AN NINH TRONG CƠ SỞ DỮ LIỆU

Mục lục

Lời nói đầu.....1
I.Tổng quan về Oracle.....5
1. Giới thiệu: 5
2.Đặc điểm của hệ quản trị CSDL Oracle 5
2.1. Cấu trúc của Oracle 5
3.Những đặc trưng của hệ quản trị CSDL Oracle. 13
3.1.Truy xuất dữ liệu 13
3.2.Tính đồng hành 14
3.3.Tính an toàn.....14
3.4.Backup và Restore dữ liệu 14
3.5.CSDL phân tán.....15
II.Bảo mật CSDL 15
1.Các cơ chế bảo mật: 16
1.1.User CSDL và Schema 16
1.2. Privilege (Đặc quyền) 17
1.3.Role (Vai trò).....18
1.4.Các xác lập lưu trữ và quota 18
1.5.Các Profile (lược sử) và giới hạn tài nguyên 19
1.6.Sự kiểm tra(Auditting) 20
2.Trusted Oracle 20
III.Bảo mật của ứng dụng.....21
1.Tổng quan.....21
2.Sử dụng đối tượng CSDL của ứng dụng 21
3.Bảo mật ứng dụng điều khiển bằng bảng 22
IV.Tài liệu tham khảo 23

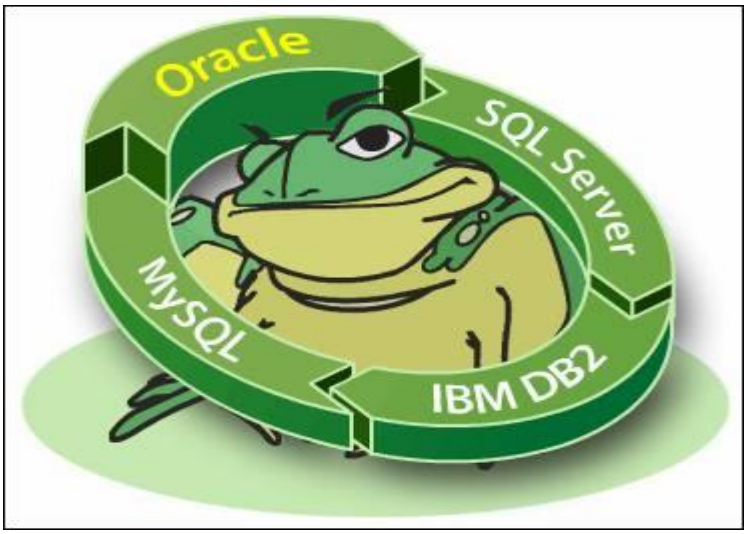
Lời nói đầu.

Khi nói đến nghề quản trị trong lĩnh vực Công nghệ thông tin tại Việt Nam, chúng ta thường liên tưởng đến nghề quản trị Mạng, chỉ khoảng gần năm năm trở lại đây thì tại Việt Nam mới hình thành lên một nghề mới, đó là nghề quản trị hệ thống Cơ sở dữ liệu (CSDL), thường gọi là DBA (Database Administrator). Hiện có ba hệ quản trị CSDL được biết đến trên thế giới và tại Việt Nam, đó là: Oracle, Microsoft SQL-Server và IBM-DB2. Nhưng với số liệu thống kê năm 2010, Oracle chiếm đến 76,8% thị phần CSDL trên toàn thế giới và theo báo cáo tài chính quí 3 của hãng thì Oracle tăng 35% doanh thu so với cùng kỳ năm trước, từ các thông tin này chúng ta có thể đánh giá bước đầu Oracle là Cơ sở dữ liệu được các doanh nghiệp ứng dụng nhiều nhất trên thế giới.

Đặc điểm sử dụng các Cơ sở dữ liệu tại Việt Nam:

Oracle thường được các doanh nghiệp đang phát triển và doanh nghiệp lớn sử dụng, do đó những ai làm việc với công nghệ Oracle thường có mức thu nhập tốt.

- Microsoft SQL-Server thì được các doanh nghiệp vừa và nhỏ (Small and Medium Bussiness - SMB) thích dùng, vì tính đơn giản và dễ sử dụng của CSDL này. Mức thu nhập khi làm việc với SQL-Server vì thế cũng có phần không cao như Oracle.
- IBM DB2 thì hơi đặc biệt, chỉ một số ít doanh nghiệp lớn tại Việt Nam sử dụng, người quản trị hệ thống này thường được ưu đãi tốt nhưng số lượng không nhiều.



Đối với một số doanh nghiệp, hiệu quả kinh doanh phụ thuộc hoàn toàn vào giá trị thông tin thì vấn đề an toàn thông tin được đặt trọng tâm hàng đầu. Hiện nay, các doanh nghiệp thường gặp khó khăn hoặc không thể trả lời các câu hỏi như: ai đã truy cập đến dữ liệu quan trọng này và vào thời gian nào ? ai đã chỉnh sửa thông tin nhạy cảm của khách hàng ? tại sao dữ liệu này lại bị thay đổi ? hệ thống dữ liệu hiện có bị ai xâm nhập trái phép hay không ?. Để trả lời các câu hỏi này, doanh nghiệp cần thiết phải trang bị giải pháp giám sát đầy đủ và chính xác, đồng thời giải pháp phải đảm bảo không tăng tải và làm chậm hệ thống hiện tại. Trong một số lĩnh vực kinh doanh nhạy cảm, tính bí mật của thông cần được bảo đảm tuyệt đối. Thông tin cần đạt tính bí mật và nguyên vẹn ngay khi được lưu trên đĩa cứng, và khi thông tin được trao đổi vào ra trong mạng cũng phải được đảm bảo an toàn.

Điều đó cho thấy tầm quan trọng của việc bảo vệ dữ liệu trong một hệ quản trị CSDL. Dựa trên những lí do đó, chúng em quyết định chọn đề tài An ninh trong hệ quản trị CSDL Oracle – hệ quản trị CSDL mạnh nhất cho tới nay để tìm hiểu. Trong đây sẽ đề cập tới những đặc trưng tính bảo mật CSDL và quan trọng hơn là các chức năng an ninh trong hệ quản trị CSDL Oracle.

Trong quá trình tìm hiểu sẽ không tránh khỏi thiếu sót, mong Thầy và các bạn quan tâm theo dõi và đóng góp ý kiến, chúng em xin chân thành tiếp thu và sửa đổi. Chúng em có lời cảm ơn chân thành tới Thầy Phạm Trung Thành đã động viên, chỉ ra hướng tìm hiểu cho tất cả các sinh viên trong đó có nhóm em cùng tất cả các ý kiến đóng góp của các bạn!

Nhóm sinh viên: Lê Thị Nhung.

Nguyễn Thị Luận.

I.Tổng quan về Oracle.

1. Giới thiệu:

Cơ sở dữ liệu oracle là một cơ sở dữ liệu mạnh mẽ nhất thế giới. Được thiết kế để triển khai cho mọi môi trường doanh nghiệp. Việc cài đặt, quản lý rất dễ dàng, các công cụ để phát triển các ứng dụng một cách hoàn thiện và nhanh chóng. Cơ sở dữ liệu oracle phù hợp cho mọi loại dữ liệu, các ứng dụng và các môi trường khác nhau bao gồm cả windows và linux với chi phí tối thiểu.

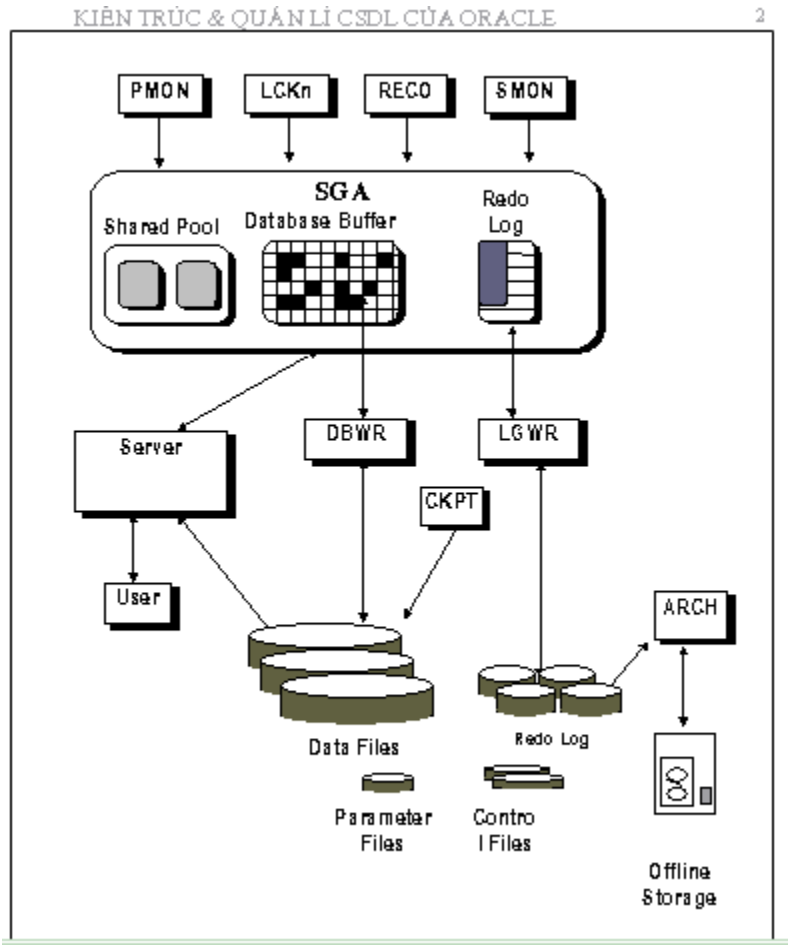
2.Đặc điểm của hệ quản trị CSDL Oracle.

- Oracle hỗ trợ việc quản lí dữ liệu có dung lượng hàng trăm Gb
- Hỗ trợ mạnh cho các tính năng nhiều user truy xuất cùng đơn vị dữ liệu trong cùng thời gian.
- Có thể làm việc 24/24.
- Tuân thủ các chuẩn công nghiệp trên các yếu tố: Hệ điều hành, Mạng, Ngôn ngữ truy xuất.
- Cung cấp cơ chế an toàn và bảo mật tốt.
- Đảm bảo tạo các ràng buộc toàn vẹn đơn giản và phức tạp trên CSDL.
- Hỗ trợ nhiều thuận lợi cho mô hình Cilent/ Server.
- Cho phép nhiều chủng loại máy, hệ điều hành khác nhau chia sẻ trên dữ liệu mạng .
- Dùng để phát triển ứng dụng và có thể sử dụng trên bất kì hệ thống nào.
  ⇨ Có thể nói đây là CSDL mạnh nhất trên thế giới.

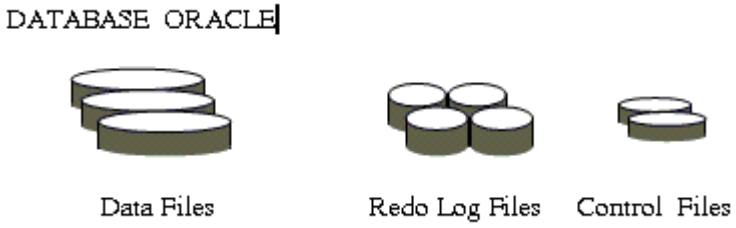
2.1. Cấu trúc của Oracle

Để thiết kế và quản lý CSDL của ORACLE được tốt chúng ta cần phải hiểu được kiến trúc cơ bản của ORACLE.

-Sơ đồ kiến trúc cơ bản ORACLE như sau:



Một CSDL của ORACLE thì bao gồm các file điêu khiển (Control files),các file dữ liệu (data files) và các file đăng nhập lại (Redo log files).

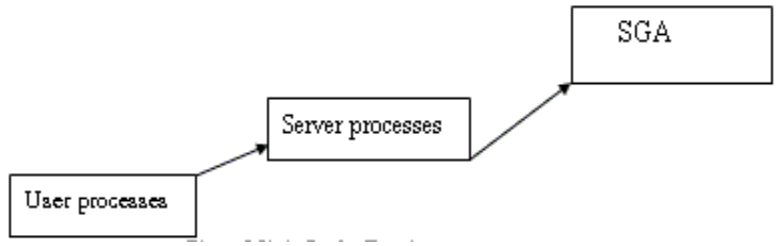


- File dữ liệu (data files) :chứa đựng tất cả các dữ liệu của CSDL có cấu trúc logic như các table,index,và chúng được lưu giữ vật lý trong các file CSDL

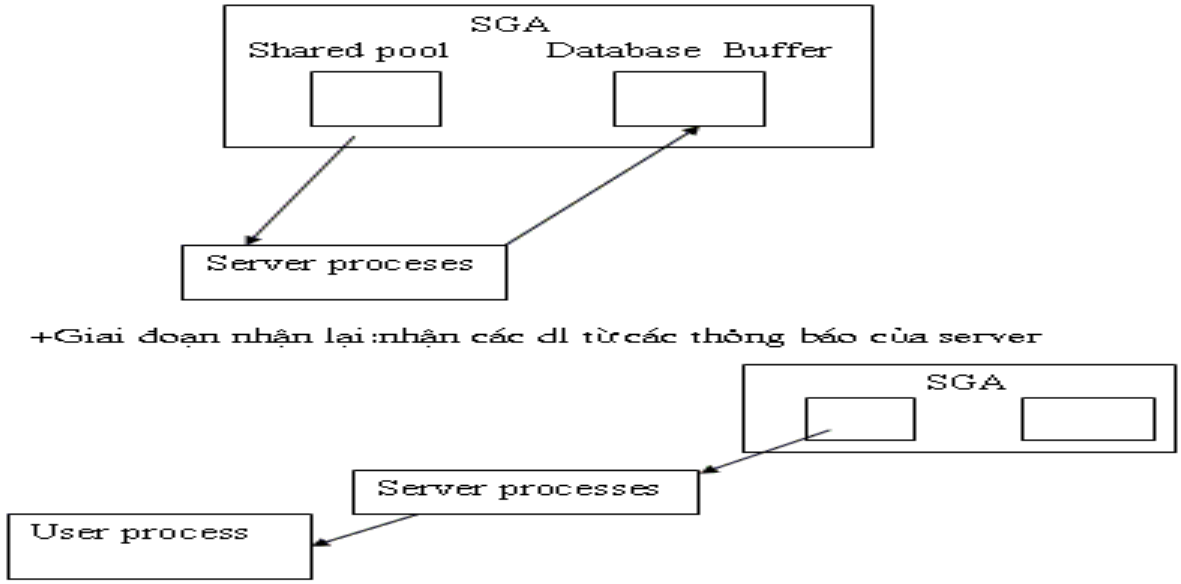
- File đăng nhập lại (reedo log files)ùng cho các xử lý đăng nhập lại.Nó ghi lại tất cả những thay đổi được tạo cho CSDL và chứa các thông tin cho việc khôi phục .
- Các file điều khiển (control files) : ghi lại cấu trúc vật lý của CSDL
- File biến (parameter) :chứa các tham số cho việc thiết lạp môi trường làm việc , xác định các đặc tính của các thể hiện ,thường được định danh là Init.Ora.
*Mỗi khi ORACLE bắt đầu SGA được định vị và các tiến trình nền được bắt đầu .Sự kết hợp của một vùng nhớ đệm và một tiến trình nền được gọi là một thể hiện (Instance) của ORACLE
- Các tiến trình nền tương ứng với các công việc khác nhau phục vụ cho tất cả mọi người dùng CSDL.

Vd: Tiến trình DBWR được dùng để ghi lại đĩa những dữ liệu từ vùng nhớ đệm CSDL (Database buffer cache)
- Các tiến trình phục vụ (server processes) : khi một user chạy một ứng dụng tương ứng hoặc một hỏi đáp dữ liệu thì các tiến trình của người dùng được đưa ra. ORACLE sẽ đưa ra một tiến trình phục vụ để trao đổi yêu cầu với các tiến trình của người sử dụng.

User processes <---------> Server processes
Các tiến trình phục vụ sau khi thi hành các yêu cầu của người sử dụngvà đưa trở lại kết quả cho user.
+ Các thông báo của SQL sẽ được thi hành bởi các server process ,các tiến trình phục vụ sẽ đọc khối dữ liệu từ đĩa vào vùng nhớ dữ liệu được dùng chung của SGA (vùng Shared pool).Sau đó đưa ra kết quả từ các thông báo SQL.Các tiến trình phục vụ có ba giai đoạn:
+ Giai đoạn phân tích:kiểm tra cú pháp,kiểm tra quyền truy nhập,đường dẫn tìm kiếm có hiệu quả nhất rồi xác định cây phân tíchảCây phân tích sẽ được cất giữ trong vùng Shared SQL Area (Thuộc một phần của Shared pool trong SGA) vùng này chứa các thông tin cho việc thi hành các thông báo SQL



+ Giai đoạn thi hành các thông báo:
gắn cây phân tích tới vùng đệm dữ liệu

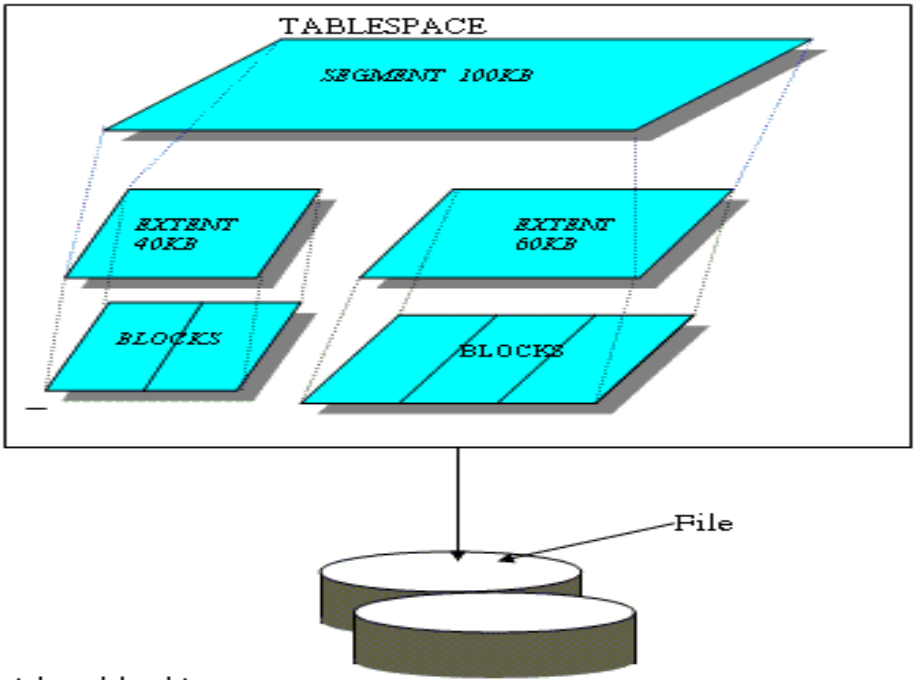


*Từ đó ta có thể kết luận về hai kiểu tiến trình (Background process và server processes):chúng đều là những tiến trình nhưng chúng khác nhau trong phạm vi tiến trình của mình.Các tiến trình phục vụ trao đổi các yêu cầu từ user,còn background processes thi hành những công việc của một instance.
*Giới thiệu về SGA(System global area :Vùng bộ nhớ hệ thống): là một tập hợp các vùng bộ nhớ đệm dùng chung các tiến trình khác nhau có một vùng đệm riêng và SGA được định vị bởi ORACLE cho một instance.

Các vùng đệm trong SGA:
+ Shared spool: (như đã nói ở phần server processes)
ngoài ra:kích cỡ của shared pool có thể được xác định bởi biến Shared -pool-size.
+ Database buffer cache (vùng nhớ đệm cơ sở dữ liệu) :
Trước khi dl có thể được truy nhập một tiến trình phục vụ sẽ đặt dữ liệu vào trong vùng đệm Database buffer cache,vùng đệm này sẽ được cát giữ những bản sao dữ liệu đọc từ đĩa bởi một tiến trình phục vụ
Số block trong vùng này được xác định bởi biến DB_Block_buffer (kích thước của block được xác định bởi biến DB_block_size)
+ Redo log buffer (Vùng đăng nhập lại) :

Vùng này chứa các thông tin về những thay đổi tới CSDL .ORACLE sẽ ghi lại những thay đổi tới CSDL trong vùng này tới các Redo log file.Kích cỡ của vùng này được xác định bởi biến Log-buffer

* Các tiến trình nền (Backgound processes)
- DBWR: các khối dữ liệu đã được sửa đổi trong vùng Database buffer cache được viết trở lại đĩa bởi 1 tiến trình nền DBWR .Tiến trình này quản lý vùng Database buffer.Cho nên các tiến trình của user luôn luôn có thể tìm được vùng đệm rỗi.
- Logwriter (LGWR): ghi lại tất cả những thay đổi tới CSDL trong vùng Redo log buffer tới đĩa bởi một tiến trình nền có tên LGWR,một tiến trình nền khác là ARCH(Achiver) có thể được bắt đầu để lưu giữ các thông tin đăng nhập lại trên 1 thiết bị cất giữ đã định trước (tape,disk)dùng cho việc khôi phục bị thất bại.
LGWR viết Redo log buffer vào Redo log files khi:
+ khi user đưa ra chỉ thị commit
+ vùng redo log buffer vượt quá 1/3 kích cỡ cho phép của vùng
+ DBWR cần để xoá sạch các khối đệm cho 1 điểm kiểm tra hoặc quá thời gian xuất hiện
- Process Monitor (PMON): thi hành những tiến trình khôi phục khi 1 tiến trình của user bị thất bại.
- System monitor (SMON): tự động thi hành việc khôi phục instance
+ Tái tạo lại kiểu không gian bảng (table space)
- Recover process(RECO):giải quyết những thất bại liên quan tới 1 tiến trình được phân bố.
- Checkpoint (CKPT): thay LGWR viết các thông tin dữ liệu từ vùng log buffer tới các header của các file dl và file điều khiển.
- LCKn (Lock process): Xử lý tình trạng bế tắc giữa các instance trong hệ tiến trình song song. Quản lý việc cất giữ dữ liệu của oracle
- Oracle định vị không gian cho dữ liệu trong CSDL theo :đơn vị nhỏ nhất là các block tiếp đến là extent,segment,tablespace



* Khối dữ liệu (data block)

-Khối dữ liệu của Oracle là khối nhỏ nhất của công việc I/O nó có thể gồm nhiều khối file vật lý được định vị từ file CSDL.

-Các khối CSDL được gọi là các khối logic.Nó tương ứngvới một hoặc 0nhiều khối vật lý trên đĩa.

- Kích cỡ 1 khối CSDL ORACLE phải được xác định trước khi tạo CSDL và nó là hằng số trong tất cả các các file của CSDL.Khi CSDL đã được thì kích cỡ của nó không thể được thay đổi ngoài trừ việc tạo lại CSDL.

Kích cỡ thường 2-:-4 KB được xác định bởi biến DB_block_size

- Thành phần của một khối:

+ Header: chứa những thông tin chung trong khối như địa chỉ khối,kiểu segmentả

+ Table directory:bao gồm các thông tin về các bảng trong cluster và được sử dụng với các segment hợp lại

+ Row data:cất giữ dữ liệu của các table hoặc index

+Row directory:bao gồm các thông tin về các hàng thực sự trong khối

+Free space:bao gồm tập các byte trong khối mà còn có sẵn cho việc update,insert,delete,việc điều khiển free space cho việc update,insert,delete trong CSDL bằng cách thay đổi giá trị của các biến:

PCTfree,PCTused,Initrsns,Matrans.

+Biến PCTfree: xác định tỷ lệ % của khối để sử dụng việc update các hàng đã tồn tại trong khối.

VdCT free=20 tức là 20% của khối dùng cho việc update khi tỷ lệ của khối còn rỗi >% của PCTfree thì việc chèn mới cho phép.

+Pctused: cho phép 1 khối được xem xét lại cho việc chèn các hàng mới,các hàng mới sẽ được chèn khi tỷ lệ % của khối đang được sử dụng nhỏ hơn giá trị Pctused,giá trị ngầm định = 40.

Vd: Pctused =40 thì sau khi không gian đã được sử dụng <40% thì hàng mới có thể được insert vào khối này.

- Để xác định các transaction đang hoạt động trên 1 khối bởi việc sử dụng 2 biến:

+INITRANS:là số các transaction đưa vào lúc bắt đầu có thể xẩy ra đồng thời được định vị trong header của mỗi khối (ngầmđịnh=1,min=1,max=255)

+Maxtrans:là số lớn nhất các tiến trình đồng thời của khối được cung cấp (mặc định là 255,min=1,max=255)

*EXTENT: Là 1 tập các khối kề cận được định vị tới 1 segment.Khi 1 đối tượng của CSDL được hình thành thì không gian được định vị cho nó(các đối tượng có thể là:table,cluster,index,roll back segment,table space)

Mỗi segment cần có ít nhất là 1 extent,riêng đoạn khôi phục cần có ít nhất là 2 extent

- Các extent đầu tiên gọi là Initial extent. Các extent tiếp theo gọi là incrmental extent.

+ Một đối tượng sẽ được cấp 1 extent mới nếu tất cả các extent đã định vị hiện tại của nó đã được sử dụng.

+ Các extent có thể được định vị cho đối tượng khác khi segment dùng nó bị huỷ bỏ hoặc nó thuộc phần cắt bớt đi của segment

- Điều chỉnh extent tới các segment bởi các biến cất giữ sau:

+ Initialác định kích cỡ của extent đầu tiên được định vị cho segment ngầm định là 5 data block.

+ Nextác: định kích cỡ của các extent tiếp theo ngầm định là 5 blocks

+ Maxextents: tổng số lớn nhất các extent cấp cho segment ngầm định là 99

+ Minextents:tổng số extent được cấp cho segment khi segment được tạo ngầm định là 1.

+ PCTỉncease: tỷ lệ % mà mỗi extent tiếp theo được phát triển có thể vượt quá extent cuối cùng ngầm định là 50%.

+ Optimal: kích cỡ tối ưu cho 1 đoạn khôi phục mgầm định la null.
+Free lits: số danh sách của các khối rỗi cho insert vào bảng ngầm định là 1

*SEMENT : là tập các extent (1 hoặc nhiều) mà chứa đựng tất cả dữ liệu của 1 kiểu dữ liệu riêng.
-Các kiểu segment:
+Đoạn dữ liệu (data segment):tập tất cả cá extent mà chúng cất giữ tất cả dữ liệu cho một bảng hoặc 1 cluster
+Index segment:chứa tất cả các dữ liệu chỉ số cho việc tìm kiếm được nhanhchóng.
+Temporary Segment:tập các exyent cất giữ các giữ liệu thuộc vào các bảng tạm thời được tạo trong suốt sự thi hành 1 sắp xếp.
+Roll back segment:tập các extent mà chúng cất giữ các giữ liệu khôi phục cho việc khôi phục lại
+Boostrap segment:là 1 extent chứa các định nghĩa về từ điển cho các bảng từ điển và được nạp vào khi CSDL được mở
TABLESPACE
-Dữ liệu trong một cơ sở dữ liệu của Oracle dược cất giữ logic trên các tablespace và được cất giữ vật lý trong các file cơ sở dữ liệu.Một cơ sở dữ liệu của Oracle có thể bao gồm một hoặc nhiều Tablespace
+ Mỗi một tablespace có một chức năng riêng Tablespace hệ thống dành cho các hoạt đọng cửa hệ thống , Tablespace user dành cho các hoạt động của user .

MỘT SỐ LỆNH SQL Để Hiển thị thông tin về việc cất giữ:
- Liệt kê tất cả các cột trong tablespace của người dùng (ser_tablespace) :
Code:

```
SQL> Desc dba_tablespaces;
```

-Liệt kê các thông tin về mỗi tablespace
Code:

```
SQL> Select * from Dba_tablespaces;
```

-Hiện các Extent của không gian rỗi trong mỗi tablespace :
Code:

```
SQL> Select * from Dba_free_space;
```

-Hiện các thông tin chung về các file dữ liệu thuộc về mỗi tablespace :

Code:

```
SQL>Select file_name, tablespace_name, bytes from Dba_data_files;
```

- Hiện những thông tin chung về tất cả các segment trong CSDL :

Code:

```
SQL> Select owner,segment_name,extents,max_extents

from Dba_segments;
```

3.Những đặc trưng của hệ quản trị CSDL Oracle.

3.1.Truy xuất dữ liệu

- SQL (SEQUEL – Structured English Query Language) là ngôn ngữ truy xuất dữ liệu cho hệ quản trị CSDL quan hệ, SQL trong Oracle tương thích với SQL theo dạng chuẩn ANSI (American National Standards Intistute)
- SQL chia làm 4 lớp:
  - Ngôn ngữ định nghĩa dữ liệu (DDL – Data Difinition Language)
    + Mức quan niệm: CREATE DATABASE, TABLE…….
    + Mức ngoài: CREATE VIEW, GRANT……
    + Mức trong: CREATE INDEX,…
  - Ngôn ngữ thao tác dữ liệu (DML – Data Manipulation Language)
    + Truy xuất: SELECT
    + Cập nhật: INSERT, UPDATE, DELETE.
  - Ngôn ngữ điều khiển dữ liệu (DCL- Data Control Language)
  - Ngôn ngữ diều khiển hệ thống (SYCL- System Control Language)
- PL/ SQL (Procedure Language / SEQUEL) là ngôn ngữ mở rộng của SQL kết hợp với ngôn ngữ lập trình có cấu trúc

3.2.Tính đồng hành

Các yêu cầu sau đây cần thỏa mãn: Dữ liệu đọc và bổ sung nhất quán , thực hiện tốt các yêu cầu của người dùng

- Tính đồng hành: Vấn đề cần điều khiển tính ra vào với các user khi truy xuất cùng một đơn vị dữ liệu sao cho tại cùng một thời điểm cho giản ước thời gian chờ là tối thiểu, tới mức gần bằng 0.
- Tính nhất quán:
  + Đảm bảo dữ liệu nhất quán trong thời gian đọc.
  + Đảm bảo người đọc không chờ người viết và ngược lại.
  + Đảm bảo người viết phải chờ người viết khác nếu cùng cập nhật một CSDL

3.3.Tính an toàn

Đảm bảo tính an toàn dữ liệu nhờ các kiểm soát:

- Quyền truy cập vào CSDL
- Quyền trên CSDL nào
- Quyền trên đối tượng nào trong CSDL, bao gồm:
  - Quyền hệ thống: Cho phép tác động đến hệ thống, là quyền rất mạnh.
  - Quyền đối tượng: Cho phép tác động lên một đối tượng cụ thể.

3.4.Backup và Restore dữ liệu

- Backup: Trong khi backup CSDL vẫn có thể hoạt động, cơ chế backup cần uyển chuyển.Người ta cần backup với những lí do:
  - Xóa nhầm dữ liệu.
  - Phát biểu nhầm các câu lệnh: Insert, Update.
  - Lỗi đĩa cứng.
  - Mất mát do mất cắp.

  Một cách khắc phục là tạo file backup gồm 2 dạng :

  - Full backup: Toàn bộ file tạo CSDL
  - Partial backup: Chỉ backup một phần của dữ liệu
- Restore :
  - Là quá trình hồi phục lại CSDL ban đầu từ CSDL được backup
  - Thứ tự backup phụ thuộc vào: + Kiểu backup.

+ Thời gian biểu (schedule)

3.5.CSDL phân tán

3.5.1 Xử lí phân tán

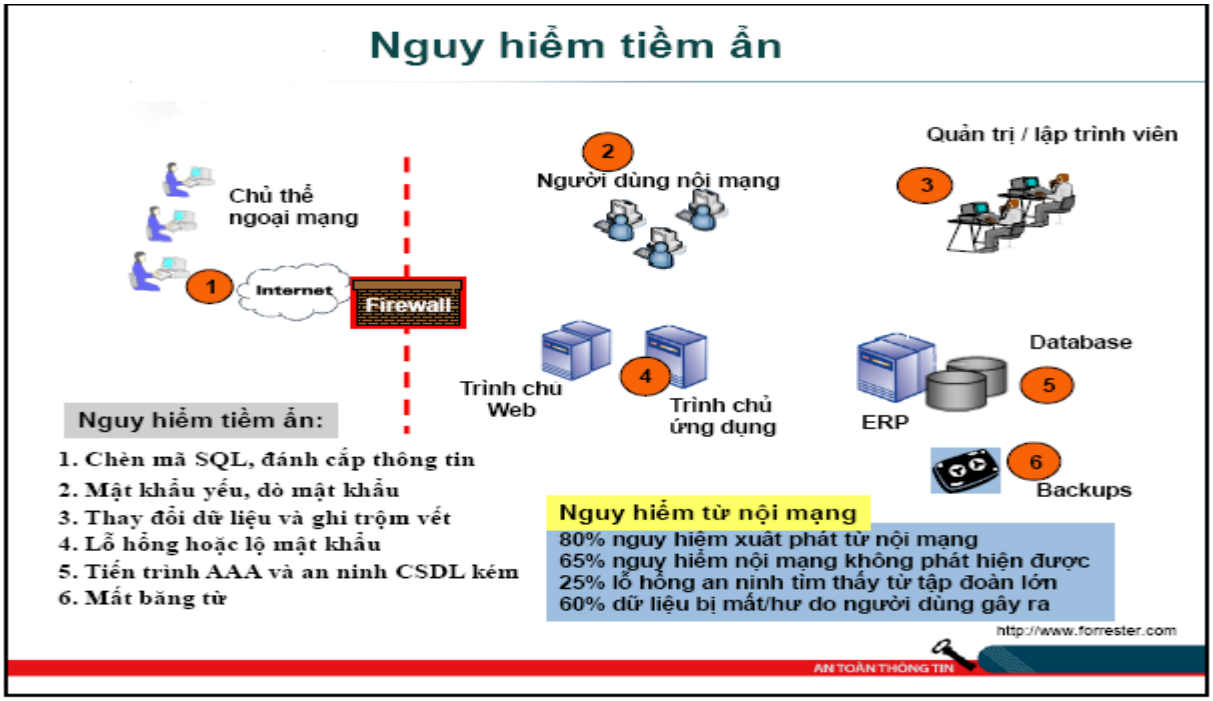
- Dùng nhiều bộ xử lí thiến trình thành các tập công việc có liên quan. Phân công công việc cho các bộ xử lí khác cùng hợp tác làm việc . Oracle hỗ trợ công việc xử lí phân tán bằng kiến trúc Client/ Server.
- Client tập trung cho công việc ra yêu cầu trình bày dữ liệu
- Server tiếp nhận các yêu cầu từ Client, xử lí và trả về kết quả cho Client

3.5.2 CSDL phân tán

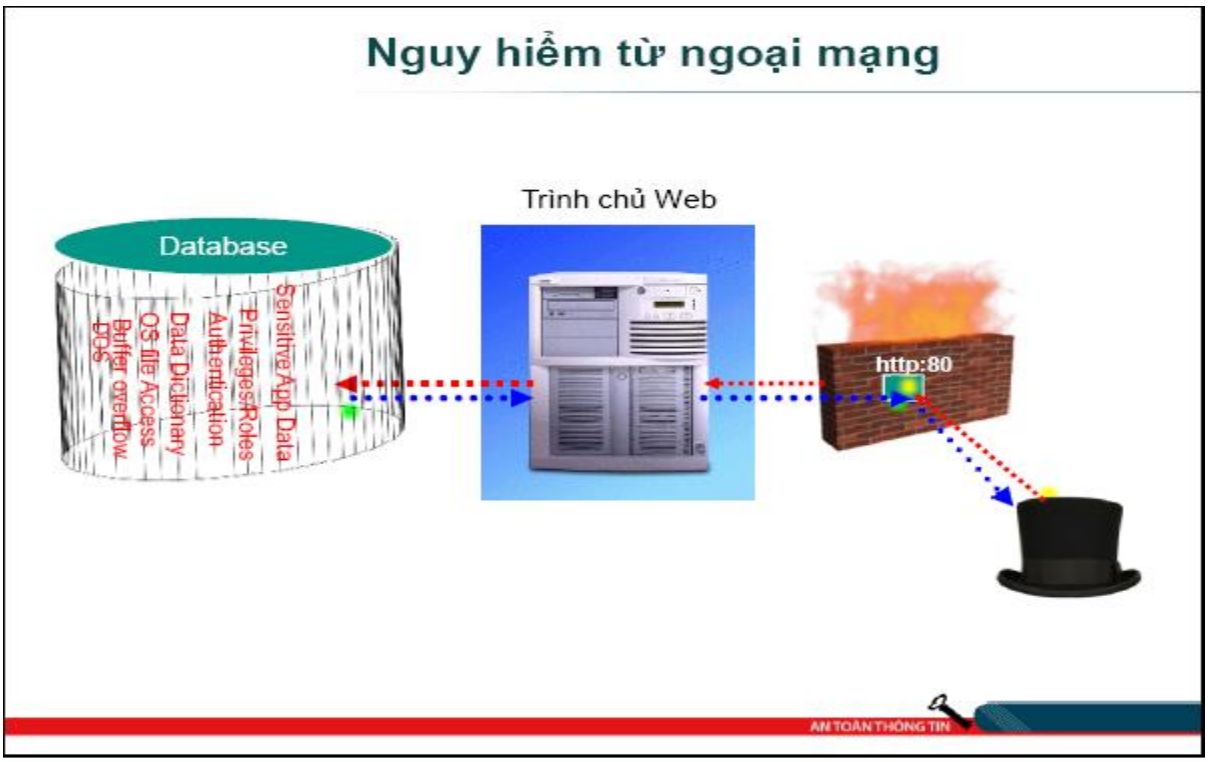
- Là một mạng CSDL quản lí bởi nhiều server CSDL , xuất hiện đối với người dùng như một CSDL duy nhất
- Dữ liệu có thể truy xuất bổ sung đồng thời tại cùng một thời điểm
- Một số khái niệm:
  - Local Database: CSDL mà người dùng kết nối trực tiếp
  - CSDl còn lại được gọi là Remove Database
  - Client Database: CSDl cục bộ truy xuất từ xa.

II.Bảo mật CSDL

Có rất nhiều mối nguy hiểm tiềm ẩn:



Cũng như mối nguy hiểm từ ngoại mạng:



Vì thế các hệ thống nhiều user như Oracle luôn bao gồm các đăc trưng bảo mật kiểm soát việc được truy xuất và sử dụng CSDL.

1.Các cơ chế bảo mật:

- Ngăn cản sự truy xuất dữ liệu trái phép
- Ngăn cản sự truy xuất trái phép các đối tượng schema
- Điều khiển việc sử dụng đĩa
- Điều khiển việc sử dụng các tài nguyên hệ thống (như thời gian CPU)
- Giám sát các hành động của user

Các cơ chế bảo mật trên được thực hiện bởi các phương tiện sau

1.1.User CSDL và Schema

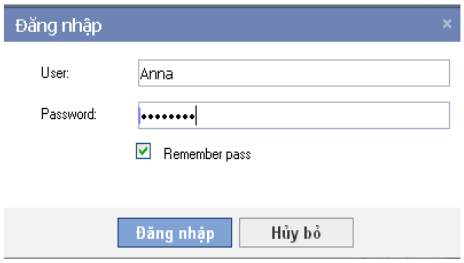
Mỗi CSDL Oracle có một danh sách username. Để tru xuất một CSDL, một username cần sử dụng CSDL gắn với một username hợp lệ của CSDL. Mỗi username có một password kết hợp để ngăn ngừa sự truy xuất trái phép. Một CSDL cần có một phương pháp nhận biết user của mình, phân biệt được tầm vực

hoạt đông và quyền hạn của user đó để có thể cung ứng đủ và đúng các yêu cầu được phép của user

- Nguyên lí: hoàn toàn giống nguyên lí của các CSDL khác, của hệ điều hành, các trò chơi,…… Oracle sẽ hỏi user về định danh username và password tương ứng. Nếu câu trả lời khớp với số liệu đã được lưu trong hệ thống bảo mật CSDL, user được quyền truy cập CSDL, ngoài ra user sẽ phải nhập lại hay sẽ bị cấm truy nhập. Nếu dùng khả năng giao tiếp user của công cụ thì password mà ta nhập sẽ không thể hiện trên màn hình, giúp bảo mật nội dung của password.

  VD: Cách nhập password thông qua giao tiếp của chương trình

```
%sqlplus
 Enter User: Anna
Enter Password:........
SQL> show user
user is Anna
```



- Domain Security (Bảo mật vùng)

  Mỗi user có một Domain security là một tập các tính chất xác định những điều như:

  - Các hoạt động (Privilege và Role) sẵn có cho user
  - Các quota tablespace (Không gian đĩa sẵn có cho user)
  - Giới hạn tài nguyên hệ thống (ví dụ thời gian xử lí CPU) cho user

1.2. Privilege (Đặc quyền)

Một Privilege là quyền thực hiện một loại lệnh SQL đặc biệt , một số Privilege như:

- Quyền kết nối vào CSDL (tạo ra một session)

- Quyền tạo bảng trong schema của mình
- Quyền chọn các hàng từ các user khác
- Quyền thực hiện các stored procedure của user khác

Các Privilege của một CSDL Oracle có thể chia thành 2 loại riêng biệt: Privilege hệ thống và Privilege các đối tượng schema

- Privilege hệ thống

Cho phép user thực hiện một hành động mức hệ thống đặc biệt hay một hành động đặc biệt trên một loại đối tượng schema đặc biệt. VD: đặc quyền tạo ra một tablespace hay xóa các hang trong CSDL. Nhiều đặc quyền hệ thống chỉ dành cho người quản trị và các nhà phát triển ứng dụng do chúng rất mạnh.

- Privilege đối tượng schema

Cho phép các user thực hiện một hành động đặc biệt trên một đối tượng schema đặc biệt. VD: đặc quyền xóa một số hang của một bảng, đặc biệt là một Privilege đối tượng . Privilege đối tượng được gán cho các người dùng cuối

- Cấp Privilege

Một user có thể được nhận Privilege từ 2 cách sau:

- Cấp tường minh
- Các Privilege có thể gán cho các Role sau đó role có thể được gán cho một hay nhiều user(phổ biến hơn)

1.3.Role (Vai trò)

Role là một nhóm các Privilege có liên quan được đặt tên để cấp cho các user hay các role khác.Các tính chất cho phép quản lí Privilege thông qua role:

- Giảm việc cấp Privilege
- Tính sẵn có chọn lọc của Privilege
- Hỗ trợ ứng dụng

1.4.Các xác lập lưu trữ và quota

Oracle cung cấp các biện pháp để định hướng và giới hạn việc sử dụng không gian của CSDL trên từng user , bao gồm:

- Tablespace mặc định : gíup Oracle chỉ định nơi lưu trữ trong trường hợp vị trí của đối tượng schema không được chỉ định
- Tablespace tạm thời: giảm sự tranh chấp xuất nhập giữa các segment tạm và các loại segment khác.
- Các quota Tablespace: điều khiển chon lọc lượng không gian đĩa tiêu tốn bởi các đối tượng của các schema xác định.

1.5.Các Profile (lược sử) và giới hạn tài nguyên

Mỗi user được gán một profile chỉ định các giới hạn trên một số tài nguyên hệ thông sẵn có cho user, gồm:

- Số session có thể được thiết lập đồng thời bởi user.
- Thời gian truy xuất CPU
  + Sẵn có cho session của user
  + Sẵn có cho một lần gọi Oracle của một lệnh SQL
- Lượng xuất nhập logic:
  + Sẵn có cho session của user
  + Sẵn có cho một lần gọi Oracle của một lệnh SQL
- Lượng thời gian rỗi cho phép của session của user
- Lượng thời gian kết nối cho phép của một session của user
- Các giới hạn password
  + Khóa account sau nhiều lần cố gắng login thất bại
  + Sự hết hạn của password và chu kì gia hạn
  + Sự sử dụng lại password và các hạn chế phức tạp

Các profile khác nhau có thể tạo ra và gán riêng biệt cho từng user. Các user không được gắn tường minh một profile riêng biệt so dùng profile mặc định

- Oracle chỉ sử dụng Profile khi tham số RESOURCE_LIMIT trong tập tin INIT.ORA có giá trị là TRUE(mặc định là False). Dưới đây là danh sách một số tài nguyên có thể chỉ định rõ trong các profile :
  - SESSION_PER_USER: xác định số tối đa các phiên làm việc mà user có thể truy cập CSDL đồng thời.
  - CPU_PER_SESSION: chỉ định tổng thời gian, tính bằng phút của CPU mà user có thể sử dụng trong 1 kết nối CSDL đơn

- CPU_PER_CALL: chỉ định tổng thời gian CPU dành cho user trong một lần gọi thực hiện lệnh SQL
- LOGICAL_READS_PER SESSION: chỉ định tổng các blocks được đọc logic mà user có thể thực hiện trong phiên làm việc đã cho.
- LOGICAL_READS_PER_CALL: chỉ định tổng các blocks được đọc logic trong một lần thực hiện lệnh gọi SQL.
- IDLE_TIME: chỉ định thời gian (tính bằng phút) không hoạt động tối đa mà một kết nối có thể nhận. Khi thời gian không hoạt động vượt quá giới hạn này, kết nối so ị ngắt.
- CONNECT_TIME: chỉ định thời gian tối đa mà một kết nối sử dụng (tính bằng phút). Nó bao gồm cả thời gian hoạt động và không hoạt động. Cần lưu ý rằng, khi thời gian két nối vượt quá giới hạn, kết nối sẽ bị ngắt, nhưng công cụ thực hiện kết nối (như SQL*Plus,….) không bị đóng.

1.6.Sự kiểm tra(Auditting)

Oracle cho phép kiểm tra chọn lọc các uhành động của user. Sự giám sát có thể được thực hiện ở 3 mức:

- Kiểm tra lệnh: kiểm tra lệnh SQL mà khong quan tâm tới các đối tượng schema
- Kiểm tra đặc quyền : kiểm tra các đặc quyền hệ thống quan trọng mà không quan tâm đến các đối tượng schema
- Kiểm tra các đối tượng schema: Kiểm tra đến sự truy xuất đến một schema nhất định mà không quan tâm đến user

2.Trusted Oracle

Trusted Oracle là một sản phẩm quản lí bảo mật CSDL nhiều mức của công ty Oracle:

- Được thiết kế để cung cấ khả năng bảo mật cao nhằm phục vụ yêu cầu các tổ chức xử lí những dữ liệu tế nhị hay mật.
- Tương thích với các sản phẩm cơ bản của Oracle
- Hỗ trợ tất cả các tính năng cơ bản của Oracle

- Thực hiện kiểm soát truy nhập ủy nhiệm(mandatory access control-MAC)trên một giả rộng cá môi trường hệ điều hành bảo mật nhiều cấp.
  - MAC là biện pháp giới hạn sự truy xuất thông tin dựa trên các nhãn .
    + Nhãn của user chỉ định thông tin nào user được phép truy xuất và loại truy xuất mà user được thực hiện
    + Nhãn của đối tượng schema chỉ định độ quan trọng của thông tin mà nó chứa

III.Bảo mật của ứng dụng

1.Tổng quan

Chế độ bảo mật của ứng dụng giới hạn việc truy cập CSDL từ phía Cilent của hệ thống, giới hạn việc truy xuất các đối tượng giao diện đặc biệt. Nói rộng ra nó bao gồm cả việc sử dụng bất kì đối tượng CSDL nào của ứng dụng được dùng để nâng cao chế độ bảo mật

Nguyên do cần có thêm chế độ Bảo mật của ứng dụng bên cạnh bảo mật của CSDL :

- Tăng cường và bổ sung chế độ bảo mật của CSDL
- Có thể nâng cao mức bảo mật CSDL bằng cách giới hạn hơn nữa việc truy xuất đối tượng CSDL và cung cấp một lớp trừu tượng bổ xung
- Trong các giao diện ứng dụng thông thường , giao diện cho phép các user làm việc với CSDL thông qua các trình đơn, các forms tương ứng với các tiến trình và các đối tượng hoạt động , thuận lợi cho việc ngăn ngừa user hiểu biết tên các bảng và các cột được truy xuất.
- Trong nhiều trường hợp thông tin của CSDL so dùng để quản lí các hoạt động của giao diện, dựa trên các role của user. Các bảng hệ thống chứa các thông tin bảo mật cho ứng dụng sẽ cho phép các khả năng nào của giao diện là sẵn có. Các tùy chọn trình đơn và các forms sẽ được hiện ra cho các user có đặc quyền và sẽ ẩn đi khi user không được phép
- Có thể truy xuất CSDL từ nhiều ứng dụng khác nhau

2.Sử dụng đối tượng CSDL của ứng dụng

Các đối tượng cơ bản (cluster, tables, indexes, sequence) và các quyền cấp cho các đối tượng này thuộc loại bảo mật CSDL. Các đối tượng khác như view,

procedure và function được coi là các đối tượng CSDL ứng dụng và các quyền cấp về chúng thuộc loại bảo mật ứng dụng

- Script là nơi bảo mật CSDL chấm dứt và là nơi bảo mật ứng dụng bắt đầu .
- Synonym chung được tạo để che ID hệ thống của DBA, giới hạn quyền được gán cho super-user của ứng dụng
- Các đối tượng ứng dụng bao gồm các procedure, các views, functions, packages
- Ta cần phải dùng các stored procedure và function để thực hiện tất cả các chuyển tác, vì:
  - Sử dụng các subroutine của CSDL sẽ giữ được tính toàn vẹn.
  - Sử dụng procedure và function cũng nâng cáo tính bảo mật CSDL
  - Sử dụng package ta có thể gói trọn các biến, cursor, hằng và các subprogram trong một đơn vị logic giúp đơn giản tiến trình gán các đặc quyền và nâng cao hiệu suất chung

3.Bảo mật ứng dụng điều khiển bằng bảng

Thường dùng trong các trường hợp giới hạn của chế độ bảo mật đòi hỏi sự linh hoạt và thường xuyên thay đổi

Ta có thể dùng các thông tin lưu trong bảng của CSDL Oracle để điều khiển chế độ bảo mật ứng dụng

Quy trình:

- Tạo ra 1 bảng chứa các thông tin về các role của user trong ứng dụng, trong đó mỗi user có một role.
- Tạo một bảng xác định các tùy chọn của trình đơn mà một role có thể sử dụng hay không
- Xây dựng một module khởi tạo để xử lí nội dung của bảng trên , nhằm tạo ra một trình đơn thực sự
- Tạo một trình đơn và các nút control động khi khởi tạo cửa sổ ứng dụng
- Cần chú ý các vấn đề :
  - Cần phải có một cách xác định role của user trước khi khởi tạo giao diện, menu.

- Trong bảng lưu trữ dữ liệu cấn phải phân định rõ và thống nhất về các định danh trong ứng dụng và trong CSDL

Tóm lại, bảo mật ứng dụng không thể thay thế bảo mật CSDL nhưng nó nâng cao độ an toàn của hệ thống bảo mật CSDL. Nó xác lập các nguyên tắc mà không thể xây dựng được bằng ràng buộc toàn vẹn

Trên đây là chúng em đã trình bày một cách tổng quát và ngắn gọn về đặc trưng cũng như các chức năng an ninh trong hệ quản trị CSDL Oracle. Rất cảm ơn Thầy và các bạn đã tập trung theo dõi.

IV.Tài liệu tham khảo

- Giáo trình lý thuyết và thực hành Oracle.
- Giáo trình kiến trúc và quản trị Oracle.
- http://download.oracle.com.
- http://tt0601.files.wordpress.com
- Google.com

Quản trị Cơ sở dữ liệu và Phần mềm ứng dụng

Bộ môn CNTT
Khoa Tin học Thương mại

Chương II: *Thiết kế CSDL quan hệ*

1. Giới thiệu chung
   1.1. Thiết kế CSDL QH và các cách tiếp cận
   1.2. Phụ thuộc hàm
2. Chuẩn hóa lược đồ quan hệ
   2.1. Các dạng chuẩn
   2.2. Tách lược đồ quan hệ theo chuẩn
3. Ràng buộc toàn vẹn trong CSDL quan hệ
   3.1. Khái niệm ràng buộc toàn vẹn
   3.2. Ràng buộc toàn vẹn trên thuộc tính
   3.3. Ràng buộc toàn vẹn trên quan hệ

1. Giới thiệu chung

1.1. Thiết kế CSDL QH và các cách tiếp cận

- Thiết kế cơ sở dữ liệu quan hệ ⇔ xây dựng lược đồ CSDL QH gồm một tập các lược đồ quan hệ thỏa mãn hai yêu cầu:
  - Lưu trữ thông tin không dư thừa
  - Tìm kiếm thông tin dễ dàng
- Ví dụ
  - Lược đồ quan hệ
    - CUNG_UNG(**MaNCC**, TenNCC, DiaChi, **SanPham**, Gia)

Dư thừa dữ liệu

- Quan hệ CUNG_UNG_0

MaNCC	TenNCC	DiaChi	SanPham	Gia
1	Hải Hà	Hà Nội	Kẹo mềm	100
1	Hải Hà	Hà Nội	Kẹo cứng	150
1	Hải Hà	Hà Nội	Bánh	200
2	Kinh đô	Hồ Chí Minh	Kẹo	120
2	Kinh đô	Hồ Chí Minh	Bánh	150

- Một nhà cung cấp cung cấp nhiều mặt hàng.
- Lặp các thông tin về nhà cung cấp ứng với mỗi một mặt hàng khác nhau của cùng nhà cung cấp đó.

Không nhất quán

- Quan hệ CUNG_UNG_0

MaNCC	TenNCC	DiaChi	SanPham	Gia
1	Hải Hà	Đà Nẵng	Kẹo mềm	100
1	Hải Hà	Hà Nội	Kẹo cứng	150
1	Hải Hà	Hà Nội	Bánh	200
2	Kinh đô	Hồ Chí Minh	Kẹo	120
2	Kinh đô	Hồ Chí Minh	Bánh	150

- Dị thường khi cập nhật thông tin về nhà cung cấp như thay đổi địa chỉ.

Dị thường khi thêm bộ

- Quan hệ CUNG UNG 0

MaNCC	TenNCC	DiaChi	SanPham	Gia
1	Hải Hà	Hà nội	Kẹo mềm	100
1	Hải Hà	Hà Nội	Kẹo cứng	150
1	Hải Hà	Hà Nội	Bánh	200
2	Kinh đô	Hồ Chí Minh	Kẹo	120
2	Kinh đô	Hồ Chí Minh	Bánh	150
3	Bibica	Đà nẵng	NULL	NULL

- Dị thường khi thêm mới thông tin về nhà cung cấp nhưng nhà cung cấp chưa cung cấp mặt hàng nào.

Dị thường khi xóa bộ

- Quan hệ CUNG_UNG_0

MaNCC	TenNCC	DiaChi	SanPham	Gia
1	Hải Hà	Hà nội	Kẹo mềm	100
1	Hải Hà	Hà Nội	Kẹo cứng	150
1	Hải Hà	Hà Nội	Bánh	200
2	Kinh đô	Hồ Chí Minh	Kẹo	120

  - Tồn tại nhà cung cấp chỉ cung cấp một mặt hàng.
  - Dị thường khi xóa thông tin về sự cung cấp ➔ xóa luôn thông tin về nhà cung cấp.

Tìm kiếm thông tin

MaNCC	TenNCC	DiaChi
1	Hải Hà	Hà Nội
2	Kinh đô	Hồ Chí Minh

CUNG_UNG_11

MaNCC	SanPham	Gia
1	Kẹo mềm	100
1	Kẹo cứng	150
1	Bánh	200
2	Kẹo	120
2	Bánh	200

CUNG_UNG_12

- Quan hệ CUNG_UNG_0 tách thành 2 quan hệ CUNG_UNG_11 và CUNG_UNG_12
  - Lưu trữ thông tin không dư thừa ???
  - Tìm kiếm thông tin dễ dàng ???

Các cách tiếp cận

- **Từ trên xuống*(Topdown)*:**
  - Xây dựng sơ đồ thực thể liên kết ER từ các đặc tả
  - Chuyển đổi sơ đồ ER thành lược đồ CSDL quan hệ.
  - Chuẩn hóa lược đồ CSDL quan hệ (nếu cần)
- **Từ dưới lên *(Bottom Up)*:**
  - Xây dựng lược đồ quan hệ ban đầu từ các đặc tả.
  - Chuẩn hóa lược đồ quan hệ.

1.2. Phụ thuộc hàm

a. Khái niệm

- Cho quan hệ R, thuộc tính B của quan hệ R được gọi là phụ thuộc hàm vào thuộc tính A của quan hệ R nếu với mỗi giá trị của A xác định duy nhất một giá trị của B. A được gọi là xác định hàm của B.
- Ký hiệu: A➔B

a. Khái niệm (t)

- *Tập các phụ thuộc hàm F của 1 lược đồ quan hệ* R là một tập gồm các phụ thuộc hàm xác định trên R.
  - Ví dụ: Tập phụ thuộc hàm F={A➔B, B➔C} của R(A,B,C)
- *Trong quan hệ R, ký hiệu A, B, C dành cho các thuộc tính đơn, X, Y, Z dành cho tập các thuộc tính.*

Ví dụ

- Tập tất cả các thuộc tính của quan hệ phải phụ thuộc hàm vào khóa.
  - MaNCC➔ TenNCC
  - MaNCC➔ SoNV
  - MaNCC➔ DiaChi
  - MaNCC: Khóa

MaNCC	TenNCC	SoNV	DiaChi
S1	Hải Hà	20	Hà Nội
S2	Kinh Đô	10	Hà Nội
S3	Bibica	30	HCM

- F={ MaNCC➔ TenNCC, MaNCC➔ SoNV, MaNCC➔ DiaChi}

Ví dụ

- Một tập thuộc tính là xác định hàm của các thuộc tính khác thì chưa chắc là một khóa.
  - TenNCC➔ DiaChi
  - TenNCC không phải là khóa

MaNCC	TenNCC	SoNV	DiaChi
S1	Hải Hà	20	Hà Nội
S2	Kinh Đô	10	Hà Nội
S3	Bibica	30	HCM
S4	Hải Hà	10	Hà Nội

b. Hệ tiên đề Amstrong

- Giả thiết
  - Lược đồ quan hệ R.
  - X,Y,Z: tập các thuộc tính thuộc R.
  - $XY = X \cup Y$
- Hệ 3 tiên đề với các phụ thuộc hàm:
  - Phản xạ: $XY \rightarrow X$; $XY \rightarrow Y$
  - Tăng trưởng: $X \rightarrow Y$ thì $XZ \rightarrow YZ$
  - Bắc cầu: $X \rightarrow Y$, $Y \rightarrow Z$ thì $X \rightarrow Z$

Luật suy ra từ hệ tiên đề

- Luật hợp
  - Nếu X➔Y, X➔Z thì X➔YZ
- Luật tựa bắc cầu
  - Nếu X➔Y, WY➔Z thì XW➔Z
- Luật tách
  - Nếu X➔Y, Z thuộc Y thì X➔Z

c. Phụ thuộc hàm đầy đủ và phụ thuộc bắc cầu

- **Phụ thuộc hàm đầy đủ**
  - Y phụ thuộc hàm đầy đủ vào X nếu Y phụ thuộc hàm vào X nhưng không phụ thuộc hàm vào bất kỳ một tập con thực sự nào của X.
  - Ví dụ
    - Lược đồ R(A, B, C, D)
    - F={AB➔C; AB➔D; B➔D}
    - C phụ thuộc hàm đầy đủ vào {A,B}
    - D không phụ thuộc hàm đầy đủ vào {A,B}

Phụ thuộc bắc cầu

- Phụ thuộc hàm $X \rightarrow A$, A được gọi là **phụ thuộc bắc cầu** vào X nếu tồn tại Y để cho $X \rightarrow Y$, $Y \rightarrow A$, $Y \not\rightarrow X$ và $A \notin XY$
- Ví dụ:
  - F = {A➔B, B➔C}
  - A➔C: C phụ thuộc bắc cầu vào A

d. Bao đóng và phủ của tập các phụ thuộc hàm

- *Cho tập các phụ thuộc hàm F xác định trên R.*
  - **Bao đóng F^+** *của tập các phụ thuộc hàm F* là tập tất cả các phụ thuộc hàm được *suy diễn logic* từ F.
  - **Phủ G** của tập các phụ thuộc hàm F ($G \approx F$) là tập các phụ thuộc hàm xác định trên R sao cho $G^+ = F^+$.

X^+ ?

- Bao đóng X^+ của thuộc tính X đối với tập phụ thuộc hàm F là tất cả các thuộc tính A mà phụ thuộc hàm X➔A có thể được suy diễn logic từ F nhờ hệ tiên đề Amstrong.
- Một phụ thuộc hàm X➔Y thuộc F^+ nếu Y thuộc X^+: Kiểm tra X➔Y có thuộc F^+

Ý nghĩa của phụ thuộc hàm

- Chỉ ra các phụ thuộc dữ liệu/ràng buộc có thể xảy ra giữa tập thuộc tính của một lược đồ quan hệ.
  - Giúp xác định khóa tối thiểu, khóa chính của quan hệ.
  - Giúp chuẩn hóa lược đồ quan hệ

2. Chuẩn hóa lược đồ quan hệ

- Khái niệm
  - Là quá trình phân tách các lược đồ quan hệ thành các lược đồ quan hệ nhỏ hơn theo một số **tiêu chuẩn** nhằm loại bỏ việc lưu trữ dư thừa dữ liệu.
  - Phép tách thành các lược đồ quan hệ đơn giản hơn, nhỏ hơn phải đảm bảo **không làm mất mát thông tin**.

2.1. Các dạng chuẩn

- Dạng chuẩn 1
- Dạng chuẩn 2
- Dạng chuẩn 3
- Dạng chuẩn Boye-Codd
- *Chuẩn 4 và các dạng chuẩn khác*

a. Dạng chuẩn 1(1NF)

- **Định nghĩa**
  - Một lược đồ quan hệ R ở dạng chuẩn 1 nếu và chỉ nếu toàn bộ các miền giá trị của các thuộc tính trong R đều chỉ chứa các giá trị nguyên tố.
  - Một quan hệ xác định trên lược đồ quan hệ ở dạng chuẩn 1 được gọi là quan hệ ở dạng chuẩn 1.

Dạng chuẩn 1 (t)

- Một quan hệ thuộc dạng chuẩn 1 là một quan hệ trong đó mỗi miền giá trị của một thuộc tính chỉ chứa những giá trị nguyên tố (không phân chia được nữa).
- Một quan hệ thuộc dạng chuẩn 1 nếu mỗi một ô trong bảng chỉ chứa duy nhất một giá trị

Ví dụ

- Quan hệ CUNG_UNG_0 **chưa thuộc dạng chuẩn 1**

MaNCC	TenNCC	DiaChi	SanPham	Gia
1	Hải Hà	Hà Nội	Kẹo mềm Kẹo cứng Bánh	100 150 200
2	Kinh Đô	Hồ Chí Minh	Kẹo Bánh	120 200

Ví dụ(t)

- Quan hệ CUNG_UNG_1 **đã thuộc dạng chuẩn 1**

MaNCC	TenNCC	DiaChi	SanPham	Gia
1	Hải Hà	Hà Nội	Kẹo mềm	100
1	Hải Hà	Hà Nội	Kẹo cứng	150
1	Hải Hà	Hà Nội	Bánh	200
2	Kinh đô	Hồ Chí Minh	Kẹo	120
2	Kinh đô	Hồ Chí Minh	Bánh	200

b. Dạng chuẩn 2 (2NF)

- **Định nghĩa**
  - Một lược đồ quan hệ R được gọi là ở dạng chuẩn 2 nếu nó đã ở dạng chuẩn 1 và mọi thuộc tính không khóa đều phụ thuộc hàm đầy đủ vào khóa chính.
  - Một quan hệ xác định trên lược đồ quan hệ ở dạng chuẩn 2 được nói là quan hệ ở dạng chuẩn 2.

Ví dụ

- Quan hệ CUNG_UNG_1 **chưa thuộc dạng chuẩn 2**.

	MaNCC	TenNCC	DiaChi	SanPham	Gia
Lặp →	1	Hải Hà	Hà Nội	Kẹo mềm	100
	1	Hải Hà	Hà Nội	Kẹo cứng	150
	1	Hải Hà	Hà Nội	Bánh	200
Lặp →	2	Kinh đô	Hồ Chí Minh	Kẹo	120
	2	Kinh đô	Hồ Chí Minh	Bánh	200

Ví dụ(t)

- R(M, T, D, S, G) = MTDSG= Lược đồ của quan hệ CUNG_UNG_1
- Phụ thuộc hàm
  - M➔ TD, MS➔G
- MS: Khóa tối thiểu
  - M, S: Thuộc tính khóa
  - T, D, G: Thuộc tính không khóa
- **T, D không phụ thuộc hàm đầy đủ vào MS**

➔ Lược đồ quan hệ CUNG_UNG_1 không thuộc dạng chuẩn 2.

Ví dụ

MaNCC	TenNCC	DiaChi
1	Hải Hà	Hà Nội
2	Kinh đô	Hồ Chí Minh

CUNG_UNG_11

MaNCC	SanPham	Gia
1	Kẹo mềm	100
1	Kẹo cứng	150
1	Bánh	200
2	Kẹo	120
2	Bánh	200

CUNG_UNG_12

- Ví dụ 2 thành quan hệ CUNG_UNG_11 và CUNG_UNG_12 tách từ quan hệ CUNG_UNG_1 **đã thuộc dạng chuẩn 2**.
  - TenNCC, DiaChi phụ thuộc hàm đầy đủ vào MaNCC
  - Gia phụ thuộc hàm đầy đủ vào {MaNCC, SanPham}

c. Dạng chuẩn 3

- **Định nghĩa**
  - Một lược đồ quan hệ R được gọi là ở dạng chuẩn 3 nếu nó đã ở dạng chuẩn 2 và mọi thuộc tính không khóa của R đều chỉ phụ thuộc hàm duy nhất vào khóa chính.
  - Một quan hệ xác định trên lược đồ quan hệ ở dạng chuẩn ba được nói là quan hệ ở dạng chuẩn 3.

Ví dụ

MaNCC	TenNCC	DiaChi
1	Hải Hà	Hà Nội
2	Kinh đô	Hồ Chí Minh

CUNG_UNG_11

MaNCC	SanPham	Gia
1	Kẹo mềm	100
1	Kẹo cứng	150
1	Bánh	200
2	Kẹo	120
2	Bánh	200

CUNG_UNG_12

- Ví dụ 2 quan hệ CUNG_UNG_11 và CUNG_UNG_12 **đã thuộc dạng chuẩn 3**.
- MTDSG tách thành 1 lược đồ con MTD và MSG:
  - MTD: T, D phụ thuộc chỉ vào khóa M
  - MSG: G phụ thuộc hàm chỉ vào MS

Ví dụ

- Ví dụ
  - Lược đồ quan hệ:
    - R(Store, Item, Departement, Manager) ⇔ R(S,I,D,M)
  - Tập các phụ thuộc hàm:
    - F={SI➔D, SD➔M}
  - Khóa tối thiểu: SI
  - Lược đồ thuộc chuẩn 2: D, M phụ thuộc hàm đầy đủ vào SI
  - Lược đồ **không thuộc chuẩn 3**: M phụ thuộc bắc cầu vào SI
    - SI➔D; SI➔SD;SD➔M;SI➔M(*)

d. Dạng chuẩn Boye-Codd (BCNF)

- Chuẩn 3 không đáp ứng được những lược đồ quan hệ
  - Có nhiều hơn một khóa tối thiểu
  - Các khóa tối thiểu là khóa kép
  - Các khóa tối thiểu giao nhau
- **Định nghĩa**
  - Một lược đồ quan hệ R thuộc dạng chuẩn Boye-Codd khi và chỉ khi mọi xác định hàm đều là một khóa.

Ví dụ

- Lược đồ quan hệ: R(CITY, STREET, ZIP)
- Phụ thuộc hàm:
  - CITY, STREET ➔ ZIP, ZIP ➔ CITY
- Khóa tối thiểu:
  - {CITY, STREET}, {STREET, ZIP}
- Thuộc tính khóa:
  - CITY, STREET, ZIP ➔ không có thuộc tính không khóa, thuộc dạng chuẩn 3.
- *Xác định hàm* ZIP không phải là một khóa **➔ lược đồ không thuộc chuẩn Boye-Codd**

Ví dụ

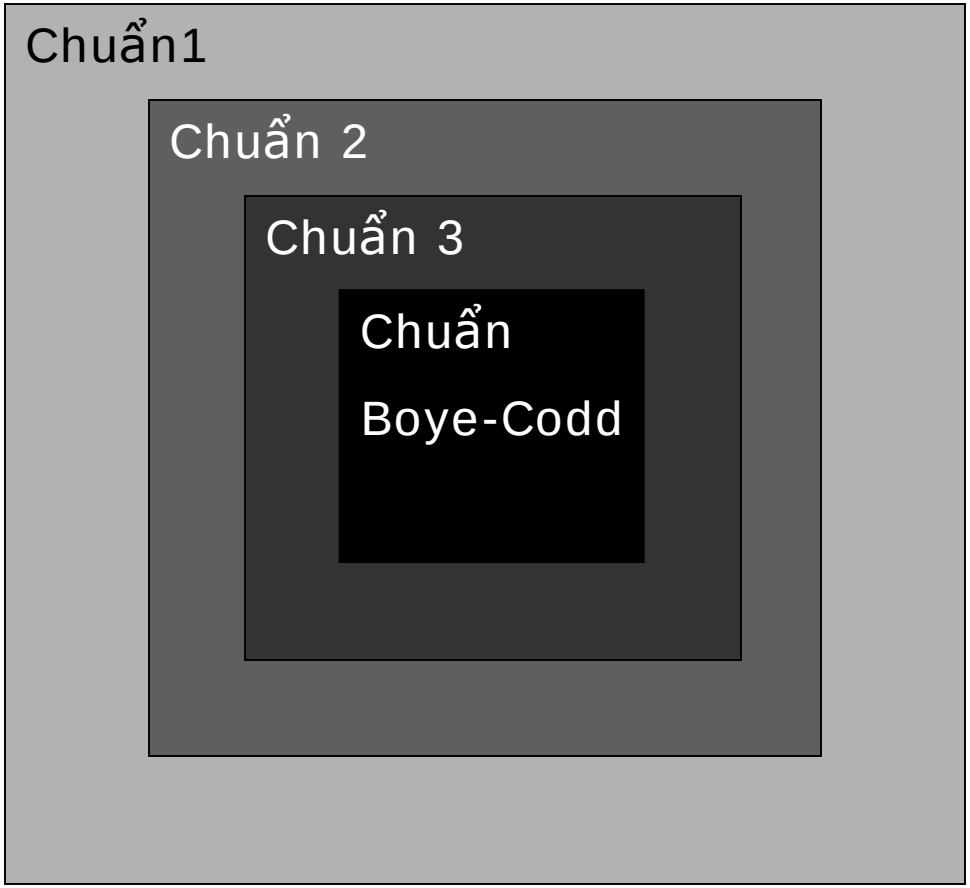
- 2 lược đồ con R1(CITY, ZIP) và R2(STREET, ZIP) **thuộc dạng chuẩn Boye-Codd**.

CITY	STREET	ZIP
Hà Nội	Láng	0423
Hà Nội	Phạm Hùng	0425
HCM	Võ Thị Sáu	0811

CITY	ZIP
Hà Nội	0423
Hà Nội	0425
HCM	0811

STREET	ZIP
Láng	0423
Phạm Hùng	0425
Võ Thị Sáu	0811

Quan hệ giữa các dạng chuẩn



2.2. Tách lược đồ quan hệ theo chuẩn

2.2.1. Tách bảo toàn thông tin và tách bảo toàn tập phụ thuộc hàm

2.2.2. Các thuật toán tách lược đồ quan hệ

2.2.3. Tách lược đồ quan hệ theo từng bước

2.2.1.Tách bảo toàn thông tin và tách bảo toàn tập phụ thuộc hàm

a. Phép tách bảo toàn thông tin

- Khái niệm
  - Phép tách bảo toàn thông tin là phép tách lược đồ quan hệ sao cho khi **kết nối tự nhiên** các quan hệ xác định trên các lược đồ con, kết quả cho lại quan hệ ban đầu.

Phép kết nối tự nhiên

- Phép ghép các cặp bộ của hai quan hệ trên các thuộc tính bằng nhau của hai quan hệ, một trong hai thuộc tính của phép so sánh "=" được loại bỏ sau khi thực hiện phép ghép.

- R(A, B, C)

a1	b1	1
a2	b2	2
a3	b3	3

- S(D, E)

1	e1
2	e2
3	e3

- R kết nối tự nhiên với s với C=D

a1	b1	1	e1
a2	b2	2	e2
a3	b3	3	e3

Ví dụ

- Lược đồ quan hệ MTDSG = (MaNCC, TenNCC, DiaChi, SanPham, Gia) ⇔(M, T, D, S, G);
- Tập phụ thuộc hàm F={M➔TD, MS➔G}
- Quan hệ xác định trên lược đồ MTDSG:

MaNCC	TenNCC	DiaChi	SanPham	Gia
1	Hải Hà	Hà Nội	Kẹo mềm	10
1	Hải Hà	Hà Nội	Kẹo cứng	15
1	Hải Hà	Hà Nội	Bánh ngọt	40
1	Hải Hà	Hà Nội	Bánh mỳ	16
2	Kinh Đô	Hồ Chí Minh	Kẹo	12
2	Kinh Đô	Hồ Chí Minh	Bánh quy	45
3	Kinh Đô	Đà Nẵng	Bánh mỳ	10
3	Kinh Đô	Đà Nẵng	Bánh quy	30

Ví dụ(t)

MaNCC	TenNCC	DiaChi
1	Hải Hà	Hà Nội
2	Kinh Đô	Hồ Chí Minh
3	Kinh Đô	Đà Nẵng

MaNCC	SanPham	Gia
1	Kẹo mềm	10
1	Kẹo cứng	15
1	Bánh ngọt	40
1	Bánh mỳ	16
2	Kẹo	12
2	Bánh quy	45
3	Bánh mỳ	10
3	Bánh quy	30

- Phép tách bảo toàn thông tin
  - Lược đồ con 1: MTD=(M, T, D)
  - Lược đồ con 2: MSG=(M, S, G)

Ví dụ(t)

MaNCC	TenNCC	DiaChi
1	Hải Hà	Hà Nội
2	Kinh Đô	Hồ Chí Minh
3	Kinh Đô	Đà Nẵng

TenNCC	SanPham	Gia
Hải Hà	Kẹo mềm	10
Hải Hà	Kẹo cứng	15
Hải Hà	Bánh ngọt	40
Hải Hà	Bánh mỳ	16
Kinh Đô	Kẹo	12
Kinh Đô	Bánh quy	45
Kinh Đô	Bánh mỳ	10
Kinh Đô	Bánh quy	30

- Phép tách mất mát thông tin
  - Lược đồ con 1: MTD=(M, T, D)
  - Lược đồ con 2: MSG=(M, S, G)

Thuật toán kiểm tra

- *Kiểm tra phép tách không mất mát thông tin của một lược đồ quan hệ thành nhiều lược đồ quan hệ con.*
  - *Vào*
    - Lược đồ quan hệ: $R=(A_1, A_2,... A_n)$
    - Tập phụ thuộc hàm F trên R
    - Phép tách $\rho=(R_1, R_2,... R_k)$
  - *Ra*
    - Một khẳng định rằng phép tách ρ có mất mát thông tin hay không?

Thuật toán kiểm tra (t)

- *Phương pháp*
  - Bước 1: Xây dựng một bảng k hàng, n cột
    - Hàng i tương ứng với R_i
    - Cột j tương ứng với thuộc tính A_j
    - Giá trị tại hàng i, cột j
      - a_j: Nếu A_j thuộc R_i
      - b_{ij}: Nếu A_j không thuộc R_i

Ví dụ (bước 1)

- Lược đồ quan hệ MTDSG
- Tập phụ thuộc hàm F={M➔TD, MS➔G}
- Tách thành hai lược đồ MTD, MSG

	M	T	D	S	G
R_1= MTD	a1	a2	a3	b14	b15
R_2= MSG	a1	b22	b23	a4	a5

Phương pháp

- Bước 2: Lặp
  - Áp dụng các phụ thuộc hàm cho bảng vừa được xây dựng: X➔Y: Nếu tồn tại hai hàng có cùng giá trị trên X thì làm bằng nhau các giá trị trên Y.
    - Nếu có giá trị một hàng thuộc Y là a_j thì các giá trị khác thuộc Y gán bằng a_j.
    - Nếu không gán bằng một trong các giá trị bij.
- Dừng khi các giá trị trong bảng không thể thay đổi được nữa

Ví dụ (bước 2)

	M	T	D	S	G
R_1 = MTD	**a1**	**a2**	**a3**	b14	b15
R_2 = MSG	**a1**	**b22**	**b23**	a4	a5

- Áp dụng phụ thuộc hàm M➔TD
  - Thay $b_{22} = a_2$
  - Thay $b_{23} = a_3$

Phương pháp

- Bước 3: Kiểm tra
  - Nếu trong bảng có một hàng gồm toàn ký hiệu a_1, a_2, a_3,... a_n thì phép tách là không mất mát thông tin.
  - Ngược lại phép tách là mất mát thông tin.

Ví dụ

	M	T	D	S	G
R_1 = MTD	a1	a2	a3	b14	b15
R_2 = MSG	**a1**	**a2**	**a3**	**a4**	**a5**

- Phép tách trên là không mất mát thông tin.

Định lý kiểm tra

- *Kiểm tra phép tách không mất mát thông tin của một lược đồ quan hệ thành hai lược đồ quan hệ con.*
  - Cho
    - lược đồ quan hệ R
    - Tập phụ thuộc hàm F trên R
    - $\rho = (R_1, R_2)$
  - Phép tách là không mất mát thông tin nếu $R_1 \cap R_2 \rightarrow R1 \backslash R2$ hoặc $R1 \cap R2 \rightarrow R2 \backslash R1$.

Ví dụ

- Lược đồ MTDSG tách thành hai lược đồ con MTD và MSG:
  - MTD∩MSG = M
  - MTD\MSG = TD
  - M➔TD thuộc F^+: Phép tách là bảo toàn thông tin.
- Lược đồ MTDSG tách thành hai lược đồ con MTD và TSG
  - MTD TSG = T
  - MTD∩SG = MD
  - TSG\MTD = SG
  - T➔MD và T➔SG không thuộc F^+: Phép tách là mất mát thông tin.

b. Phép tách bảo toàn tập phụ thuộc hàm

- Khái niệm
  - Phép tách lược đồ quan hệ R thành các lược đồ quan hệ con R_i là bảo toàn các tập phụ thuộc hàm nếu hợp của các phụ thuộc hàm là hình chiếu của F trên R_i suy diễn logic được tất cả các phụ thuộc hàm trong F.
  - Hình chiếu của F lên R_i là các phụ thuộc hàm X➔Y thỏa mãn:
    - X, Y thuộc R_i
    - X➔Y thuộc F^+

Ví dụ

- Cho
  - Lược đồ R = ACBCD
  - Tập phụ thuộc hàm F = { A➔B, C➔D}
  - Phép tách $\rho = (R_1,R_2)$: R_1= AB, R_2 = CD
- Phép tách bảo toàn tập phụ thuộc hàm
  - Hình chiếu của F lên R_1: A➔B, AB➔A, AB➔B ...
  - Hình chiếu của F lên R_2: C➔D, CD➔C, CD➔D ...
  - Các phụ thuộc hàm trong F có thể được suy diễn từ các hình chiếu này
- Phép tách không bảo toàn thông tin
  - $AB \cap CD = \Phi$
  - AB\CD = AB
  - CD\AB = CD

Ví dụ

- Cho
  - Lược đồ R = CSZ
  - Tập phụ thuộc hàm F={CS➔Z, Z➔C}
  - Phép tách ρ = (R_1,R_2): R_1= CZ, R_2 = SZ
- Phép tách không bảo toàn tập phụ thuộc hàm
  - Hình chiếu của F lên R_1: Z➔C, CZ➔C, CZ➔Z,...
  - Tập phụ thuộc hàm trong R_2: SZ➔S, SZ➔Z,...
  - Phụ thuộc hàm CS➔Z không thể được suy ra từ các hình chiếu này.
- Phép tách bảo toàn thông tin
  - CZ∩SZ ➔ CZ\SZ hay Z➔C

Ví dụ

- Cho
  - Lược đồ R = MTDSG
  - Tập phụ thuộc hàm F={M➔TD, MS➔G}
  - Phép tách ρ = (R_1 ,R_2): R_1= MTD, R_2 = MSG
- Phép tách bảo toàn tập phụ thuộc hàm
  - Hình chiếu của F lên R_1: M➔TD, …
  - Hình chiếu của F lên R_2: MS➔G,…
  - Các phụ thuộc hàm trong F có thể được suy diễn logic từ các hình chiếu này.
- Phép tách bảo toàn thông tin
  - Đã chứng minh

2.2.2. Các thuật toán tách lược đồ quan hệ

a. *Thuật toán tìm một khóa tối thiểu của lược đồ quan hệ dựa vào tập phụ thuộc hàm*

b. *Thuật toán tách không mất mát thông tin và bảo toàn tập phụ thuộc hàm về dạng chuẩn 3*

c. *Thuật toán tách không mất mát thông tin về dạng chuẩn Boye-Codd*

a. *Thuật toán tìm một khóa tối thiểu của lược đồ quan hệ dựa vào tập phụ thuộc hàm*

- Khóa của lược đồ quan hệ
  - Giá trị của tập thuộc tính khóa trên mỗi bộ của quan hệ là duy nhất
  - Mọi thuộc tính của quan hệ phải phụ thuộc hàm vào khóa.
- Thuật toán tìm một khóa tối thiểu
  - Loại bỏ dần từng thuộc tính thuộc khóa của quan hệ cho tới khi tập thuộc tính nhỏ nhất còn lại vẫn thỏa mãn là một khóa➔ khóa tối thiểu của quan hệ.

b. *Thuật toán tách không mất mát thông tin và bảo toàn tập phụ thuộc hàm về dạng chuẩn 3*

- *Vào:*
  - Lược đồ quan hệ R
  - Tập phụ thuộc hàm F (phủ tối thiểu)
- *Ra:*
  - Tập sơ đồ con, trong đó mỗi sơ đồ con
    - Thuộc dạng chuẩn 3
    - Phụ thuộc hàm là hình chiếu của F lên nó

b. *Thuật toán tách không mất mát thông tin và bảo toàn tập phụ thuộc hàm về dạng chuẩn 3 (t)*

- *Phương pháp:*
  - Nếu tồn tại một thuộc tính thuộc R không có mặt ở vế trái hay vế phải của bất kỳ phụ thuộc hàm nào thì tách thuộc tính này ra khỏi R.
  - Nếu tồn một tại phụ thuộc hàm liên quan tới mọi thuộc tính của R thì kết quả là R.
  - Nếu các phụ thuộc hàm dạng:
    - $X \rightarrow A$: lược đồ con ở dạng XA.
    - $X \rightarrow A_1, \ldots X \rightarrow A_n$: lược đồ con ở dạng $XA_1 \ldots A_n$

Ví dụ

- Cho lược đồ R = ABCDEG
- Tập phụ thuộc hàm F = {AB➔C, C➔A, BC➔D, ACD➔B, D➔EG, BE➔C, CG➔BD, CE➔AG}
- Tập phụ thuộc hàm tối thiểu F' = {AB➔C, C➔A, BC➔D, D➔E, D➔G, BE➔C, CG➔B, CE➔G}
- Phép tách ρ = (ABC, CA, BCD, DEG, BEC, CGB, CEG) = (ABC, BCD, DEG, BEC, CGB, CEG) gồm các lược đồ con đã ở dạng chuẩn 3 bảo toàn tập phụ thuộc hàm và bảo toàn thông tin.

c. *Thuật toán tách không mất mát thông tin về dạng chuẩn Boye-Codd*

- *Vào :*
  - Lược đồ quan hệ R
  - Tập phụ thuộc hàm F
- *Ra:*
  - Tập sơ đồ con, trong đó mỗi sơ đồ con
    - Thuộc dạng chuẩn Boye-Codd
    - Phụ thuộc hàm là hình chiếu của F lên nó

c. *Thuật toán tách không mất mát thông tin về dạng chuẩn Boye-Codd (t)*

- *<u>Phương pháp:</u>*
  - ρ = (R)
  - Lặp
    - S là một sơ đồ quan hệ trong ρ không ở dạng chuẩn Boye-Codd. Xét một phụ thuộc hàm X➔A của S. Nếu X không chứa khóa của S, A không thuộc X thì S được tách thành:
      - S_1 = A U{X}
      - S_2 = S\{A}
  - Dừng cho tới khi mọi sơ đồ con trong ρ đã thuộc dạng chuẩn Boye-Codd

2.2.3. Tách lược đồ quan hệ theo từng bước

- Quy tắc
  - *Quy tắc 1:* Loại bỏ các thuộc tính phụ thuộc chỉ một phần vào khóa chính.

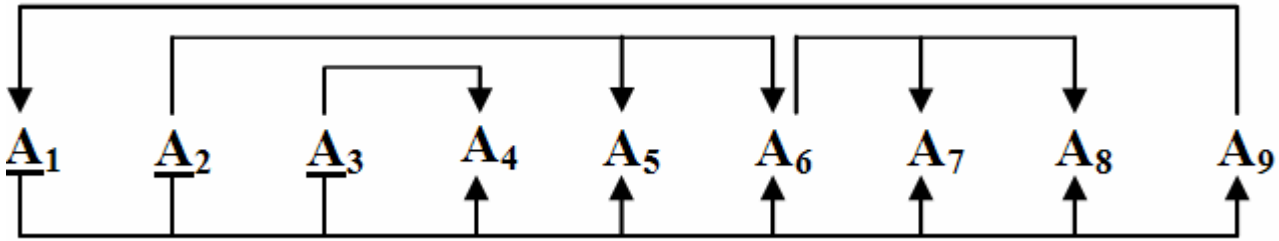
  ➔ Chuẩn hóa về 2NF
  - *Quy tắc 2:* Loại bỏ các thuộc tính không khóa không phụ thuộc vào khóa chính.

  ➔Chuẩn hóa về 3NF
  - Quy tắc 3: Loại bỏ các thuộc tính là giao của các khóa tối thiểu.

  ➔Chuẩn hóa về BCNF

Ví dụ

- Cho quan hệ R(A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9) trong đó $A_1A_2A_3$ là khóa với sơ đồ phụ thuộc hàm:



- Quan hệ R ở dạng chuẩn nào? Tại sao? Tách R thành các quan hệ ở dạng chuẩn BCNF.

3. Ràng buộc toàn vẹn trong CSDL quan hệ (Integrity Constraint)

3.1. Khái niệm ràng buộc toàn vẹn

- *Ràng buộc toàn vẹn là điều kiện bất biến không được vi phạm trong một CSDL.*
  - Các điều kiện mà mọi thể hiện của quan hệ phải thỏa mãn
  - RBTV xuất phát từ các quy tắc quản lý được áp đặt lên thế giới thực
- Ví dụ:
  - RBTV1: Mỗi bộ của quan hệ DON_DAT_HANG phải ứng với một đơn đặt hàng với mã đơn đặt hàng (MaDDH) duy nhất.
  - RBTV2: Mọi chi tiết về đơn đặt hàng phải có mã hàng (MaHang) thuộc về danh mục hàng.
  - RBTV3: Mã đơn đặt hàng (MaDDH) khác NULL.
  - RBTV4: Tổng các trị giá của các mặt hàng trong CHI_TIET_DON_HANG có cùng MaDDH phải bằng TongGT ghi trong DON_DAT_HANG
- Mục đích của RBTV
  - Đảm bảo tính nhất quán của dữ liệu(RBTV2, RBTV4)
  - Đảm bảo ngữ nghĩa nghĩa thực tế của dữ liệu(RBTV1, RBTV3)

Khái niệm ràng buộc toàn vẹn

- RBTV được xác định và mô tả trong quá trình thiết kế csdl. RBTV có 3 yếu tố chính:
  - Nội dung
  - Bối cảnh
  - Tầm ảnh hưởng
- RBTV được khai báo thông qua ngôn ngữ định nghĩa và thao tác dữ liệu và được hỗ trợ bởi hqt csdl.
  - Mệnh đề check, arssertion, triger
- RBTV được kiểm tra và xử lý bởi hqt csdl.

Nội dung của RBTV

- Được phát biểu
  - Ngôn ngữ tự nhiên: Đơn giản dễ hiểu
  - Ngôn ngữ hình thức: khó hiểu
    - Đại số quan hệ, phép tính quan hệ, mã giả
    - Biểu thức toán học
- Ví dụ:
  - RBTV5:
    - Mỗi nhân viên có một mã số dùng để phân biệt với những nhân viên khác

      $$\forall t_1, t_2 \in NHAN_VIEN(t_1 \neq t_2 \Rightarrow t_1.MaNV \neq t_2.MaNV)$$
  - RBTV6:
    - Mỗi nhân viên làm việc trong một phòng ban

      $$NHAN_VIEN[MaPhong] \in PHONG_BAN[MaPhong]$$
  - RBTV7:
    - Mỗi phòng phải có ít nhất một nhân viên

      $$\forall s \in PHONG_BAN(\exists t \in NHAN_VIEN(t.MaPhong = s.MaPhong))$$

Bối cảnh của RBTV

- Là những quan hệ mà RBTV có hiệu lực: Một hoặc nhiều quan hệ
- Ví dụ
  - RBTV5 có bối cảnh là quan hệ NHAN_VIEN
  - RBTV6, RBTV7 có bối cảnh là quan hệ NHAN_VIEN, PHONG_BAN

Tầm ảnh hưởng

- RBTV có thể bị vi phạm khi thực hiện các thao tác cập nhật trên bối cảnh: Thêm, xóa, sửa
- Bảng tầm ảnh hưởng dùng để xác định thời điểm cần kiểm tra RBTV

Xây dựng bảng tầm ảnh hưởng cho từng RBTV

- Nội dung mỗi ô
  - + : Cần phải kiểm tra RBTV
  - - : Không cần phải kiểm tra RBTV

Tên RBTV	Thêm	Xóa	Sửa
Quan hệ 1	+	+	-
...	...	...	...
Quan hệ k	+	-	-

Các quan hệ bối cảnh

Ví dụ

RBTV5	Thêm	Xóa	Sửa
NHAN_VIEN	+	-	-

RBTV6	Thêm	Xóa	Sửa
NHAN_VIEN	+	-	+
PHONG_BAN	-	+	-

RBTV7	Thêm	Xóa	Sửa
NHAN_VIEN	-	-	+
PHONG_BAN	+	-	-

Xây dựng bảng tầm ảnh hưởng tổng hợp

- Xây dựng trên cơ sở bảng tầm ảnh hưởng của các RBTV
- Để xác định thời điểm kiểm tra RBTV khi một thao tác cập nhật trên quan hệ nào đó được thực hiện

	Tên RBTV 1				Tên RBTV r		
	T	X	S	...	T	X	S
Quan hệ 1	+	+	-		+	-	+
...	...	...	...	...	...	...	...
Quan hệ n	-	-	+		+	+	+

Ví dụ

- Xây dựng bảng tầm ảnh hưởng cho các ràng buộc toàn vẹn RBTV1…, RBTV7 cho lược đồ CSDL quan hệ siêu thị M ?

Phân loại RBTV

- Dựa trên yếu tố bối cảnh của RBTV
  - RBTV có bối cảnh là một quan hệ/RBTV trên thuộc tính
    - RBTV miền giá trị
    - RBTV liên thuộc tính
    - RBTV liên bộ
  - RBTV có bối cảnh là nhiều quan hệ/RBTV trên quan hệ
    - RBTV tham chiếu
    - RBTV thuộc tính tổng hợp
    - RBTV liên thuộc tính
    - RBTV liên bộ

3.2. Ràng buộc toàn vẹn trên thuộc tính

a.Ràng buộc toàn vẹn về miền giá trị của một thuộc tính

- RBTV đặc tả tập giá trị có thể kết hợp với một thuộc tính.
- Ví dụ: quan hệ NHAN_VIEN
  - RBTV8: Tuổi của nhân viên trong công ty phải lớn hơn 18 và nhỏ hơn 65.

RBTV8 $\forall t \in NHAN_VIEN(18 \le t.Tuoi \le 65)$	T	X	S
NHAN_VIEN	+	-	+

b. RBTV liên thuộc tính

- RBTV có liên quan tới nhiều thuộc tính của một quan hệ. Thông thường đó là các phụ thuộc tính toán, hoặc một suy diễn từ giá trị của một hay nhiều thuộc tính trong cùng một bộ giá trị.
- Ví dụ
  - Quan hệ NHAN_VIEN
  - RBTV9: Nếu nhân viên có giới tính là nữ tuổi của nhân viên trong công ty phải lớn hơn 18 và nhỏ hơn 55.

    $\forall t \in NHAN_VIEN(18 \le t.Tuoi \le 55(t.GioiTinh = Nu))$

RBTV9	T	X	S
NHAN_VIEN	+	-	+

c. RBTV liên bộ, liên thuộc tính

- RBTV có liên quan tới nhiều bộ và có thể tới nhiều thuộc tính của (các) bộ giá trị trong một quan hệ.
- Ví dụ: RBTV5

3.3. Ràng buộc toàn vẹn trên quan hệ

a. RBTV tham chiếu/RBTV về phụ thuộc tồn tại (phụ thuộc về khóa ngoại)

- Bộ giá trị của quan hệ này được thêm vào một cách hợp lệ nếu tồn tại một bản ghi tương ứng trong một quan hệ khác.
- Đảm bảo rằng giá trị xuất hiện trong một quan hệ đối với một tập các thuộc tính đã cho cũng xuất hiện đối với một tập các thuộc tính nhất định trong một quan hệ khác.

a. Ràng buộc toàn vẹn về phụ thuộc tồn tại (phụ thuộc về khóa ngoại)

- Ví dụ:
  - RBTV1 : Mỗi bộ của CHI_TIET_DON_HANG phải có một đơn hàng với MaDDH tương ứng

$$\forall t \in CHI_TIET_DON_HANG(\exists u \in DON_HANG(t.MaDDH = u.MaDDH))$$

RVTV4	T	X	S
CHI_TIET_DON_HANG	+	-	-
DON_HANG	-	+	-

Ví dụ

- RBTV2 : Mỗi bộ của CHI_TIET_DON_HANG phải có MaHang thuộc về danh mục hàng trong quan hệ MAT_HANG
- Biểu diễn hình thức ? Bảng xác định tầm ảnh hưởng?

b. *Ràng buộc toàn vẹn tổng hợp*

- Khi có sự hiện diện của 1 thuộc tính mang tính chất tổng hợp (tức là giá trị của thuộc tính có thể được tính toán từ giá trị của các thuộc tính khác trên một hay nhiều bộ giá trị của các quan hệ trong CSDL)
- Ví dụ:
  - DON_DAT_HANG(**MaDDH**, NgayLap, TenKH, TongGT)
  - CHI_TIET_DON_HANG(**MaDDH**, **TenSanPham**, SoLuong, Giatri).
  - RBTV4: Tổng các trị giá của các mặt hàng trong CHI_TIET_DON_HANG có cùng MaDDH phải bằng TongGT ghi trong DON_DAT_HANG .

$$\forall t \in DON_DAT_HANG(t.TongGT = \sum u.GiaTri(u \in CHI_TIET_DON_HANG \wedge u.MaDDH = t.MaDDH))$$

RVTV4	T	X	S
CHI_TIET_DON_HANG	+	+	+
DON_DAT_HANG	+	-	+

c. RBTV liên thuộc tính

- Mối quan hệ giữa các thuộc tính trong nhiều lược đồ quan hệ
- Ví dụ
  - Quan hệ:
    - NHAN_VIEN (**TenNV**, Luong, NgaySinh TenPhong)
    - PHONG_BAN(**TenPhong**, MaPhong, NguoiQuanLy, NgayNhanChuc)
  - RBTV 10:
    - Ngày nhận chức của trưởng phòng phải lớn hơn ngày sinh
    - Biểu diễn hình thức. Bảng xác định tầm ảnh hưởng?

d.Ràng buộc toàn vẹn liên bộ

- Mối quan hệ giữa các bộ trên nhiều lược đồ quan hệ
- Ví dụ
  - Quan hệ NHAN_VIEN, PHONG_BAN
  - RBTV11:
    - Lương của nhân viên không được cao hơn lương trưởng phòng
    - Biểu diễn hình thức? Bảng xác định tầm ảnh hưởng?

Chỉ mục

- Functional dependency : Phụ thuộc hàm
- Functional determinant: Xác định hàm
- Normal Form: Dạng chuẩn

Phụ lục

- Thuật toán tìm phủ tối thiểu của một tập phụ thuộc hàm
  - Thuật toán 4.2 (128, Nguyên lý của các hệ csdl)

Hệ quản trị cơ sở dữ liệu

CHƯƠNG VI
HỆ THỐNG PHỤC HỒI
(Recovery system)

MỤC ĐÍCH

Một hệ thống máy tính, cũng giống như các thiết bị cơ - điện khác, luôn có nguy cơ bị hỏng hóc do nhiều nguyên nhân hư đĩa, mất nguồn, lỗi phần mềm v..v... Điều này dẫn đến hậu quả là sự mất thông tin. Vì vậy, hệ quản trị cơ sở dữ liệu phải có các cơ chế đáp ứng lại nguy cơ hệ thống bị hỏng hóc, nhằm đảm bảo ***tính nguyên tử và tính lâu bền*** của các giao dịch. Chương này trình bày các nguyên lý của một hệ thống phục hồi nhằm khôi phục CSDL đến một trạng thái nhất quán trước khi xảy ra sự cố.

YÊU CẦU

Hiểu rõ các sự cố có thể xảy ra trong đời sống của một cơ sở dữ liệu, các nguyên nhân của sự không nhất quán dữ liệu.

Hiểu các kỹ thuật phục hồi, các ưu nhược điểm của mỗi kỹ thuật.

PHÂN LỚP HỎNG HÓC:

Có nhiều kiểu hỏng hóc có thể xảy đến với hệ thống, mỗi một trong chúng cần được ứng xử một cách riêng biệt. Trong chương này ta chỉ xét các kiểu hỏng hóc sau:

- **Hỏng hóc trong giao dịch**: Có hai loại lỗi làm cho giao dịch bị hỏng hóc: **1. Lỗi luận lý**: Giao dịch không thể tiếp tục thực hiện bình thường được nữa do một số điều kiện bên trong không được thoả. ví dụ như: dữ liệu đầu vào không đúng, không tìm thấy dữ liệu, trào dữ liệu hoặc do việc sử dụng tài nguyên vượt hạn định. **2. Lỗi hệ thống:** Hệ thống rơi vào trạng thái không mong muốn ví dụ như trạng thái deadlock.
- **Hệ thống bị hư hỏng:** Có một phần cứng sai chức năng hoặc có một sai sót trong phần mềm cơ sở dữ liệu hay hệ điều hành.
- **Đĩa bị hư hỏng:** Một khối đĩa bị mất nội dung.

Để hệ thống có thể đề ra được chiến lược phục hồi lỗi phù hợp, trước tiên cần phải xác định các loại hỏng hóc trên các thiết bị lưu trữ dữ liệu. Sau đó, cần xác định những hỏng hóc này ảnh hưởng như thế nào đến nội dung cơ sở dữ liệu. Nhiệm vụ quan trọng sau cùng là đề ra các giải pháp nhằm đảm bảo tính nhất quán của cơ sở dữ liệu và tính nguyên tử của giao dịch mỗi khi hỏng hóc đã phát sinh. Các giải pháp này thường được gọi là các giải thuật phục hồi (recovery algorithms).

Các giải thuật phục hồi gồm có hai phần:

1. Các hành động được thực hiện trong suốt quá trình hoạt động bình thường của giao dịch nhằm đảm bảo có đầy đủ thông tin cho việc phục hồi sau này.
2. Các hành động được thực hiện sau khi lỗi phát sinh. Nhằm khôi phục nội dung của cơ sở dữ liệu trở về một trạng thái trước đó, và trạng thái này thoã mãn được các yêu cầu về tính nhất quán của cơ sở dữ liệu, tính bền và tính nguyên tử của giao dịch .

CẤU TRÚC LƯU TRỮ:

Như đã xét trong chương II, các hạng mục dữ liệu khác nhau của cơ sở dữ liệu có thể được lưu trên nhiều phương tiện lưu trữ khác nhau. Để nắm được cách thức đảm bảo tính nguyên tử và tính lâu bền của một giao dịch, cần phải có cái nhìn sâu hơn về các loại thiết bị lưu trữ dữ liệu và cách thức truy xuất chúng.

CÁC LOẠI LƯU TRỮ:

- **Lưu trữ không ổn định (volatile storage):** Thông tin lưu trong thiết bị lưu trữ không ổn định sẽ bị mất khi hệ thống bị hỏng hóc. Ví dụ của thiết bị lưu trữ không ổn định là: bộ nhớ chính, bộ nhớ cache. Sự truy cập đến các thiết bị lưu trữ không ổn định là cực nhanh. Lý do: một là: do tính chất của bộ nhớ cho phép như vậy; hai là: có thể truy xuất trực tiếp các hạng mục dữ liệu chứa trong nó.
- **Lưu trữ ổn định (nonvolatile storage):** Thông tin lưu trữ trong thiết bị lưu trữ ổn định thường không bị mất khi hệ thống bị sự cố. Tuy nhiên, nguy cơ bản thân thiết bị lưu trữ ổn định bị hỏng vẫn có thể xảy ra. Ví dụ của thiết bị lưu trữ ổn định là: đĩa từ và băng từ. Trong hầu hết các hệ cơ sở dữ liệu, thiết bị lưu trữ ổn định thường được dùng là đĩa từ. Các loại thiết bị lưu trữ ổn định khác được dùng để lưu trữ phòng hờ (back up) dữ liệu.

- **Lưu trữ bền (stable storage):** Theo lý thuyết thì thông tin chứa trong thiết bị lưu trữ bền ***không bao giờ*** bị mất khi hệ thống bị hư hỏng. Tuy nhiên, trong thực tế, ta khó lòng tạo ra được một thiết bị đạt được tính chất lý tưởng như vậy. Chỉ có giải pháp tăng cường độ bền mà thôi.

THỰC THI LƯU TRỮ BỀN:

Tiêu chí để thực hiện việc lưu trữ bền là nhân bản thông tin cần thiết trong một vài phương tiện lưu trữ ổn định khác nhau với các phương thức hỏng hóc độc lập và cập nhật các phiên bản thông tin này một cách có tổ chức, sao cho dù có lỗi xuất hiện trong quá trình chuyển dữ liệu, thông tin vẫn không bị hư hại.

- Các hệ thống RAID đảm bảo rằng việc hỏng hóc của một đĩa không gây sự mất dữ liệu. Dạng thức đơn giản và nhanh nhất của RAID là dùng đĩa gương (mirrored disk). Các dạng thức khác giúp tiết kiệm chi phí, nhưng cái giá phải trả là thời gian đọc ghi chậm hơn.
- Tuy nhiên các hệ thống RAID vẫn không đảm bảo được tính an toàn dữ liệu khi gặp phải tai họa như: cháy nổ, lụt lội. Người ta đề nghị một hệ thống lưu trữ mới an toàn hơn hoạt động theo nguyên tắc sau: Sao lưu dữ liệu sang một vài vị trí địa lý khác nhau thông qua mạng máy tính.

Sau đây là cách thức đảm bảo thông tin lưu trữ không bị lỗi trong quá trình đọc ghi dữ liệu:

Việc chuyển một khối dữ liệu giữa bộ nhớ và đĩa có thể dẫn đến kết quả:

- ***Thành công hoàn toàn***: Thông tin được chuyển đến đích an toàn.
- ***Bị lỗi một phần***: Có lỗi xuất hiện trong quá trình chuyển dữ liệu và khối đích chứa thông tin không đúng.
- ***Bị lỗi hoàn toàn***: Lỗi xuất hiện ngay ở giai đoạn đầu của quá trình truyền dữ liệu. Khối đích giữ nguyên như ban đầu.

Nếu có lỗi xuất hiện trong quá trình truyền dữ liệu, hệ thống phải phát hiện được và thực thi thủ tục phục hồi lỗi. Để làm được như vậy, hệ thống phải duy trì hai khối dữ liệu vật lý cho mỗ khối dữ liệu luận lý. (Trong tình huống dùng hệ thống đĩa gương thì hai khối vật lý này ở cùng một địa điểm, trong tình huống dùng hệ thống sao lưu từ xa, hai khối này ở hai địa điểm khác nhau).

Một thao tác ghi dữ liệu được thực thi như sau:

1. Viết thông tin lên khối vật lý thứ nhất.
2. Khi hành động ghi thứ nhất thành công, tiếp tục ghi phần thông tin trên lên khối vật lý thứ hai.
3. Thao tác ghi được coi là thành công khi thao tác ghi thứ hai thành công.

Trong quá trình phục hồi, từng cặp khối vật lý được kiểm tra:

1. Nếu nội dung của cả hai như nhau và không có lỗi có thể phát hiện, khi đó không cần làm gì thêm.
2. Nếu một trong hai khối có lỗi phát hiện được, khi đó thay thế khối bị lỗi bởi nội dung của khối còn lại.
3. Nếu cả hai khối không có lỗi phát hiện được, nhưng nội dung của chúng khác nhau, thay thế khối thứ nhất bởi khối thứ hai.

Yêu cầu phải so sánh từng cặp khối vật lý một đòi hỏi phải mất nhiều thời gian. Người ta có thể cải thiện tình huống này bằng cách lưu vết những thao tác viết khối trong tiến trình thực thi. Khi phục hồi, chỉ những khối nào thao tác ghi ở trong tiến trình thực thi mới cần được đem so sánh. Giao thức để viết ra một khối đến một site xa tương tự như viết khối trong hệ thống đĩa gương..

TRUY CẬP DỮ LIỆU

Như đã xét trong chương II, hệ cơ sở dữ liệu nằm thường trực trên các thiết bị lưu trữ ổn định (thường là đĩa từ) và thường được phân thành các đơn vị lưu trữ kích thước cố định được gọi là khối (blocks). Khối là đơn vị truyền nhận dữ liệu từ/ra đĩa. Một khối có thể chứa vài hạng mục dữ liệu. Ta giả thiết không có hạng mục dữ liệu nào trải ra trên nhiều hơn một khối.

Các giao dịch nhập (input) thông tin từ đĩa vào bộ nhớ chính và xuất (output) thông tin theo chiều ngược lại. Các thao tác nhập/xuất này được thực hiện theo đơn vị khối. Khối nằm trên đĩa được gọi là khối vật lý (physical block), khối được trữ tạm trong bộ nhớ chính được gọi là khối đệm (buffer block). Vùng bộ nhớ tạm chứa các khối dữ liệu được gọi là vùng đệm đĩa (disk buffer).

Việc di chuyển khối giữa đĩa và bộ nhớ được thực hiện thông qua hai thao tác:

1. **Input(B)** chuyển khối vật lý B vào bộ nhớ chính.
2. **Output(B)** chuyển khối đệm B ra đĩa và thay thế cho khối vật lý tương ứng ở đó.

Hình dưới đây sẽ mô phỏng cho hai thao tác này

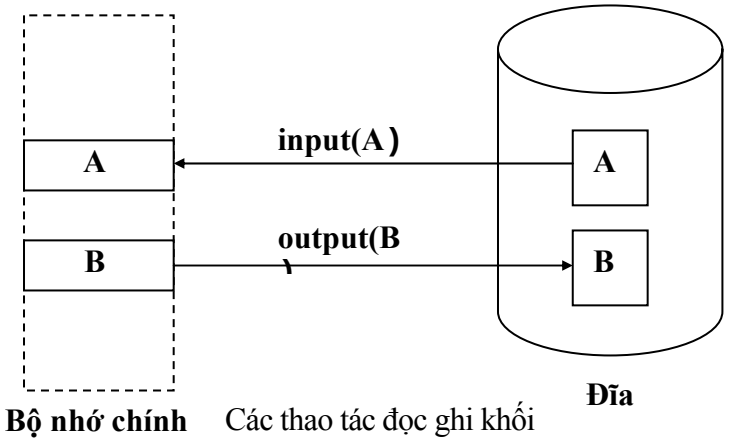


figure VI- 1

Mỗi giao dịch T_i có một vùng làm việc riêng ở đó các bản sao của tất cả các hạng mục dữ liệu được truy xuất và cập nhật được lưu giữ. Vùng làm việc này được tạo ra khi giao dịch khởi động. Nó bị xoá đi khi giao dịch bàn giao (commit) hoặc huỷ bỏ (abort). Mỗi hạng mục dữ liệu ***x*** được trữ trong vùng làm việc của giao dịch T_i sẽ được ký hiệu là x_i. Giao dịch T_i trao đổi với hệ cơ sở dữ liệu bằng cách chuyển dữ liệu đến/ra vùng làm việc của nó sang vùng đệm của hệ thống.

Hai thao tác dùng để chuyển dữ liệu:

1. **read(X)** gán giá trị của hạng mục dữ liệu X cho biến cục bộ x_i. Thao tác này được thực hiện như sau:
   a. Nếu khối B_X chứa X không có trong bộ nhớ chính thì thực hiện thao tác **input(B_X).**
   b. Gán cho x_i giá trị của X trong khối đệm.
2. **write(X)** gán giá trị của biến cục bộ x_i cho hạng mục dữ liệu X trong khối đệm. Thao tác này được thực hiện như sau:
   a. Nếu khối B_X chứa X không có trong bộ nhớ thì thực hiện thao tác **input(B_X).**
   b. Gán giá trị của x_i cho X trong vùng đệm B_X.

Chú ý rằng cả hai thao tác đều có thể đòi hỏi chuyển một khối từ đĩa vào bộ nhớ chính nhưng không yêu cầu chuyển một khối từ bộ nhớ chính ra đĩa.

Đôi khi một khối đệm bị ghi bắt buộc ra đĩa do bộ quản lý vùng đệm cần không gian bộ nhớ cho các mục đích khác hoặc do hệ cơ sở dữ liệu muốn phản ánh những thay đổi trong khối dữ

liệu B trên đĩa. Khi hệ cơ sở dữ liệu thực hiện thao tác **Output(B)** ta nói nó đã xuất bắt buộc khối đệm B ra đĩa.

Khi một giao dịch cần truy xuất hạng mục dữ liệu X lần đầu, nó phải thực hiện **Read(X)**. Khi đó tất cả các cập nhật đối với X được thực hiện trên x_i. Sau khi giao dịch truy xuất X lần cuối, nó thực hiện **Write(X)** để ghi lại sự thay đổi của X trong CSDL.

Không nhất thiết phải thực hiện thao tác **Output(B_X)** ngay sau khi thao tác **write(X)** hoàn thành. Lý do là: khối đệm B_X có thể còn chứa các hạng mục dữ liệu khác đang được truy xuất. Nếu hệ thống bị hư hỏng ngay sau khi thao tác **write(X)** hoàn thành, nhưng trước khi thực hiện thao tác **Output(B_X)**, giá trị mới của X sẽ không bao giờ được ghi ra đĩa, do đó, nó bị mất!

PHỤC HỒI VÀ TÍNH NGUYÊN TỬ:

Trở lại với ví dụ đơn giản về hệ thống ngân hàng: Giao dịch T_i thực hiện việc chuyển \$50 từ tài khoản A sang tài khoản B. Giả sử giá trị ban đầu của các tài khoản A và B là \$1000 và \$2000. Giả sử hệ thống bị hư hỏng trong khi T_i đang thực thi: sau khi thao tác **output(B_A)** được thực hiện và trước khi thực hiện thao tác **output(B_B)** (B_A và B_B là hai khối đệm chứa hai hạng mục A và B). Người ta có thể thực hiện một trong hai giải pháp phục hồi sau:

1. Thực hiện lại T_i. Thủ tục này sẽ dẫn đến kết quả: giá trị của A là \$900 thay vì phải là \$950. Do đó, hệ thống ở trong trạng thái không nhất quán.
2. Không thực hiện lại T_i . Kết quả: giá trị của A và B tương ứng sẽ là \$950 và \$2000. Hệ thống cũng trong trạng thái không nhất quán.

Vấn đề phát sinh ở chỗ: T_i thực hiện nhiều thao tác sửa đổi nội dung cơ sở dữ liệu, do đó cần nhiều thao tác xuất dữ liệu ra đĩa, nhưng lỗi phát sinh không cho phép tất cả các thao tác xuất dữ liệu hoàn thành.

Giải pháp nhằm đạt được tính nguyên tử là: trước khi thực hiện các thao tác sửa đổi cơ sở dữ liệu, cần ghi ra các thiết bị lưu trữ bền những thông tin mô tả các sửa đổi này. Cụ thể của giải pháp trên sẽ được trình bày trong các phần V.4, V.5 và V.6.

PHỤC HỒI DỰA TRÊN SỔ GHI LỘ TRÌNH (Log-based recovery)

Một cấu trúc thường được dùng để ghi lại những thay đổi trên cơ sở dữ liệu là sổ ghi lộ trình (log). Log là một dãy các mẩu tin lộ trình (log records). Một thao tác cập nhật trên cơ sở dữ liệu sẽ được ghi nhận bằng một log record. Một log record kiểu mẫu chứa các trường sau:

- **Định danh giao dịch (transaction identifier)**: định danh duy nhất của giao dịch thực hiện hoạt động **write**
- **Định danh hạng mục dữ liệu (Data-item identifier)**: định danh duy nhất của hạng mục dữ liệu được viết (thường là vị trí của hạng mục dữ liệu trên đĩa)
- **Giá trị cũ (Old value)**: giá trị của hạng mục dữ liệu trước khi viết
- **Giá trị mới (New value)**: giá trị hạng mục dữ liệu sẽ có sau khi viết

Có một vài log record đặc biệt mang các ý nghĩa riêng. Bảng sau đây chỉ ra một số loại log record và ý nghĩa của chúng:

LOẠI LOG RECORD	Ý NGHĨA
< T_i **start** >	Giao dịch T_i đã khởi động.
< T_i, X_j, V_1, V_2 >	Giao dịch T_i đã thực hiện thao tác ghi trên hạng mục dữ liệu X_j, X_j có giá trị V_1 trước khi ghi và nhận giá trị V_2 sau khi ghi.
< T_i **commit** >	Giao dịch T_i đã bàn giao.
< T_i **abort** >	Giao dịch Ti đã huỷ bỏ.

Mỗi khi một giao dịch thực hiện một thao tác ghi, trước tiên phải tạo ra một log record cho thao tác ghi đó (trong log file), trước khi giao dịch thay đổi cơ sở dữ liệu. Như vậy, hệ thống có cơ sở để huỷ bỏ (undo) một thay đổi đã được làm trên cơ sở dữ liệu bằng cách sử dụng trường **Old-value** trong log record.

log phải được lưu trong những thiết bị lưu trữ bền. Mỗi một log record mới ngầm định sẽ được thêm vào cuối tập tin log.

CẬP NHẬT TRÌ HOÃN CƠ SỞ DỮ LIỆU (Deferred Database Modification):

Kỹ thuật cập nhật trì hoãn đảm bảo tính nguyên tử của giao dịch bằng cách ghi lại tất cả những sửa đổi cơ sở dữ liệu vào sổ ghi lộ trình (log), nhưng trì hoãn sự thực hiện tất cả các thao tác viết dữ liệu ra đĩa của giao dịch cho đến khi giao dịch bàn giao một phần (partially commits). Nhắc lại rằng: một giao dịch được gọi là bàn giao một phần khi hành động cuối cùng của nó được thực hiện xong. Kỹ thuật cập nhật trì hoãn được trình bày trong phần này giả thiết rằng các giao dịch được thực hiện một cách tuần tự.

Khi giao dịch bàn giao một phần, thông tin trên log kết hợp với giao dịch được sử dụng trong việc viết trì hoãn. Nếu hệ thống có sự cố trước khi giao dịch hoàn thành việc thực hiện của nó hoặc giao dịch bị bỏ dở khi đó thông tin trên log bị bỏ lơ.

Sự thực thi của một giao dịch được tiến triển như sau:

- Trước khi giao dịch T_i bắt đầu thực hiện, một mẫu tin < T_i **start** > được ghi ra sổ lộ trình.
- Trước khi T_i thực hiện thao tác **write(X),** một mẫu tin < T_i, X, V_2 > được ghi ra sổ lộ trình.
- Cuối cùng, khi giao dịch T_i bàn giao một phần, mẫu tin < T_i **commit** > được ghi ra sổ lộ trình.

Khi giao dịch bàn giao một phần, các mẫu tin trong sổ lộ trình kết hợp với giao dịch sẽ được sử dụng để thực hiện việc ghi trì hoãn các hạng mục dữ liệu ra đĩa. Nhà thiết kế phải đảm bảo rằng, trước khi hoạt động ghi hạng mục dữ liệu diễn ra, các mẫu tin log đã được ghi thành công ra các thiết bị lưu trữ bền. Ngoài ra cũng cần để ý: kỹ thuật cập nhật trì hoãn chỉ cần ghi lại giá trị mới của hạng mục dữ liệu (V_2) mà thôi.

Để minh hoạ, ta sử dụng ví dụ hệ thống ngân hàng đơn giản. Gọi T_0 là giao dịch có nhiệm vụ chuyển $50 từ tài khoản A sang tài khoản B, T_1 là giao dịch có nhiệm vụ rút $100 từ tài khoản C. Giả sử giá trị ban đầu của các tài khoản A, B, C là $1000, $2000 và $700. Hành động của T_0 và T_1 được mô tả như sau:

T_0	T_1
read(A)	Read(C)
A:=A-50	C:=C-100
write(A)	write(C)
read(B)	
B:=B+50	
write(B)	

Giả thiết các giao dịch được thực hiện tuần tự: T_0 rồi tới T_1. Một phần của sổ lộ trình ghi lại những thông tin liên quan đến hoạt động của hai giao dịch trên được cho trong bảng dưới đây:

<T_0 start>
<T_0 ,A, 950>
<T_0 ,B, 2050>

<T_0 commit>
<T_1 start>
<T_1 ,C, 600>
<T_1 commit>

figure VI- 2

Sau khi có sự cố xảy ra, hệ thống phục hồi sẽ tham khảo sổ lộ trình để chọn ra những giao dịch nào cần được làm lại (redo). Giao dịch T_i cần được làm lại khi và chỉ khi sổ nhật ký có chứa cả hai mẫu tin <T_i start> và <T_i commit>.

Thủ tục làm lại giao dịch T_i như sau:

- **redo(T_i)** đặt giá trị mới cho tất cả các hạng mục dữ liệu được cập nhật bởi giao dịch T_i. Các giá trị mới sẽ được tìm thấy trong sổ lộ trình (log).

Hoạt động redo phải đồng hiệu lực (idempotent) có nghĩa là việc thực hiện nó nhiều lần tương đương với việc thực hiện nó một lần.

Trở lại ví dụ vừa nêu, ta có bảng mô tả trạng thái của sổ ghi lộ trình và cơ sở dữ liệu như sau:

LOG	CƠ SỞ DỮ LIỆU
<T_0 start>	
<T_0 ,A, 950>	
< T_0 ,B, 2050>	
<T_0 commit>	
	A=950
	B=2050
<T_1 start>	
<T_1 ,C, 600>	
<T_1 commit>	
	C=600

figure VI- 3

Sau đây là một số tình huống mô phỏng:

1. Giả sử lỗi hệ thống xảy ra sau khi mẫu tin log cho hành động **write(B)** của giao dịch T_0 vừa được ghi ra thiết bị lưu trữ bền. Khi hệ thống khởi động trở lại, sẽ không có hành động "thực hiện lại giao dịch" nào cần phải làm, do không có mẫu tin ghi **commit** nào xuất hiện trong sổ lộ trình. Nghĩa là giá trị của A,B và C vẫn giữ nguyên là $1000, $2000 và $700.
2. Giả sử lỗi hệ thống xảy ra sau khi mẫu tin log cho hành động **write(C)** của giao dịch T_1 vừa được ghi ra thiết bị lưu trữ bền. Khi hệ thống hoạt động trở lại, thủ tục **redo(T_0)** sẽ được thực hiện do có sự xuất hiện của mẫu tin **<T_0 commit>** trong sổ lộ trình. Sau khi thủ tục này được thực thi, giá trị của A và B sẽ là $950 và $2050.

CẬP NHẬT TỨC THỜI CƠ SỞ DỮ LIỆU (Immediate Database Modification):

Kỹ thuật cập nhật tức thời cho phép các thao tác sửa đổi cơ sở dữ liệu có quyền xuất dữ liệu tức thời ra đĩa trong khi giao dịch vẫn còn ở trong trạng thái hoạt động (active state). Hành động thay đổi nội dung dữ liệu tức thời của các giao dịch đang hoạt động được gọi là "những thay đổi chưa được bàn giao" (uncommitted modifications).

Sự thực thi của một giao dịch được tiến hành như sau:

- Trước khi giao dịch T_i bắt đầu sự thực hiện, một mẫu tin < T_i **start** > được ghi ra sổ lộ trình.

- Trước khi T_i thực hiện thao tác **write(X),** một mẫu tin < T_i, X, V_1, V_2 > được ghi ra sổ lộ trình.
- Cuối cùng, khi giao dịch T_i bàn giao một phần, mẫu tin < T_i **commit** > được ghi ra sổ lộ trình.

Cần phải đảm bảo rằng, trước khi hoạt động ghi hạng mục dữ liệu diễn ra, các mẫu tin log đã được ghi thành công ra các thiết bị lưu trữ bền. Ngoài ra, cũng cần chú ý là mẫu tin log cho hành động **write(X)** của giao dịch T_i, tức là mẫu tin <T_i, X, V_1, V_2> có chứa cả hai giá trị mới (V_2) và cũ (V_1) của hạng mục dữ liệu X.

Trở lại với ví dụ trong phần V.4.1, ta có một phần của sổ lộ trình liên quan đến các hoạt động của T_0 và T_1 như sau:

<T_0 start>
<T_0 , A, 1000, 950>
<T_0 , B, 2000, 2050>
<T_0 commit>
<T_1 start>
<T_1 , C, 700, 600>
<T_1 commit>

figure VI- 4

Bảng mô tả trạng thái của sổ ghi lộ trình và cơ sở dữ liệu như sau:

LOG	CƠ SỞ DỮ LIỆU
<T_0 start>	
<T_0 , A, 1000, 950>	
< T_0 , B, 2000, 2050>	
	A=950
	B=2050
<T_0 commit>	
<T_1 start>	
<T_1 , C, 700, 600>	
	C=600
<T_1 commit>	

figure VI- 5

Kỹ thuật cập nhật tức thời sử dụng hai thủ tục khôi phục sau lỗi:

- **undo(T_i)** đặt lại giá trị cũ cho tất cả các hạng mục dữ liệu được cập nhật bởi giao dịch T_i. Các giá trị cũ sẽ được tìm thấy trong sổ lộ trình (log).
- **redo(T_i)** đặt giá trị mới cho tất cả các hạng mục dữ liệu được cập nhật bởi giao dịch T_i. Các giá trị mới sẽ được tìm thấy trong sổ lộ trình (log).

Sau khi lỗi xuất hiện, hệ thống phục hồi tham khảo sổ ghi để quyết định những giao dịch nào cần được làm lại (redo) và những giao dịch nào cần được huỷ bỏ (undo).

- Giao dịch T_i cần được huỷ bỏ khi sổ ghi chứa mẫu tin <T_i start> nhưng không có mẫu tin <T_i commit>.
- Giao dịch T_i cần được làm lại khi sổ ghi có chứa cả mẫu tin <T_i start> lẫn mẫu tin <T_i commit>.

Sau đây là một số tình huống mô phỏng:

1. Giả sử lỗi hệ thống xảy ra sau khi mẫu tin log cho hành động **write(B)** của giao dịch T_0 vừa được ghi ra thiết bị lưu trữ bền. Khi hệ thống khởi động trở lại, nó sẽ tìm thấy mẫu tin <T_0 start> trong sổ ghi, nhưng không có mẫu tin <T_0 commit> tương ứng. Do đó giao dịch T_0 cần phải được huỷ bỏ. Nghĩa là thủ tục **undo(T_0)** sẽ được gọi và giá trị của A,B và C vẫn giữ nguyên là $1000, $2000 và $700.
2. Giả sử lỗi hệ thống xảy ra sau khi mẫu tin log cho hành động **write(C)** của giao dịch T_1 vừa được ghi ra thiết bị lưu trữ bền. Khi hệ thống hoạt động trở lại, hai thủ tục **redo(T_0)** và **undo(T_1)** sẽ được thực hiện. Do có sự xuất hiện của các mẫu tin **<T_0 start>**, **<T_0 commit>**, **<T_1 start>** trong sổ lộ trình. Sau khi hai thủ tục này được thực thi, giá trị của A, B, C sẽ là $950,ì $2050 và $700.

ĐIỂM KIỂM SOÁT (Checkpoint):

Khi lỗi hệ thống xuất hiện, hệ thống phục hồi phải tham khảo sổ ghi lộ trình để quyết định những giao dịch nào cần được làm lại và những giao dịch nào cần được huỷ bỏ. Theo nguyên lý thì cần phải tìm kiếm toàn bộ nội dung của sổ ghi để có được quyết định trên.

Hướng tiếp cận trên sẽ gặp phải hai khó khăn lớn:

1. Quá trình tìm kiếm mất nhiều thời gian.
2. Theo các giải thuật vừa nêu, hầu hết các giao dịch cần được làm lại đã ghi những dữ liệu được cập nhật ra cơ sở dữ liệu rồi. Việc làm lại chúng tuy không có hại gì, nhưng lại làm cho tiến trình khôi phục trở nên lâu hơn.

Công cụ "điểm kiểm soát" (checkpoint) được sử dụng để cải thiện hiệu năng của quá trình khôi phục. Trong quá trình hoạt động của mình, hệ thống sẽ duy trì một sổ ghi lộ trình bằng cách sử dụng một trong hai kỹ thuật được giới thiệu trong phần V.4.1 và V.4.2. Ngoài ra, hệ thống còn phải thực hiện một cách chu kỳ các hành động đặt điểm kiểm soát. Hành động này đòi hỏi một dãy các thao tác sau:

1. Xuất ra lưu trữ bền tất cả các mẫu tin ghi nhận lộ trình (log record) đang nằm trong bộ nhớ chính.
2. Xuất ra đĩa tất cả những khối đệm đã được cập nhật.
3. Xuất ra thiết bị lưu trữ bền một log-record <**checkpoint**>

Các giao dịch sẽ không được phép thực hiện bất kỳ thao tác cập nhật dữ liệu nào (ví dụ như ghi các khối đệm, ghi các mẫu tin log) khi hành động đặt điểm kiểm soát đang được thực hiện.

Sự hiện diện của điểm kiểm soát trong sổ ghi cho phép hệ thống tổ chức quá trình phục hồi tốt hơn. Xét một giao dịch T_i đã bàn giao (commit) trước một điểm kiểm soát. Ta có mẫu tin <**T_i commit**> xuất hiện trước mẫu tin **<checkpoint>.** Có nghĩa là tất cả các thay đổi mà T_i đã làm đối với cơ sở dữ liệu phải được thực hiện trước khi người ta đặt điểm kiểm soát trên. Vì vậy, trong giai đoạn phục hồi sau lỗi, người ta không cần phải làm lại (**redo)** giao dịch T_i.

Dựa trên điểm cải tiến này, ta cải tiến lại các kỹ thuật đã được trình bày trong phần V.4.1 và V.4.2 như sau:

1. Sau khi lỗi hệ thống xuất hiện, hệ thống phục hồi sẽ kiểm tra lại sổ lộ trình (log) để tìm ra giao dịch T_i thoả điều kiện: đó là giao dịch gần đây nhất được khởi động trước điểm kiểm soát gần đây nhất. Qui trình tìm T_i như sau: dò ngược trong sổ ghi lộ trình cho đến khi tìm thấy mẫu tin <**checkpoint**> đầu tiên. Từ điểm kiểm soát này, lại tiếp tục dò ngược trong sổ ghi cho đến khi tìm thấy mẫu tin <**T_i start**> đầu tiên. Mẫu tin này chỉ ra giao dịch T_i .
2. Khi đã xác định được giao dịch T_i rồi, các thủ tục **undo** và **redo** chỉ được áp dụng cho giao dịch T_i và các giao dịch diễn ra sau T_i. Chúng ta ký hiệu tập những giao dịch vừa nói là **T**.
3. Với kỹ thuật "Cập nhật tức thời cơ sở dữ liệu", tiến trình phục hồi như sau:

- Với mọi giao dịch $T_k \in T$ mà không có mẫu tin <**T_k commit**> trong sổ ghi lộ trình, thực thi **undo(T_k)**.
- Với mọi giao dịch $T_k \in T$ mà có mẫu tin <**T_k commit**> trong sổ ghi lộ trình, thực thi **redo(T_k)**.

4. Không cần thực thi thao tác **undo** khi sử dụng kỹ thuật "Cập nhật có trì hoãn cơ sở dữ liệu".

PHÂN TRANG BÓNG (Shadow Paging):

Kỹ thuật "Phân trang bóng" cũng là kỹ thuật cho phép phục hồi sau lỗi, nhưng ý tưởng thực hiện khác với các kỹ thuật dựa trên sổ ghi lộ trình vừa trình bày ở phần trên.
Sau đây là một số khái niệm cần được giải trình:

- ***Trang (page) là gì***? Như đã trình bày ở các phần trước, cơ sở dữ liệu được lưu vào thiết bị lưu trữ không phai thành nhiều khối có kích thước cố định. Người ta gọi những khối này là trang (page).
- ***Bảng trang và ý nghĩa của nó***: Khái niệm trang đã nói được mượn từ lý thuyết về Hệ điều hành. Cách quản lý trang cũng được thừa kế từ đó. Giả sử rằng cơ sở dữ liệu được phân thành n trang và sự phân bố trên đĩa của chúng có thể không theo một thứ tự cụ thể nào cả. Tuy nhiên, phải có cách để tìm ra nhanh và đúng trang thứ i của cơ sở dữ liệu ($1 \le i \le n$). Người ta dùng bảng trang (được mô phỏng như trong hình 5.2) cho mục đích này. Bảng trang có n đầu vào (entry). Mỗi đầu vào ứng với một trang. Một đầu vào chứa một con trỏ, trỏ đến một trang trên đĩa. Đầu vào đầu tiên chỉ đến trang đầu tiên của cơ sở dữ liệu, đầu vào thứ hai chỉ đến trang thứ hai ...

Ý tưởng then chốt của kỹ thuật "Phân trang bóng" là người ta sẽ duy trì hai bảng trang trong suốt chu kỳ sống của giao dịch, một bảng trang gọi là "*bảng trang hiện hành"* (current page table), bảng trang còn lại gọi là "*bảng trang bóng"* (shadow page table). Khi giao dịch khởi động, hai bảng trang này giống nhau. Bảng trang bóng sẽ không thay đổi suốt quá trình hoạt động của giao dịch. Bảng trang hiện hành sẽ bị thay đổi mỗi khi giao dịch thực hiện tác vụ **write.** Tất cả các tác vụ **input** và **output** đều sử dụng bảng trang hiện hành để định vị các trang trong đĩa. Điểm quan trọng khác là nên lưu bảng trang bóng vào thiết bị lưu trữ bền.

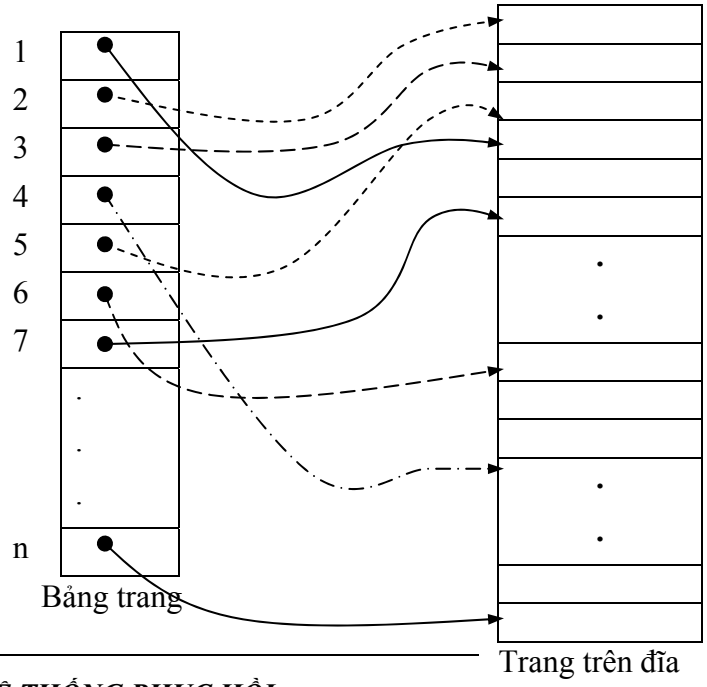


figure VI- 6

Giả sử giao dịch thực hiện tác vụ **write(X)** và hạng mục dữ liệu X được chứa trong trang thứ ***i.*** Tác vụ **write** được thực thi như sau:

1. Nếu trang thứ ***i*** chưa có trong bộ nhớ chính, thực hiện **input(X).**
2. Nếu đây là lệnh ghi được thực hiện ***lần đầu tiên*** trên trang thứ ***i*** bởi giao dịch, sửa đổi bảng trang hiện hành như sau:
   a. Tìm một trang chưa được dùng trên đĩa.
   b. Xoá trang vừa được tìm xong ở bước 2.a khỏi danh sách các khung trang tự do.
   c. Sửa lại bảng trang hiện hành sao cho đầu vào thứ ***i*** trỏ đến trang mới vừa tìm được trong bước 2.a.
3. Gán giá trị x_i cho X trong trang đệm (buffer page).

Để bàn giao một giao dịch, cần làm các bước sau:

1. Đảm bảo rằng tất cả các trang đệm trong bộ nhớ chính đã được giao dịch sửa đổi phải được xuất ra đĩa.
2. Xuất bảng trang hiện hành ra đĩa. chú ý là không được viết đè lên trang bóng
3. Xuất địa chỉ đĩa của bảng trang hiện hành ra vị trí cố định trong thiết bị lưu trữ bền. Vị trí này chính là nơi chứa địa chỉ của bảng trang bóng. Hành động này sẽ ghi đè lên địa chỉ của bảng trang bóng cũ. Như vậy, bảng trang hiện hành sẽ trở thành bảng trang bóng và giao dịch được bàn giao.

Nếu sự cố xảy ra trước khi hoàn thành bước thứ 3, hệ thống sẽ trở về trạng thái trước khi giao dịch được thực hiện. Nếu sự cố xảy ra sau khi bước thứ 3 hoàn thành, hiệu quả của giao dịch được bảo tồn; không cần thực hiện thao tác **redo** nào cả. Ví dụ trong hình 5.3 dưới đây mô phỏng lại trạng thái của các bảng trang hiện hành và bảng trang bóng khi giao dịch thực hiện thao tác ghi lên trang thứ tư của cơ sở dữ liệu có 10 trang.

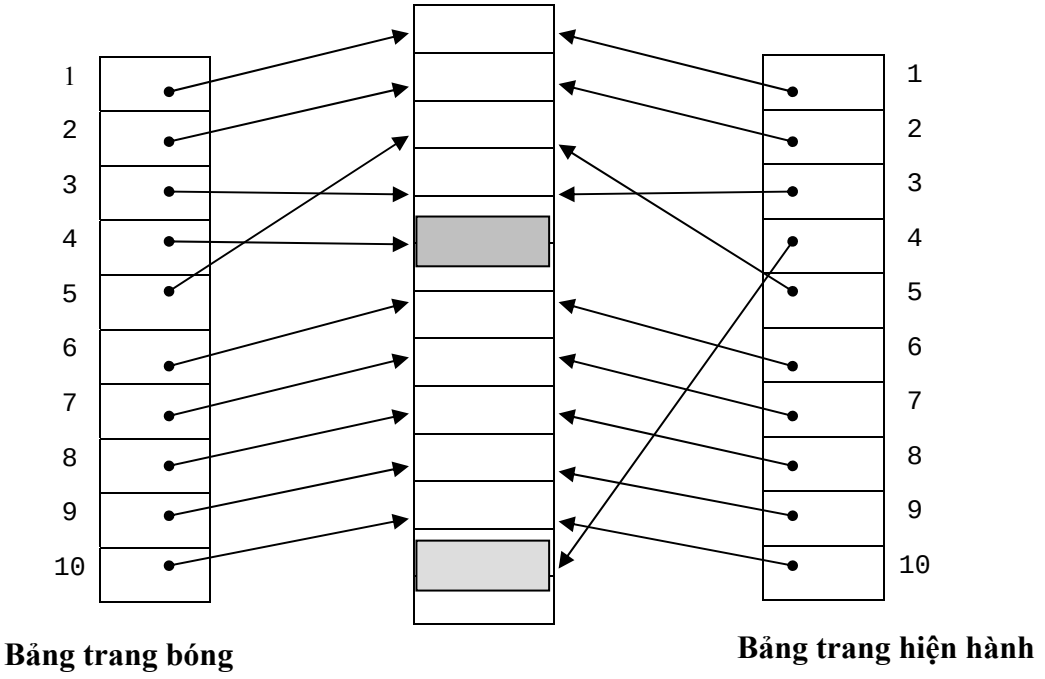


figure VI- 7 Ví dụ về bảng trang bóng và bảng trang hiện hành

Kỹ thuật phân trang bóng có một số điểm lợi hơn so với các kỹ thuật dựa trên sổ ghi:

1. Không mất thời gian ghi ra các log record.
2. Khôi phục sau sự cố nhanh hơn, do không cần các thao tác **undo** hoặc **redo**.

Tuy nhiên kỹ thuật phân trang bóng lại có nhiều nhược điểm:

- **Tổng phí bàn giao.** Xuất nhiều khối ra đĩa: các khối dữ liệu hiện tại, bảng trang hiện hành, địa chỉ của bảng trang hiện hành. Trong kỹ thuật dựa vào sổ ghi, chỉ cần xuất ra các log record, mà thông thường, các log record này vừa đủ chứa trong một khối.
- **Sự phân mảnh dữ liệu**. Trong chương II có trình bày chiến lược gom cụm vật lý các trang dữ liệu có liên quan với nhau. Sự gom cụm này cho phép việc vận chuyển dữ liệu nhanh hơn. Kỹ thuật phân trang bóng lại đổi vị trí của trang khi trang này bị sửa đổi. Điều này dẫn đến tính gom cụm dữ liệu không còn, hoặc phải dùng các giải pháp gom cụm lại rất mất thời gian.
- **Phải thu nhặt rác**. Mỗi khi giao dịch bàn giao, các trang chứa giá trị dữ liệu cũ đã bị sửa đổi bởi giao dịch sẽ trở thành không truy xuất được. Vì chúng không thuộc danh sách các trang tự do nhưng cũng không chứa dữ liệu hữu dụng. Ta gọi chúng là "rác". Cần thiết phải định kỳ tìm kiếm và thêm các trang rác vào trong danh sách các trang tự do. Hành động này được gọi là "thu nhặt rác".
- Ngoài ra, kỹ thuật phân trang bóng sẽ gặp nhiều khó khăn hơn kỹ thuật dựa vào sổ ghi khi cần được tinh chỉnh để đáp ứng cho yêu cầu phục vụ song song cho nhiều giao dịch. Vì những lý do trên, kỹ thuật phân trang bóng không được sử dụng rộng rãi lắm.

PHỤC HỒI VỚI CÁC GIAO DỊCH CẠNH TRANH

Cho dến bây giờ, ta chỉ xét các kỹ thuật phục hồi áp dụng cho các giao dịch được thực thi tuần tự. Bây giờ chúng ta sẽ tìm cách cải tiến kỹ thuật dựa vào sổ ghi nhằm đáp ứng yêu cầu phục vụ đồng thời cho nhiều giao dịch cạnh tranh. Ý tưởng thực hiện là: Không quan tâm đến số lượng các giao dịch cạnh tranh, hệ thống vẫn sử dụng một vùng đệm đĩa và một sổ ghi lộ trình. Các khối đệm được chia sẻ bởi tất cả các giao dịch. Chúng ta sẽ cho phép việc cập nhật tức thời cơ sở dữ liệu và cho phép một khối đệm có nhiều hạng mục dữ liệu được cập nhật bởi một hoặc nhiều giao dịch.

TRAO ĐỔI VỚI ĐIỀU KHIỂN CẠNH TRANH

Cơ chế phục hồi phụ thuộc rất nhiều vào cơ chế điều khiển cạnh tranh được sử dụng. Để cuộn lại một giao dịch thất bại (failed transaction), người ta phải huỷ bỏ (undo) các cập nhật được thực hiện bởi giao dịch. Giả sử giao dịch T_0 phải bị cuộn lại và một hạng mục dữ liệu Q đã bị T_0 thay đổi giá trị và cần phải được đặt lại giá trị cũ. Bằng cách sử dụng kỹ thuật dựa vào sổ ghi lộ trình, ta trả lại giá trị cũ cho Q bằng cách sử dụng một log record. Giả thiết lại có giao dịch thứ hai T_1 cũng vừa cập nhật Q xong, trước khi T_0 bị cuộn lại. Như vậy, sự cập nhật được thực hiện bởi T_1 sẽ bị mất đi nếu T_0 bị cuộn lại.

Biện pháp khắc phục là: nếu một giao dịch T đã cập nhật một hạng mục dữ liệu Q, thì không một giao dịch nào khác có quyền cập nhật lên hạng mục dữ liệu đó trong khi T chưa bàn giao hoặc chưa bị cuộn lại. Chúng ta có thể đảm bảo yêu cầu trên được thoả bằng cách sử dụng kỹ thuật "chốt hai kỳ nghiêm ngặt" (strict two-phase locking).

CUỘN LẠI GIAO DỊCH:

Phương pháp để cuộn lại (rollback) một giao dịch T_i, sử dụng sổ ghi, trong môi trường cạnh tranh như sau:

1. Dò ngược sổ ghi lộ trình để tìm ra các log record có dạng $<T_i, Xj, V_1, V_2>$.

2. Hạng mục dữ liệu Xj sẽ được trả lại giá trị cũ V_1.
3. Việc dò tìm kết thúc khi tìm thấy mẫu tin <T_i start>.

Việc dò ngược sổ ghi lộ ¹eó một ý nghĩa rất quan trọng, do một giao dịch có thể đã cập nhật một hạng mục dữ liệu nhiều hơn một lần. Một ví dụ: Xét một cặp log records như sau:

<T_i, A, 10, 20>
<T_i, A, 20, 30>

Cặp mẫu tin này thể hiện hai hành động cập nhật hạng mục dữ liệu A của giao dịch T_i. Nếu dò ngược sổ ghi lộ trình, A sẽ được trả về giá trị đúng là 10. Ngược lại, A sẽ nhận giá trị sai là 20.

Nếu kỹ thuật strict two-phase locking được sử dụng để điều khiển cạnh tranh, thì việc trả về giá trị cũ cho một hạng mục dữ liệu sẽ không xoá đi những tác động của các giao dịch khác lên hạng mục dữ liệu này.

CÁC ĐIỂM KIỂM SOÁT

Ở phần V.4.3, người ta đã sử dụng điểm kiểm soát (checkpoint) để làm giảm số lượng các log record mà hệ thống phục hồi phải dò tìm trong sổ ghi trong giai đoạn phục hồi sau lỗi. Nhưng, do đã giả thiết là không có cạnh tranh nên giải pháp V.4.3 chỉ xét đến những giao dịch sau trong quá trình khôi phục lỗi:

- Những giao dịch được khởi động sau điểm kiểm soát gần đây nhất.
- Một giao dịch (nếu có) đang trong trạng thái hoạt động (active) tại thời điểm người ta đặt điểm kiểm soát gần đây nhất.

Tình huống càng phức tạp khi các giao dịch được thực thi cạnh tranh. Có nghĩa là tại thời điểm đặt điểm kiểm soát, có thể có nhiều giao dịch đang ở trong trạng thái hoạt động.

Trong một hệ thống xử lý các giao dịch cạnh tranh, ta yêu cầu rằng: một mẫu tin ghi dấu kiểm soát (checkpoint log record) phải có dạng như sau:

<**checkpoint L**>

Trong đó L là danh sách các giao dịch đang hoạt động tại thời điểm đặt điểm kiểm soát.

Một lần nữa, ta qui ước rằng: khi hành động đặt điểm kiểm soát đang diễn ra, các giao dịch không được phép thực hiện bất kỳ thao tác cập nhật dữ liệu nào cả trên các khối đệm lẫn trên sổ ghi lộ trình.

Tuy nhiên, qui ước trên lại gây phiền toái, bởi vì các giao dịch phải ngừng hoạt động khi đặt điểm kiểm soát. Một kỹ thuật nâng cao giải quyết điểm phiền toái này là "Điểm kiểm soát mờ" (fuzzy checkpoint).

PHỤC HỒI KHỞI ĐỘNG LẠI (Restart Recovery)

Khi hệ thống phục hồi sau lỗi, nó tạo ra hai danh sách: *undo-list* bao gồm các giao dịch cần phải huỷ bỏ và *redo-list* bao gồm danh sách các giao dịch cần được làm lại.

Qui trình tạo lập hai danh sách *redo-list, undo-list* được thực hiện như sau:

1. Đầu tiên, chúng sẽ rỗng.
2. Dò ngược sổ ghi lộ trình, kiểm tra các mẫu tin cho đến khi tìm được mẫu tin <**checkpoint**> đầu tiên:
   - **a.** Với mỗi mẫu tin được tìm thấy theo dạng <**T_i commit**>, ta thêm T_i vào trong *redo-list.*
   - **b.** Với mỗi mẫu tin được tìm thấy theo dạng <**T_i start**>, nếu T_i không thuộc *redo-list* thì thêm T_i vào trong *undo-list.*
   - **c.** Khi tất cả các log record đã được xem xét, ta kiểm tra danh sách **L** trong mẫu tin <**checkpoint L**>. Với mọi giao dịch T_i trong **L**, nếu T_i không thuộc *redo-list* thì thêm T_i vào *undo-list*.

Khi hai danh sách *redo-list, undo-list* được thiết lập xong, tiến trình phục hồi được tiến hành như sau:

1. Dò ngược lại sổ ghi và thực hiện thủ tục **undo** đối với mỗi log record thuộc về giao dịch T_i trong *undo-list.* Các log record của các giao dịch nằm trong danh sách *redo-list* sẽ bị bỏ qua trong giai đoạn này. Việc dò ngược sẽ ngưng khi mẫu tin <**T_i start**> được tìm thấy cho mọi giao dịch T_i thuộc danh sách *undo-list.*
2. Định vị mẫu tin <**checkpoint L**> gần đây nhất trong log.
3. Dò sổ ghi theo chiều xuôi bắt đầu từ mẫu tin <**checkpoint L**> gần đây nhất và thực hiện thủ tục **redo** đối với mỗi log record thuộc về giao dịch T_i nằm trong danh sách *redo-list*. Trong giai đoạn này, bỏ qua các log record của các giao dịch thuộc danh sách *undo-list*.

Việc xử lý ngược ở bước 1 là rất quan trọng, nhằm đảm bảo kết quả trả về của cơ sở dữ liệu là đúng.

Sau khi tất cả các giao dịch trong danh sách *undo-list* bị huỷ bỏ, tất cả các giao dịch trong danh sách *redo-list* sẽ được làm lại. Sau khi tiến trình phục hồi thành công, xử lý giao dịch được tiếp tục.

Việc thực hiện huỷ bỏ các giao dịch trong *undo-list* trước khi làm lại các giao dịch trong *redo-list* có ý nghĩa rất quan trọng. Nếu làm ngược lại, vấn đề sau sẽ phát sinh: Giả sử hạng mục dữ liệu A có giá trị khởi đầu là 10. Giao dịch T_i đổi A thành 20 sau đó T_i bị huỷ bỏ. Sau đó, một giao dịch khác T_j cập nhật A thành 30. Đến đây thì hệ thống bị lỗi ngừng hoạt động. Hiện trạng của sổ ghi tại thời điểm hệ thống bị lỗi như sau:

<T_i, A, 10, 20>
<T_j, A, 10, 30>
<T_j commit>

Nếu thực hiện redo trước, A sẽ được đặt giá trị 30. Sau đó thực hiện undo, A sẽ được đặt giá trị 10, mà giá trị này sai. Giá trị cuối cùng của A phải là 30.

ĐIỂM KIỂM SOÁT MỜ (fuzzy checkpoint):

Kỹ thuật fuzzy checkpoint cho phép các giao dịch được cập nhật dữ liệu trên các khối đệm khi checkpoint-record đã được viết xong nhưng trước thời điểm các khối đệm đã sửa đổi được ghi ra đĩa.

Ý tưởng thực hiện fuzzy checkpoint như sau: Thay vì phải dò ngược sổ ghi để tìm mẫu tin checkpoint, ta sẽ lưu vị trí của mẫu tin checkpoint cuối cùng trong sổ ghi vào một chỗ cố định trong đĩa gọi là ***last_checkpoint***. Tuy nhiên, thông tin này sẽ không được cập nhật khi một mẫu tin checkpoint được ghi ra đĩa. Thay vào đó, trước khi một mẫu tin checkpoint được ghi ra sổ ghi, ta tạo ra một danh sách các khối đệm bị sửa đổi. Thông tin ***last_checkpoint*** được cập nhật chỉ sau khi tất cả các khối đệm bị sửa đổi được ghi ra đĩa.

last_checkpoint chỉ được dùng cho mục đích ***undo***.

BÀI TẬP CHƯƠNG VI

VI.1. Trình bày các điểm khác nhau giữa 3 kiểu lưu trữ: lưu trữ không ổn định, lưu trữ ổn định và lưu trữ bền theo tiêu chuẩn đánh giá là *chi phí cài đặt*.

VI.2. Thực tế, lưu trữ bền không thể thực hiện được. Giải thích tại sao? Hệ cơ sở dữ liệu giải quyết vấn đề này như thế nào?

VI.3. So sánh các kỹ thuật cập nhật tức thời và cập nhật có trì hoãn trong sơ đồ phục hồi dựa vào sổ ghi lộ trình theo các tiêu chuẩn: *tính dễ cài đặt và tổng chi phí thực hiện*.

VI.4. Giả sử rằng kỹ thuật cập nhật tức thời được sử dụng trong hệ thống. Bằng ví dụ, hãy chỉ ra rằng: tình trạng không nhất quán dữ liệu sẽ xảy ra nếu các log record không được ghi ra thiết bị lưu trữ bền trước khi giao dịch bàn giao (commit).

VI.5. Giải thích mục đích của cơ chế điểm kiểm soát (checkpoint). Hành động đặt điểm kiểm soát nên được thực hiện theo chu kỳ bao lâu là hợp lý?

VI.6. Khi hệ thống phục hồi sau lỗi, nó xây dựng 2 danh sách: *undo-list* và *redo-list*. Giải thích tại sao các log record của các giao dịch trong danh sách *undo-list* phải được xử lý theo thứ tự ngược, trong khi những log record trong danh sách *redo-list* lại được xử lý theo chiều xuôi?

VI.7. So sánh sơ đồ phục hồi phân trang bóng và sơ đồ phục hồi bằng sử dụng sổ ghi lộ trình theo tiêu chuẩn: *tính dễ cài đặt và tổng chi phí thực hiện*.

VI.8. Giả sử một cơ sở dữ liệu có 10 khối đĩa liên tiếp (khối 1, 2, 3, ..., 10). Hãy thể hiện trạng thái của buffer và thứ tự vật lý có thể có của các khối sau các thao tác cập nhật sau, giả sử: kỹ thuật phân trang bóng được sử dụng, buffer trong bộ nhớ chỉ đủ chứa 3 khối, chiến lược quản lý buffer là LRU (Least Recently Used)

Đọc khối 3
Đọc khối 7
Đọc khối 5

Đọc khối 3

Đọc khối 1

Sửa đổi khối 1

Đọc khối 10

Sửa khối 5

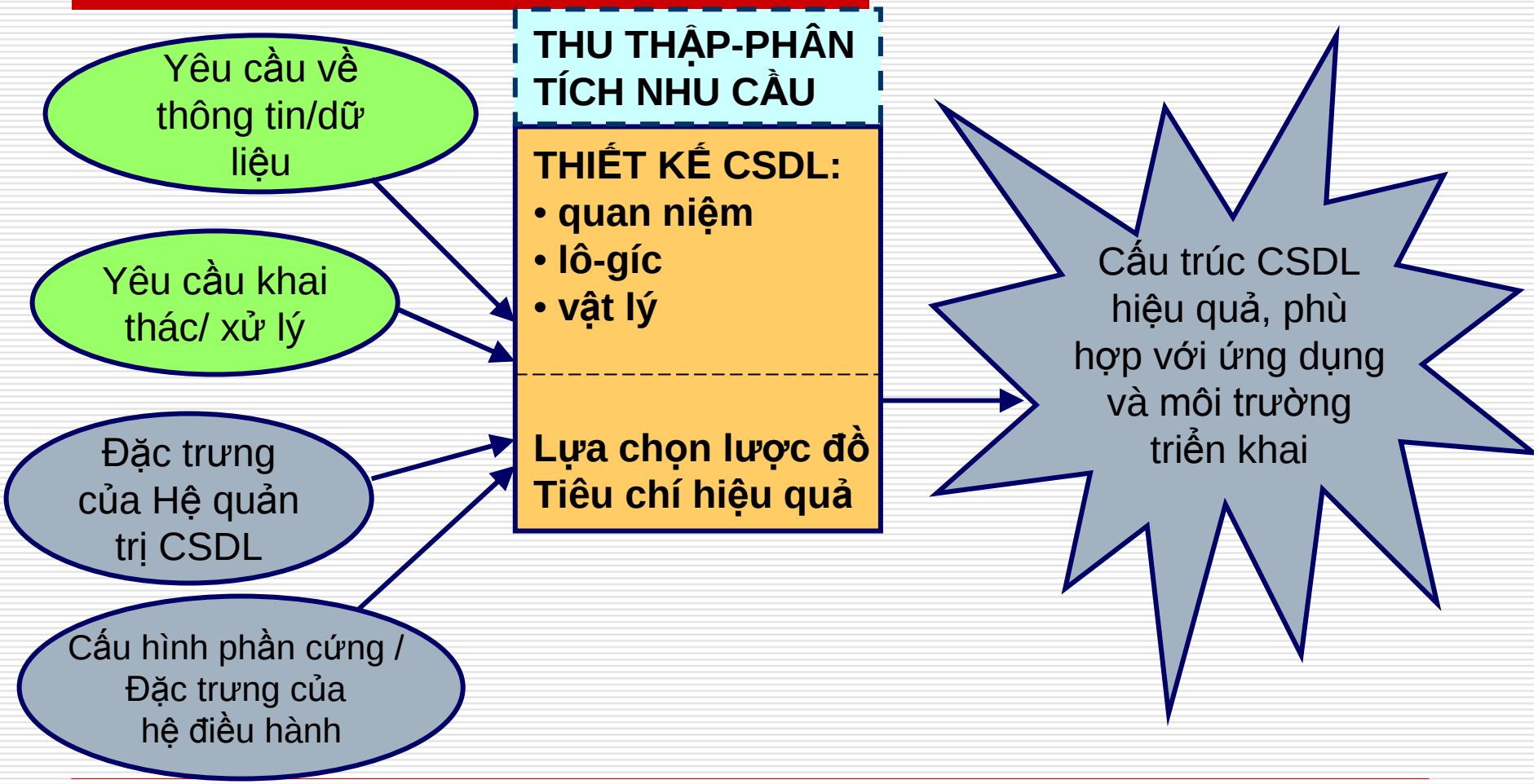
Buổi # 2
(lớp TH08/1)

Cơ sở dữ liệu nâng cao

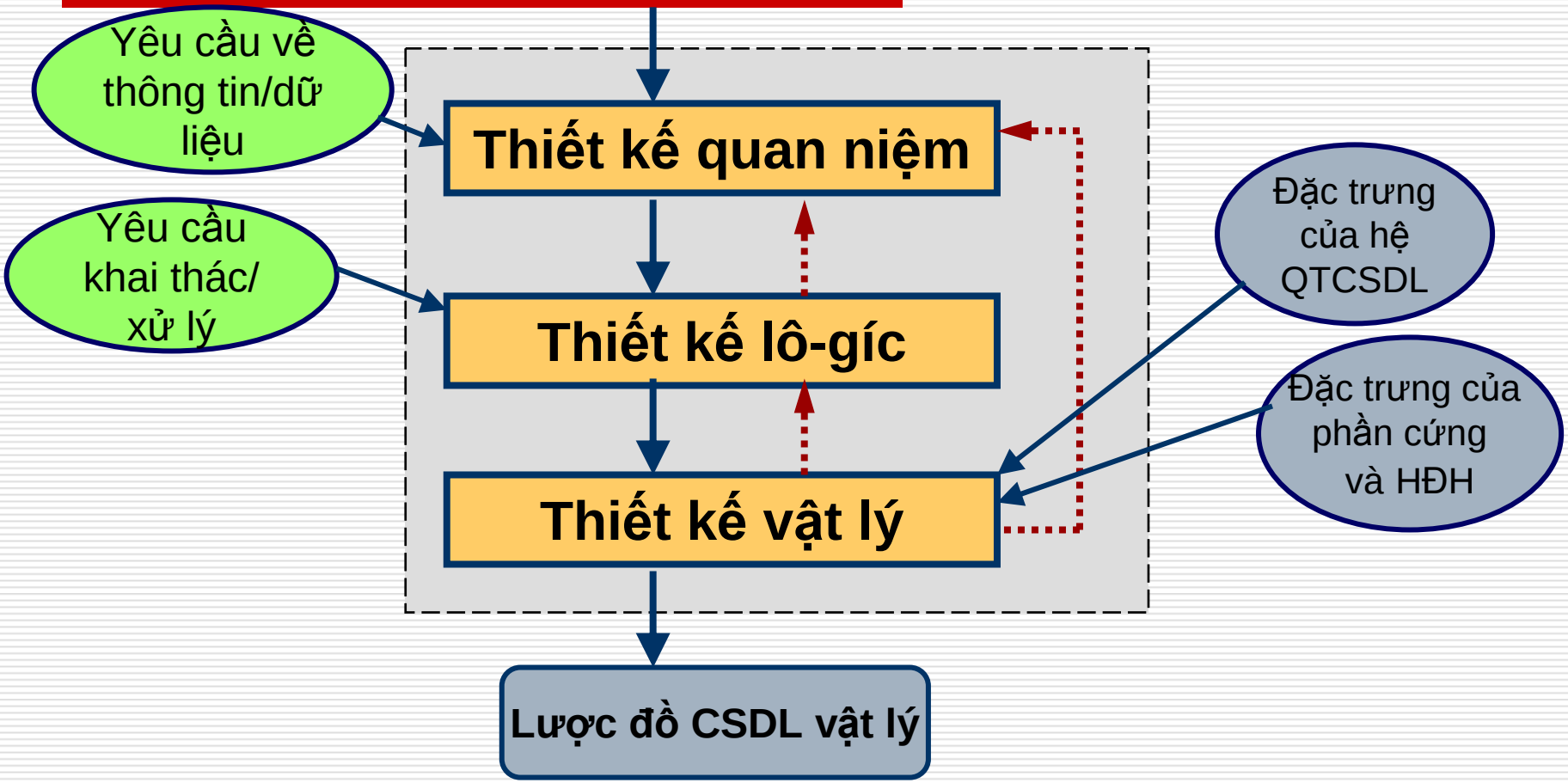
Đồng Thị Bích thủy

14/9/2010

Nhắc lại ý chính tuần trước



Nhắc lại ý chính tuần trước (2)



CHƯƠNG II:
Giai đoạn thiết kế quan niệm

Nội dung tuần này

- II.1 Dẫn nhập
  - Quy trình thiết kế quan niệm tổng thể
  - Giả thuyết *Quan hệ phổ quát*
  - Pth trong vai trò RBTV - Cơ chế kiểm tra pth
- II.2 Các mục tiêu của thiết kế quan niệm

II.1 Dẫn nhập

- ***Đầu vào***: kết quả của giai đoạn phân tích nhu cầu, có thể là:
  - Sơ phác cấu trúc CSDL gồm các quan hệ con $Q'_1, Q'_2, \ldots Q'_m$ và các phụ thuộc dữ liệu định nghĩa trên mỗi quan hệ con: $D'_1, D'_2, \ldots D'_m$:

    $\underline{C}_0' = \{<Q'_1, D'_1>, <Q'_2, D'_2>, \ldots <Q'_m, D'_m>\}$ (a)
  - Hoặc một sơ phác tổng thể:

    $\underline{C}_0 = <Q_0, D_0>$ (b)

II.1 Dẫn nhập (2)

- **Thiết kế CSDL mức quan niệm:**

  Biến đổi đầu vào thành một cấu trúc CSDL quan niệm chất lượng hơn, dựa trên một số tiêu chuẩn thiết kế

$\underline{\mathscr{C}}_0' = \{<Q'_i, \mathscr{D}'_i>\}$ (a)

$\underline{\mathscr{C}}_0 = <Q_0, \mathscr{D}_0>$ (b)

Thiết kế CSDL mức quan niệm

⇦ **tiêu chuẩn thiết kế**

$\underline{\mathscr{C}} = <Q_j, \mathscr{D}_j>$

II.1 Dẫn nhập (3)
Giả thiết quan hệ phổ quát

- **Là nền tảng của mô hình quan hệ và lý thuyết phụ thuộc dữ liệu**
- **Khái niệm quan hệ phổ quát:**
  - **Mỗi thế giới thực của ứng dụng → một quan hệ duy nhất (= quan hệ phổ quát, ký hiệu U)**
  - **Quan hệ phổ quát tập hợp tất cả các thuộc tính của môi trường ứng dụng**

II.1 ...*Giả thuyết quan hệ phổ quát (2)*

- **Nếu cấu trúc sơ phác CSDL là $\underline{C}_0$' (*trường hợp a*), ngầm hiểu quan hệ phổ quát U như sau:**
  - **U chứa tất cả các thuộc tính của các quan hệ con Q'_i.**
  - **Mỗi quan hệ con Q'_i là kết quả của phép chiếu của U lên Q'^+_i:**

    $\forall i, Q'_i = U[Q'^+_i]$ (Q'^+_i : tập thuộc tính của Q'_i)

II.1 ... *Giả thuyết quan hệ phổ quát (3)*

- **Nếu cấu trúc sơ phác là $\underline{C_0}$ (*trường hợp b*), khái niệm quan hệ phổ quát đã ngầm được sử dụng.**
- ***<u>Hệ quả</u>*:**
  - **Mỗi một thuộc tính có một ý nghĩa duy nhất. Dù thuộc tính này hiện diện trong quan hệ con hay quan hệ phổ quát, ý nghĩa của nó vẫn như nhau.**
  - **Giữa hai thuộc tính chỉ có một mối kết hợp với một ý nghĩa duy nhất.**

II.1 ... *Giả thuyết quan hệ phổ quát (4)*

- **Ví dụ 2.1: Cho môi trường ứng dụng với 3 quan hệ con như sau:**

  <ĐẶT_HÀNG (Mã_ĐH, Ngày); $F_{ĐH}$ = {Mã_ĐH → Ngày}>

  <TK_HG_GIAO(Mã_HG, Ngày, Lượng);

  F_{TKHG} = {Mã_HG, Ngày → Lượng }>

  <CHI_TIẾT_ĐẶT_HÀNG (Mã_ĐH, Mã_HG, Lượng);

  $F_{CTĐH}$ = {Mã_ĐH, Mã_HG → Lượng }>

  Các thuộc tính Ngày và Lượng có ý nghĩa khác nhau trong những quan hệ khác nhau.

II.1 ... *Giả thuyết quan hệ phổ quát (5)*

- **Ví dụ 2.1 (tt):**
  - ***Nhập nhằng về ngữ nghĩa, dẫn đến kết quả sai khi áp dụng luật dẫn cho pth trên toàn cục:***

Mã_ĐH → Ngày (*ĐH*)
MãHG, Ngày (*GH*) → Lượng (*GH*)
} **MãHG, Mã_ĐH → Lượng (*ĐH*)**

  - ***Phải đặt tên lại cho các thuộc tính:***
    - **Ngày trong ĐẶT_HÀNG thành Ngày_ĐH**
    - **Ngày trong TK_HG_GIAO thành Ngày_TK**
    - **...**

II.1 ... *Giả thuyết quan hệ phổ quát (6)*

- **Ví dụ 2.2:**

  Cho quan hệ NHÂN_VIÊN(Mã_NV, Tên_NV, Ngày)

  - **Ý nghĩa mối kết hợp giữa Mã_NV và Ngày : Ngày sinh của nhân viên**
  - **Mối kết hợp này là duy nhất : nếu cần thêm thông tin về ngày tuyển dụng, ta phải thêm một thuộc tính mới Ngày_TD, không thể thêm một ý nghĩa khác cho mối kết hợp giữa Mã_NV và Ngày.**

II.1 Dẫn nhập (4)
Cơ chế kiểm tra pth

- **Phụ thuộc hàm: một loại ràng buộc toàn vẹn quan trọng trong CSDL quan hệ**
- **Cần chọn cấu trúc CSDL sao cho việc kiểm tra pth hiệu quả nhất ⇨ tiêu chuẩn thiết kế quan trọng**
- **Xem xét:**
  - **Cơ chế kiểm tra pth nào được hệ quản trị CSDL hỗ trợ?**
  - **Thời điểm cần kiểm tra pth trong quá trình khai thác CSDL? Lúc cập nhật? Lúc truy vấn?**

II.1 ... *Cơ chế khóa – cơ chế chỉ mục trong hệ QTCSDL*

- **Định nghĩa:**

  Cho một quan hệ Q, một khóa K của Q, và I, một chỉ mục duy nhất định nghĩa trên Q

  - **Cơ chế khóa là một cơ chế có sẵn của hệ quản trị CSDL nhằm đảm bảo:**

    $$\forall q, q' \in Q \text{ sao cho } q \neq q' , q.K \neq q'.K$$

  - **Cơ chế chỉ mục với giá trị duy nhất là một cài đặt khác đảm bảo tình trạng tương tự như cơ chế khóa:**

    $$\forall q, q' \in Q \text{ sao cho } q \neq q' , q.I \neq q'.I$$

II.1 ... *Cơ chế kiểm tra phụ thuộc hàm (3)*

- Cơ chế khóa đảm bảo kiểm tra hiệu quả các phụ thuộc hàm dạng $K \rightarrow A$, với $A \in Q^+$ và K là khóa chính của Q

  (Không cần duyệt tuần tự qua tất cả các bộ trong TQ)
- Cơ chế chỉ mục duy nhất được dùng để kiểm tra hiệu quả $K' \rightarrow A$, với K' là một khóa khác của Q

II.1 ... *Hợp lệ hóa tình trạng một quan hệ*

- **Cho Q(ABC) với khóa là A, và tập phụ thuộc hàm được định nghĩa trên Q:**

$$F = \{B \rightarrow C,\ A \rightarrow BC\}$$

- **Cho một tình trạng của Q:**

TQ:

A	B	C
a_1	b_1	c_1
a_2	b_2	c_1
a_3	b_3	c_3
a_4	b_2	c_1

II.1 ... *Hợp lệ hóa tình trạng của một quan hệ (tt)*

- **Tình trạng TQ được xem là hợp lệ đối với F nếu TQ thỏa *tất cả* các pth của F.**
- **Kiểm tra B $\rightarrow$ C, đảm bảo:**

  $\forall$q, q' $\in$ Q, nếu q.B = q'.B thì q.C = q'.C

  → **Cần duyệt qua tất cả các bộ của TQ để kiểm tra**

  → **Tốn kém và mất thời gian khi TQ lớn**
- **Kiểm tra A$\rightarrow$ BC: nhờ cơ chế khóa của HQT CSDL, việc kiểm tra pth này sẽ hiệu quả hơn.**

II.1 ... *Hợp lệ hoá một biến đổi tình trạng của Q*

- **Cho một tình trạng TQ1 đã hợp lệ**
- **TQ1 được biến đổi thành TQ2 sau một vài thao tác:**
  - **Các thao tác biến TQ1 thành TQ2 có thể gây ra tình trạng không hợp lệ cho TQ2 → cần phải hợp lệ hóa TQ2 (= kiểm tra lại xem F có còn được thỏa trong TQ2)**

II.1 ... *Hợp lệ hoá một biến đổi tình trạng của Q (tt)*

- **Các thao tác có thể xảy ra trên TQ1:**
  - **Thêm một bộ: phải kiểm tra lại các pth trong F .**
  - **Xóa một bộ: không cần kiểm tra lại pth (nhưng có nguy cơ bị mất thông tin)**
  - **Sửa một bộ: phải kiểm tra lại pth và có nguy cơ bị mất thông tin (Sửa = Xoá + Thêm)**

II.2 Các mục tiêu thiết kế quan niệm

- Hai tiêu chuẩn quan trọng cần đạt được trong quá trình thiết kế CSDL ở mức quan niệm:
  - Cấu trúc CSDL kết quả (đầu ra của giai đoạn thiết kế mức quan niệm) cần đạt dạng chuẩn cao nhất
  - Cấu trúc CSDL kết quả phải tương đương với cấu trúc ban đầu

II.2 Các mục tiêu thiết kế (2)

- Tiêu chuẩn về **dạng chuẩn**:
  - Giảm tối đa sự trùng lắp thông tin → tránh được một số bất tiện khi cập nhật CSDL
  - Các phụ thuộc dữ liệu được kiểm tra dễ dàng, đơn giản và tương đối ít tốn kém nhất
- Tiêu chuẩn **tương đương**:
  - Đảm bảo các thông tin trong quan hệ phổ quát sẽ được tìm thấy đầy đủ trong CSDL của cấu trúc kết quả (gồm nhiều quan hệ con).

II.2 Các mục tiêu thiết kế (3)

- **Một CSDL thỏa mãn hai tiêu chuẩn về DC và tính tương đương, đảm bảo cho việc khai thác nó được thuận lợi trên cả ba phương diện:**
  - ***Truy vấn*: tiêu chuẩn tương đương đảm bảo các thông tin được truy xuất từ CSDL là những thông tin đã được phân tích.**
  - ***Cập nhật*: Tiêu chuẩn dạng chuẩn giảm bớt các tình huống thông tin mâu thuẫn sau khi cập nhật.**
  - ***Kiểm tra RBTV*: cả hai tiêu chuẩn đều hướng đến mục tiêu là kiểm tra RBTV dạng phụ thuộc dữ liệu được thuận lợi.**

Câu hỏi?

Buổi #3
(lớp TH2008/1)

Cơ sở dữ liệu nâng cao

Đồng Thị Bích thủy

21/9/2010

CHƯƠNG II:
Giai đoạn thiết kế quan niệm (2)

Nội dung tuần này

II.2 Các mục tiêu chính của gđ thiết kế quan niệm – *nhắc lại*

II.3 Tiêu chuẩn *Dạng chuẩn*

II.4 Tiêu chuẩn *Cấu trúc CSDL tương đương – đặt vấn đề.*

Nhắc lại:
II.1 Đầu vào – Kết quả của gđ TKQN

- **Thiết kế CSDL mức quan niệm:**

  Biến đổi đầu vào thành một cấu trúc CSDL quan niệm chất lượng hơn, dựa trên một số tiêu chuẩn thiết kế

$\underline{\mathscr{C}}_0' = \{<Q'_i, \mathscr{D}'_i>\}$

$\underline{\mathscr{C}}_0 = <Q_0, \mathscr{D}_0>$

Thiết kế CSDL mức quan niệm

$\underline{\mathscr{C}} = <Q_j, \mathscr{D}_j>$

II.2 Các mục tiêu thiết kế quan niệm

- Hai tiêu chuẩn quan trọng cần đạt được trong quá trình thiết kế CSDL ở mức quan niệm:
  - Cấu trúc CSDL kết quả (đầu ra của giai đoạn thiết kế mức quan niệm) cần đạt dạng chuẩn cao nhất
  - Cấu trúc CSDL kết quả phải tương đương với cấu trúc ban đầu

II.2 Các mục tiêu thiết kế QN (2)

- Tiêu chuẩn về dạng chuẩn:
  - ↙ tối đa sự trùng lắp thông tin ➔ tránh được một số bất tiện khi cập nhật CSDL
  - Các phụ thuộc dữ liệu được kiểm tra dễ dàng, đơn giản và tương đối ít tốn kém nhất
- Tiêu chuẩn tương đương:
  - Đảm bảo các thông tin trong quan hệ phổ quát sẽ được tìm thấy đầy đủ trong CSDL của cấu trúc kết quả (gồm nhiều quan hệ con).

II.2 Các mục tiêu thiết kế QN (3)

- **Một CSDL thỏa mãn hai tiêu chuẩn về DC và tính tương đương, đảm bảo cho việc khai thác nó được thuận lợi trên cả ba phương diện:**
  - ***Truy vấn*: tiêu chuẩn tương đương đảm bảo các thông tin được truy xuất từ CSDL là những thông tin đã được phân tích.**
  - ***Cập nhật*: Tiêu chuẩn dạng chuẩn giảm bớt các tình huống thông tin mâu thuẫn sau khi cập nhật.**
  - ***Kiểm tra RBTV*: cả hai tiêu chuẩn đều hướng đến mục tiêu là kiểm tra RBTV dạng phụ thuộc dữ liệu được thuận lợi.**

II.3 Quan hệ được chuẩn hóa

- Trước khi có mô hình quan hệ, công nghệ lưu trữ dữ liệu thường dùng cấu trúc dữ liệu phân cấp, trong đó có nhiều trường lặp.
- Ví dụ 2.3:

  DS_LỚP (Tên_L, SỉSố, (TênHS,(Điểm)*15)*30)

  →Điểm được lặp lại 15 lần: mỗi học sinh có tối đa 15 cột điểm

  →Thông tin về mỗi học sinh (tên học sinh, các cột điểm) được lặp lại 30 lần: lớp học có tối đa 30 học sinh

II.3 Quan hệ được chuẩn hóa (2)

- **Ví dụ 2.3 (tt):**
  - **Hình sau minh họa dữ liệu của quan hệ DANH_SÁCH_LỚP:**

Tên_L	SĩSố	TênHS1	Điểm11	Điểm12	…	Điểm15	TênHS2	Điểm21	…
11A1	30	Nguyễn Văn A	7	8	…	9	Trần Thị B	6	…
11A2	28	Võ Văn C	8	6	…	10	…	…	…

II.3 Quan hệ được chuẩn hóa (3)

- Cấu trúc lưu trữ chuẩn hóa của ví dụ 2.3 trở thành:
  - DS_HS (Tên_L, SỉSố, STT1, TênHS)
  - ĐiỂM_HS_LỚP (Tên_L, STT1, STT2, Điểm)

Các thuộc tính STT1, STT2 → thay bằng tên gọi có tính gợi nhớ, phù hợp với ngữ nghĩa của chúng:

- STT1: mã số học sinh trong lớp → "MãHS".
- STT2: môn học → "Môn"

II.3 Quan hệ được chuẩn hóa (4)

- **Dữ liệu minh họa cho của cấu trúc đã chuẩn hóa:**

T_ĐiỂM_HS_LỚP:

Tên_L	MãHS	Môn	Điểm
11A1	0001	Toán	7
11A1	0001	Lý	9
11A1	0002	Lý	10
11A1	0002	Toán	6
11A1	0001	Văn	7
11A2	0031	Lý	6
…	…	…	…

T_DS_HS:

Tên_L	SỉSố	MãHS	TênHS
11A1	30	0001	Nguyễn Văn A
11A1	30	0002	Trần Thị B
11A2	28	0031	Võ Văn C

II.3.1 Dạng chuẩn 1 (DC1)

- **Các quan hệ trong cấu trúc CSDL được chuẩn hóa như trên gọi là quan hệ ở *dạng chuẩn 1***
- ***Định nghĩa dạng chuẩn 1:*** **Một quan hệ ở dạng chuẩn 1 không có các trường lặp và các trường kép, còn được gọi là *cấu trúc phẳng*.**

II.3.1 Dạng chuẩn 1 (DC1) (2)

- **Ưu điểm của cấu trúc chuẩn hóa:**
  - **cho phép truy vấn chỉ dựa trên ngữ nghĩa của CSDL, không dựa vào cách thức lưu trữ thông tin**
  - **Ví dụ: Cho biết điểm môn Văn của học sinh có mã 0001 thuộc lớp 11A1:**

II.3.1 Dạng chuẩn 1 (DC1) (3)

- Với cấu trúc chuẩn hóa, câu truy vấn là:

  $\Pi_{\text{Điểm, Tên_HS}}\ \sigma$ (Tên_L = ’11A1’, MãHS = ‘0001’,
  Môn = ‘Văn’) ĐIỂM_HS_LỚP

- Với cấu trúc chưa chuẩn hóa, câu truy vấn là:

  DS_LỚP(11A1)(1,2)

  Học sinh có mã 0001 lớp 11A1 được lưu thứ 1 trong danh sách, điểm môn Văn được lưu ở cột điểm thứ 2 trong bảng điểm của HS.

II.3.1 Dạng chuẩn 1 (DC1) (4)

- **Khuyết điểm:**
  - **Với DC1, yêu cầu về sự giảm thiểu trùng lắp thông tin chưa được quan tâm**
  - **Các vấn đề gây ra do trùng lắp thông tin:**
    - **Bất tiện khi thêm/cập nhật:**

      để đảm bảo dữ liệu nhất quán, phải kiểm tra và cập nhật hàng loạt trên các thông tin trùng lắp

      VD: Thêm một học sinh (11A1, 31, 0003, Võ Thị H) vào danh sách
    - **Mất thông tin khi xóa:**

      VD: xóa học sinh duy nhất của lớp 12A5 đi, ta sẽ mất thông tin về tên lớp 12A5 trong CSDL.

II.3.1 Dạng chuẩn 1 – Những bất tiện

- **Ví dụ 2.4: Cho quan hệ sau:**

  LỊCH_COI_THI (GVCT, N,G,P,M,GV)

  F = { f1: GVCT → N,G,P : một giảng viên coi thi (GVCT) chỉ coi vào một ngày(N), một giờ (G) và trong một phòng (P) duy nhất.

  f2: M → GV : mỗi môn thi (M) có một giảng viên (GV) duy nhất phụ trách.

  f3:N,G,P → M : mỗi ngày, vào một giờ, trong một phòng, chỉ có một môn thi duy nhất }

II.3.1 Dạng chuẩn 1 – Những bất tiện (2)

- **Ví dụ 2.4 (tt):**
  - **Một thể hiện của quan hệ LỊCH_COI_THI:**

GVCT	N	G	P	M	GV
Nguyễn Văn A	2	8:00 – 10:00	101	Giải thuật	X
Nguyễn Thị B	2	10:00 – 12:00	101	CSDL	Y
Trần Văn C	2	10:00 – 12:00	101	CSDL	Y
Võ Thị D	2	8:00 – 10:00	103	Giải thuật	X

  - **Nhận xét về sự trùng lắp thông tin và những bất tiện khi thêm/ xóa/ cập nhật?**

II.3.1 Dạng chuẩn 1 – Những bất tiện (3)

- Cần **tách nhỏ** quan hệ theo các tiêu chuẩn nghiêm ngặt hơn để **giải quyết các vấn đề về trùng lắp thông tin**, để CSDL hoạt động hiệu quả hơn

II.3.2 Dạng chuẩn 2 (DC2)

- **Một số khái niệm bổ sung**
  - ***Thuộc tính khóa (không khóa):*** **A là một thuộc tính khóa (không khóa) của Q nếu:**
    - **A $\in$ Q^+**
    - **A có (không có) tham gia vào một khóa của Q**

Ví dụ 2.5: Q1(<u>MNO</u>PX) ; Q2(<u>P</u>Y)

M,N,O là các thuộc tính khóa của Q1, P là thuộc tính khóa của Q2 nhưng là thuộc tính không khóa của Q1

II.3.2 Dạng chuẩn 2 (DC2) (2)

- ***Thuộc tính phụ thuộc đầy đủ (không phụ thuộc đầy đủ) vào một tập thuộc tính:***
  - **A là một thuộc tính phụ thuộc đầy đủ (không phụ thuộc đầy đủ) vào X nếu X $\rightarrow$ A là một phụ thuộc hàm đầy đủ (không đầy đủ)**

II.3.2 Dạng chuẩn 2 – Định nghĩa

- Một quan hệ Q ở dạng chuẩn 2 nếu và chỉ nếu tất cả thuộc tính không khóa đều *phụ thuộc đầy đủ* vào khóa
- Ví dụ 2.6:

  <COI_THI (<u>GVCT</u>, N,G,P); F1 = {GVCT → N,G,P}>

  <MON_KT(<u>N,G,P</u>,M,GV); F2 = {N,G,P→M; M →GV}>

  ➔ Các quan hệ này có đạt DC2 ?
- Ví dụ 2.7: <Q(<u>BD</u>CZ) F = {B → C; BD → Z}>

  ➔ Q có đạt DC2 ?

II.3.2 DC2: Các vấn đề còn tồn tại

- Vẫn còn thông tin trùng lắp.
- Các bất tiện trong thao tác thêm/xóa/cập nhật do trùng lắp thông tin gây ra.

II.3.2 DC2: Các vấn đề còn tồn tại (2)

Các bảng sau minh họa cho dữ liệu của các quan hệ trong ví dụ 2.6 (đều đạt DC2):

GV_CT	N	G	P
Nguyễn Văn A	2	8:00 – 10:00	101
Nguyễn Thị B	2	10:00 – 12:00	101
Trần Văn C	2	10:00 – 12:00	101
Võ Thị D	2	8:00 – 10:00	103

N	G	P	M	GV
2	8:00 – 10:00	101	Giải thuật	X
2	10:00 – 12:00	101	CSDL	Y
2	8:00 – 10:00	103	Giải thuật	X

➔ Nhận xét về sự trùng lắp thông tin và các bất tiện khi thêm/ xoá/ sửa?

II.3.3 Dạng chuẩn 3 (DC3)

- **Khái niệm bổ sung: *Thuộc tính phụ thuộc bắc cầu* vào một tập thuộc tính :**

A phụ thuộc bắc cầu vào X nếu có 4 điều kiện sau:

- $X \rightarrow Y \in F^+$ **(i)**
- $Y \rightarrow A \in F^+$ **(ii)**
- $Y \rightarrow X \notin F^+$ **(iii)**
- $A \notin (X \cup Y)$ **(iv)**

II.3.3 DC3 – Khái niệm bổ sung (2)

- **Ví dụ 2.8 : Cho F = {MN→OPRX; NO →M; P →RY}**
  - **P có phụ thuộc bắc cầu vào NO?**

    NO → M ⇒ NO →MN : thỏa (i)

    MN → P:thỏa (ii)

    MN → O ⇒ MN →ON : không thỏa (iii)

    } **P không phụ thuộc bắc cầu vào NO**
  - **R có phụ thuộc bắc cầu vào NO?**

    NO → MN và MN→P ⇒ NO →P (i)

    P → R (ii)

    P → NO ∉ F^+ (iii)

    R ∉ NOP (iv)

    } **R phụ thuộc bắc cầu vào NO**

II.3.3 Dạng chuẩn 3 – Định nghĩa

- **Định nghĩa 1:**

  Q ở DC3 nếu và chỉ nếu mọi thuộc tính không khóa *phụ thuộc đầy đủ* và *không phụ thuộc bắc cầu* vào khóa

- **Định nghĩa 2:**

  Q ở DC3 nếu và chỉ nếu $\forall X \rightarrow A$ không hiển nhiên định nghĩa trên Q (A là thuộc tính đơn, X là tập thuộc tính), một trong hai điều kiện sau được thỏa:

  - **Hoặc X chứa một khóa của Q**
  - **Hoặc A là thuộc tính khóa của Q**

II.3.3 DC3 – Ví dụ

Ví dụ 2.9: Xét cấu trúc CSDL ở ví dụ 2.6 được cải tiến để các quan hệ đều đạt DC3:

COI_THI (GVCT, N, G, P)

LỊCH_KT(N,G,P,M)

GD(M,GV)

T_COI_THI

GV_CT	N	G	P
Nguyễn Văn A	2	8:00 – 10:00	101
Nguyễn Thị B	2	10:00 – 12:00	101
Trần Văn C	2	10:00 – 12:00	101
Võ Thị D	2	8:00 – 10:00	103

T_LỊCH_KT

N	G	P	M
2	8:00 – 10:00	101	Giải thuật
2	10:00 – 12:00	101	CSDL
2	8:00 – 10:00	103	Giải thuật

Không còn trùng lắp thông tin

T_GD

M	GV
Giải thuật	X
CSDL	Y

II.3.3 DC3- Vấn đề còn tồn tại

Ví dụ 2.10: môi trường ứng dụng của ví dụ 2.9 có thêm một ràng buộc mới: f4: M →P

→ Quan hệ LICH_KT có hai khóa (N,G,P) và (N,G,M), vẫn đạt chuẩn 3.

Xét thể hiện sau:

T_LỊCH_KT

	N	G	P	M	
	2	8:00 – 10: 00	101	Giải thuật	
	2	10:00 – 12: 00	101	CSDL	
Thêm →	3	8:00 – 10: 00	101	CSDL	**→ Tạo thông tin trùng lắp**
Thêm →	2	8:00 – 10: 00	105	CSDL	**→ vi phạm f_4**

II.3.4 Dạng chuẩn Boyce – Codd – Kent (DC BCK)

- **Định nghĩa:**

  Q ở DC BCK nếu và chỉ nếu $\forall$ X $\rightarrow$ A không hiển nhiên định nghĩa trên Q thì $\forall$B $\in$ Q^+, ta luôn có (X$\rightarrow$B) là một PTH thuộc F^+;

  hay nói cách khác, X chứa một khóa của Q

- **Ví dụ 2.11:**
  - **Quan hệ LỊCH_KT với hai khóa ở ví dụ 3.10 không đạt dạng chuẩn BCK vì có M $\rightarrow$ P.**

III.3.4 DC BCK - Giới hạn

- DC BCK quan tâm giải quyết vấn đề trùng lắp thông tin, nhưng xem nhẹ một tiêu chuẩn khác: sự thuận lợi khi kiểm tra phụ thuộc dữ liệu.
- Ví dụ 2.12:

  Thay đổi cấu trúc CSDL của ví dụ 2.10 để tất cả quan hệ đạt DC BCK:

  COI_THI (<u>GVCT</u>, N,G,P) ; GD (<u>M</u>,GV)

  LICH_KT_1 (<u>M</u>,P) ; LICH_KT_2 (<u>M,N,G</u>)

III.3.4 DC BCK – Giới hạn (2)

- **Minh họa thể hiện của các quan hệ:**

T_COI_THI

GV_CT	N	G	P
Nguyễn Văn A	2	8:00 – 10:00	101
Nguyễn Thị B	2	10:00 – 12:00	101
Trần Văn C	2	10:00 – 12:00	101

T_GD

M	GV
Giải thuật	X
CSDL	Y

T_LỊCH_KT_1

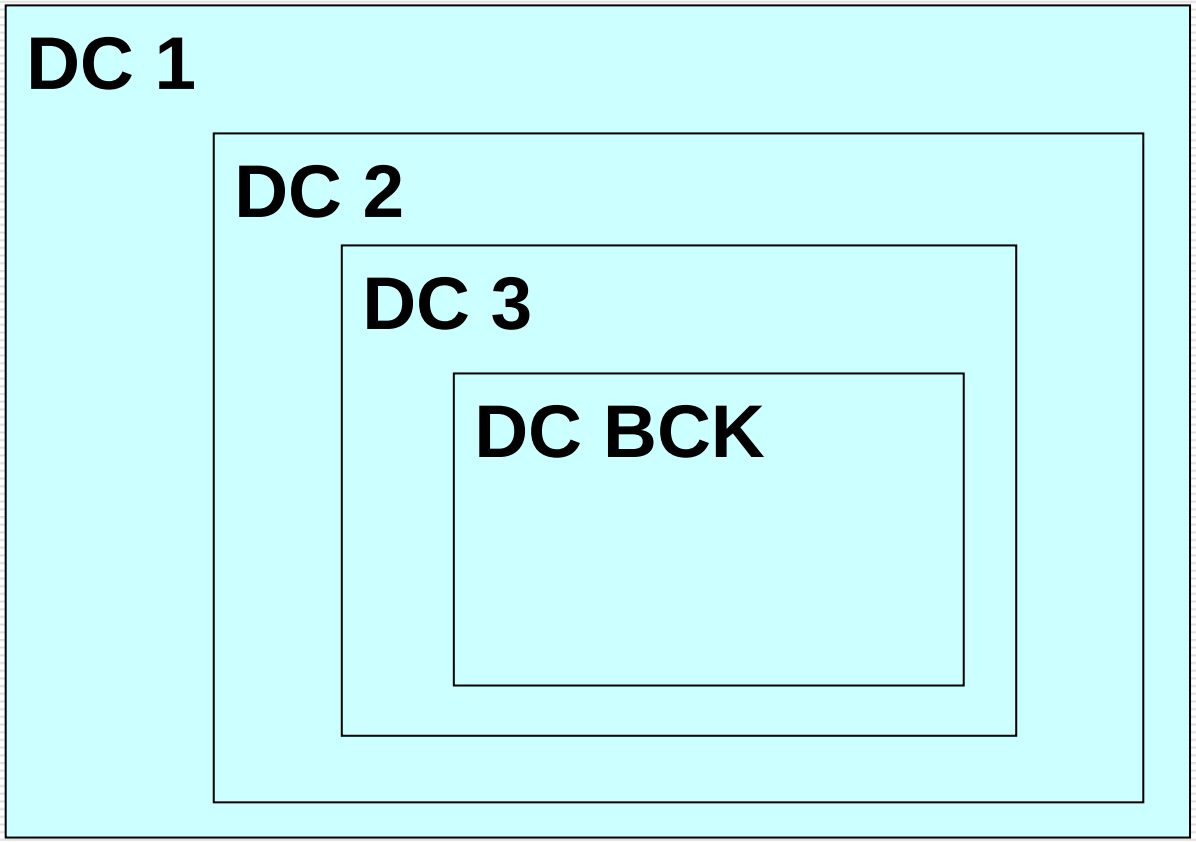
P	M
101	Giải thuật
101	CSDL

T_LỊCH_KT_2

N	G	M
2	8:00 – 10: 00	Giải thuật
2	10:00 – 12: 00	CSDL

→Không còn trùng lắp thông tin, nhưng bất tiện khi kiểm tra N,G,P → M : phải kết LICH_KT_1 và LICH_KT_2

III.3.5 Tính chất lồng nhau của các dạng chuẩn



III.3.6 Dạng chuẩn của cấu trúc CSDL

- **Định nghĩa:**
  - **Cho một cấu trúc CSDL $\underline{C} = \{<Q_i, F_i>\}$, i = 1..n**
  - **Dạng chuẩn của $\underline{C}$ là dạng chuẩn thấp nhất trong các dạng chuẩn của các Q_i**
- **Ví dụ 2.13:**
  - **Cấu trúc CSDL ở ví dụ 2.12 đạt DC BCK**
  - **Cấu trúc CSDL ở ví dụ 2.10 đạt DC3 (vì có một quan hệ con chỉ đạt DC3)**

II.4 Cấu trúc CSDL tương đương - Đặt vấn đề

- **Nguyên tắc tương đương:**
  Nếu C ’= {<Q_i, D_i>} (i = 1..n) được dùng để lưu trữ dữ liệu thay cho C = <U,D>, có nghĩa là:
  - Thông tin và dữ liệu được xác định ở gđ phân tích & thu thập nhu cầu phải được tìm thấy đầy đủ trong C’
  - Thông tin và dữ liệu được lưu trong C’ là những TT&DL lẽ ra được lưu trong C .

II.4 ... Đặt vấn đề (2)

- **Ví dụ 2.14: với cấu trúc quan hệ phổ quát của Vd 2.4:**

  < LỊCH_COI_THI (GVCT, N,G,P,M,GV) ,

  F = {f_1: GVCT→N,G,P; f_2: M → GV; f_3: N,G,P→M} >

  Cho cấu trúc CSDL sau:

  C1={<COI_THI_1 (GVCT,N,G,M),

  F_1' ={f'$_1$:GVCT→N,G,M}>;

  <GD (M,GV), F_2' = {f'$_2$: M → GV}>;

  <LICH_KT (N,G,P,M)>; F_3' ={f'$_3$: N,G,P →M}> }

II.4 ... Đặt vấn đề (3)

T_LICH_KT

N	G	P	M
2	8:00–10:00	101	Giải thuật
2	10:00–12:00	101	CSDL
2	8:00–10:00	103	Giải thuật

T_COI_THI_1

GV_CT	N	G	M
Ng.V.A	2	8:00–10:00	Giải thuật
Ng.T.B	2	10:00–12:00	CSDL
Tr.V.C	2	10:00–12:00	CSDL
Võ.T.D	2	8:00–10:00	Giải thuật

T_COI_THI_1 ⋈ T_LICH_KT ⋈ T_GD

GV_CT	N	G	P	M	GV
Ng.V.A	2	8:00–10:00	101	Giải thuật	X
Ng.V.A	2	8:00–10:00	103	Giải thuật	X
Ng.T.B	2	10:00–12:00	101	CSDL	Y
Tr.V.C	2	10:00–12:00	101	CSDL	Y
Võ.T.D	2	8:00–10:00	101	Giải thuật	X
Võ.T.D	2	8:00–10:00	103	Giải thuật	X

T_GD

M	GV
Giải thuật	X
CSDL	Y

Giống với TU?
(trang 17)

II.4 ... Đặt vấn đề (4)

- **Kết luận:**

  Cần có những tiêu chí cụ thể giúp người thiết kế xác định lược đồ CSDL QN có tương đương với lược quan hệ phổ quát không?

Câu hỏi?

Buổi #4
(lớp TH2008/1)

Cơ sở dữ liệu nâng cao

Đồng Thị Bích thủy

28/9/2010

CHƯƠNG II:
Giai đoạn thiết kế quan niệm (3)

Nội dung tuần này

II.2-3 Nhắc lại: Ý nghĩa của tiêu chuẩn
Dạng chuẩn (ưu điểm & hạn chế)

II.4 Tiêu chuẩn *Cấu trúc CSDL tương đương*

Nhắc lại:
II.1 Đầu vào – Kết quả của gđ TKQN

- **Thiết kế CSDL mức quan niệm:**

  Biến đổi đầu vào thành một cấu trúc CSDL quan niệm chất lượng hơn, dựa trên một số tiêu chuẩn thiết kế

$\underline{\mathscr{C}}_0' = \{<Q'_i, \mathscr{D}'_i>\}$

$\underline{\mathscr{C}}_0 = <Q_0, \mathscr{D}_0>$

Thiết kế CSDL mức quan niệm

$\underline{\mathscr{C}} = <Q_j, \mathscr{D}_j>$

II.2 Các mục tiêu thiết kế quan niệm

- Hai tiêu chuẩn quan trọng cần đạt được trong quá trình thiết kế CSDL ở mức quan niệm:
  - Cấu trúc CSDL kết quả (đầu ra của giai đoạn thiết kế mức quan niệm) cần đạt dạng chuẩn cao nhất
  - Cấu trúc CSDL kết quả phải tương đương với cấu trúc ban đầu

II.2 Các mục tiêu thiết kế QN (2)

- Tiêu chuẩn về dạng chuẩn:
  - ↙ tối đa sự trùng lắp thông tin ➔ tránh được một số bất tiện khi cập nhật CSDL
  - Các phụ thuộc dữ liệu được kiểm tra dễ dàng, đơn giản và tương đối ít tốn kém nhất
- Tiêu chuẩn tương đương:
  - Đảm bảo các thông tin trong quan hệ phổ quát sẽ được tìm thấy đầy đủ trong CSDL của cấu trúc kết quả (gồm nhiều quan hệ con).

II.3.3 Dạng chuẩn 3 – Định nghĩa

- **Định nghĩa 1:**

  Q ở DC3 nếu và chỉ nếu mọi thuộc tính không khóa *phụ thuộc đầy đủ* và *không phụ thuộc bắc cầu* vào khóa

- **Định nghĩa 2:**

  Q ở DC3 nếu và chỉ nếu $\forall X \rightarrow A$ không hiển nhiên định nghĩa trên Q (A là thuộc tính đơn, X là tập thuộc tính), một trong hai điều kiện sau được thỏa:

  - **Hoặc X chứa một khóa của Q**
  - **Hoặc A là thuộc tính khóa của Q**

Ví dụ 2.4:
Nhắc lại lược đồ quan hệ phổ quát

LỊCH_COI_THI (GVCT, N,G,P,M,GV)

F = { f1: GVCT → N,G,P : một giảng viên coi thi (GVCT) chỉ coi vào một ngày(N), một giờ (G) và trong một phòng (P) duy nhất.

f2: M → GV : mỗi môn thi (M) có một giảng viên (GV) duy nhất phụ trách.

f3:N,G,P → M : mỗi ngày, vào một giờ, trong một phòng, chỉ có một môn thi duy nhất }

II.3.3 DC3 – Ví dụ

Ví dụ 2.9: Các quan hệ của cấu trúc CSDL sau đều đạt DC3:

<COI_THI (GVCT, N, G, P); {f1}>

<LỊCH_KT(N,G,P,M); {f3}>

<GD(M,GV); {f2}>

T_LỊCH_KT

N	G	P	M
2	8:00 – 10:00	101	Giải thuật
2	10:00 – 12:00	101	CSDL
2	8:00 – 10:00	103	Giải thuật

T_COI_THI

GV_CT	N	G	P
Nguyễn Văn A	2	8:00 – 10:00	101
Nguyễn Thị B	2	10:00 – 12:00	101
Trần Văn C	2	10:00 – 12:00	101
Võ Thị D	2	8:00 – 10:00	103

Không còn trùng lắp thông tin

T_GD

M	GV
Giải thuật	X
CSDL	Y

Kiểm tra F hiệu quả

II.3.3 DC3- Vấn đề còn tồn tại

Ví dụ 2.10: môi trường ứng dụng của ví dụ 2.9 có thêm một ràng buộc mới: f4: M →P

→ Quan hệ LICH_KT có hai khóa (N,G,P) và (N,G,M), vẫn đạt chuẩn 3.

Xét thể hiện sau:

T_LỊCH_KT

	N	G	P	M	
	2	8:00 – 10: 00	101	Giải thuật	
	2	10:00 – 12: 00	101	CSDL	
Thêm →	3	8:00 – 10: 00	101	CSDL	**→ Tạo thông tin trùng lắp**
Thêm →	2	8:00 – 10: 00	105	CSDL	**→ vi phạm f_4**

II.3.4 Dạng chuẩn Boyce – Codd – Kent (DC BCK)

- **Định nghĩa:**

  Q ở DC BCK nếu và chỉ nếu $\forall$ X $\rightarrow$ A không hiển nhiên định nghĩa trên Q thì $\forall$B $\in$ Q^+, ta luôn có (X$\rightarrow$B) là một PTH thuộc F^+;

  hay nói cách khác, X chứa một khóa của Q

- **Ví dụ 2.11:**
  - **Quan hệ LỊCH_KT với hai khóa ở ví dụ 2.10 không đạt dạng chuẩn BCK vì có M $\rightarrow$ P.**

III.3.4 DC BCK - Giới hạn

- DC BCK quan tâm giải quyết vấn đề trùng lắp thông tin, nhưng xem nhẹ một tiêu chuẩn khác: sự thuận lợi khi kiểm tra phụ thuộc dữ liệu.
- Ví dụ 2.12:

  Thay đổi cấu trúc CSDL của ví dụ 2.10 để tất cả quan hệ đạt DC BCK:

  COI_THI (<u>GVCT</u>, N,G,P) ; GD (<u>M</u>,GV)

  LICH_KT_1 (<u>M</u>,P) ; LICH_KT_2 (<u>M,N,G</u>)

III.3.4 DC BCK – Giới hạn (2)

- **Minh họa thể hiện của các quan hệ:**

T_COI_THI

GV_CT	N	G	P
Nguyễn Văn A	2	8:00 – 10:00	101
Nguyễn Thị B	2	10:00 – 12:00	101
Trần Văn C	2	10:00 – 12:00	101

T_GD

M	GV
Giải thuật	X
CSDL	Y

T_LỊCH_KT_1

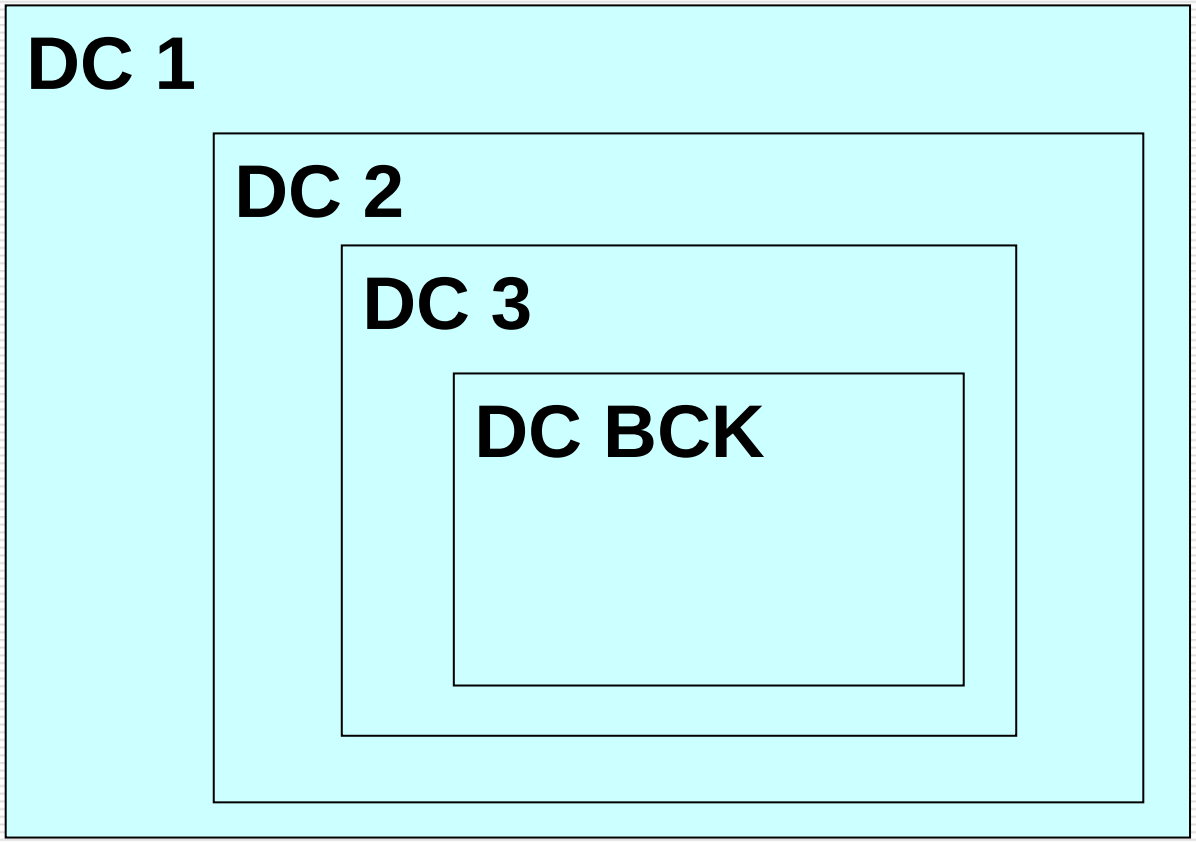
P	M
101	Giải thuật
101	CSDL

T_LỊCH_KT_2

N	G	M
2	8:00 – 10: 00	Giải thuật
2	10:00 – 12: 00	CSDL

→Không còn trùng lắp thông tin, nhưng bất tiện khi kiểm tra N,G,P → M : phải kết LICH_KT_1 và LICH_KT_2

III.3.5 Tính chất lồng nhau của các dạng chuẩn



III.3.6 Dạng chuẩn của cấu trúc CSDL

- **Định nghĩa:**
  - **Cho một cấu trúc CSDL $\underline{C} = \{<Q_i, F_i>\}$, i = 1..n**
  - **Dạng chuẩn của $\underline{C}$ là dạng chuẩn thấp nhất trong các dạng chuẩn của các Q_i**
- **Ví dụ 2.13:**
  - **Cấu trúc CSDL ở ví dụ 2.9 đạt DC BCK**
  - **Cấu trúc CSDL ở ví dụ 2.10 đạt DC3 (vì có một quan hệ con chỉ đạt DC3)**

II.4 Cấu trúc CSDL tương đương - Đặt vấn đề

- **Nguyên tắc tương đương:**
  Nếu C ’= {<Q_i, D_i>} (i = 1..n) được dùng để lưu trữ dữ liệu thay cho C = <U,D>, có nghĩa là:
  - Thông tin và dữ liệu được xác định ở gđ phân tích & thu thập nhu cầu phải được tìm thấy đầy đủ trong C’
  - Thông tin và dữ liệu được lưu trong C’ là những TT&DL lẽ ra được lưu trong C .

II.4 Cấu trúc CSDL tương đương

Khái niệm TT được lưu trữ trong CSDL:

- **TT được lưu trữ trong Q:** q, một bộ của Q, là TT được lưu trong Q. q là tường minh nếu không chứa giá trị Trống (Null).
- **TT được lưu trữ trong $\underline{\mathscr{C}}$:** TT lẽ ra được lưu trong U (hoặc trong $\underline{\mathscr{C}}$) được lưu thực tế trong các thể hiện của Q_i:
  - Cho TU, một thể hiện bất kỳ của U, $TQ_i = TU[Q_i^+]$
  - 2 điều kiện sau phải được thỏa:

    1. $\forall\, u \in TU,\ u \in \bowtie_{i=1}^{n} TQ_i$

    2. $\forall q \in \bowtie_{i=1}^{n} TQ_i,\ q \in TU$

II.4 Cấu trúc CSDL tương đương (2)

- Hai điều kiện này không phải luôn luôn được thỏa
- …mà thông thường ta có:

$$TU \subseteq \bowtie_{i=1}^{n} TU[Q_i^+], \forall TU$$

- Ví dụ 2.14: với cấu trúc quan hệ phổ quát của Vd 2.4:

  < LỊCH_COI_THI (GVCT, N,G,P,M,GV) ,

  $F = \{f_1: GVCT \rightarrow N,G,P;\ \ f_2: M \rightarrow GV;\ \ f_3: N,G,P \rightarrow M\}$ >

  Cho cấu trúc CSDL sau:

  C1={<COI_THI_1 (GVCT,N,G,M), $F_1' = \{f'_1: GVCT \rightarrow N,G,M\}$>;

  <GD (M,GV), $F_2' = \{f'_2: M \rightarrow GV\}$>;

  <LICH_KT (N,G,P,M)>; $F_3' = \{f'_3: N,G,P \rightarrow M\}$> }

II.4 Cấu trúc CSDL tương đương (3)

T_LICH_KT

N	G	P	M
2	8:00–10:00	101	Giải thuật
2	10:00–12:00	101	CSDL
2	8:00–10:00	103	Giải thuật

T_COI_THI_1

GV_CT	N	G	M
Ng.V.A	2	8:00–10:00	Giải thuật
Ng.T.B	2	10:00–12:00	CSDL
Tr.V.C	2	10:00–12:00	CSDL
Võ.T.D	2	8:00–10:00	Giải thuật

T_COI_THI_1 ⋈ T_LICH_KT ⋈ T_GD

GV_CT	N	G	P	M	GV
Ng.V.A	2	8:00–10:00	101	Giải thuật	X
Ng.V.A	2	8:00–10:00	103	Giải thuật	X
Ng.T.B	2	10:00–12:00	101	CSDL	Y
Tr.V.C	2	10:00–12:00	101	CSDL	Y
Võ.T.D	2	8:00–10:00	101	Giải thuật	X
Võ.T.D	2	8:00–10:00	103	Giải thuật	X

T_GD

M	GV
Giải thuật	X
CSDL	Y

Giống với TU?
(trang 15)

II.4 Cấu trúc CSDL tương đương (4)

- **Ba quan niệm về tính tương đương lược đồ CSDL:**
  - **Tính chất bảo toàn phụ thuộc**
  - **Tính chất bảo toàn thông tin**
  - **Biểu diễn trọn vẹn**

II.4.1 Tính chất bảo toàn phụ thuộc (i)

- Ưu tiên việc kiểm tra các phụ thuộc dữ liệu
- Quan niệm: các thông tin của CSDL được thể hiện thông qua phụ thuộc dữ liệu

II.4.1 Tính chất bảo toàn phụ thuộc (i) (2)

- **Đối với phụ thuộc hàm:**

  Cho $\mathscr{C}_1$= <Q,$\mathscr{F}$>

  và $\mathscr{C}_2$ = {<Q_i, $\mathscr{F}_i$>}, i=1..n, với $\mathscr{F}_i = \mathscr{F}^+[Q_i^+]$

  ***($\mathscr{F}_i$ là những pth của F** được định nghĩa trên* **Q_i)**

  $\mathscr{C}_1 \equiv \mathscr{C}_2$ nếu thỏa hai điều kiện sau:

  (i.1) $\cup_i Q_i^+ = Q^+$

  (i.2) $(\cup_i \mathscr{F}_i)^+ = \mathscr{F}^+$ **($\cup_i \mathscr{F}_i \equiv \mathscr{F}$)**

  <u>Ví dụ 2.16</u>:

  - **Kiểm tra tính tương đương (theo tiêu chuẩn btpth) của cấu trúc CSDL trong Vd 2.6 và 2.9 với cấu trúc ban đầu trong Vd 2.4**

II.4.2 Tính chất bảo toàn thông tin (ii)

- $\underline{\mathscr{C}}_1 \equiv \underline{\mathscr{C}}_2$ **nếu thoả hai điều kiện sau:**

  (ii.1) $\cup_i Q_i^+ = Q^+$

  (ii.2) $\bowtie_{i=1}^{n} Q[Q_i^+] = Q$ **, nghĩa là:**

$$\forall TQ \in Q,\ TQ = \bowtie_{i=1}^{n} TQ[Q_i^+]$$

II.4.2 Tính chất bảo toàn thông tin (ii) - Phương tiện kiểm tra

- **Cho:** $\underline{\mathscr{C}}_1 = <Q, \mathscr{F}>$

  và $\underline{\mathscr{C}}_2 = \{<Q_i, \mathscr{F}_i>\}$, **i=1..n, với** $\mathscr{F}_i = \mathscr{F}^+[Q_i^+]$

- **Bảng Tableau và qui trình thay thế đuổi, dựa vào *luật pth*, được sử dụng để kiểm tra tính bảo toàn thông tin của** $\underline{\mathscr{C}}_2$ **.**

II.4.2 Tính chất bảo toàn thông tin (ii) - Phương tiện kiểm tra (2)

- **Bảng Tableau:**
  - **Có m cột tương ứng với m thuộc tính của quan hệ Q**
  - **n dòng tương tương ứng với n quan hệ con Q_i**
  - **Mỗi ô ở dòng i và cột j của bảng chứa một trong hai giá trị:**
    - **a_j nếu Q_i^+ có chứa thuộc tính tương ứng với cột j**
    - **b_k trong trường hợp ngược lại, với k bắt đầu từ 1 và tăng dần mỗi khi cần dùng đến giá trị b**

II.4.2 Tính chất bảo toàn thông tin (ii) - Phương tiện kiểm tra (3)

- **Luật phụ thuộc hàm áp dụng vào Tableau:**

  Với $X \rightarrow A \in F$:

  Chọn hai dòng w_1 và w_2 trong Tableau sc $w_1.X = w_2.X$:

  <u>Nếu</u> $w_1.A \neq w_2.A$ thì:

  Nếu $w_1.A = a_j$ và $w_2.A = b_k$ thì thay thế b_k trong $w_2.A$ bằng a_j

  Nếu $w_1.A = b_k$ và $w_2.A = a_j$ thì thay thế b_k trong $w_1.A$ bằng a_j

  Nếu $w_1.A = b_k$ và $w_2.A = b_{k'}$ thì thay thế $b_{k'}$ trong $w_1.A$ bằng b_k

  <u>Cuối nếu</u>

II.4.2 Tính chất bảo toàn thông tin (ii) - Phương tiện kiểm tra (4)

- **Qui trình thay thế đuổi:**

1. $\forall f \in F$:

Áp dụng luật pth cho f

Cuối $\forall$

2. Lặp lại bước 1 cho đến khi nào không còn thay đổi được giá trị nào trong bảng Tableau thì dừng.

II.4.2 Tính chất bảo toàn thông tin (ii) - Phương tiện kiểm tra (5)

- Gọi T' là bảng kết quả sau khi đã áp dụng qui trình thay thế đuổi trên bảng Tableau T
  - Nếu T' có một dòng chứa toàn các giá trị a_j thì $\underline{\mathscr{C}}_2$ bảo toàn thông tin.
  - Ngược lại thì $\underline{\mathscr{C}}_2$ không bảo toàn thông tin.

II.4.2 Tính chất bảo toàn thông tin (ii) - Phương tiện kiểm tra (6)

- **Ví dụ 2.15:**

Xét cấu trúc sau có tương đương với cấu trúc trong Vd 2.4:

$\mathscr{E}_2$ = { <COI_THI (GVCT, N, G, P), F1 = {f1:GVCT→N,G,P}>;

<GD(M,GV); F2 = {f2 :M→GV}>;

<CT_MON(GVCT, M); F3 = {f'1: GVCT→M}>}

	GV_CT	N	G	P	M	GV
COI_THI	a_1	a_2	a_3	a_4	b_1	b_2
GD	b_3	b_4	b_5	b_6	a_5	a_6
CT_MON	a_1	b_7 a_2	b_8 a_3	b_9 a_4	a_5	b_{10} a_6

Áp dụng f_1

Áp dụng f_2

II.4.2 Tính chất bảo toàn thông tin (ii) (3)

Một thể hiện minh họa:

T_COI_THI

GVCT	N	G	P
Ng.V.A	2	8:00 – 10:00	101
Ng.T.B	2	10:00 – 12:00	101
Tr.V.C	2	10:00 – 12:00	101
Vo.T.D	2	8:00 – 10:00	103

T_GD

M	GV
Giải thuật	X
CSDL	Y

T_CT_MON

GVCT	M
Ng.V.A	Giải thuật
Ng.T.B	CSDL
Tr.V.C	CSDL
Vo.T.D	Giải thuật

T_COI_THI ⋈ T_CT_MON ⋈ T_GD

GVCT	N	G	P	M	GV
Ng.V.A	2	8:00–10:00	101	Giải thuật	X
Ng.T.B	2	10:00–12:00	101	CSDL	Y
Tr.V.C	2	10:00–12:00	101	CSDL	Y
Võ.T.D	2	8:00–10:00	101	Giải thuật	X

II.4.2 Tính chất bảo toàn thông tin (ii) (4)

- **Ví dụ 2.16 :**
  - **Dùng bảng Tableau và qui trình thay thế đuổi để kiểm tra tính bảo toàn thông tin của cấu trúc trong ví dụ 3.15**

	GV_CT	N	G	P	M	GV
COI_THI_1	a_1	a_2	a_3	b_1	a_5	b_2 ~~a_6~~
LICH_KT	b_3	a_2	a_3	a_4	a_5	b_4 ~~a_6~~
GD	b_5	b_6	b_7	b_8	a_5	a_6

Áp dụng f_2 (M→GV)

→ Không thay thế được nữa và không tìm được dòng nào chứa toàn a_j → C2 không bảo toàn thông tin.

II.4.3 Biểu diễn trọn vẹn

- Cấu trúc CSDL $\underline{\mathscr{C}}_2$ là một biểu diễn trọn vẹn của $\underline{\mathscr{C}}_1$ nếu 3 điều kiện (i.1), (i.2) và (ii.2) đều được thỏa.
- $\underline{\mathscr{C}}_2$ là biểu diễn trọn vẹn ≡ $\underline{\mathscr{C}}_1$ vừa bảo toàn thông tin, vừa tạo thuận lợi cho việc kiểm tra PTH

II.4.3 Biểu diễn trọn vẹn (2)

Ví dụ 2.17:

- **Cấu trúc trong Vd 2.6 và 2.9 đạt tiêu chuẩn biểu diễn trọn vẹn**
- **Cấu trúc trong Vd 2.15 đạt tiêu chuẩn bảo toàn thông tin, nhưng không bảo toàn pth**

  (f_3: N,G,P$\rightarrow$ M không còn được định nghĩa trên quan hệ nào)

Câu hỏi?

Buổi # 5
(lớp TH2008/1)

Cơ sở dữ liệu nâng cao

Đồng Thị Bích thủy

12/10/2010

CHƯƠNG II:
Giai đoạn thiết kế quan niệm
(tt và hết)

Nội dung tuần này

*** Nhắc lại hai tiêu chuẩn thiết kế

II.5 Hai cách tiếp cận để thiết kế

- II.5.1 Cách tiếp cận phận rã
- II.5.2 Cách tiếp cận tổng hợp
- II.5.3 So sánh hai hướng tiếp cận

**** Nhắc lại hai tiêu chuẩn thiết kế

- **Dạng chuẩn, để:**
  - ↙ tối đa thông tin trùng lắp
  - Kiểm tra pth hiệu quả (ít tốn kém)
- **Cấu trúc tương đương, với ba quan niệm:**
  - Tính chất bảo toàn phụ thuộc (btpth)
  - Tính chất bảo toàn thông tin (bttt)
  - Biểu diễn trọn vẹn (btpth + bttt)

Về các điều kiện của btpth & bttt

Cho $\mathscr{C}_1$= <Q,$\mathscr{F}$>

và $\mathscr{C}_2$ = {<Q_i, $\mathscr{F}_i$>}, i=1..n, với $\mathscr{F}_i = \mathscr{F}^+[Q_i^+]$

($\mathscr{F}_i$ là những pth của F** được định nghĩa trên **Q_i)

- **Về tính chất btpth:**

  $\mathscr{C}_1 \equiv \mathscr{C}_2$ nếu thỏa hai điều kiện sau:

  (i.1) $\cup_i Q_i^+ = Q^+$ (i.2) $(\cup_i \mathscr{F}_i)^+ = \mathscr{F}^+$ ($\cup_i \mathscr{F}_i \equiv \mathscr{F}$)

- **Về tính chất bttt:**

  $\mathscr{C}_1 \equiv \mathscr{C}_2$ nếu thoả hai điều kiện sau:

  (ii.1) $\cup_i Q_i^+ = Q^+$ (ii.2) $\bowtie_{i=1}^{n} Q[Q_i^+] = Q$

Về một biểu diễn trọn vẹn

- Cấu trúc CSDL $\underline{\mathscr{C}}_2$ là một biểu diễn trọn vẹn của $\underline{\mathscr{C}}_1$ nếu 3 điều kiện (i.1), (i.2) và (ii.2) đều được thỏa.

- $\underline{\mathscr{C}}_2$ là biểu diễn trọn vẹn

$\Updownarrow$

$\underline{\mathscr{C}}_2$ vừa bảo toàn thông tin, vừa tạo thuận lợi cho việc kiểm tra PTH

Vài ví dụ lược đồ CSDL

- Với quan hệ phổ quát sau:

  LỊCH_COI_THI (GVCT, N,G,P,M,GV)

  F = {f1: GVCT $\rightarrow$ N,G,P; f2: M $\rightarrow$ GV;

  f3: N,G,P $\rightarrow$ M; f4: M $\rightarrow$ P}

- Lược đồ gồm COI_THI (<u>GVCT</u>, N, G, P);

  LỊCH_KT(<u>P,N,G, M</u>); GD(<u>M</u>,GV)

  là một biểu diễn trọn vẹn (vừa btpth, vừa bttt).

Vài ví dụ lược đồ CSDL (tt)

- Lược đồ gồm COI_THI (GVCT, N,G,P); LICH_KT_1 (M,P); LICH_KT_2 (M,N,G); GD (M,GV)

  không btpth (thiếu f3) và không bttt.

- Lược đồ gồm COI_THI (GVCT, N, G, P); GD(M,GV); CT_MON(GVCT, M)

  không btpth (thiếu f3, f4), nhưng có bttt.

II.5 Hai cách tiếp cận để thiết kế

- Cách tiếp cận **phân rã**
  - Thường được áp dụng để tinh chỉnh một lược đồ CSDL
- Cách tiếp cận **tổng hợp**
  - Thường được áp dụng để kiểm nghiệm một lược đồ CSDL

II.5.1 Cách tiếp cận phân rã – Cơ sở lý thuyết

- **Đối với phụ thuộc hàm:**
  - ***<u>Định lý 1</u>:*** **Cho <Q, $\mathscr{F}$>**

    $Q = Q_1 \bowtie Q_2$ không mất thông tin

    (* nghĩa là bttt: thoả (ii.1) & (ii.2) *)

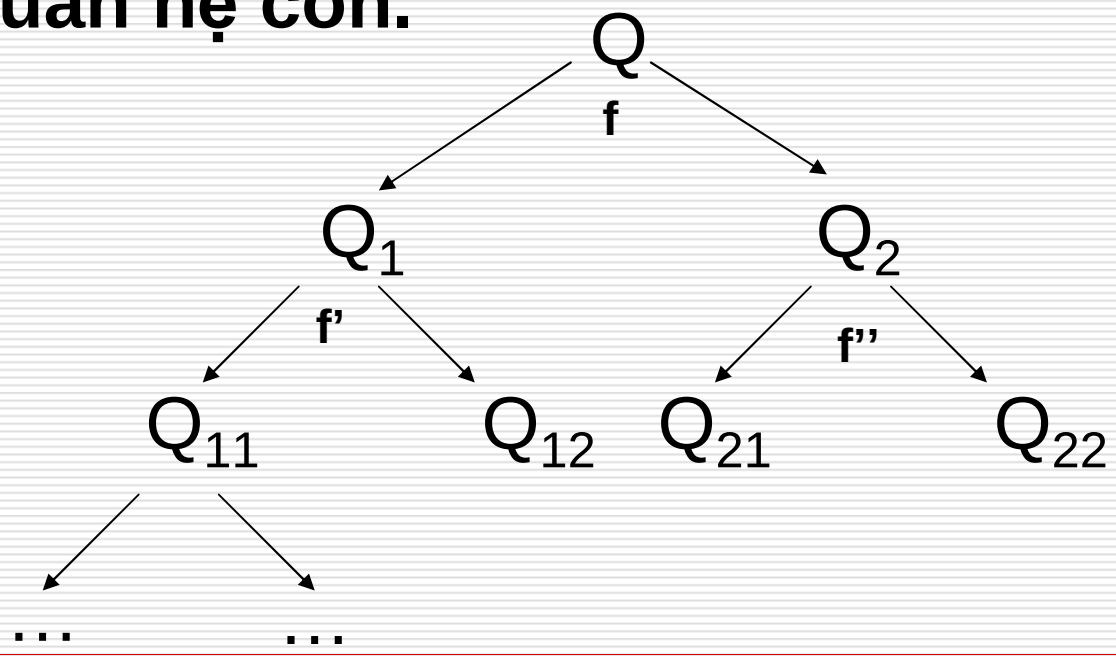
    nếu và chỉ nếu:

    $$((Q_1^+ \cap Q_2^+) \rightarrow (Q_1^+ - Q_2^+)) \in \mathscr{F}^+$$

    hoặc $((Q_1^+ \cap Q_2^+) \rightarrow (Q_2^+ - Q_1^+)) \in \mathscr{F}^+$
  - **Vận dụng định lý 1: $\forall f: X \rightarrow A \in \mathscr{F}$, Q được phân rã thành $Q_1(XA)$ và $Q_2(Q^+ - A)$**

II.5.1 Cách tiếp cận phân rã (tt)

- **Ứng dụng định lý 1, từ một quan hệ cần phân rã, ta sẽ có cây phân rã nhị phân, trong đó, ở mỗi cấp, một quan hệ được tách thành hai quan hệ con.**



II.5.1 Thuật toán phân rã

- **Vào: <Q,D>**
- **Ra: $\rho = \{Q_i\}$, i=1..n hoặc**

  $\underline{C}' = \{<Q_i, F_i>\}$, i=1..n

II.5.1 Thuật toán phân rã (tt)

- Thủ tục Phân_rã (Q)
  - Xác định tập $F_Q = \{f: X \rightarrow Y \in D^+ \mid (XY) \subseteq Q^+\}$
  - Nếu: $\forall (f: X \rightarrow Y \in F_Q), (XY) = Q^+$

    thì: $\rho := \rho \cup \{Q\}$

    nếu không:
    - Chọn $(f: X \rightarrow Y) \in D_Q$ với $(XY) \subset Q^+$ để phân rã (*)
    - Q1 = Q[XY] ; Q2 = Q[Q^+ -Y]
    - Phân_rã (Q1)
    - Phân_rã (Q2)
- Cuối thủ tục Phân_rã

(*): xem nhận xét

II.5.1 Thuật toán phân rã – Ví dụ

- **Ví dụ 2.18:**

  Cho quan hệ : Q(MS_CĐ, MSSV, CĐ, HG)

  D = { **f1: MS_CĐ → CĐ;**

  f2: CĐ →MS_CĐ;

  f3: CD, MS_SV → HG;

  f4: MS_CĐ, HG →MS_SV;

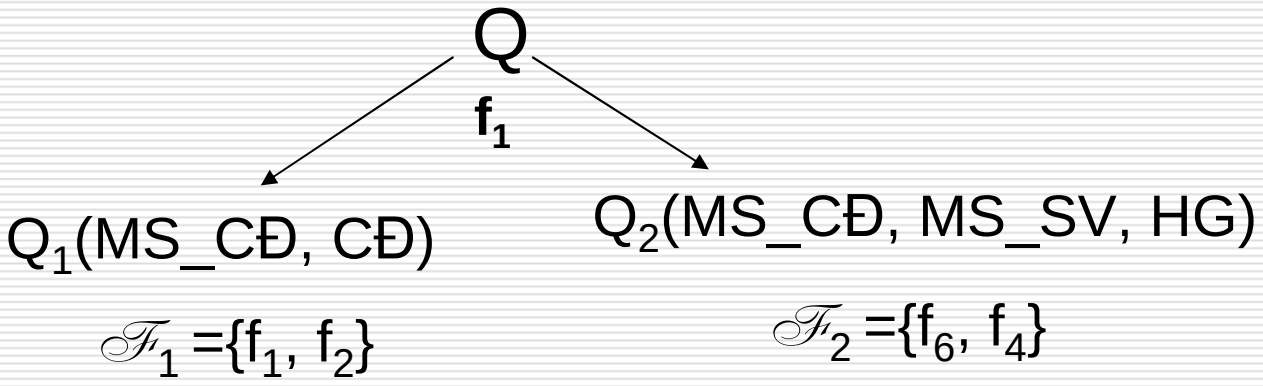
  f5: CĐ, HG →MS_SV;

  f6: MS_CĐ, MS_SV →HG;

  }

II.5.1 Thuật toán phân rã – Ví dụ (tt)

- **Phân rã Q:**



- **C1 = { <Q1(MS_CĐ, CĐ), D1>;**
  <Q2(MS_CĐ, MS_SV, HG), D2}
- **C1 có bảo toàn phụ thuộc?**

II.5.1 Thuật toán phân rã – Ví dụ (tt)

- **Phân rã Q với D có thứ tự khác:**

Q

f_4

Q'_1(MS_CĐ, HG, MS_SV) Q'_2(MS_CĐ, CĐ, HG)

$\mathscr{F}_1 = \{f_3, f_5\}$ $\mathscr{F}_2 = \{f_1, f_2\}$

f_1

Q'_{21}(MS_CĐ, CĐ) Q'_{22}(MS_CĐ, HG)

$\mathscr{F}_{21} = \{f_1, f_2\}$ $\mathscr{F}_{22} = \varnothing$

- **C2 = {< Q'1(MS_CĐ, HG, MS_SV), F'1>;**
  <Q'21(MS_CĐ, CĐ), F'21>;
  < Q'22(MS_CĐ, HG), F'22> }

II.5.1 Thuật toán phân rã – Nhận xét

- Tất cả các quan hệ trong lược đồ kết quả đều đạt DC BCK (nếu chỉ có pth).
- Phân rã bảo toàn thông tin (do áp dụng định lý 1), nhưng không đảm bảo bảo toàn pth.
- Thuật toán không quan tâm đến "chất lượng" của tập phụ thuộc F ban đầu: F có thể chứa các pth không đầy đủ, pth dư thừa.
- Thuật toán không quan tâm đến việc xác định khóa của các quan hệ con.

II.5.1 Thuật toán phân rã – Nhận xét (tt):

- Tùy vào cách chọn phụ thuộc để phân rã (bước 3) mà cấu trúc kết quả sẽ khác nhau:
  - Trong ví dụ 2.18: Chọn theo thứ tự từ trên xuống: kết quả khác nhau tùy theo thứ tự của F, và hậu quả là trong lược đồ kết quả có thể chứa những quan hệ không có ý nghĩa (ví dụ Q'22)
  - Có một số nghiên cứu để cải tiến thuật toán phân rã: xây dựng chiến lược chọn phụ thuộc để phân rã hiệu quả hơn (bảo toàn pth,…)

II.5.2 Cách tiếp cận tổng hợp

- **Mục tiêu:**

  Từ một lược đồ quan hệ phổ quát Q với tập phụ thuộc hàm F được định nghĩa trên Q, tìm một cấu trúc C ={ <Q_i, F_i>} sao cho:

  - **Các pth được kiểm tra dễ dàng trong các Q_i, nghĩa là với mọi pth trong quan hệ phổ quát, chỉ cần kiểm tra pth này một cách cục bộ tại quan hệ Q_i mà nó được định nghĩa**
  - **Các Q_i đạt tối thiểu DC3.**
  - **Bảo toàn phụ thuộc hàm.**

II.5.2 Thuật toán tổng hợp

- Vào: <U, F> với F = {f: X→A}
- Ra: C = {<Q_i, F_i>}, i=1..n
- Thuật toán Tổng_hợp(U,F)
  1. Tìm phủ tối thiểu PTT của F
  2. Chia PTT thành những nhóm F_i', i=1..p, mỗi F_i' gồm những pth thuộc PTT có cùng vế trái
     - *Ký hiệu là K_i, vế trái của các PTH trong mỗi F_i' đó là các khoá hoặc siêu khóa của các quan hệ con trong tương lai*

II.5.2 Thuật toán tổng hợp (tt)

3. Gộp các F_i' có các (siêu) khoá tương đương lại thành cùng một nhóm, ký hiệu F_i (*K_i và K_j là tương đương nếu các pth $K_i \rightarrow K_j$ và $K_j \rightarrow K_i$ thuộc F^+*) :

3.1. $J = \varnothing$

3.2. $\forall$ F_i' , F_j' sao cho $K_i \leftrightarrow K_j \in F^+$ thì

Nhập F_i ' và F_j ' lại thành một nhóm, ký hiệu F_i

Cuối $\forall$

3.3 $J := J \cup \{K_i \rightarrow K_j , K_j \rightarrow K_i\}$

II.5.2 Thuật toán tổng hợp (tt)

3.4. $\forall A \in K_j$:

Nếu $K_i \rightarrow A \in$ PTT **thì**

PTT:=PTT- $\{K_i \rightarrow A\}$;

Loại $K_i \rightarrow A$ ra khỏi nhóm F_i

Cuối nếu

Cuối $\forall$

$\forall B \in K_i$:

Nếu $K_j \rightarrow B \in$ PTT **thì**

PTT:=PTT- $\{K_j \rightarrow B\}$;

Loại $K_j \rightarrow B$ ra khỏi nhóm Fj

Cuối nếu

Cuối $\forall$

II.5.2 Thuật toán tổng hợp (tt)

4. **Hủy các pth bắc cầu:**

   4.1 Tìm PTT' $\subseteq$ PTT sao cho:

   $$(PTT' \cup J) \equiv (PTT \cup J)$$

   4.2 $\forall f \in$ PTT - PTT' : loại f ra khỏi nhóm F_i mà nó hiện diện

   4.3 $\forall f: K_i \rightarrow K_j \in J$: đưa trở vào F_i

5. **Tạo các quan hệ con:**

   $\forall F_i$: Tạo Q_i = U[Các thuộc tính của tất cả $f \in F_i$]

   $\underline{C} = \underline{C} \cup \{<Q_i, F_i>\}$

   Cuối $\forall$

Cuối thuật toán Tổng hợp

II.5.2 Thuật toán tổng hợp – Ví dụ

- **Cho quan hệ phổ quát của ví dụ 2.18. Dùng thuật toán tổng hợp để tìm một lược đồ <u>C</u> tương ứng đạt DC3:**
  1. **PTT = {f1, f2, f3, f4}**
  2. **Có 4 nhóm F_1' = {f1}; F_2' = {f2}; F_3' = {f3}; F_4' = {f4}**
  3. **3.1 J = ∅**

     3.2 F_1 = {f1, f2}; F_2 = {f3, f4}

II.5.2 Thuật toán tổng hợp – Ví dụ (tt)

3.3 J = { f1: MS_CĐ → CĐ;

f2: CĐ →MS_CĐ;

f3: CĐ, MS_SV → HG;

f'3 :CĐ, MS_SV → MS_CĐ;

f4: MS_CĐ, HG →MS_SV;

f'4: MS_CĐ, HG →CĐ

}

3.4 PTT = ∅; F_1 = ∅; F_2 = ∅

4.1, 4.2 : không có

II.5.2 Thuật toán tổng hợp – Ví dụ (tt)

4.3 F_1 = {f1, f2}

F_2 = {f3, f'3, f4, f'4}

5. <u>C</u> = {<Q12(MS_CĐ, CĐ), F_1>;

<Q34(CĐ, MS_SV, MS_CĐ, HG), F_2>}

II.5.2 Thuật toán tổng hợp – Đánh giá

- Các quan hệ trong lược đồ kết quả đều đạt DC3
- Kết quả tìm PTT và PTT' ở bước 1 và 4.1 có thể khác nhau tùy vào thứ tự các pth, do đó, dẫn đến cấu trúc kết quả của thuật toán tổng hợp cũng sẽ khác nhau.
- Thuật toán không xác định được đầy đủ các khoá của các Q_i
  - <u>Ví dụ 2.19</u>: Q34 còn 2 khoá khác là (MS_CĐ, MSSV), (CĐ,HG) nhưng thuật toán không xác định được.

II.5.2 Thuật toán tổng hợp – Đánh giá

- **Trong lược đồ kết quả của thuật toán, các F_i chỉ chứa các pth mà vế trái là một khóa, có thể có những pth khác định nghĩa trên Qi nhưng không ở dạng trên thì không được thuật toán đưa vào tập Fi.**

  <u>Ví dụ 2.20</u>: Q(ABC), F = {f1: AB→C; f2:C→ B}

  Lược đồ kết quả <u>C</u> = { <Q1(<u>AB</u>C), F1 = {f1}>;

  <Q2(<u>C</u>B), F2 = {f2}> }

  → Trên Q1 còn định nghĩa PTH f2, nhưng không có trong kết quả của thuật toán.

II.5.2 Thuật toán tổng hợp – Đánh giá (tt)

- **Đảm bảo tính chất (i.2) của tiêu chuẩn bảo toàn phụ thuộc hàm, nhưng tính chất (i.1) có thể không thỏa**

  <u>Ví dụ 2.21</u>: Q(ABCD), F = {f1: AB→C; f2:C→ B}

  Lược đồ kết quả <u>C</u> = { <Q1(<u>AB</u>C), F1 = {f1}>;

  <Q2(<u>C</u>B), F2 = {f2}> }

  → Thuộc tính D không hiện diện trong quan hệ nào của lược đồ kết quả

- **Kết quả của thuật toán không đảm bảo tính bảo toàn thông tin.**

II.5.2 Thuật toán tổng hợp – Cải tiến

- **Thêm một bước sau cùng vào thuật toán để đảm bảo tính chất bảo toàn thông tin:**

  Bước 6:

  Tìm khóa K_0 của quan hệ phổ quát U

  Nếu không tồn tại Q_i có khóa là một khóa của U thì:

  Thêm vào lược đồ kết quả C một quan hệ

  $Qk = U[K_0]$ với $F_k = \varnothing$

  Cuối nếu

- ***Ví dụ:*** **trong ví dụ 2.21, tìm được một khóa của Q là ACD, ta thêm vào trong kết quả:**

  <Q3(ACD), F3 = ∅>

II.5.3 So sánh hai hướng tiếp cận

Cách tiếp cận / Tiêu chí phân tích	Phân rã	Tổng hợp (có cải tiến)
1. Loại phụ thuộc dữ liệu được dùng	Phụ thuộc hàm và *Phụ thuộc đa trị*	Phụ thuộc hàm
2. Dạng chuẩn	DC BCK hoặc DC4	DC3
3. Bảo toàn thông tin	Có	Có
4. Bảo toàn phụ thuộc	Không	Có
5. Có sự lựa chọn cấu trúc CSDL	Ngầm có, thông qua việc chọn phụ thuộc để phân rã	Ngầm có, thông qua việc xác định PTT của $\mathcal{F}$
6. Xác định đủ khóa của quan hệ con	Không quan tâm	Không đủ

Câu hỏi?

Buổi # 6
(lớp TH2008/1)

Cơ sở dữ liệu nâng cao

Đồng Thị Bích thủy

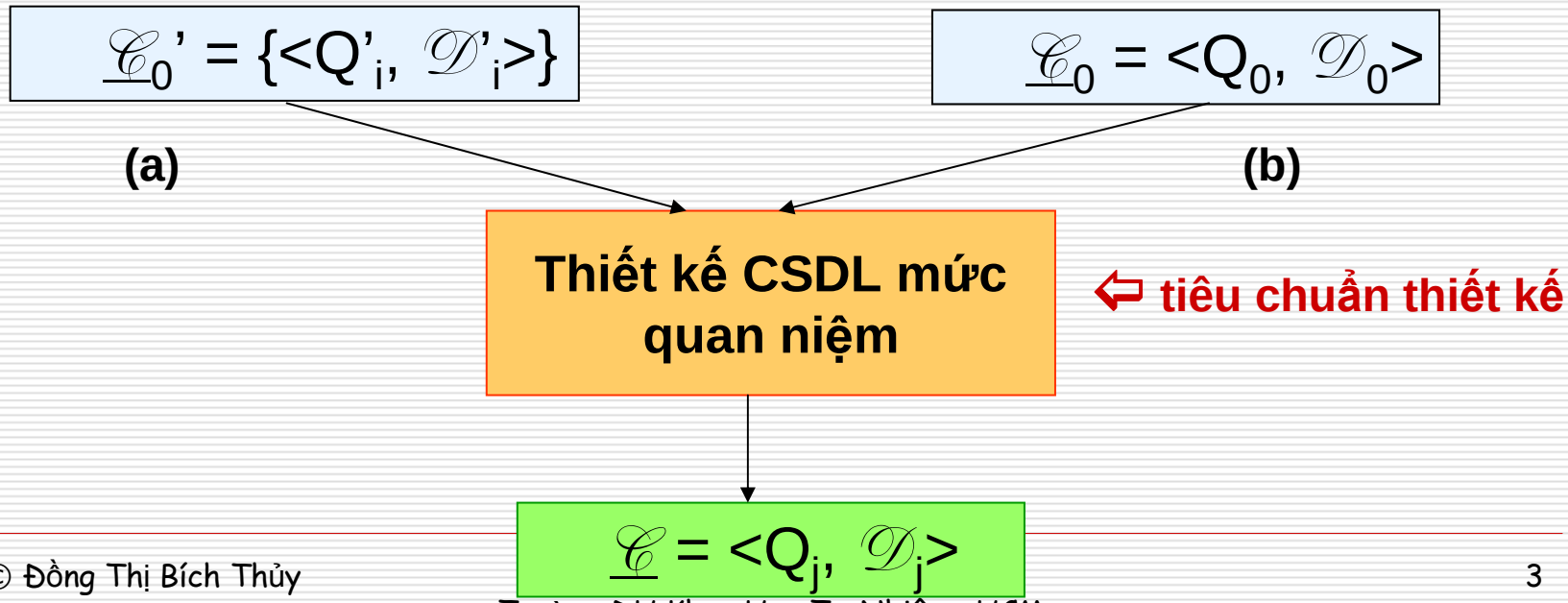
19/10/2010

CHƯƠNG II:
Giai đoạn thiết kế quan niệm
(ôn lại)

Giai đoạn thiết kế quan niệm

- **Tiếp nối gđ Phân tích-thu thập nhu cầu:**

  Biến đổi đầu vào thành một cấu trúc CSDL quan niệm chất lượng hơn, dựa trên một số tiêu chuẩn thiết kế



Hai tiêu chuẩn thiết kế

- **Dạng chuẩn, để:**
  - ↙ tối đa thông tin trùng lắp
  - Kiểm tra pth hiệu quả (ít tốn kém)
- **Cấu trúc tương đương, với ba quan niệm:**
  - Tính chất bảo toàn phụ thuộc (btpth)
  - Tính chất bảo toàn thông tin (bttt)
  - Biểu diễn trọn vẹn (btpth + bttt) ←

Tổng kết ưu-khuyết điểm của 2 DC cuối

DC của lược đồ / Tiêu chí	DC3	DC BCK
Trùng lắp thông tin	Có thể còn	Không
Kiểm tra F_0	Hiệu quả → tương đối hiệu quả	Có thể còn pth có bối cảnh trên nhiều Qi
Bảo toàn pth	Có	Có thể không

Về tiêu chuẩn đánh giá lược đồ CSDL tương đương

Cho $\mathscr{C}_1$= <Q,$\mathscr{F}$>

và $\mathscr{C}_2$ = {<Q_i, $\mathscr{F}_i$>}, i=1..n, với $\mathscr{F}_i = \mathscr{F}^+[Q_i^+]$

($\mathscr{F}_i$ là những pth của F** được định nghĩa trên **Q_i)

- **Về tính chất btpth:**

  $\mathscr{C}_1 \equiv \mathscr{C}_2$ nếu thỏa hai điều kiện sau:

  (i.1) $\cup_i Q_i^+ = Q^+$ (i.2) $(\cup_i \mathscr{F}_i)^+ = \mathscr{F}^+$ ($\cup_i \mathscr{F}_i \equiv \mathscr{F}$)

- **Về tính chất bttt:**

  $\mathscr{C}_1 \equiv \mathscr{C}_2$ nếu thoả hai điều kiện sau:

  (ii.1) $\cup_i Q_i^+ = Q^+$ (ii.2) $\bowtie_{i=1}^{n} Q[Q_i^+] = Q$

Về tiêu chuẩn đánh giá lược đồ CSDL tương đương (2)

- **Về một biểu diễn trọn vẹn**

  Cấu trúc CSDL $\underline{\mathscr{C}}_2$ là một biểu diễn trọn vẹn của $\underline{\mathscr{C}}_1$ nếu 3 điều kiện:

  (i.1), (i.2) và (ii.2)

  đều được thỏa.

- **$\underline{\mathscr{C}}_2$ là biểu diễn trọn vẹn**

$\Updownarrow$

$\underline{\mathscr{C}}_2$ vừa bảo toàn thông tin, vừa tạo thuận lợi cho việc kiểm tra PTH

Vài ví dụ lược đồ CSDL

- Với quan hệ phổ quát sau:

  LỊCH_COI_THI (GVCT, N,G,P,M,GV)

  F = {f1: GVCT $\rightarrow$ N,G,P; f2: M $\rightarrow$ GV;

  f3: N,G,P $\rightarrow$ M; f4: M $\rightarrow$ P}

- Lược đồ gồm COI_THI (<u>GVCT</u>, N, G, P);

  LỊCH_KT(<u>P,N,G, M</u>); GD(<u>M</u>,GV)

  là một biểu diễn trọn vẹn (vừa btpth, vừa bttt) và đạt DC3.

Vài ví dụ lược đồ CSDL (2)

- Lược đồ gồm COI_THI (GVCT, N,G,P); LICH_KT_1 (M,P); LICH_KT_2 (M,N,G); GD (M,GV)

  không btpth (thiếu f3), không bttt nhưng đạt DC BCK.

- Lược đồ gồm COI_THI (GVCT, N, G, P); GD(M,GV); CT_MON(GVCT, M)

  không btpth (thiếu f3, f4), nhưng có bttt và đạt DC BCK.

Hai cách tiếp cận để thiết kế

- Cách tiếp cận **phân rã**
  - Thường được áp dụng để tinh chỉnh một lược đồ CSDL
- Cách tiếp cận **tổng hợp**
  - Thường được áp dụng để kiểm nghiệm một lược đồ CSDL

Cách tiếp cận phân rã

- **Đối với phụ thuộc hàm:**
  - ***<u>Định lý 1</u>:*** **Cho <Q, $\mathscr{F}$>**

    $Q = Q_1 \bowtie Q_2$ không mất thông tin

    (* nghĩa là bttt: thoả (ii.1) & (ii.2) *)

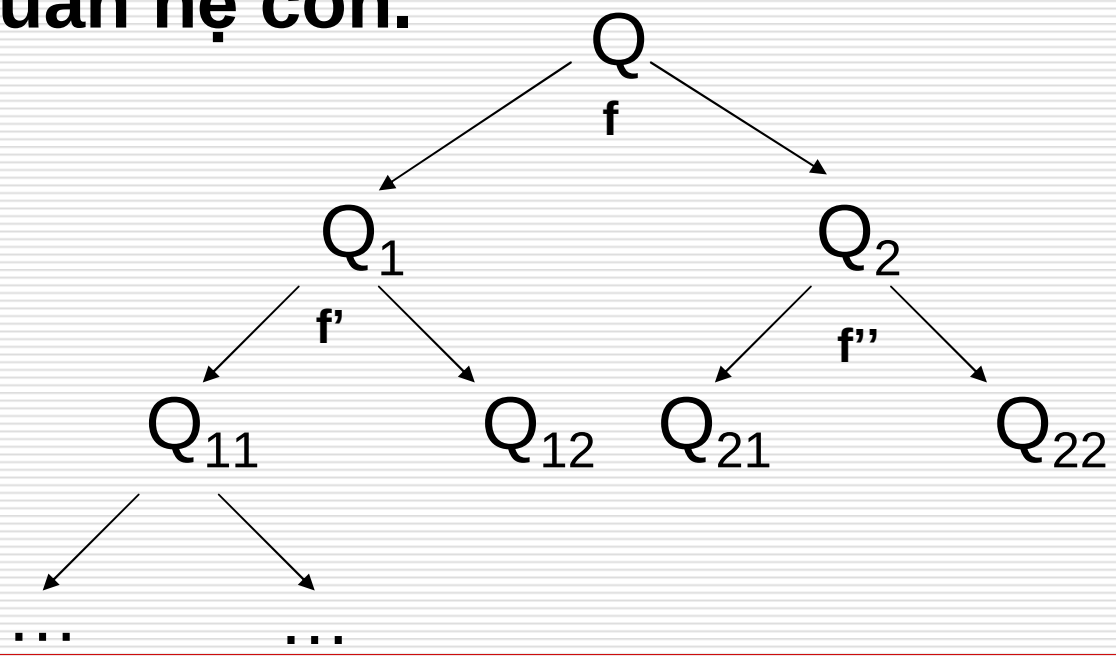
    nếu và chỉ nếu:

    $$((Q_1^+ \cap Q_2^+) \rightarrow (Q_1^+ - Q_2^+)) \in \mathscr{F}^+$$

    hoặc $((Q_1^+ \cap Q_2^+) \rightarrow (Q_2^+ - Q_1^+)) \in \mathscr{F}^+$
  - **Vận dụng định lý 1: $\forall f: X \rightarrow A \in \mathscr{F}$, Q được phân rã thành $Q_1(XA)$ và $Q_2(Q^+ - A)$**

Cây phân rã

- **Ứng dụng định lý 1, từ một quan hệ cần phân rã, ta sẽ có cây phân rã nhị phân, trong đó, ở mỗi cấp, một quan hệ được tách thành hai quan hệ con.**



Cách tiếp cận tổng hợp

- **Mục tiêu:**

  Từ một lược đồ quan hệ phổ quát Q với tập phụ thuộc hàm F được định nghĩa trên Q, tìm một cấu trúc C ={ <Q_i, F_i>} sao cho:

  - **Các pth được kiểm tra dễ dàng trong các Q_i, nghĩa là với mọi pth trong quan hệ phổ quát, chỉ cần kiểm tra pth này một cách cục bộ tại quan hệ Q_i mà nó được định nghĩa**
  - **Các Q_i đạt tối thiểu DC3.**
  - **Bảo toàn phụ thuộc hàm.**

So sánh hai hướng tiếp cận

Cách tiếp cận / Tiêu chí phân tích	Phân rã	Tổng hợp (có cải tiến)
1. Loại phụ thuộc dữ liệu được dùng	Phụ thuộc hàm và *Phụ thuộc đa trị*	Phụ thuộc hàm
2. Dạng chuẩn	DC BCK hoặc DC4	DC3
3. Bảo toàn thông tin	Có	Có
4. Bảo toàn phụ thuộc	Không	Có
5. Có sự lựa chọn cấu trúc CSDL	Ngầm có, thông qua việc chọn phụ thuộc để phân rã	Ngầm có, thông qua việc xác định PTT của $\mathcal{F}$
6. Xác định đủ khóa của quan hệ con	Không quan tâm	Không đủ

Câu hỏi?

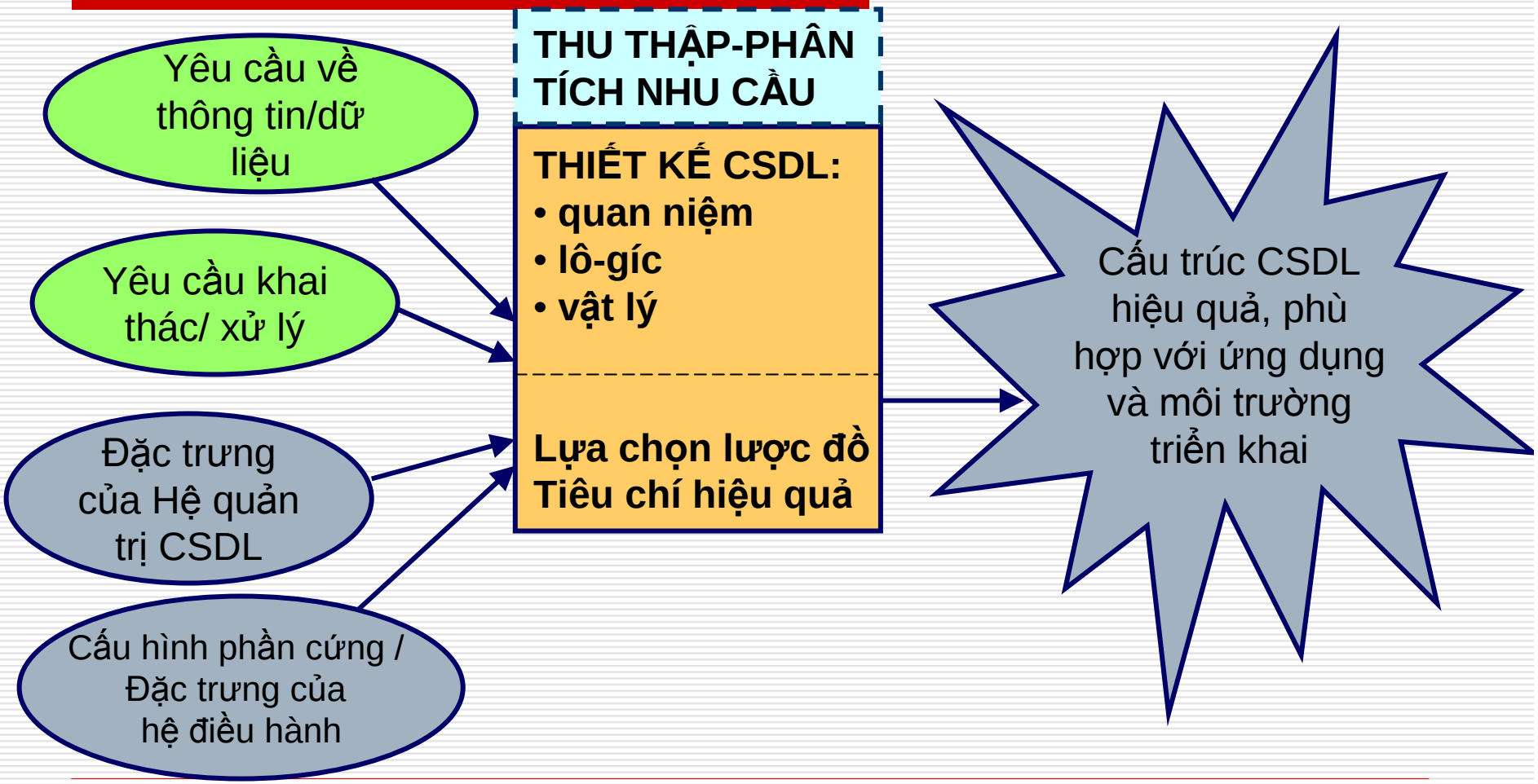
Buổi # 7
(lớp TH2008-1)

Cơ sở dữ liệu nâng cao

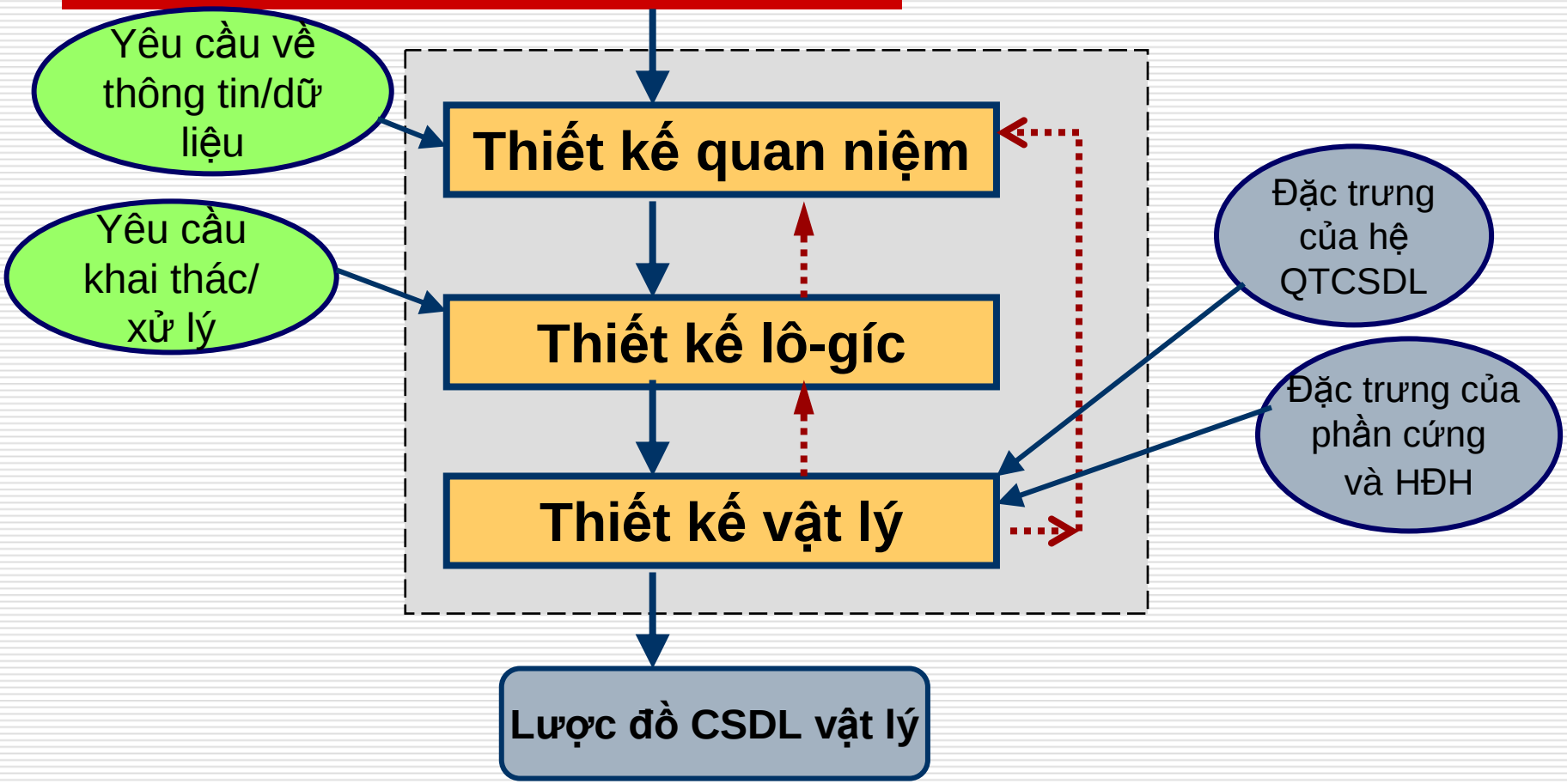
Đồng Thị Bích thủy

22/10/2010

Nhắc lại bối cảnh thiết kế CSDL



Nhắc lại bối cảnh thiết kế CSDL (tt)



CHƯƠNG III:
Giai đoạn thiết kế lô-gíc

Nội dung tuần này

III.1 Mục tiêu của giai đoạn thiết kế lô-gíc

III.2 Nhắc lại một số khái niệm của lý thuyết đồ thị

III.3 Đồ thị các con đường truy xuất

III.4 Đồ thị quan hệ

III.5 Chuyển đổi ĐTQH → ĐT CĐTX và ngược lại

III.6 Chuỗi kết được cài đặt trên đồ thị

III.1 Mục tiêu của giai đoạn

- Giai đoạn thiết kế logic là bước trung gian để giai đoạn thiết vật lý được thực hiện dễ dàng.
  - Biểu diễn lại mô hình CSDL mức quan niệm thành mô hình dữ liệu phù hợp với hệ quản trị sẽ dùng để cài đặt sau này.
    - Độc lập với các đặc trưng khác của hệ quản trị.
  - Chọn một biểu diễn ở dạng đồ thị phù hợp, làm cơ sở cho việc lựa chọn chỉ mục ở giai đoạn thiết kế vật lý.

III.1 Mục tiêu của giai đoạn (tt)

- Kết quả của gđ phải hoàn toàn "trung thực" với cấu trúc quan niệm:
  - Bảo toàn nội dung CSDL
  - Bảo toàn sự truy xuất trực tiếp đến các quan hệ của cấu trúc quan niệm.
    - Khi một quan hệ con Qi tồn tại trong cấu trúc quan niệm C, một bộ của Qi có thể được truy xuất trực tiếp (không cần thông qua một quan hệ nào khác). Cấu trúc logic phải bảo toàn khả năng này.
  - Bảo toàn tính hiệu quả trong việc kiểm tra ràng buộc toàn vẹn.

III.1 Mục tiêu của giai đoạn (tt)

- **Ví dụ 3.1**

  Cho cấu trúc mức quan niệm:

  NhânViên (Ma_NV, HoTen_NV, Ma_P)

  Phòng (Ma_P, Ten_P)

  ĐềÁn (Ma_DA, Ten_DA, Ma_P)

  PhânCông (Ma_NV, Ma_DA),

  ràng buộc: Một nhân viên được phân công vào tất cả các đề án do phòng ban mà nhân viên đó trực thuộc phụ trách)

Ví dụ 3.1 (tt)

Xét cấu trúc logic sau:

NhânViên (Ma_NV, HoTen_NV, Ma_P)

Phòng (Ma_P, Ten_P)

ĐềÁn_1 (Ma_DA, Ten_DA)

PhụTrách (Ma_DA, Ma_P)

Không có quan hệ *PhânCông*: có bảo toàn nội dung không?

Có bảo toàn sự truy xuất trực tiếp đến quan hệ *PhânCông* không?

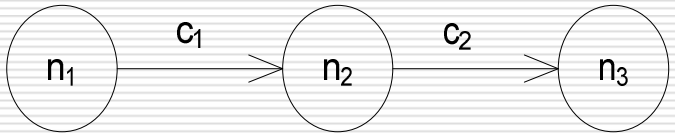
III.2 Lý thuyết đồ thị - một số khái niệm

- **Đồ thị G(N,C) được định nghĩa trên một tập nút $N=\{n_1,n_2,\ldots n_n\}$ và một tập cung $C=\{c_1, c_2,\ldots c_m\}$**
  - **Đồ thị có hướng nếu tồn tại một cung có hướng (khi đó các nút trong đồ thị gọi là nút đi hoặc nút đến), ngược lại đồ thị vô hướng (các nút gọi là nút xuất phát).**
  - ***Đường đi* (đối với đồ thị vô hướng) và *mạch đi* (đối với đồ thị có hướng)**
  - ***Khuyên & Chu trình***

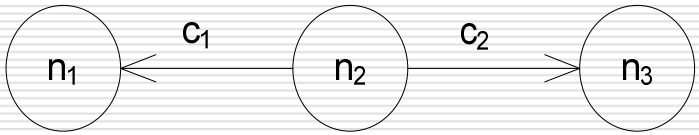
III.2 Lý thuyết đồ thị - một số khái niệm (tt)

- **Dòng có gốc n_1 là một tập cung D = (c_1, c_2, …,c_p) sao cho:**
  - **Một cung trong tập D có nút xuất phát (hoặc nút đi) là n_1**
  - **$\forall$ n_i là nút xuất phát (hoặc nút đi/ đến) của $c_i \in$ D, tồn tại một đường đi hoặc mạch đi từ gốc n_1 đến n_i qua các cung của D.**

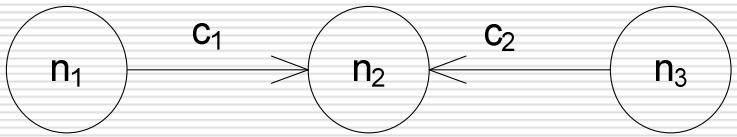
III.2 Lý thuyết đồ thị - Ví dụ 3.2



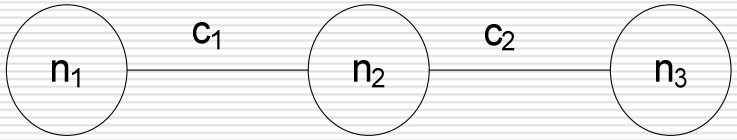
(c_1, c_2) là một dòng có gốc n_1
(c_1, c_2) không là một dòng có gốc n_2
(c_1, c_2) là một mạch đi



(c_1, c_2) là một dòng có gốc n_2
(c_1, c_2) không là một mạch đi



(c_1, c_2) không là dòng của gốc nào cả



(c_1, c_2) là dòng của gốc n_1, n_2 hoặc n_3

III.3 Đồ thị con đường truy xuất

- **Định nghĩa:**
  - **Đồ thị con đường truy xuất là một đồ thị có hướng với:**
    - **N: Tập các nút của đồ thị**
    - **C $\subseteq$ (N x N): Tập các cung (có hướng)**
    - **Q: tập các quan hệ Qi.**
    - **Cđ: Tập các con đường truy xuất**

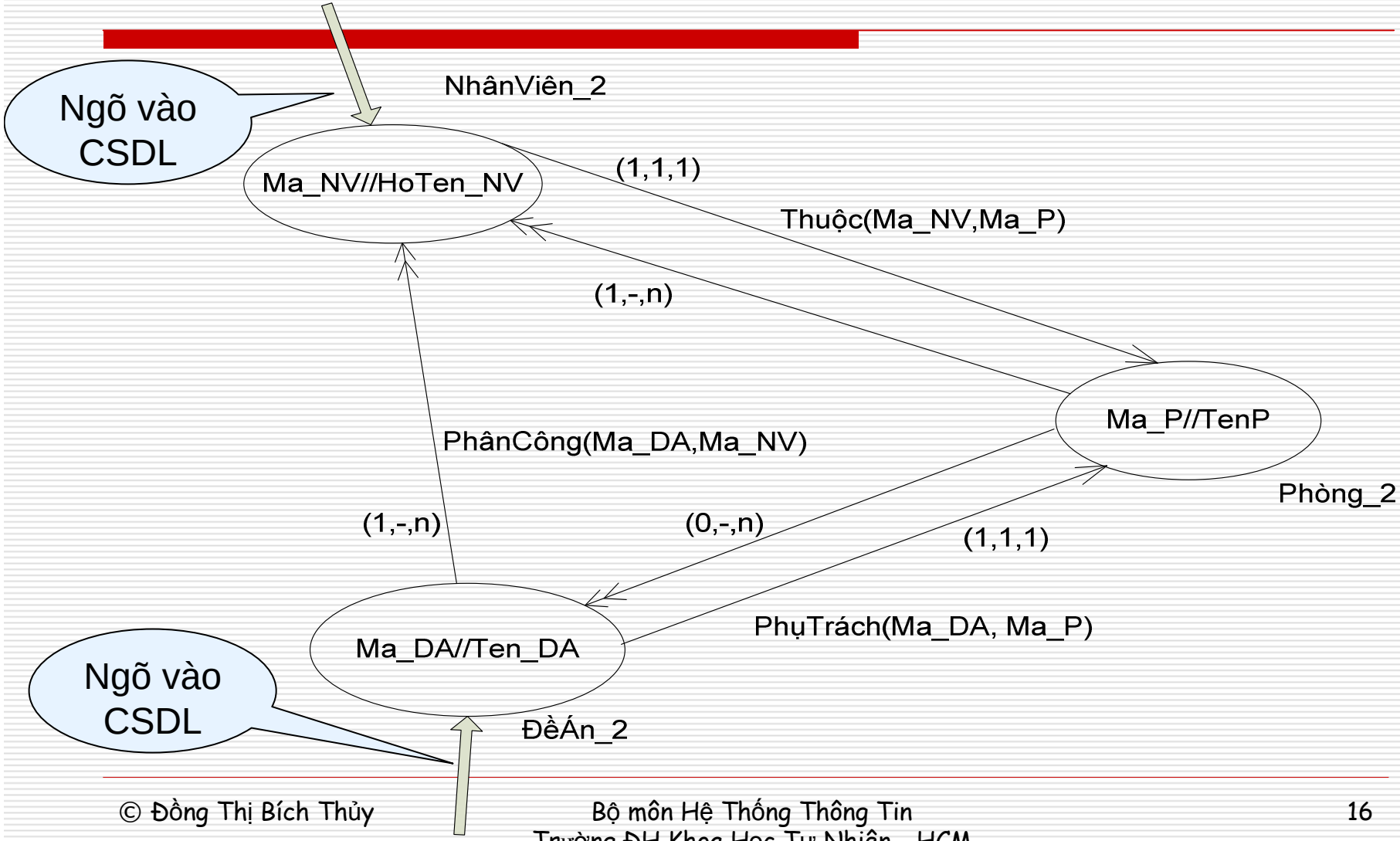
III.3 Đồ thị CĐTX – Định nghĩa (tt)

- **Mỗi cung trên đồ thị tương ứng với một con đường truy xuất đến 1 hoặc n bộ của quan hệ nút đến.**
- **Một quan hệ Qi$\in$ Q có thể là quan hệ nút (nếu nó tương ứng với một nút trên đồ thị) hoặc quan hệ cung (nếu nó ứng với một cung trên đồ thị).**
- **Mỗi quan hệ cung có thể tương ứng với tối đa hai cung ngược chiều nhau trên đồ thị CĐTX, nút đến của cung này là nút đi của cung kia và ngược lại.**

III.3 Đồ thị CĐTX – Định nghĩa (tt)

- **Ni $\xrightarrow{C_{ij}}$ Nj: Từ một bộ của quan hệ nút QNi có thể truy xuất từ 1 đến n bộ của quan hệ nút QNj thông qua con đường truy xuất tương ứng với cij.**
- **Ni $\xrightarrow{C_{ij}}$ Nj: Từ một bộ của quan hệ nút QNi có thể truy xuất đến 1 bộ của quan hệ nút QNj thông qua con đường truy xuất tương ứng với cij.**
- **Trên mỗi con đường truy xuất có gắn một bản số (n1, n2, n3) thể hiện số bộ tối thiểu, trung bình và tối đa có thể được truy xuất**

III.3 Đồ thị CĐTX – Ví dụ 3.3



Ví dụ 3.3 (tt)

- **Diễn giải:**
  - Có hai ngõ vào CSDL: NhânViên_2 và ĐềÁn_2, nghĩa là cung cấp một giá trị Ma_NV (Ma_DA), ta có thể truy xuất ngay một bộ tương ứng trong quan hệ NhânViên_2 (ĐềÁn_2).
  - Từ một bộ NhânViên_2, ta có thể truy xuất trực tiếp một bộ của Phòng_2 mà nhân viên trực thuộc, thông qua con đường truy xuất NhânViên_2 → Phòng_2.
  - Từ một bộ của Phòng_2, ta có thể truy xuất trực tiếp danh sách các nhân viên của phòng thông qua con đường truy xuất Phòng_2 → NhânViên_2.
  - Từ một bộ của NhânViên, ta không thể truy xuất trực tiếp danh sách các đề án mà nhân viên được phân công, do không có con đường truy xuất NhânViên_2 → ĐềÁn_2.

Đồ thị con đường truy xuất thô

- Là đồ thị con đường truy xuất đặc biệt, trong đó, nếu giữa hai nút của đồ thị có một cung thì bao giờ cũng tồn tại một cung theo chiều ngược lại.

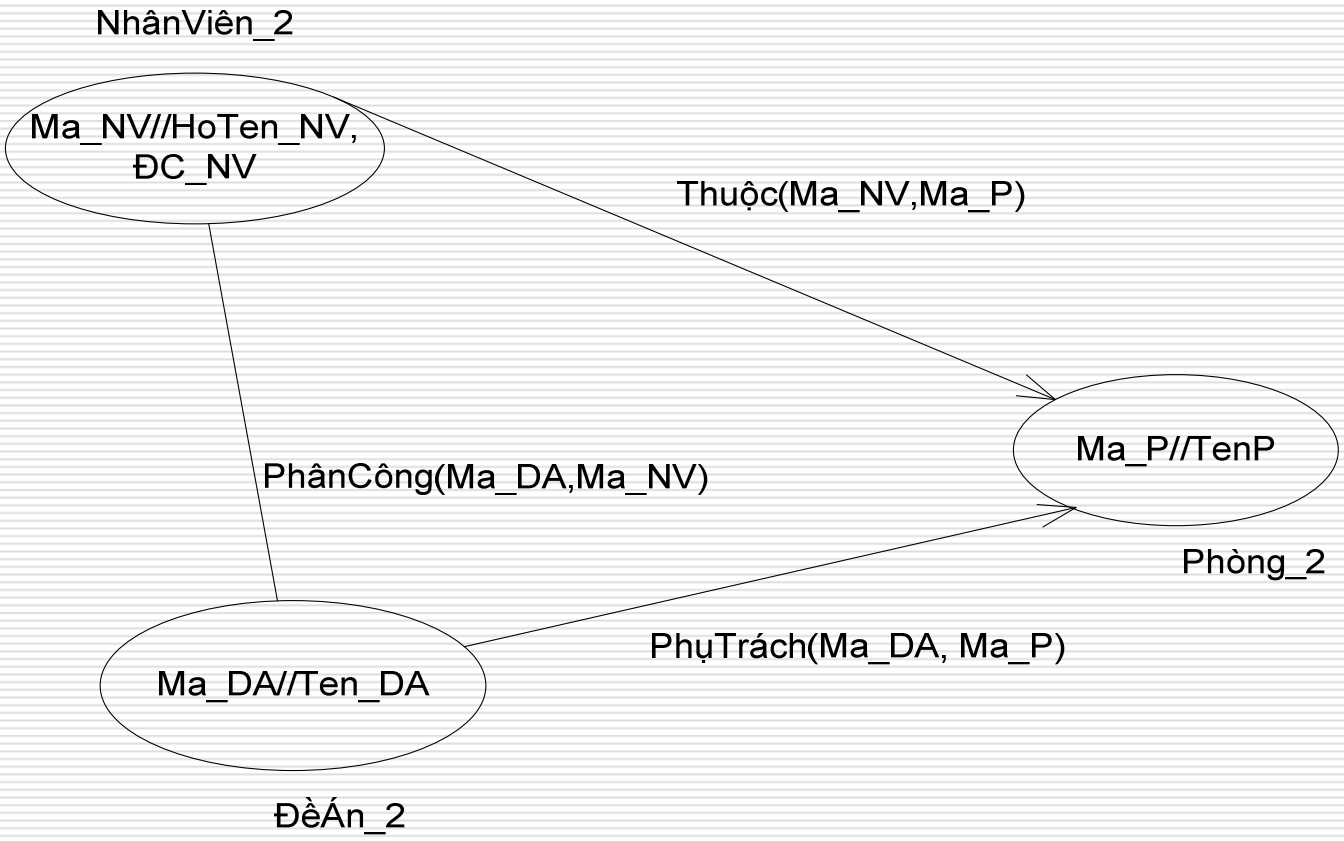
III.4 Đồ thị quan hệ

- Đồ thị quan hệ là một dạng đồ thị con đường truy xuất được đơn giản hoá
- Giúp người thiết kế dễ dàng hơn trong việc đánh giá chất lượng của biểu diễn cấu trúc CSDL ở dạng đồ thị.
- Đồ thị quan hệ là một đồ thị có hướng, với:
  - NQ: Tập nút
  - CQ $\in$ NQ x NQ: tập cung có hướng/vô hướng
  - QQ : tập quan hệ Qi

III.4 Đồ thị quan hệ (tt)

- **Diễn giải:**
  - **Ni $\xrightarrow{Q_{ij}}$ Nj:**
    - **Qi, Qj là các quan hệ lần lượt ứng với hai nút Ni và Nj**
    - **có một phụ thuộc hàm KQi $\rightarrow$ KQj, với KQi và KQj lần lượt là một khóa của Qi và Qj**
    - **Quan hệ cung Qij được hình thành từ tất cả các thuộc tính khóa của Qi, Qj: $Qji^+ = KQi^+ \cup KQj^+$**

III.4 Đồ thị quan hệ của ví dụ 3.3 (tt)



III.5 a.Biến đổi từ đồ thị CĐTX thô sang ĐTQH

- NQ = N
- QQ = Q
- $\forall$c,c' $\in$ C có chiều ngược nhau và cùng ứng với một quan hệ cung Qc, $\exists$ cQ $\in$ CQ sao cho cQ cũng ứng với quan hệ Qc trong QQ.
  - Cung cQ là cung vô hướng nếu bản số của c và c' đều có giá trị tối đa (max(c), max(c')) lớn hơn 1
  - Ngược lại cQ là cung có hướng

III.5 b.Chuyển từ ĐTQH sang đồ thị CĐTX thô

- N =NQ
- Q= QQ
- $\forall$cQ = (n_1, n_2) $\in$ CQ, (cQ ứng với một quan hệ Qc $\in$ QQ), $\exists$ c,c' $\in$ C có chiều ngược nhau và cùng ứng với một quan hệ cung Qc $\in$ Q.
  - Nếu cQ là cung vô hướng: max(c) >1 và max(c')>1
  - Ngược lại max(c) =<1 hoặc max(c') <= 1
- Các nút trong N đều là nút vào.

III.6 Chuỗi kết được cài đặt trên ĐTQH / đồ thị CĐTX

- **Khái niệm *chuỗi kết được cài đặt trên đồ thị quan hệ* là cơ sở để đánh giá tính hiệu quả của cấu trúc lô-gíc khi thực hiện phép kết.**

III.6 a. Chuỗi kết được cài đặt trên đồ thị CĐTX

- **Một chuỗi kết p = Q1 $\bowtie$ Q2… $\bowtie$ Qm được cài đặt trên đồ thị CĐTX (N,C,Q,Cđ) nếu và chỉ nếu:**
  - **$\forall$ Qi, i=1..m, Qi $\in$ Q**
  - **$\exists$ một dòng D = (c_1, c_2,… ,c_p) trên đồ thị CĐTX sao cho :**
    - **$\forall$ cung c_i của D, c_i ứng với một quan hệ Qj, trong chuỗi kết**
    - **$\forall$ Qi trong chuỗi kết : $\exists$ một cung c của D ứng với Qi**

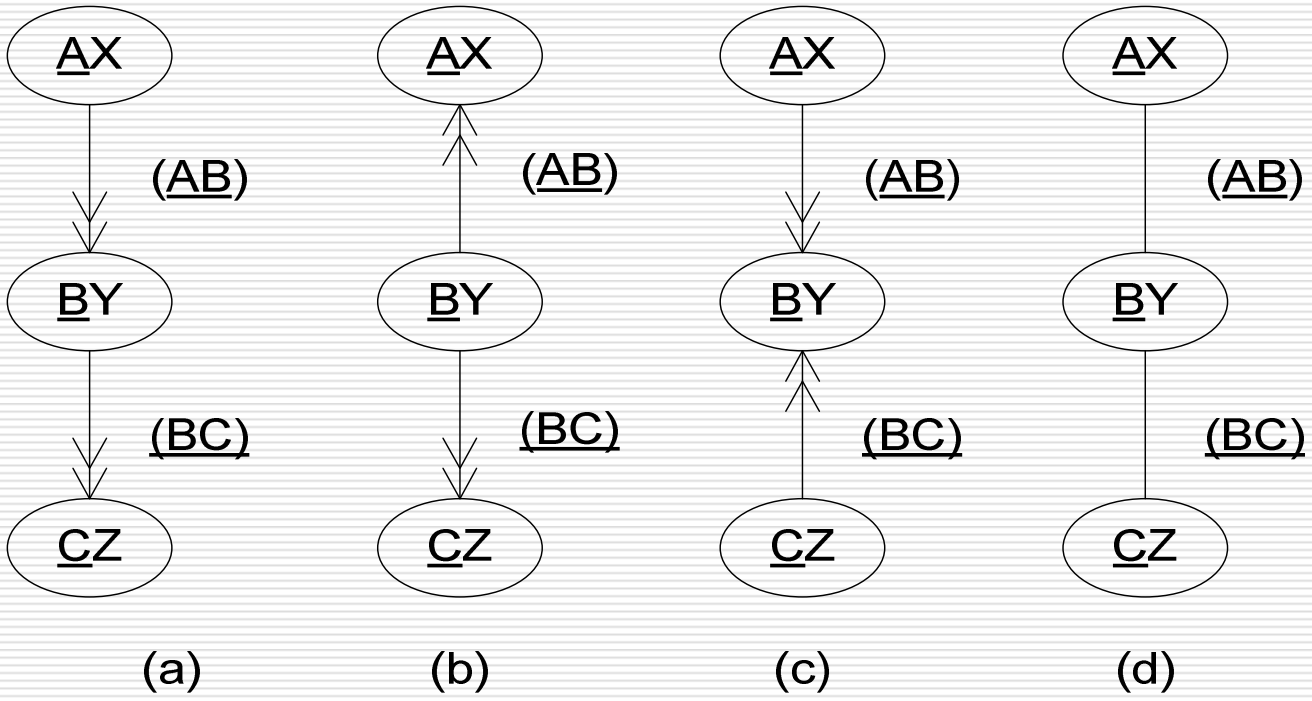
      Hoặc $\exists$ một nút n trên D ứng với Qi

III.6 b. Chuỗi kết được cài đặt trên ĐTQH

- **Một chuỗi kết p = Q1 $\bowtie$ Q2… $\bowtie$ Qm được cài đặt trên ĐTQH (NQ,CQ,QQ) nếu và chỉ nếu:**
  - **$\forall$ Qi, i=1..m, Qi $\in$ QQ**
  - **$\exists$ một dòng D = (c_1, c_2,… ,c_p) trên đồ thị quan hệ sao cho :**
    - **$\forall$ cung c_i của D, c_i ứng với một quan hệ Qj, trong chuỗi kết**
    - **$\forall$ Qi trong chuỗi kết: $\exists$ một cung c của D ứng với Qi**

      Hoặc $\exists$ một nút n trên D ứng với Qi

Chuỗi kết trên đồ thị - Ví dụ 3.4



Chuỗi kết (AX) ⋈ (AB) ⋈ (BY) ⋈ (BC) ⋈ (CZ) được cài đặt trên (a), (b), (d) nhưng không được cài đặt trên (c)

Chuỗi kết trên đồ thị - Diễn giải

- Nếu chuỗi kết ρ được cài đặt trên đồ thị, tồn tại một dòng D có gốc là n_g.
  - Từ quan hệ Qg ứng với n_g ta có thể truy xuất <u>nhanh</u> đến những bộ của Qi trong ρ thông qua các đường đi hoặc mạch đi xuất phát từ n_g

Câu hỏi?

Buổi # 8
(lớp TH2008-1)

Cơ sở dữ liệu nâng cao

Đồng Thị Bích thủy

26/10/2010

Nội dung tuần này

III.6 Chuỗi kết được cài đặt trên đồ thị (nhắc lại)

III.7 Thuật toán biến đổi cấu trúc CSDLQH

III.8 Chuyển từ đồ thị quan hệ sang cấu trúc CSDL quan hệ

III.6 Chuỗi kết được cài đặt trên ĐTQH / đồ thị CĐTX (nhắc lại)

- **Khái niệm *chuỗi kết được cài đặt trên đồ thị quan hệ* là cơ sở để đánh giá tính hiệu quả của cấu trúc lô-gíc khi thực hiện phép kết.**

Lấy lại Ví dụ 3.1

Cho cấu trúc mức quan niệm:

NhânViên (Ma_NV, HoTen_NV, Ma_P)

NhânViên_2 (Ma_NV, HoTen_NV)

Thuộc (Ma_NV, Ma_P)

Phòng (Ma_P, Ten_P)

ĐềÁn (Ma_DA, Ten_DA, Ma_P)

ĐềÁn_2 (Ma_DA, Ten_DA)

PhụTrách (Ma_DA, Ma_P)

PhânCông (Ma_NV, Ma_DA),

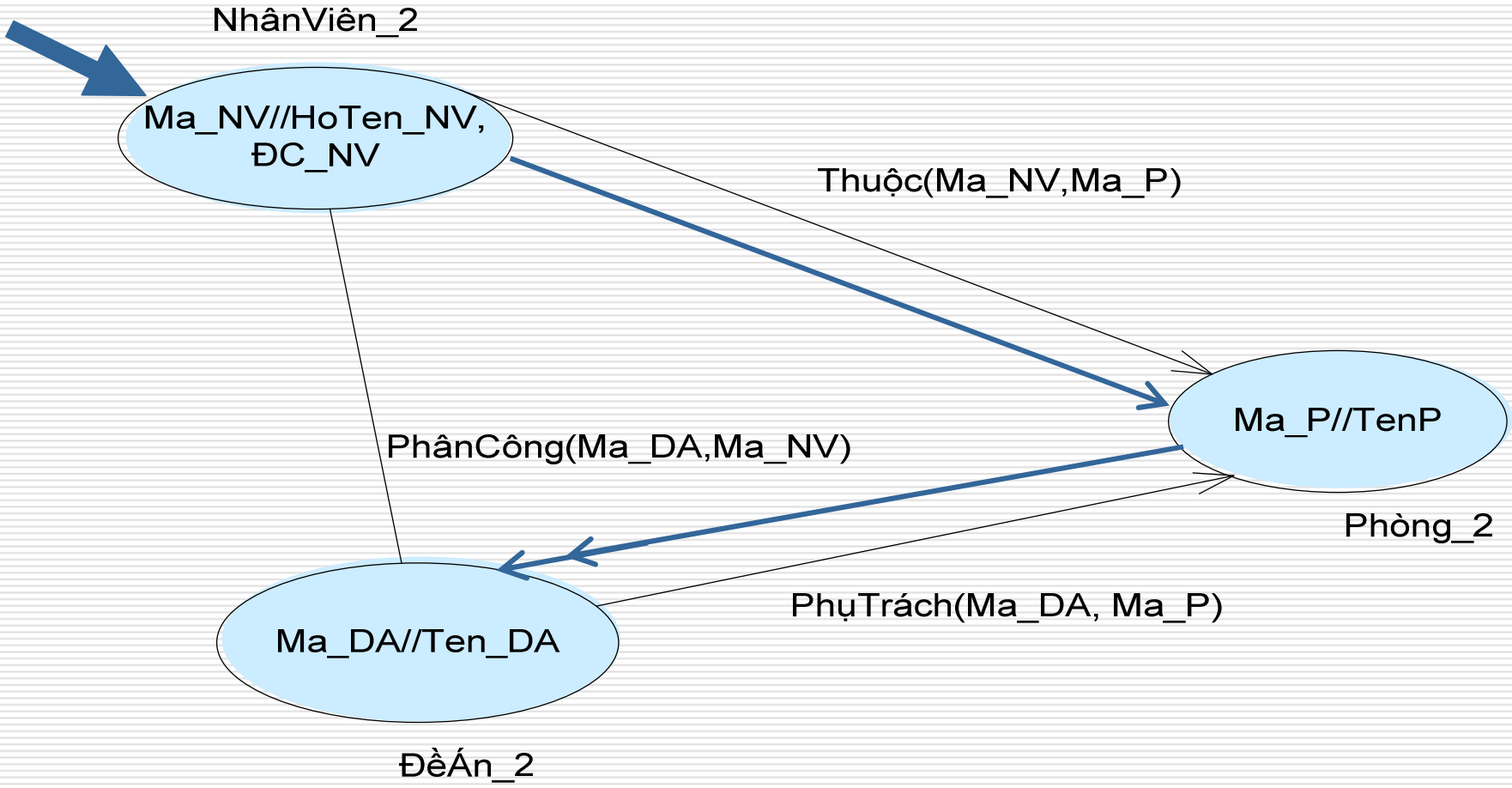
Chuỗi kết thể hiện câu truy vấn

- **Tên các đề án do phòng của nhân viên NVA phụ trách:**

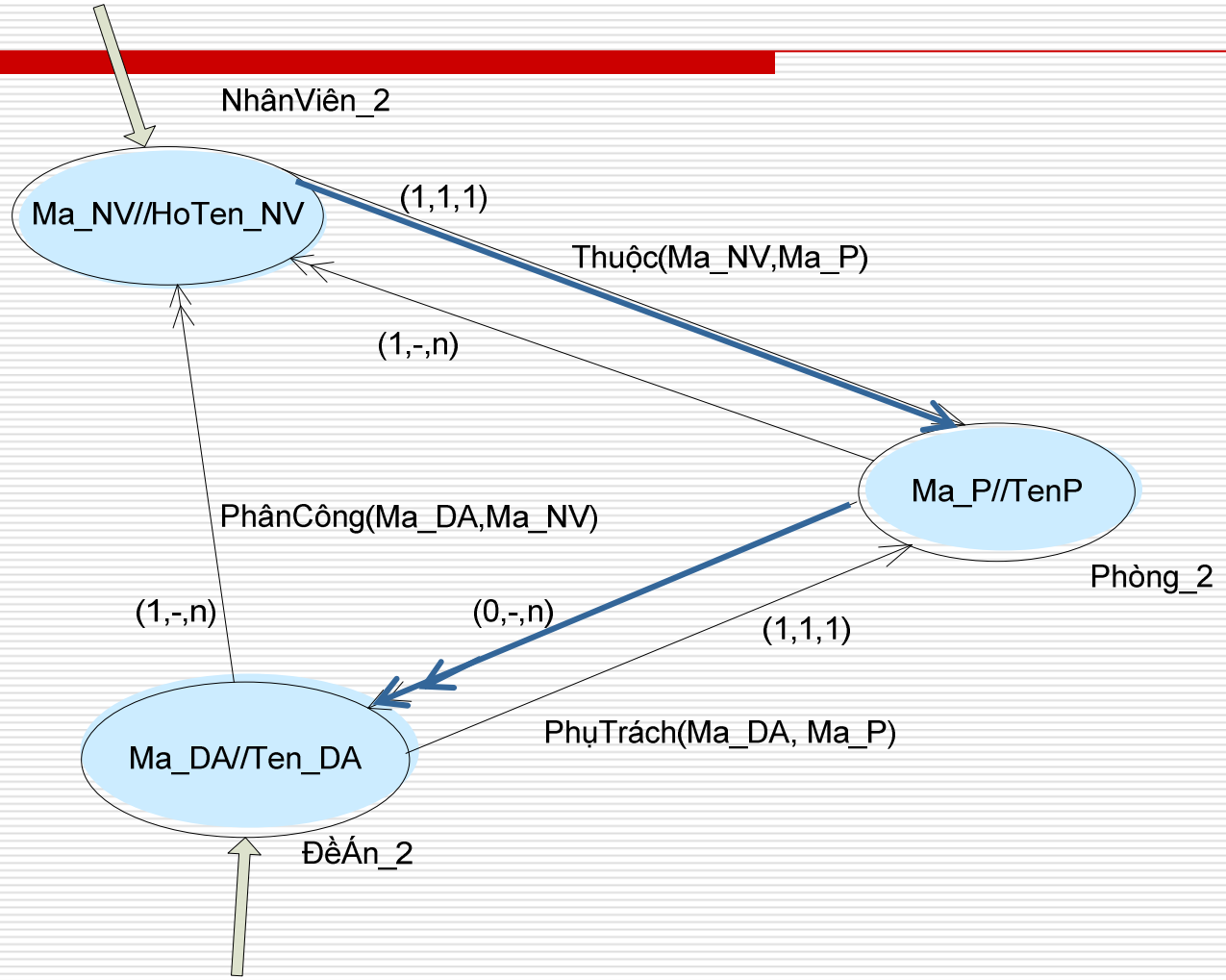
$$\pi_{\text{Ten_DA}}\, \sigma_{\text{HoTen_NV = "NVA"}}$$

$$(\text{NhânViên_2} \bowtie \text{Thuộc} \bowtie \text{Phòng} \bowtie \text{PhụTrách} \bowtie \text{ĐềÁn_2})$$

Đồ thị quan hệ của ví dụ 3.1



Đồ thị CĐTX của ví dụ 3.1



III.7 Thuật toán biến đổi lược đồ CSDLQH

- **Vào: Cấu trúc CSDL mức quan niệm :**

  ρ = {<Qi>}, mỗi Qi có tập khóa {Ki}

- **Ra: Đồ thị quan hệ tương ứng với ρ**

III.7 Thuật toán (2)

- **Bước 1 - Biến ρ thành một phân rã đồng nhất:**

  1.1. ∀ **(Qi, Qj):**

  nếu Ki ↔Kj, với Ki và Kj lần lượt là một khóa của Qi và Qj,

  thì gộp Qi, Qj lại thành một quan hệ.

  1.2. ∀**(Qi, Qj):**

  nếu Qi^+ có chứa một khóa Kj của Qj

  thì Qi^+ phải chứa tất cả các khoá của Qj

  (khi đó Qi có thể có thêm khoá)

- **Bước 2 - Tạo nút và quan hệ nút:**

  ∀ **Qi: tạo một nút Ni với QNi = Qi**

III.7 Thuật toán (3)

- **Bước 3 - Tạo nút bản lề và quan hệ nút tương ứng:**

****** Mục đích: làm nổi bật các thuộc tính chung của mỗi cặp quan hệ nút ******

3.1. $\forall$ (Qi, Qj), $Qij^+ = Qi^+ \cap Qj^+$

3.2. Chừng nào $Qij^+ \neq \varnothing$:

- **Xác định tất cả các khóa củaQ[Qij^+], ký hiệu $KQij^+$ là tập thuộc tính khóa của Q[Qij^+]**
- **Nếu $\neg\exists$ Qh $\in \rho$ sao cho một khóa của Qh là một khoá của Q[Qij+]**

  thì Tạo nút bản lề Nbl với quan hệ tương ứng Qbl = Q[$KQij^+$]

  Cuối nếu
- **$Qij^+ := Qij^+ - KQij^+$**

Cuối Chừng nào

III.7 Thuật toán (4)

☐ Bước 4 - Tạo cung

****** chỉ tạo số cung tối thiểu từ một nút ******

4.1. ∀Ni với Qi tương ứng, xác định:

PTH(Ni) = {Nj với Qj tương ứng sc $KQj^+ \subset Qi^+$}

PTH_Thừa(Ni) = {Nj ∈ PTH(Ni) |

∃Nh ∈ PTH(Ni) với $KQj^+ \subset KQh^+$}

Lồng_Khoá(Ni) = {Nj với Qj tương ứng | $KQj^+ \subset KQi^+$}

Lồng_Khoá_Thừa (Ni) =

{Nj ∈ Lồng_Khoá(Ni) | ∃Nh ∈ Lồng_Khoá(Ni) với Qh

sao cho $KQj^+ \subset KQh^+$}

Cung (Ni) = (PTH(Ni) – PTH_Thừa(Ni)) ∪

(Lồng_Khoá(Ni) – Lồng_Khoá_Thừa(Ni))

Cuối ∀

III.7 Thuật toán (5)

- **Bước 4 (tiếp)**

4.2. $\forall Nj \in$ Cung(Ni) :

Tạo một cung có hướng từ Ni $\rightarrow$ Nj,

ký hiệu cij

Cuối $\forall$

4.3. $Qij = Qi[KQi^{+} \cup KQj^{+}]$

III.7 Thuật toán (6)

- **Bước 5 - Hủy những nút bản lề thừa**
  - **∀Nk thỏa các điều kiện:**
    - **Qk có một khóa duy nhất Kk**
    - **Không có thuộc tính nào khác ngoài khóa**
    - **Chỉ có một cung vào Nk, xuất phát từ nút Ni**

    Thì ***Vai trò bản lề của Nk không còn cần thiết nữa
    - **Nhập Nk vào Ni (nhập Qk^+ vào Qi^+)**
    - **Hủy cung cik**
  - **Cuối ∀**

III.7 Thuật toán (7)

- **Bước 6 - Mịn hóa các quan hệ nút:**
  - **∀ Ni với Qi tương ứng thì:**
    - **∀ Nj ∈ Cung(Ni) với Qj tương ứng thì:**
      - **Loại khỏi Qi^+ những thuộc tính khóa của Qj mà không phải là thuộc tính khóa của Qi**
    - **Cuối ∀**
  - **Cuối ∀**

III.7 Thuật toán (8)

- **Bước 7 - Tạo cung vô hướng:**

  ∀ Nk thỏa:

  - **Qk chỉ có toàn thuộc tính khóa ($Qk^+ = KQk^+$)**
  - **Chỉ có hai cung ra khỏi Nk (không có cung vào) đến Ni và Nj với Qi và Qj sao cho $KQk^+ = KQi^+ \cup KQj^+$**

  Thì

  - **Tạo một cung vô hướng nối Ni, Nj với Qij = Qk**
  - **Hủy nút Nk**
  - **Hủy hai cung cki và ckj**

  Cuối ∀

Ví dụ 3.5

Cho cấu trúc quan niệm sau:

ĐĐH (Số_ĐĐH, Ngày_ĐH, TrịGiá)

MặtHàng (Mã_MH, Tên_MH, ĐơnGiá)

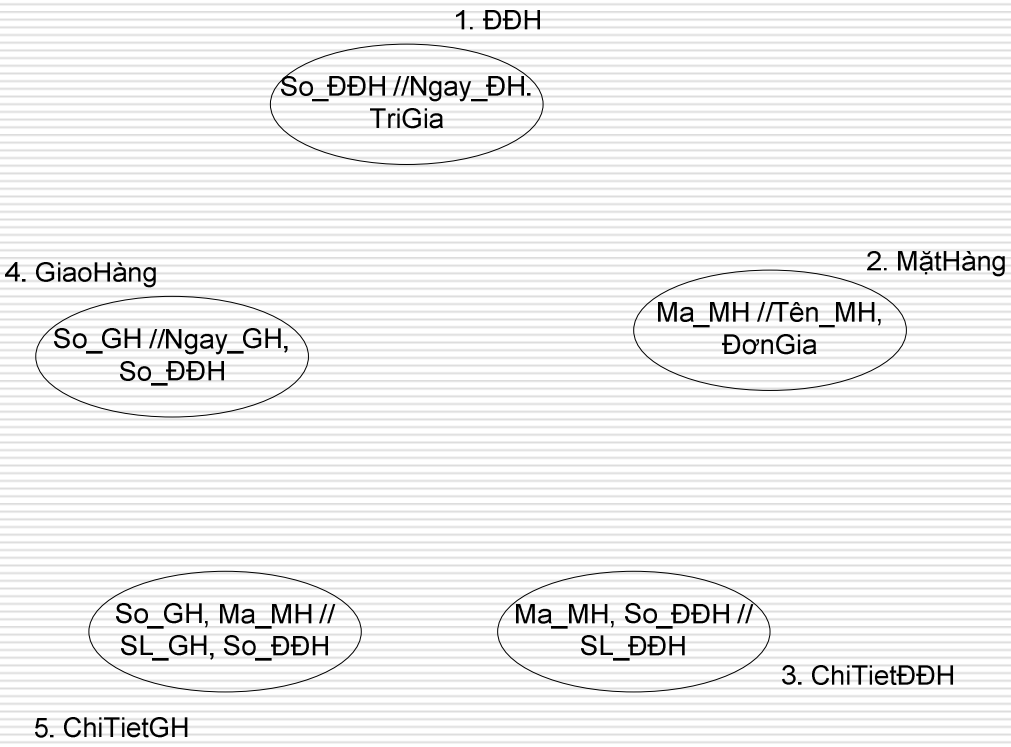
ChiTiếtĐĐH (Mã_MH, Số_ĐĐH, SL_ĐH)

Giao hàng (Số_GH, Ngày_GH, Số_ĐĐH)

ChiTiếtGH (Số_GH, Mã_MH, SL_GH, Số_ĐĐH)

Ví dụ 3.5 (2)

- **Bước 1: không có khóa tương đương giữa các quan hệ**
- **Bước 2: Tạo nút**



Ví dụ 3.5 (3)

- **Bước 3:**
- **Các tập thuộc tính chung khác rỗng của các cặp quan hệ:**
  - **(1 và 3), (1 và 4), (1 và 5): So_ĐĐH, khoá của 1**
  - **(2 và 3): Ma_MH, khoá của 2**
  - **(2 và 5): Ma_MH, khoá của 2**
  - **(3 và 5): Ma_MH, So_ĐĐH → Khóa = (Ma_MH, So_ĐĐH) = khóa của 3**
  - **(4 và 5): So_GH, So_ĐĐH → Khóa = (So_GH) = Khoá của 4; loại bỏ So_GH;**

    khóa của tập còn lại là (So_ĐĐH) = khóa của 1
- **→Kết luận: Không tạo nút bản lề nào cả**

Ví dụ 3.5 (4)

- **Bước 4**

Ghi chú "-" : không cần tính

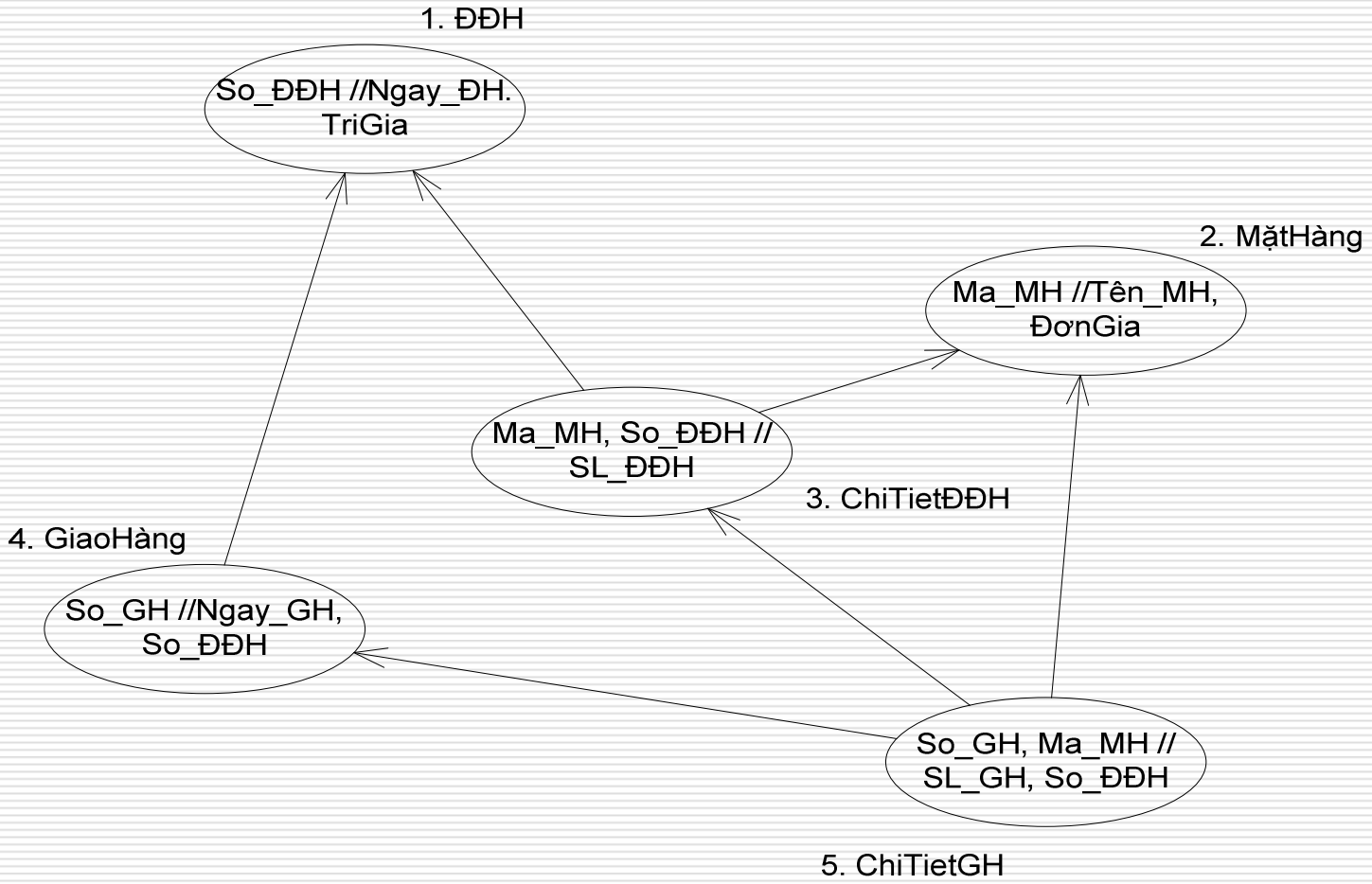
	PTH(Q_i)	PTH_ Thừa(Q_i)	Lồng_ Khoá(Q_i)	LK_Thừa (Q_i)	Cung(Q_i)
1. ĐĐH	∅	-	-	-	∅
2.MặtHàng	∅	-	-	-	∅
3. ChiTiếtĐĐH	1,2	∅	1,2	∅	1,2
4. GiaoHàng	1	∅	∅	-	∅
5.ChiTiếtGH	1,2,3,4	1,2	2,4	∅	2,3,4

Ví dụ 3.5 (5)

- **Các quan hệ cung:**
  - **Cung 31: CTĐĐH_ĐĐH (Ma_MH, So_ĐĐH)**
  - **Cung 32: CTĐĐH_MH(Ma_MH, So_ĐĐH)**
  - **Cung 41: GH_ĐĐH(So_GH, So_ĐĐH)**
  - **Cung 52: CTGH_MH (So_GH, Ma_MH)**
  - **Cung 53: CTGH_CTĐĐH (So_GH, Ma_H, So_ĐĐH)**
  - **Cung 54: CTGH_GH (So_GH, Ma_MH)**

Ví dụ 3.5 (6)

- **Kết quả của bước 1-4**

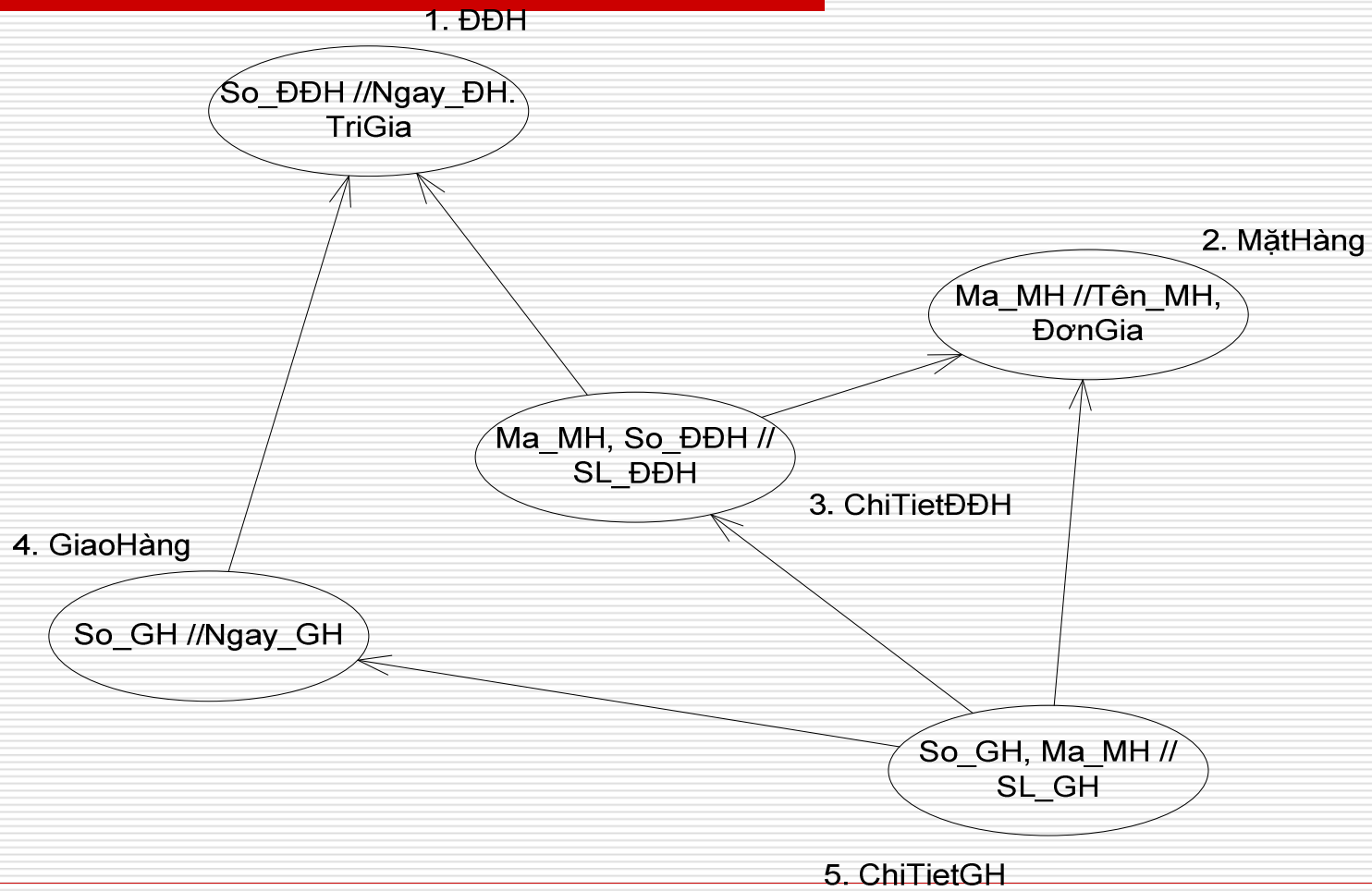


Ví dụ 3.5 (7)

- **Bước 5: Không thực hiện, vì không tạo nút bản lề nào**
- **Bước 6:**
  - **Trong quan hệ nút GiaoHàng, loại bỏ thuộc tính Số_ĐĐH**
  - **Trong quan hệ nút ChiTiếtGH, loại bỏ thuộc tính Số_ĐĐH**
- **Bước 7: không tạo được cung vô hướng nào cả.**

Ví dụ 3.5 (8)

- **Kết quả thuật toán:**



III.8 Biến đổi từ đồ thị quan hệ sang lược đồ CSDL quan hệ

- **Mục đích:** kiểm chứng xem cấu trúc quan hệ biểu diễn dưới dạng đồ thị quan hệ có hoàn toàn tương đương với cấu trúc ban đầu hay không.
- **Thuật toán:**
  - **Gọi ρ^{-1} là tập quan hệ con có được sau khi biến đổi từ đồ thị quan hệ về cấu trúc CSDL quan hệ:**
  - **$\rho^{-1} = \{Qi\} \cup \{Qij\}$, với Qi là quan hệ nút và Qij là quan hệ cung**
  - **Gộp các quan hệ có cùng khóa trong ρ^{-1} lại thành một.**

Câu hỏi?

Buổi # 9
(lớp TH2008-1)

Cơ sở dữ liệu nâng cao

Đồng Thị Bích thủy

09/11/2010

Nội dung tuần này

III.9 Trường hợp đáng chú ý

III.10 Cách chọn lựa và khai báo đồ thị CĐTX với hệ QTCSDL

- III.10.1 Đặt vấn đề
- III.10.2 Cài đặt cung (con đường truy xuất)
- III.10.3 Cài đặt nút (bảng)

III.9 Trường hợp đáng chú ý

Cho cấu trúc quan hệ phổ quát:

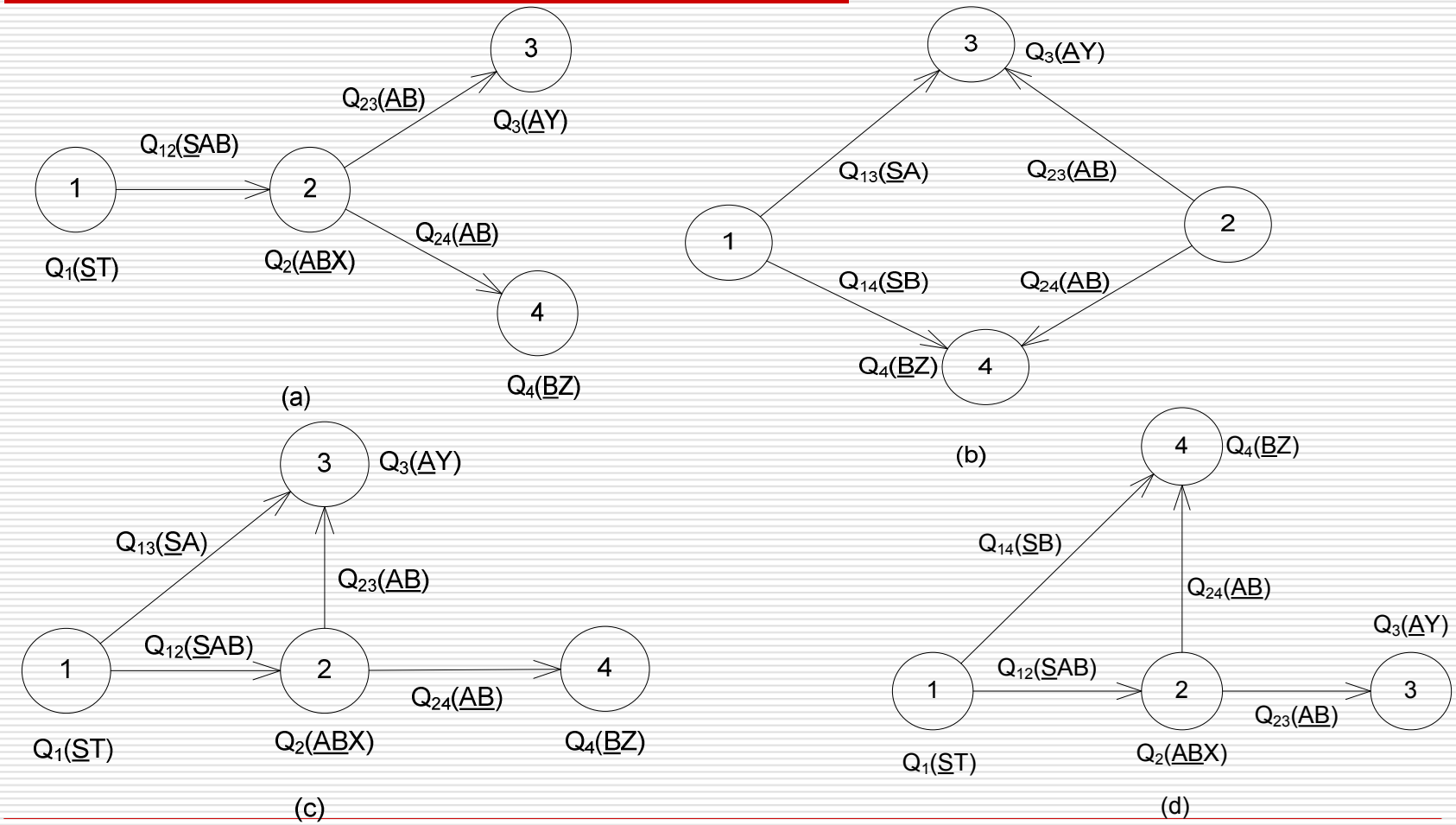
$\underline{C_0}$ = <Q0(SABXYZT), F_0={S $\rightarrow$ABT; AB $\rightarrow$X;
A $\rightarrow$Y; B $\rightarrow$Z }>

Và cấu trúc CSDL quan niệm:

$\underline{C_1}$ = {<Q_1(SABT), F_1 = {S $\rightarrow$ABT}>;
<Q_2(ABX) , F_2 = {AB $\rightarrow$X}> ;
<Q_3(AY) , F_3 = {A $\rightarrow$Y}>;
<Q_4(BZ) , F_4 ={B $\rightarrow$Z}> }

III.9 Trường hợp đáng chú ý (2)

Có thể biểu diễn $\underline{C_1}$ ở dạng đồ thị theo 4 cách sau:



III.9 Trường hợp đáng chú ý (3)

- **Nhận xét:**
  - **Các đồ thị tương đương nhau ở góc độ thao tác quan hệ: các chuỗi kết dựa trên thuộc tính chung có thể đặt ra trên cấu trúc CSDL này đều được cài đặt trên cả 4 đồ thị:**
  - **Ví dụ: muốn truy xuất π_{STAY} Q_0**
    - **Đồ thị (a) và (d) : dòng (12, 23)**
    - **Đồ thị (b) : dòng (13)**
    - **Đồ thị (c): dòng (13) hoặc (12, 23)**

III.9 Trường hợp đáng chú ý (4)

Nhận xét (2)

- Tuy nhiên, cách truy xuất sẽ khác nhau trên các đồ thị
- Ví dụ: Truy xuất $\pi_{AB}\ Q_0$:
  - Đồ thị (a), (c), (d): truy xuất **chỉ ở nút 2** : $\pi_{AB}\ Q_2$
  - Đồ thị (b): **dòng (13, 14) và nút 2**: $\pi_{AB}\ (Q_{13} \bowtie Q_{14}) \cup \pi_{AB}\ Q_2$
- Một thể hiện minh họa cho đồ thị (b):

TQ_1

S	A	B	T
s_1	a_1	b_3	t_1
s_2	a_3	b_2	t_2

TQ_2

A	B	X
a_1	b_1	x_1
a_1	b_3	x_2

TQ_3

A	Y
a_1	y_1
a_2	y_2
a_3	y_3

TQ_4

B	Z
b_1	z_1
b_2	z_2
b_3	z_3

III.9 Trường hợp đáng chú ý (5)

- **Nhận xét (3)**
  - **Các đồ thị cũng có ý nghĩa khai thác khác nhau:**
    - **Với đồ thị (c):** người sử dụng có thể tạo các bộ Q_0[SA] độc lập với những bộ Q_0[SAB], thông qua cung (13) , tức là có thể nhập trước những bộ dạng (s,a,-,t)
    - **Tương tự, với đồ thị (d),** người sử dụng có thể tạo các bộ Q_0[SB] độc lập với những bộ Q_0[SAB], thông qua cung (14)
    - Trong đồ thị (a): không có khả năng này.
  - **Khi cài đặt quan hệ Q_1 thành một bảng TQ_1 , sẽ phải khai báo khác nhau (xem phần III.10)**

III.9 Trường hợp đáng chú ý (6)

- **Nhận xét (4)**
  - **Thuật toán chuyển từ cấu trúc CSDL sang đồ thị quan hệ đã trình bày đề nghị đồ thị (a):**
    - **Mục tiêu là cung cấp một biểu diễn đồ thị không trùng lắp trên con đường truy xuất để người thiết kế có cái nhìn gọn nhất**
    - **Các biến thể (đồ thị (b), (c), (d)) sẽ được cân nhắc ở giai đoạn sau với các tiêu chí : ý đồ khai thác, tần suất khai thác,…**

III.10

Cách chọn lựa và khai báo đồ thị CĐTX với hệ QTCSDL

III.10.1 Đặt vấn đề

- Các cơ sở để cân nhắc khi thiết kế vật lý:
  - Ý đồ khai thác CSDL
  - Đặc trưng kỹ thuật của hệ quản trị CSDL

III.10.1 Đặt vấn đề (2)

- Ý đồ khai thác CSDL: là các chuỗi kết thể hiện nội dung cốt lõi của các xử lý
  - Cung (con đường truy xuất):
    - Nên giữ lại con đường truy xuất nào?
    - Có nên thêm vào con đường truy xuất trùng lặp?
  - Nút (bảng = quan hệ):
    - Gộp các nút - giữ nguyên - tách một nút ?
    - Đảo nút?

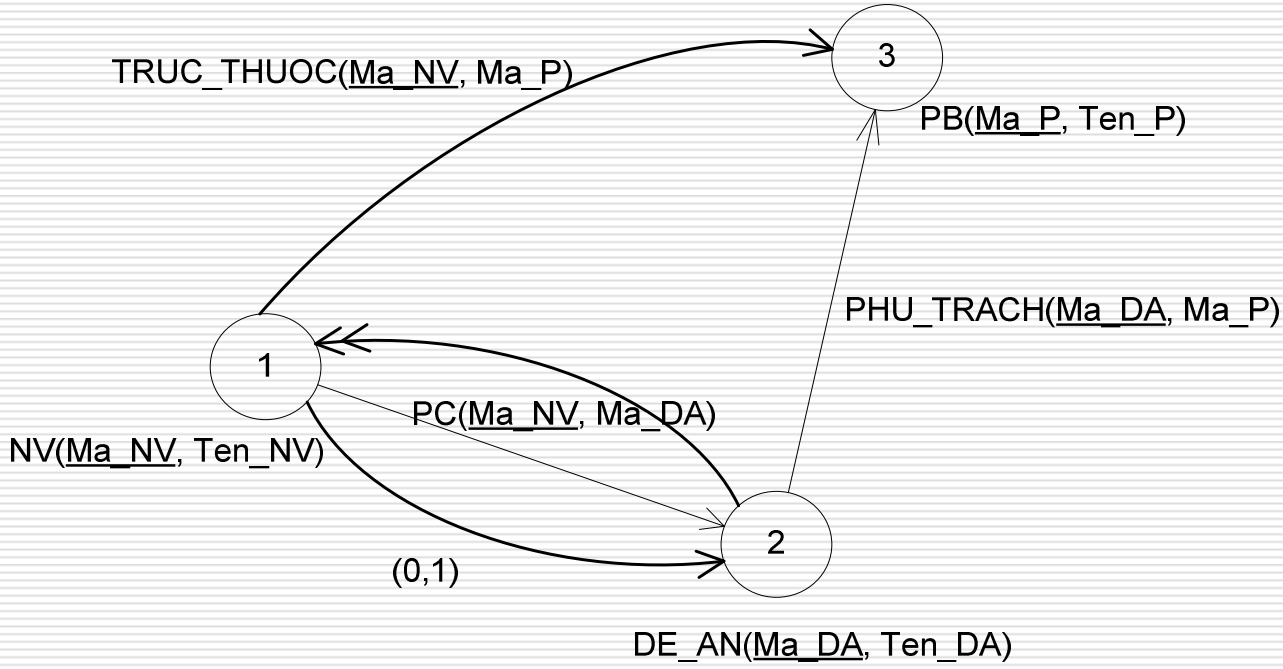
  → Kết quả là đồ thị con đường truy xuất được cài đặt

III.10.1 Đặt vấn đề (3)

- **Đặc trưng kỹ thuật của hệ quản trị CSDL chọn để cài đặt:**
  - **Cài đặt trên Hệ quản trị nào, đồ thị con đường truy xuất cài đặt có phù hợp với hệ quản trị này hay không?**
  - **Lựa chọn các hỗ trợ của hệ quản trị để thể hiện các khái niệm lý thuyết trên đồ thị con đường truy xuất**

III.10.2 Các vấn đề liên quan đến cài đặt cung (CĐTX)

- **Đưa trở vào một con đường truy xuất trùng lặp:**



III.10.2 Các vấn đề liên quan đến cài đặt cung (CĐTX) (2)

Ta có: **Ma_NV → Ma_DA** **(12)**

Ma_DA → Ma_P **(23)**

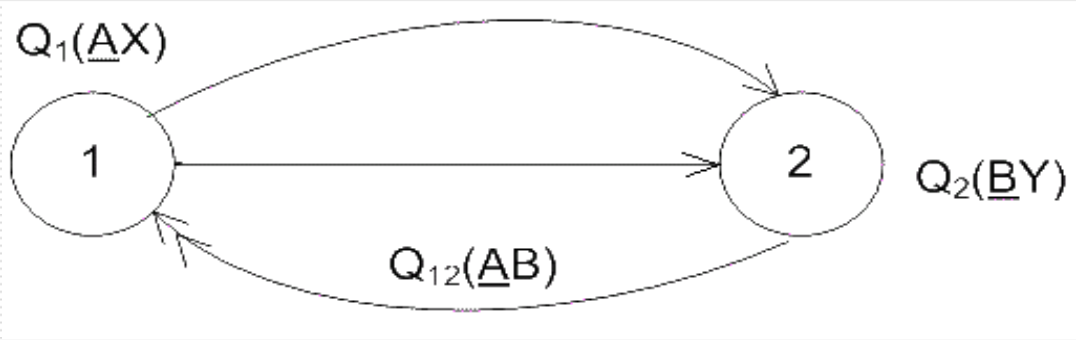
==> Ma_NV → Ma_P **(13)**

Đưa vào con đường truy xuất trùng lắp (13) vì những lý do sau:

- **Bản số cung (12) là (0,1). Cung (13) cho biết được phòng của NV dù nhân viên đó không tham gia vào đề án nào. Khi cần danh sách nhân viên của một phòng, dùng con đường (13), vì dùng (12, 23) sẽ thiếu các nhân viên không tham gia đề án.**
- **Nếu bản số cung (12) là (1,1), việc đưa vào cung (13) có ý nghĩa khi nhân viên có thể tham gia vào đề án của phòng ban khác phòng ban mà nhân viên trực thuộc.**

III.10.2 Các vấn đề liên quan đến con đường truy xuất (3)

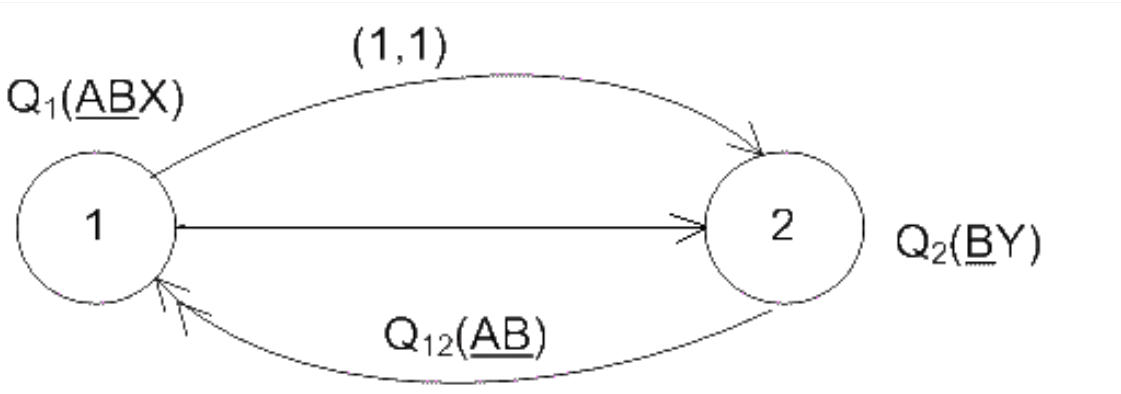
- **Khai báo vật lý**
  - **Trường hợp 1:**



  - **Khai báo vật lý:**
    - **Q2(BY), B là khóa chính**
    - **Q1(AXB), A là khóa chính, B là khoá ngoại, chỉ mục (B)**

III.10.2 Các vấn đề liên quan đến con đường truy xuất (4)

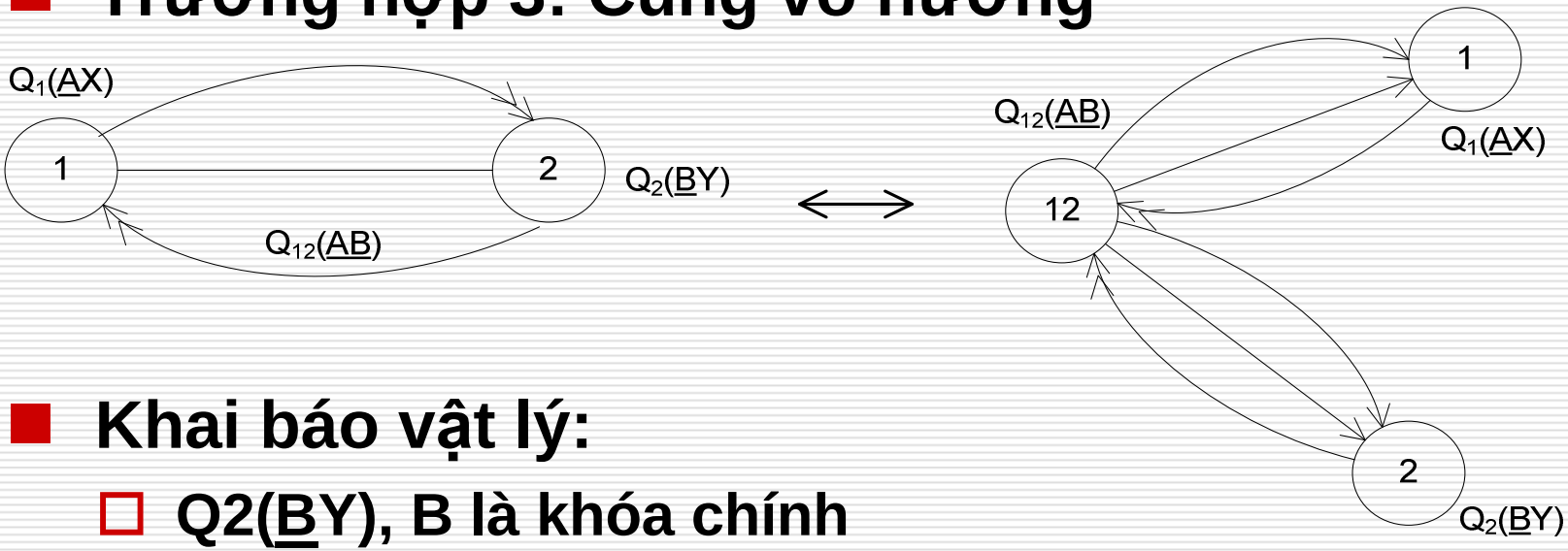
- **Trường hợp 2:**



- **Khai báo vật lý:**
  - **Q2(BY), B là khóa chính**
  - **Q1(ABX), AB là khóa chính, B là khoá ngoại, chỉ mục (B)**

III.10.2 Các vấn đề liên quan đến con đường truy xuất (5)

- **Trường hợp 3: Cung vô hướng**



- **Khai báo vật lý:**
  - **Q2(BY), B là khóa chính**
  - **Q1(AX), A là khóa chính**
  - **Q12(AB), (AB) là khóa chính, hai khóa ngoại: (A) và (B), hai chỉ mục (A) và (B)**

III.10.3 Các vấn đề liên quan đến cài đặt một nút (=bảng)

- **Đảo một quan hệ nút – Chọn chỉ mục cho một nút**
  - **Vd 3.7: NV(Mã_NV, Tên_NV,Mã_P)**
  - **Có nhu cầu truy xuất các bộ của nhân viên dựa trên Mã_P → Khai báo Mã_P là chỉ mục của quan hệ NV.**

  → Đảo quan hệ NV theo Mã_P

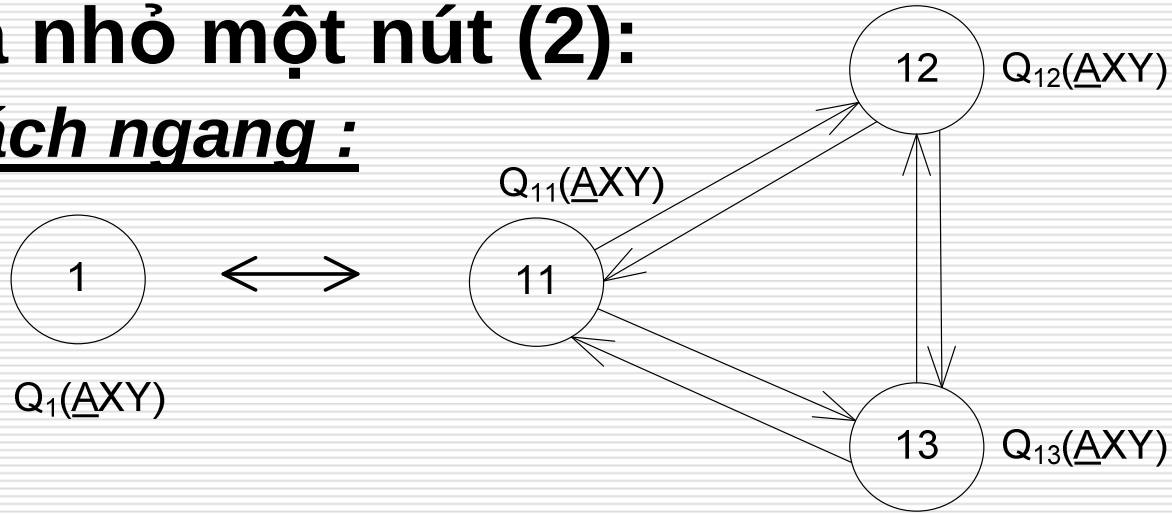
III.10.3 Các vấn đề liên quan đến cài đặt một nút (2)

- **Chia nhỏ một nút:**
  - **Để nguyên hay tách một nút?**
  - ***Tách dọc:***

    Xét quan hệ Q(AXY), trong đó Y có thể là một tập thuộc tính lớn, và trong quá trình khai thác ta thường xuyên quan tâm tới AX, ít nhắc tới Y
    - **tách Q thành hai nút Q1(AX) và Q2(AY)**

III.10.3 Các vấn đề liên quan đến cài đặt một nút (3)

- **Chia nhỏ một nút (2):**
  - ***<u>Tách ngang :</u>***



  - **<u>Ví dụ</u>: Sổ thu chi trong năm từ 1/1 đến 31/12 có dữ liệu rất lớn → chia ra thành 12 quan hệ con, mỗi quan hệ chứa thông tin của từng kỳ (tháng)**

III.10.3 Các vấn đề liên quan đến cài đặt một nút (4)

- **Gộp nút – Đưa thêm thuộc tính vào một quan hệ nút**

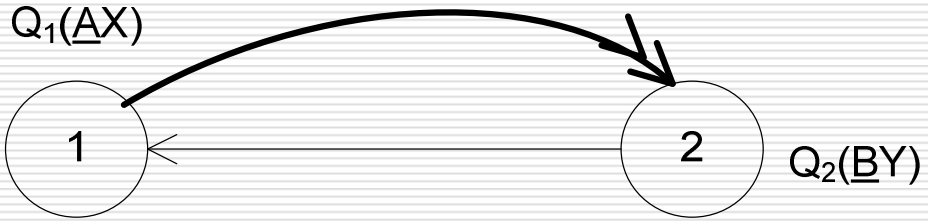
  Trường hợp 1:

  - **Từ một bộ của Q1, rất thường xuyên có nhu cầu truy xuất các bộ của Q2 tương ứng. Do đó gộp hai quan hệ này lại thành một để truy xuất cho nhanh. Khi đó khai báo vật lý:**
  - **Q12 (ABXY), khoá chính (AB), chỉ mục (A)**
  - **→ Q12 chỉ đạt DC1, ta chấp nhận trùng lắp thông tin để truy xuất nhanh**

III.10.3 Các vấn đề liên quan đến cài đặt một nút (5)

- **Gộp nút (2)**

  <u>Trường hợp 2</u>:



  - **Từ một bộ của Q1, *rất thường xuyên* có nhu cầu truy xuất các bộ của Q2 tương ứng. Do đó gộp hai quan hệ này lại thành một để truy xuất cho nhanh. Khi đó khai báo vật lý:**
  - **Q12 (<u>AB</u>XY), khoá chính (B), chỉ mục (A)**
  - **→ Q12 chỉ đạt DC2, tương tự ví dụ 1, ta chấp nhận trùng lắp thông tin để truy xuất nhanh**

III.10.3 Các vấn đề liên quan đến cài đặt một nút (5)

- **Cài đặt nút vào:**
  - **Cài đặt nút vào cho phép ta truy xuất nhanh đến một bộ của quan hệ từ giá trị khóa mà không cần duyệt tuần tự qua các bộ trong thể hiện.**
  - **Tùy vào hỗ trợ của hệ quản trị, ta chọn cách cài đặt nút vào:**
    - **Cài đặt bằng một chỉ mục có giá trị duy nhất , xem chỉ mục đặc biệt này là khóa của nút**
    - **Cài đặt bằng hàm băm,…**
    - **Hiện nay đa số các hệ quản trị CSDL đều có hỗ trợ cơ chế khóa chính.**

Ví dụ của trường hợp đáng chú ý

- Khi cài đặt quan hệ Q_1 thành một bảng TQ_1 , sẽ phải khai báo khác nhau, tùy theo chọn cài đặt theo ĐTQH nào, vì mỗi đồ thị có số lượng CĐTX đến nút của Q_1 khác nhau:
  - Đồ thị (a): một chỉ mục (AB)
  - Đồ thị (b): hai chỉ mục (A) và (B)
  - Đồ thị (c): hai chỉ mục (AB) và (A)
  - Đồ thị (d): hai chỉ mục (AB) và (B)

Câu hỏi?

Buổi # 10
(lớp TH2008-1)

Cơ sở dữ liệu nâng cao

Đồng Thị Bích thủy

16/11/2010

Nội dung tuần này

III.10 Nhắc lại: Cách chọn lựa và khai báo đồ thị CĐTX với hệ QTCSDL

- Cài đặt cung (con đường truy xuất)
- Cài đặt nút (bảng)

III.11 Cấu trúc vật lý của bảng

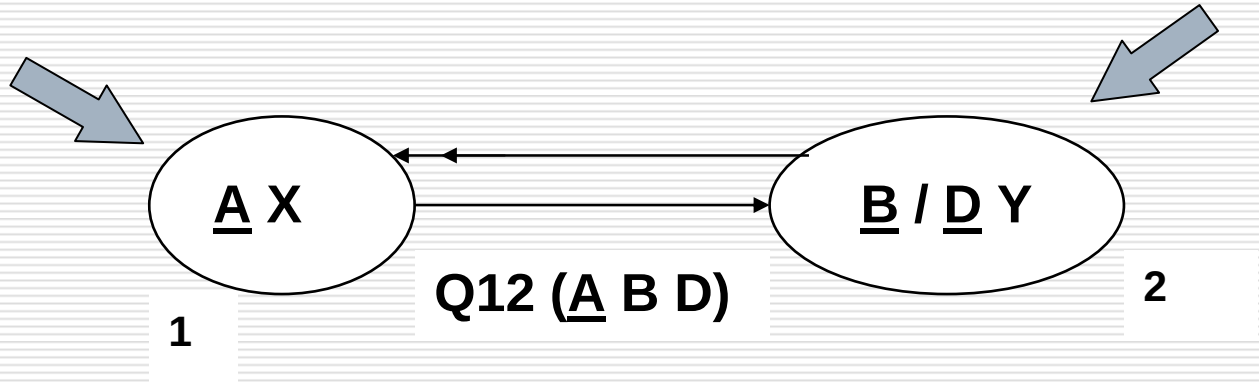
- Tập tin
- Chỉ mục

III.10 Cách chọn lựa và khai báo đồ thị CĐTX với hệ QTCSDL – nhắc lại

- Một ĐTCĐTX gồm những nút và cung có hướng
- Nút có thể là <u>nút vào</u> hoặc không
- Cung có hướng gồm:
  - cung với mũi tên đơn (=cđtx đơn trị = theo chiều của pth)
  - Cung với mũi tên kép (=cđtx đa trị = ngược chiều pth hoặc không có pth)

III.10 Cách chọn lựa ... – nhắc lại (2)

- Nút vào => khai báo khóa chính của quan hệ nút.
  - Nếu quan hệ nút có nhiều khóa → phải chọn 1 làm khóa chính. Khóa này cũng phải xuất hiện như khóa ngoại trong những quan hệ nút khác
  - Ví dụ:



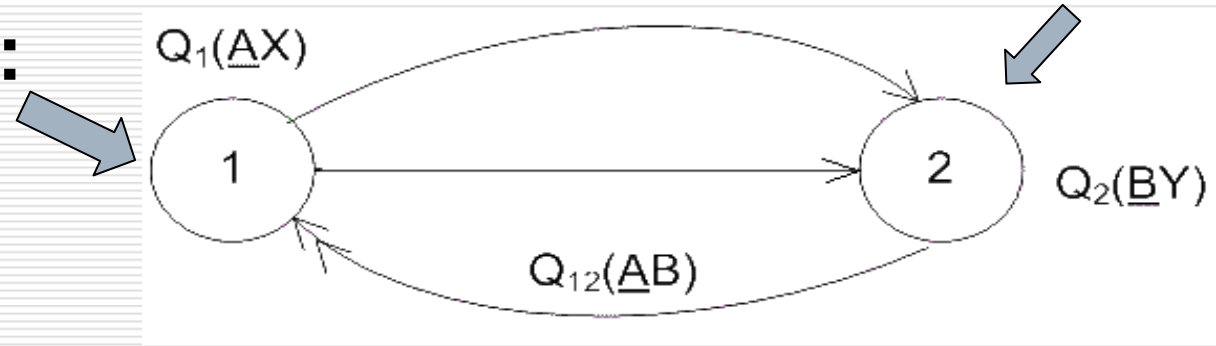
Nếu khai báo khóa chính của Q2 là B thì trong Q12 phải ưu tiên khóa ngoại B

III.10 Cách chọn lựa ... – nhắc lại (3)

- Không phải nút vào: không khai báo khóa chính của quan hệ nút
- Cung có mũi tên đơn:
  - khai báo khoá ngoại trong quan hệ cung
  - Ví dụ: trong Q12, khai báo B là khóa ngoại khi muốn cài đặt cung 12
- Cung có mũi tên kép:
  - Khai báo chỉ mục trên khóa ngoại trong quna hệ cung
  - Ví dụ: trong quan hệ Q12, khai báo B là chỉ mục

III.10 Cách chọn lựa ... – nhắc lại (4)

- Tóm lại:



- Với đồ thị cđtx trên, nếu không tách hoặc gộp nút, thì sẽ khai báo thành 2 bảng:
  - Q2(BY), B là khóa chính
  - Q1(AXB), A là khóa chính, B là khoá ngoại, chỉ mục (B)

Lưu ý: bảng Q1 bao gồm quan hệ nút 1 và quan hệ cung 12

III.11 Cấu trúc vật lý của bảng

- Là một tập tin, với cách truy xuất đến bản ghi (=dòng của bảng = bộ của quan hệ):
  - trực tiếp qua khóa của quan hệ, nếu có khai báo khóa quan hệ là khóa chính
  - tuần tự, nếu không khai báo khóa chính
  - "trực tiếp" qua chỉ mục, nếu có khai báo chỉ mục trên bảng

III.11 Cấu trúc vật lý của bảng (2)

- Có thể khai báo bao nhiêu chỉ mục trên một bảng?
  - Trên lý thuyết: nhiều (256 hoặc không hạn chế, tùy hệ QTCSDL)
  - Trên thực tế: cần cân nhắc, vì ảnh hưởng đến tốc độ thêm/xóa/sửa => càng nhiều chỉ mục , tốc độ càng chậm vì mỗi thao tác có liên quan đến giá trị chỉ mục sẽ kéo theo chuỗi cập nhật chỉ mục

III.11 Cấu trúc vật lý của bảng (3)

- Ngoài khóa ngoại, còn những thuộc tính/cụm thuộc tính nào có thể tạo thành chỉ mục?
  - Thuộc tính/cụm thuộc tính xuất hiện thường xuyên trong các điều kiện của phép chọn

III.11 Cấu trúc vật lý của bảng (4)

- Có bao nhiêu loại chỉ mục?

➔Theo cách sắp xếp:

- Chỉ mục gom nhóm (clustering index): các dòng trong bảng được sắp xếp vật lý theo thứ tự của chỉ mục
- Chỉ mục thường: các dòng trong bảng được sắp xếp theo thứ tự lô-gíc của chỉ mục
- Trong một bảng chỉ được phép có một chỉ mục gom nhóm duy nhất
- Các hệ QTCSDLQH thường cài đặt khóa chính theo loại chỉ mục gom nhóm

III.11 Cấu trúc vật lý của bảng (5)

➔**Theo cách lập chỉ mục:**

- **Chỉ mục dày (*dense index*): mỗi giá trị chỉ mục trỏ đến một dòng của bảng**
- **Chỉ mục thưa (*sparse index*): mỗi giá trị chỉ mục trỏ đến một khối dữ liệu (*block*) chứa n dòng**

III.11 Cấu trúc vật lý của bảng (6)

- **Cấu trúc cài đặt chỉ mục:**
  - **Cây B^+ → phổ biến nhất**
  - **ISAM (Indexed Sequential Access Method)**
  - **Hàm Băm**

III.11 Cấu trúc vật lý của bảng (7)

- Các hệ QTCSDL có những chọn lựa khác nhau cho các chỉ mục:
  - Sybase chọn cài đặt chỉ mục gom nhóm theo kiểu chỉ mục thưa với cấu trúc cây B^+
  - Vài phiên bản của Oracle cài đặt chỉ mục gom nhóm theo kiểu chỉ mục dày
  - Ingres cung cấp chỉ mục gom nhóm theo kiểu thưa với cấu trúc ISAM và chỉ mục gom nhóm theo kiểu dày với cấu trúc cây B^+

III.11 Cấu trúc vật lý của bảng (8)

- Cần lưu ý: quan sát thuộc tính/cụm thuộc tính tạo nên chỉ mục
  - → nếu giá trị của chúng bị thay đổi thường xuyên qua thao tác cập nhật thì hệ thống chỉ mục có thể bị nhiều chỗ "lủng"
  - → hoạt động kém hiệu quả
  - → Cần cho sắp xếp lại hệ thống chỉ mục

Câu hỏi?

Giáo trình

Hệ quản trị cơ sở dữ liệu Visual Foxpro

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Bài 1 - Mở đầu

I. Giới thiệu

- Nội dung môn học

  Khai thác hệ quản trị cơ sở dữ liệu Visual Foxpro

  Tiếp cận với công việc xây dựng một hệ thống quản lý

- Tài liệu tham khảo

  [1] - Sử dụng và khai thác Microsoft Visual Foxpro 6.0 - Nguyễn Ngọc Minh, Nguyễn Đình Đề - NXB Lao động 2003

  [2] - Bộ Help MSDN của Microsoft (tiếng anh)

  [3] - Phân tích và thiết kế hệ thống thông tin - Đào Thanh Tĩnh - NXB Quân đội 2004

  [4] - Help của Visual Foxpro

II. Hệ quản trị csdl

- Hệ quản trị cơ sở dữ liệu (Database Management System) là một hệ thống phần mềm máy tính được thiết kế chuyên biệt cho bài toán quản lý. Mô hình dữ liệu của các hệ quản trị csdl là mô hình quan hệ.

- Hệ quản trị csdl cung cấp 2 tính năng chính sau:

  1- Lưu trữ, quản lý dữ liệu

  2- Công cụ truy cập, khai thác dữ liệu

- Các Hệ quản trị csdl thường dùng hiện nay:

  1- SQL Server: Phát triển bởi Microsoft; là hệ CSDL dùng cho các tổ chức, doanh nghiệp lớn; Khả năng đáp ứng trên mạng cao; Đảm bảo tính an toàn, bảo mật.

  2- Oracle: Phát triển bởi Oracle; là hệ CSDL dùng cho các tổ chức, doanh nghiệp lớn; Khả năng đáp ứng trên mạng cao; Đảm bảo tính an toàn, bảo mật. (Được đánh giá là mạnh hơn SQL verver và thực tế giá phần mền này cũng đắt hơn SQL Server)

  3- Foxpro, Access: Phát triển bởi Microsoft; là hệ CSDL dùng cho các tổ chức, doanh nghiệp vừa và nhỏ; có khả năng triển khai trên mạng; tính bảo mật, an toàn là không cao.

  4- MySQL: Hệ CSDL mã nguồn mở, được sử dụng chủ yếu cho các ứng dụng Web.

III. Mô hình dữ liệu quan hệ

Trong mô hình này người ta mô tả các [đối tượng] (như con người, hàng hóa, môn học ...) thông qua các bảng (Table), mỗi bảng lại được mô tả qua các trường dữ liệu.

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

- Bảng (Table): Mô tả qua các cột - trường (Field) và các dòng - Bản ghi (Record) (ví dụ) (Mỗi bảng có thể hiểu như một [quan hệ])
- Trường (Field): Mô tả 1 thuộc tính nào đó của đối tượng thông qua tên trường (Field name), kiểu (Type) và động rộng (Width) (ví dụ).
- Bản ghi (Record): Tập các giá trị cụ thể của các trường của cùng 1 đối tượng (ví dụ).
- Quan hệ giữa các bảng (Relationship): Là cách thức mô tả một đối tượng lớn hơn từ những đối tượng nhỏ (ví dụ).

IV. bài toán quản lý

- Bài toán quản lý có ở hầu hết mọi đơn vị, tổ chức, thậm chí với cả một số cá nhân. Những bài toán quản lý có thể kể đến như:
  - Quản lý nhân sự
  - Quản lý hàng hóa
  - Quản lý kế toán
  - Quản lý lương
  - Quản lý sinh viên, điểm
  - Quản lý sách
  - ....
- Đặc điểm chung của những bài toán này:
  - Không cần sử dụng những thuật toán phức tạp mà yêu cầu chủ yếu là: Thống kê, tìm kiếm, lọc thông tin.
  - Khối lượng dữ liệu thường là rất lớn. Đặc điểm này thực sự là một thách thức lơn nếu chúng ta phải quản lý thủ công (việc tìm kiếm, thống kê, lọc -> mất rất nhiều thời gian và không chính xác)

Bài tập môn học - Xây dựng hệ quản lý điểm

A. Mô tả bài toán

- Bài toán **Quản lý điểm** (đối với các trường Đại học, Cao đẳng): Công tác quản lý điểm (hay kết quả học tập) của sinh viên đóng vai trò hết sức quan trọng đối với hoạt động của các trường ĐH và CĐ. Trong khoảng 10 năm trở lại đây công tác quản lý điểm (ở một số trường trong đó có HVKTQS) đã dần được tin học hóa nhằm đem lại hiệu quả cao hơn, tin cậy hơn.

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

- Bài toán **Quản lý điểm** đặt ra các vấn đề cơ bản như sau: Thể hiện được mô hình tổ chức quản lý sinh viên theo khóa, theo lớp, các loại hình đào tạo; Quản lý các môn học của các lớp theo học kỳ và kết quả học tập của sinh viên đối với các môn học đó. Hệ thống còn phải cho phép tạo ra các báo cáo từ kết quả học tập của sinh viên nhằm phục vụ công tác điều hành huấn luyện như: Tổng kết kết quả học tập theo kỳ, theo năm, theo khóa; In Danh sách thi lại; In Bảng điểm học kỳ; In Bảng điểm các nhân…

  Ngoài các chức năng chính như trên, hệ thống này còn cần thêm một số chức năng khác như: cập nhật các loại danh mục dữ liệu (danh mục lớp, danh mục loại hình đào tạo, danh mục ngành học …); các chức năng sao chép/copy dữ liệu; chuyển đổi – tách lớp; các chức năng trợ giúp …

- Có thể mô tả sơ lược các công việc chính (đối với 1 khóa học) trong hệ thống quản lý điểm như sau:

  1. Với mỗi khóa học mới -> cần phải có mô tả cho khóa mới này.
  2. Mô tả danh mục các lớp cho khóa mới.
  3. Với mỗi lớp đã có cập nhật danh sách sinh viên của lớp
  4. Với mỗi học kỳ cần cập nhật danh sách môn học/lớp
  5. Khi có kết quả các môn học -> Cập nhật điểm môn học/lớp
  6. Tổng kết kết quả học tập học kỳ, năm học, khóa học ..
  7. In Danh sách thi lại
  8. In Bảng điểm học kỳ
  9. In Bảng điểm các nhân.

B. Yêu cầu

Xây dựng hệ thống **Quản lý điểm** phục vụ công tác quản lý điểm của các trường ĐH và CĐ với các yêu cầu sau:

1. Cập nhật từ điển
   - Cập nhật danh mục khoá
   - Cập nhật danh mục lớp
   - Cập nhật danh mục ngành học
   - Cập nhật danh mục loại hình đào tạo
   - Cập nhật danh mục giáo viên
2. Cập nhật dữ liệu
   - Cập nhật hồ sơ sinh viên

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

- Cập nhật môn học/lớp/học kỳ
- Cập nhật điểm thi kết thúc môn

3. Báo cáo/ thống kê
   - In danh sách lớp
   - In Danh sách thi
   - In phiếu điểm
   - In tổng kết kết quả học kỳ
   - In tổng kết kết quả năm học
   - In tổng kết kết quả toàn khóa học
   - In bảng điểm học kỳ
   - In bảng điểm cá nhân

C. Một số gợi ý

- Các bảng dữ liệu trong hệ thống quản lý điểm có thể bao gồm các bảng sau:

1. Danh mục khoá DMKHOA.DBF

Tên trường	Kiểu	Độ rộng	Ghi chú
Ma	***C***	***10***	***Mã khoá (Khoá chính)***
Ten	C	30	Tên khoá
Nienkhoa	C	11	Niên khoá (Ghi 2004 – 2005, hoặc 2005 – 2006 …)
Ghichu	C	200	Ghi chú

2. Danh mục lớp DMLOP.DBF

Tên trường	Kiểu	Độ rộng	Ghi chú
Ma	***C***	***10***	***Mã lớp (khoá chính)***
Ten	C	30	Tên lớp
NgayKG	D	8	Ngày khai giảng
NgayBG	D	8	Ngày bế giảng
Manganh	*C*	*10*	*Mã ngành (khoá ngoài – dùng để đặt liên kết đến bảng DMNGANH)*
MaLHDT	*C*	*10*	*Mã loại hình đào tạo (khoá ngoài – dùng để đặt liên kết đến bảng DMLHDT)*
Makhoa	*C*	*10*	*Ma khoá (khoá ngoài – dùng để đặt quan hệ đến bảng DMKHOA)*
Ghichu	C	200	Ghi chú

3. Danh mục ngành học DMNGANH.DBF

Tên trường	Kiểu	Độ rộng	Ghi chú
Ma	***C***	***10***	***Mã ngành (khoá chính)***

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Ten	C	30	Tên ngành
Ghichu	C	200	Ghi chú

4. Danh mục loại hình đào tạo DMLHDT.DBF

Tên trường	Kiểu	Độ rộng	Ghi chú
Ma	***C***	***10***	***Mã loại hình đào tạo (khoá chính)***
Ten	C	30	Tên loại hình
Ghichu	C	200	Ghi chú

5. Hồ sơ sinh viên HOSOSV.DBF

Tên trường	Kiểu	Độ rộng	Ghi chú
Ma	***C***	***10***	***Mã sinh viên (Khoá chính)*** Mã sinh viên có dạng XXXXYYYYYY, trong đó XXXX có dạng là năm sinh viên đó trúng tuyển vào trường, YYYYYY là số thứ tự của sinh viên nhập học năm XXXX (Ví dụ: mã 2004000012 là mã của sinh viên thứ 12 trúng tuyển năm 2004….)
HoTen	C	30	Họ tên sinh viên
Ngaysinh	D	8	Ngày sinh
Gioi	L	1	Logic, .T. – Nam, .F. – Nữ
Hokhau	C	200	Hộ khẩu trường trú
Diachi	C	200	Địa chỉ
Malop	*C*	*10*	*Mã lớp (khoá ngoài – dùng để liên kết đến bảng DMLOP)*
Ghichu	C	200	Ghi chú

6. Môn học MONHOC.DBF

Tên trường	Kiểu	Độ rộng	Ghi chú
Ma	***C***	***10***	***Mã môn (Khoá chính)***
Tenmon	C	50	Tên môn
Sotiet	N	3	Số tiết
SoDVHT	N	3,1	Số đơn vị học trình
Hocky	N	1	Học kỳ (ghi 1 – Học kỳ 1, ghi 2 – Học kỳ 2)
Namhoc	N	4	Năm học (Ghi năm học đầu trong)
NgayThi	D	8	Ngày thi
Malop	*C*	*10*	*Mã lớp (khoá ngoài – dùng để liên kết đến bảng DMLOP)*
MaGV	*C*	*10*	*Mã giáo viên (Khoá ngoài – dùng để liên kết đến bảng GIAOVIEN)*
Ghichu	C	200	Ghi chú

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

7. Bảng GIAOVIEN.DBF

Tên trường	Kiểu	Độ rộng	Ghi chú
Ma	***C***	***10***	***Mã giáo viên***
Hoten	C	30	Họ tên giáo viên
Hocham	C	10	Học hàm (GV, GVC, GS, PGS)
Hocvi	C	10	Học vị (CN, KS, ThS, TS, TSHK)
Ghichu	C	200	Ghi chú

8. Bảng điểm DIEM.DBF

Tên trường	Kiểu	Độ rộng	Ghi chú
Mamon	***C***	***10***	***Mã môn***
MaSV	***C***	***10***	***Mã sinh viên***
Diem1	N	5,2	Điểm thi lần 1
Diem2	N	5,2	Điểm thi lần 2
Diem3	N	5,2	Điểm thi lần 3
Ghichu	C	200	Ghi chú

- Các chức năng của chương trình có thể thiết kế như sau:
  - Cập nhật danh mục khoá

Giao diện này có dạng như sau:

A1 – Danh sách khoá (Sử dụng GRID) Khi chọn 1 dòng (1 khoá) thông tin về khoá được chọn sẽ hiển thị trong ô A2
A2 – Chi tiết khoá được chọn
A3 Các chức năng : Thêm, sửa, xoá …(1 khoá)

  - Cập nhật danh mục lớp

A1 Chọn khoá : …… (sử dụng ô combox) Chức năng [OK] Khi chọn OK, danh sách các lớp của khoá được chọn sẽ hiển thị trong ô A2	A3 – Chi tiết về lớp được chọn
A2 – Danh sách lớp của khoá được chọn (Sử dụng GRID) Khi chọn 1 dòng (1 lớp) thông tin chi tiết về lớp này sẽ hiển thị trong ô A3	Các chức năng: Thêm, sửa, Xoá … (1 lớp)

  - Cập nhật danh mục ngành

Tương tự cập nhật danh mục khoá

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

- Cập nhật danh mục loại hình đào tạo

Tương tự cập nhật danh mục khoá

- Cập nhật danh mục giáo viên

Tương tự cập nhật danh mục khoá

- Cập nhật hồ sơ sinh viên

Giao diện này có dạng như sau:

A1 Chọn khoá: ….. (Sử dụng ô Combox) Chọn lớp: …… (Sử dụng ô Combox) Chức năng [OK] Khi chọn [OK] danh sách sinh viên của lớp/khoá đã chọn sẽ hiển thị trong ô A2	A3 - Chi tiết về sinh viên được chọn
A2 – Danh sách sinh viên của lớp/khoá đã chọn (sử dụng GRID) Khi chọn 1 dòng (1 sinh viên) thì thông tin chi tiết về sinh viên này sẽ hiển thị trong ô A3	Các chức năng: Thêm, Sửa, Xoá … (1 sinh viên)

- Cập nhật môn học/lớp/học kỳ

Giao diện này có dạng như sau:

A1 Chọn khoá: …. (sử dụng ô Combox) Chọn lớp: …. (Sử dụng ô Combox) Chọn học kỳ: …. Năm học: …… Chức năng [OK] Khi chọn [OK] danh sách môn học của lớp/khoá/học kỳ-năm học đã chọn sẽ hiển thị trong ô A2	A3 – Chi tiết về môn học được chọn
A2 – Danh sách môn học của lớp/khoá/học kỳ-năm học đã chọn (sử dụng GRID) Khi chọn 1 dòng (1 môn học) thì thông tin chi tiết về môn học này sẽ hiển thị trong ô A3	Các chức năng: Thêm, Sửa, Xoá … (1 môn học)

- Cập nhật điểm thi kết thúc môn

Giao diện này có dạng như sau:

A1 Chọn khoá: …. (sử dụng ô Combox) Chọn lớp: …. (Sử dụng ô Combox)

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Chọn học kỳ: …. Năm học: …… Chọn môn học: …….(sử dụng ô Combox) Thi lần: ……..(1 – thi lần đầu, 2 thi lại lần 1, 3 – thi lại lần 2) Chức năng [OK] Khi chọn [OK] danh sách "lớp-môn học" đã chọn sẽ hiển thị trong ô A2
A2 – Danh sách "lớp-môn học" đã chọn (Sử dụng GRID) (nếu là thi lần 1 – cho phép nhập vào cột điểm 1, lần 2 cho phép nhập vào cột điểm 2, lần 3 cho phép nhập vào cột điểm 3)

Lưu ý: Danh sách "lớp-môn học" do chương trình tự động sinh ra, cần lưu ý khi thiết kế để tránh sinh ra nhiều lần.

- In danh sách lớp

Giao diện này có dạng như sau:

A1 Chọn khoá: …. (sử dụng ô Combox) Chọn lớp: …. (Sử dụng ô Combox) Chức năng: Xem, In, Thoát (chức năng Xem cho phép xem report "danh sách lớp" trên màn hình chức năng In cho phép đưa report "danh sách lớp" ra máy in)

Report "danh sách lớp" có dạng như sau:

Học viện KTQS

Phòng Đào tạo

Danh sách lớp

Lớp: ……, Khoá: ……

STT	Mã SV	Họ tên	Ngày sinh	Hộ khẩu	Ghi chú
…	…	…	…	…	…

- In danh sách thi

Giao diện này có dạng như sau:

A1 Chọn khoá: …. (sử dụng ô Combox) Chọn lớp: …. (Sử dụng ô Combox)

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Chọn học kỳ: …. Năm học: ……

Chọn môn học: …….(sử dụng ô Combox)

Thi lần: ……..(1 – thi lần đầu, 2 thi lại lần 1, 3 – thi lại lần 2)

Chức năng: Xem, In, Thoát

(chức năng Xem cho phép xem report "danh sách thi" trên màn hình

chức năng In cho phép đưa report "danh sách thi" ra máy in)

Report "Danh sách thi" có dạng như sau:

Học viện KTQS

Phòng Đào tạo

Danh sách thi

Lớp: …….., khoá:……

Môn học: …….Học kỳ: .... năm học:……

Giáo viên: …….., ngày thi:……..

STT	Mã SV	Họ tên	Ngày sinh	Đề số	Kí nhận	Ghi chú
…	…	…	…	…	…	…

Giáo viên coi thi (Ký và ghi rõ họ tên)

- In phiếu điểm

Giao diện này có dạng như sau:

A1

Chọn khoá: …. (sử dụng ô Combox)

Chọn lớp: …. (Sử dụng ô Combox)

Chọn học kỳ: …. Năm học: ……

Chọn môn học: …….(sử dụng ô Combox)

Thi lần: ……..(1 – thi lần đầu, 2 thi lại lần 1, 3 – thi lại lần 2)

Chức năng: Xem, In, Thoát

(chức năng Xem cho phép xem report "phiếu điểm" trên màn hình

chức năng In cho phép đưa report "phiếu điểm" ra máy in)

Report "phiếu điểm" có dạng như sau:

Học viện KTQS

Phòng Đào tạo

Phiếu điểm

Lớp: …….., khoá:……

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Môn học: ……Học kỳ: .... năm học……

Giáo viên: …….., ngày thi:……..

STT	Mã SV	Họ tên	Ngày sinh	Đề số	Điểm	Ghi chú
…	…	…	…	…	…	…

Giáo viên chấm thi (Ký và ghi rõ họ tên)

- In tổng kết kết quả học kỳ

Giao diện này có dạng như sau:

A1

Chọn khoá: …. (sử dụng ô Combox)

Chọn lớp: …. (Sử dụng ô Combox)

Chọn học kỳ: …. Năm học: ……

Chức năng: Xem, In, Thoát

(chức năng Xem cho phép xem report "Tổng kết học kỳ" trên màn hình

chức năng In cho phép đưa report "Tổng kết học kỳ" ra máy in)

Report "Tổng kết học kỳ" có dạng như sau:

Học viện KTQS

Phòng Đào tạo

Tổng kết kết quả học kỳ

Học kỳ: .... năm học:……

Lớp: …….., khoá:……

STT	Mã SV	Họ tên	Ngày sinh	Điểm TBC	Xếp loại	Ghi chú
…	…	…	…	…	…	…

Xếp loại theo điểm trung bình chung được tính như sau:

Điểm TBC>=9 -> Xuất sắc; Điểm TBC>=8 -> Giỏi; Điểm TBC>=7 -> Khá; Điểm TBC>=6 -> Trung bình khá; Điểm TBC>=5 -> Trung bình; Điểm TBC<9 -> Không đạt

- In tổng kết kết quả năm học

Giao diện này có dạng như sau:

A1

Chọn khoá: …. (sử dụng ô Combox)

Chọn lớp: …. (Sử dụng ô Combox)

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Chọn Năm học: ……

Chức năng: Xem, In, Thoát

(chức năng Xem cho phép xem report "Tổng kết năm học" trên màn hình

chức năng In cho phép đưa report "Tổng kết năm học" ra máy in)

Report "Tổng kết năm học" có dạng như sau:

Học viện KTQS

Phòng Đào tạo

Tổng kết kết quả năm học

Năm học:……

Lớp: …….., khoá:……

STT	Mã SV	Họ tên	Ngày sinh	Điểm TBC	Xếp loại	Ghi chú
…	…	…	…	…	…	…

Xếp loại theo điểm trung bình chung được tính như sau:

Điểm TBC>=9 -> Xuất sắc; Điểm TBC>=8 -> Giỏi; Điểm TBC>=7 -> Khá; Điểm TBC>=6 -> Trung bình khá; Điểm TBC>=5 -> Trung bình; Điểm TBC<9 -> Không đạt

- In tổng kết kết quả toàn khoá

Giao diện này có dạng như sau:

A1

Chọn khoá: …. (sử dụng ô Combox)

Chọn lớp: …. (Sử dụng ô Combox)

Chức năng: Xem, In, Thoát

(chức năng Xem cho phép xem report "Tổng kết khoá học" trên màn hình

chức năng In cho phép đưa report "Tổng kết khoá học" ra máy in)

Report "Tổng kết năm học" có dạng như sau:

Học viện KTQS

Phòng Đào tạo

Tổng kết kết quả khoá học

Lớp: …….., khoá:……

STT	Mã SV	Họ tên	Ngày sinh	Điểm TBC	Xếp loại	Ghi chú
…	…	…	…	…	…	…

Xếp loại theo điểm trung bình chung được tính như sau:

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Điểm TBC>=9 -> Xuất sắc; Điểm TBC>=8 -> Giỏi; Điểm TBC>=7 -> Khá; Điểm TBC>=6 -> Trung bình khá; Điểm TBC>=5 -> Trung bình; Điểm TBC<9 -> Không đạt

- In bảng điểm học kỳ

Giao diện này có dạng như sau:

A1

Chọn khoá: …. (sử dụng ô Combox)

Chọn lớp: …. (Sử dụng ô Combox)

Chọn học kỳ: …. Năm học: ……

Chức năng: Xem, In, Thoát

(chức năng Xem cho phép xem report "Bảng điểm học kỳ" trên màn hình

chức năng In cho phép đưa report "Bảng điểm học kỳ" ra máy in)

Report "Bảng điểm học kỳ" có dạng như sau:

Học viện KTQS

Phòng Đào tạo

Bảng điểm học kỳ

Học kỳ: .... năm học:……

Lớp: …….., khoá:……

STT	Mã SV	Họ tên	Môn 1	Môn 2	…	…	…	…	TBC
…	…	…	…	…	…	…	…	…	…

- In bảng điểm cá nhân

Giao diện này có dạng như sau:

A1

Chọn khoá: …. (sử dụng ô Combox)

Chọn lớp: …. (Sử dụng ô Combox)

Chọn sinh viên: ……(sử dụng Combox)

Chức năng: Xem, In, Thoát

(chức năng Xem cho phép xem report "Bảng điểm cá nhân" trên màn hình

chức năng In cho phép đưa report "Bảng điểm cá nhân" ra máy in)

Report "Bảng điểm cá nhân" có dạng như sau:

Học viện KTQS

Phòng Đào tạo

Bảng điểm cá nhân

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Họ tên: ……, Ngày sinh

Loại hình đào tạo: ……, Ngành:……., Khoá học: ……..

STT	Mã môn	Tên môn	Số ĐVHT	Điểm 1	Điểm 2	Điểm 3	Ghi chú

Điểm trung bình chung toàn khoá: ……

Xếp loại: …….

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Bài 2 - Tổng quan về visual foxpro

I. Cài đặt visual foxpro

- Công cụ Visual Foxpro 6.0 là một thành phần trong bộ Visual Studio của Microsoft. Do đó cần cài đặt bộ Visual Studio và lựa chọn thành phần Visual Foxpro.

  [Start menu -> Visual Foxpro một thành phần của bộ Visual Studio]

- Bộ Visual Foxpro 7.0 (8.0, 9.0) được phát triển độc lập do đó có thể cài đặt riêng.

- Việc cài đặt các công cụ này tương đối đơn giản và giống với các phần mềm khác. Do đó việc cài đặt công cụ này các bạn tự tham khảo.

II. Một số thành phần trong giao diện của Visual foxpro

[Chạy Foxpro -> Giao diện chính của Visual Foxpro 6.0]

- **Cửa sổ lệnh (Command Windows)**

  Cho phép thực hiện các câu lệnh (Command).

  Ví dụ: Lệnh Create, Use, Browse ...

  [Chạy Foxpro -> Cửa sổ lệnh]

- Các menu của Foxpro (Tham khảo tài liệu)

- Các Tools bar của Foxpro (tham khảo tài liệu)

- **Tạo mới một Project**

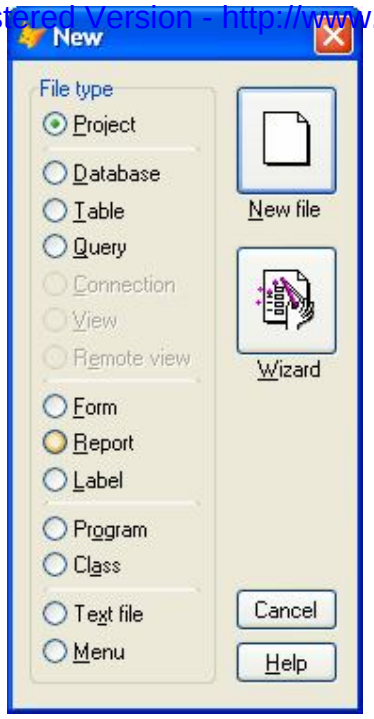
  Việc phát triển một ứng dụng trong Visual Foxpro thường được tổ chức thành một Project - Dự án.

  Việc tạo mới một Project có thể được thực hiện bằng 1 trong hai cách sau:

  Cách 1: Trong cửa sổ lệnh gõ vào lệnh CREATE PROJECT <Tên Project>

  Cách 2: Vào menu File ->Chọn New, khi đó giao diện New của Visual Foxpro xuất hiện như hình:

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com



Tiếp đến chọn File type = Project (Mặc định) rồi chọn **New file** để tạo một Project trống hoặc **Wizard** để tạo một Project theo một trong các mẫu thiết kế sẵn của Microsoft, ở đây chúng ta sẽ chọn New file. Khi chọn New file hộp thoại Create hiển thị cho phép nhập vào tên project cần tạo, tiếp chọn chọn **Save**.

Kết thúc việc tạo mới cửa sổ Project Manager xuất hiện (xem phần tiếp theo) cho phép quản lý, làm việc với Project này.

- **Cửa sổ Project (Project Manager)**

[Chạy Foxpro -> Project Manager]

Các thành phần chính trong Project Manager (Giới thiệu chi tiết các thành phần này)

+ Data: Database, Free Tables, Queries

+ Documents: Forms, Reports, Labels.

+ Class

+ Code: Program, API Libraries, Applications

+ Other: Menu, Text file, Other file

III. Những nét đặc trung cơ bản

1. Công cụ quản lý ứng dụng - Project manager đem lại một cách nhìn tổng quan, hiệu quả trong việc phát triển một ứng dụng nào đó.
2. Công cụ trực quan cho phép thiết kế giao diện được nhanh chóng, đẹp.

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

3. Công cụ Report Designer tiện dụng, cho phép tạo ra nhiều report đa dạng.

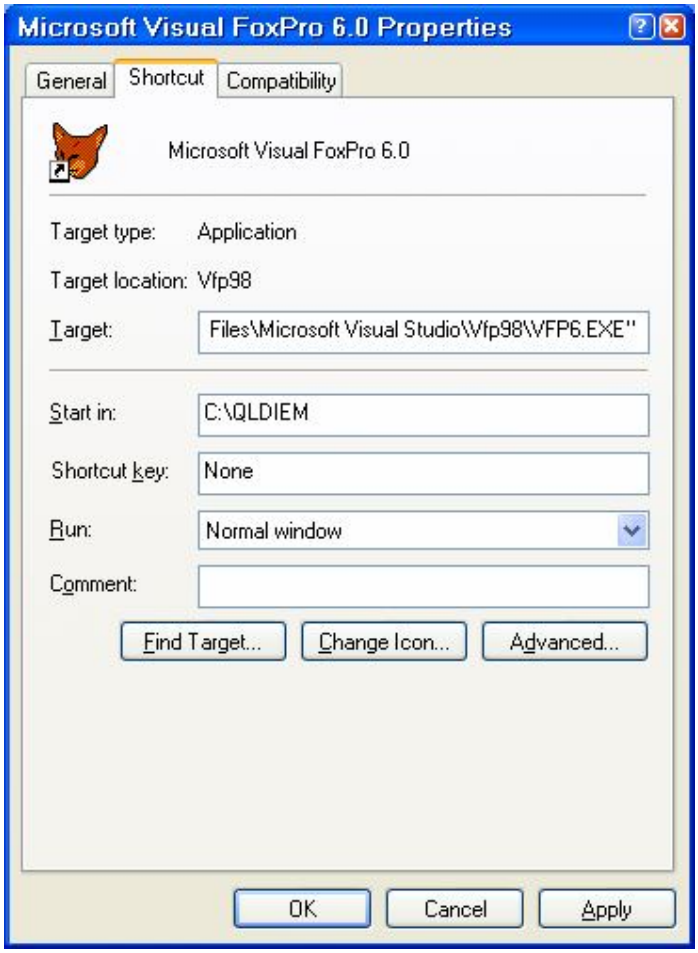
4. Công cụ tool tip trong Visual Foxpro 7.0 trợ giúp nhanh và hữu hiệu với việc lập trình.

Bài tập

1. Cài đặt Visual Foxpro 6.0 hoặc 7.0

2. Tạo thư mục QLDIEM trên ổ đĩa C:

3. Tạo shortcut trên Destop đến chương trình Visual Foxpro và đặt lại thuộc tính [Start in] vào thư mục C:\QLDIEM (như hình vẽ).



4. Tạo Project QLDIEM trống (Project sẽ được lưu trong thư mục C:\QLDIEM).

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Bài 3 - Các thành phần cơ bản của ngôn ngữ

I. từ khóa, kí hiệu, LệnH

- Từ khóa (Key Word): Là các từ mà Visual Foxpro đã sử dụng cho một mục đích nào đó. Và người dùng không được phép đặt tên trùng với từ khóa. Tham khảo tài liệu về tất cả các khóa của Visual Foxpro. Trong Foxpro từ khoá có thể không cần viết đầy đủ mà chỉ cần viêt 4 ký tự đầu của từ khoá đó, ví dụ từ khoá CREATE có thể chỉ cần viết Crea (Visual Foxpro không phân biệt chữ hoa, chữ thường).
- Kí hiệu comment: Đối với phần cuối 1 dòng dùng kí tự &&

  Với cả dòng dùng ký tự * hoặc &&

- **Lệnh của Visual Foxpro**

  Trong Visual Foxpro có hia loại câu lệnh: 1. Các câu lệnh có thể thực hiện trực tiếp từ cửa sổ lệnh (Command Windows) - gọi là Lệnh trực tiếp , 2. Các lệnh chỉ thực hiện trong chương trình. Các lệnh trực tiếp cũng thực hiện được trong chương trình.

  Cấu trúc lệnh trực tiếp của Foxpro có dạng:

  <Động từ> [Các tham số] ↵

  Trong đó: <Động từ> - Động từ tiếng anh (tương ứng với công việc cần làm); [Tham số] - Các tham số trong lệnh của fox có thể có, có thể không, hoặc có thể có nhiều; trường hợp có nhiều tham số thì thứ tự các tham số này có thể không quan trọng.

  Một số lệnh trực tiếp của Visual Foxpro

  1. Lệnh ?, ?? -> In kết quả lên màn hình
  2. QUIT -> Thoát khỏi foxpro
  3. CLEAR -> Xoá màn hình
  4. SET DEFAULT TO <Path> -> Thiết lập đường dẫn mặc định
  5. SET DATE <FRENCH | AMERICA> -> Thiết lập ngày tháng

II. kiểu dữ liệu

Kiểu dữ liệu là thuộc tính gắn liền với biến hay trường trong các bảng. Các kiểu dữ liệu trong Visual Foxpro gồm có: Character (Xâu kí tự), Date và DateTime (Ngày, giờ), Currency (Tiền tệ), Logic, Numeric (Số); các kiểu chỉ dùng cho các trường của bảng gồm có: Memo (Văn bản), General (Tổng hợp - ảnh).

1. Character

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Là xâu ký tự a->z, A->Z, 0->9 và một số các ký tự khác như ‘.’, ‘*’, ‘/’, ‘+’ ...

Hằng xâu kí tự: Đặt trong dấu “”, ‘’, hoặc []; ví dụ “Hà nội”, ‘Việt nam’ hoặc [Visual Fox]

Các phép toán:

+: Nối chuỗi 2 vào chuỗi 1.

-: Nối 2 chuỗi nhưng huỷ bỏ các khoảng trắng trước và sau mỗi chuỗi.

$: Chứa ở trong (st1$st2 – st1 có trong st2 ->TRUE, ngược lại ->FALSE)

2. Date and DateTime

Kiểu ngày, giờ dùng 8 bytes để lưu trữ.

Hằng Date và DateTime: {^yyyy-mm-dd [hh[:mm[:ss]][alp]]}; ví dụ: {^2005-12-10}, {^2005-12-10 10:20:25}

Các phép toán:

Ngày + Số	= Ngày
Số + Ngày	= Ngày
Ngày - Số	= Ngày
Ngày - Ngày	= Số

Tương tự cho ngày giờ

3. Currency

Kiểu tiền tệ (thực chất là kiểu số và thêm kí tự tiền $ hoặc ký tự bất kỳ khác) vào số đó. Kiểu này thường ít được sử dụng.

4. Logic

Kiểu logíc, tập giá trị là Đúng hoặc Sai

Hằng Logic: .T. ->TRUE đúng (hoặc .t.); .F. ->FALSE sai (hoặc .f.)

Các phép toán: NOT, AND, OR.

5. Numeric

Kiểu số, dùng tối đa 20 byte.

Hằng số: 10, 20, 15.7 ...

Các phép toán: +, - , *, /, ^ (mũ), % (chia lấy phần dư)

6. Memo

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Kiểu kí ức, dùng 4 byte để ánh xạ đến 1 file khác có phần mở rộng là fpt. Kiểu này cho phép lưu 1 văn bản rất dài.

7. General

Kiểu kí ức, dùng 4 byte để ánh xạ đến 1 file khác có phần mở rộng là fpt. Kiểu này cho phép lưu 1 đối tượng bất kỳ như file văn bản, file word, file excel, file ảnh và thường được sử dụng để lưu trữ file ảnh.

III. Biến

Có 3 loại biến: Biến do người sử dụng tự định nghĩa: Biến là trường của bảng đang mở; hoặc Biến hệ thống (do Visual Foxpro tạo ra).

Đối với các biến do người sử dụng định nghĩa:

- Là đối tượng lưu trữ dữ liệu trong bộ nhớ, được xác định qua tên biến, kiểu và phạm vi tác động của biến
  - Tên biến: Dài không quá 254 kí tự, có thể dùng các chữ cáI, chữ số, dấu gạch dưới (_) nhưng không được bắt đầu bằng chữ số (và không được trùng với từ khoá).
  - Kiểu: Các kiểu dữ liệu kể trên trừ kiểu General. Kiểu của biến trong Visual Foxpro có thể thay đổi.
  - Phạm vi tác động của biến phụ thuộc vào vị trí mà nó được khai báo. Ví dụ

Chương trình chính

Biến A

Chương trình Con 1

Biến B

Gọi chương trình con 2

Chương trình con 2

Biến C

Trong ví dụ trên: Biến A có tác động trong toán bộ chương trình chính, chương trình con 1 và chương trình con 2; Biến C chỉ có tác động trong chương trình con 2; Biến B có tác động trong chương trình con 1, nếu trong chương trình con 1 có lời gọi chương trình con 2 thì biến B có thể

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

tác động trong chương trình con 2 (nhưng nếu lời gọi chương trình con 2 tại vị trí khác mà ở đó không khai báo B thì lúc đó sẽ phát sinh lỗi).

- Khai báo biến: Trong Visual Foxpro thực ra không có khảI niệm khai báo biến (như trong PASCAL) mà các biến sẽ được sinh ra khi thực hiện lệnh gán (=) hoặc lệnh STORE. Ví dụ
  - A=150 -> Tạo ra biến có tên là **a**, kiểu là số
  - STORE ”Visual Foxpro“ TO monhoc -> Tạo ra biến **monhoc**, kiểu xâu kí tự
- Xoá bỏ các biến: Khi không dùng nữa thì có thể xoá bỏ các biến để giảI phóng bộ nhớ. Visual Foxpro dùng lệnh RELEASE <tenbien> để xoá bỏ các biến. Ví dụ RELEASE **a**; RELEASE **monhoc.**

Biến trường và biến hệ thống có phạm vi tác động trong toàn bộ chương trình.

IV. Hằng

- Là đối tượng dữ liệu mà giá trị của nó chỉ xác định 1 lần khi khai báo và không thay đổi được. Hằng cũng được xác định qua tên, kiểu và phạn vi tác động của nó.
- Khai báo hằng: Visual Foxpro dùng từ khoá #DEFINE <Tênhằng> <Giá trị> để khai báo hằng. Kiểu của hằng phụ thuộc vào kiểu của biểu thức <Giá trị>
- Phạn vi tác động của hằng được xác định tương tự với Biến.

V. biểu thức

- Biểu thức được tạo nên từ các biến, hằng, hàm và các toán tử (phép toán) theo một qui tắc nào đó (các toán tử và toán hạng phảI phù hợp nhau).
- Thứ tự ưu tiên của các phép toán trong 1 biểu thức

1. Các phép số học: Dấu âm, dương
   Luỹ thừa
   Nhân, chía
   Phần dư
   Cộng, trừ
2. Các phép chuỗi: +, -
3. Các phép quan hệ: =,<,>,>=,<=,<>
4. Các phép logic: NOT, AND, OR

- Sử dụng cập dấu () trong biểu thức để tránh sai sót về thứ tự ưu tiên các phép toán.

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

VI. Hàm

Trong Visual Foxpro có hai loại hàm: Hàm do Visual Foxpro cung cấp và hàm do người sử dụng tự tạo ra. Hàm bao giờ cũng trả về một giá trị nào đó. Trong phần này sẽ giới thiệu một số hàm do Visual Foxpro cung cấp.

- Hàm xâu kí tự

  1. ALLTRIM(st), LTRIM(st), RTRIM(st)
  2. LEFT(st,n), RIGHT(st,2), SUBSTRING(st,p,n)
  3. AT(st1,st2,n), RAT(st1,st2,n)
  4. LOWER(st), UPPER(st)
  5. ASC(ch), CHR(n)
  6. DTOC(d)
  7. LEN(st)
  8. STR(ex)

- Hàm Date và DateTime

  1. DATE()
  2. DATETIME()
  3. TIME()
  4. CTOD(st)
  5. DAY(d)
  6. MONTH(d)
  7. YEAR(d)

- Hàm số

  1. ABS(ex)
  2. INT(ex)
  3. ROUND(ex,n)
  4. VAL(st)

- Hàm tự tạo của người sử dụng

  Cú pháp

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

FUNCTION <Tenham>

[PARAMETER <danh sách các tham số>]

.... (Các lệnh của fox)

RETURN <Biểu thức>

Trong đó TenHam: Tên hàm do người sử dụng tự định nghĩa

Parameter: Từ khoá cho phép truyền vào các tham số cho hàm

Bài tập

1. Giả sử Họ tên người trong tiếng Việt luôn có dạng "Họ Đệm Tên", trong đó các phần Họ, Đệm, Tên luôn cách nhau 1 dấu cách; phần Đệm có thể có hoặc không hoặc có nhiều hơn 1 chữ. Hãy viết hàm TENVN(hoten) để lấy ra phần Tên của tên người trong tiếng Việt, trong đó hoten là xâu họ tên người trong tiếng Việt.

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Bài 4 - thiết kế database và tables

I. Giới thiệu

- Bảng dữ liệu (gọi tắt là bảng) là nơi lưu trữ dữ liệu và nó đóng một vai trò quan trọng trong các bài toán quản lý. Xuất phát từ việc phân tích yêu cầu, các mẫu biểu của bài toán để từ đó đưa ra cấu trúc các bảng sao cho phù hợp, đáp ứng các yêu cầu đặt ra là một công việc khá phức tạp, đòi hỏi phảI có kiến thức và kinh nghiệm trong lĩnh vực cụ thể đó.
- Có thể tham khảo một số vấn đề giúp các bạn rõ hơn trong việc phân tích, thiết kế dữ liệu trong tài liệu [1] từ trang 131 đến 142, hoặc tham khảo tài liệu [3].

II. Tạo Database

1. Tạo mới Database

Cách 1: Sử dụng câu lệnh

Bằng cách này chúng ta có thể tạo ta một Database, tuy nhiên database này là một Database độc lập (không liên quan đến Project nào)

Câu lệnh:

CREATE DATABASE <TenDatabase> ↵

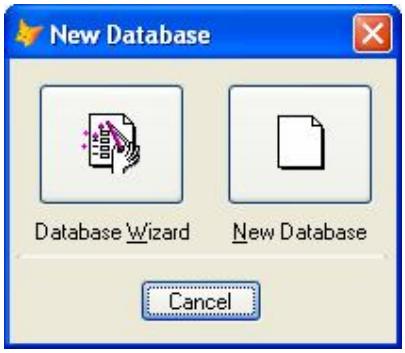
Kết thúc lệnh này Visual Foxpro sẽ tạo ra file Database có tên là TenDatabase (phần mở rộng là DBC)

Về sau có thể thêm Database này vào trong 1 Project nào đó.

Cách 2: Sử dụng Project Manager

Bằng cách này thì Database được tạo ra sẽ là một thành phần của Project

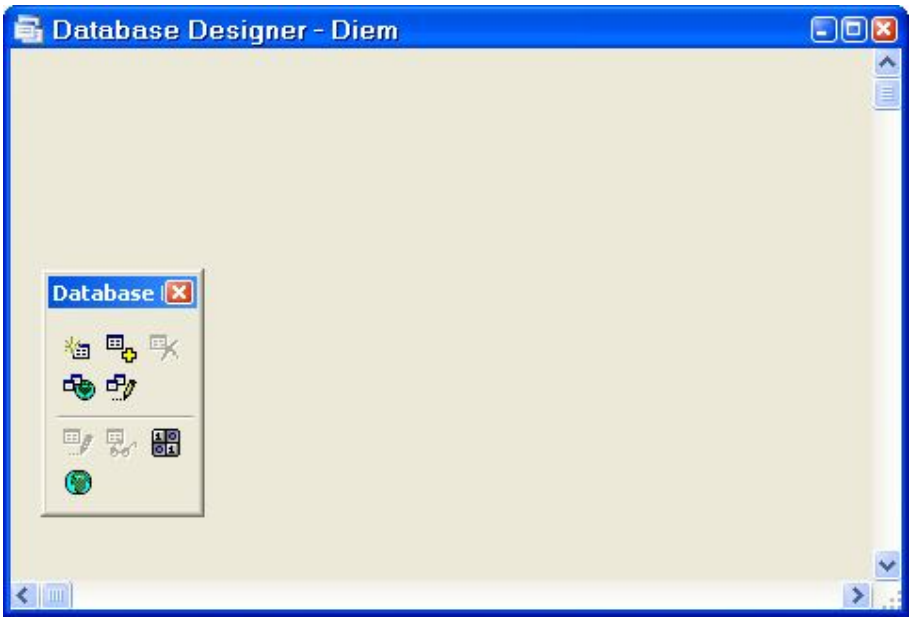
1. Mở Project QLDIEM
2. Trong Tab Data, chọn mục Database rồi chọn New, khi đó giao diện New database xuất hiện như hình 1.



Hình 1 - Hộp thoại New database

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

3. Tiếp đến chọn New Database để tạo một database trống (chọn Database Wizard để tạo một Database theo một trong các mẫu thiết kế sẵn của Visual Foxpro).
4. Tiếp đến Visual Foxpro yêu cầu nhập vào tên Database (file có phần mở rộng là DBC), rồi chọn Save để ghi lại. Khi đó giao diện Database designer xuất hiện như hình 2.
5. Giao diện Database Designer cho phép xem một cách tổng thể các thành phần (bảng, view, mỗi quan hệ giữa các bảng ...) trong csdl. Hình 2 cho ta thấy Database Diem tại thời điểm này còn trống, chúng ta sẽ trở lại giao diện này trong các phần sau (đóng cửa sổ này lại).



Hình 2 - Giao diện Database Designer

2. Sửa đổi Database

1. Trong Project Manager, chọn Tab Data, tiếp đến chọn Database và mở rộng đối tượng này (bằng cách kích vào dấu + phía trước) .
2. Chọn Database cần sửa (vì trong một Project có thể có nhiều hơn một Database) rồi chọn lệnh Modify, khi đó cửa sổ Database designer (hình 2) xuất hiện.

3. Thêm vào Project một Database đã có

Trường hợp Project sử dụng một Database đã có (đã thiết kế trước đó) thì chúng ta có thể thêm (Add) Database đó vào Project.

1. Trong Project Manager, chọn Tab Data, tiếp đến chọn mục Database (đặt vệt sáng ở mục này).
2. Chọn lệnh Add, khi đó sẽ xuất hiện cửa số open file (hình 3) cho phép chúng ta chọn csdl cần thêm vào.

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Hình 3 - Hộp thoại Open file (Select Database)

4. Loại bỏ một Database khỏi Project

Khi không cần dùng một Database nào đó, chúng ta có thể loại bỏ Database đó khỏi Project như sau:

1. Trong Project Manager, chọn Tab Data, tiếp đến chọn mục Database và mở rộng đối tượng này (bằng cách kích vào dấu + phía trước).
2. Chọn Database cần bỏ, tiếp đến chọn lệnh Remove... khi đó xuất hiện hộp thoại như hình 4.

Hình 4 - Hộp thoại Remove Database

3. Chọn Remove (khỏi Project) nếu không muốn xoá hẳn Database này; chọn Delete nếu muốn xoá hẳn Database này; chọn Cancel khi muốn huỷ bỏ thao tác này.

III. Tạo bảng

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

1. Tạo mới Tables

Cách 1: Sử dụng câu lệnh

Bằng cách này chúng ta có thể tạo ta một Table, tuy nhiên Table này là một Table độc lập (không liên quan đến Project nào).

Câu lệnh (dạng đơn giản):

CREATE TABLE <TableName>(FieldName1 Ftype1(width1) [, ...])

Trong đó: TableName - tên bảng (không chứa dấu cách), FieldName1 - Tên trường 1, Ftype1 - Kiểu trường 1 (dùng các chữ cái đầu của kiểu trong fox để mô tả), with1 - độ rộng của trường. Tương tự như vậy cho trường thứ 2.

Ví dụ: CREATE TABLE Hososv(masv C(10), Hoten C(30))

Cách 1: Sử dụng Project Manager

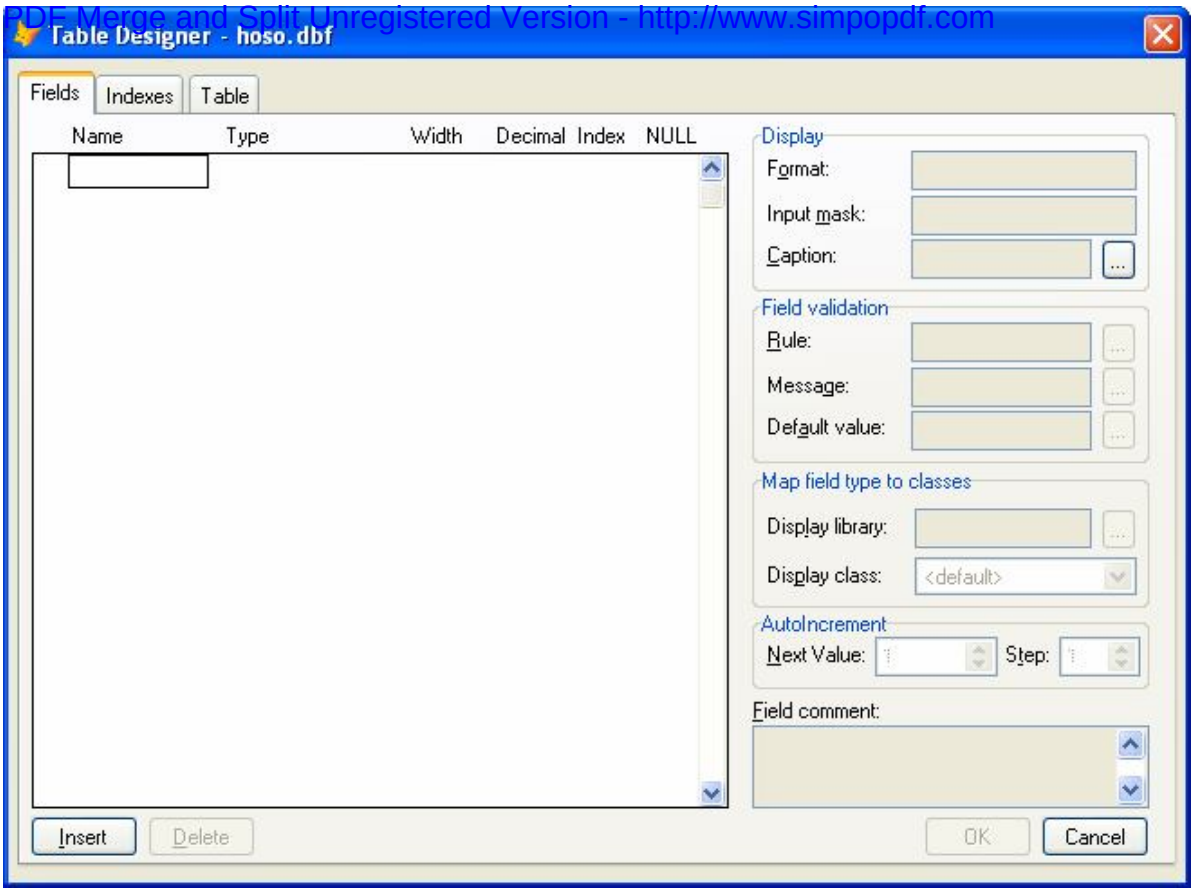
1. Trong Project Manager chọn Tab Data, chọn mục Database và mở rộng đối tượng này (bằng cách kích vào dấu + phía trước).
2. Chọn Database cần làm việc và mở rộng đối tượng này (bằng cách kích vào dấu + phía trước), tiếp đến chọn Tables rồi chọn New. Khi đó giao diện New table có dạng như hình 5.

Hình 5 - New table

3. Tiếp đến chọn New Tbale để tạo một Table trống (chọn Table Wizard để tạo một Table theo một trong các mẫu thiết kế sẵn của Visual Foxpro).
4. Tiếp đến Visual Foxpro yêu cầu nhập vào tên Table (file có phần mở rộng là dbf), rồi chọn Save để ghi lại. Giao diện Table Designer xuất hiện như hình 6.

Giao diện Table Designer

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com



Hình 6 - Table Designer

- **Tab Field:** Phần khai báo chính cho 1 trường của bảng

  a. Name : Tên trường

  b. Type: Kiểu

  c. Width, Decimal

  d. Index: Sắp xếp (Ascending, Descending, None)

  e. Null: Checked or Unchecked

  f. Format: Định dạng dữ liệu hiển thị (sử dụng các ký tự qui định cho fornat của Foxpro xem bảng sau)

Format Code:	**Description:**	**Applies to data of type:**
!	Force text to uppercase.	Character only
B	Left justify .	All types
J	Right justify.	All types
I	Center justify.	All types
L	Leading zeros.	Numeric only
Z	Blank if zero.	Numeric, Date, and DateTime

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

(	Surround negative numbers with ().	Numeric only
C	Appends " CR" to positive, non-zero numbers.	Numeric only
X	Appends " DB" to negative, non-zero numbers.	Numeric only
$	Uses currency notation.	Numeric only
D	Use current SET DATE setting.	All types
E	Display as if SET DATE BRITISH was in effect.	All types
YL	Display using system Long Date format.	Date
YS	Display using system Short Date format.	Date
R	Prevents non-formatting characters in the template string from replacing the corresponding character of the field expression value. When @R is used, the template string *interleaves* instead of *overlaying*. (See example below.)	Character and Numeric

g. Input Mask: Định dạng dữ liệu nhập (sử dụng các ký tự qui định cho Input mask của Visual Foxpro - xem bảng sau)

cMask	**Description**
!	Converts lowercase letters to uppercase letters.
#	Permit entry of digits, spaces, and signs, such as the minus sign (–).
$	Displays the current currency symbol, as specified by the **SET CURRENCY** command, in a fixed position.
$$	Displays floating currency symbol that always appears adjacent to the digits in a spinner or text box.
,	Displays the current digit-grouping, or separator, symbol as set by the Regional and Language Options setting in the Windows Control Panel.
.	Displays the current decimal point character as specified by the **SET POINT** command setting. (Default is a period (.))
9	Permits entry of digits and signs.
A	Permits alphabetic characters only.
H	Prevents entry of non-hexadecimal symbols in the specified position.
L	Permits logical data only.
N	Permits letters and digits only.
U	Permits alphabetic characters only and converts them to uppercase (A - Z).
W	Permits alphabetic characters only and converts them to lowercase (a - z).
X	Permits any character.
Y	Permits the letters, Y, y, N, and n for the logical values True (.T.) and False (.F.), respectively.

h. Rule: Luật kiểm tra đối với dữ liệu của trường này (dữ liệu nhập vào cho trường này phải thoả mãn luật này)

i. Message: Thông báo khi dữ liệu nhập không thoả mãn luật qui định với trường ở Rule.

j. Default: Giá trị mặc định cho trường dữ liệu này khi thêm mới 1 bản ghi.

k. Display library, Display class: Xác định lớp gắn với ứng trường này (ví dụ nếu kiểu của trường này là Character - > Lớp mặc định được dùng là Textbox, Là Logic ->Lớp mặc

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

định được dùng là Checkbox ...), người sử dụng có thể tạo ra các lớp của riêng mình và sử dụng ở đây.

- **Tab Index:** Phần khai báo các index (chi mục) cho bảng

  a. Order: Ascending, Descening

  b. Name: Tag name

  c. Type: Kiểu

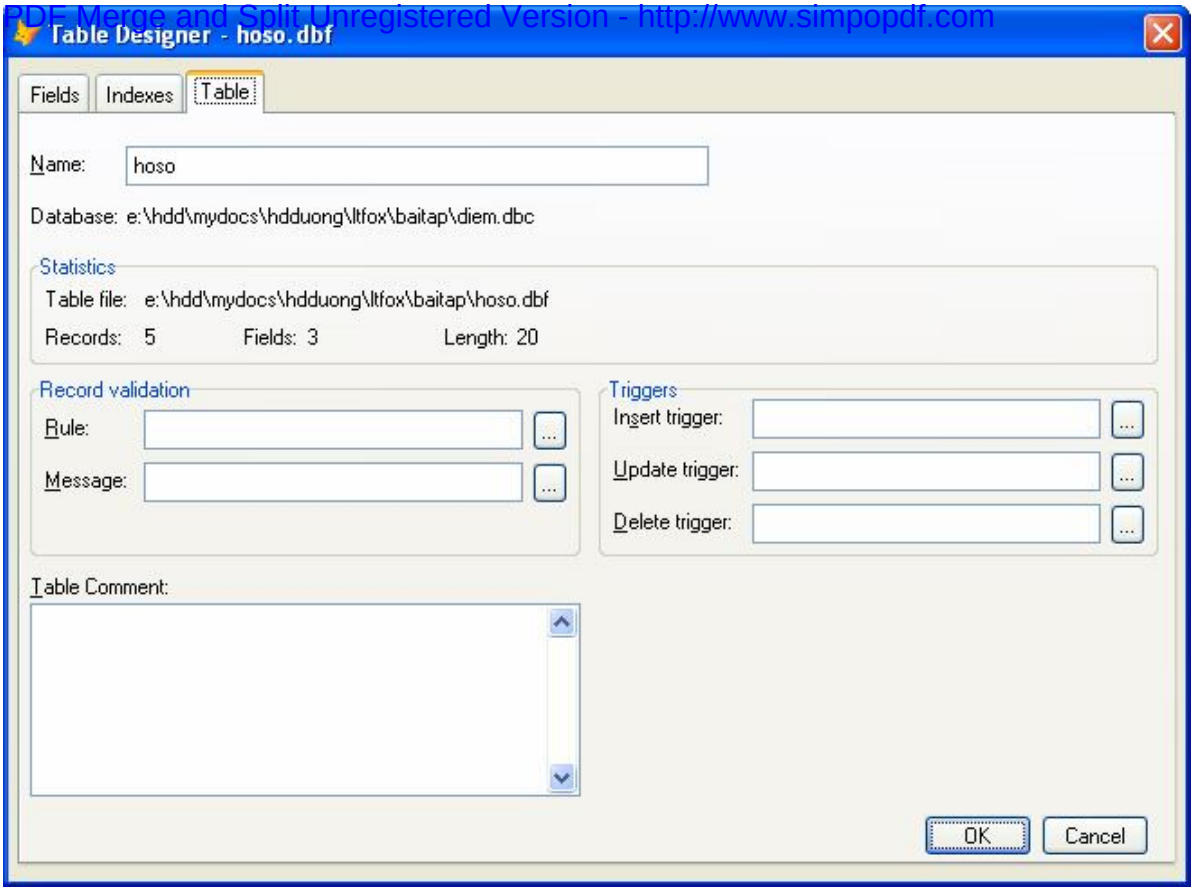
type	Description	Number allowed
Primary	Does not permit duplicate values in the specified fields or expression.	One per table
Candidate	Does not permit duplicate values in the specified fields or expression.	Many per table
Regular	Permits duplicate values in the specified fields or expression.	Many per table
Binary	Indexes records based on a valid and non-null logical expression.	Many per table
Unique	Selects a subset of records based on the first occurrence of a specified value. Included for backward compatibility.	Many per table

  d. Expression: Biểu thức index

  e. Filter: Biểu thức lọc

- **Tab Table:** Tính chất chung của bảng (hình 7)

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com



Hình 7

a. Name: Tên bảng

b. Database: Thuộc vào Database nào

c. Rule: Luật cho bản ghi. Một bản ghi chỉ được cập nhật khi thoả mã luật này (ví dụ)

d. Message: Thông báo lỗi khi luật Rule không thoả mãn.

e. Insert Trigger: Tại đây cho phép nhập vào 1 Biểu thức, biểu thức này sẽ được tính khi sảy ra sự hiện Insert đối với bảng này.

f. Update Trigger, Delete Trigger (tương tự như Insert Trigger)

Bài tập

1. Mở Project QLDIEM, tạo Database DIEMDB để lưu trữ cơ sở dữ liệu cho hệ thống quản lý điểm.

2. Tạo các bảng cho Database DIEMDB như đã nêu trong Bài tập môn học. Tự xác định các tính chất Format, Caption, Input Mask, Rule ... cho các trường của các bảng này.

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Bài 5 - Làm việc với bảng và bản ghi

I. Làm việc với bảng

 Tạo bảng

Lệnh:

CREATE TABLE <TableName>(FieldName1 Ftype1(width1) [, ...]) (xem bài 4)

CREATE [TableName] (Tương tự lệnh CREATE TABLE)

 Mở một bảng đã có

Lệnh:

USE <?|TableName> [ALIAS <AliasName>] [EXCLUSIVE]

Trong đó TableName là tên bảng cần mở hoặc khi không nhớ có thể dùng dấu ? để mở hộp thoại Open File; tùy chọn ALIAS cho phép đặt tên "vùng" làm việc với tên là <AliasName>, khi không co tùy chọn này thì tên "vùng" của bảng được mở trùng với tên bảng; tùy chọn EXCLUSIVE không cho phép nhiều người cùng sử dụng (trên mạng), tham số này thường được dùng khi cần mở bảng để sửa chữa.

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Ví dụ:

USE HOSOSV -> (tên vùng làm việc là HOSOSV)

hoặc

USE HOSOSV ALIAS SINHVIEN -> (tên vùng làm việc là SINHVIEN)

hoặc

USE ? -> Tiếp đến chọn file (bảng) cần mở

 Sửa cấu trúc của bảng

Lệnh:

MODIFY STRUCTURE

Lệnh này cho phép sửa đổi cấu trúc của bảng đang mở, giao diện này chính là giao diện CREATE TABLE (xem bài 4). Trường hợp chưa có bảng nào mở thì lệnh này sẽ mở hộp thoại Open file cho phép chọn file(bảng) cần mở và sau đó sẽ tiếp tục. Ví dụ ...

Lệnh:

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

ALTER TABLE <TableName> <ADD|ALTER|DROP> [COLUMN] FiledName1 FType1(n) ...

Lệnh này cho phép tự động sửa chữa cấu trúc của 1 bảng nào đó (thường sử dụng trong chương trình). Ví dụ

USE DIEM EXCLUSIVE

ALTER TABLE DIEM ADD COLUMN Test1 C(20)->Thêm trường Test1 là C

ALTER TABLE DIEM ALTER COLUMN Test1 N(20) ->Sửa trường Test1 thành N

ALTER TABLE DIEM DROP COLUMN Test1 ->Xóa trường Test1

- ***Đóng bảng đã mở***

Lệnh

USE && Không tham số sẽ đóng bảng hiện thời

USE IN <AliasName>

Trong đó AliasName là tên vùng một bảng nào đó đang mở, ví dụ:

USE IN DIEM

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

II. thêm, sửa, xoá các bản ghi

- ***Con trỏ bản ghi***

Con trỏ bản ghi trỏ đến bản ghi hiện thời của bảng đang mở. Khi mở bảng con trỏ bản ghi sẽ đặt ở bản ghi đầu tiên.

Lệnh:

GO TOP -> Chuyển đến bản ghi đầu tiên

GO BOTTOM -> Chuyển đến bản ghi cuối cùng

GO <n> -> Chuyển đến bản ghi thứ n

Ví dụ

```
USE DIEM
GO TOP
GO 10
GO BOTTOM
```

Lưu ý: Nếu n lớn hơn số bản ghi hiện có của bảng thì chương trình sẽ trả về thông báo Record is out of range.

Lệnh

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

SKIP <n>

Dịch chuyển con trỏ bản ghi đi n bản ghi kể từ bản ghi hiện thời (n có thể >0 hoặc <0). Ví dụ:

USE DIEM

GO 10

SKIP -5

GO 20

SKIP 5 (Sau lệnh này con trỏ ở bản ghi 30)

Lưu ý: Khi vượt quá phía đầu, phía cuối file (bảng) thì chương trình sẽ trả về thông báo: Beginning of file encountred hoặc End of file encountred.

Lệnh

LOCATE FOR <BieuThuc>

Trong đó BieuThuc là một biểu thức Logic. Lệnh này sẽ đưa con trỏ bản ghi đên bản ghi đầu tiên thoả mãn BieuThuc nếu tìm thấy. Ví dụ:

USE HOSOSV

LOCATE FOR (Hoten="Phan Văn Việt") and (malop="3A-04")

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

LOCATE FOR (Hoten= "Phan Văn Việt") and (gioi=.F.)

Lưu ý: Khi tìm thấy thì hàm FOUND() sẽ trả về giá trị TRUE, ngược lại trả về FALSE.

Lệnh

COUNTINUE

Khi sử dụng lệnh LOCATE, ta có thể dùng lệnh COUNTINUE để tìm bản ghi tiếp theo thoả mãn <BieuThuc>.

- ***Thêm mới bản ghi***

Lệnh

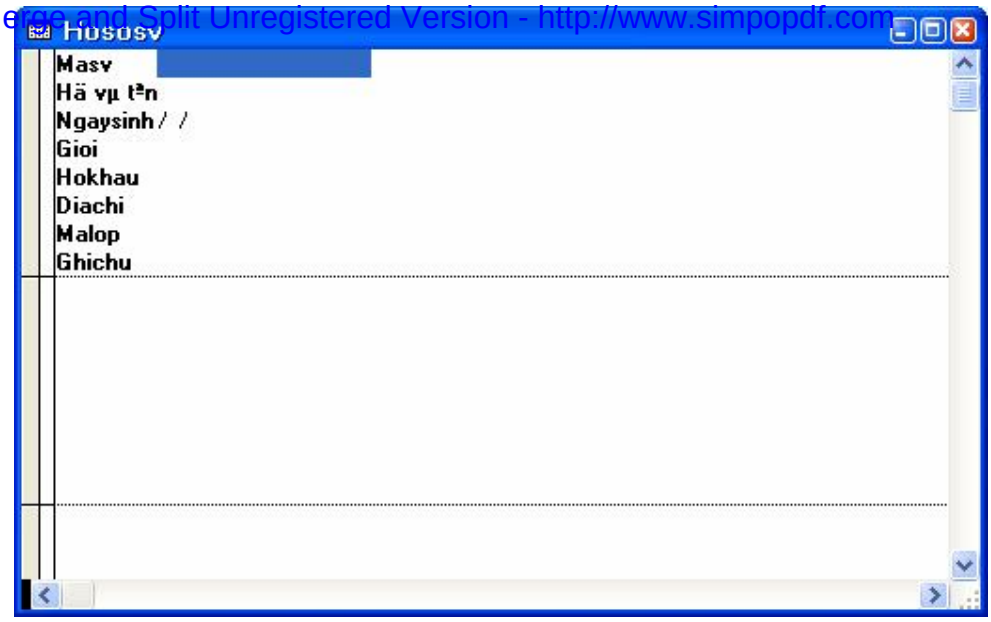
APPEND [BLANK]

Lệnh này cho phép thêm 1 bản ghi vào bảng và hiển thị giao diện cập nhật. Ví dụ

USE HOSOSV

APPEN Khi đó giao diện Appen xuất hiện như hình sau:

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com



Lưu ý: Bản ghi này chỉ thực sự được thêm vào nếu có cập nhật thông tin vào 1 trong các trường của bản ghi đó. Trường hợp nếu muốn thêm một bản ghi trống thì có thể thêm vào tham số BLANK, khi sử dụng tham số này thì bản ghi sẽ thực sự được thêm vào CSDL cả khi không cập nhật 1 trường thông tin nào (thường sử dụng trong chương trình).

Lệnh

BROWSE [FIELD <FieldName List>]

[FOR <BieuThucLogic>]

[FONT <FontName> [, FontSize]] [STYLE cFontStyle]

[TITLE <TieuDe>]

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Browse là một lệnh đa năng cho phép thực hiện hầu hết các công việc với bản ghi như xem, sửa, xóa, thêm mới ... Trong cửa sổ BROWSE, sử dụng phím Ctrl+Y để thêm 1 bản ghi trống vào bảng. Ví dụ

BROW

Tùy chọn FIELD cho phép chỉ hiển thị các trường trong <FieldName List>, ví dụ chỉ hiện thị tên và ngày sinh của HOSOSV:

BROW FIELD Hoten,ngaysinh.

Tùy chọn FOR cho phép chỉ hiển thị các bản ghi mà BieuThucLogic =TRUE, ví dụ chỉ thiển thị những người thuộc lớp “3A-04”.

BROW FOR malop=”3A-04”.

Tùy chọn FONT, STYLE cho phép chọn Font chữ dùng trong cửa sổ Brow, ví dụ sử dụng font “.vntime”:

BROW FONT “.vntime”,12

Tùy chọn TITLE cho phép đặt tên cho cửa sổ Browse là TieuDe. ví dụ

BROW FONT “.vntime”,12 TITLE “Ho so sinh vien”

Lệnh

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

INSERT INTO <TableName>(FieldName List) VALUE(Value List)

Thêm mới 1 bản ghi đồng thời xác định giá trị cho các trường của bản ghi đó. Ví dụ

USE HOSOSV

INSERT INTO HOSOSV(masv,hoten) VALUE("00001","TEST")

- ***Sửa bản ghi***

Lệnh

EDIT -> Sửa bản ghi - tương tự giao diện Append

BROWSE -> Xem mục trên

Lệnh

REPLACE <FieldName1 with BieuThuc1 [, FieldName2 with Bieuthuc2 ...]>

[Scope]

[FOR <BieuThucLogic>]

Lệnh này cho phép cập nhật giá trị của FieldName1 với giá trị của BieuThuc1, FieldName2 với giá trị của

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

BieuThuc2 .... Lệnh này thường được sử dụng trong chương trình.

Tham số Scope xác định phạm vi tác động của lệnh, có thể là:

ALL :Tất cả các bản ghi

NEXT n : n bản ghi tính từ bản ghi hiện thời

RECOR n : Bản ghi thứ n

REST : Từ bản ghi hiện thời đến cuối

Trường hợp mặc định là bản ghi hiện thời

Tùy chọn FOR cho phép thực hiện lệnh với những bản ghi thỏa mãn BieuThucLogic.

Ví dụ:

```
USE HOSOSV
REPLACE ALL ghichu with “Ghi chu ALL”
GO 5
REPLACE ghichu with “Ghi chu 5”
GO 15
REPLACE NEXT 10 ghichu with “Ghi chu Next 10”
REPLACE REST ghichu with “Ghi chu REST”
```

- ***Xoá bản ghi***

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Lệnh

DELETE [Scope]

[FOR <BieuThucLogic>]

Đánh dấu xóa các bản ghi nằm trong Scope và thỏa mãn BieuThucLogic. Việc xóa các bản ghi trong các bảng được thực hiện theo 2 bước, bước 1: là **đánh dấu xóa** các bản ghi, bước 2: xóa các bản ghi đã đánh dấu khỏi bảng, lệnh DELETE thực hiện bước 1.

Tùy chọn Scope và FOR như mục trên.

Ví dụ

USE HOSOSV

DELELE ALL FOR empty(malop)

Lệnh

PACK

Lệnh PACK thực hiện bước thứ 2 là xóa các bản ghi đã đánh dấu xóa. ví dụ PACK

Lệnh

RECALL [Scope]

[FOR <BieuThucLogic>]

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Đối với các bản ghi đã bị đánh dấu xóa (bằng lệnh DELETE) khi cần có thể khôi phục lại bằng lệnh RECALL. Tùy chọn Scope và FOR tương tự mục trên.

Lệnh

ZAP

Trường hợp đã xác định rõ cần xóa toàn bộ các bản ghi khỏi bảng thì có thể sử dụng lệnh ZAP (ZAP = DELETE +PACK).

III. sắp xếp, lọc các bản ghi

- ***Sắp xếp***

Lệnh

INDEX ON <BieuThuc> <TO IDXFile | TAG TagName>

[ASCENDING | DESCENDING]

Với các bảng dữ liệu, có thể sắp xếp các bản ghi theo một thứ tự (tiêu chuẩn nào đó) nào đó. Lệnh INDEX cho phép tạo ra các tiêu chuẩn sắp xếp (chỉ mục - index) đối với bảng, tham số <BieuThuc> chính là tiêu chuẩn sắp xếp.

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com
Tùy chọn TO IDXFile cho phép tạo ra file chỉ mục dạng IDX, với tên file là IDXFile và phần mở rộng là IDX (tạo chỉ mục dạng này sẽ không thể chọn được chiều DESCENDING và mỗi file chỉ xác định 1 tiêu chuẩn sắp xếp).

Tùy chọn TAG TagName cho phép tao ra một chỉ mục có tên là TagName trong file chỉ mục kết hợp (file chỉ mục này có tên trùng với tên bảng, và phần mở rộng là CDX, tạo chỉ mục dạng này sẽ cho phép chọn ASCENDING hoặc DESCENDING và mỗi file có thể có chứa nhiều tiêu chuẩn sắp xếp).

Ví dụ

```
USE HOSOSV
INDEX ON masv TO masvidx
BROW
INDEX ON hoten TAG hoten2 DESCENDING
BROW
```

Lệnh

SET ORDER TO [IDXFile | TagName]

Lệnh này cho phép thay đổi tiêu chuẩn sắp xếp của bảng. Trong đó: IDXFile - tên file dạng IDX (đã tạo bằng lệnh

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com
INDEX trước đó), TagName - Tên chỉ mục trong file kết hợp (file dạng CDX). Ví dụ

```
USE HOSOSV
SET ORDER TO Masv
BROW
SET ORDER TO Hoten2
BROW
SET ORDER TO masvidx
```

SET ORDER TO (không tham số) dùng để bỏ việc sắp xếp (trở lại thứ tự nhập).

- ***Lọc***

Lệnh

SET FILTER TO <BieuThucLogic>

Trường hợp chỉ cần làm việc với một số các bản ghi nào đó (lọc - filter) thì chúng ta sử dụng lệnh này, trong đó BieuThucLogic là tiêu chuẩn để lọc. Ví du

```
USE HOSOSV
SET FILTER TO malop="3A-01"
BROWSE
SET FILTER TO ngaysinh=date()
BROWS TITLE "Sinh nhat: "+ DTOC(date())
```

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

SET FILTER TO (không tham số) dùng để bỏ lọc.

IV. Làm việc với nhiều bảng

Vùng làm việc

Visual Foxpro cho phép mở nhiều bảng dữ liệu đồng thời, mỗi bảng trên một vùng làm việc (WorkArea) nhất định. Visual Foxpro cho phép mở tối đa là 255 vùng làm việc đồng thời, mỗi vùng được xác định qua tên hoặc số hiệu vùng.

Lệnh

SELECT <n | TableName| AliasName >

Lệnh này cho phép chọn vùng làm việc, trong đó: n là số thứ tự của vùng cần chuyển đén; TableName là tên bảng hoặc aliasname. Ví dụ

```
SELECTE A
USE DMLOP  ALIAS LOP
SELECTE B
USE HOSOSV ALIAS SINHVIEN
SELECTE 0
USE DIEM
```

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Trong ví dụ trên đã tạo ra 3 vùng làm việc: Vùng 1 (hoặc vùng A hoặc LOP), Vùng 2 (hoặc vùng B hoặc SINHVIEN), Vùng 3 (hoặc DIEM).

Lưu ý: Khi chọn n=0 thì Visual Foxpro sẽ tự động chọn vùng cao nhất cong trống.

- ***Liên kết giữa các bảng***

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Visual Foxpro cho phép đặt liên kết (mối "quan hệ") giữa các bảng để có thể tạo ra một bảng to hơn từ các bảng con.

Lệnh

SET RELATION TO <BieuThuc> INTO <n| TableName | AliasName>

Lệnh này cho phép đặt quan hệ từ bảng hiện thời vào bảng được mở ở vùng n (hoặc TableName hoặc AliasName), trong đó BieuThuc là tiêu chuẩn để đặt quan hệ.

Lưu ý: Bảng được mở trong vùng AliasName phải được index theo BieuThuc này rồi.

Ví dụ:

```
SELECTE A
USE DMLOP  ALIAS LOP
INDE ON ma TO malopidx
BROW
SELECT B
USE HOSOSV ALIAS SINHVIEN
SET RELATION TO malop INTO LOP
```

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

BROW

(trong ví dụ này bảng DMLOP gọi là bảng cha, bảng HOSOSV gọi là bảng con)

Khi con trỏ di chuyển trong bảng con thì con trỏ trong bảng cha cũng thay đổi tương ứng.

V. Một số hàm và lệnh khác

 Một số hàm

Hàm RECNO() - Trả về số hiệu bản ghi hiện thời

Hàm RECCOUNT() - Trả về số bản ghi hiện có của bảng

Hàm ALIAS() - Định danh của vùng hiện thời

Hàm DBF() - Tên file DBF

Hàm EOF(), BOF() - Trạng thái TRUE nếu con trỏ vượt quá Beginning và End

 Một số lệnh khác

LIST [Scope] [Field <FieldName List>] [FOR <BieuthucLogic>]

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

COUNT [TO VarName] [FOR expL1] [While expL2] [Phạm vi]

AVERAGE expN TO VarName [FOR expL1] [While expL2] [Phạm vi]

SUM <Danh sỏch biểu thức> TO <Danh sỏch biến số>
FOR expL1] [While expL2] [Phạm vi]

CALC <Danh sỏch biểu thức> TO <Danh sỏch biến số>
[FOR expL1] [While expL2] [Phạm vi]
max(expr), min(expr), aver(expN)

COPY TO <FileName> [FOR expL1] [While expL2] [Phạm vi]

COPY STRUCTURE TO <FileName> [EXTENDED]

APPEND FROM <FileName> [FOR <ExpL>]

Bài tập

1. Cập nhật dữ liệu cho tất cả các bảng của Database DIEMDB (đã tạo trong bài 4, dữ liệu là danh sách lớp).
2. Thực hiện các ví dụ trong bài học.

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Bài 6 - Tạo queries và views

I. Thiết lập quan hệ giữa các bảng

Trên thực tế, các bảng của CSDL thường có quan hệ nào đó với nhau. Mối quan hệ giữa các bảng có thể thiết lập qua lệnh (Lệnh SET RELATION - xem bài 5) - mối quan hệ tạm thời , hoặc trong quá trình thiết kế cơ sở dữ liệu - mối quan hệ bền vững. Trong phần này chúng ta chỉ xét mối quan hệ bền vững.

Để đặt quan hệ giữa các bảng thì việc đầu tiên cần làm đó là xác định các Index cho bảng. Việc xác định các Index cho bảng có thể thực hiện qua lệnh (lệnh INDEX - xem bài 5) hoặc có thể xác định khi thiết kế bảng trong giao diện Table Designer.

- ***Tạo index trong Table Designer*** (Xem bài 4)
- ***Thiết lập quan hệ bền vững giữa các bảng trong CSDL***

Mối quan hệ bền vững được tạo trong quá trình thiết kế CSDL, và được lưu lại trong file csdl (DBC) và có những đặc điểm sau:

- Được tự động sử dụng như các kết nối mặc nhiên (khi tạo Queries và Views)
- Được thể hiện bằng các đường nối trong Database Designer.
- Được thể hiện trong môi trường Data
- Dùng để lưu thông tin về mối quan hệ toàn vẹn.

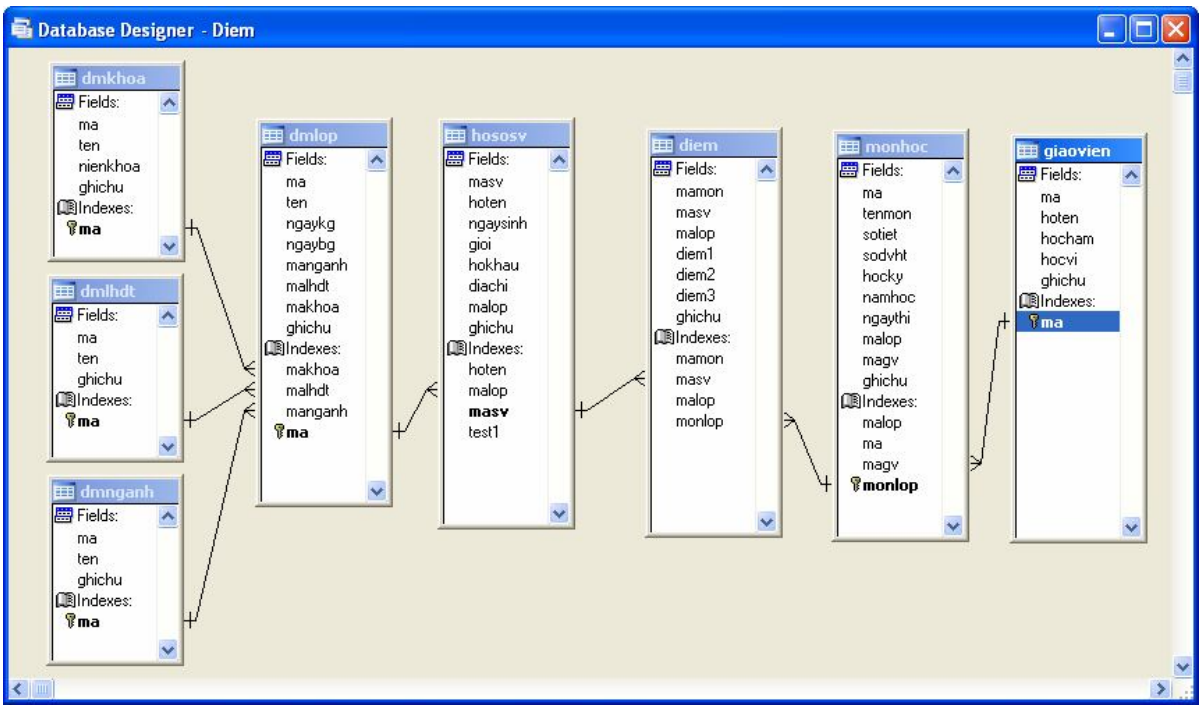
Không giống như các mối quan hệ tạm thời được tạo bằng lệnh SET RELATION, mối quan hệ bền vững không không cần phải tạo lại mỗi khi sử dụng đến các bảng dữ liệu. Tuy nhiên bởi mối quan hệ bền vững không điều khiển các con trỏ bản ghi giữa các bảng dữ liệu nên mối quan hệ tạm thời vẫn rất cần thiết khi phát triển ứng dụng.

Các bước thiết lập quan hệ bền vững

1. Mở Project
2. Chọn Tab Data
3. Chọn mục Database và mở rộng (expand)
4. Chọn Database cụ thể cần làm việc (DIEMDB)
5. Chọn mục Modify, khi đó giao diện Database Designer xuất hiện.
6. Trong giao diện Database Designer để tạo quan hệ giữa 2 bảng chỉ cần kéo-thả trường Index (Primary) từ bảng cha đến trường Index (Regular) của bảng con. ở đây cần lưu ý việc tạo Index cho các trường trước khi thiết lập quan hệ giữa các bảng.

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Sau khi thiết lập xong quan hệ giữa các bảng ta có giao diện Database Designer có dạng như hình 6.1.



Hình 6.1 Quan hệ giữa các bảng trong CSDL

Đường nối thể hiện quan hệ giữa các bảng.

- ***Thiết lập quan hệ toàn vẹn giữa các bảng***

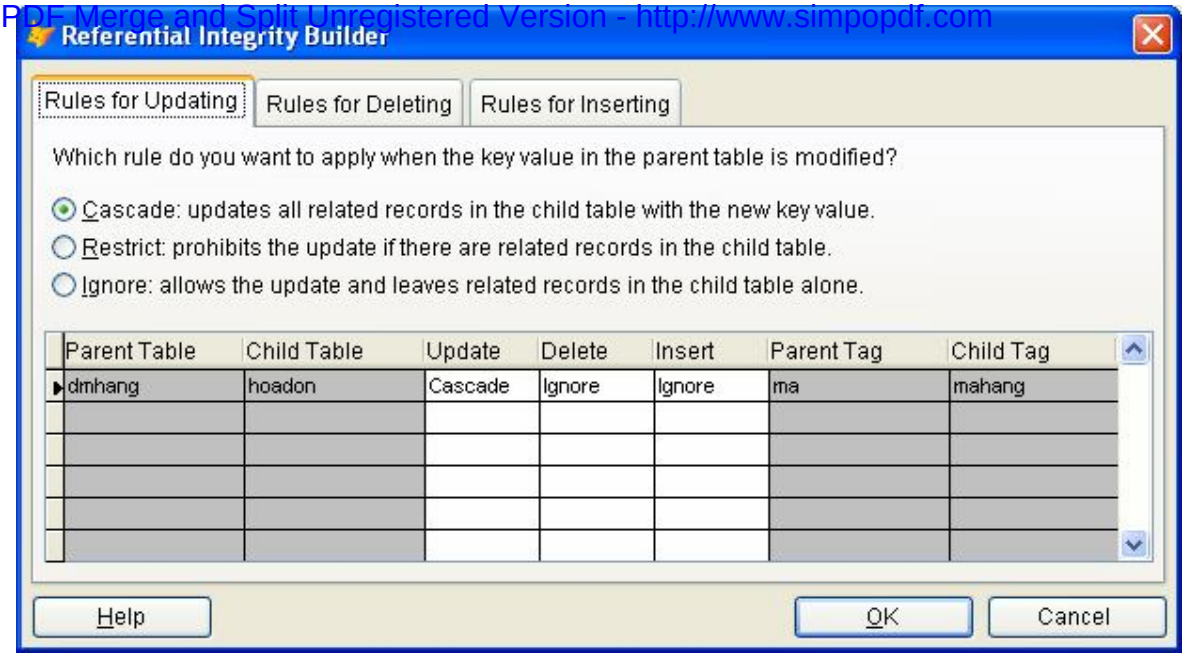
Thực chất là xác định các qui tắc ứng xử giữa các bảng có quan hệ với nhau. Các qui tắc đó sẽ kiểm xoát các việc như thay đổi (Update), Thêm (Insert), Xoá (Delete) sao cho vẫn đảm bảo mối quan hệ giữa các bảng này không bị vi phạm.

Để thiết lập quan hệ toàn vẹn giữa các bảng, nháy đúp chuột trái vào đường liên kết bất kỳ giữa 2 bảng, tiếp đến chọn lệnh Referential Integrity như hình 6.2

Hình 6.2

Khi chọn Referential Integrity giao diện này xuất hiện như hình 6.3

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com



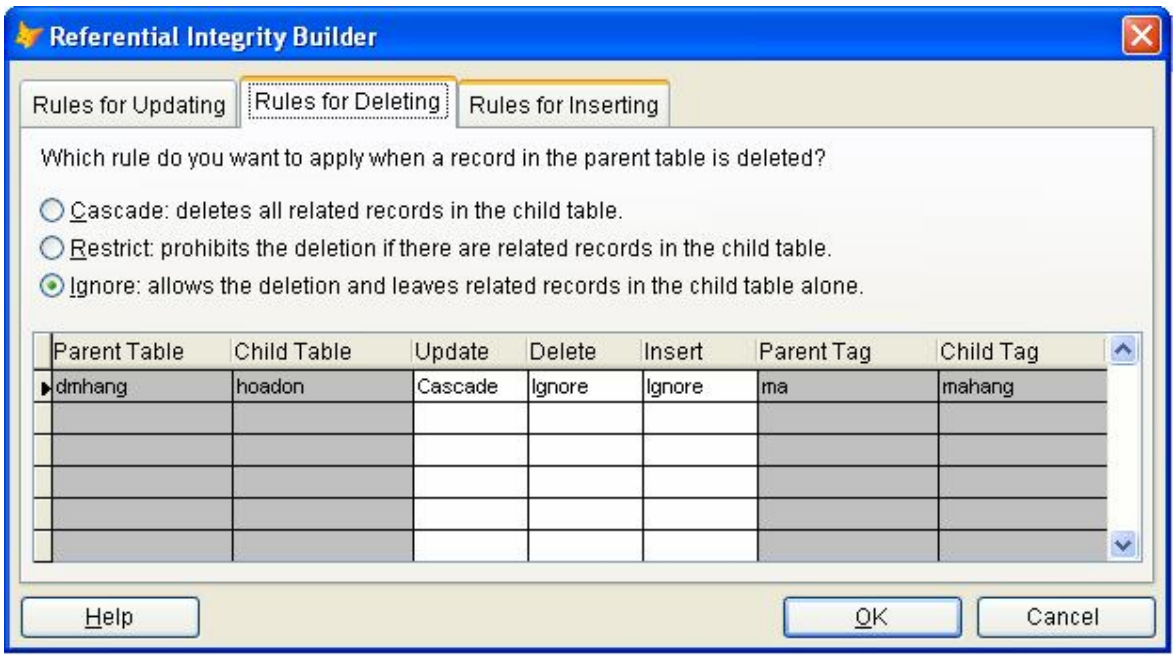
Hình 6.3 Giao diện Referential Integrity, Rules For Updating

Trong đó:

- Dòng (trong grid phía dưới): Thể hiện các quan hệ hiện có của CSDL, các thông tin của một quan hệ gồm có: Bảng cha (Parent Table), Bảng con (Child Table), Biểu thức quan hệ trên bảng cha (Parent Tag), Biểu thức quan hệ trên bảng con (Child Tag).
- Tab Rules for Updating: cho phép thiết lập qui tắc ứng xử giữa 2 bảng cha/con của một quan hệ nào đó (phía dưới) khi sửa đổi. Các qui tắc ứng xử ở đây là:
  - Ignore: Cho phép sửa các bản ghi liên quan trên bảng con một cách độc lập (không phụ thuộc vào bảng cha), đây là giá trị mặc định. Ví dụ
  - Restrict: Không cho phép sửa đổi các bản ghi trong bảng cha nếu tồn tại những bản ghi liên quan trong bảng con. Ví dụ
  - Cascade: Sửa tất cả những bản ghi có quan hệ trong bảng con với giá trị khoá mới tương ứng của bảng cha. Ví dụ
- Tab Rules for Deleting: cho phép thiết lập qui tắc ứng xử giữa 2 bảng cha/con của một quan hệ nào đó (phía dưới) khi xoá. Các qui tắc ứng xử ở đây là:
  - Ignore: Cho phép xoá các bản ghi liên quan trên bảng con một cách độc lập (không phụ thuộc vào bảng cha), đây là giá trị mặc định. Ví dụ
  - Restrict: Không cho phép xoá các bản ghi trong bảng cha nếu tồn tại những bản ghi liên quan trong bảng con. Ví dụ

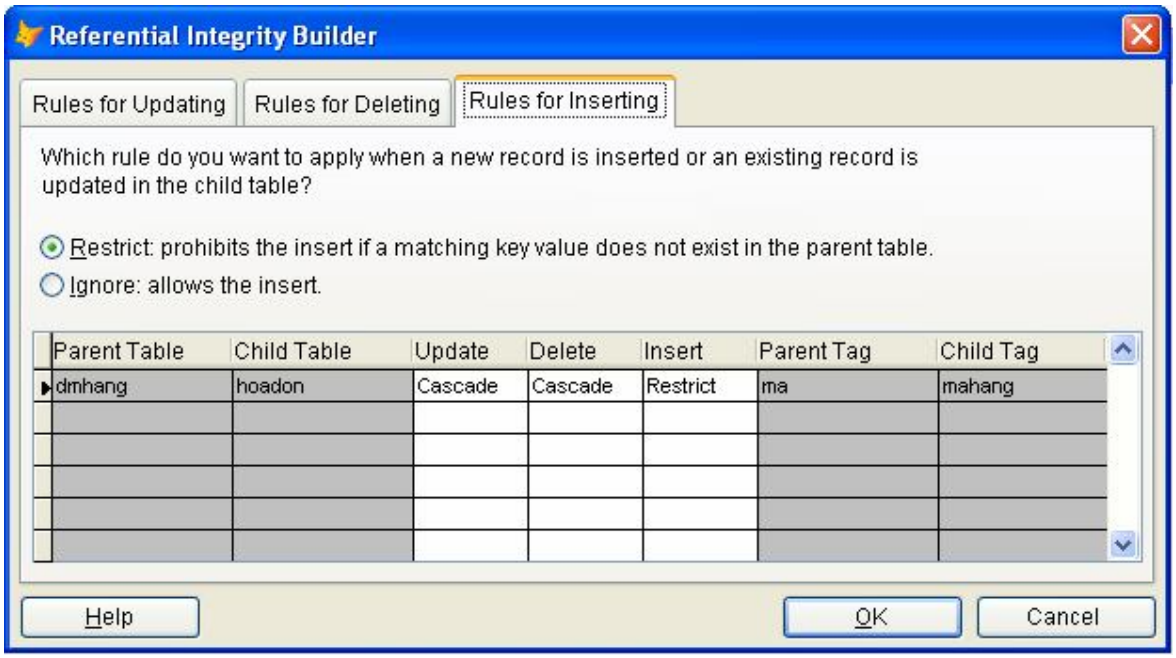
Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

- Cascade: Xoá tất cả những bản ghi có quan hệ trong bảng con tương ứng với bản ghi bị xoá của bảng cha. Ví dụ



Hình 6.4 Rules For Deleting

- Tab Rules for Inserting: cho phép thiết lập qui tắc ứng xử giữa 2 bảng cha/con của một quan hệ nào đó (phía dưới) khi thêm mới. Các qui tắc ứng xử ở đây là:



Hình 6.4 Rules For Inserting

- Ignore: Cho phép thêm các bản ghi vào bảng con một cách độc lập (không phụ thuộc vào bảng cha), đây là giá trị mặc định. Ví dụ

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

- Restrict: Không cho phép thêm các bản ghi vào bảng con nếu không tồn tại một bản ghi tương ứng trong bảng cha. Ví dụ

II. Tạo các queries

Query: Công cụ truy vấn dữ liệu. Có thể tạo ra query với nhiều lý do khác nhau như: để tổ chức thông tin cho Report, để trả lời nhanh một câu hỏi .... Ví dụ, với bảng DIEM khi biết mã sinh viên và cần xem sinh viên đó có tên là gì? Bảng điểm của một môn học nào đó. ..

- ***Tạo mới một Query sử dụng Wizard***

1. Trong Project Manager chọn Tab Data
2. Chọn Queries
3. Chọn New
4. Chọn Query Wizard
5. Chọn kiểu Query Wizard cần tạo, có các kiểu là Cross-Tab, Graph và Query.

Ví dụ 1: Tạo một Cross Tab Query

Giả sử cần xác định xem số lượng thí sinh thi trượt lần 1 các môn học theo lớp có số lượng là như thế nào? Số liệu cần đưa ra dạng sau:

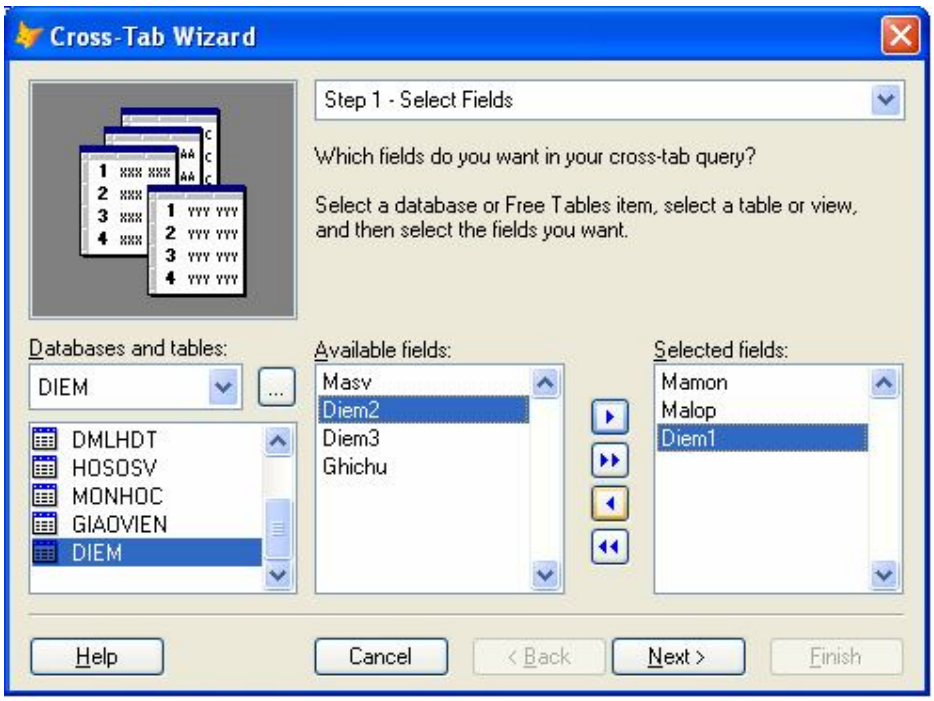
Lớp /môn	MH1	MH2	...	...	...	MHn
3A-04	4	7	...	...	...	...
3B-04	3	5	...	...	...	...
...	...	...	...	...	...	...

Khi đó cần sử dụng một Cross-Tab.

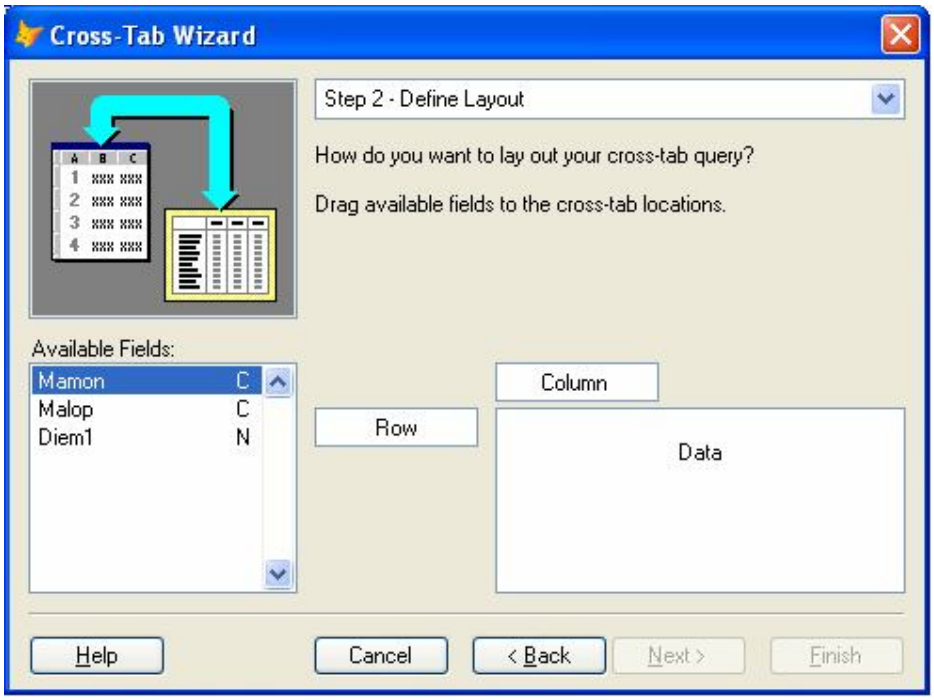
1. Chọn Cross-Tab Wizard, chọn OK.

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

2. Step 1 - Select Fields: Chọn Database, Tables và Fields (Sử dụng lệnh >, >> để chọn các trường cần thiết). Khi đã chọn xong các trường, chọn Next để qua bước 3.



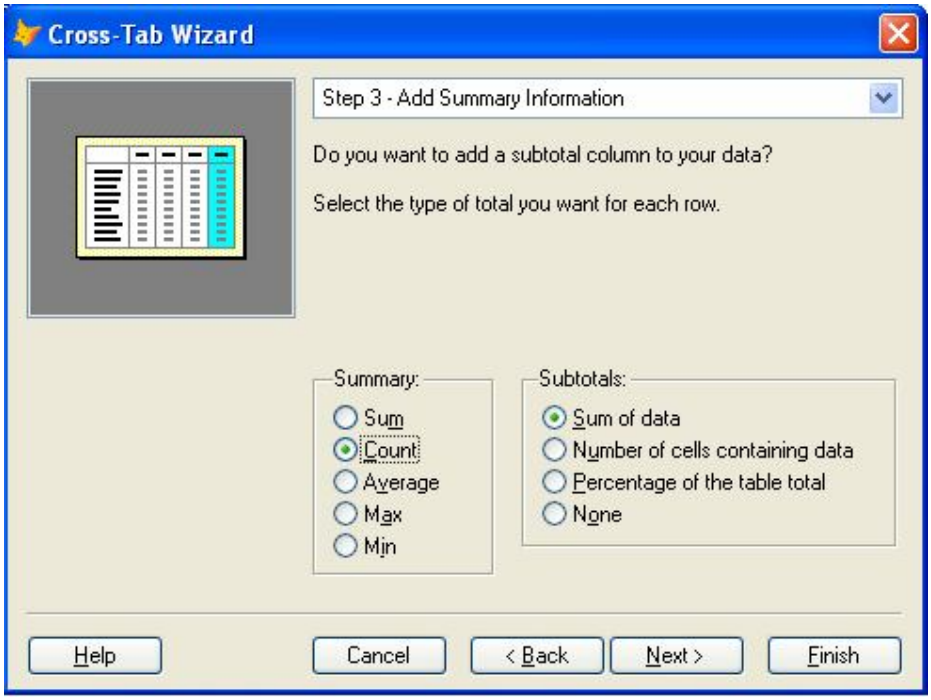
3. Step 2 - Define Layout:



Bước này cần định nghĩa 3 đối tượng cho Cross-Tab đó là: Row (trong ví dụ này là Malop), Column (Mamon) và Data (Diem1) bàng cách nhấp-thả (Drag) các trường vào các ô (Row, Column, Data) tương ứng. Chọn Next để tiếp tục

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

4. Step 3 - Add summary information: Chọn các hàm thống kê đối với đối tượng Data, các hàm này có thể là: Sum, Countm Average, Max, Min, trong ví dụ này thì Summary chọn là Count. Tuỳ chọn Subtotal cho phép thêm vào cột cuối để tính subtotal cho mỗi dòng. Tiếp đến chọn Finish và nhập vào tên file Query để hoàn tất công việc (hoặc chọn Next sẽ chuyển đến bước 4 Step 4 - Finish).



Trong cửa sổ Project Manager chọn Query vừa tạo rồi chọn lệnh Run - > Được kết quả thể hiển của Query.

Ví dụ 2: Tạo một Graph Form bằng Graph Wizard

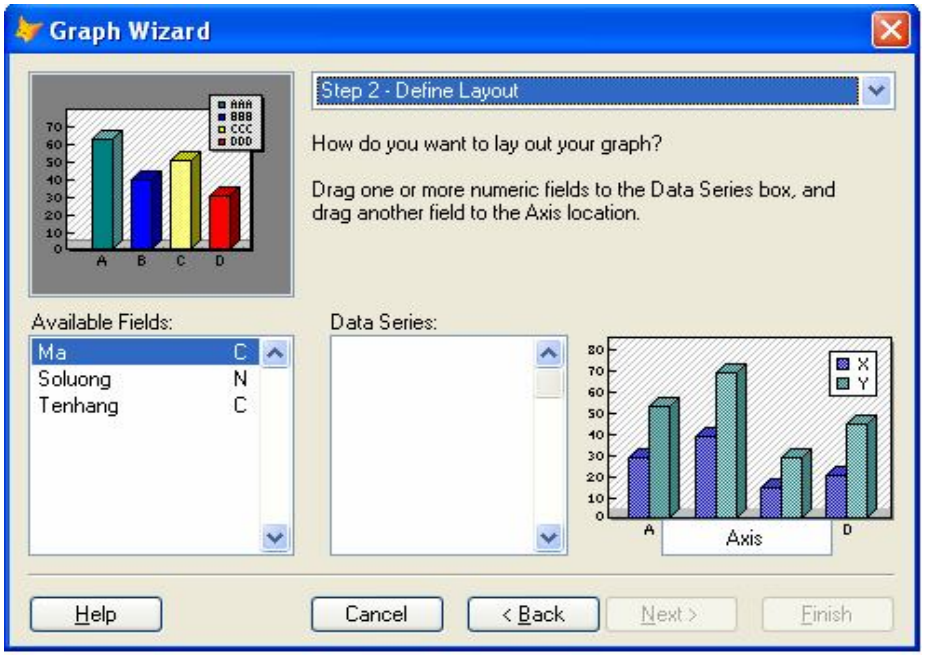
Giả sử cần thể hiện số lượng sản phẩm của các mặt hàng hiện có trong kho dạng biểu đồ. Khi đó sử dụng Graph Wizard như sau:

1. Chọn Graph Wizard, chọn OK.

2. Step 1 - Select Fields: Như ví dụ trên

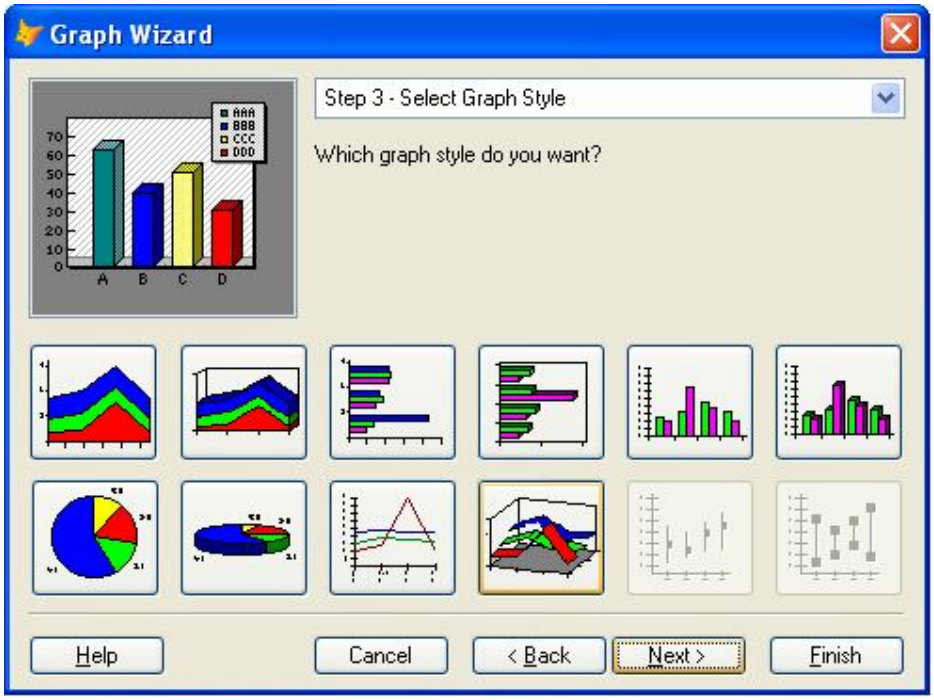
Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

3. Step 2 - Define Layout:



Bước này cần định nghĩa 2 đối tượng cho là: Data series - Trường số (trong ví dụ này là Soluong) và Axis - Trong ví dụ này là Ma. Chọn Next để tiếp tục (lưu ý là Next và Finish chỉ cho phép khi đã chọn đủ Data series và Axis).

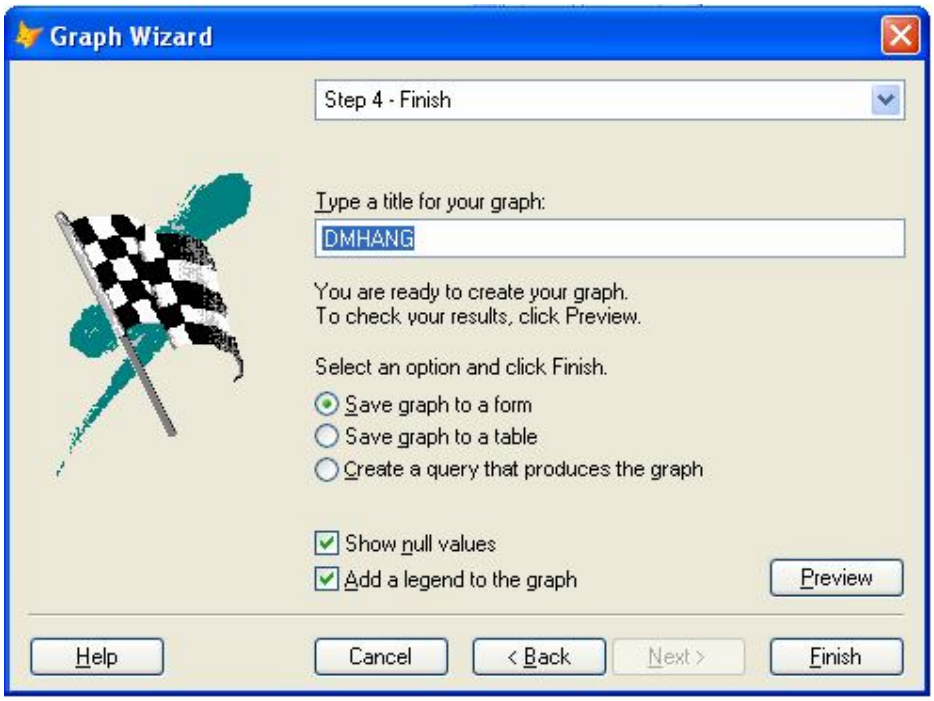
4. Step 3 - Select Graph Style



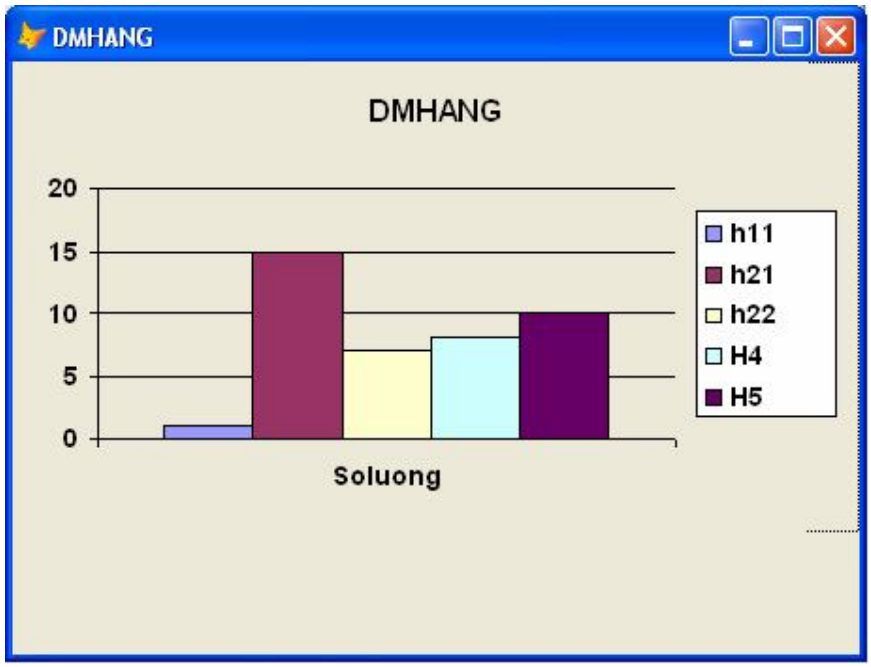
Bước này cho phép chọn kiểu biểu đồ, trong ví dụ này chọn biểu đồ cột (thứ 6). Chọn Next để tiếp tục

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

5. Step 4 - Finish: Chọn Finish rồi nhập vào tên form cần lưu (Kết quả của Wizard này là một Form).



Trong cửa sổ Project Manager chọn Form vừa tạo rồi chọn lệnh Run - > Được kết quả thể hiển của Query. Với ví dụ 2 ta có form DMHANG dạng biểu đồ như hình sau:

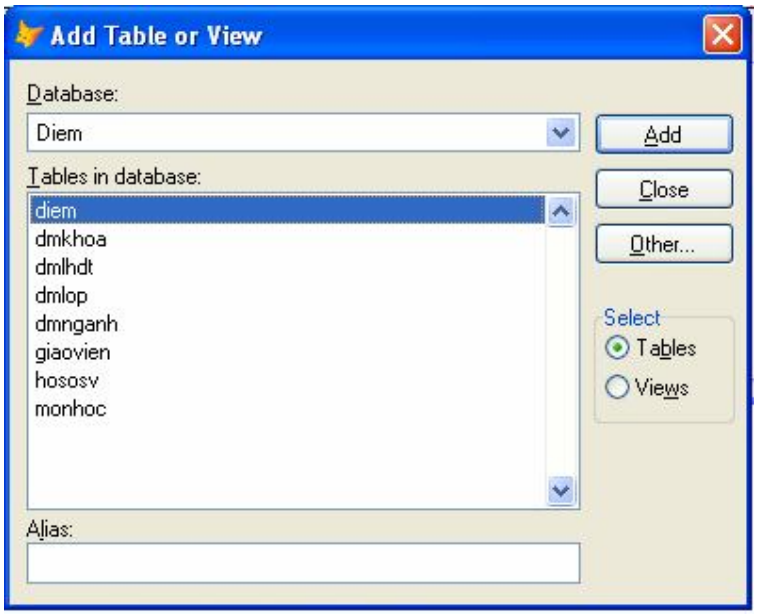


- ***Tạo mới một Query sử dụng Queries Designer***

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Giả sử cần xem bảng điểm một môn học của một lớp nào đó, bảng điểm bao gồm Masv, Hoten, Ngaysinh và điểm. Với yêu cầu này có thể sử dụng Query để lấy thông tin theo các bước sau:

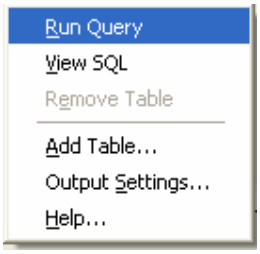
1. Trong Project Manager chọn Tab Data
2. Chọn Queries
3. Chọn New
4. Chọn New Query
5. Chọn Table hoặc View cần dùng cho Query như hình sau. Trong ví dụ này chọn Table Diem (rồi Add) và Hososv (rồi Add) (bước chọn Bảng này có thể thực hiện về sau). Chọn Close để đóng cửa sổ Add Table or View



6. Tiếp đến giao diện Query xuất hiện như hình 6.6

Giao diện Query Designer:

- **<u>Phía trên:</u>** Thể hiện các bảng (hoặc View) sử dụng trong Query. Nháy chuột phải vào phần này ta có Popup menu như sau:



Run Query: Thực hiện (chạy) query này

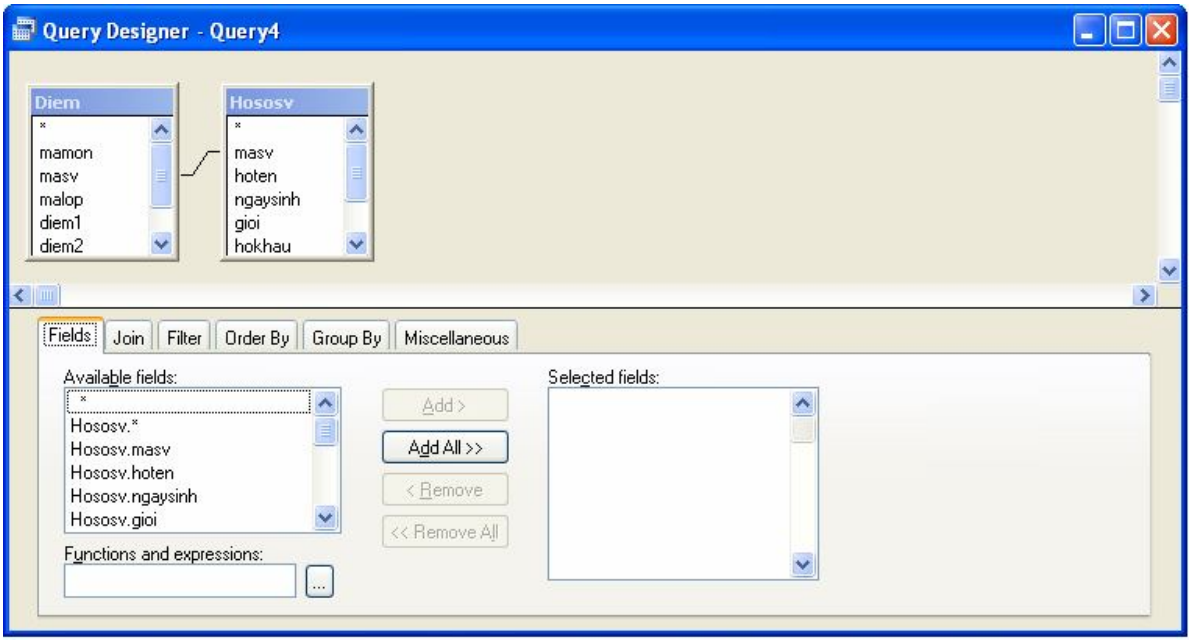
View SQL: Xem query ở dạng SQL

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Remove Table: Loại bỏ bảng đã chọn khỏi Query (khi chọn vào bảng)

Add Table: Show giao diện Add Table or View

Output Setting: Tuỳ chọn đầu ra cho query (xem sau).



Hình 6.6 - Query Designer

Đường nối thể hiện quan hệ giữa các bảng.

- **<u>Phía dưới:</u>** Các tuỳ chọn cho thiết kế Query

Tab Fields: Chọn các trường cho query.

- Available Fields: Các filed có thể chọn cho query.
- Selected Fields: Các trường đã chọn.
- Lệnh Add: Thêm trường đã chọn từ Available Fields vào Selected Fields.
- Lệnh Add All: Thêm tất cả các trường từ Available Fields vào Selected Fields.
- Lệnh Remove: Loại bỏ 1 trường đã chọn trong Selected Fields.
- Lệnh Remove All: Loại bỏ tất cả các trường đã chọn trong Selected Fields.
- Mục Functions and Expressions: Khai báo 1 biểu thức và có thể thêm (Add) biểu thức này vào Selected fields (và nó có vai trò như 1 trường, dùng khai báo AS <TenTruong> để đặt tên cho biểu thức này).

Tab Joint: Thiết lập các thuộc tính về kết nối giữa 2 bảng.

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Các bảng dùng cho 1 query thường có quan hệ nào đó với nhau. Khi truy xuất dữ liệu, kiểu liên kết (Joint Type) có vai trò quan trọng trong việc ứng xử giữa các bảng. Ví dụ có 2 bảng Dmhang và Hoadon được dùng trong 1 query và có số liệu như hình sau:

Dmhang

Ma	Tenhang	Soluong
H1	Hang hoa 1	1.00
H2	Hang hoa 2	15.00
H3	Hang hoa 3	7.00
H4	Hang hoa 4	8.00
H5	Hang hoa 5	10.00

Hoadon

Sohd	Mahang	Soluong	Ghichu
1	H1	2	
2	H1	2	
3	H2	3	
4	H1	5	
5	H3	3	
6	H3	2	
7	H10	4	

Bảng Dmhang có chứa các bản ghi H4, H5, các bản ghi này không có trong Hoađon, và ngược lại trong Hoadon có mặt hàng H10 là loại hàng chưa có trong Dmhang. Như vậy khi ghép 2 bảng này mặt hang H4, H5 có số hóa đơn là bao nhiêu? Hóa đơn 7 mặt hàng H10 thì tên hàng là gì, đơn giá là bao nhiêu? Để giải quyết vấn đề này chúng ta sẽ xét các Joint sau đây.

- Left Table: Bảng phía trái (bảng cha)
- Right Table: Bảng phía phải (bảng con)
- Joint Type: Có thể 1 trong các loại sau:
  - Inner joint: Chỉ trả về các bản ghi có thể so khớp được với nhau trên 2 bảng (khi đó các bản ghi H4, H5 và H7 trong cả 2 bảng sẽ không được trả về)
  - Left Outer Joint:
  - Right Outer joint:
  - Full Joint:

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Bài tập

1. Tạo chỉ mục cho các bảng như đã chỉ ra trong bài 1.

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Bài 7 - lập trình trong Visual Foxpro

I. Tạo và chạy một chương trình

- Visual Foxpro làm việc ở hai chế độ:
  - Cửa sổ lệnh - Các lệnh được đưa vào cửa sổ lệnh và thực thi.
  - Chương trình - Tập các lệnh được thực hiện theo một thứ tự nào đó.
- Chương trình trong Visual Foxpro được lưu trong các file có phần mở rộng là PRG.

- ***Tạo một chương trình***

Lệnh (từ cửa sổ lệnh)

MODIFY COMMAND <TenChuongtrinh> ↵

Trong đó: <Tenchuongtrinh> là tên chương trình

Khi đó xuất hiện cửa sổ soạn thảo cho phép soạn thảo văn bản chương trình.

Kết thúc soạn: Nhấn Ctrl+W để ghi lại.

Sửa chương trình: cũng dùng lệnh MODI COMM <tenchuongtrinh>

Một chương trình đơn giản

Chương trình trong Visual Foxpro không qui định theo một cấu trúc nhất định.

Xét một ví dụ đơn giản sau đây:

```
* Phần 1 : Bắt đầu
*                Dùng các lệnh SET để thiết lập môi trường làm việc ...
SET CENTURY ON          && Hiển thị năm dạng yyyy
SET DATE FREN           && Hiển thị ngày tháng dạng dd/mm/yyyy
SET EXCLUSIVE  ON       && Đặt chế độ mở bảng dạng dùng riêng

* Phần 2:       Chuẩn bị dữ liệu
*               Mở các bản cần thiết, đặt quan hệ ....
CLOSE ALL
SELECT 0
USE Hososv ALIAS SINHVIEN
SET ORDER TO MASV
SELECT 0
USE diem
SET RELATION TO masv INTO SINHVIEN

* Phần 3: Chương trình chính
*               Xử lý các yêu cầu đặt ra
BROWSE          FIELDS masv,SINHVIEN.HOTEN,MAMON,DIEM1        ;
                FOR malop="3H-04" AND mamon="GT101"         ;
                TITLE "Bảng điểm môn GT101 - Lớp 3H-04" ;
                NOEDIT
```

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

```
* Phần 4: Kết thúc
*                Đóng các bảng đả mở, thiết lập lại chế độ làm việc ...
USE IN SINHVIEN
USE IN DIEM
RETURN
```

Như trên ta thấy chương trình được tổ chức thành 4 phần:

Phần 1: Bắt đầu - chuẩn bị môi trường (sử dụng các lệnh SET)

Phần 2: Chuẩn bị dự liệu (mở các bảng, đặt quan hệ ...)

Phần 3: Thân chương trình (xử lý các yêu cầu đặt ra)

Phần 4: Kết thúc (Đóng các bảng đã mở, trả lại trạng thái ...)

- ***Chạy (thực thi) một chương trình***

Lệnh (thực hiện từ cửa sổ lệnh)

DO <TenChuongTrinh> ↵

II. lệnh có cấu trúc

1. Lệnh IF .. ENDIF

Thực hiện có điều kiện một tập các lệnh dựa vào giá trị của biểu thức logic

Cú pháp:

```
IF  <BieuThucLG> [THEN]
        Lệnh1
[ELSE
        Lệnh2]
ENDIF
```

Trong đó:

- BieuThucLG - Biểu thức logic
- BieuThucLG = TRUE Thì các **Lệnh1** được thực hiện
- BieuThucLG = FALSE và sử dụng mệnh đề ELSE khi đó các **Lệnh2** được thực hiện.

Ví dụ 1: Kiểm tra xem có ai sinh nhật vào hôm nay hay không? Nếu có thì hiển thị họ tên người đó.

```
SET CENTURY ON
SET DATE FREN
SET EXCLUSIVE  ON

CLOSE ALL
```

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

```
SELECT 0
USE HOSOSV ALIAS HSSV
LOCA FOR DAY(DATE())=DAY(Ngaysinh) and MONT(DATE())=MONT(Ngaysinh)
IF Found() THEN
  WAIT WIND "Hom nay:"+DTOC(Date()) "la sinh nhat cua:"+HSSV.hoten
ELSE
  WAIT WIND "Khong co ai sinh nhat vao hom nay:"+DTOC(Date())
ENDIF
USE IN HOSOSV
RETURN
```

Chú ý: Có thể sử dụng cấu trúc IF ENDIF lồng nhau.

```
IF BT1 THEN                    && IF thứ 1
    ....
    IF BT2 THEN        && IF thứ 2
        ....
    ELSE
        ....
    ENDIF              && Kết thúc IF thứ 2
ELSE
    ....
ENDIF                          && Kết thúc IF thứ nhất.
```

2. Lệnh DO CASE ... ENDCASE

Kiểm tra nhiều điều kiện logic, và thực hiện tập lệnh đầu tiên mà biểu thức logic có giá trị TRUE.

Cú pháp:

```
DO CASE
    CASE BieuThucLogic1
        Lệnh 1
    CASE BieuThucLogic2
        Lệnh 2
    ...
    CASE BieuThucLogicN
        Lệnh N
    [ OTHERWISE
        Lệnh N+1 ]
ENDCASE
```

Trong đó:

- BieuThucLogicK - Biểu thức logic

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

- Thực hiện lệnh: Kiểm tra từ BieuThucLogic1 đến N nếu tồn tại một BieuThucLogicK =TRUE thì chương trình sẽ thực hiện các câu **Lệnh K** và kết thúc, trường hợp tất cả các BieuThucLogic=FALSE và sử dụng mệnh đề OTHERWISE thì các **Lệnh N+1** được thực hiện.

Ví dụ 2: Tính điểm trung bình chung và tính xếp loại của một sinh viên nào đó.

```
SET TALK OFF
CLOSE ALL
SELECT 0
USE monhoc
SET ORDER TO Ma
SELECT 0
USE diem
SET RELATION TO mamon INTO Monhoc
SUM monhoc.sodvht*diem.diem1,monhoc.sodvht TO tongdiem,tongdvht FOR masv="SV-0624
"
DTBC=tongdiem/tongdvht
DO CASE
	CASE dtbc>9
		WAIT WINDOW "Xep loai: Xuat sac"
	CASE dtbc>8
		WAIT WINDOW "Xep loai: Gioi"
	CASE dtbc>7
		WAIT WINDOW "Xep loai: Kha"
	CASE dtbc>6
		WAIT WINDOW "Xep loai: Trung Binh Kha"
	CASE dtbc>5
		WAIT WINDOW "Xep loai: Trung Binh"
	OTHERWISE
		WAIT WINDOW "Xep loai: Yeu"
ENDCASE
USE IN monhoc
USE IN diem
RETURN
```

3. Lệnh DO WHILE ... ENDDO

Thực hiện một tập các lệnh trong vòng lặp có điều kiện.

Cú pháp:

DO WHILE <BieuThucLogic>

Lệnh 1

[LOOP]

[EXIT]

Lệnh 2

ENDDO

Trong đó:

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

- BieuThucLogic - Biểu thức logic

- Thực hiện: Nếu BieuThucLogic=TRUE sẽ thực các **Lệnh 1, Lệnh 2**, khi đến ENDDO sẽ quay trở lại kiểm tra BieuThucLogic; trường hợp BieuThucLogic=FALSE sẽ kết thúc lệnh.

  Lệnh LOOP cho phép (ngay lập tức) trở lại bắt đầu vòng lặp (Kiểm tra BieuThucLogic), bỏ qua các lệnh giữa LOOP và ENDDO. Lệnh LOOP có thể đặt vào bất cứ đâu giữa DO WHILE và ENDDO.

  Lệnh EXIT cho phép (ngay lập tức) kết thúc lệnh DO WHILE ...ENDDO và bỏ qua các lệnh giữa EXIT và ENDDO. Lệnh EXIT có thể đặt vào bất cứ đâu giữa DO WHILE và ENDDO.

Ví dụ 3: Tìm xem trong danh sách sinh viên có bạn Nữ (Gioi=.T.) nào có điểm trung binhg chung đạt >=8.0?

```
SET TALK OFF
CLEAR
CLEAR ALL
CLOSE ALL
SELECT 0
USE monhoc
SET ORDER TO MONLOP   && MA+MALOP
SELECT 0
USE diem
SET RELATION TO mamon+malop INTO Monhoc
SELECT 0
USE Hososv
GO TOP
DO WHILE !EOF()
        IF hososv.gioi=.F. Then
                SKIP
                LOOP
        ENDIF
        SELECT diem
        SUM monhoc.sodvht*diem.diem1,monhoc.sodvht ;
           TO tdiem,tdvht FOR masv=Hososv.masv
        dtbc=tongdiem/tongdvht
        IF dtbc>=8 THEN
                WAIT WINDOW "SV: "+ hososv.hoten +" DTBC: "+ STR(dtbc)
                EXIT
        ENDIF
        SELECT hososv
        SKIP
ENDDO
USE IN hososv
USE IN monhoc
USE IN diem
RETURN
```

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

4. Lệnh FOR ... ENDFOR

Thực hiện một tập các lệnh với số lần lặp xác định.

Cú pháp:

FOR Bien = initValue TO FinalValue [STEP K]

Lệnh 1

[LOOP]

[EXIT]

Lệnh 2

ENDFOR

Trong đó:

- **Bien** - Đóng vai trò như một biến đếm, initValue - Giá trị đầu, FinalValue - Giá trị cuối.
- Thực hiện: Thực hiện các lệnh giữa FOR và ENDFOR khi biến **Bien** có giá trị từ initValue đến FinalValue.

  Tham số STEP K qui định bước thay đổi của biến **Bien** bằng K (K có thể âm), mặc định (khi không thên tùy chọn STEP K) K =1.

  Lệnh LOOP cho phép (ngay lập tức) trở lại bắt đầu vòng FOR, bỏ qua các lệnh giữa LOOP và ENDFOR. Lệnh LOOP có thể đặt vào bất cứ đâu giữa FOR và ENDFOR.

  Lệnh EXIT cho phép (ngay lập tức) kết thúc lệnh FOR ...ENDFOR và bỏ qua các lệnh giữa EXIT và ENDFOR. Lệnh EXIT có thể đặt vào bất cứ đâu giữa FOR và ENDFOR.

Ví dụ 4: Tính điểm trung bình chung học kỳ 1 năm 2004-2005 của lớp 3E-04.

```
SET TALK OFF
SET SAFETY OFF
CLEAR
CLEAR ALL
CLOSE ALL
SELECT 0
USE monhoc
SET ORDER TO MONLOP   && MA+MALOP
SELECT 0
USE diem
SET RELATION TO mamon+malop INTO Monhoc
SET FILTER TO monhoc.hocky=1 AND monhoc.namhoc=2004
SELECT 0
USE Hososv
COPY TO c:\temp\HOSOLOP FOR malop="3E-04"
USE IN hososv
SELECT 0
USE c:\temp\HOSOLOP ALIAS Hososv EXCLUSIVE
```

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

```
ALTER table c:\temp\HOSOLOP ADD COLUMN Dtbc N(5,2)
n=RECCOUNT()
FOR i=1 TO n
        SELECT diem
        SUM monhoc.sodvht*diem.diem1,monhoc.sodvht ;
                TO tongdiem,tongdvht FOR masv=Hososv.masv
        SELECT hososv
        REPLACE dtbc WITH IIF(tongdvht>0,tongdiem/tongdvht,0)
        SKIP
ENDFOR
SELECT Hososv
BROWSE FIELDS masv,hoten,ngaysinh,dtbc ;
         TITLE "Tong ket ket qua hoc ky 1, 2004-2005"

USE IN hososv
USE IN monhoc
USE IN diem
RETURN
```

5. Lệnh SCAN ... ENDSCAN

Di chuyển con trỏ bản ghi trên một vùng làm việc nào đó, tương ứng với mỗi bản ghi tìm thấy sẽ thực hiện một tập các lệnh.

Cú pháp:

SCAN [Scope] [FOR BieuthucLogic]

Lệnh 1

[LOOP]

[EXIT]

Lệnh 2

ENDSCAN

Trong đó:

- Tùy chọn Scope cho phép xác định phạm vi tác động của lệnh (ALL, NEXT n, RECORD n, REST).
- Tùy chọn [FOR BieuThucLogic] cho phép lệnh SCAN duyệt trên các bản ghi thỏa mãn BieuThucLogic. Nếu không đưa tham số này vào thì lệnh SCAN sẽ duyệt trên tất cả các bản ghi.
- Thực hiện: (*) Nếu còn bản ghi (trong phạm vi Scope và thỏa mãn BieuThucLogic) trong vùng làm việc -> Thực hiện các lệnh giữa SCAN và ENDSCAN, tiếp đến dịch chuyển con trỏ bản ghi đi 1 vị trí (SKIP 1) rồi trở lại (*); Nếu không còn bản ghi nào-> Kết thúc lệnh SCAN.

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Lệnh LOOP cho phép (ngay lập tức) trở lại bắt đầu vòng SCAN bỏ qua các lệnh giữa LOOP và ENDSCAN. Lệnh LOOP có thể đặt vào bất cứ đâu giữa SCAN và ENDSCAN.

Lệnh EXIT cho phép (ngay lập tức) kết thúc lệnh SCAN ...ENDSCAN và bỏ qua các lệnh giữa EXIT và ENDSCAN. Lệnh EXIT có thể đặt vào bất cứ đâu giữa SCAN và ENDSCAN.

Ví dụ 5: Tính điểm trung bình chung học kỳ 1 năm 2004-2005 của lớp 3E-04 (như ví dụ 4 nhưng thay FOR bằng SCAN).

```
SET SAFETY OFF
SET TALK OFF
CLEAR
CLEAR ALL
CLOSE ALL
SELECT 0
USE monhoc
SET ORDER TO MONLOP   && MA+MALOP
SELECT 0
USE diem
SET RELATION TO mamon+malop INTO Monhoc
SET FILTER TO monhoc.hocky=1 AND monhoc.namhoc=2004
SELECT 0
USE Hososv
COPY TO c:\temp\HOSOLOP FOR malop="3E-04"
USE IN hososv
SELECT 0
USE c:\temp\HOSOLOP ALIAS Hososv EXCLUSIVE
ALTER table c:\temp\HOSOLOP ADD COLUMN Dtbc N(5,2)
GO TOP
SCAN
        SELECT diem
        SUM monhoc.sodvht*diem.diem1,monhoc.sodvht ;
                TO tongdiem,tongdvht FOR masv=Hososv.masv
        SELECT hososv
        REPLACE dtbc WITH IIF(tongdvht>0,tongdiem/tongdvht,0)
ENDSCAN
SELECT Hososv
BROWSE      FIELDS masv,hoten,ngaysinh,dtbc ;
            TITLE "Tong ket ket qua hoc ky 1, 2004-2005"
USE IN hososv
USE IN monhoc
USE IN diem
RETURN
```

III. Chương trình con

1. Thủ tục (PROCEDURE)

Người sử dụng có thể tạo ra các thủ tục riêng bằng cách sử dụng khai báo PROCEDURE

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Cú pháp:

PROCEDURE <ProcName>

[PARAMETERS <para1, para2 ... paraN>]

Lệnh

[ENDPROC]

Trong đó:

- ProcName: Tên thủ tục (do người sử dụng tự đặt)
- Tùy chọn PARAMETERS cho phép khai báo các tham số "hình thức", các tham số này sẽ nhận giá trị thực từ chương trình chính khi thực hiện lời gọi thủ tục, para1, para2 ,..., paraN là tên các tham số hình thức.

Lời gọi thủ tục: Các thủ tục sau khi hoàn tất có thể được gọi từ một chương trình nào đó. Lời gọi chương trình con có dạng như sau.

DO <ProcName> [With value1, value2 ... valueN] [IN ProgName]

Trong đó: ProcName là tên thủ tục cần gọi; value1, value ... valueN là giá trị thực được truyền tương ứng cho các tham số hình thức của chương trình con; tùy chọn IN ProgName cho phép xác định file chương trình chứa thủ tục cần gọi (mặc định là trong bản thân chương trình đang thực hiện).

Ví dụ 6: Viết thủ tục cho phép hiển thị thứ (trong tuần) của một ngày nào đó.

```
CLEAR ALL
CLOSE ALL
CLEAR
DO HienThiThu WITH DATE()
* --------------- Khai bao thu tuc -------------------
PROCEDURE HienThiThu
        PARAMETERS date1
        WAIT WINDOW CDOW(Date1)
ENDPROC
```

Trong ví dụ trên, date1 là tham số "hình thức" trong thủ tục HienThiNgay và nhận tham số thực là Date() trong lời gọi thủ tục trong chương trình chính.

Giả sử ví dụ 6 được lưu vào file Vidu6.PRG, khi đó trong một chương trình khác có thể thực hiện lời gọi thủ tục HienThiThu như sau:

```
DO HienThiThu WITH DATE() IN Vidu6.PRG
```

Chú ý: Với các tham số hình thức Para1, para2 ... có thể không truyền giá trị cho các tham số này khi thực hiện lời gọi thủ tục, tuy nhiên các tham số khi đó sẽ nhận giá trị FALSE (giá trị Logic) và như vậy rất có thể dẫn đến lỗi.

2. Hàm (FUNCTION)

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Ngoài các hàm do Visual Foxpro cung cấp sẵn, người sử dụng có thể tạo ra các thủ tục riêng bằng cách sử dụng khai báo FUNCTION

Cú pháp:

FUNCTION <FuncName>
[PARAMETERS <para1, para2 ... paraN>]
Lệnh
RETURN <BieuThuc>
[ENDFUNC]

Trong đó:

- FuncName: Tên hàm (do người sử dụng tự đặt)
- Tùy chọn PARAMETERS cho phép khai báo các tham số "hình thức", các tham số này sẽ nhận giá trị thực từ chương trình chính khi thực hiện lời gọi hàm para1, para2 ,..., paraN là tên các tham số hình thức.
- BieuThuc: Giá trị của biểu thức này là giá trị trả về của hàm.

Lời gọi hàm: Các hàm tự tạo hoặc hàm do Visual Foxpro cung cấp đều được gọi như sau:

FuncName(With value1, value2 ... valueN)

Trong đó: FuncName là tên hàm cần gọi; value1, value ... valueN là giá trị thực được truyền tương ứng cho các tham số hình thức của chương trình con. Lời gọi hàm chỉ tham gia vào trong các biểu thức.

Ví dụ 7: Viết hàm tính tổng của 2 số.

```
CLEAR ALL
CLOSE ALL
CLEAR
a=10
b=20
WAIT WINDOW STR(a)+"+"+STR(b)+"="+STR(tong(a,b))
* --------------- Khai bao ham -------------------
FUNCTION Tong
        PARAMETERS x,y
        RETURN x+y
ENDPROC
```

Ví dụ 8: Viết hàm trả về thứ (tiếng việt) của một ngày nào đó.

```
CLEAR ALL
CLOSE ALL
CLEAR
WAIT WINDOW "Hom nay la: "+ ThuVN(DATE())
* --------------- Khai bao thu tuc -------------------
FUNCTION ThuVN
```

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

```
        PARAMETERS d
        kq=""
        i=DOW(d)
        DO CASE
                CASE i=1
                        kq="Chu Nhat"
                CASE i=2
                        kq="Thu Hai"
                CASE i=3
                        kq="Thu Ba"
                CASE i=4
                        kq="Thu Tu"
                CASE i=5
                        kq="Thu Nam"
                CASE i=6
                        kq="Thu Sau"
                CASE i=7
                        kq="Thu Bay"
        ENDCASE
        RETURN kq
ENDPROC
```

Bài tập

1. Thực hiện các ví dụ trong bài.

2. Hoàn thành hàm TENVN đã nêu trong bài 3.

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Bài 08 - thiết kế form

I. Lập trình trên windows

- Lập trình trên Windows có nhiều khác biệt so với lập trình trên DOS (như lập trình bằng PASCAL), các điểm khác biệt chính là: Sử dụng nhiều đối tượng trực quan trong thiết kế giao diện; Logic chương trình diễn ra theo sự kiện ...
- **Đối tượng** (điều khiển - **Control**): Các công cụ lập trình trên Windows thường cung cấp sẵn các đối tượng điều kiển như: Cửa sổ (FORM), Nút lệnh (Command Button), Ô text (Textbox), Lưới (GRID) ... và nhiều đối tượng điều khiển khác thường dùng cho việc thiết kế giao diện.
- **Thuộc tính** (**Properties**): Mỗi đối tượng điều khiển được mô tả qua một số thuộc tính nào đó, các thuộc tính này qui định tính chất hiển thị, định danh của đối tượng như: Name - Tên đối tượng, Font - Font chữ, Width - độ rộng, Height - chiều cao ... Các thuộc tính của đối tượng do người lập trình xác lập.
- **Sự kiện** (**Events**): Mỗi đối tượng điều khiển có khả năng đáp ứng một số sự kiện nào đó. Ví dụ: đối tượng Command Button có khả năng đáp ứng sự kiện Click, Double Click ... đối tượng Text box có khả năng đáp ứng sự kiện Click, Change ... Các sự kiện này do người sử dụng tạo ra. Các sự kiện do người lập trình viết mã nhằm thực thi một công việc nào đó.
- **Phương thức** (**Methods**): Mỗi đối tượng điều khiển thường có một vài phương thức tương ứng với một "hành động" nào đó. Ví

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

dụ với FORM có thể sử dụng phương thức Cls để xoá Form, phương thúc Line để vẽ một đoạn thẳng; với đối tượng TextBox sử dụng phương thức Refresh để làm tươi dữ liệu, phương thức Move để di chuyển vị trí ... Các phương thức được thiết lập sẵn nhằm thực thi một công việc cụ thể.

II. thiết kế form

1. Tạo mới một FORM sử dụng Wizard

Các bước thực hiện

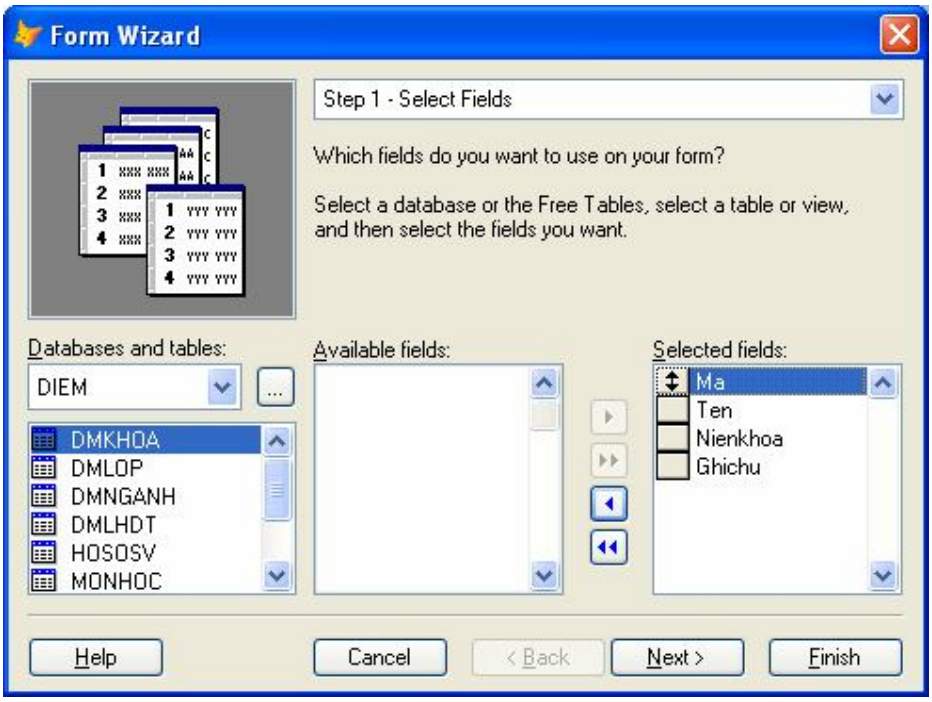
1. Mở Project cần làm, trong Project Manager chọn Tab Documents
2. Chọn Form
3. Chọn New
4. Chọn Form Wizard, khi đó giao diện chọn kiểu Form Wizard xuất hiện như hình sau:

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

5. Chọn kiểu Form Wizard: Form Wizard - Tạo Form từ một bảng đơn, hoặc One-To-Many Form Wizard - Tạo Form từ 2 bảng có quan hệ (1-nhiều) với nhau.

Ví dụ 1: Tạo Form từ bảng đơn - Form cập nhật danh mục khoá (DMKHOA)

1. Bước 0 - Bước 5 ở trên chọn Form Wizard, chọn OK
2. Step 1 - Select Fields: Chọn Database, Tables và Fields (Sử dụng lệnh >, >> để chọn các trường cần thiết), Trong ví dụ này chọn bảng DMKHOA, chọn tất cả các trường. Khi đã chọn xong các trường, chọn Next để qua bước 2.



3. Step 2 - Choose Form Style: Chọn kiểu form, kiểu nút lệnh

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

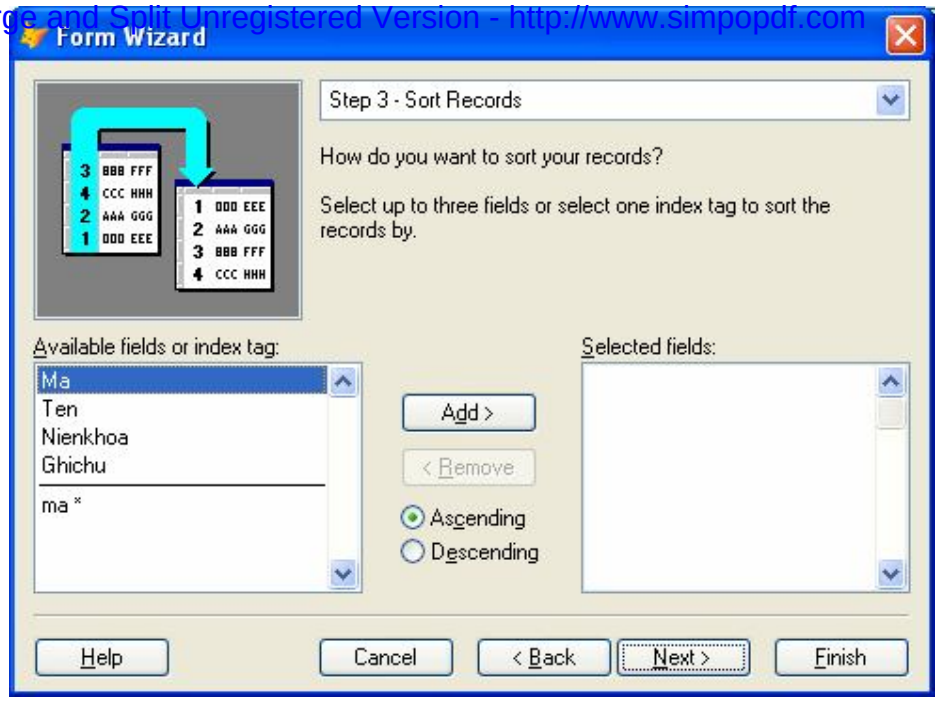
Style: Kiểu Form. Khi chọn một kiểu nào đó, phần ảnh bên trái hiển thị dạng form tương ứng với kiểu đó.

Button type: Kiểu nút lệnh. Text - Nút lệnh dạng chữ (Text), Picture - Nút lệnh dạng ảnh, No button - Không có button. Custom - Tuỳ chọn. Nên chọn là Text hoặc Picture.

Chọn Next để tiếp tục

4. Step 3 - Sort Records

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com



Chọn trường để sắp xếp: Chọn1 trong các trường để sắp xếp hoặc có thể bỏ qua.

5. Step 4 - Finish

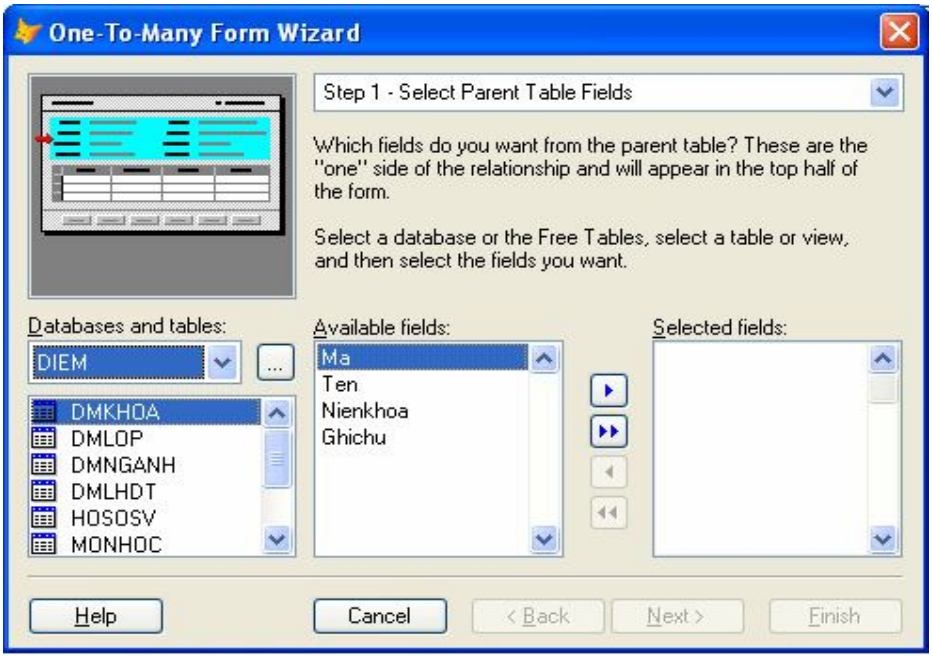
Nhập vào tiêu đề cho form rồi chọn Finish. Tiếp đến nhập vào tên form rồi chọn Save để ghi lại.

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Ví dụ 2: Tạo Form từ 2 bảng - Form cập nhật danh mục Lớp (DMLOP)

1. Bước 0 - Bước 5 ở trên chọn One-To-Many Form Wizard, chọn OK

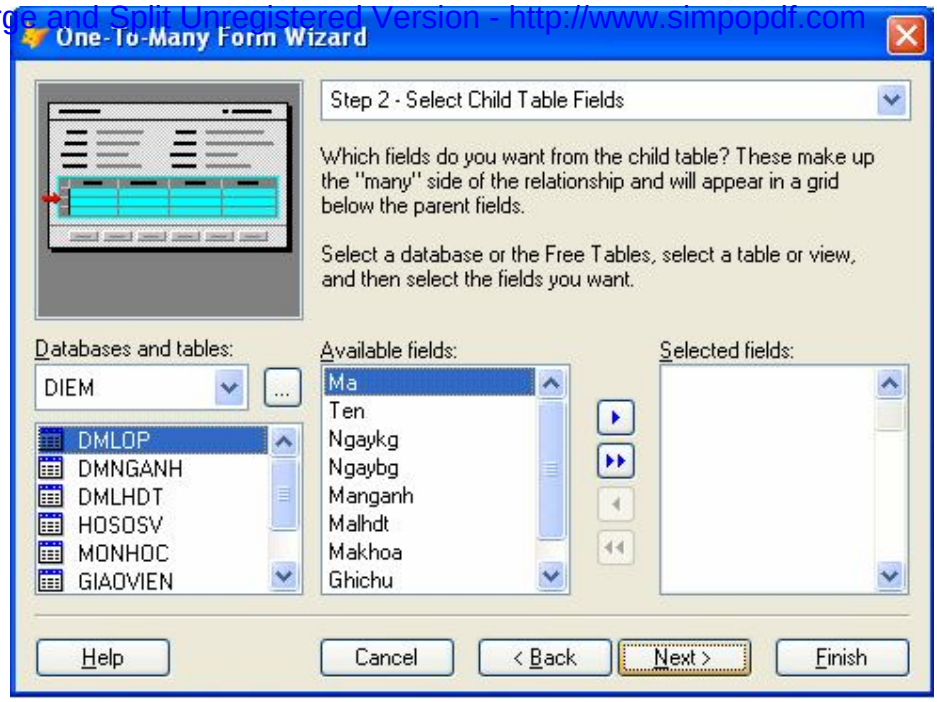
2. Step 1 - Select Parent Table Fields: Chọn trường cho bảng cha, rồi chọn Next.



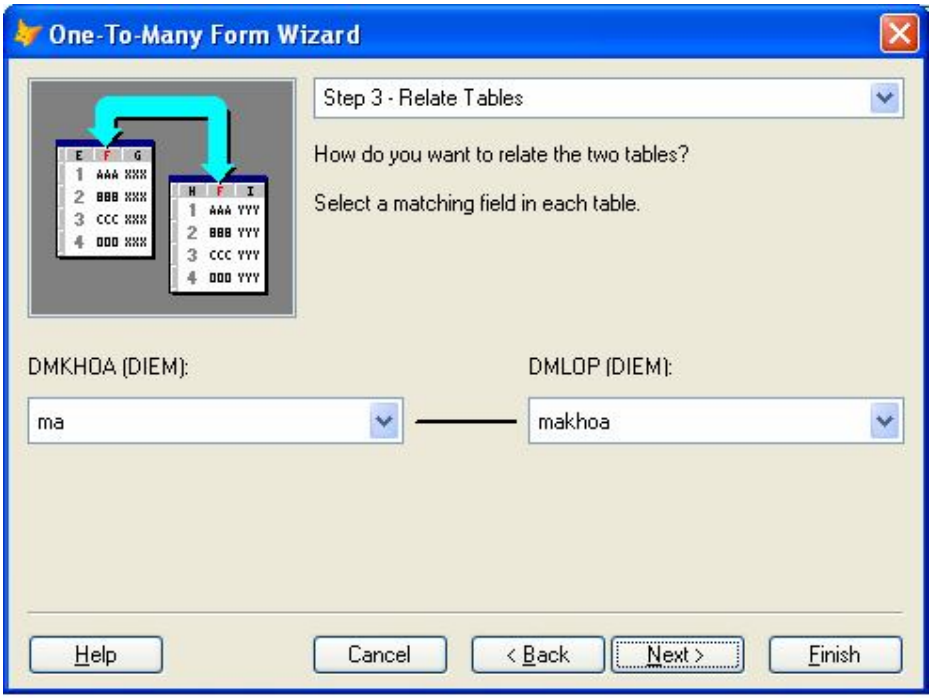
(Trong ví dụ này chúng ta dùng 2 bảng là DMKHOA - bảng cha, và DMLOP - bảng con).

3. Step 2 - Select Child Table Fields: Chọn trường cho bảng con, chọn Next để tiếp tục.

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com



4. Step 3 - Relate Tables: Xác định quan hệ giữa 2 bảng, trường hợp đã thiết lập quan hệ bền vững giữa các bảng thì quan hệ này sẽ được xác lập ngay (như hình vẽ). Chọn Next để tiếp tục.



5. Step 4 - Choose Form Style : Chọn kiểu Form, Chọn Next để tiếp tục (xem phần trước).

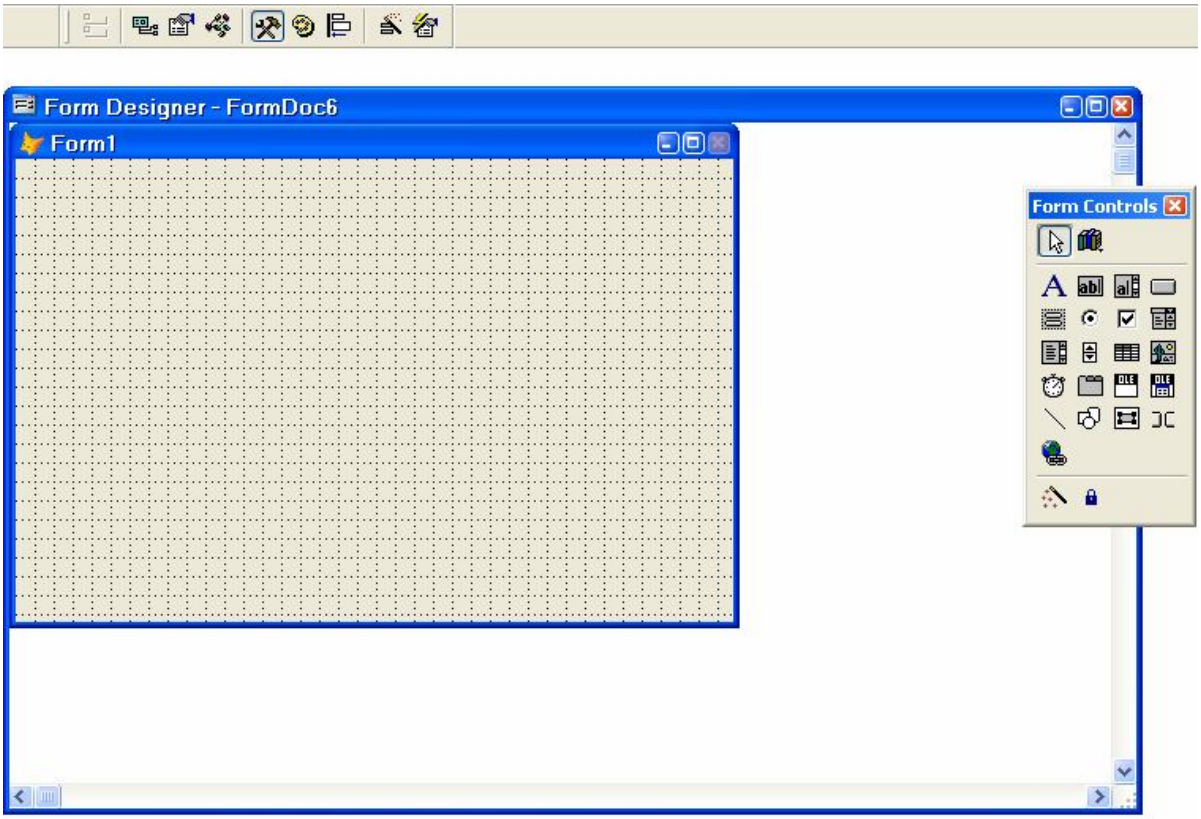
Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

6. Step 5 - Sort Records: Chọn trường để sắp xếp. Chọn Next để tiếp tục (xem phần trước).
7. Step 6 - Finish: Kết thúc (xem phần trước).

2. Tạo mới một FORM sử dụng Form Designer

Các bước thực hiện

1. Mở Project cần làm, trong Project Manager chọn Tab Documents.
2. Chọn Form
3. Chọn New
4. Chọn New Form. Khi đó giao diện Form Designer xuất hiện như hình sau:



Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Trong đó: Phía trên là thanh toolbar Form Designer, bên phải là thanh toolbar Form Controls. Chúng ta sẽ tìm hiểu các chức năng này trong các phần sau.

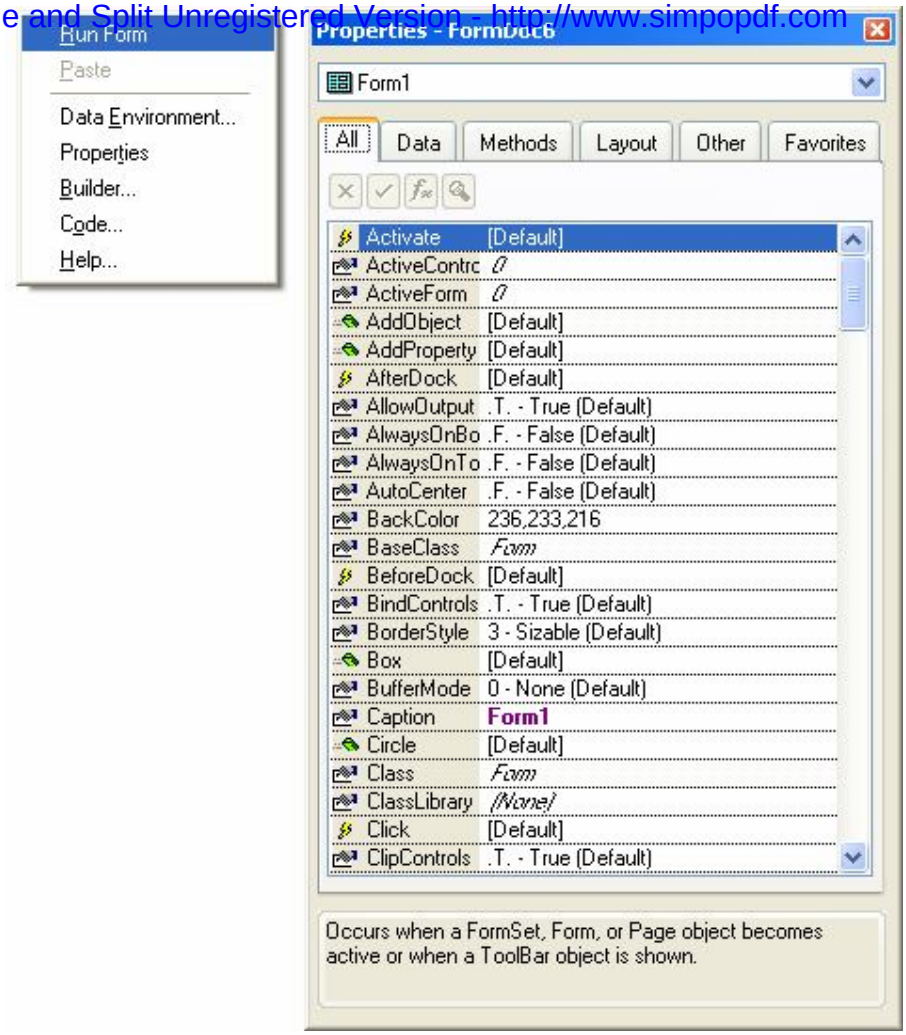
Một số thuộc tính của FORM

Form là đối tượng quan trọng trong thiết kế giao diện trên WIndows, cũng như các đối tượng khác Form có các thuộc tính riêng của nó. Có hai cách để thiết lập các thuộc tính của form: 1. là sử dụng cửa sổ Properties và

2. là sử dụng lệnh để thiết lập (có những thuộc tính chỉ cho phép thiết lập ở chế độ thứ 2).

Hiển thị cửa sổ Properties: Kích phải chuột vào phần nền của Form đang thiết kế, khi đó một Pop-up menu xuất hiện và ta chọn chức năng Properties. Pop-up menu và cửa sổ Properties xuất hiện như hình sau:

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com



Sau đây chúng ta tìm hiểu một số thuộc tính thông dụng của Form.

1. AlwaysOnTop: Ngăn cản các cửa sổ khác đè lên. Thuộc tính này có thể nhận 1 trong hai giá trị là: (1) .T. - Form luôn nằm trên đối tượng khác; (2) .F. - Cửa sổ khác có thể đè lên (mặc định).

2. AutoCenter: Chỉ ra khi xuất hiện vị trí của Form có ở giữa màn hình hay không. Thuộc tính này có thể nhận 1 trong 2 giá trị: (1) .T. - Form tự động được canh giữa; (2) .F. - Vị trí của Form phụ thuộc vào thuộc tính Top, Left (mặc định).

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

3. BackColor: Màu nền của Form. Trong khi thiết kế có thể dùng hộp mầu để chọn hoặc khi chạy chương trình có thể sử dụng hàm RBG() để xác định mầu.

4. BorderStyle: Dạng đường viền của Form. Thuộc tính này có thể nhận một trong các giá trị: (1) 0 - Không có đường viền (No Border); (2) 1 - Dạng đơn, cố định (Fixed Single); (3) 2 - Theo dạng cửa sổ hộp thoại (Fixed Dialog); (4) 3 - Có thể thay đổi kích thước (Sizable - Mặc định).

5. Caption: Tiêu đề của Form. Thuộc tính này nhận giá trị là một xâu kí tự, mặc định caption là tên form.

6. Closable: Cho phép sử dụng lệnh Close trong hộp Control - menu hoặc nút Close góc trên bên phải form. Thuộc tính này có thể nhận các giá trị: (1) .T. - Cho phép đóng cửa sổ bằng lệnh Close (Mặc định); (2) .F. - Không cho phép sử dụng lệnh Close.

7. ControlBox: Hiển thị hoặc không hộp Control menu box góc trên bên trái form. Thuộc tính này nhận các giá trị: (1) .T. - Hiển thị contrrol menu box; (2) - Không hiển thị control menu box.

8. Enabled: Cho phép có hiệu lực hay vô hiệu khả năng đáp ứng sự kiện của các control trên form (và cả form). Thuộc tính này nhận các giá trị: (1) .T. - có hiệu lực; (2) .F. - Vô hiệu lực.

9. FontName: Xác định font chữ dùng để hiển thị các đối tượng text trên form. Khi thiết kế có thể dùng hộp thoại chọn forn để chọn các font chữ hiện có trong máy tính. Trong chương trình

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

thuộc tính này nhận xâu text thể hiện tên của font chữ nào đó, ví dụ ".vnTime", ".vnArial".

10. FontSize: Kích thước font chữ.
11. Height: Chiều cao form.
12. Left: Vị trí cạnh trái của form
13. Maxbutton, MinButton: Hiển thị hay không các nút lệnh Maximize, Minimize. Thuộc tính này nhận các giá trị: (1) .T. - Hiển thị (mặc định); (2) .F. Không hiển thị.
14. Name: Tên (định danh) Form, Tên form là xâu kí tự không chứa dấu cách, và ký tự đặc biệt.
15. Top: Vị trí cạnh trên của Form.
16. Width: Độ rộng của form.
17. WindowState: Xác định trạng thái form khi chạy. Thuộc tính này có thể nhận các giá trị: (1) 0 - Bình thường (Normal), tức là vị trí, kích thước của form như thiết kế (mặc định); (2) 1 - Thu thành biểu tượng trên task bar (Minimize); (3) 2 - Phóng to hết cỡ màn hình (Maximize).

Một số tình huống (Event) thường dùng của Form

1. Activate: Tình huống này sảy ra khi form được "kích hoạt" (active).

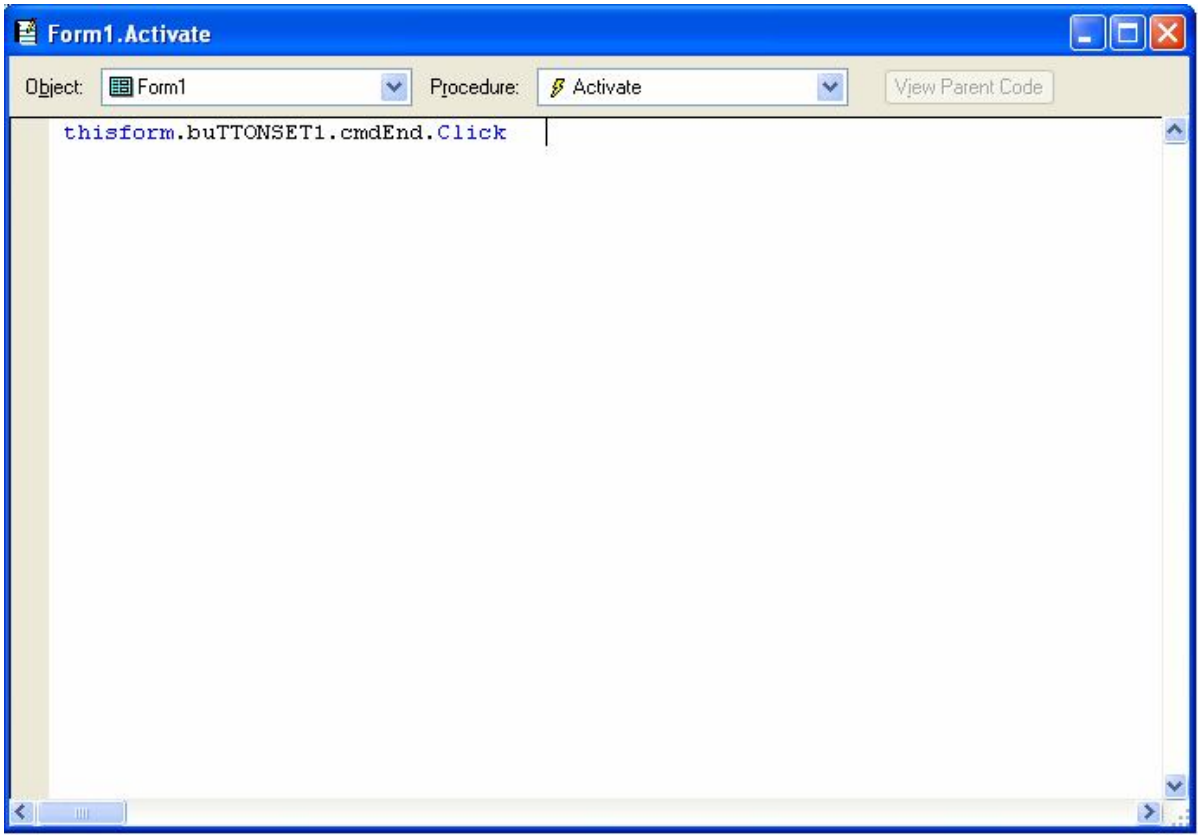
   Ví dụ: Khi kích hoạt form DMKHOA (đã tạo bằng Form Wizard trước đó) cần đưa về bản ghi (khoá) cuối cùng.

   Cách thực hiện:

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

- Trong cửa sổ Project Manager chọn mục Documents.
- Chọn mục Form và mở rộng.
- Chọn form DMKHOA
- Chọn Modify, khi đó giao diện Form Designer xuất hiện

Trong Form Designer nháy đúp chuột trái vào nền Form, khi đó giao diện lập trình trong form xuất hiện như hình sau:



- Object: hiển thị tên đối tượng (trong tình huống này là Form1 - Tên của form DMHANG). Kích chuột trái vào mũi tên xuống (bên cạnh) để chọn Object nếu cần.
- Procedure: Hiện thị tình huống (event) có thể đáp ứng của đối tượng (Objêct) đã chọn ở ô bên cạnh. Kích chuột trái

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

vào mũi tên xuống (bên cạnh) để chọn tình huống nếu cần, trong ví dụ này tình huống cần chọn là Activate.

- Viết lệnh: Khi đã chọn đúng đối tượng và tình huống cần thiết thì cửa sổ bên dưới cho phép soạn thảo lệnh nhằm đáp ứng tình huống đặt ra. Trong ví dụ này chúng ta cần đưa về bản ghi cuối nên nhập vào lệnh:

  thisform.butTONSET1.cmdEnd.click

  Trong đó:

  Thisform: là tên thay thế của form hiện thời

  butTONSET1: Tên lớp chứa các nút lệnh

  cmdEnd: Tên nút lệnh Buttom

  Click: Tình huống click của đối tượng cmdEnd

2. Click: Tình huống này sảy ra khi người sử dụng click phím trái chuột trên form.

   Ví dụ khi click chuột trái lên form DMKHOA -> Hiển thị thông báo “Ban vua click chuot trai len form!”.

   Thực hiện tương tự như trên nhưng chọn Event là click và viết lệnh:

   MESSAGEBOX("Ban vua click chuot trai tren form!")

3. Dbclick: Tình huống này sảy ra khi người sử dụng nháy đúp chuột trái trên form.

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Ví dụ khi Dbclick chuột trái lên form DMKHOA -> Hiển thị thông báo “Ban vua Double click chuot trai len form!”.

Thực hiện tương tự như trên nhưng chọn Event là Dbclick và viết lệnh:

MESSAGEBOX("Ban vua Double click chuot trai tren form!")

4. Deactivate: Tình huống này sảy ra khi form bị thôi kích hoạt (ngược với kích hoạt). Ví dụ hiển thị một thông báo khi thôi kích hoạt 1 form.

5. Destroy: Tình huống này sảy ra khi form được hủy bỏ (destroy).

   Ví dụ: Hiển thị thông báo “Destroy” khi tình huống này xảy ra.

   Thực hiện tương tự như trên nhưng chọn Event là Destroy và viết lệnh:

   MESSAGEBOX("Destroy!")

6. Init: Tình huống này sảy ra khi form được khởi tạo.

   Ví dụ: Hiển thị thông báo “Init” khi tình huống này xảy ra.

   Thực hiện tương tự như trên nhưng chọn Event là Init và viết lệnh:

   MESSAGEBOX("Init!")

7. Load: Tình huống này sảy ra trước khi form được khởi tạo.

   Ví dụ: Hiển thị thông báo “Load” khi tình huống này xảy ra.

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Thực hiện tương tự như trên nhưng chọn Event là Destroy và viết lệnh:

MESSAGEBOX("Load!")

Một số phương thức (Methods) thường dùng của Form

1. Circle(r, x,y): Cho phép vẽ lên form 1 vòng tròn bán kính r, tâm tại x,y.
2. Cls: Xóa sạch nền form.
3. Hide: ẩn form
4. Line(x1,y1,x2,y2): Vẽ đoạn thẳng từ x1,y1 đến x2,y2
5. Point(x,y): Trả về số hiệu mầu tại điểm x,y
6. Print(cText): In một dòng text ra màn hình
7. Pset(x,y): Vẽ điểm x,y
8. Release: Xóa form hiện thời.
9. Show: Hiển thị form.

3. Thi hành (chạy) một FORM

Thực thi một Form bằng 1 trong hai cách:

Cách 1: Từ cửa sổ Project Manager, chọn Documents, chọn Form rồi chọn Form cần thi hành rồi nhấn Run.

Cách 2: Thực hiện qua lệnh Do FORM như sau:

DO FORM <Tên Form>

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

III. Các thành phần liên quan đến công cụ form designer

1. Thanh công cụ Color Palette (Tự tìm hiểu)

2. Thanh công cụ Layout (Tự tìm hiểu)

3. Thanh công cụ Toolbar (Tự tìm hiểu)

4. Data Environment

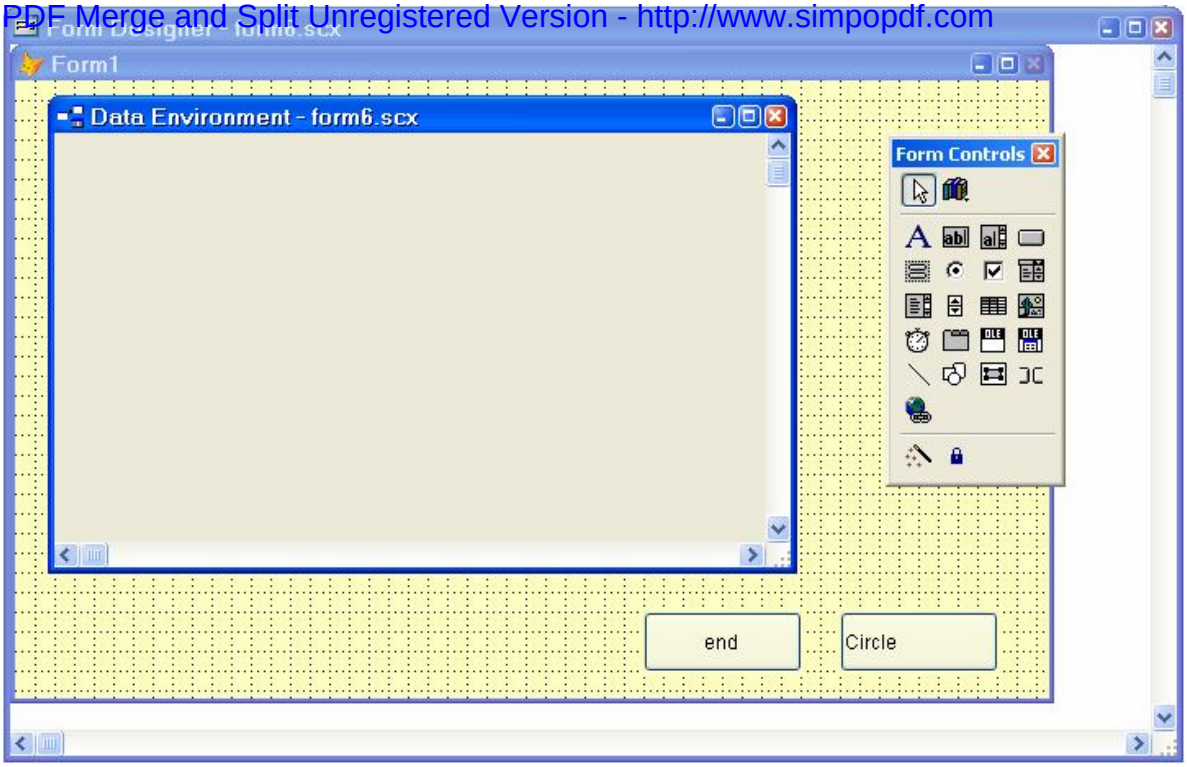
Data Environment là đối tượng dùng để chứa các bảng dữ liệu hay View mà form cần sử dụng.

Data Environment tự động mở các bảng và view khi Form được khởi động và cũng tự đóng chúng khi kết thúc làm việc với form. Sử dụng Data Environment giúp chúng ta dễ dàng thiết kế các giao diện với bảng hoặc view.

Thiết lập Data Environment:

1. Mở Form cần làm việc ở chế độ Form Designer
2. Trong Menu View chọn mục Data Environment ... khi đó giao diện Data Environment xuất hiện như hình sau:

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com



3. Click chuột phải lên vùng trống trong Data Environment rồi chọn lệnh Add để thêm các bảng, view vào Data Environment.

Bài tập

Bước đầu thiết kế các form nhập liệu như đã nêu trong bài tập môn học - bài 1.

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Bài 09 - sử dụng các control

I. giới thiệu

- Các đối tượng điều khiển (Control) được sử dụng trong việc thiết kế giao diện trên Form, Report, Label (Trong bài này chúng ta chỉ xét đến các Control dùng trên FORM). Ví dụ giao diện sau được thiết kế từ FORM và các control.

Hình 1

- Mỗi loại đối tượng thường được thiết kế cho một mục đích nhất định. Ví dụ đối tượng Label (nhãn) được dùng để hiển thị một chú thích, đối tượng TextBox được dùng để cho phép nhập liệu, đối tượng Command Button để thể hiện các nút lệnh... Tuy nhiên người dùng có thể sử dụng với mục đích khác.
- Các Control được xác định qua các Properties (thuộc tính), Methods (Phương thức) và Events (sự kiện - tương tự như FORM).
- Thanh công cụ **Form Controls**: Khi đang mở Form ở chế độ thiết kế (Form Designer), Visual Foxpro hiển thị thanh công cụ Form Controls như hình sau:

Có thể hiển thị/ẩn Form Control bằng cách trong menu **View** chọn **Form Control Toolbar**

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

- **Đưa một Control vào Form**: Từ thanh công cụ Form Controls "nhấp" đối tượng cần dùng và "thả" vào Form. Ví dụ

II. Các đối tượng thường dùng

1. Label:

- Sử dụng cho: hiển thị nhãn, chú thích, ví dụ các ô text sau các label **Khóa, Lớp, Học kỳ, Năm học** ... trong **Hình 1** được dùng để nhập khóa, lớp, học kỳ và năm học.
- Các Properties thường dùng:

1) Alignment: Canh lề phần text hiển thị của label. Thuộc tính này có thể nhận các giá trị: (1) 0 - Left Canh trái, đây là giá trị mặc định; (2) 1-Right Canh phải; (3) 2 - Center canh giữa.
2) Autosize: Tự động thay đổi kích thước phù hợp cho việc hiển thị thuộc tính Caption của nó.
3) BackColor: Đặt mầu nền của Label. Trong khi thiết kế có thể dùng hộp mầu để chọn hoặc khi chạy chương trình có thể sử dụng hàm RBG() để xác định mầu.
4) BackStyle: Thiết lập tính chất nền. Thuộc tính này có thể nhận một trong các giá trị: (1) 0 - Transparent xác định tính chất trong suốt của nền (có thể nhìn thấy mọi đối tượng "phía sau"); (2) 1 - Opaque không cho quan sát các đối tượng phía sau.
5) BorderStyle: Thiết kập thuộc tính biên của đối tượng. Thuộc tính này có thể nhận các giá trị: 0 (none) - Không có đường biên; 1 (Fixed single) - Đường biên đơn.
6) Caption: Text hiển thị như một nhãn, chú thích. Thuộc tính này nhận chuỗi text bất kỳ.
7) FontName: Đặt Font chữ.
8) ForeColor: Đặt mầu chữ. Trong khi thiết kế có thể dùng hộp mầu để chọn hoặc khi chạy chương trình có thể sử dụng hàm RBG() để xác định.
9) Height: Chiều cao
10) Left: Vị trí cạnh trái
11) Name: Tên của đối tượng (dùng để sử dụng trong chương trình)
12) Top: Vị trí cạnh trên
13) Visible: Thiết lập tính chất Hiện/ẩn. Thuộc tính này nhận 2 giá trị: (1) .T. - Hiện ; (2) .F. - ẩn.

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

14) Width: Độ rộng của label.

- Label cũng có nhiều Methods và Events, tuy nhiên vì mục đích của đối tượng này là hiển thị một đoạn text nên trong phần này không giới thiệu các methods và events này.

2. TextBox, EditBox

- Sử dụng cho: Nhập một dòng text, các số, ví dụ các nhãn **Khóa, Lớp, Học kỳ, Năm học** ... trong **Hình 1** được dùng làm tiêu đề cho các ô dữ liệu bên cạnh.
- Các Properties thường dùng:

1) Alignment: Xem phần II.1
2) BackColor: Xem phần II.1
3) BackStyle: Xem phần II.1
4) BorderColor: Xem phần II.1
5) BorderStyle: Xem phần II.1
6) ControlSource: Cho phép "gắn kết" (bound) textbox với 1 trường hoặc biến nào đó (thông thường với 1 trường). Việc "Gắn kết" ở đây được hiểu là nếu text trên textbox thay đổi thì trường hoặc biến "gắn kết" với nó cũng thay đổi theo. Ví dụ, thêm 1 textbox vào form và đặt thuộc tính controlSource="Hososv.hoten" ->Run Form.
7) DisabledBackColor: Mầu nền của textbox khi thuộc tính Enable=.F.
8) DisabledForeColor: Mầu của text khi thuộc tính Enable=.F.
9) Enabled: Cho phép có hiệu lực hay vô hiệu khả năng đáp ứng sự kiện của textbox. Thuộc tính này nhận các giá trị: .T. - có hiệu lực; .F. - Vô hiệu lực
10) FontName: Xem phần II.1
11) ForeColor: Xem phần II.1
12) Format: Định dạng dữ liệu hiển thị. Visual Foxpro qui định kí hiệu sử dụng cho Format của textbox (tham khảo trong help của Visual Foxpro 9 (Microsoft Visual Foxpro 9.0/Reference/Language Reference/Properties/ Format Property). Ví dụ đặt Format là !A (! - Hiển thị chữ hoa, A - Chỉ nhập các ký tự)
13) Height: Xem phần II.1
14) InputMask: Định dạng dữ liệu nhập. Visual Foxpro qui định kí hiệu sử dụng cho InputMask của textbox (tham khảo trong help của Visual Foxpro 9 (Microsoft

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Visual Foxpro 9.0/Reference/Language Reference/Properties/InputMask Property). Ví dụ đặt InputMask là ### (# - nhập chữ số) cho phép nhập 3 chữ số.

15) Left: Xem phần II.1

16) MaxLength: Số ký tự tối đa có thể nhập (nếu MaxLength=0 thì số ký tự là bất kỳ, đây là giá trị mặc định).

17) SelectOnEntry: Thiết lập chế độ đánh dấu text khi đối tượng này nhận quyền điều khiển. Thuộc tính này nhận các giá trị: .T. - Cho phép chế độ đánh dấu; .F. Không cho phép.

18) TabIndex: Thứ tự theo Index của đối tượng (khi sử dụng phím Tab con trỏ sẽ di chuyển theo thứ tự Index từ thấp đến cao)

19) Top: Xem phần II.1

20) Value: Giá trị là xâu text hiển thị trên đối tượng này.

21) Visible: Xem phần II.1

22) Width: Xem phần II.1

- Các Methods và Event thường dùng:

1) Refresh (method): "Làm tươi" lại dữ liệu hiển thị trên textbox. Ví dụ, đặt thuộc tính ControlSource = text1 (text1 là 1 biến), sau đó gán cho text1 ="Visual Foxpro", khi đó cần thực hiện Refresh để làm tươi lại dữ liệu trên Text.

2) SetFocus (method): Đưa con trỏ đến đối tượng này.

3) Valid (Event): Sự kiện này sảy ra trước khi chuyển quyền điều khiển tới đối tượng khác. Vì lý do như vậy nên sự kiện này thường dùng để kiểm tra tính hợp lệ của dữ liệu, nếu không hợp lệ thì có thể yêu cầu người sử dụng nhập lại.

Ví dụ 1: Trong hình 1, xét ô text học kỳ. Ta có học kỳ chỉ nhận 2 giá trị là 1 hoặc 2, để đảm bảo yêu cầu này chúng ta sẽ viết đoạn lệnh đáp ứng sự kiện valid để kiểm.

Giá thiết textbox được thiết lập các thuộc tính

Name = **hocky**

Maxlength=1

khi đó sự kiện **hocky.valid** được viết như sau:

```
hk=this.value
IF !hk$"1;2" then
	WAIT WINDOW "Gia tri khong dung! Hoc ky = 1 hoac 2" NOWAIT
	RETURN 0
ENDIF
```

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Ví dụ 2: Trong hình 1, xét ô text Khoa. Ta thấy các khóa nhập vào ô đây phải nằm trong DMKHOA thì mới có nghĩa -> Viết sự kiện valid để kiểm tra tính hợp lệ này của ô text khóa.

Giá thiết textbox được thiết lập các thuộc tính

Name = **Khoa**

Maxlength=10

khi đó sự kiện **Khoa.valid** được viết như sau:

```
khoahoc=this.Value
SELECT dmkhoa
LOCATE FOR (ma=khoahoc)
IF FOUND() THEN
	thisform.lblTenKhoa.Caption=dmkhoa.ten
ELSE
	IF !EMPTY(khoahoc) THEN
		WAIT WINDOW "Ma khoa khong dung!!" NOWAIT
		RETURN 0
	ELSE
		thisform.lblTenKhoa.Caption=""
	ENDIF
ENDIF
```

- Đối tượng **EditBox** tương đối giống **TextBox (**khác biệt chủ yếu là cho phép soạn text trên nhiều dòng) các bạn tự tìm hiểu thêm.

3. Command Button

- Sử dụng cho: Tạo các nút lệnh. Ví dụ như các nút lệnh **Tạo BC, To Excel, Xem in** ... trong hình 1.
- Các Properties thường dùng: Trong phần này chỉ giới thiệu một số thuộc tính khác so với các thuộc tính đã giới thiệu ở các đối tượng trên.
    1) Caption: Text hiển thị như tên (khác với name) của nút lệnh.
    2) FontName: Font chữ cho caption.
    3) Picture: Cho phép đưa một ảnh vào nút lệnh.
    4) ToolTipText: Đặt dòng text (tooltip) khi đưa con trỏ tới nút lệnh.
- Các Methods và Events thường dùng:
    1) Click (Event): Sự kiện này sảy ra khi người sử dụng click chuột trái vào nút lệnh (nút lệnh được thiết kế chính để đáp ứng sự kiện này).

Ví dụ 1: Thêm vào form nút lệnh "Thoát" để đóng form, thiết lập các thuộc tính:

Caption=Thoát

Name=cmdThoat

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Khi đó dự kiện **cmdThoat.Click** được viết như sau:

```
IF MESSAGEBOX("Dong form nay lai?",4+32)=6 THEN
        thisform.Release
ENDIF
```

Ví dụ 2: Tạo form DMKHOA cho phép cập nhật danh mục khóa. Thêm vào Form các nút lệnh "Đầu", "Trước", "Sau" và "Cuối" để di chuyển con trỏ bản ghi của bảng DMKHOA về bản ghi đầu, bản ghi trước, bản ghi tiếp và bản ghi cuối.

Lần lượt đặt tên cho các nút lệnh đó là cmdDau, cmdTruoc, cmdSau và cmdCuoi, láy ví dụ.

Sửa lại chức năng trong các nút lệnh sao cho khi chuyển đến bản ghi đầu thì nút lệnh "Đầu", "Trước" sẽ disable, khi chuyển đến bản ghi cuối thì các nút lệnh "Sau", "Cuối" sẽ disable.

4. OptionGroup Box

- Sử dụng cho: Lựa chọn một trong nhiều khả năng.
- Các Properties thường dùng:
    1) AutoSize: Tự động thay đổi kích thước của đối tượng cho phù hợp (phần thiết kế). Thuộc tính này nhận các giá trị .T. và .F.
    2) ButtonCount:Số các tùy chọn
    3) Value: Giá tị trả về của option, giá trị này là số thứ tự mục chọn.

  Khi thiết kế, có thể thay đổi một số thuộc tính khác của OptionGroup bằng cách nhấp chuột phải vào đối tượng, tiếp đến chon **Builder..**

  Vì đây là một dạng container nên để thay đổi các thuộc tính các đối tượng con của nó bằng cách: nhấp chuột phải vào đối tượng, tiếp đến chọn **Edit**...
- Các Methods và Events thường dùng: Không có methods, hoặc event nào đặt biệt nên không giới thiệu ở đây.

Ví dụ:

5. CheckBox

- Sử dụng cho: Lựa chọn một trong 2 khả năng.
- Các Properties thường dùng:
    1) Caption: Thể hiện phần text của đối tượng
    2) Value: Checked - Giá trị 1 hoặc .T.; Unchecked - Giá trị 0 hoặc .F.
- Các Methods và Events thường dùng:

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

1) Click (Event): Sự kiện này sảy ra khi người sử dụng click chuột trái vào check box.

Ví dụ: Giả sử dùng check box để hiển thị thông tin về giới tính với qui ước checked - Nam, và Unchecked là nữ.

Khi đó sự kiện click được viết như sau:

```
IF this.Value THEN
        this.Caption="Nam"
ELSE
        this.Caption="N÷"
ENDIF
```

6. **Combo Box**

7. **List Box**

8. **Grid**

9. **Image**

10. **Timer**

11. **Page Frame**

Bài tậP

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Bài 10 - Tạo reports và labels

I. Giới thiệu

- Report là công cụ dùng để thiết kế các trang in báo cáo.
- Cấu trúc của report thường được thiết kế theo một trong các dạng sau:

1. Dạng cột (Column report): Mỗi Field là một cột và mỗi Record nằm trên 1 dòng, như danh sách lớp; bảng điểm. Ví dụ

HOSOSV

05/06/06

Masv	Họ và tên	Ngaysinh
20010914	Vũ Thanh Tùng	05/09/84
CHKH0901	Nguyễn Thành Luân	11/16/86
SV-0001	Lương Ngọc ánh Moi sua	01/07/84
SV-0002	Cao Mai An	12/06/85
SV-0003	Đỗ Minh Anh	09/30/86
SV-0004	Lê Thị Kim Anh	04/15/85

2. Dạng dòng (Row report): Mỗi field nằm trên 1 hàng và mỗi, hết record này đến record khác. Ví dụ

HOSOSV

05/06/06

Masv: 20010914
Họ và tên: Vũ Thanh Tùng
Ngaysinh: 05/09/84

Masv: CHKH0901
Họ và tên: Nguyễn Thành Luân
Ngaysinh: 11/16/86

Masv: SV-0001
Họ và tên: Lương Ngọc ánh Moi sua
Ngaysinh: 01/07/84

3. Dạng một - nhiều (One - To - Many Report): Dữ liệu được lấy từ 2 bảng, 1 bảng cha và một bảng con (có quan hệ One-Many với nhau). Cứ mỗi record cha sẽ có các record con phía dưới. Ví dụ

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Masv: SV-0002
Họ và tên: Cao Mai An
Ngaysinh: 12/06/85

Mamon	Diem1	Diem2
GT101	7.00	0.00
HGT01	8.00	0.00
VL101	5.00	0.00
TRI01	6.00	0.00
TA101	8.00	0.00
TNL01	7.00	0.00

Masv: SV-0003
Họ và tên: Đỗ Minh Anh
Ngaysinh: 09/30/86

Mamon	Diem1	Diem2
GT101	6.00	0.00
HGT01	7.00	0.00
VL101	8.00	0.00
TRI01	7.00	0.00
TA101	9.00	0.00
TNL01	7.00	0.00

4. Dạng nhiều cột (Multi - Column report): Dạng cột và dòng như trên là dạng 1 "cột", tuy nhiên có thể tạo các report trên thành dạng nhiều "cột". Ví dụ

HOSOSV

05/06/06

Masv	Họ và tên	Ngaysinh	Masv	Họ và tên	Ngaysinh
20010	Vũ Thanh Tùng	05/09/84	SV-00	Nguyễn Thị Huyền	04/27/85
CHKH	Nguyễn Thành Luân	11/16/86	SV-00	Đặng Đức Khôi	11/01/86
SV-00	Lương Ngọc ánh Moi sua	01/07/84	SV-00	Đàm Quang Khang	12/01/84
SV-00	Cao Mai An	12/06/85	SV-00	Lê Xuân Khiêm	09/05/85
SV-00	Đỗ Minh Anh	09/30/86	SV-00	Vũ Trọng Kiên	07/08/83
SV-00	Lê Thị Kim Anh	04/15/85	SV-00	Nguyễn Phùng Lương	10/10/85

II. Thiết kế report sử dụng Wizard

Các bước thực hiện

1. Mở Project cần làm, trong Project Manager chọn Tab Documents
2. Chọn Report
3. Chọn New

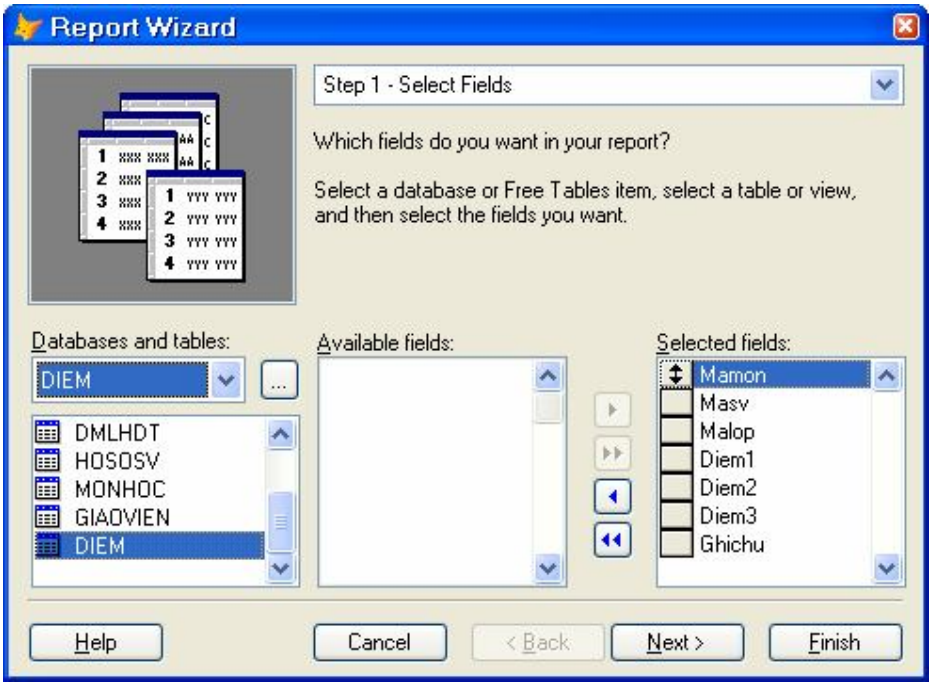
Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

4. Chọn Report Wizard, khi đó giao diện chọn kiểu Report Wizard xuất hiện như hình sau:

5. Chọn kiểu Report Wizard: Report Wizard - Tạo Repor từ một bảng đơn, hoặc One-To-Many Report Wizard - Tạo Report từ 2 bảng có quan hệ (1-nhiều) với nhau.

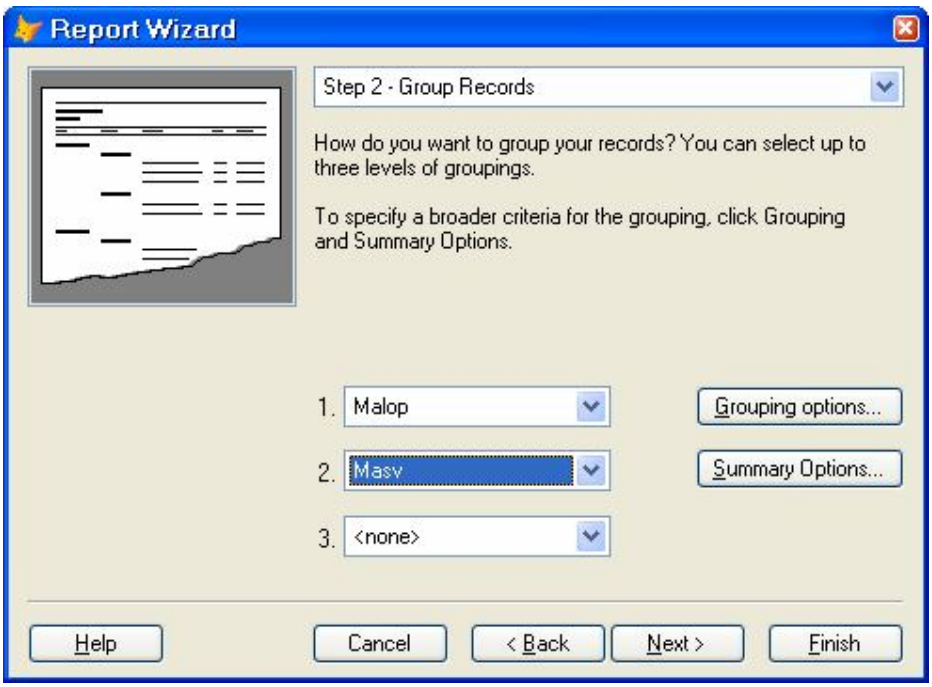
Ví dụ 1: Tạo Report từ bảng đơn (sử dụng Report Wizard)

1. Bước 0 - Bước 5 ở trên chọn Report Wizard, chọn OK
2. Step 1 - Select Fields: Chọn Database, Tables và Fields (Sử dụng lệnh >, >> để chọn các trường cần thiết. Khi đã chọn xong các trường, chọn Next để qua bước 2 (ví dụ chọn tất cả các trường của bảng DIEM).

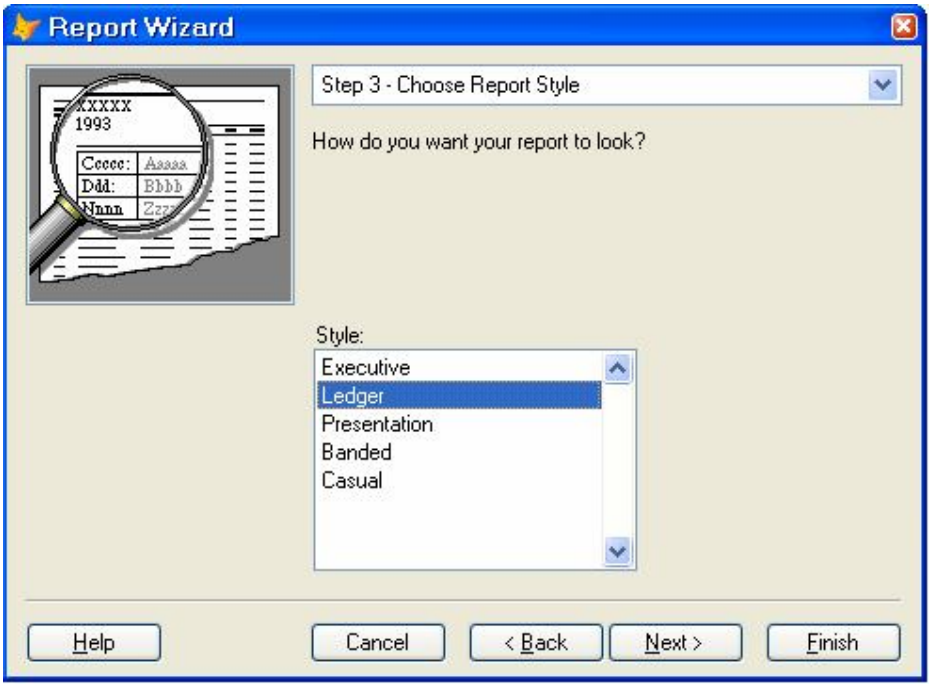


Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

3. Step 2 - Group Records: Chọn trường để nhóm dữ liệu, có thể tạo ra nhiều cấp nhóm (ví dụ chọn 2 cấp nhóm: 1 là malop, 2 là masv), chọn Next để qua bước 3.



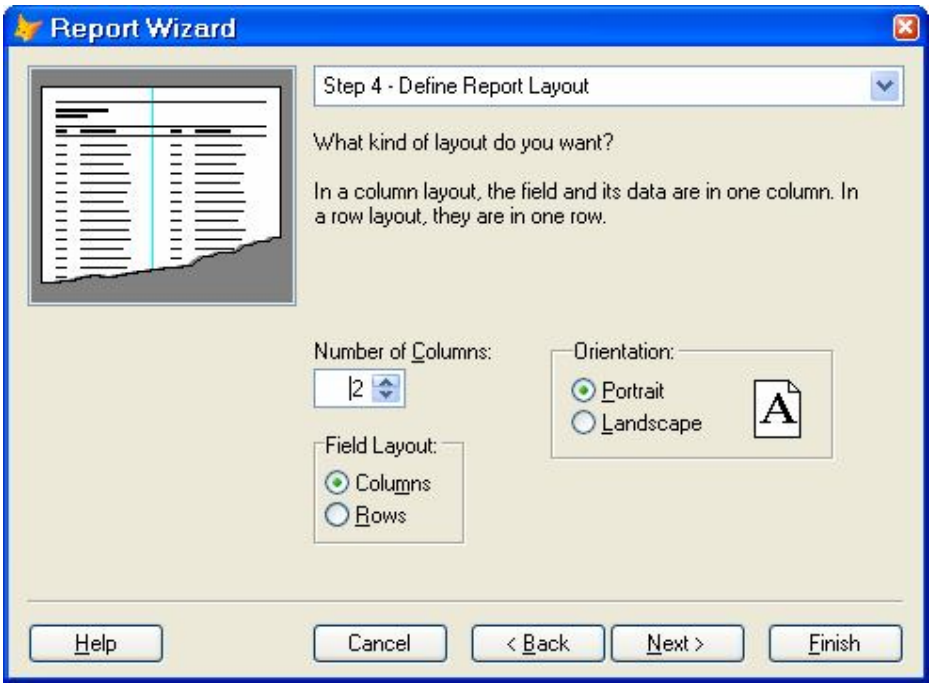
4. Step 3 - Choose Report Style: Chọn kiểu report (hình thức mỗi style khi được chọn sẽ hiển thị ở hình bên - ví dụ này chọn Ledger), chọn Next để qua bước 4.



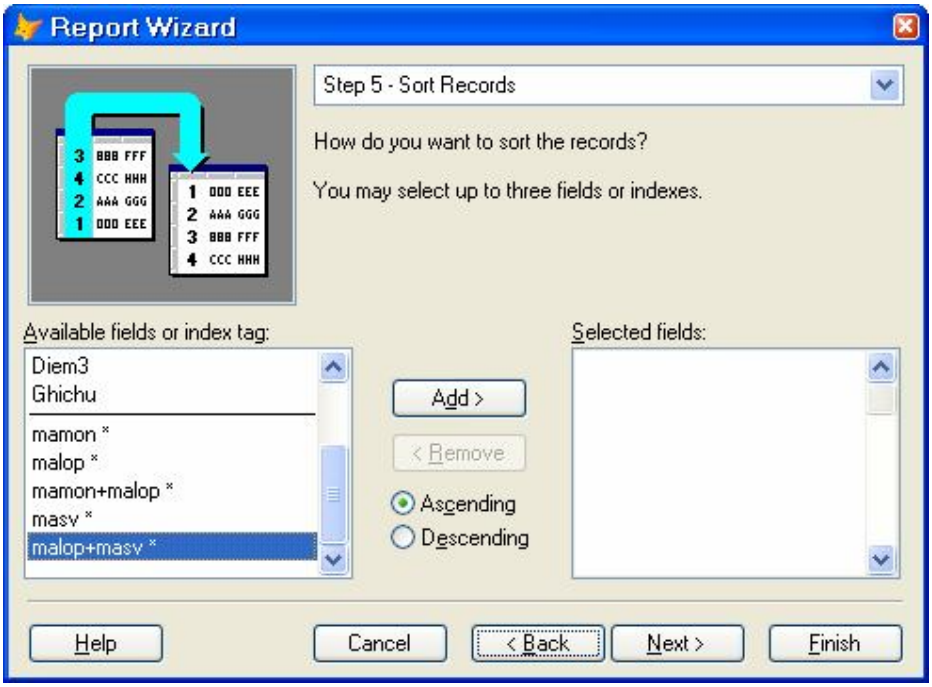
5. Step 4 - Define Report Layout: Thiết lập cấu trúc report Trong đó: **Number of Columns**: Số "Cột" (chỉ cho phép khi không chọn Group); **Field Layout Columns**: Các trường ở

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

dạng cột (dạng 1 - chỉ cho phép khi không chọn Group); **Field Layout Row**: Các trường ở dạng dòng (dạng 2 - chỉ cho phép khi không chọn Group); **Orientation Portrait**: Hướng giấy theo chiều đứng; **Orientation Landscape**: Hướng giấy theo chiều ngang (Trong ví dụ này chúng ta bỏ qua Step 4).



6. Step 5 - Sort Records: Chọn cách sắp xếp các bản ghi.



7. Step 6 - Finish: Kết thúc, chọn finish rồi nhập vào tên file cần ghi để kết thúc.

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

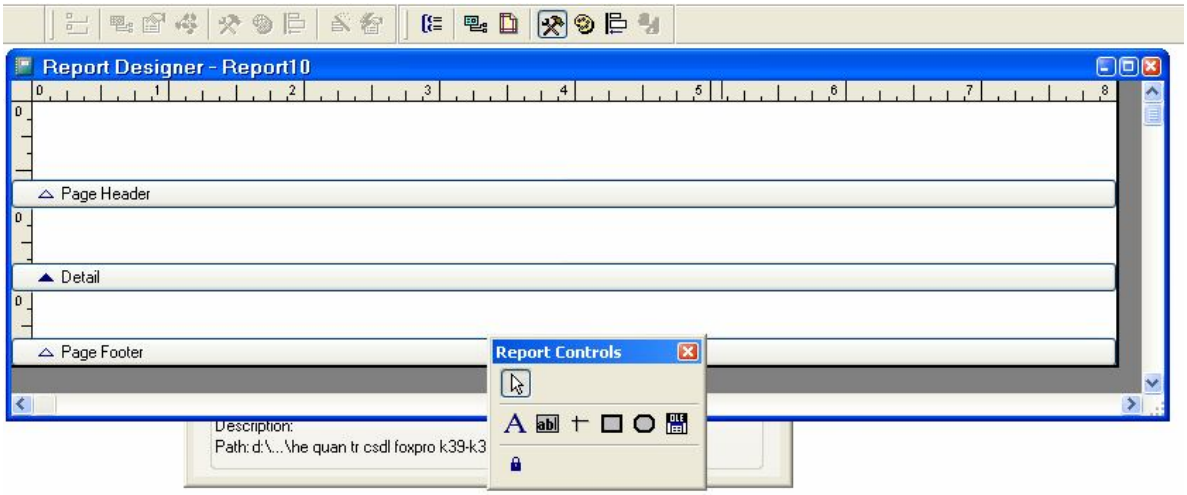
Ví dụ 2: Tạo One - To - Many Report

(Tương tự như **ví dụ 1**, sinh viên tự tìm hiểu)

III. Thiết kế report sử dụng Report designer

Các bước thực hiện

1. Mở Project cần làm, trong Project Manager chọn Tab Documents
2. Chọn Report
3. Chọn New
4. Chọn New Report, khi đó giao diện Report Designer xuất hiện như hình sau:



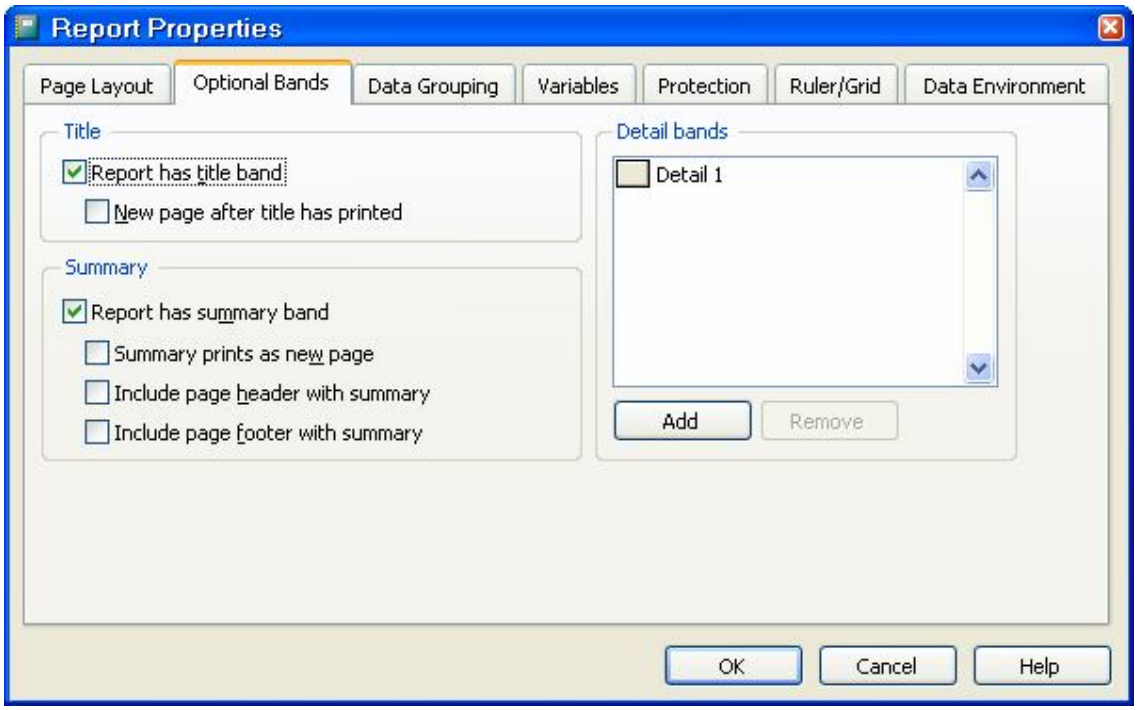
Trong đó: Phía trên là thanh công cụ, cửa sổ Report Designer là phần thiết kế report, phía dưới là thanh Report Controls.

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

1. Bands

- Trong cửa sổ Report Designer có 3 phần: **Page Header; Detail; Page Footer**, các phần này gọi là các Band (nhóm).
- Mặc định (khi tạo mới) mỗi Report có 3 band như trên.
- **Page Header**: Những thiết kế trong phần này sẽ in lặp lại ở đầu mỗi trang
- **Detail**: Những thiết kế trong phần này sẽ in tương ứng với mỗi bản ghi của vùng làm việc hiện thời.
- **Page Footer**: Những thiết kế trong phần này sẽ in lặp lại ở cuối mỗi trang
- Ngoài các Band trên chúng ta còn có thể thêm vào band Title và Summary như sau:

  1. Click chuột trái trên nền Report
  2. Chọn Optional Bands, khi đó giao diện chương trình xuất hiện như hình sau:



Đánh dấu chọn (check) vào các mục: **Report has title band** và **Report has summary band**, chọn OK để kết thúc. Khi đó giao diện Report Designer có dạng như hình sau:

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

- **Title**: Những thiết kế trong phần này sẽ in một lần duy nhất ở đầu của Report.
- **Summary**: Những thiết kế trong phần này sẽ in một lần duy nhất ở cuối của Report.

2. Thanh công cụ Report Controls

- Trong thiết kế Report (Report Designer), Visual Foxpro cung cấp công cụ Report Control để đưa các điều kiển vào trong Report. Thanh Report control có dạng như hình sau.

- Các Control chính:

1. **Label**: Tạo nhãn, ví dụ như tiêu đề cột ...

   Cách thực hiện:

   a. Nhấp chuột trái vào biểu tượng label trên Report Controls

   b. Đưa chuột (khi đó có hình là chữ I) vào vị trí cần đặt label trên report rồi nhấp chuột trái.

   c. Nhập vào dong text cần thể hiện của label.

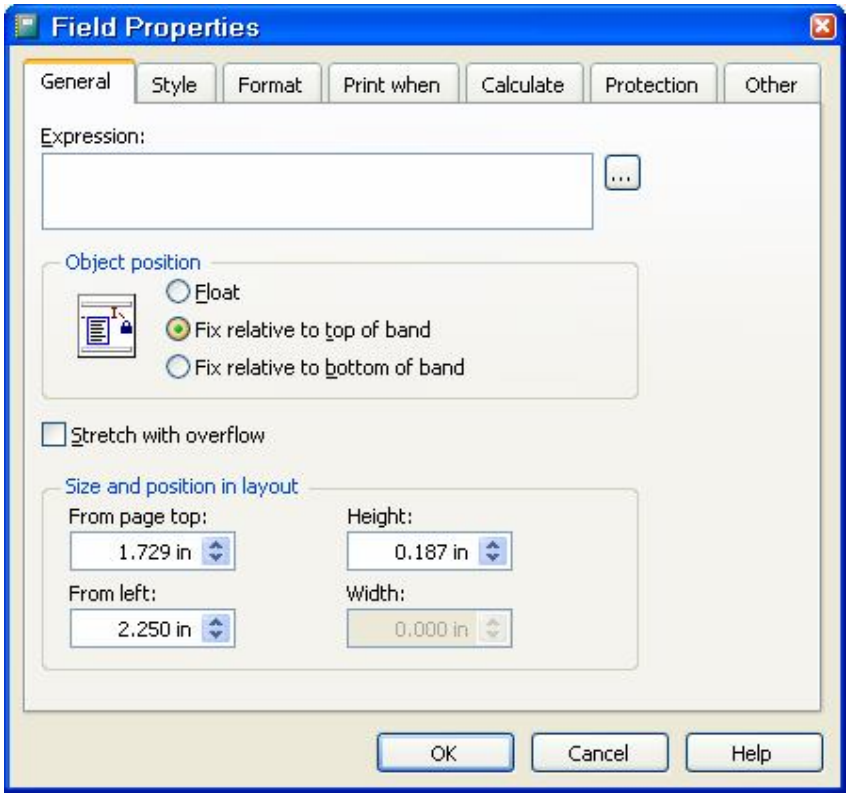
2. **Field**: Đưa dữ liệu từ các trường vào report.

   Cách thực hiện:

   a. Nhấp chuột trái vào biểu tượng Field trên Report Controls

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

b. Đưa chuột (khi đó có hình là chữ "+") vào vị trí cần đặt field trên report rồi nhấp chuột trái. Khi đó giao diện Field Properties xuất hiện như hình sau:



c. Nhập vào **Expression** biểu thức hoặc 1 trường.

d. Chọn OK để kết thúc.

3. **Line**: Vẽ các đoạn thẳng.

   Cách thực hiện:

   a. Nhấp chuột trái vào biểu tượng Line trên Report Controls

   b. Vẽ một đoạn thẳng lên report.

4. **Rectangle**: Vẽ hình chữ nhật (tương tự Line)

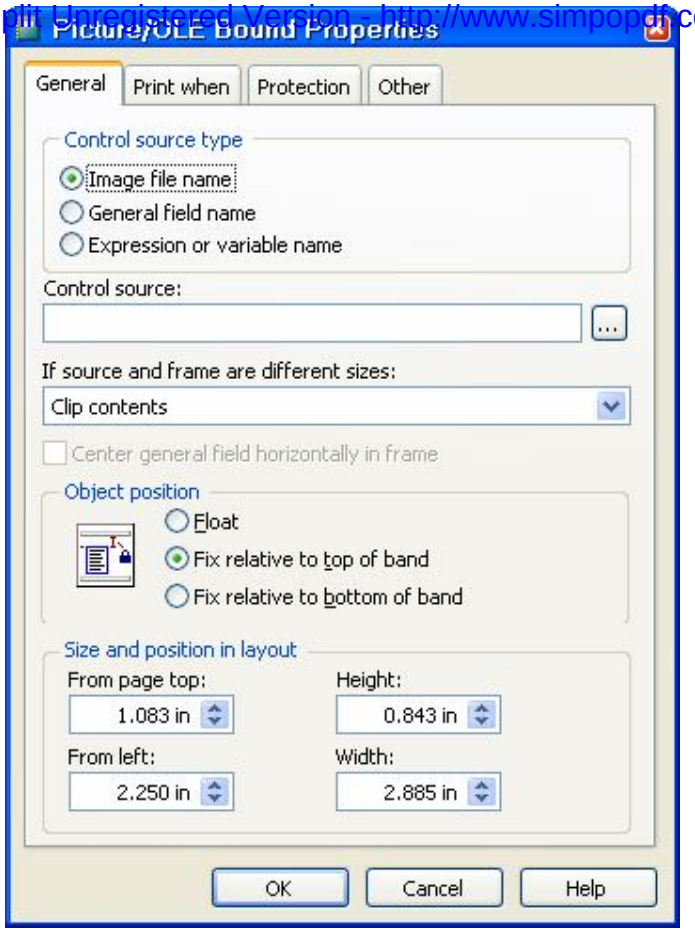
5. **Picture/Ole bound control**: Đưa ảnh (hoặc đối tượng Ole khác) vào report.

   Cách thực hiện:

   a. Nhấp chuột trái vào biểu tượng Picture/Ole trên Report Controls

   b. Đưa chuột vào vị trí cần đặt trên report rồi nhấp chuột trái, khi đó giao diện Picture/Ole properties xuất hiện như hình sau:

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com



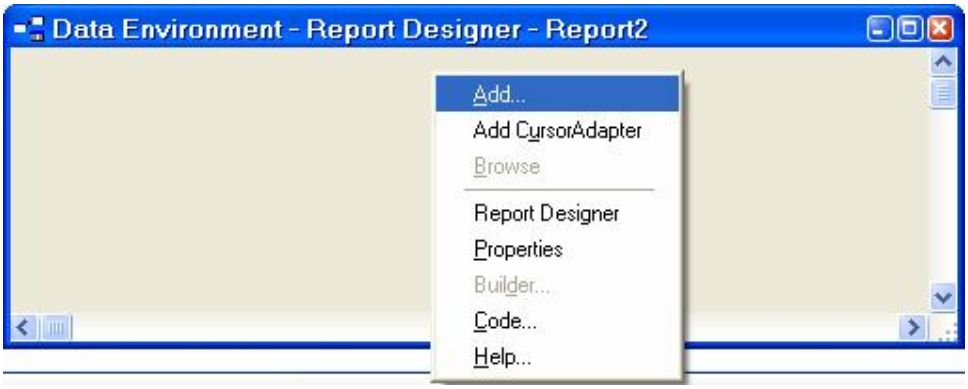
c. Chọn General Field Name rồi gõ vào Control source tên trường ảnh.

3. DataEnvironment

Thiết lập môi trường dữ liệu cho Report.

Cách thực hiện:

1. Kích phải chuột vào report (trong cửa sổ report designer)
2. Chọn mục DataEnvironment, khi đó giao diện DataEnvironment xuất hiện như sau:



3. Kích chuột phải vào nền giao diện DataEnvironment chọn Add và thêm để thêm Table, View vào DataEnvironment.

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Lưu ý: Không nhất thiết phải sử dụng DataEnvironment, mà người sử dụng có thể tự tạo ra môi trường này bằng các lệnh mở bảng, view trước khi gọi thực hiện một report nào đó.

4. Ví dụ

Thiết kế một report "danh sách lớp" như mẫu sau: (Ví dụ trên Visual Foxpro)

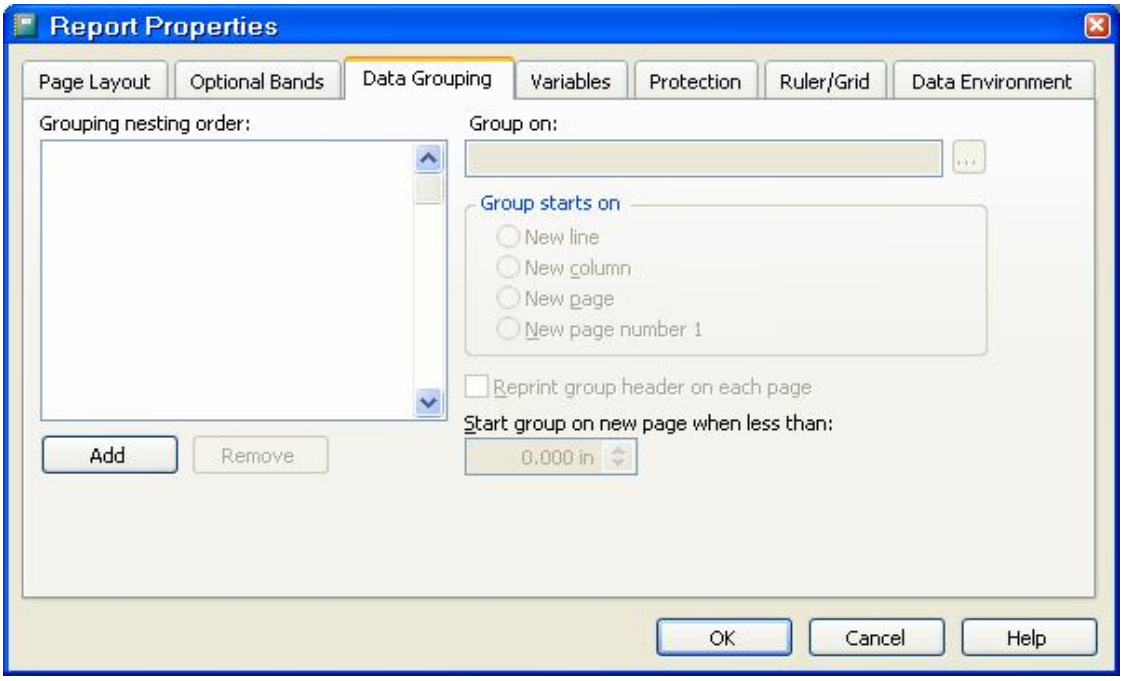
Học viện KTQS Phòng Đào tạo Danh sách lớp Lớp: ……, Khoá: ……					
STT	Mã SV	Họ tên	Ngày sinh	Hộ khẩu	Ghi chú
…	…	…	…	…	…

5. Nhóm dữ liệu (Data Grouping)

Tùy chọn này cho phép "nhóm" dữ liệu theo một tiêu chuẩn nào đó. Ví dụ in danh mục các lớp có thể nhóm theo khóa, theo ngành, theo loại hình đào tạo ...

Cách thực hiện:

1. Kích phải chuột vào report (trong cửa sổ report designer)
2. Chọn mục Data Grouping, khi đó giao diện Data Grouping xuất hiện như sau:



3. Chọn Add để thêm vào các tiêu chuẩn nhóm, có thể nhóm nhiều cấp. Khi một nhóm được thêm vào Report, thì xuất hiện thêm hai Band là Group Header và Group Footer

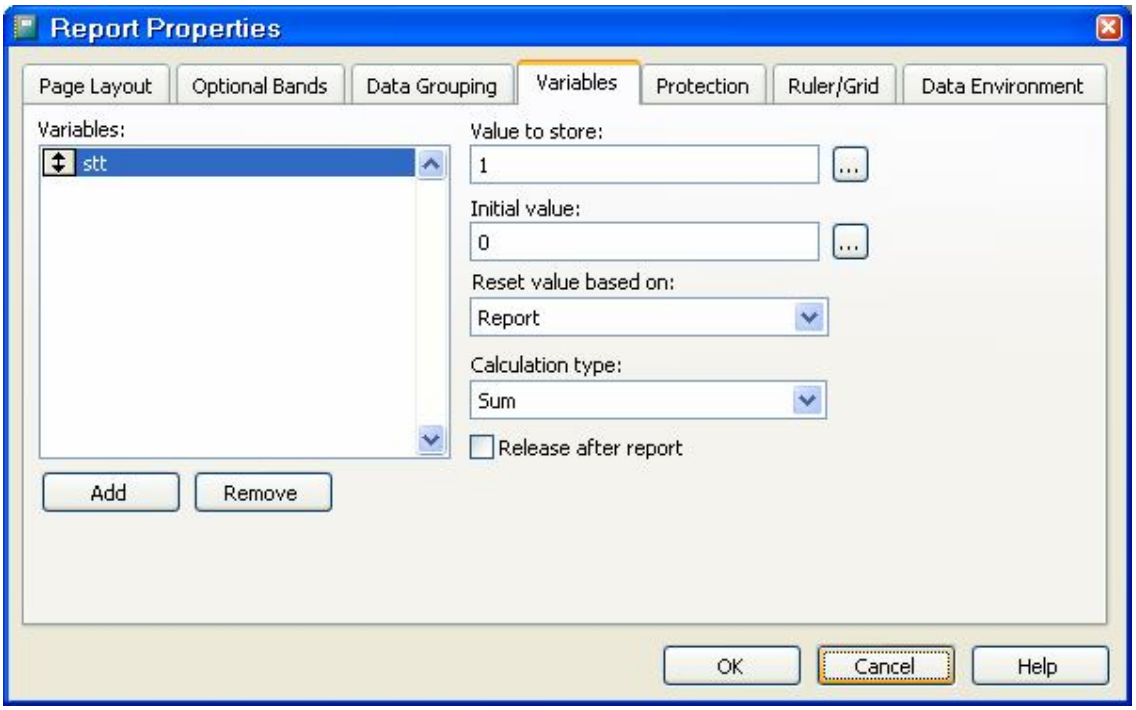
Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Group Header: Những thiết kế trong phần này sẽ in lặp lại ở đầu mỗi nhóm

Group Footer: Những thiết kế trong phần này sẽ in lặp lại ở cuối mỗi nhóm

6. Tạo biến trong Report

1. Kích phải chuột vào report (trong cửa sổ report designer)
2. Chọn mục Variables..., khi đó giao diện Variables xuất hiện như sau:



3. Chọn Add để thêm vào các biến. Với mỗi biến có thể thiết lập các tùy chọn sau:
   a. Value to store: Giá trị lưu vào biến
   b. Initial Value: Giá trị khởi tạo
   c. Reset value based on: Khởi tạo lại giá trị của biến tùy thuộc vào đối tượng được chọn.
   d. Calculation type: Hàm tính
   e. Release After report: Xóa biến này khi kết thúc report.

IV. thực hiện (chạy) một report

Trong cửa sổ Project Manager

Cách thực hiện:

1. Chọn mục Documents
2. Chọn Report và mở rộng

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

3. Chọn Report cần thực hiện

4. Chọn lệnh Preview

Sử dụng lệnh

Lệnh REPORT FORM

```
REPORT FORM <TenFileReport>
            [PREVIEW | TO PRINT [PROMPT]]
            [NOCONSOLE]
```

Trong đó:

- TenFileReport: Tên file report
- PREVIEW: Xem report ở chế độ preview
- TO PRINT: In report ra máy in, PROMPT: Xuất hiện hộp thoại Print dialog
- NOCONSOLE: Không cho in ra màn hình nền.

Bài tập

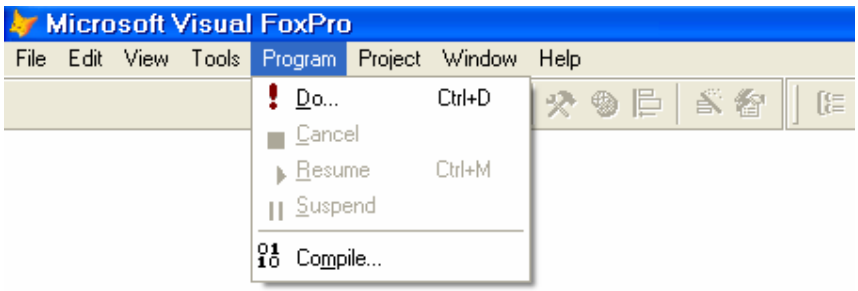
Tạo các report đã yêu cầu trong bài 1.

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Bài 11 - Tạo menu và toolbar

I. Giới thiệu

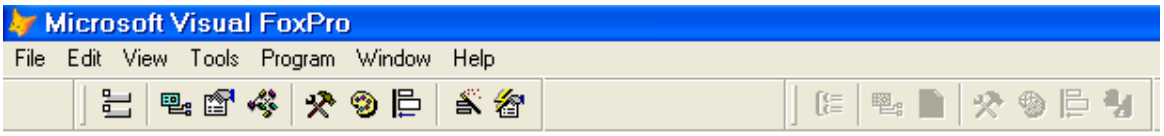
- Menu và Toolbar là các công cụ tiện lợi, khoa học cho việc tổ chức các chức năng của hệ thống.
- Menu, ví dụ menu của Visual Foxpro (hình sau), được dùng để tổ chức các chức năng của Visual Foxpro.



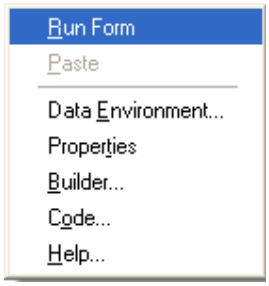
Trong đó:

  - Thanh nganh (chứa các mục File, Edit ...) gọi Menu Bar
  - Các mục như File, Edit ... trên Menu Bar gọi là PAD
  - Khi kích hoạt mỗi PAD (ví dụ như PAD Program trong hình trên)-> Xuất hiện một POPUP menu.
  - Mỗi mục trên POPUP menu (như Do, Cancel ... trong hình trên) được gọi là một BAR.

- ToolBar được dùng để tổ chức các chức năng thường xuyên được sử dụng. Ví dụ như Form Designer Toolbar của Visual Foxpro



- ShortCut menu là popup menu xuất hiện khi click chuột phải (Lên một đối tượng nào đó). Ví dụ shortcut sau xuất hiện khi right click trên nền form (khi thiết kế):



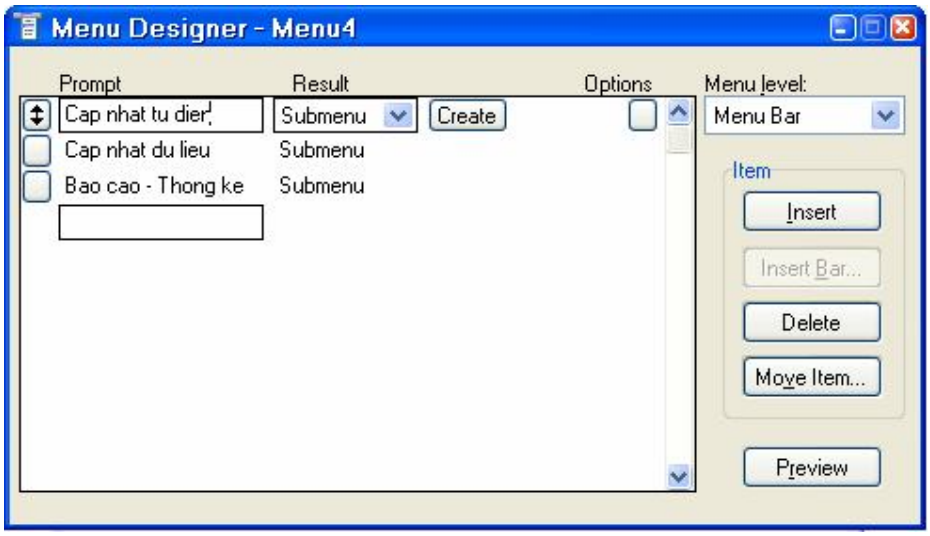
II. Thiết kế menu

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Các bước thực hiện

1. Mở Project cần làm, trong Project Manager chọn Tab Other
2. Chọn Menus
3. Chọn New, khi đó giao diện New menu xuất hiện như hình sau:

4. Chọn Menu. Khi đó giao diện Menu Designer xuất hiện như hình sau:



Giao diện Menu Designer

1. Menu level: Chỉ ra đối tượng (menu) đang thiết kế (ở khung bên trái) là đối tượng nào. Menu Bar (như hình trên) là mức cao nhất (Menu bar), hoặc các Submenu ...
2. Mỗi dòng trong khung thiết kế là một mục của Menu level. Ví dụ như hình trên thì “Cap nhat tu dien”, “Cap nhat du lieu” và “Bao cao - Thong ke” là các mục của Menu Bar.
3. Prompt: Tên các mục của menu (như là caption chứ không phải là định danh). Ví dụ như hình trên thì menu bar có 3 mục là: “Cap nhat tu dien”, “Cap nhat du lieu” và “Bao cao - Thong ke”.

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

4. Result: Hành động xảy ra khi một mục được chọn. Tham số này có thể nhận giá trị:. Command, PAD name, Submenu và Procedure, tùy thuộc vào Result mà việc thiết lập các tham số khác của một mục là khác nhau:
    a. Command: Nhập vào 1 lệnh của FOX;
    b. PAD Name: Tên một PAD menu (đã được tạo trước đó)
    c. Submenu (menu con): Lệnh Create (hoặc Edit) để sửa đổi menu con
    d. Procedure: Lệnh Create (hoặc Edit) để viết thủ tục
    e. BAR #: Tên một BAR (đã được tạo trước đó)
5. **Insert**: Chèn thêm 1 mục vào menu level
6. **Delete**: Xóa 1 mục khỏi menu level
7. **Move Item**: Di chuyển các mục
8. **Preview**: Xem thử menu đang thiết kế

Prompt Options

Với mỗi mục của menu có thể được thiết lập thêm các lựa chọn như phím nóng, hình ảnh ... để thiết lập các lựa chọn này chọn mục Option (ứng với mỗi mục), khi đó giao diện xuất hiện như hình sau:

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

1. **Key Label**: "Phím nóng" (ví dụ như Ctrl+C của Word).
2. **Key Text**: Text thể hiện của "phím nóng" (nên đặt trùng với Key Label).
3. **Negotiate**: Chỉ áp dụng đối với các PAD, và chỉ có hiệu ứng khi chạy ở chế độ OLE edit.
4. **Skip for**: Cho phép nhập vào 1 biểu thức logic, nếu biểu thức này có giá trị FALSE thì mục này sẽ Disabled.
5. **Message**: Thông báo ở Status bar khi di chuyển con trỏ đến mục này.
6. **Picture - File**: đường dẫn + File ảnh
7. **Picture - Resource**: Chọn ảnh theo các thiết kế sẵn của Visual Foxpro
8. **Comment**: Giải thích, ghi chú.

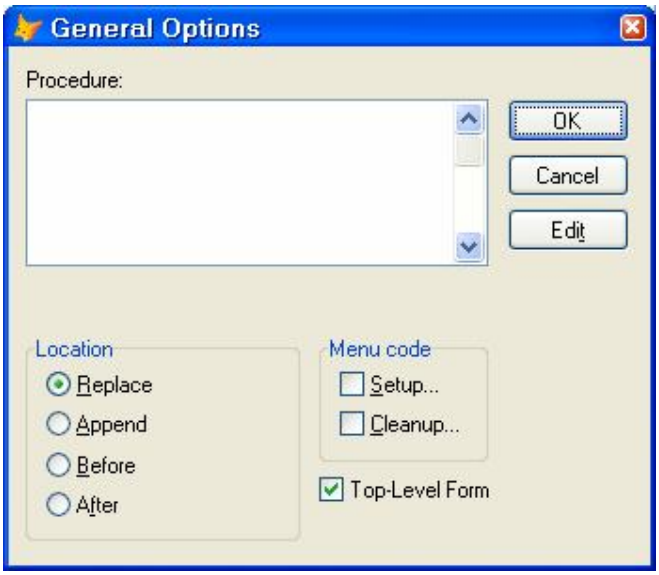
Ví dụ

Thiết kế menu theo yêu cầu trong bài 1 (demo trong Visual Foxpro).

Gắn một menu vào Form

Để gắn một menu vào form chúng ta thực hiện như sau:

1. Thiết lập thuộc tính **Top - Level Form** cho menu: Trong chế độ thiết kế menu (menu Designer) vào menu **View/General Option** khi đó giao diện chương trình có dạng:



   Đánh dấu (check) ô Top-Level Form như hình trên rồi chọn OK. Nếu không chọn Top-Level Form thì menu được tạo sẽ xuất hiện thay thế menu hệ thống.

2. Đối với Form:

   a. Thiết lập thuộc tính **ShowWindow** là **2 - As Top-Level Form**.

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

b. Trong sự kiện Init (Form.Init) viết dòng lệnh sau:

```
DO <MenuName> WITH THIS, .T.
```

Thực thi (chạy) một menu

Chỉ thực hiện đối với các menu không phải là **Top-Level Form** (xem phần trên)

Các bước thực hiện

1. Mở Project cần làm, trong Project Manager chọn Tab Other
2. Chọn Menus, và mở rộng
3. Chọn menu cần thực hiện
4. Chọn lệnh **Run**.

Hoặc thực hiện lệnh

```
DO <MenuName>
```

Lưu ý: Thực hiện lệnh **SET SYSMENU TO DEFAULT** để trả lại menu hệ thống của Visual Foxpro .

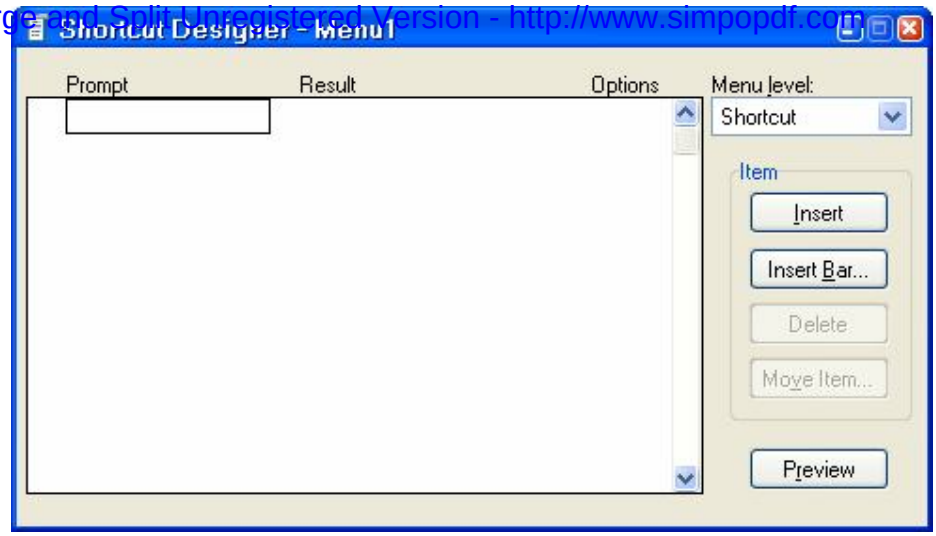
III. Thiết kế SHORTCUT

Các bước thực hiện

1. Mở Project cần làm, trong Project Manager chọn Tab Other
2. Chọn Menus
3. Chọn New, khi đó giao diện New menu xuất hiện như hình sau:

4. Chọn Shortcut. Khi đó giao diện Shortcut Designer xuất hiện như hình sau:

Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com



Giao diện Shortcut Designer

Tương tự giao diện **Menu Designer** vc

Sử dụn trong Form

Có thể thiết lập shortcut cho bất kỳ đối tượng nào trên form. Ví dụ cần thiết lập shortcut cho form, khi đó viết vào sự kiện rightClick của form dòng lệnh sau:

DO <TenShortCut>

IV. Thiết kế toolbar

Các bước thực hiện

1. Mở Project cần làm, trong Project Manager chọn Tab Class
2. Chọn New, khi đó giao diện New Class xuất hiện như hình sau:

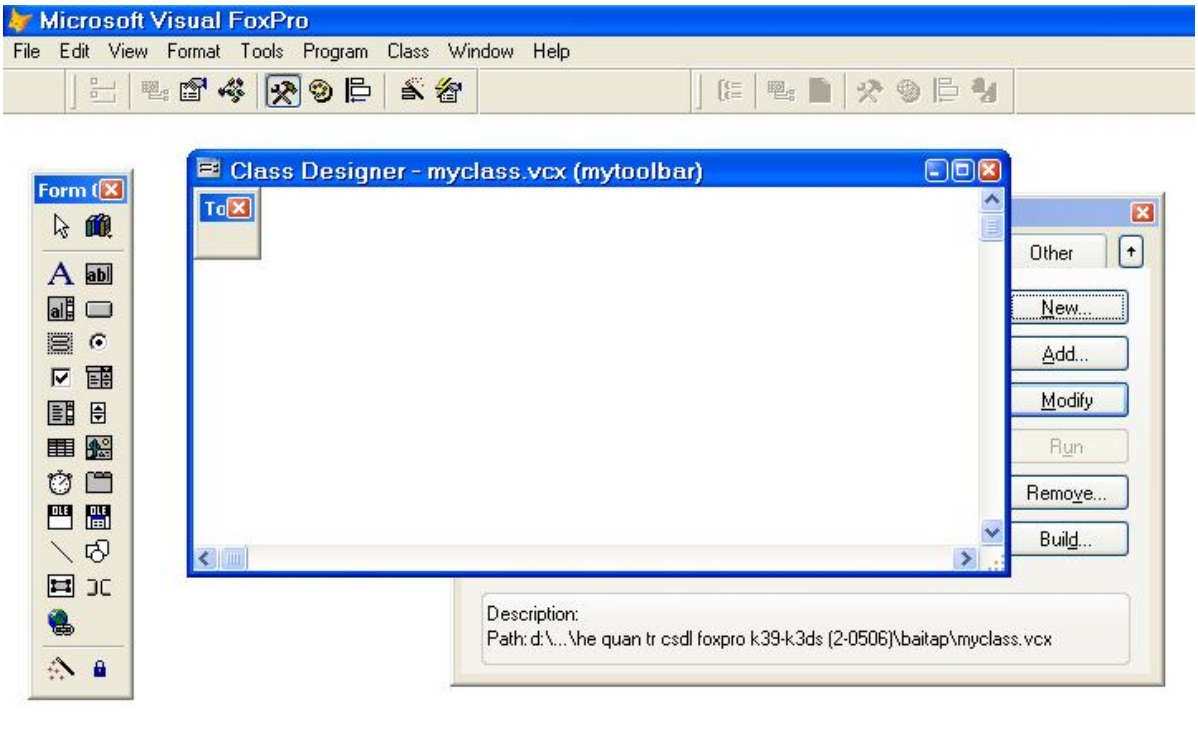
Trong đó:

- Class name: Tên lớp do người sử dụng đặt (trong hình trên đặt là myToolBar)
- Based On: Cho phép chọn 1 trong các lớp cơ sở của Visual Foxpro, trường hợp này cần tạo ToolBar nên chúng ta chọn ToolBar.

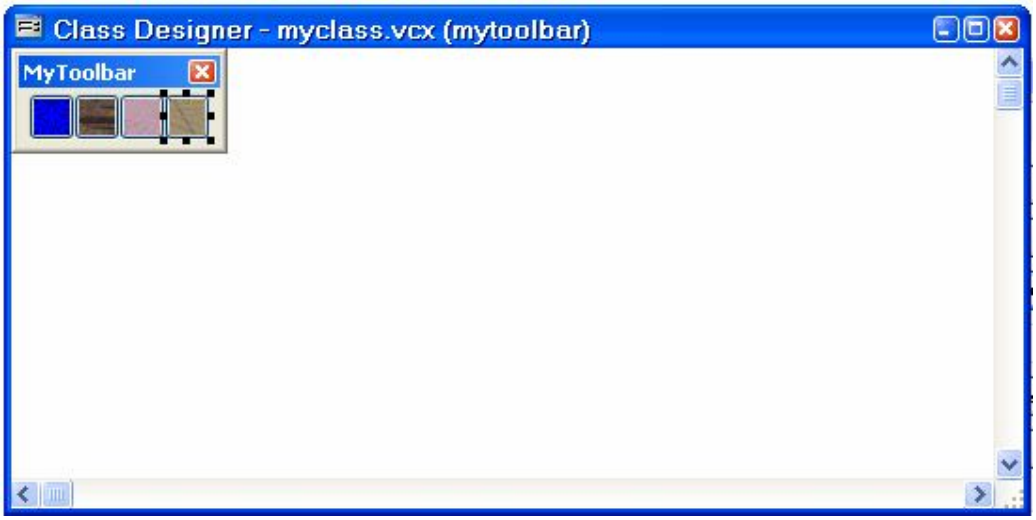
Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

- Store in: Tên thư viện do người sử dụng đặt (trong hình trên là MyClass), thư viện có phần mở rộng là VCX.

3. Sau khi đã thiết lập xong các tùy chọn (như trên) chọn OK để tiếp tục, khi đó giao diện Class Designer xuất hiện như hình sau:



4. Thiết kế Class nói chung và ToolBar nói riêng là tương đối giống với thiết kế Form. Trường hợp này là thiết kế ToolBar nên chỉ cần đưa các Command control vào, viết lệnh đáp ứng sự kiện cho chúng ... Ví dụ chúng ta thêm vào 4 nút lệnh khi đó ToolBar của chúng ta có dạng như sau:



Simpo PDF Merge and Split Unregistered Version - http://www.simpopdf.com

Đưa một toolbar vào form

Các bước thực hiện

1. Mở form cần có toolbar ở chế độ thiết kế.
2. Trong thanh công cụ **Form controls** chọn lệnh View Classes như hình sau:

3. Chọn lệnh Add, khi đó hộp thoại open file xuất hiện cho phép chọn file clas (*.VCX), ở đây chúng ta chọn file class vừa tạo ở bước trên. Khi đó thanh công cụ Form controls sẽ có dạng:

Trường hợp file class có nhiều đối tượng hơn thì thanh công cụ này sẽ khác.

4. "Nhắp - Thả" đối tượng mytoolbar vào form.

Lưu ý: đối tượng mytoolbar như chúng ta vừa thiết kế chỉ làm việc với Formset, do đó nếu chưa tạo Formset thì khi "Nhắp - thả" mytoolbar vào form chúng ta sẽ nhận được thông báo sau:

Chọn Yes để đồng ý hoặc No để hủy bỏ (không tạo formset và cũng không đưa mytoolbar vào form).

Bài tập

Tạo hệ thống Menu cho hệ quản lý điểm (như đã chỉ ra trong bài 1).

PGS.TS. Vũ Đức Thi

Giáo trình cơ sở dữ liệu

Bài Giảng

Hà Nội

Lời nói đầu

Cơ sở dữ liệu là một lĩnh vực phát triển mạnh của công nghệ thông tin. Cùng với sự phát triển công nghệ thông tin ở nước ta, việc sử dụng các kiến thức về cơ sở dữ liệu vào thực tiễn ngày càng trở lên cần thiết.

Trong bài giảng này chúng tôi cung cấp cho sinh viên những kiến thức cơ bản nhất về cơ sở dữ liệu. Mục tiêu chính là với số kiến thức cơ bản này sinh viên có thể ứng dụng các kiến thức về cơ sở dữ liệu vào thực tiễn và tiếp tục nghiên cứu học tập được các môn tin học khác.

Giáo trình gồm 4 chương chính (Ngoài chương mở đầu và tài liệu tham khảo).

Chương 2 cung cấp cho sinh viên những kiến thức cơ bản về cơ sở dữ liệu, mà cụ thể là về cơ sở dữ liệu quan hệ. Trong chương này, chúng tôi trình bày những khái niệm cơ bản nhất của cơ sở dữ liệu quan hệ, cũng như những thuật toán thiết kế chúng.

Chương 3 trình bày các kiến thức liên quan đến các dạng chuẩn.

Chương 4 giới thiệu các phép toán xử lí các bảng (quan hệ).

Chương 5 và chương 6 là các chương trình bày các ứng dụng của cơ sở dữ liệu vào thực tiễn.

Trong chương 5 chúng tôi nêu một số các ứng dụng của cơ sở dữ liệu trong các hệ quản trị cơ sở dữ liệu hiện có. Trong đó có những vấn đề liên quan đến các thực thể, các khoá, các dạng chuẩn trong các hệ quản trị cơ sở dữ liệu.

Chương 6 trình bày một số các công đoạn xây dựng các dự án thiết kế tổng thể các hệ thống thông tin.

Trong chương 7, chúng tôi trình bày một số các kiến thúc cơ bản về thuật toán và độ phức tạp thuật toán. Những kiến thức này giúp cho bạn đọc tiếp thu các kiến thức của các chương trên.

Giáo trình này phục vụ cho các sinh viên ngành công nghệ thông tin hoặc các cán bộ đang công tác trong lĩnh vực tin học muốn bổ xung kiến thức cho mình.

Tại tất cả các trường đại học có giảng dạy về tin học, cơ sở dữ liệu là môn học chính cho các sinh viên khoa công nghệ thông tin. Vì thế giáo trình này có thể làm tư liệu học tập cho sinh viên hệ cử nhân tin học, cử nhân cao đẳng tin học, kĩ sư tin học, hoặc có thể làm tài liệu tham khảo cho các học viên cao học, nghiên cứu sinh và các giảng viên tin học.

PGS.TS. Vũ Đức Thi

Chương mở đầu

Cơ sở dữ liệu (CSDL) là một trong những lĩnh vực được tập trung nghiên cứu và phát triển của công nghệ thông tin, nhằm giải quyết các bài toán quản lí, tìm kiếm thông tin trong những hệ thống lớn, đa dạng, phức tạp cho nhiều người sử dụng trên máy tính điện tử. Cùng với sự ứng dụng mạnh mẽ công nghệ thông tin vào đời sống xã hội, kinh tế, quốc phòng ...Việc nghiên cứu CSDL đã và đang phát triển ngày càng phong phú và hoàn thiện. Từ những năm 70, mô hình dữ liệu quan hệ do E.F. Codd đưa ra với cấu trúc hoàn chỉnh đã tạo lên cơ sở toán học cho các vấn đề nghiên cứu lí thuyết về CSDL. Với ưu điểm về tính cấu trúc đơn giản và khả năng hình thức hoá phong phú, CSDL quan hệ dễ dàng mô phỏng các hệ thống thông tin đa dạng trong thưc tiễn, tạo điều kiện lưu trữ thông tin tiết kiệm, có tính độc lập dữ liệu cao, dễ sửa đổi, bổ sung cũng như khai thác dữ liệu. Mặt khác, việc khai thác và áp dụng các kĩ thuật tổ chức và sử dụng bộ nhớ cho phép việc cài đặt các CSDL quan hệ đưa lại hiệu quả cao và làm cho CSDL quan hệ chiếm ưu thế trên thị trường.

Nhiều hệ quản trị CSDL đã được xây dựng và đưa vào sử dụng rộng rãi như : DBASE, **FOXBASE,**

FOXPRO, PARADOX, ORACLE, MEGA, IBM DB2, SQL for WINDOWS NT...

Mô hình dữ liệu quan hệ đặt trọng điểm hàng đầu không phải là khai thác các tiềm năng của máy mà ở sự mô tả trực quan dữ liệu theo quan điểm của người dùng, cung cấp một mô hình dữ liệu đơn giản, trong sáng, chặt chẽ, dễ hiểu và tạo khả năng tự động hoá thiết kế CSDL quan hệ. Có thể nói lí thuyết thiết kế và cài đặt CSDL, nhất là mô hình dữ liệu quan hệ đã phát triển ở mức độ cao và đạt được những kết quả sâu sắc. Hàng loạt vấn đề đã được nghiên cứu giải quyết như:

- Lí thuyết thiết kế CSDL, các phương pháp tách và tổng hợp các lược đồ quan hệ theo tiêu chuẩn không tổn thất thông tin hay bảo toàn tính nhất thể của các ràng buộc trên dữ liệu .

- Các loại ràng buộc dữ liệu, cấu trúc và các tính chất của chúng, ngữ nghĩa và khả năng áp dụng phụ thuộc dữ liệu ví dụ như phụ thuộc hàm, phụ thuộc đa trị, phụ thuộc kết nối, phụ thuộc lôgic...

- Các vấn đề tối ưu hoá: ở mức vật lí trong việc tổ chức quản lí các tệp; ở mức đường truy nhập với các tệp chỉ số hay các danh sách sắp xếp; ở mức lôgic trên cơ sở rút gọn các biểu thức biểu diễn các câu hỏi, ...vv

.....

Trong Giáo trình này sẽ trình bày một số kiến thức cơ bản nhất về CSDL bao gồm các kiến thức liên quan đến phụ thuộc hàm, khoá và dạng chuẩn, các thuật toán nhận dạng và thiết kế chúng, việc xây dựng các khái niệm này trong các hệ CSDL lớn như MEGA, ORACLE...., việc nghiên cứu và áp dụng chúng để xây dựng các dự án thiết kế tổng thể các hệ thống CSDL hiện nay.

Chương 2

Các kiến thức cơ bản về cơ sở dữ liệu

2.1. Khát quát về mô hình dữ liệu

Thông thường đối với việc thiết kế và xây dựng các hệ thông tin quản lí, chúng ta cần xử lí các file dữ liệu. Những file này bao gồm nhiều bản ghi (record) có cùng một cấu trúc xác định (loại bản ghi). Đồng thời, mỗi bản ghi được phân chia thành các trường dữ liệu (fild). Một cơ sở dữ liệu là một hệ thống các file dữ liệu, mỗi file này có cấu trúc bản ghi khác nhau, nhưng về mặt nội dung có quan hệ với nhau. Một hệ quản trị cơ sở dữ liệu là một hệ thống quản lí và điều hành các file dữ liệu. Nói chung một hệ quản trị cơ sở dữ liệu thường có những đặc tính sau :

- Có tính độc lập với các công cụ lưu trữ,

- Có tính độc lập với các chương trình phần mềm của người sử dụng (có nghĩa là các ngôn ngữ lập trình khác nhau có thể được dùng trong hệ này),

- Có khả năng tại một thời điểm truy nhập vào nhiều nơi trong hệ này ,

- Có khả năng khai thác tốt tiềm năng của máy,

- Người dùng với kiến thức tối thiểu cúng có thể xử dụng được hệ này,

- Bảo đảm an toàn dữ liệu và bảo mật dữ liệu,

- Thuận lợi và mềm dẻo trong việc bổ xung, loại bỏ, thay đổi dữ liệu

- Giảm bớt sự dư thừa dữ liệu trong lưu trữ,

Trong quá trình thiết kế và xây dựng các hệ quản trị cơ sở dữ liệu, người ta tiến hành xây dựng các mô hình dữ liệu. Mô hình dữ liệu phải thể hiện được các mối quan hệ bản chất của các dữ liệu mà các dữ liệu này phản ánh các mối quan hệ và các thực thể trong thế giới hiện thực. Có thể thấy mô hình dữ liệu phản ánh khía cạnh cấu trúc lôgic mà không đi vào khía cạnh vật lí của các cơ sở dữ liệu. Khi xây dựng các mô hình dữ liệu cần phân biệt các thành phần cơ bản sau :

- Thực thể (Entity): Đó là đối tượng có trong thực tế mà chúng ta cần mô tả các đặc trưng của nó.

- Thuộc tính: Đó là các dữ liệu thể hiện các đặc trưng của thực thể.

- Ràng buộc: Đó là các mối quan hệ lôgic của các thực thể.

Tuy vậy, ba thành phần cơ bản trên được thể hiện ở hai mức :

- Mức loại dữ liệu (Type): Đó là sự khái quát hoá các ràng buộc, các thuộc tính, các thực thể cụ thể.

- Mức thể hiện: Đó là một ràng buộc cụ thể, hoặc là các giá trị thuộc tính, hoặc là một thực thể cụ thể

Thông thường chúng ta sẽ nhận được các loại dữ liệu (Type) của các đối tượng cần khảo sát trong quá trình phân tích các thể hiện cụ thể của chúng.

yếu tố quan trọng nhất của cấu trúc cơ sở dữ liệu là dạng cấu trúc dữ liệu mà trong đó các mối quan hệ giữa các dữ liệu lưu trữ được mô tả. Có thể thấy rằng loại dữ liệu nền tảng của việc mô tả các mối quan hệ là loại bản ghi (Record type). Bởi vì các ràng buộc giữa các loại bản ghi tạo ra bản chất cấu trúc của cơ sở dữ liệu. Vì thế, dựa trên việc xác định các ràng buộc giữa các loại dữ lịêu được cho như thế nào mà chúng ta phân loại các mô hình dữ liệu. Có nghĩa là từ cách nhìn của người xử dụng việc mô tả các dữ liệu và các ràng buộc giữa các dữ liệu được thực hiện như thế nào. Trên thực tế chúng ta phân biệt hai loại mô hình dữ liệu:

- Mô hình dữ liệu mạng: Trong đó chúng ta thể hiện trực tiếp các ràng buộc tuỳ ý giữa các loại bản ghi,

- Mô hình dữ liệu quan hệ: Trong mô hình này các ràng buộc trên được thể hiện qua các quan hệ (bảng).

Mô hình dữ liệu quan hệ là một công cụ rất tiện lợi để mô tả cấu trúc lôgic của các cơ sở dữ liệu. Như vậy, ở mức lôgic mô hình này bao gồm các file được biểu diễn dưới dạng các bảng. Do đó đơn vị của CSDL quan hệ là một bảng (Một quan hệ được thể hiện trong Định nghĩa 1), trong đó các dòng của bảng là các bản ghi dữ liệu cụ thể (Đó là các thể hiện cụ thể của loại bản ghi), còn tên các cột là các thuộc tính.

Theo cách nhìn của người xử dụng thì một cơ sở dữ liệu quan hệ là một tập hợp các bảng biến đổi theo thời gian.

2.2. Các khái niệm cơ bản và hệ tiên đề Armstrong:

Trong mục này, chúng ta trình bày những khái niệm cơ bản nhất về mô hình dữ liệu quan hệ của E.F. Codd. Những khái niệm cơ bản này gồm các khái niệm về quan hệ, thuộc tính, phụ thuộc hàm, hệ tiên đề Armstrong, khóa, dạng chuẩn....

Những khái niệm này đóng vai trò rất quan trọng trong mô hình dữ liệu quan hệ. Chúng được áp dụng nhiều trong việc thiết kế các hệ quản trị cơ sở dữ liệu hiện nay.

Những khái niệm này có thể tìm thấy trong [1,2,3,4,7,9,10,15,16,17].

Định nghĩa 1. (Quan hệ)

Cho R = $\{a_{1, \dots}, a_n\}$ là một tập hữu hạn và không rỗng các thuộc tính. Mỗi thuộc tính a_i có miền giá trị là D_{ai}. Khi đó r là một tập các bộ $\{h_1, \dots, h_m\}$ được gọi là một quan hệ trên R với h_j (j = 1,...m) là một hàm :

$$h_j : R \rightarrow \cup D_{ai}$$

$$a_i \in R$$

sao cho: $h_j(a_i) \in D_{ai}$

Chúng ta có thể biểu diễn quan hệ r thành bảng sau:

	a_1	a_2		a_n
h_1	$h_1(a_1)$	$h_1(a_2)$	.....	$h_1(a_n)$
h_2	$h_2(a_1)$	$h_2(a_2)$	.....	$h_2(a_n)$
.	.....			

.

.

h_m $h_m(a_1)$ $h_m(a_2)$ ..... $h_m(a_n)$

Ví dụ: Trong một cơ quan, chúng ta quản lý nhân sự theo biểu gồm các thuộc tính sau:

Nhân sự

Số TT	Họ tên	Giới tính	Năm sinh	Trình độ đào tạo	Lương
001	Nguyễn Văn A	Nam	1970	Đại học	300000
002	Nguyễn Kim Anh	Nữ	1971	Trung cấp	210000
003	Trần Văn ánh	Nam	1969	Đại học	500000
004	Trần Bình	Nam	1965	PTS	450000
.....					
120	Trần Thị yến	Nữ	1967	PTS	455000

Chúng ta quy định kích thước cho các thuộc tính (các trường) như sau:

Tên thuộc tính	Kiểu	Kích thước
STT	Kí tự	3
HOTEN	Ký tự	30
GIOITINH	Ký tự	3
NAMSINH	Số	4
TRINHDO	Ký tự	10
LUONG	Số	7

Có nghĩa là qui định cho thuộc tính STT là các dãy gồm 3 kí tự, thuộc tính HOTEN là các dãy gồm 30 kí tự, ..., cho thuộc tính LUONG là các số có nhiều nhất 7 chữ số.

Như vậy chúng ta có tập thuộc tính

NHANSU = {STT, HOTEN, GIOITINH, NAMSINH, TRINHDO, LUONG}

ở đây D_{STT} là tập các dãy gồm 3 kí tự,...., D_{LUONG} là tập các số có nhiều nhất 7 chữ số.

Khi đó chúng ta có quan hệ r = {h_1, h_2,..., h_{120}}, ở đây ví dụ như đối với bản ghi thứ 2 (dòng thứ 2) chúng ta có:

h_2 (STT) = 002, h_2 (HOTEN) = Nguyễn Kim ánh

h_2 (GIOITINH) = Nữ, h_2 (NAMSINH) = 1971

h_2 (TRINHDO) = Trung cấp, h_2 (LUONG) = 240000

Định nghĩa 2. (Phụ thuộc hàm)

1. Cho R = $\{a_1,...,a_n\}$ là tập các thuộc tính, r = $\{h_1,...,h_m\}$ là một quan hệ trên R, và A, B $\subseteq$ R.

2. Khi đó chúng ta nói A xác định hàm cho B hay B phụ thuộc hàm vào A trong r (Kí pháp A $\overset{f}{\underset{r}{\longrightarrow}}$ B) nếu

$(\forall h_i,h_j \in r)((\forall a \in A)(h_i(a)= h_j(a)) \Rightarrow (\forall b \in B)(h_i(b)=h_j(b)))$

Đặt F_r = { (A,B): A,B $\subseteq$ R, A $\overset{f}{\underset{r}{\longrightarrow}}$ B }. Lúc đó F_r được gọi là họ đầy đủ các phụ thuộc hàm của r.

Khái niệm phụ thuộc hàm miêu tả một loại ràng buộc (phụ thuộc dữ liệu) xảy ra tự nhiên nhất giữa các tập thuộc tính. Dù hiện nay đã có nhiều loại phụ thuộc dữ liệu được nghiên cứu, xong về cơ bản các hệ quản trị cơ sở dữ liệu lớn sử dụng phụ thuộc hàm.

Định nghĩa 3.

Phụ thuộc hàm (PTH) trên tập các thuộc tính R là một dãy kí tự có dạng A $\rightarrow$B, ở đây A,B $\subseteq$ R. Chúng

ta nói PTH $A \rightarrow B$ đúng trong quan hệ r if $A \overset{f}{\underset{r}{>}} B$. Chúng ta cũng nói rằng r thoả mãn

$A \rightarrow B$.

Dễ thấy, F_r là tập tất cả các PTH đúng trong r.

Chú ý: Trong giáo trình này chúng ta có thể viết (A,B) hoặc $A \rightarrow B$ thay cho $A \overset{f}{\underset{r}{>}} B$ mà không bị lẫn về mặt kí pháp.

Định nghĩa 4. (Hệ tiên đề của Armstrong)

Giả sử R là tập các thuộc tính và kí pháp P(R) là tập các tập con của R. Cho $Y \subseteq P(R) \times P(R)$. Chúng ta nói Y là một họ f trên R nếu đối với mọi A, B, C, $D \subseteq R$

(1) $(A,A) \in Y$,

(2) $(A,B) \in Y, (B,C) \in Y \Rightarrow (A,C) \in Y$,

(3) $(A,B) \in Y, A \subseteq C, D \subseteq B \rightarrow (C,D) \in Y$,

(4) $(A,B) \in Y, (C,D) \in Y \Rightarrow (A \cup C, B \cup D) \in Y$.

Rõ ràng, F_r là một họ f trên R.

Trong [l] A. A. Armstrong đã chứng minh một kết quả rất quan trọng như sau : Nếu Y là một họ f

bất kì thì tồn tại một quan hệ r trên R sao cho F_r = Y.

Kết quả này cùng với định nghĩa của phụ thuộc hàm chứng tỏ rằng hệ tiên đề Armstrong là đúng đắn và đầy đủ.

Mặt khác, hệ tiên đề này cho ta những đặc trưng của họ các phụ thuộc hàm, mà các đặc trưng này không phụ thuộc vào các quan hệ (bảng) cụ thể . Nhờ có hệ tiên đề này các công cụ của toán học được áp dụng để nghiên cứu làm sáng tỏ cấu trúc lôgic của mô hình dữ liệu quan hệ. Đặc biệt chúng ta xử dụng công cụ thuật toán để thiết kế các công đoạn xây dựng các hệ quản trị cơ sở dữ liệu.

Chúng ta đưa ra ví dụ chỉ ra có nhiều quan hệ khác nhau xong các họ đầy đủ các phụ thuộc hàm của chúng lại như nhau.

Cho r_1 và r_2 là các quan hệ sau:

r_1 =

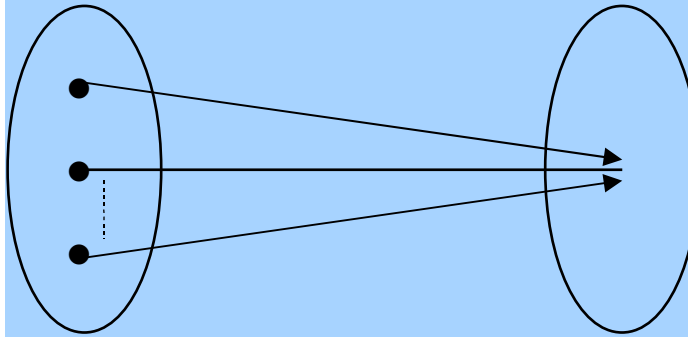
a	b
0	0
1	1
2	1
3	2

r_2 =

a	b
0	0
1	1
2	1
3	1

Có thể thấy r_1 và r_2 khác nhau nhưng $F_{r1} = F_{r2}$.

Như vậy, tương quan giữa lớp các quan hệ với lớp các họ phụ thuộc hàm có thể được thể hiện bằng hình vẽ sau.



Lớp các quan hệ Lớp các phụ thuộc hàm

Định nghĩa 5.

Một hàm L : P(R) $\rightarrow$ P(R) được gọi là một hàm đóng trên R nếu với mọi A, B $\in$ P(R) thì :

- A $\subseteq$ L(A),
- Nếu A $\subseteq$ B thì L(A) $\subseteq$ L(B),
- L(L(A)) = L(A).

Địnhlí 6.

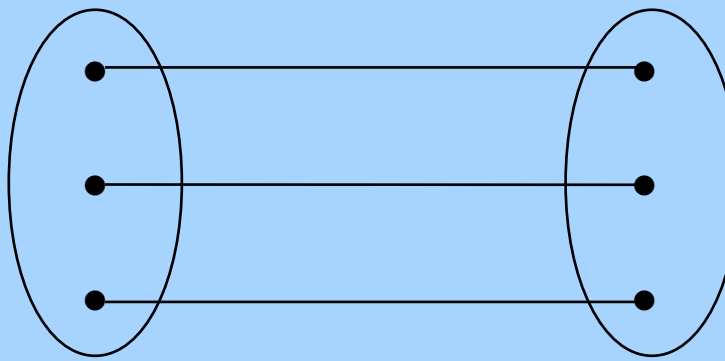
Nếu F là một họ f và chúng ta đặt

$L_F = \{a : a \in R$ và $(A, \{a\}) \in F\}$

thì L_F là một hàm đóng. Ngược lại, nếu L là một hàm đóng thì tồn tại duy nhất một họ f F trên R sao cho $L = L_F$, ở đây

$F = \{ (A,B) : A, B \subseteq R , B \subseteq L(A) \}$.

Như vậy, chúng ta thấy có một tương ứng 1-1 giữa lớp các hàm đóng và lớp các họ f . Chúng ta có hình vẽ sau



Lớp các họ phụ thuộc hàm
Lớp các hàm đóng

Định lí 6 chỉ ra rằng để nghiên cứu phân tích các đặc trưng của họ các phụ thuộc hàm chúng ta có thể dùng công cụ hàm đóng.

Sau này trong mục 2.3 chúng tôi sẽ trình bày nhiều công cụ nữa để nghiên cứu cấu trúc lôgic của họ các phụ thuộc hàm.

Định nghĩa 7. (Sơ đồ quan hệ)

Chúng ta gọi sơ đồ quan hệ (SĐQH) s là một cặp <R,F>, ở đây R là tập các thuộc tính và F là tập các phụ thuộc hàm trên R. Kí pháp F^+ là tập tất cả các PTH được dẫn xuất từ F bằng việc áp dụng các qui tắc trong Định nghĩa 4.

Đặt $A^+ = \{a: A \rightarrow \{a\} \in F^+\}$. A^+ được gọi là bao đóng của A trên s.

Có thể thấy rằng $A \rightarrow B \in F^+$ nếu và chỉ nếu $B \subseteq A^+$.

Tương tự chúng ta đặt $A_r^+ = \{a : A \overset{f}{\underset{r}{>}} \{a\}\}$. A_r^+ được gọi là bao đóng của A trên r.

Theo [1] chúng ta có thể thấy nếu s = <R,F> là sơ đồ quan hệ thì có quan hệ r trên R sao cho $F_r=F^+$. Quan hệ r như vậy chúng ta gọi là quan hệ Armstrong của s.

Trong trường hợp này hiển nhiên các PTH của s đúng trong r.

Định nghĩa 8. (Khoá)

Giả sử r là một quan hệ , s = <R,F> là một sơ đồ quan hệ, Y là một họ f trên R, và $A \subseteq R$. Khi đó A là một khoá của r (tương ứng là một khoá của s, một khoá của Y) nếu $A \overset{f}{\underset{r}{>}} R$ ($A \rightarrow R \in F^+$, $(A,R) \in Y$). Chúng ta gọi A là một khoá tối tiểu của r (tương ứng của s, của Y) nếu

- A là một khoá của r (s, Y),

- Bất kì một tập con thực sự của A không là khoá của r (s, Y).

Chúng ta kí pháp K_r, (K_s, K_y) tương ứng là tập tất cả các khoá tối tiểu của r (s, Y).

Chúng ta gọi K (ở đây K là một tập con của P(R)) là một hệ Sperner trên R nếu với mọi $A,B \in K$ kéo theo $A \subseteq B$).

Có thể thấy K_r,K_s, K_y là các hệ Sperner trên R.

Định nghĩa 9.

Giả sử K là một hệ Sperner trên R. Chúng ta định nghĩa tập các phản khoá của K, kí pháp là K^{-1}, như sau:

$K^{-1} = \{A \subset R : (B \in K) \Rightarrow (\not\exists B \subseteq A)$ and $(A \subset C) \Rightarrow (\exists B \in K)(B \subseteq C)\}$

Dễ thấy K^{-1} cũng là một hệ Sperner trên R.

Tập phản khoá đóng vai trò rất quan trọng trong quá trình nghiên cứu cấu trúc lôgic của các họ phụ thuộc hàm, khóa, dạng chuẩn, quan hệ Armstrong, đặc biệt đối với các bài toán tổ hợp trong mô hình dữ liệu quan hệ.

Trong [5] người ta đã nêu ra rằng nếu s = <R,F> là một sơ đồ quan hệ trên R, thì K_s là hệ Sperner trên R. Ngược lại, nếu K là một hệ Sperner bất kì trên R, thì tồn tại một sơ đồ quan hệ s sao cho $K_s = K$.

Ví dụ: Cho $K = \{ A_1,.....,A_m \}$ là một hệ Sperner. Khi đó s = <R,F>, ở đây $F = \{ A_1 \rightarrow R, ... , A_m \rightarrow R\}$ là sơ đồ quan hệ mà $K_s = K$.

Nhận xét :

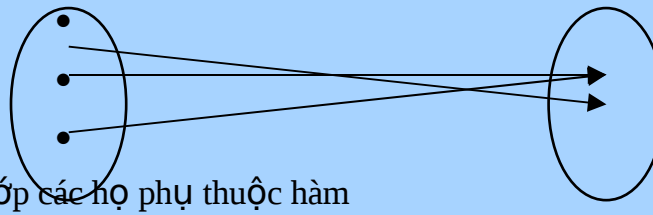
- Có thể cho ví dụ chỉ ra rằng có nhiều sơ đồ quan hệ khác nhau nhưng tập các khoá tối tiểu của chúng giống nhau. Có nghĩa là tồn tại

$s_1 = <R,F_1>, ..., s_t = <R,F_t>$ $(2 \leq t)$ mà $F_1^+ \neq .... \neq F_t^+$, nhưng

$K_{s1} = ... = K_{st}$.

Tất nhiên, nếu $F_1 = F_2$ thì $K_{s1} = K_{s2}$.

Mối quan hệ giữa lớp họ phụ thuộc hàm và lớp các hệ Sperner thể hiện qua hình vẽ sau



Lớp các họ phụ thuộc hàm
Lớp các hệ Sperner

- Nếu K đóng vai trò là một tập các khoá tối tiểu của một sơ đồ quan hệ nào đó, thì theo định nghĩa K^{-1} là tập tất cả các tập không phải khoá lớn nhất.

Trong Giáo trình này, chúng ta qui ước rằng nếu hệ Sperner đóng vai trò là tập các khoá tối tiểu (tập các phản khoá), thì hệ này không rỗng (không chứa R).

Định nghĩa 10.

Cho $I \subseteq P(R)$. Khi đó I được gọi là nửa dàn giao nếu

$R \in I$ và $A,B \in I \Rightarrow A \cap B \in I$.

Giả sử $M \subseteq P(R)$. Đặt $M^+ = \{\cap M' : M' \subseteq M\}$. Khi đó chúng ta nói rằng M là một hệ sinh của I nếu $M^+ = I$.

Chú ý rằng $R \in M^+$ nhưng R không là một phần tử của M, bởi vì chúng ta theo thông lệ cho R là giao của một tập rỗng các tập con của M.

Kí pháp $N_I = \{A \in I : A \neq \cap \{A' \in I : A \subset A'\}\}$.

Trong [4] người ta đã chỉ ra rằng N_I là hệ sinh nhỏ nhất và duy nhất của I. Có nghĩa là đối với mọi hệ sinh N' của I chúng ta có $N_I \subseteq N'$.

Định nghĩa 11.

Cho r là một quan hệ trên R. Chúng ta đặt $E_r = \{E_{ij} : 1 \leq i \leq j \leq |r|\}$, ở đây $E_{ij} = \{a \in R: h_i(a) = h_j(a)\}$. E_r được gọi là hệ bằng nhau của r.

Đặt $M_r = \{ A \in P(R) : \exists E_{ij} = A, \nexists E_{pq}: A \subset E_{pq}\}$. Khi đó chúng ta gọi M_r là hệ bằng nhau cực đại của r.

Sau này ta sẽ thấy hệ bằng nhau và hệ bằng nhau cực đại được dùng rất nhiều trong các thuật toán thiết kế.

Mối quan hệ giữa lớp các quan hệ và lớp các phụ thuộc hàm đóng một vai trò quan trọng trong quá trình nghiên cứu cấu trúc lôgic của lớp các phụ thuộc hàm.

Định nghĩa 12.

Cho trước r là một quan hệ r và F là một họ f trên R. Chúng ta nói rằng r là thể hiện họ F nếu $F_r = F$. Chúng ta cũng có thể nói r là một quan hệ Armstrong của F.

Bây giờ, chúng ta đưa ra một điều kiện cần và đủ để một quan hệ là thể hiện một họ f cho trước.

Định lý 13.

Giả sử $r = \{h_1, ..., h_m\}$ là một quan hệ, F là một họ f trên R thì r thể hiện F nếu và chỉ nếu với mọi $A \subseteq R$

$$L_F(A) = \begin{cases} \cap E_{ij} \text{ nếu tồn tại } E_{ij} \in E_r : A \subseteq E_{ij}, \\ A \subseteq E_{ij} \\ R \quad \text{ngược lại.} \end{cases}$$

ở đây $L_F(A) = \{a \in R : (A, \{a\}) \in F\}$ và E_r là hệ bằng nhau của r.

Lời giải: Đầu tiên chúng ta chứng minh rằng trong một quan hệ r bất kì với mọi $A \subseteq R$

$$L_{Fr}(A) = \begin{cases} \cap E_{ij} \text{ nếu tồn tại } E_{ij} \in E_r : A \subseteq E_{ij}, \\ A \subseteq E_{ij} \\ R \quad \text{ngược lại.} \end{cases}$$

Giả sử đầu tiên chúng ta công nhận rằng A là một tập mà không có $E_{ij} \in E_r$ với $A \subseteq E_{ij}$ với mọi $h_i, h_j \in r$, $a \in A : h_i(a) = h_j(a)$. Theo định nghĩa của phụ thuộc hàm điều này kéo theo $A \rightarrow R$ và bởi định nghĩa của L_{Fr} ta thu được $L_F(A) = R$. Rõ ràng là

$$L_{Fr}(\emptyset) = \cap E_{ij}$$

$$E_{ij} \in E_r$$

Nếu $A \neq \varnothing$ và có một $E_{ij} \in E_r$ mà $A \subseteq E_{ij}$ thì chúng ta đặt

$$V = \{ E_{ij} : A \subseteq E_{ij}, E_{ij} \in E_r \}$$

và $E = \cap E_{ij}$

$$E_{ij} \in V$$

Dễ dàng nhận thấy rằng $A \subseteq E$. Nếu $V = E_r$ thì chúng ta nhận thấy rằng $(A,E) \in F_r$ nếu $V \neq E_r$ thì có thể coi như với mọi $E_{ij} \in V$ chúng ta có

(Với mọi $a \in A$) $(h_i(a) = h_j(a)) \rightarrow$ (Với mọi $b \in B$) $(h_i(b) = h_j(b))$ và với mọi $E_{ij} \notin V$ có một $a \in A$ mà $h_i(a) \neq h_j(a)$. Như vậy, $(A, E) \in F_r$.

Từ định nghĩa của L_{Fr} ta có $E \subseteq L_{Fr}(A)$. bởi vì r là một quan hệ trên R, chúng ta có $E \subset R$. Sử dụng $A \subseteq E \subseteq L_{Fr}(A)$ ta thu được $(E, L_{Fr}(A)) \in F_r$.

Bây giờ, ta giả sử rằng c là một thuộc tính mà $c \notin E$. Khi đó có một $E_{ij} \in V$ mà $c \notin E_{ij}$. Điều này kéo theo sự tồn tại của một cặp $h_i, h_j \in r$ mà với mọi $b \in E$: $h_i(b) = h_j(b)$ nhưng $h_i(c) \neq h_j(c)$. Có thể thấy rằng theo định nghĩa phụ thuộc hàm $(E \cup \{c\})$ không phụ thuộc vào E. Như vậy, với mọi thuộc tính $c \notin E$ ta có $(E, E \cup \{c\}) \notin F_r$. Bằng định nghĩa của L_{Fr} ta thu được :

$$L_{Fr}(A) = \bigcap_{E_{ij} \in V} E_{ij}$$

Trên cơ sở Định lí 6 chúng ta dễ dàng thấy rằng $F_r = F$ nếu và chỉ nếu $L_{Fr} = L_F$. □

Giả sử L là một hàm đóng. Đặt $Z(L) = \{A \subseteq R : L(A) = A\}$.

Rõ rằng, Z(L) là tập đóng với phép giao.

Có thể thấy là với mọi E_{ii} ($E_{ii} \in E_r$), chúng ta có $E_{ii} \in Z(L_{Fr})$, có nghĩa là $E^+_r \subseteq Z(L_{Fr})$

Nhờ Định lý 13 chúng ta có $Z(L_{Fr}) \subseteq E^+_R$. Như vậy chúng ta có

Hệ quả 14.

Giả sử r quan hệ, F là một họ *f* trên R. Khi đó r thể hiện F nếu và chỉ nếu $Z(L_F) = E^+_R$.

Trong [5] người ta đã chỉ ra rằng nếu cho một hệ Sperner không rỗng tuỳ ý K thì tồn tại một quan hệ r để $K = K_r$.

Bây giờ, chúng ta đưa ra một định nghĩa dưới đây

Định nghĩa 15.

Cho trước quan hệ r và hệ Sperner K trên R. Chúng ta nói rằng r thể hiện K nếu $K_r = K$.

Định nghĩa 16.

Cho F là một họ f trên R, và (A,B) là một phần tử của F. Chúng ta nói (A,B) là một phụ thuộc có vế phải cực đại của F nếu với mọi B' (B B') và (A,B') ∈ F kéo theo B = B'.

Chúng ta kí pháp M(F) là tập tất cả các phụ thuộc có vế phải cực đại của F. Chúng ta nói rằng B là vế phải cực đại của F nếu có A sao cho (A,B) ∈ M(F). Kí pháp I(F) là tập tất cả các vế phải cực đại của F.

Dưới đây chúng ta cho một điều kiện cần và đủ để một quan hệ thể hiện một hệ Sperner.

Định lý 17.

Giả sử K là một hệ Sperner không rỗng, r là một là một quan hệ trên R. Khi đó r thể hiện K nếu và chỉ nếu $K^{-1} = M_r$, ở đây M_r là hệ bằng nhau cực đại của r.

Lời giải: Có thể xem như là nếu K là một hệ Sperner không rỗng thì K^{-1} tồn tại. Mặt khác, K và K^{-1} là xác định duy nhất. Cho nên, chúng ta có $K_r = K$ nếu và chỉ nếu $K_r^{-1} = K^{-1}$.

Bây giờ chúng ta có thể chỉ cần chứng minh rằng $K_r^{-1} = M_r$. Rõ ràng, F_r là một họ f trên R. Đầu tiên chúng ta có thể giả thiết rằng A là một phản khoá của K_r. Rõ ràng A ≠ R. Nếu có một B sao cho

$A \subset B$ và $A \rightarrow B$, thì bằng định nghĩa của phản khoá, chúng ta có $B \rightarrow R$ và $A \rightarrow R$. Đây là một điều phi lý. Vì vậy $A \in I(F_r)$. Nếu có một B' sao cho $B' \neq R$, $B' \in I(F_r)$ và $A \subset B'$, thì B' là một khoá của r. Đây là một điều mâu thuẫn $B' \neq R$. Do đó $A \in I(F_r) - R$ và không tồn tại B' ($B' \in I(F_r) - R$) để $A \subset B'$.

Mặt khác, theo định nghĩa của một quan hệ, $R \notin M_r$. Rõ ràng, $E_{ij} \in I(F_r)$. Như vậy chúng ta có $M_r \subseteq I(F_r)$. Nếu Dlà một tập sao cho $\forall C \in M_r : D \subseteq C$, thì D là một khoá của r. Bởi vậy, M_r là tập phần tử rời nhau cực đại của $I(F_R)$. Vì vậy chúng ta có $A \in M_r$

Ngược lại, nếu $A \in M_r$, thì theo định nghĩa của quan hệ và M_r Chúng ta có $A \rightarrow R$. Có nghĩa là $\forall K \in K_r : K \subseteq A$. Mặt khác, bởi vì A là một phần tử của tập bằng nhau cực đại, cho nên đối với tất cả D($A \subset D$) chúng ta có $D \rightarrow R$. Đồng thời theo định nghĩa của các phản khoá $A \in K^{-1}_r$. □

Cho trước s = <R,F> là một sơ đồ quan hệ trên R, K_s là tập tất cả các khoá tối tiểu của s. Kí pháp K_s^{-1} là tập các phản khoá của s. Từ Định lí 17 chúng ta có kết quả sau.

Hệ quả 18.

Cho trước s = <R,F> là một sơ đồ quan hệ và r là một quan hệ trên R. Khi đó $K_r = K_s$ nếu và chỉ nếu $K_s^{-1} = M_r$, ở đây M_r là hệ bằng nhau cực đại của r.

Chúng ta đưa ra một kết quả liên quan đến cả K^{-1} và K.

Định lý 19.

Giả sử K là một hệ Sperner trên R. Giả sử s(K) = min{m:K=K_r, |r|=m, r là quan hệ trên R}. Khi đó

$$\sqrt{2|K^{-1}|}\,2 \le s(K) \le |K^{-1}|+1.$$

Đánh giá này chỉ ra mối quan hệ giữa kích cỡ của quan hệ tối tiểu mà thể hiện một hệ Sperner (ở đây hệ này đóng vai trò là một hệ khoá tối tiểu) cho trước với lực lượng của hệ phản khoá tương ứng của nó.

Cho F là một họ f trên R. Theo Định nghĩa 16 thì dễ thấy I(F) là một nửa dàn giao. Khi đó

$$N_{I(F)} = \{A \in I(F) : A \neq \cap \{A' \in I : A \subset A'\}\}.$$

Trên cơ sở này chúng ta có định lí sau đánh giá quan hệ Armstrong nhỏ nhất (minimal Armstrong relation) của một họ f.

Định lý 20.

Giả sử F là một họ f trên R. Đặt

s(F) = min{m:F=F_r, |r|=m, r là quan hệ trên R}. Khi đó $\sqrt{2|N_{I(F)}|} \le s(F) \le |N_{I(F)}|+1$.

Đánh giá này cho chúng ta mối quan hệ giữa kích thước của quan hệ Armstrong nhỏ nhất của họ F với lực lượng của hệ sinh nhỏ nhất của I(F).

Bây giờ, chúng ta đánh giá sâu hơn kích thước của các quan hệ Armstrong nhỏ nhất trên R, cũng như các quan hệ nhỏ nhất mà thể hiện một hệ Sperner K cho trước.

Chúng ta đặt

P(n) = max { s(K) : K là hệ Sperner tùy ý trên R = $\{a_1,..., a_n\}$ }

và Q(n) = max { s(F) : F là họ f tùy ý trên R = $\{a_1,..., a_n\}$ }

Khi đó chúng ta có

Định lý 21.

- $1/n^2 . C_{[n/2]}^n \leq P(n) \leq C_{[n/2]}^n + 1$

- $1/n^2 . C_{[n/2]}^n \leq Q(n) \leq (1 + C(1/n^{1/2})).C_{[n/2]}^n$

Đánh giá này cho toàn bộ các hệ Sperner (ở đây hệ này đóng vai trò là một hệ khoá tối tiểu) và các họ f có thể có trên R.

Định nghĩa 22.

Giả sử r là một quan hệ trên R và K_r là tập của tất cả các khoá tối tiểu của r. Chúng ta nói rằng a là

một thuộc tính cơ bản của r nếu tồn tại một khoá tối tiểu K ($K \in K_r$) để a là một phần tử của K.

Nếu a không thoả mãn tính chất trên thì a là thuộc tính thứ cấp.

Trong chương 3 chúng ta có thể thấy các thuộc tính cơ bản và thứ cấp đóng một vai trò quan trọng trong việc chuẩn hoá các sơ đồ quan hệ và các quan hệ.

Trong [24] đã chứng minh kết quả sau

Định lí 23.

Cho trước một sơ đồ quan hệ s = <R,F> và một thuộc tính a. Bài toán xác định a là thuộc tính cơ bản hay không là bài toán NP- đầy đủ.

Có nghĩa rằng cho đén nay không có một thuật toán có độ phức tạp thời gian đa thức để giải quyết bài toán này.

Tuy vậy, chúng ta chỉ ra rằng đối với quan hệ thì bài toán này được giải bằng một thuật toán thời gian đa thức.

Trước tiên chúng ta chứng minh kết quả sau.

Định lý 24.

Giả sử K là một hệ Sperner trên R. thì

$$\cup K = R - \cap K^{-1}.$$

Lời giải:

Nếu $c \in \cup K$, thì tồn tại một khoá tối tiểu K sao cho $c \in K$. Đặt $H = K - c$. Rõ ràng H không chứa một khoá nào. Như vậy, tồn tại một phản khoá B để B chứa H. Có thể thấy c không là phần tử của B, vì ngược lại chúng ta có B chứa K. Điều này là vô lí. Vì thế chúng ta có

$$c \in R - B \subseteq R - \cap K^{-1}.$$

Bây giờ chúng ta giả thiết $c \notin \cup K$ và $B \in K^{-1}$. Có thể thấy $c \in B$. Vì ngược lại $c \notin B$, thì $\{c\} \cup B$ hình thành một khoá chứa khoá tối tiểu K ($K \in K$). Như vậy $K \subseteq B$, và chúng ta có $c \in K$. Điều này là vô lý. □

Trên cơ sở của Định lý 17 và Định lí 24 chúng ta chỉ ra rằng đối với một quan hệ, thì vấn đề về thuộc tính cơ bản có thể là giải quyết bằng một thuật toán thời gian đa thức.

Đầu tiên chúng ta xây dựng một thuật toán xác định tập các thuộc tính cơ bản của quan hệ cho trước.

Thuật toán 25.

Vào: $r = \{h_1, ..., h_m\}$ là một quan hệ trên R

Ra: V là tập tất cả thuộc tính cơ bản của r

Bước 1: Từ r chúng ta xây dựng một tập $E_r = \{E_{ij} : m \geq j > i \geq 1\}$ và $E_{ij} = \{ a \in R : h_j(a) = h_j(a) \}$

Bước 2 : Từ E_r chúng ta xây dựng tập

$M = \{B \in P(R) :$ Tồn tại $E_{ij} \in E_r : E_{ij} = B\}$

Bước 3: Từ M xây dựng tập $M_r = \{ B \in M :$ Với mọi $B' \in M : B \not\subset B'\}$

Có thể thấy rằng M_r tính được bằng một thuật toán thời gian đa thức.

Bước 4: Xây dựng tập $V = R - \cap M_r$.

Rõ ràng $m.(m+1)/2 \geq |E_r| \geq |M| \geq |M_r|$. Bởi vậy thời gian tính của Thuật toán 25 là một đa thức theo số hàng và số cột của r.

Từ các Định lý 17, 24 và Thuật toán 25 chúng ta có hệ quả sau.

Hệ quả 26.

Tồn tại thuật toán đối với một quan hệ r cho trước, xác định một thuộc tính bất kì là cơ bản hay không với thời gian tính đa thức theo số hàng và cột của r.

Một vấn đề thường xuyên hay xảy ra là đối với một sơ đồ quan hệ cho trước s = <R,F>, và một phụ thuộc hàm $A \rightarrow B$, chúng ta muốn biết $A \rightarrow B$ có là phần tử của F^+ hay không. Để trả lời câu hỏi này

chúng ta cần tính bao đóng F^+ của tập các phụ thuộc hàm F. Tuy nhiên tính F^+ trong trường hợp tổng quát là rất khó khăn và tồn kém thời gian vì tập các phụ thuộc hàm thuộc F^+ là rất lớn cho dù F có thể là nhỏ. Chẳng hạn $F = \{A \rightarrow B_1, A \rightarrow B_2, .. A \rightarrow B_n\}$, F^+ khi đó còn bao gồm cả những phụ thuộc hàm $A \rightarrow Y$ với $Y \subseteq \{B_1 \cup B_2 \cup ... \cup B_n\}$. Như vậy sẽ có 2^n tập con Y . Trong khi đó, việc tính bao đóng của tập thuộc tính A lại không khó. Theo kết quả đã trình bày ở trên việc kiểm tra $A \rightarrow B \in F^+$ không khó hơn việc tính A^+.

Ta có thể tính bao đóng A^+ qua thuật toán sau:

Thuật toán 27

Vào: s = <R,F>, ở đây $R = \{ a_1, ..., a_n \}$ tập hữu hạn các thuộc tính, F tập các phụ thuộc hàm, $A \subseteq R$

Ra: A^+ bao đóng của A đối với F

Thuật toán thực hiện như sau: Tính các tập thuộc tính A_0, A_1... theo qui tắc:

1) $A_0 = A$

2) $A_i = A_{i-1} \cup \{a\}$ sao cho $\exists (C \rightarrow D) \in F, \{a\} \in Y$ và $C \subseteq A_{i-1}$

Vì $A = A_0 \subseteq ... \subseteq A_i \subseteq R$, và R hữu hạn nên tồn tại một chỉ số i nào đó mà $A_i = A_{i+1}$, khi đó thuật toán dừng và $A^+ = A_i$

Chúng ta có thể thấy độ phức tạp thời gian của thuật toán này là đa thức theo kích thước của s.

Để tiện kí pháp chúng ta thay $a \cup b$ bởi viết ab.

Ví dụ: s = <R,F>, ở đâyR = { a,b,c,d,e,g }, F bao gồm 8 phụ thuộc hàm

$ab \rightarrow c$	$d \rightarrow eg$
$c \rightarrow a$	$be \rightarrow c$
$bc \rightarrow d$	$cg \rightarrow bd$
$acd \rightarrow b$	$ce \rightarrow ag$

và A = bd.

Dùng Thuật toán 27 chúng ta có thể thấy

A_0 = bd,

A_1 = bdeg,

A_2 = bcdeg,

A_3 = abcdeg,

A^+ = abcdeg.

Để chứng minh tính đúng đắn của Thuật toán 27 chúng ta có thể dùng phương pháp quy nạp để chỉ ra rằng nếu a thuộc A_i thì a cũng thuộc A^+.

Việc chỉ ra điều ngược lại cũng bằng qui nạp nhưng khó khăn hơn là nếu a nằm trong A^+ thì a nằm trong một số A_i nào đó.

Định lý 28.

Thuật toán 27 tính chính xác A^+.

Như vậy, để xác định một phụ thuộc hàm $A \rightarrow B$ có thuộc F^+ hay không chúng ta chỉ cần kiểm tra $B \subseteq A^+$?.

Bây giờ, chúng ta đi tìm thuật toán tìm bao đóng cho một tập các thuộc tính trên một quan hệ bất kì

Đối với một quan hệ bất kì theo Định lí 13 chúng ta đã chúng minh với mọi $A \subseteq R$

$$L_{Fr}(A) = \begin{cases} \cap E_{ij} \text{ nếu tồn tại } E_{ij} \in E_r : A \subseteq E_{ij}, \\ A \subseteq E_{ij} \\ R \quad \text{ngược lại.} \end{cases}$$

Có thể thấy $L_{Fr}(A) = A_r^+$. Do vậy chúng ta có ngay thuật toán sau để tính bao đóng cho một tập bất kì trên quan hệ r.

Thuật toán 29

Vào: $r = \{h_1, ..., h_m\}$ là một quan hệ trên R

Ra: V là tập tất cả thuộc tính cơ bản của r

Bước 1: Từ r chúng ta xây dựng một tập $E_r = \{E_{ij} : m \geq j > i \geq 1\}$ và $E_{ij} = \{ a \in R : h_j(a) = h_j(a) \}$

Bước 2: Từ E_r chúng ta xây dựng một tập $M = \{B \in P(R) :$ Tồn tại $E_{ij} \in E_r : E_{ij} = B\}$

Bước 3:

$$A_r^+ = \begin{cases} \cap B & \text{nếu tồn tại } B \in M : A \subseteq B, \\ R & \text{ngược lại.} \end{cases}$$

Dễ thấy, độ phức tạp thời gian của thuật toán này là một đa thức theo kích thước của r.

Định nghĩa 30

Giả sử s = <R,F>, t = <R,G > là hai sơ đồ quan hệ trên R. Khi đó chúng ta nói s tương đương với t nếu $F^+ = G^+$. Nếu s và t tương đương thì đôi khi chúng ta có thể nói rằng s là một phủ của t hoặc t là một phủ của s.

Dễ dàng kiểm tra rằng s và t có tương đương với nhau không.

Mỗi phụ thuộc hàm $Y \rightarrow Z$ ở trong F chúng ta kiểm tra lại $Y \rightarrow Z$ ở trong G^+ bằng vi ệc sử dụng Thuật toán 27 để tính Y^+ và kiểm tra $Z \subseteq Y^+$. Nếu có phụ thuộc hàm $Y \rightarrow Z$ không nằm trong G^+ hoặc ngược lại nếu có phụ thuộc hàm $X \rightarrow W$ ở trong G

nhưng không ở trong F^+ thì điều chắc chắn là F^+ khác G^+. Nếu mỗi phụ thuộc hàm ở trong F thì cũng nằm trong G^+ và mỗi phụ thuộc hàm nằm trong G thì cúng nằm trong F^+, khi đó chúng ta có s và t là tương đương với nhau.

Hiện nay chúng ta cho định nghĩa sau nói về sự tương đương của hai quan hệ.

Định nghĩa 31

Giả sử r và v là hai quan hệ trên R. Khi đó ta nói r và v tương đương với nhau nếu $F_r = F_v$.

Chúng tôi trình bày định lí sau liên quan đến sự tương đương của hai quan hệ.

Định lí 32

Giả sử r và v là hai quan hệ trên R. Khi đó s tương đương với v khi và chỉ khi $N_r = N_v$.

Trên cơ sở Định lí 32 chúng ta đưa ra một thuật toán kiểm tra xem r có tương đương với v hay không.

Thuật toán 33

Vào: r và t là hai quan hệ trên R

Ra: r có tương đương với v hay không

Bước 1: Từ r tính N_r,

Bước 2: Từ v tính N_v

Bước 3 : So sánh M_r với M_v.

Bây giờ, chúng ta quay lại với sơ đồ quan hệ. Chúng ta muốn gọt giũa các phụ thuộc hàm của sơ đồ quan hệ để có tập phụ thuộc hàm tốt hơn.

Định nghĩa 34

Chúng ta nói một sơ đồ quan hệ s = <R,F> là chính tắc nếu

1. Vế phải của mỗi phụ thuộc hàm trong F là thuộc tính đơn.

2. Không có $X \rightarrow a$ nào ở trong F để F- $\{X \rightarrow a\}$ tương đương với F.

3. Không có $X \rightarrow a$ và một tập con Z của X để $F-\{X \rightarrow A\} \cup \{Z \rightarrow a\}$ tương đương với F.

Chúng ta sẽ chỉ ra rằng có thuật toán để tìm một phủ chính tắc cho một sơ đồ quan hệ bất kì.

Trước tiên chúng ta đưa ra mệnh đề sau

Mệnh đề 35:

Mỗi một sơ đồ quan hệ s = <R,F> đều có một phủ tương đương t = <R,G> sao cho vế phải của mỗi phụ thuộc hàm trong G không có hơn một thuộc tính.

Chứng minh: Đặt G là tập phụ thuộc hàm có dạng $X \rightarrow a$, với $X \rightarrow Y$ nằm trong F và a là một

phần tử của Y. Trên cơ sở hệ tiên đề của Armstrong, chúng ta dễ thấy t tương đương với s. □

Chúng ta trình bày thuật toán dưới đây để tìm phủ chính tắc cho một sơ đồ quan hệ cho trước

Thuật toán 36

Vào: s= <R,F>, F = { $A_1 \rightarrow B_1$,..., $A_m \rightarrow B_m$ }

Ra: t = <R,G> là chính tắc và tương đương với s

Do Mệnh đề 35 chúng ta có thể coi s thoả mãn điều kiện 1.

Bước 1: Đặt F_0 = F, với i = 1, ..., m

$F_i = F_{i-1}$ - { $A_i \rightarrow B_i$ } nếu F_{i-1} - { $A_i \rightarrow B_i$ } tương đương với F_i . Trong trương hợp ngược lại thì $F_i = F_{i-1}$.

Bước 2: Nhờ thuật toán tính bao đóng, từ F_m chúng ta lần lượt loại bỏ các thuộc tính thừa trong mỗi vế trái của từng phụ thuộc hàm thuộc F_m.

Kết quả nhận được chính là G.

Dễ thấy rằng thuật toán trên có độ phức tạp thời gian là đa thức theo kích thước của s.

Chúng ta đưa ra một khái niệm sau

Định nghĩa 37

Giả sử s = <R,F> là một sơ đồ quan hệ trên R. Khi đó chúng ta nói rằng s là sơ đồ quan hệ tối tiểu nếu với mọi sơ đồ quan hệ t = < R,G> tương đương với s có | F | < |G|, ở đây |F| là số lượng các phụ thuộc hàm trong F.

David Maier [25] đã chứng minh định lí sau

Định lí 38

Tồn tại một thuật toán có độ phức tạp thời gian đa thức để tìm phủ tối tiểu cho một sơ đồ quan hệ cho trước.

Ví dụ:

Chúng ta xem xét tập các phụ thuộc hàm F trong ví dụ minh hoạ Thuật toán 27. Ban đầu chúng ta tách vế phải ra thành các thuộc tính đơn:

$ab \rightarrow c$,

$c \rightarrow a$,

$bc \rightarrow d$,

$acd \rightarrow b$,

$d \rightarrow e$,

$d \rightarrow g$,

$be \rightarrow c$,

$cg \rightarrow b$,

cg→d,

ce→a,

ce →g.

Rõ ràng ce →a là không cần thiết bởi vì nó suy ra được từ c →a.

cg →b là dư thừa bởi vì có cg →d , c →a và acd →b. Ngoài ra không có một phụ thuộc hàm nào là dư thừa nữa. Dễ thấy acd →b có thể thay thế bởi cd →b. Vậy chúng ta có một phủ chính tắc là :

ab →c,

c →a,

bc→d,

cd→b,

d→e,

d→g,

be→c,

cg→d,

ce→g.

2.3 Các mô tả tương đương của họ các phụ thuộc hàm

Trong mục trên chúng ta đã định nghĩa hàm đóng. Nó là một mô tả tương đương của họ các phụ thuộc hàm. Trong mục này chúng tôi cung cấp cho bạn đọc một số các mô tả tương đương khác của họ này. Chúng chính là các công cụ để chúng ta có thể nghiên cứu phong phú hơn nữa cấu trúc lôgic của họ các phụ thuộc hàm.

Định nghĩa 1

Cho R là tập các thuộc tính và P(R) là tập các tập con của R.

Một hàm C: P(R) $\rightarrow$ P(R) được gọi là một hàm chọn trên R nếu với mọi A $\in$ P(R) thì C (A) $\subseteq$ A.

Giả sử L là một hàm đóng trên R . Chúng ta đặt C(A) = R - L(R-A) (*)

Dễ thấy C là một hàm chọn trên R

Trong [6] người ta đã chứng minh được kết quả sau

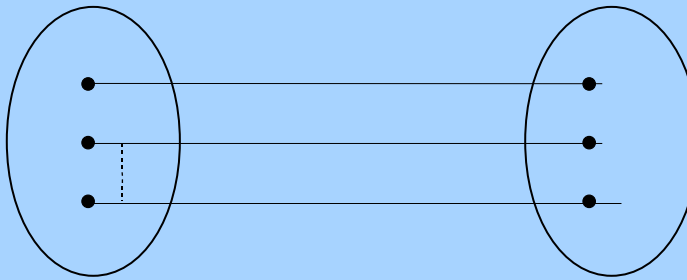
Định lí 2

Tương ứng được xác định như (*) là tương ứng 1-1 giữa tập các hàm đóng và tập các hàm chọn thoả mãn 2 điều kiện sau: với mọi A,B $\subseteq$ R

(1) C(A) $\subseteq$ B $\subseteq$ A kéo theo C(A) = C(B)

(2) A $\subseteq$ B kéo theo C(A) $\subseteq$ C(B).

Chúng ta có hình vẽ sau thể hiện mối quan hệ giữa lớp hàm đóng và lớp hàm chọn đặc biệt trên



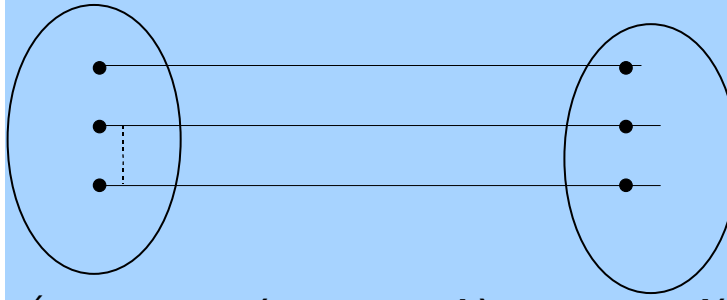
Lớp các hàm đóng Lớp các hàm chọn đặc biệt

Định lí dưới đây chỉ ra một tương ứng 1-1 giữa lớp các hàm đóng và lớp các nửa dàn giao trên R.

Định lí 4

Giả sử L là một hàm đóng trên R. Đặt Z(L) = { A : A $\in$ P(R) và L(A) = A }. Khi đó Z(L) là một nửa dàn giao trên R.

Ngược lại, nếu I là một nửa dàn giao trên R, thì tồn tại duy nhất một hàm đóng L sao cho Z(L) = I, ở đây L(A) = $\cap$ {A' $\in$ I : A $\subseteq$ A'}.



Lớp các hàm đóng
Lớp các nửa dàn giao

Như vậy, từ Định lí 6 mục trên và Định lí 4, chúng ta thấy có một tương ứng 1-1 giữa lớp các nửa dàn giao và lớp các họ các phụ thuộc hàm trên R .

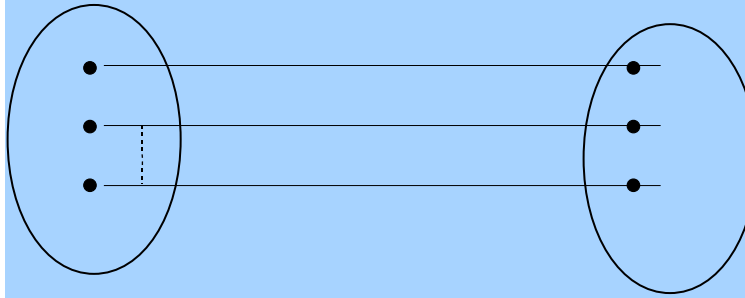
Định nghĩa 5

Giả sử $N \subseteq P(R)$. Khi đó N được gọi là tập không giao nếu với mọi $A \in N$ thì $A \neq \cap \{A' \in N : A \subset A'\}$.

Định lí 6

Nếu I là nửa dàn giao, thì N_I là tập không giao. Ngược lại nếu N là tập không giao thì tồn tại duy nhất một nửa dàn giao I sao cho $N_I = N$.

Như vậy, chúng ta thấy có một tương ứng 1-1 giữa lớp các nửa dàn giao và lớp các tập không giao.



Lớp các nửa dàn giao Lớp các tập không giao

Từ Định lí 6 mục trên và các Định lí 4 và 6 chúng ta rút ra kết luận là có một tương ứng 1-1 giữa lớp các họ phụ thuộc hàm với lớp các tập không giao.

Như vậy, để nghiên cứu phân tích các đặc trưng của họ các phụ thuộc hàm chúng ta có thể dùng công cụ nửa dàn giao hoặc tập không giao.

Bây giờ chúng tôi đưa ra khái niệm họ cực đại các thuộc tính. Đồng thời chúng ta chỉ ra rằng họ này là một mô tả tương đương của họ phụ thuộc hàm.

Định nghĩa 7

Giả sử R là tập các thuộc tính. Họ $M=\{(A,\{a\}): A \subset R, a \in R\}$ được gọi là họ cực đại các thuộc tính trên R nếu nó thoả mãn các điều kiện sau

(1) $a \notin A$,

(2) Đối với mọi $(B, \{b\}) \in M$, $a \notin B$ và $A \subseteq B$ kéo theo $A = B$.

(3) $\exists (B,\{b\}) \in M : a \notin B$, $a \neq b$, và $L_a \cup B$ là một hệ Sperner trên R, ở đây $L_a = \{A: (A, \{a\}) \in M\}$.

Nhận xét 8.

- Có thể có $(A, \{a\}), (B, \{b\}) \in M$ mà $a \neq b$, nhưng $A = B$.

- Do (1) và (2) có thể thấy đối với $a \in R$ chúng ta có L_a là một hệ Sperner trên R. Đặc biệt có thể L_a là một hệ Sperner rỗng.

- Trên cơ sở Định nghĩa 7 chúng ta có thể thấy tồn tại một thuật toán thời gian tính đa thức để xác định một tập $Y \subseteq P(R) \times P(R)$ có là một họ cực đại các thuộc tính trên R hay không.

Giả sử H là một hàm đóng trên R. Đặt $Z(H) = \{A : H(A) = A\}$ và $M(H) = \{(A, \{A\}) : A \notin A, A \in Z(H)$ và $B \in Z(H), A \subseteq B, A \notin B$ kéo theo $A = B\}$.

Z(H) là họ các tập đóng của H. Dễ thấy, với mỗi $(A, \{a\}) \in M(H)$, A là tập đóng cực đại mà không chứa a.

Có thể tồn tại $(A, \{a\}), (B, \{b\}) \in M(H)$ mà $a \neq b$, nhưng $A = B$.

Nhận xét 9.

Giả sử r là một quan hệ trên R và F_r là họ các phụ thuộc hàm của r. Đặt $A_r^+ = \{a: A \to \{a\} \in F_r\}$ và $Z_r = \{A: A = A_r^+\}$ và N_r là hệ sinh cực tiểu của nó. Có thể thấy $N_r \subseteq E_r$ với

$N_r = \{A \in E_r : A \neq \cap\{B: B \in E_r, A \subset B\}\}$, ở đây E_r là hệ bằng nhau r.

Chúng ta cho định lí dưới đây chỉ ra rằng giữa các hàm đóng và họ cực đạI các thuộc tính có tương ứng 1-1.

Định lí 10.

Giả sử H là một hàm đóng trên R. Khi đó M(H) là họ cực đại các thuộc tính. Ngược lại, nếu M là họ cực đạI các thuộc tính trên R thì tồn tại đúng một hàm đóng H trên R để M(H) = M, ở đây với mọi $B \in P(R)$.

$$H(B) = \begin{cases} \cap A \text{ if } \exists A \in L(M) : B \subseteq A, \\ B \subseteq A \\ R \quad \text{ngược lại,} \end{cases}$$

và $L(M) = \{A: (A, \{a\} \in M\}$.

Lời giải:

Giả sử H là hàm đóng trên R. Cơ sở trên định nghĩa của M(H) ta có (1) và (2). Ta đặt $L'_a = \{ A : (A,$

$\{a\} \in M(H)\}$. Giả thiết có $S(B, \{b\}) \in M(H) : a \neq b$, $a \notin B$, $L'_a \cup B$ là hệ Sperner trên R (*). Khi đó ta chọn $(B,\{b\}) \in M(H)$ sao cho B là lớn nhất cho (*). Do (2) trong Định nghĩa 7 ta có L_a' là hệ Sperner trên R. Do đó, không có một A nào mà $A \in L_a'$ và $B \subseteq A$. Phù hợp với định nghĩa của M(H) ta có $(B,\{a\}) \in M(H)$. Như vậy, $B \in L_a'$. Điều này là vô lí. Từ đó ta có (3) trong Định nghĩa 7. Có nghĩa là M(H) là họ cực đại các thuộc tính trên R.

Ngược lại, giả sử M là họ cực đại các thuộc tính trên R. Đặt $L(M) = \{A : (A, \{a\}) \in M\}$. Đầu tiên ta chứng tỏ rằng L(M) là tập không giao trên R. Đối với bất kì $(A, \{a\}) \in M$ do Nhận xét 2.2 ta có $A \neq A' \cap A''$ và $A \neq A' \cap B$, ở đây $A', A'' \in L_a$ và $B \in L(M) : A \neq B$.

Nếu tồn tại $(B, \{b\}), (C, \{c\}) \in M$ sao cho $b \neq a$, $c \neq a$, $A \subset B$, $A \subset C$ thì bởi (2) trong Định nghĩa 2.1 ta có $a \in B$, $a \in C$. Từ đó, $A \subset B \cap C$. Như vậy, đối với A,B,C, $\in L(M)$ nếu $A = B \cap C$ thì $A = B$ hoặc $A = C$. Do đó, L(M) là hệ không giao trên R. Chúng ta biết rằng các họ không giao và các hàm đóng xác định duy nhất lẫn nhau. Mặt khác phù hợp với Nhận xét 8 và Định lí 13 mục trên ta có H là một hàm đóng trên R và L(M) là hệ sinh tối tiểu của Z(H).

Hiện ta sẽ chứng minh M(H) = M. Nếu (A, {a}) $\in$ M thì A $\in$ L(M). Giả thiết rằng đối với mỗi b $\notin$ A có B $\in$ Z(H) : A $\subset$ B, b $\notin$ B. Có thể thấy rằng A là giao của những B như thế. Điều này mâu thuẫn với A $\in$ L(M). Nếu (A, {a}) $\in$ M thì có b $\notin$ A để (A, {b}) $\in$ M(H) (**).

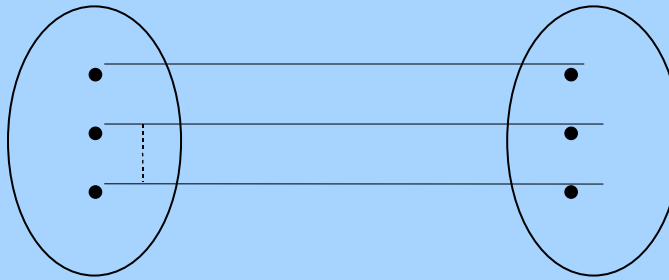
Nếu (A, {a}) $\in$ M(H) thì phù hợp với định nghĩa của M(H) B $\in$Z(H), và A $\subset$ B kéo theo a $\in$ B. Vì a $\notin$ A, ta thấy A không là giao của các B như thế. Theo cấu trúc của H ta có A $\in$ L(M). Như vậy, nếu (A, {a}) $\in$ M(H) thì A $\in$ L(M) (***).

Bây giờ ta giả thiết là (A,{a}) $\in$ M, nhưng (A, {a}) $\notin$ M(H). Vì A là tập đóng của H,a $\notin$ A và bởi định nghĩa của M(H) thì tồn tại (B,{a}) $\in$ M(H) sao chot A $\subset$ B. Do (***) ta có B $\in$ L(M). Điều này mâu thuẫn với điều kiện (2) của Định nghĩa 7. Từ đó, ta có (A,{a}) $\in$ M(H).

Giả thiết (A,{a}) $\in$ M(H), nhưng (A,{a}) $\notin$ M. Nếu có A' $\in$ L_a để A $\subset$ A' thì bởi (**) ta có A' là một tạp đóng của H. Phù hợp với định nghĩa của M(H) ta có (A,{a}) $\notin$ M. Điều này là vô lí. Nếu A' $\subset$ A thì do (***) ta có (A',{a}) $\notin$ M, nó cũng vô lí. Nếu A $\cup$ L_a là hệ Sperner trên R thì bởi (***) ta có thể thấy nó trái với điều kiện (3) của Định nghĩa 7. Do đó tồn tại

một A' mà A' ∈ L_a và A =A'. Từ đó ta có M(H) = M.

Nếu ta giả thiết rằng có H' mà M(H') = M. Đặt L(H') = {A : (A,{a}) ∈ M(H'). Theo (**) và (***) của chứng minh trên ta có L(H') là một hệ sinh nhỏ nhất của Z(H'). Do M(H') = M(H) = M ta có L(H') = L(M) = L(H). Vì các hàm đóng và các tập không giao là xã định duy nhất lẫn nhau nên H = H'. □



Lớp các hàm đóng cực đại các thuộc tính Lớp các họ

Vì các hàm đóng và các họ các phụ thuộc hàm tương ứng xác định duy nhất lẫn mhau, từ Định lí 2.4 ta có

Hệ quả 11.

Tồn tại tương ứng 1-1 giữa lớp các họ cực đại các thuộc tính và họ các phụ thuộc hàm.

2.4. Các thuật toán liên quan đến các khoá

Khi giải quyết các bài toán thông tin quản lí, chúng ta thường sử dụng các hệ quản trị cơ sở dữ liệu mà trong đó chứa cơ sở dữ liệu quan hệ. Các phép xử lí đối với lớp bài toán này thường là tìm kiếm bản ghi sau đó thay đổi nội dung bản ghi, thêm bản ghi mới hoặc xoá bản ghi cũ. Trong các thao tác trên việc tìm kiếm bản ghi là rất quan trọng. muốn tìm được bản ghi trong một file dữ liệu chúng ta phải xây dựng khoá cho file dữ liệu đó.

Việc xây dựng khoá ở đây chính là xây dựng khoá tối tiểu. Vì thế trong mục này chúng tôi cung cấp cho bạn đọc hai thuật toán tìm khoá tối tiểu.

2.4.1 Thuật toán tìm khoá tối tiểu của một sơ đồ quan hệ

Vào : sơ đồ quan hệ s = < F, R> trong đó :

F là tập các phụ thuộc hàm và

R={ a_1 ,...., a_n } là tập các thuộc tính

Ra : K là một khoá tối tiểu

Thuật toán thực hiện như sau:

Tính liên tiếp các tập thuộc tính K_0, K_1....., K_n như sau:

K_o = R = { a_1 ,...., a_n }

K_I = K_{i-1} nếu K_{i-1} - {a_i} $\rightarrow R \notin F^+$,

K_{i-1} -{a_i} ngược lại

...

K = K_n là khoá tối tiểu

2.4.2. Thuật toán tìm một khoá tối tiểu của một quan hệ

Cho trước r = {h_1,.... h_m} là một quan hệ trên tập thuộc tính R = { a_1 ,...., a_n }

Hệ bằng nhau của quan hệ r được định nghĩa ở phần trên như sau:

E_r ={E_{ij} : 1≤ i ≤ j ≤ m}, ở đây E_{ij} = {a ∈ R : h_i (a) = h_j (a)}.

Dễ dàng thấy rằng E_r được tính bằng thời gian đa thức từ r

Thuật toán tìm một khoá tối tiểu của một quan hệ r:

Vào: r = {h_1,.... h_m} là một quan hệ trên tập thuộc tính R= { a_1 ,...., a_n }

Ra: K là một khoá tối tiểu của r

Thuật toán thực hiện như sau

Bước 1: Tính E_r ={ A_1 ....A_t}

Bước 2 : Tính M_r = { A : có A_j ∈ E_r : A = A_j và không có A_i : A_i∈E_r: A ⊂ A_i }

Bước 3 : Lần lượt tính các tập thuộc K_1, K_2, ...K_n tính theo qui tắc :

$K_0 = R = \{ a_1, ..., a_n \}$ hoặc K_0 = một khoá đã biết

$K_i = K_{i-1} - \{ a_i \}$ nếu không tồn tại $A \in M_r$ sao cho $K_{i-1} - \{a_i\} \subseteq A$ hoặc $K_i = K_{i-1}$ trong trường hợp ngược lại

Bước 4: Đặt $K = K_n$, khi đó K là khoá tối tiểu

2.5. Mối quan hệ giữa quan hệ Armstrong và sơ đồ quan hệ

Việc xây dựng quan hệ Armstrong của một sơ đồ quan hệ cho trước và ngược lại từ quan hệ cho trước ta xây dựng một SĐQH sao cho quan hệ cho trước này là quan hệ Armstrong của nó có vai trò rất quan trọng trong việc phân tích cấu trúc lôgic của mô hình dữ liệu quan hệ cả trong thiết kế lẫn trong ứng dụng. Đã có nhiều tác giả nghiên cứu vấn đề này. Trong mục này chúng tôi trình bày hai thuật toán giải quyết bài toán trên và đưa ra việc đánh giá các thuật toán này cũng như đánh giá độ phức tạp của bài toán trên.

Trước tiên, chúng ta cho một thuật toán tìm tập tất cả các phản khoá của hệ Spernner cho trước.

Thuật toán 1 (Tìm tập phản khoá)

Vào: $K = \{B_1,...,B_n\}$ là hệ Sperner trên R.

Ra: K^{-1}.

Bước 1: Ta đặt $K_1 = \{R - \{a\}: a \in B_1\}$. Hiển nhiên $K_1 = \{B_1\}^{-1}$.

Bước q + 1: (q < m). Ta giả thiết rằng $K_q = F_q \cup \{X_1... X_{tq}\}$, ở đây $X_1,....,X_{tq}$ chứa B_{q+1} và $F_q = \{ A \in K_q : B_{q+1} \subseteq A \}$. Đối với mỗi I (I = 1,..., t_q) ta tìm các phản khoá của { B_{q+1} } trên X_i tương tự như K_1. Kí pháp chúng là $A^i_1,..., A^i_{ri}$. Đặt

$K_{q+1} = F_q \cup \{A^i_p : A \in F_q$ kéo theo $A^i_p \not\subset A, 1 \leq i \leq t_q. 1 \leq p \leq r_i\}$

Cuối cùng ta đặt $K^{-1} = K_m$

Định lí 2.

Với mọi q ($1 \leq q \leq m$), $K_q = \{ B_1,.... B_q\}^{-1}$, có nghĩa là $K_m = K^{-1}$.,

Rõ ràng, K và K^{-1} là xác định duy nhất lẫn nhau và từ định nghĩa của K^{-1} có thể thấy thuật toán của chúng ta không phụ thuộc vào thứ tự của dãy $B_1,..., B_m$. Đặt $K_q = F_q \cup \{X_1,..., X_{tq}\}$ và l_q ($1 \leq q \leq m-1$) là số các phần tử của K_q. Khi đó ta có

Mệnh đề 3

Độ phức tạp thời gian tồi nhất của Thuật toán **2.1** là

m-1

$$O\left(|R|^2 \sum_{q=1} t_q.u_q\right).$$

ở đây $u_q = \begin{cases} l_q - t_q, & \text{nếu } l_q > t_q, \\ 1 & \text{nếu } l_q = t_q \end{cases}$

Rõ ràng trong mỗi bước thuật toán ta có K_q là hệ Sperner trên R. Ta biết rằng[5] kích thước của hệ Sperner bất kì trên R không vượt quá $C_n^{[n/2]}$, ở đây n = |R|. Có thể thấy $C_n^{[n/2]}$ xấp xỉ bằng $2^{n+1/2}/(\Pi.\ n^{1/2})$. Từ đó độ phức tạp thời gian tồi nhất của thuật toán trên không nhiều hơn hàm số mũ theo n. Trong trường hợp mà $l_q \leq l_m$ (q = 1,..., m-1), dễ thấy rằng độ phức tạp thuật toán không lớn hơn $O(|R|^2 |K| |K^{-1}|^2)$. Như vậy, trong các trường hợp này độ phức tạp của Thuật toán 1 tìm K^{-1} là đa thức theo |R|, |K|, and $|K^{-1}|$. Có thể thấy nếu số lượng các phần tử của K là nhỏ thì Thuật toán 1 là rất hiệu quả. Nó chỉ đòi hỏi thời gian đa thức theo |R|

Định nghĩa 4

Cho s = (R,F) là SĐQH trên R và $a \in R$. Đặt

$K_a = \{ A \subseteq R: A \rightarrow \{a\}, \nexists B: (B \rightarrow \{a\})(B \subset A)\}$. K_a được gọi là họ các tập tối tiểu của thuộc tính a.

Rõ ràng, $R \notin K_a$, $\{a\} \in K_a$ và K_a là hệ Sperner trên R.

Thuật toán 5. (Tìm tập tối tiểu của thuộc tính a)

Vào: Cho s = (R = $\{a_1,..., a_n\}$, F) là SĐQH, $a = a_1$.

Ra: $A \in K_a$.

Bước 1: Ta đặt L(0) = R.

Bước i + 1: Đặt

$$L(i+1) = \begin{cases} L(i) - a_{i+1} & \text{nếu } L(i) - a_{i+1} \rightarrow \{a\}, \\ L(i), & \text{ngược lại.} \end{cases}$$

Khi đó A = L(n).

Bổ đề 6.

$L(n) \in K_a$

Lời giải: Bằng phương pháp chứng minh qui nạp có thể thấy $L(n) \rightarrow \{a\}$, và $L(n) \subseteq ... \subseteq L(0)$ (1). Nếu L(n) = a, thì bởi định nghĩa của tập tối tiểu của thuộc tính a ta thu được $L(n) \in K_a$. Bây giờ ta giả thiết là tồn tại một B sao cho $B \subset L(n)$ và $B \neq 0$. Như vậy sẽ có a_j sao cho $a_j \not\subset B$, $a_j \in L(n)$. Theo các xây dựng thuật toán này ta có $L(j-1) - a_j \rightarrow \{a\}$. Rõ ràng bởi (1) ta thu được $L(n) - a_j \subseteq L(j-1) - a_j$ (2). Dễ thấy $B \subseteq L(n) - a_j$.

Từ (1), (2) ta có $B \rightarrow \{a\}$. □

Dễ thấy, vì thuật toán xác định một phụ thuộc hàm bất kì có phải là phụ thuộc hàm của một SĐQH hay không là có độ phức tạp thời gian đa thức, nên độ phức tạp của Thuật toán 5 là $O(|R|^2 |F|)$.

Bổ đề 7.

Cho s = (R,F) là SĐQH trên R và $a \in R$, K_a là họ các tập tối tiểu của a, $L \subseteq K_a$, $\{a\} \in L$. Khi đó $L \subset K_a$ nếu và chỉ nếu tồn tại C, $A \rightarrow B$ sao cho $C \in L$ và $A \rightarrow B \in F$ và $\forall E \in L \Rightarrow E \subseteq A \cup (C - B)$.

Lời giải. =>: Ta giả thiết rằng $L \subset K_a$. Do đó, tồn tại $D \in K_a - L$. Bởi $\{a\} \in L$ và K_a là hệ Sperner trên R, chúng ta có thể xây dựng một tập cực đại Q sao cho $D \subseteq Q \subset U$ và $L \cup Q$ là hệ Spernner. Từ định nghĩa của K_a, chúng ta thu được $Q \rightarrow \{a\}$ (1) và $a \notin Q$ (2). Nếu $A \rightarrow B \in F$ kéo theo (A tb Q, $B \subseteq Q$) hoặc $A \subseteq Q$ thì $Q^+ = Q$. Bởi (2) ta có $Q \rightarrow \{a\}$. Điều này mâu thuẫn với (1). Do đó, tồn tại một phụ thuộc hàm $A \rightarrow B$ sao cho $A \subseteq Q$, $B \subseteq Q$. Từ cách xây dựng của Q có C sao cho $C \in L$, $A \subseteq Q$, $C - B \subseteq Q$. Hiển nhiên rằng $A \cup (C-B) \subseteq Q$.

Rõ ràng, $E \subseteq A \cup (C-B)$ đối với mọi $E \in L$.

<=: Ta giả thiết rằng có C, và $A \rightarrow B$ sao cho $C \in L$, $A \rightarrow B \in F$ và $E \subseteq A \cup (C-B)$ đối với mọi $E \in L$ (3). Bởi định nghĩa của L chúng ta thu được $A \cup$

$U(C-B) \rightarrow \{a\}$. Bởi $\{a\}$ thuộc L nên có D sao cho $D \in K_a$, $a \notin D$, D thuộc $A \cup (C-B)$. Bởi (3) ta có $D \in K_a - L$. □

Cơ sở trên bổ đề này và thuật toán 5, chúng ta xây dựng một thuật toán sau đây bằng qui nạp.

Thuật toán 8. Tìm họ các tập cực tiểu của thuộc tính a.

Vào: Cho s = (R, F) là một sơ đồ quan hệ và a thuộc R.

Ra: K_a

Bước 1: Đặt $L(1) = E_1 = \{a\}$

Bước i + 1: Nếu có C và $A \rightarrow B$ mà $C \in L(i)$, $A \rightarrow B \in F$, $\forall E \in L(i) \rightarrow E \notin A \cup (C - B)$, thì bởi thuật toán 5 chúng ta xây dựng E_{i+1}, ở đây $E_{i+1} \subseteq A \cup (C - B)$, $E_{i+1} \in K_a$. Chúng ta đặt $K(i+1) = K(i) \cup E_{i+1}$. Trong trường hợp ngược lại ta đặt $K_a = L(i)$.

Bởi bổ đề 7 hiển nhiên rằng tồn tại một số tự nhiên t để $K_a = L(t)$

Có thể thấy rằng độ phức tạp thời gian tồi nhất của thuật toán là $O(|U| \; |F| \; |K_a| \; (|U| + ||K_a|))$. Như vậy, độ phức tạp thời gian của thuật toán này là đa thức theo $|U|$, $|F|$, và $|K_a|$.

Rõ ràng, nếu số lượng các phần tử của K_a đối với sơ đồ quan hệ s = <R, F> là đa thức theo kích thước của s, thì thuật toán này là rất hiệu quả. Đặc biệt khi $|K_a|$ là nhỏ.

Hiển nhiên rằng nếu đối với mỗi $A \rightarrow B \in F$ kéo theo $a \in A$ hoặc $a \notin B$, thì $K_a = \{a\}$

Nhận xét 9

Biết rằng [27] nếu s = <R, F> là một sơ đồ quan hệ, Z(F) = $\{A : A^+ = A\}$ và N(F) là hệ sinh nhỏ nhất của Z(F), thì

$$N(F) = MAX(F^+) = \bigcup_{a \in R} MAX(F^+, a)$$

ở đây $MAX(F^+,a) = \{A \subseteq U : A \rightarrow \{a\} \notin F^+, A \subset B \rightarrow B \rightarrow \{a\} \in F^+$. Rõ ràng rằng, K_a là một hệ Sperner trên R. Có thể thấy MAX (F^+, a) là tập các phản khoá của K_a đối với mọi $a \in R$. Như vậy, $MAX(F^+, a) = K^-_{1\,a}$.

Định lý 10

Cho $r = \{h_1,...h_m\}$ là một quan hệ, và F là một họ f trên R. Khi đó $F_R = F$ nếu và chỉ nếu với mọi $A \in P(R)$

$$\bigcap E_{ij} \quad \text{nếu } \exists E_{ij} \in E_r ; A \subseteq E_{ij}$$

$L_F(A) = \quad A \subseteq E_{i\,j}$

R ngược lại,

ở đây $L_F(A) = \{a \in R : (A, \{a\}) \in F\}$ và E_r là hệ bằng nhau của r.

Trên cơ sở nhận xét 9, định lý 10, các thuật toán 1, 8, chúng ta xây dựng một thuật toán tìm quan hệ Armstrong từ một sơ đồ quan hệ cho trước như sau:

Thuật toán 11 (Tìm quan hệ Armstrong)

Vào: Cho s = <R, F> là một sơ đồ quan hệ

Ra: r là quan hệ sao cho $F_r = F^+$

Bước 1: Đối với mỗi $a \in R$ bởi thuật toán 2.8 chúng ta tính K_a, và từ thuật toán 2.1 xây dựng tập các phản khoá K_a^{-1} .

Bước 2: $N = \bigcup_{a \in R} K_a^{-1}$

Bước 3: Giả sử các phần tử của N là $A_1,...,A_t$, chúng ta xây dựng quan hệ $r = \{ h_0, h_1, ..., h_0 \}$ như sau: Với mỗi $a \in R$, $h_0(a) = 0$, $\forall i = 1,...,t$

$h_i(a) = 0$ nếu $a \in A_i$, hoặc $h_i(a) = 1$ trong trường hợp ngược lại.

Do nhận xét 9 rõ ràng rằng, nếu chúng ta có N(F), thì chúng ta có thể trực tiếp xây dựng r. Độ phức tạp của việc xây dựng này phụ thuộc vào |

N(F) |. Dễ thấy độ phức tạp của thuật toán 11 là độ phức tạp của bước 1. Bởi bổ đề 3 và đánh giá của thuật toán 8, dễ thấy rằng độ phức tạp tồi nhất của thuật toán 11 là

$$O\left(n \sum_{i=1}^{n} \left(\sum_{q-1}^{m_i - 1} t_{iq}\, u_{iq} + |F|\, m_i (m_i + n)\right)\right).$$

ở đây $R = \{a_1,...,a_n\}$, $m_i = |K_a,|$ and

$u_{iq} = l_{iq}$ nếu $l_{iq} > t_{iq}$ hoặc $u_{iq} = 1$ nếu $l_{iq} = t_{iq}$

Trong trường hợp $l_{iq} \leq (\forall i, \forall q : 1 \leq q \leq m_i)$, độ phức tạp thuật toán của chúng ta là

$$O\left(n \sum_{i=1}^{n} |K_{ai}| \left(n|F| + |K_{ai}|F| + n|K_a^{-1}|^2\right)\right).$$

Như vậy, độ phức tạp thuật toán 2.11 là đa thức theo $|R|$, $|F|$, $|K_{ai}|$, $|K_a^{-1}|$. Rõ ràng, trong các trường hợp này nếu $|K_{ai}|$ và $|K_a^{-1}|$ là đa thức (đặc biệt nếu chúng là nhỏ) theo $|R|$ và $|F|$, thì thuật toán của chúng ta là hiệu quả.

Bây giờ chúng ta sử dụng thuật toán 11 để xây dựng quan hệ Armstrong cho sơ đồ quan hệ trong ví dụ dưới đây.

Ví dụ 13

Cho s = <R, F> là một sơ đồ quan hệ, ở đây R = {a,b,c,d} và F = {{a,d} $\rightarrow$R, {a} $\rightarrow${a,b,c},{b,d} $\rightarrow$ {b,c,d}}.

Bởi thuật toán 8, chúng ta thu được K_a = {a}.K_b = {{a},{b}},K_c ={{a},{b,d},{c}}, K_d = {d}.

Trên cơ sở thuật toán 1, ta có K_a^{-1} = {b,c,d }, K_b^{-1} = {c,d}, K_c^{-1} = {{b},{d}}, K^{-1}_d = {a,b,c}.

Do đó, N(F) = {{a,b,c }, {b,c,d}, {c,d}, {b}, {d}}. Khi đó ta xây dựng quan hệ r như sau:

a	b	c	d
0	0	0	0
0	0	0	1
2	0	0	0
3	3	0	0
4	0	4	4
5	5	5	0

Bây giờ chúng ta xây dựng một thuật toán tìm một sơ đồ quan hệ s từ một quan hệ cho trước sao cho quan hệ này là quan hệ Armstrong của s.

Thuật toán 14 (Tìm một khoá tối thiểu từ tập các phản khoá).

Vào: Cho K là một hệ Sperner, H là một hệ Sperner, và $C = \{b_1,...,b_m\} \subseteq R$ sao cho $H^{-1} = K$ và $\exists B \in K : B \subseteq C$.

Ra : $D \in H$

Bước 1: Đặt $T(0) = C$

Bước i+1: Đặt $T = T(i) - b_{i+1}$

$$T(i+1) = \begin{cases} T & \text{nếu } \forall B \in K : T \notin B \\ T(i) & \text{ngược lại} \end{cases}$$

Cuối cùng đặt $D = T(m)$

Bổ đề 15. Nếu K là tập các phản khoá thì $T(m) \in H$.

Bổ đề 16. Cho H là một hệ Sperner trên R, và $H^{-1} = \{B_1,...,B_m\}$ là tập các phản khoá của H, $T \subseteq H$ .Khi đó $T \subset H$, $T \neq \emptyset$ nếu và chỉ nếu tồn tại $B \subseteq U$ sao cho $B \in T^{-1}$, $B \notin B_i$ ($\forall i : 1 \leq i \leq m$).

Cơ sở trên bổ đề 16 và thuật toán 14 chúng ta xây dựng thuật toán sau.

Thuật toán 17. Tìm tập các khoá tối thiểu từ tập các phản khoá.

Vào: Cho $K = \{B_1,...,B_k\}$ là một hệ Sperner trên R

Ra: H mà $H^{-1} = K$

Bước 1: Nhờ thuật toán 2.14 chúng ta tính A_1, đặt $K(1) = A_1$

Bước i+1: Nếu có $B \in K_i^{-1}$ sao cho $B \notin B_j$ ($\forall$ j: $1 \leq j \leq k$), thì bởi Thuật toán 2.14 chúng ta tính A_{i+1}, ở đây $A_{i+1} \in H$, $A_{i+1} \subseteq B$. Đặt $K(i+1) = K(i) \cup A_{i+1}$. Trong trường hợp ngược lại ta đặt H=K(i).

Mệnh đề 18. Độ phức tạp của Thuật toán 17 là

$$O(n (\sum_{q=1}^{m-1} (k\, l_q + n\, t_q\, u_q) + k^2 + n))$$

ở đây $|R| = n$, $|K| = k$, $|H| = m$, ý nghĩa của l_q, t_q, u_q, xem trong mệnh đề 3.

Rõ ràng , trong các trường hợp mà $l_q \leq k$ ($\forall$ q : $1 \leq q \leq m-1$) độ phức tạp thời gian của thuật toán là $O(|R|^2 |K|^2 |H|)$. Dễ thấy trong các trường hợp này thuật toán 2.17 tìm tập các khoá tối tiểu có độ phức tạp thời gian là đa thức trong kích thước của R, K, H.

Nếu $|H|$ là đa thức theo $|R|$ và $|K|$, thì thuật toán là hiệu quả. Có thể thấy rằng nếu số lượng các phần tử của H là nhỏ thì thuật toán 17 là rất hiệu quả.

Bổ đề 19.

Cho F là một họ f trên R, $a \in R$. Đặt $L_F(A) = \{a \in R: (A , \{a\}) \in F\}$, $Z_F = \{A : L_F (A) = A\}$. Rõ ràng,

$R \in Z_F$, $A,B \in Z_F \rightarrow A \cap B \in Z_F$. Kí pháp N_F là hệ sinh tối tiểu Z_F. Đặt $M_a = \{ A \in N_F : a \notin A, \exists B \in N_F: a \notin B, A \subset B\}$. Khi đó $M_a = MAX(F,a)$, ở đây $MAX(F,a) = \{A \subseteq U : A$ là một tập cực đại không rỗng mà $(A,\{a\}) \notin F\}$.

Lời giải:

Biết rằng [27] $MAX(F,a) \subseteq N_F$ (1). Giả thiết rằng $A \in M_a$. Bởi $A \in N_F$, có nghĩa là $L_F(A) = A$, và $a \notin A$, ta thu được $(A, \{a\}) \notin F$. Từ (1) và phù hợp với định nghĩa của M_a ta có $A \in MAX(F,a)$.

Ngược lại, Nếu $A \in MAX(F,a)$ thì do (1) ta có $A \in N_F$ (2). Do $(A,\{a\}]) \notin F$ và từ (2) ta thu được $a \notin A$. Phù hợp với định nghĩa của $MAX(F,a)$ ta có $A \in M_a$.□

Trên cơ sở Thuật toán 17 và Bổ đề 19, ta xây dựng thuật toán dưới đây để tìm SĐQH $s = \langle R,F \rangle$ cho một quan hệ r cho trước sao cho $F^+ = F_r$.

Thuật toán 20. (Tìm SĐQH)

Vào: r là quan hệ trên R

Ra: $s = \langle R,F \rangle$ mà $F^+ = F_R$

Bước1: Từ r ta tính hệ bằng nhau E_r

Bước 2: Đặt $N_r = \{ A \in E_r : A \neq \cap \{ B \in E_r : A \subset B\}\}$

Bước 3: Với mỗi $a \in R$ ta xây $N_a = \{A \in N_r : a \notin A \ \exists \ B \in N_r : a \notin B , A \in B \}$. Sau đó, bởi Thuật toán 17 ta xây họ H_a($H_a^{-1} = N_a$)

Bước 4: Xây s = <R,F>, ở đây $F = \{A \rightarrow \{a\} : \forall a \in R , A \in H_a, A \neq \{a\}\}$

Mệnh đề 21.

$F_R = F^+$

Lời giải: Vì F_R là một họ f trên R, có thể thấy $N_{F\,r} \subset E_r$, ở đây N_{Fr} là hệ sinh nhỏ nhất của Z_{Fr}. Do định nghĩa của hệ sinh nhỏ nhất ta có $N_r = N_{Fr}$. Do đó ta có $N_a = M_a$. Từ định nghĩa của tập phản khoá và định nghĩa của tập K_a ta có $H_a = K_a$. Tù đó ta thu được $F^+ \subseteq F_R$.

Ngược lại, nếu $A \rightarrow B = \{b_1,...,b_t\} \in F_R$ thì bởi việc xây dựng của F ta thu được $A \rightarrow \{b_i\} \in F^+$ với mỗi i=l,...,t. Vì không có phụ thuộc hàm tầm thường $\{a\} \rightarrow \{a\}$ trong F, dễ thấy với mọi i=1,...,t, nếu không có phụ thuộc hàm $B \rightarrow \{b_i\} \in F$, ở đây $B \subseteq U - b_i$, thì $b_i \in A$. Từ đó ta có $A \rightarrow B \in F^+$. □

Có thể thấy E_r, N_r, N_a với $a \in R$ được xây trong thời gian đa thức theo kích thước của r. Rõ ràng, việc xây dựng F phụ thuộc vào kích thước của H_a($\forall a \in R$). Do đó, độ phức tạp thời gian tồi nhất của Thuật toán 20 là

$$O\left(n \sum_{i=1}^{n} \left(\sum_{q=1}^{m_i-1} (k_i l_{iq} + n t_{iq} u_{iq}) + k_i^2 + n\right)\right)$$

ở đây R = { a_1,..., a_n}, $|N_{ai}| = k_i$, $|H_{ai}| = m_i$, ý nghĩa của các l_{iq}, t_{iq}, u_{iq} xem các Mệnh đề 3 và 18.

Dễ thấy, nếu $l_{iq} \le k_i$ ($\forall$ i, $\forall$ q : $1 \le q \le m_i - 1$), thì độ phức tạp thời gian của thuật toán của chúng ta là

$$\mathbf{O}\left(n^2 \sum_{i=1}^{n} k_i^2 m_i\right)$$

Bởi vì k_i là đa thức theo kích thước của r, trong các trường hợp nếu is m_i là đa thức theo kích thước của r, thì thuật toán của chúng ta là hiệu quả. Lúc đó độ phức tạp của nó là đa thức theo kích thước của r. Nếu $|H_a|$ là nhỏ thì thuật toán của ta rất hiệu quả.

Bây giờ, nhờ Thuật toán 20 chúng ta xây dựng SĐQH s = <R,F> cho quan hệ sau đây.

Ví dụ 22 r là quan hệ sau đây trên R = {a, b, c, d}:

a	b	c	d
6	6	6	0
0	2	0	2
0	0	0	0
0	0	0	3
4	4	0	0
5	0	5	5
1	0	0	0

Dễ thấy là

E_R = {{a,b,c}, {b,c,d}. { a,c}, { b,c}, {c,d}.{b}, {c},{d}, ∅},

N_R = {[a,b,c}, {b,c,d},{a,c},{c,d},{b},{d}},

N_a = {b,c,d}, N_b= {{a,c},{c,d}},

N_c = {{b},{d}}, N_d ={a,b,c}

Ta có H_a = {a}, H_b = {{b}, {a,d}}, H_c = {{a}, {b,d},{c}}, H_d = {d}.

Ta xây dựng s = (R,F) như sau:

R = {a,b,c,d}, F = {{a,d} → {b},{a} → {c},{b,d} → {c}}.

Chúng ta trình bày hai kết quả cơ bản về độ phức tạp thuật toán cho việc xây dựng quan hệ Armstrong cho một SĐQH cho trước và ngược lại.

Định lí 23

Độ phức tạp thời gian cho việc tìm kiếm một quan hệ Armstrong của một SĐQH cho trước là hàm số mũ theo số lượng của các thuộc tính.

Định lí 24

Độ phức tạp thời gian cho việc tìm kiếm một SĐQH s = <R,F> từ một quan hệ r cho trước sao cho $F_r = F^+$ là hàm số mũ theo số lượng các thuộc tính.

Chương 3

Các dạng chuẩn và các thuật toán liên quan

Việc chuẩn hoá các quan hệ cũng như các sơ đồ quan hệ đóng một vai trò cực kì quan trọng trong việc thiết kế các hệ quản trị cơ sở dữ liệu trên mô hình dữ liệu của Codd. Nhờ có chuẩn hoá các quan hệ và các sơ đồ quan hệ chúng ta tránh được việc dư thừa dữ liệu và tăng tốc độ của các phép toán xử lí quan hệ.

3.1 Các khái niệm cơ bản

Chúng ta định nghĩa các dạng chuẩn như sau.

Cho $r = \{h_1,...,h_m\}$ là quan hệ trên $R = \{a_1 ...., a_n\}$

Định nghĩa 1. (Dạng chuẩn 1 - 1NF):

r là dạng chuẩn 1 nếu các phần tử của nó là sơ cấp.

Khái niệm sơ cấp hiểu ở đây là giá trị $h_i(a_j)$ (i=1,...,m; j=1,...,n) không phân chia được nữa.

Định nghĩa 2 (Dạng chuẩn 2 - 2NF)

r là dạng chuẩn 2 nếu:

- r là dạng chuẩn 1

- $A \rightarrow \{a\} \notin F_r$ đối với mọi khoá tối thiểu K, $A \subset K$ và a là thuộc tính thứ cấp.

Định nghĩa 3. (Dạng chuẩn 3 - 3NF):

r là dạng chuẩn 3 nếu:

$A \rightarrow \{a\} \notin F_r$ đối với A mà $A^+ \neq R$, $a \notin A$, $a \notin \cup K$

Có nghĩa rằng :

- K là một khoá tối tiểu

- a là thuộc tính thứ cấp

- A không là khoá

- $A \rightarrow \{a\}$ không đúng trong r

Định nghĩa 4. (Dạng chuẩn Boye-Codd - BCNF)

r là dạng chuẩn của Boye-Codd nếu:

$A \rightarrow \{a\} \notin F_r$ đối với A mà $A^+ \neq R$, $a \notin A$

Nhận xét 5

Qua định nghĩa, ta có thể thấy dạng chuẩn

BCNF là 3NF và 3NF là 2NF. Tuy vậy, chúng ta có thể đưa ra các ví dụ chứng tỏ có quan hệ là 2NF nhưng không là 3NF và có quan hệ là 3NF nhưng không là BCNF.

Nói cách khác là lớp các quan hệ BCNF là lớp con thực sự của lớp các quan hệ 3NF và lớp các quan hệ 3NF này lại là lớp con thực sự của lớp các quan hệ 2NF.

Đối với s = <F,R> thì các dạng chuẩn 2NF, 3NF, BCNF trong đó ta thay F_r bằng F^+.

Chú ý là đối với sơ đồ quan hệ ta không có dạng 1NF.

Nhận xét 5 cũng đúng cho các dạng chuẩn của sơ đồ quan hệ. Chúng ta xem ví dụ sau

Ví dụ 6.

Cho s = <R, F>, s' = <R, F' > là hai SĐQH trên R = {a, b, c, d} và

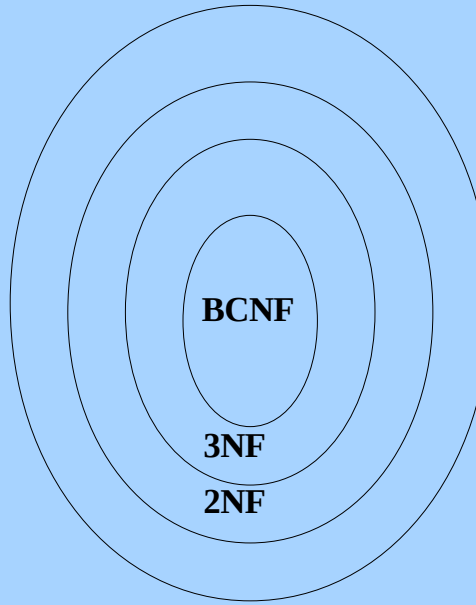
F = {{a} →{c}, {b} →{d}, {c} →{a, b, d}}.

F' = {{a, b} →{c}, {d} →{b}, {c} →{a, b, d}}.

Dễ thấy {a}, {c} là các khoá tối tiểu của s, {b} là thuộc tính thứ cấp. Do đó, s là 2NF, nhưng không là 3NF.

Rõ ràng, {a, b}, {c} là các khoá tối tiểu của s'. Hiển nhiên s là 3NF. Vì ta có {d} →{b}, nên s không là BCNF.

Như vậy việc phân lớp các dạng chuẩn có thể được thể hiện quan hình vẽ sau



2NF

3.2 Dạng chuẩn 2NF

Bây giờ chúng ta nêu ra loại phụ thuộc hàm đặc biệt, mà phụ thuộc dữ liệu này đóng vai trò quan trọng trong dạng chuẩn 2.

Định nghĩa 1.

Một phụ thuộc hàm $A \rightarrow B$ được gọi là sơ cấp nếu không tồn tại một tập hợp $A' \subset A$ sao cho $A' \rightarrow B$. Trong trường hợp này ta cũng nói B phụ thuộc hoàn toàn vào A. Như vậy nếu A là một thuộc tính sơ cấp thì phụ thuộc hàm $A \rightarrow B$ cũng là sơ cấp. Trong trường hợp ngược lại, ta nói B phụ thuộc bộ phận vào A

Định lí 2.

Cho r là một quan hệ trên R. Khi đó r là 2NF khi và chỉ khi

- r là 1NF
- Mỗi thuộc tính thứ cấp của r đều phụ thuộc hoàn toàn vào mọi khoá tối tiểu.

Vì SĐQH không có dạng chuẩn 1, từ Định lí 2 ta có mệnh đề sau

Mệnh đề 3.

Cho s là một sơ đồ quan hệ trên R. Khi đó s là 2NF khi và chỉ khi mọi thuộc tính thứ cấp của s đều phụ thuộc hoàn toàn vào khoá tối tiểu bất kì.

Có thể thấy, bản chất dạng chuẩn 2NF là loại bỏ các phụ thuộc bộ phận giữa các thuộc tính thứ cấp với các khoá tối tiểu.

Định lí 4.

Giả sử s = <R, F> là sơ đồ quan hệ. Đặt $M_s = \{A - a; a \in A, A \in K_s\}$, và F_n là tập tất cả các thuộc tính thứ cấp của s. Đặt $l_s = \{B : B = C^+ , C \in M_s\}$. Khi đó ta có các tương đương sau:

(1) s là 2NF.

(2) Với mỗi $C \in M_s$: $C^+ \cap F_n = \varnothing$;

(3) Với mỗi $B \in l_s$ và $a \in F_n$: $(B - a)^+ = B - a$.

Lời giải: Giả sử s là 2NF. Nếu $F_n = \varnothing$ thì (2) là rõ ràng. Giả thiết rằng $F_n \neq \varnothing$. Do định nghĩa của F_n và của M_s, (3) là hiển nhiên.

Giả sử rằng chúng ta có (2) và $F_n \neq \varnothing$. Nếu có $B \in l_s$, và $a \in F_n$: $B - a \subset (B - a^+)$. Từ định nghĩa của l_s có $C \in M_s$: $C^+ = B$. Rõ ràng rằng $a \in (B - a)^+$. Phù hợp với định nghĩa của bao đóng ta có $(B - a)^+ = C^+ = B$. Từ đó ta thu được $a \in C^+$. Do vậy, $C^+ \cap F_n \neq \varnothing$. Điều này là vô lý. Do đó ta thu được (3).

Bây giờ, giả sử chúng ta có (3) và $F_n \neq \varnothing$. Giả thiết rằng tồn tại $D \subset A$, $A \in K_s$(*) và $a \in F_n$, $a \notin D$, nhưng $D \rightarrow \{a\} \in F^+$. Do (*) và phù hợp với việc xây

dựng của M_s và l_s thì có $C \in M_s : D \subseteq C$. Hiển nhiên rằng $a \notin C$. Rõ ràng, $D^+ \subseteq C^+$ và $a \in C^+$. Đặt $B = C^+$. Có thể thấy $C \subseteq B - a$. Do đó, $B - a \subset C^+ = (B - a)^+$. Điều này mâu thuẫn với $(B - a)^+ = B - a$. Như vậy chúng ta có (1). □

Từ định lý 4 trực tiếp suy ra kết quả sau:

Hệ quả 5.

Giả sử $s = (R, F)$ là một sơ đồ quan hệ. Ký pháp F_n là tập tất cả những thuộc tính thứ cấp của s, và $G_s = \{B - F_n : B \in K_s^{-1}\}$. Khi đó nếu đối với mọi $C \in G_s : C^+ = C$ thì s là 2NF.

Cho $s = \langle R, F\rangle$ là một sơ đồ quan hệ trên R. Đặt $Z(s) = \{X^+ : X \subseteq R\}$. Chúng ta nói rằng s là đơn nếu F chứa chỉ các phụ thuộc hàm dạng $\{a\} \rightarrow \{b\}$. Biết rằng [16] s là đơn nếu và chỉ nếu đối với mọi $A, B \in Z(s) : A \cup B \in Z(s)$. Rõ ràng, từ điều đó ta có $(A \cup B)^+ = A^+ \cup B^+$.

Mệnh đề 6.

Cho $s = \langle R, F\rangle$ là một sơ đồ quan hệ đơn. Đặt F_n là tập tất cả những thuộc tính thứ cấp của s, và $G_s = \{B - F_n : B \in K_s^{-1}\}$. Khi đó s là 2NF nếu và chỉ nếu với mọi $C \in G_s : C^+ = C$.

Lời giải: Giả sử rằng s là 2NF. Nếu $F_n = \phi$ thì bởi định nghĩa của phản khoá ta có $C^+ = C$. Nếu $F_n \neq$

ϕ thì giả thiết rằngcó $C \in G_s : C^+ \neq C$. Bởi định nghĩa của G_s thì tồn tại $B \in K_s^{-1} : C \cup F_n = B$. Biết rằng [9] F_n là giao của tất cả các phản khoá chúng ta có $C \subset B$. Do đó, $C^+ \subseteq B$, $C^+ \cap F_n \neq \phi$. Ta ký pháp các phần tử của C là $c_1,..., c_l$. Bởi vì s là đơn chúng ta thu được $\{c_1\}^+ \cup ... \cup \{c_l\}^+ = C^+$. Do đó tồn tại $c_1 \in C$ sao cho $\{c_1\}^+ \cap F_n \neq \emptyset$. Hiển nhiên c_1 là thuộc tính cơ bản. Điều này trái với việc s có 2NF. Do đó, $C^+ = C$.

Ngược lại bởi hệ quả 5 nếu ta có $C^+ = C$ đối với mọi $C \in G_s$, thì s là 2NF. □

Cho r là một quan hệ trên R. Đặt $A^+_r = \{a : a \in R, A \rightarrow^f_r \{a\}\}$, r là quan hệ đơn nếu đối với mọi $A, B \subseteq R : (A \cup B)^+_r = A^+_r \cup B^+_r$.

Biết rằng đối với một quan hệ r cho trước thì tập hợp tất cả các phản khoá của r được xây dựng trong thời gian đa thức. Trong [9] chúng ta đã chỉ ra rằng giao của tất cả các phản khoá đúng bằng tập tất cả các thuộc tính thứ cấp. Mặt khác biết rằng [6] nếu sơ đồ quan hệ s = <R, F> là đơn thì độ phức tạp thời gian tìm quan hệ r sao cho $F^+ = F_r$ là đa thức. Từ điều này và mệnh đề 6 ta có

Mệnh đề 7.

Cho s là một sơ đồ quan hệ đơn và r là một quan hệ đơn trên R. Khi đó tồn tại một thuật toán đa thức xác định rằng s (r, tương ứng) có là 2NF.

3.3 Dạng chuẩn 3NF

Trong mục này, chúng ta đưa ra một khái niệm quan trọng thường dùng để mô tả dạng chuẩn 3NF

Định nghĩa 1.

Một phụ thuộc hàm $A \rightarrow C$ được gọi là trực tiếp nếu không có B ($B \neq A$ và $B \neq C$) sao cho $A \rightarrow B$ và $B \rightarrow C$ nhưng B không phụ thuộc hàm vào A hoặc C không phụ thuộc hàm vào B. Trong trường hợp nếu có B như vậy thì B được gọi là tập thuộc tính bắc cầu và $A \rightarrow C$ là phụ thuộc bắc cầu.

Ta có một đặc trưng sau cho dạng chuẩn 3**.**

Định lí 2.

Cho r là một quan hệ trên R. Khi đó r là 3NF nếu và chỉ nếu

- r là 2NF

- Không có một thuộc tính thứ cấp nào của r phụ thuộc bắc cầu vào một khoá tối tiểu.

Từ Định lí 2 đối với SĐQH ta cũng có kết quả sau

Mệnh đề 3.

Giả sử s là một sơ đồ quan hệ trên R. Khi đó s là 3NF nếu và chỉ nếu

- s là 2NF

- Mọi thuộc tính thứ cấp của s phụ thuộc trực tiếp vào mỗi khoá tối tiểu.

Trong dạng chuẩn 3NF chúng ta loại bỏ các phụ thuộc bộ phận, phụ thuộc bắc cầu giữa các thuộc tính thứ cấp với các khoá tối tiểu.

Chúng ta trình bày thêm một số đặc trưng của các SĐQH dạng 3NF.

Cho s = <R, F> là một sơ đồ quan hệ trên R. Từ s chúng ta xây dựng Z(s) = $\{X^+ : X \subseteq R\}$, và tính hệ sinh N_s của Z(s). Chúng ta đặt

$T_s = \{A \in N_s : \exists\ B \in N_s : A \subset B\}$

Biết rằng [1] đối với một sơ đồ quan hệ s cho trước thì tồn tại một quan hệ r là quan hệ Amstrong của s. Mặt khác bởi Định lý 17 và Hệ quả 18 mục 2.2, mệnh đề dưới đây là rõ ràng

Mệnh đề 4.

S là một SĐQH. Khi đó $K_s^{-1} = T_s$.

Định lí 5.

Cho s = <R, F> là một sơ đồ quan hệ. Đặt F_n là tập tất cả các thuộc tính thứ cấp của s. Khi đó s là

3NF nếu và chỉ nếu $\forall B \in K_s^{-1}, a \in F_n : (B - a)^+ = B - a$.

Lời giải: Giả sử $F_n \neq \emptyset$. Có thể thấy rằng s là 3NF thì đối với mỗi $B \in K_s^{-1}, a \in F_n : B - a = (B - a)^+$.

Ngược lại, nếu s không là 3NF thì tồn tại một tập A và $a \in F_n : a \notin A$ sao cho $A \rightarrow \{a\} \in F^+$ và $A^+ \neq R$. Phù hợp với Mệnh đề 4 có $B \in K^{-1}_r$ sao cho $A^+ \subseteq B$. Từ $a \in A^+$ chúng ta có $a \in B$. Do $a \notin A$ ta có $A \subseteq B - a$. Do đó ta thu được $B - a \subset (B - a)^+$. □

Định lí 6.

Giả sử r là một quan hệ trên R. Khi đó r là 3NF nếu và chỉ nếu với mọi $A \in E_r$, $a \in A$ và a là thuộc tính thứ cấp thì $\{A- a\}_r^+ = A- a$, ở đây E_r là hệ bằng nhau của r.

Từ Định lí 5 ta có hệ quả sau

Hệ quả 7.

Giả sử s là một sơ đồ quan hệ trên R. Khi đó s là 3NF nếu và chỉ nếu với mọi A : $A^+ = A$, $a \in A$ và a là thuộc tính thứ cấp thì $\{A - a\}^+ = A- a$.

3.4. Dạng chuẩn BCNF

Trong mục này, chúng ta đưa ra một số các đặc trưng của dạng chuẩn BCNF cho sơ đồ quan hệ và quan hệ.

Định nghĩa 1.

Giả sử r là một quan hệ trên R, $A,B \subseteq R$ và $A \rightarrow B$.

Khi đó ta nói A là tập sinh của B nếu

- $|A| < |B|$,

- Không tồn tại tập con thực sự của A mà xác định hàm cho B.

Tập C là tập sinh của quan hệ r nếu có một tập D nào đó để C là tập sinh của D.

Định lí 2

Giả sử r là quan hệ trên R. Khi đó r là BCNF khi và chỉ khi mọi tập sinh của r đều là khoá.

Mệnh đề 3

Cho s = <R, F> là một sơ đồ quan hệ. Đặt F_n là tập tất cả các thuộc tính thứ cấp của s. Khi đó

s là BCNF nếu và chỉ nếu $\forall B \in K_s^1, a \in B : (B - a)^+ = B - a$.

Lời giải: Dễ thấy nếu s là BCNF thì $(B - a)^+ = B - a$ đối với $B \in K_s^{-1}$ và $a \in B$.

Ngược lại giả thiết s không là BCNF. Khi đó tồn tại $A \rightarrow \{a\} \in F^+$ ở đây $A^+ \neq R$ và $a \notin A$. Bởi mệnh đề 4 mục trên, ta có $B \in K^{-1}$ sao cho $A^+ \subseteq B$. Rõ ràng, $a \in B$ và $A \subseteq B - a$. Từ đó, ta có $(B - a)^+ = B$. □

Định lí 4.

Giả sử r là một quan hệ trên R. Khi đó r là BCNF nếu và chỉ nếu với mọi $A \in M_r$, $a \in A$ thì $\{A- a\}_r^+ = A- a$, ở đây M_r là hệ bằng nhau cực đại của r.

Giả sử $A \rightarrow B$ là một phụ thuộc hàm. Chúng ta gọi phụ thuộc hàm này là tầm thường nếu $B \subseteq A$. Ngược lại trong trường hợp này, chúng ta gọi nó là phụ thuộc hàm không tầm thường.

Định lí 5

Giả sử s = < R,F> là một sơ đồ quan hệ trên R. Khi đó s là BCNF nếu và chỉ nếu với mọi $A \rightarrow B \in F$ và $A \rightarrow B$ không tầm thường thì $A^+ = R$.

3.5. Các thuật toán liên quan

Trên cơ sở các định lí đã trình bày ở các mục trên, chúng ta xây dựng các thuật toán để xác định dạng chuẩn cho các quan hệ hoặc sơ đồ quan hệ cho trước.

Đầu tiên chúng ta xây dựng thuật toán xác định một quan hệ cho trước có là 3NF hay không.

Thuật toán 1.

Đầu vào: $r = \{h_1, ..., h_m\}$ là một quan hệ trên R

Đầu ra : r là 3NF ?

Bước 1: Từ r chúng ta xây dựng một tập $E_r = \{E_{ij} : m \geq j > i \geq 1\}$, ở đây $E_{ij} = \{ a \in R : h_j(a) = h_j(a)\}$.

Bước 2: Từ E_r chúng ta xây dựng một tập $M = \{B \in P(R) :$ Tồn tại $E_{ij} \in E_r : E_{ij} = B\}$.

Bước 3: Từ M xây dựng tập $M_r = \{ B \in M :$ Với mọi $B' \in M : B \not\subset B'\}$.

Có thể thấy rằng M_r tính được bằng một thuật toán thời gian đa thức.

Bước 4: Xây dựng tập $V = \cap M_r$.

Bước 5: r là 3NF nếu với mọi $B \in M_r$, $a \in V$: $\{B - a\}_r^+ = B - a$. Ngược lại r không là 3NF.

Ví dụ : Cho quan hệ r sau

A	B	C	D	E
0	0	1	0	1
1	1	0	0	1
2	2	0	1	3
1	2	3	1	0

1 1 1 0 2

Khi đó E_{12}= DE, E_{13}= ∅, E_{14}= ∅, E_{15}= D, E_{23}= C, E_{24}= A, E_{25}=AB,

E_{34}=BD, E_{35}=∅, E_{45}=A.

Như vậy ta có M_r= {DE,AB,BD,C}. Dễ thấy DE ∩ AB ∩ BD ∩ C = ∅.

Cho nên r là 3NF.

Trên cơ sở Định lí 4 mục trên chúng ta xây dựng thuật toán dưới đây

Thuật toán 2.

Đầu vào: r = {h_1, ..., h_m }là một quan hệ trên R

Đầu ra: r là BCNF ?

Bước 1: Từ r chúng ta xây dựng một tập E_r = {E_{ij} : m ≥ j > i ≥ 1} và E_{ij} = {a ∈ R : $h_j(a) = h_j(a)$}

Bước 2: Từ E_r chúng ta xây dựng một tập M = {B ∈ P(R) : Tồn tại E_{ij} ∈ E_r : E_{ij} = B}

Bước 3: Từ M xây dựng tập M_r = {B ∈ M : Với mọi B' ∈ M : B ⊄ B'}. Có thể thấy rằng M_r tính được bằng một thuật toán thời gian đa thức.

Bước 4: r là BCNF nếu với mọi $B \in M_r$, $a \in B$: $\{B - a\}_r^+ = B - a$. Ngược lại r không là BCNF.

Ví dụ : Cho quan hệ r :

A	B	C	D	E
2	0	1	1	1
1	1	0	0	1
2	2	0	3	3
1	2	3	3	0
1	1	1	0	2

Khi đó E_{12}= {E}, E_{13}= {A}, $E_{14}=\varnothing$, E_{15}= {C}, E_{23}= {C}, E_{24}= {A}, E_{25}={A,B}, E_{34}={B,D}, $E_{35}=\varnothing$, E_{45}={A}.

Như vậy ta có M_r= {{A,B},{B,D},{C},{E}}. Có thể kiểm tra rằng {A,D} - A = D và $\{D\}^+_r$ = {B,D}. Vì thế r không là BCNF.

Nhờ Định lí thuật toán dưới đây được xây dựng

Thuật toán 3.

Đầu vào: s = <R,F> là một sơ đồ quan hệ trên R, với

$F = \{ A_1 \rightarrow B_1,..., A_m \rightarrow B_m \}$

Đầu ra: s là BCNF ?

Bước 1: Nếu $A_1 \rightarrow B_1$ là phụ thuộc hàm không tầm thường và A_1^+ # R thì dừng và kết luận s không là BCNF. Ngược lại thì chuyển sang bước tiếp theo.

.....

Bước m: Giống như bước 1 nhưng đối với $A_m \rightarrow B_m$.

Bước m+1: s là BCNF.

Ví dụ : Cho sơ đồ quan hệ s =<R,F>

R = {a,b,c,d,e}

F={{a,b}→{d},{b,c}→{e}, {d}→{c}}

Ta có $\{a,b\}^+ = R$, nhưng $\{b,c\}^+ \neq R$. Vậy s không là BCNF.

Vì thời gian tính bao đóng của một tập thuộc tính bất kì của một sơ đồ quan hệ hoặc một quan hệ là đa thức. Cho nên chúng ta có các kết luận sau.

Định lí 4.

Cho trước một quan hệ r và một sơ đồ quan hệ s. Khi đó đều tồn tại một thuật toán có độ phức tạp thời gian đa thức theo kích thước của r (s) để kiểm tra r (s) có là BCNF hay không.

Định lí 5

Cho trước r là một quan hệ trên R. Khi đó tồn tại một thuật toán có độ phức tạp thời gian đa thức để kiểm tra r có là 3NF hay không.

Tuy vậy, đối với đầu vào là s thì đây lại là bài toán NP đầy đủ.

Định lí 6

Cho trước s là một sơ đồ quan hệ trên R. Khi đó bài toán xác định s có là 3NF hay không là NP - đầy đủ.

Có nghĩa là cho đến nay, độ phức tạp thời gian của bài toán này không là đa thức.

- Với trường hợp 2NF, các câu hỏi tương tự cho cả r lẫn s còn là bài toán mở (Chúng tôi phỏng đoán có độ phức tạp thời gian là hàm mũ trở lên)

3.6 Dạng chuẩn của các hệ khoá

Bây giờ chúng ta khảo sát các sơ đồ quan hệ mà tập các khoá tối thiểu của nó là một hệ Sperner cho trước.

Định nghĩa 1.

Cho K là hệ Sperner trên R. Ta nó rằng K là 2NF (3NF, BCNF, tương ứng) nếu với mỗi sơ đồ quan hệ s = <R, F> mà K_s = K thì s là 2NF (3NF, BCNF tương ứng).

Bây giờ chúng ta cho một điều kiện cần và đủ để một hệ Sperner bất kỳ là 2NF.

Cho K là một hệ Sperner trên R. Đặt $K_p = \{a \in R : \exists A \in K : a \in A\}$, và $K_n = R - K_p$. K_p (K_n) được gọi là tập các thuộc tính cơ bản (thứ cấp) của K.

Cho trước sơ đồ quan hệ s = <R, F>, chúng ta nói rằng phụ thuộc hàm $A \rightarrow B \in F$ là thừa nếu hoặc A = B hoặc có $C \rightarrow D \in F$ sao cho $C \subseteq A$ và $B \subseteq D$.

Định lí 2.

Cho K là hệ Sperner trên R. Khi đó K là 2NF nếu và chỉ nếu $K_n = \varnothing$.

Lời giải: Theo định nghĩa của quan hệ 2NF, hệ Sperner 2NF và K_n ta có thể thấy nếu $K_n = \varnothing$ thì K là 2NF.

Bây giờ ta giả thiết là K là 2NF. Kí pháp K^{-1} là tập các phản khoá của K. Từ K, K^{-1} ta xây một SĐQH như sau

Đối với mỗi $A \subset R$ có $B \in K^{-1}$ sao cho $A \subseteq B$. Đặt $C = \cap \{B \in K^{-1} : A \subseteq B\}$. Ta lập $A \rightarrow C$. Kí pháp T là tập tất cả các phụ thuộc hàm như thế. Đặt $F = \{ E \rightarrow R : E \in K\} \cup (T - Q)$, ở đây $Q = \{X \rightarrow Y \in T : X \rightarrow Y$ là phụ thuộc hàm thừa}. Từ Định lí 13 phần 2.2 và định nghĩa của hệ Sperner ta thu được $K_s = K$.

Rõ ràng, với mỗi SĐQH bất kì $s_1 = (R, F_1)$ sao cho $K_{s1} = K$ và $A \subseteq R$ ta có $A^+_{s1} \subseteq A^+_s$, ở đây $A^+_{s1} = \{a: A \rightarrow \{a\} \in F_1^+\}$. Chúng ta đã chứng minh rằng [9]

K_n là giao của các phản khoá của s. Trên cơ sở việc xây dựng s = <R,F> và phù hợp với định nghĩa của hệ Sperner 2NF ta có $K_n = \varnothing$. □

Dễ thấy SĐQH 3NF là 2NF và nếu tập các thuộc tính thứ cấp là rỗng thì SĐQH này 3NF. Do đó từ Định lí 2 ta suy ra ngay hệ quả sau

Hệ quả 3.

Cho K là hệ Sperner trên R. Khi đó K là 3NF nếu và chỉ nếu $K_n=\varnothing$.

Định nghĩa 4.

Cho K là hệ Sperner trên R. Ta nói K là đơn nhất nếu K xác định duy nhất SĐQH s = <R,F>, theo nghĩa đối với mọi SĐQH s' =<R,F '> mà $K_{s'}$ = K thì ta có $F^+ = F'^+$.

Từ định nghĩa hệ Sperner BCNF và Định nghĩa 4 ta có

Mệnh đề 5.

K là BCNF nếu và chỉ nếu K là đơn nhất.

Như đã biết [5] đối với hệ Sperner cho trước K tồn tại SĐQH s (tương ứng quan hệ r) sao cho K_s =K (tương ứng K_r =K). Ta nói s (tương ứng r) là đơn nhất nếu K_s (tương ứng K_r) xác định duy nhất s

(tương ứng r). Có nghĩa là K_s (tương ứng K_r) là đơn nhất.

Bây giờ ta cho một điều kiện cần và đủ để một SĐQH là đơn nhất.

Định lí 6.

Cho s = (R,F) là SĐQH trên R. Khi đó s là đơn nhất nếu và chỉ nếu đối với mọi $a \in A$, $A \in K^{-1}_s$: $A-a = \cap \{B \in K^{-1}_s : (A- a) \subset B\}$

Lời giải:

Chúng ta biết rằng hệ Sperner K là đơn nhất khi và chỉ khi đối với mọi $B \subseteq A, A \in K^{-1}$, B là giao của các phản khoá. Đặt $P_s = \{A-a: \in K^{-1}_s, a \in A\}$

Có thể thấy nếu s=<R,F> là đơn nhất thì $B \in P_s$ kéo theo B là giao của các phản khoá. Có nghĩa là $B = \cap \{A \in K_s^{-1}:B \subset A\}$.

Ngược lại giả sử với mỗi $B \in P_s$ ta có $B = \cap \{A \in K_s^{-1}:B \subseteq A\}$(*).. Do mệnh đề 3 mục 3.4 và Mệnh đề 4 mục 3.3 ta có $N_s \subseteq (P_s \cup K_s^{-1})$. Có thể thấy s là BCNF. Trên cơ sở định nghĩa của N_s và Mệnh đề 4 mục 3.3 ta có $K_s^{-1} \subseteq N_s$. Phù hợp với (*) ta thu được $K_s^{-1} = N_s$. Vì s là BCNF nên đối với mọi $B \subseteq A$, $A \in K_s^{-1}$: $B^+ = B$. Như vậy B là giao của các phản khoá của s. □

Theo định nghĩa của hệ Sperner BCNF, Định lí 6 và Mệnh đề 5 ta cho một điều kiện cần và đủ để một hệ Sperner là BCNF.

Địnhlí 7.

Giả sử K là hệ Sperner trên R. Khi đó K là BCNF nếu và chỉ nếu đối với mọi $a \in A$, $A \in K^{-1}$: $A - a = \cap \{B \in K^{-1} : (A - a) \subset B\}$.

Phù hợp với Định lí 6 và thuật toán đa thức tìm tập các phản khoá của một quan hệ cho trước, ta có mệnh đề sau

Mệnh đề 8.

Tồn tại thuật toán xác định một quan hệ cho trước có là đơn nhất hay không. Độ phức tạp thời gian của thuật toán này là đa thức theo kích thước của R và r.

Từ Định lí 7 và Mệnh đề 8 trực tiếp kéo theo mệnh đề sau

Mệnh đề 9.

Tồn tại thuật toán đa thức xác định tập các khoá tối tiểu của một quan hệ cho trước là BCNF hay không.

Từ Định lí 6 suy ra ngay hệ quả sau

Hệ quả 10.

Cho K là hệ Sperner trên R . Khi đó có thuật toán đa thức xác định hệ Sperner H có là đơn nhất hay không, ở đây $H^{-1} = K$.

3.7. Ví dụ

Dưới đây chúng ta cho ví dụ minh hoạ việc phân tách một bảng (quan hệ) thành các bảng ở dạng chuẩn 3NF.

Trong một nhà máy, hàng ngày người ta xuất vật tư theo phiếu xuất kho như sau:

Phiếu xuất kho

Số phiếu	Ngày xuất	Người nhận	Địa chỉ người nhận	Mã vật tư
1010001	26/10/96	Phạm An	2 Phố Huế, Hà Nội	10100 20018 10703
1020004	12/01/97	Trần Hà	14 Lê Lợi, TP. HCM	30101
1170003	17/03/97	Trần Hà	14 Lê Lợi, TP. HCM	10100 20904

Trong ví dụ này có hai thuộc tính không sơ cấp. Đó là :

- 'Địa chỉ người nhận' là một thuộc tính tổng hợp những thuộc tính sơ cấp sau: 'Số và phố', 'Tên TP'.

- 'Mã vật tư' là danh sách các vật tư của một hoá đơn, có chiều dài không nhất định, cần được tách riêng ra từng vật tư.

Ta có thể biến đổi quan hệ phiếu xuất kho sang dạng chuẩn 1 như sau

Phiếu xuất kho

Số phiếu	Ngày xuất	Người nhận	Số và phố	Thành phố	Mã vật tư
1010001	26/10/96	Phạm An	2 Phố Huế	Hà Nội	10100
1010001	26/10/96	Phạm An	2 Phố Huế	Hà Nội	20018
1010001	26/10/96	Phạm An	2 Phố Huế	Hà Nội	10703
1020004	12/01/97	Trần Hà	14 Lê Lợi	TPHCM	30101
1170003	17/03/97	Trần Hà	14 Lê Lợi	TPHCM	10100
1170003	17/03/97	Trần Hà	14 Lê Lợi	TPHCM	20904

Trong quan hệ Phiếu xuất kho ta nhận thấy là tập {Số phiếu, Mã vật tư} là khoá tối tiểu. Dễ thấy các thuộc tính Ngày xuất, Người nhận, Số và phố, Thành phố phụ thuộc hàm vào thuộc tính số phiếu. Như vậy, quan hệ Phiếu xuất kho không là 2NF (Nếu lưu trữ và xử lí trên quan hệ này sẽ dẫn đến trùng lặp dữ liệu). Do đó, ta tách thành 2 quan hệ riêng biệt:

Phiếu kho

Số phiếu	Ngày xuất	Người nhận	Số và phố	Thành phố
1010001	26/10/96	Phạm An	2 Phố Huế	Hà Nội
1020004	12/01/97	Trần Hà	14 Lê Lợi	TPHCM
1170003	17/03/97	Trần Hà	14 Lê Lợi	TPHCM

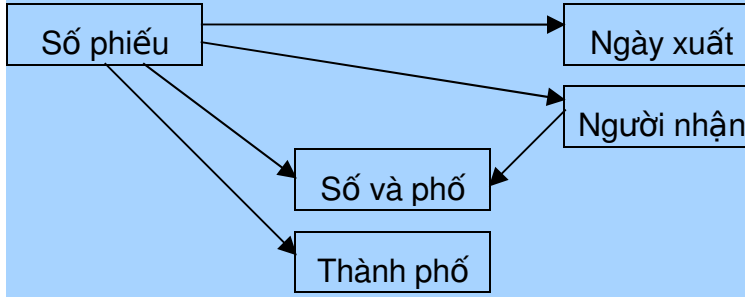
Vật tư

Số phiếu	Mã vật tư
1010001	10100
1010001	20018
1010001	10703

1020004	30101
1170003	10100
1170003	20904

Ta có thể thấy quan hệ vật tư là 3NF.

Trong quan hệ Phiếu kho là 2NF ở trên, ta thấy trên đồ thị của các phụ thuộc hàm có hai con đường để đi từ Số phiếu đến {Số và phố, Thành phố} hoặc đi qua thuộc tính Người nhận. Như vậy tồn tại phụ thuộc bắc cầu trong quan hệ này.



Điều này chứng tỏ quan hệ này chưa là 3NF, Nếu ta lưu quan hệ này thì khi xử lí sẽ dẫn đến trùng

lặp địa chỉ của người nhận. Do vậy ta tách nó thành hai thực thể riêng biệt:

Phiếu

Số phiếu	Ngày xuất	Người nhận
1010001	26/10/96	Phạm An
1020004	12/01/97	Trần Hà
1170003	17/03/97	Trần Hà

Người nhận

Người nhận	Số và phố	Thành phố
Phạm An	2 Phố Huế	Hà Nội
Trần Hà	14 Lê Lợi	TP. HCM

Như vậy, ta đã tách quan hệ phiếu xuất kho thành 3 quan hệ dạng chuẩn 3 là phiếu, người nhận, vật tư.

Chương 4
Một số phép toán xử lí bảng

Ngôn ngữ xử lí dữ liệu là một phần quan trọng trong các hệ quản trị cơ sở dữ liệu. Ngay từ năm 1970 E.F.Codd đã đưa ra hai ngôn ngữ xử lí dữ liệu chính. Đó là ngôn ngữ đại số quan hệ và ngôn ngữ tính toán quan hệ (Chủ yếu dựa vào phép toán tân từ). Hầu hết ngôn ngữ xử lí của các hệ quản trị cơ sở dữ liệu lớn hiện nay đều chứa ngôn ngữ đại số quan hệ. Trong chương này chúng tôi trình bày các phép toán cơ bản của ngôn ngữ đại số quan hệ này.

Trong Giáo trình này chúng ta coi bảng, file dữ liệu, quan hệ (theo dạng Codd) là tương đương nhau.

4.1 Các phép toán cơ bản

Để minh hoạ bằng các ví dụ làm sáng tỏ tính chất của các phép toán xử lí bảng chúng ta cho 2 quan hệ sau:

A	B	C
a	a	b
a	c	b
b	c	d
a	a	d

Quan hệ r

D	A	E
a	a	b
b	c	d
e	f	g

Quan hệ t

Hình 1

1. Phép hợp (r ∪ t)

Giả sử r và t là các quan hệ n cột

Khi đó quan hệ hợp r ∪ t là quan hệ n cột bao gồm các bản ghi (dòng) của cả r lẫn t.

Chú ý những bản ghi giống nhau chỉ giữ lại một.

Nếu r và t là các quan hệ có tên các cột khác nhau thì quan hệ hợp không có tên các cột

Với r, t trong hình 1, ta có r ∪ t =

a	a	b
a	c	b
b	c	d
a	a	d
e	f	g

2. Phép trừ (r - t)

Giả sử r và t là hai quan hệ n cột.

Quan hệ hiệu (kí hiệu là r-t) là quan hệ n cột mà bao gồm các dòng của r nhưng không có mặt trong t

Nếu r và t có các tên cột khác nhau, thì quan hệ hiệu không có tên các cột.

Ví dụ: r - t =

a	c	b
a	a	d

3. Phép giao (r ∩ t)

Giả sử t và t là hai quan hệ n cột.

Khi đó quan hệ giao của r và t là quan hệ n cột bao gồm các bản ghi có mặt cả ở trong r lẫn t.

Trong trường hợp nếu r và t có các tên cột khác nhau (tên các thuộc tính) thì các cột của quan hệ giao không có tên.

Ví dụ: r ∩ t =

a	a	b
b	c	d

4. Tích Đề các

Giả sử có quan hệ r có n cột (R_1= {a_1,...,a_n}) và t có m cột (R_2= {b_1,...,b_m})

a_1 .....a_n b_1 .....b_m

r = ... t = ...

Tích Đề các:

r x t nếu $(r_1.....r_n) \in r$ và

$(t_1.....t_m) \in t$

Thì $(r_1,....,r_n, t_1,....., t_m) \in rxt$

Ví dụ:

	A	B	C
	a	a	b
r =	a	c	b
	b	c	d
	a	a	d

	D	A	E
	a	a	b
t =	b	c	d
	e	f	g

Tích Đề các:

	r.A	B	C	D	t.A	E
r x t =	a	a	b	a	a	b
	a	a	b	b	c	d
	a	a	b	e	f	g
	a	c	b	a	a	b

a	c	b	b	c	d
a	c	b	e	f	g
b	c	d	a	a	b
b	c	d	b	c	d
b	c	d	e	f	g
a	a	d	a	a	b
a	a	d	b	c	d
a	a	d	e	f	g

Tích Đề các là một quan hệ. Trong trường hợp có cột giống nhau (tên giống nhau) cần đánh dấu để phân biệt. Trong ví dụ trên đó là r.A và t.A

Số lượng bản ghi (dòng) là n.m.

5. Phép chiếu

Giả sử có r là một quan hệ gồm m cột ($R_1 = \{a_1...a_m\}$). Khi ấy phép chiếu (Ký hiệu là: ∏) lên tập r:

$\prod_{i1,i2...ip}$ (r)

ij là số thứ tự lấy trong tập từ 1 đến m

j = 1,...,p, ở đây chỉ số p ≤ m.

hoặc $\prod_{a,b,...,\ t}$ (r), ở đây a, b, ... t là tên các thuộc tính.

Khi đó ta thực hiện phép chiếu như sau:

Giữ lại p cột có số hiệu là i_1, i_2, i_p, loại bỏ các dòng bị trùng nhau.

Ví dụ: (lấy ví dụ trước).

$\Pi_{1,2}$ (r) =

A	B
a	a
a	c
b	c

6. Phép chọn

Phép toán này rút ra các bản ghi thoả mãn điều kiện nào đấy.

Gọi F là một điều kiện nếu nó bao gồm:

- Các toán hạng (hằng, tên thuộc tính)

- Các quan hệ số học (<, =, >, ≤ , ≥ , #)

- Các phép toán logic: và (∧), hoặc (∨) và phủ định. Riêng với hằng ta bao bằng dấu ' '

Ví dụ: F là điều kiện A = 'a' v C = 'b'

Ký hiệu phép chọn: δ_F (r) (F: điều kiện).

Khi đó δ_F là phép chọn được thực hiện bằng việc rút ra từ r (và loại bỏ các dòng trùng) các bản ghi thoả mãn F.

Ví dụ:

δ_A = 'a' $\vee$C= 'b' (r) =

A	B	C
a	a	b
a	c	b
a	a	d

4.2 Các phép toán khác

Các phép toán trình bày tiếp theo đây được biểu diễn qua các phép toán ở mục trên.

1. Phép chia (r ÷ s):

Giả sử r là quan hệ n cột, t là quan hệ m cột

ở đây n > m và t ≠ ∅. Quan hệ thương của r và t là quan hệ được tạo ra như sau:

- Tích $A_1 = \prod_{1,2,...,n-m}(r)$
- Tích $A_2 = A_1 \times t$
- Tích $A_3 = A_2 - r$
- Tích $A_4 = \prod_{1,2,....,n-m}(A_3)$

Cuối cùng $r \div s = \prod_{1,2,....,n-m}(r) - A_4$

Như vậy chúng ta có:

$r \div s = \prod_{1,2,....,n-m}(r) - \prod_{1,2,....,n-m}((\prod_{1,2,....,n-m}(r) \times t) - r)$

Ví dụ:

a	b	c	d
a	b	e	f
b	c	e	f
e	d	c	d
e	d	e	f
a	b	d	c
b	b	b	b

Quan hệ r

c	d
e	f

a	b
e	d

Quan hệ t

Quan hệ r ÷ t

2. Phép nối θ

Giả sử r là quan hệ n cột, t là quan hệ m cột, θ là một trong các quan hệ số học: < , ≤ , = , # , > , ≥

i và j là tên hoặc là số cột tương ứng của r và t

Khi đó kết quả của phép nối θ hai quan hệ r và t là quan hệ (n + m) cột bao gồm các bản ghi của tích đề các r x t, mà các bản ghi này thoả mãn quan hệ số học θ giữa các giá trị thuộc tính i và j.

Ký pháp nó là r >< t

Có thể thấy $r \underset{i\theta j}{\overset{\theta}{><}} t = \delta_{i\,\theta\,(n+j)}(r \times t)$

Với ví dụ như ở mục trên ta có:

r.A	B	C	D	t.A	E
a	a	b	a	a	b
a	c	b	a	a	b
a	a	d	a	a	b
b	c	d	b	c	d

$r \underset{r.A=D}{><} t$

3. Phép nối

Giả sử có r là một quan hệ n cột ($R_1 = \{a_1...a_n\}$), r là quan hệ m cột ($R_2 = \{b_1,...,b_m\}$). Khi đó chúng ta ký hiệu $r \bowtie t$ là phép nối.

- Nó được thực hiện bởi biểu thức trên cơ sở các phép toán sau:

$r \bowtie t = \prod_{i_1, i_2...i_{n+m-q}} (\delta r.A_1 = t.A_1 \wedge ... \wedge r.A_q = t.A_q (r \times t))$

$A_1,...,A_q$ là q cột có tên trùng nhau ở hai quan hệ r và t.

- Phép toán được thực hiện:

+ Tích Đề các r và t.

+ Chọn trong tích Đề các những bản ghi thoả mãn những cột giống tên nhau phải trùng giá trị trên p cột.

+ Phép chiếu loại bỏ p cột giống nhau (Chỉ giữ lại một).

Ví dụ như hai quan hệ trong hình 1 của phần này, chúng ta có thể thấy r ⋈ t là bảng sau.

A	B	C	D	E
a	a	b	a	b
a	c	b	a	b
a	a	d	a	b

Phép nối là một phép xử lý bảng rất quan trọng. Thông thường chúng ta phân tách file dữ liệu lớn ban đầu thành các file dữ liệu nhỏ ở dạng chuẩn 3NF. Nhờ phép nối các file dữ liệu nhỏ với nhau chúng ta có thể phục hồi file dữ liệu lớn ban đầu và dung tích bộ nhớ để lưu trữ các file dữ liệu nhỏ thường nhỏ hơn dung tích dùng để lưu trữ file dữ liệu lớn ban

đầu. Như vậy, nhờ phép nối chúng ta tiết kiệm được việc lưu trữ các bảng.

4.3. Các ví dụ

Bây giờ chúng ta đưa ra ví dụ minh hoạ việc sử dụng các phép toán xử lý bảng

Một cửa hàng tổng hợp mỗi ngày có bản tổng kết bán hàng như sau:

1. Bản tổng kết bán hàng một ngày từ những hoá đơn bán ra

Đơn vị tính 1000đ

Ngày tháng	Mã hàng	Tên hàng	Đơn vị tính	Số lượng
210397	M1	Radio	1 000	1
	M3	TV	4 000	2
	M6	Xe đạp	1 000	1
			Tổng giá tiền:	10 000
			Đã thanh toán:	6 000

ở đây 210397 là ngày bán: Ngày 21 tháng 03 năm 1997. Trên cơ sở của bản tổng kết này chúng ta xây

dựng một quan hệ (bảng) bán hàng như sau gồm 7 cột và cho các số liệu cụ thể.

Bán hàng

Ngày tháng	Mã hàng	Tên hàng	Đơn giá	Số lượng	Tổng	Thanh toán
210397	M1	Radio	1 000	1	10 000	6 000
210397	M3	TV	4 000	2	10 000	6 000
210397	M6	Xe đạp	1 000	1	10 000	6 000
220397	M2	Máy giặt	3 000	2	6 000	2 000
230397	M1	Radio	1 000	3	15 000	11 000
230397	M4	Video	5 000	2	15 000	11 000

230397	M9	Máy ảnh	2 000	1	15 000	11 000

Có thể thấy quan hệ bán hàng có khoá tối tiểu là Ngày tháng, Mã hàng.

2. Chúng ta sẽ tách từ bảng bán hàng thành 4 bảng sau:

khối lượng

Ngày tháng	Mã hàng	Số lượng
210397	M1	1
210397	M3	2
210397	M6	1
220397	M2	2
230397	M1	3
230397	M4	2
230397	M9	1

Khoá tối tiểu của quan hệ Khối lượng là Ngày tháng, Mã hàng

Doanh số

Ngày tháng	Tổng	Thanh toán

210397	10000	6000
220397	6000	2000
230397	15000	11000

Khoá tối tiểu là Ngày tháng

Hàng

Mã hàng	Tên hàng
M1	Radio
M2	Máy giặt
M3	TV
M4	Video
M6	Xe đạp
M9	Máy ảnh

Khoá tối tiểu là Mã hàng

Mặt hàng

Tên hàng	Đơn giá
Radio	1000
Máy giặt	3000

TV	4000
Video	5000
Xe đạp	1000
Máy ảnh	2000

Khoá tối tiểu là Tên hàng

3. Có thể thấy (Nếu không kể đến thứ tự của cột và hàng):

bán hàng = khối lượng >< doanh số >< Hàng ⋈ Mặt hàng

Không khó khăn lắm chúng ta thấy 4 quan hệ khối lượng, doanh số, Hàng, mặt hàng là 3NF, còn quan hệ bán hàng chưa được chuẩn hoá. Để thực hiện việc xử lí thông tin, chúng ta lưu trữ 4 quan hệ đã được chuẩn hóa, chứ không lưu trữ quan hệ bán hàng.

Như vậy nhờ phép nối chúng ta có thể hồi phục được quan hệ Bán Hàng

Bây giờ chúng ta xử dụng các phép toán xử lí bảng để tìm kiếm và in ra các thông tin sau

- Chúng ta muốn biết doanh số bán ra sau ngày 21 tháng 03 năm 1997 đó là

$\Pi_{\text{Ngày tháng,Tổng}}(\delta_{\text{Ngày tháng' > '210397'}}$ (doanh số)) và in ra bảng sau:

Ngày tháng	Tổng
220397	6 000
230397	15 000

- Chúng ta muốn biết các tên hàng và số lượng đã bán trong ngày 21 tháng 03 năm 1997.

$\Pi_{\text{Tên hàng, số lượng}}$ ($\delta_{\text{Ngày tháng='210397'}}$ (Khối lượng) $\bowtie$ Hàng)

Tên hàng	Số lượng
Radio	1
TV	2
Xe đạp	1

- Tìm các ngày mà trong các ngày đó doanh số bán ra ít nhất là 10.000.000đ

$\Pi_{\text{Ngày tháng}}$ ($\delta_{\text{Tổng}\geq\text{'10 000'}}$ (doanh số))

Ngày tháng
210197
230397

- In ra các mã và tên hàng mà đơn giá của nó nhỏ hơn 3.000.000đ

$\Pi_{\text{Tên hàng, Mã hàng}}$ ($\delta_{\text{Đơn giá'<'3 000'}}$ (Mặt hàng)).

Tên hàng	Mã hàng
Radio	M1
Xe đạp	M6
Máy ảnh	M9

- Cho các mã hàng và đơn giá của chúng trong ngày 23 tháng 03 năm 1997.

$\Pi_{\text{Mã hàng}}$ ($\delta_{\text{Ngày tháng = '230397'}}$ (khối lượng)) >< $\Pi_{\text{Mã hàng, Đơn giá}}$ (Hàng $\bowtie$ Mặt Hàng).

Mã hàng	Đơn giá
M1	1000
M4	5000
M9	2000

- Tìm tên, đơn giá của mã hàng M1 và số lượng bán ra của mặt hàng này trong ngày 23 tháng 03 năm 1997

$\Pi_{\text{Tên hàng, Đơn giá, Số lượng}}$ ($\Pi_{\text{Mã hàng, Số lượng}}$ ($\delta_{\text{Ngày tháng = '230397' } \wedge \text{Mã hàng = M1}}$ (khối lượng) $\bowtie$ hàng $\bowtie$ Mặt hàng).

Tên hàng	Đơn giá	Số lượng
Radio	1000	3

Ví dụ: Cho 2 quan hệ

r_1 =

Chuyến bay	Máy bay
83	727
83	747
84	727
84	747
109	707

r_2 =

Phi công	Máy bay

Tuấn	707
Tuấn	727
Thành	747
Thắng	727
Thắng	747

Chúng ta cần in ra một bảng chỉ ra các phi công có thể lái cho mỗi chuyến bay. Khi đó chúng ta chỉ cần thực hiện phép nối tự nhiên giữa r_1 và r_2.

$$r_3 = r_1 \bowtie r_2$$

Chuyến bay	Máy bay	Phi công
83	727	Tuấn
83	727	Thắng
83	747	Thành
83	747	Thắng

84	727	Tuấn
84	727	Thắng
84	747	Thành
84	747	Thắng
109	707	Tuấn

Đơn giản hơn ta thực hiện

$\Pi_{\text{Chuyến bay, Phi công}}(r_3)$

Chuyến bay	Phi công
83	Tuấn
83	Thắng
83	Thành
84	Tuấn
84	Thắng
84	Thành

109	Tuấn

Chương 5

Một số áp dụng mô hình dữ liệu trong các hệ quản trị CSDL hiện có

5.1. Mô tả chung

Chương này mô tả việc áp dụng các khái niệm của chương 2, 3 trong các hệ QTCSDL hiện có trên thị trường. Các khái niệm này bao gồm thực thể,

thuộc tính, khoá , quan hệ, phụ thuộc hàm, các dạng chuẩn.

5.2. Những khái niệm cơ bản

Trong phần này chúng tôi nêu lại một vài khái niệm đã được trình bày sơ bộ ở chương 2.

5.2.1. Thực thể

Thực thể là một hình ảnh tượng trưng cho một đối tượng cụ thể hay một khái niệm trừu tượng nhưng có mặt trong thế giới thực.

Ví dụ:

Dự án, con người, sản phẩm, ...

Thông thường khi xây dựng mô hình dữ liệu các thực thể được biểu diễn bằng những hình chữ nhật ví dụ như

Sản phẩm

5.2.2. Thuộc tính

Trong một hệ thông tin, ta cần lựa chọn một số tính chất đặc trưng để diễn tả một thực thể, các tính chất này được gọi là thuộc tính của thực thể được mô tả và đây cũng chính là các loại thông tin dữ liệu cần quản lí.

Ví dụ:

Họ tên, địa chỉ, ngày sinh của thực thể 'sinh viên'

Nhãn hiệu, giá của thực thể 'sản phẩm'

Giá trị các thuộc tính của một thực thể cho phép diễn tả một trường hợp cụ thể của thực thể, gọi là một thể hiện của thực thể đó .

Ví dụ:

('Trần Văn Sơn', '204 Triệu Việt Vương - Hà Nội', 12-5-1975) là một thể hiện của 'sinh viên'

('Máy vi tính ACER', 1349) là một thể hiện của 'sản phẩm'

Một thuộc tính là sơ cấp khi ta không cần phân tích nó thành nhiều thuộc tính khác, tuỳ theo nhu cầu xử lí trong hệ thông tin đối với một thực thể.

Thông thường một thực thể tương ứng với một bảng (hay một quan hệ của Codd).

Mỗi thực thể phải có ít nhất một thuộc tính mà mỗi giá trị của nó vừa đủ cho phép nhận diện một cách duy nhất một thể hiện của thực thể, gọi là thuộc tính nhận dạng hay là khoá. Có nhiều trường hợp chúng ta phải dùng một tập các thuộc tính để nhận diện thực thể. Khi một thực thể có nhiều khoá, người ta chọn một trong số đó làm khoá chính (khóa

tối tiểu). Giá trị của một khoá luôn luôn được xác định.

Ví dụ:

Số hoá đơn là thuộc tính nhận dạng của thực thể "Hoá đơn".

Không thể có hai hay nhiều hoá đơn có cùng số hoá đơn trong cùng một hệ thông tin.

Ví dụ:

Hoá Đơn
Số Hoá Đơn Khách Hàng Giá Tiền

5.2.3. Quan hệ

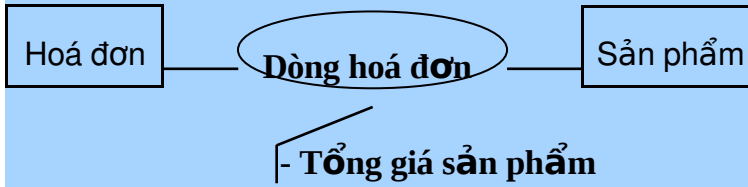
Khái niệm quan hệ ở mục này (khác với khái niệm quan hệ của Codd) được dùng để nhóm họp 2 hay nhiều thực thể với nhau nhằm biểu hiện một mối liên quan tồn tại trong thế giới thực giữa các thực thể này. Kích thước của một quan hệ là số thực thể đã cấu thành nên quan hệ, và có thể là một số

nguyên bất kỳ. Tuy vậy, trong thực tiễn, người ta luôn tìm cách tránh dùng đến những quan hệ có kích thước lớn hơn 3.

Trong một mô hình dữ liệu các quan hệ được biểu diễn bằng những hình tròn hoặc elipse. Trong một số trường hợp, mối quan hệ cũng có thể có những thuộc tính riêng.

Ví dụ:

Hoá đơn dùng để thanh toán một số sản phẩm bán ra. Mỗi dòng hoá đơn cho biết tổng giá của mỗi sản phẩm.Đây là một quan hệ có kích thước là 2, còn gọi là quan hệ nhị nguyên.



Người ta đưa ra khái niệm những vai trò khác nhau của cùng một thực thể để có thể biểu diễn mối quan hệ giữa thực thể này với chính nó. Vì loại quan hệ này ít dùng nên trong Giáo trình này chúng tôi không trình bày loại quan hệ đó.

5.2.4. Phân loại các quan hệ

Xét R là một tập quan hệ và E là một thực thể cấu thành của R, mỗi cặp (E,R) được biểu thị trên sơ đồ khái niệm dữ liệu bằng một đoạn thẳng. Với thực thể E, ta có thể xác định được:

- X là số tối thiểu các thể hiện tương ứng với E mà R có thể có trong thực tế.

Giá trị của X như vậy chỉ có thể bằng 0 hay 1.

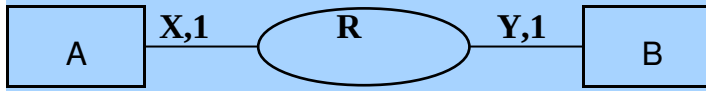
- Y là số tối đa các thể hiện tương ứng với E mà R có thể có trong thực tế.

Giá trị của Y có thể bằng 1 hay một số nguyên N > 1 .

Cặp số (X, Y) được định nghĩa là bản số của đoạn thẳng (E,R) và có thể lấy các giá trị sau: (0,1), (1,1), (0,N) hay (1,N), với N > 1.

Đối với loại quan hệ nhị nguyên R liên kết giữa hai thực thể A và B, ta phân thành ba loại quan hệ cơ bản sau:

- Quan hệ 1-1 (một - một): mỗi thể hiện của thực thể A được kết hợp với 0 hay 1 thể hiện của B và ngược lại.

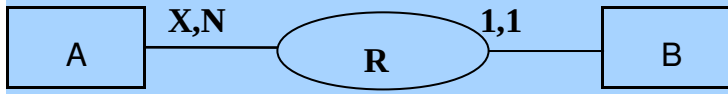


X và Y có thể lấy các giá trị 0 và 1

Ví dụ:

Mỗi độc giả ở một thời điểm chỉ được đọc một quyển sách.

- Quan hệ 1 - N (một - nhiều): Mỗi thể hiện của thực thể A được kết hợp với 0,1 hay nhiều thể hiện của B và mỗi thể hiện của B được kết hợp với một thể hiện duy nhất của A. Đây là một loại quan hệ thông dụng và đơn giản nhất.

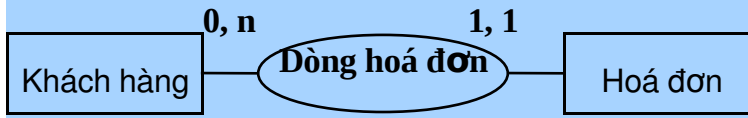


X có thể lấy các giá trị 0 và 1

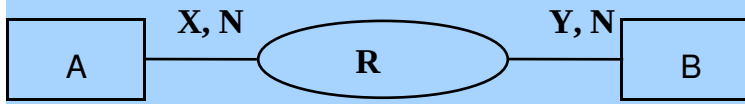
Ví dụ:

Một khách hàng có thể có nhiều hoá đơn

Một hoá đơn chỉ có thể mang tên một khách hàng.



- Quan hệ N - P (nhiều - nhiều): Mỗi thể hiện của một thực thể A được kết hợp với 0, 1 hay nhiều thể hiện của B và ngược lại, mỗi thể hiện của B được kết hợp 0, 1 hay nhiều thể hiện của A



X và Y có thể lấy các giá trị 0 hoặc 1

Ví dụ:

Một hoá đơn dùng để thanh toán một hay nhiều sản phẩm

Một sản phẩm có thể xuất hiện trong 0, 1 hay nhiều hoá đơn.

Thông thường quan hệ N - P chứa các thuộc tính. Chúng ta biến đổi loại quan hệ này thành thực thể và thực thể này cần được nhận dạng bởi một khoá chính.

5.3. Các dạng chuẩn trong các hệ QTCSDL hiện có

Trong mục này chúng tôi trình bày một số ý nghĩa của phụ thuộc hàm và mối liên hệ của nó với việc chuẩn hoá trong thực tiễn

5.3.1. Một số phụ thuộc hàm đặc biệt

Trên cơ sở của định nghĩa phụ thuộc hàm đã trình bày ở chương 2 chúng ta có thể thấy:

- Nếu B phụ thuộc hàm vào A (A→B), thì với mỗi giá trị của A tương ứng với một giá trị duy nhất của B. Hay nói cách khác, tồn tại một hàm (ánh xạ) từ tập hợp những giá trị của A đến tập hợp những giá trị của B.

Ví dụ: Trong một hoá đơn bao gồm các thuộc tính 'số hoá đơn', 'tên khách hàng', 'mã sản phẩm', 'tổng giá trị sản phẩm'....

Ta thấy 'Số hoá đơn' →'Tên khách hàng'

'Số hoá đơn', 'Mã sản phẩm' →'Tổng giá trị sản phẩm'

Chúng ta có thể mở rộng khái niệm phụ thuộc hàm khi cho phép A hoặc B là một thực thể hoặc là một quan hệ.

Ví dụ: Ta có hai thực thể 'Hoá đơn' và 'Khách hàng'

Khi đó: Thực thể 'Hoá đơn' →'Thuộc tính 'Tên khách hàng'

Thực thể 'Hoá đơn' →thực thể 'Khách hàng'

Thuộc tính 'Số hoá đơn' →thực thể 'Khách hàng'

Trong chương 3 ta trình bày phụ thuộc hàm hoàn toàn và phụ thuộc hàm trực tiếp. Hai loại phụ thuộc hàm này đóng vai trò quan trong các dạng chuẩn.

Các dạng chuẩn được đề ra với mục đích để đảm bảo tính nhất quán và tránh việc trùng lặp các thông tin. Trong mục này chúng ta sẽ quay trở lại với các dạng chuẩn. Các dạng chuẩn này có những biến đổi điều kiện ràng buộc đơn giản hơn so với các dạng chuẩn đã trình bày trong chương 3**.**

5.3.2. Dạng chuẩn 1

Chúng ta nói rằng một thực thể hay quan hệ ở dạng chuẩn 1 nếu tất cả giá trị các thuộc tính của nó là sơ cấp. Điều kiện ràng buộc giống như 1NF của chương 3. Định nghĩa của dạng chuẩn 1 mang tính mô tả. Thông thường giá trị các thuộc tính là các dãy kí tự hoặc là các số như trong FOXPRO, khi đó chúng ta cho các giá trị này là sơ cấp

Để minh họa ta đưa ra thực thể sau đã trình bày trong mục 4.3.

Bán hàng

Ngày tháng	Mã hàng	Tên hàng	Đơn giá	Số lượng	Tổng	Thanh toán
210397	M1	Radio	1 000	1	10 000	6 000

210397	M3	TV	4 000	2	10 000	6 000
210397	M6	Xe đạp	1 000	1	10 000	6 000
220397	M2	Máy giặt	3 000	2	6 000	2 000
230397	M1	Radio	1 000	3	15 000	11 000
230397	M4	Video	5 000	2	15 000	11 000
230397	M9	Máy ảnh	2 000	1	15 000	11 000

Chúng ta chọn khoá chính (nó là một khoá tối tiểu) cho thực thể bán hàng là tập {Ngày tháng, Mã hàng}

5.3.3. Dạng chuẩn 2

Một thực thể hay quan hệ là 1NF được xem là dạng chuẩn 2 nếu tất cả các phụ thuộc hàm giữa khoá chính và các thuộc tính khác của nó đều là hoàn toàn .

Chú ý rằng định nghĩa dạng chuẩn 2 trong chương 2 chặt hơn vì điều kiện phụ thuộc hoàn toàn liên quan đến mọi khoá tối tiểu, chứ không chỉ liên

quan đến một khoá tối tiểu được chọn làm khoá chính.

Trong ví dụ trên, thực thể Bán hàng đã là 1NF, ta nhận thấy đối với khoá chính {số phiếu, mã vật tư} các thuộc tính Tổng và Thanh toán phụ thuộc hàm vào thuộc tính Ngày tháng, các thuộc tính Tên hàng, Đơn giá phụ thuộc hàm vào thuộc tính Mã hàng. Ngày tháng, Mã hàng là hai thuộc tính của khóa chính. Do đó dẫn đến trùng lặp dữ liệu. Thực thể Bán hàng không là 2NF. Để thoả dạng chuẩn 2NF, ta phải tách nó thành 3 thực thể riêng biệt:

hàng hoá

Mã hàng	Tên hàng	Đơn giá
M1	Radio	1 000
M2	Máy giặt	3 000
M3	TV	4 000
M4	Video	5 000
M6	Xe đạp	1 000
M9	Máy ảnh	2 000

Ta chọn khoá chính của thực thể hàng hoá là Mã hàng

khối lượng

Ngày tháng	Mã hàng	Số lượng
210397	M1	1
210397	M3	2
210397	M6	1
220397	M2	2
230397	M1	3
230397	M4	2
230397	M9	1

Ta chọn khoá chính (nó là khoá tối tiểu) cho thực thể Khối lượng là Ngày tháng, Mã hàng

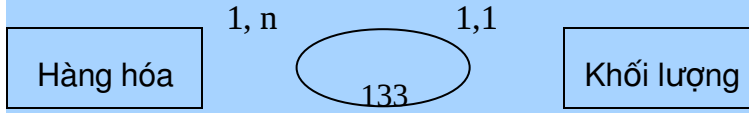
Doanh số

Ngày tháng	Tổng	Thanh toán
210397	10000	6000
220397	6000	2000
230397	15000	11000

Khoá chính là Ngày tháng

Có thể thấy thực thể Khối lượng →thực thể Hàng hoá

Ta có biểu diễn sau:



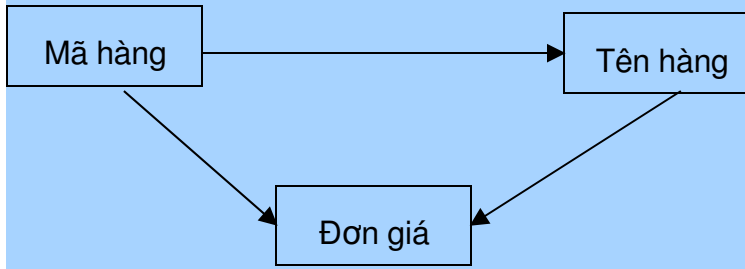
R

5.3.4. Dạng chuẩn 3 (3NF)

Một thực thể (hay quan hệ) đã là 2NF được xem là dạng chuẩn 3NF nếu tất cả các phụ thuộc hàm giữa khoá chính và các thuộc tính khác của nó đều là trực tiếp.

Hay nói cách khác, mọi thuộc tính không nằm trong khoá chính đều không phụ thuộc hàm vào một thuộc tính không phải là khóa chính. Ta có thể rút ra nhận xét: Một thực thể có nhiều khoá nhận dạng không thể thoả mãn dạng chuẩn 3NF. Mặt khác định nghĩa 3NF trong chương 2 chặt hơn vì điều kiện phụ thuộc hoàn toàn và phụ thuộc trực tiếp liên quan đến mọi khoá tối tiểu, chứ không chỉ liên quan đến một khoá tối tiểu được chọn làm khoá chính.

Trong thực thể Hàng hoá là 2NF ở trên, ta thấy trên đồ thị của các phụ thuộc hàm có hai con đường để đi từ ‘Mã hàng’ đến ‘Đơn giá’ hoặc đi qua thuộc tính ‘Tên hàng’



Điều này chứng tỏ thực thể chưa là 3NF, dẫn đến trùng lặp đơn giá của tên hàng. Để là dạng 3NF, ta tách nó thành hai thực thể riêng biệt:

Hàng

Mã hàng	Tên hàng
M1	Radio
M2	Máy giặt
M3	TV
M4	Video
M6	Xe đạp
M9	Máy ảnh

Khoá chính là Mã hàng

Mặt hàng

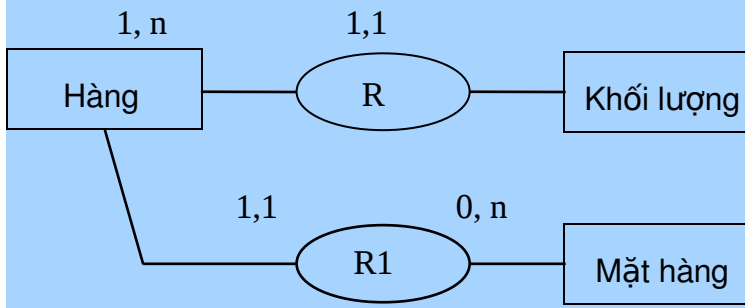
Tên hàng	Đơn giá
Radio	1000
Máy giặt	3000
TV	4000

Video	5000
Xe đạp	1000
Máy ảnh	2000

Khoá chính là Tên hàng

Có thể thấy thực thể Hàng → thực thể Mặt hàng

Tổng hợp với phần trên, ta có sơ đồ sau:



5.3.5. Dạng chuẩn của Boyce - Codd (BCNF)

Dạng chuẩn 3NF cho phép một thuộc tính thành phần của khoá chính phụ thuộc hàm vào một thuộc tính không phải là khoá

Ví dụ :

Lớp	Môn	Thầy
12	Toán	A
11	Toán	D
10	Toán	A
12	Địa	C
11	Địa	C
10	Địa	D

Thực thể này thoả dạng 3NF. Khoá chính của nó gồm các thuộc tính 'Lớp' và 'Môn'.

Nhưng do qui tắc 'Mỗi thầy chỉ dạy một môn', ta thấy có sự phụ thuộc hàm của Môn (Là một thành phần của khoá chính) vào Thầy (Là một thuộc tính bình thường):

'Thầy' $\rightarrow$'Môn'

Ta nói rằng thực thể thoả mãn dạng chuẩn Boyce-Codd (BCNF) khi tất cả các phụ thuộc hàm của nó đều thuộc dạng K $\rightarrow$ a, trong đó K là khoá chính và a là một thuộc tính bất kỳ.

Để thoả dạng BCNF, ta có thể tách thực thể trên thành hai thực thể riêng biệt như sau:

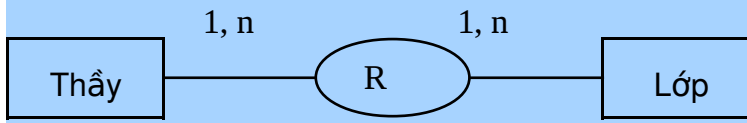
Thực thể 'Lớp':

Lớp
12
11
10

Thực thể 'Thầy':

Thầy	Môn
A	Toán
B	Toán
C	Sử Địa
D	Sử Địa

Chúng ta có biểu diễn sau:



5.3.6. Nhận xét về việc chuẩn hoá

Khi không có yêu cầu gì đặc biệt, người ta thường tìm cách chuẩn hoá mô hình dữ liệu nhằm tăng hiệu hiệu năng và giảm sơ xuất trong các giai đoạn phát triển hệ thông tin về sau.

Tuy vậy, việc chuẩn hoá không phải lúc nào cũng đạt đến mức tối đa. Thông thường chúng ta chuẩn hoá đến dạng chuẩn 2NF và 3NF.

Chương 6

Một số công đoạn xây dựng các dự án thiết kế tổng thể các hệ thống cơ sở dữ liệu hiện nay

6.1. Mô tả chung

Trong dự án thiết kế tổng thể người ta thường trình bày một số vấn đề cơ bản sau:

- Hiện trạng và tiềm lực CNTT của cơ quan chủ trì dự án.

- Hiện trạng về việc thu thập, lưu trữ và xử lí các dữ liệu liên quan hệ cơ sở dữ liệu cần xây dựng,

- Ước đoán khối lượng lưu trữ và trao đổi thông tin trong hệ cơ sở dữ liệu,

- Phân tích thiết kế hệ CSDL,

- Thiết kế hạ tầng kĩ thuật,

- Chuẩn hoá, bảo mật và an toàn thông tin,

- Tính pháp lí về quyền và nghĩa vụ trong việc thu thập, cập nhật, khai thác và bảo vệ thông tin trong hệ CSDL,

- Nội dung, đối tượng và kế hoạch triển khai công tác đào tạo,

- Tổng hợp và cân đối kinh phí

- Các biện pháp tổ chức thực hiện .

Một trong các phần quan trọng nhất của dự án là khâu phân tích thiết kế.

Thông thường để thực hiện việc phân tích thiết kế khi xây dựng dự án người ta sử dụng một số phương pháp phân tích thiết kế, ví dụ như phương pháp phân tích thiết kế có cấu trúc (Structured Analysis and Design), phương pháp GALASCI, phương pháp phân tích MCX, phương pháp phân tích MERISE (Methode pour Rassembler les Idees Sans Effort)...

Trong các phương pháp phân tích thiết kế trên MERISE có đặc tính là khi phân tích nó tách rời xử lí với dữ liệu, tổ chức theo nhiều mức, đi từ phân tích tổng thể đến chi tiết một cách tuần tự theo chiều tăng dần của mức độ phức tạp. MERISE được nhiều người xử dụng và đặc biệt tốt cho việc phân tích thiết kế cho các hệ thông tin lớn. Tại Việt Nam một số cơ quan đã được trang bị công cụ phân mềm MEGA tuân theo các chuẩn của phương pháp phân tích MERISE. Cho đến nay nhiều dự án "Thiết kế

tổng thể hệ thống cơ sở dữ liệu" đã được xây dựng trên cơ sở phương pháp phân tích MERISE.

Dựa trên phương pháp MERISE, khi phân tích thiết kế tổng thể, chúng ta tiến hành các công đoạn sau :

① Mô tả qui trình nghiệp vụ bằng lời:

Trong công đoạn này chúng ta mô tả đầy đủ, mạch lạc bằng lời qui trình nghiệp vụ của bài toán cần thiết kế. Công đoạn này thường phân biệt những yếu tố nghiệp vụ hay thay đổi với những yếu tố nghiệp vụ tương đối bất biến trong khoảng thời gian dài. Những yếu tố bất biến này thường được tập trung mô tả kĩ hơn. Trong công đoạn mô tả bằng lời này, chúng ta không chỉ mô tả những yếu tố nghiệp vụ hiện có mà còn mô tả các kiến nghị sửa đổi, các dự báo tương lai phù hợp với quá trình phát triển của hệ thống thông tin. Công đoạn này phân rã mỗi lĩnh vực nghiệp vụ thành các chức năng nghiệp vụ, và mỗi chức năng nghiệp vụ được phân rã thành các thao tác xử lí. Toàn bộ các yếu tố này đều được mô tả rõ ràng.

② Xây dựng các mô hình

Sau khi mô tả các qui trình nghiệp vụ một cách rõ ràng và mạch lạc, chúng ta tiến hành xây dựng các mô hình. Các mô hình này được phân chia theo 4 mức độ khái quát khác nhau. Đó là :

- Mức khái niệm: Mức này mô tả sự hoạt động của đơn vị theo một cấu trúc khái quát nhất. Trong mức này các chức năng hoạt động được mô tả độc lập với những bộ phận, vị trí hay nhân viên thực hiện chúng.

- Mức tổ chức: Mức này thể hiện các mục tiêu đã được khái niệm hóa lên thực tế của đơn vị trong đó có tính đến những điều kiện ràng buộc về mặt tổ chức.

- Mức lôgic: Mức này qui định các công cụ tin học mà các công cụ này được người dùng trong các thao tác xử lí.

- Mức vật lí: Mức này liên quan chặt chẽ với các trang thiết bị tin học cụ thể.

Trong mỗi mức chúng ta có 3 mô hình. Đó là mô hình trao đổi thông tin, mô hình xử lí và mô hình dữ liệu.

Cụ thể chúng ta có:

*Mô hình khái niệm trao đổi (MCC modele conceptuel de communication):

MCC phân rã lĩnh vực nghiệp vụ thành các chức năng nghiệp vụ. MCC mô tả sự trao đổi thông tin giữa các chức năng nghiệp vụ nằm trong bài toán và sự trao đổi thông tin giữa các lĩnh vực nghiệp vụ với các lĩnh vực nghiệp vụ khác và các đối tượng bên ngoài.

MCC cho ta một cái nhìn tổng thể về một lĩnh vực nghiệp vụ và các chức năng nghiệp vụ của nó cũng như các nhu cầu thông tin của nó.

*Mô hình khái niệm xử lí (MCT modele conceptuel de traitements):

MCT dùng để mô tả một lĩnh vực, qui trình, chức năng, thao tác.

MCT phân rã một chức năng nghiệp vụ thành các thao tác xử lí . Mỗi thao tác xử lí có thể được xem như một phép biến đổi thông tin. Một thao tác có thể có điều kiện khởi động là các sự kiện, các thông báo mà khi xuất hiện chúng thao tác bắt đầu được thực hiện . Trong quá trình thực hiện, thao tác cần phải truy nhập đến các thông tin đã được lưu giữ , biến đổi chúng và cập nhật lại theo một số qui luật tính toán và ràng buộc nhất định.

* Mô hình khái niệm dữ liệu (MCD modele conceptuel de donneees):

Việc mô tả dữ liệu trong mô hình MCD thông qua ngôn ngữ " Thực thể / Quan hệ " cùng với các thuộc tính của các thực thể và các quan hệ . Trên mô hình này, khóa chính, các thuộc tính chính được khai báo và lưu trữ trong hệ thống.

MCD được xây dựng cho một lĩnh vực nghiệp vụ nhằm xác định đầy đủ những dữ liệu đòi hỏi khi thực hiện các chức năng , trong đó đặc biệt là những dữ liệu cần thiết cho việc trao đổi .

* Mô hình tổ chức trao đổi (MOC):

MOC mô tả một lĩnh vực nghiệp vụ, đơn vị tổ chức.

Mô hình này mô tả các vị trí làm việc và việc luân chuyển thông tin trong đơn vị.

* Mô hình tổ chức xử lí (MOT):

MOT là mô hình tổ chức để thực hiện các thao tác của một chức năng nghiệp vụ đã được mô tả trong MCT. MOT thể hiện qui trình làm việc , trong đó nhấn mạnh tính tuần tự của các thao tác và nêu rõ những ràng buộc về thời điểm bắt đầu xử lí hay truyền thông tin. Một thao tác trong MCT có thể ứng vối nhiều thao tác trong MOT và ngược lại một thao

tác trong MOT cũng có thể ứng với nhiều thao tác trong MOT tuỳ theo các hoàn cảnh cụ thể .

* Mô hình tổ chức dữ liệu (MOD):

MOD mô tả dữ liệu cần ghi nhớ trong từng địa điểm, cho từng vị trí thực hiện.

Trong khi MCD chỉ định nghĩa các khái niệm dữ liệu, thì MOC cụ thể hóa những điều kiện có thể xảy ra để một thực thể thuộc vào mô hình.

* Mô hình lôgic trao đổi (MLC):

MLC xác định sự trao đổi giữa người với người (thông qua các mẫu biểu), giữa người với máy (thông qua giao diện) và giữa các phần mềm với nhau.

* Mô hình lôgic xử lí (MLT): Thường để mô tả công cụ tin học.

* Mô hình lôgic dữ liệu (MLD):

Nhờ MLD, chúng ta có thể chuyển các MOD sang dạng quen thuộc cho các chuyên gia tin học. Thông thường, chúng ta chuyển từ ngôn ngữ thực thể - quan hệ sang dạng biểu báo.

Các mô hình vật lí trao đổi (MPC), mô hình vật lí xử lí (MPT), mô hình vật lí dữ liệu (MPD) gắn liền với các trang thiết bị tin học cụ thể.

Các mức và mô hình của MERISE có thể tóm tắt như sau:

Mức	Trao đổi	Chức năng	Dữ liệu
Khái niệm	MCC	MCT	MCD
Tổ chức	MOC	MOT	MOD
Lôgic	MLC	MLT	MLD
Vật lí	MPC	MPT	MPD

Quan trọng nhất trong các mô hình trên là các mô hình liên quan đến dữ liệu vì nó làm nền tảng cho việc xây dựng các mô hình khác.

Trong phạm vi Giáo trình này chúng tôi trình bày việc xây dựng mô hình khái niệm dữ liệu. Đó là mô hình dữ liệu thường có trong các dự án thiết kế tổng thể các hệ thống thông tin.

Quá trình xây dựng mô hình khái niệm dữ liệu có thể được chia thành các giai đoạn sau đây:

A. Khảo sát thực tế

- Thu thập và trình bày thông tin.

B. Thiết kế mô hình dữ liệu

- Kiểm kê các dữ liệu.

- Xác định các phụ thuộc hàm.

- Xây dựng mô hình khái niệm:

- Xác định tập hợp các khoá chính.

- Nhận diện các thực thể.

- Nhận diện các quan hệ.

- Phân bổ các thuộc tính còn lại.

- Vẽ sơ đồ khái niệm dữ liệu.

C. Kiểm soát và chuẩn hoá mô hình

6.2. Khảo sát thực tế

6.2. Khảo sát thực tế

Mục tiêu của giai đoạn này là qua quá trình quan sát, phỏng vấn, tìm hiểu và phân tích, chúng ta mô tả đầy đủ hiện trạng, các bài toán nghiệp vụ và các nhu cầu của người sử dụng mà hệ thống thông tin cần phải thoả mãn. Do đó, nó không chỉ giới hạn trong việc xây dựng mô hình dữ liệu mà còn là nguồn gốc các thông tin cần thiết cho việc xây dựng mô hình xử lí.

Để đạt mục tiêu này, ta cần thu thập và trình bày tất cả những thông tin dù thuộc phương diện dữ liệu

hay chức năng có thể hữu ích cho việc thiết kế hệ thông tin về sau. Các thông tin này cần được quan sát dưới dạng: tĩnh (dữ liệu sơ cấp, tài liệu, quảng cáo, đơn vị, vị trí làm việc..), dạng động (luồng luân chuyển các thông tin, tài liệu, chu kì, thời lượng), và dạng biến đổi của chúng (thủ tục, qui tắc quản lí, công thức tính toán,..).

Các thông tin thu thập được phải đầy đủ và chính xác vì chúng là nền tảng của hệ thông tin tương lai. Nhưng cũng không nên đi quá sâu vào chi tiết và phải biết gạn bỏ những thông tin không cần thiết, để không làm chệch hướng và gây khó khăn nặng nề cho việc phân tích thiết kế.

Công việc khảo sát không chỉ tập trung hoàn toàn vào giai đoạn đầu của quá trình phân tích thiết kế, mà có thể chạy dài trong suốt quá trình này để thu thập thêm thông tin, đào sâu vấn đề hay kiểm chứng một giả thiết cùng với người sử dụng khi hệ thông tin cần xây dựng quá lớn và phức tạp, ta nên chia nó thành nhiều tiểu hệ. Mỗi tiểu hệ có thể được khảo sát, phân tích hay thiết kế độc lập với nhau, trước khi được tập hợp lại .

Để chia một hệ thông tin thành nhiều tiểu hệ, người ta thường sử dụng một trong hai phương pháp tiếp cận sau đây:

- Phương pháp 1: Các tiểu hệ độc lập được định ra, dựa trên cơ sở những bài toán , chức năng nghiệp vụ chủ yếu của tổ chức. Đôi khi dựa trên một kế hoạch thực hiện theo thứ tự ưu tiên hay để thoả những yêu cầu về thời gian.

- Phương pháp 2: Một cuộc khảo sát tổng quát sơ khởi sẽ cho phép nhận diện những tiểu hệ tương đối độc lập với nhau.

Tiếp theo đó chúng ta tiến hành thu thập thông tin về hệ thống cần xây dựng

Công việc này chủ yếu là tham khảo tài liệu và tiếp xúc với những người sử dụng, đòi hỏi những khả năng như óc quan sát, kinh nghiệm, tài giao tiếp và ứng biến... Các phương pháp gò bó, cứng nhắc sẽ chẳng đem lại kết quả mong muốn. Do đó, phần này chỉ liệt kê và phân loại các thông tin có thể gặp được trong quá trình khảo sát, xem như để trợ giúp trí nhớ.

Trong quá trình thu thập thông tin thông thường ta tập hợp các dữ liệu sau

- Dữ liệu về hệ thông tin hiện tại bao gồm các dữ liệu liên quan trực tiếp đến hệ thông tin theo mọi dạng (tĩnh, động, biến đổi) và thông tin về môi trường làm việc

- Dữ liệu về hệ thống tương lai bao gồm các nhu cầu, mong muốn cho hệ thông tin của ta. Trong các dữ liệu này ta cần phân biệt những vấn đề đã được nhận thức và phát biểu rõ ràng, những vấn đề được nhận thức nhưng chưa được công nhận, những vấn đề còn chưa nhận thức và còn đang tranh luận.

Với mỗi loại dữ liệu cần thu thập đã nêu trên, nếu cần ta còn có thể tìm hiểu thêm về một số khía cạnh khác như: số lượng, độ chính xác cần có, ai là người có trách nhiệm...

Nhận xét:

- Nếu có thể, các cuộc phỏng vấn phải được tiến hành tuần tự theo cấu trúc phân cấp của tổ chức theo từng bộ phận, lĩnh vực, chức năng hay đi từ cấp lãnh đạo qua cấp quản lí đến những người thừa hành.

- Phải luôn nhớ sao chụp mẫu các hồ sơ, tài liệu, để có được cấu trúc chính xác các thông tin làm căn bản cho việc xây dựng mô hình dữ liệu sau này.

- Cần thiết nhất là luôn phân biệt những thông tin nói về hệ thông tin đang xây dựng với những thông tin thuộc về hệ này

Các thông tin thu thập, sau khi được tổng hợp sẽ được trình bày dưới hai dạng:

a. Mô tả các bài toán nghiệp vụ, các chức năng và tổ chức của cơ quan, các nhu cầu và mong muốn của người sử dụng một cách đầy đủ, nhưng ngắn gọn và mạch lạc, bằng một ngôn ngữ thông thường, gần gũi với mọi người.

b. Minh hoạ và hệ thống hoá phần trình bày trên bằng một ngôn ngữ hình thức, thường là dưới dạng phiếu mô tả, danh sách và đồ hoạ.

Trên thực tế có nhiều phương pháp trình bày thông dụng khác nhau,

6.3. Thiết kế mô hình dữ liệu

Thiết kế một mô hình khái niệm dữ liệu là liệt kê và định nghĩa chính xác những dữ liệu có liên quan đến các chức năng, hoạt động của một tổ chức. Sau đó ta nhóm chúng lại thành thực thể và quan hệ giữa các thực thể, rồi dùng một số qui ước đã định trước để trình bày dưới dạng mô hình khái niệm.

6.3.1. Kiểm kê dữ liệu

Danh sách này chủ yếu được rút từ những thông tin thu thập được trong giai đoạn khảo sát ban đầu; tài liệu thu thập được; nhu cầu, giải thích của người sử dụng.

Có thể phân biệt hai kiểu dữ liệu:

- Loại dữ liệu xuất hiện trực tiếp trên các tài liệu, màn hình, tập tin thu thập được.

- Loại dữ liệu không xuất hiện nhưng cần thiết để chứa kết quả trung gian, các thông tin đang chờ được xử lý, hay để tính toán các dữ liệu thuộc loại thứ nhất.

Một công cụ thông dụng, hữu ích cho giai đoạn này là bảng, dùng để phân tích các tài liệu thu thập và liệt kê ra danh sách các dữ liệu. Trong bảng này, ta trình bày mỗi cột là một tài liệu và mỗi hàng là một loại dữ liệu. Tại mỗi ô giao điểm, ta đánh dấu khi loại dữ liệu có xuất hiện trên tài liệu. Nên dùng hai loại dấu khác nhau để phân biệt loại dữ liệu trực tiếp với loại được tính toán thành.

Khi xây dựng bảng này, ta nên bắt đầu bằng những tài liệu cơ bản, quan trọng nhất và chỉ cần trình bày một loại tài liệu khi nó cho phép nhận dạng ít nhất một loại dữ liệu mới.

Ví dụ: Trong một công ty có:

- Kho hàng làm nhiệm vụ lưu giữ và quản lí hàng hoá và khi cần thì đề nghị mua thêm hàng

- Phòng đặt hàng có nhiệm vụ làm đơn đặt hàng gửi cho các công ty cung cấp

- Phòng kế toán lưu bản sao đơn đặt hàng để kiểm tra hàng.

Ta dùng 1 để đánh dấu dữ liệu trực tiếp và 2 cho dữ liệu được tính toán.

Tài liệu Loại dữ liệu	Phiếu đề nghị đặt hàng	Đơn đặt hàng	Phiếu giao hàng	Tệp tin về nhà cung cấp
Tên kho	1			
Địa chỉ kho	1			
Ngày đề nghị đặt hàng	1			
Số lượng hàng cần đặt	1			
Mã số sản phẩm	1	1	1	
Nhãn hiệu sản phẩm	1	1	1	
Mã số của công ty cung cấp		1	1	1
Tên công ty cung cấp		1	1	1

Địa chỉ công ty cung cấp		1	1	1
Đơn giá sản phẩm		1	1	1
Ngày đặt hàng		1		
Số lượng hàng đặt mua		1		
Tổng giá đơn đặt hàng		2		
Ngày giao hàng			1	
Số lượng hàng được giao			1	
Tổng giá hàng được giao			2	

Từ danh sách này, người ta cần kiểm tra bằng một số công tác thanh lọc như sau:

- Bỏ bớt các dữ liệu đồng nghĩa nhưng khác tên, chỉ giữ lại một.

Ví dụ: Mã số sản phẩm = danh mục mặt hàng

- Phân biệt các dữ liệu cùng tên nhưng khác nghĩa thành nhiều loại dữ liệu khác nhau.

Ví dụ: Giá bán của một cửa hiệu khác với giá bán của công ty sản xuất.

- Nhập chung các loại dữ liệu luôn luôn xuất hiện đồng thời với nhau trên mọi loại tài liệu thành một loại dữ liệu sơ cấp.

Ví dụ: số nhà và tên đường; ngày, tháng và năm sinh.

- Loại bỏ những loại dữ liệu có thể được xác định một cách duy nhất từ các loại dữ liệu khác, hoặc bằng công thức tính toán, hoặc do các qui luật của tổ chức

Ví dụ: Tổng giá đơn đặt hàng = Số lượng hàng* Đơn giá

Giả sử do qui luật của tổ chức, mọi đề nghị mua hàng phải được giải quyết nội trong ngày, ta suy ra:

Ngày đề nghị mua hàng = Ngày đặt hàng.

6.3.2. Định các phụ thuộc hàm giữa các dữ liệu

Từ danh sách dữ liệu đã được thanh lọc của hệ thông tin đạt được qua giai đoạn trên, ta phải định ra tất cả các phụ thuộc hàm có giữa chúng .

Cụ thể, ta phải tự đặt câu hỏi: Mỗi giá trị của một loại dữ liệu A có tương ứng với một giá trị duy nhất của loại dữ liệu B hay không?

Nếu câu trả lời là "Có" thì B phụ thuộc hàm vào A: $A \rightarrow B$

Ngoài các phụ thuộc hàm có vế trái A là một loại dữ liệu sơ cấp (gọi là phụ thuộc hàm sơ cấp) tương đối dễ xác định, ta còn phải nhận diện cả các phụ thuộc hàm trong đó vế trái A là một tập hợp của nhiều loại dữ liệu (gọi là phụ thuộc hàm đa phần). Trong trường hợp này, ta nên tự đặt câu hỏi: Cần ấn định giá trị của những loại dữ liệu nào để có thể suy ra một giá trị duy nhất của loại dữ liệu B? Các phụ thuộc hàm sẽ được trình bày dưới dạng một bảng như sau:

Loại dữ liệu Phụ thuộc hàm Phụ thuộc hàm

sơ cấp đa phần

Mã số của công ty cung cấp .....

Tên công ty cung cấp.....

Địa chỉ công ty cung cấp.....

Tên kho .....

Địa chỉ kho.....

Ngày đề nghi đặt hàng .....

Ngày đặt hàng .....

Ngày giao hàng .....

Đơn giá sản phẩm .....

Mã số sản phẩm .....

Nhãn hiệu sản phẩm.....

Số lượng hàng cần đặt .....

Số lượng hàng đặt mua .....

Số lượng hàng được giao .....

6.3.3. Xây dựng mô hình dữ liệu

Giai đoạn này bao gồm 5 động tác:

- Định tập hợp các khoá chính
- Nhận diện các thực thể
- Nhận diện các quan hệ

- Phân bổ các thuộc tính còn lại

- Vẽ sơ đồ khái niệm dữ liệu

6.3.3.1. Xác định tập hợp các khoá chính

Tập hợp K của những khoá chính là tập hợp tất cả những loại dữ liệu đóng vai trò nguồn (thuộc vế trái) trong ít nhất một phụ thuộc hàm.

Trong ví dụ trên ta có : K =

{'Mã số công ty cung cấp'

'Tên kho'

'Ngày đề nghi đặt hàng'

'Ngày đặt hàng'

'Ngày giao hàng'

'Mã số sản phẩm'}

6.3.3.2 Nhận diện các thực thể

Mỗi phần tử của tập hợp K sẽ là khoá chính của một thực thể

Trong ví dụ trên, ta nhận ra được 4 thực thể:

'Nhà cung cấp' với khoá chính là 'Mã số của công ty cung cấp'

'Kho' với khoá chính là 'Tên kho'

'Lịch' với khoá chính là 'Ngày'

(các thực thể 'Lịch đặt hàng', 'Lịch đề nghi mua hàng' và 'Lịch giao hàng' hoàn toàn tương đương với nhau nên được gộp thành một thực thể duy nhất là 'Lịch')

'Sản phẩm' với khoá chính là 'Mã số sản phẩm'

6.3.3.3 Nhận diện các quan hệ

Có 2 trường hợp :

a. Nếu gốc của một phụ thuộc hàm bao gồm ít nhất 2 phần tử thuộc tập hợp K thì nó tương ứng với một quan hệ N - P giữa các thực thể có khoá chính là các phần tử này.

Trong ví dụ trên, ta nhận ra được 4 quan hệ :

'Đơn giá của nhà cung cấp'

'Dòng đề nghị mua hàng'

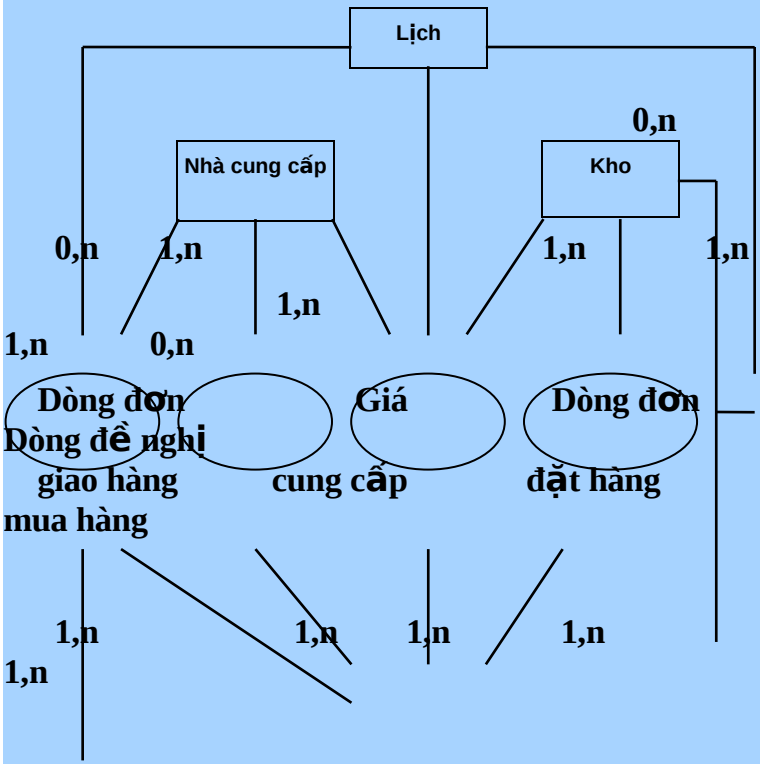
'Dòng đơn đặt hàng'

'Dòng phiếu giao hàng'

b. Một phụ thuộc hàm giữa 2 phần tử của tập hợp K xác định một quan hệ nhị nguyên kiểu 1-N giữa hai thực thể có khoá chính là các phần tử này.

6.3.3.5. Vẽ sơ đồ khái niệm dữ liệu

Từ các thực thể và quan hệ đã nhận diện, ta có thể vẽ lên một sơ đồ khái niệm dữ liệu sau:


Lịch
Nhà cung cấp
Kho
0,n
0,n
1,n
1,n
1,n
1,n
1,n
0,n
Dòng đơn giao hàng
Giá cung cấp
Dòng đơn đặt hàng
Dòng đề nghị mua hàng
1,n
1,n
1,n
1,n
1,n

6.3.4 Xây dựng mô hình dữ liệu bằng trực giác

Phương pháp phân tích hệ thống nêu trên là một công cụ hữu hiệu và chuẩn xác để xây dựng phần lớn các loại mô hình dữ liệu. Nhưng nếu áp dụng hoàn toàn trong một hệ thông tin cỡ lớn sẽ đòi hỏi nhiều thời gian và công sức. Trong thực tế, các thiết kế viên kinh nghiệm, sau khi đã nhận thức được vấn đề qua khảo sát thường chọn cách xây dựng trực tiếp một mô hình sơ khởi rồi đi thẳng vào giai đoạn sau để kiểm soát và chuẩn hoá mô hình .

Phương pháp trực giác này có ưu điểm là ít tốn thời gian và đôi khi tạo ra những mô hình đơn giản và gần thực tế hơn. Nhưng ngược lại, nó chứa nhiều rủi ro hơn.

6.4. Kiểm soát và chuẩn hoá mô hình

Để đơn giản hoá và đồng thời đảm bảo tính nhất quán của mô hình dữ liệu, ta cần kiểm soát lại mô hình vừa xây dựng bằng một số qui tắc thực tiễn sau đây:

6.4.1 Chuẩn hoá mô hình

Chú ý việc chuẩn hoá toàn bộ mô hình dữ liệu thành các dạng BCNF không phải là bắt buộc. Tuy

vậy các dạng 1FN, 2FN, 3NF nên luôn được thực hiện.

6.4.2 Tạo thêm một thực thể

Việc tạo thêm một thực thể là cần thiết khi có ít nhất một quan hệ đang được xử lý liên quan tới nó.

Việc tạo thêm một thực thể là hợp lí khi:

a) Thuộc tính sẽ được chọn làm khoá chính của thực thể này là một loại dữ liệu thông dụng trong hoạt động của tổ chức đang khảo sát.

b) Ngoài khoá chính này, quan hệ còn có chứa những thuộc tính khác .

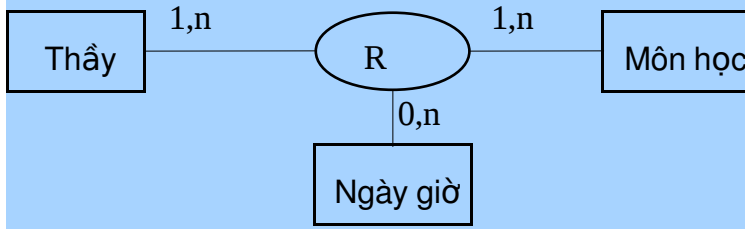
6.4.3. Biến một quan hệ thành thực thể

Một quan hệ có kích thước lớn hơn 3 nên được biến thành thực thể để đơn giản hoá.

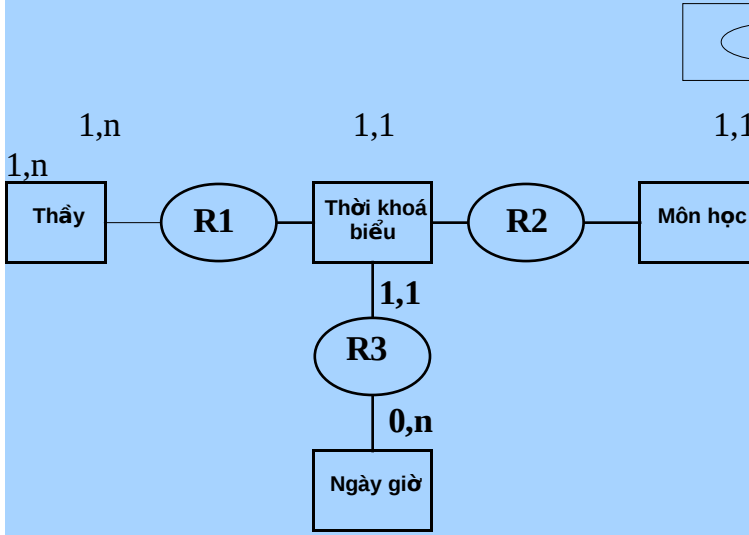
Có thể biến một quan hệ thành thực thể khi hội đủ các điều kiện sau:

- Quan hệ này có một khoá chính độc lập

- Quan hệ này tương ứng với một khái niệm quen thuộc, thông dụng trong hoạt động của tổ chức.



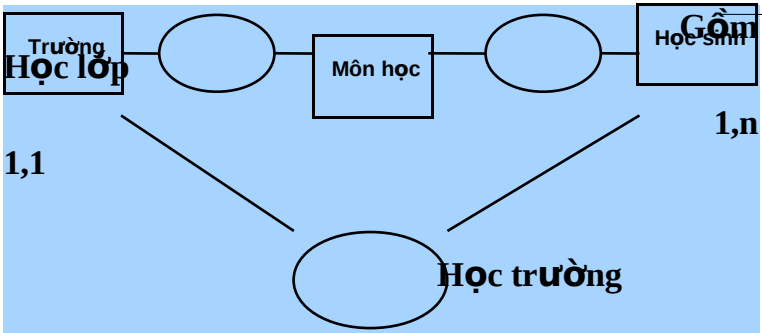
Quan hệ R được biến thành thực thể 'Thời khoá biểu':



6.4.4 Xoá một quan hệ

Một quan hệ 1 - N phải được loại bỏ khỏi mô hình dữ liệu nếu nó là tổng hợp của 2 hay nhiều quan hệ 1 - N khác.

1,n **1,1** **1,n**
1,1



Ta loại bỏ quan hệ Học trường

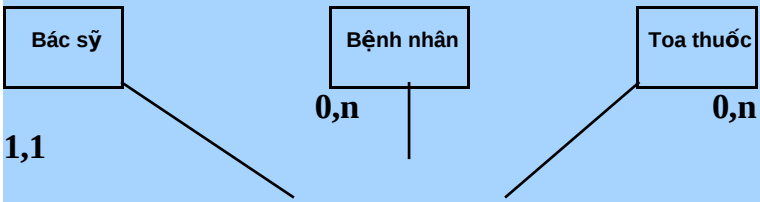
6.4.5 Phân tách một quan hệ phức tạp

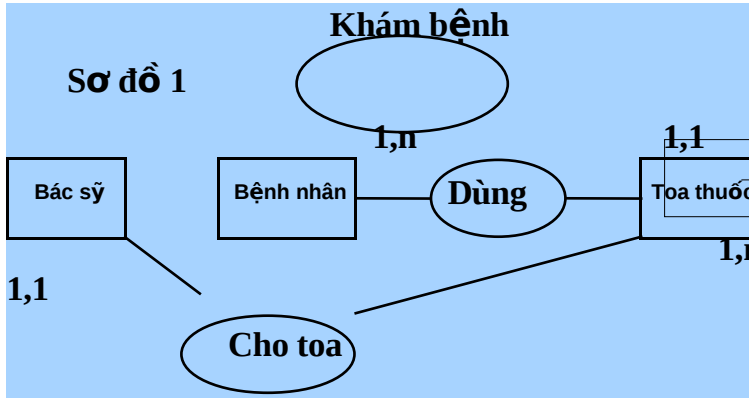
Xét một quan hệ có kích thước lớn hơn hoặc bằng 3. Quan hệ có thể được phân tách thành nhiều quan hệ khác với kích thước nhỏ hơn mà không mất thông tin nếu tồn tại ít nhất một phụ thuộc hàm giữa các thực thể cấu thành quan hệ.

6.4.5.1 Trường hợp phụ thuộc hàm ẩn

Trong trường hợp này, một trong các bản số của quan hệ bằng (1,1) hoặc (0,1). Điều này chứng tỏ sự tồn tại của một số phụ thuộc hàm ẩn.

Ví dụ:





Sơ đồ 1

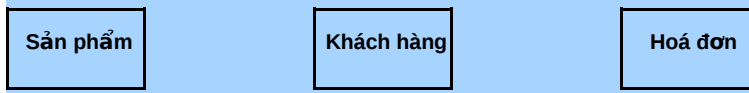
Sơ đồ 2

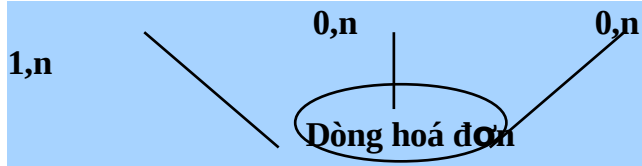
Chúng ta chọn sơ đồ 2

6.4.5.2. Trường hợp phụ thuộc hàm hiện

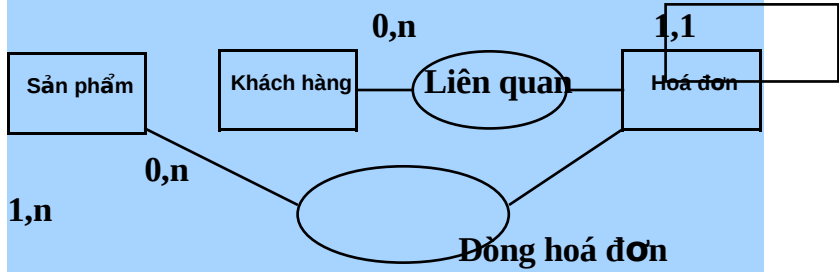
Cho một quan hệ R giữa 3 thực thể A, B, và C. Nếu tồn tại một hàm phụ thuộc A →B thì R có thể được phân thành quan hệ giữa A với B và giữa B với C.

Trong ví dụ sau đây, phụ thuộc hàm Hoá đơn → Khách hàng (mỗi hoá đơn chỉ liên quan đến một khách hàng duy nhất) cho phép ta đưa vào một quan hệ mới để diễn tả sự phụ thuộc này và đơn giản hoá mô hình.





Sơ đồ 1



Sơ đồ 2

Chúng ta chọn sơ đồ 2 đơn giản hơn.

Chương 7
Thuật toán và độ phức tạp

7.1. Khái niệm thuật toán

Nếu cho trước một bài toán, thì một cách giải bài toán được phân định ra thành một số hữu hạn bước, có kết thúc cuối cùng gọi là thuật toán.

Khi giải bài toán sẽ nẩy sinh ra các vấn đề sau:

- Độ phức tạp bài toán. Mỗi bài toán có độ phức tạp khác nhau.

- Một bài toán có nhiều thuật toán giải nó.

- Dùng nhiều ngôn ngữ lập trình để cài đặt phần mềm cho một thuật toán

- Có thể dùng nhiều cấu trúc dữ liệu cho một thuật toán.

Một số sai sót cơ bản khi giải bài toán:

- Hiểu sai bài toán.

- Tìm sai thuật toán.

- Do không hiểu ngôn ngữ lập trình nên có nhầm lẫn.

- Bản thân dữ liệu quét không hết trường hợp.

- Yêu cầu giải quyết bài toán.

- Phần mềm dễ sử dụng .

- Tốc độ tính toán nhanh .

- Bộ nhớ phù hợp .

Trong đó tính dễ sử dụng là một yêu cầu cơ bản nhất.

7.2. Độ phức tạp thuật toán

Khái niệm thuật toán chính xác liên quan đến các khái niệm máy Turing, hàm đệ qui, thuật toán Marcop, ngôn ngữ hình thức của N.Chomsky. Những khái niệm này không nằm trong khuôn khổ Giáo trình này. Chúng tôi trình bày một khái niệm quan trọng liên quan trực tiếp đến thuật toán. Đó là độ phức tạp thuật toán. Nhờ có khái niệm này chúng ta có thể đánh giá và so sánh được các thuật toán với nhau. Hay nói một cách khác, chúng ta có thể có công cụ đo, để lựa chọn một thuật toán tốt cho lời giải bài toán cần giải quyết. Thông thường chúng ta có hai loại đánh giá: Một là độ phức tạp về thời gian tính của thuật toán, hai là độ phức tạp về phạm vi bộ nhớ dùng cho thuật toán. Đối với một thuật toán, thời gian tính và phạm vi bộ nhớ cần dùng thường mâu thuẫn nhau. Có nghĩa là, nếu thời gian tính của thuật toán là ngắn thì thông thường phạm vi bộ nhớ dùng cho thuật toán đó lại lớn. Mà chúng ta lại muốn chọn một thuật toán thời gian tính thì ngắn và bộ nhớ dùng cũng nhỏ. Như vậy, trong từng trường hợp cụ thể, chúng ta sẽ quyết định chọn lựa thuật toán nào. Trong phạm vi Giáo trình này chúng ta chỉ trình bày về độ phức tạp thời gian tính. Đó là độ phức tạp thường được đề cập nhiều nhất. Đồng thời,

trong phạm vi giới hạn của giáo trình, chúng ta cũng chỉ trình bày độ phức tạp của thuật toán theo góc độ tin học.

Giả sử T là thuật toán giải quyết bài toán A. Chúng ta gọi T(n) là độ phức tạp thời gian của thuật toán T. Thông thường T(n) được biểu diễn dưới dạng sau: T(n) = O(g(n)). Trong đó hàm g(n) là cấp của T(n); n là độ dài thông tin đưa vào.

Ví dụ: $T(n) = O(n^2)$

Chúng ta hiểu f(n) = O(g(n)), nếu ∃ hằng số C và số nguyên n_0.

Sao cho:

$$\forall n \geq n_0 \text{ ta luôn có: } f(n) \leq Cg(n)$$

(Nói cách khác là g(n) là hàm chặn trên của f(n) từ một chỉ số nào đó trở đi).

Rõ ràng, trong quá trình đánh giá thuật toán, nếu có g(n) nhỏ nhất thì đó là sự đánh giá chính xác nhất.

Có thể thấy rằng, bài toán tìm g(n) nhỏ nhất khá phức tạp.

Bây giờ, chúng ta đưa ra độ phức tạp thời gian là hàm nhiều biến. Giả sử $T(n_1,\ldots, n_k)$ là độ phức tạp thời gian của thuật toán T và $T(n_1,\ldots, n_k) = O(g(n_1,\ldots n_k))$. Khi đó chúng ta hiểu rằng tồn

tại các số C , $n_{01}...n_{0k}$ sao cho với mọi $n_1 \geq n_{01}$, ..., $n_k \geq n_{0k}$

$$T(n_1,...,n_k) \leq Cg(n_1,..., n_k)$$

Ví dụ: Đầu vào là R = {a_1,....., a_n},

r = {h_1,..., h_m}

Chúng ta có T(n,m) = O(g(n,m))

Trong trường hợp có nhiều đối số thì phức tạp thời gian được tính theo đối số có giá trị lớn nhất.

Ví dụ: $T(n,m) = O(n^2+2^m)$. Khi đó độ phức tạp thời gian của thuật toán T là hàm số mũ.

Việc đánh giá như trên gọi là độ phức tạp thời gian tồi nhất.

Trong thực tế có nhiều cách đánh giá độ phức tạp thời gian. Ví dụ như độ phức tạp thời gian trung bình. Độ phức tạp này gắn với nhiều độ đo khác nhau (độ đo xác suất). Giáo trình này đánh giá độ phức tạp thời gian theo cách tồi nhất (tìm g(n) chặn trên).

- Giả sử một độ phức tạp thuật toán chia làm nhiều đoạn, mỗi đoạn có độ phức tạp tương ứng là $T_1(n)$,...,$T_q(n)$.

Khi đó chúng ta đặt $T(n) = O(\max(T_1(n),..., T_q(n))$.

- Nếu T_i là hàm nhiều biến: $T_1(n_1,...,n_m)$, ..., $T_q(n_1,...,n_m)$, thì lúc đó $T_1(n_1,..., n_m) = O(\max(T_1(n_1,..., n_m),..., T_q(n_1,..., n_m))$.

- Giả sử T là thuật toán giải quyết bài toán A.

Ta chia hai khúc T_1 và T_2 và T_2 lồng trong T_1. Khi đó $T(n) = O(T_1(n). T_2(n))$.

Ví dụ: x:= 3;

x:= x + 1.

Dễ thấy độ phức tạp T(n) = O(c) (hay O(1)).

Ví dụ: x:= 3;

For i:= 1 to n do x:= x+1

Thì: T(n) = O(cn).

Ví dụ: For i: = 1 to n do

For j: = 1 to n do x:= x+1

$T(n) = O(c.n.n) = O(c.n^2)$

Trong trường hợp thuật toán có các đoạn lồng thắt vào nhau thì độ phức tạp là tích (Thể hiện bài toán có những toán tử lặp chu trình).

- Trong thuật toán chia thành nhiều đoạn. Có đoạn lồng thắt của đoạn khác: tính tích, có đoạn rời rạc: tính max.

Thông thường người ta chia các bài toán thành ba lớp. Đó là lớp bài toán được giải quyết bằng một thuật toán có độ phức tạp là hàm mũ, lớp bài toán NP - đầy đủ và lớp bài toán được giải quyết bằng một thuật toán có độ phức tạp là hàm đa thức.

- Đối với lớp bài toán được giải bằng thuật toán là hàm mũ và lớp bài toán NP - đầy đủ (Thường gọi là các bài toán không khả thi) thực tế trong tin học các bài toán này không có khả năng thực hiện vì thời gian tính quá lớn. Khi đó, chúng ta phải tách bài toán thành các dạng riêng biệt, và cố gắng đưa nó về lớp bài toán có độ phức tạp là hàm đa thức.

- Đối với các bài toán được giải bằng thuật toán có độ phức tạp là hàm đa thức, chúng ta cố gắng giảm số mũ k xuống (gần sát tuyến tính (mũ 1)).

Để hạ k (tăng tốc độ) thông thường người ta dùng cấu trúc dữ liệu, sử dụng ngôn ngữ gần ngôn ngữ máy.

Thông thường người ta coi các thông số vào (input) bình đẳng với nhau.

Tài liệu tham khảo

[1] Armstrong W.W. Dependency Structures of Database Relationships. Information Processing 74, Holland publ. Co. (1974), 580-583.

[2] Beeri C. Bernstein P.A. Computational problems related to the design of normal form relational schemas. ACM trans on Database Syst. 4.1 (1979), 30-59

[3] Beeri C. Dowd M., Fagin R.. Staman R. On the structure of Armstrong relations for Functional Dependencies . J.ACM 31,1 (1984), 30-46.

[4] Codd E. F. A relational model for large shared data banks. Communications ACM 13 (1970), 377-387.

[5] Demetrovics J., Logical and structural Investigation of Relational Datamodel. MTA - SZTAKI Tanulmanyok, Budapest, 114 (1980), 1-97.

[6] Demetrovics J., Libkin, L. Functional dependencies in relational databases : A Lattice point of view. Discrete Aplied Mathematics 40 (1992), 155-185.

[7] Demetrovics J., Thi V.D. Some results about functional dependencies. Acta Cybernetica 8,3 (1988), 273-278.

[8] Demetrovics J., Thi V.D. Relations and minimal keys. Acta Cybernetica 8,3 (1988), 279-285.

[9] Demetrovics J., Thi V.D. On keys in the Relational Datamodel. Inform. Process Cybern. EIK 24, 10 (1988), 515 - 519

[10] Demetrovics J., Thi V.D. Algorithm for generating Armstrong relations and inferring functional dependencies in the relational datamodel. Computers and Mathematics with Applications. Great Britain. 26,4(1993), 43 - 55.

[11] Demetrovics J., Thi V.D. Some problems concerning Keys for relation Schemes and Relationals in the Relational Datamodel. Information Processing Letters. North Holland 46,4(1993),179-183

[12] Demetrovics J., Thi. V.D. Some Computational Problems Related to the functional Dependency in the Relational Datamodel. Acta Scientiarum Mathematicarum 57, 1 - 4 (1993), 627 - 628.

[13] Demetrovics J., Thi V.D. Armstrong Relation, Functional Dependencies and Strong Dependencies. Comput. and AI. (submited for publication).

[14] Demetrovics J., Thi V.D. Keys, antikeys and prime attributes. Ann. Univ. Scien. Budapest Sect. Comput. 8 (1987), 37 - 54.

[15] Demetrovics J., Thi V.D. On the Time Complexity of Algorithms Related to Boyce-Codd Normal Forms. J. Serdica, the Bulgarian Academy of Sciencies, No.19 (1993), 134 - 144.

[16] Demetrovics J., Thi V.D. Generating Armstrong Relations for Relation schemes and Inferring Functional Dependencies from Relations.

International Jounal on Information Theories and Applications, ITA-93, 1, 4 (1993), 3 - 12.

[17] Demetrovics J., Thi V.D. Some problems concerning Armstrong relations of dual schemes and

relation schemes in the relational datamodel. Acta Cybernetica 11, 1-2 (1993), 35 - 47.

[18] Demetrovics J., Thi V.D. Normal Forms and Minimal Keys in the Relational Datamodel. Acta Cybernetica Vol. 11,3 (1994), 205 - 215.

[19] Demetrovics J., Thi, V.D. Some results about normal forms for functional dependency in the relational datamodel. Discrete Aplied Mathematics 69 (1996), 61 - 74.

[20] Garey M.R., Johnson D.S Computers and Intractability: A Guide to theory of NP - Completeness. Bell Laboratories. W.H Freeman and Company. San Francisco 1979.

[21] Gottlob G. Libkin L. Investigations on Armstrong relations dependency inference, and excluded functional dependencies. Acta Cybernetica Hungary IX/4 (1990), 385 - 402.

[22] Jou J.H, Fischer P.C. The complexity of recognizing 3NF relation schemes . IPL 14 (1982), 187 - 190.

[23] * Libkin L. Direct product decompositions of lattices, closures and relation schemes. Discrete Mathematics, North-Holland, 112 (1993), 119-138.

[24] Lucchesi C.L., Osborn S.L. Candidate keys for relations. J. Comput. Syst. Scien 17,2 (1978), 270 - 279

[25] Maier D. Minimum covers in the relational database model. JACM 27,4 (1980), 664 - 674.

[26] * Mannila H., Raiha K.J. Algorithms for inferring functional dependencies from relations. Data and Knowledge Engineering, North - Holland, V. 12, No. 1 (1994), 83 - 99.

[27] * Mannila H., Raiha K.J. On the complexity of inferring functional dependencies. Discrete Applied Mathematics, North - Holland, 40 (1992), 237 - 243.

[28] * Thalheim B. The number of keys in relational and nested relational databases. Discrete Applied Mathematics, North - Holland, 40 (1992), 265 - 282.

[29] Thi V.D. Investigation on Combinatorial Characterizations Related to Functional Dependency in the Relational Datamodel. MTA-SZTAKI Tanulmanyok. Budapest, 191 (1986), 1 - 157. Ph.D. dissertation.

[30] Thi V.D. Minimal keys and Antikeys. Acta Cybernetica 7.4 (1986), 361 - 371

[31] Thi V.D. Minimal keys and Antikeys. Acta Cybernetica 7, 4 (1986), 361 - 371.

[32] Thi V.D. Strong dependencies and s-semilattices. Acta Cybernetica 7, 2 (1987), 175 - 202.

[33] Thi V.D. Logical dependencies and irredundant relations. Computers and Artificial Intelligence 7 (1988), 165 - 184.

[34] Thi V.D., Anh N.K. Weak dependencies in the relational datamodel. Acta Cybernetica 10, 1-2 (1991), 93 - 100.

[35] Thi V.D., Thanh L.T. Some remark on Functional Dependencies in the relational Datamodel J. Acta Cybernetica, Hungary Vol. 11, 4 (1994), 345 - 352.

[36] Thi V.D. On the equivalent descriptions of family of functional dependencies in the relational datamodel. J. Computer Science and Cybernetic, Hanoi Vietnam, Vol 11, 4 (1995), 40 - 50.

[37] Thi V.D. Some Computational problems related to normal forms. J. Computer Science and Cybernetic, Hanoi Vietnam, Vol 13, 1 (1997), 53 - 65.

[38] Thi V.D. On the nonkeys. J. Computer Science and Cybernetic, Hanoi Vietnam, Vol 13, 1 (1997), 11 - 15.

[39] Thi V.D. Some results about hypergraph. J. Computer Science and Cybernetic, Hanoi Vietnam, Vol 13, 2 (1997), 8 - 15.

[40] Tsou D.M., Fischer P.C. Decomposition of a relation scheme into Boyce-Codd normal form. SIGACT NEWS 14 (1982), 23 - 29.

[41] Ullman J.D. Principles of database and knowledgw base systems. Computer Science Press, Second Edition (1992).

[42] Yannakakis M., Paradimitriou C. Algebraic dependencies. J. Comp. Syst. Scien. 25 (1982), 2 - 41.

[43] Yu C.T., Johnson D.T. On the complexity of finding the set of candidate keys for a given set of functional dependencies. IPL 5, 4 (1976), 100 - 101.

[44] Zaniolo C. Analysis and design of relational schemata for database systems. Ph. D. UCLA (1976).

[45] Collongues A., Hugues J., Laroche B. MERISE - Phương pháp thiết kế hệ thống thông tin tin học hoá phục vụ quản lí doanh nghiệp (Bản dịch). Nhà xuất bản Khoa học kĩ thuật, 1994.

[46] Hồ Sĩ Khoa. Các phương pháp xây dựng các mô hình khái niệm dữ liệu

[47] Các tài liệu hướng dẫn sử dụng hệ MEGA

[48] Demetrovics J., Denev. , Pavlov R. Cơ sở toán của khoa học tính toán. Hungary, 1985.

Mục lục

Trang

Chương mở đầu 9

Chương 2
Các kiến thức cơ bản về cơ sở dữ liệu

2.1. Khái quát về mô hình dữ liệu
2.2. Các khái niệm cơ bản và hệ tiên đề Armstrong
2.3. Họ phụ thuộc hàm và các mô tả tương đương
2.4. Các thuật toán liên quan đến các khoá
2.5. Mối liên hệ giữa quan hệ Armstrong và sơ đồ quan hệ

Chương 3
Các dạng chuẩn
và các thuật toán liên quan

3.1. Các khái niệm chung

3.2. Dạng chuẩn 2 (2NF)
3.3. Dạng chuẩn 3 (3NF)
3.4. Dạng chuẩn Boyce - Codd (BCNF)
3.5. Các thuật toán liên quan
3.6. Dạng chuẩn của các hệ khoá
3.7. Ví dụ

Chương 4
Các phép toán xử lí bảng

4.1. Các phép toán cơ bản
4.2. Các phép toán khác
4.3. Các ví dụ

Chương 5
Một số áp dụng mô hình dữ liệu trong các hệ quản trị cơ sở dữ liệu (QTCSDL) hiện có

5.1. Mô tả chung
5.2. Những khái niệm cơ bản
5.3. Mối quan hệ giữa các thực thể
5.4. Các dạng chuẩn trong các hệ QTCSDL hiện có

Chương 6

Một số công đoạn xây dựng
các dự án thiết kế tổng thể các hệ thống
cơ sở dữ liệu hiện nay

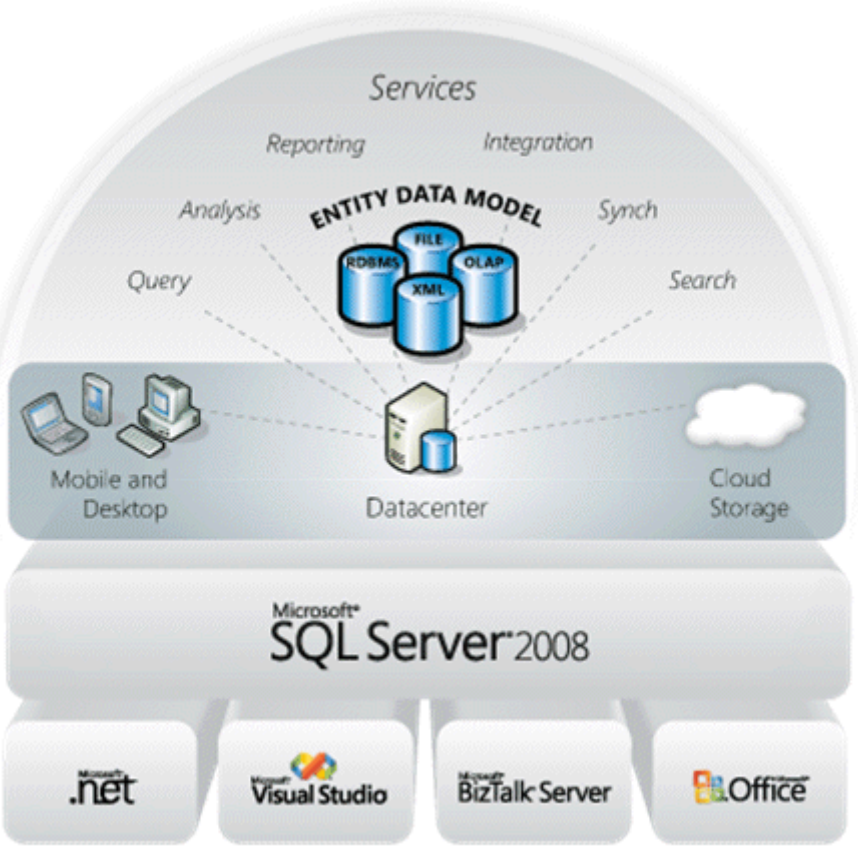
6.1. Khảo sát thông tin
6.2. Thiết kế mô hình dữ liệu
6.3. Kiểm soát và chuẩn hoá mô hình

Chương 7
Thuật toán và độ phức tạp

7.1. Khái niệm thuật toán
7.2. Độ phức tạp thuật toán

Tài liệu tham khảo

Giáo trình “Các hệ quản trị cơ sở dữ liệu



Chương 1 TỔNG QUAN CƠ SỞ DỮ LIỆU ACCESS

1. GIỚI THIỆU MICROSOFT ACCESS

Microsoft Access là một Hệ Quản Trị Cơ Sở Dữ Liệu (QTCSDL) tương tác người sử dụng chạy trong môi trường Windows. Microsoft Access cho chúng ta một công cụ hiệu lực và đầy sức mạnh trong công tác tổ chức, tìm kiếm và biểu diễn thông tin.

Microsoft Access cho ta các khả năng thao tác dữ liệu, khả năng liên kết và công cụ truy vấn mạnh mẽ giúp quá trình tìm kiếm thông tin nhanh. Người sử dụng có thể chỉ dùng một truy vấn để làm việc với các dạng cơ sở dữ liệu khác nhau. Ngoài ra, có thể thay đổi truy vấn bất kỳ lúc nào và xem nhiều cách hiển thị dữ liệu khác nhau chỉ cần động tác nhấp chuột.

Microsoft Access và khả năng kết xuất dữ liệu cho phép người sử dụng thiết kế những biểu mẫu và báo cáo phức tạp đáp ứng đầy đủ các yêu cầu quản lý, có thể vận động dữ liệu và kết hợp các biểu mẫu va báo cáo trong một tài liệu và trình bày kết quả theo dạng thức chuyên nghiệp.

Microsoft Access là một công cụ đầy năng lực để nâng cao hiệu suất công việc. Bằng cách dùng các Wizard của MS Access và các lệnh có sẵn (macro) ta có thể dễ dàng tự động hóa công việc mà không cần lập trình. Đối với những nhu cầu quản lý cao, Access đưa ra ngôn ngữ lập trình Access Basic (Visual Basic For application) một ngôn ngữ lập trình mạnh trên CSDL.

2. KHỞI ĐỘNG VÀ THOÁT KHỎI ACCESS

2.1. Khởi động ACCESS

Chọn nút Start trên thanh Task bar

Chọn Programs

Chọn Microsoft ACCESS

Khung hội thoại Microsoft ACCESS gồm:

Create a New Database Using : Tạo CSDL ứng dụng mới.

Blank Database : Tạo CSDL trống.

Database Wizard : Tạo với sự trợ giúp của Wizard.

Open an Existing Database : Mở một CSDL có sẵn.

2.2. Thoát khỏi ACCESS

Chọn File/Exit hoặc nhấn tổ hợp phím ALT+F4

3. CÁC THAO TÁC TRÊN TẬP TIN CƠ SỞ DỮ LIỆU ACCESS

3.1. Tạo một tập tin CSDL

Thực hiện các thao tác sau:

Chọn File/New hoặc chọn biểu tượng New trên thanh công cụ

Chọn Database, chọn OK

Trong mục *Save in:* Chọn thư mục cần chứa tên tập tin.

File name: Chọn tên tập tin cần tạo

(Phần mở rộng mặc định là MDB)

3.2. Mở một CSDL đã tồn tại trên đĩa

Chọn File/Open database (Hoặc click biểu tượng Open)

Trong mục *Look in :* Chọn thư mục cần chứa tên tập tin cần mở.

File name: Chọn tên tập tin cần mở.

Chọn Open

3.3. Đóng một CSDL

Chọn File/Close hoặc ALT+F4

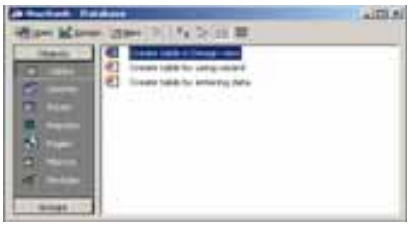
3.4. Các thành phần cơ bản của một tập tin CSDL ACCESS

Một tập tin CSDL ACCESS gồm có 6 thành phần cơ bản sau

Bảng (Tables) : Là nơi chứa dữ liệu

Truy vấn (Queries) : Truy vấn thông tin dựa trên một hoặc nhiều bảng.

Biểu mẫu (Forms) : Các biểu mẫu dùng để nhập dữ liệu hoặc hiển thị dữ liệu.



Báo cáo (Reports) : Dùng để in ấn.

Pages (Trang) : Tạo trang dữ liệu.

Macros (Tập lệnh) : Thực hiện các tập lệnh.

Modules (Đơn thể) : Dùng để lập trình Access Basic

4. CÁCH SỬ DỤNG CỬA SỔ DATABASE

Như đã nói ở trên, một CSDL của Access chứa trong nó 7 đối tượng chứ không đơn thuần là bảng dữ liệu. Sau khi tạo mới một CSDL hoặc mở một CSDL có sẵn Access sẽ hiển thị một cửa sổ Database, trên đó hiển thị tên của CSDL đang mở và liệt kê 7 đối tượng mà nó quản lý, mỗi lớp đối tượng đều được phân lớp rõ ràng để tiện theo dõi.

4.1. Tạo một đối tượng mới

Trong cửa sổ Database, chọn tab chứa đối tượng cần tạo (Bảng, Truy vấn, Biểu mẫu, Báo cáo,...) hoặc thực hiện lệnh

View/D atabase Object - Table/Query/Form/ReportPages/Macros/Modules

Chọn nút New.

4.2. Thực hiện một đối tượng trong CSDL

Trong cửa sổ Database, chọn tab cần thực hiện. Cửa sổ Database liệt kê tên các đối tượng có sẵn, chọn tên đối tượng cần mở.

Chọn nút Open (đối với Bảng, Truy vấn, Biểu mẫu, Trang) hoặc Preview (đối với Báo biểu) hoặc Run (đối với Macro và Module).

4.3. Sửa đổi một đối tượng có sẵn trong CSDL

Trong cửa sổ Database, chọn tab cần thực hiện. Cửa sổ Database liệt kê tên các đối tượng có sẵn, chọn tên đối tượng cần mở, Chọn nút Design.

Chương 2 BẢNG DỮ LIỆU

Bảng là đối tượng chủ yếu chứa các thông tin cần quản lý, có thể đó chỉ là một vài địa chỉ đơn giản hay cả vài chục nghìn bản ghi chứa đựng thông tin liên quan đến các hoạt động SXKD của một công ty xuất nhập khẩu nào đó. Trước khi ta muốn làm việc với bất kỳ một CSDL nào thì ta phải có thông tin để quản lý, các thông tin đó nằm trong các bảng, nó là cơ sở để cho người sử dụng tạo các đối tượng khác trong CSDL như truy vấn, biểu mẫu, báo biểu...

1. THIẾT KẾ CƠ SỞ DỮ LIỆU

Một CSDL được thiết kế tốt cho phép người sử dụng truy cập nhanh chóng đến những thông tin cần tham khảo, giúp tiết kiệm được thời gian truy xuất thông tin. Một CSDL thiết kế tốt giúp người sử dụng rút ra được những kết quả nhanh chóng và chính xác hơn.

Để thiết kế một CSDL tốt chúng ta phải hiểu cách mà một Hệ QTCSDL quản trị các CSDL như thế nào. MS Access hay bất kỳ một Hệ QTCSDL nào có thể cung cấp các thông tin cho chúng ta một cách chính xác và hiệu quả nếu chúng được cung cấp đầy đủ mọi dữ kiện về nhiều đối tượng khác nhau lưu trữ trong các bảng dữ liệu. Ví dụ ta cần một bảng để chứa thông tin về lý lịch của cán bộ, một bảng khác để chứa các đề tài nguyên cứu khoa học của các cán bộ...

Khi bắt tay thiết kế CSDL, chúng ta phải xác định và phân tích các thông tin muốn lưu trữ thành các đối tượng riêng rẽ, sau đó báo cho Hệ QTCSDL biết các đối tượng đó liên quan với nhau như thế nào. Dựa vào các quan hệ đó mà Hệ QTCSDL có thể liên kết các đối tượng và rút ra các số liệu tổng hợp cần thiết.

CÁC BƯỚC THIẾT KẾ CSDL

Bước 1: Xác định mục tiêu khai thác CSDL của chúng ta. Điều này quyết định các loại sự kiện chúng ta sẽ đưa vào MS Access.

Bước 2: Xác định các bảng dữ liệu cần thiết. Mỗi đối tượng thông tin sẽ hình thành một bảng trong CSDL của chúng ta.

Bước 3: Sau khi đã xác định xong các bảng cần thiết, tiếp đến ta phải chỉ rõ thông tin nào cần quản lý trong mỗi bảng, đó là xác định các trường. Mỗi loại thông tin trong bảng gọi là trường. Mọi mẫu in trong cùng một bảng đều có chung cấu trúc các trường. Ví dụ: Trong lý lịch khoa học cán bộ, những trường (thông tin) cần quản lý là: “HỌ VÀ TÊN”, “CHUYÊN MÔN”, “HỌC VỊ”, “HỌC HÀM”,...

Bước 4: Xác định các mối quan hệ giữa các bảng. Nhìn vào mỗi bảng dữ liệu và xem xét dữ liệu trong bảng này liên hệ thế nào với dữ liệu trong bảng khác. Thêm trướng hoặc tạo bảng mới để làm rõ mối quan hệ này. Đây là vấn đề hết sức quan trọng, tạo được quan hệ tốt sẽ giúp chúng ta nhanh chóng truy tìm tìm và kết xuất dữ liệu.

Bước 5: Tinh chế, hiệu chỉnh lại thiết kế. Phân tích lại thiết kế ban đầu để tim lỗi, tạo bảng dữ liệu và nhập vào vài bản ghi, thử xem CSDL đó phản ánh thế nào với những yêu cầu truy xuất của chúng ta, có rút được kết quả đúng từ những bảng dữ liệu đó không. Thực hiện các chỉnh sửa thiết kế nếu thấy cần thiết.

2. KHÁI NIỆM VỀ BẢNG

Bảng là nơi chứa dữ liệu về một đối tượng thông tin nào đó như SINH VIÊN, HÓA ĐƠN,... Mỗi hàng trong bảng gọi là một **bản ghi** (record) chứa các nội dung riêng của đối tượng đó. Mỗi bản ghi của một bảng đều có chung cấu trúc, tức là các **trường** (field). Ví dụ: Cho bảng dưới đây để quản lý lý lịch khoa học cán bộ trong trường đại học, có các trường MACB (Mã cán bộ), TRINHDOVH (Trình độ văn hóa), CHUYENMON (Chuyên môn),...

Trong một CSDL có thể chứa nhiều bảng, thường mỗi bảng lưu trữ nhiều thông tin (dữ liệu) về một đối tượng thông tin nào đó, mỗi một thông tin đều có những kiểu đặc trưng riêng, mà với Access nó sẽ cụ thể thành những kiểu dữ liệu của các trường.

3. TẠO BẢNG MỚI TRONG CƠ SỞ DỮ LIỆU

Trong MS Access có hai cách để tạo bảng, một là cách dùng Table Wizard, nhưng các trường ở đây MS Access tự động đặt tên và không có bàn tay can thiệp của người sử dụng. Ở đây, sẽ đưa ra cách tạo mới bảng hoàn toàn do người sử dụng.

3.1. Tạo bảng không dùng Table Wizard

Trong cửa sổ Database, chọn tab Table (hoặc Lệnh *View/Daatbase object - Table*)

Chọn nút **New,** xuất hiện hộp thoại

Datasheet View: Trên màn hình sẽ xuất hiện một bảng trống với các trường (tiêu đề cột) lần lượt Field1, field2

Design View: Trên màn hình xuất hiện cửa sổ thiết kế bảng, người sử dụng tự thiết kế bảng.

Table Wizard: Thiết kế bảng với sự trợ giúp của MS Access

Import table: Nhập các bảng và các đối tượng từ các tập tin khác vào CSDL hiện thời.

Link table: Tạo bảng bằng cách nối vào CSDL hiện thời các bảng của CSDL khác.

Chọn chức năng Design View, chọn OK.

3.2. Sử dụng Design View

Field Name: Tên trường cần đặt (thông tin cần quản lý)

Data Type: Kiểu dữ liệu của trường

Desciption: Mô tả trường, phần này chỉ mang ý nghĩa làm rõ thông tin quản lý, có thể bỏ qua trong khi thiết kế bảng.

Field properties: Các thuộc tính của trường

Xác định khoá chính của bảng (nếu có)

Xác định thuộc tính của bảng, Lưu bảng dữ liệu

Đặt tên trường

Tên trường ở đây không nhất thiết phải có độ dài hạn chế và phải sát nhau, mà ta có thể đặt tên trường tùy ý nhưng không vượt quá 64 ký tự kể cả ký tự trắng. Lưu ý rằng, tên trường có thể đặt dài nên nó dẽ mô tả được thông tin quản lý, nhưng sẽ khó khăn hơn khi ta dùng các phát biểu SQL và lập trình Access Basic. Do đó khi đặt tên trường ta nên đặt ngắn gọn, dễ gợi nhớ và không chứa ký tự trắng.

Kiểu dữ liệu

MS Access cung cấp một số kiểu dữ liệu cơ bản sau:

Kiểu dữ liệu	Dữ liệu vào	Kích thước
Text	Văn bản	Tối đa 255 byte
Memo	Văn bản nhiều dòng, trang	Tối đa 64000 bytes
Number	Số	1,2,4 hoặc 8 byte
Date/Time	Ngày giờ	8 byte
Currency	Tiền tệ (Số)	8 byte
Auto number	ACCESS tự động tăng lên một khi một bản ghi được tạo	4 byte
Yes/No	Lý luận (Boolean)	1 bit
OLE Object	Đối tượng của phần mềm khác	Tối đa 1 giga byte
Lookup Wizard		Trường nhận giá trị do người dùng chọn từ 1 bảng khác hoặc 1 danh sách giá trị định trước
Hyper link	Liên kết các URL	

Quy định thuộc tính, định dạng cho trường

Đặt thuộc tính là một phần không kém quan trọng, nó quyết định đến dữ liệu thực sự lưu giữa trong bảng, kiểm tra độ chính xác dữ liệu khi nhập vào, định dạng

dữ liệu nhập vào ... Mỗi một kiểu dữ liệu sẽ có các thuộc tính và các đặc trưng và khác nhau. Sau đây là các thuộc tính, định dạng của các kiểu dữ liệu.

Để tăng thêm tốc độ xử lý khi nhập dữ liệu cũng như các công việc tìm kiếm sau này thì việc quy định dữ liệu rất quan trọng.

Các trường trong ACCESS có các thuộc tính sau:

3.2.1. Field Size

Quy định kích thước của trường và tùy thuộc vào từng kiểu dữ liệu

Kiểu Text: Chúng ta quy định độ dài tối đa của chuỗi.

Kiểu Number: Có thể chọn một trong các loại sau:

Byte: 0..255

Integer: -32768..32767

Long Integer: -3147483648.. 3147483647

Single:$-3,4x10^{38}..3,4x10^{38}$ (Tối đa 7 số lẻ)

Double: $-1.797x10^{308}$ ..$1.797x10^{308}$ (Tối đa 15 số lẻ)

Decimal Places

Quy định số chữ số thập phân (Chỉ sử dụng trong kiểu Single và Double)

Đối với kiểu Currency mặc định decimal places là 2

3.2.2. Format

Quy định dạng hiển thị dữ liệu, tùy thuộc vào từng kiểu dữ liệu.

Kiểu chuỗi: Gồm 3 phần

<Phần 1>;<Phần 2>;<Phần 3>

Trong đó:

<Phần 1>: Chuỗi định dạng tương ứng trong trường hợp có chứa văn bản.

<Phần 2>: Chuỗi định dạng tương ứng trong trường hợp không chứa văn bản.

<Phần 3>: Chuỗi định dạng tương ứng trong trường hợp null

Các ký tự dùng để định dạng chuỗi

Ký tự	Tác dëch vuû du lëchụng
@	Chuỗi ký tự
>	Đổi tất cả ký tự nhập vào thành in hoa
<	Đổi tất cả ký tự nhập vào thành in thường
“Chuỗi ký tự “	Chuỗi ký tự giữa 2 dấu nháy
\<ký tự>	Ký tự nằm sau dấu \
[black] [White] [red] Hoặc [<số>] Trong đó 0<=số<=56	Màu

Ví dụ

Cách định dạng	Dữ liệu	Hiển thị
@@@-@@@@	123456 abcdef	123-456 abc-def
>	Tinhoc	TINHOC
<	TINHOC	Tinhoc
@;”Không có”;”Không biết”	Chuỗi bất kỳ Chuỗi rỗng Giá trị trống (Null)	Hiển thị chuỗi Không có Không biết

Kiểu Number

Định dạng do ACCESS cung cấp

Dạng	Dữ liệu	Hiển thị
General Number	1234.5	1234.5
Currency	1234.5	$1.234.50
Fixed	1234.5	1234

Standard	1234.5	1,234.50
Pecent	0.825	82.50%
Scientific	1234.5	1.23E+03

Định dạng do người sử dụng

<Phần 1>;<Phần 2>;<Phần 3>;<Phần 4>

<Phần 1>:Chuỗi định dạng tương ứng trong trường hợp số dương.

<Phần 2>: Chuỗi định dạng tương ứng trong trường hợp số âm.

<Phần 3>: Chuỗi định dạng tương ứng trong trường hợp số bằng zero.

<Phần 4>: Chuỗi định dạng tương ứng trong trường hợp null.

Các ký tự định dạng

Ký tự	Tác dụng
.(Period)	Dấu chấm thập phân
,(comma)	Dấu phân cách ngàn
0	Ký tự số (0-9)
#	Ký tự số hoặc khoảng trắng
$	Dấu $
%	Phần trăm

Ví dụ

Định dạng	Hiển thị
0;(0);;”Null”	Số dương hiển thị bình thường Số âm được bao giữa 2 dấu ngoặc Số zero bị bỏ trống Null hiện chữ Null
+0.0;-0.0;0.0	Hiển thị dấu + phía trước nếu số dương Hiển thị dấu - phía trước nếu số âm Hiển thị 0.0 nếu âm hoặc Null

Kiểu Date/Time

Các kiểu định dạng do ACCESS cung cấp

Dạng	Hiển thị
General date	10/30/99 5:10:30PM
Long date	Friday, may 30 , 1999
Medium date	30-jul-1999
Short date	01/08/99
Long time	6:20:00 PM
Medium time	6:20 PM
Short time	18:20

Các ký tự định dạng

Ký tự	Tác dụng
: (colon)	Dấu phân cách giờ
/	Dấu phân cách ngày
d	Ngày trong tháng (1-31)
dd	Ngày trong tháng 01-31)
ddd	Ngày trong tuần (Sun -Sat0
W	Ngày trong tuần (1-7)
WW	Tuần trong năm (1-54)
M	Tháng trong năm (1-12)
MM	Tháng trong năm (01-12)
q	Quý trong năm (1-4)
y	Ngày trong năm (1-366)
yy	Năm (01-99)
h	Giờ (0-23)
n	Phút (0-59)
s	Giây (0-59)

Ví dụ

Định dạng	Hiển thị
Ddd,”mmm d”,yyyy	Mon,jun 2, 1998
Mm/dd/yyyy	01/02/1998

Kiểu Yes/No

Các kiểu định dạng

Định dạng	Tác dụng
Yes/No	Đúng/Sai
True/False	Đúng/Sai
On/Off	Đúng/Sai

Định dạng do người sử dụng: Gồm 3 phần

<Phần 1>;<Phần 2>;<Phần 3>

Trong đó: <Phần 1>: Bỏ trống

<Phần 2>: Trường hợp giá trị trường đúng

<Phần 3>: Trường hợp giá trị trường sai

Ví dụ

Định dạng	Hiển thị	
	Trường hợp True	Trường hợp False
;”Nam”;”Nu”	Nam	Nu
;”co”;”Khong”	Co	Khong

3.2.3. Input mask (Mặt nạ)

Thuộc tính này dùng để quy định mặt nạ nhập dữ liệu cho một trường.

Các ký tự định dạng trong input mask

Ký tự	Tác dụng
0	Bắt buộc nhập ký tự số
9	Không bắt buộc nhập, ký tự số
#	Không bắt buộc nhập, số 0-9, khoảng trắng, dấu + và -
L	Bắt buộc nhập, ký tự chữ
?	Không bắt buộc nhập, ký tự chữ hoặc khoảng trắng
a	Bắt buộc nhập, ký tự chữ hoặc số
A	Không bắt buộc nhập, ký tự chữ hoặc số
&	Bắt buộc nhập, ký tự bất kỳ
C	Không bắt buộc nhập ký tự bất kỳ
<	Các ký tự bên phải được đổi thành chữ thường
>	Các ký tự bên phải được đổi thành chữ hoa
!	Dữ liệu được ghi từ phải sang trái
\<Ký tự>	Ký tự theo sau \ sẽ được đưa thẳng vào

Ví dụ

Input mask	Dữ liệu nhập vào
(000)000-0000	(054)828-8282
(000)AAA-A	(123)124-E

☞ **Chú ý:** Nếu muốn các ký tự gõ vào quy định thuộc tính input mask là password (Khi nhập dữ liệu vào tại các vị trí đó xuất hiện dấu *).

3.2.4. Caption

Quy định nhãn là một chuỗi ký tự sẽ xuất hiện tại dòng tiêu đề của bảng. Chuỗi ký tự này cũng xuất hiện tại nhãn các của các điều khiển trong các biểu mẫu hoặc báo cáo.

3.2.5. Default value

Quy định giá trị mặc định cho trường trừ Auto number và OEL Object

3.2.6. Validation rule và Validation Text

Quy định quy tắc hợp lệ dữ liệu (Validation rule) để giới hạn giá trị nhập vào cho một trường. Khi giới hạn này bị vi phạm sẽ có câu thông báo ở Validation text.

Các phép toán có thểt dùng trong Validation rule

Các phép toán	Phép toán	Tác dụng
Phép so sánh	>, <, >=, <=, =, <>	
Phép toán logic	Or, and , not	Hoặc, và, phủ định
Phép toán về chuỗi	Like	Giống như

☞ ***Chú ý:*** Nếu hằng trong biểu thức là kiểu ngày thì nên đặt giữa 2 dấu #.

Ví dụ

Validation rule	Tác dụng
<>0	Khác số không
Like “*HUE*”	Trong chuỗi phải chứa HUE
<#25/07/76#	Trước ngày 25/07/76
>=#10/10/77# and <=#12/11/77#	Trong khoảng từ 10/10/77 đến 12/12/77

3.2.7. Required

Có thể quy định thuộc tính này để bặt buộc hay không bắt buộc nhập dữ liệu cho trường.

Required	Tác dụng
Yes	Bắt buộc nhập dữ liệu
No	Không bắt buộc nhập dữ liệu

3.2.8. AllowZeroLength

Thuộc tính này cho phép quy định một trường có kiểu Text hay memo có thể hoặc không có thể có chuỗi có độ dài bằng 0.

☞ ***Chú ý:*** Cần phân biệt một trường chứa giá trị null (chưa có dữ liệu) và một trường chứa chuỗi có độ dài bằng 0 (Có dữ liệu nhưng chuỗi rỗng “”).

AllowZeroLength	Tác dụng
Yes	Chấp nhận chuỗi rỗng
No	Không chấp nhận chuỗi rỗng

3.2.9. Index

Quy định thuộc tính này để tạo chỉ mục trên một trường. Nếu chúng ta lập chỉ mục thì việc tìm kiếm dữ liệu nhanh hơn và tiện hơn.

Index	Tác dụng
Yes(Dupplicate OK)	Tạo chỉ mục có trùng lặp
Yes(No Dupplicate)	Tạo chỉ mục không trùng lặp
No	Không tạo chỉ mục

3.2.10. New value

Thuộc tính này chỉ đối với dữ liệu kiểu auto number, quy định cách thức mà trường tự động điền số khi thêm bản ghi mới vào.

New value	Tác dụng
Increase	Tăng dần
Random	Lấy số ngẫu nhiên

4. THIẾT LẬP KHOÁ CHÍNH (*primary key*)

4.1. Khái niệm khoá chính

Sức mạnh của một Hệ QTCSDL như Microsoft Access, là khả năng mau chóng truy tìm và rút dữ liệu từ nhiều bảng khác nhau trong CSDL. Để hệ thống có thể làm được điều này một cách hiệu quả, mỗi bảng trong CSDL cần có một trường hoặc một nhóm các trường có thể xác định duy nhất một bản ghi trong số rất nhiều bản ghi đang có trong bảng. Đây thường là một mã nhận diện như Mã nhân viên hay Số Báo Danh của học sinh. Theo thuật ngữ CSDL trường này được gọi là **khóa chính** (primary key) của bảng. MS Access dùng trường khóa chính để kết nối dữ liệu nhanh chóng từ nhiều bảng và xuất ra kết quả yêu cầu.

Nếu trong bảng chúng ta đã có một trường sao cho ứng với mỗi trị thuộc trường đó chúng ta xác định duy nhất một bản ghi của bảng, chúng ta có thể dùng trường đó làm trường khóa của bảng. Từ đó cho ta thấy rằng tất cả các trị trong trường khóa chính phải khác nhau. Chẳng hạn đừng dùng tên người làm trường khóa vì tên trường là không duy nhất.

Nếu không tìm được mã nhận diện cho bảng nào đó, chúng ta có thể dùng một trường kiểu **Autonunter** (ví dụ Số Thứ Tự) để làm trường khóa chính.

Khi chọn trường làm khóa chính chúng ta lưu ý mấy điểm sau:

MS Access không chấp nhận các giá trị trùng nhau hay trống (null) trong trường khóa chính.

Chúng ta sẽ dùng các giá trị trong trường khóa chính để truy xuất các bản ghi trong CSDL, do đó các giá trị trong trường này không nên quá dài vì khó nhớ và khó gõ vào.

Kích thước của khóa chính ảnh hưởng đến tốc độ truy xuất CSDL. Để đạt hiệu quả tối ưu, dùng kích thước nhỏ nhất để xác định mọi giá trị cần đưa vào trường.

4.2. Cách đặt khoá chính

Ta có thể tự chọn trường làm khóa chính cho bảng bằng các bước sau đây:

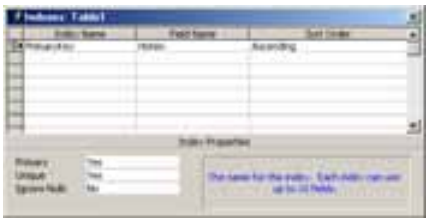
Mở bảng ở chế độ Design View

Nhắp chọn trường cần đặt

Thực hiện lệnh *Edit - Primary Key* hoặc nhắp chọn nút trên thanh công cụ của mục này .

☞***chú ý:*** Không phải mọi trường đều có thể làm khóa chính, mà chỉ có các trường có các kiểu dữ liệu không phải là **Memo** và **OLE Object**., **Hyper Link**.

Để hủy bỏ khóa chính hoặc các đã thiết lập thì thực hiện lệnh *View - Indexes*, trong hộp thoại này chọn và xóa đi những trường khóa đã thiết lập:



5. LƯU BẢNG

Sau khi thiết kế xong, ta tiến hành lưu bảng vào CSDL, có thể thực hiện một trong hai thao tác sau:

Thực hiện lệnh *File - Save*.

Nhắp chọn nút trên thanh công cụ của mục này (Table Design)

6. HIỆU CHỈNH BẢNG

6.1. Di chuyển trường: Các thao tác để di chuyển thứ tự các trường

Đưa con trỏ ra đầu trường đến khi con trỏ chuột chuyển thành ➡ thì nhắp chọn.

Đưa con trỏ ra đầu trường vừa chọn, nhấn và kéo đếnvị trí mới.

6.2. Chen trường : Các thao tác lần lượt như sau

Chọn trường hiện thời là trường sẽ nằm sau trường được chen vào

Thực hiện lệnh *Insert/ Row*

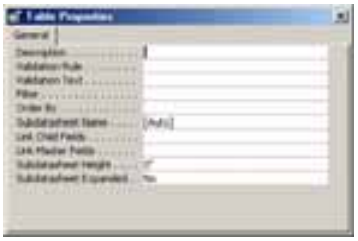
6.3. Xóa trường: Các thao tác lần lượt như sau

Chọn trường cần xóa

Thực hiện lệnh *Edit - Delete Rows*

6.4. Quy định thuộc tính của bảng

Mở bảng ở chế độ Design View, chọn View/Properties



Description: Dòng mô tả bảng

Validation Rule: Quy tắc hợp lệ dữ liệu cho toàn bảng.

Validation Text: Thông báo lỗi khi dữ liệu không hợp lệ.

7. XEM THÔNG TIN VÀ BỔ SUNG BẢN GHI

7.1. Xem thông tin ở chế độ datasheet

Muốn xem thông tin trong một bảng chúng ta phải chuyển bảng sang một chế độ hiển thị khác gọi là *Datasheet*. Trong chế độ hiển thị này, mỗi bản ghi được thể hiện trên một hàng ngang, hàng đầu tiên là các tên trường.

Sau đây là các cách để chuyển sang chế độ hiển thị Datasheet:

Trong của sổ Database của CSDL đang mở, nhấp chọn tab Table. Trong mục này chọn bảng cần hiển thị rồi chọn nút **Open**, bảng sẽ được mở để bổ sung và chỉnh sửa dữ liệu.

Ta có thể chuyển sang chế độ Datasheet ngay khi đang ở trong chế Design, bằng cách nhấp chọn nút thì bảng sẽ chuyển sang chế độ Datasheet, để quay trở về chế độ Design, ta nhấp chọn lại nút . Hoặc chọn lệnh *View - /Design View*.

7.2. Bổ sung bản ghi cho bảng

Sau khi hoàn thành công việc thiết kế cấu trúc bảng, ta tiến hành nhập dữ liệu, tức là bổ sung các bản ghi, cho bảng. Hiển thị bảng ở chế độ hiển thị Datasheet, mỗi hàng đại diện cho một bản ghi. Ở đây có các ký hiệu chúng ta cần biết công dụng của chúng

Bản ghi hiện thời

Bản ghi đang nhập dữ liệu

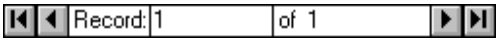
Bản ghi mới

7.2.1. Bổ sung bản ghi mới cho CSDL

Đang đứng tại một bản ghi nào đó (không phải là bản ghi mới), chọn nút trên thanh công cụ. Hoặc thực hiện lệnh *Record - Go To - New*.

7.2.2. Di chuyển giữa các bản ghi

Ta có thể di chuyển qua lại giữ các bản ghi bằng cách dùng công cụ *Chọn lựa bản ghi* (Record Selector).



7.2.3. Nhập dữ liệu cho bản ghi

Khi đang nhập dữ liệu cho một bản ghi nào đó, thì đầu hàng của bản ghi đó xuất hiện biểu tượng .

Tổ hợp phím	Tác dụng
Tab	Sang ô kế tiếp
Shift Tab	Sang ô phía trước
Home	Đến đầu dòng
End	Đến cuối dòng
Ctrl Home	Đến bản ghi đầu tiên
Ctrl End	Đến bản ghi cuối cùng
Shift F2	Zoom

Theo chuẩn, khi nhập dữ liệu thì Access sẽ lấy font mặc định là MS San Serif, điều này có thể giúp cho ta hiển thị được tiếng Việt chỉ khi MS San Serif đó là của VietWare hoặc của ABC.

Để không phụ thuộc vào điều này, ta nên chọn font trước khi tiến hành nhập dữ liệu. Trong chế độ hiển thị Datasheet, thực hiện lệnh *Format - Font...*

Khi nhập dữ liệu là trường cho trường **OLE Object**, ta thực hiện như sau: Lệnh *Edit - Object...*

7.2.4. Chọn các bản ghi

Đánh dấu chọn bản ghi:

Chọn Edit/Select Record: Để chọn bản ghi hiện hành

Chọn Edit/ Select all Record để chọn toàn bộ

7.2.5. Xóa bản ghi

Chọn các bản ghi cần xóa, sau đó thực hiện lệnh *Edit - Delete* (hoặc nhấn phím DELETE) .

8. THIẾT LẬP QUAN HỆ GIỮA CÁC BẢNG

8.1. Các loại quan hệ trong cơ sở dữ liệu ACCESS

8.1.1. Quan hệ một -một (1-1)

Trong quan hệ một -một, mỗi bản ghi trong bảng A có tương ứng với một bản ghi trong bảng B và ngược lại mỗi bản ghi trong bảng B có tương ứng duy nhất một bản ghi trong bảng A.

***Ví dụ*:** Cho 2 bảng dữ liệu

Bảng Danhsach(**Masv**, ten, Ngaysinh, gioitinh) và bảng Diemthi(**Masv**, diem)

Ten	Ngaysinh	Gioitinh	Masv
An	20/10/77	Yes	A001
Bình	21/07/80	No	A002
Thuỷ	02/12/77	Yes	A003
Lan	03/04/80	No	A004
Hồng	12/11/77	No	A005

Masv	diem
A001	9
A002	7
A003	9
A004	4
A005	5

Bảng **Danhsach** và **diemthi** có mối quan hệ **1-1** dựa trên trường Masv.

8.1.2. Quan hệ một nhiều (1-∞)

Là mối quan hệ phổ biến nhất trong CSDL, trong quan hệ một nhiều : Một bản ghi trong bảng A sẽ có thể có nhiều bản ghi tương ứng trong bảng B, nhưng ngược lại một bản ghi trong bảng B có duy nhất một bản ghi tương ứng trong bảng A.

Ví dụ: Trong một khoa của một trường học nào đó có nhiều sinh viên, những một sinh viên thuộc một khoa nhất định. Ta có 2 bảng dữ liệu như sau:

Bảng Danhsachkhoa(**Makhoa**, tenkhoa, sodthoai)

Bảng danhsachsv(Makhoa, Ten, Quequan, lop)

Tenkhoa	Sodthoai	Makhoa
CNTT	826767	01
TOÁN	878787	02
LÝ	868785	03

Makhoa	Ten	Queuqan	Lop
01	Thanh	Huế	K23
01	Tùng	Qbình	K24
02	Thuỷ	Huế	K25
02	Hùng	ĐN	K26
03	Lan	Huế	K25
03	Hương	ĐN	K26

Bảng Danhsachkhoa và bảng danhsachsv có mối quan hệ **1-∞** dựa trên trường Makhoa.

8.3. Quan hệ nhiều nhiều(∞-∞)

Trong quan hệ nhiều nhiều, mỗi bản ghi trong bảng A có thể có không hoặc nhiều bản ghi trong bảng B và ngược lại mỗi bản ghi trong bảng B có thể có không hoặc nhiều bản ghi trong bảng A.

Khi gặp mối quan hệ nhiều- nhiều để không gây nên sự trùng lặp và dư thừa dữ liệu thì người ta tách quan hệ nhiều-nhiều thành 2 quan hệ một-nhiều bằng cách tạo ra một bảng phụ chứa khóa chính của 2 bảng đó.

Ví dụ: Một giáo viên có thể dạy cho nhiều trường và một trường có nhiều giáo viên tham gia giảng dạy. Đây là một mối quan hệ nhiều-nhiều

Bảng Danhsachgv(**Magv**,ten)

Bảng Danhsachtruong(**Matruong**, Tentruong)

Tạo ra bảng Phancongday(**Magv, matruong**)

Ten	Magv
Thanh	G1
Thuý	G2
Hùng	G3

Magv	Matruong
G1	KH
G1	SP
G2	YK
G2	SP
G3	KH
G3	YK

Matruong	tentruong
KH	DHKH
SP	DHSP
YK	DHYK

Bảng **Danhsachgv** và bảng **Phancongday** có mối quan hệ **1-∞** dựa trên trường **Magv**.

Bảng **Danhsachtruong** và bảng **Phancongday** có mối quan hệ **1-∞** dựa trên trường **Matruong**.

8.4. Thiết lập mối quan hệ giữa các bảng dữ liệu (Relationships)

Tại cửa sổ Database, thực hiện lệnh Tools/Relationship

Trong cửa sổ Show

bảng cần thiết lập quan hệ, sau đó chọn **Add và Close**.

Kéo trường liên kết của bảng quan hệ vào trường của bảng được quan hệ (Table related).

Bật chức năng Enforce Referential Integrity (Nếu muốn quan hệ này bị ràng buộc tham chiếu toàn vẹn), chọn mối quan hệ (one-many) hoặc (one-one).

Chọn nút **Create**.

☞ ***Chú ý***

Quan hệ có tính tham chiếu toàn vẹn sẽ đảm bảo các vấn đề sau:

Khi nhập dữ liệu cho trường tham gia quan hệ ở bên nhiều thì phải tồn tại bên một.

Không thể xoá một bản ghi của bảng bên một nếu trong quan hệ đã tồn tại những bản ghi bên nhiều có quan hệ với bản ghi bên một đó.

Trường hợp vi phạm các quy tắc trên thì sẽ nhận được thông báo lỗi.

8.4.1. Thiết lập thuộc tính tham chiếu toàn vẹn trong quan hệ

Trong khi chọn mối quan hệ giữa các bảng, có 2 thuộc tính tham chiếu toàn vẹn đó là *Cascade update related fields,Cascade Delete related records*, có thiết lập 2 thuộc tính này.

Nếu chọn thuộc tính *Cascade update related fields*, khi dữ liệu trên khoá chính của bảng bên một thay đổi thì Access sẽ tự động cập nhật sự thay đổi đó vào các trường tương ứng (có quan hệ) trên các bảng bên Nhiều, hay nói cách khác, dữ liệu ở bảng bên nhiều cũng thay đổi theo.

Nếu chọn thuộc tính *Cascade Delete related records*, khi dữ liệu trên bảng bên một bị xoá thì dữ liệu trên bảng bên nhiều cũng sẽ bị xoá..

8.4.2. Kiểu kết nối (Join type)

Trong quá trình thiết lập quan hệ giữa các bảng, nếu không chọn nút Create, chọn nút join type để chọn kiểu liên kết

Mục 1: Liên kết nội (Inner join)

Mục 2 và mục 3 là liên kết ngoại (outer join)

8.4.3. Điều chỉnh các mối quan hệ

Mở cửa sổ quan hệ (Tools/Relationship)

Click chuột phải, chọn edit relationship

8.4.4. Xoá các mối quan hệ

Mở cửa sổ quan hệ (Tools/Relationship)

Chọn mối quan hệ giữa các bảng, nhấn delete.

9. SẮP XẾP VÀ LỌC DỮ LIỆU

9.1. Một số phép toán và hàm

9.1.1. Một số phép toán

Ký tự thay thế: ? : Thay thế cho một ký tự bất kỳ

Ký tự * : Thay thế cho một dãy các ký tự.

Phép toán **Like**: Giống như

In: Kiểm tra một giá trị có thuộc một tập các giá trị hay không?

Is Null: Giá trị của một trường là Null.

Is not Null:Giá trị của một trường là không Null.

Between.....and: Kiểm tra xem một giá trị có thuộc một "đoạn" nào đó hay không?

9.1.2. Một số hàm

Hàm **Left$(<Chuỗi>,<n>)**: Trích bên trái chuỗi n ký tự.

Hàm **Right$(<Chuỗi>,<n>)**: Trích bên phải chuỗi n ký tự.

Hàm **Ucase(<Chuỗi>)**: Trả lại một chuỗi in hoa.

Hàm **Lcase(<Chuỗi>)**: Trả lại một chuỗi in thường.

Hàm **IIF(<Điều kiện>,<Giá trị 1>,< Giá trị 2>)**: Nếu <Điều kiện> nhận giá trị true thì hàm trả lại <Giá trị 1>, ngược lại hàm trả lại <Giá trị 2>.

9.2. Sắp xếp dữ liệu

9.2.1. sắp xếp trên một trường

Đặt con trỏ tại trường cần sắp xếp

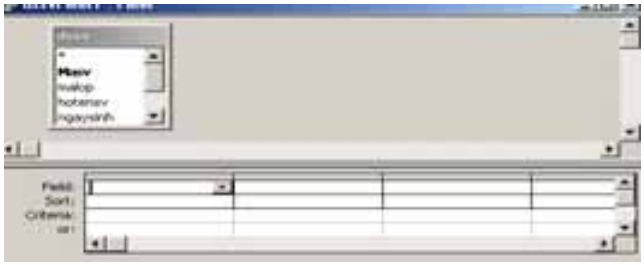
Thực hiện lệnh Records/ Sort/ Sort Ascending (Nếu sắp xếp tăng dần)

/ Sort Descending (Nếu sắp xếp giảm dần)

9.2.2. sắp xếp trên nhiều trường

Mở bảng trong chế độ Datasheet rồi thực hiện lệnh

Records/ Filter/ Advanced Filter sort



Trong hàng Field: Chọn các trường cần sắp xếp (Thứ tự ưu tiên từ trái sang phải)

Trong hàng Sort: Chọn tiêu chuẩn sắp xếp.

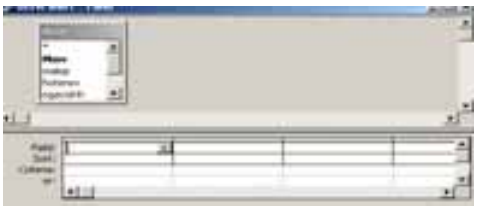
Trong hàng Criteria: Chọn điều kiện sắp xếp (nếu có).

Xem kết quả.: Chọn Filter/Apply filter.

9.3. Lọc dữ liệu

Mở bảng trong chế độ Datasheet rồi thực hiện lệnh

Records/ Filter/ Advanced Filter sort



Trong hàng Field: Chọn các trường làm tiêu chuẩn lọc dữ liệu

Trong hàng Criterie: Chọn tiêu chuẩn lọc dữ liệu.

Thực hiện lệnh Filter/ Apply filter sort để xem kết quả..

Chương 3 TRUY VẤN (QUERY)

Sức mạnh thực sự của CSDL là khả năng tìm đúng và đầy đủ thông tin mà chúng ta cần biết, trình bày dữ liệu sắp xếp theo ý muốn. Để đáp ứng yêu cầu trên, Acces cung cấp một công cụ truy vấn cho phép đặt câu hỏi với dữ liệu đang chứa bên trong các bảng trong CSDL.

1. KHÁI NIỆM TRUY VẤN

Truy vấn là một công cụ cho phép đặt câu hỏi với dữ liệu trong bảng dữ liệu trong CSDL.

Loại truy vấn thông dụng nhất là truy vấn chọn (Select Query). Với kiểu truy vấn này chúng ta có thể xem xét dữ liệu trong các bảng, thực hiện phân tích và chỉnh sửa trên dữ liệu đó, có thể xem thông tin từ 1 bảng hoặc có thể thêm nhiều trường từ nhiều bảng khác nhau.

Ví dụ:

Cho 2 bảng dữ liệu KHOHANG (**MAHANG**, TENHANG, GIA) BANHANG(MAHANG, TENKHACH, SOLUONG, NGAYMUA). Hãy hiển thị những khách hàng mua hàng trong tháng 7 bao gồm các thông tin: MAHANG, TENHANG, GIA, TENKHACH.

Sau khi thực hiện truy vấn, dữ liệu thỏa mãn yêu cầu được rút ra và tập hợp vào một bảng kết qủa gọi là **Dynaset** (Dynamic set). Dynaset cũng hoạt động như 1 bảng (Table) nhưng nó không phải là bảng vfa kết quả khi hiển thị có thể cho phép sửa đổi.

Một loại bảng thể hiện kết quả truy vấn khác là **Snapshot**, nó tương tự như dynaset tuy nhiên không thể sửa đổi thông tin (Như truy vấn Crosstab....).

1.1. Các loại truy vấn trong Access

Select Query : Truy vấn chọn

Crosstab Query : Truy vấn tham khỏa chéo (Thể hiện dòng và cột)

Action Query : Truy vấn hành động gồm

Truy vấn tạo bảng (make table Query)

Truy vấn nối (append Query)

Truy vấn cập nhật (Update Query)

Truy vấn xóa dữ liệu (Delete Query)

SQL Query : Truy vấn được viết bởi ngôn ngữ SQL.

Pass throught Query : Gởi các lệnh đến một CSDL SQL như Microsoft SQL server.

1.2. Sự cần thiết của truy vấn

Khi đứng trước một vấn đề nào đó trong CSDL, nếu sử dụng công cụ truy vấn thì có thể thực hiện được các yêu cầu sau:

Sự lựa chọn các trường cần thiết.

Lựa chọn những bản ghi.

Sắp xếp thứ tự các bản ghi.

Lấy dữ liệu chứa trên nhiều bảng khác nhau trong CSDL.

Thực hiện các phép tính.

Sử dụng truy vấn làm nguồn dữ liệu cho một biểu mẫu (Form), báo cáo (report) hoặc một truy vấn khác (Query).

Thay đổi dữ liệu trong bảng.

2. CÁC CHẾ ĐỘ HIỂN THỊ TRUY VẤN

2.1. Cửa sổ thiết kế truy vấn (Design view).

Trong chế độ này, người sử dụng có thể tạo, sửa chữa một truy vấn nào đó. Màn hình truy vấn chứa hai phần, phần thứ nhất chứa các bảng (hoặc truy vấn) tham gia truy vấn, phần thứ hai gọi là vùng lưới QBE (Query By Example).

2.2. Cửa sổ hiển thị truy vấn (DataSheet view).

Sử dụng chế độ này để xem kết quả.

2.3. Cửa sổ lệnh SQL (SQL view).

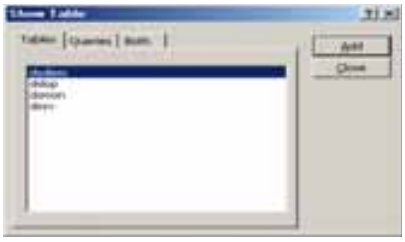
Sử dụng chế độ này để xem mã lệnh của truy vấn đang tạo

3. TẠO TRUY VẤN

3.1.Tạo mới 1 truy vấn

Từ cửa sổ Database, click vào đối tượng Queries.

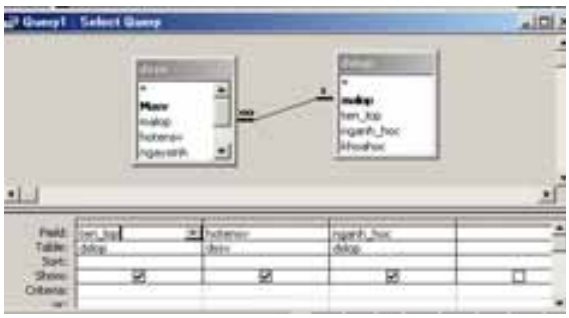
Chọn nút New.



Chọn Design View, chọn OK

Trong bảng Show Table chọn tables để hiển thị các bảng, chọn các bảng tham gia vào truy vấn và nhấn nút **Add**, sau đó nhấn ***Close***. (Nếu chọn Queries thì hiển thị truy vấn, chọn both thì hiển thị cả truy vấn và bảng dữ liệu).

Đưa các trường từ các bảng vào tham gia truy vần bằng cách kéo các trường và thả vào hàng **Field** trong vùng lưới QBE.



Trong hàng Sort: Sắp xếp dữ liệu (nếu có)

Trong hàng **Criteria** đặt tiêu chuẩn (nếu có)

Lưu truy vấn.

☞ ***Chú ý***

Mỗi truy vấn có:

Tối đa là 32 bảng tham gia.

Tối đa là 255 trường.

Kích thước tối đa của bảng dữ liệu (do truy vấn tạo ra) là 1 gigabyte.

Số trường dùng làm khóa sắp xếp tối đa là 10.

Số truy vấn lồng nhau tối đa là 50 cấp.

Số ký tự tối đa trong ô của vùng lưới là 1024.

Số ký tự tối đa trong dòng lệnh SQL là 64000.

Số ký tự tối đa trong tham số là 255.

3.2. Thay đổi thứ tự, xóa các trường

Các trường trong truy vấn sẽ hiển thị theo thứ tự như xuất hiện trong vùng lưới QBE.

3.2.1.Thay đổi thứ tự của trường

Đưa con trỏ vào thanh chọn sao cho con trỏ biến thành hình mũi tên trỏ xuống

Click để chọn trường

Drag để thay đổi vị trí.

3.2.2.xoá trường

Đưa con trỏ vào thanh chọn sao cho con trỏ biến thành hình mũi tên trỏ xuống

Click để chọn trường

Nhấn phím delete (Nếu muốn xoá tất cả các trường trong vùng lưới QBE: chọn Edit/clear grid)

3.3. Thể hiện hoặc che dấu tên bảng trong vùng lưới QBE

Muốn biết tên trường hiện tại trong vùng lưới QBE là của tên bảng nào, tại chế độ Design View người sử dụng thực hiện View/tables name.

3.4. Xem kết quả của truy vấn.

Tại cửa sổ Database chọn tên truy vấn rồi chọn Open, hoặc trong khi thiết kế truy vấn thực hiện lệnh View/datasheet View.

3.5. Đổi tiêu đề cột trong truy vấn.

Đổi tên tiêu đề cột trong truy vấn mục đích là làm cho bảng kết xuất dễ đọc hơn (Trừ khi đã quy định thuộc tính Caption).

Muốn đổi tên tiêu đề cột thực hiện theo các bước sau:

Mở truy vấn ở chế độ Design View

Chọn vào bên trái ký tự đầu tiên của tên trường trong vùng lưới QBE

Gõ vào tên mới, theo sau là dấu 2 chấm (:).

Ví dụ:

3.6. Định thứ tự sắp xếp

Có thể sử dụng nhanh trong chế độ datasheet View .

Có thể tạo sắp xếp trong khi thiết kế truy vấn bằng cách chọn Ascending (tăng dần) hoặc Descending (giám dần) trong hàng Sort của vùng lưới QBE .

☞ ***Chú ý:*** Nếu có nhiều trường định vị sắp xếp thì theo thứ tự ưu tiên từ trái sang phải.

3.7. Che dấu hay thể hiện các trường trong Dynaset

Tại hàng Show ứng với trường cần che dấu chúng ta không chọn mặt dù nó vẫn tồn tại, vẫn tham gia truy vấn.

3.8. Mối quan hệ giữa thuộc tính của trường trong truy vấn và trong bảng dữ liệu

Theo mặc nhiên, các trường trong truy vấn kế thừa tất cả các thuộc tính của trường trong bảng làm nguồn dữ liệu. Nếu không quy định lại trong truy vấn, các trường trong Dynaset hoặc snapshot luôn kế thừa các thuộc tính của bảng làm nguồn dữ liệu. Nếu thay đổi thiết kế trong bảng làm nguồn dữ liệu và thay đổi thuộc tính của các trường thì thuộc tính này cũng được thay đổi trong truy vấn. Tuy nhiên, nếu quy định lại các thuộc tính cho các trường trong truy vấn thì các thuộc tính của các trường trong bảng làm nguồn dữ liệu không thay đổi.

4. THIẾT KẾ TRUY VẤN CHỌN

4.1. Định nghĩa truy vấn chọn

Truy vấn chọn là loại truy vấn được chọn lựa, rút trích dữ liệu từ các bảng dữ liệu thỏa mãn một hoặc nhiều điều kiện nào đó. Khi thực hiện truy vấn chọn, Access tác động lên dữ liệu và thể hiện các bản ghi thoả mãn các điều kiện đăt ra trong một bảng kết quả gọi là Recordset.

4.2. Lập phép chọn trong truy vấn

4.2.1. Chọn một nhóm các bản ghi thỏa mãn một điều kiện nào đó

Muốn thực hiện các phép chọn trong khi thể hiện truy vấn người ta thường sử dụng các phép toán sau:

Phép toán	Ví dụ	Ý nghĩa
<	<#20/10/99#	Trước ngày 20/10/99
>	>#10/10/98#	Sau ngày 10/10/98
>=	>= #05/05/90#	Sau và trong ngày 05/05/90
<>	<>#01/01/99#	Khác ngày 01/01/99
=	= #10/10/97#	Trong ngày 10/10/97
Between .... and ...	Between #1/2/97# and #1/7/97#	Từ ngày 1/2/97 đến 1/7/97

Ví dụ:

Cho 2 bảng dữ liệu Dslop(**Malop**, Tenlop, Nganh_hoc, khoahoc)

Dssv(Masv, malop, hotensv, ngaysinh, quequan, giotinh, hocbong)

Tạo một truy vấn để hiển thị danh sách những sinh viên có ngaysinh trong khoảng thời gian từ 05/05/75 đến 05/05/79 bao gồm các trường: Tenlop, Hotensv, Ngaysinh, nganh_hoc.

Tạo truy vấn chọn và đưa 2 bảng dslop và dssv vào tham gia truy vấn

Đưa các trường Tenlop, hotensv, ngaysinh, nganh_hoc vào vùng lưới QBE

Trong hàng Criteria của trường Ngaysinh: Between #05/05/75# and #05/05/79#

4.2.2. Ký tự thay thế

Ký tự * : Thay thế một nhóm ký tự bất kỳ.

Ký tự ? : Thay thế 1 ký tự.

Ký tự [ ] : Thay thế các ký tự trong ngoặc vuông.

Ký tự ! : Phủ định.

Ký tự - : Từ ký tự đến ký tự.

Ví dụ

Cho 2 bảng dữ liệu Dslop(**Malop**, Tenlop, Nganh_hoc, khoahoc)

Dssv(Masv, malop, hotensv, ngaysinh, quequan, giotinh, hocbong)

Tạo một truy vấn để hiển thị danh sách những sinh viên có Tenlop bắt đầu là **"T"** bao gồm các trường: Tenlop, Hotensv, Ngaysinh, nganh_hoc.

Tạo truy vấn chọn và đưa 2 bảng dslop và dssv vào tham gia truy vấn

Đưa các trường Tenlop, hotensv, ngaysinh, nganh_hoc vào vùng lưới QBE

Trong hàng Criteria của trường Tenlop: Like "T*"

4.2.3. Chọn các bản ghi không phù hợp với một giá trị nào đó

Dùng toán tử **Not**

Ví dụ: Tạo một truy vấn để hiển thị danh sách những sinh viên có Tenlop không bắt đầu là **"T"** bao gồm các trường: Tenlop, Hotensv, Ngaysinh, nganh_hoc.

4.2.4. Định nhiều tiêu chuẩn trong lựa chọn

Dùng phép “Và” và phép “Hoặc” trong một trường

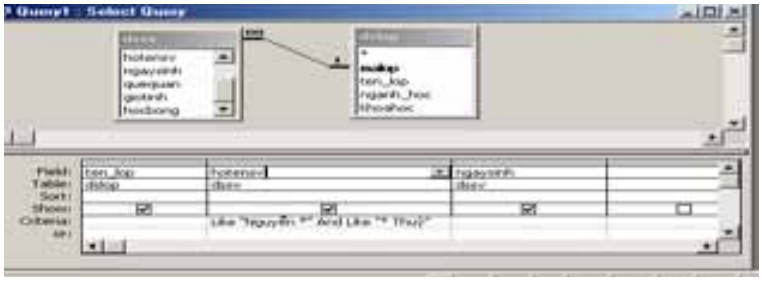
Muốn quy định nhiều tiêu chuẩn trong cùng một trường, chúng ta phải sử dụng toán tử AND (và) cùng toán tử OR (hoặc).

Ví dụ: Tạo một truy vấn để hiển thị danh sách những sinh viên có Họ là "**Nguyễn"** và Tên "**Thuỷ"** bao gồm các trường: Tenlop, Hotensv, Ngaysinh.

Tạo truy vấn chọn và đưa 2 bảng dslop và dssv vào tham gia truy vấn

Đưa các trường Tenlop, hotensv, ngaysinh vào vùng lưới QBE

Trong hàng Criteria của trường Hotensv : Like "Nguyễn *" and "* Thuỷ"



Dùng phép “Và” và phép “Hoặc”trên nhiều trường

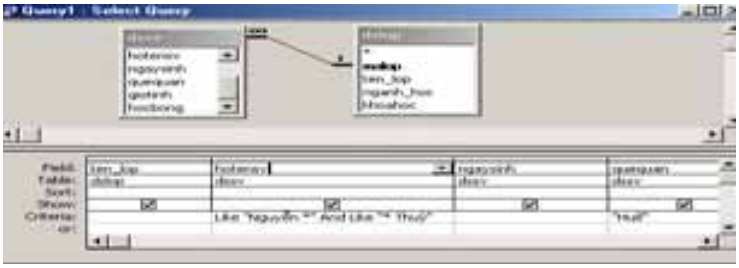
Ví dụ: Tạo một truy vấn để hiển thị danh sách những sinh viên có Họ là "**Nguyễn"** và tên "**Thuỷ**" và có quê quán ở "**Huế"** bao gồm các trường: Tenlop, Hotensv, ngaysinh Quequan.

Tạo truy vấn chọn và đưa 2 bảng dslop và dssv vào tham gia truy vấn

Đưa các trường Tenlop, hotensv, ngaysinh, quequan vào vùng lưới QBE

Trong hàng Criteria của trường Hotensv : Like "Nguyễn *" and "* Thuỷ"

Quequan : Huế



Tạo một truy vấn để hiển thị danh sách những sinh viên có Họ là "**Lê"** hoặc có quê quán ở "**Đà Nẵng** bao gồm các trường: Tenlop, Hotensv, Ngaysinh, Quequan.

Tạo truy vấn chọn và đưa 2 bảng dslop và dssv vào tham gia truy vấn

Đưa các trường Tenlop, hotensv, ngaysinh , Quequanvào vùng lưới QBE

Trong hàng Criteria của trường Hotensv : Like "Lê *"

Trong hàng or của trường Quequan : Đà Nẵng

4.2.5. Chọn các bản ghi có chứa có giá trị

Chúng ta có thể chọn các bản ghi có chứa hoặc không chứa giá trị, chẳng hạn như tìm những sinh viên mà không có số điện thoại nhà ở.

Access cung cấp 2 phép toán

Phép toán	Ý nghĩa
IS NULL	Trường không chứa giá trị
IS NOT NULL	Trường có chứa giá trị

Ví dụ:

Tạo một truy vấn để hiển thị danh sách những sinh viên không có học bổng bao gồm các trường: Tenlop, Hotensv, hocbong.

4.2.6. Chọn các bản ghi thuộc danh sách các giá trị nào đó

Chúng ta có thể sử dụng phép toán **IN** (***Danh sách giá trị***)

Ví dụ

Tạo một truy vấn để hiển thị danh sách những sinh viên thuộc lớp "**Tin K23**" hoặc "**Tin K24**" hoặc "**Tin K25**" bao gồm các trường: Tenlop, Hotensv, hocbong.

4.2.7. Tham chiếu đến các trường khác

Nếu trong biểu thức chọn của truy vấn, các tính toán tham chiếu đến các trường phải đặt trong dấu [ ], trường hợp tham chiếu đến trường của bảng khác phải chỉ rõ bảng nguồn của nó. **[Tên bảng]![Tên trường].**

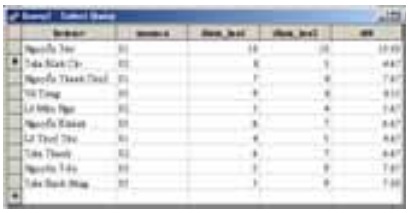
4.2.8. Tạo trường kiểu biểu thức

Ví dụ

Cho 2 bảng dữ liệu Dssv(**Masv**, hotensv, ngaysinh, quequan, gioitinh)

Dsdiem(**Masv**, mamon, diem_lan1, diem_lan2)

Tạo truy vấn để hiển thị các thông tin: Hotensv, mamon, diem_lan1, diem_lan2, dtb, trong đó dtb=(diem_lan1+ diem_lan2*2)/3.



☞ Chú ý

Sau khi thực hiện truy vấn chúng ta không thể thay đổi giá trị trong trường kiểu biểu thức, tuy nhiên nếu thay đổi giá trị trong trường tham gia biểu thức thì kết quả trong trường kiểu biểu thức cũng thay đổi theo.

4.2.9 .Chọn giá trị duy nhất

Theo mặc định, access sẽ chọn tất cả các bản ghi thoã mãn điều kiện, tuy nhiên đôi khi có nhiều giá trị giống nhau được lặp đi lặp lại, do đó để cô đọng dữ liệu thì chúng ta có thể quy định thuộc tính duy nhất trong khi hiển thị

Thuộc tính Unique-values

Chọn Yes: Không thể hiện các giá trị trùng nhau

Chọn No: Thể hiện các giá trị trùng nhau

Thuộc tính Unique-Records

Chọn Yes: Không thể hiện các bản ghi trùng nhau

Chọn No: Thể hiện các bản ghi trùng nhau

4.2.10. Chọn các giá trị đầu

Khi hiển thị truy vấn đôi khi chúng ta muốn hiển thị một số bản ghi đầu tiên nào đó thoả mãn các điều kiện thì sử dụng thuộc tính **Top values**.

5. TRUY VẤN DỰA TRÊN NHIỀU BẢNG DỮ LIỆU

Để tạo truy vấn dựa trên nhiều bảng dữ liệu thì các bảng đó phải được thiết lập mối quan hệ, nếu các bảng không thiết lập mối quan hệ thì khi truy vấn dữ liệu access sẽ cho ra những bộ dữ liệu là tích Đề-Các giữa các bộ dữ liệu trong các bảng.

5.1. Liên kết các bảng trong truy vấn

Khi các bảng dữ liệu được thiết lập mối quan hệ thì trường nối với nhau gọi là trường liên kết, trong access phân biệt 3 loại liên kết sau

5.1.1. Liên kết nội (Inner join)

Đây là loại liên kết rất phổ biến nhất giữa 2 bảng dữ liệu. Trong đó dữ liệu khi thể hiện trên Dynaset sẽ gồm những bản ghi mà dữ liệu chứa trong trường liên kết ở hai bảng phải giống nhau hoàn toàn.

5.1.2. Liên kết ngoại (Outer join)

Đây là loại liên kết cho phép dữ liệu thể hiện trên Dynaset của một trong hai bảng tham gia có nội dung trường liên kết không giống nội dung trong trường tương ứng của bảng còn lại. Liên kết ngoại được chia làm hai loại

Left Outer Join: Trong kiểu liên kết này, dữ liệu ở bảng bên "1" thể hiện toàn bộ trên Dynaset và chỉ những bản ghi bên bảng "nhiều" có nội dung trong trường liên kết giống trường tương ứng bên bảng "1".

Right Outer Join: Trong kiểu liên kết này, dữ liệu ở bảng bên "nhiều" thể hiện toàn bộ trên Dynaset và chỉ những bản ghi bên bảng "1" có nội dung trong trường liên kết giống trường tương ứng bên bảng "nhiều".

5.1.3. Tự liên kết (Seft join)

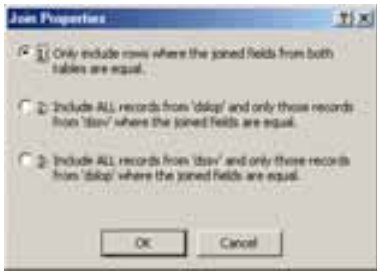
Là kiểu liên kết của một bảng dữ liệu với chính nó. Trong đó một bản ghi trong bảng dữ liệu sẽ liên kết với những bản ghi khác trong bảng dữ liệu đó. Tự liên kết có thể hiểu như là liên kết nội hay liên kết ngoại từ một bảng vào một bảng sao chính nó. Để thực hiện việc tạo tự liên kết chúng ta phải đưa một bảng vào tham gia truy vấn 2 lần.

5.2.Tạo liên kết ngoại

Muốn tạo liên kết ngoại giữa 2 bảng dữ liệu ta thực hiện

Tools/Relationships

Double click vào đường liên kết giữa 2 bảng dữ liệu, chọn Join Type



Trong hộp thoại Join Properties chọn mục 2 hoặc mục 3.

5.3.Tạo một tự liên kết

Để tạo một tự liên kết chúng ta thực hiện

Tạo truy vấn mới và đưa bảng dữ liệu vào truy vấn 2 lần

Tạo các liên kết

Ví dụ:

Cho bảng Dsdiem(Masv, Hoten, Diem_lan1, Diem_lan2)

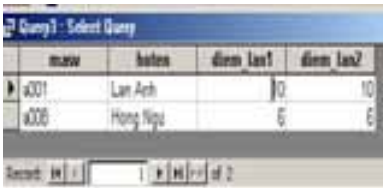
Tạo truy vấn để hiển thị danh sách những sinh viên có điểm thi Lần 1 bằng điểm thi lần 2...

Tạo truy vấn mới và đưa bảng Dsdiem vào tham gia truy vấn 2 lần

Tạo liên kết nội (Inner join) giữa 2 trường Masv

Tạo liên kết nội từ trường Diem_lan1 vào Diem_lan2

Đưa các trường vào vùng lưới QBE và xem kết quả



5.3. Tự động tìm kiếm dữ liệu (Auto lookup)

Khi nhập dữ liệu vào Dynaset, chức năng tự động tìm kiếm dữ liệu cho phép chúng ta chỉ nhập dữ liệu ở các trường của bảng bên "nhiều" (Ở quan hệ 1-∞) còn Acces sẽ tự động tìm kiếm dữ liệu tương ứng trên bảng "1" để hiển thị.

Chức năng Auto Lookup hoạt động trong các truy vấn mà hai bảng tham gia có mối quan hệ 1-∞.

Tạo một truy vấn có sử dụng chức năng Auto Lookup chúng ta thực hiện:

Tạo truy vấn và đưa 2 bảng vào tham gia truy vấn

Đưa trường liên kết cuả bảng bên nhiều vào vùng lưới QBE

Đưa các trường cần hiển thị dữ liệu của bảng bên "1".

☞ ***Chú ý:*** Khi nhập dữ liệu chỉ nhập dữ liệu ở các trường của bảng "nhiều"

6. TÍNH TỔNG TRONG TRUY VẤN CHỌN

Trong thực tế, chúng ta thường có những câu hỏi đặt ra về việc nhóm dữ liệu nào đó, chẳng hạn trong tháng 10 công ty xăng dầu XYZ bán được bao nhiêu lít xăng, tổng thành tiền bao nhiêu?

Trong Access chúng ta có thể thực hiện một số phép tính lên một nhóm bản ghi bằng cách dùng truy vấn tính tổng

Một số phép toán thường sử dụng

Phép toán	Ý nghĩa
Sum	Tính tổng các giá trị của một trường
Avg	Tính giá trị trung bình của một trường
Min	Tính giá trị nhỏ nhất của một trường
Max	Tính giá trị lớn nhất của một trường
Count	Đếm số giá trị khác rỗng có trong một trường
First	Giá trị của trường ở bản ghi đầu tiên trong bảng
Last	Giá trị của trường ở bản ghi cuối cùng trong bảng
Where	Giới hạn điều kiện khi tính tổng
Expression	Trường kiểu biểu thức

☞ ***Chú ý:*** Khi thực hiện truy vấn Total, dữ liệu trong bảng kết quả của nó trình bày không thể chỉnh sửa.

6.1. Tạo truy vấn tính tổng

Tạo truy vấn chọn và đưa các bảng vào tham gia truy vấn

Thực hiện lệnh: View/Totals

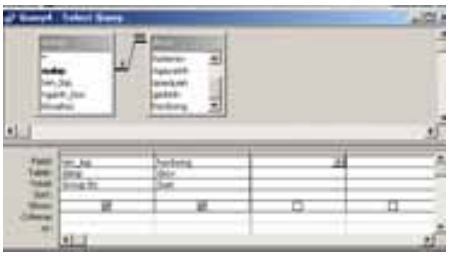
Trong vùng lưới QBE:

Tại hàng Field chọn các trường

Total chọn các phép toán tương ứng.

Criteria: Chọn điều kiện giới hạn tính tổng (Nếu có)

Lưu và thực hiện truy vấn



6.2. Tính tổng của tất cả các bản ghi

Tạo truy vấn chọn.

Đưa các bảng cần thiết vào truy vấn.

Đưa các trường cần thiết vào vùng lưới QBE

Chọn menu View/Totals, dòng Total sẽ xuất hiện trên vùng lưới.

Trong hàng total của mỗi trường chọn phương pháp tính tổng (Sum, Avg, count...). Vì đang tính tổng của tất cả các bản ghi nên không được phép chọn “Group by” ở bất kỳ trường nào.

Chuyển sang DataSheet View để xem kết quả. (View/ Datasheet View)

Ví dụ:

Để quản lý các mặt hàng bán ra trong một cửa hàng người ta sử dụng 2 bảng dữ liệu như sau:

Dshang(**Mahang**, tenhang, dongia)

Dskhach(Mahang, tenkhach, ngaymua, diachi, soluong, thanhtien)

Tạo truy vấn để thống kê xem trong cửa hàng bán bao nhiêu mặt hàng và trung bình đơn giá của mỗi mặt hàng là bao nhiêu?

Tạo truy vấn và đưa bảng Dshang vào tham gia truy vấn

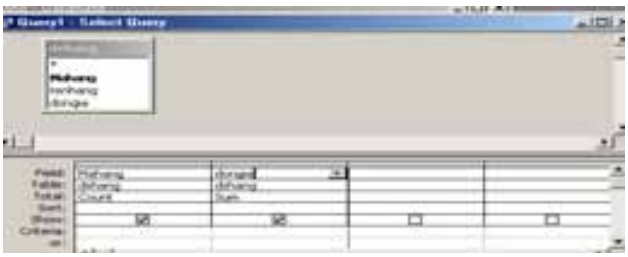
Đưa 2 trường Mahang và dongia vào vùng lưới QBE.

Chọn View/Totals

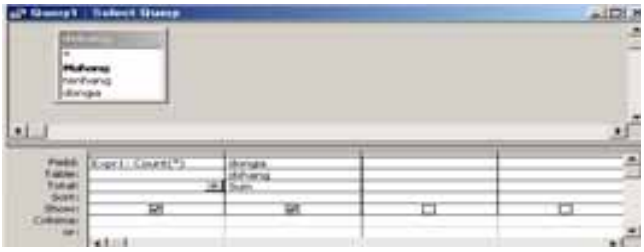
Trong hàng Total của trường Mahang chọn phép toán Count

Trong hàng Total của trường Dongia chọn phép toán Avg

Chọn View/Datasheet View để xem kết quả.



☞ ***Chú ý:*** Có thể đếm số bản ghi bằng cách dùng Count(*)



6.3. Tính tổng trên từng nhóm bản ghi

Trong nhiều trường hợp chúng ta có thể tính toán trên một nhóm bản ghi nào đó. Chúng ta quy định khi thiết kế truy vấn những trường nào sẽ được tính theo nhóm, trường nào sẽ được tính tổng.

Tạo truy vấn

Đưa các bảng tham gia vào truy vấn

Đưa các trường vào vùng lưới

Chọn View/Totals

Tại hàng total

Chọn “Group by” cho trường làm khóa để nhóm

Chọn các phép toán tính tổng (Sum,count..) cho các trường còn lại

Chọn View/Datasheet View để xem kết quả.

Ví dụ:

Tạo một truy vấn để tính tổng soluong, thanhtien của mỗi mặt hàng bán được là bao nhiêu?

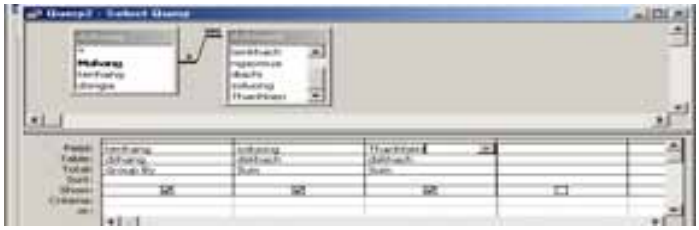
Tạo truy vấn và đưa 2 bảng dshang và dskhach vào tham gia truy vấn

Chọn View/Totals

Đưa các trường Tenhang, soluong, thanhtien vào vùng lưới QBE.

Tại hàng Total của trường Tenhang: Chọn phép toán Group by

Tại hàng Total của trường Soluong, thanhtien: Chọn phép toán SUM.



Chọn View/Datasheet View để xem kết quả.

Query2 : Select Query

tenhang	SumOfsoluong	SumOfThanhtien
Dau hoa	35	140000
Xang A83	83	415000
XangA92	78	468000

Record: 3 of 3

Ta có thể thay đổi tiêu đề cột trong khi thực hiện truy vấn tính tổng như sau

tenhang	Tong so luong	Tong thanh tien
Dau hoa	35	140000
Xang A83	83	415000
XangA92	78	468000

6.4. Tính tổng trên nhiều nhóm bản ghi

Access cho phép tính tổng không chỉ trên một mà còn nhiều nhóm bản ghi. Để làm được điều đó chúng ta chọn “Group by” trên nhiều trường và khi thực hiện

Access sẽ theo thứ tự từ trái sang phải trường bên trái là nhóm mức cao hơn, trường kế tiếp theo là nhóm mức thấp hơn.

Ví dụ:

Tạo truy vấn để tính tổng thanhtien của mỗi mặt hàng bán được theo từng năm nào đó?

Tạo truy vấn và đưa 2 bảng dshang và dskhach vào tham gia truy vấn

Chọn View/Totals

Đưa các trường tenhang, ngaymua, thanhtien vào vùng lưới QBE

Tại hàng Total của trường tenhang chọn phép toán Group by

Tại hàng field của trường ngayban sử dụng hàm year([ngayban]) và tại hàng total chọn phép toán Group by.

Tại hàng Total của trường Thanhtien chọn phép toán Sum.

Thay đổi tiêu đề cột trong truy vấn.

Chọn View/ Datasheet View để xem kết quả

Query4 : Select Query

tenhang	Nam	Tong thanh tien
Dau hoa	1997	24000
Dau hoa	1998	88000
Dau hoa	1999	28000
Xang A83	1997	45000
Xang A83	1998	250000
Xang A83	1999	120000
XangA92	1997	120000
XangA92	1998	168000
XangA92	1999	180000

ecord: 3 of 9

6.5. Lập biểu thức chọn cho các trường dùng để nhóm khi tính tổng

Cũng như với những truy vấn khác, chúng ta có thể lập biểu thức chọn cho truy vấn tính tổng theo từng nhóm.

Để thực hiện công việc này, chúng ta lập biểu thức điều kiện ngay hàng Criteria của trường “group by”.

Ví dụ:

Tạo truy vấn để tính tổng thanhtien của mỗi mặt hàng bán được trong năm 1999.

Tạo truy vấn và đưa 2 bảng dshang và dskhach vào tham gia truy vấn

Chọn View/Totals

Đưa các trường tenhang, ngaymua, thanhtien vào vùng lưới QBE

Tại hàng Total của trường tenhang chọn phép toán Group by

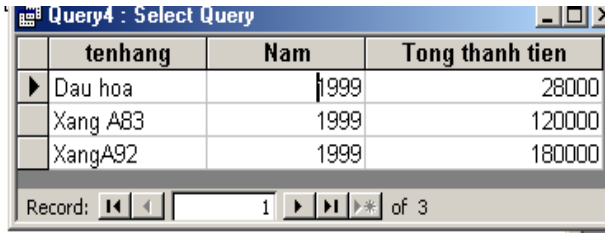
Tại hàng field của trường ngayban sử dụng hàm year([ngayban]) và tại hàng total chọn phép toán Group by.

Tại hàng Total của trường Thanhtien chọn phép toán Sum.

Tại hàng Criteria của trường ngaymua: gõ vào giá trị 1999

Thay đổi tiêu đề cột trong truy vấn.

Chọn View/ Datasheet View để xem kết quả



Query4 : Select Query

tenhang	Nam	Tong thanh tien
Dau hoa	1999	28000
Xang A83	1999	120000
XangA92	1999	180000

Record: 1 of 3

6.6. Lập biểu thức chọn để giới hạn những bản ghi.

Trong các phần trước, chúng ta đã biết cách giới hạn các bản ghi trong truy vấn theo một điều kiện nào đó, sự giới hạn này gọi là **giới hạn sau khi tính tổng.**

Bây giờ chúng ta lập biểu thức chọn giới hạn số bản ghi trước khi đưa vào tính tổng trong truy vấn gọi là **giới hạn trước khi tính tổng** .

Cách tạo

Tạo truy vấn mới và đưa các bảng tham gia vào truy vấn

Đưa các trường vào vùng lưới QBE.

Chọn menu View/Total

Thiết lập hàng Total thành Where đối với trường chúng ta muốn dùng để đặt biểu thức điều kiện giới hạn số bản ghi trước khi tính tổng.

Gõ biểu thức điều kiện tại hàng Criteria tương ứng.

Chuyển sang chế độ datasheet view để xem kết quả.

Ví dụ:

Tạo truy vấn để tính tổng soluong, thanhtien của mỗi mặt hàng bán được đối với khách mua hàng có Queuqan ở "**Huế**"

☞ ***Chú ý:*** Trong đa số trường hợp, đặt điều kiện lọc trước và sau khi tính tổng có giá trị khác nhau.

6.7. Dùng truy vấn để cập nhật bản ghi

Khi truy vấn chỉ dựa trên một bảng, hoậchi bảng có quan hệ 1-1 thì tất cả các trường đều có thể thay đổi, cập nhật. Trong trường hợp có nhiều hơn hai bảng tham gia truy vấn mà có quan hệ 1-∞ thì sẽ phức tạp hơn.

6.7.1. Khi nào dữ liệu trong trường có thể sửa đổi được

Bảng sau đây liệt kê các trường hợp khi nào một trường trong kết quả truy vấn hay trong biểu mẫu có thể sửa đổi được.

Loại truy vấn hay truờng	Dữ liệu trong trường có cho phép sửa đổi hay không?
Truy vấn dựa trên 1 bảng	Có
Truy vấn dựa trên nhiều bảng có quan hệ 1-1	Có

Truy vấn dựa trên nhiều bảng có quan hệ 1-∞	Thông thường
Truy vấn Tham khảo chéo	Không
Truy vấn tính tổng	Không
Truy vấn với thuộc tính Unique values được thiết lập thành Yes	Không
Truy vấn hội	Không
Truy vấn chuyển nhượng	Không
Trường kiểu biểu thức	Không
Trường trong bản ghi đã bị xoá hoặc bị khoá bởi một người khác trong môi trường nhiều người sử dụng	Không

6.7.1. Chỉnh sửa bản ghi trong truy vấn dựa trên hai bảng có quan hệ 1-∞

Trong truy vấn dựa trên dữ liệu là hai bảng có quan hệ 1-∞, chúng ta có thể sửa đổi tất cả các trường trừ trường liên kết của bảng bên "1". Tuy nhiên có hai trường hợp mà vẫn có thể sửa đổi dữ liệu trong trường liên kết bên "1" là:

Có thể sửa đổi dữ liệu trường liên kết bên bảng "1" trong trường hợp liên kết ngoại và trương tương ứng của liên kết bên bảng "nhiều" không chứa giá trị.

Có thể sửa đổi dữ liệu trường liên kết bên bảng "1" trong trường hợp đã khai báo thuộc tính tham chiếu toàn vẹn.

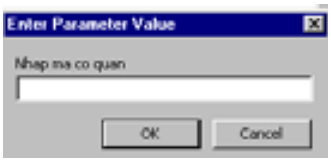
7. TRUY VẤN THAM SỐ (Parameter Query)

7.1. Khái niệm

Nếu thường xuyên chạy cùng một truy vấn, nhưng mỗi lần một tiêu chuẩn khác nhau, thay vì phải thiết kế lại truy vấn sau mỗi lần thực hiện, có thể tiết kiệm thời gian bằng cách tạo truy vấn tham số. Khi thực hiện loại này Access sẽ nhắc nhập điều kiện chọn trong hộp thoại enter parameter Value.

Ví dụ:

Giả sử thường xuyên chạy một truy vấn để liệt kê danh sách nhân viên của một cơ quan nào đó có mã cơ quan nhập vào bất kỳ.



☞ ***Chú ý:*** Nội dung các tham số mà chúng ta nhập vào có thể là hằng (số,chuỗi,ngày..) nhưng không được biểu thức.

7.2. Tạo truy vấn tham số

Tạo truy vấn chọn và đưa các bảng cần thiết vào tham gia truy vấn.

Kéo các trường cần thiết vào vùng lưới QBE.

Tại hàng Criteria gõ vào biểu thức có chứa tham số với chú ý tên tham số phải nằm giữa 2 dấu ngoặc vuông ([ ])

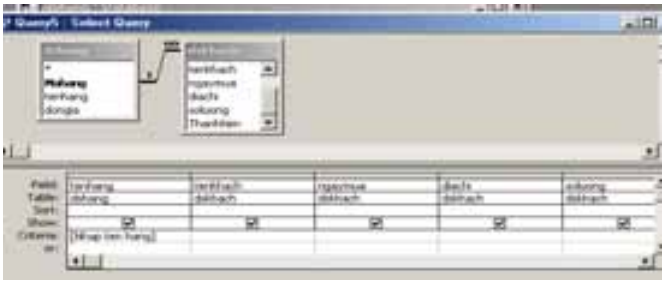
Tên tham số cũng là chuỗi nhắc nhở.Access cho phép có khoảng trắng và độ dài tối đa 255 ký tự.

Quy định kiểu dữ liệu cho tham số: Chọn queries/ parameter query.

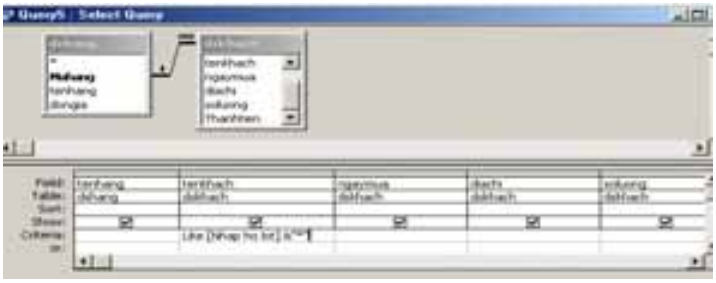
Trong hộp thoại query parameters: Trong mục Parameter chọn tham số, trong mục Data type chọn kiểu dữ liệu tương ứng.

Ví dụ:

Tạo truy vấn để hiển thị danh sách các khách hàng mua một mặt hàng nào đó (mặt hàng được nhập bất kỳ từ bàn phím).



Tạo truy vấn để hiển thị danh sách các khách hàng mua hàng có họ lót được nhập từ bàn phím.



7.3. Truy vấn nhiều tham số

Có thể tạo truy vấn , khi chạy truy vấn nhập nhiều dữ liệu cho điều kiện chọn lựa. Muốn vậy tạo truy vấn nhiều tham số.

Ví dụ:

Tạo một truy vấn hiển thị danh sách các khách hàng mua hàng trong khoảng thời gian nào đó (Thời gian được nhập từ bàn phím).

Tạo truy vấn chọn và đưa các 2 bảng dshang và dskhach vào tham gia truy vấn.

Kéo các trường tenhang, tenkhach, ngaymua vào vùng lưới QBE.

Tại hàng Criteria của trường NGAYSINH chọn:

Between [Từ ngày] and [Đến ngày]

Chọn query/Parameter khai báo kiểu dữ liệu cho 2 tham số là date/time.

Khi chạy truy vấn sẽ cho kết quả sau

7.4. Kết hợp giữa truy vấn tham số và truy vấn tính tổng

Trong nhiều bài toán quản lý người ta thường gặp những yêu cầu như: Hãy thống kê xem mỗi loại hàng trong một tháng nào đó bán được với tổng số lượng là bao nhiêu? Tổng thành tiền là bao nhiêu? (Tháng được nhập từ bàn phím). Vì vậy trước hết chúng ta phải thực hiện truy vấn tính tổng xong mới kết hợp truy vấn tham số.

Ví dụ:

Hãy tạo một truy vấn để hiển thị tổng thanhtien của mỗi mặt hàng bán được trong một năm nào đó (Năm được nhập từ bàn phím).

Tạo một truy vấn chọn, đưa bảng Dshang và dskhach vào tham gia truy vấn.

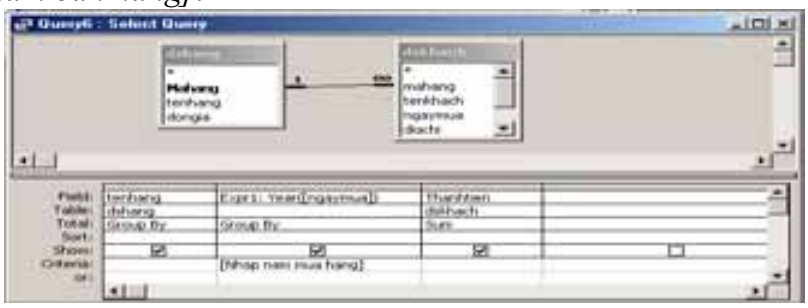
Đưa các trường tenhang, ngayban, thanhtien vào vùng lưới QBE.

Chọn View/Totals

Trong hàng Total: Chọn Group by đối với trường tenhang, và year([ngayban])

Trong hàng Criteria đối với trường Ngaysinh, ta chọn tham số sau:

[Nhap vao nam ban hang].



8. TRUY VẤN THAM KHẢO CHÉO (Crosstab query)

8.1. Khái niệm

Truy vấn tham khảo chéo là loại truy vấn dùng để tóm lược dữ liệu và trình bày kết quả theo dạng như một bảng tính. Truy vấn tham khảo chéo cũng có thể thống kê một khối lượng dữ liệu lớn và trình bày đơn giản hơn do đó thường sử dụng để so sánh dữ liệu.

8.2. Tạo truy vấn tham khảo chéo

Muốn tạo một truy vấn tham khảo chéo chúng ta phải xác định được 3 yếu tố chính: Trường làm tiêu đề cột (Duy nhất 1 trường), trường làm tiêu đề hàng (Có thể nhiều trường), trường tính giá trị (Duy nhất 1 trường).

Cách tạo

Tạo truy vấn chọn và đưa các bảng vào tham gia truy vấn

Đưa các trường vào vùng lưới QBE

Chọn Query/Crosstab

Quy định trường làm tiêu đề cột

Tại hàng Total: Bắt buộc chọn phép toán Group by

Tại hàng Crosstab: Chọn Column heading

Quy định trường làm tiêu đề hàng

Tại hàng Total: Ít nhất một trong các trường phải chọn phép toán Group by

Tại hàng Crosstab: Chọn Row heading

Quy định trường tính giá trị

Tại hàng Total: Chọn phép toán thích hợp

Tại hàng Crosstab: Chọn Value

Ví dụ:

Cho 2 bảng dữ liệu Dstruong(**matruong**, tentruong, sodt)

Danhsach(matruong, hoten, ngaysinh, gioitinh, xeploai)

Tạo một truy vấn Crosstab để phản ánh tổng số lượng sinh viên xếp mỗi loại của trong từng trường bao nhiêu.?

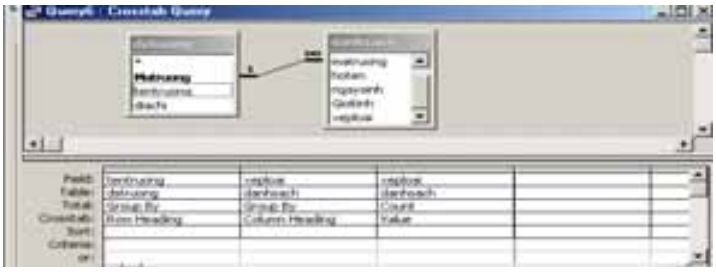
Tạo truy vấn và đưa 2 bảng dữ liệu vào tham gia truy vấn, đưa các trường tentruong và xeploai vào vùng lưới QBE. (Trường Xeploai đưa vào 2 lần)

Chọn Query/ crosstab query

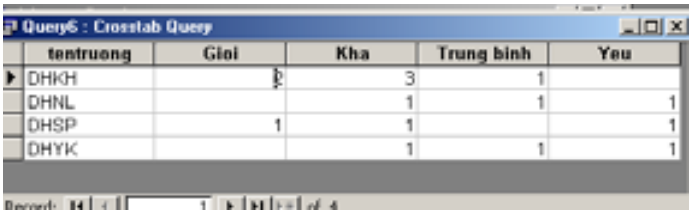
Tại hàng Total của trường tentruong: Chọn phép toán Group by, hàng crosstab: chọn Row heading

Tại hàng Total của truờng Xeploai: Chọn phép toán Group by, hàng Crosstab chọn Column heading.

Tại hàng Total của truờng Xeploai: Chọn phép toán Count, hàng Crosstab chọn Value.



Chọn View/ Datasheet View để xem kết quả



Query6 : Crosstab Query

tentruong	Gioi	Kha	Trung binh	Yeu
DHKH	2	3	1	
DHNL		1	1	1
DHSP	1	1		1
DHYK		1	1	1

8.3. Định dạng cho tiêu đề cột

Với truy vấn Crosstab, chúng ta có thể can thiệp nhiều hơn về cách trình bày tiêu đề cột trong bảng. Chúng ta có thể thay đổi bằng cách đặt lại thuộc tính Column Heading của truy vấn. Thuộc tính này cho phép chúng ta: Chỉ định sắp xếp các tiêu đề cột.

Muốn định dạng tiêu đề cột thực hiện các bước sau:

Tạo truy vấn Crosstab.

Chuyển sang chế độ Design View.

Mở bảng thuộc tính truy vấn.

Tại hàng Column Heading : Gõ các tiêu đề cột theo thứ tự mà chúng ta muốn

Các giá trị này phải cách nhau bởi dấu chấm phẩy (;)

9. TRUY VẤN HÀNH ĐỘNG

9.1. Các loại truy vấn hành động

Truy vấn hành động giúp người sử dụng tạo bảng mới hay sửa đổi dữ liệu trong các bảng. Có 4 loại truy vấn hành động:

Truy vấn tạo bảng (Make table query): Tạo bảng mới từ một bảng hay nhiều bảng đã tồn tại dữ liệu.

Truy vấn cập nhật (Update query): Dùng để cập nhật dữ liệu cho một hoặc nhiều trường trong bảng dữ liệu.

*Truy vấn xoá (Delete query):*Xoá các bản ghi thoả mãn các điều kiện từ một hay nhiều bảng dữ liệu.

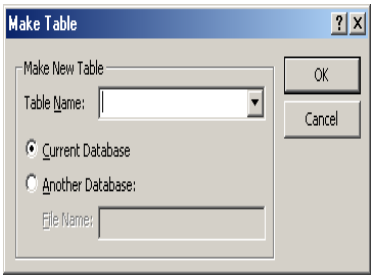
Truy vấn nối (Append query): Nối một số bản ghi từ một hoặc nhiều bảng dữ liệu vào sau một hoặc nhiều bảng dữ liệu khác.

9.2. Truy vấn tạo bảng

Truy vấn tạo bảng sẽ tạo ra một bảng mới bằng cách rút các bản ghi thoã mãn các điều kiện nào đó.

Cách tạo truy vấn

Để tạo truy vấn tạo bảng chúng ta tạo truy vấn chọn và đưa bảng vào tham gia truy vấn. Đưa các trường vào vùng lưới QBE



Chọn query/ make table query

Trong mục Table name: Đặt tên mới cho bảng muốn tạo.

Chọn Current Database: CSDL hiện thời

Another Database: Tạo bảng trong CSDL khác.

Chọn các điều kiện (Nếu có).

9.3. Truy vấn xoá

Truy vấn xoá giúp chúng ta loại bỏ các bản ghi thoả mãn một số điều kiện nào đó

Cách tạo truy vấn

Để tạo truy vấn xoá chúng ta tạo truy vấn chọn và đưa bảng vào tham gia truy vấn.

Chọn query/ Delete query

Trong vùng lưới QBE tại hàng Field chọn các trường cần so sánh với điều kiện xoá

Tại hàng Delete: Chọn phép toán Where

Tại hàng Criteria: Chọn điều kiện xoá

Ví dụ:

Tạo truy vấn để xoá những sinh viên có matruong là "SP"

9.3. Truy vấn cập nhật

Truy vấn này dùng để cập nhật giá trị hoặc sửa đổi giá trị của các trường trong bảng dữ liệu.

Cách tạo truy vấn

Tạo một truy vấn chọn và đưa bảng vào tham gia truy vấn

Chọn Query/Update query

Tại hàng Field: Chọn trường cần cập nhật dữ liệu

Tại hàng Update to: Chọn Biểu thức cần tính giá trị

Tại hàng Criteria: Chọn điều kiện (nếu có).

Ví dụ:

Cho 2 bảng dữ liệu Dsphong(**tenphong**, dongia)

Dskhachtro (tenphong, ngayden, ngaydi, thanhtien)

Tạo truy vấn để cập nhật giá trị cho trường Thanhtien=(ngaydi-ngayden)*dongia

9.4. Truy vấn nối dữ liệu

Truy vấn nối dữ liệu dùng để nối dữ liệu từ một bảng này vào sau một bảng khác.

Cách tạo truy vấn

Tạo truy vấn chọn và đưa bảng dữ liệu vào để nối với bảng khác tham gia truy vấn.

Chọn Queries/Append query

Trong mục Table name: Chọn bảng cần nối vào và chọn OK.

Chọn Current Database: CSDL hiện thời

Another Database: Tạo bảng trong CSDL khác.

Trong vùng lưới QBE của truy vấn tại hàng Field:

Đưa các trường của bảng gốc vào.

Trong hàng Append to: Đưa các trường tương ứng của bảng cần nối và đặt điều kiện nếu cần thiết.

☞ ***Chú ý***

Trong truy vấn nối dữ liệu thì các trường được nối với nhau tương ứng phải cùng kiểu dữ liệu. Nếu các trường tương ứng không có cùng kiểu dữ liệu thì sẽ không được nối. Nếu các trường có Field size không phù hợp thì tuỳ theo việc nối dữ liệu mà sẽ cắt bớt hoặc thêm vào ký tự trắng.

Chương 4 NGÔN NGỮ SQL

1. GIỚI THIỆU VỀ NGÔN NGỮ SQL

SQL là một ngôn ngữ dùng để truy xuất dữ liệu, cập nhật, thay đổi và quản lý các CSDL quan hệ.

Khi tạo một truy vấn thì ACCESS tự động xây dựng các câu lệnh SQL tương ứng.

Một số truy vấn của ngôn ngữ SQL như sau:

Truy vấn hội (Union Query)

Truy vấn chuyển nhượng (pass through Query)

Truy vấn định nghĩa dữ liệu (Data Definition Query)

Truy vấn con (Sub Query)

☞ ***Chú ý***

Trong SQL mỗi câu lệnh có thể được viết trên nhiều hàng và kết thúc mỗi câu lệnh là dấu chấm phẩy (;)

2. SQL XỬ LÝ TRÊN BẢNG DỮ LIỆU

2.1. Tạo bảng mới

Cú pháp

Create table <Table name>(<Field1> <Type>[(Size)][, <Field2> <Type>[(Size)]..... [Constraint < Primary key name> primary key <Fieldname key>] [Constraint <Index name> Unique <Field name Index>]

Chức năng: Tạo cấu trúc của một bảng mới.

Giải thích chức năng các tham số

Table name: Tên bảng cần tạo

Fieldname1, Fieldname2....: Các trường trong bảng cần tạo, tối thiểu 1 trường.

Type: Các kiểu dữ liệu tương ứng.

Size : Kích thước của trường

Primary key name : Tên khoá chính

Fieldname key : Trường làm khoá chính

Index name : Tên chỉ mục

Fieldname Index : Truờng làm chỉ mục

Constraint.....Primary key : Dùng để thiết lập khoá chính

Constraint..... Unique: Thiết lập chỉ mục (Không trùng nhau)

☞ ***Chú ý:*** Đối với các kiểu dữ liệu khi sử dụng trong SQL

Kiểu dữ liệu chuẩn	**Khai báo tương ứng trong SQL**
Text	Text(Size)
Byte	Byte
Integer	Short
Long Integer	Long
Single	Single
Double	Double
Date/Time	Datetime
Yes/No	Yesno
OLE Object	OLE Object
Currency, Memo, Counter	Currency, Memo, Counter

Ví dụ : Tạo bảng DSTRUONG có cấu trúc sau

Fieldname	Data Type	size
Matruong	Text	2
Tenruong	Text	20
SODT	Text	9

Create table dstruong(matruong text(2), tentruong text(20), Sodt text(9));

Ví dụ Tạo bảng DSHOCVIEN có cấu trúc sau

Fieldname	Data Type	size	
Mahv	Text	4	(Khoá chính)
Tenhv	Text	30	
Ngaysinh	Date/time		

SODT Text 9 (Lập chỉ mục)

Create table dshocvien(mahv text(4), tenhv text(30), ngaysinh Datetime, sodt text(9), Constraint khoa primary key (mahv), Constraint chimuc unique (sodt));

☞ ***Chú ý:*** Nếu tên trường có ký tự trắng hoặc tên bảng, tên khoá chính, tên chỉ mục thì phải đặt trong cặp dấu [...]

Ví dụ

Create table [Bang NV] ([Ma nv] text(2), [ho va ten] text(30));

2.2. Thay đổi cấu trúc của bảng

2.2.1.Thêm hoặc loại bỏ trường

Cú pháp

Alter table <Table name>[add column <fieldname><type>]

[Drop column <Field name>]

[Add Constraint <Index name> unique <fieldname>]

[Drop Constraint <Index name>]

Chức năng: Thay đổi cấu trúc của bảng

Giải thích:

ADD Column..: Thêm trường vào bảng

Drop column..: Loại bỏ trường ra khỏi bảng

Add Constraint.....: Thêm tên chỉ mục

Drop Constraint..... Loại bỏ tên chỉ mục

Ví dụ: Giả sử đã tồn tại bảng MATHANG cấu trúc sau

Fieldname	*Data Type*	*size*
MAHANG	Text	4
TENHANG	Text	20
GIA	Integer	
MAXN	Text	2
Ngaynhap	Date/time	

Thêm trường SOLUONG có kiểu byte vào bảng MATHANG

Alter Table mathang soluong byte;

Ví dụ : Thêm chỉ mục có ten cmx cho trường MAXN

Alter table mathang add constraint cmx unique Maxn

Ví dụ : Loại bỏ chỉ mục cmx

Alter table mathang drop constraint cmx

Ví dụ : Loại bỏ trường ngaynhap ra khỏi bảng MATHANG

Alter table mathang drop column ngaynhap

2.2.2. Loại bỏ chỉ mục

Cú pháp

Drop Index <Index name> on <Table name>

Chức năng: Loại bỏ 1 chỉ mục nào đó.

2.3. Xoá bảng

Cú pháp

Drop table <Table name>

Chức năng: Xoá bảng dữ liệu nào đó.

Ví dụ: Xoá bảng MATHANG

Drop table MATHANG

3. SQL XỬ LÝ TRÊN TRUY VẤN

3.1.Truy vấn chọn (Select query)

Cú pháp

Select <Scope> <Fieldname1> [AS <New name>].....

From <Table name>

[Where <Condition>]

Chức năng: Tạo truy vấn chọn

Trong đó:

Scope: Phạm vi (Mặc định là ALL, Top n: Lấy n bản ghi đầu tiên)

Nếu có AS <New name> thì sẽ thay thế tên cho Fieldname tương ứng.

Table name: Tên bảng cần lấy dữ liệu.

Condition: Điều kiện để hạn chế dữ liệu.

Ví dụ: Cho bảng dữ liệu DOCGIA sau

Fieldname	Data Type	Size	Description
MADG	Text	2	Mã độc giả (Khoá chính)
MASACH	Text	4	
HOTEN	Text	30	
QUEQUAN	Text	30	
NGAYSINH	Date/time	8	
NGAYMUON	Date/time	8	

Chọn 2 trường MADG và HOTEN

Select MADG, hoten

From docgia;

Chọn 2 trường Masach và hoten mà chỉ những masach bắt đầu là T

Select MADG, hoten

From docgia where masach like “T*”;

Chọn Hoten, quequan, madg cho những độc giả có quê quán ở Huế và đổi tên trường hoten thành Họ và tên

Select hoten AS [Họ và tên], queuqan, madg

From docgia where quequan=”Huế”;

Chọn những độc giả mượn sách trong tháng 8 hoặc năm 1999.

Select * from docgia where month([ngaymuon])=8 or year([ngaymuon])=1999;

3.2. Truy vấn tính tổng (Total query)

Cú pháp

Select .....from.....[where < Condition>]

Group by [Group fieldname]

[Having <Group Condition>];

Chức năng: Tạo một truy vấn tính tổng.

Ví dụ: Cho bảng dữ liệu BANHANG có cấu trúc

Fieldname	Data type	size
Mahang	text	2
Soluong	integer	

Ngayban Date/time 8

Tạo một truy vấn để thống kê xem mỗi loại hàng bán được với số lượng là bao nhiêu?

```
Select mahang, sum([soluong])
From banhang
Group by mahang;
```

Tạo truy vấn để thống kê xem mỗi loại hàng trong thàng 7 bán được với số lượng bao nhiêu? Chỉ hiển thị những loại hàng mà số lượng bán trên 20.

```
Select mahang, sum([soluong])
From banhang
where month([ngayban])=7
Group by mahang
 having sum([soluong])>20;
```

3. 3.Truy vấn tham khảo chéo

Cú pháp

Transform <Value Express>

Select.....From.....Where.....

Group by <Row Heading Field>

Pivot <Column heading Field>

Chức năng: Tạo truy vấn tham khảo chéo

Ví dụ

```
Transform sum([soluong])
select   Mahang, tenhang, sum([soluong]) from dskhang
Group by  tenhang
Pivot Mahang;
```

3.4. Truy vấn tạo bảng

Cú pháp

Select <Field select> into <New Table name)

From <Old Table name>

[Where <Condition>]

Chức năng: Tạo một truy vấn tạo bảng

Ví dụ

Select Mahang, tenhang into Luu

From Dskhang

Where Mahang Like “A*”;

3.5. Truy vấn nối dữ liệu

Cú pháp

Insert into <append Table name>

Select <field select>

From <Table name>

[Where <Condition>]

Chức năng: Tạo truy vấn nối dữ liệu

Nếu chỉ thêm 1 bản ghi với các giá trị cụ thể thì ta thực hiện câu lệnh

Insert into <Table name and Field list>

values <append values>

Ví dụ:

Insert into luu1(Hoten, quequan)

Values (“Nguyen an”, “Hue”)

3.6. Truy vấn cập nhật dữ liệu

Cú pháp

Update <Update Table name>

Set <Field name>=<Express>

[Where <Condition>]

Chức năng: Tạo một truy vấn dùng để cập nhật dữ liệu

Ví dụ:

Update dssv

set [hocbong]=[hocbong]+200000

Where Uutien=”1”;

3.7. Truy vấn xoá

Cú pháp

Delete <Table.>*

From <Delete Table name>

[Where <Condition>]

Chức năng: Dùng để tạo một truy vấn xoá các bản ghi trong bảng theo một hoặc nhiều điều kiện nào đó.

☞ ***Chú ý:*** nếu mệnh đề From chỏ có 1 bảng duy nhất thì không cần liệt kê các bảng trong mệnh đề DELETE.

Ví dụ:

Có 2 bảng dữ liệu DSHS và DSDTHI có quan hệ 1-1 trên trường MAHS. Hãy xoá những học sinh có điểm thi <5 trong bảng DSHS và DSDTHI.

Delete DSHS.*

From DSHS INNER JOIN DSDTHI ON DSHS.MAHS=DSDTHI.MAHS

Where diem<5.;

3.8. Tạo mối quan hệ giữa các bảng

Muốn tạo một truy vấn để truy xuất dữ liệu từ 2 hay nhiều bảng thì phải tạo các mối quan hệ giữa các bảng đó.

Cú pháp

..... From <Table name 1> inner join <Table name 2> ON <Table name 1>.<Field name1>=<Table name 2>.<Field name 2>.....

Hoặc

.....From <Table name 1> Left join/ Right join <Table name 2> ON <Table name 1>.<Field name1>=<Table name 2>.<Field name 2>.....

Ví dụ:

Tạo truy vấn gồm : Matruong, tentruong, hoten từ 2 bảng Dstruong và DSHS dựa vào trường liên kết Matruong.

Select Matruong, tentruong, hoten From Dstruong inner join dshs on Dstruong.matruong=dshs.matruong;

Ví dụ:

Cho 3 bảng dữ liệu Dstruong(**Matruong**, tentruong)

Dskhoa(Matruong, tenkhoa, **Makhoa**)

DSSV(Makhoa, Hoten, Ngaysinh, quequan)

Tạo một truy vấn để hiển thị danh sách sinh viên thuộc mỗi khoa của mỗi trường

Select Distinctrow

Matruong, tentruong, tenkhoa, hoten

From Dstruong inner join (Dskhoa inner join DSSV ON Dskhoa.Makhoa=DSSV.Makhoa)

ON Dstruong.Matruong=Dskhoa.Matruong;

☞ ***Chú ý***

Có thể sử dụng liên kết ngoại trái Left join hoặc phải Right join

3.9. Truy vấn con (Sub query)

Truy vấn con là một mệnh đề Select.....From.....Wheres được lồng ghép vào một trong các mệnh đề sau:

Select.....From.....Where

Select.....Into.....

Insert.....Into.....

Delete.....

Update.....

Cú pháp

Select.....From.....Where.....

<Biểu thức so sánh> ANY|SOME|ALL <Mệnh đề truy vấn con>

<Biểu thức tìm kiếm> IN | NOT IN <Mệnh đề truy vấn con>

EXIST | NOT EXISTS <Mệnh đề truy vấn con>;

Chức năng: Tạo một truy vấn con

Giải thích các tham số

<Biểu thức so sánh>: Là một biểu thức và một phép toán so sánh.

<Biểu thức tìm kiếm>: Là một biểu thức mà tập hợp kết quả của truy vấn con sẽ được tìm kiếm.

<Mệnh đề truy vấn con>: Là dạng mệnh đề ở trong cú pháp và đặt giữa hai dấu ( ).

ANY, SOME: Các bản ghi trong truy vấn chính thoả mãn điều kiện so sánh với bất kỳ hoặc một vài các bản ghi nào truy xuất được từ truy vấn con.

ALL: Các bản ghi trong truy vấn chính thoả mãn với điều kiện so sánh với tất cả bản ghi nào truy xuất được từ truy vấn con.

IN: Các bản ghi trong truy vấn chính mà có tồn tại một vài bản ghi trong truy vấn con có giá trị bằng nó.

NOT IN: Các bản ghi trong truy vấn chính mà không tồn tại một vài bản ghi trong truy vấn con có giá trị bằng nó.

EXISTS (NOT EXISTS): Phép so sánh True/ False để xác định nhận truy vấn con có kết quả là bản ghi nào không.

Ví dụ:

Cho 2 bảng dữ liệu KHO(Mahang, Tenhang, Dongia)

NKBAN (Mahang, Hoten, Ngaymua, Giamgia, Dongia)

Tìm tất cả những mặt hàng mà đơn giá lớn hơn vài mặt hàng được bán với Giamgia là 20%.

*Select * From Kho*

Where dongia> ANY (select dongia From NKBAN Where giamgia=20%);

Tìm những mặt hàng bán ra với giảm giá >=10%.

*Select * from Kho*

Where Mahang IN (Select mahang From NKBAN Where giamgia>=0.1);

Ví dụ:

Cho 2 bảng danh sách DSKH(MAKHACH, HOTEN, QUEQUAN, SDT)

DATHANG(MAKHACH, SOLUONG, NGAYDAT)

Tìm những người khách đặt hàng trước 10/10/99 bao gồm Hoten, Quequan.

Select Hoten, quequan From DSKH

Where Makhach IN (Select makhach from DATHANG Where NGAYDAT<=#10/10/99#;.

3.10. Truy vấn hội (Union Query)

Dùng để nối (Kết hợp) dữ liệu các trường tương ứng từ 2 hay nhiều bảng hoặc truy vấn vào trường.

Cú pháp: *Select .....From.....Where.....*

UNION | UNION ALL

Select.....

Chức năng: Tạo truy vấn hội

Giải thích:

UNION: Không muốn các bản ghi trùng nhau hiển thị

UNION ALL: Hiển thị các bản ghi trùng nhau

Ví dụ:

Cho 2 bảng dữ liệu DHSVIEN(Hoten, Lop, Matruong, Diachi)

CDHSVIEN(Hoten, Lop, Matruong, Diachi)

Sử dụng truy vấn hội để liệt kê Hoten, Lop, Matruong của sinh viên 2 hệ (Địa học, Cao đẳng).

Select Hoten, Lop, Matruong from DHSVIEN

UNION Select Hoten, Lop, Matruong from DHSVIEN

Liệt kê Hoten, Lop của những sinh viên 2 hệ và có quê quán ở Huế

Select Hoten, Lop from DHSVIEN

UNION Select Hoten, Lopfrom DHSVIEN Where Diachi=”Huế”;

Ch□□ng 5 BIỂU MẪU

Từ trước đến nay chúng ta vẫn làm việc một cách đơn điệu với các bảng, truy vấn với cách trình bày dữ liệu hiệu quả nhưng không đẹp mắt. Với biểu mẫu (form) trong Access sẽ giúp chúng ta khắc phục điều này. Biểu mẫu trong Access rất linh động, chúng ta có thể dùng biểu mẫu để nhập, xem, hiệu chỉnh dữ liệu. Hoặc là dùng biểu mẫu để tạo ra các bảng chọn công việc làm cho công việc của chúng ta thuận lợi và khoa học hơn. Hoặc dùng biểu mẫu để tạo ra các hộp thoại nhằm thiết lập các tùy chọn cho công việc quản lý của mình.

1. KHÁI NIỆM VỀ BIỂU MẪU

Nếu chúng ta đã quen điền các tờ biểu, mẫu trong cuộc sống hàng ngày thì chúng ta có thể hình dung một biểu mẫu trong Access cũng vậy. Một biểu mẫu trong Access định nghĩa một tập dữ liệu chúng ta muốn lấy và từ đó đưa vào CSDL. Cũng vậy biểu mẫu cùng có thể dùng để xem xét dữ liệu hay in ra máy in.

Trong môi trường của Hệ QTCSDL Access chúng ta có thể thiết kế các biểu mẫu có hình thức trình bày đẹp, dễ sử dụng và thể hiện đúng các thông tin cần thiết. Chúng ta có thể đưa vào biểu mẫu các đối tượng như văn bản, hình ảnh, đường vẽ két hợp với các màu sắc sao cho biểu mẫu của chúng ta đạt được nội dung và hình thức trình bày ưng ý nhất. Hình thức và cách bố trí các đối tượng ra sao trên biểu mẫu hoàn toàn tùy thuộc vào khả năng thẩm mỹ và ng khiếu trình bày của chúng ta.

2. TÁC DỤNG VÀ KẾT CẤU CỦA BIỂU MẪU

2.1.Tác dụng của biểu mẫu

Biểu mẫu cung cấp một khả năng thuận lợi để hiển thị dữ liệu. Chúng ta có thể xem mọi thông tin của một bản ghi thay vì ở chế độ Datasheet nghèo nàn trước đây bằng chế độ Form View, một phương cách tiên tiến hơn.

Sử dụng biểu mẫu tăng khả năng nhập dữ liệu, tiết kiệm thời gian và ngăn ngừa các lỗi do đánh sai. Chẳng hạn thay vì gõ vào các giá trị của tất cả các trường chúng ta có thể tạo những danh sách (gọi là combo box) để chọn trên biểu mẫu (đây là phương cách áp dụng rất hiệu quả để tránh đánh sai dữ liệu).

Biểu mẫu cung cấp một hình thức trình bày hết sức tiện nghi để xem, nhập và hiệu chỉ các bản ghi trong CSDL. Access cung cấp các công cụ thiết kế biểu mẫu hỗ trợ rất đắc lực cho chúng ta trong việc thiết kê những biểu mẫu dễ sử dụng mà lại có thể tận dụng được các khả năng:

Hình thức thể hiện dữ liệu đẹp, trình bày lôi cuốn với các kiểu font và hiệu ứng đồ họa đặc biệt khác ...

Quen thuộc với người sử dụng vì nó giống các biểu mẫu trên giấy thông thường.

Có thể tính toán được.

Có thể chứa cả biểu đồ.

Có thể hiển thị dữ liệu từ nhiều bảng (hoặc truy vấn)

Tự động hóa một số thao tác phải làm thường xuyên.

2.2. Kết cấu của biểu mẫu

Các thông tin trên biểu mẫu có thể lấy dữ liệu từ một bảng hay truy vấn nào đó, nhưng cũng có thể độc lập đối với cả bảng lẫn truy vấn, chẳng hạn như các đối tượng đồ họa. Dáng vẻ trình bày của biểu mẫu được thực hiện trong quá trình thiết kế.

Tất cả các thông tin thể hiện trên biểu mẫu được chứa trong những đối tượng gọi là *điều khiển* (control). Điều khiển có thể dùng để thể hiện dữ liệu hoặc thực hiện các hàng động hoặc trang trí cho biểu mẫu.

Một số điều khiển được buộc vào với các trường của bảng hay truy vấn, gọi là bảng cơ sở hay truy vấn cơ sở. Do đó chúng ta có thể dùng biểu mẫu để nhập dữ liệu vào

các trường hay lấy dữ liệu từ các trường đó ra để xem. Ví dụ dùng Text box để nhập hay hiển thị chuỗi và số, dùng Object frame để thể hiện hình ảnh.

Một số điều khiển khác trình bày thông tin được lưu trữ trong thiết kế bảng. Ví dụ dùng Label (nhãn) để thể hiện thông tin có tính chất mô tả; đường và các hình khối để tổ chức dữ liệu và làm biểu mẫu có hình thức hấp dẫn hơn.

3. TẠO BIỂU MẪU

3.1. Tạo biểu mẫu tự động với Autoform

Access cung cấp chức năng Autoform cho phép chúng ta tạo biểu mẫu dựa trên các bảng hoặc truy vấn đã được xây dựng trước đó.

Cách tạo

Trong cửa sổ Database, chọn form, chọn New

Trong mục *Choose the table or query Where the object's data comes from:*

Chọn bảng hoặc truy vấn làm nguồn dữ liệu cho form.

Chọn **Autoform Columnar**: Nếu muốn tạo lập biểu mẫu dạng cột, trong đó mỗi trường trong bảng hay truy vấn là một dòng.

Chọn **Autoform Tabular**: Nếu muốn tạo lập biểu mẫu dạng hàng, trong đó mỗi trường trong bảng hay truy vấn là một cột và một bản ghi trong một dòng.

Chọn **Autoform Datasheet**: Nếu muốn tạo lập biểu mẫu theo dạng bảng, trong đó mỗi cột tương ứng một trường và mỗi dòng là một bản ghi.

Chọn OK.

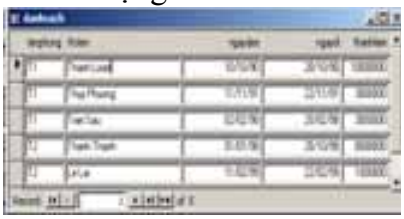
Ví dụ

Cho bảng Danhsach(Tenphong, hoten, ngayden, ngaydi, thanhtien)

Hãy thiết kế biểu mẫu dựa trên chức năng Autoform sử dụng bảng Danhsach làm nguồn dữ liệu.

Biểu mẫu dạng **Autoform Columnar**

Biểu mẫu dạng **Autoform tabular**



Biểu mẫu dạng **Autoform Datasheet**

3.2. Tạo biểu mẫu sử dụng Wizard

Tạo biểu mẫu sử dụng công cụ Autoform thì Access không cho phép người sử dụng can thiệp vào quá trình tạo biểu mẫu, chẳng hạn như hạn chế số trường..... thì Form Wizard cho phép người sử dụng can thiệp vào quá trình tạo biểu mẫu.

Cách tạo

Trong cửa sổ Database chọn Form, chọn New

Trong mục *Choose the table or query Where the object's data comes from:*

Chọn bảng hoặc truy vấn làm nguồn dữ liệu cho form.

Chọn Form Wizard

Chọn OK

Trong mục Avaiable Field: Chọn các trường đưa vào biểu mẫu, nhấn nút >>

Chọn nút **Next**.

Chọn *Columnar* : Biểu mẫu hiển thị theo dạng cột

Tabular : Biểu mẫu hiển thị theo dạng hàng

Datasheet : Biểu mẫu hiển thị theo dạng bảng

Justified : Biểu mẫu hiển bình thường (đều).

Chọn **Next**

Chọn loại biểu mẫu

Chọn **Next**

Đặt tiêu đề cho Form

Chọn *Open the form to view or enter information*: Nếu muốn mở Form sau khi chọn Finish.

Chọn *Modify the form’s design:* Nếu muốn form ở dạng thiết kế.

Chọn **Finish** .

Lưu form.

3.3. Tạo biểu mẫu không sử dụng Wizard (Do người sử dụng tự thiết kế)

Tạo biểu mẫu sử dụng công cụ Autoform và Form wizard người sử dụng có thể nhanh chóng thiết kế các biểu mẫu nhờ vào các đặc tính hỗ trợ của Access. Nhưng đối với hai cách trên chỉ cung cấp một số hạn chế các phương án xây dựng biểu mẫu mà không thoã mãn yêu cầu của người sử dụng khi muốn thiết kế biểu mẫu teho ý của riêng mình. Do đó người sử dụng phải tự thiết kế một biểu mẫu không cần sự hỗ trợ của Access.

Cách tạo

Trong cửa sổ Database chọn Form, chọn New

Chọn bảng dữ liệu hoặc truy vấn làm nguồn dữ liệu cho form, chọn **OK**

Xây dựng các điều khiển cho biểu mẫu (Đưa các trường trong bảng dữ liệu vào biểu mẫu).

Thiết lập các thuộc tính cho các điều khiển.

Lưu biểu mẫu.

4. CÁC CHẾ ĐỘ HIỂN THỊ VÀ CÁC THÀNH PHẦN CỦA BIỂU MẪU

4.1. Các chế độ hiển thị

Có 4 chế độ hiển thị của biểu mẫu

4.1.1.Chế độ Design View

Dùng để tạo biểu mẫu mới hay thay đổi cấu trúc của một biểu mẫu đã tồn tại.

Khi đang ở trong cửa sổ Database: Chọn form /chọn tên form/ chọn Design (Có

thể click chuột phải rồi chọn Design).

Khi đang ở chế độ Form view: Chọn View/ Form Design

4.1.2. Chế độ Form View

Dùng để nhập, thay đổi và xem dữ liệu. Trong chế độ Form View người sử dụng có thể xem tất cả các trường của một bản ghi tại một thời điểm.

Khi đang ở trong cửa sổ Database: Chọn form /chọn tên form/ chọn Open (Có thể click chuột phải rồi chọn Open).

Khi đang ở chế độ Design view: Chọn View/ Form View.

4.1.3. Chế độ hiển thị biểu mẫu dưới dạng bảng (Datasheet View)

Dùng để nhập, thay đổi và xem dữ liệu trong biểu mẫu theo dạng bảng biểu.

Để mở chế độ hiển thị dạng Datasheet View chúng ta thực hiện như sau:

Khi đang ở chế độ Design View: Chọn View/Datasheet.

Khi đang ở chế độ Form View: Chọn View/ Datasheet View.

4.1.4. Chế độ hiển thị Print Preview

Dùng để xem biểu mẫu trước khi quyết định in ấn. Trong chế độ Print Preview sẽ duy trì hình dạng trình bày dữ liệu đã được thiết kế trước đó.

Khi đang ở trong cửa sổ Database: Chọn form /chọn tên form/ Chọn File/Print Preview.

4.2. Các thành phần biểu mẫu trong chế độ Design View

Khi muốn thiết kế biểu mẫu thì người sử dụng phải làm việc trong chế dộ Design View khi đó biểu mẫu có các thành phần chính sau:

Thước(Ruler): Điều chỉnh kích thước của các điều khiển.

Tiêu đề form (form header):Sử dụng để trình bày tiêu đề của form, tiêu đề form luôn được trình bày phần trên cùng, đầu tiên của biểu mẫu và trang in biểu mẫu.

Chân form (Form Footer): Sử dụng để trình bày chân của form, chân form luôn được trình bày phần dưới cùng, xuất hiện cuối biểu mẫu và trang in biểu mẫu.

Tiêu đề trang (Page header): Sử dụng để chứa tiêu đề trang

Chân trang (Page footer): Sử dụng để chứa chân trang nhưng xuất hiện phần trước của Form footer trong trang biểu mẫu in.

☞ ***Chú ý***

Page header và Page footer chỉ xuất hiện trong trang biểu mẫu in nên chúng không có những tính chất thông thường như Form header và Form footer.

Chọn View/ Page header/ footer (Nếu 2 thành phần này chưa xuất hiện trên biểu mẫu).

Chi tiết form (Detail): Đây là phần rất quan trọng chứa các điều khiển nhằm trình bày các dạng dữ liệu từ các bảng dữ liệu hoặc các truy vấn. Các loại điều khiển có thể là điều khiển buộc, không buộc hoặc tính toán.

5. CÁC LOẠI ĐIỀU KHIỂN

Tất cả thông tin trên biểu mẫu được chứa trong những đối tượng gọi là điều khiển (Control). Điều khiển có thể dùng để thể hiện dữ liệu, thực hiện các hành động hoặc thiết kế biểu mẫu đẹp mắt. Trong ACCESS hệ thống định nghĩa một số loại điều khiển như sau:

Điều khiển nhãn (Label).

Điều khiển hộp văn bản (Text box).

Điều khiển nhóm lựa chọn (Option group).

Điều khiển loại hộp Combo (Combo box) và hộp danh sách (List Box).

Ngoài ra còn có một số điều khiển khác như command button.....

Khi tạo lập điều khiển, chúng ta thuờng xác định hình thức dữ liệu trình bày trong chúng. Có những điều khiển lấy dữ liệu từu các trường trong bảng hay truy vấn, có điều khiển chỉ dùng vào mục đích trang trí, làm tiêu đề, có những điều khiển lấy dữ liệu từ một biểu thức nào đó. Vì vậy người ta phân ra thành ba nhóm điều khiển chính:

Điều khiển buộc (Bound control)

Điều khiển không buộc (Unbound control)

Điều khiển tính toán (Caculated control)

5.1. Điều khiển bị buộc (bound), không buộc (unbound) và tính toán được (calculated).

Khi tạo một điều khiển trong biểu mẫu thì phải xác định nó lấy dữ liệu từ nguồn nào để thể hiện.

Ví dụ

Tạo một điều khiển loại hộp văn bản (Text box) để hiển thị tên các mặt hàng, chúng ta phải chỉ định cho điều khiển lấy dữ liệu trong trường TEN_HANG của bảng MAT_HANG. Hộp văn bản này gọi là bị buộc.

Điều khiển cũng có thể thể hiện những thông tin không có trong CSDL (Không bị buộc).

Ví dụ: Tạo tiêu đề cho biểu mẫu...

Tóm lại

Điều khiển bị buộc (Bound Control) là điều khiển mà nguồn dữ liệu của nó lấy từ một trường trong bảng hoặc truy vấn Trong biểu mẫu dùng điều khiển buộc vào các trường để hiển thị nội dung hoặc cập nhật các trường của CSDL, các giá trị cập nhật có thể là: Văn bản, Date,Num ber, yes/No, Picture, chart trong đó dạng văn bản là phổ biến nhất.

Điều khiển không bị buộc (Unbound Control) là điều khiển không lấy dữ liệu từ một nguồn nào cả là điều khiển không bị buộc. Dùng điều khiển không buộc để trình bày thông tin không có trong các bảng hay rút được từ truy vấn.

Điều khiển tính toán (Calculated Control) là điều khiển mà nguồn dữ liệu của nó không phải là một trường mà là một biểu thức gọi là điều khiển tính toán (Calculated Control). Chúng ta qui định giá trị xuất hiện trong điều khiển bằng cách lập biểu thức cho nó. Biểu thức này là nguồn dữ liệu của điều khiển .Trong biểu thức có thể dùng các toán tử (+, -,=...) với tên điều khiển.

Ví dụ: Tạo một điều khiển THANHTIEN=SOLUONG*DONGIA

5.2. Tạo điều khiển loại hộp văn bản (text box)

Text box có thể là một điều khiển bị buộc, không buộc hoặc tính toán.

Tạo hộp văn bản bị buộc (Bound Text box).

Chúng ta buộc điều khiển Text box vào một trường bằng cách chỉ định điều khiển đó lấy dữ liệu trên trường nào. Chọn trường để buộc vào điều khiển bằng cách Click biểu

tượng Field List để mở danh sách các trường của bảng hay truy vấn làm nền tảng cho biểu mẫu. Theo mặc định như vậy thì hệ thống sẽ tạo một điều khiển loại Text box.

Một cách khác dùng hộp dụng cụ Toolbox để tạo điều khiển và sau đó gõ tên trường muốn buộc vào hộp văn bản.

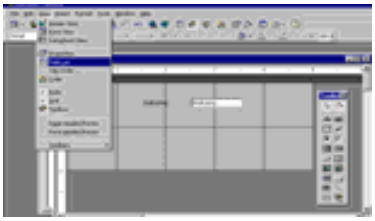
Mở hộp danh sách trường

Mở biểu mẫu trong chế độ Design (Tạo biểu mẫu trước).

Trong cửa sổ Database Chọn Form/New (Cọn bảng hoặc truy vấn làm nền cho biểu mẫu).

Chọn View/Field List hoặc click vào biểu tượng Field List trên thanh công cụ).

Tạo một Text Box bị buộc



Từ danh sách trường chọn một hoặc nhiều trường kéo và đưa vào biểu mẫu.

5.3. Tạo một điều khiển khác dùng hộp công cụ

Dùng hộp công cụ (Toolbox) để tạo những điều khiển không buộc (Unboud control) hoặc để tính toán. Đặc biệt dùng các tính năng của hộp này để tạo những điều khiển bị buộc khác ngoài buộc khác ngoài (Text box). Để bật hộp công cụ ta chọn View/Toolbars.

Tạo điều khiển dùng hộp công cụ

Click vào biểu tượng công cụ tuơng ứng với điều khiển muốn tạo.

Tạo điều khiển bị buộc bằng cách chọn một trường trong danh sách trường (Field List) và kéo vào biểu mẫu.

Hoặc tạo điều khiển không buộc hay dùng để tính toán bằng cách click vào một vị trí trên biểu mẫu.

Tạo điều khiển dùng để tính toán

Nếu muốn trình bày kết quả của một phép toán trong biểu mẫu, khai báo nguồn dữ liệu của điều khiển là một biểu thức. Sau này mỗi lần mở biểu mẫu, ACCESS tính toán lại kết quả của vùng dữ liệu được cập nhật mới nhất từ các bảng.

Tạo điều khiển nhãn (Label Control)

Muốn trình bày một chuỗi ký tự trên biểu mẫu như để làm tiêu đề... chúng ta dùng loại điều khiển gọi là điều khiển nhãn (label control). Nhãn không thể hiện dữ liệu của một trường hay biểu thức nào , chúng luôn luôn là không bị buộc.

Click vào biểu tượng Label trong Toolbox

Click vị trí muốn đặt nhãn trên biểu mẫu.

Thay đổi các thuộc tính của điều khiển.

Sau khi tạo biểu mẫu, biến đổi một số thuộc tính của điều khiển có thể hoàn thiện thêm về thiết kế và hình thức trình bày của số liệu.

Nhắp kép vào điều khiển.

Trong hộp lựa chọn trên đầu bảng thuộc tính, chọn một lớp thích hợp để làm việc.

All Properties: Trình bày tất cả các thuộc tính của điều khiển

Data Properties: Ấn định các đặc tính thể hiện dữ liệu trong điều khiển như giá trị mặc nhiên, định dạng số.

Even Properties: Qui định một tập lệnh (Macro) hay thủ tục (Procedure)..

Layout Properties: Định nghĩa các hình thức của điều khiển như cao, rộng.

Other Properties: Một số thuộc tính khác như tên điều khiển, thông tin mô tả ở dòng trạng thái.

Click chọn một trong các thuộc tính của bảng để thực hiện.

5.4. Thiết lập một số thuộc tính bổ sung

5.4.1. Các thuộc tính hỗ trợ nhập liệu

Thuộc tính **Default value:** Gán giá trị mặc định vào nội dung trình bày trong điều khiển.

Thuộc tính **ValidationRule và ValidationRule Text**: Kiểm tra tính hợp lệ khi nhập dữ liệu cho một điều khiển và thông báo lỗi nếu dữ liệu không hợp lệ

5.4.2. Các thuộc tính gióng lề

Thuộc tính **General:** Gióng hàng văn bản theo lề trái, dữ liệu số và ngày tháng theo lề phải.

Thuộc tính **Left:** Gióng hàng văn bản theo lề trái.

Thuộc tính **Center:** Xác lập hàng văn bản ở chính giữa.

Thuộc tính **Center:** Gióng hàng văn bản theo lề phải.

5.4.3. Các thuộc tính màu sắc

Thuộc tính **Backcolor**: Thiết lập màu nền cho điều khiển hay cho biểu mẫu

Thuộc tính **Forecolor**: Thiết lập màu cho hàng chữ trong điều khiển.

Thuộc tính **Bodercolor**: Thiết lập màu cho khung bao quanh trong điều khiển.

5.4.3. Các thuộc tính khung bao

Thuộc tính **BoderStyle**:Thiết lập loại khung cho điều khiển

Thuộc tính **BoderWidth**: Thiết lập độ dày hay đậm của khung bao.

Thuộc tính **BoderColor**: Thiết lập màu của khung bao.

6. NÂNG CẤP BIỂU MẪU

Access cung cấp nhiều loại điều khiển để làm cho biểu mẫu dễ sử dụng và có nhiều hiệu ứng tốt. Có thể thay thế một Textbox với một List box hay Combo box để chọn từ các giá trị có sẵn thay vì buộc người sử dụng phải nhớ để nhập giá trị vào.

6.1. Dùng điều khiển List box và Combo box để tạo danh sách chọn lựa.

Trong nhiều trường hợp, chọn một danh sách có sẵn thường tiện lợi hơn phải nhớ để gõ vào từ bàn phím. Access cung cấp hai khả năng điều khiển tạo danh sách chọn lựa:

List box và Combo box.

List box đơn giản là một danh sách để chọn, combo box tương tự như một text box và một combo box kết hợp vào một điều khiển, có nghĩa là có thể gõ thẳng giá trị vào text box hay chọn từ một danh sách có sẵn.

Ưu điểm của List box: Danh sách luôn được thể hiện và người dùng chỉ được phép chọn trong danh sách, do đó dữ liệu nhập luôn luôn là hợp lệ.

Ưu điểm của Combo box: Danh sách không được thể hiện cho đến khi người dùng mở hộp điều khiển, do đó ít tốn chỗ trên biểu mẫu hơn.

6.2. Tạo List box và Combo box không sử dụng Wizard

Tắt chức năng Control Wizard trong menu View hoặc trong thanh công cụ.

Click biểu tượng List box hoặc Combo box trong hộp công cụ.

Tạo điều khiển buộc bằng cách click biểu tượng Field list trên thanh công cụ để mở danh sách trường làm nền cho biểu mẫu. Chọn một trường trong danh sách kéo vào biểu mẫu, hoặc tạo điều khiển không buộc bằng cách click biểu mẫu nơi muốn đặt điều khiển.

Lập các thuộc tính của điều khiển để định nghĩa các hàng dùng làm chọn lựa trong danh sách.

Muốn danh sách thể hiện	**Lập thuộc tính Row Source Type thành**	**Lập Row Source thành**
Các hàng từ một bảng hay truy vấn	Table/Query (Default)	Tên của bảng hay truy vấn đó
Các hàng lấy từ lệnh Select của SQL	Table/Query (Default)	Một câu lệnh SQL
Một danh sách với các giá trị do người dùng tự đặt	Value List	Danh sách các giá trị đó phân cách nhau bởi dấu chấm phẩy
Tên các trường trong một bảng hoặc truy vấn	Field List	Tên của bảng hoặc truy vấn đó
Các giá trị định nghĩa bởi một hàm trong Access Basic	Tên hàm xây dựng	Để trống

6.2.1. Định nghĩa các hàng trong danh sách

Định nghĩa nguồn dữ liệu để hình thành các hàng dùng làm trong danh sách lựa chọn trong List box và Combo box bằng cách lập thuộc tính thích hợp cho RowSource Type và RowSource của điều khiển, hai thuộc tính này phối hợp với nhau để tạo nên các hàng trong danh sách.

Thuộc tính RowSource Type cho biết danh sách sẽ hình thành từ nguồn dữ liệu loại nào, thuộc tính RowSource là thuộc tính xác định nguồn dữ liệu của danh sách.

6.2.2. Danh sách lấy dữ liệu từ bảng hay truy vấn

Có thể tạo List box hay Combo box lấy dữ liệu từ một hoặc nhiều trường trong bảng (hoặc truy vấn) để lập thành danh sách chọn lựa cho điều khiển. Thực hiện điều này bằng cách đặt thuộc tính RowSource Type thành Table/Query và RowSource thành tên bảng hay tên truy vấn có chứa dữ liệu để lập thành danh sách đó.

6.2.3. Tạo danh sách nhiều cột

Có thể tạo danh sách gồm hai hay nhiều cột để có thêm thông tin lựa chọn. Xác định số cột bằng cách lập thuộc tính Columcount và ColumnWidth cho điều khiển. Thuộc tính columcount quy định số cột và nếu trong RowSource là tên một bảng hay truy vấn thì các trường đầu tiên tương ứng số cột của bảng hay truy vấn đó (tính từ trái sang phải) sẽ được đưa vào danh sách. Thuộc tính columnwidth quy định bề rộng mỗi cột tính theo inch hay cm.

6.2.4. Sử dụng danh sách giá trị

Trường hợp điều khiển chỉ có một số nhỏ chọn lựa và các giá trị này không thay đổi, có thể dùng phương pháp đơn giản là danh sách giá trị. Danh sách này chúng ta tự lập bằng cách đưa các giá trị dùng làm chọn lựa vào thuộc tính RowSource của điều khiển. Sử dụng dấu chấm phẩy để phân cách các mục chọn lựa của danh sách.

6.3. Sử dụng hộp kiểm tra (check box), nút chọn lựa (Option button), nút bật tắt (Toggle button).

Hộp kiểm tra, nút chọn lựa, nút bật tắt thường được sử dụng như các điều khiển độc lập để nhận các chọn lựa Yes/No. Các điều khiển này thực chất chỉ khác nhau về hình thức, do đó chúng ta có thể sử dụng bất cứ nút nào.

Khi được chọn điều khiển này biểu diễn trị Yes (hay True), ở trạng thái không chọn thì nhận giá nị No (hay False).

Check box : Được chọn khi có dấu **x** trong hộp.

Option button : Được chọn khi có hạt đậu trong nút.

Toggle button : Được chọn khi nó trông như bị nhấn xuống.

6.4. Sử dụng nhóm chọn lựa (Option group)

Chúng ta có thể dùng nhóm chọn lựa để trình bày một số nhỏ các chọn lựa trên biểu mẫu. Nhóm chọn lựa gồm một khung nhóm (Group frame) có gắn nhãn và một số check box, option button hay toggle button.

Nếu nhóm lựa chọn được buộc vào một trường, chỉ có khung nhóm là bị buộc vào trường đó, không phải các thành phần để chọn lựa trong khung. Khung nhóm có thể không bị buộc vào trường nào hoặc chứa một biểu thức, mỗi lần chỉ có thể chọn một thành phần trong nhóm.

Khi sử dụng hộp kiểm tra, nút chọn lựa và nút bật tắt trong nhóm chọn lựa, thuộc tính của chúng khác với như khi được dùng với các điều khiển độc lập. Mỗi phần tử trong nhóm chọn lựa không có thuộc tính Control Source mà thay bằng thuộc tính Option Value. Lập thuộc tính Option Value của mỗi phần tử dùng làm chọn lựa trong nhóm thành một trị số có nghĩa đối với trường mà khung nhóm được buộc vào. Khi một phần tử của nhóm được chọn, Access đưa giá trị lập trong Option value của phần tử đó cho điều khiển Option Group. Đến lượt Option Group trả lại giá trị đó cho trường mà nó bị buộc vào.

7. BIỂU MẪU DỰA TRÊN NHIỀU BẢNG

7.1. Biểu mẫu phụ (Subform)

Biểu mẫu phụ là phương pháp để đưa thông tin từ nhiều bảng vào một biểu mẫu. Biểu mẫu phụ có nghĩa là một biểu mẫu được lồng trong biểu mẫu khác. Trong Access biểu mẫu chính gọi là Main form, biểu mẫu nằm trong Main form gọi là biểu mẫu phụ (Sub form). Khi dùng biểu mẫu phụ chúng ta dễ nhận thấy mối quan hệ giữa các bản ghi của hai hay nhiều bảng.

Biểu mẫu phụ đặc biệt hữu hiệu khi dùng để hiển thị dữ liệu từ nhiều bảng hay truy vấn có quan hệ một-nhiều với nhau. Biểu mẫu chính đại diễn cho bên một, biểu mẫu phụ đại diện cho bên nhiều.

7.2. Các loại biểu mẫu phụ

Khi tạo biểu mẫu phụ chúng ta có thể thiết kế nó thành dạng bảng, hoặc dạng biểu mẫu, hoặc cả hai dạng trên.

Biểu mẫu dạng bảng: Là loại dễ tạo nhất và có thể sử dụng như bất kỳ bảng nào khác như sắp xếp....

Biểu mẫu phụ dạng biểu mẫu: Cho chúng ta thực sự linh hoạt và mềm dẽo khi thiết kế hơn.

7.3. Thiết kế biểu mẫu phụ

Thông thường chúng ta dùng bảng hay truy vấn làm nguồn dữ liệu cho biểu mẫu chính, một bảng hay truy vấn khác làm nguồn dữ liệu cho biểu mẫu phụ. Nếu dữ liệu trong biểu mẫu chính và biểu mẫu phụ có liên quan với nhau, chúng ta cần đánh giá một số vấn đề sau:

Các bảng hoặc truy vấn có quan hệ một-nhiều với nhau không? Nếu dùng biểu mẫu phụ để thể hiện quan hệ một-nhiều, chúng ta nên dùng bảng bên một đối với bảng chính, bảng bên nhiều đối với bảng phụ.

Các bảng hoặc truy vấn làm nguồn dữ liệu cho biểu mẫu chính/phụ có các trường liên kết không? Access dùng trường kết nối để giới hạn số lượng bản ghi thể hiện trong biểu mẫu phụ.

Cách tạo biểu mẫu chính/phụ

Thiết kế hai biểu mẫu riêng biệt, sau đó kéo biểu mẫu phụ vào biểu mẫu chính.

Thiết kế biểu mẫu chính

Tạo biểu mẫu chính, dành chỗ trên biểu mẫu này để chứa biểu mẫu phụ.

Lưu và đóng biểu mẫu chính.

Thiết kế biểu mẫu phụ

Có thể thiết kế biểu mẫu phụ để chỉ thể hiện dữ liệu dưới dạng bảng, biểu mẫu này có cả hai khả năng trên.

Tạo biểu mẫu mới, lập hai thuộc tính ViewAllowed và Default View của biểu mẫu tuỳ theo yêu cầu sử dụng như sau:

Biểu mẫu phụ chỉ trình bày dưới dạng bảng: Đặt các trường trên biểu mẫu theo thứ tự chúng ta muốn chúng xuất hiện trong bảng. Lập cả hai thuộc tính ViewAllowed và Default View thành Datasheet.

Biểu mẫu phụ chỉ trình bày dữ liệu dưới dạng biểu mẫu: Sắp đặt các điều khiển như trên. Lập thuộc tính ViewAllowed thành Form và Default View thành Single form hay Continuous form.

Đưa biểu mẫu phụ vào biểu mẫu chính

Mở biểu mẫu chính ở chế độ Design View

Chuyển sang cửa sổ Database, nhần F11

Kéo biểu mẫu phụ tử cửa sổ Database và đặt vào một vị trí trên biểu mẫu chính.

Di chuyển biểu mẫu phụ đến vị trí khác, thay đổi nội dung nhãn hoặc kích thước nếu cần. Chuyển sang chế độ Form View để xem kết quả.

7.4. Liên kết biểu mẫu chính và biểu mẫu phụ

Trong chế độ Design View của biểu mẫu chính, mở bảng thuộc tính của điều khiển biểu mẫu phụ.

Lập thuộc tính LinkChildFields thành tên trường nối kết trong biểu mẫu phụ, nếu có nhiều trường nối kết, phân cách nhau bởi dấu phẩy.

Lập thuộc tính LinkMasterFields thành tên trường nối kết hoặc tên điều khiển trong biểu mẫu chính, nếu có nhiều trường nối kết, phân cách nhau bởi dấu phẩy.

Chương 6 BÁO CÁO (REPORT)

1. TỔNG QUAN VỀ BÁO CÁO

Báo cáo là phương thức hữu hiệu giúp người sử dụng trình bày dữ liệu dưới dạng đầy đủ và dễ hiểu, nhanh chóng, đẹp mắt để khi in ấn. Người sử dụng có thể tích hợp trong báo cáo các dạng thức trình bày dữ liệu khác nhau như: Hình ảnh, biểu đồ, văn bản.....

Báo cáo được xây dựng trên một nguồn dữ liệu đó là bảng hoặc truy vấn, một câu lệnh SQL hoặc một dạng biểu mẫu nào đó

1.1. Các dạng mẫu của báo cáo

Báo cáo dạng cột (columnar): báo cáo dạng này sẽ được trình bày theo dạng một cột và kèm theo phần nhãn của mỗi cột dữ liệu bên trái, mỗi dòng tương ứng với một trường dữ liệu.

Báo cáo dạng hàng (Tabular): Báo cáo sẽ trình bày dữ liệu theo dạng bảng bao gồm nhiều hàng và nhiêu cột.

Báo cáo dạng nhóm/ Tổng (Group/Total): Báo cáo dạng này sẽ tổ chức dữ liệu thành các nhóm, mỗi nhóm sẽ trình bày dữ liệu theo dạng Tabular. Người sử dụng có thể nhóm dữ liệu theo cấp và có thể tính toán giá trị tổng cho mỗi nhóm và một giá trị tính tổng cho toàn bộ các nhóm

Báo cáo dạng biểu đồ (Chart)

Báo cáo dạng nhãn (Label Report)

Báo cáo với báo cáo con

1.2. Các chế độ hiển thị của báo cáo

Báo cáo có thể được trình bày theo 3 chế độ sau

Report design: Chế độ thiết kế báo cáo.

Layout PreView: Chế độ trình bày dữ liệu trong báo cáo.

Print PreView: Chế độ xem hình thức báo cáo trước khi in ấn.

2. TẠO BÁO CÁO SỬ DỤNG CÔNG CỤ AUTO REPORT VÀ REPORT WIZARD

2.1.Tạo báo cáo sử dụng Auto report

Click biểu tượng Report trong cửa sổ database (hoặc chọn View/Report)

Chọn New

Chọn bảng hoặc Truy vấn làm nguồn dữ liệu báo cáo .

Chọn AutoReport Columnar: Nếu muốn báo cáo hiển thị dạng cột.

AutoReport Tabular: Nếu muốn báo cáo hiển thị dạng hàng

Chọn OK

Lưu Báo cáo.

2.2.Tạo báo cáo sử dụng Report Wizard

Click biểu tượng Report trong cửa sổ database (hoặc chọn View/Report)

Chọn New

Chọn bảng hoặc Truy vấn làm nguồn dữ liệu báo cáo .

Chọn Report Wizard

Chọn OK

Chọn các trường cần thiết cho báo cáo.

Chọn **Next**

Chọn các trường cần nhóm, chọn **Next**

Chọn các trường cần sắp xếp, chọn **Next**

Chọn dạng thể hiện của Report, chọn **Next**

Chọn nền thể hiện của Report, chọn **Next**

Đặt tiêu đề cho Report, chọn **Finish**

3. TẠO ĐIỀU KHIỂN TRONG BÁO CÁO

3.1. Tạođiều khiển Text Box (Hộp văn bản)

Text box có nhiều tính năng, có thể là điều khiển bị buộc, không buộc hay dùng tính toán. Nếu trong báo cáo có cả 3 loại điều khiển này, chúng ta nên ưu tiên điều khiển bị buộc trước.

Tạo một text box bị buộc và không buộc

Chúng ta buộc điều khiển Text box vào một trường bằng cách chỉ định trường cho text box đó lấy dữ liệu. Text box đó lấy dữ liệu có thể thực hiện bằng cách kéo trường muốn buộc vào điều khiển từ danh sách trường (Field List) vào biểu mẫu đang thiết kế.

Cách khác để tạo Text box là dùng hộp công cụ (Toolbox), sau đó gõ tên trường muốn buộc vào hộp văn bản hoặc bảng thuộc tính của điều khiển.

Dùng danh sách trường là phương pháp tốt nhất để tạo một điều khiển Text box bị buộc vì hai lý do sau:

Điều khiển đó được hệ thống tự động gắn nhãn và nhãn lấy tên trường được kéo làm tiêu đề.

Text box bị buộc đó thừa kế các thiết lập thuộc tính của trường từ bảng hay truy vấn.

Muốn chuyển một điều khiển không buộc thành bị buộc, lập thuộc tính Control Source của điều khiển thành một trường.

Một số thao tác khi thực hiện thiết kế báo cáo.

Mở báo cáo trong chế độ Design View

Chọn Field List từ menu View (Hoặc click biểu tượng Field List trên thanh công cụ)

Tạo điều khiển Text box bị buộc.

Mở bảng danh sách trường, chọn trường hoặc các trường muốn đặt vào báo cáo

Chọn một trường, click vào trường đó.

Chọn nhiều trường liền nhau, clik trường đầu, giữ phím shift, click trường cuối.

Chọn nhiều trường không liền nhau, giữ phím Ctrl và lần lượt click từng trường.

Chọn tất cả các trường trong danh sách, nhấp kép vào thanh tiêu đề của danh sách trường.

Kéo trường (hoặc các trường) được chọn và đặt vào một vị trí trên mẫu báo cáo.

Click biểu tượng Simple preview trên thanh công cụ để xem kết quả.

3.2. Tạo các điều khiển khác dùng hộp công cụ

Muốn tạo các điều khiển không buộc hay dùng để tính toán, phải dùng công cụ trong Toolbox.

Chọn View/Toolbars để hiển thị thanh công cụ

Tạo điều khiển dùng Toolbox

Click công cụ tương ứng loại điều khiển muốn tạo trong báo cáo.

Tạo điều khiển bị buộc bằng cách chọn một trường trong Field list và kéo nó vào trong báo cáo.

3.3. Tạo điều khiển dùng tính toán

Click vào biểu tượng Text box trong hộp công cụ

Click vào một vị trí trên báo cáo. Access tự động gắn nhãn cho điều khiển vừa tạo, tiêu đề mặc nhiên thường có dạng “Field0”, có thể thay đổi tiêu đề này theo ý thích.

Đưa con trỏ vào bên trong Text box.

Gõ dấu = , theo sau là biểu thức muốn lập

Ví dụ: =[SOLUONG]*[DONGIA]

Click vào Sample Preview để xem kết quả.

3.4. Tạo điều khiển nhãn

Nhãn là một điều khiển không buộc, nội dung nhãn không thay đổi từ trang này qua trang khác hay từ bản ghi này qua bản ghi khác.

Click biểu tượng Label trong Toolbox.

Click vào một vị trí trên báo cáo để tạo nhãn. Nhãn được tạo như vậy sẽ có kích thước tự mở rộng khi gõ nội dung vào.

Nếu muốn trình bày văn bản thành nhiều dòng, bấm Ctrl+Enter cuối dòng thứ nhất.

4. CÁC THUỘC TÍNH CỦA ĐIỀU KHIỂN TRONG BÁO CÁO

Thuộc tính xác định các đặc trưng của đối tượng, mỗi điều khiển trong báo cáo cũng có những thuộc tính riêng. Muốn mở bảng thuộc tính của điều khiển, chọn điều khiển đó và click biểu tượng Properties trên thanh công cụ.

Ta xét một số thuộc tính sau:

Cangrow: Dùng thuộc tính này để làm cho Text box có thể tự điều chỉnh kích thước theo phương dọc đối với khối dữ liệu chứa trong trường nó bị buộc.(chọn Yes).

CanShrink: Khi Text box khôngcó dữ liệu hoặc dữ liệu là chuỗi rỗng Access sẽ chừa trống chỗ đó trên giấy. Điều này có thể làm cho báo cáo quá trống trải nếu có nhiều chỗ như vây. Chúng ta lập thuộc tính của Text box này thành Yes.

HideDuplicate: Dùng thuộc tính này để che Text box khi giá trị trong đó trùng bản ghi trước

5. SẮP XẾP VÀ TẬP HỢP DỮ LIỆU THEO NHÓM

Sắp xếp là phương pháp phổ biến nhằm tổ chức dữ liệu theo một trật tự nào đó để tìm kiếm và phân loại thông tin.

5.1. Sắp xếp dữ liệu

Khi in báo cáo người dùng thường muốn tổ chức các bản ghi theo một trật tự nào đó. Ví dụ in danh sách cán bộ theo thứ tự giảm dần của lương.

Các bước thực hiện sắp xếp trên báo cáo.

Mở báo cáo ở chế độ Design View.

Chọn Sorting And Grouping trong menu View.

Trong hộp thoại.

Field/Expression: Chỉ định sắp xếp theo trường hoặc biểu thức nào đó.

Sort Order: Chọn Tăng dần hoặc giảm dần.

5.2. Nhóm dữ liệu

Trong nhiều báo cáo, sắp xếp các bản ghi không cũng chưa đủ mà cần phân thành các nhóm. Nhóm là tập hợp các bản ghi cùng với thông tin tóm lược tiêu biểu cho một thể loại thông tin. Một nhóm thường được cấu tạo như sau:

Tiêu đề nhóm (group header), nhóm con (nếu có), các bản ghi chi tiết và chân nhóm (Group footer)

Tiêu đề nhóm 1

Tiêu đề nhóm 2

Tiêu đề nhóm 3

.....

.....

Tiêu đề nhóm 10

Các bản ghi chi tiết

Chân nhóm 10

.....

.....

Chân nhóm 3

Chân nhóm 2

Chân nhóm 1

Ch□□ng 7 MACRRO

1. KHÁI NIỆM

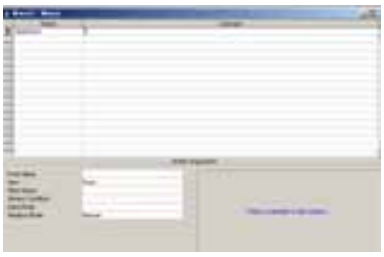
Macro trng MS Access là tập hợp các lệnh (Hành động, hành động.....) được định sẵn nhằm tự động thực hiện chuỗi các tác vụ nào đó mà không cần sự can thiệp từng bước của người sử dụng. Macro có thể liên kết các đối tương trong tập tin cơ ở dữ liệu (CSDL) như: Table, Query, form, report..... nhằm tạo ra các ứng dụng để khai thác có hiệu quả..

Macro được dùng khi có các hành động nào thường xuyên lặp lại trong MS Access hoặc được dùng khi cần kết hợp các hành động đơn giản nhằm giải quyết một vấn đề nào đó khi xây dựng các ứng dụng. Việc tự động hoá các hành động này bởi macro sẽ được thực hiện một cách nhanh chóng và chính xác.

2.TẠO VÀ THI HÀNH MỘT MACRO

2.1.Tạo một macro

Tại cửa sổ database, chọn đối tượng Macro/ New



Trong mục Action : chọn các hành động cho Macro.

Tong mục Action arguments: Chọn các tham số tương ứng cho hành động.

Lưu trữ Macro: File/save.

2.2.Thi hành macro

Tại cửa sổ database: Chọn đối tượng Macro/Run.

3. CÁC HÀNH ĐỘNG VÀ CÁC THAM SỐ

Ms Acces cung cấp một số hành động để tạo macro, mỗi hành động thực hiện một tác vụ nào đó trên CSDL và tuỳ thuộc vào tham số của mỗi hành động.

Open table: Mở 1 bảng dữ liệu

Table name: Tên bảng cần mở

View: Chọn dạng thể hiện của bảng (Datasheet/ Design/ Print Preview

Data mode:

Add: Dùng để nhập dữ liệu

Edit: Dùng để thêm, xoá, sửa dữ liệu trong bảng

Read Only: Mở bảng để chỉ xem dữ liệu

Open Query: Mở 1 truy vấn

Query name: Tên truy vấn cần mở

View: Chọn dạng thể hiện của bảng (Datasheet/ Design/ Print Preview

Data mode:

Add: Dùng để nhập dữ liệu

Edit: Dùng để thêm, xoá, sửa dữ liệu trong bảng

Read Only: Mở bảng để chỉ xem dữ liệu

Open Form: Mở 1 biểu mẫu

Form name: Tên biểu mẫu

View: Chọn dạng thể hiện (Form/ Design/ Print preview/ Datasheet)

Filter name: Tên Query lọc các dữ liệu để hiển thị trong form.

Where condition: Điều kiện lọc dữ liệu hiển thị trong form.

Data mode:

Add : Dùng để nhập dữ liệu

Edit : Dùng để thêm, xoá, sửa dữ liệu trong bảng

Read Only : Mở bảng để chỉ xem dữ liệu

Window mode:

Normal : Dạng cửa sổ form bình thường.

Hidden : Dạng cửa sổ form được ẩn đi.

Icon : Cửa sổ form thu nhỏ thành 1 biểu tượng.

Dialog : Dạng hộp thoại.

Open Report: Mở 1 báo cáo

Report name: Tên báo cáo

View: Chọn kiểu in

Print preview:In ra màn hình.

Design: Dạng thiết kế báo cáo

Print: In ra máy in

Filter name: Tên Query lọc các dữ liệu để hiển thị trong Report.

Where condition: Điều kiện lọc dữ liệu hiển thị trong Report.

Run macro: Thực hiện một tập lệnh

Macro name: Tên macro cần thực hiện

Repeat count: Số lần thực hiện macro sẽ lặp lại.

Repeat Expression: Biểu thức điều kiện để lặp lại khi thực hiện macro. Macro chỉ dừng khi khi biểu thức điều kiện nhận giá trị False.

Open module: Mở cửa sổ soạn thảo thủ tục trong 1 module.

Module name: Tên module chứa thủ tục cần mở.

Procedure name: Tên thủ tục sẽ mở.

Run code: Gọi thực hiện một hàm của Access Basic

Function name: Tên hàm cần thực hiện và các đối số của hàm.

Run App: Cho thực hiện một ứng dụng nào đó trong môi trường Windows

Command line: đường dẫn đến tập tin của một ứng dụng.

Run SQL: Cho thực hiện câu lệnh SQL

SQL Statement: Nội dung câu lệnh SQL

Maximize: Cực đại cửa sổ hiện thời

Minimize:Cực tiểu cửa sổ hiện thời thành một biểu tượng.

Restore: Phục hồi cửa sổ trở về kích thước cũ.

Move size: Di chuyển hoặc thay đổi kích thước cửa sổ hiện thời.

Right: Khoảng cách từ góc trên trái của cửa sổ này đến cạnh trái của cửa sổ chứa nó.

Down: Khoảng cách từ góc trên trái của cửa sổ này đến đến cạnh trên của cửa sổ chứa nó.

Width: Chiều rộng của cửa sổ này.

Height: Chiều cao của cửa sổ này.

Stop Macro: Dừng macro đang thực hiện

Beep: Phát tiếng kêu bíp

Hourglass: Đổi dạng con trỏ thành đồng hồ cát trong khi macro đang chạy

Hourglass On: Yes/No (Đổi/ Không đổi)

Close: Đóng một cửa sổ đang hoạt động

Object Type: Loại cửa sổ của đối tượng cần đóng như Table, Query, form, Report, Macro hoặc Module.

Object name: Tên của đối tượng cần đóng.

Quit: Thoát khỏi MS Access và trở về Windows

Option

Prompt: Hiển thị hộp thoại có lưu trữ không? Nếu đối tượng có thay đổi.

Save all: Lưu trữ tất cả mọi đối tượng.

Exit: Thoát mà không cần lưu trữ.

Print: In đối tượng hiện thời

Print Range: Phạm vi cần in ấn.

All: In tất cả các đối tượng

Selection: In phần trang được chọn

Pages: In các trang được chọn

Page from: Trang bắt đầu in

Page to: Trang kết thúc in

Print Quality: Chất lượng in

Copies: Số bản cần in

Collate Copies: Có sắp xếp thứ tự các bản in teo trang.

Msg Box: Hiển thị hộp thông báo

Message: Câu thông báo cần hiển thị

Beep: Yes/ No: Có/ Không phát ra tiếng Bíp khi hiển thị hộp thông báo.

Type: Loại hộp thông báo.

Title: Tiêu đề của hộp thông báo.

CancelEvent: Huỷ bỏ một sự kiện đang thực hiện

Requery: Cập nhật dữ liệu cho một đối tượng đang hoạt động bằng cách cập nhật lại dữ liệu nguồn của đối tượng đó.

Control name: Tên của đối tượng cần cập nhật dữ liệu (Nếu không chỉ ra thì sẽ cập nhật lại dữ liệu nguồn của chính đối tượng đang hoạt động).

Select Object: Chọn đối tượng trong CSDL

Object Type: loại đối tượng cần chọn.

Object name: Tên đối tượng cần chọn

In Database Window: (Yes/No) Xác định MS access có chọn đối tượng trong cửa sổ CSDL không, mặc định là No.

Set value: Gán một giá trị cho 1 trường, 1 điều khiển, hoặc một thuôc tính trên một Form hoặc 1 Report.

*Item:*Tên trường, đối tượng hay thuộc tính muốn gán giá trị.

Expression: Biểu thức cần gán giá trị cho Item.

☛ ***Chú ý:*** Nếu tên trường, tên đối tượng, tên thuộc tính ở 1 Form hoặc 1 Report khác thì phải mô tả đầy đủ.

Trong Form khác: [Forms]![Tên Form]![Tên trường/Tên đối tượng]

Trong Report khác: [Reports]![Tên Report]![Tên trường/Tên đối tượng]

Đối với các thuộc tính

[Forms/Reports]![Tên Form/Tên Report]![Tên trường].[Tên thuộc tính]

Add menu: Tạo thêm một Drop Down Menu vào một menu bar cho một form hoặc Report.

Menu name: Tên của Drop Down Menu muốn thêm vào menu bar.

Menu macro name: Tên macro chứa các lệnh về việc tạo menu.

Status bar: Thông báo ở thanh trạng thái khi chọn menu này.

Apply Filter: Lọc (Truy vấn) các dữ liệu khi xử lý Table, Form, Report.

Filter name: Tên của truy vấn lọc dữ liệu.

Where condition: Điều kiện lọc dữ liệu.

FindRecord: Tìm bản ghi đầu tiên nằm trong phạm vi và thoả mãn điêu kiện.

Find What: Nội dung dữ liệu cần tìm là một giá trị hoặc một biểu thức, nếu là biểu thức sẽ bắt đầu dấu "=".

Where: Qui định cách so sánh giá trị cần tìm với giá trị của trường.

Any part of field: Một phần bất kỳ của trường.

Match Whole field: Giá trị cần tìm bằng giá trị của trường.

Start of field: Giá trị cần tìm là phần đầu của trường.

Match Case: Yes/No: Có/Không phân biệt chữ in hoa và chữ in thường.

Direction: Quy định hướng tìm

All: Tìm toàn bộ

Up: Tìm từ bản ghi hiện thời lên phía trên.

Down: Tìm từ bản ghi hiện thời lên phía dưới.

Search As Formatted: Qui định việc tìm có dựa trên dữ liệu sau khi đã định dạng trong các trường hay không?

Search in: Qui định việc tìm trên trường hiện thời hoặc trên tất cả các trường.

Current Field: Tìm trên trường hiện thời.

All Fields: Tìm tất cả các trường.

Find First: Yes/No: Qui định tìm từ bản ghi đầu tiên hay tìm từ bản ghi hiện thời.

Findnext:Tìm bản ghi kế tiếp thoả mãn điều kiện tìm kiếm của lệnh FindRecord.

CopyObject: Sao chép một đối tượng trong tập tin CSDL hiện thời thành một đối tượng khác của tập tin CSDL khác trong MS Access.

Destionation Database: Tên tập tin CSDL đích

New name: Tên mới của đối tượng sau khi sao chép.

Source Object Type: Kiểu của đối tượng nguồn.

Source Object Name: Tên của đối tượng nguồn.

DeleteObject: Xoá một đối tượng trong tập tin CSDL hiện thời.

Object Type: Kiểu của đối tượng.

Object Name: Tên của đối tượng

4. NHÓM TẬP LỆNH VÀ TẬP LỆNH CÓ ĐIỀU KIỆN

4.1. Nhóm tập lệnh

Là Macro chứa các macro con, thay vì tạo ra nhiều macro với nhiều tên khác nhau thì các macro này được gom lại thành một tên chung nhằm giảm bớt số lượng và thuận lợi trong quá trình sử dụng. Tuy nhiên các Macro được nhóm khi chúng có liên quan với nhau.

Có thể có nhiều Macro trong nhóm có cùng hành động, tuy nhiên chúng được phân biệt bởi tên Macro.

Đặt tên cho Macro ta thực hiện:

Tại chế độ thiết kế Macro:

View/Macro name

Đặt tên cho Macro tại cột Macro name.

Cách thực hiện 1 macro trong macro name

<Tên Macro Group>.<Tên Macso cần thực hiện>

4.2. Macro có điều kiện

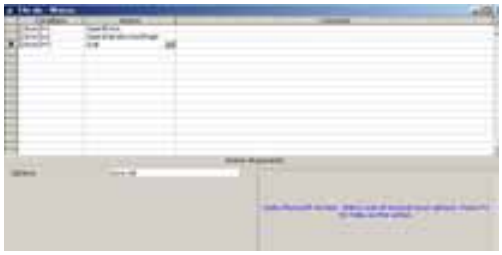
Là macro có chứa các điều kiện thi hành cho mỗi hành động.

Cách tạo Macro có điều kiện

Tại chế độ thiết kế Macro: Chọn View/Conditions

Tại cột Condition : Đặt điều kiện thi hành cho mỗi hành động.

Ví dụ



4.3. Áp dụng Macro cho form và Report

4.3.1. Quy tắc chung khi gọi một đối tượng

Đối với form : *Forms![Tên form]![Tên đối tượng]*

Đối với Report : *Reports![Tên Report]![Tên đối tượng]*

4.3.2. Các thuộc tính của một đối tượng

Muốn gắn một nút lệnh trên một biểu mẫu hoặc báo cáo với một Macro nào đó vào nút lệnh này ta thực hiện: Click chuột phải vào nút lệnh, chọn Properties và gắn Macro vào các hành động tương ứng.

On Enter: Macro thi hành khi nhấn Enter vào bên trong đối tượng

On Exit: Macro thi hành khi thoát khỏis đối tượng

On Got Focus: Thiết lập nhận biết khi có di chuyển con trỏ đến một form hoặc 1 trường trên form đang mở.

On Click: Macro thi hành khi click vào đối tượng

On Dbl Click: Macro thi hành khi Double click vào đối tượng.

On Mouse Down: Macro thi hành khi ấn và giữ chuột tại đối tượng

On Mouse Move: Macro thi hành khi di chuyển chuột ra khỏi đối tượng

On Mouse Up: Macro thi hành khi nhã chuột ra khỏi đối tượng.

On Key Down: Macro thi hành khi ấn và giữ một phím đối tượng.

On Key Press: Macro thi hành khi ấn một phím đối tượng.

On Key Up: Macro thi hành khi nhã một phím đối tượng.

4.3.3. Macro tự động thực hiện sau khi mở tập tin CSDL

Chúng ta có thể tạo ra một Macro mà mỗi khi mở một tập tin CSDL thì Macro này tự động thực hiện.

Để tạo Macro tự động thực hiện ta tiến hành các thao tác sau:

Tạo macro

Lưu trữ Macro với tên AutoExec.

5. THIẾT KẾ MENU TRONG ACCESS

Trong các ứng dụng, thường chúng ta phải tổ chức Menu để cho phép người sử dụng thực hiện các hành động thông qua các chức năng trên Menu này.

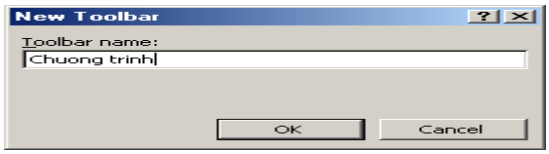
Có 2 cách hiển thị các Menu của người sử dụng (Custom Menu) trong các ứng dụng đó là:

Menu ứng với 1 form xác định nào đó: Menu này chỉ xuất hiện khi truy xuất đến Form này.

Menu toàn cục (Globally): Menu toàn cục này sẽ xuất hiện trong các ứng dụng và nó chỉ được thay thế khi có 1 menu ứng với form nào đó được mở.

5.1. Tạo menu của người sử dụng

Chọn View/Toolbars/Customize/New/ Đặt tên cho thanh menu

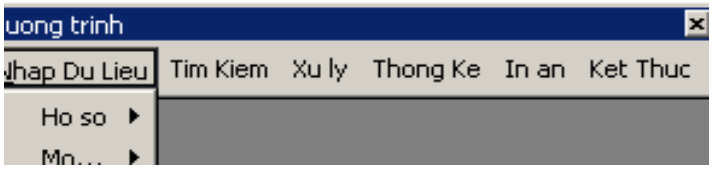


Chọn Properties và chọn Type là Menu bar, close

Chọn command, trọng mục Categories chọn New menu và kéo sang thanh menu vừa tạo.

Click chuột phải để thay đổi các tiêu đề cho phù hợp trên thanh menu.

Tương tự cho các nhóm khác.



Có thể tạo các chức năng là các hành động như mở bảng, truy vấn, biểu mẫu.... bằng cách kéo các biểu tượng này trong mục command vào menu đang tạo.

5.2. Tạo menu toàn cục

Nếu muốn tạo một menu toàn cục sẽ thay thế menu có sẵn trong Access khi CSDL này được mở ta thực hiện các bước sau:

Mở hoặc tạo ra một Macro có tên là Autoexec.

Thêm vào hành động: SetValue

Trong mục các tham số (arguments):

Item: Application.menubar

Expression: Tên của Menubar người sử dụng tạo ra và đặt giữa cặp dấu ngoặc kép "".

BÀI SỐ 1

Khởi động MSACCESS, tạo một CSDL có tên NHANSU.MDB rồi lần lượt tạo các bảng dữ liệu sau

Bảng 1: Nhanvien

STT	Fieldname	Data Type	Fieldsize	Note
1	**Manv**	Text	4	Khoá chính
2	Holot	Text	20	
3	Ten	Text	10	
4	Ngaysinh	Date/time	8	
5	Gioitinh	Yes/No	1	**Yes:Nam, No:Nu**
6	Maphong	Text (Lookup)	2	Lấy từ bảng Dsphong
7	MaPXuong	Text (Lookup)	2	Lấy từ bảng dspxuong
8	Diachi	Text	30	
9	Ghichu	Memo		

Bảng 2: Dsphong

STT	Fieldname	Data Type	Fieldsize	Note
1	**Maphong**	Text	2	Khoá chính
2	Tenphong	Text	30	
3	Sodthoai	Text	11	

Bảng 3: Thunhap_NV

STT	Fieldname	Data Type	Fieldsize	Note
1	**Manv**	Text (Lookup)	4	Khoá chính
2	Luongchinh	Number	Double	
3	Heso	Number	Integer	
4	Phucap	Number	Double	
5	Thue	Number	Double	Thuế
6	Thamnien	Number	Byte	Số năm thâm niên
7	Tongluong	Number	Double	

Bảng 4: DSpxuong

STT	Fieldname	Data Type	Fieldsize	Note
1	**Mapxuong**	Text	2	Khoá chính
2	Tenpxuong	Text	20	
3	Sodoanvien	Number	Byte	Số đoàn viên

1. Hãy nhập dữ liệu thích hợp cho các bảng trên.

2. Mở bảng Nhanvien sắp xếp tăng dần theo **manv**, sắp xếp giảm dần theo **ten**.

3. Sắp xếp tăng dần theo **ten**, nếu **ten** trùng nhau thì sắp xếp giảm dần theo **holot** nếu **holot** trùng nhau thì sắp xếp giảm dần theo **maphong**.

4. Lọc và hiển thị danh sách những nhân viên có tên **Thanh**.

5. Lọc và hiển thị những nhân viên có họ **Nguyen** hoặc **Cao** và có tên **anh**.

6. Lọc và hiển thị nhữnh nhân viên có **Manv** bắt đầu là **B** và **maphong** là **A1** hoặc **Manv** bắt đầu là **C** và **maphong** là **A2**.

7. Lọc và hiển thị những nhân viên **nam** và có năm sinh **1975**.

8. Lọc và hiển thị những nhân viên **nam** sinh trong thời gian từ **12/12/76** đến **12/12/79**.

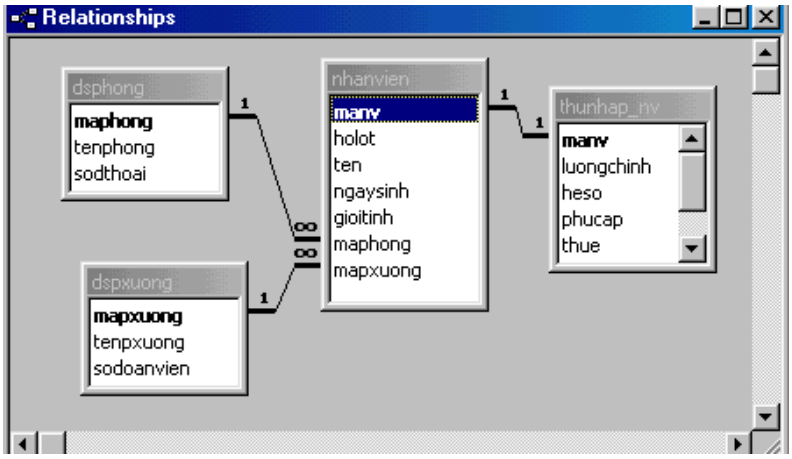
9. Lọc và hiển thị những nhân viên **nữ** có **Mapxuong** là **P1** và sinh trong tháng **10** năm **1975** hoặc nhân viên **nam** có năm sinh **1972** đến **1976**.

10. Mở bảng **Thunhap_nv** để hiển thị những nhân viên có **heso>100 và** có **luongchinh<500.000** hoặc có **phucap** từ **100.000** đến **500.000**.

BÀI SỐ 2

1. Mở CSDL NHANSU.MDB trong bài thực hành số 1 rồi thực hiện các nhiệm vụ sau:

- ❖ Đặt khoá chính cho trường Manv (nhanvien), Maphong(dsphong), manv(thunhap_nv) và mapxuong (dspxuong).
- ❖ Thiết lập các mối quan hệ giữa các bảng dữ liệu theo sơ đồ sau



❖ Hãy thiết lập các thuộc tính tham chiếu toàn vẹn và các thuộc tính như Format, caption, inputmask..... cho các trường một cách hợp lệ.

Đặt thuộc tính Format của trường **ngaysinh :** dd-mm-yy , Inputmask: 00/00/00

Đặt thuộc tính Inputmask của trường **sodthoai** : (000)000000

2. Tạo một truy vấn **query1** để hiển thị những thông tin sau:

Manv, holot, ten (trong bảng **nhanvien**), Luongchinh, phucap (Trong bảng **Thunhap_nv**), tenphong, maphong (Trong **dsphong**).

3. Tạo một truy vấn **query 2** để hiển thị những thông tin sau:

Manv, hoten (Nối holot và ten), maphong, tenphong, mapxuong, tenpxuong, tongthunhap trong đó tongthunhap được tính theo công thức:

Tongthunhap=luongchinh*heso+phucap-thue nếu là nhân viên **nam**.

Tongthunhap=luongchinh*heso+phucap nếu là nhân viên **nữ** .

4. Tạo một truy vấn **query3** để hiển thị những nhân viên có **maphong** là **p1** và **mapxuong** là **x1** sinh trong tháng **7** năm **1976** bao gồm những tin sau:

Hoten, gioitinh, maphong, tenphong, mapxuong, tenpxuong.

5. Tạo một truy vấn **query4** để hiển thị những nhân viên **nữ** và có **maphong** là **p2** hoặc **p3** sinh trong ngày **20** tháng **12** bao gồm những thông tin:

Hoten, maphong, ngaysinh, gioitinh (chú ý giá trị trường gioitinh phải hiển thị nam hoặc nữ).

6. Tạo một truy vấn **query5** để hiển thị những nhân viên **nam** có **tongthunhap>=60000** hoặc thuộc **maphong** là **p2** và không phải họ **Nguyãùn** bao gồm những thông tin: Ten, maphong, tenphong, tongthunhap.

7. Tạo một truy vấn **query6** để hiển thị những nhân viên **nữ** có **maphong** không bắt đầu là **p** hoặc những nhân viên thuộc phân xưởng không có đoàn viên nào bao gồm những thông tin sau: Holot, ten, gioitinh, tenpxuong.

8. Tạo một truy vấn **query7** để hiển thị những nhân viên **nam** tên **Thanh** hoặc **Long** hoặc những nhân viên **nữ** không phải họ **Lê** hoặc **trần** bao gồm những thông tin: Hoten, gioitinh, maphong, tenphong.

9. Tạo một truy vấn **query8** để hiển thị những nhân viên **nam** sinh trong khoảng thời gian từ năm **1973** đến **1980** thuộc phân xưởng có **mapxuong** là **x2** hoặc những nhân viên không có **thuế** sinh trong tháng **4** đến tháng **8** năm **1975** bao gồm những thông tin: Holot, ten, thangsinh, namsinh, mapxuong, tenpxuong.

10. Tạo một truy vấn **query9** để hiển thị những thông tin: Hoten, namsinh, thamnien, luongchinh, trong đó nếu

Luongchinh>=3000 và nhân viên **nữ** thì thamnien là **35**.

Luongchinh>=4000 và nhân viên **nam** thì thamnien là **30**.

BÀI SỐ 3

Tạo một CSDL có tên QLTV.MDB, rồi lần lượt tạo các bảng dữ liệu sau:

Bảng 1: **Loaisach**

STT	Fieldname	Data Type	Fieldsize	Note
1	**Masach**	Text	4	Khoá chính
2	Tensach	Text	20	
3	Tentacgia	Text	20	
4	Namxb	Date/time	8	
5	Soluongco	Number	3	Số lượng có
6	Sotrang	Number	5	Số trang sách
7	Manxb	Text	4	Mã nhà xuất bản

Bảng 1: **Docgiamuon**

STT	Fieldname	Data Type	Fieldsize	Note
1	Hoten	Text	30	
2	Quequan	Text	30	
3	**Madocgia**	Text	4	Khoá chính
4	Masach	Text (Lookup)	4	Lấy dữ liệu từ Loaisach
5	Sluong	Number	1	Số lượng mượn
6	Ngaymuon	Date/time	8	Ngày mượn
7	Ngayhen	Date/time	8	Ngày hẹn trả

Bảng 3: **Docgiatra**

STT	Fieldname	Data Type	Fieldsize	Note
1	**Madocgia**	Text	4	Khoá chính
2	Masach	Text	4	
3	Ngaytra	Date/time	8	Ngày trả

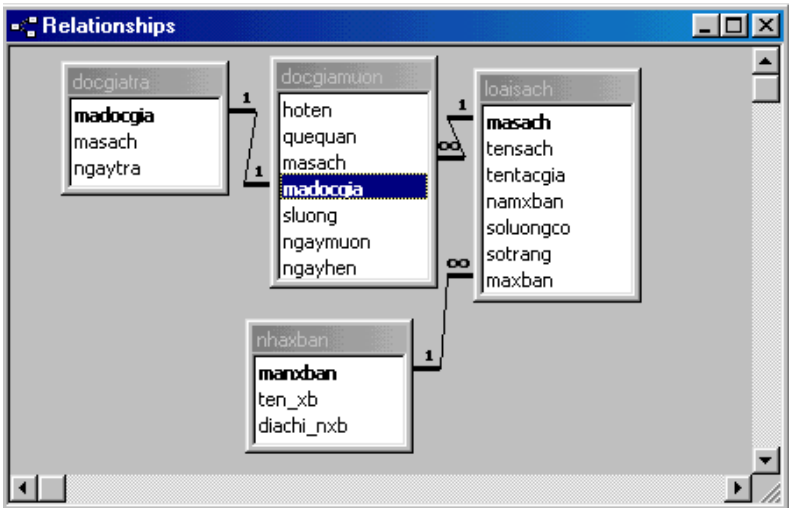
Bảng 4: : **Nhaxban**

STT	Fieldname	Data Type	Fieldsize	Note
1	**Manxban**	Text	4	Khoá chính
2	Ten_xb	Text	30	Tên nhà xuất bản
3	Diachi_nxb	Text	30	Địa chỉ Nhà XB

1. Nhập dữ liệu thích hợp cho các bảng dữ liệu.

2. Đặt khoá chính cho các trường: masach(Loaisach), Madocgia(Docgiamuon), Madocgia(Docgiatra), manxban(Nhaxban).

3. Thiết lập các mối quan hệ giữa các bảng dữ liệu theo sơ đồ sau



4. Tạo một truy vấn **TV1** để hiển thị danh sách độc giả mượn sách trong tháng 10 năm 98 bao gồm những thông tin: Hoten, ngaymuon, tensach, tentacgia.

5. Tạo một truy vấn **TV2** để hiển thị số lượng còn của mỗi loại sách của nhà xuất bản **Kim đồng** bao gồm những thông tin: tensach, nxban, soluongcon.

6. Tạo một truy vấn **TV3** để hiển thị danh sách những độc giả mượn sách quá hạn bao gồm những thông tin: Hoten, Ngaymuon, Ngayhen, songayqua (Số ngày quá hạn).

7. Tạo một truy vấn **TV4** để hiển thị danh sách những độc giả mượn sách quá hạn bao gồm những thông tin Hoten, sluong, mucquahan

Trong đó: Mucquahan là Mức 1 nếu songayqua<5

Mucquahan là Mức 2 nếu songayqua<10

Mucquahan là Mức 3 nếu songayqua>=10

8. Tạo một truy vấn **TV5** để hiển thị danh sách những độc giả trả sách đúng hạn hoặc sớm hạn bao gồm những thông tin: Hoten, masach, tensach.

9. Tạo một truy vấn **TV6** để hiển thị những loại sách có số trang >100 và số lượng còn là 20 của **nhà xuất bản giáo dục** hoặc tên sách có chữ **Tin học** bao gồm Tensach, sotrang, ten_xb, tentacgia.

BÀI SỐ 4

1. Sử dụng CSDL QLTV.MDB, tạo một truy vấn **Truyvan1** để hiển thị tổng số lượng mượn của từng loại sách trong thư viện.

2. Tạo một truy vấn **Truyvan2** để hiển thị tổng số lượng mượn của từng loại sách trong tháng **12** năm **1998**.

3. Tạo một truy vấn **Truyvan3** để hiển thị tổng số lượng mượn của từng loại sách theo từng tháng trong năm **1999**.

4. Tạo một truy vấn **Truyvan4** để hiển thị tổng số lượng mượn của từng loại sách theo từng tháng của một năm nào đó(Tháng và năm được nhập từ bàn phím).

5. Tạo một truy vấn **Truyvan8** để hiển thị số lượng mượn của từng loại sách trong năm **1999** và có số lượng mượn của mỗi độc giả **>2**.

6. Tạo một truy vấn **Truyvan9** để hiển thị tổng số loại sách có trong thư viện.

7. Tạo một truy vấn **Truyvan10** để hiển thị tổng số loại sách trong thư viện cho mượn trong năm **1998**.

8. Tạo một truy vấn **Truyvan11** để hiển thị tên của độc giả nào đó mượn sách trong **tháng 1 năm 2001** (Tên được nhập từ bàn phím).

9. Tạo một truy vấn **Truyvan12** để hiển thị tên sách và tên tác giả xuất bản sách trong năm **2002** (Tên sách nhập từ bàn phím, Họ của tác giả nhập từ bàn phím).

10. Tạo một truy vấn tham khảo chéo **Truyvan13** để phản ánh tổng số mỗi loại sách mượn cho mượn trong **tháng 11 năm 2000**.

11. Tạo một truy vấn tham khảo chéo **Truyvan14** để hiển thị tổng số laọi sách xuất bản trong năm **1995**.

12. Tạo một truy vấn **Truyvan14** để xoá những sinh viên đã mượn sách quá hạn **5** ngày.

BÀI SỐ 5

Tạo một CSDL có tên QLSV.MDB, rồi tạo các bảng dữ liệu sau:

Bảng 1: **DSSV**

STT	Fieldname	Data Type	Fieldsize	Note
1	**Masv**	Text	4	Khoá chính
2	Malop	Text	4	
3	Hotensv	Text	30	
4	Ngaysinh	Date/time	8	
5	Quequan	Text	30	
6	Gioitinh	Yes/No	1	
7	Hocbong	Number	Double	

Bảng 2: DSDIEM

STT	Fieldname	Data Type	Fieldsize	Note
1	**Masv**	Text	4	Khoá chính
2	Mamon	Text	2	Mã môn học
3	Diem_lan1	Number	Double	Điểm thi lần 1
4	Diem_lan2	Number	Double	Điểm thi lần 2

Bảng 3: DSLOP

STT	Fieldname	Data Type	Fieldsize	Note
1	**Malop**	Text	4	Khoá chính
2	Tenlop	Text	20	
3	Nganh_hoc	Text	30	
4	Khoahoc	Text	2	

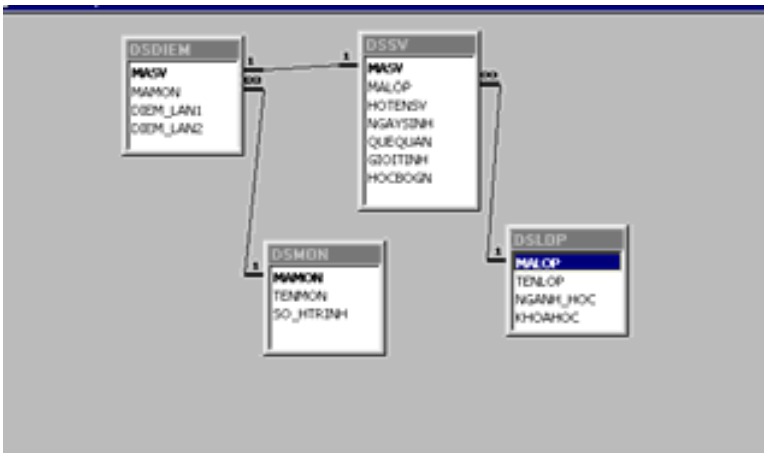
Bảng 3: DSMON

STT	Fieldname	Data Type	Fieldsize	Note
1	**Mamon**	Text	2	Khoá chính
2	Tenmon	Text	20	
3	So_htrinh	Number	Byte	Số học trình

1. Nhập dữ liệu thích hợp cho các bảng dữ liệu.

2. Đặt khoá chính cho các trường: MASV(DSSV), MASV(DSDIEM), MALOP(DSLOP), Mamon(DSMON).

3. Thiết lập các mối quan hệ giữa các bảng theo sơ đồ sau:



4. Tạo một truy vấn **BT1** để hiển thị tổng số sinh viên của mỗi lớp là bao nhiêu?

5. Tạo một truy vấn tham khảo chéo **BT2** để phản ánh tổng số sinh viên xếp loại ***Xsắc, Giỏi, khá, Tb, Yếu*** của mỗi lớp.

6. Từ bảng dữ liệu DSSV, tạo một truy vấn **BT3** để tạo ra bảng dữ liệu DSLUU lưu trữ những sinh viên có Mã lớp bắt đầu là **T** hoặc **H.**

7. Từ bảng dữ liệu DSSV, tạo một truy vấn **BT4** để tạo ra một bảng dữ liệu DSNAM để lưu trữ những sinh viên **nam** sinh trong năm **1976** bao gồm các thông tin: HOTENV, QUEQUAN, GIOTINH.

8. Từ bảng dữ liệu DSSV, tạo một truy vấn **BT5** để tạo ra một bảng dữ liệu DSNU để lưu trữ những sinh viên **nữ** sinh trong **quý 3** năm **1979** bao gồm các thông tin: HOTENV, QUEQUAN, GIOTINH.

9. Tạo một truy vấn **BT6** để tăng học bổng cho những sinh viên **nữ** thêm **30%**.

10. Tạo một truy vấn **BT7** để nối dữ liệu từ bảng DSNAM vào bảng DSNU.

11. Tạo truy vấn **BT8** để xoá những sinh viên có quê quán ở **Đà Nẵng** và có mã lớp bắt đầu là **B**.

BÀI SỐ 6

1. Sử dụng ngôn ngữ SQL để tạo cấu trúc của các bảng dữ liệu sau:

DSTRUONG (**MATRUONG**, TENTRUONG, DIACHI)
DSKHOA (MATRUONG, **MAKHOA**, TENKHOA, SODT)
DANHSACH (**MASV**, MAKHOA, HOTEN, NGAYSINH, LOP, HOCBONG)
BANGDIEM (**MASV**, DTBK1, DTBK2, DTBK3, DTBK4)

Trong đó: Các trường in đậm và gạch chân là khoá chính, kiểu dữ liệu và kích thuớc của các được mô tả như sau:

Bảng DSTRUONG

MATRUONG TEXT(2), TENTRUONG TEXT(20), DIACHI TEXT(30)

Bảng DSKHOA

MATRUONG TEXT(2), MAKHOA TEXT(4), TENKHOA TEXT(10), SODT TEXT(6). Trường SODT được lập chỉ mục.

Bảng DANHSACH

MASV TEXT(4), MAKHOA TEXT(4), HOTEN TEXT(30), NGAYSINH(DATE/TIME), LOP TEXT(10), HOCBONG (DOUBLE). Trường MASV được lập chỉ mục.

Bảng BANGDIEM

MASV TEXT(4), DTBK1 (DOUBLE), DTBK2 (DOUBLE), DTBK3 (DOUBLE), DTBK4 (DOUBLE).

☞ **Chú ý**

Sau khi tạo cấu trúc các bảng dữ liệuxong, hãy nhập dữ liệu và thiết lập mối quan hệ giữa các bảng phù hợp. Trường HOCBONG trong bảngg DANHSACH chỉ nhập một trong ba giá trị 120000, 180000 hoặc 240000.

2. Sử dụng ngôn ngữ SQL để thay đổi cấu trúc của bảng dữ liệu

a. Thêm trường GHICHU có kiểu MEMO vào trong bảng DANHSACH

b. Thêm trường TBCONG có kiểu DOUBLE vào trong bảng BANGDIEM

c. Thêm trường QUEQUAN có kiểu TEXT và GIOITINH có kiểu YES/NO vào bảng DANHSACH và lập chỉ mục trường QUEQUAN.

3. Sử dụng ngôn ngữ SQL để tạo các truy vấn chọn sau

a. Chọn MATRUONG, MAKHOA, TENKHOA trong bảng DSKHOA.

b. Chọn MATRUONG, MAKHOA, SODT trong bảng DSKHOA của những trường có MATRUONG bắt đầu là Q.

c. Chọn MASV, MAKHOA, HOTEN của những sinh viên sinh trong khoảng thời gian từ 20/10/74 đến 20/10/76 trong bảng DANHSACH.

d. Chọn MASV, HOTEN, LOP, HOCBONG của những sinh viên có MASV bắt đầu là T và thuộc lớp Tin học hoặc Hoá học hoặc kinh tế trong bảng DANHSACH (HOTEN đổi thành Họ và tên).

e. Chọn những sinh viên có tên THANH sinh trong tháng 10/76 hoặc có HOCBONG trong khoảng từ 150000 đến 200000.

4. Sử dụng ngông ngữ SQL để tạo các truy vấn tính tổng sau:

a. Tạo một truy vấn để tính tổng HOCBONG của mỗi khoa.

b. Tạo một truy vấn thống kê xem mỗi khoa số lượng sinh viên là bao nhiêu?

c. Tạo một truy vấn để tính tổng HOCBONG của mỗi lớp trong mỗi khoa.

d. Tạo một truy vấn thống kê xem mỗi trường có bao nhiêu khoa?

5. Sử dụng ngôn ngữ SQL để tạo các truy vấn tham số sau:

a. Tạo một truy vấn tham số thống kê xem mỗi mức HOCBONG mỗi khoa có bao nhiêu sinh viên.

b. Tạo một truy vấn tham số để thống kê xem số lượng sinh viên sinh trong mỗi tháng của năm 1979 của mỗi khoa là bao nhiêu?

6. Sử dụng ngông ngữ SQL để tạo các truy vấn tạo bảng sau:

a. Tạo một bảng DIEMLUU gồm tất cả các trường trong bảng DIEMTHI.

b. Tạo một bảng DSLUU gồm các trường MASV, MAKHOA, HOTEN từ bảng DANHSACH của những sinh viên sinh trước ngày 20/11/76.

c. Tạo một bảng DSLUU1 gồm các trường HOTEN, LOP của những sinh viên thuộc lớp **Tin K25A** và sinh năm 1985 hoặc trước năm 1978.

7. Sử dụng ngôn ngữ SQL để tạo các truy vấn nối dữ liệu sau:

a. Tạo một truy vấn nối dữ liệu từ bảng DIEMLUU vào DIEMTHI nhưng chỉ gồm các trường DTBK1, DTBK2.

b. Tạo một truy vấn nối dữ liệu từ bảng DIEMLUU vào DIEMTHI nhưng chỉ gồm các trường DTBK1, DTBK2, DTBK3 đối với những bản ghi ghi có DTBK3>=8.

8. Sử dụng ngôn ngữ SQL để tạo các truy vấn cập nhật dữ liệu sau:

a. Tính giá trị trường DTBCONG của bảng DIEMTHI theo công thức (DTBK1+ DTBK2+ DTBK3+ DTBK4)/4.

b. Tăng HOCBONG thêm 100.000 cho những sinh viên có MASV bắt đầu là A trong bảng DANHSACH.

c. Giảm HOCBONG đi 50.000 cho những sinh viên có năm sinh từ 1975 đến 1978 hoặc những sinh viên có tên NHAN.

9. Sử dụng ngôn ngữ SQL để tạo các truy vấn xoá sau:

a. Tạo một truy vấn xoá những sinh viên thuộc khoa toán.

b. Tạo một truy vấn xoá những sinh viên sinh trong khoảng thời gian từ 20/10/74 đến 20/10/76 hoặc có HOCBONG=120.000.

c. Tạo một truy vấn xoá những sinh viên có họ NGUYEN hoặc tên THANH sinh trong tháng 7 năm 1978.

10. Sử dụng ngôn ngữ SQL để tạo các truy vấn dựa trên nhiều bảng sau:

a. Tạo truy vấn để hiển thị HOTEN, TENKHOA, LOP, HOCBONG từ 2 bảng dữ liệu DSKHOA và DANHSACH.

b. Tạo một truy vấn để hiển thị TENTRUONG, TENKHOA, HOTEN của những sinh viên sinh trong năm 1980 từ 3 bảng DSTRUONG, DSKHOA, DANHSACH.

c. Tạo một truy vấn để hiển thị TENTRUONG, TENKHOA, HOTEN, DTBK1, DTBK2 tử 4 bảng dữ liệu DSTRUONG, DSKHOA, DANHSACH, DIEMTHI.

11. Sử dụng ngôn ngữ SQL để tạo các truy vấn con sau:

a. Tạo một truy vấn để hiển thị HOTEN, NGAYSINH, LOP của những sinh viên có DTBK4>=5.

b. Tạo một truy vấn để hiển thị it nhất một sinh viên có DTBK2<=4.

12. Sử dụng ngôn ngữ SQL để tạo các truy vấn hội sau:

a. Tạo truy vấn hội để hiển thị HOTEN, DTBK1, DTBK2 từ 2 bảng dữ liệu DIEMTHI và DIEMLUU.

b. Tạo truy vấn hội để hiển thị HOTEN, DTBK1, DTBK2, DTBK3 từ 2 bảng dữ liệu DIEMTHI và DIEMLUU nhưng chỉ hiển thị những sinh viên có DTBK3>=7.

BÀI SỐ 7

Sử dụng CSDL **QLSV.MDB** trong **bài tập số 5** để thực hiện các yêu cầu sau:

1. Tạo một form có tên **Nhaplop** để nhập dữ liệu cho bảng **dslop** như sau:

2. Tạo một form chính phụ như sau:

3. Tạo một form có tên **Hienthi** để hiển thị danh sách sinh viên như sau :

4. Hãy thiết kế form theo yêu cầu sau:

Hãy gắn các chức năng phù hợp với các mục chọn.

5. Hãy thiết kế một **Report** theo yêu cầu sau:

BÀI SỐ 8

Cho CSDL **QLSVIEN.MBD** gồm các bảng dữ liệu sau:

DSSVIEN(**MASV**, MALOP, HOTENSVIEN, NGAYSINH, GTINH, QQUAN)

DSMONHOC(**MAMON**, TENMON, SOH_TRINH)

DSDIEMTHI(**MASV**, MAMON, DIEM_LAN1, DIEM_LAN2)

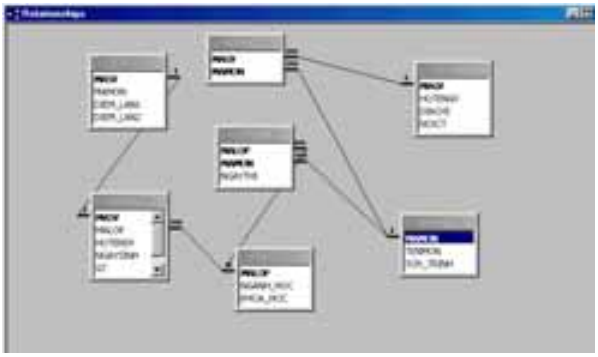
DSGIAOVIEN(**MAGV**, HOTENGVIEN, DIACHI, NOI_CT)

DSLOPHOC(**MALOP**, NGANH_HOC, KHOA_HOC)

GVMONHOC(**MAGV**, **MAMON**)

DSLOPMON(**MALOP**, **MAMON**, NGAYTHI)

Mối quan hệ giữa các bảng dữ liệu theo sơ đồ sau:



1. Tạo các biểu mẫu để cập nhật dữ liệu cho các bảng trên, chú ý sử dụng combo box hoặc List box đối với các trường cần thiết.

2. Tạo báo cáo để hiển thị danh sách sinh viên trong một lớp học nào đó, bao gồm cả ngành học tương ứng.

3. Tạo báo cáo để hiển thị danh sách sinh viên trong từng lớp học, bao gồm cả ngành học, trong đó các sinh viên cùng một lớp thì tên lớp được nhóm lại với nhau.

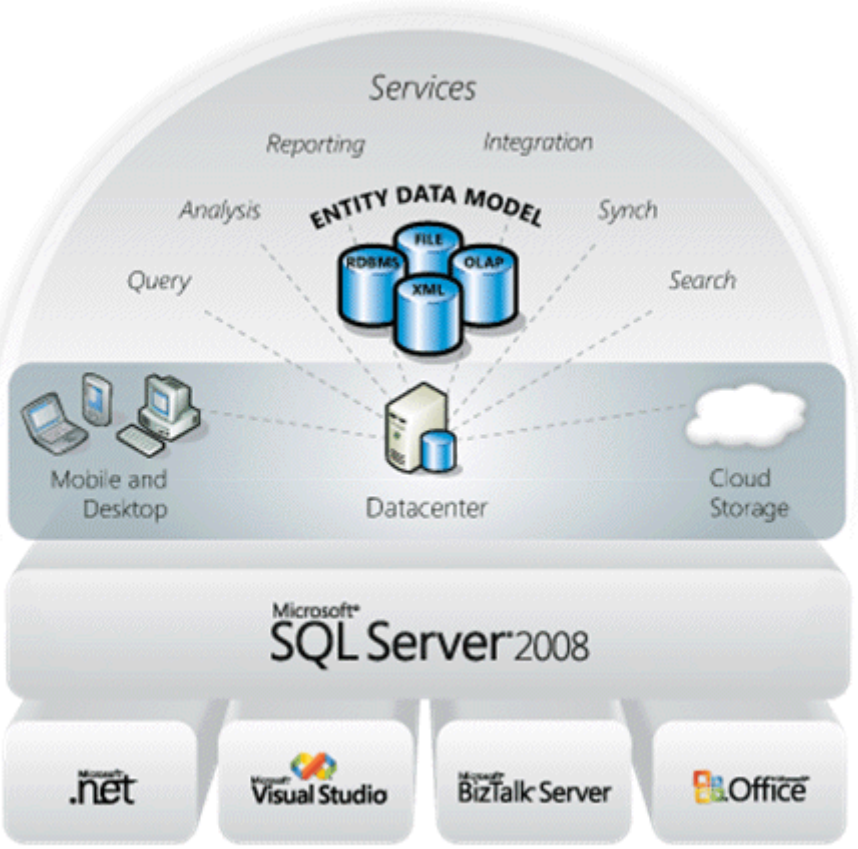
4. Tạo báo cáo để hiển thị danh sách giáo viên đã giảng dạy các một học cho một lớp nào đó.

5. Tạo báo cáo để hiển thị danh sách giáo viên đã tham gia giảng dạy các môn học cho nhiều lớp học, trong đó các giáo viên giảng dạy các môn học cho một lớp học được nhóm lại với nhau.

6. Tạo báo cáo để hiển thị điểm thi các môn học của một sinh viên.

7. Tạo báo cáo để hiển thị điểm thi một môn học của một lớp nào đó.

Giáo trình “Các hệ quản trị cơ sở dữ liệu



Chương 1 TỔNG QUAN CƠ SỞ DỮ LIỆU ACCESS

1. GIỚI THIỆU MICROSOFT ACCESS

Microsoft Access là một Hệ Quản Trị Cơ Sở Dữ Liệu (QTCSDL) tương tác người sử dụng chạy trong môi trường Windows. Microsoft Access cho chúng ta một công cụ hiệu lực và đầy sức mạnh trong công tác tổ chức, tìm kiếm và biểu diễn thông tin.

Microsoft Access cho ta các khả năng thao tác dữ liệu, khả năng liên kết và công cụ truy vấn mạnh mẽ giúp quá trình tìm kiếm thông tin nhanh. Người sử dụng có thể chỉ dùng một truy vấn để làm việc với các dạng cơ sở dữ liệu khác nhau. Ngoài ra, có thể thay đổi truy vấn bất kỳ lúc nào và xem nhiều cách hiển thị dữ liệu khác nhau chỉ cần động tác nhấp chuột.

Microsoft Access và khả năng kết xuất dữ liệu cho phép người sử dụng thiết kế những biểu mẫu và báo cáo phức tạp đáp ứng đầy đủ các yêu cầu quản lý, có thể vận động dữ liệu và kết hợp các biểu mẫu va báo cáo trong một tài liệu và trình bày kết quả theo dạng thức chuyên nghiệp.

Microsoft Access là một công cụ đầy năng lực để nâng cao hiệu suất công việc. Bằng cách dùng các Wizard của MS Access và các lệnh có sẵn (macro) ta có thể dễ dàng tự động hóa công việc mà không cần lập trình. Đối với những nhu cầu quản lý cao, Access đưa ra ngôn ngữ lập trình Access Basic (Visual Basic For application) một ngôn ngữ lập trình mạnh trên CSDL.

2. KHỞI ĐỘNG VÀ THOÁT KHỎI ACCESS

2.1. Khởi động ACCESS

Chọn nút Start trên thanh Task bar

Chọn Programs

Chọn Microsoft ACCESS

Khung hội thoại Microsoft ACCESS gồm:

Create a New Database Using : Tạo CSDL ứng dụng mới.

Blank Database : Tạo CSDL trống.

Database Wizard : Tạo với sự trợ giúp của Wizard.

Open an Existing Database : Mở một CSDL có sẵn.

2.2. Thoát khỏi ACCESS

Chọn File/Exit hoặc nhấn tổ hợp phím ALT+F4

3. CÁC THAO TÁC TRÊN TẬP TIN CƠ SỞ DỮ LIỆU ACCESS

3.1. Tạo một tập tin CSDL

Thực hiện các thao tác sau:

Chọn File/New hoặc chọn biểu tượng New trên thanh công cụ

Chọn Database, chọn OK

Trong mục *Save in:* Chọn thư mục cần chứa tên tập tin.

File name: Chọn tên tập tin cần tạo

(Phần mở rộng mặc định là MDB)

3.2. Mở một CSDL đã tồn tại trên đĩa

Chọn File/Open database (Hoặc click biểu tượng Open)

Trong mục *Look in :* Chọn thư mục cần chứa tên tập tin cần mở.

File name: Chọn tên tập tin cần mở.

Chọn Open

3.3. Đóng một CSDL

Chọn File/Close hoặc ALT+F4

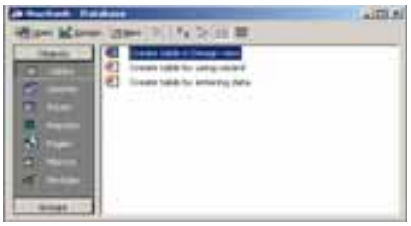
3.4. Các thành phần cơ bản của một tập tin CSDL ACCESS

Một tập tin CSDL ACCESS gồm có 6 thành phần cơ bản sau

Bảng (Tables) : Là nơi chứa dữ liệu

Truy vấn (Queries) : Truy vấn thông tin dựa trên một hoặc nhiều bảng.

Biểu mẫu (Forms) : Các biểu mẫu dùng để nhập dữ liệu hoặc hiển thị dữ liệu.



Báo cáo (Reports) : Dùng để in ấn.

Pages (Trang) : Tạo trang dữ liệu.

Macros (Tập lệnh) : Thực hiện các tập lệnh.

Modules (Đơn thể) : Dùng để lập trình Access Basic

4. CÁCH SỬ DỤNG CỬA SỔ DATABASE

Như đã nói ở trên, một CSDL của Access chứa trong nó 7 đối tượng chứ không đơn thuần là bảng dữ liệu. Sau khi tạo mới một CSDL hoặc mở một CSDL có sẵn Access sẽ hiển thị một cửa sổ Database, trên đó hiển thị tên của CSDL đang mở và liệt kê 7 đối tượng mà nó quản lý, mỗi lớp đối tượng đều được phân lớp rõ ràng để tiện theo dõi.

4.1. Tạo một đối tượng mới

Trong cửa sổ Database, chọn tab chứa đối tượng cần tạo (Bảng, Truy vấn, Biểu mẫu, Báo cáo,...) hoặc thực hiện lệnh

View/D atabase Object - Table/Query/Form/ReportPages/Macros/Modules

Chọn nút New.

4.2. Thực hiện một đối tượng trong CSDL

Trong cửa sổ Database, chọn tab cần thực hiện. Cửa sổ Database liệt kê tên các đối tượng có sẵn, chọn tên đối tượng cần mở.

Chọn nút Open (đối với Bảng, Truy vấn, Biểu mẫu, Trang) hoặc Preview (đối với Báo biểu) hoặc Run (đối với Macro và Module).

4.3. Sửa đổi một đối tượng có sẵn trong CSDL

Trong cửa sổ Database, chọn tab cần thực hiện. Cửa sổ Database liệt kê tên các đối tượng có sẵn, chọn tên đối tượng cần mở, Chọn nút Design.

Chương 2 BẢNG DỮ LIỆU

Bảng là đối tượng chủ yếu chứa các thông tin cần quản lý, có thể đó chỉ là một vài địa chỉ đơn giản hay cả vài chục nghìn bản ghi chứa đựng thông tin liên quan đến các hoạt động SXKD của một công ty xuất nhập khẩu nào đó. Trước khi ta muốn làm việc với bất kỳ một CSDL nào thì ta phải có thông tin để quản lý, các thông tin đó nằm trong các bảng, nó là cơ sở để cho người sử dụng tạo các đối tượng khác trong CSDL như truy vấn, biểu mẫu, báo biểu...

1. THIẾT KẾ CƠ SỞ DỮ LIỆU

Một CSDL được thiết kế tốt cho phép người sử dụng truy cập nhanh chóng đến những thông tin cần tham khảo, giúp tiết kiệm được thời gian truy xuất thông tin. Một CSDL thiết kế tốt giúp người sử dụng rút ra được những kết quả nhanh chóng và chính xác hơn.

Để thiết kế một CSDL tốt chúng ta phải hiểu cách mà một Hệ QTCSDL quản trị các CSDL như thế nào. MS Access hay bất kỳ một Hệ QTCSDL nào có thể cung cấp các thông tin cho chúng ta một cách chính xác và hiệu quả nếu chúng được cung cấp đầy đủ mọi dữ kiện về nhiều đối tượng khác nhau lưu trữ trong các bảng dữ liệu. Ví dụ ta cần một bảng để chứa thông tin về lý lịch của cán bộ, một bảng khác để chứa các đề tài nguyên cứu khoa học của các cán bộ...

Khi bắt tay thiết kế CSDL, chúng ta phải xác định và phân tích các thông tin muốn lưu trữ thành các đối tượng riêng rẽ, sau đó báo cho Hệ QTCSDL biết các đối tượng đó liên quan với nhau như thế nào. Dựa vào các quan hệ đó mà Hệ QTCSDL có thể liên kết các đối tượng và rút ra các số liệu tổng hợp cần thiết.

CÁC BƯỚC THIẾT KẾ CSDL

Bước 1: Xác định mục tiêu khai thác CSDL của chúng ta. Điều này quyết định các loại sự kiện chúng ta sẽ đưa vào MS Access.

Bước 2: Xác định các bảng dữ liệu cần thiết. Mỗi đối tượng thông tin sẽ hình thành một bảng trong CSDL của chúng ta.

Bước 3: Sau khi đã xác định xong các bảng cần thiết, tiếp đến ta phải chỉ rõ thông tin nào cần quản lý trong mỗi bảng, đó là xác định các trường. Mỗi loại thông tin trong bảng gọi là trường. Mọi mẫu in trong cùng một bảng đều có chung cấu trúc các trường. Ví dụ: Trong lý lịch khoa học cán bộ, những trường (thông tin) cần quản lý là: “HỌ VÀ TÊN”, “CHUYÊN MÔN”, “HỌC VỊ”, “HỌC HÀM”,...

Bước 4: Xác định các mối quan hệ giữa các bảng. Nhìn vào mỗi bảng dữ liệu và xem xét dữ liệu trong bảng này liên hệ thế nào với dữ liệu trong bảng khác. Thêm trướng hoặc tạo bảng mới để làm rõ mối quan hệ này. Đây là vấn đề hết sức quan trọng, tạo được quan hệ tốt sẽ giúp chúng ta nhanh chóng truy tìm tìm và kết xuất dữ liệu.

Bước 5: Tinh chế, hiệu chỉnh lại thiết kế. Phân tích lại thiết kế ban đầu để tim lỗi, tạo bảng dữ liệu và nhập vào vài bản ghi, thử xem CSDL đó phản ánh thế nào với những yêu cầu truy xuất của chúng ta, có rút được kết quả đúng từ những bảng dữ liệu đó không. Thực hiện các chỉnh sửa thiết kế nếu thấy cần thiết.

2. KHÁI NIỆM VỀ BẢNG

Bảng là nơi chứa dữ liệu về một đối tượng thông tin nào đó như SINH VIÊN, HÓA ĐƠN,... Mỗi hàng trong bảng gọi là một **bản ghi** (record) chứa các nội dung riêng của đối tượng đó. Mỗi bản ghi của một bảng đều có chung cấu trúc, tức là các **trường** (field). Ví dụ: Cho bảng dưới đây để quản lý lý lịch khoa học cán bộ trong trường đại học, có các trường MACB (Mã cán bộ), TRINHDOVH (Trình độ văn hóa), CHUYENMON (Chuyên môn),...

Trong một CSDL có thể chứa nhiều bảng, thường mỗi bảng lưu trữ nhiều thông tin (dữ liệu) về một đối tượng thông tin nào đó, mỗi một thông tin đều có những kiểu đặc trưng riêng, mà với Access nó sẽ cụ thể thành những kiểu dữ liệu của các trường.

3. TẠO BẢNG MỚI TRONG CƠ SỞ DỮ LIỆU

Trong MS Access có hai cách để tạo bảng, một là cách dùng Table Wizard, nhưng các trường ở đây MS Access tự động đặt tên và không có bàn tay can thiệp của người sử dụng. Ở đây, sẽ đưa ra cách tạo mới bảng hoàn toàn do người sử dụng.

3.1. Tạo bảng không dùng Table Wizard

Trong cửa sổ Database, chọn tab Table (hoặc Lệnh *View/Daatbase object - Table*)

Chọn nút **New,** xuất hiện hộp thoại

Datasheet View: Trên màn hình sẽ xuất hiện một bảng trống với các trường (tiêu đề cột) lần lượt Field1, field2

Design View: Trên màn hình xuất hiện cửa sổ thiết kế bảng, người sử dụng tự thiết kế bảng.

Table Wizard: Thiết kế bảng với sự trợ giúp của MS Access

Import table: Nhập các bảng và các đối tượng từ các tập tin khác vào CSDL hiện thời.

Link table: Tạo bảng bằng cách nối vào CSDL hiện thời các bảng của CSDL khác.

Chọn chức năng Design View, chọn OK.

3.2. Sử dụng Design View

Field Name: Tên trường cần đặt (thông tin cần quản lý)

Data Type: Kiểu dữ liệu của trường

Desciption: Mô tả trường, phần này chỉ mang ý nghĩa làm rõ thông tin quản lý, có thể bỏ qua trong khi thiết kế bảng.

Field properties: Các thuộc tính của trường

Xác định khoá chính của bảng (nếu có)

Xác định thuộc tính của bảng, Lưu bảng dữ liệu

Đặt tên trường

Tên trường ở đây không nhất thiết phải có độ dài hạn chế và phải sát nhau, mà ta có thể đặt tên trường tùy ý nhưng không vượt quá 64 ký tự kể cả ký tự trắng. Lưu ý rằng, tên trường có thể đặt dài nên nó dẽ mô tả được thông tin quản lý, nhưng sẽ khó khăn hơn khi ta dùng các phát biểu SQL và lập trình Access Basic. Do đó khi đặt tên trường ta nên đặt ngắn gọn, dễ gợi nhớ và không chứa ký tự trắng.

Kiểu dữ liệu

MS Access cung cấp một số kiểu dữ liệu cơ bản sau:

Kiểu dữ liệu	Dữ liệu vào	Kích thước
Text	Văn bản	Tối đa 255 byte
Memo	Văn bản nhiều dòng, trang	Tối đa 64000 bytes
Number	Số	1,2,4 hoặc 8 byte
Date/Time	Ngày giờ	8 byte
Currency	Tiền tệ (Số)	8 byte
Auto number	ACCESS tự động tăng lên một khi một bản ghi được tạo	4 byte
Yes/No	Lý luận (Boolean)	1 bit
OLE Object	Đối tượng của phần mềm khác	Tối đa 1 giga byte
Lookup Wizard		Trường nhận giá trị do người dùng chọn từ 1 bảng khác hoặc 1 danh sách giá trị định trước
Hyper link	Liên kết các URL	

Quy định thuộc tính, định dạng cho trường

Đặt thuộc tính là một phần không kém quan trọng, nó quyết định đến dữ liệu thực sự lưu giữa trong bảng, kiểm tra độ chính xác dữ liệu khi nhập vào, định dạng

dữ liệu nhập vào ... Mỗi một kiểu dữ liệu sẽ có các thuộc tính và các đặc trưng và khác nhau. Sau đây là các thuộc tính, định dạng của các kiểu dữ liệu.

Để tăng thêm tốc độ xử lý khi nhập dữ liệu cũng như các công việc tìm kiếm sau này thì việc quy định dữ liệu rất quan trọng.

Các trường trong ACCESS có các thuộc tính sau:

3.2.1. Field Size

Quy định kích thước của trường và tùy thuộc vào từng kiểu dữ liệu

Kiểu Text: Chúng ta quy định độ dài tối đa của chuỗi.

Kiểu Number: Có thể chọn một trong các loại sau:

Byte: 0..255

Integer: -32768..32767

Long Integer: -3147483648.. 3147483647

Single:$-3,4x10^{38}..3,4x10^{38}$ (Tối đa 7 số lẻ)

Double: $-1.797x10^{308}$ ..$1.797x10^{308}$ (Tối đa 15 số lẻ)

Decimal Places

Quy định số chữ số thập phân (Chỉ sử dụng trong kiểu Single và Double)

Đối với kiểu Currency mặc định decimal places là 2

3.2.2. Format

Quy định dạng hiển thị dữ liệu, tùy thuộc vào từng kiểu dữ liệu.

Kiểu chuỗi: Gồm 3 phần

<Phần 1>;<Phần 2>;<Phần 3>

Trong đó:

<Phần 1>: Chuỗi định dạng tương ứng trong trường hợp có chứa văn bản.

<Phần 2>: Chuỗi định dạng tương ứng trong trường hợp không chứa văn bản.

<Phần 3>: Chuỗi định dạng tương ứng trong trường hợp null

Các ký tự dùng để định dạng chuỗi

Ký tự	Tác dëch vuû du lëchụng
@	Chuỗi ký tự
>	Đổi tất cả ký tự nhập vào thành in hoa
<	Đổi tất cả ký tự nhập vào thành in thường
“Chuỗi ký tự “	Chuỗi ký tự giữa 2 dấu nháy
\<ký tự>	Ký tự nằm sau dấu \
[black] [White] [red] Hoặc [<số>] Trong đó 0<=số<=56	Màu

Ví dụ

Cách định dạng	Dữ liệu	Hiển thị
@@@-@@@@	123456 abcdef	123-456 abc-def
>	Tinhoc	TINHOC
<	TINHOC	Tinhoc
@;”Không có”;”Không biết”	Chuỗi bất kỳ Chuỗi rỗng Giá trị trống (Null)	Hiển thị chuỗi Không có Không biết

Kiểu Number

Định dạng do ACCESS cung cấp

Dạng	Dữ liệu	Hiển thị
General Number	1234.5	1234.5
Currency	1234.5	$1.234.50
Fixed	1234.5	1234

Standard	1234.5	1,234.50
Pecent	0.825	82.50%
Scientific	1234.5	1.23E+03

Định dạng do người sử dụng

<Phần 1>;<Phần 2>;<Phần 3>;<Phần 4>

<Phần 1>:Chuỗi định dạng tương ứng trong trường hợp số dương.

<Phần 2>: Chuỗi định dạng tương ứng trong trường hợp số âm.

<Phần 3>: Chuỗi định dạng tương ứng trong trường hợp số bằng zero.

<Phần 4>: Chuỗi định dạng tương ứng trong trường hợp null.

Các ký tự định dạng

Ký tự	Tác dụng
.(Period)	Dấu chấm thập phân
,(comma)	Dấu phân cách ngàn
0	Ký tự số (0-9)
#	Ký tự số hoặc khoảng trắng
$	Dấu $
%	Phần trăm

Ví dụ

Định dạng	Hiển thị
0;(0);;”Null”	Số dương hiển thị bình thường Số âm được bao giữa 2 dấu ngoặc Số zero bị bỏ trống Null hiện chữ Null
+0.0;-0.0;0.0	Hiển thị dấu + phía trước nếu số dương Hiển thị dấu - phía trước nếu số âm Hiển thị 0.0 nếu âm hoặc Null

Kiểu Date/Time

Các kiểu định dạng do ACCESS cung cấp

Dạng	Hiển thị
General date	10/30/99 5:10:30PM
Long date	Friday, may 30 , 1999
Medium date	30-jul-1999
Short date	01/08/99
Long time	6:20:00 PM
Medium time	6:20 PM
Short time	18:20

Các ký tự định dạng

Ký tự	Tác dụng
: (colon)	Dấu phân cách giờ
/	Dấu phân cách ngày
d	Ngày trong tháng (1-31)
dd	Ngày trong tháng 01-31)
ddd	Ngày trong tuần (Sun -Sat0
W	Ngày trong tuần (1-7)
WW	Tuần trong năm (1-54)
M	Tháng trong năm (1-12)
MM	Tháng trong năm (01-12)
q	Quý trong năm (1-4)
y	Ngày trong năm (1-366)
yy	Năm (01-99)
h	Giờ (0-23)
n	Phút (0-59)
s	Giây (0-59)

Ví dụ

Định dạng	Hiển thị
Ddd,”mmm d”,yyyy	Mon,jun 2, 1998
Mm/dd/yyyy	01/02/1998

Kiểu Yes/No

Các kiểu định dạng

Định dạng	Tác dụng
Yes/No	Đúng/Sai
True/False	Đúng/Sai
On/Off	Đúng/Sai

Định dạng do người sử dụng: Gồm 3 phần

<Phần 1>;<Phần 2>;<Phần 3>

Trong đó: <Phần 1>: Bỏ trống

<Phần 2>: Trường hợp giá trị trường đúng

<Phần 3>: Trường hợp giá trị trường sai

Ví dụ

Định dạng	Hiển thị	
	Trường hợp True	Trường hợp False
;”Nam”;”Nu”	Nam	Nu
;”co”;”Khong”	Co	Khong

3.2.3. Input mask (Mặt nạ)

Thuộc tính này dùng để quy định mặt nạ nhập dữ liệu cho một trường.

Các ký tự định dạng trong input mask

Ký tự	Tác dụng
0	Bắt buộc nhập ký tự số
9	Không bắt buộc nhập, ký tự số
#	Không bắt buộc nhập, số 0-9, khoảng trắng, dấu + và -
L	Bắt buộc nhập, ký tự chữ
?	Không bắt buộc nhập, ký tự chữ hoặc khoảng trắng
a	Bắt buộc nhập, ký tự chữ hoặc số
A	Không bắt buộc nhập, ký tự chữ hoặc số
&	Bắt buộc nhập, ký tự bất kỳ
C	Không bắt buộc nhập ký tự bất kỳ
<	Các ký tự bên phải được đổi thành chữ thường
>	Các ký tự bên phải được đổi thành chữ hoa
!	Dữ liệu được ghi từ phải sang trái
\<Ký tự>	Ký tự theo sau \ sẽ được đưa thẳng vào

Ví dụ

Input mask	Dữ liệu nhập vào
(000)000-0000	(054)828-8282
(000)AAA-A	(123)124-E

☞ **Chú ý:** Nếu muốn các ký tự gõ vào quy định thuộc tính input mask là password (Khi nhập dữ liệu vào tại các vị trí đó xuất hiện dấu *).

3.2.4. Caption

Quy định nhãn là một chuỗi ký tự sẽ xuất hiện tại dòng tiêu đề của bảng. Chuỗi ký tự này cũng xuất hiện tại nhãn các của các điều khiển trong các biểu mẫu hoặc báo cáo.

3.2.5. Default value

Quy định giá trị mặc định cho trường trừ Auto number và OEL Object

3.2.6. Validation rule và Validation Text

Quy định quy tắc hợp lệ dữ liệu (Validation rule) để giới hạn giá trị nhập vào cho một trường. Khi giới hạn này bị vi phạm sẽ có câu thông báo ở Validation text.

Các phép toán có thểt dùng trong Validation rule

Các phép toán	Phép toán	Tác dụng
Phép so sánh	>, <, >=, <=, =, <>	
Phép toán logic	Or, and , not	Hoặc, và, phủ định
Phép toán về chuỗi	Like	Giống như

☞ ***Chú ý:*** Nếu hằng trong biểu thức là kiểu ngày thì nên đặt giữa 2 dấu #.

Ví dụ

Validation rule	Tác dụng
<>0	Khác số không
Like “*HUE*”	Trong chuỗi phải chứa HUE
<#25/07/76#	Trước ngày 25/07/76
>=#10/10/77# and <=#12/11/77#	Trong khoảng từ 10/10/77 đến 12/12/77

3.2.7. Required

Có thể quy định thuộc tính này để bặt buộc hay không bắt buộc nhập dữ liệu cho trường.

Required	Tác dụng
Yes	Bắt buộc nhập dữ liệu
No	Không bắt buộc nhập dữ liệu

3.2.8. AllowZeroLength

Thuộc tính này cho phép quy định một trường có kiểu Text hay memo có thể hoặc không có thể có chuỗi có độ dài bằng 0.

☞ ***Chú ý:*** Cần phân biệt một trường chứa giá trị null (chưa có dữ liệu) và một trường chứa chuỗi có độ dài bằng 0 (Có dữ liệu nhưng chuỗi rỗng “”).

AllowZeroLength	Tác dụng
Yes	Chấp nhận chuỗi rỗng
No	Không chấp nhận chuỗi rỗng

3.2.9. Index

Quy định thuộc tính này để tạo chỉ mục trên một trường. Nếu chúng ta lập chỉ mục thì việc tìm kiếm dữ liệu nhanh hơn và tiện hơn.

Index	Tác dụng
Yes(Dupplicate OK)	Tạo chỉ mục có trùng lặp
Yes(No Dupplicate)	Tạo chỉ mục không trùng lặp
No	Không tạo chỉ mục

3.2.10. New value

Thuộc tính này chỉ đối với dữ liệu kiểu auto number, quy định cách thức mà trường tự động điền số khi thêm bản ghi mới vào.

New value	Tác dụng
Increase	Tăng dần
Random	Lấy số ngẫu nhiên

4. THIẾT LẬP KHOÁ CHÍNH (*primary key*)

4.1. Khái niệm khoá chính

Sức mạnh của một Hệ QTCSDL như Microsoft Access, là khả năng mau chóng truy tìm và rút dữ liệu từ nhiều bảng khác nhau trong CSDL. Để hệ thống có thể làm được điều này một cách hiệu quả, mỗi bảng trong CSDL cần có một trường hoặc một nhóm các trường có thể xác định duy nhất một bản ghi trong số rất nhiều bản ghi đang có trong bảng. Đây thường là một mã nhận diện như Mã nhân viên hay Số Báo Danh của học sinh. Theo thuật ngữ CSDL trường này được gọi là **khóa chính** (primary key) của bảng. MS Access dùng trường khóa chính để kết nối dữ liệu nhanh chóng từ nhiều bảng và xuất ra kết quả yêu cầu.

Nếu trong bảng chúng ta đã có một trường sao cho ứng với mỗi trị thuộc trường đó chúng ta xác định duy nhất một bản ghi của bảng, chúng ta có thể dùng trường đó làm trường khóa của bảng. Từ đó cho ta thấy rằng tất cả các trị trong trường khóa chính phải khác nhau. Chẳng hạn đừng dùng tên người làm trường khóa vì tên trường là không duy nhất.

Nếu không tìm được mã nhận diện cho bảng nào đó, chúng ta có thể dùng một trường kiểu **Autonunter** (ví dụ Số Thứ Tự) để làm trường khóa chính.

Khi chọn trường làm khóa chính chúng ta lưu ý mấy điểm sau:

MS Access không chấp nhận các giá trị trùng nhau hay trống (null) trong trường khóa chính.

Chúng ta sẽ dùng các giá trị trong trường khóa chính để truy xuất các bản ghi trong CSDL, do đó các giá trị trong trường này không nên quá dài vì khó nhớ và khó gõ vào.

Kích thước của khóa chính ảnh hưởng đến tốc độ truy xuất CSDL. Để đạt hiệu quả tối ưu, dùng kích thước nhỏ nhất để xác định mọi giá trị cần đưa vào trường.

4.2. Cách đặt khoá chính

Ta có thể tự chọn trường làm khóa chính cho bảng bằng các bước sau đây:

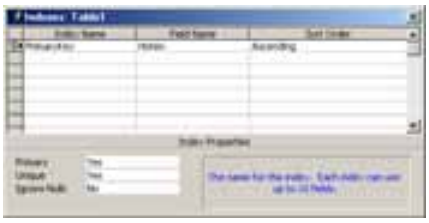
Mở bảng ở chế độ Design View

Nhắp chọn trường cần đặt

Thực hiện lệnh *Edit - Primary Key* hoặc nhắp chọn nút trên thanh công cụ của mục này .

☞***chú ý:*** Không phải mọi trường đều có thể làm khóa chính, mà chỉ có các trường có các kiểu dữ liệu không phải là **Memo** và **OLE Object**., **Hyper Link**.

Để hủy bỏ khóa chính hoặc các đã thiết lập thì thực hiện lệnh *View - Indexes*, trong hộp thoại này chọn và xóa đi những trường khóa đã thiết lập:



5. LƯU BẢNG

Sau khi thiết kế xong, ta tiến hành lưu bảng vào CSDL, có thể thực hiện một trong hai thao tác sau:

Thực hiện lệnh *File - Save*.

Nhắp chọn nút trên thanh công cụ của mục này (Table Design)

6. HIỆU CHỈNH BẢNG

6.1. Di chuyển trường: Các thao tác để di chuyển thứ tự các trường

Đưa con trỏ ra đầu trường đến khi con trỏ chuột chuyển thành ➡ thì nhắp chọn.

Đưa con trỏ ra đầu trường vừa chọn, nhấn và kéo đếnvị trí mới.

6.2. Chen trường : Các thao tác lần lượt như sau

Chọn trường hiện thời là trường sẽ nằm sau trường được chen vào

Thực hiện lệnh *Insert/ Row*

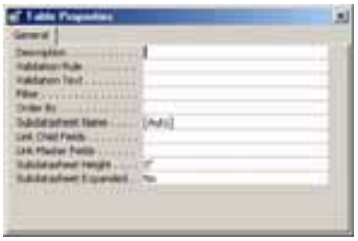
6.3. Xóa trường: Các thao tác lần lượt như sau

Chọn trường cần xóa

Thực hiện lệnh *Edit - Delete Rows*

6.4. Quy định thuộc tính của bảng

Mở bảng ở chế độ Design View, chọn View/Properties



Description: Dòng mô tả bảng

Validation Rule: Quy tắc hợp lệ dữ liệu cho toàn bảng.

Validation Text: Thông báo lỗi khi dữ liệu không hợp lệ.

7. XEM THÔNG TIN VÀ BỔ SUNG BẢN GHI

7.1. Xem thông tin ở chế độ datasheet

Muốn xem thông tin trong một bảng chúng ta phải chuyển bảng sang một chế độ hiển thị khác gọi là *Datasheet*. Trong chế độ hiển thị này, mỗi bản ghi được thể hiện trên một hàng ngang, hàng đầu tiên là các tên trường.

Sau đây là các cách để chuyển sang chế độ hiển thị Datasheet:

Trong của sổ Database của CSDL đang mở, nhấp chọn tab Table. Trong mục này chọn bảng cần hiển thị rồi chọn nút **Open**, bảng sẽ được mở để bổ sung và chỉnh sửa dữ liệu.

Ta có thể chuyển sang chế độ Datasheet ngay khi đang ở trong chế Design, bằng cách nhấp chọn nút thì bảng sẽ chuyển sang chế độ Datasheet, để quay trở về chế độ Design, ta nhấp chọn lại nút . Hoặc chọn lệnh *View - /Design View*.

7.2. Bổ sung bản ghi cho bảng

Sau khi hoàn thành công việc thiết kế cấu trúc bảng, ta tiến hành nhập dữ liệu, tức là bổ sung các bản ghi, cho bảng. Hiển thị bảng ở chế độ hiển thị Datasheet, mỗi hàng đại diện cho một bản ghi. Ở đây có các ký hiệu chúng ta cần biết công dụng của chúng

Bản ghi hiện thời

Bản ghi đang nhập dữ liệu

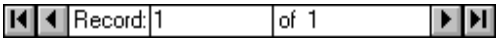
Bản ghi mới

7.2.1. Bổ sung bản ghi mới cho CSDL

Đang đứng tại một bản ghi nào đó (không phải là bản ghi mới), chọn nút trên thanh công cụ. Hoặc thực hiện lệnh *Record - Go To - New*.

7.2.2. Di chuyển giữa các bản ghi

Ta có thể di chuyển qua lại giữ các bản ghi bằng cách dùng công cụ *Chọn lựa bản ghi* (Record Selector).



7.2.3. Nhập dữ liệu cho bản ghi

Khi đang nhập dữ liệu cho một bản ghi nào đó, thì đầu hàng của bản ghi đó xuất hiện biểu tượng .

Tổ hợp phím	Tác dụng
Tab	Sang ô kế tiếp
Shift Tab	Sang ô phía trước
Home	Đến đầu dòng
End	Đến cuối dòng
Ctrl Home	Đến bản ghi đầu tiên
Ctrl End	Đến bản ghi cuối cùng
Shift F2	Zoom

Theo chuẩn, khi nhập dữ liệu thì Access sẽ lấy font mặc định là MS San Serif, điều này có thể giúp cho ta hiển thị được tiếng Việt chỉ khi MS San Serif đó là của VietWare hoặc của ABC.

Để không phụ thuộc vào điều này, ta nên chọn font trước khi tiến hành nhập dữ liệu. Trong chế độ hiển thị Datasheet, thực hiện lệnh *Format - Font...*

Khi nhập dữ liệu là trường cho trường **OLE Object**, ta thực hiện như sau: Lệnh *Edit - Object...*

7.2.4. Chọn các bản ghi

Đánh dấu chọn bản ghi:

Chọn Edit/Select Record: Để chọn bản ghi hiện hành

Chọn Edit/ Select all Record để chọn toàn bộ

7.2.5. Xóa bản ghi

Chọn các bản ghi cần xóa, sau đó thực hiện lệnh *Edit - Delete* (hoặc nhấn phím DELETE) .

8. THIẾT LẬP QUAN HỆ GIỮA CÁC BẢNG

8.1. Các loại quan hệ trong cơ sở dữ liệu ACCESS

8.1.1. Quan hệ một -một (1-1)

Trong quan hệ một -một, mỗi bản ghi trong bảng A có tương ứng với một bản ghi trong bảng B và ngược lại mỗi bản ghi trong bảng B có tương ứng duy nhất một bản ghi trong bảng A.

***Ví dụ*:** Cho 2 bảng dữ liệu

Bảng Danhsach(**Masv**, ten, Ngaysinh, gioitinh) và bảng Diemthi(**Masv**, diem)

Ten	Ngaysinh	Gioitinh	Masv
An	20/10/77	Yes	A001
Bình	21/07/80	No	A002
Thuỷ	02/12/77	Yes	A003
Lan	03/04/80	No	A004
Hồng	12/11/77	No	A005

Masv	diem
A001	9
A002	7
A003	9
A004	4
A005	5

Bảng **Danhsach** và **diemthi** có mối quan hệ **1-1** dựa trên trường Masv.

8.1.2. Quan hệ một nhiều (1-∞)

Là mối quan hệ phổ biến nhất trong CSDL, trong quan hệ một nhiều : Một bản ghi trong bảng A sẽ có thể có nhiều bản ghi tương ứng trong bảng B, nhưng ngược lại một bản ghi trong bảng B có duy nhất một bản ghi tương ứng trong bảng A.

Ví dụ: Trong một khoa của một trường học nào đó có nhiều sinh viên, những một sinh viên thuộc một khoa nhất định. Ta có 2 bảng dữ liệu như sau:

Bảng Danhsachkhoa(**Makhoa**, tenkhoa, sodthoai)

Bảng danhsachsv(Makhoa, Ten, Quequan, lop)

Tenkhoa	Sodthoai	Makhoa
CNTT	826767	01
TOÁN	878787	02
LÝ	868785	03

Makhoa	Ten	Queuqan	Lop
01	Thanh	Huế	K23
01	Tùng	Qbình	K24
02	Thuỷ	Huế	K25
02	Hùng	ĐN	K26
03	Lan	Huế	K25
03	Hương	ĐN	K26

Bảng Danhsachkhoa và bảng danhsachsv có mối quan hệ **1-∞** dựa trên trường Makhoa.

8.3. Quan hệ nhiều nhiều(∞-∞)

Trong quan hệ nhiều nhiều, mỗi bản ghi trong bảng A có thể có không hoặc nhiều bản ghi trong bảng B và ngược lại mỗi bản ghi trong bảng B có thể có không hoặc nhiều bản ghi trong bảng A.

Khi gặp mối quan hệ nhiều- nhiều để không gây nên sự trùng lặp và dư thừa dữ liệu thì người ta tách quan hệ nhiều-nhiều thành 2 quan hệ một-nhiều bằng cách tạo ra một bảng phụ chứa khóa chính của 2 bảng đó.

Ví dụ: Một giáo viên có thể dạy cho nhiều trường và một trường có nhiều giáo viên tham gia giảng dạy. Đây là một mối quan hệ nhiều-nhiều

Bảng Danhsachgv(**Magv**,ten)

Bảng Danhsachtruong(**Matruong**, Tentruong)

Tạo ra bảng Phancongday(**Magv, matruong**)

Ten	Magv
Thanh	G1
Thuý	G2
Hùng	G3

Magv	Matruong
G1	KH
G1	SP
G2	YK
G2	SP
G3	KH
G3	YK

Matruong	tentruong
KH	DHKH
SP	DHSP
YK	DHYK

Bảng **Danhsachgv** và bảng **Phancongday** có mối quan hệ **1-∞** dựa trên trường **Magv**.

Bảng **Danhsachtruong** và bảng **Phancongday** có mối quan hệ **1-∞** dựa trên trường **Matruong**.

8.4. Thiết lập mối quan hệ giữa các bảng dữ liệu (Relationships)

Tại cửa sổ Database, thực hiện lệnh Tools/Relationship

Trong cửa sổ Show

bảng cần thiết lập quan hệ, sau đó chọn **Add và Close**.

Kéo trường liên kết của bảng quan hệ vào trường của bảng được quan hệ (Table related).

Bật chức năng Enforce Referential Integrity (Nếu muốn quan hệ này bị ràng buộc tham chiếu toàn vẹn), chọn mối quan hệ (one-many) hoặc (one-one).

Chọn nút **Create**.

☞ ***Chú ý***

Quan hệ có tính tham chiếu toàn vẹn sẽ đảm bảo các vấn đề sau:

Khi nhập dữ liệu cho trường tham gia quan hệ ở bên nhiều thì phải tồn tại bên một.

Không thể xoá một bản ghi của bảng bên một nếu trong quan hệ đã tồn tại những bản ghi bên nhiều có quan hệ với bản ghi bên một đó.

Trường hợp vi phạm các quy tắc trên thì sẽ nhận được thông báo lỗi.

8.4.1. Thiết lập thuộc tính tham chiếu toàn vẹn trong quan hệ

Trong khi chọn mối quan hệ giữa các bảng, có 2 thuộc tính tham chiếu toàn vẹn đó là *Cascade update related fields,Cascade Delete related records*, có thiết lập 2 thuộc tính này.

Nếu chọn thuộc tính *Cascade update related fields*, khi dữ liệu trên khoá chính của bảng bên một thay đổi thì Access sẽ tự động cập nhật sự thay đổi đó vào các trường tương ứng (có quan hệ) trên các bảng bên Nhiều, hay nói cách khác, dữ liệu ở bảng bên nhiều cũng thay đổi theo.

Nếu chọn thuộc tính *Cascade Delete related records*, khi dữ liệu trên bảng bên một bị xoá thì dữ liệu trên bảng bên nhiều cũng sẽ bị xoá..

8.4.2. Kiểu kết nối (Join type)

Trong quá trình thiết lập quan hệ giữa các bảng, nếu không chọn nút Create, chọn nút join type để chọn kiểu liên kết

Mục 1: Liên kết nội (Inner join)

Mục 2 và mục 3 là liên kết ngoại (outer join)

8.4.3. Điều chỉnh các mối quan hệ

Mở cửa sổ quan hệ (Tools/Relationship)

Click chuột phải, chọn edit relationship

8.4.4. Xoá các mối quan hệ

Mở cửa sổ quan hệ (Tools/Relationship)

Chọn mối quan hệ giữa các bảng, nhấn delete.

9. SẮP XẾP VÀ LỌC DỮ LIỆU

9.1. Một số phép toán và hàm

9.1.1. Một số phép toán

Ký tự thay thế: ? : Thay thế cho một ký tự bất kỳ

Ký tự * : Thay thế cho một dãy các ký tự.

Phép toán **Like**: Giống như

In: Kiểm tra một giá trị có thuộc một tập các giá trị hay không?

Is Null: Giá trị của một trường là Null.

Is not Null:Giá trị của một trường là không Null.

Between.....and: Kiểm tra xem một giá trị có thuộc một "đoạn" nào đó hay không?

9.1.2. Một số hàm

Hàm **Left$(<Chuỗi>,<n>)**: Trích bên trái chuỗi n ký tự.

Hàm **Right$(<Chuỗi>,<n>)**: Trích bên phải chuỗi n ký tự.

Hàm **Ucase(<Chuỗi>)**: Trả lại một chuỗi in hoa.

Hàm **Lcase(<Chuỗi>)**: Trả lại một chuỗi in thường.

Hàm **IIF(<Điều kiện>,<Giá trị 1>,< Giá trị 2>)**: Nếu <Điều kiện> nhận giá trị true thì hàm trả lại <Giá trị 1>, ngược lại hàm trả lại <Giá trị 2>.

9.2. Sắp xếp dữ liệu

9.2.1. sắp xếp trên một trường

Đặt con trỏ tại trường cần sắp xếp

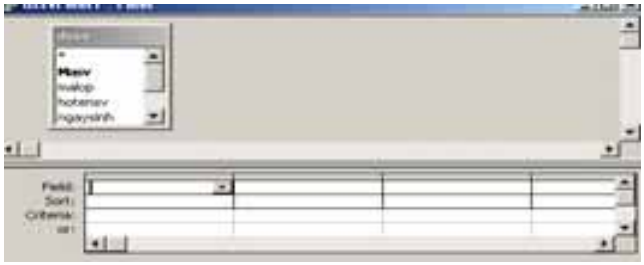
Thực hiện lệnh Records/ Sort/ Sort Ascending (Nếu sắp xếp tăng dần)

/ Sort Descending (Nếu sắp xếp giảm dần)

9.2.2. sắp xếp trên nhiều trường

Mở bảng trong chế độ Datasheet rồi thực hiện lệnh

Records/ Filter/ Advanced Filter sort



Trong hàng Field: Chọn các trường cần sắp xếp (Thứ tự ưu tiên từ trái sang phải)

Trong hàng Sort: Chọn tiêu chuẩn sắp xếp.

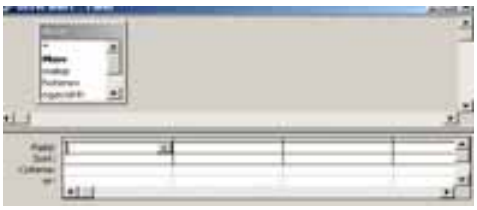
Trong hàng Criteria: Chọn điều kiện sắp xếp (nếu có).

Xem kết quả.: Chọn Filter/Apply filter.

9.3. Lọc dữ liệu

Mở bảng trong chế độ Datasheet rồi thực hiện lệnh

Records/ Filter/ Advanced Filter sort



Trong hàng Field: Chọn các trường làm tiêu chuẩn lọc dữ liệu

Trong hàng Criterie: Chọn tiêu chuẩn lọc dữ liệu.

Thực hiện lệnh Filter/ Apply filter sort để xem kết quả..

Chương 3 TRUY VẤN (QUERY)

Sức mạnh thực sự của CSDL là khả năng tìm đúng và đầy đủ thông tin mà chúng ta cần biết, trình bày dữ liệu sắp xếp theo ý muốn. Để đáp ứng yêu cầu trên, Acces cung cấp một công cụ truy vấn cho phép đặt câu hỏi với dữ liệu đang chứa bên trong các bảng trong CSDL.

1. KHÁI NIỆM TRUY VẤN

Truy vấn là một công cụ cho phép đặt câu hỏi với dữ liệu trong bảng dữ liệu trong CSDL.

Loại truy vấn thông dụng nhất là truy vấn chọn (Select Query). Với kiểu truy vấn này chúng ta có thể xem xét dữ liệu trong các bảng, thực hiện phân tích và chỉnh sửa trên dữ liệu đó, có thể xem thông tin từ 1 bảng hoặc có thể thêm nhiều trường từ nhiều bảng khác nhau.

Ví dụ:

Cho 2 bảng dữ liệu KHOHANG (**MAHANG**, TENHANG, GIA) BANHANG(MAHANG, TENKHACH, SOLUONG, NGAYMUA). Hãy hiển thị những khách hàng mua hàng trong tháng 7 bao gồm các thông tin: MAHANG, TENHANG, GIA, TENKHACH.

Sau khi thực hiện truy vấn, dữ liệu thỏa mãn yêu cầu được rút ra và tập hợp vào một bảng kết qủa gọi là **Dynaset** (Dynamic set). Dynaset cũng hoạt động như 1 bảng (Table) nhưng nó không phải là bảng vfa kết quả khi hiển thị có thể cho phép sửa đổi.

Một loại bảng thể hiện kết quả truy vấn khác là **Snapshot**, nó tương tự như dynaset tuy nhiên không thể sửa đổi thông tin (Như truy vấn Crosstab....).

1.1. Các loại truy vấn trong Access

Select Query : Truy vấn chọn

Crosstab Query : Truy vấn tham khỏa chéo (Thể hiện dòng và cột)

Action Query : Truy vấn hành động gồm

Truy vấn tạo bảng (make table Query)

Truy vấn nối (append Query)

Truy vấn cập nhật (Update Query)

Truy vấn xóa dữ liệu (Delete Query)

SQL Query : Truy vấn được viết bởi ngôn ngữ SQL.

Pass throught Query : Gởi các lệnh đến một CSDL SQL như Microsoft SQL server.

1.2. Sự cần thiết của truy vấn

Khi đứng trước một vấn đề nào đó trong CSDL, nếu sử dụng công cụ truy vấn thì có thể thực hiện được các yêu cầu sau:

Sự lựa chọn các trường cần thiết.

Lựa chọn những bản ghi.

Sắp xếp thứ tự các bản ghi.

Lấy dữ liệu chứa trên nhiều bảng khác nhau trong CSDL.

Thực hiện các phép tính.

Sử dụng truy vấn làm nguồn dữ liệu cho một biểu mẫu (Form), báo cáo (report) hoặc một truy vấn khác (Query).

Thay đổi dữ liệu trong bảng.

2. CÁC CHẾ ĐỘ HIỂN THỊ TRUY VẤN

2.1. Cửa sổ thiết kế truy vấn (Design view).

Trong chế độ này, người sử dụng có thể tạo, sửa chữa một truy vấn nào đó. Màn hình truy vấn chứa hai phần, phần thứ nhất chứa các bảng (hoặc truy vấn) tham gia truy vấn, phần thứ hai gọi là vùng lưới QBE (Query By Example).

2.2. Cửa sổ hiển thị truy vấn (DataSheet view).

Sử dụng chế độ này để xem kết quả.

2.3. Cửa sổ lệnh SQL (SQL view).

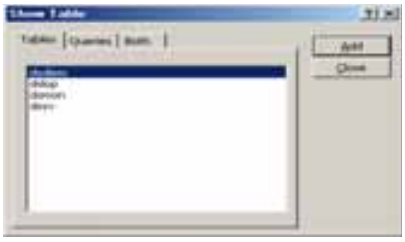
Sử dụng chế độ này để xem mã lệnh của truy vấn đang tạo

3. TẠO TRUY VẤN

3.1.Tạo mới 1 truy vấn

Từ cửa sổ Database, click vào đối tượng Queries.

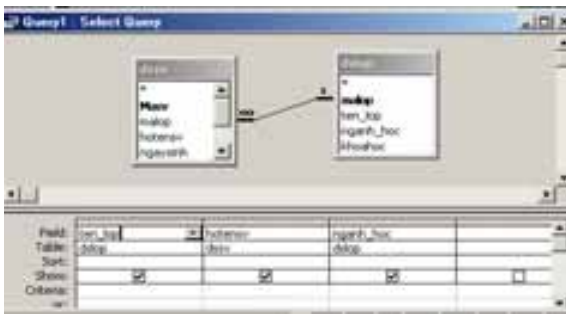
Chọn nút New.



Chọn Design View, chọn OK

Trong bảng Show Table chọn tables để hiển thị các bảng, chọn các bảng tham gia vào truy vấn và nhấn nút **Add**, sau đó nhấn ***Close***. (Nếu chọn Queries thì hiển thị truy vấn, chọn both thì hiển thị cả truy vấn và bảng dữ liệu).

Đưa các trường từ các bảng vào tham gia truy vần bằng cách kéo các trường và thả vào hàng **Field** trong vùng lưới QBE.



Trong hàng Sort: Sắp xếp dữ liệu (nếu có)

Trong hàng **Criteria** đặt tiêu chuẩn (nếu có)

Lưu truy vấn.

☞ ***Chú ý***

Mỗi truy vấn có:

Tối đa là 32 bảng tham gia.

Tối đa là 255 trường.

Kích thước tối đa của bảng dữ liệu (do truy vấn tạo ra) là 1 gigabyte.

Số trường dùng làm khóa sắp xếp tối đa là 10.

Số truy vấn lồng nhau tối đa là 50 cấp.

Số ký tự tối đa trong ô của vùng lưới là 1024.

Số ký tự tối đa trong dòng lệnh SQL là 64000.

Số ký tự tối đa trong tham số là 255.

3.2. Thay đổi thứ tự, xóa các trường

Các trường trong truy vấn sẽ hiển thị theo thứ tự như xuất hiện trong vùng lưới QBE.

3.2.1.Thay đổi thứ tự của trường

Đưa con trỏ vào thanh chọn sao cho con trỏ biến thành hình mũi tên trỏ xuống

Click để chọn trường

Drag để thay đổi vị trí.

3.2.2.xoá trường

Đưa con trỏ vào thanh chọn sao cho con trỏ biến thành hình mũi tên trỏ xuống

Click để chọn trường

Nhấn phím delete (Nếu muốn xoá tất cả các trường trong vùng lưới QBE: chọn Edit/clear grid)

3.3. Thể hiện hoặc che dấu tên bảng trong vùng lưới QBE

Muốn biết tên trường hiện tại trong vùng lưới QBE là của tên bảng nào, tại chế độ Design View người sử dụng thực hiện View/tables name.

3.4. Xem kết quả của truy vấn.

Tại cửa sổ Database chọn tên truy vấn rồi chọn Open, hoặc trong khi thiết kế truy vấn thực hiện lệnh View/datasheet View.

3.5. Đổi tiêu đề cột trong truy vấn.

Đổi tên tiêu đề cột trong truy vấn mục đích là làm cho bảng kết xuất dễ đọc hơn (Trừ khi đã quy định thuộc tính Caption).

Muốn đổi tên tiêu đề cột thực hiện theo các bước sau:

Mở truy vấn ở chế độ Design View

Chọn vào bên trái ký tự đầu tiên của tên trường trong vùng lưới QBE

Gõ vào tên mới, theo sau là dấu 2 chấm (:).

Ví dụ:

3.6. Định thứ tự sắp xếp

Có thể sử dụng nhanh trong chế độ datasheet View .

Có thể tạo sắp xếp trong khi thiết kế truy vấn bằng cách chọn Ascending (tăng dần) hoặc Descending (giám dần) trong hàng Sort của vùng lưới QBE .

☞ ***Chú ý:*** Nếu có nhiều trường định vị sắp xếp thì theo thứ tự ưu tiên từ trái sang phải.

3.7. Che dấu hay thể hiện các trường trong Dynaset

Tại hàng Show ứng với trường cần che dấu chúng ta không chọn mặt dù nó vẫn tồn tại, vẫn tham gia truy vấn.

3.8. Mối quan hệ giữa thuộc tính của trường trong truy vấn và trong bảng dữ liệu

Theo mặc nhiên, các trường trong truy vấn kế thừa tất cả các thuộc tính của trường trong bảng làm nguồn dữ liệu. Nếu không quy định lại trong truy vấn, các trường trong Dynaset hoặc snapshot luôn kế thừa các thuộc tính của bảng làm nguồn dữ liệu. Nếu thay đổi thiết kế trong bảng làm nguồn dữ liệu và thay đổi thuộc tính của các trường thì thuộc tính này cũng được thay đổi trong truy vấn. Tuy nhiên, nếu quy định lại các thuộc tính cho các trường trong truy vấn thì các thuộc tính của các trường trong bảng làm nguồn dữ liệu không thay đổi.

4. THIẾT KẾ TRUY VẤN CHỌN

4.1. Định nghĩa truy vấn chọn

Truy vấn chọn là loại truy vấn được chọn lựa, rút trích dữ liệu từ các bảng dữ liệu thỏa mãn một hoặc nhiều điều kiện nào đó. Khi thực hiện truy vấn chọn, Access tác động lên dữ liệu và thể hiện các bản ghi thoả mãn các điều kiện đăt ra trong một bảng kết quả gọi là Recordset.

4.2. Lập phép chọn trong truy vấn

4.2.1. Chọn một nhóm các bản ghi thỏa mãn một điều kiện nào đó

Muốn thực hiện các phép chọn trong khi thể hiện truy vấn người ta thường sử dụng các phép toán sau:

Phép toán	Ví dụ	Ý nghĩa
<	<#20/10/99#	Trước ngày 20/10/99
>	>#10/10/98#	Sau ngày 10/10/98
>=	>= #05/05/90#	Sau và trong ngày 05/05/90
<>	<>#01/01/99#	Khác ngày 01/01/99
=	= #10/10/97#	Trong ngày 10/10/97
Between .... and ...	Between #1/2/97# and #1/7/97#	Từ ngày 1/2/97 đến 1/7/97

Ví dụ:

Cho 2 bảng dữ liệu Dslop(**Malop**, Tenlop, Nganh_hoc, khoahoc)

Dssv(Masv, malop, hotensv, ngaysinh, quequan, giotinh, hocbong)

Tạo một truy vấn để hiển thị danh sách những sinh viên có ngaysinh trong khoảng thời gian từ 05/05/75 đến 05/05/79 bao gồm các trường: Tenlop, Hotensv, Ngaysinh, nganh_hoc.

Tạo truy vấn chọn và đưa 2 bảng dslop và dssv vào tham gia truy vấn

Đưa các trường Tenlop, hotensv, ngaysinh, nganh_hoc vào vùng lưới QBE

Trong hàng Criteria của trường Ngaysinh: Between #05/05/75# and #05/05/79#

4.2.2. Ký tự thay thế

Ký tự * : Thay thế một nhóm ký tự bất kỳ.

Ký tự ? : Thay thế 1 ký tự.

Ký tự [ ] : Thay thế các ký tự trong ngoặc vuông.

Ký tự ! : Phủ định.

Ký tự - : Từ ký tự đến ký tự.

Ví dụ

Cho 2 bảng dữ liệu Dslop(**Malop**, Tenlop, Nganh_hoc, khoahoc)

Dssv(Masv, malop, hotensv, ngaysinh, quequan, giotinh, hocbong)

Tạo một truy vấn để hiển thị danh sách những sinh viên có Tenlop bắt đầu là **"T"** bao gồm các trường: Tenlop, Hotensv, Ngaysinh, nganh_hoc.

Tạo truy vấn chọn và đưa 2 bảng dslop và dssv vào tham gia truy vấn

Đưa các trường Tenlop, hotensv, ngaysinh, nganh_hoc vào vùng lưới QBE

Trong hàng Criteria của trường Tenlop: Like "T*"

4.2.3. Chọn các bản ghi không phù hợp với một giá trị nào đó

Dùng toán tử **Not**

Ví dụ: Tạo một truy vấn để hiển thị danh sách những sinh viên có Tenlop không bắt đầu là **"T"** bao gồm các trường: Tenlop, Hotensv, Ngaysinh, nganh_hoc.

4.2.4. Định nhiều tiêu chuẩn trong lựa chọn

Dùng phép “Và” và phép “Hoặc” trong một trường

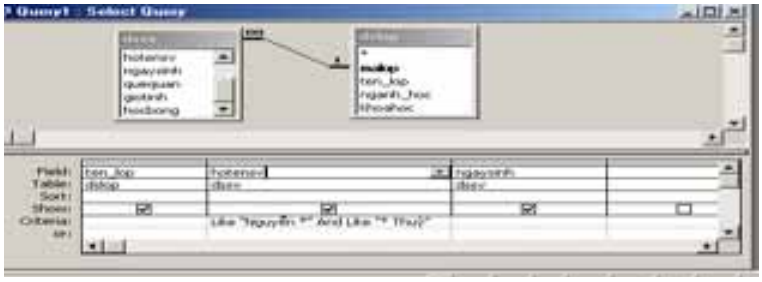
Muốn quy định nhiều tiêu chuẩn trong cùng một trường, chúng ta phải sử dụng toán tử AND (và) cùng toán tử OR (hoặc).

Ví dụ: Tạo một truy vấn để hiển thị danh sách những sinh viên có Họ là "**Nguyễn"** và Tên "**Thuỷ"** bao gồm các trường: Tenlop, Hotensv, Ngaysinh.

Tạo truy vấn chọn và đưa 2 bảng dslop và dssv vào tham gia truy vấn

Đưa các trường Tenlop, hotensv, ngaysinh vào vùng lưới QBE

Trong hàng Criteria của trường Hotensv : Like "Nguyễn *" and "* Thuỷ"



Dùng phép “Và” và phép “Hoặc”trên nhiều trường

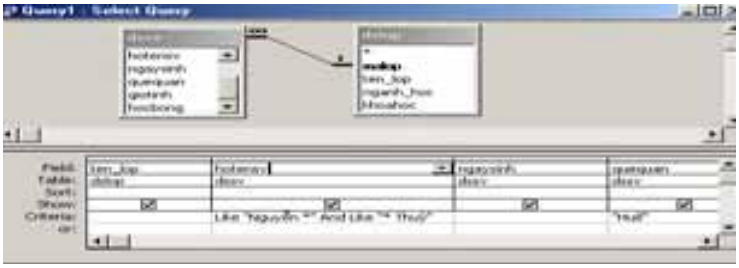
Ví dụ: Tạo một truy vấn để hiển thị danh sách những sinh viên có Họ là "**Nguyễn"** và tên "**Thuỷ**" và có quê quán ở "**Huế"** bao gồm các trường: Tenlop, Hotensv, ngaysinh Quequan.

Tạo truy vấn chọn và đưa 2 bảng dslop và dssv vào tham gia truy vấn

Đưa các trường Tenlop, hotensv, ngaysinh, quequan vào vùng lưới QBE

Trong hàng Criteria của trường Hotensv : Like "Nguyễn *" and "* Thuỷ"

Quequan : Huế



Tạo một truy vấn để hiển thị danh sách những sinh viên có Họ là "**Lê"** hoặc có quê quán ở "**Đà Nẵng** bao gồm các trường: Tenlop, Hotensv, Ngaysinh, Quequan.

Tạo truy vấn chọn và đưa 2 bảng dslop và dssv vào tham gia truy vấn

Đưa các trường Tenlop, hotensv, ngaysinh , Quequanvào vùng lưới QBE

Trong hàng Criteria của trường Hotensv : Like "Lê *"

Trong hàng or của trường Quequan : Đà Nẵng

4.2.5. Chọn các bản ghi có chứa có giá trị

Chúng ta có thể chọn các bản ghi có chứa hoặc không chứa giá trị, chẳng hạn như tìm những sinh viên mà không có số điện thoại nhà ở.

Access cung cấp 2 phép toán

Phép toán	Ý nghĩa
IS NULL	Trường không chứa giá trị
IS NOT NULL	Trường có chứa giá trị

Ví dụ:

Tạo một truy vấn để hiển thị danh sách những sinh viên không có học bổng bao gồm các trường: Tenlop, Hotensv, hocbong.

4.2.6. Chọn các bản ghi thuộc danh sách các giá trị nào đó

Chúng ta có thể sử dụng phép toán **IN** (***Danh sách giá trị***)

Ví dụ

Tạo một truy vấn để hiển thị danh sách những sinh viên thuộc lớp "**Tin K23**" hoặc "**Tin K24**" hoặc "**Tin K25**" bao gồm các trường: Tenlop, Hotensv, hocbong.

4.2.7. Tham chiếu đến các trường khác

Nếu trong biểu thức chọn của truy vấn, các tính toán tham chiếu đến các trường phải đặt trong dấu [ ], trường hợp tham chiếu đến trường của bảng khác phải chỉ rõ bảng nguồn của nó. **[Tên bảng]![Tên trường].**

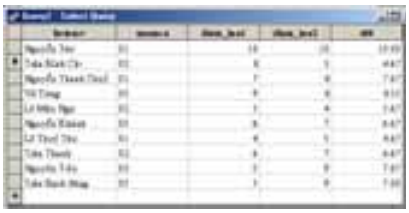
4.2.8. Tạo trường kiểu biểu thức

Ví dụ

Cho 2 bảng dữ liệu Dssv(**Masv**, hotensv, ngaysinh, quequan, gioitinh)

Dsdiem(**Masv**, mamon, diem_lan1, diem_lan2)

Tạo truy vấn để hiển thị các thông tin: Hotensv, mamon, diem_lan1, diem_lan2, dtb, trong đó dtb=(diem_lan1+ diem_lan2*2)/3.



☞ Chú ý

Sau khi thực hiện truy vấn chúng ta không thể thay đổi giá trị trong trường kiểu biểu thức, tuy nhiên nếu thay đổi giá trị trong trường tham gia biểu thức thì kết quả trong trường kiểu biểu thức cũng thay đổi theo.

4.2.9 .Chọn giá trị duy nhất

Theo mặc định, access sẽ chọn tất cả các bản ghi thoã mãn điều kiện, tuy nhiên đôi khi có nhiều giá trị giống nhau được lặp đi lặp lại, do đó để cô đọng dữ liệu thì chúng ta có thể quy định thuộc tính duy nhất trong khi hiển thị

Thuộc tính Unique-values

Chọn Yes: Không thể hiện các giá trị trùng nhau

Chọn No: Thể hiện các giá trị trùng nhau

Thuộc tính Unique-Records

Chọn Yes: Không thể hiện các bản ghi trùng nhau

Chọn No: Thể hiện các bản ghi trùng nhau

4.2.10. Chọn các giá trị đầu

Khi hiển thị truy vấn đôi khi chúng ta muốn hiển thị một số bản ghi đầu tiên nào đó thoả mãn các điều kiện thì sử dụng thuộc tính **Top values**.

5. TRUY VẤN DỰA TRÊN NHIỀU BẢNG DỮ LIỆU

Để tạo truy vấn dựa trên nhiều bảng dữ liệu thì các bảng đó phải được thiết lập mối quan hệ, nếu các bảng không thiết lập mối quan hệ thì khi truy vấn dữ liệu access sẽ cho ra những bộ dữ liệu là tích Đề-Các giữa các bộ dữ liệu trong các bảng.

5.1. Liên kết các bảng trong truy vấn

Khi các bảng dữ liệu được thiết lập mối quan hệ thì trường nối với nhau gọi là trường liên kết, trong access phân biệt 3 loại liên kết sau

5.1.1. Liên kết nội (Inner join)

Đây là loại liên kết rất phổ biến nhất giữa 2 bảng dữ liệu. Trong đó dữ liệu khi thể hiện trên Dynaset sẽ gồm những bản ghi mà dữ liệu chứa trong trường liên kết ở hai bảng phải giống nhau hoàn toàn.

5.1.2. Liên kết ngoại (Outer join)

Đây là loại liên kết cho phép dữ liệu thể hiện trên Dynaset của một trong hai bảng tham gia có nội dung trường liên kết không giống nội dung trong trường tương ứng của bảng còn lại. Liên kết ngoại được chia làm hai loại

Left Outer Join: Trong kiểu liên kết này, dữ liệu ở bảng bên "1" thể hiện toàn bộ trên Dynaset và chỉ những bản ghi bên bảng "nhiều" có nội dung trong trường liên kết giống trường tương ứng bên bảng "1".

Right Outer Join: Trong kiểu liên kết này, dữ liệu ở bảng bên "nhiều" thể hiện toàn bộ trên Dynaset và chỉ những bản ghi bên bảng "1" có nội dung trong trường liên kết giống trường tương ứng bên bảng "nhiều".

5.1.3. Tự liên kết (Seft join)

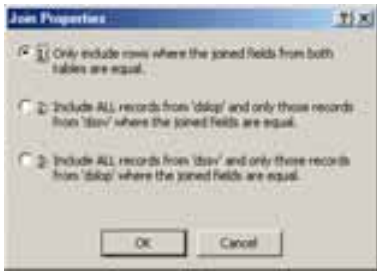
Là kiểu liên kết của một bảng dữ liệu với chính nó. Trong đó một bản ghi trong bảng dữ liệu sẽ liên kết với những bản ghi khác trong bảng dữ liệu đó. Tự liên kết có thể hiểu như là liên kết nội hay liên kết ngoại từ một bảng vào một bảng sao chính nó. Để thực hiện việc tạo tự liên kết chúng ta phải đưa một bảng vào tham gia truy vấn 2 lần.

5.2.Tạo liên kết ngoại

Muốn tạo liên kết ngoại giữa 2 bảng dữ liệu ta thực hiện

Tools/Relationships

Double click vào đường liên kết giữa 2 bảng dữ liệu, chọn Join Type



Trong hộp thoại Join Properties chọn mục 2 hoặc mục 3.

5.3.Tạo một tự liên kết

Để tạo một tự liên kết chúng ta thực hiện

Tạo truy vấn mới và đưa bảng dữ liệu vào truy vấn 2 lần

Tạo các liên kết

Ví dụ:

Cho bảng Dsdiem(Masv, Hoten, Diem_lan1, Diem_lan2)

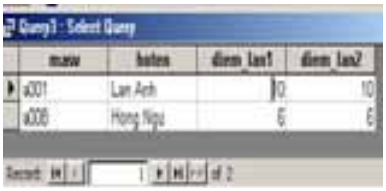
Tạo truy vấn để hiển thị danh sách những sinh viên có điểm thi Lần 1 bằng điểm thi lần 2...

Tạo truy vấn mới và đưa bảng Dsdiem vào tham gia truy vấn 2 lần

Tạo liên kết nội (Inner join) giữa 2 trường Masv

Tạo liên kết nội từ trường Diem_lan1 vào Diem_lan2

Đưa các trường vào vùng lưới QBE và xem kết quả



5.3. Tự động tìm kiếm dữ liệu (Auto lookup)

Khi nhập dữ liệu vào Dynaset, chức năng tự động tìm kiếm dữ liệu cho phép chúng ta chỉ nhập dữ liệu ở các trường của bảng bên "nhiều" (Ở quan hệ 1-∞) còn Acces sẽ tự động tìm kiếm dữ liệu tương ứng trên bảng "1" để hiển thị.

Chức năng Auto Lookup hoạt động trong các truy vấn mà hai bảng tham gia có mối quan hệ 1-∞.

Tạo một truy vấn có sử dụng chức năng Auto Lookup chúng ta thực hiện:

Tạo truy vấn và đưa 2 bảng vào tham gia truy vấn

Đưa trường liên kết cuả bảng bên nhiều vào vùng lưới QBE

Đưa các trường cần hiển thị dữ liệu của bảng bên "1".

☞ ***Chú ý:*** Khi nhập dữ liệu chỉ nhập dữ liệu ở các trường của bảng "nhiều"

6. TÍNH TỔNG TRONG TRUY VẤN CHỌN

Trong thực tế, chúng ta thường có những câu hỏi đặt ra về việc nhóm dữ liệu nào đó, chẳng hạn trong tháng 10 công ty xăng dầu XYZ bán được bao nhiêu lít xăng, tổng thành tiền bao nhiêu?

Trong Access chúng ta có thể thực hiện một số phép tính lên một nhóm bản ghi bằng cách dùng truy vấn tính tổng

Một số phép toán thường sử dụng

Phép toán	Ý nghĩa
Sum	Tính tổng các giá trị của một trường
Avg	Tính giá trị trung bình của một trường
Min	Tính giá trị nhỏ nhất của một trường
Max	Tính giá trị lớn nhất của một trường
Count	Đếm số giá trị khác rỗng có trong một trường
First	Giá trị của trường ở bản ghi đầu tiên trong bảng
Last	Giá trị của trường ở bản ghi cuối cùng trong bảng
Where	Giới hạn điều kiện khi tính tổng
Expression	Trường kiểu biểu thức

☞ ***Chú ý:*** Khi thực hiện truy vấn Total, dữ liệu trong bảng kết quả của nó trình bày không thể chỉnh sửa.

6.1. Tạo truy vấn tính tổng

Tạo truy vấn chọn và đưa các bảng vào tham gia truy vấn

Thực hiện lệnh: View/Totals

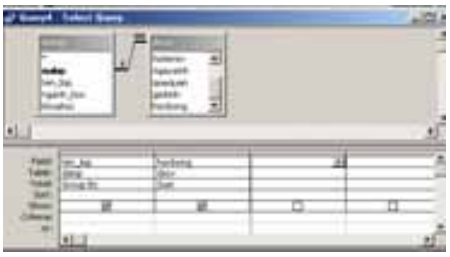
Trong vùng lưới QBE:

Tại hàng Field chọn các trường

Total chọn các phép toán tương ứng.

Criteria: Chọn điều kiện giới hạn tính tổng (Nếu có)

Lưu và thực hiện truy vấn



6.2. Tính tổng của tất cả các bản ghi

Tạo truy vấn chọn.

Đưa các bảng cần thiết vào truy vấn.

Đưa các trường cần thiết vào vùng lưới QBE

Chọn menu View/Totals, dòng Total sẽ xuất hiện trên vùng lưới.

Trong hàng total của mỗi trường chọn phương pháp tính tổng (Sum, Avg, count...). Vì đang tính tổng của tất cả các bản ghi nên không được phép chọn “Group by” ở bất kỳ trường nào.

Chuyển sang DataSheet View để xem kết quả. (View/ Datasheet View)

Ví dụ:

Để quản lý các mặt hàng bán ra trong một cửa hàng người ta sử dụng 2 bảng dữ liệu như sau:

Dshang(**Mahang**, tenhang, dongia)

Dskhach(Mahang, tenkhach, ngaymua, diachi, soluong, thanhtien)

Tạo truy vấn để thống kê xem trong cửa hàng bán bao nhiêu mặt hàng và trung bình đơn giá của mỗi mặt hàng là bao nhiêu?

Tạo truy vấn và đưa bảng Dshang vào tham gia truy vấn

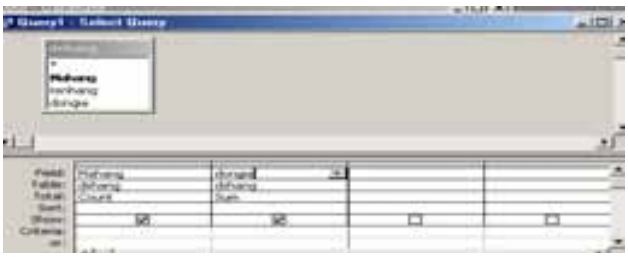
Đưa 2 trường Mahang và dongia vào vùng lưới QBE.

Chọn View/Totals

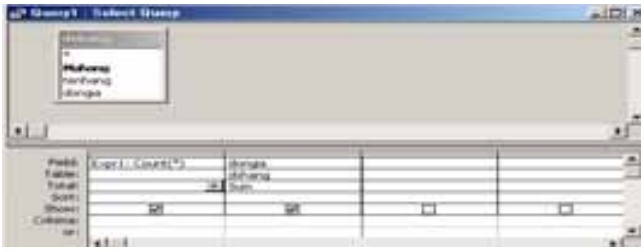
Trong hàng Total của trường Mahang chọn phép toán Count

Trong hàng Total của trường Dongia chọn phép toán Avg

Chọn View/Datasheet View để xem kết quả.



☞ ***Chú ý:*** Có thể đếm số bản ghi bằng cách dùng Count(*)



6.3. Tính tổng trên từng nhóm bản ghi

Trong nhiều trường hợp chúng ta có thể tính toán trên một nhóm bản ghi nào đó. Chúng ta quy định khi thiết kế truy vấn những trường nào sẽ được tính theo nhóm, trường nào sẽ được tính tổng.

Tạo truy vấn

Đưa các bảng tham gia vào truy vấn

Đưa các trường vào vùng lưới

Chọn View/Totals

Tại hàng total

Chọn “Group by” cho trường làm khóa để nhóm

Chọn các phép toán tính tổng (Sum,count..) cho các trường còn lại

Chọn View/Datasheet View để xem kết quả.

Ví dụ:

Tạo một truy vấn để tính tổng soluong, thanhtien của mỗi mặt hàng bán được là bao nhiêu?

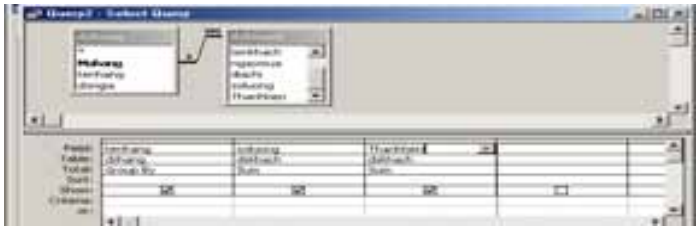
Tạo truy vấn và đưa 2 bảng dshang và dskhach vào tham gia truy vấn

Chọn View/Totals

Đưa các trường Tenhang, soluong, thanhtien vào vùng lưới QBE.

Tại hàng Total của trường Tenhang: Chọn phép toán Group by

Tại hàng Total của trường Soluong, thanhtien: Chọn phép toán SUM.



Chọn View/Datasheet View để xem kết quả.

Query2 : Select Query

tenhang	SumOfsoluong	SumOfThanhtien
Dau hoa	35	140000
Xang A83	83	415000
XangA92	78	468000

Record: 3 of 3

Ta có thể thay đổi tiêu đề cột trong khi thực hiện truy vấn tính tổng như sau

tenhang	Tong so luong	Tong thanh tien
Dau hoa	35	140000
Xang A83	83	415000
XangA92	78	468000

6.4. Tính tổng trên nhiều nhóm bản ghi

Access cho phép tính tổng không chỉ trên một mà còn nhiều nhóm bản ghi. Để làm được điều đó chúng ta chọn “Group by” trên nhiều trường và khi thực hiện

Access sẽ theo thứ tự từ trái sang phải trường bên trái là nhóm mức cao hơn, trường kế tiếp theo là nhóm mức thấp hơn.

Ví dụ:

Tạo truy vấn để tính tổng thanhtien của mỗi mặt hàng bán được theo từng năm nào đó?

Tạo truy vấn và đưa 2 bảng dshang và dskhach vào tham gia truy vấn

Chọn View/Totals

Đưa các trường tenhang, ngaymua, thanhtien vào vùng lưới QBE

Tại hàng Total của trường tenhang chọn phép toán Group by

Tại hàng field của trường ngayban sử dụng hàm year([ngayban]) và tại hàng total chọn phép toán Group by.

Tại hàng Total của trường Thanhtien chọn phép toán Sum.

Thay đổi tiêu đề cột trong truy vấn.

Chọn View/ Datasheet View để xem kết quả

Query4 : Select Query

tenhang	Nam	Tong thanh tien
Dau hoa	1997	24000
Dau hoa	1998	88000
Dau hoa	1999	28000
Xang A83	1997	45000
Xang A83	1998	250000
Xang A83	1999	120000
XangA92	1997	120000
XangA92	1998	168000
XangA92	1999	180000

ecord: 3 of 9

6.5. Lập biểu thức chọn cho các trường dùng để nhóm khi tính tổng

Cũng như với những truy vấn khác, chúng ta có thể lập biểu thức chọn cho truy vấn tính tổng theo từng nhóm.

Để thực hiện công việc này, chúng ta lập biểu thức điều kiện ngay hàng Criteria của trường “group by”.

Ví dụ:

Tạo truy vấn để tính tổng thanhtien của mỗi mặt hàng bán được trong năm 1999.

Tạo truy vấn và đưa 2 bảng dshang và dskhach vào tham gia truy vấn

Chọn View/Totals

Đưa các trường tenhang, ngaymua, thanhtien vào vùng lưới QBE

Tại hàng Total của trường tenhang chọn phép toán Group by

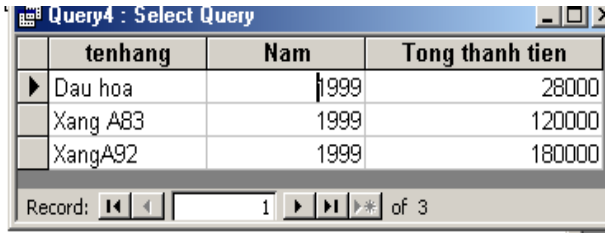
Tại hàng field của trường ngayban sử dụng hàm year([ngayban]) và tại hàng total chọn phép toán Group by.

Tại hàng Total của trường Thanhtien chọn phép toán Sum.

Tại hàng Criteria của trường ngaymua: gõ vào giá trị 1999

Thay đổi tiêu đề cột trong truy vấn.

Chọn View/ Datasheet View để xem kết quả



Query4 : Select Query

tenhang	Nam	Tong thanh tien
Dau hoa	1999	28000
Xang A83	1999	120000
XangA92	1999	180000

Record: 1 of 3

6.6. Lập biểu thức chọn để giới hạn những bản ghi.

Trong các phần trước, chúng ta đã biết cách giới hạn các bản ghi trong truy vấn theo một điều kiện nào đó, sự giới hạn này gọi là **giới hạn sau khi tính tổng.**

Bây giờ chúng ta lập biểu thức chọn giới hạn số bản ghi trước khi đưa vào tính tổng trong truy vấn gọi là **giới hạn trước khi tính tổng** .

Cách tạo

Tạo truy vấn mới và đưa các bảng tham gia vào truy vấn

Đưa các trường vào vùng lưới QBE.

Chọn menu View/Total

Thiết lập hàng Total thành Where đối với trường chúng ta muốn dùng để đặt biểu thức điều kiện giới hạn số bản ghi trước khi tính tổng.

Gõ biểu thức điều kiện tại hàng Criteria tương ứng.

Chuyển sang chế độ datasheet view để xem kết quả.

Ví dụ:

Tạo truy vấn để tính tổng soluong, thanhtien của mỗi mặt hàng bán được đối với khách mua hàng có Queuqan ở "**Huế**"

☞ ***Chú ý:*** Trong đa số trường hợp, đặt điều kiện lọc trước và sau khi tính tổng có giá trị khác nhau.

6.7. Dùng truy vấn để cập nhật bản ghi

Khi truy vấn chỉ dựa trên một bảng, hoậchi bảng có quan hệ 1-1 thì tất cả các trường đều có thể thay đổi, cập nhật. Trong trường hợp có nhiều hơn hai bảng tham gia truy vấn mà có quan hệ 1-∞ thì sẽ phức tạp hơn.

6.7.1. Khi nào dữ liệu trong trường có thể sửa đổi được

Bảng sau đây liệt kê các trường hợp khi nào một trường trong kết quả truy vấn hay trong biểu mẫu có thể sửa đổi được.

Loại truy vấn hay truờng	Dữ liệu trong trường có cho phép sửa đổi hay không?
Truy vấn dựa trên 1 bảng	Có
Truy vấn dựa trên nhiều bảng có quan hệ 1-1	Có

Truy vấn dựa trên nhiều bảng có quan hệ 1-∞	Thông thường
Truy vấn Tham khảo chéo	Không
Truy vấn tính tổng	Không
Truy vấn với thuộc tính Unique values được thiết lập thành Yes	Không
Truy vấn hội	Không
Truy vấn chuyển nhượng	Không
Trường kiểu biểu thức	Không
Trường trong bản ghi đã bị xoá hoặc bị khoá bởi một người khác trong môi trường nhiều người sử dụng	Không

6.7.1. Chỉnh sửa bản ghi trong truy vấn dựa trên hai bảng có quan hệ 1-∞

Trong truy vấn dựa trên dữ liệu là hai bảng có quan hệ 1-∞, chúng ta có thể sửa đổi tất cả các trường trừ trường liên kết của bảng bên "1". Tuy nhiên có hai trường hợp mà vẫn có thể sửa đổi dữ liệu trong trường liên kết bên "1" là:

Có thể sửa đổi dữ liệu trường liên kết bên bảng "1" trong trường hợp liên kết ngoại và trương tương ứng của liên kết bên bảng "nhiều" không chứa giá trị.

Có thể sửa đổi dữ liệu trường liên kết bên bảng "1" trong trường hợp đã khai báo thuộc tính tham chiếu toàn vẹn.

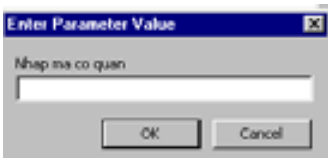
7. TRUY VẤN THAM SỐ (Parameter Query)

7.1. Khái niệm

Nếu thường xuyên chạy cùng một truy vấn, nhưng mỗi lần một tiêu chuẩn khác nhau, thay vì phải thiết kế lại truy vấn sau mỗi lần thực hiện, có thể tiết kiệm thời gian bằng cách tạo truy vấn tham số. Khi thực hiện loại này Access sẽ nhắc nhập điều kiện chọn trong hộp thoại enter parameter Value.

Ví dụ:

Giả sử thường xuyên chạy một truy vấn để liệt kê danh sách nhân viên của một cơ quan nào đó có mã cơ quan nhập vào bất kỳ.



☞ ***Chú ý:*** Nội dung các tham số mà chúng ta nhập vào có thể là hằng (số,chuỗi,ngày..) nhưng không được biểu thức.

7.2. Tạo truy vấn tham số

Tạo truy vấn chọn và đưa các bảng cần thiết vào tham gia truy vấn.

Kéo các trường cần thiết vào vùng lưới QBE.

Tại hàng Criteria gõ vào biểu thức có chứa tham số với chú ý tên tham số phải nằm giữa 2 dấu ngoặc vuông ([ ])

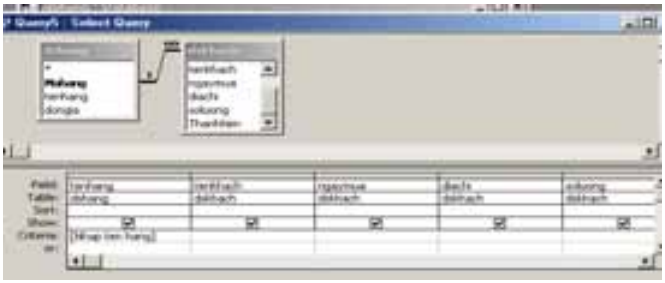
Tên tham số cũng là chuỗi nhắc nhở.Access cho phép có khoảng trắng và độ dài tối đa 255 ký tự.

Quy định kiểu dữ liệu cho tham số: Chọn queries/ parameter query.

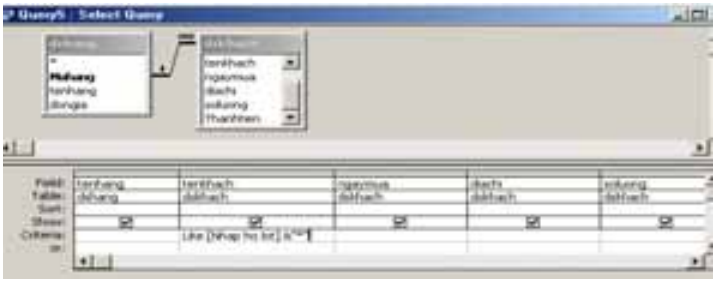
Trong hộp thoại query parameters: Trong mục Parameter chọn tham số, trong mục Data type chọn kiểu dữ liệu tương ứng.

Ví dụ:

Tạo truy vấn để hiển thị danh sách các khách hàng mua một mặt hàng nào đó (mặt hàng được nhập bất kỳ từ bàn phím).



Tạo truy vấn để hiển thị danh sách các khách hàng mua hàng có họ lót được nhập từ bàn phím.



7.3. Truy vấn nhiều tham số

Có thể tạo truy vấn , khi chạy truy vấn nhập nhiều dữ liệu cho điều kiện chọn lựa. Muốn vậy tạo truy vấn nhiều tham số.

Ví dụ:

Tạo một truy vấn hiển thị danh sách các khách hàng mua hàng trong khoảng thời gian nào đó (Thời gian được nhập từ bàn phím).

Tạo truy vấn chọn và đưa các 2 bảng dshang và dskhach vào tham gia truy vấn.

Kéo các trường tenhang, tenkhach, ngaymua vào vùng lưới QBE.

Tại hàng Criteria của trường NGAYSINH chọn:

Between [Từ ngày] and [Đến ngày]

Chọn query/Parameter khai báo kiểu dữ liệu cho 2 tham số là date/time.

Khi chạy truy vấn sẽ cho kết quả sau

7.4. Kết hợp giữa truy vấn tham số và truy vấn tính tổng

Trong nhiều bài toán quản lý người ta thường gặp những yêu cầu như: Hãy thống kê xem mỗi loại hàng trong một tháng nào đó bán được với tổng số lượng là bao nhiêu? Tổng thành tiền là bao nhiêu? (Tháng được nhập từ bàn phím). Vì vậy trước hết chúng ta phải thực hiện truy vấn tính tổng xong mới kết hợp truy vấn tham số.

Ví dụ:

Hãy tạo một truy vấn để hiển thị tổng thanhtien của mỗi mặt hàng bán được trong một năm nào đó (Năm được nhập từ bàn phím).

Tạo một truy vấn chọn, đưa bảng Dshang và dskhach vào tham gia truy vấn.

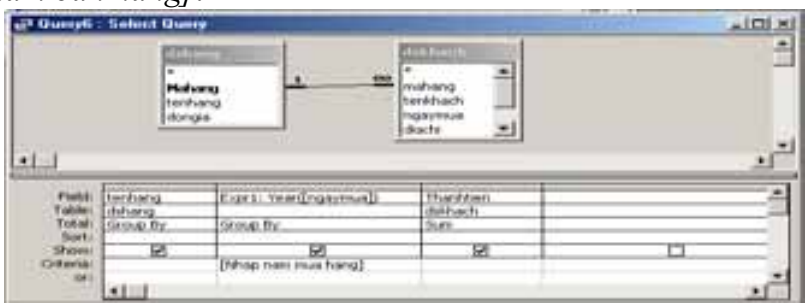
Đưa các trường tenhang, ngayban, thanhtien vào vùng lưới QBE.

Chọn View/Totals

Trong hàng Total: Chọn Group by đối với trường tenhang, và year([ngayban])

Trong hàng Criteria đối với trường Ngaysinh, ta chọn tham số sau:

[Nhap vao nam ban hang].



8. TRUY VẤN THAM KHẢO CHÉO (Crosstab query)

8.1. Khái niệm

Truy vấn tham khảo chéo là loại truy vấn dùng để tóm lược dữ liệu và trình bày kết quả theo dạng như một bảng tính. Truy vấn tham khảo chéo cũng có thể thống kê một khối lượng dữ liệu lớn và trình bày đơn giản hơn do đó thường sử dụng để so sánh dữ liệu.

8.2. Tạo truy vấn tham khảo chéo

Muốn tạo một truy vấn tham khảo chéo chúng ta phải xác định được 3 yếu tố chính: Trường làm tiêu đề cột (Duy nhất 1 trường), trường làm tiêu đề hàng (Có thể nhiều trường), trường tính giá trị (Duy nhất 1 trường).

Cách tạo

Tạo truy vấn chọn và đưa các bảng vào tham gia truy vấn

Đưa các trường vào vùng lưới QBE

Chọn Query/Crosstab

Quy định trường làm tiêu đề cột

Tại hàng Total: Bắt buộc chọn phép toán Group by

Tại hàng Crosstab: Chọn Column heading

Quy định trường làm tiêu đề hàng

Tại hàng Total: Ít nhất một trong các trường phải chọn phép toán Group by

Tại hàng Crosstab: Chọn Row heading

Quy định trường tính giá trị

Tại hàng Total: Chọn phép toán thích hợp

Tại hàng Crosstab: Chọn Value

Ví dụ:

Cho 2 bảng dữ liệu Dstruong(**matruong**, tentruong, sodt)

Danhsach(matruong, hoten, ngaysinh, gioitinh, xeploai)

Tạo một truy vấn Crosstab để phản ánh tổng số lượng sinh viên xếp mỗi loại của trong từng trường bao nhiêu.?

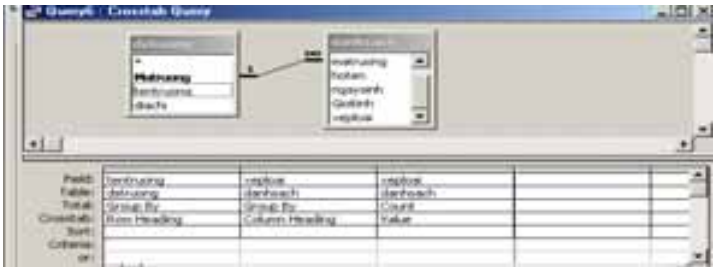
Tạo truy vấn và đưa 2 bảng dữ liệu vào tham gia truy vấn, đưa các trường tentruong và xeploai vào vùng lưới QBE. (Trường Xeploai đưa vào 2 lần)

Chọn Query/ crosstab query

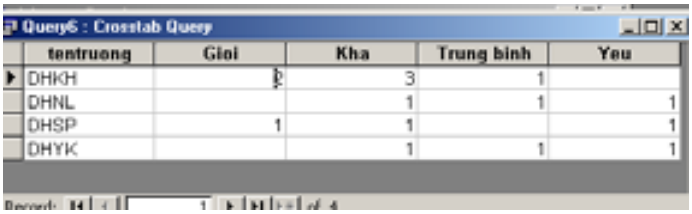
Tại hàng Total của trường tentruong: Chọn phép toán Group by, hàng crosstab: chọn Row heading

Tại hàng Total của truờng Xeploai: Chọn phép toán Group by, hàng Crosstab chọn Column heading.

Tại hàng Total của truờng Xeploai: Chọn phép toán Count, hàng Crosstab chọn Value.



Chọn View/ Datasheet View để xem kết quả



Query6 : Crosstab Query

tentruong	Gioi	Kha	Trung binh	Yeu
DHKH	2	3	1	
DHNL		1	1	1
DHSP	1	1		1
DHYK		1	1	1

8.3. Định dạng cho tiêu đề cột

Với truy vấn Crosstab, chúng ta có thể can thiệp nhiều hơn về cách trình bày tiêu đề cột trong bảng. Chúng ta có thể thay đổi bằng cách đặt lại thuộc tính Column Heading của truy vấn. Thuộc tính này cho phép chúng ta: Chỉ định sắp xếp các tiêu đề cột.

Muốn định dạng tiêu đề cột thực hiện các bước sau:

Tạo truy vấn Crosstab.

Chuyển sang chế độ Design View.

Mở bảng thuộc tính truy vấn.

Tại hàng Column Heading : Gõ các tiêu đề cột theo thứ tự mà chúng ta muốn

Các giá trị này phải cách nhau bởi dấu chấm phẩy (;)

9. TRUY VẤN HÀNH ĐỘNG

9.1. Các loại truy vấn hành động

Truy vấn hành động giúp người sử dụng tạo bảng mới hay sửa đổi dữ liệu trong các bảng. Có 4 loại truy vấn hành động:

Truy vấn tạo bảng (Make table query): Tạo bảng mới từ một bảng hay nhiều bảng đã tồn tại dữ liệu.

Truy vấn cập nhật (Update query): Dùng để cập nhật dữ liệu cho một hoặc nhiều trường trong bảng dữ liệu.

*Truy vấn xoá (Delete query):*Xoá các bản ghi thoả mãn các điều kiện từ một hay nhiều bảng dữ liệu.

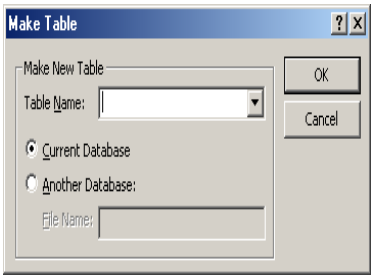
Truy vấn nối (Append query): Nối một số bản ghi từ một hoặc nhiều bảng dữ liệu vào sau một hoặc nhiều bảng dữ liệu khác.

9.2. Truy vấn tạo bảng

Truy vấn tạo bảng sẽ tạo ra một bảng mới bằng cách rút các bản ghi thoã mãn các điều kiện nào đó.

Cách tạo truy vấn

Để tạo truy vấn tạo bảng chúng ta tạo truy vấn chọn và đưa bảng vào tham gia truy vấn. Đưa các trường vào vùng lưới QBE



Chọn query/ make table query

Trong mục Table name: Đặt tên mới cho bảng muốn tạo.

Chọn Current Database: CSDL hiện thời

Another Database: Tạo bảng trong CSDL khác.

Chọn các điều kiện (Nếu có).

9.3. Truy vấn xoá

Truy vấn xoá giúp chúng ta loại bỏ các bản ghi thoả mãn một số điều kiện nào đó

Cách tạo truy vấn

Để tạo truy vấn xoá chúng ta tạo truy vấn chọn và đưa bảng vào tham gia truy vấn.

Chọn query/ Delete query

Trong vùng lưới QBE tại hàng Field chọn các trường cần so sánh với điều kiện xoá

Tại hàng Delete: Chọn phép toán Where

Tại hàng Criteria: Chọn điều kiện xoá

Ví dụ:

Tạo truy vấn để xoá những sinh viên có matruong là "SP"

9.3. Truy vấn cập nhật

Truy vấn này dùng để cập nhật giá trị hoặc sửa đổi giá trị của các trường trong bảng dữ liệu.

Cách tạo truy vấn

Tạo một truy vấn chọn và đưa bảng vào tham gia truy vấn

Chọn Query/Update query

Tại hàng Field: Chọn trường cần cập nhật dữ liệu

Tại hàng Update to: Chọn Biểu thức cần tính giá trị

Tại hàng Criteria: Chọn điều kiện (nếu có).

Ví dụ:

Cho 2 bảng dữ liệu Dsphong(**tenphong**, dongia)

Dskhachtro (tenphong, ngayden, ngaydi, thanhtien)

Tạo truy vấn để cập nhật giá trị cho trường Thanhtien=(ngaydi-ngayden)*dongia

9.4. Truy vấn nối dữ liệu

Truy vấn nối dữ liệu dùng để nối dữ liệu từ một bảng này vào sau một bảng khác.

Cách tạo truy vấn

Tạo truy vấn chọn và đưa bảng dữ liệu vào để nối với bảng khác tham gia truy vấn.

Chọn Queries/Append query

Trong mục Table name: Chọn bảng cần nối vào và chọn OK.

Chọn Current Database: CSDL hiện thời

Another Database: Tạo bảng trong CSDL khác.

Trong vùng lưới QBE của truy vấn tại hàng Field:

Đưa các trường của bảng gốc vào.

Trong hàng Append to: Đưa các trường tương ứng của bảng cần nối và đặt điều kiện nếu cần thiết.

☞ ***Chú ý***

Trong truy vấn nối dữ liệu thì các trường được nối với nhau tương ứng phải cùng kiểu dữ liệu. Nếu các trường tương ứng không có cùng kiểu dữ liệu thì sẽ không được nối. Nếu các trường có Field size không phù hợp thì tuỳ theo việc nối dữ liệu mà sẽ cắt bớt hoặc thêm vào ký tự trắng.

Chương 4 NGÔN NGỮ SQL

1. GIỚI THIỆU VỀ NGÔN NGỮ SQL

SQL là một ngôn ngữ dùng để truy xuất dữ liệu, cập nhật, thay đổi và quản lý các CSDL quan hệ.

Khi tạo một truy vấn thì ACCESS tự động xây dựng các câu lệnh SQL tương ứng.

Một số truy vấn của ngôn ngữ SQL như sau:

Truy vấn hội (Union Query)

Truy vấn chuyển nhượng (pass through Query)

Truy vấn định nghĩa dữ liệu (Data Definition Query)

Truy vấn con (Sub Query)

☞ ***Chú ý***

Trong SQL mỗi câu lệnh có thể được viết trên nhiều hàng và kết thúc mỗi câu lệnh là dấu chấm phẩy (;)

2. SQL XỬ LÝ TRÊN BẢNG DỮ LIỆU

2.1. Tạo bảng mới

Cú pháp

Create table <Table name>(<Field1> <Type>[(Size)][, <Field2> <Type>[(Size)]..... [Constraint < Primary key name> primary key <Fieldname key>] [Constraint <Index name> Unique <Field name Index>]

Chức năng: Tạo cấu trúc của một bảng mới.

Giải thích chức năng các tham số

Table name: Tên bảng cần tạo

Fieldname1, Fieldname2....: Các trường trong bảng cần tạo, tối thiểu 1 trường.

Type: Các kiểu dữ liệu tương ứng.

Size : Kích thước của trường

Primary key name : Tên khoá chính

Fieldname key : Trường làm khoá chính

Index name : Tên chỉ mục

Fieldname Index : Truờng làm chỉ mục

Constraint.....Primary key : Dùng để thiết lập khoá chính

Constraint..... Unique: Thiết lập chỉ mục (Không trùng nhau)

☞ ***Chú ý:*** Đối với các kiểu dữ liệu khi sử dụng trong SQL

Kiểu dữ liệu chuẩn	**Khai báo tương ứng trong SQL**
Text	Text(Size)
Byte	Byte
Integer	Short
Long Integer	Long
Single	Single
Double	Double
Date/Time	Datetime
Yes/No	Yesno
OLE Object	OLE Object
Currency, Memo, Counter	Currency, Memo, Counter

Ví dụ : Tạo bảng DSTRUONG có cấu trúc sau

Fieldname	Data Type	size
Matruong	Text	2
Tenruong	Text	20
SODT	Text	9

Create table dstruong(matruong text(2), tentruong text(20), Sodt text(9));

Ví dụ Tạo bảng DSHOCVIEN có cấu trúc sau

Fieldname	Data Type	size	
Mahv	Text	4	(Khoá chính)
Tenhv	Text	30	
Ngaysinh	Date/time		

SODT Text 9 (Lập chỉ mục)

Create table dshocvien(mahv text(4), tenhv text(30), ngaysinh Datetime, sodt text(9), Constraint khoa primary key (mahv), Constraint chimuc unique (sodt));

☞ ***Chú ý:*** Nếu tên trường có ký tự trắng hoặc tên bảng, tên khoá chính, tên chỉ mục thì phải đặt trong cặp dấu [...]

Ví dụ

Create table [Bang NV] ([Ma nv] text(2), [ho va ten] text(30));

2.2. Thay đổi cấu trúc của bảng

2.2.1.Thêm hoặc loại bỏ trường

Cú pháp

Alter table <Table name>[add column <fieldname><type>]

[Drop column <Field name>]

[Add Constraint <Index name> unique <fieldname>]

[Drop Constraint <Index name>]

Chức năng: Thay đổi cấu trúc của bảng

Giải thích:

ADD Column..: Thêm trường vào bảng

Drop column..: Loại bỏ trường ra khỏi bảng

Add Constraint.....: Thêm tên chỉ mục

Drop Constraint..... Loại bỏ tên chỉ mục

Ví dụ: Giả sử đã tồn tại bảng MATHANG cấu trúc sau

Fieldname	*Data Type*	*size*
MAHANG	Text	4
TENHANG	Text	20
GIA	Integer	
MAXN	Text	2
Ngaynhap	Date/time	

Thêm trường SOLUONG có kiểu byte vào bảng MATHANG

Alter Table mathang soluong byte;

Ví dụ : Thêm chỉ mục có ten cmx cho trường MAXN

Alter table mathang add constraint cmx unique Maxn

Ví dụ : Loại bỏ chỉ mục cmx

Alter table mathang drop constraint cmx

Ví dụ : Loại bỏ trường ngaynhap ra khỏi bảng MATHANG

Alter table mathang drop column ngaynhap

2.2.2. Loại bỏ chỉ mục

Cú pháp

Drop Index <Index name> on <Table name>

Chức năng: Loại bỏ 1 chỉ mục nào đó.

2.3. Xoá bảng

Cú pháp

Drop table <Table name>

Chức năng: Xoá bảng dữ liệu nào đó.

Ví dụ: Xoá bảng MATHANG

Drop table MATHANG

3. SQL XỬ LÝ TRÊN TRUY VẤN

3.1.Truy vấn chọn (Select query)

Cú pháp

Select <Scope> <Fieldname1> [AS <New name>].....

From <Table name>

[Where <Condition>]

Chức năng: Tạo truy vấn chọn

Trong đó:

Scope: Phạm vi (Mặc định là ALL, Top n: Lấy n bản ghi đầu tiên)

Nếu có AS <New name> thì sẽ thay thế tên cho Fieldname tương ứng.

Table name: Tên bảng cần lấy dữ liệu.

Condition: Điều kiện để hạn chế dữ liệu.

Ví dụ: Cho bảng dữ liệu DOCGIA sau

Fieldname	Data Type	Size	Description
MADG	Text	2	Mã độc giả (Khoá chính)
MASACH	Text	4	
HOTEN	Text	30	
QUEQUAN	Text	30	
NGAYSINH	Date/time	8	
NGAYMUON	Date/time	8	

Chọn 2 trường MADG và HOTEN

Select MADG, hoten

From docgia;

Chọn 2 trường Masach và hoten mà chỉ những masach bắt đầu là T

Select MADG, hoten

From docgia where masach like “T*”;

Chọn Hoten, quequan, madg cho những độc giả có quê quán ở Huế và đổi tên trường hoten thành Họ và tên

Select hoten AS [Họ và tên], queuqan, madg

From docgia where quequan=”Huế”;

Chọn những độc giả mượn sách trong tháng 8 hoặc năm 1999.

Select * from docgia where month([ngaymuon])=8 or year([ngaymuon])=1999;

3.2. Truy vấn tính tổng (Total query)

Cú pháp

Select .....from.....[where < Condition>]

Group by [Group fieldname]

[Having <Group Condition>];

Chức năng: Tạo một truy vấn tính tổng.

Ví dụ: Cho bảng dữ liệu BANHANG có cấu trúc

Fieldname	Data type	size
Mahang	text	2
Soluong	integer	

Ngayban Date/time 8

Tạo một truy vấn để thống kê xem mỗi loại hàng bán được với số lượng là bao nhiêu?

```
Select mahang, sum([soluong])
From banhang
Group by mahang;
```

Tạo truy vấn để thống kê xem mỗi loại hàng trong thàng 7 bán được với số lượng bao nhiêu? Chỉ hiển thị những loại hàng mà số lượng bán trên 20.

```
Select mahang, sum([soluong])
From banhang
where month([ngayban])=7
Group by mahang
 having sum([soluong])>20;
```

3. 3.Truy vấn tham khảo chéo

Cú pháp

Transform <Value Express>

Select.....From.....Where.....

Group by <Row Heading Field>

Pivot <Column heading Field>

Chức năng: Tạo truy vấn tham khảo chéo

Ví dụ

```
Transform sum([soluong])
select   Mahang, tenhang, sum([soluong]) from dskhang
Group by  tenhang
Pivot Mahang;
```

3.4. Truy vấn tạo bảng

Cú pháp

Select <Field select> into <New Table name)

From <Old Table name>

[Where <Condition>]

Chức năng: Tạo một truy vấn tạo bảng

Ví dụ

Select Mahang, tenhang into Luu

From Dskhang

Where Mahang Like “A*”;

3.5. Truy vấn nối dữ liệu

Cú pháp

Insert into <append Table name>

Select <field select>

From <Table name>

[Where <Condition>]

Chức năng: Tạo truy vấn nối dữ liệu

Nếu chỉ thêm 1 bản ghi với các giá trị cụ thể thì ta thực hiện câu lệnh

Insert into <Table name and Field list>

values <append values>

Ví dụ:

Insert into luu1(Hoten, quequan)

Values (“Nguyen an”, “Hue”)

3.6. Truy vấn cập nhật dữ liệu

Cú pháp

Update <Update Table name>

Set <Field name>=<Express>

[Where <Condition>]

Chức năng: Tạo một truy vấn dùng để cập nhật dữ liệu

Ví dụ:

Update dssv

set [hocbong]=[hocbong]+200000

Where Uutien=”1”;

3.7. Truy vấn xoá

Cú pháp

Delete <Table.>*

From <Delete Table name>

[Where <Condition>]

Chức năng: Dùng để tạo một truy vấn xoá các bản ghi trong bảng theo một hoặc nhiều điều kiện nào đó.

☞ ***Chú ý:*** nếu mệnh đề From chỏ có 1 bảng duy nhất thì không cần liệt kê các bảng trong mệnh đề DELETE.

Ví dụ:

Có 2 bảng dữ liệu DSHS và DSDTHI có quan hệ 1-1 trên trường MAHS. Hãy xoá những học sinh có điểm thi <5 trong bảng DSHS và DSDTHI.

Delete DSHS.*

From DSHS INNER JOIN DSDTHI ON DSHS.MAHS=DSDTHI.MAHS

Where diem<5.;

3.8. Tạo mối quan hệ giữa các bảng

Muốn tạo một truy vấn để truy xuất dữ liệu từ 2 hay nhiều bảng thì phải tạo các mối quan hệ giữa các bảng đó.

Cú pháp

..... From <Table name 1> inner join <Table name 2> ON <Table name 1>.<Field name1>=<Table name 2>.<Field name 2>.....

Hoặc

.....From <Table name 1> Left join/ Right join <Table name 2> ON <Table name 1>.<Field name1>=<Table name 2>.<Field name 2>.....

Ví dụ:

Tạo truy vấn gồm : Matruong, tentruong, hoten từ 2 bảng Dstruong và DSHS dựa vào trường liên kết Matruong.

Select Matruong, tentruong, hoten From Dstruong inner join dshs on Dstruong.matruong=dshs.matruong;

Ví dụ:

Cho 3 bảng dữ liệu Dstruong(**Matruong**, tentruong)

Dskhoa(Matruong, tenkhoa, **Makhoa**)

DSSV(Makhoa, Hoten, Ngaysinh, quequan)

Tạo một truy vấn để hiển thị danh sách sinh viên thuộc mỗi khoa của mỗi trường

Select Distinctrow

Matruong, tentruong, tenkhoa, hoten

From Dstruong inner join (Dskhoa inner join DSSV ON Dskhoa.Makhoa=DSSV.Makhoa)

ON Dstruong.Matruong=Dskhoa.Matruong;

☞ ***Chú ý***

Có thể sử dụng liên kết ngoại trái Left join hoặc phải Right join

3.9. Truy vấn con (Sub query)

Truy vấn con là một mệnh đề Select.....From.....Wheres được lồng ghép vào một trong các mệnh đề sau:

Select.....From.....Where

Select.....Into.....

Insert.....Into.....

Delete.....

Update.....

Cú pháp

Select.....From.....Where.....

<Biểu thức so sánh> ANY|SOME|ALL <Mệnh đề truy vấn con>

<Biểu thức tìm kiếm> IN | NOT IN <Mệnh đề truy vấn con>

EXIST | NOT EXISTS <Mệnh đề truy vấn con>;

Chức năng: Tạo một truy vấn con

Giải thích các tham số

<Biểu thức so sánh>: Là một biểu thức và một phép toán so sánh.

<Biểu thức tìm kiếm>: Là một biểu thức mà tập hợp kết quả của truy vấn con sẽ được tìm kiếm.

<Mệnh đề truy vấn con>: Là dạng mệnh đề ở trong cú pháp và đặt giữa hai dấu ( ).

ANY, SOME: Các bản ghi trong truy vấn chính thoả mãn điều kiện so sánh với bất kỳ hoặc một vài các bản ghi nào truy xuất được từ truy vấn con.

ALL: Các bản ghi trong truy vấn chính thoả mãn với điều kiện so sánh với tất cả bản ghi nào truy xuất được từ truy vấn con.

IN: Các bản ghi trong truy vấn chính mà có tồn tại một vài bản ghi trong truy vấn con có giá trị bằng nó.

NOT IN: Các bản ghi trong truy vấn chính mà không tồn tại một vài bản ghi trong truy vấn con có giá trị bằng nó.

EXISTS (NOT EXISTS): Phép so sánh True/ False để xác định nhận truy vấn con có kết quả là bản ghi nào không.

Ví dụ:

Cho 2 bảng dữ liệu KHO(Mahang, Tenhang, Dongia)

NKBAN (Mahang, Hoten, Ngaymua, Giamgia, Dongia)

Tìm tất cả những mặt hàng mà đơn giá lớn hơn vài mặt hàng được bán với Giamgia là 20%.

*Select * From Kho*

Where dongia> ANY (select dongia From NKBAN Where giamgia=20%);

Tìm những mặt hàng bán ra với giảm giá >=10%.

*Select * from Kho*

Where Mahang IN (Select mahang From NKBAN Where giamgia>=0.1);

Ví dụ:

Cho 2 bảng danh sách DSKH(MAKHACH, HOTEN, QUEQUAN, SDT)

DATHANG(MAKHACH, SOLUONG, NGAYDAT)

Tìm những người khách đặt hàng trước 10/10/99 bao gồm Hoten, Quequan.

Select Hoten, quequan From DSKH

Where Makhach IN (Select makhach from DATHANG Where NGAYDAT<=#10/10/99#;.

3.10. Truy vấn hội (Union Query)

Dùng để nối (Kết hợp) dữ liệu các trường tương ứng từ 2 hay nhiều bảng hoặc truy vấn vào trường.

Cú pháp: *Select .....From.....Where.....*

UNION | UNION ALL

Select.....

Chức năng: Tạo truy vấn hội

Giải thích:

UNION: Không muốn các bản ghi trùng nhau hiển thị

UNION ALL: Hiển thị các bản ghi trùng nhau

Ví dụ:

Cho 2 bảng dữ liệu DHSVIEN(Hoten, Lop, Matruong, Diachi)

CDHSVIEN(Hoten, Lop, Matruong, Diachi)

Sử dụng truy vấn hội để liệt kê Hoten, Lop, Matruong của sinh viên 2 hệ (Địa học, Cao đẳng).

Select Hoten, Lop, Matruong from DHSVIEN

UNION Select Hoten, Lop, Matruong from DHSVIEN

Liệt kê Hoten, Lop của những sinh viên 2 hệ và có quê quán ở Huế

Select Hoten, Lop from DHSVIEN

UNION Select Hoten, Lopfrom DHSVIEN Where Diachi=”Huế”;

Ch□□ng 5 BIỂU MẪU

Từ trước đến nay chúng ta vẫn làm việc một cách đơn điệu với các bảng, truy vấn với cách trình bày dữ liệu hiệu quả nhưng không đẹp mắt. Với biểu mẫu (form) trong Access sẽ giúp chúng ta khắc phục điều này. Biểu mẫu trong Access rất linh động, chúng ta có thể dùng biểu mẫu để nhập, xem, hiệu chỉnh dữ liệu. Hoặc là dùng biểu mẫu để tạo ra các bảng chọn công việc làm cho công việc của chúng ta thuận lợi và khoa học hơn. Hoặc dùng biểu mẫu để tạo ra các hộp thoại nhằm thiết lập các tùy chọn cho công việc quản lý của mình.

1. KHÁI NIỆM VỀ BIỂU MẪU

Nếu chúng ta đã quen điền các tờ biểu, mẫu trong cuộc sống hàng ngày thì chúng ta có thể hình dung một biểu mẫu trong Access cũng vậy. Một biểu mẫu trong Access định nghĩa một tập dữ liệu chúng ta muốn lấy và từ đó đưa vào CSDL. Cũng vậy biểu mẫu cùng có thể dùng để xem xét dữ liệu hay in ra máy in.

Trong môi trường của Hệ QTCSDL Access chúng ta có thể thiết kế các biểu mẫu có hình thức trình bày đẹp, dễ sử dụng và thể hiện đúng các thông tin cần thiết. Chúng ta có thể đưa vào biểu mẫu các đối tượng như văn bản, hình ảnh, đường vẽ két hợp với các màu sắc sao cho biểu mẫu của chúng ta đạt được nội dung và hình thức trình bày ưng ý nhất. Hình thức và cách bố trí các đối tượng ra sao trên biểu mẫu hoàn toàn tùy thuộc vào khả năng thẩm mỹ và ng khiếu trình bày của chúng ta.

2. TÁC DỤNG VÀ KẾT CẤU CỦA BIỂU MẪU

2.1.Tác dụng của biểu mẫu

Biểu mẫu cung cấp một khả năng thuận lợi để hiển thị dữ liệu. Chúng ta có thể xem mọi thông tin của một bản ghi thay vì ở chế độ Datasheet nghèo nàn trước đây bằng chế độ Form View, một phương cách tiên tiến hơn.

Sử dụng biểu mẫu tăng khả năng nhập dữ liệu, tiết kiệm thời gian và ngăn ngừa các lỗi do đánh sai. Chẳng hạn thay vì gõ vào các giá trị của tất cả các trường chúng ta có thể tạo những danh sách (gọi là combo box) để chọn trên biểu mẫu (đây là phương cách áp dụng rất hiệu quả để tránh đánh sai dữ liệu).

Biểu mẫu cung cấp một hình thức trình bày hết sức tiện nghi để xem, nhập và hiệu chỉ các bản ghi trong CSDL. Access cung cấp các công cụ thiết kế biểu mẫu hỗ trợ rất đắc lực cho chúng ta trong việc thiết kê những biểu mẫu dễ sử dụng mà lại có thể tận dụng được các khả năng:

Hình thức thể hiện dữ liệu đẹp, trình bày lôi cuốn với các kiểu font và hiệu ứng đồ họa đặc biệt khác ...

Quen thuộc với người sử dụng vì nó giống các biểu mẫu trên giấy thông thường.

Có thể tính toán được.

Có thể chứa cả biểu đồ.

Có thể hiển thị dữ liệu từ nhiều bảng (hoặc truy vấn)

Tự động hóa một số thao tác phải làm thường xuyên.

2.2. Kết cấu của biểu mẫu

Các thông tin trên biểu mẫu có thể lấy dữ liệu từ một bảng hay truy vấn nào đó, nhưng cũng có thể độc lập đối với cả bảng lẫn truy vấn, chẳng hạn như các đối tượng đồ họa. Dáng vẻ trình bày của biểu mẫu được thực hiện trong quá trình thiết kế.

Tất cả các thông tin thể hiện trên biểu mẫu được chứa trong những đối tượng gọi là *điều khiển* (control). Điều khiển có thể dùng để thể hiện dữ liệu hoặc thực hiện các hàng động hoặc trang trí cho biểu mẫu.

Một số điều khiển được buộc vào với các trường của bảng hay truy vấn, gọi là bảng cơ sở hay truy vấn cơ sở. Do đó chúng ta có thể dùng biểu mẫu để nhập dữ liệu vào

các trường hay lấy dữ liệu từ các trường đó ra để xem. Ví dụ dùng Text box để nhập hay hiển thị chuỗi và số, dùng Object frame để thể hiện hình ảnh.

Một số điều khiển khác trình bày thông tin được lưu trữ trong thiết kế bảng. Ví dụ dùng Label (nhãn) để thể hiện thông tin có tính chất mô tả; đường và các hình khối để tổ chức dữ liệu và làm biểu mẫu có hình thức hấp dẫn hơn.

3. TẠO BIỂU MẪU

3.1. Tạo biểu mẫu tự động với Autoform

Access cung cấp chức năng Autoform cho phép chúng ta tạo biểu mẫu dựa trên các bảng hoặc truy vấn đã được xây dựng trước đó.

Cách tạo

Trong cửa sổ Database, chọn form, chọn New

Trong mục *Choose the table or query Where the object's data comes from:*

Chọn bảng hoặc truy vấn làm nguồn dữ liệu cho form.

Chọn **Autoform Columnar**: Nếu muốn tạo lập biểu mẫu dạng cột, trong đó mỗi trường trong bảng hay truy vấn là một dòng.

Chọn **Autoform Tabular**: Nếu muốn tạo lập biểu mẫu dạng hàng, trong đó mỗi trường trong bảng hay truy vấn là một cột và một bản ghi trong một dòng.

Chọn **Autoform Datasheet**: Nếu muốn tạo lập biểu mẫu theo dạng bảng, trong đó mỗi cột tương ứng một trường và mỗi dòng là một bản ghi.

Chọn OK.

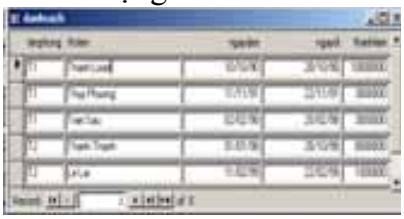
Ví dụ

Cho bảng Danhsach(Tenphong, hoten, ngayden, ngaydi, thanhtien)

Hãy thiết kế biểu mẫu dựa trên chức năng Autoform sử dụng bảng Danhsach làm nguồn dữ liệu.

Biểu mẫu dạng **Autoform Columnar**

Biểu mẫu dạng **Autoform tabular**



Biểu mẫu dạng **Autoform Datasheet**

3.2. Tạo biểu mẫu sử dụng Wizard

Tạo biểu mẫu sử dụng công cụ Autoform thì Access không cho phép người sử dụng can thiệp vào quá trình tạo biểu mẫu, chẳng hạn như hạn chế số trường..... thì Form Wizard cho phép người sử dụng can thiệp vào quá trình tạo biểu mẫu.

Cách tạo

Trong cửa sổ Database chọn Form, chọn New

Trong mục *Choose the table or query Where the object's data comes from:*

Chọn bảng hoặc truy vấn làm nguồn dữ liệu cho form.

Chọn Form Wizard

Chọn OK

Trong mục Avaiable Field: Chọn các trường đưa vào biểu mẫu, nhấn nút >>

Chọn nút **Next**.

Chọn *Columnar* : Biểu mẫu hiển thị theo dạng cột

Tabular : Biểu mẫu hiển thị theo dạng hàng

Datasheet : Biểu mẫu hiển thị theo dạng bảng

Justified : Biểu mẫu hiển bình thường (đều).

Chọn **Next**

Chọn loại biểu mẫu

Chọn **Next**

Đặt tiêu đề cho Form

Chọn *Open the form to view or enter information*: Nếu muốn mở Form sau khi chọn Finish.

Chọn *Modify the form’s design:* Nếu muốn form ở dạng thiết kế.

Chọn **Finish** .

Lưu form.

3.3. Tạo biểu mẫu không sử dụng Wizard (Do người sử dụng tự thiết kế)

Tạo biểu mẫu sử dụng công cụ Autoform và Form wizard người sử dụng có thể nhanh chóng thiết kế các biểu mẫu nhờ vào các đặc tính hỗ trợ của Access. Nhưng đối với hai cách trên chỉ cung cấp một số hạn chế các phương án xây dựng biểu mẫu mà không thoã mãn yêu cầu của người sử dụng khi muốn thiết kế biểu mẫu teho ý của riêng mình. Do đó người sử dụng phải tự thiết kế một biểu mẫu không cần sự hỗ trợ của Access.

Cách tạo

Trong cửa sổ Database chọn Form, chọn New

Chọn bảng dữ liệu hoặc truy vấn làm nguồn dữ liệu cho form, chọn **OK**

Xây dựng các điều khiển cho biểu mẫu (Đưa các trường trong bảng dữ liệu vào biểu mẫu).

Thiết lập các thuộc tính cho các điều khiển.

Lưu biểu mẫu.

4. CÁC CHẾ ĐỘ HIỂN THỊ VÀ CÁC THÀNH PHẦN CỦA BIỂU MẪU

4.1. Các chế độ hiển thị

Có 4 chế độ hiển thị của biểu mẫu

4.1.1.Chế độ Design View

Dùng để tạo biểu mẫu mới hay thay đổi cấu trúc của một biểu mẫu đã tồn tại.

Khi đang ở trong cửa sổ Database: Chọn form /chọn tên form/ chọn Design (Có

thể click chuột phải rồi chọn Design).

Khi đang ở chế độ Form view: Chọn View/ Form Design

4.1.2. Chế độ Form View

Dùng để nhập, thay đổi và xem dữ liệu. Trong chế độ Form View người sử dụng có thể xem tất cả các trường của một bản ghi tại một thời điểm.

Khi đang ở trong cửa sổ Database: Chọn form /chọn tên form/ chọn Open (Có thể click chuột phải rồi chọn Open).

Khi đang ở chế độ Design view: Chọn View/ Form View.

4.1.3. Chế độ hiển thị biểu mẫu dưới dạng bảng (Datasheet View)

Dùng để nhập, thay đổi và xem dữ liệu trong biểu mẫu theo dạng bảng biểu.

Để mở chế độ hiển thị dạng Datasheet View chúng ta thực hiện như sau:

Khi đang ở chế độ Design View: Chọn View/Datasheet.

Khi đang ở chế độ Form View: Chọn View/ Datasheet View.

4.1.4. Chế độ hiển thị Print Preview

Dùng để xem biểu mẫu trước khi quyết định in ấn. Trong chế độ Print Preview sẽ duy trì hình dạng trình bày dữ liệu đã được thiết kế trước đó.

Khi đang ở trong cửa sổ Database: Chọn form /chọn tên form/ Chọn File/Print Preview.

4.2. Các thành phần biểu mẫu trong chế độ Design View

Khi muốn thiết kế biểu mẫu thì người sử dụng phải làm việc trong chế dộ Design View khi đó biểu mẫu có các thành phần chính sau:

Thước(Ruler): Điều chỉnh kích thước của các điều khiển.

Tiêu đề form (form header):Sử dụng để trình bày tiêu đề của form, tiêu đề form luôn được trình bày phần trên cùng, đầu tiên của biểu mẫu và trang in biểu mẫu.

Chân form (Form Footer): Sử dụng để trình bày chân của form, chân form luôn được trình bày phần dưới cùng, xuất hiện cuối biểu mẫu và trang in biểu mẫu.

Tiêu đề trang (Page header): Sử dụng để chứa tiêu đề trang

Chân trang (Page footer): Sử dụng để chứa chân trang nhưng xuất hiện phần trước của Form footer trong trang biểu mẫu in.

☞ ***Chú ý***

Page header và Page footer chỉ xuất hiện trong trang biểu mẫu in nên chúng không có những tính chất thông thường như Form header và Form footer.

Chọn View/ Page header/ footer (Nếu 2 thành phần này chưa xuất hiện trên biểu mẫu).

Chi tiết form (Detail): Đây là phần rất quan trọng chứa các điều khiển nhằm trình bày các dạng dữ liệu từ các bảng dữ liệu hoặc các truy vấn. Các loại điều khiển có thể là điều khiển buộc, không buộc hoặc tính toán.

5. CÁC LOẠI ĐIỀU KHIỂN

Tất cả thông tin trên biểu mẫu được chứa trong những đối tượng gọi là điều khiển (Control). Điều khiển có thể dùng để thể hiện dữ liệu, thực hiện các hành động hoặc thiết kế biểu mẫu đẹp mắt. Trong ACCESS hệ thống định nghĩa một số loại điều khiển như sau:

Điều khiển nhãn (Label).

Điều khiển hộp văn bản (Text box).

Điều khiển nhóm lựa chọn (Option group).

Điều khiển loại hộp Combo (Combo box) và hộp danh sách (List Box).

Ngoài ra còn có một số điều khiển khác như command button.....

Khi tạo lập điều khiển, chúng ta thuờng xác định hình thức dữ liệu trình bày trong chúng. Có những điều khiển lấy dữ liệu từu các trường trong bảng hay truy vấn, có điều khiển chỉ dùng vào mục đích trang trí, làm tiêu đề, có những điều khiển lấy dữ liệu từ một biểu thức nào đó. Vì vậy người ta phân ra thành ba nhóm điều khiển chính:

Điều khiển buộc (Bound control)

Điều khiển không buộc (Unbound control)

Điều khiển tính toán (Caculated control)

5.1. Điều khiển bị buộc (bound), không buộc (unbound) và tính toán được (calculated).

Khi tạo một điều khiển trong biểu mẫu thì phải xác định nó lấy dữ liệu từ nguồn nào để thể hiện.

Ví dụ

Tạo một điều khiển loại hộp văn bản (Text box) để hiển thị tên các mặt hàng, chúng ta phải chỉ định cho điều khiển lấy dữ liệu trong trường TEN_HANG của bảng MAT_HANG. Hộp văn bản này gọi là bị buộc.

Điều khiển cũng có thể thể hiện những thông tin không có trong CSDL (Không bị buộc).

Ví dụ: Tạo tiêu đề cho biểu mẫu...

Tóm lại

Điều khiển bị buộc (Bound Control) là điều khiển mà nguồn dữ liệu của nó lấy từ một trường trong bảng hoặc truy vấn Trong biểu mẫu dùng điều khiển buộc vào các trường để hiển thị nội dung hoặc cập nhật các trường của CSDL, các giá trị cập nhật có thể là: Văn bản, Date,Num ber, yes/No, Picture, chart trong đó dạng văn bản là phổ biến nhất.

Điều khiển không bị buộc (Unbound Control) là điều khiển không lấy dữ liệu từ một nguồn nào cả là điều khiển không bị buộc. Dùng điều khiển không buộc để trình bày thông tin không có trong các bảng hay rút được từ truy vấn.

Điều khiển tính toán (Calculated Control) là điều khiển mà nguồn dữ liệu của nó không phải là một trường mà là một biểu thức gọi là điều khiển tính toán (Calculated Control). Chúng ta qui định giá trị xuất hiện trong điều khiển bằng cách lập biểu thức cho nó. Biểu thức này là nguồn dữ liệu của điều khiển .Trong biểu thức có thể dùng các toán tử (+, -,=...) với tên điều khiển.

Ví dụ: Tạo một điều khiển THANHTIEN=SOLUONG*DONGIA

5.2. Tạo điều khiển loại hộp văn bản (text box)

Text box có thể là một điều khiển bị buộc, không buộc hoặc tính toán.

Tạo hộp văn bản bị buộc (Bound Text box).

Chúng ta buộc điều khiển Text box vào một trường bằng cách chỉ định điều khiển đó lấy dữ liệu trên trường nào. Chọn trường để buộc vào điều khiển bằng cách Click biểu

tượng Field List để mở danh sách các trường của bảng hay truy vấn làm nền tảng cho biểu mẫu. Theo mặc định như vậy thì hệ thống sẽ tạo một điều khiển loại Text box.

Một cách khác dùng hộp dụng cụ Toolbox để tạo điều khiển và sau đó gõ tên trường muốn buộc vào hộp văn bản.

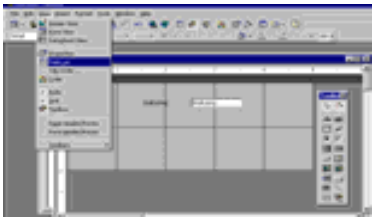
Mở hộp danh sách trường

Mở biểu mẫu trong chế độ Design (Tạo biểu mẫu trước).

Trong cửa sổ Database Chọn Form/New (Cọn bảng hoặc truy vấn làm nền cho biểu mẫu).

Chọn View/Field List hoặc click vào biểu tượng Field List trên thanh công cụ).

Tạo một Text Box bị buộc



Từ danh sách trường chọn một hoặc nhiều trường kéo và đưa vào biểu mẫu.

5.3. Tạo một điều khiển khác dùng hộp công cụ

Dùng hộp công cụ (Toolbox) để tạo những điều khiển không buộc (Unboud control) hoặc để tính toán. Đặc biệt dùng các tính năng của hộp này để tạo những điều khiển bị buộc khác ngoài buộc khác ngoài (Text box). Để bật hộp công cụ ta chọn View/Toolbars.

Tạo điều khiển dùng hộp công cụ

Click vào biểu tượng công cụ tuơng ứng với điều khiển muốn tạo.

Tạo điều khiển bị buộc bằng cách chọn một trường trong danh sách trường (Field List) và kéo vào biểu mẫu.

Hoặc tạo điều khiển không buộc hay dùng để tính toán bằng cách click vào một vị trí trên biểu mẫu.

Tạo điều khiển dùng để tính toán

Nếu muốn trình bày kết quả của một phép toán trong biểu mẫu, khai báo nguồn dữ liệu của điều khiển là một biểu thức. Sau này mỗi lần mở biểu mẫu, ACCESS tính toán lại kết quả của vùng dữ liệu được cập nhật mới nhất từ các bảng.

Tạo điều khiển nhãn (Label Control)

Muốn trình bày một chuỗi ký tự trên biểu mẫu như để làm tiêu đề... chúng ta dùng loại điều khiển gọi là điều khiển nhãn (label control). Nhãn không thể hiện dữ liệu của một trường hay biểu thức nào , chúng luôn luôn là không bị buộc.

Click vào biểu tượng Label trong Toolbox

Click vị trí muốn đặt nhãn trên biểu mẫu.

Thay đổi các thuộc tính của điều khiển.

Sau khi tạo biểu mẫu, biến đổi một số thuộc tính của điều khiển có thể hoàn thiện thêm về thiết kế và hình thức trình bày của số liệu.

Nhắp kép vào điều khiển.

Trong hộp lựa chọn trên đầu bảng thuộc tính, chọn một lớp thích hợp để làm việc.

All Properties: Trình bày tất cả các thuộc tính của điều khiển

Data Properties: Ấn định các đặc tính thể hiện dữ liệu trong điều khiển như giá trị mặc nhiên, định dạng số.

Even Properties: Qui định một tập lệnh (Macro) hay thủ tục (Procedure)..

Layout Properties: Định nghĩa các hình thức của điều khiển như cao, rộng.

Other Properties: Một số thuộc tính khác như tên điều khiển, thông tin mô tả ở dòng trạng thái.

Click chọn một trong các thuộc tính của bảng để thực hiện.

5.4. Thiết lập một số thuộc tính bổ sung

5.4.1. Các thuộc tính hỗ trợ nhập liệu

Thuộc tính **Default value:** Gán giá trị mặc định vào nội dung trình bày trong điều khiển.

Thuộc tính **ValidationRule và ValidationRule Text**: Kiểm tra tính hợp lệ khi nhập dữ liệu cho một điều khiển và thông báo lỗi nếu dữ liệu không hợp lệ

5.4.2. Các thuộc tính gióng lề

Thuộc tính **General:** Gióng hàng văn bản theo lề trái, dữ liệu số và ngày tháng theo lề phải.

Thuộc tính **Left:** Gióng hàng văn bản theo lề trái.

Thuộc tính **Center:** Xác lập hàng văn bản ở chính giữa.

Thuộc tính **Center:** Gióng hàng văn bản theo lề phải.

5.4.3. Các thuộc tính màu sắc

Thuộc tính **Backcolor**: Thiết lập màu nền cho điều khiển hay cho biểu mẫu

Thuộc tính **Forecolor**: Thiết lập màu cho hàng chữ trong điều khiển.

Thuộc tính **Bodercolor**: Thiết lập màu cho khung bao quanh trong điều khiển.

5.4.3. Các thuộc tính khung bao

Thuộc tính **BoderStyle**:Thiết lập loại khung cho điều khiển

Thuộc tính **BoderWidth**: Thiết lập độ dày hay đậm của khung bao.

Thuộc tính **BoderColor**: Thiết lập màu của khung bao.

6. NÂNG CẤP BIỂU MẪU

Access cung cấp nhiều loại điều khiển để làm cho biểu mẫu dễ sử dụng và có nhiều hiệu ứng tốt. Có thể thay thế một Textbox với một List box hay Combo box để chọn từ các giá trị có sẵn thay vì buộc người sử dụng phải nhớ để nhập giá trị vào.

6.1. Dùng điều khiển List box và Combo box để tạo danh sách chọn lựa.

Trong nhiều trường hợp, chọn một danh sách có sẵn thường tiện lợi hơn phải nhớ để gõ vào từ bàn phím. Access cung cấp hai khả năng điều khiển tạo danh sách chọn lựa:

List box và Combo box.

List box đơn giản là một danh sách để chọn, combo box tương tự như một text box và một combo box kết hợp vào một điều khiển, có nghĩa là có thể gõ thẳng giá trị vào text box hay chọn từ một danh sách có sẵn.

Ưu điểm của List box: Danh sách luôn được thể hiện và người dùng chỉ được phép chọn trong danh sách, do đó dữ liệu nhập luôn luôn là hợp lệ.

Ưu điểm của Combo box: Danh sách không được thể hiện cho đến khi người dùng mở hộp điều khiển, do đó ít tốn chỗ trên biểu mẫu hơn.

6.2. Tạo List box và Combo box không sử dụng Wizard

Tắt chức năng Control Wizard trong menu View hoặc trong thanh công cụ.

Click biểu tượng List box hoặc Combo box trong hộp công cụ.

Tạo điều khiển buộc bằng cách click biểu tượng Field list trên thanh công cụ để mở danh sách trường làm nền cho biểu mẫu. Chọn một trường trong danh sách kéo vào biểu mẫu, hoặc tạo điều khiển không buộc bằng cách click biểu mẫu nơi muốn đặt điều khiển.

Lập các thuộc tính của điều khiển để định nghĩa các hàng dùng làm chọn lựa trong danh sách.

Muốn danh sách thể hiện	**Lập thuộc tính Row Source Type thành**	**Lập Row Source thành**
Các hàng từ một bảng hay truy vấn	Table/Query (Default)	Tên của bảng hay truy vấn đó
Các hàng lấy từ lệnh Select của SQL	Table/Query (Default)	Một câu lệnh SQL
Một danh sách với các giá trị do người dùng tự đặt	Value List	Danh sách các giá trị đó phân cách nhau bởi dấu chấm phẩy
Tên các trường trong một bảng hoặc truy vấn	Field List	Tên của bảng hoặc truy vấn đó
Các giá trị định nghĩa bởi một hàm trong Access Basic	Tên hàm xây dựng	Để trống

6.2.1. Định nghĩa các hàng trong danh sách

Định nghĩa nguồn dữ liệu để hình thành các hàng dùng làm trong danh sách lựa chọn trong List box và Combo box bằng cách lập thuộc tính thích hợp cho RowSource Type và RowSource của điều khiển, hai thuộc tính này phối hợp với nhau để tạo nên các hàng trong danh sách.

Thuộc tính RowSource Type cho biết danh sách sẽ hình thành từ nguồn dữ liệu loại nào, thuộc tính RowSource là thuộc tính xác định nguồn dữ liệu của danh sách.

6.2.2. Danh sách lấy dữ liệu từ bảng hay truy vấn

Có thể tạo List box hay Combo box lấy dữ liệu từ một hoặc nhiều trường trong bảng (hoặc truy vấn) để lập thành danh sách chọn lựa cho điều khiển. Thực hiện điều này bằng cách đặt thuộc tính RowSource Type thành Table/Query và RowSource thành tên bảng hay tên truy vấn có chứa dữ liệu để lập thành danh sách đó.

6.2.3. Tạo danh sách nhiều cột

Có thể tạo danh sách gồm hai hay nhiều cột để có thêm thông tin lựa chọn. Xác định số cột bằng cách lập thuộc tính Columcount và ColumnWidth cho điều khiển. Thuộc tính columcount quy định số cột và nếu trong RowSource là tên một bảng hay truy vấn thì các trường đầu tiên tương ứng số cột của bảng hay truy vấn đó (tính từ trái sang phải) sẽ được đưa vào danh sách. Thuộc tính columnwidth quy định bề rộng mỗi cột tính theo inch hay cm.

6.2.4. Sử dụng danh sách giá trị

Trường hợp điều khiển chỉ có một số nhỏ chọn lựa và các giá trị này không thay đổi, có thể dùng phương pháp đơn giản là danh sách giá trị. Danh sách này chúng ta tự lập bằng cách đưa các giá trị dùng làm chọn lựa vào thuộc tính RowSource của điều khiển. Sử dụng dấu chấm phẩy để phân cách các mục chọn lựa của danh sách.

6.3. Sử dụng hộp kiểm tra (check box), nút chọn lựa (Option button), nút bật tắt (Toggle button).

Hộp kiểm tra, nút chọn lựa, nút bật tắt thường được sử dụng như các điều khiển độc lập để nhận các chọn lựa Yes/No. Các điều khiển này thực chất chỉ khác nhau về hình thức, do đó chúng ta có thể sử dụng bất cứ nút nào.

Khi được chọn điều khiển này biểu diễn trị Yes (hay True), ở trạng thái không chọn thì nhận giá nị No (hay False).

Check box : Được chọn khi có dấu **x** trong hộp.

Option button : Được chọn khi có hạt đậu trong nút.

Toggle button : Được chọn khi nó trông như bị nhấn xuống.

6.4. Sử dụng nhóm chọn lựa (Option group)

Chúng ta có thể dùng nhóm chọn lựa để trình bày một số nhỏ các chọn lựa trên biểu mẫu. Nhóm chọn lựa gồm một khung nhóm (Group frame) có gắn nhãn và một số check box, option button hay toggle button.

Nếu nhóm lựa chọn được buộc vào một trường, chỉ có khung nhóm là bị buộc vào trường đó, không phải các thành phần để chọn lựa trong khung. Khung nhóm có thể không bị buộc vào trường nào hoặc chứa một biểu thức, mỗi lần chỉ có thể chọn một thành phần trong nhóm.

Khi sử dụng hộp kiểm tra, nút chọn lựa và nút bật tắt trong nhóm chọn lựa, thuộc tính của chúng khác với như khi được dùng với các điều khiển độc lập. Mỗi phần tử trong nhóm chọn lựa không có thuộc tính Control Source mà thay bằng thuộc tính Option Value. Lập thuộc tính Option Value của mỗi phần tử dùng làm chọn lựa trong nhóm thành một trị số có nghĩa đối với trường mà khung nhóm được buộc vào. Khi một phần tử của nhóm được chọn, Access đưa giá trị lập trong Option value của phần tử đó cho điều khiển Option Group. Đến lượt Option Group trả lại giá trị đó cho trường mà nó bị buộc vào.

7. BIỂU MẪU DỰA TRÊN NHIỀU BẢNG

7.1. Biểu mẫu phụ (Subform)

Biểu mẫu phụ là phương pháp để đưa thông tin từ nhiều bảng vào một biểu mẫu. Biểu mẫu phụ có nghĩa là một biểu mẫu được lồng trong biểu mẫu khác. Trong Access biểu mẫu chính gọi là Main form, biểu mẫu nằm trong Main form gọi là biểu mẫu phụ (Sub form). Khi dùng biểu mẫu phụ chúng ta dễ nhận thấy mối quan hệ giữa các bản ghi của hai hay nhiều bảng.

Biểu mẫu phụ đặc biệt hữu hiệu khi dùng để hiển thị dữ liệu từ nhiều bảng hay truy vấn có quan hệ một-nhiều với nhau. Biểu mẫu chính đại diễn cho bên một, biểu mẫu phụ đại diện cho bên nhiều.

7.2. Các loại biểu mẫu phụ

Khi tạo biểu mẫu phụ chúng ta có thể thiết kế nó thành dạng bảng, hoặc dạng biểu mẫu, hoặc cả hai dạng trên.

Biểu mẫu dạng bảng: Là loại dễ tạo nhất và có thể sử dụng như bất kỳ bảng nào khác như sắp xếp....

Biểu mẫu phụ dạng biểu mẫu: Cho chúng ta thực sự linh hoạt và mềm dẽo khi thiết kế hơn.

7.3. Thiết kế biểu mẫu phụ

Thông thường chúng ta dùng bảng hay truy vấn làm nguồn dữ liệu cho biểu mẫu chính, một bảng hay truy vấn khác làm nguồn dữ liệu cho biểu mẫu phụ. Nếu dữ liệu trong biểu mẫu chính và biểu mẫu phụ có liên quan với nhau, chúng ta cần đánh giá một số vấn đề sau:

Các bảng hoặc truy vấn có quan hệ một-nhiều với nhau không? Nếu dùng biểu mẫu phụ để thể hiện quan hệ một-nhiều, chúng ta nên dùng bảng bên một đối với bảng chính, bảng bên nhiều đối với bảng phụ.

Các bảng hoặc truy vấn làm nguồn dữ liệu cho biểu mẫu chính/phụ có các trường liên kết không? Access dùng trường kết nối để giới hạn số lượng bản ghi thể hiện trong biểu mẫu phụ.

Cách tạo biểu mẫu chính/phụ

Thiết kế hai biểu mẫu riêng biệt, sau đó kéo biểu mẫu phụ vào biểu mẫu chính.

Thiết kế biểu mẫu chính

Tạo biểu mẫu chính, dành chỗ trên biểu mẫu này để chứa biểu mẫu phụ.

Lưu và đóng biểu mẫu chính.

Thiết kế biểu mẫu phụ

Có thể thiết kế biểu mẫu phụ để chỉ thể hiện dữ liệu dưới dạng bảng, biểu mẫu này có cả hai khả năng trên.

Tạo biểu mẫu mới, lập hai thuộc tính ViewAllowed và Default View của biểu mẫu tuỳ theo yêu cầu sử dụng như sau:

Biểu mẫu phụ chỉ trình bày dưới dạng bảng: Đặt các trường trên biểu mẫu theo thứ tự chúng ta muốn chúng xuất hiện trong bảng. Lập cả hai thuộc tính ViewAllowed và Default View thành Datasheet.

Biểu mẫu phụ chỉ trình bày dữ liệu dưới dạng biểu mẫu: Sắp đặt các điều khiển như trên. Lập thuộc tính ViewAllowed thành Form và Default View thành Single form hay Continuous form.

Đưa biểu mẫu phụ vào biểu mẫu chính

Mở biểu mẫu chính ở chế độ Design View

Chuyển sang cửa sổ Database, nhần F11

Kéo biểu mẫu phụ tử cửa sổ Database và đặt vào một vị trí trên biểu mẫu chính.

Di chuyển biểu mẫu phụ đến vị trí khác, thay đổi nội dung nhãn hoặc kích thước nếu cần. Chuyển sang chế độ Form View để xem kết quả.

7.4. Liên kết biểu mẫu chính và biểu mẫu phụ

Trong chế độ Design View của biểu mẫu chính, mở bảng thuộc tính của điều khiển biểu mẫu phụ.

Lập thuộc tính LinkChildFields thành tên trường nối kết trong biểu mẫu phụ, nếu có nhiều trường nối kết, phân cách nhau bởi dấu phẩy.

Lập thuộc tính LinkMasterFields thành tên trường nối kết hoặc tên điều khiển trong biểu mẫu chính, nếu có nhiều trường nối kết, phân cách nhau bởi dấu phẩy.

Chương 6 BÁO CÁO (REPORT)

1. TỔNG QUAN VỀ BÁO CÁO

Báo cáo là phương thức hữu hiệu giúp người sử dụng trình bày dữ liệu dưới dạng đầy đủ và dễ hiểu, nhanh chóng, đẹp mắt để khi in ấn. Người sử dụng có thể tích hợp trong báo cáo các dạng thức trình bày dữ liệu khác nhau như: Hình ảnh, biểu đồ, văn bản.....

Báo cáo được xây dựng trên một nguồn dữ liệu đó là bảng hoặc truy vấn, một câu lệnh SQL hoặc một dạng biểu mẫu nào đó

1.1. Các dạng mẫu của báo cáo

Báo cáo dạng cột (columnar): báo cáo dạng này sẽ được trình bày theo dạng một cột và kèm theo phần nhãn của mỗi cột dữ liệu bên trái, mỗi dòng tương ứng với một trường dữ liệu.

Báo cáo dạng hàng (Tabular): Báo cáo sẽ trình bày dữ liệu theo dạng bảng bao gồm nhiều hàng và nhiêu cột.

Báo cáo dạng nhóm/ Tổng (Group/Total): Báo cáo dạng này sẽ tổ chức dữ liệu thành các nhóm, mỗi nhóm sẽ trình bày dữ liệu theo dạng Tabular. Người sử dụng có thể nhóm dữ liệu theo cấp và có thể tính toán giá trị tổng cho mỗi nhóm và một giá trị tính tổng cho toàn bộ các nhóm

Báo cáo dạng biểu đồ (Chart)

Báo cáo dạng nhãn (Label Report)

Báo cáo với báo cáo con

1.2. Các chế độ hiển thị của báo cáo

Báo cáo có thể được trình bày theo 3 chế độ sau

Report design: Chế độ thiết kế báo cáo.

Layout PreView: Chế độ trình bày dữ liệu trong báo cáo.

Print PreView: Chế độ xem hình thức báo cáo trước khi in ấn.

2. TẠO BÁO CÁO SỬ DỤNG CÔNG CỤ AUTO REPORT VÀ REPORT WIZARD

2.1.Tạo báo cáo sử dụng Auto report

Click biểu tượng Report trong cửa sổ database (hoặc chọn View/Report)

Chọn New

Chọn bảng hoặc Truy vấn làm nguồn dữ liệu báo cáo .

Chọn AutoReport Columnar: Nếu muốn báo cáo hiển thị dạng cột.

AutoReport Tabular: Nếu muốn báo cáo hiển thị dạng hàng

Chọn OK

Lưu Báo cáo.

2.2.Tạo báo cáo sử dụng Report Wizard

Click biểu tượng Report trong cửa sổ database (hoặc chọn View/Report)

Chọn New

Chọn bảng hoặc Truy vấn làm nguồn dữ liệu báo cáo .

Chọn Report Wizard

Chọn OK

Chọn các trường cần thiết cho báo cáo.

Chọn **Next**

Chọn các trường cần nhóm, chọn **Next**

Chọn các trường cần sắp xếp, chọn **Next**

Chọn dạng thể hiện của Report, chọn **Next**

Chọn nền thể hiện của Report, chọn **Next**

Đặt tiêu đề cho Report, chọn **Finish**

3. TẠO ĐIỀU KHIỂN TRONG BÁO CÁO

3.1. Tạođiều khiển Text Box (Hộp văn bản)

Text box có nhiều tính năng, có thể là điều khiển bị buộc, không buộc hay dùng tính toán. Nếu trong báo cáo có cả 3 loại điều khiển này, chúng ta nên ưu tiên điều khiển bị buộc trước.

Tạo một text box bị buộc và không buộc

Chúng ta buộc điều khiển Text box vào một trường bằng cách chỉ định trường cho text box đó lấy dữ liệu. Text box đó lấy dữ liệu có thể thực hiện bằng cách kéo trường muốn buộc vào điều khiển từ danh sách trường (Field List) vào biểu mẫu đang thiết kế.

Cách khác để tạo Text box là dùng hộp công cụ (Toolbox), sau đó gõ tên trường muốn buộc vào hộp văn bản hoặc bảng thuộc tính của điều khiển.

Dùng danh sách trường là phương pháp tốt nhất để tạo một điều khiển Text box bị buộc vì hai lý do sau:

Điều khiển đó được hệ thống tự động gắn nhãn và nhãn lấy tên trường được kéo làm tiêu đề.

Text box bị buộc đó thừa kế các thiết lập thuộc tính của trường từ bảng hay truy vấn.

Muốn chuyển một điều khiển không buộc thành bị buộc, lập thuộc tính Control Source của điều khiển thành một trường.

Một số thao tác khi thực hiện thiết kế báo cáo.

Mở báo cáo trong chế độ Design View

Chọn Field List từ menu View (Hoặc click biểu tượng Field List trên thanh công cụ)

Tạo điều khiển Text box bị buộc.

Mở bảng danh sách trường, chọn trường hoặc các trường muốn đặt vào báo cáo

Chọn một trường, click vào trường đó.

Chọn nhiều trường liền nhau, clik trường đầu, giữ phím shift, click trường cuối.

Chọn nhiều trường không liền nhau, giữ phím Ctrl và lần lượt click từng trường.

Chọn tất cả các trường trong danh sách, nhấp kép vào thanh tiêu đề của danh sách trường.

Kéo trường (hoặc các trường) được chọn và đặt vào một vị trí trên mẫu báo cáo.

Click biểu tượng Simple preview trên thanh công cụ để xem kết quả.

3.2. Tạo các điều khiển khác dùng hộp công cụ

Muốn tạo các điều khiển không buộc hay dùng để tính toán, phải dùng công cụ trong Toolbox.

Chọn View/Toolbars để hiển thị thanh công cụ

Tạo điều khiển dùng Toolbox

Click công cụ tương ứng loại điều khiển muốn tạo trong báo cáo.

Tạo điều khiển bị buộc bằng cách chọn một trường trong Field list và kéo nó vào trong báo cáo.

3.3. Tạo điều khiển dùng tính toán

Click vào biểu tượng Text box trong hộp công cụ

Click vào một vị trí trên báo cáo. Access tự động gắn nhãn cho điều khiển vừa tạo, tiêu đề mặc nhiên thường có dạng “Field0”, có thể thay đổi tiêu đề này theo ý thích.

Đưa con trỏ vào bên trong Text box.

Gõ dấu = , theo sau là biểu thức muốn lập

Ví dụ: =[SOLUONG]*[DONGIA]

Click vào Sample Preview để xem kết quả.

3.4. Tạo điều khiển nhãn

Nhãn là một điều khiển không buộc, nội dung nhãn không thay đổi từ trang này qua trang khác hay từ bản ghi này qua bản ghi khác.

Click biểu tượng Label trong Toolbox.

Click vào một vị trí trên báo cáo để tạo nhãn. Nhãn được tạo như vậy sẽ có kích thước tự mở rộng khi gõ nội dung vào.

Nếu muốn trình bày văn bản thành nhiều dòng, bấm Ctrl+Enter cuối dòng thứ nhất.

4. CÁC THUỘC TÍNH CỦA ĐIỀU KHIỂN TRONG BÁO CÁO

Thuộc tính xác định các đặc trưng của đối tượng, mỗi điều khiển trong báo cáo cũng có những thuộc tính riêng. Muốn mở bảng thuộc tính của điều khiển, chọn điều khiển đó và click biểu tượng Properties trên thanh công cụ.

Ta xét một số thuộc tính sau:

Cangrow: Dùng thuộc tính này để làm cho Text box có thể tự điều chỉnh kích thước theo phương dọc đối với khối dữ liệu chứa trong trường nó bị buộc.(chọn Yes).

CanShrink: Khi Text box khôngcó dữ liệu hoặc dữ liệu là chuỗi rỗng Access sẽ chừa trống chỗ đó trên giấy. Điều này có thể làm cho báo cáo quá trống trải nếu có nhiều chỗ như vây. Chúng ta lập thuộc tính của Text box này thành Yes.

HideDuplicate: Dùng thuộc tính này để che Text box khi giá trị trong đó trùng bản ghi trước

5. SẮP XẾP VÀ TẬP HỢP DỮ LIỆU THEO NHÓM

Sắp xếp là phương pháp phổ biến nhằm tổ chức dữ liệu theo một trật tự nào đó để tìm kiếm và phân loại thông tin.

5.1. Sắp xếp dữ liệu

Khi in báo cáo người dùng thường muốn tổ chức các bản ghi theo một trật tự nào đó. Ví dụ in danh sách cán bộ theo thứ tự giảm dần của lương.

Các bước thực hiện sắp xếp trên báo cáo.

Mở báo cáo ở chế độ Design View.

Chọn Sorting And Grouping trong menu View.

Trong hộp thoại.

Field/Expression: Chỉ định sắp xếp theo trường hoặc biểu thức nào đó.

Sort Order: Chọn Tăng dần hoặc giảm dần.

5.2. Nhóm dữ liệu

Trong nhiều báo cáo, sắp xếp các bản ghi không cũng chưa đủ mà cần phân thành các nhóm. Nhóm là tập hợp các bản ghi cùng với thông tin tóm lược tiêu biểu cho một thể loại thông tin. Một nhóm thường được cấu tạo như sau:

Tiêu đề nhóm (group header), nhóm con (nếu có), các bản ghi chi tiết và chân nhóm (Group footer)

Tiêu đề nhóm 1

Tiêu đề nhóm 2

Tiêu đề nhóm 3

.....

.....

Tiêu đề nhóm 10

Các bản ghi chi tiết

Chân nhóm 10

.....

.....

Chân nhóm 3

Chân nhóm 2

Chân nhóm 1

Ch□□ng 7 MACRRO

1. KHÁI NIỆM

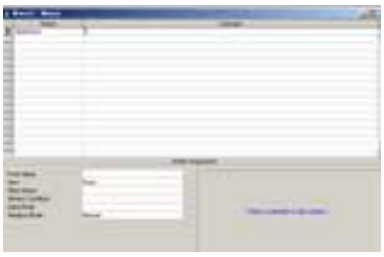
Macro trng MS Access là tập hợp các lệnh (Hành động, hành động.....) được định sẵn nhằm tự động thực hiện chuỗi các tác vụ nào đó mà không cần sự can thiệp từng bước của người sử dụng. Macro có thể liên kết các đối tương trong tập tin cơ ở dữ liệu (CSDL) như: Table, Query, form, report..... nhằm tạo ra các ứng dụng để khai thác có hiệu quả..

Macro được dùng khi có các hành động nào thường xuyên lặp lại trong MS Access hoặc được dùng khi cần kết hợp các hành động đơn giản nhằm giải quyết một vấn đề nào đó khi xây dựng các ứng dụng. Việc tự động hoá các hành động này bởi macro sẽ được thực hiện một cách nhanh chóng và chính xác.

2.TẠO VÀ THI HÀNH MỘT MACRO

2.1.Tạo một macro

Tại cửa sổ database, chọn đối tượng Macro/ New



Trong mục Action : chọn các hành động cho Macro.

Tong mục Action arguments: Chọn các tham số tương ứng cho hành động.

Lưu trữ Macro: File/save.

2.2.Thi hành macro

Tại cửa sổ database: Chọn đối tượng Macro/Run.

3. CÁC HÀNH ĐỘNG VÀ CÁC THAM SỐ

Ms Acces cung cấp một số hành động để tạo macro, mỗi hành động thực hiện một tác vụ nào đó trên CSDL và tuỳ thuộc vào tham số của mỗi hành động.

Open table: Mở 1 bảng dữ liệu

Table name: Tên bảng cần mở

View: Chọn dạng thể hiện của bảng (Datasheet/ Design/ Print Preview

Data mode:

Add: Dùng để nhập dữ liệu

Edit: Dùng để thêm, xoá, sửa dữ liệu trong bảng

Read Only: Mở bảng để chỉ xem dữ liệu

Open Query: Mở 1 truy vấn

Query name: Tên truy vấn cần mở

View: Chọn dạng thể hiện của bảng (Datasheet/ Design/ Print Preview

Data mode:

Add: Dùng để nhập dữ liệu

Edit: Dùng để thêm, xoá, sửa dữ liệu trong bảng

Read Only: Mở bảng để chỉ xem dữ liệu

Open Form: Mở 1 biểu mẫu

Form name: Tên biểu mẫu

View: Chọn dạng thể hiện (Form/ Design/ Print preview/ Datasheet)

Filter name: Tên Query lọc các dữ liệu để hiển thị trong form.

Where condition: Điều kiện lọc dữ liệu hiển thị trong form.

Data mode:

Add : Dùng để nhập dữ liệu

Edit : Dùng để thêm, xoá, sửa dữ liệu trong bảng

Read Only : Mở bảng để chỉ xem dữ liệu

Window mode:

Normal : Dạng cửa sổ form bình thường.

Hidden : Dạng cửa sổ form được ẩn đi.

Icon : Cửa sổ form thu nhỏ thành 1 biểu tượng.

Dialog : Dạng hộp thoại.

Open Report: Mở 1 báo cáo

Report name: Tên báo cáo

View: Chọn kiểu in

Print preview:In ra màn hình.

Design: Dạng thiết kế báo cáo

Print: In ra máy in

Filter name: Tên Query lọc các dữ liệu để hiển thị trong Report.

Where condition: Điều kiện lọc dữ liệu hiển thị trong Report.

Run macro: Thực hiện một tập lệnh

Macro name: Tên macro cần thực hiện

Repeat count: Số lần thực hiện macro sẽ lặp lại.

Repeat Expression: Biểu thức điều kiện để lặp lại khi thực hiện macro. Macro chỉ dừng khi khi biểu thức điều kiện nhận giá trị False.

Open module: Mở cửa sổ soạn thảo thủ tục trong 1 module.

Module name: Tên module chứa thủ tục cần mở.

Procedure name: Tên thủ tục sẽ mở.

Run code: Gọi thực hiện một hàm của Access Basic

Function name: Tên hàm cần thực hiện và các đối số của hàm.

Run App: Cho thực hiện một ứng dụng nào đó trong môi trường Windows

Command line: đường dẫn đến tập tin của một ứng dụng.

Run SQL: Cho thực hiện câu lệnh SQL

SQL Statement: Nội dung câu lệnh SQL

Maximize: Cực đại cửa sổ hiện thời

Minimize:Cực tiểu cửa sổ hiện thời thành một biểu tượng.

Restore: Phục hồi cửa sổ trở về kích thước cũ.

Move size: Di chuyển hoặc thay đổi kích thước cửa sổ hiện thời.

Right: Khoảng cách từ góc trên trái của cửa sổ này đến cạnh trái của cửa sổ chứa nó.

Down: Khoảng cách từ góc trên trái của cửa sổ này đến đến cạnh trên của cửa sổ chứa nó.

Width: Chiều rộng của cửa sổ này.

Height: Chiều cao của cửa sổ này.

Stop Macro: Dừng macro đang thực hiện

Beep: Phát tiếng kêu bíp

Hourglass: Đổi dạng con trỏ thành đồng hồ cát trong khi macro đang chạy

Hourglass On: Yes/No (Đổi/ Không đổi)

Close: Đóng một cửa sổ đang hoạt động

Object Type: Loại cửa sổ của đối tượng cần đóng như Table, Query, form, Report, Macro hoặc Module.

Object name: Tên của đối tượng cần đóng.

Quit: Thoát khỏi MS Access và trở về Windows

Option

Prompt: Hiển thị hộp thoại có lưu trữ không? Nếu đối tượng có thay đổi.

Save all: Lưu trữ tất cả mọi đối tượng.

Exit: Thoát mà không cần lưu trữ.

Print: In đối tượng hiện thời

Print Range: Phạm vi cần in ấn.

All: In tất cả các đối tượng

Selection: In phần trang được chọn

Pages: In các trang được chọn

Page from: Trang bắt đầu in

Page to: Trang kết thúc in

Print Quality: Chất lượng in

Copies: Số bản cần in

Collate Copies: Có sắp xếp thứ tự các bản in teo trang.

Msg Box: Hiển thị hộp thông báo

Message: Câu thông báo cần hiển thị

Beep: Yes/ No: Có/ Không phát ra tiếng Bíp khi hiển thị hộp thông báo.

Type: Loại hộp thông báo.

Title: Tiêu đề của hộp thông báo.

CancelEvent: Huỷ bỏ một sự kiện đang thực hiện

Requery: Cập nhật dữ liệu cho một đối tượng đang hoạt động bằng cách cập nhật lại dữ liệu nguồn của đối tượng đó.

Control name: Tên của đối tượng cần cập nhật dữ liệu (Nếu không chỉ ra thì sẽ cập nhật lại dữ liệu nguồn của chính đối tượng đang hoạt động).

Select Object: Chọn đối tượng trong CSDL

Object Type: loại đối tượng cần chọn.

Object name: Tên đối tượng cần chọn

In Database Window: (Yes/No) Xác định MS access có chọn đối tượng trong cửa sổ CSDL không, mặc định là No.

Set value: Gán một giá trị cho 1 trường, 1 điều khiển, hoặc một thuôc tính trên một Form hoặc 1 Report.

*Item:*Tên trường, đối tượng hay thuộc tính muốn gán giá trị.

Expression: Biểu thức cần gán giá trị cho Item.

☛ ***Chú ý:*** Nếu tên trường, tên đối tượng, tên thuộc tính ở 1 Form hoặc 1 Report khác thì phải mô tả đầy đủ.

Trong Form khác: [Forms]![Tên Form]![Tên trường/Tên đối tượng]

Trong Report khác: [Reports]![Tên Report]![Tên trường/Tên đối tượng]

Đối với các thuộc tính

[Forms/Reports]![Tên Form/Tên Report]![Tên trường].[Tên thuộc tính]

Add menu: Tạo thêm một Drop Down Menu vào một menu bar cho một form hoặc Report.

Menu name: Tên của Drop Down Menu muốn thêm vào menu bar.

Menu macro name: Tên macro chứa các lệnh về việc tạo menu.

Status bar: Thông báo ở thanh trạng thái khi chọn menu này.

Apply Filter: Lọc (Truy vấn) các dữ liệu khi xử lý Table, Form, Report.

Filter name: Tên của truy vấn lọc dữ liệu.

Where condition: Điều kiện lọc dữ liệu.

FindRecord: Tìm bản ghi đầu tiên nằm trong phạm vi và thoả mãn điêu kiện.

Find What: Nội dung dữ liệu cần tìm là một giá trị hoặc một biểu thức, nếu là biểu thức sẽ bắt đầu dấu "=".

Where: Qui định cách so sánh giá trị cần tìm với giá trị của trường.

Any part of field: Một phần bất kỳ của trường.

Match Whole field: Giá trị cần tìm bằng giá trị của trường.

Start of field: Giá trị cần tìm là phần đầu của trường.

Match Case: Yes/No: Có/Không phân biệt chữ in hoa và chữ in thường.

Direction: Quy định hướng tìm

All: Tìm toàn bộ

Up: Tìm từ bản ghi hiện thời lên phía trên.

Down: Tìm từ bản ghi hiện thời lên phía dưới.

Search As Formatted: Qui định việc tìm có dựa trên dữ liệu sau khi đã định dạng trong các trường hay không?

Search in: Qui định việc tìm trên trường hiện thời hoặc trên tất cả các trường.

Current Field: Tìm trên trường hiện thời.

All Fields: Tìm tất cả các trường.

Find First: Yes/No: Qui định tìm từ bản ghi đầu tiên hay tìm từ bản ghi hiện thời.

Findnext:Tìm bản ghi kế tiếp thoả mãn điều kiện tìm kiếm của lệnh FindRecord.

CopyObject: Sao chép một đối tượng trong tập tin CSDL hiện thời thành một đối tượng khác của tập tin CSDL khác trong MS Access.

Destionation Database: Tên tập tin CSDL đích

New name: Tên mới của đối tượng sau khi sao chép.

Source Object Type: Kiểu của đối tượng nguồn.

Source Object Name: Tên của đối tượng nguồn.

DeleteObject: Xoá một đối tượng trong tập tin CSDL hiện thời.

Object Type: Kiểu của đối tượng.

Object Name: Tên của đối tượng

4. NHÓM TẬP LỆNH VÀ TẬP LỆNH CÓ ĐIỀU KIỆN

4.1. Nhóm tập lệnh

Là Macro chứa các macro con, thay vì tạo ra nhiều macro với nhiều tên khác nhau thì các macro này được gom lại thành một tên chung nhằm giảm bớt số lượng và thuận lợi trong quá trình sử dụng. Tuy nhiên các Macro được nhóm khi chúng có liên quan với nhau.

Có thể có nhiều Macro trong nhóm có cùng hành động, tuy nhiên chúng được phân biệt bởi tên Macro.

Đặt tên cho Macro ta thực hiện:

Tại chế độ thiết kế Macro:

View/Macro name

Đặt tên cho Macro tại cột Macro name.

Cách thực hiện 1 macro trong macro name

<Tên Macro Group>.<Tên Macso cần thực hiện>

4.2. Macro có điều kiện

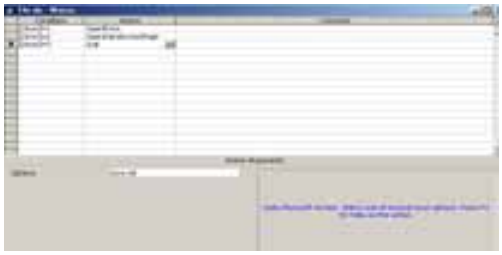
Là macro có chứa các điều kiện thi hành cho mỗi hành động.

Cách tạo Macro có điều kiện

Tại chế độ thiết kế Macro: Chọn View/Conditions

Tại cột Condition : Đặt điều kiện thi hành cho mỗi hành động.

Ví dụ



4.3. Áp dụng Macro cho form và Report

4.3.1. Quy tắc chung khi gọi một đối tượng

Đối với form : *Forms![Tên form]![Tên đối tượng]*

Đối với Report : *Reports![Tên Report]![Tên đối tượng]*

4.3.2. Các thuộc tính của một đối tượng

Muốn gắn một nút lệnh trên một biểu mẫu hoặc báo cáo với một Macro nào đó vào nút lệnh này ta thực hiện: Click chuột phải vào nút lệnh, chọn Properties và gắn Macro vào các hành động tương ứng.

On Enter: Macro thi hành khi nhấn Enter vào bên trong đối tượng

On Exit: Macro thi hành khi thoát khỏis đối tượng

On Got Focus: Thiết lập nhận biết khi có di chuyển con trỏ đến một form hoặc 1 trường trên form đang mở.

On Click: Macro thi hành khi click vào đối tượng

On Dbl Click: Macro thi hành khi Double click vào đối tượng.

On Mouse Down: Macro thi hành khi ấn và giữ chuột tại đối tượng

On Mouse Move: Macro thi hành khi di chuyển chuột ra khỏi đối tượng

On Mouse Up: Macro thi hành khi nhã chuột ra khỏi đối tượng.

On Key Down: Macro thi hành khi ấn và giữ một phím đối tượng.

On Key Press: Macro thi hành khi ấn một phím đối tượng.

On Key Up: Macro thi hành khi nhã một phím đối tượng.

4.3.3. Macro tự động thực hiện sau khi mở tập tin CSDL

Chúng ta có thể tạo ra một Macro mà mỗi khi mở một tập tin CSDL thì Macro này tự động thực hiện.

Để tạo Macro tự động thực hiện ta tiến hành các thao tác sau:

Tạo macro

Lưu trữ Macro với tên AutoExec.

5. THIẾT KẾ MENU TRONG ACCESS

Trong các ứng dụng, thường chúng ta phải tổ chức Menu để cho phép người sử dụng thực hiện các hành động thông qua các chức năng trên Menu này.

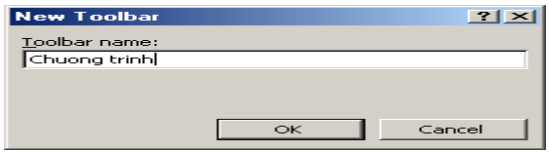
Có 2 cách hiển thị các Menu của người sử dụng (Custom Menu) trong các ứng dụng đó là:

Menu ứng với 1 form xác định nào đó: Menu này chỉ xuất hiện khi truy xuất đến Form này.

Menu toàn cục (Globally): Menu toàn cục này sẽ xuất hiện trong các ứng dụng và nó chỉ được thay thế khi có 1 menu ứng với form nào đó được mở.

5.1. Tạo menu của người sử dụng

Chọn View/Toolbars/Customize/New/ Đặt tên cho thanh menu

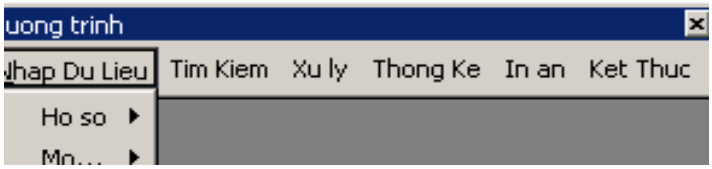


Chọn Properties và chọn Type là Menu bar, close

Chọn command, trọng mục Categories chọn New menu và kéo sang thanh menu vừa tạo.

Click chuột phải để thay đổi các tiêu đề cho phù hợp trên thanh menu.

Tương tự cho các nhóm khác.



Có thể tạo các chức năng là các hành động như mở bảng, truy vấn, biểu mẫu.... bằng cách kéo các biểu tượng này trong mục command vào menu đang tạo.

5.2. Tạo menu toàn cục

Nếu muốn tạo một menu toàn cục sẽ thay thế menu có sẵn trong Access khi CSDL này được mở ta thực hiện các bước sau:

Mở hoặc tạo ra một Macro có tên là Autoexec.

Thêm vào hành động: SetValue

Trong mục các tham số (arguments):

Item: Application.menubar

Expression: Tên của Menubar người sử dụng tạo ra và đặt giữa cặp dấu ngoặc kép "".

BÀI SỐ 1

Khởi động MSACCESS, tạo một CSDL có tên NHANSU.MDB rồi lần lượt tạo các bảng dữ liệu sau

Bảng 1: Nhanvien

STT	Fieldname	Data Type	Fieldsize	Note
1	**Manv**	Text	4	Khoá chính
2	Holot	Text	20	
3	Ten	Text	10	
4	Ngaysinh	Date/time	8	
5	Gioitinh	Yes/No	1	**Yes:Nam, No:Nu**
6	Maphong	Text (Lookup)	2	Lấy từ bảng Dsphong
7	MaPXuong	Text (Lookup)	2	Lấy từ bảng dspxuong
8	Diachi	Text	30	
9	Ghichu	Memo		

Bảng 2: Dsphong

STT	Fieldname	Data Type	Fieldsize	Note
1	**Maphong**	Text	2	Khoá chính
2	Tenphong	Text	30	
3	Sodthoai	Text	11	

Bảng 3: Thunhap_NV

STT	Fieldname	Data Type	Fieldsize	Note
1	**Manv**	Text (Lookup)	4	Khoá chính
2	Luongchinh	Number	Double	
3	Heso	Number	Integer	
4	Phucap	Number	Double	
5	Thue	Number	Double	Thuế
6	Thamnien	Number	Byte	Số năm thâm niên
7	Tongluong	Number	Double	

Bảng 4: DSpxuong

STT	Fieldname	Data Type	Fieldsize	Note
1	**Mapxuong**	Text	2	Khoá chính
2	Tenpxuong	Text	20	
3	Sodoanvien	Number	Byte	Số đoàn viên

1. Hãy nhập dữ liệu thích hợp cho các bảng trên.

2. Mở bảng Nhanvien sắp xếp tăng dần theo **manv**, sắp xếp giảm dần theo **ten**.

3. Sắp xếp tăng dần theo **ten**, nếu **ten** trùng nhau thì sắp xếp giảm dần theo **holot** nếu **holot** trùng nhau thì sắp xếp giảm dần theo **maphong**.

4. Lọc và hiển thị danh sách những nhân viên có tên **Thanh**.

5. Lọc và hiển thị những nhân viên có họ **Nguyen** hoặc **Cao** và có tên **anh**.

6. Lọc và hiển thị nhữnh nhân viên có **Manv** bắt đầu là **B** và **maphong** là **A1** hoặc **Manv** bắt đầu là **C** và **maphong** là **A2**.

7. Lọc và hiển thị những nhân viên **nam** và có năm sinh **1975**.

8. Lọc và hiển thị những nhân viên **nam** sinh trong thời gian từ **12/12/76** đến **12/12/79**.

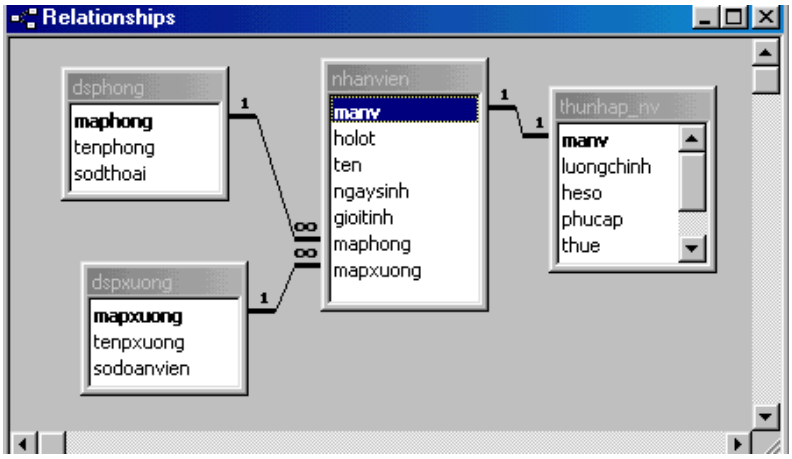
9. Lọc và hiển thị những nhân viên **nữ** có **Mapxuong** là **P1** và sinh trong tháng **10** năm **1975** hoặc nhân viên **nam** có năm sinh **1972** đến **1976**.

10. Mở bảng **Thunhap_nv** để hiển thị những nhân viên có **heso>100 và** có **luongchinh<500.000** hoặc có **phucap** từ **100.000** đến **500.000**.

BÀI SỐ 2

1. Mở CSDL NHANSU.MDB trong bài thực hành số 1 rồi thực hiện các nhiệm vụ sau:

- ❖ Đặt khoá chính cho trường Manv (nhanvien), Maphong(dsphong), manv(thunhap_nv) và mapxuong (dspxuong).
- ❖ Thiết lập các mối quan hệ giữa các bảng dữ liệu theo sơ đồ sau



❖ Hãy thiết lập các thuộc tính tham chiếu toàn vẹn và các thuộc tính như Format, caption, inputmask..... cho các trường một cách hợp lệ.

Đặt thuộc tính Format của trường **ngaysinh :** dd-mm-yy , Inputmask: 00/00/00

Đặt thuộc tính Inputmask của trường **sodthoai** : (000)000000

2. Tạo một truy vấn **query1** để hiển thị những thông tin sau:

Manv, holot, ten (trong bảng **nhanvien**), Luongchinh, phucap (Trong bảng **Thunhap_nv**), tenphong, maphong (Trong **dsphong**).

3. Tạo một truy vấn **query 2** để hiển thị những thông tin sau:

Manv, hoten (Nối holot và ten), maphong, tenphong, mapxuong, tenpxuong, tongthunhap trong đó tongthunhap được tính theo công thức:

Tongthunhap=luongchinh*heso+phucap-thue nếu là nhân viên **nam**.

Tongthunhap=luongchinh*heso+phucap nếu là nhân viên **nữ** .

4. Tạo một truy vấn **query3** để hiển thị những nhân viên có **maphong** là **p1** và **mapxuong** là **x1** sinh trong tháng **7** năm **1976** bao gồm những tin sau:

Hoten, gioitinh, maphong, tenphong, mapxuong, tenpxuong.

5. Tạo một truy vấn **query4** để hiển thị những nhân viên **nữ** và có **maphong** là **p2** hoặc **p3** sinh trong ngày **20** tháng **12** bao gồm những thông tin:

Hoten, maphong, ngaysinh, gioitinh (chú ý giá trị trường gioitinh phải hiển thị nam hoặc nữ).

6. Tạo một truy vấn **query5** để hiển thị những nhân viên **nam** có **tongthunhap>=60000** hoặc thuộc **maphong** là **p2** và không phải họ **Nguyãùn** bao gồm những thông tin: Ten, maphong, tenphong, tongthunhap.

7. Tạo một truy vấn **query6** để hiển thị những nhân viên **nữ** có **maphong** không bắt đầu là **p** hoặc những nhân viên thuộc phân xưởng không có đoàn viên nào bao gồm những thông tin sau: Holot, ten, gioitinh, tenpxuong.

8. Tạo một truy vấn **query7** để hiển thị những nhân viên **nam** tên **Thanh** hoặc **Long** hoặc những nhân viên **nữ** không phải họ **Lê** hoặc **trần** bao gồm những thông tin: Hoten, gioitinh, maphong, tenphong.

9. Tạo một truy vấn **query8** để hiển thị những nhân viên **nam** sinh trong khoảng thời gian từ năm **1973** đến **1980** thuộc phân xưởng có **mapxuong** là **x2** hoặc những nhân viên không có **thuế** sinh trong tháng **4** đến tháng **8** năm **1975** bao gồm những thông tin: Holot, ten, thangsinh, namsinh, mapxuong, tenpxuong.

10. Tạo một truy vấn **query9** để hiển thị những thông tin: Hoten, namsinh, thamnien, luongchinh, trong đó nếu

Luongchinh>=3000 và nhân viên **nữ** thì thamnien là **35**.

Luongchinh>=4000 và nhân viên **nam** thì thamnien là **30**.

BÀI SỐ 3

Tạo một CSDL có tên QLTV.MDB, rồi lần lượt tạo các bảng dữ liệu sau:

Bảng 1: **Loaisach**

STT	Fieldname	Data Type	Fieldsize	Note
1	**Masach**	Text	4	Khoá chính
2	Tensach	Text	20	
3	Tentacgia	Text	20	
4	Namxb	Date/time	8	
5	Soluongco	Number	3	Số lượng có
6	Sotrang	Number	5	Số trang sách
7	Manxb	Text	4	Mã nhà xuất bản

Bảng 1: **Docgiamuon**

STT	Fieldname	Data Type	Fieldsize	Note
1	Hoten	Text	30	
2	Quequan	Text	30	
3	**Madocgia**	Text	4	Khoá chính
4	Masach	Text (Lookup)	4	Lấy dữ liệu từ Loaisach
5	Sluong	Number	1	Số lượng mượn
6	Ngaymuon	Date/time	8	Ngày mượn
7	Ngayhen	Date/time	8	Ngày hẹn trả

Bảng 3: **Docgiatra**

STT	Fieldname	Data Type	Fieldsize	Note
1	**Madocgia**	Text	4	Khoá chính
2	Masach	Text	4	
3	Ngaytra	Date/time	8	Ngày trả

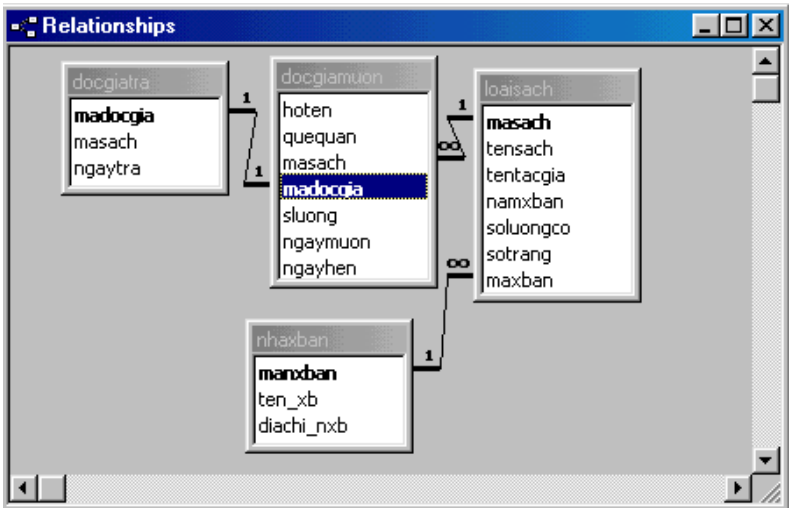
Bảng 4: : **Nhaxban**

STT	Fieldname	Data Type	Fieldsize	Note
1	**Manxban**	Text	4	Khoá chính
2	Ten_xb	Text	30	Tên nhà xuất bản
3	Diachi_nxb	Text	30	Địa chỉ Nhà XB

1. Nhập dữ liệu thích hợp cho các bảng dữ liệu.

2. Đặt khoá chính cho các trường: masach(Loaisach), Madocgia(Docgiamuon), Madocgia(Docgiatra), manxban(Nhaxban).

3. Thiết lập các mối quan hệ giữa các bảng dữ liệu theo sơ đồ sau



4. Tạo một truy vấn **TV1** để hiển thị danh sách độc giả mượn sách trong tháng 10 năm 98 bao gồm những thông tin: Hoten, ngaymuon, tensach, tentacgia.

5. Tạo một truy vấn **TV2** để hiển thị số lượng còn của mỗi loại sách của nhà xuất bản **Kim đồng** bao gồm những thông tin: tensach, nxban, soluongcon.

6. Tạo một truy vấn **TV3** để hiển thị danh sách những độc giả mượn sách quá hạn bao gồm những thông tin: Hoten, Ngaymuon, Ngayhen, songayqua (Số ngày quá hạn).

7. Tạo một truy vấn **TV4** để hiển thị danh sách những độc giả mượn sách quá hạn bao gồm những thông tin Hoten, sluong, mucquahan

Trong đó: Mucquahan là Mức 1 nếu songayqua<5

Mucquahan là Mức 2 nếu songayqua<10

Mucquahan là Mức 3 nếu songayqua>=10

8. Tạo một truy vấn **TV5** để hiển thị danh sách những độc giả trả sách đúng hạn hoặc sớm hạn bao gồm những thông tin: Hoten, masach, tensach.

9. Tạo một truy vấn **TV6** để hiển thị những loại sách có số trang >100 và số lượng còn là 20 của **nhà xuất bản giáo dục** hoặc tên sách có chữ **Tin học** bao gồm Tensach, sotrang, ten_xb, tentacgia.

BÀI SỐ 4

1. Sử dụng CSDL QLTV.MDB, tạo một truy vấn **Truyvan1** để hiển thị tổng số lượng mượn của từng loại sách trong thư viện.

2. Tạo một truy vấn **Truyvan2** để hiển thị tổng số lượng mượn của từng loại sách trong tháng **12** năm **1998**.

3. Tạo một truy vấn **Truyvan3** để hiển thị tổng số lượng mượn của từng loại sách theo từng tháng trong năm **1999**.

4. Tạo một truy vấn **Truyvan4** để hiển thị tổng số lượng mượn của từng loại sách theo từng tháng của một năm nào đó(Tháng và năm được nhập từ bàn phím).

5. Tạo một truy vấn **Truyvan8** để hiển thị số lượng mượn của từng loại sách trong năm **1999** và có số lượng mượn của mỗi độc giả **>2**.

6. Tạo một truy vấn **Truyvan9** để hiển thị tổng số loại sách có trong thư viện.

7. Tạo một truy vấn **Truyvan10** để hiển thị tổng số loại sách trong thư viện cho mượn trong năm **1998**.

8. Tạo một truy vấn **Truyvan11** để hiển thị tên của độc giả nào đó mượn sách trong **tháng 1 năm 2001** (Tên được nhập từ bàn phím).

9. Tạo một truy vấn **Truyvan12** để hiển thị tên sách và tên tác giả xuất bản sách trong năm **2002** (Tên sách nhập từ bàn phím, Họ của tác giả nhập từ bàn phím).

10. Tạo một truy vấn tham khảo chéo **Truyvan13** để phản ánh tổng số mỗi loại sách mượn cho mượn trong **tháng 11 năm 2000**.

11. Tạo một truy vấn tham khảo chéo **Truyvan14** để hiển thị tổng số laọi sách xuất bản trong năm **1995**.

12. Tạo một truy vấn **Truyvan14** để xoá những sinh viên đã mượn sách quá hạn **5** ngày.

BÀI SỐ 5

Tạo một CSDL có tên QLSV.MDB, rồi tạo các bảng dữ liệu sau:

Bảng 1: **DSSV**

STT	Fieldname	Data Type	Fieldsize	Note
1	**Masv**	Text	4	Khoá chính
2	Malop	Text	4	
3	Hotensv	Text	30	
4	Ngaysinh	Date/time	8	
5	Quequan	Text	30	
6	Gioitinh	Yes/No	1	
7	Hocbong	Number	Double	

Bảng 2: DSDIEM

STT	Fieldname	Data Type	Fieldsize	Note
1	**Masv**	Text	4	Khoá chính
2	Mamon	Text	2	Mã môn học
3	Diem_lan1	Number	Double	Điểm thi lần 1
4	Diem_lan2	Number	Double	Điểm thi lần 2

Bảng 3: DSLOP

STT	Fieldname	Data Type	Fieldsize	Note
1	**Malop**	Text	4	Khoá chính
2	Tenlop	Text	20	
3	Nganh_hoc	Text	30	
4	Khoahoc	Text	2	

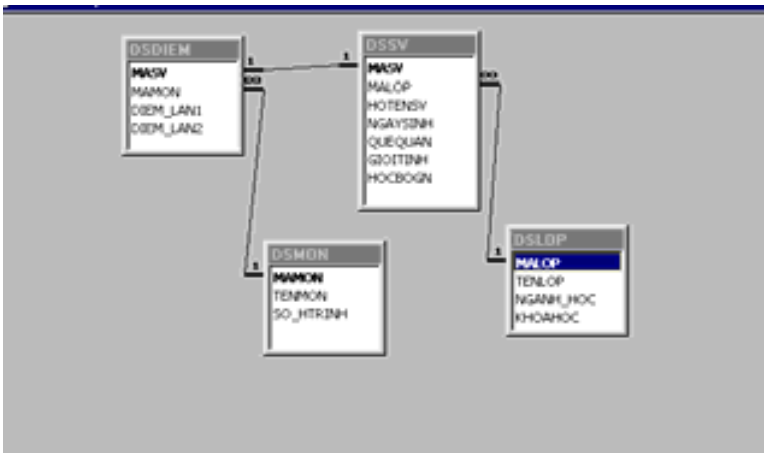
Bảng 3: DSMON

STT	Fieldname	Data Type	Fieldsize	Note
1	**Mamon**	Text	2	Khoá chính
2	Tenmon	Text	20	
3	So_htrinh	Number	Byte	Số học trình

1. Nhập dữ liệu thích hợp cho các bảng dữ liệu.

2. Đặt khoá chính cho các trường: MASV(DSSV), MASV(DSDIEM), MALOP(DSLOP), Mamon(DSMON).

3. Thiết lập các mối quan hệ giữa các bảng theo sơ đồ sau:



4. Tạo một truy vấn **BT1** để hiển thị tổng số sinh viên của mỗi lớp là bao nhiêu?

5. Tạo một truy vấn tham khảo chéo **BT2** để phản ánh tổng số sinh viên xếp loại ***Xsắc, Giỏi, khá, Tb, Yếu*** của mỗi lớp.

6. Từ bảng dữ liệu DSSV, tạo một truy vấn **BT3** để tạo ra bảng dữ liệu DSLUU lưu trữ những sinh viên có Mã lớp bắt đầu là **T** hoặc **H.**

7. Từ bảng dữ liệu DSSV, tạo một truy vấn **BT4** để tạo ra một bảng dữ liệu DSNAM để lưu trữ những sinh viên **nam** sinh trong năm **1976** bao gồm các thông tin: HOTENV, QUEQUAN, GIOTINH.

8. Từ bảng dữ liệu DSSV, tạo một truy vấn **BT5** để tạo ra một bảng dữ liệu DSNU để lưu trữ những sinh viên **nữ** sinh trong **quý 3** năm **1979** bao gồm các thông tin: HOTENV, QUEQUAN, GIOTINH.

9. Tạo một truy vấn **BT6** để tăng học bổng cho những sinh viên **nữ** thêm **30%**.

10. Tạo một truy vấn **BT7** để nối dữ liệu từ bảng DSNAM vào bảng DSNU.

11. Tạo truy vấn **BT8** để xoá những sinh viên có quê quán ở **Đà Nẵng** và có mã lớp bắt đầu là **B**.

BÀI SỐ 6

1. Sử dụng ngôn ngữ SQL để tạo cấu trúc của các bảng dữ liệu sau:

DSTRUONG (**MATRUONG**, TENTRUONG, DIACHI)
DSKHOA (MATRUONG, **MAKHOA**, TENKHOA, SODT)
DANHSACH (**MASV**, MAKHOA, HOTEN, NGAYSINH, LOP, HOCBONG)
BANGDIEM (**MASV**, DTBK1, DTBK2, DTBK3, DTBK4)

Trong đó: Các trường in đậm và gạch chân là khoá chính, kiểu dữ liệu và kích thuớc của các được mô tả như sau:

Bảng DSTRUONG

MATRUONG TEXT(2), TENTRUONG TEXT(20), DIACHI TEXT(30)

Bảng DSKHOA

MATRUONG TEXT(2), MAKHOA TEXT(4), TENKHOA TEXT(10), SODT TEXT(6). Trường SODT được lập chỉ mục.

Bảng DANHSACH

MASV TEXT(4), MAKHOA TEXT(4), HOTEN TEXT(30), NGAYSINH(DATE/TIME), LOP TEXT(10), HOCBONG (DOUBLE). Trường MASV được lập chỉ mục.

Bảng BANGDIEM

MASV TEXT(4), DTBK1 (DOUBLE), DTBK2 (DOUBLE), DTBK3 (DOUBLE), DTBK4 (DOUBLE).

☞ **Chú ý**

Sau khi tạo cấu trúc các bảng dữ liệuxong, hãy nhập dữ liệu và thiết lập mối quan hệ giữa các bảng phù hợp. Trường HOCBONG trong bảngg DANHSACH chỉ nhập một trong ba giá trị 120000, 180000 hoặc 240000.

2. Sử dụng ngôn ngữ SQL để thay đổi cấu trúc của bảng dữ liệu

a. Thêm trường GHICHU có kiểu MEMO vào trong bảng DANHSACH

b. Thêm trường TBCONG có kiểu DOUBLE vào trong bảng BANGDIEM

c. Thêm trường QUEQUAN có kiểu TEXT và GIOITINH có kiểu YES/NO vào bảng DANHSACH và lập chỉ mục trường QUEQUAN.

3. Sử dụng ngôn ngữ SQL để tạo các truy vấn chọn sau

a. Chọn MATRUONG, MAKHOA, TENKHOA trong bảng DSKHOA.

b. Chọn MATRUONG, MAKHOA, SODT trong bảng DSKHOA của những trường có MATRUONG bắt đầu là Q.

c. Chọn MASV, MAKHOA, HOTEN của những sinh viên sinh trong khoảng thời gian từ 20/10/74 đến 20/10/76 trong bảng DANHSACH.

d. Chọn MASV, HOTEN, LOP, HOCBONG của những sinh viên có MASV bắt đầu là T và thuộc lớp Tin học hoặc Hoá học hoặc kinh tế trong bảng DANHSACH (HOTEN đổi thành Họ và tên).

e. Chọn những sinh viên có tên THANH sinh trong tháng 10/76 hoặc có HOCBONG trong khoảng từ 150000 đến 200000.

4. Sử dụng ngông ngữ SQL để tạo các truy vấn tính tổng sau:

a. Tạo một truy vấn để tính tổng HOCBONG của mỗi khoa.

b. Tạo một truy vấn thống kê xem mỗi khoa số lượng sinh viên là bao nhiêu?

c. Tạo một truy vấn để tính tổng HOCBONG của mỗi lớp trong mỗi khoa.

d. Tạo một truy vấn thống kê xem mỗi trường có bao nhiêu khoa?

5. Sử dụng ngôn ngữ SQL để tạo các truy vấn tham số sau:

a. Tạo một truy vấn tham số thống kê xem mỗi mức HOCBONG mỗi khoa có bao nhiêu sinh viên.

b. Tạo một truy vấn tham số để thống kê xem số lượng sinh viên sinh trong mỗi tháng của năm 1979 của mỗi khoa là bao nhiêu?

6. Sử dụng ngông ngữ SQL để tạo các truy vấn tạo bảng sau:

a. Tạo một bảng DIEMLUU gồm tất cả các trường trong bảng DIEMTHI.

b. Tạo một bảng DSLUU gồm các trường MASV, MAKHOA, HOTEN từ bảng DANHSACH của những sinh viên sinh trước ngày 20/11/76.

c. Tạo một bảng DSLUU1 gồm các trường HOTEN, LOP của những sinh viên thuộc lớp **Tin K25A** và sinh năm 1985 hoặc trước năm 1978.

7. Sử dụng ngôn ngữ SQL để tạo các truy vấn nối dữ liệu sau:

a. Tạo một truy vấn nối dữ liệu từ bảng DIEMLUU vào DIEMTHI nhưng chỉ gồm các trường DTBK1, DTBK2.

b. Tạo một truy vấn nối dữ liệu từ bảng DIEMLUU vào DIEMTHI nhưng chỉ gồm các trường DTBK1, DTBK2, DTBK3 đối với những bản ghi ghi có DTBK3>=8.

8. Sử dụng ngôn ngữ SQL để tạo các truy vấn cập nhật dữ liệu sau:

a. Tính giá trị trường DTBCONG của bảng DIEMTHI theo công thức (DTBK1+ DTBK2+ DTBK3+ DTBK4)/4.

b. Tăng HOCBONG thêm 100.000 cho những sinh viên có MASV bắt đầu là A trong bảng DANHSACH.

c. Giảm HOCBONG đi 50.000 cho những sinh viên có năm sinh từ 1975 đến 1978 hoặc những sinh viên có tên NHAN.

9. Sử dụng ngôn ngữ SQL để tạo các truy vấn xoá sau:

a. Tạo một truy vấn xoá những sinh viên thuộc khoa toán.

b. Tạo một truy vấn xoá những sinh viên sinh trong khoảng thời gian từ 20/10/74 đến 20/10/76 hoặc có HOCBONG=120.000.

c. Tạo một truy vấn xoá những sinh viên có họ NGUYEN hoặc tên THANH sinh trong tháng 7 năm 1978.

10. Sử dụng ngôn ngữ SQL để tạo các truy vấn dựa trên nhiều bảng sau:

a. Tạo truy vấn để hiển thị HOTEN, TENKHOA, LOP, HOCBONG từ 2 bảng dữ liệu DSKHOA và DANHSACH.

b. Tạo một truy vấn để hiển thị TENTRUONG, TENKHOA, HOTEN của những sinh viên sinh trong năm 1980 từ 3 bảng DSTRUONG, DSKHOA, DANHSACH.

c. Tạo một truy vấn để hiển thị TENTRUONG, TENKHOA, HOTEN, DTBK1, DTBK2 tử 4 bảng dữ liệu DSTRUONG, DSKHOA, DANHSACH, DIEMTHI.

11. Sử dụng ngôn ngữ SQL để tạo các truy vấn con sau:

a. Tạo một truy vấn để hiển thị HOTEN, NGAYSINH, LOP của những sinh viên có DTBK4>=5.

b. Tạo một truy vấn để hiển thị it nhất một sinh viên có DTBK2<=4.

12. Sử dụng ngôn ngữ SQL để tạo các truy vấn hội sau:

a. Tạo truy vấn hội để hiển thị HOTEN, DTBK1, DTBK2 từ 2 bảng dữ liệu DIEMTHI và DIEMLUU.

b. Tạo truy vấn hội để hiển thị HOTEN, DTBK1, DTBK2, DTBK3 từ 2 bảng dữ liệu DIEMTHI và DIEMLUU nhưng chỉ hiển thị những sinh viên có DTBK3>=7.

BÀI SỐ 7

Sử dụng CSDL **QLSV.MDB** trong **bài tập số 5** để thực hiện các yêu cầu sau:

1. Tạo một form có tên **Nhaplop** để nhập dữ liệu cho bảng **dslop** như sau:

2. Tạo một form chính phụ như sau:

3. Tạo một form có tên **Hienthi** để hiển thị danh sách sinh viên như sau :

4. Hãy thiết kế form theo yêu cầu sau:

Hãy gắn các chức năng phù hợp với các mục chọn.

5. Hãy thiết kế một **Report** theo yêu cầu sau:

BÀI SỐ 8

Cho CSDL **QLSVIEN.MBD** gồm các bảng dữ liệu sau:

DSSVIEN(**MASV**, MALOP, HOTENSVIEN, NGAYSINH, GTINH, QQUAN)

DSMONHOC(**MAMON**, TENMON, SOH_TRINH)

DSDIEMTHI(**MASV**, MAMON, DIEM_LAN1, DIEM_LAN2)

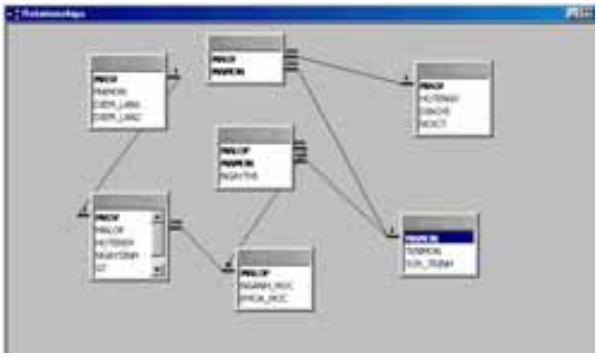
DSGIAOVIEN(**MAGV**, HOTENGVIEN, DIACHI, NOI_CT)

DSLOPHOC(**MALOP**, NGANH_HOC, KHOA_HOC)

GVMONHOC(**MAGV**, **MAMON**)

DSLOPMON(**MALOP**, **MAMON**, NGAYTHI)

Mối quan hệ giữa các bảng dữ liệu theo sơ đồ sau:



1. Tạo các biểu mẫu để cập nhật dữ liệu cho các bảng trên, chú ý sử dụng combo box hoặc List box đối với các trường cần thiết.

2. Tạo báo cáo để hiển thị danh sách sinh viên trong một lớp học nào đó, bao gồm cả ngành học tương ứng.

3. Tạo báo cáo để hiển thị danh sách sinh viên trong từng lớp học, bao gồm cả ngành học, trong đó các sinh viên cùng một lớp thì tên lớp được nhóm lại với nhau.

4. Tạo báo cáo để hiển thị danh sách giáo viên đã giảng dạy các một học cho một lớp nào đó.

5. Tạo báo cáo để hiển thị danh sách giáo viên đã tham gia giảng dạy các môn học cho nhiều lớp học, trong đó các giáo viên giảng dạy các môn học cho một lớp học được nhóm lại với nhau.

6. Tạo báo cáo để hiển thị điểm thi các môn học của một sinh viên.

7. Tạo báo cáo để hiển thị điểm thi một môn học của một lớp nào đó.

CHƯƠNG TRÌNH TỔNG THỂ CẢI CÁCH HÀNH CHÍNH

CỦA CHÍNH PHỦ GIAI ĐOẠN (2001-2010)

BAN ĐIỀU HÀNH ĐỀ ÁN 112

GIÁO TRÌNH
PHÂN TÍCH, THIẾT KẾ, XÂY DỰNG VÀ QUẢN TRỊ CÁC HỆ THỐNG CƠ SỞ DỮ LIỆU

(Giáo trình nộp lần 2)

Tháng 8 - 2004

MỤC LỤC

Lời giới thiệu 10

Chủ đề I: Khái niệm cơ sở dữ liệu 16

1. Giới thiệu chung 16
2. Các khái niệm cơ bản 17
   2.1. Cơ sở dữ liệu 17
      a. Các hệ thống xử lý tệp truyền thống 18
      b. Cách tiếp cận cơ sở dữ liệu 22
   2.2. Dữ liệu và thông tin 26
      a. Dữ liệu 26
      b. Dữ liệu và thông tin 28
      c. Siêu dữ liệu 30
   2.3. Kiều thực thể và bảng dữ liệu 32
      a. Thực thể 32
      b. Kiểu thực thể 32
      c. Bảng dữ liệu hay quan hệ 35
   2.4. Thuộc tính và trường dữ liệu 39
      a. Thuộc tính 39
      b. Trường dữ liệu 41
   2.5. Liên kết giữa các thực thể 42
   2.6. Khái niệm toàn vẹn dữ liệu 44
3. Ví dụ về một cơ sở dữ liệu 45
4. Các mô hình dữ liệu 48
   4.1. Khái quát chung 48
   4.2. Mô hình dữ liệu quan hệ 50
   4.3. Một số mô hình dữ liệu khác 51
      a. Mô hình dữ liệu mạng và mô hình dữ liệu phân cấp 51
      b. Mô hình dữ liệu hướng đối tượng 52
5. Kho dữ liệu 53

6. Kết luận 54
7. Bài tập 55
Chủ đề II: Phương pháp phân tích, thiết kế và xây dựng các hệ thống CSDL 56
1. Khái quát chung 56
2. Phân tích yêu cầu 57
2.1. Xác định dữ liệu cần lưu trữ 60
2.2. Xác định ứng dụng sẽ được cài đặt trên cơ sở dữ liệu 61
2.3. Xác định các thao tác thường xuyên được thực hiện 62
2.4. Một số kỹ thuật phân tích yêu cầu 63
a. Kỹ thuật phỏng vẫn 63
b. Kỹ thuật sử dụng phiếu khảo sát 65
c. Kỹ thuật phân tích tài liệu 65
3. Thiết kế cơ sở dữ liệu mức khái niệm 66
3.1. Lựa chọn giữa kiểu thực thể mạnh và kiểu thực thể yếu 67
a. Kiểu thực thể mạnh 67
b. Kiểu thực thể yếu 68
3.2. Lựa chọn giữa các loại thuộc tính 70
a. Thuộc tính đơn và gộp 70
b. Thuộc tính đơn trị và thuộc tính đa trị 71
c. Thuộc tính lưu trữ và thuộc tính dẫn xuất 72
d. Thuộc tính khoá hay thuộc tính định danh 73
3.3. Lựa chọn giữa các loại liên kết 77
a. Ba khái niệm cơ bản về liên kết 77
b. Các liên kết một-một (1:1) 81
c. Các liên kết một-nhiều (1:N) 82
d. Các liên kết nhiều-nhiều (M:N) 83
e. Thuộc tính cho liên kết 85

3.4. Lựa chọn giữa kiểu thực thể và thuộc tính.....87
3.5. Lựa chọn giữa kiểu thực thể và liên kết.....89
3.6. Các phân cấp "là-một".....92
3.7. Một sơ đồ thực thể liên kết.....95
3.8. Mô tả các ràng buộc trên dữ liệu 105
4. Thiết kế cơ sở dữ liệu mức logic.....109
4.1. Giới thiệu chung về ngôn ngữ SQL.....109
4.2. Ràng buộc toàn vẹn 110
a. Ràng buộc miền.....111
b. Ràng buộc khoá.....113
c. Các kích hoạt.....114
4.3. Lựa chọn hệ quản trị CSDL.....116
4.4. Bước một: Chuyển đổi các kiểu thực thể thông thường 117
4.5. Bước hai: Chuyển đổi các kiểu thực thể yếu 121
4.6. Bước ba: Chuyển đổi các liên kết giữa hai kiểu thực thể.....124
a. Chuyển đổi các liên kết một-nhiều 124
b. Chuyển đổi các liên kết nhiều-nhiều.....126
c. Chuyển đổi các liên kết một-một.....129
4.7. Bước bốn: Chuyển đổi các kiểu thực thể kết nối 132
a. Trường hợp thuộc tính định danh không xác định 133
b. Trường hợp thuộc tính định danh đã xác định...135
4.8. Bước năm: Chuyển đổi các liên kết đơn.....138
a. Chuyển đổi các liên kết đơn một-nhiều 139
b. Chuyển đổi các liên kết đơn nhiều-nhiều 141
4.9. Bước sáu: Chuyển đổi các liên kết giữa nhiều kiểu thực thể.....143
4.10. Bước bảy: Chuyển đổi các phân cấp "là một" ..146
4.11. Một sơ đồ cơ sở dữ liệu mức logic 150

4.12. Kết luận 155
5. Tinh chỉnh sơ đồ dữ liệu và các dạng chuẩn 157
5.1. Tinh chỉnh các lược đồ quan hệ 157
5.2. Phụ thuộc hàm 159
5.3. Các dạng chuẩn 161
a. Dạng chuẩn 1NF 161
b. Dạng chuẩn 2NF 163
c. Dạng chuẩn 3NF 166
6. Thiết kế cơ sở dữ liệu vật lý 171
6.1. Phân tích kích thước và nhu cầu sử dụng dữ liệu 172
6.2. Thiết kế các trường 173
a. Lựa chọn kiểu dữ liệu 173
b. Kiểm soát các toàn vẹn dữ liệu 174
6.3. Tổ chức các bản ghi vật lý và phi chuẩn 177
a. Tổ chức các bản ghi vật lý 177
b. Phi chuẩn 179
6.4. Tổ chức các tệp vật lý 188
a. Tổ chức tệp tuần tự 189
b. Tổ chức tệp chỉ mục 190
c. Tổ chức các tệp băm 196
d. Tổ chức các tệp được phân cụm (clustering) 196
6.5. Lựa chọn và sử dụng các loại chỉ mục 198
6.6. RAID 207
6.7. Tối ưu hoá hiệu năng truy vấn 217
7. Thiết kế an toàn bảo mật 219
7.1. Xác định các nhóm sử dụng và các quyền hạn ... 221
7.2. Xác định quyền hạn thao tác trên dữ liệu 222
7.3. Mã hoá dữ liệu 240
7.4. Vai trò của người quản trị 245
8. Kết luận 247
9. Bài tập 247

Chủ đề III: Quản trị hệ thống cơ sở dữ liệu 250
1. Khái quát chung 250
2. Các mô hình tổ chức cơ sở dữ liệu 251
2.1. Các hệ cơ sở dữ liệu tập trung 252
a. Hệ cơ sở dữ liệu cá nhân 252
b. Hệ cơ sở dữ liệu trung tâm 253
c. Hệ cơ sở dữ liệu khách/chủ 254
2.2. Các hệ cơ sở dữ liệu phân tán 254
a. Hệ cơ sở dữ liệu phân tán thuần nhất..... 255
b. Hệ cơ sở dữ liệu phân tán không thuần nhất..... 256
3. Quản trị hệ thống cơ sở dữ liệu 257
3.1. Quản trị nội dung cơ sở dữ liệu 263
a. Theo dõi cơ sở dữ liệu 263
b. Hiệu chỉnh cơ sở dữ liệu..... 265
c. Sao lưu và phục hồi dữ liệu 266
3.2. Quản trị kỹ thuật..... 267
a. Cài đặt, cấu hình hệ thống 268
b. Đảm bảo về đường truyền, mạng..... 269
c. Quản trị tài nguyên cơ sở dữ liệu..... 269
3.3. Đảm bảo an ninh, phân quyền truy nhập 270
3.4. Một số lỗi thường gặp và cách khắc phục 271
a. Không thể truy cập được cơ sở dữ liệu..... 271
b. Hệ thống báo lỗi timeout (chậm quá thời gian cho phép)..... 272
c. Hệ thống báo lỗi Out of Memory (tràn bộ nhớ). 273
d. Không thể cung cấp được dữ liệu 274
e. Virus 274
4. Kết luận..... 275
5. Bài tập..... 276
Chủ đề IV: Một số hệ quản trị cơ sở dữ liệu phổ biến 277

1. Giới thiệu chung 277
2. Hệ quản trị cơ sở dữ liệu 280
2.1. Kiến trúc của một hệ quản trị cơ sở dữ liệu 280
2.2. Lưu trữ dữ liệu trong một hệ quản trị cơ sở dữ liệu 282
2.3. Truy vấn trong một hệ quản trị cơ sở dữ liệu..... 283
3. Một số hệ thống quản trị cơ sở dữ liệu thông dụng . 284
3.1. Các hệ quản trị cơ sở dữ liệu cỡ vừa và nhỏ 284
a. Hệ quản trị cơ sở dữ liệu Access 284
b. Hệ quản trị cơ sở dữ liệu Visual Foxpro 295
3.2. Các hệ quản trị cơ sở dữ liệu cỡ lớn 311
a. Hệ quản trị cơ sở dữ liệu SQL Server..... 311
b. Hệ quản trị cơ sở dữ liệu Oracle 340
3.3. Hệ quản trị cơ sở dữ liệu mã nguồn mở..... 368
3.4. Nhận xét chung..... 370
4. Minh hoạ đối với một hệ quản trị cơ sở dữ liệu cụ thể 373
5. Kết luận..... 373
6. Bài tập 374

Phụ lục A: Các hệ thống CSDL sẽ được xây dựng trong khuôn khổ dự án 112 375

1. Mô hình tổ chức hệ thống CSDL các cấp..... 375
2. Các cơ sở dữ liệu phục vụ điều hành, quản lý..... 377
a. Cơ sở dữ liệu phục vụ hệ thống thông tin quản lý đơn thư khiếu nại tố cáo..... 377
b. Cơ sở dữ liệu phục vụ HTTT luật và văn bản qui phạm pháp luật..... 378
c. Các cơ sở dữ liệu ứng dụng chuyên ngành 379
3. Các cơ sở dữ liệu phục vụ dịch vụ công..... 380
4. Các cơ sở dữ liệu quốc gia 382

Phụ lục B: Phân tích, thiết kế cơ sở dữ liệu cho một cửa hàng sách 383
1. Phân tích yêu cầu 383
2. Thiết kế cơ sở dữ liệu mức khái niệm 384
3. Thiết kế cơ sở dữ liệu mức logic 386
4. Tinh chỉnh lược đồ 390
5. Thiết kế cơ sở dữ liệu vật lý 392
6. Vấn đề an toàn bảo mật 395

Tài liệu tham khảo 399

Lời giới thiệu

Tài liệu này được biên soạn trong khuôn khổ đào tạo thuộc Đề án 112 với thời gian tập huấn chính thức là 45 tiết. Mục tiêu chính của tài liệu này là:

- Nhằm trang bị cho học viên các kiến thức và kỹ năng cơ bản trong việc phân tích, thiết kế và xây dựng các hệ thống cơ sở dữ liệu.
- Giúp học viên có đủ kiến thức cơ sở và những kỹ năng cần thiết để có thể quản trị các hệ thống cơ sở dữ liệu
- Cung cấp cho học viên một số kinh nghiệm cần thiết để có thể vận hành, quản trị hệ thống và khắc phục một số sự cố và sửa lỗi thông thường đối với các hệ quản trị CSDL.

Trong khi trình bày, chúng tôi đã cố gắng Việt hoá hầu hết các thuật ngữ tiếng Anh. Với các ví dụ chúng tôi cố gắng sử dụng tiếng Việt để diễn giải và đặt tên các đối tượng bằng tiếng Anh (ví dụ: kiểu thực thể nhân viên EMPLOYEE, thuộc tính mã hiệu nhân viên Emp_no,...).

Đối với phần thực hành của giáo trình, tuỳ theo từng đối tượng học viên khác nhau, giảng viên có thể lựa chọn một trong số các hệ quản trị cơ sở dữ liệu quen thuộc như: Access, Visual Foxpro, Sql Server, Oracle,....

Khi chuẩn bị môi trường thực hành, giảng viên nên lưu ý những điểm sau:

- Đối với hai hệ quản trị cơ sở dữ liệu Access và Visual Foxpro, nên cài đặt trên mỗi máy của học viên.
- Đối với hệ quản trị cơ sở dữ liệu Microsoft SQL Server nên cài đặt một máy làm Server chứa cơ sở dữ liệu, các máy thực hành của học viên chỉ cài ở chế độ Client và kết nối đến máy Server trên. Tạo cho mỗi học viên một tài khoản và một cơ sở dữ liệu khác nhau.
- Đối với hệ quản trị cơ sở dữ liệu Oracle, nên sử dụng một máy có cấu hình tương đối mạnh để cài đặt làm máy Server. Trong trường hợp có thể cho học viên thực hành làm quen với công cụ thiết kế Oracle Designer, cần thiết lập thêm kho thiết kế repository. Sau đó tạo cho mỗi học viên một tài khoản người dùng khác nhau và cấp quyền thiết kế tương ứng. Tại mỗi máy thực hành, cài đặt thêm công cụ thiết kế Oracle Designer và cấu hình kết nối đến Server trên. Cố gắng hướng dẫn học viên từ cách tạo các kiểu thực thể, thuộc tính, liên kết,... bằng tiện ích Entity Relationship Diagrammer, sau đó chuyển đổi sang sơ đồ dữ liệu Data Diagram, rồi cuối cùng sinh ra mã để tạo cơ sở dữ liệu.
- Nếu lựa chọn một trong hai hệ quản trị cơ sở dữ liệu là Microsoft SQL Server hoặc Oracle làm bài giảng thực hành, giảng viên nên bố trí thời gian cài đặt và cấu hình trước để có thêm thời gian trên lớp.

Đối với bài toán lựa chọn để thực hành, giảng viên có thể tự lựa chọn một bài toán phù hợp với mình và học viên, hoặc có thể lựa chọn bài toán đã trình bày trong giáo trình

nêu ở mục 3 chủ đề 1, hoặc cũng có thể lựa chọn bài toán được trình bày trong phụ lục B của giáo trình.

Trong mỗi chủ đề, như trình bày dưới đây, chúng tôi có đề xuất thời gian dự kiến cho từng chủ đề đó cũng như các nội dung tối thiểu mà giảng viên cần phải đảm bảo. Tuy nhiên, tuỳ theo từng đối tượng học viên, giảng viên có thể thay đổi cho phù hợp. Cuối mỗi chủ đề, chúng tôi đều có phần tóm tắt các nội dung đã trình bày nhằm giúp học viên hệ thống hoá lại kiến thức đã được học. Ngoài ra phần bài tập giúp học viên hiểu sâu hơn nội dung đã trình bày, cũng như gợi mở các vấn đề thực tế.

Cụ thể, giáo trình gồm các nội dung chính sau:

Chủ đề 1: Khái niệm cơ sở dữ liệu

Trong chủ đề này, giáo trình tập trung trình bày các khái niệm cơ bản nhất về cơ sở dữ liệu và các khái niệm được sử dụng trong quá trình phân tích, thiết kế cơ sở dữ liệu. Thời lượng dành cho việc trình bày cho chủ đề này có thể không nhiều, khoảng từ 4 đến 6 tiết, với mục đích chủ yếu giúp học viên nắm bắt và hình dung được các khái niệm. Các nội dung như về mô hình dữ liệu, kho dữ liệu, giảng viên không cần đi sâu và có thể trình bày lướt qua.

Chủ đề 2: Phương pháp phân tích, thiết kế và xây dựng các hệ thống cơ sở dữ liệu

Đây được xem là chủ đề quan trọng nhất của giáo trình. Do vậy thời gian dành cho chủ đề này sẽ khoảng từ 14 đến 16 tiết, tuỳ từng đối tượng học viên. Nội dung của chủ đề tập trung vào 6 bước chính của quá trình phân tích, thiết kế và xây dựng hệ thống cơ sở dữ liệu: phân tích yêu cầu, thiết kế cơ sở dữ liệu mức khái niệm, thiết kế cơ sở dữ liệu mức

logic, tinh chỉnh lược đồ dữ liệu và các dạng chuẩn, thiết kế cơ sở dữ liệu mức vật lý, và bước cuối cùng là thiết kế an toàn bảo mật. Đây cũng là các nội dung mà giảng viên cần phải đảm bảo. Tuy nhiên với mục 6 của chủ đề, giảng viên có thể lựa chọn một số mục chủ yếu để giảng và lướt qua các mục còn lại, bởi một số nội dung trong mục này mang tính chất công nghệ và phụ thuộc vào phần cứng, hệ điều hành cũng như hệ quản trị cơ sở dữ liệu.

Ngoài ra, các ví dụ trong chủ đề này được chúng tôi tập trung trình bày để giải quyết một bài toán đã được giới thiệu trong mục 3 chủ đề 1. Giảng viên có thể lựa chọn thêm những ví dụ khác để minh hoạ thêm cho nội dung bài giảng.

Chủ đề 3: Quản trị hệ thống cơ sở dữ liệu

Quản trị hệ thống cơ sở dữ liệu là một phần tất yếu sau khi cơ sở dữ liệu được xây dựng đưa vào sử dụng. Việc quản trị một hệ thống cơ sở dữ liệu bao gồm rất nhiều vấn đề và thông thường, với một hệ thống cơ sở dữ liệu lớn thì cần phải có một đội ngũ chuyên gia tin học có kỹ năng. Toàn bộ nôi dung của chủ đề tập trung chủ yếu vào các vấn đề như: quản trị về mặt nội dung, về mặt kỹ thuật, về đảm bảo an ninh, phân quyền dữ liệu. Trong phần cuối của chủ đề chúng tôi dành riêng một mục để đề cập đến một số lỗi thường gặp trong quá trình quản trị một hệ thống cơ sở dữ liệu và hướng dẫn một số cách khắc phục. Thời lượng dành cho chủ đề này là từ 6 đến 8 tiết.

Trong chủ đề, đối với phần các mô hình tổ chức cơ sở dữ liệu (mục 2), giảng viên không cần tập trung nhiều thời gian và có thể cho học viên đọc trước và trên lớp chỉ cần nêu một số đặc điểm chính mang tính chất khái quát. Nội

dung tối thiểu cần phải đảm bảo trong chủ đề đó là toàn bộ mục 3, Quản trị hệ thống cơ sở dữ liệu.

Chủ đề 4: Một số hệ quản trị cơ sở dữ liệu phổ biến

Trong chủ đề này, chúng tôi tập trung trình bày về kiến trúc chung của một hệ quản trị cơ sở dữ liệu, giúp học viên có được cái nhìn tổng quát về một hệ quản trị cơ sở dữ liệu. Các phần còn lại của chủ đề được dành cho việc trình bày bốn hệ quản trị cơ sở dữ liệu thông dụng hiện nay: Access, Visual Foxpro, SQL Server, Oracle. Đối với mỗi một hệ quản trị cơ sở dữ liệu, chúng tôi trình bày lần lượt các bước để xây dựng một cơ sở dữ liệu, do vậy học viên có thể dễ dàng lựa chọn hệ quản trị cơ sở dữ liệu mà mình ưa thích để thực hành. Thời lượng dành cho chủ đề này có thể là từ 14 đến 16 tiết bao gồm cả thời gian dành cho phần thực hành.

Đối với các nội dung trong chủ đề, giảng viên cần đảm bảo giúp học viên nắm được thế nào là một hệ quản trị cơ sở dữ liệu, các thành phần thường có trong một hệ quản trị cơ sở dữ liệu. Ngoài ra, thời gian chủ yếu trong chủ đề sẽ là phần thực hành cụ thể trên một hệ quản trị cơ sở dữ liệu. Như trên đã trình bày, tuỳ từng đối tượng học viên cũng như khả năng sắp xếp thời gian, giảng viên có thể lựa chọn một trong bốn hệ quản trị cơ sở dữ liệu đã trình bày để giúp học viên thực hành. Tuy nhiên chúng tôi đề xuất là giảng viên nên lựa chọn hệ quản trị cơ sở dữ liệu Microsoft SQL Server, và nếu có điều kiện thì lựa chọn hệ quản trị cơ sở dữ liệu Oracle, bởi với hệ quản trị cơ sở dữ liệu này, giảng viên minh hoạ được cả quá trình thiết kế từ sơ đồ thực thể - liên kết, sơ đồ dữ liệu,... thông qua công cụ Oracle

Designer, phần mà ở các hệ quản trị cơ sở dữ liệu khác chưa có.

Phụ lục A: Các hệ thống CSDL sẽ được xây dựng trong khuôn khổ dự án 112

Nội dung của phụ lục này nhằm giới thiệu với các học viên một số nét chính về hệ thống các cơ sở dữ liệu mang tính chất định hướng sẽ được xây dựng trong khuôn khổ dự án 112. Tuy nhiên, giảng viên cũng nên dành thời gian cho phụ lục này khoảng từ 2 cho đến 4 tiết.

Phụ lụcB: Phân tích, thiết kế cơ sở dữ liệu cho một cửa hàng sách

Phụ lục này được xem như một bài thực hành (case study) thêm cho học viên. Bài toán được trình bày trong phụ lục theo thể loại hỏi đáp, gần gũi với học viên: từ nhu cầu của khách hàng, các luồng suy nghĩ đề giải quyết đối với người phân tích, thiết kế cơ sở dữ liệu, đến quá trình xây dựng và cài đặt cơ sở dữ liệu. Điều quan trọng hơn là bài toán đã đưa ra những vấn đề nảy sinh trong thực tế sau khi hệ cơ sở dữ liệu được đưa vào sử dụng và các hướng giải quyết, đây chính là một quá trình thường gặp với các bài toán thực tế.

Chủ đề I: Khái niệm cơ sở dữ liệu

Mục tiêu

Trong chủ đề này các bạn cần phải nắm được các khái niệm cơ bản nhất về cơ sở dữ liệu, cũng như các khái niệm được sử dụng trong quá trình phân tích, thiết kế, xây dựng cơ sở dữ liệu quan hệ như thực thể, kiểu thực thể, quan hệ, bảng dữ liệu, thuộc tính, trường dữ liệu, cùng các liên kết.

1. Giới thiệu chung

Hơn hai thập kỷ gần đây chúng ta đã được chứng kiến sự phát triển vượt bậc cả về số lượng và chất lượng của các ứng dụng dựa trên cơ sở dữ liệu, chúng ngày càng đóng vai trò quan trọng trong các hoạt động của các tổ chức, doanh nghiệp, hay cá nhân. Cơ sở dữ liệu có thể được sử dụng với những ứng dụng cài đặt riêng lẻ trong máy tính cá nhân hay dưới dạng một cơ sở dữ liệu tập trung trên mạng LAN, hoặc một cơ sở dữ liệu phân tán trên mạng diện rộng được sử dụng bởi rất nhiều các đối tượng khác nhau.

Trong chủ đề này, chúng ta sẽ tìm hiểu những khái niệm cơ bản nhất về cơ sở dữ liệu, cũng như những khái niệm sẽ được sử dụng trong quá trình phân tích, thiết kế cơ sở dữ liệu. Bên cạnh đó, chúng ta cũng sẽ xem xét các mô hình dữ liệu khác nhau đối với cơ sở dữ liệu.Tuy nhiên xin lưu ý rằng các quá trình phân tích, thiết kế, xây dựng và quản trị cơ sở dữ liệu được tập trung trình bày trong giáo trình này sẽ là cơ sở dữ liệu quan hệ, một loại cơ sở dữ liệu được xây dựng dựa trên mô hình dữ liệu quan hệ có lý thuyết toán

học vững chắc nhất. Ngoài ra, đây cũng là loại cơ sở dữ liệu phổ biến nhất, được sử dụng hầu hết trong các hệ quản trị cơ sở dữ liệu thông dụng trên thế giới.

2. Các khái niệm cơ bản

2.1. Cơ sở dữ liệu

Ngày nay, hơn bao giờ hết, sự thành công của một doanh nghiệp phụ thuộc vào khả năng có được những dữ liệu chính xác và kịp thời về các hoạt động của doanh nghiệp, để quản lý và phân tích dữ liệu một cách có hiệu quả nhằm đưa ra được chiến lược cho những hoạt động tiếp theo. Các thuật ngữ như "xa lộ thông tin" đã trở nên phổ biến và vấn đề quản lý và sử dụng thông tin dần trở thành một ngành công nghiệp có doanh số nhiều tỷ đô la.

Người ta đã coi giá trị của dữ liệu như là một thứ tài sản quý giá của doanh nghiệp với hàm lượng thông tin bao hàm trong đó. Tuy nhiên, nếu không có được sự quản lý một cách hữu hiệu đối với lượng dữ liệu ngày càng lớn này, cũng như khả năng đáp ứng yêu cầu của các câu hỏi tìm kiếm, lượng dữ liệu này sẽ trở nên vô nghĩa và khi đó doanh nghiệp sẽ mất đi một nguồn tài sản quý giá. Chính điều này đã thúc đẩy sự cần thiết có được một cơ sở dữ liệu cũng như một hệ quản trị cơ sở dữ liệu hiệu quả và linh hoạt, hỗ trợ doanh nghiệp trong công tác quản lý cũng như tìm kiếm thông tin.

Chúng ta có thể định nghĩa cơ sở dữ liệu như là một bộ sưu tập có tổ chức của các dữ liệu liên quan logic với nhau và được các hệ ứng dụng của một tổ chức cụ thể nào đó sử dụng.

Ví dụ: một người bán hàng có thể lưu danh sách khách hàng trong một cơ sở dữ liệu được cài đặt trong máy sách tay của anh ta với kích thước chỉ khoảng vài megabytes và cơ sở dữ liệu này sẽ phục vụ cho một chương trình quảng cáo sản phẩm chẳng hạn. Nhưng một doanh nghiệp có thể có một cơ sở dữ liệu vô cùng lớn, cỡ khoảng vài terabytes dữ liệu được lưu trong một máy tính lớn mainframe, và nó được sử dụng bởi một ứng dụng hỗ trợ ra quyết định cho doanh nghiệp đó.

Tuy nhiên, từ khi có máy tính, người ta đã tìm cách lưu trữ dữ liệu và xử lý chúng, một trong những hệ thống xử lý dữ liệu đầu tiên đó là các hệ thống xử lý tệp. Mặc dù ngày nay những hệ thống xử lý tệp này không còn được sử dụng rộng rãi nhưng nó vẫn tồn tại trong một số lĩnh vực hẹp. Để có thể thấy được những ưu nhược điểm của hệ thống xử lý tệp cũng như có thể hiểu sâu hơn tại sao người ta lại mong muốn xây dựng lên những cơ sở dữ liệu cùng các ưu điểm của cách tiếp cận cơ sở dữ liệu so với cách tiếp cận dựa trên hệ thống xử lý tệp, chúng ta sẽ xem xét một cách khái quát các khái niệm sau đây:.

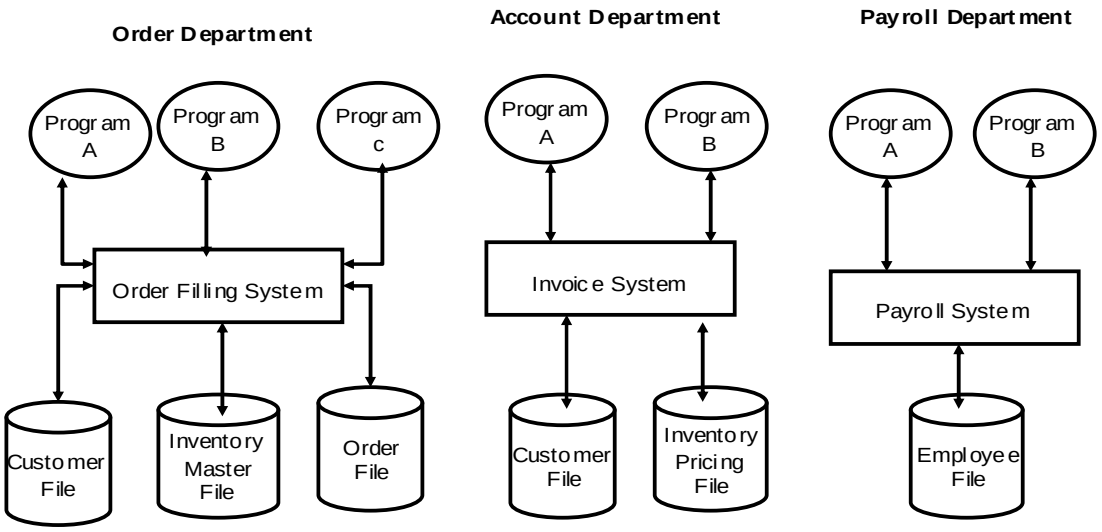
a. Các hệ thống xử lý tệp truyền thống

Như chúng ta đã biết, khi chưa có cơ sở dữ liệu, các máy tính của chúng ta phải xử lý dữ liệu thông qua các hệ thống xử lý tệp truyền thống. Ngày nay, khi khả năng của các máy tính đã tăng lên một cách đáng kể, có thể giải quyết các bài toán lớn, đòi hỏi xử lý những khối dữ liệu đồ sộ, phức tạp, các hệ thống xử lý tệp truyền thống tỏ ra không còn thích hợp và dần dần nó được thay thế bởi các hệ thống xử lý dựa trên cơ sở dữ liệu. Tuy nhiên sự hiểu biết về các hệ thống xử lý tệp vẫn là cần thiết bởi hai lý do cơ bản sau:

- Ngày nay các hệ thống xử lý tệp vẫn còn được sử dụng trong một số hệ thống, đặc biệt là các hệ thống phục vụ sao lưu cơ sở dữ liệu.
- Hiểu được các vấn đề cũng như các giới hạn của các hệ thống xử lý tệp sẽ giúp chúng ta thiết kế các cơ sở dữ liệu tốt hơn.

Ví dụ của một ứng dụng dựa trên hệ thống xử lý tệp:

Một công ty chuyên sản xuất các đồ dùng bằng gỗ đã tiến hành tin học hoá các nghiệp vụ của mình trên máy tính. Thọat đầu, mỗi một ứng dụng trong công ty đều được xây dựng theo cách tiếp cận xử lý tệp truyền thống. Với cách tiếp cận này, các hệ thống thông tin đều tập trung vào việc xử lý dữ liệu của mỗi một phòng ban đơn lẻ, trong đó các hệ thống cố gắng đáp ứng các yêu cầu của từng nhóm người cụ thể. Do vậy, nhìn toàn bộ vào hệ thống thông tin của toàn công ty thì không có một mô hình cũng như kế hoạch hay sơ đồ tổng thể nào để có thể hoạch định được cho việc phát triển tiếp theo của các ứng dụng. Hệ thống ứng dụng dựa trên trên cách tiếp cận xử lý tệp này được minh hoạ như trong hình 1.2.1.1, đó là các hệ thống tạo lập đơn hàng Order Filling System, hệ thống xử lý hoá đơn Invoice System và hệ thống quản lý lương bổng cho nhân viên Payroll System. Nhìn vào hình vẽ chúng ta thấy rằng, các tệp dữ liệu được kết nối trực tiếp với mỗi chương trình ứng dụng, mỗi một tệp là một tập các bản ghi liên quan. Ví dụ, với hệ thống Order Filling System có 3 tệp: Customer, Inventory Master và Order. Chúng ta nhận thấy rằng có thể có những tệp được lưu lặp lại và sử dụng bởi các ứng dụng khác nhau (như Customer), đây chính là một trong những nhược điểm của các hệ thống dựa trên việc xử lý tệp.



Hình 1.2.1.1 Một hệ thống xử lý tệp truyền thống

Các nhược điểm của các hệ thống xử lý tệp:

- Tính phụ thuộc giữa chương trình và dữ liệu: Các mô tả về tệp dữ liệu được lưu trong mỗi một chương trình ứng dụng sử dụng tệp đó. Như ví dụ trên, chương trình Program A truy nhập cả hai tệp Inventory và Customer, do vậy chương trình này sẽ chứa các mô tả chi tiết của cả 2 tệp này. Điều này dẫn đến, khi có bất cứ một thay đổi nào đến cấu trúc tệp sẽ đòi hỏi cũng phải có các thay đổi tương ứng đến các mô tả trong các chương trình ứng dụng sử dụng tệp đó. Ngoài ra, như trong hình 1.2.1.1, nếu tệp Customer bị sửa đổi thì cả hai hệ thống Order System và Invoice System đều bị ảnh hưởng.
- Lưu trữ trùng lặp dữ liệu: Trong hình 1.2.1.1 chúng ta nhận thấy rằng, hệ thống Order Filling có chứa một tệp Inventory Master, trong khi đó hệ thống Invoice cũng chứa một tệp là Inventory Pricing. Rõ ràng giữa

hai tệp này sẽ có sự lưu trữ trùng lặp dữ liệu, ví dụ các thông tin về mô tả sản phẩm, giá mỗi sản phẩm cũng như số lượng đang có trong kho. Việc lưu trữ dư thừa dữ liệu này gây ra rất nhiều bất cập và đòi hỏi không gian lưu trữ lớn, nó rất dễ dẫn tới tính không toàn vẹn của dữ liệu.

- Việc chia sẻ dữ liệu bị hạn chế: Trong cách tiếp cận xử lý tệp này, mỗi một ứng dụng có các tệp dữ liệu riêng của nó và do vậy giữa những người sử dụng, họ có ít cơ hội để có thể chia sẻ dữ liệu cho nhau nếu không sử dụng chung một phân hệ ứng dụng. Như trong hình 1.2.1.1, các nhân viên trong phòng kế toán Accounting chỉ có thể truy nhập được những tệp trong ứng dụng của mình mà không thể truy nhập được những tệp trong các ứng dụng của các phòng xử l;ý đơn hàng Order hay phòng quản lý tiền lương Payroll. Do vậy, khi người lãnh đạo muốn tạo ra những báo cáo tổng hợp liên phòng thì vô cùng khó khăn bởi dữ liệu được lưu riêng lẻ cũng như có thể không thống nhất giữa các phòng.
- Tốn thời gian xây dựng: Với mỗi một ứng dụng mới, người phát triển chương trình phải bắt đầu lại từ đầu với việc xây dựng các tệp, mô tả chúng cũng như xây dựng các module chương trình để truy nhập tới các tệp đó. Quá trình này tốn khá nhiều thời gian mặc dù nó luôn lặp lại giữa các hệ thống.
- Chi phí cho bảo trì chương trình lớn: Trên thực tế, có đến 80% chi phí được dành cho giai đoạn bảo trì chương trình đối với những ứng dụng dựa trên các hệ thống xử lý tệp truyền thống này.

Tuy nhiên chúng ta cũng cần phải lưu ý rằng, những hạn chế đối với cách tiếp cận dựa trên các hệ thống xử lý tệp truyền thống này cũng có thể gặp phải trong cách tiếp cận dựa trên cơ sở dữ liệu (sẽ được trình bày dưới đây), nếu như cơ sở dữ liệu không được thiết kế tốt, cũng như chúng được xây dựng một cách riêng rẽ, không thống nhất.

b. Cách tiếp cận cơ sở dữ liệu

Cách tiếp cận cơ sở dữ liệu nhấn mạnh đến tính chia sẻ và tích hợp của dữ liệu trong toàn bộ hoạt động của một doanh nghiệp (trong giáo trình này, chúng tôi sử dụng thuật ngữ doanh nghiệp để đại diện chung cho các tổ chức có nhu cầu sử dụng cơ sở dữ liệu trong hoạt động nghiệp vụ của mình). Theo cách tiếp cận này, dữ liệu được lưu trữ có tổ chức, tập trung dưới dạng một cơ sở dữ liệu và giữa các dữ liệu có sự liên kết logic với nhau.

Quay trở lại ví dụ của công ty chuyên sản xuất các sản phẩm đồ gỗ trên, sau một thời gian hoạt động, lãnh đạo của công ty nhận thấy rằng có rất nhiều bất cập cũng như hạn chế trong cách tiếp cận dựa trên hệ thống xử lý tệp, và họ quyết định chuyển sang cách tiệp cận dựa trên cơ sở dữ liệu. Khi đó họ phải tiến hành xây dựng lên một mô hình dữ liệu thống nhất, mô tả các hoạt động nghiệp vụ cho toàn bộ doanh nghiệp, và dựa trên mô hình dữ liệu này dần dần họ đã hình thành lên một cơ sở dữ liệu và được sử dụng thống nhất trong toàn bộ công ty.

Dưới đây chúng ta sẽ xem xét các ưu điểm mà cách tiếp cận dựa trên cơ sở dữ liệu mang lại so với cách tiếp cận dựa trên các hệ thống xử lý tệp truyền thống:

- Tối thiểu hoá dư thừa dữ liệu: Mục đích thiết kế trong cách tiếp cận cơ sở dữ liệu chính là cố gắng tích hợp các tệp dữ liệu riêng rẽ (và cũng là dư thừa) trong mô hình xử lý tệp trước đây vào một cấu trúc lôgic duy nhất. Mỗi một sự kiện cụ thể sẽ được lưu trữ tại một vị trí quy định trong cơ sở dữ liệu. Ví dụ với sự kiện sản phẩm có mã Product_id bằng 3 là một chiếc bàn để máy tính có giá là 170 USD sẽ được lưu trữ tại một vị trí nằm trong bảng PRODUCT. Cũng cần lưu ý rằng, cách tiếp cận cơ sở dữ liệu có thể không loại bỏ hoàn toàn vấn đề dư thừa dữ liệu, nhưng nó đã cho phép người thiết kế có thể kiểm soát được vấn đề này. Ví dụ, đối với mỗi đơn đặt hàng trong bảng ORDER có chứa một trường Customer_id để biểu diễn mối liên kết giữa các đơn đặt hàng và các khách hàng. Tuy nhiên trong một số trường hợp, để nâng cao hiệu quả truy vấn của cơ sở dữ liệu, chúng ta có thể chấp nhận hình thức lưu trữ dư thừa dữ liệu (vấn đề này sẽ được trình bày trong chủ đề 2 của giáo trình).
- Tính độc lập chương trình - dữ liệu: Tính độc lập với dữ liệu được thể hiện ở sự tách biệt rõ ràng giữa chương trình ứng dụng và các mô tả dữ liệu (siêu dữ liệu), do vậy nó cho phép dữ liệu của một doanh nghiệp được phép sửa đổi (tất nhiên là trong một giới hạn nào đó) mà không phải thay đổi lại chương trình thao tác trên các dữ liệu đó.
- Nâng cao tính nhất quán dữ liệu: Với việc loại bỏ (chính xác hơn là kiểm soát được) tính dư thừa dữ liệu, cách tiếp cận cơ sở dữ liệu đã giảm bớt được đáng kể tính không nhất quán của dữ liệu. Ví dụ, nếu

địa chỉ của khách hàng được lưu trữ duy nhất, khi đó, nếu khách hàng thay đổi địa chỉ, chúng ta chỉ việc làm một động tác đơn giản là sửa đổi lại địa chỉ của khách hàng ở vị trí duy nhất đó, các truy vấn liên quan đến giá trị này sẽ được tự động nhận giá trị mới khi được thực hiện. Như vậy, vấn đề về cập nhật, sửa đổi dữ liệu rất đơn giản và nhất quán. Một điểm quan trọng hơn, cách tiếp cận cơ sở dữ liệu sẽ làm giảm đáng kể không gian nhớ so với phương pháp lưu trữ dữ liệu trong các hệ thống xử lý tệp truyền thống.

- Nâng cao tính chia sẻ dữ liệu: cơ sở dữ liệu được thiết kế nhằm mục đích chia sẻ dữ liệu. Mỗi một người sử dụng hợp pháp được gán những quyền sử dụng cơ sở dữ liệu khác nhau, đối với mỗi một người (hoặc nhóm người) sử dụng sẽ được cung cấp những khung nhìn sử dụng khác nhau, do vậy họ có thể chủ động trong việc chia sẻ dữ liệu thông qua các quyền kiểm soát dữ liệu.
- Nâng cao tính sản phẩm của ứng dụng: Một lợi thế đáng kể của cách tiếp cận cơ sở dữ liệu đó là giảm đáng kể chi phí và thời gian cho việc phát triển những ứng dụng nghiệp vụ. Có hai lý do quan trọng mà các ứng dụng dựa trên cơ sở dữ liệu có thể được xây dựng nhanh chóng hơn so với cách tiếp cận của các ứng dụng dựa trên tệp truyền thống, đó là: khi xây dựng các ứng dụng mới dựa trên cơ sở dữ liệu, người lập trình không phải quan tâm đến vấn đề thiết kế các tệp hay các thao tác chi tiết mức thấp. Lý do thứ hai đó là các hệ quản trị cơ sở dữ liệu thường cung cấp một loạt các công cụ hỗ trợ mức cao trong quá trình

xây dựng ứng dụng, như các công cụ sản sinh mẫu biểu sinh báo cáo hay các giao diện nhập liệu cho chương trình ứng dụng.

- Áp đặt dễ dàng các tiêu chuẩn: Với cách tiếp cận cơ sở dữ liệu, các tiêu chuẩn như quy ước đặt tên, sử dụng qui trình thống nhất trong việc truy nhập, cập nhật và bảo vệ dữ liệu dễ dàng được thực hiện. Các chức năng quản trị cơ sở dữ liệu thường cung cấp một loạt các công cụ nhằm đảm bảo những tiêu chuẩn này luôn phải được thoả mãn. Tính chất này làm cho việc bảo trì và trao đổi dữ liệu trở nên dễ dàng và thuận lợi hơn.
- Nâng cao chất lượng dữ liệu: Với cách tiếp cận cơ sở dữ liệu, người thiết kế có thể xây dựng các ràng buộc toàn vẹn nhằm đảm bảo dữ liệu khi được đưa vào cơ sở dữ liệu phải thoả mãn một số qui tắc nào đó. Hoặc đối với những cơ sở dữ liệu lớn như kho dữ liệu, thì thông thường dữ liệu được làm sạch trước khi được đưa vào kho dữ liệu.
- Nâng cao khả năng truy nhập và trả lại dữ liệu: Với các cơ sở dữ liệu quan hệ, người sử dụng không cần nhiều kỹ năng về lập trình cũng có thể dễ dàng tìm kiếm và hiển thị dữ liệu thông qua một ngôn ngữ hỏi đáp dữ liệu rất thân thiện đã được tiêu chuẩn hoá.
- Giảm chi phí bảo trì chương trình: Tính chất này là hệ quả của tính độc lập chương trình - dữ liệu. Bởi chúng ta biết rằng dữ liệu có thể bị thay đổi vì những lý do khác nhau như: chúng ta có thể bổ sung thêm những kiểu dữ liệu mới, sửa đổi các định dạng cũng

như các kiểu dữ liệu đã có,... Tính độc lập chương trình - dữ liệu cho phép xây dựng chương trình tách biệt khỏi dữ liệu ở một mức độ nhất định, do vậy khi có sự sửa đổi, chúng ta chỉ việc sửa đổi một số module thao tác trên dữ liệu mà không nhất thiết phải sửa đổi lại toàn bộ chương trình. Chính điều này đã dẫn đến việc giảm bớt được chi phí bảo trì chương trình.

Trên đây là những ưu điểm mà cách tiếp cận cơ sở dữ liệu mang lại. Một cơ sở dữ liệu thường được tích hợp vào trong một hệ quản trị cơ sở dữ liệu, đó là một chương trình máy tính (hay chính xác hơn là một bộ các chương trình máy tính) được xây dựng để quản lý một cơ sở dữ liệu. Chúng ta sẽ tìm hiểu kỹ hơn về các hệ quản trị cơ sở dữ liệu trong chủ đề 4 của giáo trình.

Phần tiếp theo chúng ta sẽ tìm hiểu các khái niệm liên quan đến cơ sở dữ liệu.

2.2. Dữ liệu và thông tin

a. Dữ liệu

Như trên chúng ta đã nhắc đến dữ liệu, vậy thì dữ liệu thực sự là gì? Trong máy tính, thuật ngữ dữ liệu được xem như là các sự kiện được biết đến mà có thể ghi lại và lưu trữ trên các thiết bị ghi nhớ của máy tính.

Ví dụ trong cơ sở dữ liệu của người bán hàng nói trên, dữ liệu bao gồm các sự kiện như tên khách hàng, địa chỉ hay số điện thoại.

Ngày nay, ngoài những dữ liệu thông thường như số và ký tự, các cơ sở dữ liệu còn được sử dụng để lưu trữ các đối tượng đặc biệt hơn như văn bản, hình ảnh, âm thanh và thậm chí cả một đoạn phim. Ví dụ, với cơ sở dữ liệu của người bán hàng trên, anh ta có thể lưu ảnh của khách hàng, thậm chí có thể là các đoạn ghi âm hay băng hình ghi lại những cuộc nói chuyện với khách hàng gần đây nhất. Để có thể phản ánh được những thực tế này, chúng ta cần hiểu rộng hơn về khái niệm dữ liệu như sau:

Dữ liệu bao gồm các sự kiện, văn bản, hình ảnh, đồ hoạ, âm thanh, đoạn phim có một giá trị nào đó đối với người sử dụng chúng và được lưu trữ, xử lý trong máy tính.

Chúng ta đã định nghĩa cơ sở dữ liệu là một sưu tập các dữ liệu được tổ chức, liên quan logic với nhau. Từ được tổ chức ở đây ngụ ý rằng dữ liệu được cấu trúc sao cho chúng có thể dễ dàng được lưu trữ, thao tác, tìm kiếm và trả về cho người sử dụng. Còn từ liên quan trong định nghĩa ngụ ý rằng dữ liệu mô tả một lĩnh vực quan tâm của một người hoặc nhóm người sử dụng nào đó và do vậy họ có thể sử dụng chung để trả lời các câu hỏi liên quan đến lĩnh vực.

Ví dụ: một cơ sở dữ liệu của một cửa hàng sửa chữa xe ô tô có thể chứa các dữ liệu về khách hàng (bao gồm các khoản mục dữ liệu như tên khách hàng, địa chỉ, số điện thoại, thẻ tín dụng), các dữ liệu về xe ô tô của khách hàng (bao gồm các khoản dữ liệu như hãng xe, đời xe, năm sản xuất) và nhật ký các lần sửa chữa cho mỗi chiếc xe (như ngày phát sinh dịch vụ, tên thợ sửa, loại hình sửa chữa, và số tiền đã thanh toán).

b. Dữ liệu và thông tin

Hai thuật ngữ dữ liệu và thông tin liên quan chặt chẽ với nhau, trên thực tế chúng thường được sử dụng thay đổi cho nhau. Tuy nhiên, giữa chúng cũng có sự khác biệt và chúng ta nên phân biệt giữa dữ liệu và thông tin.

Chúng ta quan niệm thông tin là những gì được tạo nên nhằm giảm sự không xác định. Nói cách khác chúng ta có thể hiểu thông tin như là dữ liệu đã được xử lý theo cách mà chúng có thể làm tăng hàm lượng tri thức cho người sử dụng nó.

Ví dụ: Hoá đơn N=852 của nhà cung cấp A cần phải thanh toán chậm nhất là vào ngày 10-12-1999 đó là một thông tin.

Để thể hiện thông tin người ta sử dụng dữ liệu. Thông tin được thể hiện qua các giá trị của dữ liệu. Với ví dụ trên thông tin toàn bộ được thể hiện qua ba loại dữ liệu: đó là số thứ tự hoá đơn, tên nhà cung cấp và ngày giới hạn cho việc thanh toán.

Chúng ta hãy xem thêm ví dụ sau, đây là một danh sách các sự kiện:

Nguyễn Văn M.	3245213
Trần Đức T.	7214613
Mai Văn H.	8145184
Trịnh Thị M	5524812

Những sự kiện này rõ ràng thoả mãn định nghĩa của chúng ta về dữ liệu, tuy nhiên theo cách biểu diễn trên, hầu hết mọi người đều cho rằng những dữ liệu này chưa thể mang lại thông tin. Chúng ta có thể đoán rằng đây là một danh sách bao gồm tên người cùng với số bảo hiểm xã hội của

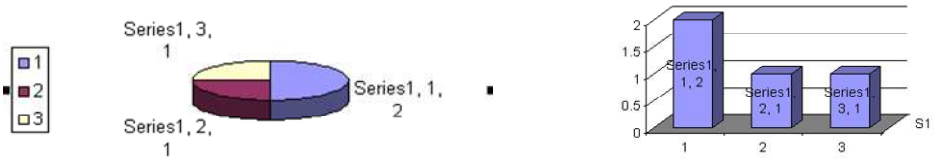
họ, nhưng thực sự chúng ta vẫn chưa có được thông tin gì về toàn bộ các số liệu này. Tuy nhiên, nếu đặt danh sách này vào một ngữ cảnh như trong hình 2.2.1a, với việc bổ sung thêm một số dữ liệu và cấu trúc của các dữ liệu đó, chúng ta sẽ nhận ra ngay rằng đó là một phần của bảng kết quả học tập cho một khoá học cụ thể. Như vậy chúng sẽ là hữu ích cho các nhân viên ghi danh hay những người quản lý lớp học chẳng hạn.

Bảng kết quả học tập

Course: KT2K38
Kinh tế
Semester: Học kỳ 2 năm 1999

Name	Id	Major	GPA
Nguyễn Văn M.	3245213	KTCT	7.5
Trần Đức T.	7214613	CNTT	6.8
Mai Văn H.	8145184	KTVM	8.2
Trịnh Thị L.	5524812	KTVM	6.9

(a) Dữ liệu được đặt trong ngữ cảnh



(b) Dữ liệu được tóm lược và biểu diễn dưới dạng trực quan

Hình 1.2.2.1 Chuyển đổi dữ liệu thành thông tin

Một cách khác để chuyển dữ liệu thành thông tin, đó là tóm lược chúng lại hoặc xử lý và trình bày chúng dưới các dạng trực quan hơn như đồ thị hay biểu đồ.

Trong hình 1.2.2.1b là các dữ liệu về kết quả học tập của các sinh viên đã được tóm lược và biểu diễn dưới dạng thông tin đồ hoạ. Rõ ràng dựa trên những thông tin này, việc quyết định có cần bổ sung thêm các khoá học mới hay thuê thêm giáo viên hay không sẽ dễ dàng được thực hiện hơn.

Trên thực tế, ngày nay các cơ sở dữ liệu có thể chứa đựng hoặc dữ liệu hoặc thông tin hoặc cả hai như trong định nghĩa chúng ta đã trình bày. Cũng có thể dữ liệu được xử lý trước và được lưu trữ dưới dạng một mẫu tóm lược trong cơ sở dữ liệu để phục vụ cho vấn đề hỗ trợ ra quyết định. Trong giáo trình này, chúng tôi sử dụng thuật ngữ cơ sở dữ liệu mà không phân biệt nội dung của nó là dữ liệu hay thông tin.

c. Siêu dữ liệu

Như chúng ta đã thấy, dữ liệu chỉ trở nên hữu ích khi chúng được đặt trong một ngữ cảnh nào đó. Cơ chế chính cung cấp ngữ cảnh cho dữ liệu đó chính là siêu dữ liệu, do vậy chúng ta có thể hiểu siêu dữ liệu là dữ liệu mô tả các tính chất hoặc đặc trưng của dữ liệu. Các tính chất này có thể là các dữ liệu về định nghĩa, về cấu trúc dữ liệu, về các luật hay các ràng buộc.

Quay trở lại ví dụ về bảng kết quả học tập trong hình 1.2.2.1a. Với mỗi khoản mục dữ liệu xuất hiện trong bảng, siêu dữ liệu cho biết các thông tin về tên, kiểu dữ liệu, độ dài, khoảng giá trị cho phép, và một số mô tả ngắn gọn đối với mỗi khoản mục dữ liệu đó.

Data Item			**Value**		
Name	**Type**	**Length**	**Mi**	**Max**	**Description**
Course	Charact	30			Course_id và
Section	Integer	1	1	9	Số hiệu học

Semeste	Charact	10			Học kỳ và
Name	Charact	30			Tên sinh viên
Id	Integer	9			Số hiệu sinh
Major	Charact	4			Môn học
GPA	Decimal	3	0.0	10.0	Điểm môn

Bảng 1.2.2.1: Ví dụ về một siêu dữ liệu cho bảng phân công lớp học

Chúng ta cần phải phân biệt rõ giữa dữ liệu và siêu dữ liệu: siêu dữ liệu là phần đã được tách khỏi dữ liệu. Cụ thể hơn, siêu dữ liệu mô tả các tính chất của dữ liệu nhưng không bao gồm dữ liệu. Do đó, siêu dữ liệu trong bảng 1.2.2.1 không bao gồm bất cứ một bộ dữ liệu nào từ bảng kết quả học tập trong hình 1.2.2.1a.

Siêu dữ liệu cho phép người thiết kế cơ sở dữ liệu và người sử dụng biết có dữ liệu gì, tồn tại ở đâu, các khoản mục dữ liệu khác nhau ở điểm gì. Việc quản lý siêu dữ liệu cũng quan trọng như việc quản lý dữ liệu liên quan với nó, bởi dữ liệu nếu không được phân biệt rõ ràng sẽ gây ra nhầm lẫn, sai sót và không thể diễn giải nổi. Trong cơ sở dữ liệu siêu dữ liệu thường được lưu trữ trong một vùng qui định mà chúng ta thường gọi là từ điển dữ liệu (data dictionary).

Dữ liệu tạo nên nội dung của cơ sở dữ liệu, tuy nhiên nó không quyết định tính hiệu quả của cơ sở dữ liệu. Bởi tính hiệu quả được thể hiện qua việc thiết kế và khai thác cơ sở dữ liệu, chúng ta cần làm quen với những khái niệm tiếp theo bởi chúng sẽ được sử dụng thường xuyên trong quá trình phân tích, thiết kế cơ sở dữ liệu.

2.3. Kiểu thực thể và bảng dữ liệu

a. Thực thể

Thực thể là một khái niệm được sử dụng thường xuyên trong quá trình phân tích, thiết kế cơ sở dữ liệu. Vậy thực thể là gì?

Chúng ta nhận thấy rằng trong thế giới thực, tồn tại rất nhiều các đối tượng khác nhau, ví dụ như một người trong một tổ chức, một sinh viên trong một trường đại học, hay thậm chí một phòng ban trong một viện nghiên cứu, tất cả những đối tượng đó đều là các thực thể khác nhau. Ngoài ra những đối tượng trừu tượng như một ngành khoa học, chúng ta cũng có thể coi là một thực thể.

Vậy ta có thể định nghĩa thực thể như là một đối tượng cụ thể hay trừu tượng trong thế giới thực mà nó tồn tại và có thể phân biệt với các đối tượng khác.

Rõ ràng một sinh viên Nguyễn Văn A là một thực thể được phân biệt hoàn toàn với một sinh viên Trần Thị B cũng là một thực thể.

b. Kiểu thực thể

Từ định nghĩa về thực thể nêu trên, chúng ta nhận thấy rằng trong thế giới thực có rất nhiều các thực thể giống nhau ở một số khía cạnh nào đó. Như các sinh viên trong một trường đại học đều là các thực thể, và họ có chung một số điểm đó là đều có mã hiệu sinh viên, có tên, ngày nhập học, số hiệu lớp. Hay với các phòng ban trong một viện nghiên cứu cũng đều là các thực thể, giữa các thực thể này đều có một điểm chung đó là chúng đều có tên, có một nhân viên

làm trưởng phòng, cũng như có một số lượng các nhân viên đang làm việc trong phòng.

Như vậy, chúng ta có thể coi một nhóm các thực thể giống nhau hình thành lên một kiểu thực thể hay còn gọi là tập thực thể.

Ví dụ: tất cả các sinh viên trong một trường đại học tạo thành một kiểu thực thể là STUDENT. Tất cả nhân viên của một doanh nghiệp hình thành lên một kiểu thực thể là EMPLOYEE. Tất cả các phòng ban trong một viện nghiên cứu tạo thành một kiểu thực thể DEPARTMENT.

Tuy nhiên chúng ta cũng nhận thấy rằng khái niệm "thực thể giống nhau" không được xác định một cách chính xác và người ta có thể đưa ra một số vô hạn các tính chất khác nhau để hình thành lên một kiểu thực thể, điều này phụ thuộc vào nhu cầu quản lý của bài toán. Như trong ví dụ về kiểu thực thể nhân viên EMPLOYEE, nếu chúng ta muốn phân biệt chi tiết hơn nữa như theo loại nhân viên là làm việc hưởng lương theo ngày công, hay làm việc hưởng lương theo hợp đồng, khi đó chúng ta phải phân nhóm tiếp và sẽ hình thành lên hai kiểu thực thể cụ thể hơn đó là kiểu thực thể nhân viên hưởng lương theo ngày EMPLOYEE_DAY và kiểu thực thể nhân viên hưởng lương theo hợp đồng EMPLOYEE_CONTRACT.

Vậy chúng ta có thể hiểu một kiểu thực thể như là một nhóm các thực thể giống nhau ở một số điểm nhất định nào đó mà chúng ta cần quan tâm.

Một trong các bước chính trong quá trình phân tích thiết kế cơ sở dữ liệu đó là: xây dựng một sơ đồ đối với thế giới thực, nói một cách cụ thể hơn tức là lựa chọn các kiểu thực

thể cần phải mô tả và mỗi quan hệ giữa các kiểu thực thể đó.

Chúng ta cần phải phân biệt sự khác nhau giữa kiểu thực thể và thể hiện của thực thể (hay thực thể):

- Một kiểu thực thể là là một tập các thực thể có chung một số tính chất hoặc đặc trưng nào đó. Mỗi một kiểu thực thể phải được đặt một tên duy nhất không trùng với các kiểu thực thể đã có.
- Còn thể hiện của thực thể là một đơn xuất hiện của kiểu thực thể.

Kiểu thực thể: EMPLOYEE

Các thuộc tính:

Emp_id	**Char(10)**
Emp_name	**Char(50)**
Address	**Char(50)**
Date_hired	**Date**
Birth_date	**Date**

Và hai thể hiện:

642-17-125	**781-78-157**
Nguyễn Văn V.	**Hoàng Xuân T.**
10A Nguyễn Huệ	**102 Lê Duẩn**
21/01/2000	**19/08/1997**
19/06/1968	**20/12/1958**

Hình 1.2.3.1 Kiểu thực thể EMPLOYEE và 2 thể hiện thực thể của nó

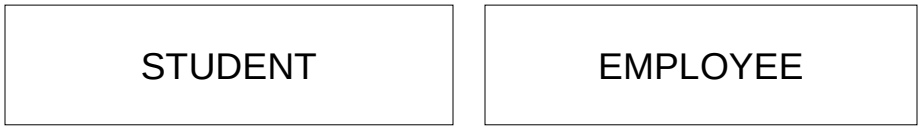
Hình 1.2.3.1 trên minh hoạ sự khác nhau giữa kiểu thực thể EMPLOYEE và 2 thể hiện thực thể của nó.

Khi phân tích thiết kế một cơ sở dữ liệu, một kiểu thực thể chỉ được mô tả một lần (sử dụng siêu dữ liệu), trong khi đó các thể hiện của kiểu thực thể đó lại được biểu diễn bởi dữ liệu lưu trong cơ sở dữ liệu.

Ví dụ có một kiểu thực thể là EMPLOYEE trong hầu hết các tổ chức, tuy nhiên có hàng trăm, thậm chí hàng nghìn thể hiện của kiểu thực thể đó được lưu trong cơ sở dữ liệu.

Trong giáo trình này, chúng tôi sẽ sử dụng thuật ngữ thực thể thay cho thuật ngữ thể hiện thực thể nếu ý nghĩa của nó là rõ ràng trong ngữ cảnh trình bày.

Để đặt tên cho kiểu thực thể, chúng ta dùng các danh từ viết liền với các chữ cái được viết hoa, bao quanh bởi một hình chữ nhật như hình 1.2.3.2 dưới đây:



Hình 1.2.3.2 Kiểu thực thể

Các kiểu thực thể được đặc tả bởi một tập các đặc trưng mà chúng ta qui ước gọi là các thuộc tính (như chúng ta sẽ thấy dưới đây). Do vậy với các thành viên tham gia vào một kiểu thực thể đòi hỏi chúng ta phải xác định được tập các thuộc tính chúng nhất giữa chúng.

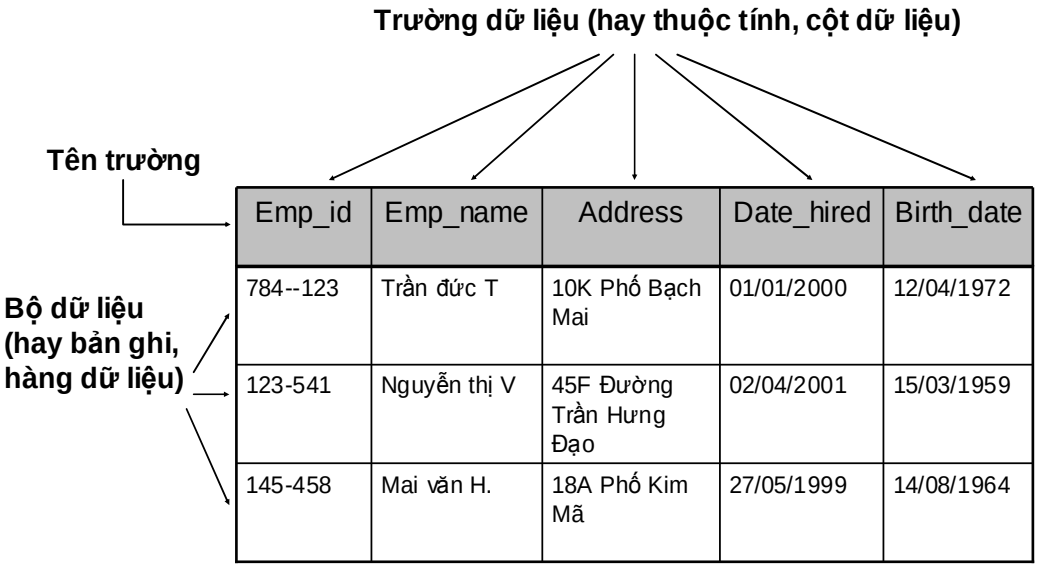
c. Bảng dữ liệu hay quan hệ

Trong quá trình phân tích cơ sở dữ liệu, chúng ta sử dụng khái niệm kiểu thực thể để minh hoạ cho một lớp các thực thể giống nhau. Bảng dữ liệu chính là hình thức thể hiện cụ

thể của kiểu thực thể khi chúng ta xây dựng cơ sở dữ liệu, và bảng dữ liệu được sử dụng để lưu dữ liệu về các thực thể trong lớp thực thể đó.

Chúng ta có thể hình dung bảng dữ liệu là một tập các bộ dữ liệu, hay bản ghi dữ liệu, mỗi bộ có cùng một số lượng thuộc tính như nhau nhưng có thể khác nhau về giá trị. Bảng dữ liệu trong cách tiếp cận cơ sở dữ liệu quan hệ, được hiểu chính xác hơn bởi cụm từ quan hệ. Một quan hệ bao gồm một lược đồ quan hệ và một thể hiện quan hệ. Thể hiện quan hệ chính là một bảng, còn lược đồ quan hệ mô tả tiêu đề các cột của bảng đó. Trong một quan hệ không thể tồn tại hai bộ dữ liệu giống nhau ở tất cả các thuộc tính. Hình 1.2.3.3 dưới đây minh hoạ cho một bảng dữ liệu, các cột của bảng tương ứng với các thuộc tính hay các trường dữ liệu, các hàng của bảng tương ứng với các bản ghi hay các bộ dữ liệu.

(Khái niệm về thuộc tính sẽ được trình bày trong phần sau).



Emp_id	Emp_name	Address	Date_hired	Birth_date
784--123	Trần đức T	10K Phố Bạch Mai	01/01/2000	12/04/1972
123-541	Nguyễn thị V	45F Đường Trần Hưng Đạo	02/04/2001	15/03/1959
145-458	Mai văn H.	18A Phố Kim Mã	27/05/1999	14/08/1964

Hình 1.2.3.3 Bảng dữ liệu hay quan hệ

Đôi khi chúng ta có thể biểu diễn các bảng dưới dạng loại bỏ các dữ liệu và chỉ đưa ra tên bảng cùng các tên cột như sau:

AUTHOR (Au_id, Au_lname, Au_fname, Address, City, Country)

TITLE (Title_id, Title, type, Price, Pub_id)

PUBLISHER (Pub_id, Pub_name, City)

AUTHOR_TITLE (Au_id, Title_id)

Mỗi bảng hay quan hệ có một số tính chất sau:

- Các dữ liệu trong bảng là các giá trị đơn, được hiểu theo nghĩa là các giá trị không cần chia nhỏ hơn nữa mà vẫn đáp ứng được nhu cầu nghiệp vụ. Ví dụ: thiết kế cơ sở dữ liệu để lưu địa chỉ của các nhân viên

trong một công ty, nếu chúng ta không cần quan tâm đến việc phân loại các nhân viên theo thành phố, quận huyện, phường xã thì với một giá trị như "Nhà số 6 phường Kim Mã quận Ba Đình thành phố Hà Nội" được chấp nhận như một giá trị đơn. Mặt khác, trong trường hợp bài toán yêu cầu quản lý địa chỉ của nhân viên đến từng cấp thành phố, quận huyện, phường xã, số nhà thì rõ ràng địa chỉ trên không phải là giá trị đơn.

- Các giá trị trong cùng một cột dữ liệu phải có cùng một kiểu. Ví dụ nếu một cột được qui định là lưu ngày vào làm việc của các nhân viên trong công ty thì cột đó không thể lưu các giá trị khác kiểu ngày tháng như tên nhân viên, tuổi tác,..
- Mỗi một hàng trong bảng là duy nhất. Trong bảng không thể tồn tại hai hàng có cùng giá trị ở tất cả các cột. Ví dụ, nếu một bảng qui ước là lưu thông tin của các nhân viên trong công ty thì không thể có một nhân viên được lưu hai lần trong cùng bảng đó.
- Thứ tự liệt kê các cột trong bảng không quan trọng.
- Thứ tự liệt kê các hàng trong bảng không quan trọng
- Mỗi một cột dữ liệu trong bảng phải có một tên duy nhất

Như trên chúng ta thấy một quan hệ là một bảng hai chiều gồm các giá trị, đó là tập hợp các bộ của quan hệ tại một thời điểm nào đó. Tuy nhiên, quan hệ lại thường được cập nhật (bổ sung, loại bỏ, điều chỉnh các bộ), do đó quan hệ có tính chất là thay đổi theo thời gian.

Tuy thay đổi về giá trị thông qua việc cập nhật, song quan hệ vẫn phải giữ một số đặc điểm về ngữ nghĩa, hay nói một cách khác nó phải tuân thủ một lược đồ quan hệ nhất đinh, như qui ước số trường trong quan hệ, kiểu dữ liệu trong quan hệ,... Chúng ta có thể hình dung một lược đồ quan hệ là một khuôn mẫu, hay nói đúng hơn là một cái lọc, cho phép thiết lập một họ các quan hệ cùng ngữ nghĩa. Trong giáo trình này, đôi lúc chúng tôi sẽ dùng thuật ngữ quan hệ thay cho thuật ngữ lược đồ quan hệ nếu ngữ cảnh trình bày cho phép.

2.4. Thuộc tính và trường dữ liệu

a. Thuộc tính

Như trong khái niệm về kiểu thực thể chúng ta đã biết, một kiểu thực thể được mô tả bằng cách sử dụng một tập các thuộc tính. Tất cả các thực thể trong một kiểu thực thể đã được xác định đều có cùng số lượng các thuộc tính đó. Chúng ta cần lưu ý rằng, việc lựa chọn các thuộc tính phản ánh mức độ chi tiết mà chúng ta muốn biểu diễn thông tin về các thực thể của kiểu thực thể đó.

Ví dụ với kiểu thực thể EMPLOYEE, chúng ta có thể lựa chọn các thuộc tính như số thẻ bảo hiểm xã hội, tên, địa chỉ, ngày bắt đầu làm việc. Tuy nhiên trong một ngữ cảnh hay bài toán khác, chúng ta có thể quan tâm thêm đến các thuộc tính khác như giới tính, tuổi hay địa chỉ của nhân viên.

Do vậy chúng ta có thể xem thuộc tính như là một tính chất hay một đặc trưng của một kiểu thực thể mà người sử dụng quan tâm.

Mỗi một kiểu thực thể luôn có một tập các thuộc tính đi kèm. Dưới đây là ví dụ về hai kiểu thực thể và các thuộc tính đi kèm với chúng:

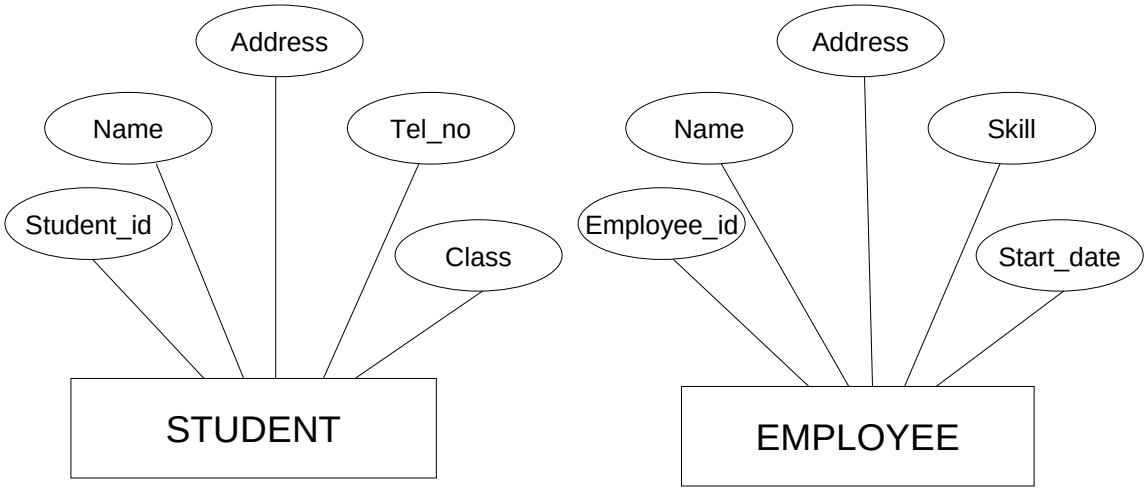
STUDENT: Student_id, Student_name, Address, Tel_no, Class

EMPLOYEE: Emp_id, Emp_name, Address, Skill, Start_date

Với mỗi thuộc tính trong kiểu thực thể, chúng ta phải xác định miền giá trị của thuộc tính đó. Ví dụ miền giá trị cho thuộc tính Emp_name của kiểu thực thể EMPLOYEE có thể là một xâu ký tự có độ dài không quá 30 ký tự. Một ví dụ khác, nếu công ty của chúng ta muốn xếp loại nhân viên theo các kỹ năng (tay nghề) từ 1 đến 7, thì miền giá trị cho thuộc tính Skill sẽ là các số nguyên có giá trị nằm trong khoảng từ 1 đến 7.

Thể hiện của thuộc tính chính là các giá trị. Ví dụ "Nguyên văn A." là một giá trị cho thuộc tính tên nhân viên Emp_name.

Để đặt tên cho các thuộc tính, chúng ta dùng các danh từ viết liền với chữ cái đầu tiên được viết hoa, nếu một thuộc tính có tên gồm 2 từ thì chúng ta sử dụng dấu gạch dưới để nối 2 từ đó lại. Chúng ta sử dụng hình elip để biểu diễn cho một thuộc tính và một đường thẳng kết nối thuộc tính đó với kiểu thực thể mà nó mô tả, ví dụ như hình 1.2.4.1 sau là hai kiểu thực thể và các thuộc tính của chúng.



Hình 1.2.4.1 Thuộc tính và kiểu thực thể

b. Trường dữ liệu

Cũng như giữa bảng dữ liệu và kiểu thực thể, khi phân tích cơ sở dữ liệu, chúng ta sử dụng các thuộc tính để mô tả cho các kiểu thực thể, còn khi xây dựng cơ sở dữ liệu, chúng ta sử dụng các trường dữ liệu để xây dựng lên các quan hệ. Do vậy chúng ta có thể hiểu trường dữ liệu như là đơn vị nhỏ nhất có tên gọi của dữ liệu lưu trữ trong cơ sở dữ liệu. Trường dữ liệu còn được gọi là cột dữ liệu khi chúng ta sử dụng với khái niệm bảng dữ liệu.

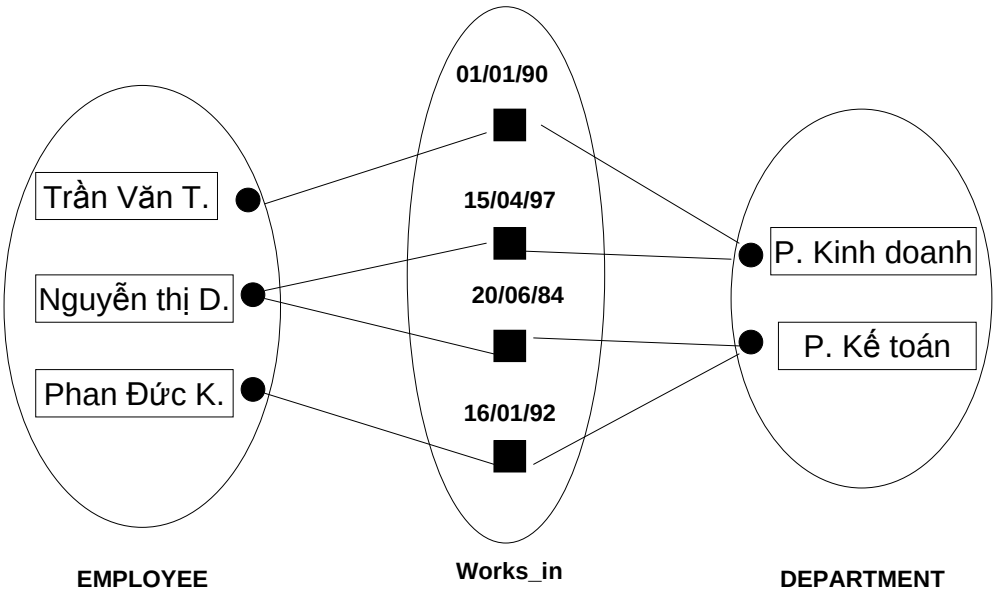
Qua đó chúng ta có thể ánh xạ các khái niệm trong khâu phân tích cơ sở dữ liệu và các khái niệm được sử dụng trong khâu thiết kế, xây dựng cơ sở dữ liệu:

- Từ kiểu thực thể chúng ta xây dựng được lên các quan hệ hay bảng dữ liệu.
- Từ các thuộc tính chúng ta xây dựng được lên các trường hay cột dữ liệu.

2.5. Liên kết giữa các thực thể

Một liên kết là một kết nối giữa các thể hiện của một hoặc nhiều kiểu thực thể. Ví dụ, chúng ta có một liên kết như nhân viên Trần Văn T. làm việc ở phòng kinh doanh; Sinh viên Nguyễn Văn A. tham gia các khoá học về C++, Visual Basic.

Chúng ta cũng có thể sử dụng các thuộc tính để mô tả cho liên kết. Các thuộc tính mô tả được sử dụng để lưu lại các thông tin về mối liên kết hơn là những thông tin về các thực thể tham gia vào liên kết đó. Ví dụ, chúng ta có thể ghi lại nhân viên Trần Văn T. làm việc ở phòng kinh doanh từ ngày 01/01/1990 thông qua một thuộc tính Start_date được đặt bên cạnh liên kết Works_in như trong hình 1.2.5.1 dưới đây:



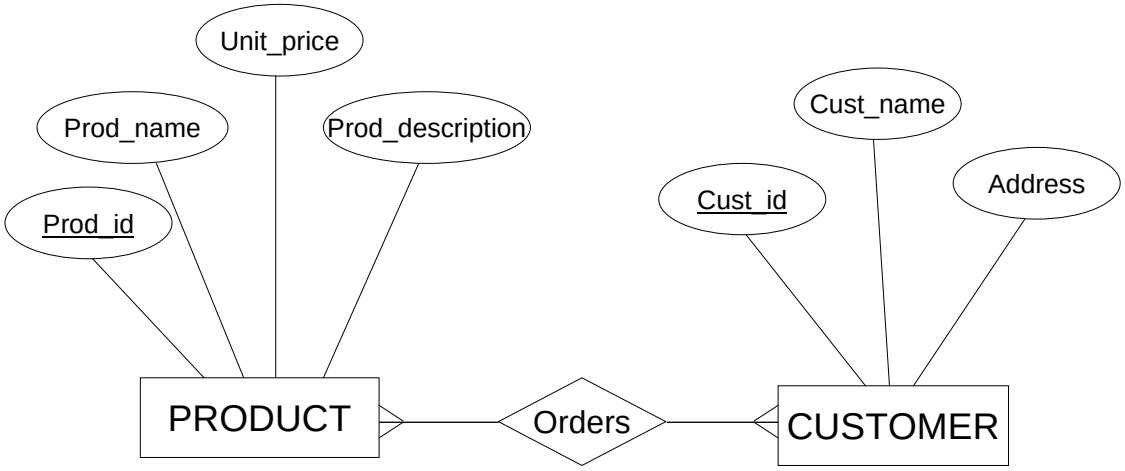
Hình 1.2.5.1 Liên kết giữa các thực thể

Một liên kết phải được xác định duy nhất bởi các thực thể tham gia liên kết mà không cần phải tham chiếu đến các thuộc tính mô tả. Như trong hình trên, mỗi một liên kết trong Works_in được xác định duy nhất bởi việc kết hợp thuộc tính đại diện Emp_name của kiểu thực thể EMPLOYEE và thuộc tính đại diện Dept_name của kiểu thực thể DEPARTMENT. Do vậy với mỗi cặp nhân viên – phòng ban trên chúng ta không thể có hơn một kết nối.

(Chú ý: trong ví dụ trên chúng ta giả thiết rằng giữa các thực thể của kiểu thực thể có thể phân biệt duy nhất với nhau thông qua các giá trị trong các thuộc tính Emp_name và Dept_name tương ứng với hai kiểu thực thể EMPLOYEE và DEPARTMENT).

Một nhóm các liên kết cùng kiểu hình thành lên một tập liên kết. Khi dùng với khái niệm kiểu thực thể, trong giáo trình này chúng tôi dùng thuật ngữ liên kết để mô tả cho sự kết nối giữa chúng thay vì sử dụng thuật ngữ tập liên kết.

Để đặt tên cho liên kết, chúng ta dùng các động từ viết liền với chữ cái đầu tiên được viết hoa, nếu một động từ có tên gồm 2 từ thì chúng ta sử dụng dấu gạch dưới để nối 2 từ đó lại. Chúng ta sử dụng hình thoi để biểu diễn cho một liên kết và các đường nối liên kết đó với các kiểu thực thể tham gia vào liên kết. Một ví dụ được cho như hình 1.2.5.2 dưới đây:



Hình 1.2.5.2 Liên kết giữa các kiểu thực thể

Hình 1.2.5.2 trên mô tả cho liên kết đặt mua "Orders" giữa hai kiểu thực thể là khách hàng CUSTOMER và mặt hàng PRODUCT với ý nghĩa: mỗi khách hàng có thể đặt mua một hoặc nhiều mặt hàng, ngược lại mỗi một mặt hàng cũng có thể được nhiều khách hàng khác nhau mua. Kiểu kết nối giữa liên kết "Orders" trên với hai kiểu thực thể còn lại sẽ được trình bày cụ thể trong chủ đề hai của giáo trình.

2.6. Khái niệm toàn vẹn dữ liệu

Như đã trình bày trong phần đầu, một trong những ưu điểm của cách tiếp cận cơ sở dữ liệu đó là dữ liệu khi được đưa vào cơ sở dữ liệu sẽ phải tuân theo một số qui định nào đó. Những qui định này được xem là những ràng buộc trên dữ liệu và nó đảm bảo tính toàn vẹn của dữ liệu. Tính toàn vẹn dữ liệu giúp cho người sử dụng có thể tin cậy ít nhất về mặt nguyên tắc đối với các dữ liệu trong cơ sở dữ liệu.

Chúng ta cũng cần lưu ý rằng toàn vẹn dữ liệu có thể đảm bảo tính hợp lệ của dữ liệu khi được lưu trong cơ sở dữ liệu,

nhưng lại khó có thể kiểm tra mỗi một giá trị như vậy về tính hợp lý. Ví dụ, khi nhập giá trị giờ làm việc của một nhân viên trong tuần, không có cách nào có thể phát hiện sự kiện là một giá trị vào bằng 35 là sai, trong khi đó thực sự phải là giá trị 38. Tuy nhiên, nếu một giá trị là 350 đối với giờ làm việc rõ ràng là sai và hệ thống sẽ không chấp nhận giá trị đó, bởi một tuần không thể có 350 giờ.

Giữ gìn tính toàn vẹn của cơ sở dữ liệu có thể xem là sự bảo vệ dữ liệu trong cơ sở dữ liệu chống lại sự cập nhật hay phá hủy không có căn cứ. Chú ý, chúng ta cũng phải phân biệt tính toàn vẹn dữ liệu với tính an toàn dữ liệu dù rằng hai khái niệm này quan hệ rất mật thiết với nhau và thực sự có thể dùng chung một cơ chế để thực hiện bảo vệ cả hai ở một mức độ nào đó.

Nói chung mỗi một bảng dữ liệu trong cơ sở dữ liệu quan hệ đều có một tập các ràng buộc toàn vẹn đi kèm với nó. Các ràng buộc toàn vẹn này cũng được lưu trữ trong từ điển dữ liệu.

Chúng ta sẽ dành cho việc tìm hiểu sâu hơn tính toàn vẹn của dữ liệu trong các chủ đề tiếp theo.

3. Ví dụ về một cơ sở dữ liệu

Trong giáo trình này, chúng tôi cố gắng đưa ra các ví dụ minh hoạ xoay quanh một bài toán xây dựng cơ sở dữ liệu cho một công ty chuyên kinh doanh đồ gỗ, chúng ta tạm đặt tên cho công ty này là Pine Furniture:

Hoạt động chung của công ty này như sau: công ty mua các nguyên vật liệu (hay bán thành phẩm) từ một số nhà cung cấp khác nhau, sau khi nhận được các nguyên vật liệu này,

công ty sẽ tiến hành sản xuất để tạo nên các sản phẩm theo các đơn đặt hàng, và gửi đến các khách hàng đã đăng ký mua các sản phẩm đó thông qua các đơn đặt hàng. Mỗi một khách hàng có thể đặt mua một hoặc nhiều loại sản phẩm khác nhau trong cùng một đơn đặt hàng.

Một số khái niệm:

- Mỗi một khách hàng CUSTOMER của công ty có thể là một cá nhân hay một tổ chức nào đó có nhu cầu đặt mua các sản phẩm của công ty. Ví dụ: công ty kinh doanh thiết bị tin học AITC muốn đặt mua 15 chiếc bàn mã hiệu MH0415 của công ty.
- Sản phẩm PRODUCT là một mặt hàng bán ra cụ thể của công ty mà khách hàng có thể đặt mua thông qua các đơn đặt hàng. Ví dụ: sản phẩm bàn máy tính mã hiệu MH0415 có kích thước chiều ngang là 2m, chiều dọc là 1.2m, chiều cao là 0.9m và được làm từ gỗ thông.
- Đơn đặt hàng ORDER là một giao dịch giữa công ty và khách hàng. Mỗi một đơn đặt hàng đều có một số hiệu duy nhất, trong mỗi đơn đặt hàng sẽ phải thể hiện được các thông tin sau: khách hàng nào đã đặt, họ đặt mua những sản phẩm nào, số lượng tương ứng cho mỗi loại sản phẩm đó và ngày đặt của đơn hàng. Ví dụ: Khách hàng là công ty A đã đặt mua các sản phẩm: giá sách loại 0600 ba chiếc; bàn làm việc loại C700 hai chiếc vào ngày 1/7/2003, số hiệu của hoá đơn đó là HD1845.
- Các nguyên vật liệu bán thành phẩm ITEM là các thành phần được sử dụng để tạo thành các sản phẩm.

Các nguyên vật liệu này sẽ do các nhà cung cấp SUPPLIER gửi đến. Ví dụ, thanh gỗ loại G200 có thể được mua từ hai nhà cung cấp là North Wood và Star Wood.

- Nhà cung cấp SUPPLIER là các công ty chuyên cung cấp các vật liệu bán thành phẩm ITEM làm nguyên liệu sản xuất cho công ty.

Một số quy tắc nghiệp vụ:

- Mỗi một nhà cung cấp SUPPLIER có thể cung cấp một hoặc nhiều nguyên liệu bán thành phẩm ITEM. Và mỗi một nguyên vật liệu ITEM phải được cung cấp ít nhất từ một nhà cung cấp.
- Mỗi một nguyên vật liệu ITEM phải được sử dụng trong ít nhất một sản phẩm PRODUCT, và tất nhiên nó thường được sử dụng trong nhiều sản phẩm.
- Một nhà cung cấp SUPPLIER có thể có nhiều giao dịch với công ty. Ngược lại, mỗi một giao dịch mua hàng chỉ được gửi đến chính xác từ một nhà cung cấp.
- Trong mỗi giao dịch với nhà cung cấp phải bao gồm một hoặc nhiều nguyên vật liệu ITEM. Và mỗi một nguyên vật liệu có thể xuất hiện trong nhiều giao dịch khác nhau.
- Một khách hàng CUSTOMER có thể đăng ký mua các sản phẩm thông qua các đơn đặt hàng ORDER. Tuy nhiên mỗi một đơn đặt hàng phải do một khách hàng duy nhất đặt.

- Trong một đơn đặt hàng ORDER, khách hàng có thể đặt mua một hoặc nhiều sản phẩm PRODUCT tạo thành những dòng đơn hàng. Và mỗi một sản phẩm có thể được đặt mua trong một hoặc nhiều đơn hàng ORDER, thậm chí có những sản phẩm không được đặt mua ở bất kỳ một đơn đặt hàng nào.

Bài toán này sẽ được sử dụng cho một số ví dụ trong chủ đề hai của giáo trình.

Trước khi chuyển sang chủ đề hai của giáo trình, trong phần sau chúng ta hãy xem xét qua một số mô hình dữ liệu.

4. Các mô hình dữ liệu

4.1. Khái quát chung

Có thể nói mô hình dữ liệu là cách biểu diễn các cấu trúc dữ liệu cho một cơ sở dữ liệu dưới dạng các khái niệm. Các cấu trúc dữ liệu bao gồm các đối tượng dữ liệu, các kết nối giữa chúng và các ràng buộc trên các đối tượng dữ liệu đó.

Bản thân thuật ngữ "mô hình dữ liệu" đã cho chúng ta thấy mô hình chỉ tập trung vào việc mô tả các dữ liệu và chúng sẽ được tổ chức như thế nào chứ không tập trung vào các thao tác sẽ được thực hiện trên các dữ liệu đó (được thực hiện bởi mô hình chức năng).

Một mô hình dữ liệu không phụ thuộc vào các ràng buộc về cả phần cứng và phần mềm. Thay vì biểu diễn dữ liệu như là một cơ sở dữ liệu, mô hình dữ liệu chủ yếu tập trung vào việc biểu diễn các dữ liệu sao cho người sử dụng có thể "nhìn thấy" được chúng trong thế giới thực.

Mô hình dữ liệu đóng vai trò như là một cầu nối giữa các khái niệm được sử dụng để mô tả cho các sự kiện và các quá trình trong thế giới thực với cách biểu diễn các khái niệm này trong một cơ sở dữ liệu.

Có hai phương pháp luận chính được sử dụng để tạo nên một mô hình dữ liệu: đó là phương pháp dựa trên cách tiếp cận thực thể - liên kết và phương pháp mô hình hoá đối tượng. Trong giáo trình này chúng ta chỉ tập trung vào cách tiếp cận thứ nhất, cách tiếp cận thực thể - liên kết bởi cơ sở dữ liệu được tập trung trình bày trong giáo trình là cơ sở dữ liệu quan hệ.

Trong quá trình phân tích, thiết kế và xây dựng cơ sở dữ liệu, có thể nói quá trình xây dựng mô hình dữ liệu chiếm nhiều thời gian và công sức nhất. Bởi mục đích của nó là đảm bảo chắc chắn rằng tất cả các đối tượng dữ liệu trong bài toán phải được trình bày đầy đủ và chính xác. Chúng ta có thể sử dụng các ngôn ngữ một cách rất tự nhiên cũng như các ký hiệu minh hoạ dễ hiểu để mô tả cho các đối tượng dữ liệu, do đó người sử dụng cuối cùng có thể dễ dàng hiểu và cho ý kiến về mô hình. Mặt khác người thiết kế cơ sở dữ liệu cũng có thể sử dụng mô hình dữ liệu như một bản thiết kế chi tiết để có thể tiến hành thiết kế một cơ sở dữ liệu mức vật lý.

Dưới đây chúng ta sẽ xem xét một số mô hình dữ liệu, trong đó đáng lưu ý nhất là mô hình dữ liệu quan hệ. Mặc dù được xây dựng khá lâu, nhưng đến nay mô hình dữ liệu quan hệ vẫn được xem như là một mô hình quan trọng nhất trong tất cả các hệ quản trị cơ sở dữ liệu phổ biến.

Ngoài mô hình dữ liệu quan hệ, chúng ta cũng sẽ lướt qua một số mô hình dữ liệu khác như mô hình dữ liệu mạng,

mô hình dữ liệu phân cấp là những “tiền thân” của mô hình dữ liệu quan hệ.

4.2. Mô hình dữ liệu quan hệ

Mô hình dữ liệu quan hệ được Codd đề xuất năm 1970, được xem là một mô hình có cơ sở toán học vững chắc nhất. Tại thời điểm đó, hầu hết các hệ cơ sở dữ liệu đều được xây dựng dựa trên hai mô hình trước đó là mô hình dữ liệu phân cấp và mô hình dữ liệu mạng. Sau khi mô hình dữ liệu quan hệ ra đời, nó đã tạo ra một cuộc cách mạng mới trong lĩnh vực cơ sở dữ liệu và nhanh chóng thay thế các mô hình dữ liệu trước đó.

Ngày nay, mô hình dữ liệu quan hệ vẫn tiếp tục đóng một vai trò thống trị và là nền tảng cho hàng loạt các hệ quản trị cơ sở dữ liệu nổi tiếng và phổ biến như Oracle, DB2, MS SQLServer, Access, ... Các hệ cơ sở dữ liệu quan hệ này có mặt ở khắp mọi nơi và đã trở thành một trong những ngành công nghiệp có doanh thu cao nhất.

Mô hình dữ liệu quan hệ tương đối đơn giản và dễ hiểu. Một cơ sở dữ liệu quan hệ là một tập của một hoặc nhiều quan hệ, trong đó mỗi một quan hệ là một bảng với các hàng và các cột. Cách biểu diễn đơn giản theo bảng gồm các hàng và cột này khiến cho người mới làm quen với cơ sở dữ liệu cũng có thể hiểu được nội dung của cơ sở dữ liệu. Hơn thế nữa, mô hình dữ liệu quan hệ đã cung cấp các khái niệm chặt chẽ được hình thức hoá cao, cho phép áp dụng các công cụ toán học, các thuật toán tối trên dữ liệu.

Mô hình dữ liệu quan hệ được trừu tượng hoá cao nhưng nó chỉ dừng ở mức logic, nghĩa là độc lập với mức vật lý, với

sự cài đặt, với các thiết bị lưu trữ. Nhờ đó, mô hình dữ liệu quan hệ đảm bảo được tính độc lập giữa dữ liệu và chương trình ứng dụng. Ngoài ra, do sự biểu diễn dữ liệu thống nhất, mô hình dữ liệu quan hệ có thể cho phép sử dụng các ngôn ngữ thao tác dữ liệu ở mức cao, dễ sử dụng và có thể chuẩn hoá. Đây cũng chính là một trong những ưu điểm lớn nhất của một mô hình dữ liệu quan hệ so với các mô hình dữ liệu trước đây

4.3. Một số mô hình dữ liệu khác

a. Mô hình dữ liệu mạng và mô hình dữ liệu phân cấp

Trong mô hình mạng và mô hình phân cấp, các thực thể được biểu diễn dưới dạng các bản ghi và các liên kết được biểu diễn bởi các móc nối hay con trỏ. Về cơ bản, các móc nối có khả năng biểu diễn các liên kết giữa một thực thể và nhiều thực thể của các kiểu thực thể liên quan. Tuy nhiên, với cách nhìn dữ liệu theo kiểu phân cấp bởi cấu trúc cây, các bản ghi dữ liệu chỉ có ý nghĩa khi nó được xem xét trong một ngữ cảnh cụ thể, nói cách khác các bản ghi phụ thuộc không thể tồn tại trong cấu trúc cây nếu không có phần tử bậc trên nó. Tuy nhiên, với mô hình phân cấp, chúng ta có một cách rất tự nhiên để mô hình hoá những cấu trúc phân cấp thực sự của thế giới thực.

Mô hình dữ liệu mạng có cấu trúc tổng quát hơn so với mô hình dữ liệu phân cấp bởi một bản ghi dữ liệu đã cho có thể có một số bất kỳ các phần tử bậc trên trực tiếp cũng như một số bất kỳ các phần tử phụ thuộc trực tiếp. Cấu trúc nội tại của mô hình mạng phức tạp hơn so với mô hình phân

cấp nhưng mô hình mạng lại đối xứng hơn mô hình phân cấp. Điểm bất lợi chủ yếu của mô hình này đó là sự quá phức tạp trong bản thân mô hình và trong các thao tác xử lý dữ liệu. Do vậy, những người sử dụng và người viết chương trình sẽ phải hao tốn nhiều thời gian vào việc giải quyết những vấn đề nảy sinh trong quá trình phát triển và bảo trì chương trình.

b. Mô hình dữ liệu hướng đối tượng

So với các mô hình dữ liệu khác, mô hình dữ liệu hướng đối tượng là một mô hình cho phép biểu diễn các đối tượng khá tự nhiên và sát với thực tế hơn cả. Các đối tượng phức tạp và các liên kết giữa chúng được phản ánh đầy đủ và giàu ngữ nghĩa. Do có khả năng biểu diễn dữ liệu khá tốt, nên mô hình dữ liệu hướng đối tượng cũng hỗ trợ các ngôn ngữ thao tác dữ liệu hướng đối tượng mạnh, cho phép người sử dụng dễ dàng biểu diễn cả các truy vấn khá phức tạp. Tuy nhiên, cho đến nay, chưa có một sự thống nhất và chuẩn hoá đối với các khái niệm của mô hình hướng đối tượng cũng như chưa có một cơ sở toán học tốt hình thức hoá ở mức cao, chặt chẽ đối với mô hình này, do vậy nó vẫn được tiếp tục nghiên cứu và phát triển trong tương lại.

Ngày nay, người ta đã nghĩ đến một mô hình dữ liệu kết hợp giữa mô hình dữ liệu quan hệ và mô hình dữ liệu hướng đối tượng gọi tắt là mô hình dữ liệu quan hệ - hướng đối tượng nhằm tận dụng được các ưu điểm của mỗi mô hình trên. Nhưng đây thực sự là một vấn đề khá khó khăn nên mô hình này vẫn đang trong giai đoạn nghiên cứu và thử nghiệm.

5. Kho dữ liệu

Như chúng ta đã biết, cơ sở dữ liệu đã được sử dụng từ những năm 60 và chúng ngày càng chứng tỏ là những thành phần không thể thiếu trong các hoạt động của doanh nghiệp. Qua năm tháng, lượng dữ liệu tác nghiệp này ngày càng nhiều và người ta đã tìm cách khai thác chúng một cách hiệu quả hơn, chủ yếu dựa vào tính lịch sử của chúng. Một xu hướng mới đã xuất hiện và phát triển khá nhanh trong các hệ cơ sở dữ liệu đó là các cơ sở dữ liệu hỗ trợ việc phân tích và khai thác dữ liệu, người ta gọi đó là những kho dữ liệu, chúng chủ yếu được phục vụ cho các ứng dụng hỗ trợ ra quyết định DSS (Decision Support System). Dưới đây là những đặc điểm cơ bản của kho dữ liệu:

- Tính tích hợp: dữ liệu tập hợp trong kho được thu thập từ nhiều nguồn và được tích hợp với nhau tạo thành một thể thống nhất.
- Tính hướng chủ đề: Các dữ liệu được thu thập, trích lọc và làm sạch để đưa vào kho theo những chủ đề nhất định.
- Gắn thời gian: Mỗi hàng thông tin trong kho dữ liệu đều chứa một yếu tố thời gian như một phần của khoá chính để đảm bảo tính đơn nhất của mỗi hàng và cung cấp một đặc trưng về thời gian cho dữ liệu.
- Tính lịch sử: Các thông tin trong kho dữ liệu được tập trung theo thời gian và thể hiện một khung nhìn về một lĩnh vực chủ đề trong một giai đoạn dài.
- Tính chỉ đọc: Dữ liệu trong kho là dữ liệu chỉ đọc và không thể bị sửa đổi.

- Tính bất biến: Dữ liệu được lưu trữ lâu dài trong kho, mặc dù những dữ liệu mới vẫn tiếp tục được thu thập nhưng dữ liệu cũ trong kho vẫn không bị xoá, bị sửa đổi, điều đó cho phép cung cấp thông tin về một khoảng thời gian dài, đủ số liệu cần thiết cho các mô hình nghiệp vụ phân tích, dự báo.
- Bao gồm cả dữ liệu tổng hợp và chi tiết: Dữ liệu chi tiết là thông tin mức thấp nhất được lưu trữ trong kho. Dữ liệu tổng hợp được tích hợp lại qua nhiều giai đoạn khác nhau.

6. Kết luận

Trong chủ đề này, chúng ta đã làm quen với các khái niệm cơ bản liên quan đến việc phân tích, thiết kế các hệ cơ sở dữ liệu.

Các khái niệm cần phải nắm được trong chủ đề này đó là: các khái niệm về cơ sở dữ liệu, hệ quản trị cơ sở dữ liệu, các khái niệm về dữ liệu, siêu dữ liệu, thông tin, từ điển dữ liệu, các khái niệm về thực thể, kiểu thực thể, thuộc tính, trường dữ liệu, liên kết, bảng dữ liệu, quan hệ, lược đồ quan hệ. Những khái niệm này được trình bày lần lượt, dễ hiểu và được minh hoạ bằng các ví dụ kèm theo. Cũng cần lưu ý rằng các khái niệm này không độc lập hoàn toàn với nhau mà giữa chúng có một sự liên quan nhất định.

Ngoài ra một số mô hình dữ liệu cũng đã được đề cập, trong đó quan trọng nhất là mô hình dữ liệu quan hệ, và việc phân tích, thiết kế các cơ sở dữ liệu được trình bày

trong giáo trình sẽ là những cơ sở dữ liệu quan hệ được xây dựng dựa trên chính mô hình dữ liệu này.

7. Bài tập

1. Hãy cho biết thế nào là một cơ sở dữ liệu

2. Hãy phân biệt giữa dữ liệu, thông tin và siêu dữ liệu

3. Thực thể là gì? Kiểu thực thể là gì? Lấy ví dụ trong thực tế cho hai khái niệm này

4. Quan hệ là gì? Quan hệ (hay bảng dữ liệu) trong khái niệm cơ sở dữ liệu khác với các bảng thông thường khác ở điểm gì?

5. Thuộc tính là gì? Hãy lấy ví dụ về một kiểu thực thể và các thuộc tính mô tả cho kiểu thực thể đó.

6. Lựa chọn một bài toán trên thực tế gần với nghiệp vụ hàng ngày của bạn, liệt kê các kiểu thực thể cùng các thuộc tính cần quản lý tương ứng trong bài toán đó.

Chủ đề II: Phương pháp phân tích, thiết kế và xây dựng các hệ thống CSDL

Mục tiêu:

Mục tiêu chính của chủ đề này là giúp người đọc hình dung, nắm bắt và làm chủ được các bước cần phải thực hiện khi phân tích, thiết kế và xây dựng các hệ thống cơ sở dữ liệu quan hệ. Các bước đó bao gồm:

- Phân tích yêu cầu
- Thiết kế cơ sở dữ liệu mức khái niệm
- Thiết kế cơ sở dữ liệu mức logic
- Tinh chỉnh sơ đồ dữ liệu và các dạng chuẩn
- Thiết kế cơ sở dữ liệu mức vật lý
- Thiết kế an toàn bảo mật cho cơ sở dữ liệu

Trong mỗi bước, các bạn sẽ được hướng dẫn cụ thể các tình huống cần phải xử lý. Các kinh nghiệm thực tế sẽ được trình bày thông qua các ví dụ minh hoạ.

1. Khái quát chung

Phân tích, thiết kế và xây dựng các hệ cơ sở dữ liệu là một trong những bước quan trọng nhất trong quá trình phân tích, thiết kế, xây dựng các hệ thống thông tin. Các cơ sở dữ liệu được xem là nền tảng của hệ thống thông tin, do vậy chúng ta có thể nói rằng một hệ thống thông tin chỉ mang lại hiệu quả khi cơ sở dữ liệu được thiết kế tốt.

Việc nắm vững các bước trong quá trình phân tích, thiết kế và xây dựng các hệ thống cơ sở dữ liệu là một đòi hỏi bắt buộc đối với mỗi một người phân tích, thiết kế các hệ thống thông tin. Ngoài ra đối với những người quản trị hệ thống, người phát triển chương trình, việc hiểu rõ các bước này cũng là điều kiện cần trong công việc của họ. Bởi chỉ có nắm vững được các công đoạn phân tích, xây dựng cơ sở dữ liệu này, họ mới có thể chủ động và thực hiện tốt công việc của mình.

Trong quá trình phân tích cơ sở dữ liệu, chúng ta sẽ phải sử dụng đến một mô hình, đó là mô hình thực thể - liên kết thông qua việc biểu diễn các đối tượng trong bài toán bằng những sơ đồ thực thể - liên kết.

Trong quá trình thiết kế cơ sở dữ liệu, chúng ta lại phải sử dụng đến các mô hình dữ liệu, mà ở đây chính là mô hình dữ liệu quan hệ được thể hiện qua việc xây dựng các sơ đồ dữ liệu mức logic với các quan hệ.

2. Phân tích yêu cầu

Bước đầu tiên trong quá trình phân tích, thiết kế một hệ thống cơ sở dữ liệu luôn luôn là bước phân tích yêu cầu.

Trong bước này chúng ta cần tìm hiểu dữ liệu gì sẽ phải được lưu trữ trong cơ sở dữ liệu, ứng dụng nào sẽ được xây dựng dựa trên cơ sở dữ liệu đó, và các thao tác gì sẽ được thực hiện trên những dữ liệu đó. Nói một cách tổng quát, chúng ta phải tìm hiểu người sử dụng họ muốn gì từ hệ thống cơ sở dữ liệu mà chúng ta sẽ xây dựng.

Quá trình phân tích yêu cầu của người sử dụng thường là quá trình thảo luận với khách hàng, nghiên cứu môi trường

tác nghiệp hiện thời và các mong muốn của khách hàng đối với hệ thống cơ sở dữ liệu sẽ được xây dựng.

Để có thể nắm được chi tiết một lĩnh vực dự định sẽ được xây dựng, chúng ta cần phải tìm hiểu hiện trạng, phân tích các yêu cầu của các đối tượng sử dụng từ những người thừa hành, đến những người quản lý và lãnh đạo.

Nếu những người thừa hành có thể cho chúng ta biết các thông tin chi tiết về một công việc cụ thể, người quản lý có thể cho chúng ta biết một chức năng, một lĩnh vực trong hệ thống thì người lãnh đạo lại có thể cho chúng ta một cách tổng thể cùng các mục tiêu trung và dài hạn của toàn bộ hệ thống.

Kỹ thuật và công cụ thường được sử dụng nhiều nhất trong giai đoạn này là phỏng vấn. Bổ sung cho kỹ thuật cơ bản này có thể là các phiếu điều tra, các bảng câu hỏi gửi tới các đối tượng mà chúng ta muốn nghiên cứu, tìm hiểu. Dù có khá nhiều kỹ thuật có thể áp dụng để tiến hành công việc phân tích yêu cầu, song để có được những hiểu biết cơ bản thì phỏng vẫn vẫn là công cụ và kỹ thuật hữu hiệu nhất.

Mục đích chính của khâu phân tích yêu cầu, đó là:

- Xác định các yêu cầu dữ liệu của cơ sở dữ liệu và trình bày chúng dưới dạng các đối tượng.
- Phân loại và mô tả các thông tin về các đối tượng đó.
- Xác định và phân loại các mối liên kết giữa các đối tượng
- Xác định các giao dịch sẽ được thực hiện trên các đối tượng dữ liệu đó.

- Xác định các qui tắc nghiệp vụ (luật) đảm bảo tính toàn vẹn của dữ liệu.

Khâu phân tích yêu cầu thường được tiến hành song song với khâu mô hình hoá dữ liệu. Khi thu thập được các thông tin, chúng ta cũng nên tiến hành xác định và phân loại các thông tin đó, chúng sẽ là các kiểu thực thể, các thuộc tính, hay là các liên kết. Chúng ta hãy đặt tên cho chúng và cố gắng sử dụng các từ ngữ thật gần gũi với người sử dụng. Những đối tượng này sau đó sẽ được mô hình hoá và phân tích bằng cách sử dụng sơ đồ thực thể - liên kết, bước phân tích thiết kế cơ sở dữ liệu mức khái niệm.

Chúng ta cần phải lưu ý rằng, sơ đồ thực thể - liên kết cũng phải được xem xét bởi người sử dụng, họ có thể sẽ phát hiện ra những chỗ mà chúng ta thể hiện còn thiếu hay chưa đúng, khi đó chúng ta cần phải có kế hoạch để tiếp tục tiến hành khảo sát, phân tích bổ sung.

Có ba điểm chính cần phải ghi nhớ trong bước phân tích yêu cầu của người sử dụng, đó là:

- Khi trao đổi với người sử dụng, chúng ta phải sử dụng các thuật ngữ trong thế giới thực. Người sử dụng thường không có các khái niệm thế nào là một kiểu thực thể, thế nào là một thuộc tính, thế nào là liên kết nhưng họ lại có thể hiểu về một người, một sự kiện, mối quan hệ giữa chúng và về các bước thực hiện để giải quyết một vấn đề mà họ tiếp xúc hàng ngày.
- Chúng ta hãy bỏ thời gian để nghiên cứu về mô hình cùng các hoạt động của doanh nghiệp. Khi hiểu các

qui trình nghiệp vụ của doanh nghiệp thì chúng ta cũng sẽ dễ dàng hơn trong việc mô hình hoá nó.

- Những người sử dụng khác nhau thường có cách suy xét và nhìn nhận dữ liệu theo các góc độ khác nhau, tương ứng với chức năng và nhiệm vụ của họ. Do vậy, việc phỏng vấn đầy đủ các đối tượng sử dụng là điều hết sức quan trọng.

2.1. Xác định dữ liệu cần lưu trữ

Đối với các hệ thống cơ sở dữ liệu, điểm cốt yếu khi tìm hiểu hệ thống đó là phải xác định được các dữ liệu nào sẽ được lưu trữ trong cơ sở dữ liệu. Nếu chúng ta xác định sai hoặc thiếu sẽ dẫn tới những thay đổi khôn lường sau khi cơ sở dữ liệu đi vào hoạt động, và khi đó vấn đề sửa đổi lại thiết kế sẽ tốn khá nhiều thời gian và công sức. Do đó, chúng ta cần phải xác định càng chi tiết đối với dữ liệu cần lưu trữ bao nhiêu sẽ càng đảm bảo tính thành công của cơ sở dữ liệu sẽ được xây dựng bấy nhiêu.

Đối với mỗi dữ liệu thu thập được chúng ta cần thể hiện đầy đủ các mục sau:

- Tên dữ liệu: chọn tên phù hợp với ngôn ngữ thường dùng của bài toán, sau đó gán cho nó một tên tượng trưng tương ứng.
- Định nghĩa: chúng ta hãy cố gắng định nghĩa sao cho người sử dụng có thể hiểu được.
- Kiểu dữ liệu: qua cấu trúc dữ liệu, chúng ta sẽ phải xác định chúng thuộc kiểu nào trong các kiểu sau: Ký

tự, số, ngày tháng, logic,... đồng thời chỉ rõ kích cỡ cần thiết hay là độ dài của dữ liệu.

- Loại dữ liệu: nó là loại sơ cấp (không thể chia nhỏ được nữa), loại kết nối (được gộp lại từ việc ghép nhiều dữ liệu với nhau), loại tính toán (được suy ra từ một số dữ liệu khác).
- Định lượng: chúng ta phải ước tính được số các giá trị khác nhau mà dữ liệu có thể nhận được.
- Ghi chú thêm: cho phép chúng ta dùng hình thức diễn đạt tự do để có thể nêu thêm các dữ kiện bổ sung cho dữ liệu.

Ví dụ về một phiếu mô tả dữ liệu:

- Tên: tên nhà cung cấp (Supplier_name)
- Định nghĩa: dùng để đặt tên, thường tóm tắt, cho phép xác định nhà cung cấp
- Cấu trúc: Kiểu ký tự C(30)
- Loại: sơ cấp
- Định lượng: khoảng 50
- Ví dụ: công ty SUNIMEX, công ty IMEXCO
- Ghi chú: có thể chỉ sử dụng 15 ký tự.

2.2. Xác định ứng dụng sẽ được cài đặt trên cơ sở dữ liệu

Như phần đầu của chủ đề đã trình bày, việc xây dựng cơ sở dữ liệu là nền tảng cho một hệ thống thông tin. Ngoài cơ sở dữ liệu trong hệ thống còn có một thành phần cũng khá

quan trọng, đó là các ứng dụng. Giữa hai thành phần này của một hệ thống thông tin có sự quan hệ mật thiết với nhau, bởi các ứng dụng phải được xây dựng và cài đặt trên một cơ sở dữ liệu cụ thể. Do vậy, trong quá trình phân tích yêu cầu, chúng ta cũng phải xác định được những ứng dụng nào sẽ được xây dựng, những đối tượng người dùng nào sẽ sử dụng, để từ đó có được một cái nhìn tổng quan hơn về cơ sở dữ liệu cần xây dựng.

2.3. Xác định các thao tác thường xuyên được thực hiện

Việc xác định các thao tác thường xuyên được thực hiện trên dữ liệu đóng một vai trò quan trọng, yếu tố này đôi khi lại quyết định tính thành công của một cơ sở dữ liệu. Một cơ sở dữ liệu được thiết kế chính xác về mặt nghiệp vụ nhưng lại không quan tâm đến các thao tác thường xuyên được thực hiện thì chưa thể được đánh giá là một thiết kế đầy đủ.

Trên thực tế, khi lượng dữ liệu tăng lên thì thời gian thực hiện các thao tác trên các dữ liệu cũng tăng lên. Do vậy, nếu trong khâu khảo sát, phân tích yêu cầu, chúng ta không xác định được những thao tác thường xuyên được thực hiện thì không thể đưa ra được những thiết kế hiệu quả, không thể tạo ra được những chỉ mục phù hợp cũng như cách phân chia dữ liệu hợp lý.

Bởi vậy, trong quá trình phân tích tìm hiểu yêu cầu, chúng ta cũng cần phải khảo sát kỹ đến vấn đề này, mặc dù không được mô tả ngay trong các sơ đồ thực thể - liên kết nhưng các thông tin thu thập được sẽ lại đóng vai trò quan trọng

trong quá trình tinh chỉnh sơ đồ cơ sở dữ liệu mức logic cũng như thiết kế cơ sở dữ liệu mức vật lý.

2.4. Một số kỹ thuật phân tích yêu cầu

a. Kỹ thuật phỏng vẫn

Khi tìm hiểu yêu cầu của cơ sở dữ liệu dự định xây dựng, một trong những kỹ thuật phân tích yêu cầu hữu hiệu nhất đó là kỹ thuật phỏng vẫn. Trong kỹ thuật phỏng vẫn, chúng ta cũng phải phân tích để lựa chọn đúng các đối tượng cần phỏng vấn.

Phỏng vẫn nên được tiến hành tuần tự theo cấu trúc phân cấp: đầu tiên là ban lãnh đạo, tiếp đến là các bộ phận, phòng ban. Trong quá trình phỏng vấn chúng ta cố gắng càng nắm được nhiều hồ sơ (đặc biệt là các quy trình nghiệp vụ), nhiều thống kê báo cáo càng tốt.

Kết quả của phỏng vấn sẽ là: các mục tiêu chính mà cơ sở dữ liệu dự định xây dựng phải được đáp ứng.

Phỏng vấn ban lãnh đạo:

Mục tiêu: Tìm hiểu được bức tranh tổng thể của doanh nghiệp trong từng lĩnh vực, từng bộ phận.

Phỏng vấn các vị trí làm việc:

Mục tiêu: Tiếp thu được tất cả các thao tác nghiệp vụ cùng những thông tin cần phải quản lý thuộc các lĩnh vực đã phỏng vấn Ban lãnh đạo.

Chúng ta cũng cần phải phỏng vấn những người phụ trách từng vị trí làm việc có liên quan đến lĩnh vực quan tâm. Ví dụ, với một bài toán quản lý sản phẩm: chúng ta cần phải phỏng vấn các vị trí sau: người phụ trách giao dịch với nhà

cung cấp, người phụ trách giao dịch với khách hàng, người quản lý các sản phẩm làm ra, …

Khi tiến hành phỏng vấn, chúng ta cố gắng tạo ra các cuộc phỏng vấn gồm những người đại diện cho mỗi phòng ban nghiệp vụ. Cố gắng ghi lại đầy đủ tất cả các thông tin trong các cuộc phỏng vấn đó dưới các dạng như tài liệu, băng ghi âm,…

Cuối cùng, chúng ta có thể tiến hành phỏng vấn từng người tham gia trực tiếp vào từng công việc cụ thể. Để có thể liệt kê và mô tả toàn bộ các công việc phải thực hiện, đối với mỗi công việc cần phải nắm bắt được những sự kiện khởi động, chu kỳ thực hiện, thời lượng thực hiện, những dữ kiện có liên quan, khối lượng các dữ kiện này, các qui tắc cần phải áp dụng để thực hiện công việc, quan sát sự luân chuyển thông tin giữa các vị trí làm việc.

Ở giai đoạn phỏng vấn, chúng ta có thể sử dụng các phiếu phỏng vấn trong đó phải thể hiện được các thông tin như: tên công việc, các điều kiện khởi động, kết quả đưa ra, tần suất, thời lượng thông tin, dữ liệu , các quy tắc nghiệp vụ cần thực hiện…

Sau các buổi phỏng vấn chúng ta cần phải xem lại tất cả những gì đã thu thập được, hệ thống sắp xếp lại dưới dạng hồ sơ khảo sát, liệt kê đầy đủ các thông tin. Việc xác định các đối tượng dữ liệu, các qui tắc nghiệp vụ ngoài việc chuẩn bị cho các giai đoạn phân tích dữ liệu tiếp theo còn nhằm để phát hiện được những vấn đề còn chưa rõ để đề ra các câu hỏi phụ cho những người đối thoại trong các buổi phỏng vấn kế tiếp.

b. Kỹ thuật sử dụng phiếu khảo sát

Có thể nói đây là một hình thức phỏng vấn gián tiếp. Các câu hỏi được liệt kê trong một phiếu điều tra, khảo sát. Người được khảo sát sẽ ghi lại các câu trả lời vào trong các phiếu điều tra, khảo sát đó.

Việc sử dụng các loại câu hỏi cũng như thứ tự các câu hỏi ở đây cũng giống như khi phỏng vấn trực tiếp, tuy nhiên thứ tự này chỉ có ý nghĩa về mặt sắp xếp vấn đề mà ít có tác dụng dẫn dắt tư duy của người được hỏi. Vì người được khảo sát có thể đọc qua một lượt các câu hỏi, rồi tuỳ tiện trả lời các câu hỏi theo bất cứ trật tự nào mà họ muốn, thậm chí họ có thể không trả lời một số câu hỏi.

So với kỹ thuật phỏng vấn trực tiếp, chỗ yếu chính của phiếu khảo sát là thiếu sự giao tiếp giữa chúng ta và khách hàng. Nhưng ngược lại, phiếu khảo sát lại có chỗ mạnh là có thể mở rộng diện khảo sát và ít tốn kém.

c. Kỹ thuật phân tích tài liệu

Các loại sổ sách, tài liệu, tệp dưới dạng điện tử,… thường là những nguồn thông tin để khảo sát về các loại dữ liệu, luồng dữ liệu và giao dịch. Kỹ thuật phân tích khai thác nguồn tài liệu này được tiến hành như sau: đầu tiên lập một danh sách các tài liệu qua sự tìm hiểu từ phía các đối tượng sử dụng, sau đó nghiên cứu từng tài liệu để phát hiện ra các dữ liệu cơ bản và các dữ liệu cấu trúc. Kỹ thuật này có thể cho phép chúng ta phát hiện ra các dữ liệu trùng lặp hoặc sự thiếu nhất quán trong lưu trữ, tên gọi.

Ngoài ra, nếu doanh nghiệp đã có một hệ thống cơ sở dữ liệu và họ muốn nâng cấp hay thay đổi toàn bộ thì chúng ta

có thể xem xét lại các tài liệu liên quan đến cơ sở dữ liệu đó, điều này sẽ giúp cho chúng ta biết thêm khá nhiều thông tin và hiểu được tại sao doanh nghiệp lại muốn xây dựng một hệ thống cơ sở dữ liệu mới.

3. Thiết kế cơ sở dữ liệu mức khái niệm

Các thông tin thu thập được trong bước phân tích yêu cầu trên sẽ được sử dụng để xây dựng lên những mô tả đầu tiên về dữ liệu cũng như các ràng buộc cần phải thoả mãn đối với dữ liệu. Bước này thường được thực hiện thông qua một mô hình mà chúng ta gọi là mô hình thực thể - liên kết.

Mô hình thực thể - liên kết thường được sử dụng như là một công cụ kết nối giữa người thiết kế cơ sở dữ liệu và người dùng cuối trong quá trình phân tích cơ sở dữ liệu. Nó được sử dụng để xây dựng lên mô hình dữ liệu mức khái niệm, mô tả cấu trúc và các ràng buộc của dữ liệu trong cơ sở dữ liệu, mô hình này độc lập với cả hệ quản trị cơ sở dữ liệu cũng như mô hình dữ liệu sẽ được sử dụng đề cài đặt nó.

Mô hình thực thể - liên kết thường được biểu diễn dưới dạng một sơ đồ thực thể - liên kết thông qua các hình vẽ mô tả cho 3 đối tượng chính, đó là kiểu thực thể, thuộc tính và liên kết. Do vậy, kết quả của bước thiết kế cơ sở dữ liệu mức khái niệm sẽ là một sơ đồ thực thể - liên kết đầy đủ.

Khi xây dựng một sơ đồ thực thể - liên kết, chúng ta phải tiến hành các lựa chọn sau:

- Một khái niệm sẽ được mô hình hoá thành một kiểu thực thể hay một thuộc tính?
- Nếu là kiểu thực thể thì nó sẽ là loại nào?

- Nếu là thuộc tính thì nó sẽ là loại nào?
- Các thuộc tính đã đủ để minh hoạ cho kiểu thực thể chưa?
- Một khái niệm sẽ được mô hình hoá thành một kiểu thực thể hay một liên kết?
- Nếu là một liên kết thì chúng kết nối những kiểu thực thể nào? Nên sử dụng loại liên kết nào?

Để có thể trả lời các câu hỏi đó, chúng ta phải phân tích kỹ các thông tin thu thập được trong bước phân tích yêu cầu cũng như kinh nghiệm thiết kế các hệ cơ sở dữ liệu của chính mình.

3.1. Lựa chọn giữa kiểu thực thể mạnh và kiểu thực thể yếu

a. Kiểu thực thể mạnh

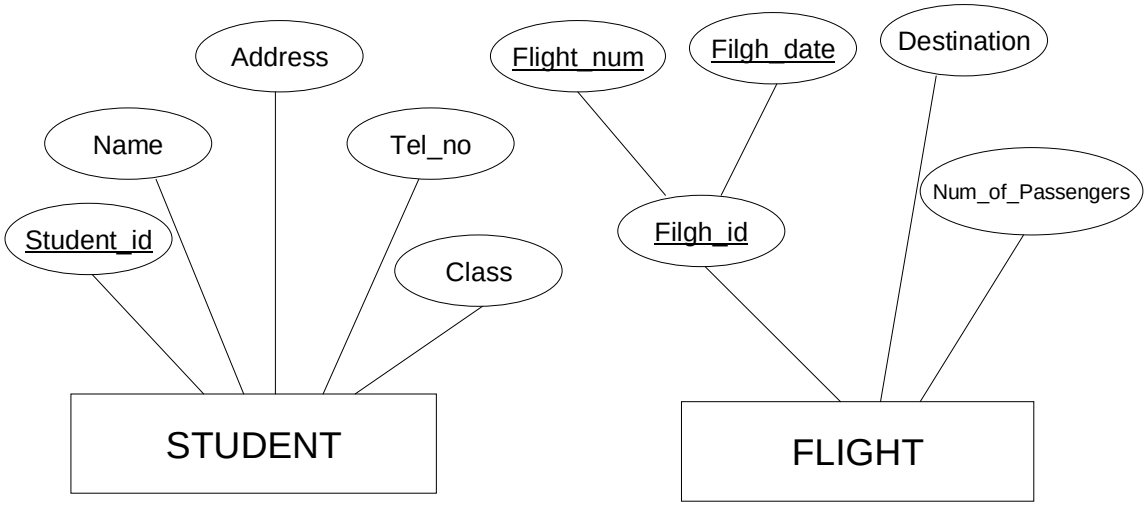
Hầu hết các kiểu thực thể được xác định trong một bài toán xây dựng cơ sở dữ liệu là các kiểu thực thể mạnh. Kiểu thực thể mạnh là một kiểu thực thể có thể tồn tại độc lập với các kiểu thực thể khác. Ví dụ như các kiểu thực thể EMPLOYEE, STUDENT trong các ví dụ đã nêu.

Trong kiểu thực thể mạnh, mỗi thực thể của nó luôn tồn tại một đặc trưng duy nhất được gọi là định danh. Định danh này có thể bao gồm một hoặc gộp của một số thuộc tính, mỗi giá trị xuất hiện trong định danh sẽ xác định duy nhất một thực thể trong kiểu thực thể. Các thuộc tính tham gia vào định danh được gọi là các thuộc tính định danh hay thuộc tính khoá. Chúng ta sẽ tìm hiểu kỹ hơn về loại thuộc

tính này trong phần lựa chọn các thuộc tính được trình bày trong các phần sau.

Lưu ý: chúng ta thường sử dụng khái niệm định danh với kiểu thực thể, còn khái niệm khoá với quan hệ.

Ví dụ chúng ta có hai kiểu thực thể như trong hình 2.3.1.1 sau:



Hình 2.3.1.1 Kiểu thực thể mạnh

Cả hai kiểu thực thể trên đều là kiểu thực thể mạnh, chúng được ký hiệu bằng một hình chữ nhật. Trong kiểu thực thể sinh viên STUDENT: định danh là thuộc tính số hiệu sinh viên Student_id. Còn trong kiểu thực thể chuyến bay FLIGHT thì định danh bao gồm kết hợp của hai thuộc tính đó là số hiệu chuyến bay Flight_num và ngày Fligh_date, bởi để xác định một chuyến bay cụ thể chúng ta cần kết hợp các giá trị trong cả hai thuộc tính này.

b. Kiểu thực thể yếu

Ngược lại, kiểu thực thể yếu là một kiểu thực thể mà sự tồn tại của nó phụ thuộc vào một kiểu thực thể khác. Trong sơ

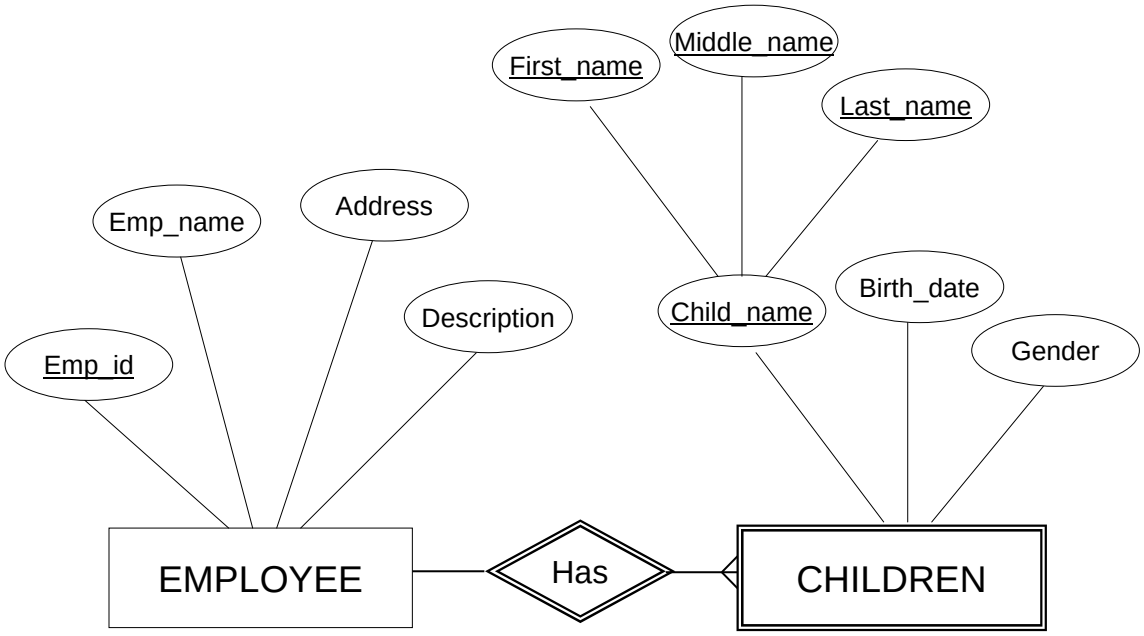
đồ thực thể - liên kết, một kiểu thực thể yếu luôn luôn phải được biểu diễn cùng với kiểu thực thể mà nó phụ thuộc. Kiểu thực thể mà kiểu thực thể yếu phụ thuộc vào được gọi là kiểu thực thể sở hữu (owner).

Trong kiểu thực thể yếu không tồn tại định danh đầy đủ. Ngoài các thuộc tính mô tả, trong kiểu thực thể yếu chỉ tồn tại một số thuộc tính được xem là định danh bộ phận. Định danh bộ phận này sẽ được kết hợp với định danh của kiểu thực thể sở hữu để có thể xác lập duy nhất một thực thể trong nó.

Chúng ta ký hiệu một kiểu thực thể yếu bởi một hình chữ nhật có nét kép như trong hình 2.3.1.2 dưới đây.

Liên kết giữa kiểu thực thể yếu và kiểu thực thể sở hữu nó được gọi là liên kết định danh và được ký hiệu bằng một hình thoi có nét kép.

Ví dụ: Chúng ta muốn lưu những thông tin về tình hình các cháu nhỏ là con của các nhân viên làm việc trong doanh nghiệp. Khi đó chúng ta phải sử dụng hai kiểu thực thể, một kiểu thực thể là EMPLOYEE mô tả các thông tin về các nhân viên và một kiểu thực thể là CHILDREN mô tả các thông tin về tình trạng con cái của họ. Rõ ràng EMPLOYEE là một kiểu thực thể mạnh với định danh là thuộc tính mã hiệu nhân viên Emp_id. Còn CHILDREN là một kiểu thực thể yếu, nó phụ thuộc vào kiểu thực thể EMPLOYEE, thuộc tính Child_name được gọi là định danh bộ phận. Và liên kết “Has” là một liên kết định danh, nó được ký hiệu bằng một hình thoi có hai nét. Ví dụ này được minh họa như hình 2.3.1.2 sau:



Hình 2.3.1.2 Kiểu thực thể yếu

3.2. Lựa chọn giữa các loại thuộc tính

a. Thuộc tính đơn và gộp

Thuộc tính đơn (hay nguyên tử) là thuộc tính không thể chia nhỏ thành các thành phần riêng biệt. Ví dụ như các thuộc tính số hiệu sinh viên Student_id, màu sắc Color, Số lượng Quantity,...

Ngược lại, có một số thuộc tính mà giá trị của nó có thể chia được ra thành nhiều phần có ý nghĩa. Ví dụ như thuộc tính địa chỉ Address, thuộc tính này có thể được chia thành các phần như: Country, Province, City và Street. Hay trong hình 2.3.1.2 trên, thuộc tính Child_name là thuộc tính gộp của 3 thuộc tính First_name, Middle_name và Last_name.

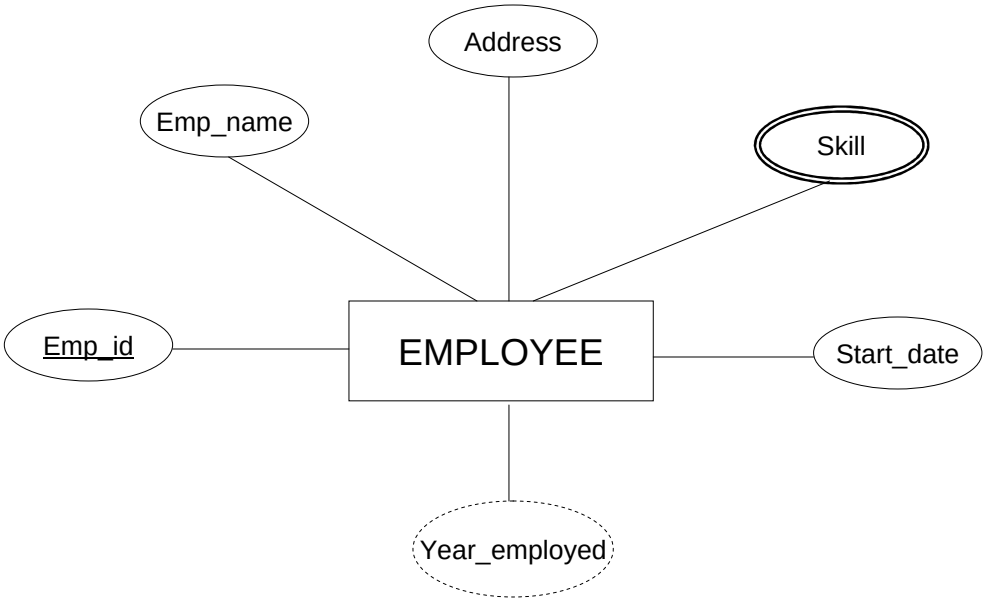
Thuộc tính gộp chỉ mang tính chất minh hoạ bởi nó được biểu diễn bởi các thuộc tính đơn. Do đó khi xây dựng sơ đồ thực thể - liên kết, chúng ta cố gắng trình bày các thuộc tính dưới dạng các thuộc tính đơn.

b. Thuộc tính đơn trị và thuộc tính đa trị

Trong quá trình phân tích bài toán đề tìm kiếm các thuộc tính cho một kiểu thực thể, chúng ta sẽ gặp những trường hợp mà một thuộc tính có thể có nhiều hơn một giá trị đối với một số thực thể.

Ví dụ: trong kiểu thực thể EMPLOYEE, chúng ta nhận thấy rằng thuộc tính kỹ năng Skill của mỗi một thực thể có thể nhận nhiều hơn một giá trị, bởi trên thực tế một nhân viên lập trình có thể biết cả ngôn ngữ C++ và ngôn ngữ Basic.

Như vậy, một thuộc tính đa trị là một thuộc tính có thể chứa một hoặc một vài giá trị. Chúng ta ký hiệu thuộc tính đa trị bằng một hình elip có một vòng tròn kép như thuộc tính Skill trong hình 2.3.2.1 sau:



Hình 2.3.2.1 Thuộc tính kép

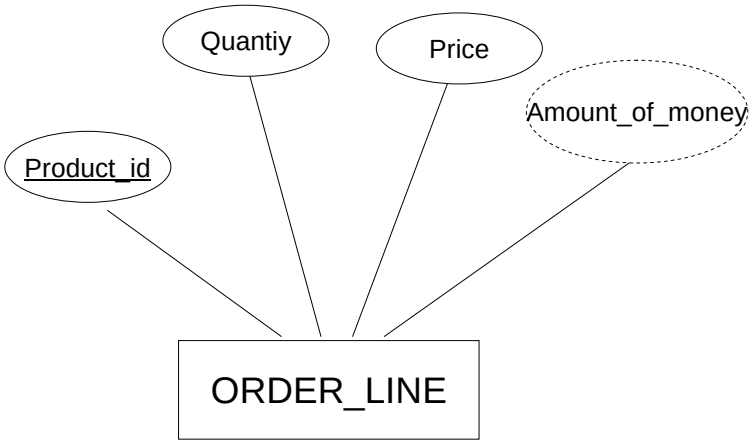
c. Thuộc tính lưu trữ và thuộc tính dẫn xuất

Có một số thuộc tính giá trị của nó có thể được tính toán hoặc suy ra từ một số giá trị của các thuộc tính liên quan.

Ví dụ: với kiểu thực thể EMPLOYEE, chúng ta có thuộc tính ngày bắt đầu làm việc Start_date cho biết ngày một nhân viên được nhận vào làm việc, rõ ràng nếu muốn biết một nhân viên nào đó đã làm việc được bao nhiêu lâu thì chúng ta có thể dựa trên giá trị trong thuộc tính này và ngày hiện thời để tính ra giá trị đó. Hay trong các bài toán kế toán, với kiểu thực thể dòng đơn hàng ORDER_LINE chúng ta biết rằng giá trị trong thuộc tính tổng tiền Amount_of_money luôn bằng giá trị trong thuộc tính đơn giá Unit_price nhân với giá trị trong thuộc tính số lượng Quantity.

Như vậy một thuộc tính dẫn xuất là một thuộc tính mà các giá trị của nó có thể được suy ra từ các giá trị của các thuộc tính liên quan (cũng có thể giá trị đó không lưu trong cơ sở dữ liệu, như giá trị ngày hiện thời như trong ví dụ trên).

Chúng ta ký hiệu thuộc tính dẫn suất bằng một hình elip có nét đứt như trong các hình 2.3.2.1 và 2.3.2.2:



Hình 2.3.2.2 Thuộc tính dẫn xuất

d. Thuộc tính khoá hay thuộc tính định danh

Như đã trình bày, trong cùng một kiểu thực thể chúng ta cần phải phân biệt được giữa các thực thể với nhau. Để làm được điều này chúng ta phải quy ước chọn ra một hoặc một số thuộc tính sao cho giá trị xuất hiện trong các thuộc tính đó sẽ xác định duy nhất một thực thể, chúng ta gọi các thuộc tính này là một định danh và các thuộc tính tham gia vào định danh đó là các thuộc tính khoá hay thuộc tính định danh.

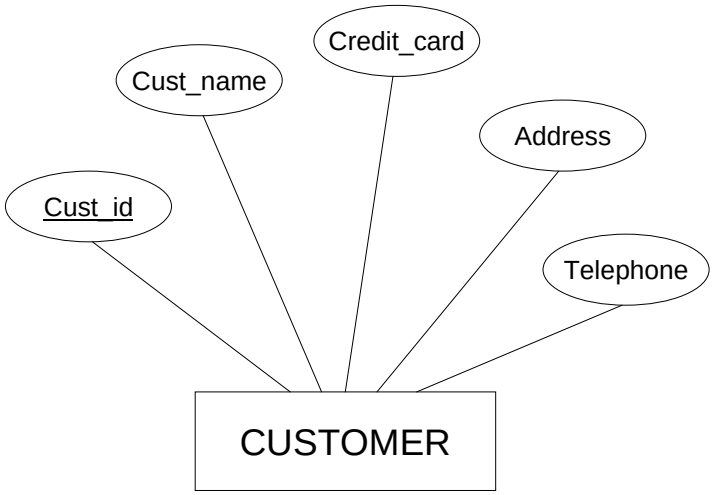
Ví dụ, định danh cho kiểu thực thể STUDENT sẽ là thuộc tính số hiệu sinh viên Student_id. Tuy nhiên, một thuộc

tính khác như tên sinh viên Student_name chẳng hạn thì không thể là định danh, bởi có thể có một số sinh viên có tên trùng nhau (bao gồm cả họ và tên), mặt khác tên của một sinh viên cũng có thể thay đổi.

Đôi khi chúng ta sẽ gặp trường hợp trong cùng một kiểu thực thể có thể có nhiều hơn một (hoặc một số) thuộc tính có thể được sử dụng làm định danh, khi đó dựa trên bài toán cùng kinh nghiệm, chúng ta phải chọn lấy một trong số các lựa chọn đó làm định danh duy nhất.

Ví dụ: với kiểu thực thể khách hàng CUSTOMER, chúng ta có các thuộc tính: Cust_id, Cust_name, Address, Tel_no, Credit_card. Rõ ràng với kiểu thực thể này thuộc tính số hiệu khách hàng Cust_id và số thẻ tín dụng Credit_card đều có thể chọn là định danh, bởi mỗi giá trị trong các thuộc tính này đều xác định duy nhất một thực thể khách hàng Tuy nhiên theo kinh nghiệm, chúng ta vẫn nên chọn thuộc tính số hiệu khách hàng Cust_id làm định danh duy nhất cho kiểu thực thể này bởi có thể có những khách hàng không sử dụng thẻ tín dụng.

Chúng ta phân biệt thuộc tính định danh với các thuộc tính khác bằng một đường gạch chân dưới thuộc tính đó như hình 2.3.2.3 sau:



Hình 2.3.2.3 Thuộc tính định danh

Dưới đây là một số gợi ý mà chúng ta có thể sử dụng để lựa chọn được một định danh duy nhất cho mỗi kiểu thực thể:

- Chọn một định danh mà giá trị của nó sẽ không bị thay đổi theo thời gian. Ví dụ trong kiểu thực thể EMPLOYEE, chúng ta có thể chọn hai thuộc tính tên nhân viên Name và địa chỉ Address kết hợp làm định danh duy nhất vì thực sự chúng luôn là duy nhất. Tuy nhiên lựa chọn này là một lựa chọn không tối ưu bởi rõ ràng một nhân viên có thể thay đổi chỗ ở của mình trong quá trình làm việc tại công ty.
- Chọn định danh gồm các thuộc tính mà giá trị của chúng không được phép mang giá trị null (giá trị không xác định). Trong trường hợp định danh là một thuộc tính gộp thì các thuộc tính con của nó cũng không được phép rỗng. Ví dụ như với kiểu thực thể chuyến bay FILGHT như hình 2.3.1.1 trên, đối với định danh gộp, số hiệu chuyến bay Flight_id, là kết hợp của 2 thuộc tính số hiệu chuyến bay Flight_num

và ngày bay Flight_date đều không thể mang giá trị null.

- Tránh sử dụng những thuộc tính mà giá trị của nó thể hiện thông tin, hay cấu trúc của nó thể hiện sự phân loại, vị trí,... làm định danh. Ví dụ nếu chúng ta có kiểu thực thể kho hàng STORE, các giá trị trong thuộc tính mã kho Store_code có qui ước một ký tự đầu được thể hiện cho địa danh đặt kho, hai ký tự sau thể hiện cho số thứ tự kho trong địa danh đó. Rõ ràng theo thời gian, mã các kho hàng này có thể thay đổi như khi số lượng các địa danh có thể đặt các kho tăng lên nhiều hơn 9 ví trí, tức là khi chúng ta cần phải có tới hai ký tự để thể hiện cho vị trí đặt kho, và khi đó các mã kho cũng phải thay đổi theo. Một ví dụ khác mà trên thực tế, một số bài toán đã gặp phải, đó là qui ước cách đánh mã số các địa bàn hành chính sử dụng bộ mã 7 ký tự, cách đánh mã số này qui ước như sau: 1 ký tự đầu ký hiệu cho vùng, 2 ký tự sau ký hiệu cho tỉnh thuộc vùng đó, 2 ký tự tiếp theo ký hiệu cho huyện thuộc tỉnh và 2 ký tự cuối cùng ký hiệu cho xã thuộc huyện. Khi nảy sinh các vấn đề tách tỉnh, gộp tỉnh, tách huyên, gộp huyện, tách xã gộp xã thì các mã này rõ ràng cũng sẽ bị sửa đổi theo qui tắc đánh mã như trên.
- Cố gắng lựa chọn những thuộc tính đơn làm định danh trong khả năng có thể thay vì sử dụng kết hợp của một số thuộc tính. Ví dụ nếu ta có một cơ sở dữ liệu về một giải bóng đá, trong đó có một kiểu thực thể là trận đấu GAME, chúng ta nên lựa chọn thuộc tính số thứ tự trận đấu Game_no làm định danh thay

vì sử dụng kết hợp hai thuộc tính tên đội khách Visiting_team và tên đội nhà Home_team làm định danh.

3.3. Lựa chọn giữa các loại liên kết

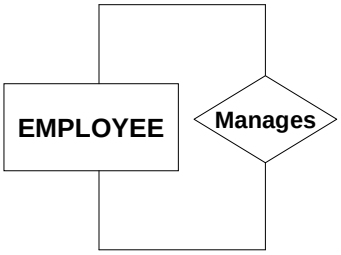
Để mô hình hoá thế giới thực, chúng ta cũng phải sử dụng đến các liên kết để minh hoạ cho mối quan hệ giữa các kiểu thực thể. Và để có thể lựa chọn được đúng loại liên kết chúng ta cần phải nắm được 3 khái niệm cơ bản sẽ được dùng trong mỗi một liên kết.

a. Ba khái niệm cơ bản về liên kết

Việc phân chia ra các loại liên kết được dựa trên 3 khái niệm:

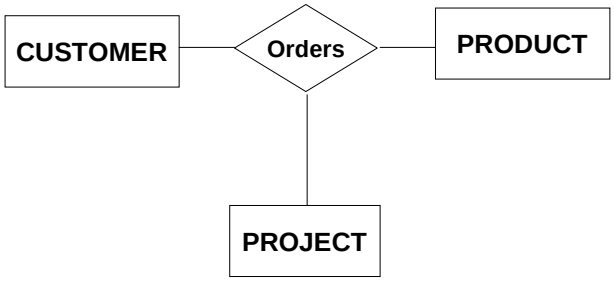
- Cấp độ của liên kết: được thể hiện bằng việc có bao nhiêu kiểu thực thể tham gia vào liên kết. Có thể liên kết kết nối giữa cùng một kiểu thực thể, hoặc cũng có thể liên kết kết nối giữa hai kiểu thực thể hay cũng có thể liên kết kết nối giữa nhiều kiểu thực thể. Tuy nhiên, trên thực tế liên kết giữa hai kiểu thực thể thường được sử dụng nhiều nhất và trong trường hợp chúng ta gặp phải những liên kết giữa nhiều kiểu thực thể thì cố gắng đưa chúng về loại liên kết giữa hai kiểu thực thể.

Chúng ta ký hiệu các cấp độ này như hình 2.3.3.1 sau:



(a) Liên kết giữa cùng một kiểu thực thể

(b) Liên kết giữa hai kiểu thực thể



(c) Liên kết giữa nhiều kiểu thực thể

Hình 2.3.3.1 Cấp độ của liên kết

Ví dụ: như trong hình 2.3.3.1 trên:

Hình (a) thể hiện cho mối liên kết trong cùng một kiểu thực thể EMPLOYEE, liên kết này cho biết một nhân viên có thể quản lý trực tiếp một số nhân viên khác, chẳng hạn giám đốc sẽ quản lý trực tiếp các trưởng phòng, các trưởng phòng lại quản lý trực tiếp các trưởng nhóm, còn các trưởng nhóm lại quản lý trực tiếp các nhân viên. Liên kết này còn được gọi là liên kết đệ qui.

Hình (b) thể hiện cho mối liên kết giữa hai kiểu thực thể là khách hàng CUSTOMER và mặt hàng PRODUCT, liên kết này cho biết khách hàng có thể đặt mua các mặt hàng.

Hình (c) thể hiện cho mối liên kết giữa ba kiểu thực thể là khách hàng CUSTOMER, mặt hàng PRODUCT và

dự án PROJECT. Liên kết này cho biết khách hàng mua các mặt hàng gì và để phục vụ cho dự án nào. Thông thường trong hầu hết các trường hợp, với những liên kết giữa ba kiểu thực thể, chúng ta nên chuyển đổi liên kết đó thành một kiểu thực thể kết nối (sẽ được trình bày dưới đây).

- Lực lượng của liên kết: khái niệm này thể hiện số lượng các thực thể của mỗi một kiểu thực thể tham gia vào liên kết đó như thể nào. Các trường hợp có thể xảy ra như sau: nếu khi mỗi một thực thể của kiểu thực thể này chỉ có thể kết nối với mỗi một thực thể của kiểu thực thể kia thì đó là liên kết một-một. Còn nếu với mỗi một thực thể của kiểu thực thể này có thể kết nối với nhiều thực thể của kiểu thực thể kia thì chúng ta có liên kết một-nhiều. Cuối cùng, nếu mỗi một thực thể của kiểu thực thể này có thể kết nối với nhiều thực thể của kiểu thực thể kia và ngược lại thì khi đó chúng ta có liên kết nhiều-nhiều.

Để phân biệt với đầu liên kết phía một, chúng ta ký hiệu đầu liên kết phía nhiều có ba đoạn thẳng nhỏ xuất phát từ một điểm như trong hình 2.3.3.2 sau:

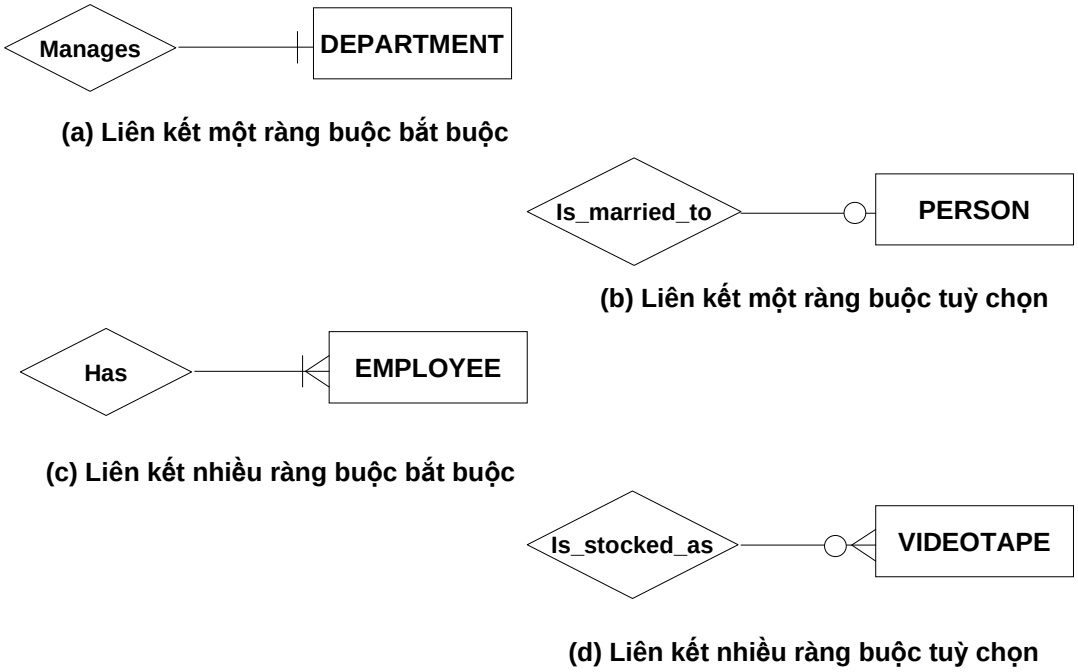
(a) Đầu liên kết phía một **(b) Đầu liên kết phía nhiều**

Hình 2.3.3.2 Lực lượng của liên kết

Chúng ta sẽ tìm hiểu kỹ hơn khái niệm lực lượng của liên kết trong các phần tiếp theo.

- Ràng buộc tham gia của liên kết: khái niệm này thể hiện cung cách của các kiểu thực thể tham gia vào liên kết. Cụ thể, nếu mỗi một thực thể của kiểu thực thể A khi xuất hiện và tham gia vào liên kết đều phải được kết nối với một hoặc một số thực thể của kiểu thực thể B khi đó chúng ta nói ràng buộc tham gia vào liên kết của kiểu thực thể A là bắt buộc. Ngược lại, khi một thực thể của kiểu thực thể A xuất hiện mà có thể có hoặc không có một thực thể nào của kiểu thực thể B cùng tham gia vào liên kết với A thì ta nói ràng buộc tham gia vào liên kết của A là tuỳ chọn.

Chúng ta ký hiệu các ràng buộc này như hình 2.3.3.3 sau:



Hình 2.3.3.3 Ràng buộc tham gia của liên kết

Như trong hình 2.3.3.3 trên chúng ta có:

Hình (a) mô tả cho liên kết một có ràng buộc tham gia là bắt buộc. Ví dụ mỗi một phòng ban phải có duy nhất một người quản lý, khi đó rõ ràng kiểu thực thể

DEPARTMENT khi tham gia vào liên kết phải là loại ràng buộc bắt buộc.

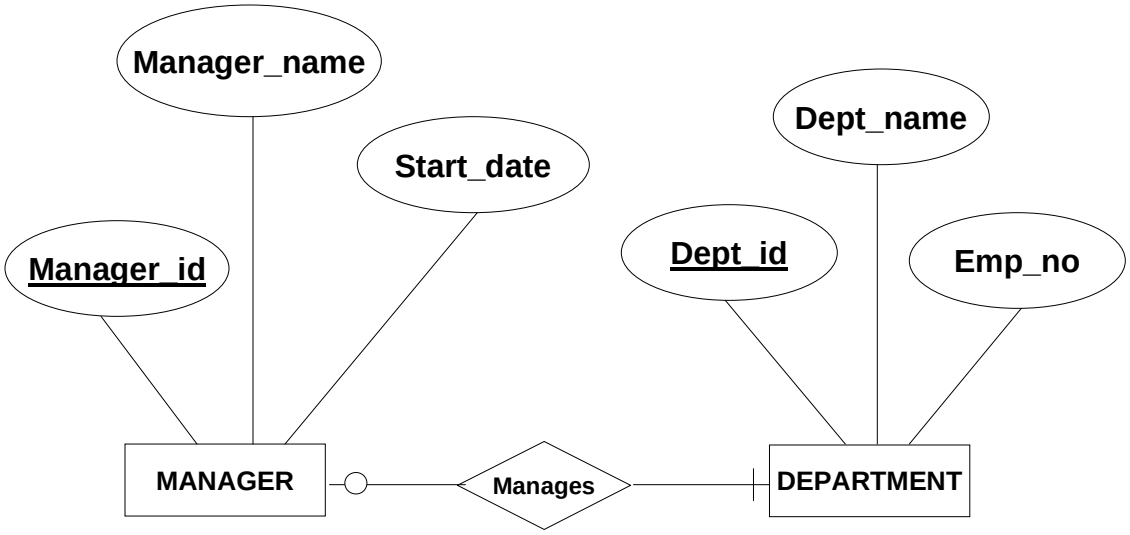
Hình (b) mô tả cho liên kết một có ràng buộc tham gia là tuỳ chọn. Ví dụ, mỗi một người chỉ có thể lập gia đình với một người khác và cũng có thể có những người không hoặc chưa lập gia định. Như vậy chúng ta thấy rằng kiểu thực thể PERSON tham gia vào liên kết một theo kiểu ràng buộc tuỳ chọn.

Hình (c) mô tả cho một liên kết nhiều với ràng buộc tham gia là bắt buộc. Ví dụ một nhân viên phải được bố trí vào làm việc ở một phòng nhất định và có thể có nhiều nhân viên cùng làm việc trong một phòng. Như vậy chúng ta thấy rằng ràng buộc tham gia của kiểu thực thể EMPLOYEE vào liên kết nhiều phải là loại bắt buộc

Hình (d) mô tả cho một liên kết nhiều với ràng buộc tham gia là tuỳ chọn. Ví dụ mỗi một hoặc một số cuốn băng sẽ được sử dụng để ghi lại cho một bộ phim. Tuy nhiên cũng có thể có những cuốn băng vẫn chưa được sử dụng nhưng chúng ta vẫn phải quản lý. Khi đó kiểu thực thể VIDEOTAPE tham gia vào liên kết theo loại ràng buộc là tuỳ chọn.

b. Các liên kết một-một (1:1)

Dạng đơn giản nhất và hiếm gặp nhất của liên kết trên hai kiểu thực thể là một-một, có nghĩa là với mỗi thực thể trong kiểu thực thể A có nhiều nhất một thực thể trong kiểu thực thể B kết nối với nó và ngược lại.



Hình 2.3.3.4 Liên kết một - một

Ví dụ như liên kết quản lý "Manages" giữa 2 kiểu thực thể DEPARTMENT và MANAGER trong hình 2.3.3.4 trên có thể được khai báo là liên kết một-một. Như vậy, chúng ta thấy rằng không bao giờ có một phòng cùng được quản lý bởi nhiều người, và ngược lại cũng không bao giờ thấy một người quản lý 2 hay nhiều phòng. Từ hình vẽ chúng ta cũng có thể thấy một người quản lý có thể ở một thời điểm nào đó không quản lý phòng nào nhưng một phòng thì luôn luôn có một người quản lý.

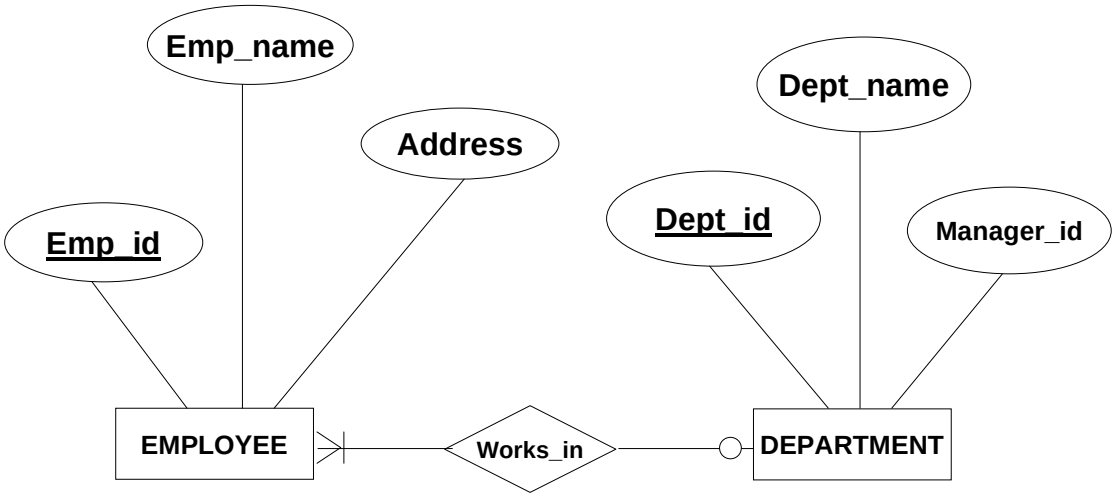
Một ví dụ khác: mỗi một nhân viên trong công ty được bố trí làm việc ở một bàn làm việc nhất định. Cũng như vậy, ở mỗi một bàn làm việc chỉ có thể có một nhân viên ngồi làm việc mà thôi.

c. Các liên kết một-nhiều (1:N)

Đây là loại liên kết quan trọng nhất và phổ biến nhất. Trong liên kết một-nhiều, một thực thể trong kiểu thực thể B được

kết nối với không, một hay nhiều thực thể trong kiểu thực thể A, nhưng mỗi thực thể trong kiểu thực thể A chỉ được kết nối với nhiều nhất một thực thể trong kiểu thực thể B. Liên kết này được gọi là liên kết một-nhiều từ B đến A. Điều này có nghĩa liên kết là một hàm bộ phận từ B đến A.

Ví dụ liên kết làm việc "Works_in" giữa hai kiểu thực thể DEPARTMENT và EMPLOYEE trong hình 2.3.3.5 dưới đây, mỗi một nhân viên được bố trí làm việc ở nhiều nhất một phòng, và mỗi một phòng có thể gồm nhiều nhân viên hoặc chưa có nhân viên nào.



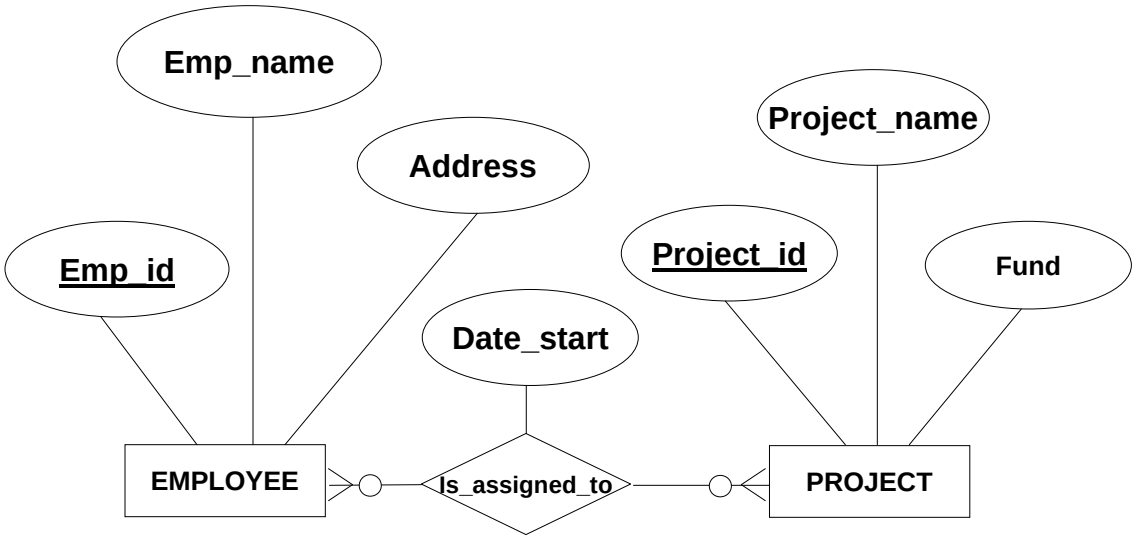
Hình 2.3.3.5 Liên kết một - nhiều

d. Các liên kết nhiều-nhiều (M:N)

Chúng ta cũng bắt gặp các liên kết nhiều-nhiều, đôi khi còn được gọi là các liên kết không cụ thể. Liên kết nhiều-nhiều là liên kết khi với mỗi một thực thể trong kiểu thực thể A, nó được kết nối với không, một hoặc nhiều thực thể trong kiểu thực thể B và ngược lại, với mỗi thực thể B trong kiểu

thực thể B, nó được kết nối với không, một hoặc nhiều thực thể trong kiểu thực thể A.

Ví dụ: Một nhân viên tại cùng một thời điểm có thể cùng tham gia vào hai hay nhiều dự án. Ngược lại mỗi một dự án có thể được thực hiện bởi một hay nhiều nhân viên khác nhau. Như vậy một nhân viên có thể được kết nối với nhiều dự án, ngược lại, một dự án cũng có thể được kết nối với nhiều nhân viên. Hình 2.3.3.6 mô tả cho liên kết tham gia này:



Hình 2.3.3.6 Liên kết nhiều - nhiều

Trong khi các liên kết nhiều-nhiều xuất hiện khá thường xuyên trong thực tế, chúng ta phải cẩn thận khi xem xét các liên kết này được biểu thị như thế nào trong mô hình khái niệm của cơ sở dữ liệu.

Đôi khi liên kết nhiều-nhiều không được chuyển đổi trực tiếp thành các bảng, thay vào đó nó sẽ được chuyển đổi

thành hai hay nhiều liên kết một-nhiều bằng cách sử dụng thêm kiểu thực thể kết nối.

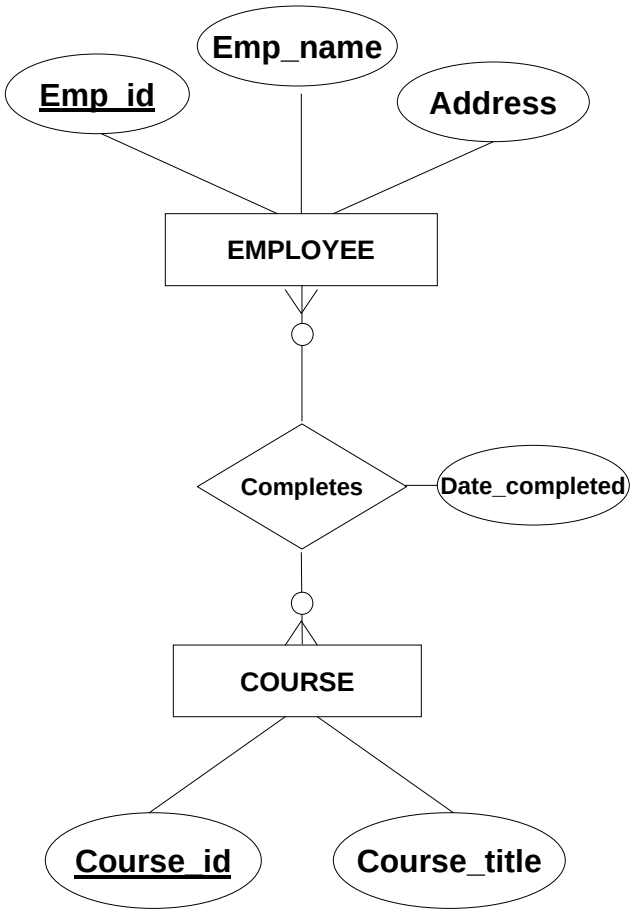
e. Thuộc tính cho liên kết

Các thuộc tính cũng có thể được sử dụng để mô tả cho các liên kết.

Ví dụ trong hình 2.3.3.7a là một số dữ liệu về tình hình tốt nghiệp các khoá học nâng cao trình độ của một số nhân viên. Giá trị trong thuộc tính ngày tháng Date_completed cho biết ngày tháng mà các nhân viên đã hoàn thành các khoá học tương ứng.

Emp_name	Course	Date_ completed
Trần Văn T.	C++	06/2003
Trần Văn T.	Java	09/2003
Mai Văn H.	Basic	05/2003
Mai Văn H.	SQL	07/2003
Lê Đức T.	C++	02/2003
Nguyễn Thị K.	C++	02/2003

(a) Bảng dữ liệu



(b) Thuộc tính cho liên kết

Hình 2.3.3.7 Thuộc tính trên liên kết

Vậy thì trong sơ đồ thực thể - liên kết, thuộc tính Date_completed sẽ được đặt ở vị trí nào? Chúng ta không thể đặt thuộc tính này trong các kiểu thực thể EMPLOYEE cũng như COURSE. Bởi Date_completed mô tả cho liên kết "Completes" chứ không phải cho các kiểu thực thể đó. Mặt khác chúng ta cũng thấy rằng, mỗi một thể hiện của liên kết "Completes" cũng tồn tại đi kèm một giá trị của thuộc tính Date_completed. Như bảng trong hình 2.3.3.7a, chúng ta thấy rằng Trần Văn T. đã hoàn thành khoá học về C++ vào tháng 06/2003.

Do vậy chúng ta phải sử dụng thuộc tính Date_completed để mô tả cho liên kết hoàn thành "Completes" giữa hai kiểu thực thể nhân viên EMPLOYEE và lớp học COURSE như trong hình 2.3.3.7b. Trên thực tế cũng có thể có một số thuộc tính khác đi kèm với liên kết "Completes" này ví dụ như kết quả tốt nghiệp Course_Grade, hay nơi tốt nghiệp Location chẳng hạn.

3.4. Lựa chọn giữa kiểu thực thể và thuộc tính

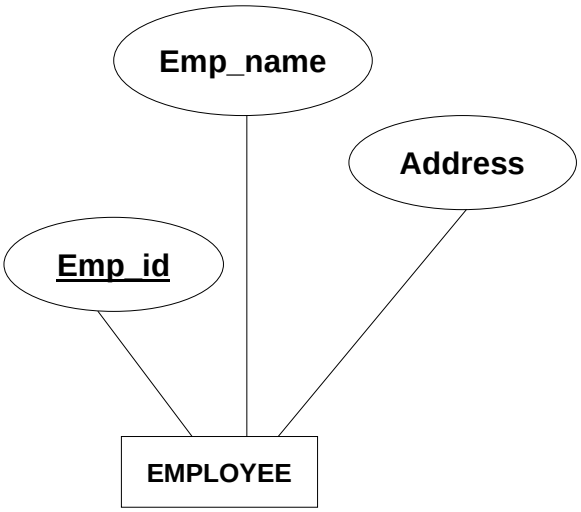
Trên đây chúng ta đã biết cách lựa chọn giữa các kiểu thực thể với nhau, giữa các liên kết với nhau, giữa các thuộc tính với nhau. Nhưng chúng ta vẫn chưa biết khi nào thì hình thành một thuộc tính hay khi nào thì hình thành một kiểu thực thể? Trong phần này chúng tôi sẽ giúp các bạn thực hiện điều đó.

Trong quá trình xác định các thuộc tính cho một tập thực thể, đôi lúc chúng ta sẽ gặp vấn đề là liệu một tính chất nào đó nên được mô hình hoá thành một thuộc tính hay thành một kiểu thực thể và nếu vậy thì nó quan hệ với các kiểu thực thể khác thông qua các liên kết như thế nào?

Ví dụ: Với bài toán quản lý doanh nghiệp, khi xem xét đến thông tin về địa chỉ của nhân viên, chúng ta phải đứng trước hai lựa chọn: chúng ta sẽ chỉ mô hình hoá nó như là một thuộc tính Address nằm trong kiểu thực thể EMPLOYEE?; hay chúng ta sẽ mô hình hoá nó thành một kiểu thực thể địa chỉ ADDRESS, và nếu vậy nó sẽ quan hệ với kiểu thực thể EMPLOYEE như thế nào?

Lựa chọn thứ nhất: Nếu bài toán không yêu cầu chúng ta cần phải quản lý đến các lần thay đổi địa chỉ của nhân viên

thì khi đó có thể coi địa chỉ như là một thuộc tính mô tả nằm trong kiểu thực thể nhân viên EMPLOYEE (hình 2.3.4.1).

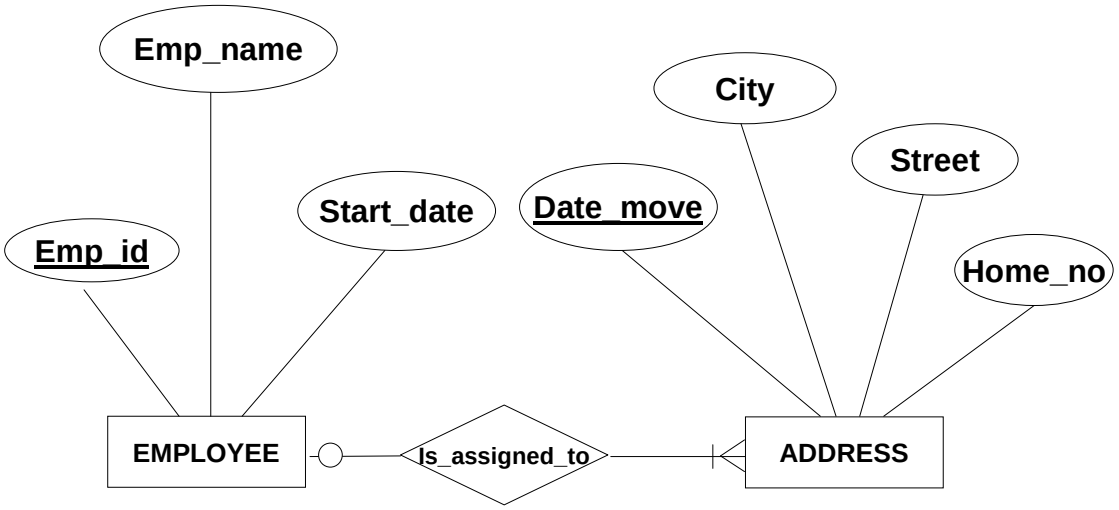


Hình 2.3.4.1 Hình thành một thuộc tính trong kiểu thực thể

Lựa chọn thứ hai: Ngược lại, nếu bài toán yêu cầu chúng ta phải quản lý cả những lần nhân viên thay đổi địa chỉ chỗ ở cũng như thời gian thay đổi thì khi đó chúng ta nên hình thành một kiểu thực thể mới, đặt tên là ADDRESS chẳng hạn, và kiểu thực thể này sẽ được kết nối với kiểu thực thể nhân viên EMPLOYEE theo kiểu liên kết một - nhiều (hình 2.3.4.2).

Như vậy, chúng ta sẽ chọn lựa chọn thứ hai này khi phải đáp ứng 2 tình huống sau:

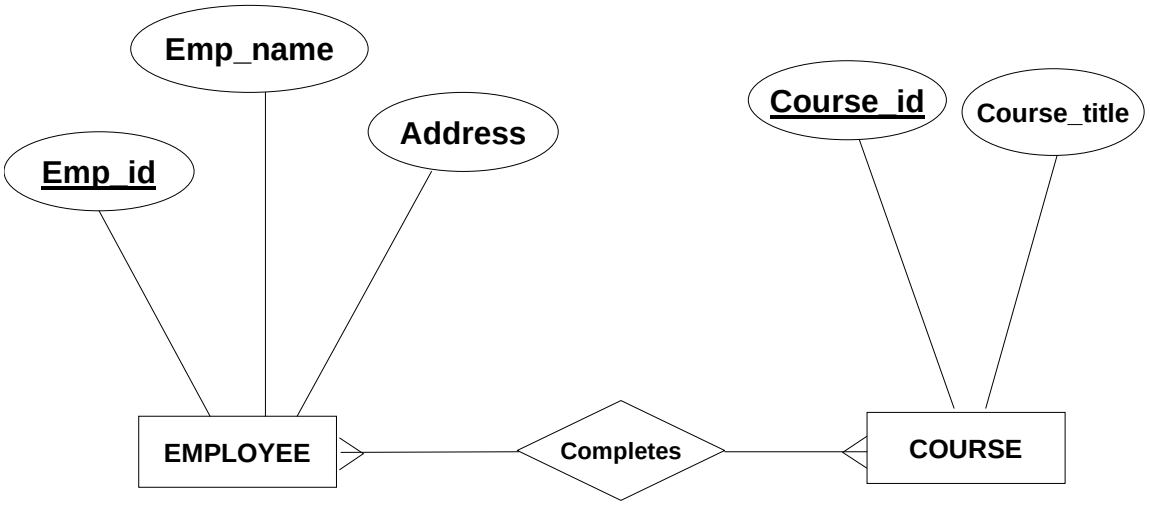
- Trên thực tế có những trường hợp chúng ta phải lưu trữ nhiều hơn một địa chỉ đối với mỗi một nhân viên.
- Chúng ta muốn quản lý chi tiết hơn nữa đối với thông tin địa chỉ như ngày chuyển đến địa chỉ mới, …



Hình 2.3.4.2 Hình thành một kiểu thực thể mới

3.5. Lựa chọn giữa kiểu thực thể và liên kết

Để bắt đầu, chúng ta hãy xét một ví dụ sau: Trong bài toán quản lý nhân sự cho công ty, chúng ta muốn quản lý về tình hình nâng cao trình độ của các nhân viên trong công ty. Giả sử chúng ta đã có hai kiểu thực thể nhân viên EMPLOYEE và khoá học COURSE. Rõ ràng giữa hai kiểu thực thể này tồn tại một liên kết mô tả cho biết các nhân viên đã hoàn tất những khoá học nào. Chúng ta tạm đặt tên liên kết này là hoàn thành "Completes". Ngoài ra cũng nhận thấy rằng để có thể mô tả được đầy đủ các trường hợp, chúng ta phải sử dụng loại liên kết nhiều - nhiều, bởi mỗi một nhân viên có thể hoàn tất và tốt nghiệp nhiều khoá học, ngược lại mỗi một khoá học có thể có nhiều nhân viên đã tốt nghiệp. Chúng ta thể hiện bằng hình 2.3.5.1 sau:



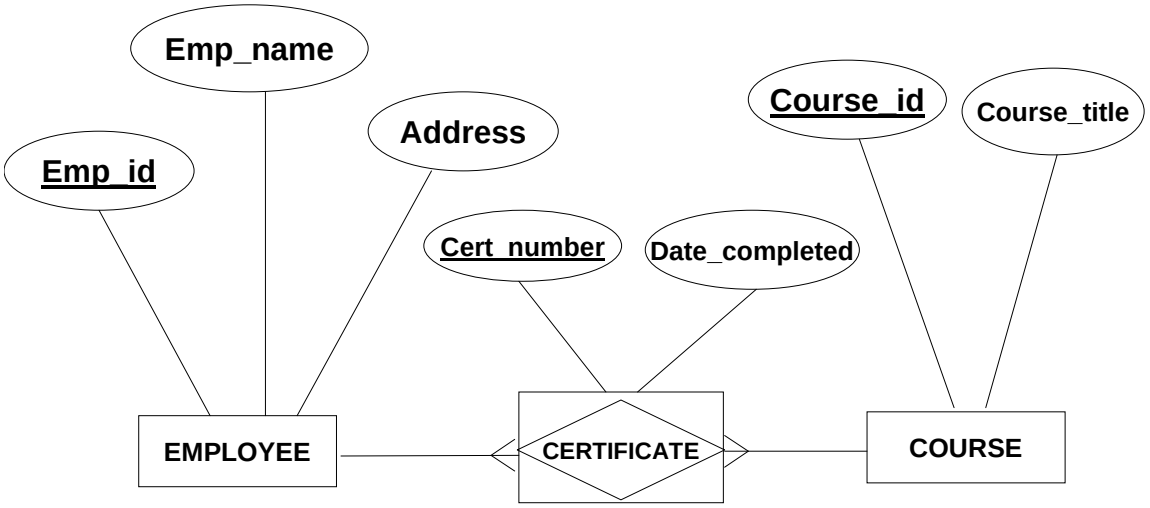
Hình 2.3.5.1

Nếu nhu cầu quản lý thông tin về tình hình nâng cao trình độ của nhân viên được mở rộng thêm, như chúng ta phải lưu lại số chứng chỉ mà mỗi một nhân viên đã nhận, cũng như ngày đã tốt nghiệp của mỗi một khoá học, thì khi đó ta phải sử dụng thêm các thuộc tính để mô tả cho liên kết "Completes" trên.

Trong liên kết nhiều - nhiều, khi phải xác định các thuộc tính cho liên kết, chúng ta hãy nghĩ đến trường hợp chuyển đổi liên kết đó thành một kiểu thực thể mới, được gọi là kiểu thực thể kết nối, nó vừa đóng vai trò là một liên kết giữa hai kiểu thực thể lại vừa đóng vai trò là một kiểu thực thể thực sự.

Quay trở lại ví dụ trên, với liên kết "Completes", rõ ràng chúng ta phải sử dụng hai thuộc tính là số chứng chỉ Cert_number và ngày tốt nghiệp Date_completed để mô tả cho liên kết này, và do vậy chúng ta nên hình thành một kiểu thực thể chứng chỉ CERTIFICATE cùng với hai thuộc tính đó.

Chúng ta biểu diễn kiểu thực thể này bằng một hình thoi (như một liên kết) và bên ngoài là một hình chữ nhật (như một kiểu thực thể) như hình 2.3.5.2 dưới đây:



Hình 2.3.5.2 Hình thành kiểu thực thể kết nối

Vậy thì khi nào thì chúng ta nên chuyển đổi một liên kết thành một kiểu thực thể kết nối? Dưới đây là bốn điều kiện có thể xem xét để chuyển đổi:

- Liên kết giữa các kiểu thực thể là liên kết nhiều-nhiều.
- Kiểu thực thể kết nối được chuyển đổi thường được xác định bởi một định danh đơn (gồm một thuộc tính).
- Trong liên kết có thêm một số thuộc tính mô tả, ngoài thuộc tính định danh.
- Khi hình thành lên kiểu thực thể kết nối thì kiểu thực thể này độc lập với các kiểu thực thể được nối bởi liên kết đó.

Như ví dụ trên, việc chuyển đổi liên kết “Completes” thành một kiểu thực thể kết nối CERTIFICATE cho biết mỗi một nhân viên hoàn thành một khoá học đều được cấp một chứng chỉ, và mỗi một chứng chỉ đều có một số Cert_number duy nhất, do vậy thuộc tính này sẽ trở thành định danh cho kiểu thực thể kết nối mới, cùng với thuộc tính mô tả khác như Date_completed.

Chúng ta cũng phải lưu ý rằng, khi chuyển đổi liên kết thành kiểu thực thể kết nối thì ký hiệu cho liên kết cũng thay đổi theo. Khi đó chiều liên kết nhiều xuất hiện ở hai kiểu thực thể ban đầu được chuyển thành các liên kết nhiều xuất phát từ kiểu thực thể kết nối vừa hình thành đến hai kiểu thực thể còn lại.

3.6. Các phân cấp “là-một”

Trên thực tế, đôi khi chúng ta gặp những trường hợp mà một tập thực thể A là một tập con của một tập thực thể B, nó có những tính chất chung với tập thực thể B, nhưng lại có một số tính chất của riêng nó. Khi đó chúng ta gọi kiểu thực thể B là kiểu thực thể gốc (supertype), kiểu thực thể A là kiểu thực thể con (subtype).

Ví dụ, trong bài toán quản lý doanh nghiệp, chúng ta có kiểu thực thể nhân viên EMPLOYEE, tuy nhiên trong số những thực thể thuộc kiểu EMPLOYEE này có những nhân viên làm việc hưởng lương theo hợp đồng hàng năm, có những nhân viên làm việc hưởng lương theo giờ, lại có những nhân viên làm việc theo hình thức là chuyên gia tư vấn. Họ đều là nhân viên của công ty, đều có số hiệu nhân viên, ngày bắt đầu tham gia vào làm việc tại công ty. Nhưng đối với mỗi hình thức làm việc đó, họ lại có thể

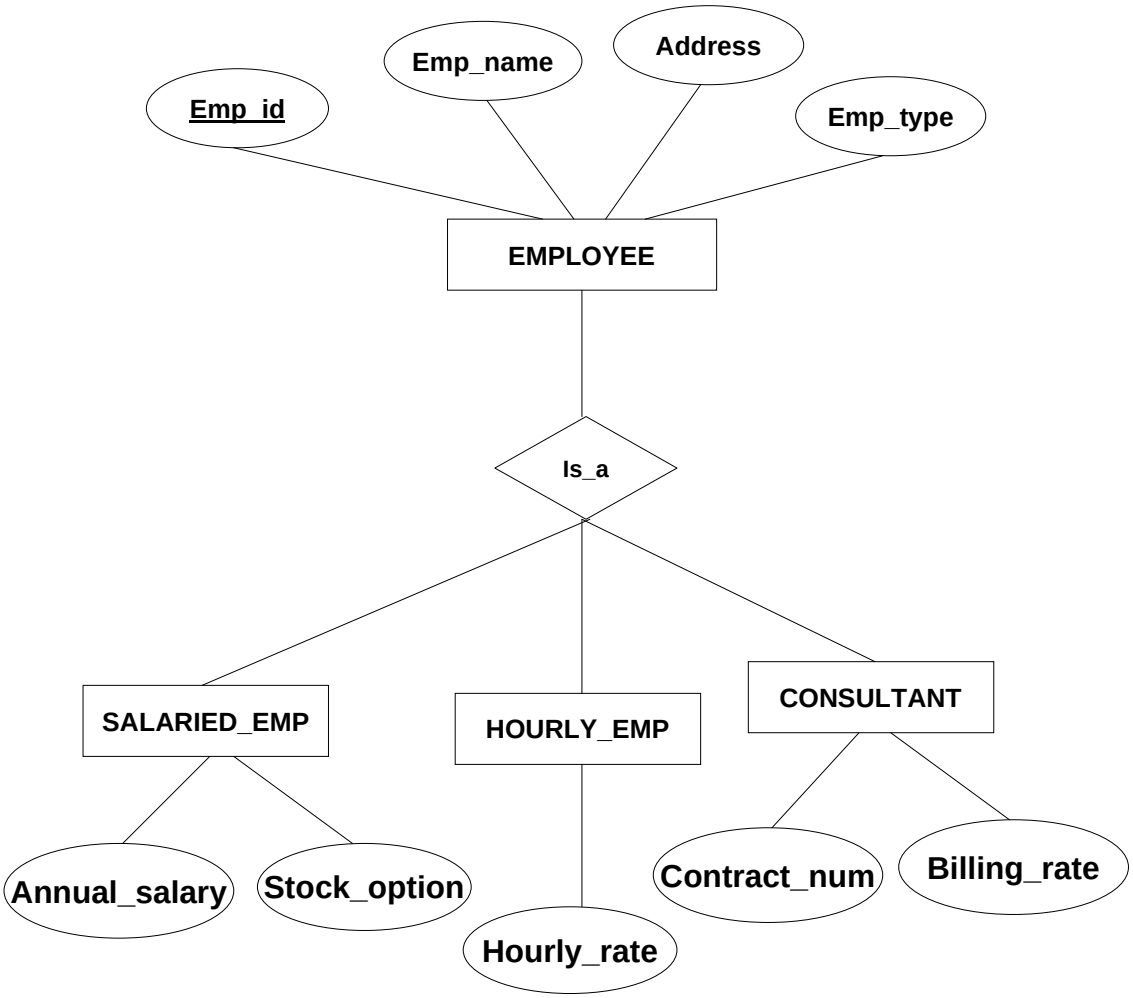
hình thành lên những kiểu thực thể mới và kết nối với kiểu thực thể EMPLOYEE theo một liên kết mà chúng ta qui ước gọi là liên kết “là-một” hay phân cấp “là-một”.

Một cách hình thức, chúng ta nói A “là-một” B nếu kiểu thực thể B là sự tổng quát hoá của kiểu thực thể A và kiểu thực thể A là sự chuyên biệt hoá của kiểu thực thể B.

Khi chúng ta hình thành liên kết “là-một” giữa hai kiểu thực thể A và B thì A sẽ kế thừa các thuộc tính của B, nhưng A cũng có thể bổ sung thêm một số thuộc tính của riêng mình mà B không có, tức là các thực thể thuộc B nhưng không thuộc A không có những thuộc tính này.

Về mặt hình thức, một thực thể a trong kiểu thực thể A có quan hệ với chính xác một thực thể b trong kiểu thực thể B, sao cho a và b thực sự là cùng một thực thể trong thế giới thực. Và chúng ta cũng cần lưu ý rằng, không có một thực thể b nào trong tập B có quan hệ với hai thực thể khác nhau của A. Các thuộc tính khoá của kiểu thực thể A thực sự cũng là các thuộc tính khoá của kiểu thực thể B, và các giá trị của những thuộc tính này đối với một thực thể a trong A được lấy từ b tương ứng trong B.

Quay trở lại ví dụ trên, một cách tốt nhất đối với việc thiết kế đó là chúng ta có thêm các kiểu thực thể mới: nhân viên hưởng lương theo hợp đồng hàng năm SALARIED_EMP, nhân viên hưởng lương theo giờ HOURLY_EMP và cộng tác viên CONSULTANT, tất cả các kiểu thực thể này đều liên kết “là-một” với kiểu thực thể EMPLOYEE như hình 2.3.6.1 sau:



Hình 2.3.6.1 Phân cấp hay liên kết "là-một"

Với kiểu thực thể SALARIED_EMP, chúng ta có thêm các thuộc tính như mức lương hàng năm Annual_salary, số cổ phần nắm giữ trong công ty Stock_option. Với kiểu thực thể HOURLY_EMP có thêm thuộc tính mức lương tính theo từng giờ Hourly_rate. Với kiểu thực thể CONSULTANT chúng ta có thêm các thuộc tính thuê theo số hợp đồng Contract_num, mức lương theo hợp đồng Billing_rate.

Một ví dụ khác: Trong công ty chúng ta có một kiểu thực thể nhân viên EMPLOYEE với các thuộc tính như Emp_id, Emp_name, Address, Start_date. Nếu công ty có một đội bóng thì rõ ràng một số nhân viên trong công ty sẽ là những cầu thủ của đội bóng, và họ có thể có thêm một số thuộc tính riêng như thể lực Physical_Strength, tốc độ Speed, vị trí Position,... mà những nhân viên khác không có. Nếu chúng ta thêm những thuộc tính này vào kiểu thực thể EMPLOYEE trên thì sẽ có rất nhiều những nhân viên khác do không tham gia vào đội bóng sẽ không có các giá trị ở những thuộc tính này. Do vậy, cách tốt nhất đối với việc thiết kế cơ sở dữ liệu là chúng ta nên có thêm một kiểu thực thể mới là BALL_PLAYER và một liên kết "là-một" kết nối đến kiểu thực thể EMPLOYEE.

3.7. Một sơ đồ thực thể liên kết

Với các hướng dẫn trên, đến đây chúng ta đã có thể tiến hành xây dựng một sơ đồ thực thể - liên kết hoàn chỉnh cho quá trình thiết kế cơ sở dữ liệu mức khái niệm.

Quay trở lại ví dụ về công ty kinh doanh đồ gỗ đã trình bày trong chủ đề 1, trong phần này chúng ta sẽ tiến hành xây dựng một sơ đồ thực thể - liên kết đầy đủ cho nó với việc áp dụng các nguyên tắc đã trình bày trong các phần trên.

Qua phân tích bài toán chúng ta thấy hình thành lên các kiểu thực thể sau:

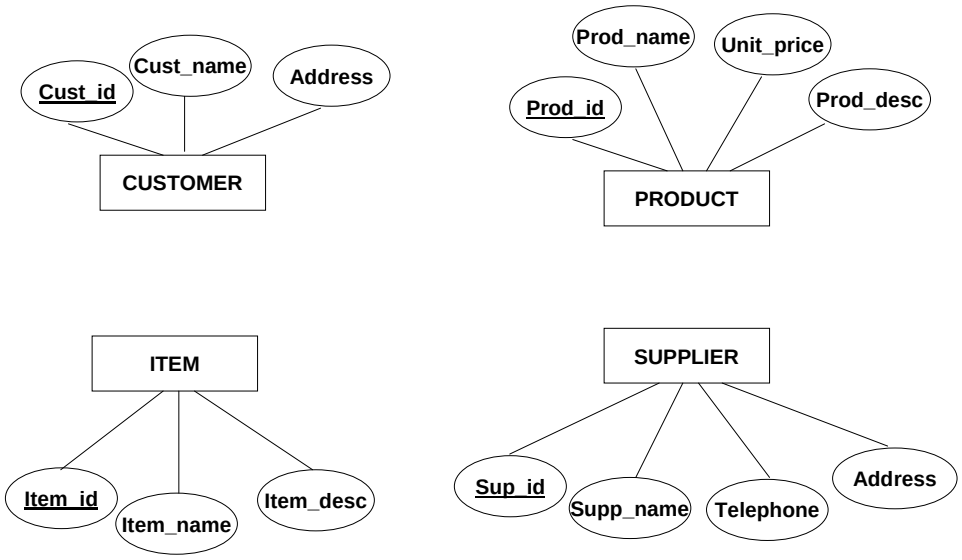
- Rõ ràng các khách hàng là những đối tác chính của công ty và công ty cũng có nhu cầu quản lý họ. Do vậy chúng ta sẽ có một kiểu thực thể đầu tiên đó là CUSTOMER, với mỗi khách hàng chúng ta cần phải

quản lý những thuộc tính sau: Tên khách hàng Cust_name, địa chỉ Address, số thẻ tín dụng Credit_card, và đối với mỗi một khách hàng chúng ta phải đặt cho họ một mã số duy nhất, chúng ta gọi là Cust_id, thuộc tính này sẽ là thuộc tính khoá của kiểu thực thể CUSTOMER.

- Công ty sản xuất và bán các sản phẩm khác nhau cho khách hàng, do vậy các sản phẩm cũng sẽ trở thành một kiểu thực thể, chúng ta đặt tên là PRODUCT. Với mỗi sản phẩm chúng ta cần quản lý tên sản phẩm Prod_name, đơn giá Unit_price, mô tả về sản phẩm Prod_desc. Cũng như kiểu thực thể CUSTOMER, với mỗi sản phẩm chúng ta cũng cần phải phân biệt chúng, do vậy chúng ta sẽ sử dụng thêm một thuộc tính Prod_id làm thuộc tính khoá cho PRODUCT.
- Để sản xuất ra các sản phẩm, công ty phải thu mua các nguyên vật liệu từ các nhà cung cấp khác nhau. Nên chúng ta sẽ có hai kiểu thực thể là nhà cung cấp SUPPLIER và nguyên vật liệu ITEM. Với kiểu thực thể SUPPLIER chúng ta cần quản lý mã nhà cung cấp Sup_id, tên nhà cung cấp Sup_name, địa chỉ Address, số điện thoại Telephone, trong đó thuộc tính Sup_id là thuộc tính khoá để phân biệt giữa các nhà cung cấp với nhau. Với kiểu thực thể ITEM chúng ta cần quản lý mã nguyên vật liệu Item_id, tên nguyên vật liệu Item_name và mô tả Item_desc, chúng ta cũng xác định thuộc tính Item_id sẽ là thuộc tính khoá của kiểu thực thể này.

Như vậy qua việc phân tích trên, chúng ta tạm thời xác định được 4 kiểu thực thể ban đầu là: CUSTOMER, PRODUCT,

SUPPLIER và ITEM cùng các thuộc tính của chúng như hình 2.3.1.7 sau:

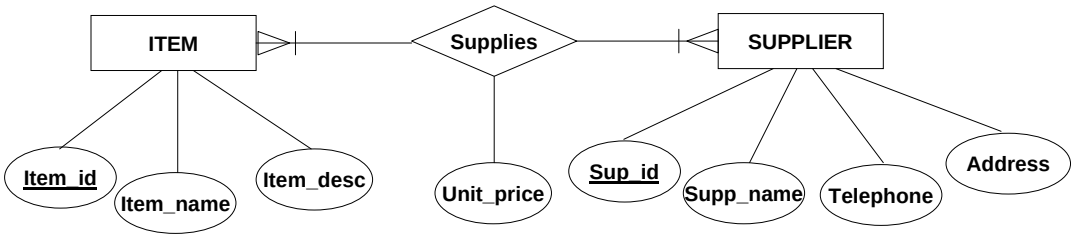


Hình 2.3.7.1 Bốn kiểu thực thể ban đầu

Chúng ta hãy tiếp tục tiến hành phân tích các mối quan hệ giữa chúng, ở đây xin lưu ý, trong quá trình phân tích để tìm ra các liên kết chũng ta vẫn có thể hình thành lên các kiểu thực thể mới như các phần trên đã trình bày.

- Để đơn giản, chúng ta hãy bắt đầu từ việc công ty mua các nguyên vật liệu từ các nhà cung cấp. Chúng ta nhận thấy rằng đây sẽ là một liên kết nhiều-nhiều, bởi mỗi một nguyên vật liệu, công ty có thể mua từ một số các nhà cung cấp khác nhau, ngược lại với mỗi một nhà cung cấp, họ có thể cung cấp cho công ty một số nguyên vật liệu. Chúng ta cũng nhận thấy rằng với liên kết nhiều-nhiều này, sẽ có một thuộc tính mô tả đi kèm, đó là thuộc tính đơn giá Unit_price do có thể cùng một nguyên vật liệu nhưng các nhà

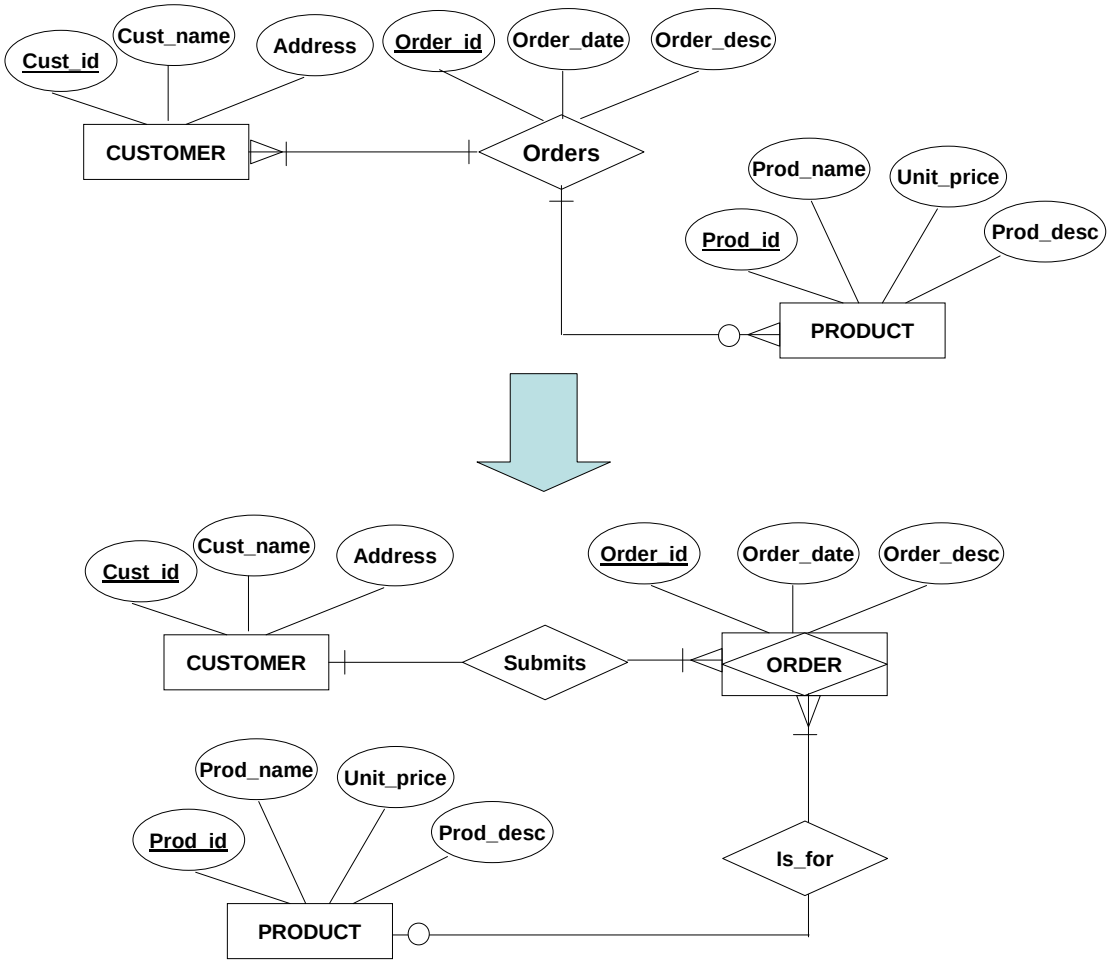
cung cấp khác nhau có thể có các giá chào bán khác nhau. Chúng ta đặt tên cho liên kết giữa hai kiểu thực thể SUPPLIER và ITEM này là cung cấp "Supplies". Về ràng buộc của liên kết, chúng ta phân tích và lựa chọn như sau: bởi công ty sẽ không quản lý những nhà cung cấp mà không bán một nguyên vật liệu nào cho công ty nên đầu liên kết với nhà cung cấp SUPPLIER sẽ là loại ràng buộc bắt buộc. Ngược lại, với các nguyên vật liệu, có thể sẽ có những thứ công ty không phải mua từ các nhà cung cấp, đó có thể là những nguyên vật liệu bán thành phẩm được tạo ra trong quá trình sản xuất ra sản phẩm cuối cùng, do vậy đầu liên kết với ITEM sẽ là loại ràng buộc tuỳ chọn. Hình 2.3.7.2 sau minh hoạ cho bước phân tích này:



Hình 2.3.7.2 Liên kết "Supplies"

- Bây giờ chúng ta hãy phân tích với một quan hệ phức tạp hơn, đó là quan hệ giữa khách hàng và công ty. Rõ ràng khách hàng muốn mua hàng của công ty thì họ phải đặt mua các sản phẩm đó. Như vậy chúng ta sẽ có một liên kết đặt mua "Orders" giữa khách hàng và sản phẩm. Chúng ta cũng nhận thấy rằng đây là một liên kết nhiều-nhiều bởi một khách hàng có thể

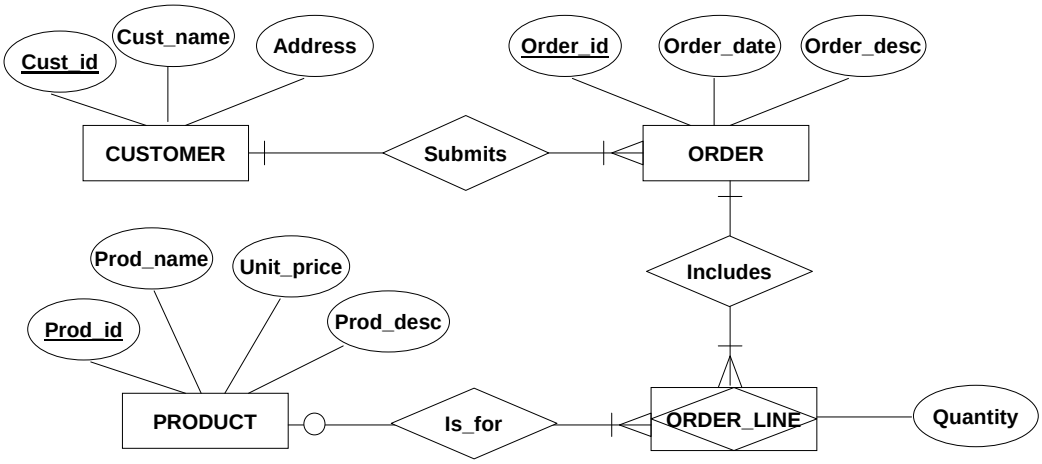
đặt mua nhiều sản phẩm cũng như một sản phẩm có thể được nhiều khách hàng khác nhau cùng mua. Cụ thể hơn, trên thực tế, đối với mỗi một giao dịch mua sản phẩm, công ty đều phải sử dụng một hoá đơn bán hàng, mỗi một hoá đơn hàng đều có một số hiệu duy nhất, cùng ngày tháng của đơn hàng. Do vậy, như trên đã trình bày, đến đây chúng ta có thể nghĩ ngay đến trường hợp chuyển đổi liên kết nhiều-nhiều này thành một kiểu thực thể kết nối, và đặt tên nó là đơn hàng ORDER như hình 2.3.7.3 sau:



Hình 2.3.7.3 Hình thành kiểu thực thể kết nối ORDER

- Phân tích tiếp nghiệp vụ này, trong mỗi hoá đơn bán hàng, khách hàng sẽ liệt kê các sản phẩm mà họ muốn đặt mua. Đi kèm với mỗi một sản phẩm sẽ là đơn giá, số lượng và thành tiền (trong đó thành tiền là thuộc tính dẫn xuất từ hai thuộc tính đơn giá và số lượng). Chúng ta thấy rằng nếu thiết kế như trên chỉ với một kiểu thực thể ORDER thì với mỗi sản phẩm mà khách hàng đặt mua, giá trị hai thuộc tính số hiệu đơn hàng và ngày đơn hàng sẽ bị lặp lại ở mỗi một

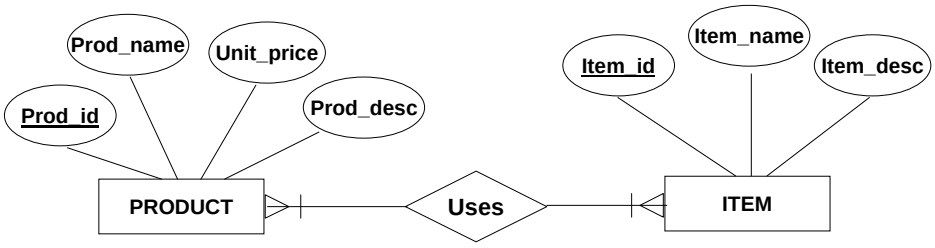
dòng đặt mua sản phẩm, mặc dù nó là duy nhất đối với mỗi một đơn hàng. Do vậy chúng ta nên tách kiểu thực thể này thành hai kiểu thực thể, một là đơn hàng gốc ORDER lưu số hiệu đơn hàng Order_id, ngày đơn hàng Order_date (có thể có thêm thuộc tính diễn giải cho đơn hàng Order_desc), và một kiểu thực thể là dòng đơn hàng ORDER_LINE. Lưu ý là trong ORDER_LINE chúng ta không hình thành thuộc tính đơn giá bởi trong kiểu thực thể PRODUCT chúng ta đã có thuộc tính này rồi, do vậy khi khách hàng lựa chọn một sản phầm đơn giá đi theo sản phẩm đó sẽ xuất hiện, cũng như thuộc tính thành tiền là thuộc tính dẫn xuất nên chúng ta có thể lựa chọn tạo ra hoặc không cũng được. (Cũng cần lưu ý rằng, nếu giá các sản phẩm có sự thay đổi theo thời gian và công ty có nhu cầu quan tâm đến những đơn hàng đã kết thúc thì chúng ta nên tạo thêm hai thuộc tính này). Ở đây để đơn giản, chúng ta giả thiết kiểu thực thể kết nối ORDER_LINE chỉ có một thuộc tính mô tả, đó là số lượng Quantity. Mô tả bước phân tích này được thể hiện trong hình 2.3.7.4 sau:



Hình 2.3.7.4 Hình thành kiểu thực thể dòng đơn hàng ORDER_LINE

- Rõ ràng liên kết giữa ORDER và ORDER_LINE là liên kết một-nhiều như phân tích trên, còn với loại ràng buộc của liên kết thì chúng ta đều phải lựa chọn loại ràng buộc là bắt buộc ở cả hai đầu của liên kết, bởi một thực thể là đơn hàng gốc a trong kiểu thực thể ORDER luôn phải có ít nhất một thực thể b là sản phẩm đặt mua trong kiểu thực thể ORDER_LINE kết nối với nó. Ngược lại, mỗi một dòng đặt mua sản phẩm b phải thuộc một đơn hàng a. Chúng ta đặt tên liên kết này là bao gồm "Includes".
- Liên kết giữa khách hàng và đơn hàng chúng ta đặt tên là đặt mua "Submits", và do công ty cũng sẽ không quản lý những khách hàng mà không bao giờ đặt mua hàng của công ty nên đầu liên kết phía kiểu thực thể CUSTOMER sẽ có kiểu ràng buộc là bắt buộc. Và ngược lại, một đơn hàng cũng phải có một khách hàng đặt nên đầu liên kết phía kiểu thực thể ORDER cũng sẽ có kiểu ràng buộc là bắt buộc.

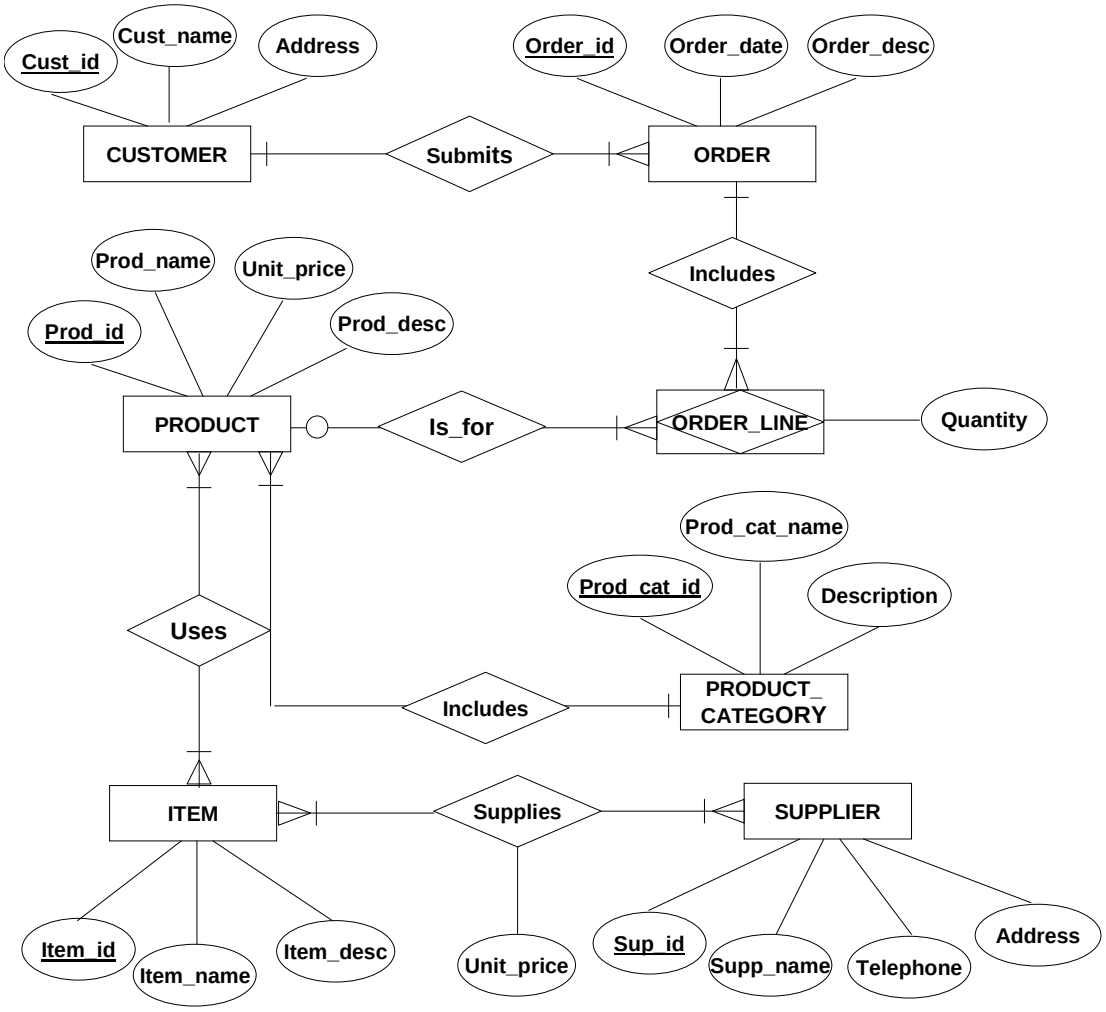
- Do chúng ta đã chuyển đổi liên kết đặt mua hàng "Orders" trên thành kiểu thực thể kết nối (ORDER và ORDER_LINE), nên liên kết "Is_for" giữa kiểu thực thể PRODUCT và dòng đơn hàng ORDER_LINE sẽ được chuyển thành liên kết một-nhiều. Rõ ràng với một dòng đơn hàng trong ORDER_LINE phải tồn tại một sản phẩm kết nối từ PRODUCT nên loại ràng buộc cho đầu liên kết với ORDER_LINE sẽ là bắt buộc. Ngược lại, ở đầu liên kết phía PRODUCT, do có thể có những sản phẩm không được đặt mua ở bất kỳ một đơn hàng nào nên loại ràng buộc sẽ là tuỳ chọn. Đến đây chúng ta đã chuyển đổi xong cho liên kết nhiều-nhiều giữa hai kiểu thực thể khách hàng CUSTOMER và sản phẩm PRODUCT thành một kiểu thực thể kết nối.
- Tiếp theo chúng ta hãy phân tích mối quan hệ giữa sản phẩm PRODUCT và nguyên vật liệu ITEM. Chúng ta nhận thấy rằng đây cũng là một liên kết nhiều-nhiều, bởi mỗi một sản phẩm có thể được làm từ nhiều nguyên vật liệu khác nhau và ngược lại, mỗi một nguyên vật liệu có thể được sử dụng trong nhiều sản phẩm khác nhau. Chúng ta đặt tên cho liên kết này là sử dụng "Uses". Loại ràng buộc đối với hai đầu của liên kết cũng sẽ là loại ràng buộc bắt buộc, bởi mỗi một nguyên vật liệu mua về công ty đều phải sử dụng đến để tạo ra các sản phẩm, ngược lại mỗi một sản phẩm đều phải được tạo từ các nguyên vật liệu khác nhau.



Hình 2.3.7.5 Liên kết giữa PRODUCT và ITEM

- Nếu công ty có nhu cầu phân loại các sản phẩm bán ra theo từng nhóm, như các sản phẩm chuyên về bàn làm việc, các sản phẩm chuyên về giá sách, các sản phẩm chuyên về ghế ngồi,… thì chúng ta sẽ có thêm một kiểu thực thể mới đó là PRODUCT_CATEGORY, kiểu thực thể này sẽ liên kết với kiểu thực thể PRODUCT trên theo dạng liên kết một-nhiều. Do mỗi một sản phẩm phải thuộc vào một nhóm nhất định và mỗi một nhóm có thể có nhiều sản phẩm cùng loại. Chúng ta đặt tên liên kết này là bao gồm "Includes". Hai đầu của liên kết đều là loại ràng buộc bắt buộc, bởi công ty qui định mỗi một sản phẩm phải thuộc vào một nhóm, cũng như trong một nhóm phải có ít nhất một sản phẩm.

Cuối cùng, chúng ta có một sơ đồ thực thể - liên kết đầy đủ như hình 2.3.7.6 sau:



Hình 2.3.7.6 Sơ đồ thực thể - liên kết đầy đủ

3.8. Mô tả các ràng buộc trên dữ liệu

Việc xác định và mô tả các ràng buộc trên dữ liệu sẽ được thực hiện dựa trên sơ đồ thực thể - liên kết và các luật nghiệp vụ

Trong chủ đề 1, chúng ta đã biết ràng buộc dữ liệu là những luật mà không thể bị xâm phạm bởi người sử dụng cơ sở dữ liệu. Các ràng buộc sẽ đảm bảo tính toàn vẹn và nhất quán của dữ liệu.

Ví dụ: Một sinh viên chỉ có thể đăng ký học tiếp một học phần khác nếu anh ta đã hoàn thành xong một số học phần nào đó. Hay một khách hàng sẽ được giảm giá 10% khi anh ta không bị quá hạn thanh toán. Chúng ta thấy rằng, một số giá trị của một thuộc tính nào đó sẽ phải phụ thuộc vào các giá trị của một số thuộc tính khác.

Ví dụ: Bậc lương của mỗi một nhân viên chỉ có thể nhận các giá trị từ 1 đến 7. Trong luật nghiệp vụ này, chúng ta thấy rằng các giá trị mà bậc lương chỉ có thể nhận là các giá trị số nằm giữa 1 và 7 mà không thể nhận các giá trị là ký tự.

Đối với mỗi một kiểu thực thể, thông thường sẽ có một tập các luật nghiệp vụ đi kèm, các luật nghiệp vụ này sẽ trở thành các ràng buộc toàn vẹn trong cơ sở dữ liệu. Những ràng buộc này về lý thuyết có thể có độ phức tạp bất kỳ, tuy rằng trong thực tế, nhiều ràng buộc lại rất đơn giản. Trong cơ sở dữ liệu quan hệ, các ràng buộc toàn vẹn thường được định nghĩa thông qua các câu lệnh SQL (sẽ được trình bày trong bước sau).

Tại bước này, chúng ta cần phải mô tả các giá trị mà mỗi một thuộc tính có thể nhận, hay nói một cách khác, đó chính là miền giá trị của thuộc tính để có thể đảm bảo rằng các giá trị đưa vào thuộc tính đó phải thoả mãn. Cũng như các luật nghiệp vụ mà chúng phải thoả mãn dưới dạng các tài liệu mô tả đi kèm với sơ đồ thực thể - liên kết.

Mỗi một thuộc tính trong sơ đồ thực thể - liên kết phải được khai báo miền xác định bao gồm các thông tin như sau:

- Kiểu dữ liệu: như kiểu số nguyên, số thập phân, kiểu ký tự, kiểu ngày tháng,… Hầu hết các cơ sở dữ liệu đều hỗ trợ các kiểu dữ liệu này.
- Độ dài: Đối với mỗi một kiều dữ liệu xác định cho thuộc tính, chúng ta cần phải xác định thêm độ dài cho kiểu dữ liệu đó. Ví dụ nếu thuộc tính tuổi có kiểu số nguyên thì các giá trị cho thuộc tính đó có thể nhận giá trị tối đa là 3 chữ số. Hay nếu một thuộc tính tên có kiểu là ký tự thì giá trị của nó có thể nhận tối đa là 30 ký tự.
- Khoảng giá trị: xác định các giá trị lớn nhất và nhỏ nhất đối với các giá trị của thuộc tính, thường được áp dụng với các kiểu dữ liệu là số và ngày tháng. Như ví dụ trên, thuộc tính bậc lương sẽ được mô tả là kiểu số có độ dài là 1 và nhận các giá trị trong khoảng từ 1 đến 7.
- Hỗ trợ giá trị null (không xác định): cho biết các giá trị của một thuộc tính có được phép nhận giá trị null hay không. Ví dụ với các thuộc tính khoá thì rõ ràng các giá trị của nó không thể nhận giá trị null. Hay với thuộc tính tên, mặc dù không tham gia vào làm khoá chính nhưng chúng ta vẫn phải mô tả cho thuộc tính này không được phép nhận giá trị null.
- Các định dạng dữ liệu: mô tả dữ liệu của thuộc tính khi đưa vào cơ sở dữ liệu phải ở định dạng nào. Ví dụ như với thuộc tính có kiểu là ngày tháng, các định dạng có thể là dd/mm/yyyy hay mm/dd/yyyy. Với thuộc tính có kiểu dữ liệu là ký tự thì có cần viết hoa tất cả các ký tự hay không.

Tất cả những thông tin mô tả ràng buộc toàn vẹn cho dữ liệu này cùng với sơ đồ thực thể - liên kết sẽ là những thông tin đầu vào cho bước thiết kế cơ sở dữ liệu mức logic. Chúng sẽ được thể hiện trong các câu lệnh định nghĩa các đối tượng cho cơ sở dữ liệu.

Như vậy, mô hình thực thể - liên kết thường được biểu diễn bằng một sơ đồ thực thể - liên kết bao gồm các kiểu thực thể (trong đó mỗi đối tượng trong thế giới thực được biểu diễn bằng một thực thể), thuộc tính và liên kết.

Một kiểu thực thể là một tập các thực thể hay đối tượng có cấu trúc giống nhau. Các thực thể được mô tả bằng các thuộc tính. Mỗi một kiểu thực thể có một hoặc một tập các thuộc tính được gọi là khoá hay định danh nhằm xác định duy nhất mỗi một thực thể. Giữa các kiểu thực thể có sự kết nối với nhau thông qua các liên kết. Các liên kết cũng chính là các ràng buộc khoá giữa các kiểu thực thể

Trong phần này này, chúng ta đã xem xét các bước mô hình hoá nhằm lựa chọn giữa các kiểu đối tượng khác nhau trong sơ đồ thực thể - liên kết. Chúng ta cần phải nắm vững được những lựa chọn này để khi đứng trước các trường hợp phức tạp có thể lựa chọn được đúng đắn đối tượng cần mô hình hoá.

Kết quả của giai đọan thiết kế cơ sở dữ liệu mức khái niệm là sơ đồ thực thể - liên kết cùng với các mô tả ràng buộc dữ liệu trên các thuộc tính. Trong các hệ quản trị cơ sở dữ liệu lớn, như Oracle, SQL Server, thường tích hợp luôn các bộ công cụ cho phép chúng ta thể hiện chung cả hai kết quả đó, do vậy khi làm quen với các hệ quản trị cơ sở dữ liệu này, chúng ta cũng nên làm quen luôn với những bộ công cụ đó,

bởi chúng hoàn toàn thích hợp và thuận lợi cho giai đoạn thiết kế mà trước đây chúng ta vẫn phải làm trên giấy.

4. Thiết kế cơ sở dữ liệu mức logic

Trong bước này, chúng ta phải tiến hành lựa chọn hệ quản trị cơ sở dữ liệu để cài đặt cho cơ sở dữ liệu, chuyển đổi các thiết kế cơ sở dữ liệu mức khái niệm ở bước trên thành các lược đồ cơ sở dữ liệu của hệ quản trị cơ sở dữ liệu đã được lựa chọn.

Nhiệm vụ chính của thiết kế cơ sở dữ liệu mức logic là chuyển đổi sơ đồ thực thể liên kết đã được thực hiện ở phần thiết kế cơ sở dữ liệu mức khái niệm thành sơ đồ cơ sở dữ liệu quan hệ mức logic.

Cụ thể, chúng ta sẽ phải tập trung vào các vấn đề sau:

- Dữ liệu sẽ được biểu diễn như thế nào?
- Loại ràng buộc toàn vẹn nào sẽ được sử dụng?
- Các bước chuyển đổi từ mô hình thiết kế mức khái niệm sang mô hình thiết kế mức logic?

4.1. Giới thiệu chung về ngôn ngữ SQL

SQL (Structured Query LanguAge) là ngôn ngữ hỏi đáp dữ liệu có cấu trúc được xây dựng đầu tiên trong hệ quản trị cơ sở dữ liệu quan hệ R-System của IBM. Ngày nay, SQL trở thành một ngôn ngữ được sử dụng rộng rãi nhất trong các hệ quản trị cơ sở dữ liệu quan hệ.

Người ta sử dụng SQL để tạo lập, thao tác và truy vấn dữ liệu. Khi số lượng các nhà phát triển tham gia vào việc xây

dựng SQL tăng lên đáng kể, người ta nhận thấy rằng cần phải tiêu chuẩn hoá SQL.

Chuẩn SQL đầu tiên được xây dựng vào năm 1986 bởi Viện tiêu chuẩn quốc gia Hoa Kỳ (ANSI) và khi đó nó được gọi là chuẩn SQL-86. Có một phiên bản phụ xuất hiện vào năm 1989 và chính thức sau đó vào năm 1992 và được gọi là SQL-92. Tổ chức tiêu chuẩn quốc tế ISO đã hợp tác với ANSI để xây dựng nên chuẩn này. Hầu hết các hệ quản trị cơ sở dữ liệu thương mại ngày nay đều hỗ trợ chuẩn SQL-92. Trong SQL người ta chia ngôn ngữ (hay các câu lệnh) ra làm các loại: ngôn ngữ định nghĩa dữ liệu DDL (Data Definition LanguAge) như các câu lệnh CREATE, ALTER, DROP; ngôn ngữ thao tác dữ liệu DML (Data Manipulation LanguAge) như các câu lệnh SELECT, INSERT, UPDATE, DELETE. Các câu lệnh liên quan đến việc phân quyền dữ liệu như GRANT, REVOKE được gọi là ngôn ngữ điều khiển dữ liệu DML (Data Control LanguAge), tuy nhiên nó cũng có thể được coi là một phần của ngôn ngữ DML.

4.2. Ràng buộc toàn vẹn

Một ràng buộc toàn vẹn là một điều kiện xác định trên một lược đồ cơ sở dữ liệu nhằm giới hạn các dữ liệu có thể được lưu trữ trong quan hệ mà nó mô tả.

Cùng với việc định nghĩa một lược đồ cơ sở dữ liệu, chúng ta phải xây dựng các ràng buộc toàn vẹn để khai báo trên các thuộc tính.

Cũng cần phải lưu ý rằng, để có thể tiến hành xây dựng các khai báo cho các thuộc tính sẽ phải sử dụng đến những mô

tả dữ liệu ở các bước trên: như thuộc tính đó có là thuộc tính khoá không?, nó có thể nhận những giá trị gì? null, kiểu ký tự, kiểu số, kiểu ngày tháng..., độ dài tối đa là bao nhiêu? hệ quản trị cơ sở dữ liệu mà chúng ta đã lựa chọn có hỗ trợ các kiểu dữ liệu đó không?

Để trả lời được các câu hỏi đó, các ràng buộc toàn vẹn phải tuân theo các qui định sau:

- Sau khi một ràng buộc được định nghĩa, bất kỳ một câu lệnh thao tác dữ liệu trên bảng không thoả mãn ràng buộc đó thì câu lệnh sẽ không được thực hiện thành công và trả lại một lỗi.
- Các ràng buộc toàn vẹn được định nghĩa trên các bảng dữ liệu và được lưu như là một phần của định nghĩa bảng trong từ điển dữ liệu, do vậy các ứng dụng thao tác trên các bảng dữ liệu đều phải tuân thủ các ràng buộc toàn vẹn đó. Cũng như vậy, khi thay đổi các ràng buộc toàn vẹn, thì chúng ta chỉ cần thay đổi ở mức cơ sở dữ liệu.
- Nếu một ràng buộc toàn vẹn được tạo ra cho một bảng và trong bảng đã tồn tại một số dữ liệu không thoả mãn ràng buộc đó thì có nghĩa ràng buộc đó không thể có hiệu lực.

Những ràng buộc toàn vẹn sau thường được hỗ trợ bởi các hệ quản trị cơ sở dữ liệu phổ biến trên thị trường hiện này:

a. Ràng buộc miền

Đó là các ràng buộc về kiểu dữ liệu mà hệ quản trị cơ sở dữ liệu hỗ trợ, ví dụ như với hệ quản trị cơ sở dữ liệu Oracle,

nó hỗ trợ khá nhiều kiểu dữ liệu, một số kiểu quen thuộc như sau:

Kiểu dữ liệu	Mô tả
VARCHAR2(n)	Kiểu ký tự chứa tối đa n ký tự. Giá trị n tối đa là bằng 4000 và tối thiểu là 1
NVARCHAR2(n)	Tương tự như kiểu dữ liệu trên nhưng được xác định theo bảng mã quốc gia.
NUMBER(p,s)	Kiểu số, trong đó p là độ dài của số và s là phần thập phân. Giá trị p có thể nhận được là từ 1 đến 38
LONG	Kiểu ký tự có thể có độ dài tới 2Gigabyte
DATE	Kiểu ngày tháng
TIMESTAMP	Kiểu ngày giờ

Các ràng buộc miền là một dạng cơ bản nhất của các ràng buộc toàn vẹn. Chúng được kiểm tra một cách dễ dàng bởi hệ thống khi có một khoản mục dữ liệu mới được đưa vào cơ sở dữ liệu.

Có thể có một số thuộc tính có cùng miền giá trị. Ví dụ các thuộc tính tên khách hàng Cust_name, tên nhà cung cấp Sup_name trong các ví dụ trên có cùng một miền giá trị, chúng đều được khai báo là các xâu ký tự có độ dài 50.

b. Ràng buộc khoá

Tất cả các khái niệm khoá: khoá chính, khoá ngoại lai, khoá duy nhất có thể được khai báo bởi các lệnh của ngôn ngữ định nghĩa dữ liệu trong SQL.

- NOT NULL: Khi một trường trong quan hệ được định nghĩa ràng buộc này thì tất cả các giá trị đựa vào trường này của quan hệ đều không được mang giá trị rỗng. Tất cả các trường khoá chính đều luôn phải thoả mãn mệnh đề ràng buộc này.
- UNIQUE KEY: Ràng buộc không cho phép có 2 giá trị giống y hệt nhau trên cột hoặc tập các cột được định nghĩa bởi mệnh đề ràng buộc loại này.
- PRIMARY KEY: Ràng buộc không cho phép 2 giá trị hoàn toàn giống nhau cũng như giá trị null được đưa vào cột hoặc tập các cột được định nghĩa thoả mãn mệnh đề ràng buộc này.
- FOREIGN KEY: Ràng buộc đòi hỏi mỗi giá trị trong cột hoặc tập các cột được định nghĩa bởi ràng buộc này phải bằng một giá trị lấy từ quan hệ liên kết tương ứng, và cột trong quan hệ liên kết tương ứng cũng phải thoả mãn các ràng buộc toàn vẹn là Unique key hoặc Primary key. Ràng buộc này đôi khi còn được gọi là ràng buộc tham chiếu. Cột được khai báo loại ràng buộc này có thể nhận giá trị null nếu nó không được khai báo cùng với mệnh đề NOT NULL.
- CHECK: Ràng buộc kiểm tra điều kiện dữ liệu đưa vào phải thoả mãn một biểu thức logic được định nghĩa trong ràng buộc.

Khoá được sử dụng trong định nghĩa của một số kiểu của ràng buộc toàn vẹn. Một khoá là một cột hoặc một tập các cột được bao gồm trong định nghĩa của một số kiểu ràng buộc toàn vẹn cụ thể nào đó. Khoá mô tả mối liên kết giữa các bảng dữ liệu khác nhau và các các cột của một cơ sở dữ liệu quan hệ. Các giá trị riêng rẽ trong một khoá được gọi là giá trị khoá.

Ví dụ: Khi tạo lập quan hệ EMPLOYEE_SKILL, chúng ta định nghĩa các ràng buộc khoá chính PRIMARY KEY cho hai thuộc tính Emp_id và Skill, ràng buộc khoá ngoại lai cho thuộc tính Emp_id tham chiếu đến khoá chính trong quan hệ EMPLOYEE như câu lệnh sau:

```
CREATE TABLE EMPLOYEE_SKILL (Emp_id   char(10),
                             Skill    Number(1) NOT NULL
                             PRIMARY KEY (Emp_id,Skill),
                             FOREIGN KEY (Emp_id)
REFERENCES EMPLOYEE)
```

c. Các kích hoạt

Ví dụ, một luật nghiệp vụ qui định rằng, việc giảm giá cho một khách hàng không được phép vượt quá 10% giá trí mặt hàng, nếu có một thao tác thêm mới hoặc sửa đổi vượt quá giá trị này sẽ phải được phòng tránh. Hay các luật qui định một sinh viên chỉ được phép ghi danh học tiếp một môn mới ở khoá tiếp theo sau khi anh ta đã hoàn tất ít nhất 5 môn học của khoá trước. Rõ ràng, với những luật nghiệp vụ này thì những loại ràng buộc trên là rất khó đáp ứng. Hầu hết các hệ quản trị cơ sở dữ liệu ngày nay đều cung cấp một

cơ chế để định nghĩa các loại ràng buộc thao tác này thông qua các kích hoạt (trigger).

Như vậy chúng ta có thể hiểu kích hoạt là một chỉ thị được thực hiện một cách tự động bởi hệ thống như một tác động phụ của một thay đổi đối với cơ sở dữ liệu. Các kích hoạt đôi khi còn được gọi là các luật sự kiện-điều kiện-hành động ECA, chúng khác biệt với các kiểu ràng buộc được trình bày ở trên như sau:

- Các kích hoạt chỉ được kiểm tra khi các sự kiện, được đặc tả bởi người thiết kế cơ sở dữ liệu, xuất hiện. Các kiểu sự kiện được cho phép thường là thêm, xoá hay sửa đổi đối với quan hệ.
- Thay cho việc ngăn chặn ngay lập tức sự kiện vừa được phát ra, một kích hoạt kiểm tra sẽ được thực hiện, nếu điều kiện này không thoả mãn thì không một hành động nào kết hợp với kích hoạt này xảy ra đáp lại sự kiện đó. Ngược lại, nếu điều kiện của kích hoạt được thoả mãn, hành động kết hợp với kích hoạt được thực hiện bởi hệ quản trị cơ sở dữ liệu. Hành động này có thể ngăn cấm sự kiện đó hoặc thực hiện một loạt các hành động khác đi kèm với sự kiện đó.

Các kích hoạt được chia ra thành hai loại chính: loại được thực hiện trước khi thao tác trên dữ liệu xảy ra BEFORE và loại được thực hiện sau khi thao tác trên dữ liệu xảy ra AFTER.

Các thao tác trên dữ liệu mà các kích hoạt được phép can thiệp đó là các lệnh: INSERT, UPDATE, DELETE.

Cũng cần phải lưu ý rằng, mặc dù các kích hoạt cho phép chúng ta định nghĩa và áp đặt các luật toàn vẹn dữ liệu,

song chúng không giống hoàn toàn như một ràng buộc toàn vẹn, bởi các kích hoạt không thể kiểm tra được những dữ liệu đã được đưa vào bảng trước đó. Do vậy, chúng ta chỉ nên sử dụng đến các kích hoạt khi mà không thể sử dụng được các loại ràng buộc toàn vẹn trên.

4.3. Lựa chọn hệ quản trị CSDL

Việc lựa chọn hệ quản trị cơ sở dữ liệu sẽ phải được thực hiện ngay từ đầu trong giai đọan thiết kế mức logic, bởi vì các hệ quản trị cơ sở dữ liệu ngày nay tuy đều hỗ trợ ngôn ngữ tiêu chuẩn SQL-92 nhưng đối với mỗi một hệ quản trị cơ sở dữ liệu cụ thể, sẽ có thể sử dụng một số cú pháp khác nhau.

Ngoài ra, đối với mỗi một hệ quản trị cơ sở dữ liệu khác nhau, chúng có thể sẽ hỗ trợ những kiểu dữ liệu khác nhau ngoài những kiểu cơ bản như ký tự, số, ngày tháng,..., ngay cả đối với các kiểu cơ bản này, chúng cũng sẽ có cú pháp cũng như độ dài hỗ trợ tối đa khác nhau tuỳ thuộc vào từng hệ quản trị cơ sở dữ liệu.

Ví dụ, với Visual Foxpro có hỗ trợ kiểu dữ liệu memo dùng để lưu những trường có giá trị vô cùng lớn, như một tệp văn bản chẳng hạn, trong khi đó hệ quản trị cơ sở dữ liệu Oracle lại hỗ trợ một kiểu tương tự là kiểu long có thể chứa tới 2Gigabyte dữ liệu.

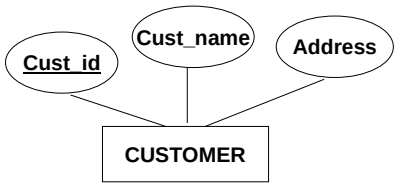
Với Oracle có hỗ trợ thêm các kiểu NVARCHAR2 để lưu các trường kiểu ký tự nhưng theo các vùng lãnh thổ khác nhau tối đa là 4000 ký tự phụ thuộc vào tập ký tự được sử dụng. Các kiểu CLOB, BLOB có thể lưu các tệp kiểu ký tự, nhị phân lên đến 4GB.

Sau khi lựa chọn xong hệ quản trị cơ sở dữ liệu chúng ta sẽ tiến hành chuyển đổi sơ đồ thực thể - liên kết sang mô hình dữ liệu quan hệ, cụ thể là các quan hệ logic sẽ được tạo ra trong bước này. Ngôn ngữ SQL sử dụng thuật ngữ bảng "TABLE" để thay cho thuật ngữ quan hệ "RELATION".

4.4. Bước một: Chuyển đổi các kiểu thực thể thông thường

Mỗi một kiểu thực thể trong sơ đồ thực thể liên kết được chuyển đổi thành một quan hệ. Tên đặt cho quan hệ có thể lấy trùng với tên đã đặt cho kiểu thực thể. Mỗi một thuộc tính của kiểu thực thể sẽ trở thành một thuộc tính (hay trường) của quan hệ. Các thuộc tính tham gia vào định danh của kiểu thực thể sẽ trở thành thuộc tính khoá chính trong quan hệ.

Hình 2.4.4.1 sau minh hoạ một ánh xạ từ kiểu thực thể CUSTOMER thành quan hệ CUSTOMER tương ứng.



CUSTOMER

Cust_id	Cust_name	Address

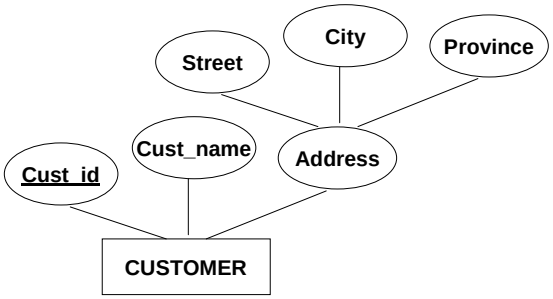
Hình 2.4.4.1 Chuyển đổi từ kiểu thực thể CUSTOMER thành quan hệ CUSTOMER

Câu lệnh SQL sau sẽ thể hiện bước chuyển đổi trên (bao gồm cả các ràng buộc về miền giá trị đối với mỗi thuộc tính và khoá):

```
CREATE TABLE CUSTOMER (Cust_id          char(10),
```

```
                    Cust_name      char(30),
                    Address    char(150)
                    PRIMARY KEY (Cust_id))
```

Nếu trong kiểu thực thể có một thuộc tính kết hợp được đại diện bằng các thuộc tính đơn, thì những thuộc tính đơn này sẽ có mặt trong quan hệ tạo thành.. Hình 2.4.4.2 minh hoạ cho trường hợp này, trong đó Address được biểu diễn như là một thuộc tính kết hợp của các thuộc tính đơn như Street, City, Province. Kiểu thực thể này được ánh xạ thành quan hệ tương ứng chỉ chứa các thuộc tính đơn.



CUSTOMER

Cust_id	Cust_name	Street	City	Province

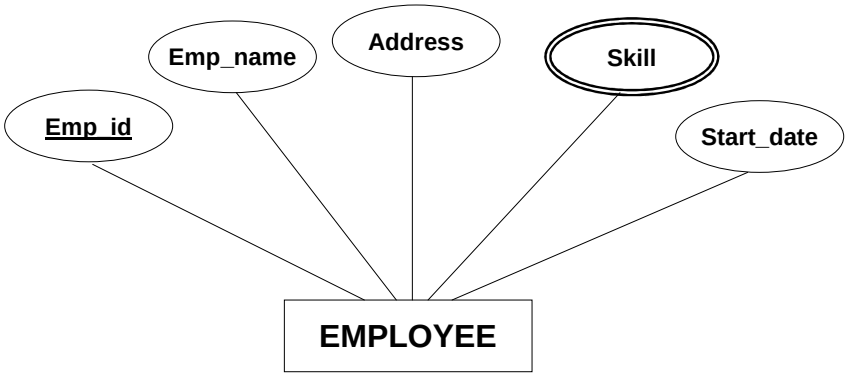
Hình 2.4.4.2 Chuyển đổi từ kiểu thực thể có thuộc tính kết hợp thành quan hệ

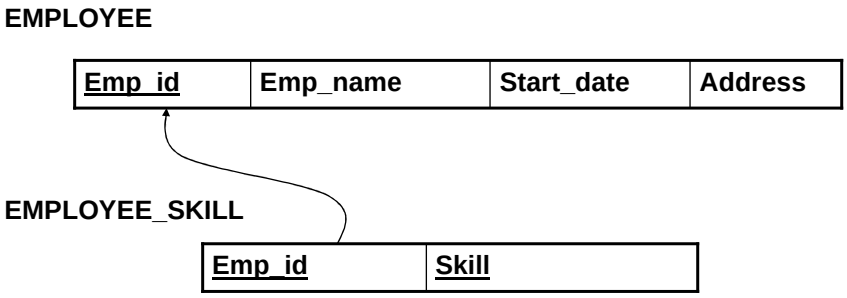
Câu lệnh tạo bảng như sau:

```
CREATE TABLE CUSTOMER (Cust_id          char(10),
                    Cust_name      char(30),
                    Street         char(50),
                    City       char(50),
                    Province   char(50)
                    PRIMARY KEY (Cust_id))
```

Nếu trong kiểu thực thể có chứa các thuộc tính đa trị, thì khi ánh xạ thành quan hệ, kiểu thực thể này sẽ tạo thành 2 quan hệ mới: quan hệ thứ nhất chứa tất cả các thuộc tính của kiểu thực thể ngoại trừ thuộc tính đa trị; và quan hệ thứ hai chỉ chứa hai thuộc tính, thuộc tính thứ nhất là khoá chính của quan hệ thứ nhất vừa tạo, và thuộc tính thứ hai là thuộc tính đa trị, cả hai thuộc tính này mới tạo thành khoá chính trong quan hệ thứ hai. Như vậy trong quan hệ thứ hai không tồn tại các thuộc tính mô tả mà chỉ gồm các thuộc tính tham gia vào khoá chính.

Hình 2.4.4.3 sau là một ví dụ minh hoạ cho kiểu thực thể EMPLOYEE của chúng ta. Như trong các ví dụ trước, thuộc tính Skill của kiểu thực thể này là một thuộc tính đa trị, tức là một nhân viên có thể có nhiều kỹ năng khác nhau, nhưng anh ta luôn chỉ có một mã duy nhất. Do vậy khi chuyển đổi kiểu thực thể này sang quan hệ chúng ta phải tạo thành hai quan hệ: Quan hệ thứ nhất vẫn được gọi là EMPLOYEE chứa khoá chính Emp_id cùng các thuộc tính đơn trị khác như Emp_name, và Address. Quan hệ thứ hai được đặt tên là EMPLOYEE_SKILL sẽ bao gồm khoá chính Emp_id và thuộc tính đa trị Skill, hai thuộc tính này kết hợp với nhau mới tạo thành khoá chính trong quan hệ mới. Liên kết giữa khoá chính và khoá ngoại lai của hai quan hệ được chỉ ra bởi đường mũi tên như trong hình.





Hình 2.4.4.3 Chuyển đổi từ kiểu thực thể có thuộc tính đa trị thành quan hệ

Quan hệ EMPLOYEE_SKILL không chứa các thuộc tính mô tả, tuy nhiên cũng có thể nhận thấy rằng, chúng ta có thể bổ sung thêm các thuộc tính đó vào cho quan hệ mới này như Certificate_Date để ghi nhận lại một nhân viên A đã thi đạt được kỹ năng B ngày nào, từ đó có thể suy ra anh ta đã có bao nhiêu năm kinh nghiệm với kỹ năng đó.

Câu lệnh tạo hai bảng này được mô tả như sau:

```
CREATE TABLE EMPLOYEE(Emp_id          char(10),
                      Emp_name        char(30) NOT NULL,
                      Start_date date NOT NULL,
                      Address         char(150)
                      PRIMARY KEY (Emp_id))
```

```
CREATE TABLE EMPLOYEE_SKILL (Emp_id   char(10),
                             Skill    Number(1) NOT NULL
                             PRIMARY KEY (Emp_id,Skill),
                             FOREIGN KEY (Emp_id)
REFERENCES EMPLOYEE)
```

4.5. Bước hai: Chuyển đổi các kiểu thực thể yếu

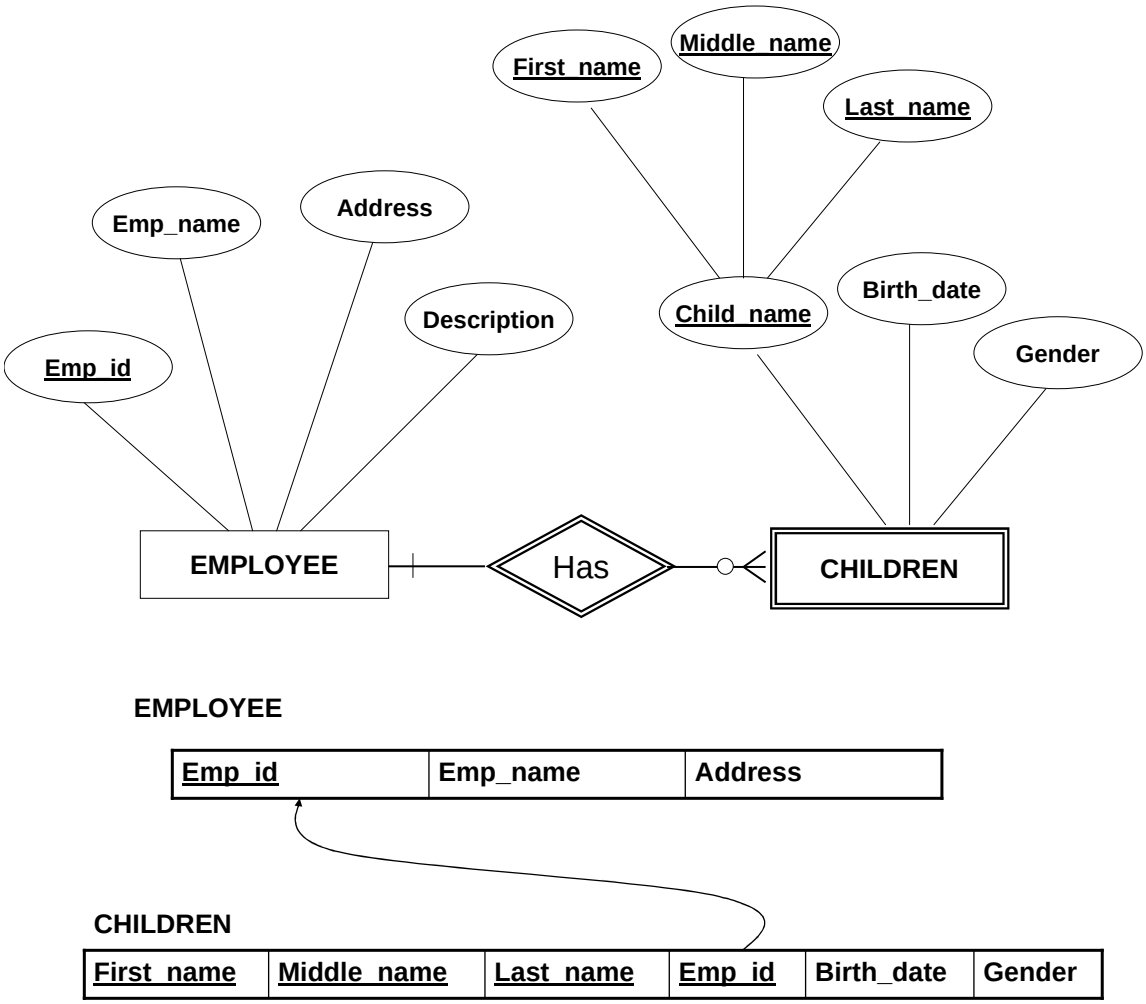
Như chúng ta đã biết, một kiểu thực thể yếu không được phép tồn tại độc lập mà luôn phụ thuộc vào một kiểu thực thể khác thông qua một liên kết định danh. Một kiểu thực thể yếu không có một định danh đầy đủ, nhưng nó cũng có một đinh danh mà chúng ta qui ước gọi là định danh bộ phận để có thể phân biệt được các thực thể trong kiểu thực thể yếu này khi biết một thực thể tương ứng trong kiểu thực thể sở hữu nó.

Khi chúng ta muốn chuyển đổi một kiểu thực thể yếu sang quan hệ, chúng ta phải chuyển đổi các kiểu thực thể sở hữu nó thành các quan hệ trước theo bước 1, sau đó mới có thể tiến hành chuyển đổi nó như sau:

Đối với mỗi kiểu thực thể yếu, chúng ta tạo một quan hệ tương ứng với nó bằng cách chuyển đổi tất cả các thuộc tính thành các thuộc tính trong quan hệ. Tất nhiên, nếu trong kiểu thực thể yếu tồn tại những thuộc tính kết hợp thì nó cũng sẽ được chuyển sang thành các thuộc tính trong quan hệ theo nguyên tắc như trong bước 1, tức là chỉ chuyển đổi các thuộc tính đơn hợp thành thuộc tính kết hợp đó. Ngoài ra, trong quan hệ mới thì khoá chính trong quan hệ sở hữu tương ứng của nó cũng sẽ phải có mặt trong quan hệ mới và trở thành các thuộc tính đóng vai trò khoá ngoại

lai. Cuối cùng, khoá chính của quan hệ mới sẽ là hợp của các khoá ngoại lai và các thuộc tính tham gia vào định danh bộ phận của kiểu thực thể yếu.

Hình 2.4.5.1 sau là một ví dụ minh hoạ cho bước thứ hai này, trong đó kiểu thực thể CHILDREN là một kiểu thực thể yếu, nó phụ thuộc vào kiểu thực thể sở hữu EMPLOYEE và kết nối thông qua liên kết “Has” (chúng ta có thể tham khảo thêm hình 2.3.1.2 trong phần trước). Ở đây cũng thấy rằng thuộc tính kết hợp Child_name bao gồm các thuộc tính đơn là First_name, Middle_name và Last_name, chúng sẽ có mặt trong quan hệ mới, đồng thời nó cũng đóng vai trò là định danh bộ phận trong kiểu thực thể yếu, nên khi chuyển sang quan hệ nó sẽ tham gia vào khoá chính.



Hình 2.4.5.1 Chuyển đổi từ kiểu thực thể yếu thành quan hệ

Như vậy khoá chính của quan hệ CHILDREN sẽ bao gồm 4 thuộc tính: First_name, Middle_name, Last_name và thuộc tính khoá ngoại lai Emp_ID từ quan hệ sở hữu EMPLOYEE tương ứng. Các thuộc tính ngày sinh Birth_date, giới tính Gender là các thuộc tính mô tả. Mũi tên trong hình minh hoạ cho liên kết giữa thuộc tính khoá ngoại lai với khoá chính.

Câu lệnh tạo hai bảng này được mô tả như sau:

```
CREATE TABLE EMPLOYEE(Emp_id          char(10),
```

```
                    Emp_name      char(30) NOT NULL,
                    Address    char(150)
                    PRIMARY KEY (Emp_id))
CREATE TABLE CHILDREN (First_name  char(10),
                Middle_name    char(10),
                Last_name      char(10),
                Emp_id         char(10),
                Birth_date date NOT NULL,
                Gender         char(1)
                PRIMARY KEY (First_name, Middle_name,
Last_name,Emp_id),
                FOREIGN KEY (Emp_id) REFERENCES
EMPLOYEE)
```

4.6. Bước ba: Chuyển đổi các liên kết giữa hai kiểu thực thể

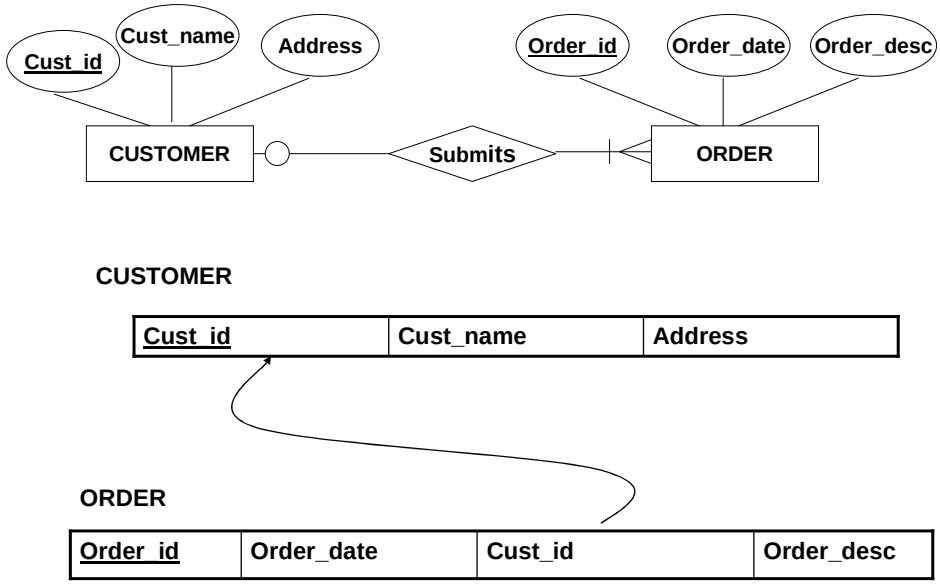
Việc chuyển đổi các liên kết này phụ thuộc vào kiểu liên kết giữa các kiểu thực thể: liên kết đơn trong cùng một kiểu thực thể, liên kết đôi giữa hai kiểu thực thể hay liên kết đa giữa nhiều kiểu thực thể; cấp độ của liên kết: liên kết một-một, một-nhiều hay liên kết nhiều-nhiều và các ràng buộc tham gia của liên kết. Chúng ta sẽ lần lượt xem xét việc chuyển đổi của từng loại một như sau:

a. Chuyển đổi các liên kết một-nhiều

Đối với loại liên kết này, đầu tiên chúng ta phải ánh xạ hai kiểu thực thể tham gia trong liên kết thành hai quan hệ

tương ứng như trong hai bước 1 và 2 trên. Tiếp theo đưa các thuộc tính khoá chính của quan hệ phía liên kết một sang thành các thuộc tính khoá ngoại lại trong quan hệ phía liên kết nhiều. Nếu đầu liên kết phía nhiều có dạng ràng buộc là bắt buộc thì các khoá ngoại lai sẽ tham gia vào làm khoá chính của quan hệ.

Ví dụ: hình 2.4.6.1 sau minh hoạ cho quá trình này, trong hình a là một liên kết một-nhiều. Hình b là kết quả của phép chuyển đổi, trong đó chúng ta thấy rằng khoá chính Cust_id trong quan hệ phía liên kết một CUSTOMER đã được chuyển sang thành thuộc tính khoá ngoại lai Cust_id trong quan hệ ORDER phía liên kết nhiều. Liên kết khoá ngoại lai giữa hai quan hệ được minh hoạ bằng đường mũi tên.



Hình 2.4.6.1 Chuyển đổi các liên kết một - nhiều

Câu lệnh tạo hai bảng này được mô tả như sau:

```
CREATE TABLE CUSTOMER(Cust_id          char(10),
```

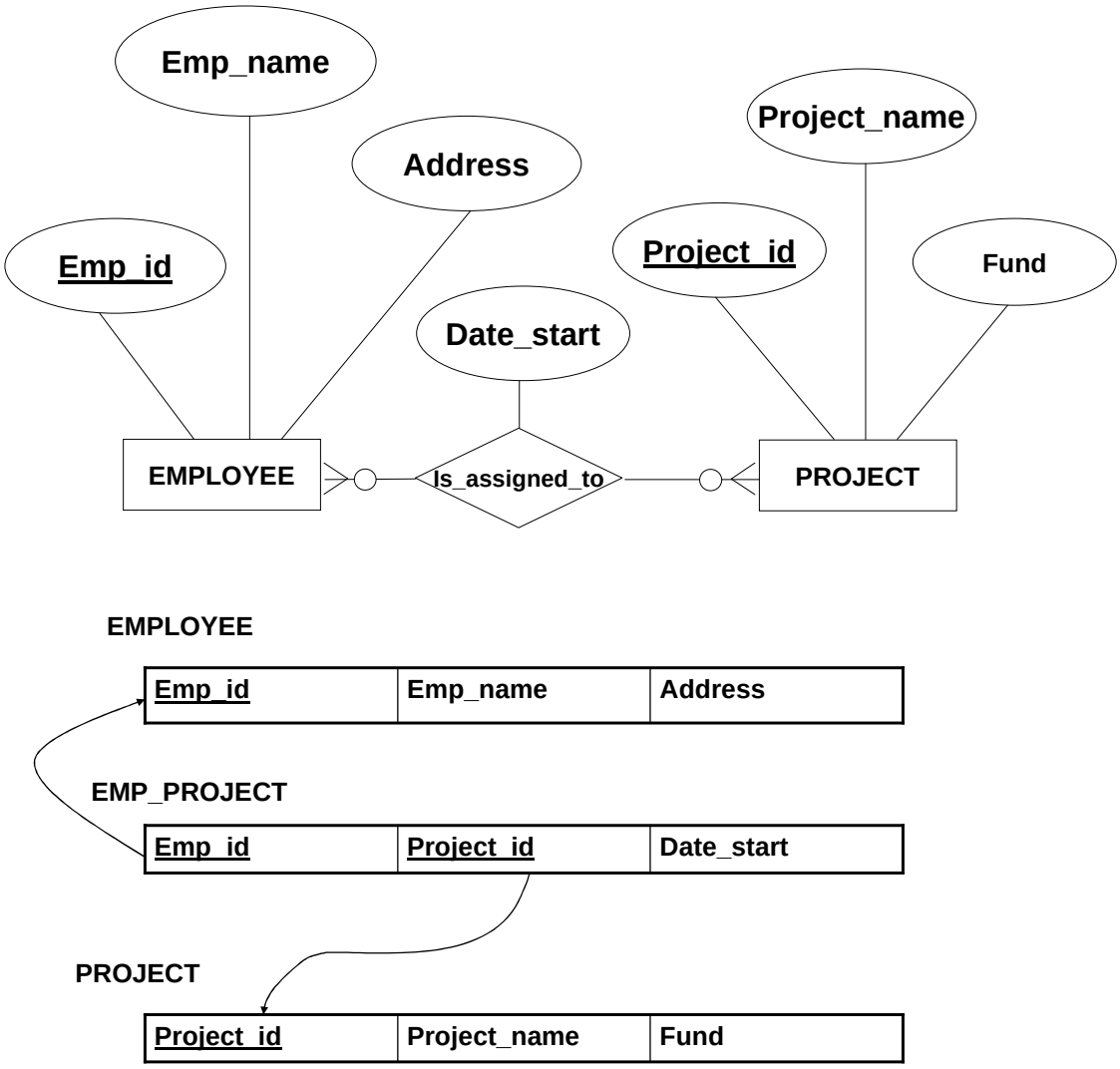
```
                    Cust_name      char(30) NOT NULL,
                    Address    char(150)
                    PRIMARY KEY (Cust_id))
CREATE TABLE ORDER(Order_id   char(10),
              Order_date      date  NOT NULL,
              Cust_id         char(10) NOT NULL,
              Order_desc      char(200)
              PRIMARY KEY (Order_id),
              FOREIGN  KEY  (Cust_id)  REFERENCES
CUSTOMER)
```

b. Chuyển đổi các liên kết nhiều-nhiều

Nếu giữa hai kiểu thực thể A và B có một liên kết kiểu nhiều-nhiều. Khi đó chúng ta phải tạo thêm một quan hệ mới C. Trong quan hệ mới C này, khoá chính của nó sẽ được tạo lập từ các khoá chính của hai quan hệ A và B. Như vậy chúng ta thấy rằng để có được quan hệ C, đầu tiên ta phải chuyển đổi được hai quan hệ A và B từ hai kiểu thực thể A và B tương ứng như trong các bước trước. Các thuộc tính mô tả cho liên kết nhiều-nhiều sẽ trở thành các thuộc tính tương ứng trong quan hệ C mới.

Ví dụ: Hình 2.4.6.2 sau là một ví dụ cho qui tắc chuyển đổi này. Liên kết “Is_assigned_to” kết nối giữa hai kiểu thực thể EMPLOYEE và PROJECT, đây là một liên kết loại nhiều-nhiều, bởi một nhân viên có thể tham gia vào nhiều dự án và một dự án có thể bao gồm nhiều nhân viên tham gia.. Thuộc tính ngày bắt đầu tham gia dự án Date_start là

thuộc tính mô tả cho liên kết này. Do đó khi chuyển đổi sang các quan hệ, hai kiểu thực thể EMPLOYEE và PROJECT được chuyển đổi đầu tiên, tiếp đó liên kết "Is_assigned_to" được chuyển đổi thành một quan hệ mới mà chúng ta đặt tên là EMP_PROJECT, trong quan hệ này chúng ta thấy rằng các thuộc tính khoá chính Emp_id và Project_id trong hai quan hệ EMPLOYEE và PROJECT tương ứng đã có mặt trong quan hệ mới và tham gia vào làm khoá chính trong quan hệ EMP_PROJECT này, còn thuộc tính Date_start trở thành thuộc tính mô tả trong quan hệ mới. Các liên kết của khoá ngoại lai được minh hoạ bằng các mũi tên giữa các quan hệ.



Hình 2.4.6.2 Chuyển đổi các liên kết nhiều - nhiều

Câu lệnh tạo các bảng này được mô tả như sau:

```
CREATE TABLE EMPLOYEE(Emp_id          char(10),
                      Emp_name     char(30) NOT NULL,
                      Address    char(150)
                      PRIMARY KEY (Emp_id))
CREATE TABLE PROJECT (Project_id          char(10),
                      Project_name   char(100) NOT NULL,
```

```
                    Fund        number(10)
                    PRIMARY KEY (Project_id))
CREATE TABLE EMP_PROJECT(Emp_id char(10),
                    Project_id  char(10),
                    Date_start date
                PRIMARY KEY (Emp_id, Project_id),
                FOREIGN  KEY  (Emp_id)  REFERENCES
EMPLOYEE
                FOREIGN KEY (Project_id) REFERENCES
            PROJECT)
```

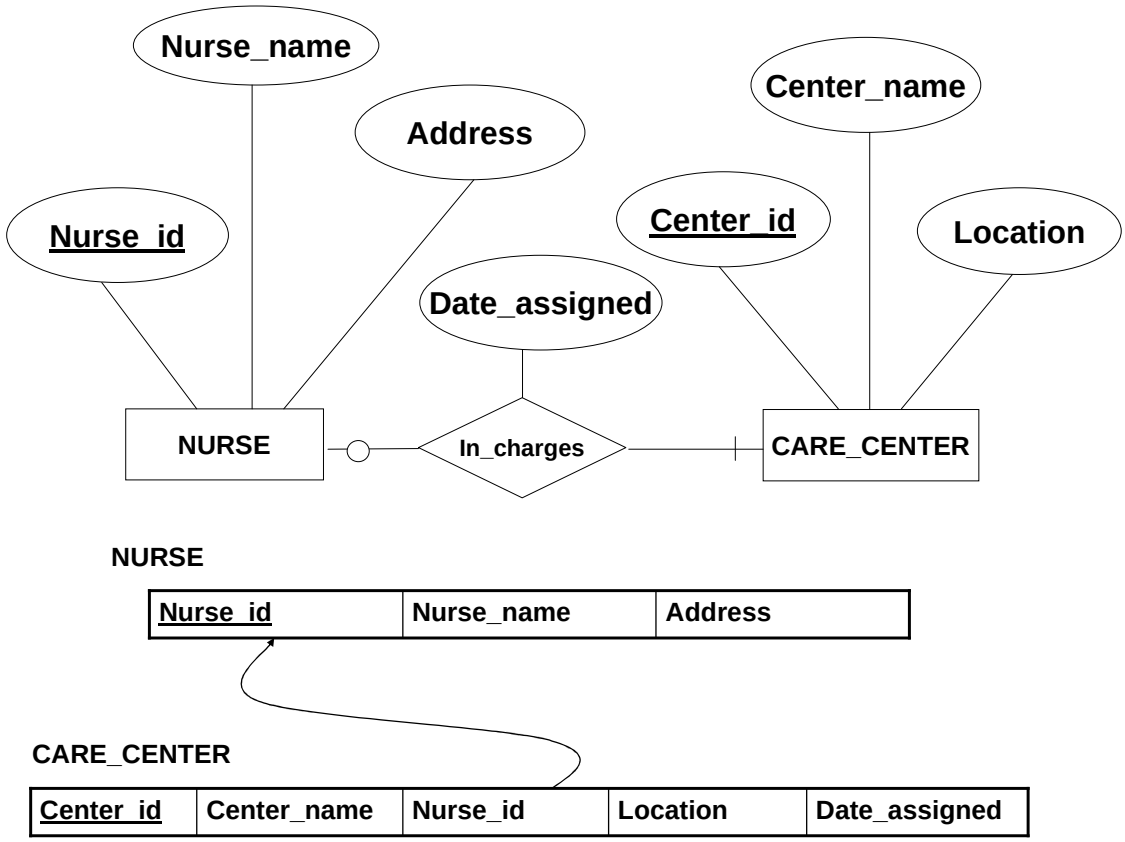
c. Chuyển đổi các liên kết một-một

Liên kết một-một có thể xem như là một trường hợp đặc biệt của liên kết một-nhiều. Quá trình chuyển đổi liên kết này qua hai giai đoạn: Đầu tiên hai quan hệ được chuyển đổi từ hai kiểu thực thể tương ứng tham gia vào liên kết. Tiếp theo khoá chính trong một quan hệ sẽ được ánh xạ thành khoá ngoại lai trong quan hệ còn lại.

Trong liên kết một-một, một đầu của liên kết luôn luôn là loại tuỳ chọn (optional), đầu còn lại luôn luôn là bắt buộc (mandatory). Do vậy khi chuyển đổi các kiểu thực thể thành các quan hệ thì kiểu thực thể ở phía liên kết bắt buộc sẽ trở thành quan hệ trong đó có khoá ngoại lai lấy từ khoá chính của quan hệ ánh xạ từ kiểu thực thể phía liên kết tuỳ chọn. Như vậy chúng ta thấy rằng, sẽ tránh được trường hợp phải lưu giá trị rỗng trong các thuộc tính khoá ngoại lai. Các thuộc tính mô tả còn lại cũng sẽ có mặt trong các quan

hệ tương ứng như các qui tắc trên. Ngoài ra nếu trong liên kết có các thuộc tính mô tả cho liên kết đó thì nó cũng sẽ trở thành các thuộc tính mô tả cho quan hệ phía liên kết tuỳ chọn.

Ví dụ minh hoạ: Hình 4.6.3 minh hoạ cho bước chuyển đổi này, giữa hai kiểu thực thể NURSE và CARE_CENTER tồn tại một liên kết một-một "In_charge". Mỗi một trung tâm chăm sóc phải có một y tá đảm nhiệm. Do đó kết nối từ CARE_CENTER đến NURSE phải là kết nối bắt buộc, tức là không thể có một trung tâm chăm sóc nào mà không có một y tá đảm nhận. Nhưng ngược lại, kết nối từ NURSE đến CARE_CENTER lại là kết nối tuỳ chọn, bởi có thể có những y tá không đảm nhận một trung tâm chăm sóc nào. Thuộc tính Date_assigned là thuộc tính mô tả cho liên kết giữa hai kiểu thực thể này nó ghi lại ngày mà y tá bắt đầu đảm nhận trung tâm chăm sóc.



Hình 2.4.6.3 Chuyển đổi các liên kết một - một

Kết quả của phép chuyển đổi liên kết một-một này cho ta thấy rằng hai quan hệ NURSE và CARE_CENTER đã được tạo ra từ hai kiểu thực thể tương ứng. Quan hệ NURSE được hình thành như trong bước 1, còn đối với quan hệ CARE_CENTER, do tham gia vào liên kết với tư cách là một kết nối bắt buộc, do vậy khóa ngoại lai từ quan hệ NURSE sẽ có mặt trong quan hệ này, ngoài ra thuộc tính Date_assigned mô tả cho liên kết cũng sẽ được bổ sung vào quan hệ.

Câu lệnh tạo bảng như sau:

```
CREATE TABLE NURSE (Nurse_id  char(10),
                    Nurse_name   char(100) NOT NULL,
```

```
                    Address    char(200)
                    PRIMARY KEY (Nurse_id))
CREATE TABLE CARE_CENTER(Center_id     char(10),
                    Center_name   char(100) NOT NULL,
                    Nurse_id   char(10) NOT NULL,
                    Location    char(200)
                    Date_assigned  date
                PRIMARY KEY (Center_id),
                FOREIGN KEY (Nurse_id) REFERENCES
NURSE)
```

4.7. Bước bốn: Chuyển đổi các kiểu thực thể kết nối

Như đã trình bày trong phần 3, khi tiến hành mô hình hoá dữ liệu với sơ đồ thực thể - liên kết, nếu chúng ta gặp phải một liên kết nhiều-nhiều, chúng ta có thể lựa chọn để mô hình hoá liên kết đó thành một kiểu thực thể kết nối. Cách tiếp cận này cũng giúp người sử dụng khi tham gia vào quá trình thiết kế dễ hình dung hơn. Việc chuyển đổi kiểu thực thể kết nối này cũng bao gồm các các khâu chuyển đổi tương tự như chuyển đổi liên kết nhiều-nhiều ở bước 3 nêu trên. Cụ thể các khâu chuyển đổi như sau:

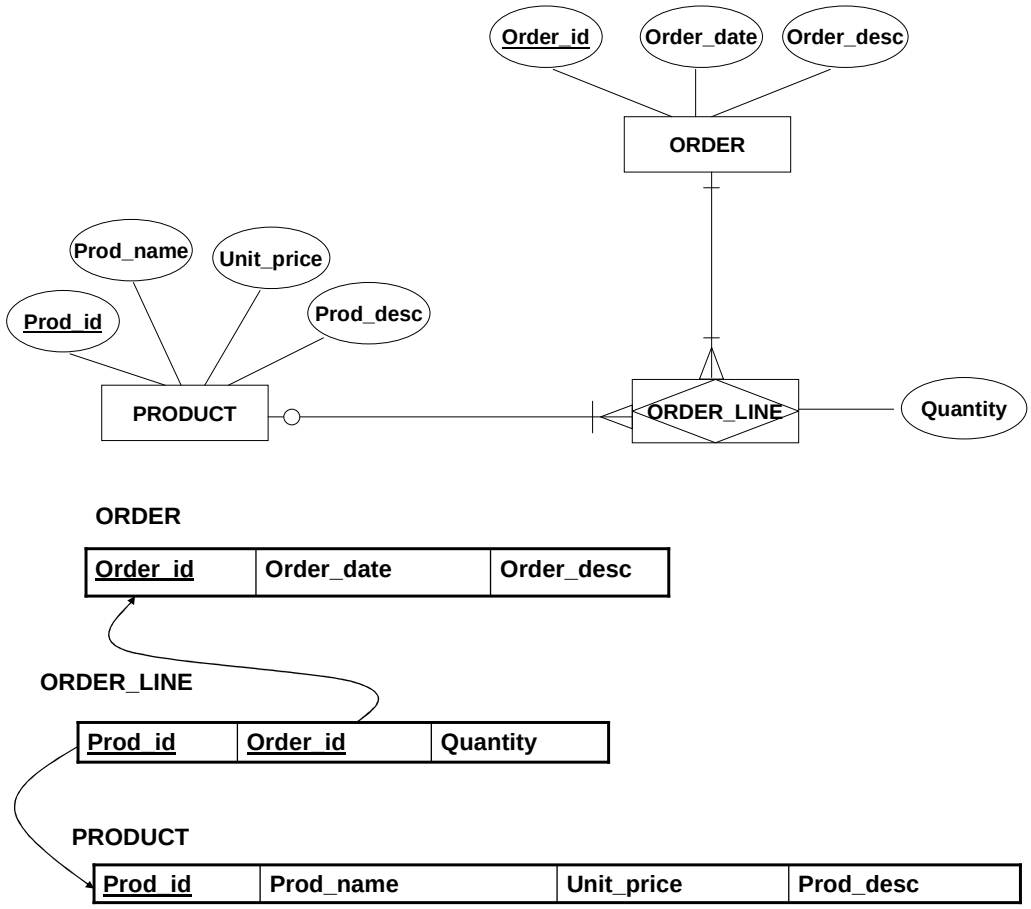
Đầu tiên chúng ta tạo ra 3 quan hệ: hai quan hệ đầu tương ứng với hai kiểu thực thể tham gia, quan hệ thứ ba được tạo nên từ kiểu thực thể kết nối, và chúng ta gọi nó là quan hệ kết nối. Ngoài ra, bước chuyển đổi còn phụ thuộc vào việc

trong kiểu thực thể kết nối có các thuộc tính định danh hay không. Để tiện theo dõi, chúng ta chia ra hai trường hợp:

a. Trường hợp thuộc tính định danh không xác định

Khi các thuộc tính định danh không được gán, khoá chính mặc định cho quan hệ kết nối sẽ bao gồm các thuộc tính tham gia vào làm khoá chính của hai quan hệ còn lại. Do đó mỗi khoá sẽ là một khoá ngoại lai trong quan hệ mới kết nối đến hai quan hệ còn lại.

Ví dụ, Hình 2.4.7.1 là một ví dụ minh hoạ cho bước chuyển đổi này. Kiểu thực thể kết nối ORDER_LINE liên kết hai kiểu thực thể ORDER và PRODUCT. Chúng ta có thể nhận thấy rằng nó tương tự như ví dụ trong trường hợp của liên kết nhiều-nhiều trong hình 2.4.6.2.



Hình 2.4.7.1 Chuyển đổi kiểu thực thể kết nối với thuộc tính định danh không xác định

Câu lệnh tạo bảng này được mô tả như sau:

```
CREATE TABLE ORDER(Order_id        char(10),
                   Order_date      date NOT NULL,
                   Order_desc      char(150)
                   PRIMARY KEY (Order_id))
CREATE TABLE PRODUCT(Prod_id       char(10),
                   Prod_name       char(30) NOT NULL,
                   Unit_price number(10) NOT NULL,
                   Prod_desc char(10)
                   PRIMARY KEY (Prod_id))
```

```
CREATE TABLE ORDER_LINE(Order_id  char(10),
                        Prod_id        char(10),
                        Quantity   number(10) NOT NULL
                        PRIMARY KEY (Order_id,Prod_id),
                    FOREIGN KEY (Order_id) REFERENCES
ORDER,
                    FOREIGN KEY (Prod_id) REFERENCES
ORDER)
```

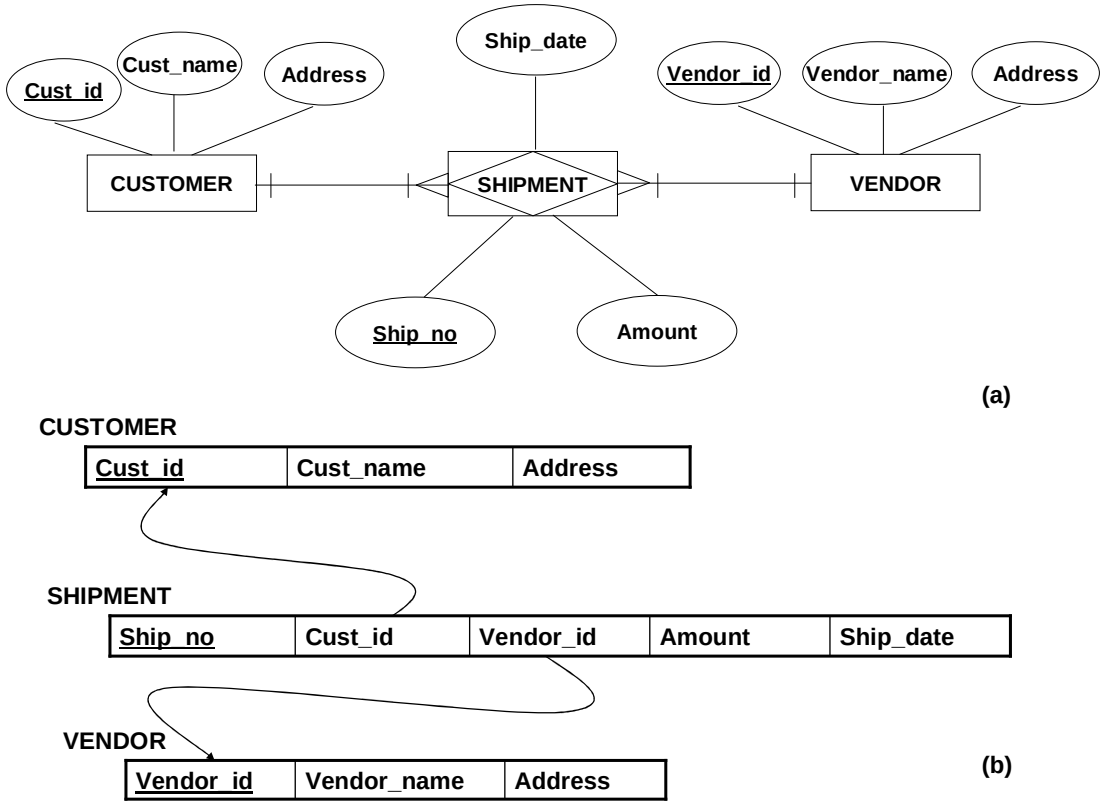
Hãy lưu ý đến thứ tự các câu lệnh tạo bảng, đầu tiên hai quan hệ ORDER và PRODUCT phải được tạo trước, tiếp theo mới đến quan hệ ORDER_LINE.

b. Trường hợp thuộc tính định danh đã xác định

Đôi khi trong quá trình mô hình hoá dữ liệu với sơ đồ thực thể liên kết, chúng ta đã xác định được định danh cho kiểu thực thể kết nối (chúng được gọi là khoá hay định danh đại diện). Có hai lý do khiến chúng ta có thể xác định được định danh này:

- Khi định danh dễ dàng xác định và phù hợp với bài toán thực tế. Ví dụ số hiệu đơn hàng luôn là duy nhất nên nó là định danh cho kiểu thực thể kết nối đơn hàng (nối hai kiểu thực thể là khách hàng và đại lý cung cấp).
- Khi định danh mặc định (xác định được bằng việc kết hợp hai định danh của hai kiểu thực thể được kết nối) không thể xác định duy nhất cho các thể hiện của kiểu thực thể kết nối.

Khi đó quá trình chuyển đổi kiểu thực thể kết nối sẽ tiến hành như sau: một quan hệ kết nối vẫn phải được tạo ra tương ứng với kiểu thực thể kết nối đó (như trường hợp nêu trên), tuy nhiên khoá chính của quan hệ kết nối mới bây giờ là khoá đại diện. Các khoá chính trong hai quan hệ còn lại vẫn có mặt trong quan hệ kết nối mới nhưng nó chỉ đóng vai trò là các khoá ngoại lai mà không tham gia vào làm khóa chính.



Hình 2.4.7.2 Chuyển đổi kiểu thực thể kết nối với thuộc tính định danh xác định

Hình 2.4.7.2 trên là ví dụ minh hoạ cho bước chuyển đổi trong trường hợp khoá định danh đã xác định. Trong hình a chúng ta thấy rằng kiểu thực thể kết nối SHIPMENT liên kết hai kiểu thực thể CUSTOMER và VENDOR. Trong

kiểu thực thể kết nối này chúng ta đã có thuộc tính Ship_no đóng vai trò định danh bởi hai lý do:

- Thứ nhất số hiệu hoá đơn vận chuyển Ship_no là có thực và người sử dụng đã quen với giá trị này, nó không thể mang các giá trị trùng nhau.
- Thứ hai, định danh mặc định ban đầu cho quan hệ kết nối mới sẽ là hai thuộc tính Cust_id và Vendor_id không thể xác định duy nhất cho một hoá đơn vận chuyển, bởi đối với mỗi một khách hàng và với mỗi một đại lý có thể có rất nhiều các hoá đơn vận chuyển, do vậy chúng ta không thể sử dụng kết hợp của hai thuộc tính định danh này thành khoá chính cho quan hệ kết nối mới.

Trên thực tế, với mỗi một nhà vận chuyển có rất nhiều hoá đơn vận chuyển đến của cùng một khách hàng, thậm chí nếu chúng ta cho rằng có thể sử dụng thêm thuộc tính ngày vận chuyển Ship_date vào cùng tham gia làm khoá chính thì cũng không thể thoả mãn, bởi trong cùng một ngày vẫn có thể phát sinh nhiều hoá đơn đến của cùng một khách hàng từ cùng một nhà đại lý. Tuy nhiên nếu sử dụng khoá đại diện Ship_no thì luôn xác định duy nhất một hoá đơn vận chuyển. Hai thuộc tính mô tả Ship_date và Amount vẫn sẽ xuất hiện trong quan hệ kết nối mới.

Kết quả của phép chuyển đổi này được thể hiện trong hình 2.4.7.2, quan hệ kết nối mới được đặt tên là SHIPMENT. Khoá chính của quan hệ này là Ship_no, hai thuộc tính Cust_id và Vendor_id đóng vai trò là khoá ngoại lai liên kết với hai quan hệ còn lại.

Câu lệnh mô tả:

```
CREATE TABLE CUSTOMER(Cust_id          char(10),
                      Cust_name     char(30) NOT NULL,
                      Address    char(150)
                      PRIMARY KEY (Cust_id))
CREATE TABLE VENDOR(Vendor_id          char(10),
                      Vendor_name   char(30) NOT NULL,
                      Address    char(150)
                      PRIMARY KEY (Vendor_id))
CREATE TABLE SHIPMENT(Ship_no          char(10),
                      Cust_id          char(10) NOT NULL,
                      Vendor_id char(10) NOT NULL,
                      Amount          number(10)      NOT
                      NULL,
                      Ship_date date  NOT NULL
                PRIMARY KEY (Ship_no),
                FOREIGN  KEY  (Cust_id)  REFERENCES
CUSTOMER,
                FOREIGN KEY (Vendor_id) REFERENCES
VENDOR)
```

4.8. Bước năm: Chuyển đổi các liên kết đơn

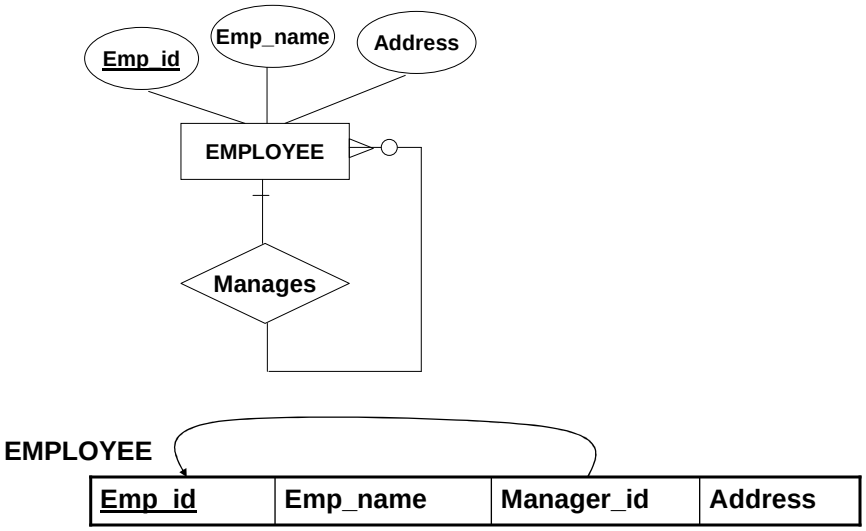
Như chúng ta đã biết liên kết đơn là liên kết giữa hai thực thể trong cùng một kiểu thực thể. Các liên kết đơn đôi khi còn được gọi là các liên kết đệ qui. Hai trường hợp quan trọng nhất của kiểu liên kết này đó là liên kết đơn một-nhiều và liên kết đơn nhiều-nhiều. Chúng ta sẽ xem xét hai

trường hợp này riêng rẽ bởi cách tiếp cận để chuyển đổi chúng là khá khác nhau:

a. Chuyển đổi các liên kết đơn một-nhiều

Trước hết, kiểu thực thể trong liên kết đơn vẫn được chuyển đổi thành một quan hệ như trong bước 1. Sau đó một thuộc tính đóng vai trò là khoá ngoại lai sẽ được đưa thêm vào quan hệ và sẽ liên kết với các khoá chính, do đó miền giá trị cho khoá ngoại lai này cũng sẽ là miền giá trị của khoá chính và trên thực tế, nó sẽ được lấy từ tập các giá trị trong khoá chính. Khoá ngoại lai này đôi khi còn được gọi là khoá ngoại lai đệ qui bởi nó tham chiếu đến chính khoá chính trong cùng quan hệ.

Ví dụ: Hình 2.4.8.1 dưới đây minh hoạ cho một liên kết đơn một-nhiều có tên là “Manages”, liên kết này cho biết một nhân viên có thể là người quản lý của một số nhân viên khác nhưng họ đều là nhân viên của công ty. Mỗi một nhân viên có duy nhất một người quản lý trực tiếp. Quan hệ EMPLOYEE được chuyển đổi tương ứng từ kiểu thực thể EMPLOYEE và liên kết đơn một-nhiều được cho trong hình b. Khoá ngoại lai đệ qui trong quan hệ mới được đặt tên là Manage_id. Miền xác định của khoá này cũng chính là miền xác định của khoá chính Emp_id. Mỗi một bản ghi của quan hệ này sẽ lưu dữ liệu về một nhân viên bao gồm: mã hiệu nhân viên Emp_id, tên nhân viên Emp_name, ngày sinh Birth_date, và mã hiệu người quản lý trực tiếp của nhân viên đó Manager_id. Lưu ý rằng vì là khoá ngoại lai nên Manager_id sẽ tham chiếu trực tiếp đến Emp_id, và chúng ta thấy rằng khoá chính của quan hệ mới vẫn là Emp_id.



Hình 2.4.8.1 Chuyển đổi các liên kết đơn một - nhiều

```
CREATE TABLE EMPLOYEE(Emp_id          char(10),
                      Emp_name        char(30) NOT NULL,
                      Manager_id      char(10),
                      Address         char(150)
                      PRIMARY KEY (Emp_id),
                      FOREIGN KEY (Manager_id)
REFERENCES EMPLOYEE)
```

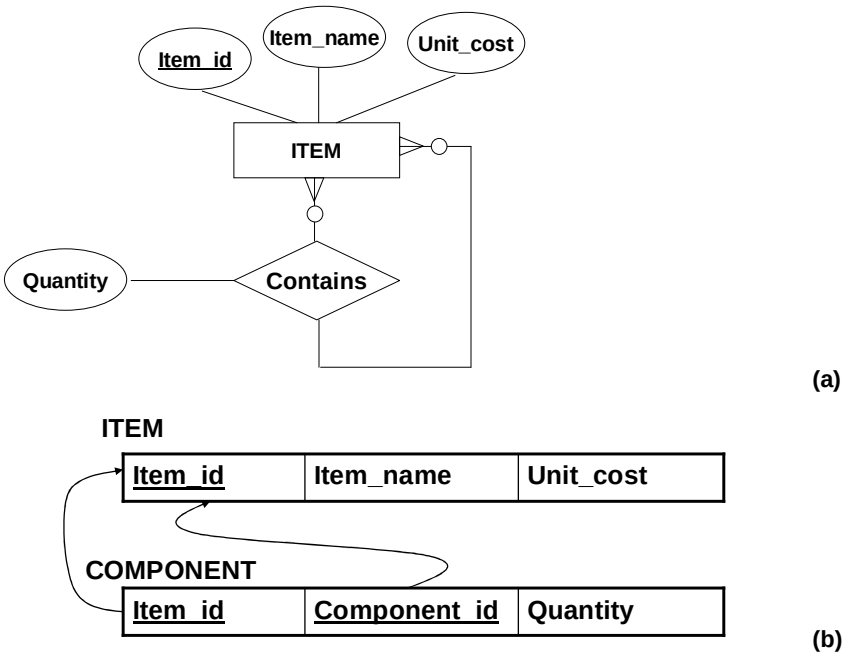
Trong thực tế chúng ta gặp liên kết đơn kiểu một-nhiều này rất nhiều. Ví dụ, để mô tả một kiểu thực thể địa bàn hành chính, chúng ta phải sử dụng liên kết này, bởi mỗi một xã sẽ thuộc một huyện nhất định, tiếp đến mỗi một huyện sẽ

phải thuộc một tỉnh thành nhất định và cuối cùng mỗi một tỉnh thành lại thuộc một vùng nhất định.

b. Chuyển đổi các liên kết đơn nhiều-nhiều

Với kiểu liên kết này, sẽ có đến hai quan hệ sẽ được tạo ra: một quan hệ được tạo ra tương ứng với chính kiểu thực thể hiện thời, và một quan hệ khác được sinh ra từ mối liên kết nhiều-nhiều và được gọi là quan hệ kết nối. Khoá chính của quan hệ kết nối sẽ bao gồm hai thuộc tính, giá trị trong cả hai thuộc tính này đều lấy chung từ các giá trị trong thuộc tính khoá chính của quan hệ còn lại. Mọi thuộc tính kết nối của liên kết đều được đặt trong quan hệ kết nối mới.

Ví dụ minh hoạ: Một liên kết đơn kiểu nhiều-nhiều được thể hiện trong hình 2.4.8.2. Mỗi một bộ phận có thể được bao gồm từ các bộ phận khác, và ngược lại một bộ phận có thể tham gia như một thành phần trong nhiều bộ phận khác, do đó liên kết đơn “Contains” này là một liên kết nhiều-nhiều. Các quan hệ ánh xạ từ kiểu thực thể cùng với liên kết đơn này được chỉ ra trong phần b của hình 2.4.8.2. Quan hệ ITEM được ánh xạ trực tiếp từ kiểu thực thể ITEM. Còn quan hệ COMPONENT là một quan hệ kết nối với khoá chính bao gồm hai thuộc tính là Item_id và Component_id. Thuộc tính mô tả liên kết số lượng Quantity cũng sẽ xuất hiện trong quan hệ kết nối này, nó cho biết số lượng của các thành phần Component_id được sử dụng trong một bộ phận Item_no. Lưu ý rằng cả hai thuộc tính Item_no và Component_id đều tham chiếu đến khoá chính Item_no của quan hệ ITEM.



Hình 2.4.8.2 Chuyển đổi các liên kết đơn nhiều - nhiều

```
CREATE TABLE ITEM(Item_id        char(10),
                  Item_name      char(30) NOT NULL,
                  Unit_cost  number(10)
                  PRIMARY KEY (Item_id))
CREATE TABLE COMPONENT(Item_id    char(10),
                       Component_id  char(10),
                       Quantity    number(5)
                  PRIMARY KEY (Item_id, Component_id),
                  FOREIGN        KEY        (Component_id)
REFERENCES ITEM)
```

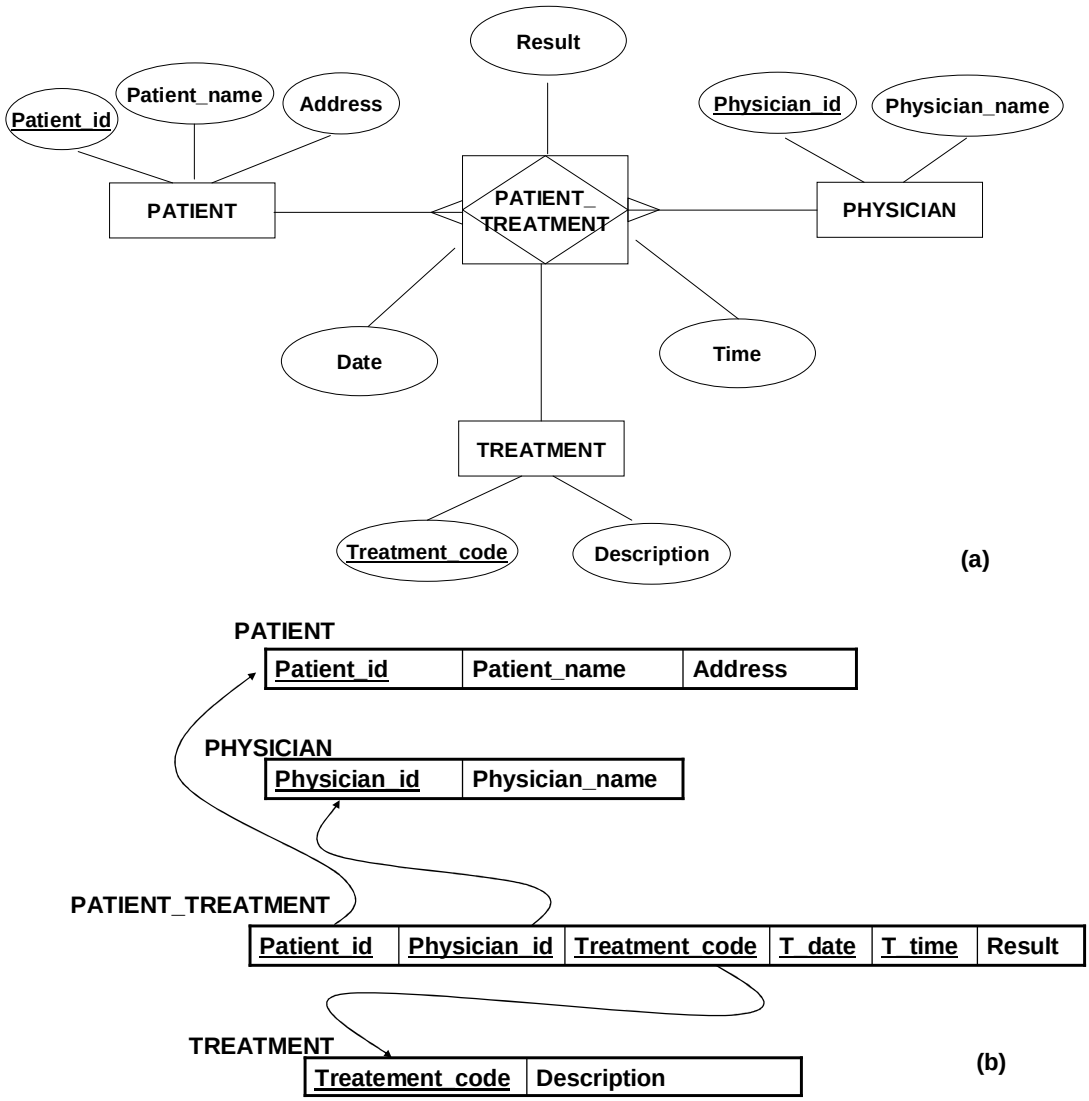
4.9. Bước sáu: Chuyển đổi các liên kết giữa nhiều kiểu thực thể

Trong phần thiết kế cơ sở dữ liệu mức khái niệm, chúng ta đã đề cập đến các liên kết giữa nhiều kiểu thực thể (liên kết đa nguyên), đó là loại liên kết kết nối ít nhất giữa 3 kiểu thực thể. Ở đây, chúng ta nên chuyển đổi loại liên kết này thành kiểu thực thể kết nối để có thể biểu diễn các ràng buộc bộ phận một cách chính xác hơn.

Giả thiết có 3 kiểu thực thể tham gia vào liên kết với kiểu thực thể kết nối. Khi đó, trước tiên chúng ta phải tiến hành chuyển đổi 3 kiểu thực thể đó thành các quan hệ tương ứng như trong các bước trên. Tiếp theo, khi chuyển đổi kiểu thực thể kết nối thành quan hệ, khoá chính mặc định cho nó sẽ là kết hợp của tất cả các thuộc tính tham gia vào làm khoá chính trong 3 quan hệ đã tạo trên. Tuy nhiên, có thể trong một số trường hợp chúng ta vẫn cần phải bổ sung thêm thuộc tính để hình thành lên một khoá chính duy nhất. Những thuộc tính này cũng đóng vai trò là khoá ngoại lai tham chiếu tương ứng đến các thuộc tính khoá chính trong ba quan hệ còn lại. Mỗi một thuộc tính trong kiểu thực thể kết nối sẽ trở thành một thuộc tính mô tả trong quan hệ kết nối mới. Để dễ hiểu hơn, chúng ta hãy xem ví dụ sau:

Ví dụ minh hoạ: Một kiểu thực thể kết nối liên kết với ba kiểu thực thể được thể hiện trong hình 2.4.9.1. Kiểu thực thể này mô tả cho mối quan hệ giữa một bệnh nhân nhận được một phương pháp điều trị từ một bác sỹ, chúng ta đặt tên cho kiểu thực thể này là PATIENT_TREATMENT, gồm các thuộc tính: ngày điều trị T_date, giờ điều trị T_time, và kết quả điều trị Result.

Kết quả của phép chuyển đổi các kiểu thực thể này được thể hiện trong phần b của hình 2.4.9.1. Các thuộc tính khoá chính Patient_id, Physician_id và Treatment_code trở thành các khoá ngoại lai trong quan hệ PATIENT_TREATMENT. Ở đây chúng ta nhận thấy rằng, chúng không xác định duy nhất một thể hiện trong quan hệ kết nối mới, bởi một bệnh nhân có thể nhận được cùng một phương pháp điều trị từ cùng một bác sỹ nhưng ở các thời điểm khác nhau. Do vậy trong quan hệ kết nối PATIENT_TREATMENT mới, chúng ta phải kết hợp thêm các thuộc tính thời gian vào để có thể xác định duy nhất một thể hiện trong quan hệ. Kết quả của phép ánh xạ này cũng được thể hiện trong hình 2.4.9.1.



Hình 2.4.9.1 Chuyển đổi các liên kết giữa nhiều kiểu thực thể

```
CREATE TABLE PATIENT(Patient_id      char(10),
                     Patient_name    char(30) NOT NULL,
                     Address         char(150)
                     PRIMARY KEY (Patient_id))
CREATE TABLE PHYSICIAN(Physiciant_id      char(10),
                       Physiciant_name    char(30) NOT NULL
```

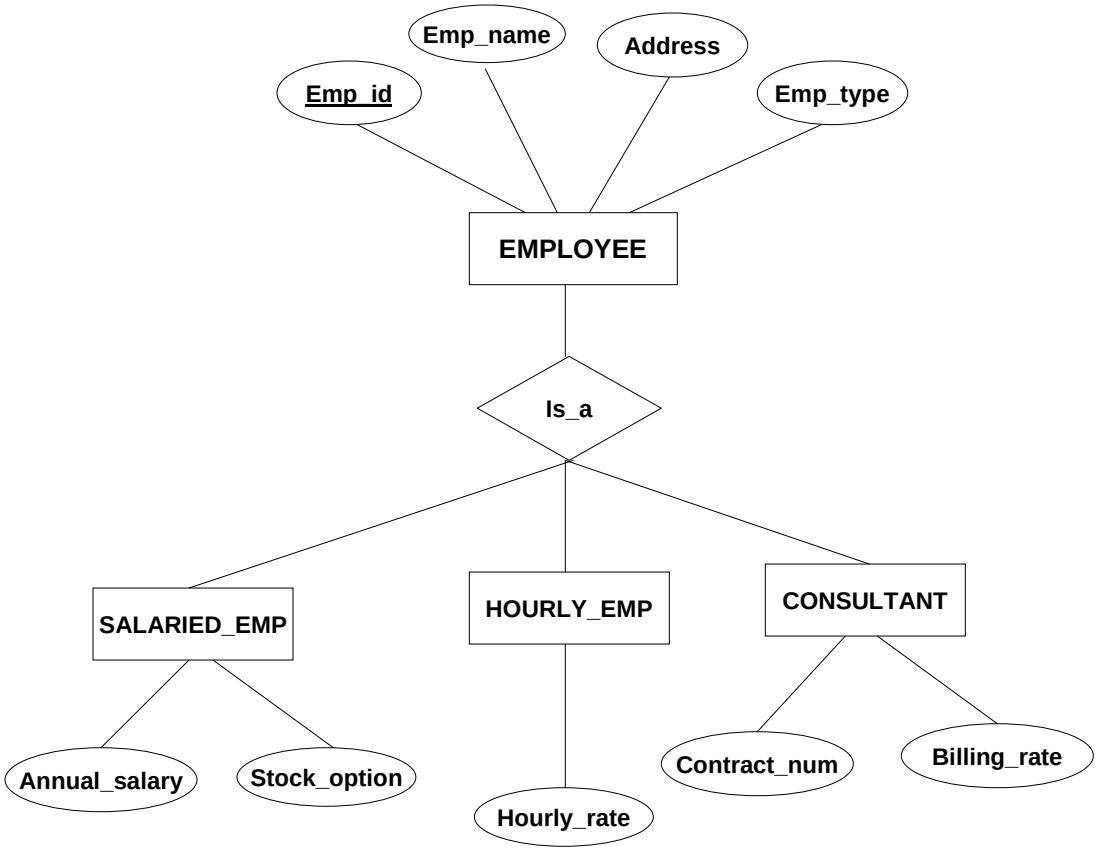
```
                    PRIMARY KEY (Physician_id))
CREATE TABLE TREATMENT(Patient_id char(10),
                    Physiciant_id        char(10),
                    Treatment_code       char(10),
                    T_date          date,
                    T_time          time,
                    Result               char(300) NOT NULL
PRIMARY KEY (Patient_id, Physician_id,Treatment_code,
T_date, T_time),
FOREIGN KEY (Patient_id) REFERENCES PATIENT),
FOREIGN KEY (Physician_id) REFERENCES
PHYSICIAN),
FOREIGN KEY (Treatment_code) REFERENCES
TREATMENT))
```

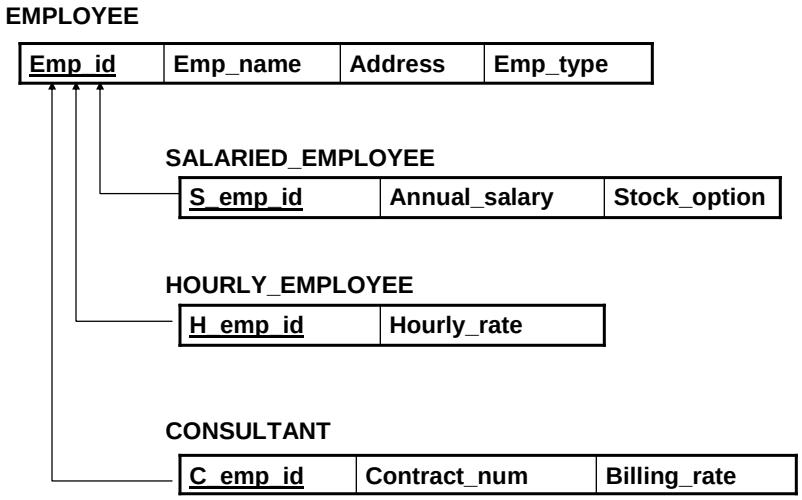
4.10. Bước bảy: Chuyển đổi các phân cấp "là một"

Trong mô hình dữ liệu quan hệ không hỗ trợ trực tiếp các phân cấp "là-một" hay liên kết "là-một". Tuy nhiên, chúng ta có thể sử dụng một số chiến lược để có thể biểu diễn được kiểu liên kết này. Một trong số những chiến lược hay được sử dụng được tiến hành như sau:

- Tạo mỗi quan hệ tách biệt cho kiểu thực thể gốc (supertype) và cho mỗi kiểu thực thể con của nó.
- Với kiểu thực thể gốc, quan hệ được tạo ra như cách áp dụng cách tạo quan hệ ở các bước trên.

- Với các kiểu thực thể con, các thuộc tính khoá chính trong mỗi quan hệ con được lấy từ tập thuộc tính khoá chính trong quan hệ gốc.
- Các thuộc tính còn lại trong mỗi kiểu thực thể con sẽ trở thành các thuộc tính mô tả tương ứng trong các quan hệ con.





Hình 2.4.10.1 Chuyển đổi các liên kết "là một"

Ví dụ minh hoạ: Hình 2.4.10.1 biểu diễn một kiểu thực thể supertype EMPLOYEE cùng với các kiểu thực thể subtype của nó là SALARIED_EMPLOYEE, HOURLY_EMPLOYEE và CONSULTANT. Khoá chính

của kiểu thực thể EMPLOYEE là Emp_id và thuộc tính Emp_type được gọi là thuộc tính phân biệt cho các kiểu thực thể con.

Kết quả của phép chuyển đổi các kiểu thực thể này cũng được thể hiện trong hình 2.4.10.1. Một quan hệ được tạo ra tương ứng với kiểu thực thể EMPLOYEE. Đối với mỗi kiểu thực thể subtype còn lại cũng sẽ hình thành các quan hệ tương ứng. Khoá chính của tất cả các quan hệ này đều là Emp_id. Tiền tố phía trước trong mỗi thuộc tính khoá chính trong mỗi quan hệ con chỉ mang tính chất phân biệt. Mỗi một thuộc tính khoá chính này sẽ đóng cả vai trò là khoá ngoại lai tham chiếu đến khoá chính trong quan hệ EMPLOYEE, chúng được thể hiện bằng các đường mũi tên trong hình. Các thuộc tính mô tả còn lại trong kiểu thực thể con trở thành các thuộc tính mô tả trong quan hệ tương ứng.

```
CREATE TABLE EMPLOYEE(Emp_id      char(10),
                      Emp_name    char(30) NOT NULL,
                      Emp_type    char(30) NOT NULL,
                      Address     char(150)
                      PRIMARY KEY (Emp_id))
CREATE TABLE SALARIED_EMPLOYEE(S_emp_id
    char(10),
                      Annual_salary   number(7) NOT NULL,
                      Stock_option    char(100)
                      PRIMARY KEY (S_emp_id),
                      FOREIGN KEY (S_emp_id) REFERENCES
                 EMPLOYEE))
```

```
CREATE TABLE HOURLY_EMPLOYEE(H_emp_id char(10),
        Hourly_rate number(5) NOT NULL
        PRIMARY KEY (H_emp_id),
        FOREIGN KEY (H_emp_id) REFERENCES EMPLOYEE))
CREATE TABLE CONSULTANT(C_emp_id char(10),
        Contract_num char(10) NOT NULL,
        Billing_rate number(8) NOT NULL
        PRIMARY KEY (C_emp_id),
        FOREIGN KEY (C_emp_id) REFERENCES EMPLOYEE))
```

4.11. Một sơ đồ cơ sở dữ liệu mức logic

Trong phần này chúng ta tiếp tục xem xét việc chuyển đổi sơ đồ thực thể - liên kết trong phần 3 thành sơ đồ dữ liệu mức logic:

Như chúng ta thấy, các kiểu thực thể CUSTOMER, SUPPLIER, ITEM, PRODUCT_CATEGORY là những kiểu thực thể thông thường nên sẽ được chuyển đổi thành những quan hệ tương ứng theo nguyên tắc đã trình bày: Mỗi một thuộc tính của kiểu thực thể sẽ trở thành một trường trong quan hệ tương ứng. Các định danh của kiểu thực thể sẽ trở thành khoá chính trong quan hệ như trong hình 2.4.11.1 sau:

CUSTOMER

Cust_id	Cust_name	Address

PRODUCT_CATEGORY

Prod_cat_id	Prod_cat_name	Description

ITEM

Item_id	Item_name	Item_desc

SUPPLIER

Sup_id	Sup_name	Telephone	Address

Hình 2.4.11.1 Các quan hệ chuyển đổi từ các kiểu thực thể thông thường

Tiếp theo chúng ta hãy xét đến kiểu thực thể ORDER. Do kiểu thực thể này có liên kết một - nhiều với kiểu thực thể CUSTOMER nên khoá chính trong quan hệ CUSTOMER sẽ được chuyển thành khoá ngoại lai trong quan hệ ORDER. Thuộc tính Order_id vẫn sẽ trở thành khoá chính trong quan hệ ORDER. Do vậy trong quan hệ ORDER chúng ta sẽ có thêm một thuộc tính khoá ngoại lai Cust_id tham chiếu đến thuộc tính Cust_id của quan hệ CUSTOMER. Kết quả của bước chuyển đổi này được thể hiện trong hình 2.4.11.2:

CUSTOMER

Cust_id	Cust_name	Address

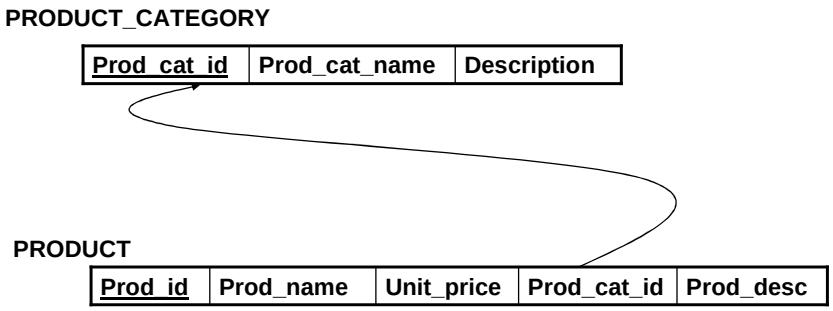
ORDER

Order_id	Cust_id	Order_date	Order_desc

Hình 2.4.11.2 Các quan hệ chuyển đổi từ các kiểu thực thể CUSTOMER và ORDER

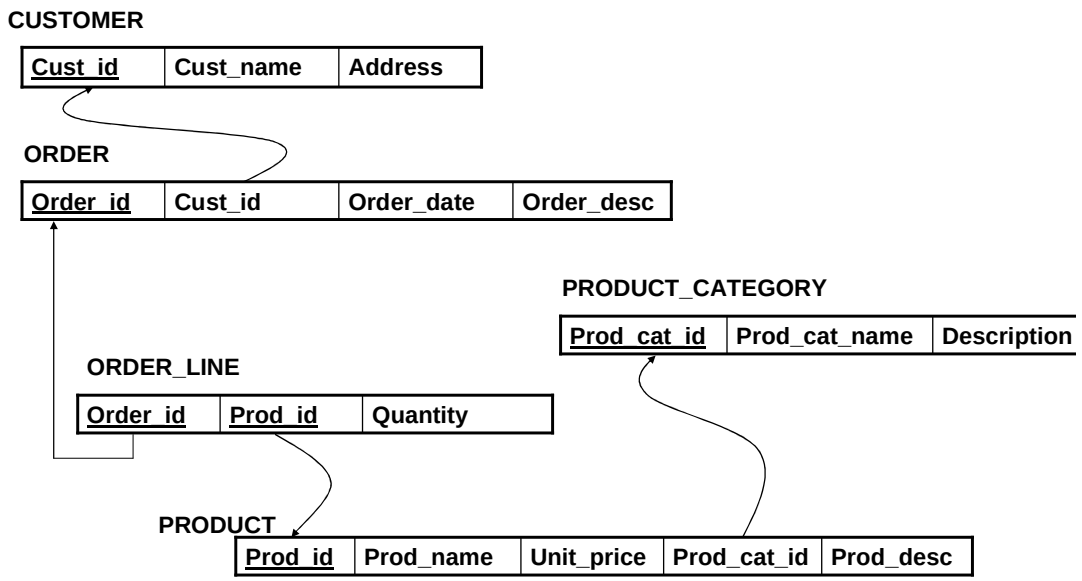
Với kiểu thực thể PRODUCT, do có liên kết một nhiều với kiểu thực thể PRODUCT_CATEGORY, nên khi chuyển

đổi thành quan hệ cho kiểu thực thể này, cũng như trường hợp trên, thuộc tính Prod_cat_id sẽ xuất hiện thêm và trở thành khoá ngoại lai tham chiếu đến khoá chính Prod_cat_id trong quan hệ PRODUCT_CATEGORY như hình 2.4.11.3 sau:



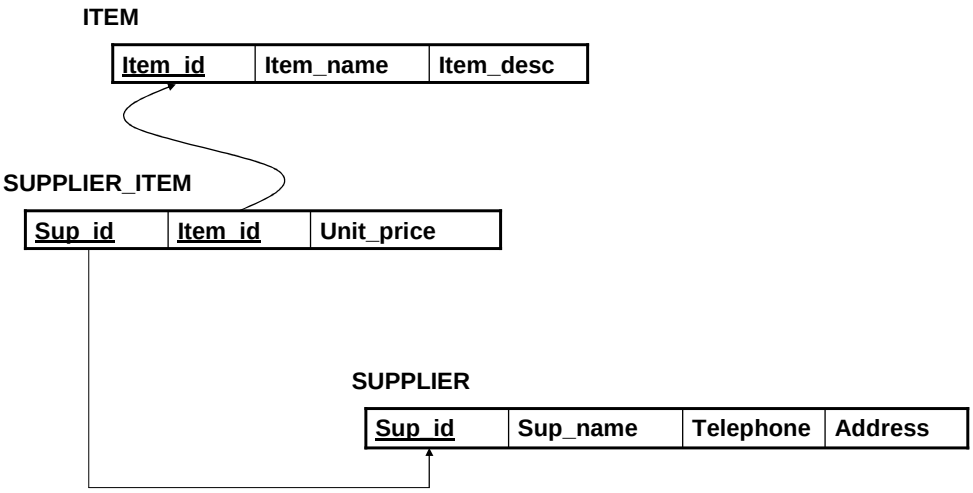
Hình 2.4.11.3 Các quan hệ chuyển đổi từ kiểu thực thể PRODUCT_CATEGORY và PRODUCT

Với kiểu thực thể liên kết ORDER_LINE, do có kết nối hai kiểu thực thể là ORDER và PRODUCT theo liên kết một-nhiều nên khi hình thành quan hệ, khoá chính của kiểu thực thể này sẽ được hình thành từ các khoá chính của hai quan hệ ORDER và PRODUCT, Order_id tham chiếu đến khoá chính của quan hệ ORDER và Prod_id tham chiếu đến khoá chính của quan hệ PRODUCT. Thuộc tính mô tả cho liên kết Quantity cũng sẽ trở thành một thuộc tính trong quan hệ mới. Hình 2.4.11.4 sau minh hoạ cho bước chuyển đổi này:



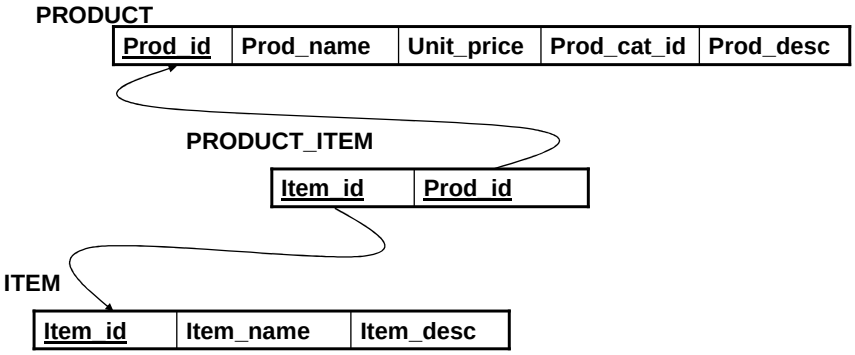
Hình 2.4.11.4 Chuyển đổi thành quan hệ cho kiểu thực thể ORDER_LINE

Liên kết nhiều-nhiều "Supplies" kết nối hai kiểu thực thể SUPPLIER và ITEM nên cũng sẽ tạo thành một quan hệ, chúng ta đặt tên là SUPPLIER_ITEM. Khoá cho quan hệ này sẽ là kết hợp khoá của hai quan hệ trong liên kết, bao gồm hai trường Sup_id tham chiếu đến khoá chính của quan hệ SUPPLIER và trường Item_id tham chiếu đến khoá chính của quan hệ ITEM, ngoài ra thuộc tính Unit_price cũng sẽ trở thành một trường quan hệ như trong hình 2.4.11.5:



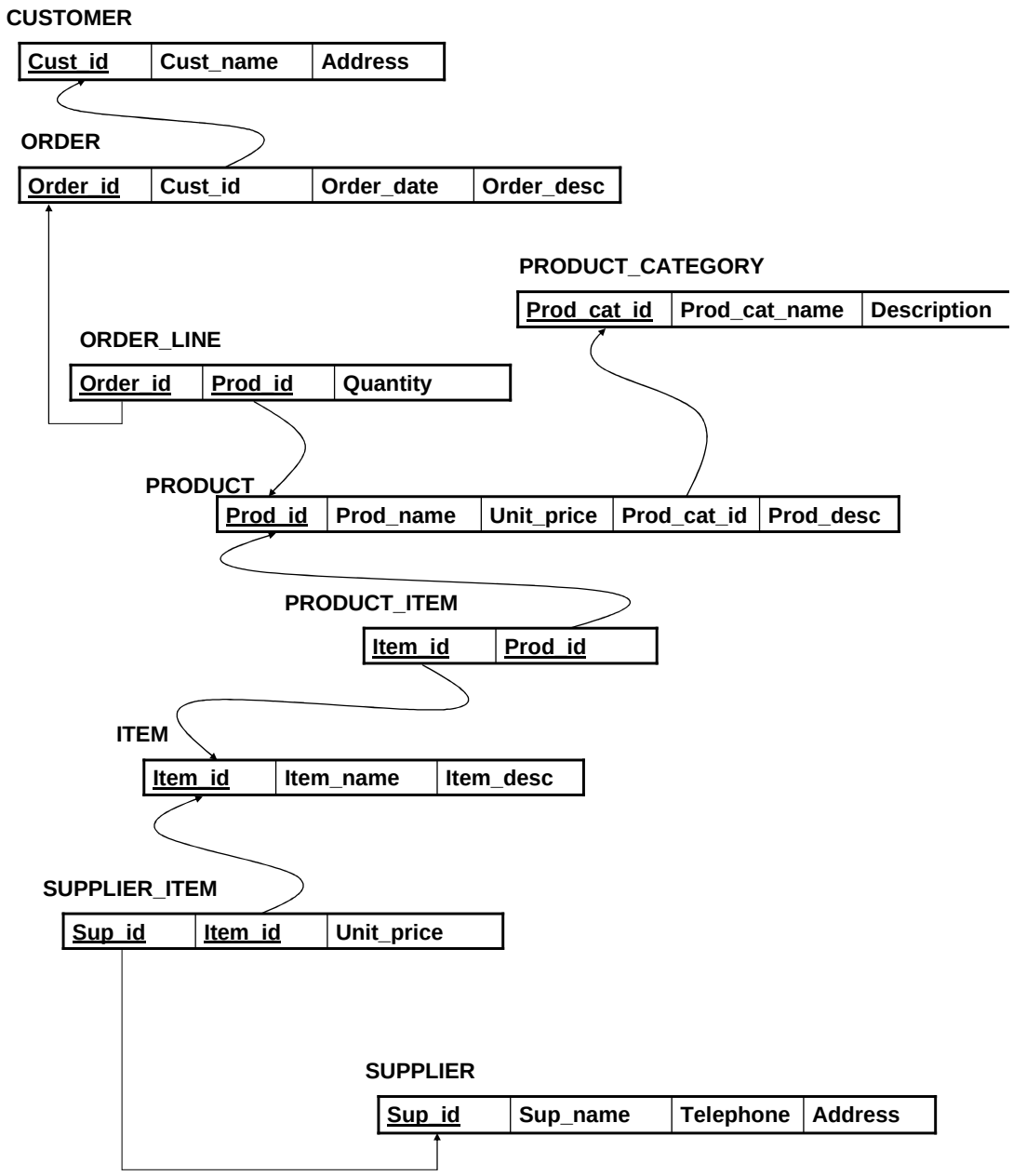
Hình 2.4.11.5 Chuyển đổi liên kết nhiều-nhiều "Supplies" thành quan hệ

Cuối cùng, liên kết nhiều-nhiều "Uses" giữa hai kiểu thực thể PRODUCT và ITEM cũng sẽ được tạo thành một quan hệ như trong trường hợp trên, chúng ta đặt tên cho quan hệ này là PRODUCT_ITEM. Tuy nhiên ở quan hệ này, mọi trường đều tham gia vào làm khoá chính, Item_id tham chiếu đến khoá chính của quan hệ ITEM, Prod_id tham chiếu đến khoá chính của quan hệ PRODUCT.



Hình 2.4.11.6 Chuyển đổi liên kết nhiều nhiều "Uses" thành quan hệ

Cuối cùng chúng ta có sơ đồ cơ sở dữ liệu mức logic như hình 2.4.11.7 sau:



Hình 2.4.11.7 Sơ đồ cơ sở dữ liệu mức logic

4.12. Kết luận

Bước thiết kế cơ sở dữ liệu mức logic là quá trình chuyển đổi từ mô hình dữ liệu mức khái niệm thành mô hình dữ liệu mức logic được mô tả dưới dạng một sơ đồ. Mô hình

dữ liệu quan hệ biểu diễn dữ liệu dưới dạng các quan hệ hay còn được gọi là các bảng. Một quan hệ là một bảng dữ liệu hai chiều và được đặt tên sao cho không trùng với các quan hệ khác trong cùng cơ sở dữ liệu. Tính chất khoá của các quan hệ đóng vai trò vô cùng quan trọng và nó cho biết không thể có hai bản ghi trong cùng một quan hệ có giá trị giống nhau ở tất cả các thuộc tính khoá . Và đặc biệt, trong quan hệ không thể chứa các thuộc tính đa trị.

Trong phần này, chúng ta đã xem xét lần lượt các trường hợp chuyển đổi từ sơ đồ thực thể - liên kết (được thực hiện ở bước 3) thành sơ đồ cơ sở dữ liệu mức logic.

Mỗi một kiểu thực thể trong sơ đồ thực thể liên kết được chuyển đổi thành một quan hệ. Một liên kết một-nhiều được biểu diễn với việc bổ sung thêm một khoá ngoại lai, chính là khoá chính trong quan hệ phía liên kết một, vào quan hệ chuyển đổi từ kiểu thực thể phía liên kết nhiều. Với liên kết nhiều-nhiều, chúng ta đã phải chuyển đổi nó thành một quan hệ mới riêng rẽ với hai quan hệ còn lại tham gia vào liên kết. Khoá chính của quan hệ mới này được tạo thành từ việc kết hợp hai khoá chính trong hai quan hệ mà nó kết nối. Chúng ta cũng đã xem xét khi nào thì khoá ngoại lai sẽ tham gia vào khoá chính trong quan hệ, khi nào thì nó chỉ đóng vai trò là tham chiếu.

Các câu lệnh mô tả cho quan hệ đã bước đầu được hình thành. Chúng sẽ tiếp tục được phân tích và sửa đổi trong hai bước sau: Tinh chỉnh sơ đồ dữ liệu và thiết kế cơ sở dữ liệu mức vật lý. Kết hợp với các câu lệnh tạo chỉ mục, ràng buộc, chúng sẽ trở thành kết quả của giai đoạn thiết kế mức vật lý và sẵn sàng được sử dụng để cài đặt trên một hệ quản trị cơ sở dữ liệu đã lựa chọn.

5. Tinh chỉnh sơ đồ dữ liệu và các dạng chuẩn

Bước thứ tư trong quá trình thiết kế cơ sở dữ liệu đó là phân tích tập các quan hệ trong sơ đồ cơ sở dữ liệu quan hệ mức logic để xác định các vấn đề còn chưa ổn và tiến hành tinh chỉnh chúng. Việc tinh chỉnh sơ đồ dữ liệu được thực hiện dựa trên lý thuyết chuẩn hoá quan hệ.

Thiết kế cơ sở dữ liệu mức khái niệm và logic cho chúng ta một sơ đồ các quan hệ và các ràng buộc toàn vẹn, chúng được xem như như những bước khởi đầu tốt cho quá trình thiết kế cơ sở dữ liệu sau này. Trong bước này, chúng ta cần phải xem xét kỹ hơn các ràng buộc toàn vẹn so với trong mô hình thực thể liên kết. Chúng ta sẽ tập trung vào một lớp ràng buộc quan trọng, đó là phụ thuộc hàm. Ngoài phụ thuộc hàm còn có một số loại ràng buộc toàn vẹn khác như phụ thuộc đa trị, phụ thuộc kết nối, chúng được sử dụng để loại bỏ các dư thừa trong các trường hợp phụ thuộc hàm không thể thực hiện được, tuy nhiên trong giáo trình này chúng ta chỉ dừng ở mức sử dụng các phụ thuộc hàm, bởi nó đã cho phép chúng ta loại bỏ sự dư thừa dữ liệu ở hầu hết các trường hợp thường gặp trong các bài toán thực tế.

5.1. Tinh chỉnh các lược đồ quan hệ

Việc tinh chỉnh sơ đồ dữ liệu chính là quá trình chuẩn hoá để đưa các lược đồ quan hệ về các dạng chuẩn, nhằm loại bỏ tối đa sự lưu trữ dư thừa dữ liệu. Như chúng ta đã biết, lưu trữ dư thừa, trùng lặp dữ liệu là nguyên nhân chính dẫn đến các vần đề về không toàn vẹn dữ liệu và dẫn đến các vấn đề sau:

- Dị thường khi cập nhật: Cùng một dữ liệu, nếu chúng được lưu trữ lặp lại ở nhiều nơi thì khi cập nhật sửa đổi một chỗ, chúng ta phải tìm tất cả những chỗ khác để cập nhật cho đồng bộ, tuy nhiên công việc này khá phức tạp và rất dễ gây ra sự bỏ sót, do đó sẽ dẫn đến tính không nhất quán của dữ liệu.
- Dị thường khi thêm mới: Đôi khi việc chèn thêm dữ liệu mới không thể thực hiện được do cũng phải chèn thêm những dữ liệu khác mà tại thời điểm thêm mới chưa có.
- Dị thường khi xoá bỏ: Khi xoá bỏ một số dữ liệu trong quan hệ, chúng ta cũng có thể đánh mất những dữ liệu khác mặc dù chủ tâm chúng ta không muốn như vậy, do đó sẽ gây ra sự mất mát thông tin.

Tất cả những vấn đề đó khiến chúng ta phải tiến hành chuẩn hoá các quan hệ. Chuẩn hoá được xem là một kỹ thuật để củng cố và nâng cao cho quá trình thiết kế logic, thông qua việc thoả mãn các ràng buộc nhằm loại bỏ các dư thừa dữ liệu. Một quan hệ được xem là ở một dạng chuẩn nào đó nếu nó thoả mãn một tập các ràng buộc nhất định. Ngày nay, trong lý thuyết chuẩn hoá người ta đã xác định được 5 dạng chuẩn, song trên thực tế chúng ta chỉ xét đến 3 dạng chuẩn đầu tiên.

Mục đích của chúng ta trong bước này đó là tạo ra được một cơ sở dữ liệu bao gồm một tập các quan hệ ở dạng chuẩn 3. Các dạng chuẩn 1 và 2 được xem là những bước trung gian để có thể đưa quan hệ đến dạng chuẩn 3.

Để có thể hiểu rõ hơn đối với các dạng chuẩn, chúng ta hãy bắt đầu bằng việc tìm hiểu thế nào là một phụ thuộc hàm.

5.2. Phụ thuộc hàm

Trong một quan hệ R, chúng ta có các thuộc tính X và Y, ta nói thuộc tính Y phụ thuộc hàm vào thuộc tính X trong quan hệ R nếu và chỉ nếu mỗi giá trị của X trong R luôn xác định rõ ràng một giá trị của Y tại mọi thời điểm. X và Y cũng có thể bao gồm nhiều thuộc tính.

Chúng ta kí hiệu phụ thuộc hàm bằng một mũi tên như sau: $X \rightarrow Y$: X xác định Y hay Y phụ thuộc hàm vào X.

Việc thuộc tính Y phụ thuộc hàm vào thuộc tính X sẽ cho chúng ta biết: khi có các giá trị của thuộc tính X thì sẽ xác định được các giá trị của thuộc tính Y.

Ví dụ: Trong quan hệ nhân viên EMPLOYEE, nếu chúng ta biết số hiệu của một nhân viên nào đó thì chúng ta có thể dễ dàng tìm ra tên của anh ta là gì, địa chỉ của anh ta ở đâu. Như vậy các thuộc tính tên Emp_name, địa chỉ Address sẽ phụ thuộc hàm vào thuộc tính số hiệu nhân viên Emp_id. Bởi như chúng ta đã phân tích, mỗi một nhân viên sẽ chỉ có một số hiệu duy nhất. Nhưng ngược lại, nếu chúng ta biết tên của một nhân viên thì chưa chắc chúng ta đã tìm ra được duy nhất số hiệu cũng như địa chỉ của anh ta bởi có thể trong công ty có rất nhiều người cùng có tên như vậy. Vậy trong quan hệ này chúng ta có một phụ thuộc hàm sau: Emp_id $\rightarrow$ {Emp_name, Address}.

Ví dụ: Trong quan hệ EMPLOYEE_PROJECT, chúng ta có các thuộc tính số hiệu nhân viên Emp_id, mã dự án Project_id, ngày bắt đầu tham gia dự án Date_Start. Nếu biết số hiệu nhân viên là NV_15721, mã dự án là DA_1534 thì chúng ta có thể biết ngay được ngày nhân viên NV_15721 tham gia vào dự án DA_1543 là ngày nào.

Nhưng nếu chúng ta chỉ biết số hiệu nhân viên là NV_15721 thì chúng ta không thể biết duy nhất giá trị ngày tham gia dự án bởi nhân viên đó có thể tham gia vào nhiều dự án trong những ngày khác nhau. Cũng như vậy, nếu chúng ta chỉ biết mã dự án là DA_1543 thì chúng ta cũng không thể biết duy nhất một ngày dự án bởi trong dự án đó có nhiều nhân viên bắt đầu tham gia vào những ngày khác nhau. Còn nếu chúng ta biết một ngày nào đó, ví dụ 14/05/2004 thì chúng ta cũng không thể xác định duy nhất các giá trị như mã số dự án, số hiệu nhân viên tham gia vào dự án, bởi cùng ngày hôm đó có thể có nhiều nhân viên tham gia vào nhiều dự án khác nhau. Như vậy chúng ta nhận thấy rằng chỉ khi cùng biết hai giá trị số hiệu nhân viên và mã số dựa án thì chúng ta mới có thể biết duy nhất ngày tham gia vào dự án, vậy chúng ta có phụ thuộc hàm sau: {Emp_id, Project_id} $\rightarrow$ Date_start.

Ví dụ: Cũng tương tự như ví dụ trên, để biết số lượng chính xác (duy nhất) của một mặt hàng đã được bán trong một đơn hàng nào đó trong quan hệ dòng đơn hàng ORDER_LINE, chúng ta cần phải biết số hiệu đơn hàng và số hiệu sản phẩm. Vậy chúng ta có phụ thuộc hàm sau: {Order_id, Product_id} $\rightarrow$ Quantity.

Nhận xét: chúng ta có thể nhận ra ngay một điểm quan trọng sau: Nếu X là khoá chính, khi đó mọi thuộc tính còn lại trong quan hệ đều phải phụ thuộc hàm vào nó. Lưu ý là X có thể là một thuộc tính hoặc một tập thuộc tính.

Sau khi hiểu thế nào là phụ thuộc hàm, chúng ta hãy tiếp tục xem các quan hệ sẽ được đưa về các dạng chuẩn như thế nào.

5.3. Các dạng chuẩn

a. Dạng chuẩn 1NF

Chúng ta nói rằng một quan hệ ở dạng chuẩn 1 nếu tất cả các giá trị các thuộc tính của nó là sơ cấp (tức là chỉ chứa các giá trị được coi là không thể phân chia nhỏ hơn được nữa).

Ví dụ: Để dễ minh hoạ cho quá trình chuẩn hoá ở các phần tiếp sau, ở đây giả thiết chúng ta có lược đồ quan hệ EMP_PROJECT bao gồm những thuộc tính như sau:

EMP_PROJECT (Emp_id, Emp_name, Department, Mgr_id, Mgr_name, Project)

Và một quan hệ của nó:

Eno	Enam	Depart	Mgr	Mgrna	Project
7902	*Sửu*	*10_Sales*	*7988*	*Dậu*	*15_Feasibility,*
7988	*Dậu*	*20_Marketing*	*7699*	*Thìn*	*15_Feasibility,*
7562	*Hợi*	*10_Sal*	*769*	*Thìn*	*25_Anal*

Quan hệ này rõ ràng là chưa ở dạng chuẩn 1. Vì lý do như sau:

Mỗi nhân viên có thể tham gia vào nhiều dự án, và số lượng dự án mà mỗi nhân viên tham gia là có thể khác nhau. Như ta thấy Dậu và Sửu tham gia vào 3 dự án, trong khi đó Hợi chỉ tham gia vào một dự án mà thôi. Do đó, trong trường hợp này, để đưa quan hệ về dạng chuẩn 1 ta phải thêm các bộ mới. Cụ thể ứng với mỗi dự án trong thuộc

tính Project của một nhân viên, ta phải thêm một dòng mới, với các trường còn lại được giữ nguyên ở dòng mới, còn các dự án (trong thuộc tính Project) phải được tách ra thành từng dự án.

Tuy nhiên, ta nhận thấy rằng trong quan hệ còn có 2 thuộc tính chưa ở dạng nguyên tố, đó là thuộc tính Department, bao gồm mã và tên bộ phận, và thuộc tính vừa tách Project (bao gồm cả mã và tên dự án). Do đó, ta tách như sau:

- Thuộc tính Department sẽ được tách thành 2 thuộc tính Dno và Dname, trong đó Dno sẽ lưu mã bộ phận và Dname lưu tên bộ phận.
- Thuộc tính Project cũng sẽ được tách thành 2 thuộc tính Pno và Pname, trong đó thuộc tính Pno lưu mã dự án và thuộc tính Pname lưu tên dự án.

Đến đây ta được quan hệ EMPLOYEE đã ở dạng chuẩn 1, để phục vụ cho việc minh hoạ ở các bước sau, chúng ta thêm vào quan hệ hai thuộc tính nữa, đó là: ngày bắt đầu tham gia dự án của nhân viên Start, và số tiền của dự án mà mỗi nhân viên được trả Bill. Ta có:

En	**Ena**	**D**	**Dnam**	**Mg**	**Mgr**	**P**	**Pnam**	**Start**	**Bi**
79	*Sửu*	*1*	*Sales*	*798*	*Dâu*	*15*	*Feasi*	*10/09*	*10*
79	*Sửu*	*1*	*Sales*	*798*	*Dậu*	*35*	*Testin*	*20/09*	*10*
79	*Sửu*	*1*	*Sales*	*798*	*Dậu*	*45*	*Hand*	*20/10*	*15*
79	*Dậu*	*2*	*Marke*	*769*	*Thìn*	*15*	*Feasi*	*15/09*	*20*
79	*Dậu*	*2*	*Marke*	*769*	*Thìn*	*25*	*Analy*	*14/09*	*25*
79	*Dậu*	*2*	*Marke*	*769*	*Thìn*	*45*	*Hand*	*20/10*	*20*
75	*Hợi*	*1*	*Sales*	*769*	*Thìn*	*25*	*Analy*	*20/05*	*15*

Như trong các phần trước, trong khi thực hiện các bước ánh xạ từ mô hình thực thể liên kết sang quan hệ, chúng ta đã trình bày quy trình để loại bỏ các thuộc tính đa trị từ các kiểu thực thể. Do vậy nếu chúng ta đã thực hiện bước thiết kế logic này rồi thì ở đây chúng ta có thể không cần quan tâm đến việc liệu một quan hệ có ở dạng chuẩn một hay không.

b. Dạng chuẩn 2NF

Một quan hệ được gọi là ở dạng chuẩn hai nếu quan hệ đó đã ở dạng chuẩn một và mọi thuộc tính không khoá đều phụ thuộc hàm đầy đủ vào khoá chính. Tức là không một thuộc tính không khoá nào phụ thuộc hàm một phần vào khoá chính.

Một quan hệ ở dạng chuẩn một sẽ ở dạng chuẩn hai nếu một trong những điều kiện sau được áp dụng:

- Khoá chính chỉ chứa duy nhất một thuộc tính. Ví dụ như thuộc tính Emp_id trong quan hệ EMPLOYEE. Thuộc tính Project_id trong quan hệ PROJECT. Thuộc tính Supplier_id trong quan hệ SUPPLIER.
- Không tồn tại các thuộc tính không khoá trong quan hệ. Tức là tất cả các thuộc tính trong quan hệ đều tham gia vào khoá chính. Ví dụ như trong quan hệ ITEM_COMPONENT, nếu chỉ bao gồm hai thuộc tính Itemp_id và Component_id thì cả 2 thuộc tính này đều là khoá của quan hệ.
- Mọi thuộc tính không khoá đều phụ thuộc hàm đầy đủ vào tất cả các thuộc tính khoá. Ví dụ, trong quan hệ SUPPLIER_ITEM, thuộc tính đơn giá Unit_price phụ thuộc vào cả hai thuộc tính mã nhà cung cấp

Sup_id và số hiệu vật liệu Item_id nên nó ở dạng chuẩn hai.

Quay trở lại quan hệ EMP_PROJECT trên của chúng ta, rõ ràng khoá của quan hệ này là kết hợp của hai thuộc tính Emp_id và Pno. Chúng ta cũng nhận thấy, mỗi một tên dự án chỉ phụ thuộc hàm vào mã dự án, tức là phụ thuộc hàm bộ phận vào một thuộc tính của khoá {Emp_id, Pno}. Cũng tương tự như vậy, các thuộc tính {Ename, Dno, Dname, Mgr_id, Mgrname} chỉ phụ thuộc hàm bộ phận vào thuộc tính Eno, là một thuộc tính của khoá chính K. Chính điều này đã làm xuất hiện những vấn đề sau khi cập nhật và lưu trữ dữ liệu:

- Phép thay đổi: Việc lưu trữ thông tin về một nhân viên, ví dụ như nhân viên Sửu phải lặp lại 3 lần tên phòng, 3 lần tên người quản lý. Khi cần thay đổi phòng làm việc của Sửu chẳng hạn, cần phải thay đổi 3 lần các thông tin này, rõ ràng là tốn kém thời gian hơn. Hơn nữa khi thay đổi thông tin về một nhân viên lại không liên quan gì tới thông tin dự án. Để xác định những thông tin về một nhân viên, chỉ cần thuộc tính Eno là xác định được duy nhất thông tin về họ.
- Phép chèn: Trong quan hệ EMPLOYEE chỉ chứa thông tin về những nhân viên đã tham gia một dự án nào đó. Nếu muốn chèn thêm một nhân viên vào quan hệ thì không được vì không rõ nhân viên đó đã tham gia vào dự án nào chưa. Bất lợi này chỉ được khắc phục nếu loại bỏ những thông tin về dự án ra khỏi quan hệ .

- Phép loại bỏ: Giả sử vì một lý do nào ta đó cần huỷ bỏ dự án Analysis ra khỏi quan hệ, khi đó rõ ràng là thông tin liên quan đến nhân viên Hợi sẽ bị mất.

Để khắc phục những bất lợi trên, ta tách quan hệ EMPLOYEE về 3 quan hệ ở dạng chuẩn 2 như sau:

Quan hệ thứ nhất: PROJECT:

P	**Pnam**
15	*Feasi*
25	*Analy*
35	*Testin*
45	*Hand*

Trong quan hệ này, thuộc tính Pno sẽ làm khoá chính.

Quan hệ thứ 2: ASSIGNMENT:

En	**P**	**Start**	**Bi**
79	*15*	*10/09*	*10*
79	*35*	*20/09*	*10*
79	*45*	*20/10*	*15*
79	*15*	*15/09*	*20*
79	*25*	*14/09*	*25*
79	*45*	*20/10*	*20*
75	*25*	*20/05*	*15*

Trong quan hệ này, tập thuộc tính {Eno,Pno} vẫn làm khoá chính.

Quan hệ thứ 3: EMPLOYEE:

En	Ena	D	Dnam	Mg	Mgr
79	*Sửu*	*1*	*Sales*	*798*	*Dậu*
79	*Dậu*	*2*	*Marke*	*769*	*Thìn*
75	*Hợi*	*1*	*Sales*	*769*	*Thìn*

Trong quan hệ này, thuộc tính Eno sẽ làm khoá chính.

Đến đây ta đã có 3 quan hệ đã ở dạng chuẩn 2.

c. Dạng chuẩn 3NF

Một quan hệ đã là 2NF được xem là ở dạng chuẩn 3NF nếu tất cả các phụ thuộc hàm giữa khoá chính và các thuộc tính khác của nó đều là trực tiếp.

Hay nói cách khác, mọi thuộc tính không nằm trong khoá chính đều không phụ thuộc hàm vào một thuộc tính không phải là khoá chính.

Để phân rã các quan hệ ở dạng chuẩn 2 thành các quan hệ ở dạng chuẩn 3, ta phải thực hiện các bước sau:

- Tìm ra các phụ thuộc hàm bắc cầu vào khoá chính có dạng : $K \rightarrow X$, $X \rightarrow Y$, trong đó K là tập các thuộc tính khoá, X và Y là tập các thuộc tính không khoá.
- Nếu tìm thấy phụ thuộc hàm bắc cầu như trên thì tiến hành tách quan hệ hiện thời thành hai quan hệ, quan hệ thứ nhất bao gồm các thuộc tính X và Y, quan hệ thứ hai bao gồm các thuộc tính trong quan hệ ban đầu trừ những thuộc tính trong Y.

- Kiểm tra với các quan hệ con xem đã ở dạng chuẩn 3 hay chưa, nếu chưa lại thực hiện tách tiếp như bước trên.

Ví dụ: quay trở lại các quan hệ đã ở dạng chuẩn 2NF trên chúng ta phân tích tiếp:

Hai quan hệ PROJECT và ASSIGMENT rõ ràng là đã ở dạng chuẩn 3. Vì ở quan hệ Projects, ngoài khoá chính ra không còn có thuộc tính nào khác.

Trong quan hệ ASSIGMENT, hai thuộc tính mô tả còn lại Start và Bill không phụ thuộc vào nhau, do đó quan hệ này cũng ở dạng chuẩn 3.

Trong quan hệ còn lại: EMPLOYEE, ta nhận thấy rằng thuộc tính Dname chỉ phụ thuộc hàm vào thuộc tính Dno, mà không phụ thuộc hàm trực tiếp vào khoá chính K = Eno, rõ ràng Dname đã phụ thuộc hàm bắc cầu vào khoá chính K thông qua thuộc tính Dno. Cũng tương tự như vậy đối với thuộc tính Mgrname, nó chỉ phụ thuộc hàm vào thuộc tính Mgr_id mà không phụ thuộc hàm trực tiếp vào thuộc tính Eno.

Do đó ở quan hệ này, ta phải tách thành 3 quan hệ là DEPARTMENT, MANAGER và EMPLOYEE ở dạng chuẩn 3 như sau:

Quan hệ thứ nhất: DEPARTMENT

D	**Dnam**
2	*Marke*
1	*Sales*

Với khoá là thuộc tính: Dno

Quan hệ thứ hai: MANAGER

Mgr	Mgr
798	*Dậu*
769	*Thìn*

Với khoá là thuộc tính: Mgr_id

Và quan hệ thứ ba: EMPLOYEE

En	Ena	D	Mgr	Mgr
79	*Sửu*	*1*	*798*	*Dâu*
79	*Dậu*	*2*	*769*	*Thìn*
75	*Hợi*	*1*	*790*	*Dần*

Với khoá là thuộc tính: Eno

Tiếp tục chuẩn hoá với 2 quan hệ trên:

Trong các quan hệ DEPARTMENT và MANAGER, vì ngoài khoá chính ra, không còn có thuộc tính nào khác, do đó chúng đã ở dạng chuẩn 3.

Trong quan hệ EMPLOYEE, lần lượt tìm phụ thuộc hàm bắc cầu vào khoá chính Eno của các thuộc tính còn lại, ta không tìm thấy phụ thuộc hàm nào thỏa phụ thuộc bắc cầu, do đó quan hệ cũng ở dạng chuẩn 3.

Như vậy kết hợp với 2 quan hệ còn lại là:

Quan hệ PROJECT:

P	Pnam
15	*Feasi*
25	*Analy*
35	*Testin*
45	*Hand*

và quan hệ ASSIGNMENT:

En	P	Start	Bi
79	*15*	*10/09*	*10*
79	*35*	*20/09*	*10*
79	*45*	*20/10*	*15*
79	*15*	*15/09*	*20*
79	*25*	*14/09*	*25*
79	*45*	*20/10*	*20*
75	*25*	*20/05*	*15*

Ta có 5 quan hệ đã ở dạng chuẩn 3NF.

Quay trở lại ví dụ chung của công ty kinh doanh thiết bị đồ gỗ, nếu trong bước thiết kế cơ sở dữ liệu mức khái niệm (ở các hình 2.3.7.3 và 2.3.7.4) chúng ta không tiến hành phân tách liên kết "Orders" thành hai kiểu thực thể là ORDER và ORDER_LINE thì khi hình thành quan hệ ở bước thiết kế cơ sở dữ liệu mức logic chúng ta sẽ có một quan hệ ORDER với các thuộc tính sau:

ORDER(Order_id, Order_date, Order_desc, Cust_id, Prod_it, Quantity).

Trong đó 2 thuộc tính Order_id và Prod_id là các thuộc tính khoá của quan hệ.

Rõ ràng quan hệ ORDER trên không ở dạng chuẩn 3NF bởi các thuộc tính Order_date, Order_desc và Prod_id chỉ phụ thuộc vào thuộc tính Order_id mà thôi, còn trường Quantity thì phụ thuộc đầy đủ vào toàn bộ các thuộc tính khoá của quan hệ (Order_id, Prod_id).

Do vậy nếu trong các bước đó chúng ta chưa phân tích kỹ được như đã trình bày thì tại bước tinh chỉnh và chuẩn hoá các quan hệ này chúng ta phải tiến hành tách quan hệ ORDER trên thành hai quan hệ mới đó là:

ORDER (Order_id, Order_date, Order_desc, Prod_id)

ORDER_LINE (Order_id, Prod_id, Quantity)

Đến đây chúng ta cũng đã có hai quan hệ đã ở dạng chuẩn 3NF như trong các bước trên. Như vậy có thể nhận thấy rằng, chỉ khi kết thúc bước này, chúng ta mới có được một sơ đồ thiết kế cơ sở dữ liệu mức khái niệm hoàn chỉnh theo nghĩa mọi quan hệ đã ở dạng chuẩn 3NF và đã tối thiểu hoá được việc lưu trữ dư thừa dữ liệu.

Trong lý thuyết thiết kế cơ sở dữ liệu, ngoài 3 dạng chuẩn trình bày trên, người ta còn phát triển tiếp các dạng chuẩn khác như dạng chuẩn BCNF, dạng chuẩn 4 với khái niệm phụ thuộc đa trị và dạng chuẩn 5 với khái niệm phụ thuộc kết nối . Tuy nhiên trong giáo trình này, chúng ta chỉ dừng ở dạng chuẩn 3, vì như trên đã trình bày, trên thực tế khi các quan hệ được chuẩn hoá ở dạng chuẩn 3, chúng đã hạn chế khá tốt tính lưu trữ dư thừa dữ liệu. Ngoài ra, cũng nhận thấy rằng, trong quá trình tiến hành chuẩn hoá các quan hệ như trên, tính toàn vẹn của dữ liệu vẫn được đảm bảo.

6. Thiết kế cơ sở dữ liệu vật lý

Trong bước này, chúng ta phải xem xét, cân nhắc kỹ các vấn đề liên quan đến khối lượng dữ liệu mà cơ sở dữ liệu đang xây dựng cần phải đáp ứng, ngoài ra, cũng trong bước này có thể chúng ta phải tiến hành tinh chỉnh thêm thiết kế các quan hệ để đảm bảo chắc chắn rằng nó đáp ứng được các mục tiêu về hiệu năng truy suất.

Bước này có thể đơn giản chỉ là việc xây dựng các chỉ số cho một số bảng dữ liệu, phân cụm một số bảng, hoặc cũng có thể bao gồm cả bước thiết kế lại một phần của sơ đồ cơ sở dữ liệu logic có được trong các bước thiết kế trước.

Chúng ta cũng cần lưu ý rằng thiết kế cơ sở dữ liệu mức vật lý không bao gồm quá trình cài đặt cơ sở dữ liệu đó.

Để có thể thực hiện được bước này, chúng ta cần phải có sự hiểu biết thấu đáo về hệ quản trị cơ sở dữ liệu sẽ được lựa chọn để cài đặt cho cơ sở dữ liệu, đặc biệt là:

- Các đối tượng trong cơ sở dữ liệu mà hệ quản trị cơ sở dữ liệu được lựa chọn hỗ trợ, cấu trúc vật lý và các tệp cần thiết cho các đối tượng đó.
- Cách thức mà hệ quản trị cơ sở dữ liệu đó hỗ trợ cho các đối tượng: chỉ mục, toàn vẹn tham chiếu, các ràng buộc, các kiểu dữ liệu,...
- Ưu nhược điểm đối với mỗi một phiên bản của hệ quản trị cơ sở dữ liệu đã lựa chọn.
- Các tham số cấu hình cho hệ quản trị cơ sở dữ liệu.
- Các câu lệnh định nghĩa dữ liệu để có thể chuyển đổi thành các đối tượng dữ liệu mức vật lý

6.1. Phân tích kích thước và nhu cầu sử dụng dữ liệu

Kích thước dữ liệu và việc thống kê những dữ liệu thường xuyên được sử dụng là những tiêu chuẩn quan trọng ở đầu vào cho quá trình thiết kế cơ sở dữ liệu mức vật lý. Do vậy chúng ta cần phải đánh giá được kích thước của cơ sở dữ liệu cũng như việc sử dụng dữ liệu.

Kích thước và nhu cầu sử dụng dữ liệu thường được phát hiện ra trong quá trình phân tích hệ thống. Kích thước của dữ liệu đại diện cho độ phức tạp của hoạt động nghiệp vụ, và nên được đánh giá trong một khoảng thời gian vài năm tuỳ thuộc vào từng ứng dụng cơ sở dữ liệu cụ thể. Nhu cầu sử dụng hay tần xuất truy nhập dữ liệu được đánh giá dựa trên thời gian của các sự kiện, độ lớn của các giao dịch, các hoạt động liên quan đến báo cáo, truy vấn tìm kiếm dữ liệu. Ngày nay, một số cơ sở dữ liệu lớn thường hỗ trợ các truy vấn dạng ad hoc, những truy vấn này chiếm đáng kể thời gian thực hiện bởi chúng phải xử lý trên những lượng dữ liệu vô cùng lớn, do vậy đôi khi với những cơ sở dữ liệu có sử dụng truy vấn này chúng ta nên để ý đến kích thước của dữ liệu hơn là tần xuất sử dụng các truy vấn đó.

Trong quá trình phân tích kích thước và nhu cầu sử dụng dữ liệu, chúng ta nên giải thích được tính tự nhiên của các truy nhập thông qua các hình vẽ mô tả các đường truy nhập.

Ví dụ chúng ta có thể mô tả trong số 200 truy nhập đến dữ liệu trong bảng đơn hàng ORDER, có 150 truy nhập dựa trên khoá chính Order_id (như hỏi thông tin về một đơn hàng cụ thể), 50 truy nhập còn lại là dựa trên các giá trị của trường ngày đơn hàng Order_date.

Những mô tả rõ ràng này sẽ giúp cho công đoạn lựa chọn chỉ mục, như sẽ trình bày trong phần dưới đây, thuận tiện hơn, chính xác hơn và phù hợp với thực tế. Cũng tương tự như vậy, chúng ta nên mô tả cả các kết quả trả về của một truy nhập: đó sẽ là một thao tác tạo mới, thao tác sửa đổi, xoá bỏ hay trả về của dữ liệu.

6.2. Thiết kế các trường

Trường là đơn vị nhỏ nhất của một dữ liệu ứng dụng mà phần mềm hệ thống có thể nhận ra, phần mềm hệ thống có thể là ngôn ngữ lập trình hay hệ quản trị cơ sở dữ liệu. Một trường tương ứng với một thuộc tính đơn trong mô hình dữ liệu logic.

Đối với mỗi trường dữ liệu, chúng ta phải quan tâm đến những vấn đề sau: kiểu dữ liệu (hay kiểu lưu trữ) sẽ được sử dụng để biểu diễn các giá trị của trường đó, các kiểm soát toàn vẹn dữ liệu, và làm thế nào để một hệ quản trị cơ sở dữ liệu có thể quản lý được vấn đề mất mát giá trị trong trường đó.

a. Lựa chọn kiểu dữ liệu

Mỗi một trường dữ liệu được lưu trữ trong cơ sở dữ liệu bắt buộc phải thuộc một kiểu dữ liệu mà hệ quản trị cơ sở dữ liệu đó hỗ trợ. Việc lựa chọn một kiểu dữ liệu phải thoả mãn 4 mục đích sau:

- Tối thiểu hoá không gian lưu trữ
- Biểu diễn được tất cả các khả năng của giá trị cần lưu trữ
- Nâng cao tính toàn vẹn dữ liệu

- Hỗ trợ tất cả các thao thác trên dữ liệu

Tuy nhiên, những mục đích này có thể có mức độ quan trọng khác nhau tuỳ thuộc vào từng bài toán cụ thể.

Lựa chọn đúng kiểu dữ liệu cho một trường, trong không gian tối thiểu, nhằm biểu diễn mọi giá trị có thể xảy ra (tất nhiên là loại trừ những giá trị không hợp lệ) cho thuộc tính tương ứng và có thể hỗ trợ các thao tác trên dữ liệu (ví dụ các kiểu dữ liệu số phải đáp ứng được các thao tác là các hàm số học tính toán, các kiểu dữ liệu ký tự phải đáp ứng được các thao tác ghép xâu, tách xâu). Trong mô hình dữ liệu mức khái niệm chúng ta đã xây dựng các ràng buộc miền cho các thuộc tính, trong bước này, những thông tin đó cũng sẽ giúp cho chúng ta khá nhiều trong việc lựa chọn kiểu dữ liệu cho các thuộc tính đó.

Việc đạt được 4 mục tiêu trên hoàn toàn không dễ dàng và chúng ta cần phải nắm bắt được toàn bộ các trường hợp có thể xảy ra. Ví dụ, chúng ta có kiểu dữ liệu SmallInt có độ rộng lớn nhất là 2 bytes. Giả sử kiểu dữ liệu này là đủ để biểu diễn cho trường số lượng bán Quantity, song nếu chúng ta lại muốn tính toán trên kiểu dữ liệu này như tính tổng, thì có thể kết quả tính sẽ vượt qua độ rộng cho phép, và do vậy có thể dẫn đến những sai sót.

Chúng ta cũng cần lưu ý rằng với những kiểu dữ liệu cụ thể sẽ có những khả năng thao tác cụ thể, ví dụ với kiểu dữ liệu DATE chỉ cho phép những phép toán số học cụ thể trên trường DATE mà thôi.

b. Kiểm soát các toàn vẹn dữ liệu

Trong nhiều hệ quản trị cơ sở dữ liệu, kiểm soát các toàn vẹn dữ liệu (tức là quản lý các giá trị có thể nhận của một

trường dữ liệu) có thể được xây dựng thành những cấu trúc vật lý cho các trường.

Kiểu dữ liệu là một loại kiểm soát toàn vẹn, bởi nó giới hạn kiểu của dữ liệu mà một trường có thể nhận được. Ví dụ nếu một trường được khai báo có kiểu dữ liệu là kiểu số thì khi đó không thể chèn các dữ liệu kiểu ký tự vào trường đó được.

Độ rộng của trường dữ liệu cũng là một kiểu kiểm soát toàn vẹn dữ liệu, bởi nó giới hạn kích thước của dữ liệu đưa vào. Ví dụ nếu một trường lưu bậc lương của nhân viên có giá trị không vượt quá một số nguyên thì không thể lưu những giá trị bằng hoặc lớn hơn 10 được. Hay trường tên của một nhân viên không bao giờ vượt quá 30 ký tự chẳng hạn.

Một số kiểm soát toàn vẹn dữ liệu khác:

- Giá trị mặc định: là giá trị được gán cho một trường nếu một bản ghi mới được tạo ra, trường được định nghĩa giá trị mặc định sẽ mang giá trị đó trừ khi người sử dụng thay đổi. Việc gán giá trị mặc định cho một trường sẽ giúp giảm thời gian nhập dữ liệu cho người sử dụng, và nó cũng giảm tỷ lệ mắc lỗi khi nhập dữ liệu (một trong những lỗi hay xảy ra nhất). Ví dụ, mỗi một chứng từ mua hàng được tạo ra đều mang giá trị là ngày hiện thời thì chúng ta có thể đặt giá trị mặc định cho thuộc tính này là ngày hệ thống (sysdate).
- Điều kiện khoảng: Một điều kiện khoảng sẽ giới hạn tập các giá trị cho phép đối với một trường dữ liệu. Một khoảng có thể là một khai báo về cận trên và cận dưới đối với một trường dữ liệu kiểu số nào đó có thể

được nhận giá trị. Nói một cách khác, giá trị của dữ liệu đưa vào được gọi là hợp lệ nếu nó nằm trong khoảng giữa cận trên và cận dưới đó. Có một lưu ý nhỏ đó là nên đặt các điều kiện khoảng này trong khi định nghĩa các bảng dữ liệu của cơ sở dữ liệu, bởi ta cũng có thể giới hạn các giá trị nhập vào thông qua chương trình ứng dụng, và khi đó thì rất dễ gây ra lỗi không nhất quán, triệt để và khó khăn trong quá trình tìm và thay đổi khi chúng ta muốn sửa đổi giá trị khoảng này. Ví dụ như nếu chúng ta qui ước thuộc tính bậc lương của nhân viên có kiểu dữ liệu là Smallint, có độ rộng là 1 và nằm trong khoảng từ 1 đến 7 thì ta có thể khai báo giá trị khoảng cho thuộc tính này có cận dưới bằng 1 và cận trên bằng 7. Khi đó cơ sở dữ liệu sẽ ngăn cản được những giá trị đưa vào không hợp lệ như các giá trị 8, 9, mặc dù độ dài của chúng thoả mãn điều kiện ràng buộc về độ rộng của trường.

- Điều kiện giá trị Null: một giá trị null được định nghĩa như là một giá trị không xác định. Đối với mỗi một thuộc tính tham gia vào khoá chính phải được đảm bảo cấm được các giá trị null. Ngoài ra một số thuộc tính khác cũng có thể khai báo điều kiện này. Ví dụ, trong quan hệ nhân viên EMPLOYEE, ngoài thuộc tính mã hiệu nhân viên Emp_id là khoá chính không được phép mang giá trị null, thuộc tính Emp_name cũng nên được khai báo không được phép mang giá trị null, bởi nếu chúng ta biết mã hiệu của một nhân viên mà không biết tên của anh ta là ai thì cũng vô nghĩa.

- Toàn vẹn tham chiếu: Toàn vẹn tham chiếu là một dạng của toàn vẹn theo điều kiện giá trị khoảng, bởi những giá trị hợp lệ phải được lấy từ (tham chiếu đến) một quan hệ nào đó được trường đó tham chiếu đến. Các khoá ngoại lai là một trường hợp phổ biến của dạng kiểm soát này. Ví dụ trong quan hệ EMPLOYEE, chúng ta có thuộc tính Dept_id là thuộc tính khoá ngoại lai, nó được tham chiếu đến thuộc tính khoá chính Dept_id của quan hệ DEPARTMENT. Do vậy mọi giá trị mà thuộc tính Dept_id trong quan hệ EMPLOYEE phải được lấy từ tập giá trị trong thuộc tính Dept_id của quan hệ DEPARTMENT.

6.3. Tổ chức các bản ghi vật lý và phi chuẩn

a. Tổ chức các bản ghi vật lý

Trong mô hình dữ liệu logic, chúng ta đã nhóm vào một quan hệ các thuộc tính được xác định bởi cùng một khoá chính. Trái lại, một bản ghi vật lý là một nhóm các trường được lưu trữ liên tiếp trong các vị trí của bộ nhớ và được trả về dưới dạng một đơn vị đọc/ghi. Việc thiết kế các bản ghi vật lý bao gồm việc lựa chọn thứ tự các trường được lưu trữ kế tiếp nhằm đạt được hai mục đích: sử dụng hiệu quả vùng lưu trữ thứ cấp và tăng tốc độ xử lý dữ liệu.

Vấn đề sử dụng hiệu quả vùng lưu trữ thứ cấp bị ảnh hưởng bởi cả kích thước của bản ghi vật lý lẫn cấu trúc của vùng lưu trữ thứ cấp. Các hệ điều hành máy tính đọc dữ liệu từ ổ cứng dưới các dạng đơn vị được gọi là trang (page), chứ không phải là các bản ghi vật lý. Một trang là lượng dữ liệu

được đọc hoặc viết vào bộ nhớ thứ cấp trong một phép vào hoặc ra. Kích thước của một trang là được xác định trước và được lựa chọn sao cho đạt hiệu quả cao nhất khi sử dụng RAM. Phụ thuộc vào từng hệ thống máy tính, một bản ghi vật lý có thể được phép hoặc không được phép nằm trên hai trang khác nhau. Nếu kích thước của một trang không là bội số của kích thước bản ghi vật lý, khi đó một số vùng cuối của trang sẽ bị lãng phí. Số lượng các bản ghi vật lý có thể lưu trong một trang được gọi là hệ số khối (blocking factor). Nếu không gian lưu trữ khan hiếm và các bản ghi vật lý không thể nằm trên các trang khác nhau, việc có nhiều bản ghi vật lý trong một quan hệ logic sẽ tối thiểu hoá việc tiêu tốn không gian lưu trữ.

Việc thiết kế một bản ghi vật lý để bảo toàn không gian lưu trữ sẽ dễ dàng hơn nếu mọi trường dữ liệu là có độ dài cố định. Trong trường hợp này, các trường được đặt theo thứ tự trường này nối tiếp trường kia. Với các trường kiểu ký tự, mỗi giá trị sẽ được gán thêm các giá trị trống vào cuối, còn với trường số nó sẽ được gán thêm các số 0 vào trước. Với các trường có kích thước thay đổi, việc lưu trữ phức tạp hơn. Ví dụ kiểu trường VARCHAR2 mà hệ quản trị cơ sở dữ liệu Oracle hỗ trợ là một kiểu trường có kích thước thay đổi.

Lưu ý: Ngày nay hầu hết các hệ quản trị cơ sở dữ liệu lớn đều hỗ trợ việc tổ chức các bản ghi vật lý thông qua các tham số thiết lập cho cơ sở dữ liệu, do vậy, chúng ta không cần đi quá sâu vào việc xem xét các bản ghi được lưu trữ và truy suất vật lý như thế nào mà nên quan tâm đến kích thước của chúng tương ứng với mỗi khối dữ liệu mà hệ quản trị cơ sở dữ liệu đó hỗ trợ. Ví dụ như với hệ quản trị

cơ sở dữ liệu Oracle, kích thước phổ biến của mỗi database block mà nó hỗ trợ là 8kb, tuy nhiên nếu các bản ghi có kích thước khá lớn thì chúng ta có thể sửa đổi tham số này thành 16kb để có thể tận dụng được thêm không gian lưu trữ, giảm thiểu sự phân mảnh.

b. Phi chuẩn

Những vấn đề được trình bày trên đây về thiết kế các bản ghi vật lý chủ yếu tập trung vào việc sử dụng hiệu quả không gian lưu trữ. Trong hầu hết các trường hợp, mục đích thứ 2 của việc thiết kế các bản ghi vật lý - tức là xử lý hiệu quả dữ liệu - chiếm hầu hết thời gian thiết kế cơ sở dữ liệu mức vật lý. Việc thiết kế cơ sở dữ liệu vật lý nhằm xử lý dữ liệu một cách hiệu quả, giống như việc sắp xếp sao cho có thể tìm kiếm một cuốn sách hiệu quả nhất trong một thư viện, phụ thuộc rất nhiều vào việc sắp xếp gần nhau những dữ liệu (sách) liên quan. Thông thường, tất cả các thuộc tính xuất hiện trong một quan hệ không được sử dụng đồng thời, và dữ liệu từ các quan hệ khác nhau thường được kết hợp lại để trả lời cho một truy vấn hay để tạo ra một báo cáo. Do vậy, mặc dù các quan hệ đã được chuẩn hoá có thể giải quyết được các dị thường trong lưu trữ, cập nhật, bảo trì cơ sở dữ liệu, nhưng nếu chúng được thiết kế quá chặt chẽ thì có thể dẫn tới việc xử lý dữ liệu không hiệu quả.

Một số kết quả so sánh trên một cơ sở dữ liệu được chuẩn hoá hoàn và một cơ sở dữ liệu chỉ được chuẩn hoá một phần. Kết quả cho thấy rằng, việc xử lý trên những cơ sở dữ liệu chỉ được chuẩn hoá một phần còn tỏ ra nhanh hơn trên những cơ sở dữ liệu được chuẩn hoá hoàn toàn. Mặc dù kết quả này phụ thuộc rất lớn vào cơ sở dữ liệu cũng

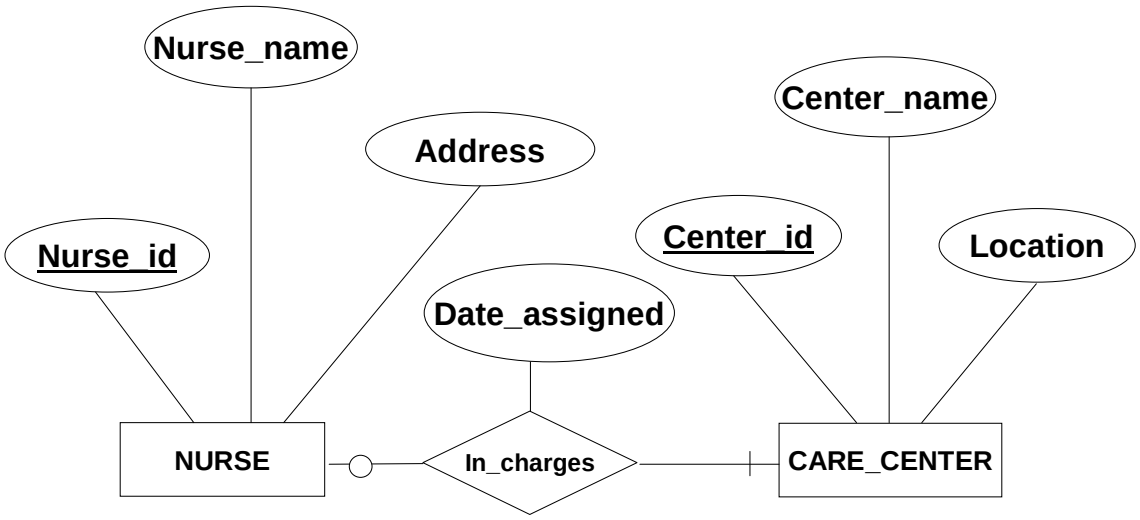
như loại xử lý dữ liệu, nhưng nó cũng cho thấy rằng chúng ta nên cân nhắc cẩn thận khi thiết kế các bản ghi vật lý với việc chuẩn hoá các quan hệ cho cơ sở dữ liệu.

Phi chuẩn là một kỹ thuật chuyển đổi các quan hệ đã được chuẩn hoá ở mức cao thành những dạng chuẩn ở mức thấp hơn nhằm tăng tốc độ truy nhập trong cơ sở dữ liệu. Tuy nhiên chúng ta cũng cần phải lưu ý rằng, việc thiết kế phi chuẩn sẽ làm gia tăng khả năng xảy ra lỗi cũng như vấn đề không nhất quán dữ liệu và khiến cho việc lập trình ứng dụng phức tạp hơn.

Dưới đây chúng ta sẽ xem xét một số trường hợp có thể tiến hành phi chuẩn, từ đó có thể rút ra kinh nghiệm trong quá trình thiết kế cơ sở dữ liệu mức vật lý:

- Trường hợp hai quan hệ được hình thành từ hai kiểu thực thể tham gia vào liên kết một - một: Nếu việc truy suất giữa hai quan hệ diễn ra thường xuyên thì chúng ta nên kết hợp chúng thành một quan hệ.

Ví dụ: Hình 2.6.3.1 dưới đây cho thấy, một quan hệ NURSE liên kết với một quan hệ CARE_CENTER thông qua khoá Nurse_id, tức là một y tá có thể đảm nhiệm một trung tâm chăm sóc bệnh nhân. Chúng ta có thể kết hợp 2 quan hệ này thành một, trong đó giá trị trong các trường Center_name, Location, Date_assigned có thể mang giá trị null đối với những y tá không phụ trách trung tâm nào. Chúng ta cũng nhận thấy rằng khi kết nối hai quan hệ như vậy thì chúng ta cũng không cần đến thuộc tính Center_id.



Hai quan hệ được chuẩn hoá

NURSE

Nurse_id	Nurse_name	Address

CARE_CENTER

Center_id	Center_name	Nurse_id	Location	Date_assigned

Quan hệ được phi chuẩn hoá

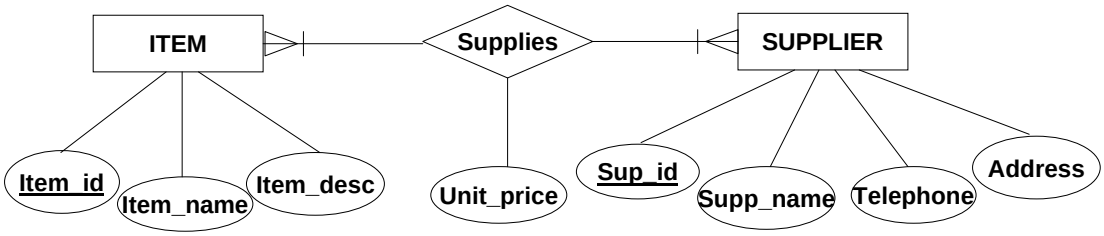
NURSE

Nurse_id	Nurse_name	Address	Centre_name	Location	Date_assigned

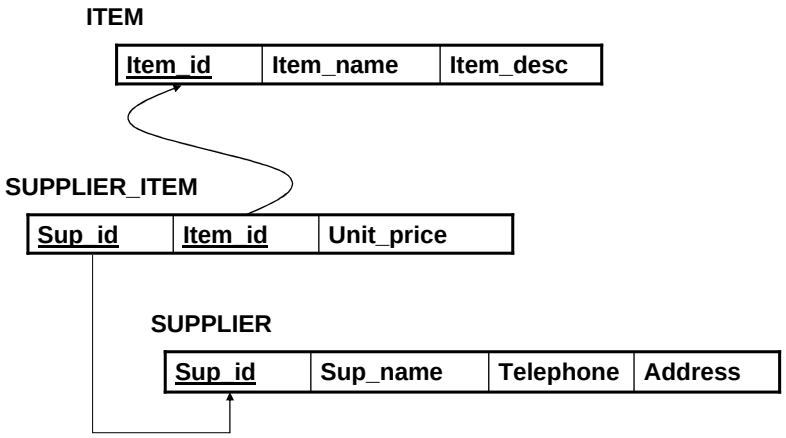
Hình 2.6.3.1 Phi chuẩn trong trường hợp liên kết một - một

- Trường hợp các quan hệ được hình thành từ một liên kết nhiều-nhiều với các thuộc tính trong liên kết không phải là khoá: Thay vì phải hình thành 3 quan hệ chúng ta có thể kết hợp thành 2 quan hệ chính. Chúng ta gộp các thuộc tính trong một quan hệ vào trong quan hệ được hình thành từ liên kết.

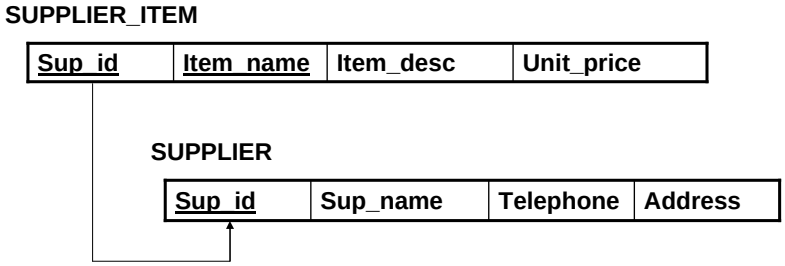
Ví dụ: Hình 2.6.3.2 sau là một ví dụ về các quan hệ và liên kết minh hoạ cho nghiệp vụ liên kết báo giá cho các mặt hàng khác nhau của các nhà cung cấp khác nhau. Trong trường hợp này, các trường của các quan hệ ITEM và SUPPLIER_ITEM có thể được gộp vào trong quan hệ hình thành từ liên kết nhiều-nhiều nhằm tránh việc phải kết nối 3 quan hệ. Chú ý rằng, chính việc kết hợp này cũng tạo ra sự lưu trữ dư thừa dữ liệu, như trong ví dụ này là các giá trị nằm trong trường Item_desc bị lặp lại trong mỗi bản ghi báo giá.



Ba quan hệ được chuẩn hoá



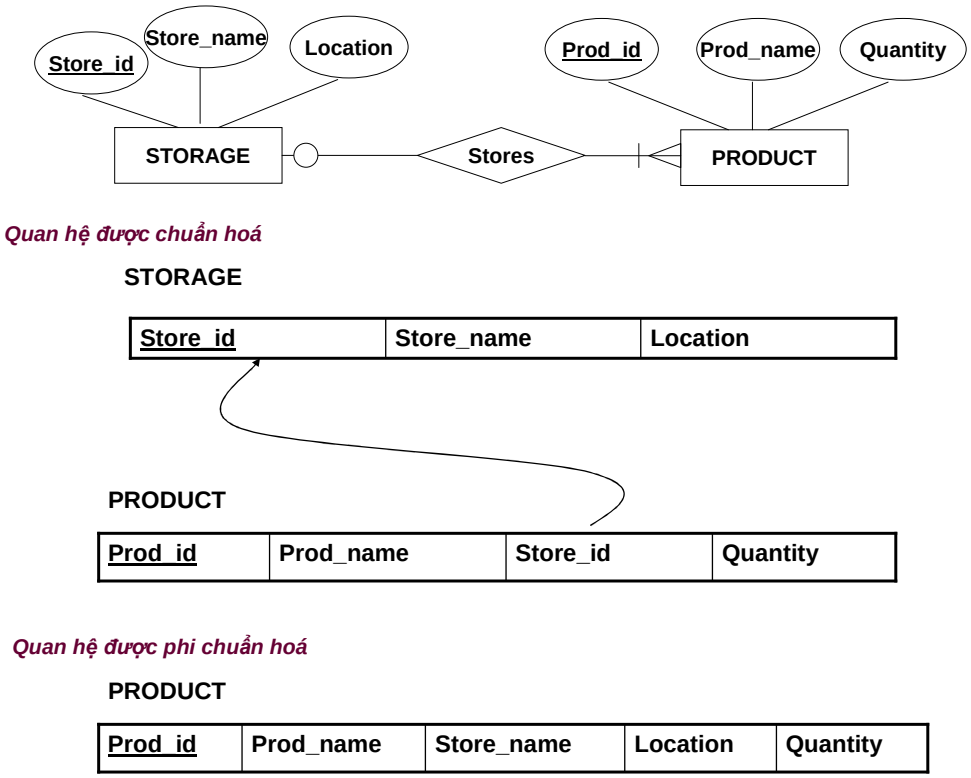
Quan hệ được phi chuẩn hoá



Hình 2.6.3.2 Phi chuẩn trong trường hợp liên kết nhiều - nhiều

- Trường hợp các quan hệ được hình thành từ liên kết một-nhiều và kiểu thực thể phía liên kết đầu một không tham gia vào bất cứ một liên kết nào khác. Khi đó chúng ta có thể cân nhắc đến việc kết hợp hai quan hệ hình thành từ hai kiểu thực thể này thành một quan hệ phi chuẩn như ví dụ sau:

Ví dụ: Giả thiết chúng ta có liên kết một-nhiều sau: mỗi một kho STORAGE sẽ được sử dụng để lưu một số mặt hàng cụ thể PRODUCT, ngược lại mỗi một mặt hàng chỉ được phép lưu trong một kho. Khi đó chúng ta có thể gộp hai quan hệ trên thành một quan hệ duy nhất là ITEM như trong hình 2.6.3.3 sau:



Hình 2.6.3.3 Phi chuẩn trong trường hợp liên kết một - nhiều

Trên đây là một số trường hợp mà chúng ta có thể gộp các quan hệ nhằm tránh việc phải thường xuyên kết nối giữa chúng. Ngược lại, phi chuẩn cũng có thể được sử dụng để tách các quan hệ bằng cách phân mảng một quan hệ thành nhiều bảng riêng biệt. Cả 2 phương pháp phân mảng theo chiều dọc và theo chiều ngang đều có thể được sử dụng.

- Phân mảng theo chiều ngang là chia một quan hệ thành nhiều vùng khác nhau có cùng cấu trúc các

trường. Chúng ta có thể hình dung công việc này như trong thư viện, ta đặt các sách khoa học vào một ngăn riêng biệt, các sách văn học vào một ngăn riêng biệt khác. Trong cơ sở dữ liệu cũng vậy, việc phân mảng theo chiều ngang tỏ ra khá hiệu quả khi các bản ghi trong quan hệ được xử lý riêng rẽ theo từng vùng. Ví dụ với quan hệ khách hàng CUSTOMER, chúng ta có thể chia quan hệ này thành 4 phân mảng khác nhau dựa trên các giá trị trong thuộc tính Region. Khi đó, các nghiệp vụ xử lý sẽ nhanh hơn do được tiến hành dựa trên các mảng khác nhau, bởi thực sự máy tính chỉ phải thao tác trên những tệp tách biệt nhau, số lượng các bản ghi cần thao tác được giảm đi đáng kể. Việc phân mảng theo chiều ngang cũng tỏ ra an toàn hơn, bởi bản chất chúng là những tệp riêng biệt, chúng ta có thể tiến hành các thao tác phân quyền đảm bảo an toàn mức tệp để giới hạn việc sử dụng đối với người sử dụng. Phương pháp này cũng giúp việc khôi phục dữ liệu dễ dàng hơn. Khi một phân mảng bị hỏng, chúng ta có thể tiến hành khôi phục riêng phân mảng đó mà không làm ảnh hưởng đến các phân mảng khác cũng như toàn bộ quan hệ. Ngay cả khi một phân mảng bị hỏng thì các phân mảng khác vẫn có thể hoạt động thay vì cả quan hệ sẽ bị hỏng. Cuối cùng, mỗi một phân mảng có thể được lưu trữ trên những ổ cứng khác nhau, điều này giúp cho việc tăng đáng kể tốc độ xử lý dữ liệu trong quan hệ.

Cũng cần phải lưu ý rằng, việc phân mảng theo chiều ngang gần tương tự như việc tạo các liên kết kiểu “là - một” bởi các thực thể khác nhau sẽ tham gia vào những liên

kết khác nhau, do vậy chúng cũng sẽ tham gia vào các quá trình xử lý khác nhau.

- Phân mảng theo chiều dọc chia các cột của một quan hệ thành những tệp khác nhau và thuộc tính tham gia vào làm khoá chính sẽ được lưu trữ ở tất cả các tệp được tách ra đó. Ví dụ trong một quan hệ chúng ta có thể tách các cột số liệu liên quan đến kế toán thành một bảng riêng, các số liệu liên quan đến kỹ thuật thành một bảng riêng và các số liệu liên quan đến bán hàng thành một bảng riêng. Khi các số liệu này cần được xử lý cùng nhau thì những bảng này có thể dễ dàng kết hợp lại (do có cùng chung các trường khoá chính), do vậy nó được xử lý như một quan hệ.

Trong một số trường hợp, chúng ta cũng có thể kết hợp cả 2 phương pháp phân mảng trên cho một quan hệ, điều này thường được áp dụng khi mà cơ sở dữ liệu khá lớn và được lưu trên những máy tính khác nhau.

Dưới đây là các ưu, nhược điểm của các phương pháp phân mảng và tạo bản sao dữ liệu:

Ưu điểm:

- Tính hiệu quả: Dữ liệu được xử lý cùng nhau (có chung một đặc điểm) sẽ được sắp xếp vật lý gần nhau, ngược lại dữ liệu được xử lý khác nhau sẽ được lưu ở những vị trí cách nhau.
- Tối ưu hoá cục bộ: Mỗi một phân mảng của dữ liệu có thể được lưu trữ khác nhau nhằm tối ưu hoá việc xử lý dựa trên đặc trưng sử dụng của chúng.
- Tính an toàn: Khi sử dụng phân mảng, mỗi một người hoặc nhóm người dễ dàng được phân các

quyền thao tác, sử dụng khác nhau trên mỗi phân mảng đó, do vậy trong cùng một quan hệ, với phân mảng này, người sử dụng A có thể cập nhật, sửa đổi dữ liệu, trong khi với các phân mảng khác anh ta lại không được phép sửa đổi thậm chí được phép nhìn thấy.

- Tính dễ khôi phục: Khi chia thành các phân mảng, mỗi một phân mảng sẽ có kích thước nhỏ hơn rất nhiều so với cả quan hệ, do vậy khôi phục lại một phân mảng sẽ nhanh hơn. Và khi một phân mảng bị hỏng, thì các phân mảng khác vẫn hoạt động được, do vậy các lỗi xảy ra đối với mỗi phân mảng sẽ được cô lập.
- Tính cân bằng tải: Các phân mảng do bản chất là những tệp riêng biệt do vậy có thể được lưu trữ trên những vùng khác nhau (các ổ cứng hoặc các thiết bị lưu trữ) và làm tăng tốc độ xử lý do chúng có thể được thực hiện song song.

Nhược điểm:

- Tốc độ truy nhập không nhất quán: Các phân mảng khác nhau sẽ được truy nhập với các tốc độ khác nhau, do vậy dễ làm cho người sử dụng hiểu lầm. Cũng như vậy, khi có các thao tác đòi hỏi kết hợp các phân mảng lại sẽ làm cho tốc độ xử lý giảm đi đáng kể.
- Phức tạp: Việc phân chia thành các phân mảng có thể khiến người lập trình phải tốn công sức viết chương trình xử lý hơn, bởi bản chất các phân mảng là những tệp vật lý khác nhau.

- Dị thường: Các dị thường trong các thao tác thêm mới, sửa đổi, xoá bỏ có thể xảy ra và do vậy cần phải cẩn thận để tránh những trường hợp này.
- Cần nhiều không gian lưu trữ hơn và thời gian cập nhât: Dữ liệu có thể được lưu trữ trùng lặp trên các phân mảng, do vậy sẽ cần nhiều không gian lưu trữ hơn, khi cập nhật cũng phải cập nhật ở tất cả những nơi lưu trữ trùng lặp đó.

6.4. Tổ chức các tệp vật lý

Ngày nay, hầu hết các hệ quản trị cơ sở dữ liệu đều quản lý các tệp dữ liệu tập trung trong một cơ sở dữ liệu, do vậy trong mục này chúng tôi sẽ không trình bày cách thức lưu trữ các bản ghi vật lý trong mỗi tệp. Mục đích chính của mục này là giúp chúng ta hiểu được các cách tổ chức tệp mà các hệ quản trị cơ sở dữ liệu ngày nay thường dùng, trên cơ sở đó có thể lựa chọn các hình thức tệp phù hợp với các quan hệ đã được xây dựng.

Một tệp vật lý là một vị trí được đặt tên trong bộ nhớ thứ cấp được cấp phát cho mục đích lưu trữ các bản ghi vật lý. Một số hệ điều hành cho phép cắt một tệp vật lý thành những đoạn riêng biệt, tuy nhiên nó vẫn là trong suốt đối với người thiết kế cơ sở dữ liệu.

Tổ chức tệp là lựa chọn kỹ thuật sắp xếp vật lý các bản ghi của một tệp dữ liệu trên thiết bị lưu trữ thứ cấp. Khi lựa chọn tổ chức lưu trữ cho một tệp dữ liệu cụ thể trong cơ sở dữ liệu, có 7 yếu tố thường được xem xét đến, đó là:

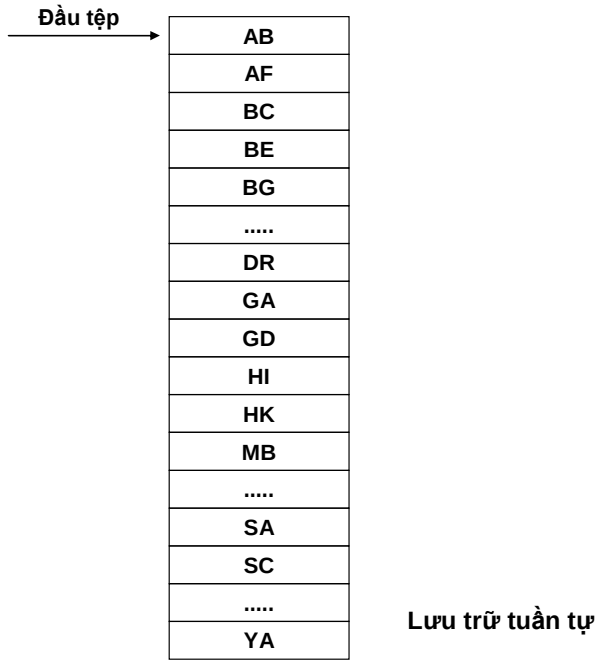
- Nâng cao tốc độ trả về dữ liệu.

- Xử lý nhanh các dữ liệu đưa vào cũng như các giao tác
- Sử dụng hiệu quả không gian lưu trữ
- Bảo vệ dữ liệu khỏi bị mất mát và lỗi
- Giảm thiểu việc phải tái tổ chức lại
- Dự trù được việc cấp phát không gian lưu trữ tiếp theo
- Đảm bảo an toàn đối với các sử dụng bất hợp pháp

Dưới đây là một số cách tổ chức tệp phổ biến trong các hệ quản trị cơ sở dữ liệu:

a. Tổ chức tệp tuần tự

Trong một tệp được tổ chức theo cơ chế tuần tự, các bản ghi trong cùng một tệp được lưu trữ tuần tự theo các giá trị của trường khoá chính. Để cấp phát vị trí cho một bản ghi, hệ quản trị cơ sở dữ liệu phải quét từ đầu tệp để tìm ra vị trí phù hợp cho bản ghi đó. Các tệp này thường được sử dụng khi bảng dữ liệu có khoá chính là các giá trị tăng dần và các bản ghi được đưa vào bảng theo đúng thứ tự tăng dần đó.



Hình 2.6.4.1 Tổ chức tệp tuần tự

b. Tổ chức tệp chỉ mục

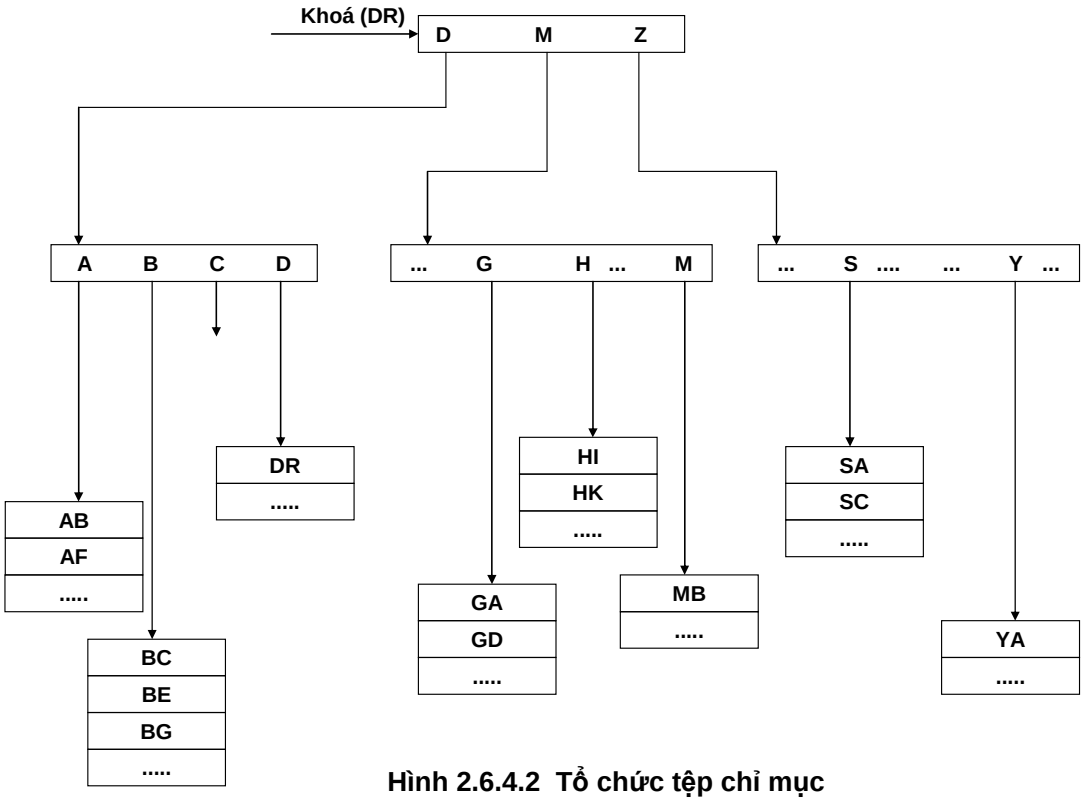
Trong phương pháp tổ chức tệp này, các bản ghi có thể được lưu trữ tuần tự hoặc không tuần tự và khi một bản ghi được tạo ra thì đi kèm với nó là một chỉ mục cũng được tạo ra tương ứng nhằm tạo ra một cơ chế để tìm kiếm nhanh chóng đến bản ghi đó. Cách tổ chức này giống cách tổ chức các ô phích tìm kiếm sách trong thư viện, thay vì độc giả phải tìm kiếm trên những tủ sách đồ sộ, họ có thể tìm kiếm trên các phiếu được sắp xếp có thứ tự trong các ô phích, trong mỗi phiếu đều lưu các giá trị mà độc giả có thể tìm kiếm cùng vị trí lưu trữ của quyển sách tương ứng. Tuy nhiên, trong máy tính có thể hơi khác, bởi mỗi một chỉ mục có thể được ánh xạ tới một giá trị khoá của một hoặc một số bản ghi. Một chỉ mục có thể trỏ tới duy nhất một bản ghi (như trong trường hợp sử dụng khoá chính làm chỉ mục) hoặc chỉ tới một số các bản ghi khi chúng có chung một giá

trị nào đó, ta gọi chỉ mục này là chỉ mục khoá thứ cấp. Các chỉ mục khoá thứ cấp đóng vai trò quan trọng trong việc hỗ trợ trong các truy vấn kiểu ad hoc đòi hỏi tốc độ cũng như trong các yêu cầu tạo báo cáo.

Một ví dụ về loại chỉ mục này đó là chỉ mục trên trường Order_date của quan hệ ORDER. Rõ ràng sẽ có một tập các bản ghi có chung giá trị trong trường Order_date bởi một ngày có thể có rất nhiều đơn hàng được đặt, và do vậy khi Order_date được sử dụng làm chỉ mục chúng ta sẽ dễ dàng tìm kiếm đến những đơn hàng được tạo ra trong một khoảng thời gian đưa vào.

Một số hệ quản trị cơ sở dữ liệu cũng hỗ trợ khả năng việc sử dụng chỉ mục trên các chỉ mục để tạo ra một phân cấp các chỉ mục. Bởi chúng ta biết rằng, một chỉ mục bản chất của nó cũng là một tệp dữ liệu, do vậy nếu nó quá lớn thì cũng có thể tổ chức theo phương pháp chỉ mục phân cấp này, tức là tạo chỉ mục của chỉ mục. Như trong hình 2.6.4.2, mỗi một chỉ mục có một giá trị khoá và một con trỏ trỏ tới một chỉ mục khác hoặc trỏ thẳng tới bản ghi dữ liệu. Ví dụ, để tìm bản ghi có khoá là ‘DR’, cấu trúc tệp được bắt đầu từ chỉ mục trên cùng và lấy con trỏ sau giá trị D, con trỏ này trỏ tới một chỉ mục khác với tất cả các khoá bắt đầu bằng chữ A đến chữ D theo thứ tự từ điển. Tiếp theo, chương trình tìm kiếm lại sử dụng tiếp con trỏ sau giá trị D trong chỉ mục này, con trỏ này trỏ đến một dãy các bản ghi có trường khoá bắt đầu bằng chữ D. Cuối cùng, việc tìm kiếm được tiến hành trên dãy các chỉ mục này để có thể chỉ ra được bản ghi có giá trị khóa là ‘DR’ hay là không có bản ghi đó.

Cấu trúc chỉ mục trong hình 2.6.4.2 được gọi là cấu trúc chỉ mục tuần tự. Trong cấu trúc này, các bản ghi chỉ mục được lưu trữ tuần tự theo giá trị của trường khoá sử dụng làm chỉ mục. Trong hình 2.6.4.2, các bản ghi chỉ mục được lưu trữ tuần tự trong mỗi khối dữ liệu, mỗi khối dữ liệu này có thể không được cấp phát gần nhau, nhưng cấu trúc chỉ mục tuần tự này có thể được sử dụng để lưu trữ các bản ghi dữ liệu của một tệp tuần tự. Một cách tổ chức khác đối với các tệp chỉ mục tuần tự này đó là sử dụng con trỏ liên kết giữa các khối, tức là thay vì được tổ chức phân cấp như trên, phần cuối mỗi khối đều có thêm một con trỏ để trỏ đến vị trí bắt đầu của khối tiếp theo. Ví dụ như trong hình 2.6.4.2, ở các khối trong mức thứ 2 chúng ta sẽ tổ chức thêm một con trỏ để trỏ sang khối tiếp theo. Loại tổ chức tệp được chỉ mục này được sử dụng với cả các tệp dữ liệu được tổ chức tuần tự hoặc không tuần tự (các bản ghi dữ liệu được lưu trữ tại những vị trí bất kỳ trong thiết bị lưu trữ thứ cấp). So với các tệp chỉ mục không tuần tự, các tệp chỉ mục tuần tự cho phép việc sử dụng không gian lưu trữ hiệu quả hơn và việc xử lý dữ liệu tuần tự rõ ràng có tốc độ nhanh hơn.



Hình 2.6.4.2 Tổ chức tệp chỉ mục

Một trong những thế mạnh của cách tổ chức tệp chỉ mục đó là khả năng có thể tạo ra các đa chỉ mục. Như trong thư viện, các đa chỉ mục có thể xây dựng là các chỉ mục dựa trên tiêu đề, dựa trên tên tác giả và thậm chí dựa trên chủ đề của cuốn sách. Do vậy, chúng ta có thể tạo ra một tệp chỉ mục khác trên cùng một tập bản ghi dữ liệu.

Ví dụ, nếu chúng ta sử dụng chỉ mục Sup_id để tìm ra địa chỉ của các bản ghi với giá trị mã nhà cung cấp bằng "CTY005", và sử dụng chỉ mục Unit_price để tìm ra những bản ghi có giá trị tại trường đơn giá nhỏ hơn 500$, sau đó có thể sử dụng giao của 2 tập bản ghi này để tìm ra những bản ghi thoả mãn cả 2 điều kiện trên.

Khả năng sử dụng nhiều chỉ mục trong cùng một truy vấn đóng vai trò quan trọng trong việc thực hiện các truy vấn

trong cơ sở dữ liệu quan hệ. Các phép toán logic AND, OR và NOT dễ dàng được thực hiện thông qua việc sử dụng các chỉ mục.

Cấu trúc tổng quát của một chỉ mục phân cấp như trong hình 2.6.4.2 được gọi là cây (có một nút trên cùng được gọi là gốc và các nút lá ở dưới cùng). Việc xử lý các nút trên cây thông qua các tính chất của cây dựa trên các cấu trúc khác nhau, do vậy việc xử lý cũng phụ thuộc vào từng cấu trúc cây đó. Tuy nhiên, hầu hết các hệ quản trị cơ sở dữ liệu đều xử dụng một cấu trúc cây đặc biệt được gọi là cây cân bằng (B-tree). Trong cấu trúc cây B-tree này, tất cả các nút lá (các bản ghi dữ liệu và các con trỏ tương ứng) đều có cùng khoảng cách đến nút gốc. Trong hình 2.6.4.2, tất cả các nút lá đều cách nút gốc 2 mức.

Tuy nhiên, như đã trình bày ở trên, mặc dù chúng ta là người thiết kế cơ sở dữ liệu mức vật lý, chúng ta cũng không cần phải can thiệp vào cấu trúc cũng như các thuật toán xử lý trên các cấu trúc cây này, bởi chúng đã được quản lý bởi hệ điều hành cũng như hệ quản trị cơ sở dữ liệu.

Ngoài ra, ngày nay phổ biến một loại chỉ mục mới, đó là chỉ mục bitmap. Trong cấu trúc này, mỗi một giá trị khoá được gán một bitmap. Ví dụ, giả sử trong quan hệ PRODUCT chúng ta có 10 bản ghi và trong thuộc tính Price chúng ta có 4 giá trị phân biệt trong 10 bản ghi đó. Một chỉ mục bitmap trên trường Price này sẽ có 4 bản ghi với 10bits cho mỗi bản ghi đó. Mỗi một bit trong bản ghi bitmap đó tương ứng với một hàng trong quan hệ Product. Giả sử, nếu bit thứ 8 tương ứng với Price có giá bằng 400$, thì điều đó cũng có nghĩa rằng bản ghi thứ 8 trong quan hệ

Product có giá trị bằng 400$ (trong mỗi một cột bit sẽ chỉ có một bit được mang giá trị 1).

Cấu trúc chỉ mục bitmap này rất thích hợp cho những trường có giá trị khác nhau ít, như các trường về giới tính, màu sắc. Một ưu điểm nữa đó là cấu trúc chỉ mục kiểu bitmap đòi hỏi rất ít không gian lưu trữ (thường khoảng bằng 25%) so với các chỉ mục theo cấu trúc cây thông thường. Sở dĩ như vậy bởi một đặc trưng cơ bản của chỉ mục bitmap đó là một bitmap có thể được sử dụng cho nhiều giá trị khoá. Ví dụ, chúng ta có thể thêm một tập các hàng vào cuối bảng trong hình 2.6.4.3 cho tất cả các giá trị của thuộc tính Room. Khi đó, mỗi cột bit có thể có 2 giá trị bằng 1, tương ứng với các giá trị cho Price và Room. Nếu một truy vấn muốn tìm kiếm các sản phẩm có giá bằng 100$ và phòng là phòng ăn có thể kết hợp giao của các bit của hàng Price 100$ và của Room dinner để tìm ra các bản ghi phù hợp với điều kiện đó. Đặc biệt thao tác với các bit tỏ ra vô cùng hiệu quả.

Price	1	2	3	4	5	6	7	8	9	10
100	0	0	1	0	1	0	0	0	0	0
200	1	0	0	0	0	0	0	0	0	0
300	0	1	0	0	0	0	0	0	0	0
400	0	0	0	1	0	1	0	1	1	0

Các sản phẩm số 3 và 5 có giá là 100 USD
Sản phẩm 1 có giá là 200 USD
Các sản phẩm 2,7 và 10 có giá là 300 USD
Các sản phẩm 4,6,8 và 9 có giá là 400 USD

Hình 2.6.4.3 Chỉ mục bitmap trên thuộc tính Price của quan hệ PRODUCT

c. Tổ chức các tệp băm

Trong kiểu tổ chức này, địa chỉ của mỗi bản ghi được xác định thông qua một thuật toán băm, đó là một thủ tục chuyển đổi giá trị của khoá chính thành địa chỉ của bản ghi.

Một thuật toán băm điển hình đó là sử dụng kỹ thuật chia giá trị khoá chính cho một số nguyên tố và sau đó sử dụng phần dư của phép chia này để gán cho vị trí lưu trữ tương đối. Ví dụ, giả sử chúng ta phải tổ chức cho một tệp lưu trữ khoảng 1000 bản ghi là nhân viên của một công ty, khi đó số nguyên tố phù hợp nhất đó là 997, bởi nó gần 1000 nhất. Giả sử ta cần tìm vị trí của bản ghi có giá trị khoá chính là 12396. Khi đó chúng ta chia giá trị này cho 997 và được số dư là 432. Do vậy bản ghi của nhân viên này sẽ được lưu trữ tại vị trí 432 trong tệp. Tuy nhiên, như chúng ta thấy, dù sao thì cũng tồn tại trường hợp có 2 bản ghi khi chia cho 997 cúng có số dư là 432, ví dụ nhân viên có giá trị khoá là 10429, khi đó phải tồn tại những thuật toán băm tiếp để xử lý những trường hợp xung đột này. Trên thực tế, phương pháp tổ chức tệp băm cũng tỏ ra khá có hiệu quả.

d. Tổ chức các tệp được phân cụm (clustering)

Một số hệ quản trị cơ sở dữ liệu cho phép lưu trữ các bản ghi của những tệp khác nhau liền kề nhau trong không gian bộ nhớ thứ cấp. Khi dó, một vùng đĩa liên tiếp không chứa các bản ghi có cùng cấu trúc và nó hình thành các phân cụm (hay phân nhóm). Ví dụ, trong Oracle, các hàng của một, hai hoặc nhiều hơn của các bảng liên quan với nhau được lưu trữ trong cùng một vùng lưu trữ trong đĩa cứng nếu chúng thường xuyên được kết nối với nhau. Như vậy chúng ta thấy rằng một phân nhóm được xác định bởi các

bảng với các cột mà giữa các bảng đó thường xuyên có sự kết nối với nhau.

Ví dụ bảng Customer và bảng Customer_Order thường xuyên được kết nối thông qua giá trị của trường Customer_ID, hoặc các hàng của bảng Price_Quote (lưu trữ báo giá của các mặt hàng của các nhà cung cấp khác nhau) có thể được phân nhóm với nhau thông qua trường Item_ID.

Việc phân nhóm giảm thời gian truy nhập đến các bản ghi liên quan với nhau so với các phương pháp cấp phát thông thường khác (các tệp khác nhau được lưu trữ ở những vị trí khác nhau).

Các lệnh định nghĩa dưới đây của cơ sở dữ liệu Oracle sẽ cho thấy các phân nhóm được định nghĩa như thế nào và các bảng được gán vào trong các phân nhóm đó ra sao. Đầu tiên, phân nhóm được xác định:

```
CREATE CLUSTER ORDERING (CLUSTERKEY CHAR(25))
```

ORDERING là tên được đặt cho vùng phân nhóm, thuật ngữ CLUSTERKEY là bắt buộc, tuy nhiên nó sẽ không được sử dụng lại.

Sau đó các bảng được gán vào phân nhóm trên như sau:

```
CREATE TABLE CUSTOMER(
Cust_id    VARCHAR2(10) NOT NULL,
Cust_name      VARCHAR2(50) NOT NULL,
Cust_address  VARCHAR2(200))
CLUSTER ORDERING (Cust_id)
```

```
CREATE TABLE ORDER(
Order_id  VARCHAR2(10) NOT NULL
Cust_id   VARCHAR2(25) NOT NULL,
Order_date      DATE)
CLUSTER ORDERING (Cust_id)
```

Truy nhập đến các bản ghi trong phân nhóm được thông qua một chỉ mục trên khoá phân nhóm hay thông qua các hàm băm trên khoá phân nhóm. Các bản ghi phân nhóm được sử dụng hiệu quả nhất khi chúng ít bị thay đổi. Ngược lại, khi các bản ghi liên tục được thêm vào, sửa đổi, xoá bỏ thì không gian nhớ rác rất dễ sinh ra và do vậy khó khăn trong việc định vị các bản ghi liên kết với nhau. Tuy nhiên, kỹ thuật phân nhóm cũng vẫn là một lựa chọn cho người thiết kế các tệp vật lý nhằm nâng cao hiệu năng sử dụng các bảng.

6.5. Lựa chọn và sử dụng các loại chỉ mục

Việc xây dựng các chỉ mục đóng vai trò quan trọng trong quá trình thiết kế cơ sở dữ liệu mức vật lý. Các chỉ mục được sử dụng phải dựa trên các truy vấn (bao gồm cả những câu lệnh cập nhật dữ liệu vì thực sự chúng cũng là những câu lệnh truy vấn). Các thuộc tính được lựa chọn làm chỉ mục phải dựa trên các điều kiện trong mệnh đề where của câu lệnh truy vấn. Với những chỉ mục hợp lý, chúng sẽ làm tăng đáng kể tốc độ của câu lệnh truy vấn dữ liệu.

Tuy nhiên, khi xây dựng các chỉ mục cho một quan hệ dữ liệu, chúng ta cũng cần phải cân nhắc đến sự ảnh hưởng

của chúng đến tốc độ cập nhật dữ liệu trên quan hệ đó. Mặc dù một chỉ mục có thể làm tăng tốc độ truy vấn, nhưng khi cập nhật một giá trị trong các thuộc tính được sử dụng làm chỉ mục đó, hệ thống sẽ phải cập nhật lại cho tất cả các chỉ mục đã sử dụng thuộc tính đó, do vậy nó sẽ làm chậm quá trình cập nhật dữ liệu.

Như vậy, việc lựa chọn được một tập chỉ mục đòi hỏi sự hiểu biết thấu đáo về các kỹ thuật chỉ mục cũng như cơ chế tối ưu hóa truy vấn. Dưới đây là một số nguyên tắc giúp chúng ta có thể lựa chọn khi xây dựng các chỉ mục:

- Nguyên tắc 1 (khi nào thì xây dựng chỉ mục): Chúng ta không nên xây dựng một chỉ mục mà không có một câu lệnh truy vấn dữ liệu nào sử dụng chúng. Bất cứ khi nào có thể, chúng ta hãy lựa chọn những chỉ mục có thể làm tăng ít nhất nhiều hơn một câu lệnh truy vấn. Với những câu lệnh thường xuyên được thực hiện trên quan hệ có số lượng bản ghi lớn hầu như nhất thiết chúng ta phải tạo các chỉ mục cho chúng.
- Nguyên tắc 2 (lựa chọn khoá tìm kiếm): Những thuộc tính xuất hiện trong mệnh đề where sẽ là những thuộc tính ứng cử cho việc xây dựng các chỉ mục. Đối với những điều kiện lựa chọn giá trị chính xác, chúng ta nên cân nhắc đến những chỉ mục kiểu băm cho những thuộc tính xuất hiện trong các đối sánh đó. Còn đối với những điều kiện lựa chọn giá trị trong khoảng, chúng ta có thể lựa chọn chỉ mục loại cây cân bằng B-tree trên những thuộc tính đó. Loại chỉ mục này rất phù hợp trong trường hợp các quan hệ được chỉ mục ít bị cập nhật.

- Nguyên tắc 3 (khoá tìm kiếm đa thuộc tính): Chúng ta lựa chọn chỉ mục được xây dựng trên nhiều thuộc tính khi trong mệnh đề where xuất hiện so sánh trên nhiều thuộc tính. Khi xây dựng các chỉ mục trên các thuộc tính khoá tìm kiếm này, chúng ta nên cân nhắc thứ tự liệt kê các điều kiện so sánh, bởi chúng cũng ảnh hưởng đến việc phân tích và sử dụng chỉ mục.
- Nguyên tắc 4 (khi nào thì sử dụng phân nhóm - cluster): chú ý rằng đối với mỗi một quan hệ chỉ có nhiều nhất một chỉ mục kiểu phân nhóm, và việc sử dụng cơ chế phân nhóm sẽ làm ảnh hưởng đáng kể đến tốc độ xử lý trên quan hệ, do vậy việc lựa chọn chỉ mục loại phân nhóm cần phải được xem xét kỹ lưỡng. Thông thường các truy vấn dựa trên giá trị khoảng thường được hưởng lợi nhiều nhất từ cơ chế phân nhóm.
- Nguyên tắc 5 (lựa chọn chỉ mục băm hay cây cân bằng): Một chỉ mục B-tree thường được sử dụng nhiều hơn bởi nó phù hợp cho cả những truy vấn kiểu giá trị nằm trong một khoảng nào đó và cả những truy vấn có điều kiện bằng chính xác một giá trị cụ thể nào đó. Chỉ mục kiểu băm tỏ ra phù hợp hơn trong những trường hợp sau: Quan hệ được chỉ mục là quan hệ trong, và khoá tìm kiếm nằm trong điều kiện kết nối giữa các quan hệ. Hoặc trong các truy vấn sử dụng điều kiện bằng một giá trị chính xác nào đó.
- Nguyên tắc 6 (cân đối chi phí cho việc bảo trì chỉ mục): Sau khi xây dựng xong một chỉ mục, chúng ta cần phải xem xét đến sự ảnh hưởng của chúng đến

các công đoạn cập nhật dữ liệu: Nếu vấn đề bảo trì các chỉ mục làm chậm các phép cập nhật, hơn nữa những phép này lại thường xuyên được thực hiện thì khi đó chỉ mục tỏ ra không hiệu quả và chúng ta nên cân nhắc đến khả năng loại bỏ chúng.

Dưới đây ta hãy xét đến một ví dụ cụ thể, ví dụ này sẽ minh hoạ cho chúng ta nên lựa chọn chỉ mục như thế nào trong quá trình thiết kế cơ sở dữ liệu vật lý: Giả thiết, chúng ta có 2 lược đồ quan hệ gồm các trường như sau:

EMPLOYEE (Eno, Ename, Age, Sal, Hobby)

DEPARTMENT (Dno, Dname, Mgr)

Chúng ta hãy xem xét đến một câu lệnh truy vấn đơn giản sau:

```
Select   E.Ename,
         D.Mgr
From         EMPLOYEE E,
         DEPARTMENT D
Where    D.Dname = 'Toy' and
         E.Dno = D.Dno
```

Đầu tiên chúng ta hãy xem xét kế hoạch đánh giá cho truy vấn trên sẽ được thực hiện như thế nào. Đầu tiên, những bộ nào thoả mãn điều kiện D.Dname = 'Toy' được thực hiện, tiếp theo, trong số những bộ đó, sẽ được so sánh qua điều kiện thứ hai E.Dno = D.Dno để lựa chọn ra tiếp những bộ thoả mãn.

Chúng ta thấy cả 2 điều kiện trong mệnh đề where đều là điệu kiện bằng. Do vậy chúng ta sẽ phải nghĩ ngay đến việc

sử dụng chỉ mục kiểu băm trên những thuộc tính này. Điều kiện D.Dname = 'Toy' cho chúng ta biết nên xây dựng một chỉ mục cho trường Dname của quan hệ DEPARTMENT. Còn đối với điều kiện E.Dno = D.Dno, chúng ta đứng trước lựa chọn: nên xây dựng chỉ mục cho thuộc tính Dno của quan hệ EMPLOYEE hay thuộc tính Dno của quan hệ DEPARTMENT hay thậm chí cho cả 2 quan hệ đó? Chúng ta thấy rằng, với việc sử dụng chỉ mục trên thuộc tính Dname, chỉ có một số ít các bộ được trả về thỏa mãn điều kiện Dname = 'Toy', với những bộ này, chúng ta cần tìm ra những bộ tương ứng bên quan hệ EMPLOYEE thoả mãn điều kiện thứ 2 E.Dno = D.Dno, do vậy chúng ta nên xây dựng một chỉ mục trên trường Dno của quan hệ EMPLOYEE. Cũng cần phải lưu ý rằng, nếu chúng ta tiếp tục xây dựng một chỉ mục khác trên trường Dno của quan hệ DEPARTMENT thì cũng không làm tăng tốc độ của kết quả trả về, bởi những bộ thoả mãn điều kiện tìm kiếm trên trong quan hệ DEPARTMENT đã được trả về bằng cách sử dụng chỉ mục của trường Dname.

Như vậy các phương án đánh giá truy vấn đã giúp chúng ta có thể xác định được việc lựa chọn các chỉ mục. Đây là công việc mà chúng ta cần phải lưu tâm trong quá trình thiết kế cơ sở dữ liệu mức vật lý.

Tiếp theo, chúng ta sửa đổi câu lệnh trên đi một chút, giả sử mệnh đề where bây giờ sẽ là:

```
Where   D.Dname = 'Toy' and
        E.Dno = D.Dno and
        E.Age = 25.
```

Chúng ta hãy xem xét phương án đánh giá cho câu truy vấn trên. Một phương án tốt, đó là trả về những bộ trong quan hệ DEPARTMENT thoả mãn điều kiện trên trường Dname và đối sánh với những bộ trong quan hệ EMPLOYEE thông qua chỉ mục trên trường Dno, sau đó việc lựa chọn những bộ thoả mãn điều kiện trên trường Age mới được áp dụng trên những bộ này. Trong truờng hợp này, chúng ta thực sự không cần đến chỉ mục trên trường Dno của quan hệ EMPLOYEE nếu chúng ta đã có một chỉ mục trên trường Age. Những bộ trong quan hệ DEPARTMENT sẽ được trả về thông qua việc sử dụng chỉ mục trên trường Dname và những bộ trong quan hệ EMPLOYEE sẽ được trả về thông qua việc sử dụng chỉ mục trên trường Age sẽ được kết nối với nhau. Vì số lượng những bộ thoả mãn 2 điều kiện này không nhiều, nên vấn đề đối sánh giữa chúng không chiếm nhiều bộ nhớ cũng như thời gian của máy tính. Với phương án đánh giá này, chúng ta nhận thấy rằng nó cũng đạt được mục đích của câu lệnh truy vấn, tuy nhiên nó có vẻ không được hợp lý lắm nếu xét trên khía cạnh thực tế thiết kế thì sử dụng chỉ mục trên trường Dno tỏ ra hợp lý hơn, bởi đây là khoá ngoại lai được lấy từ quan hệ DEPARTMENT sang quan hệ EMPLOYEE, và nó có thể sẽ được sử dụng thêm ở những quan hệ khác khi liên kết với quan hệ EMPLOYEE.

Tiếp theo chúng ta hãy xem xét đến các câu lệnh truy vấn có điều kiện tìm kiếm dựa trên giá trị khoảng, bằng cách sửa đổi câu truy vấn trên như sau:

```
Select    E.Ename,
          D.Dname
From      EMPLOYEE E,
```

```
                DEPARTMENT D
Where           E.Sal between 10.000 and 20.000 and
                E.Hobby = 'Stamps' and
                E.Dno = D.Dno
```

Câu truy vấn này đã sử dụng một điều kiện tìm kiếm trên giá trị khoảng thông qua toán tử Between. Nó cũng tương đương với điều kiện:

```
10.00 <= E.Sal and E.Sal <= 20.000
```

Tuy nhiên, chúng ta nên sử dụng toán tử Beween bởi nó dễ hiểu và bộ tối ưu hoá truy vấn dễ phân tích hơn.

Quay trở lại câu lệnh truy vấn trên, cả 2 điều kiện E.Sal between 10.000 and 20.000 và E.Hobby = 'Stamps' đều dựa trên quan hệ EMPLOYEE. Do vậy, một kế hoạch đánh giá tốt sẽ là sử dụng quan hệ EMPLOYEE làm quan hệ ngoài và quan hệ DEPARTMENT làm quan hệ trong và chúng ta nên xây dựng một chỉ mục trên thuộc tính Dno của quan hệ DEPARTMENT. Còn đối với quan hệ EMPLOYEE, chúng ta sẽ lựa chọn chỉ mục nào để xây dựng? Một chỉ mục kiểu B-tree trên thuộc tính Sal sẽ là hợp lý, đặc biết nếu nó được phân cụm. Và một chỉ mục kiểu băm trên thuộc tính Hobby cũng giúp ích cho điều kiện tìm kiếm lựa chọn bằng. Nếu một trong 2 chỉ mục này đã có thì những bộ trong quan hệ EMPLOYEE sẽ được trả về thông qua chỉ mục đó, còn nếu có cả 2 chỉ mục trên thì bộ tối ưu hoá truy vấn sẽ lựa chọn chỉ mục sẽ trả về số lượng bộ thoả mãn ít hơn (hoặc điều kiện khoảng dựa trên Sal hoặc điều kiện bằng dựa trên Hobby), và điều này phụ thuộc vào dữ liệu trong quan hệ. Nếu chỉ có một số ít nhân viên có mức lương nằm trong khoảng tìm kiếm trên và có

nhiều nhân viên có thú vui chơi tem thì chỉ mục B-tree sẽ được lựa chọn, ngược lại chỉ mục băm trên thuộc tính Hobby sẽ được lựa chọn.

Đối với những hệ quản trị cơ sở dữ liệu lớn phổ biến hiện nay như Oracle hay SQL Server thường có các thống kê về dữ liệu trên các quan hệ được sử dụng thường xuyên, khi đó việc lựa chọn một trong 2 chỉ mục trên sẽ được dựa trên những thống kê này, ngược lại, nếu không có thống kê thì chỉ mục theo điều kiện bằng thường được lựa chọn trước, tất nhiên là đối với những truy vấn có giá trị lựa chọn tường minh như ví dụ trên. Còn đối với những điều kiện lựa chọn có giá trị không tường minh, thường dựa vào các biến truyền vào, hoặc các câu lệnh truy vấn động, thì bộ tối ưu hoá sẽ tự động phân tích để lựa chọn chỉ mục tại thời điểm chạy câu lệnh (run-ime).

Với những chỉ mục được phân cụm, những câu lệnh truy vấn dựa trên giá trị khoảng thường cho tốc độ trả về tốt hơn so với những chỉ mục không được phân cụm. Ví dụ như với truy vấn sau:

```
Select    E.Dno
From      EMPLOYEE E
Where     E.Age > 40
```

Trên thực tế, có khá nhiều truy vấn tìm kiếm trên nhiều trường khác nhau. Ví dụ, nếu chúng ta muốn trả về những bộ trong quan hệ EMPLOYEE sao cho có Age = 30 và Sal = 4000, vậy thì sử dụng một chỉ mục dựa trên khoá tìm kiếm <Sal,Age> (hoặc <Age,Sal>) sẽ là tốt hơn so với việc chỉ sử dụng một chỉ mục trên khoá tìm kiếm <Sal> hoặc một chỉ mục trên khoá tìm kiếm <Age>. Hay giả sử chúng

ta cùng có 2 chỉ mục dựa trên 2 khoá tìm kiếm Sal và Age độc lập nhau, việc sử dụng đồng thời cả 2 chỉ mục đó liệu có cho kết quả tốt hơn không? Chúng ta sẽ lựa chọn chỉ mục nào để xây dựng cho truy vấn này, cách tốt nhất đó là xây dựng chỉ mục hợp.

Chúng ta gọi chỉ mục trên các khoá tìm kiếm của nhiều thuộc tính là chỉ mục hợp. Ngoài việc hỗ trợ cho các truy vấn sử dụng điều kiện tìm kiếm bằng, chỉ mục hợp cũng có thể được sử dụng trong các truy vấn tìm kiếm giá trị khoảng.

Chúng ta hãy xem xét truy vấn sau:

```
Select    E.Eid
From          EMPLOYEE E
Where     E.Age between 20 and 30 and
          E.Sal between 3000 and 5000
```

Với một chỉ mục hợp trên các trường <Age,Sal> sẽ cho kết quả tìm kiếm khá tốt. Và đặc biệt tốt nếu chỉ mục có dạng B-tree và được phân cụm. Tuy nhiên, thứ tự của các thuộc tính tìm kiếm đôi khi cũng tạo ra sự khác biệt khá lớn, hãy xem tiếp truy vấn sau:

```
Select    E.Eid
From          EMPLOYEE E
Where     E.Age = 25 and E.
          E.Sal between 3000 and 5000
```

Trong câu lệnh truy vấn này, chỉ mục hợp theo thứ tự <Age,Sal> kiểu B-tree phân cụm sẽ có thời gian thực hiện khá ít bởi các bộ đầu tiên được sắp xếp theo các giá trị

trong trường Age, sau đó (nếu có nhiều hơn 1 bộ có cùng giá trị trong trường Age) các bộ sẽ được sắp xếp tiếp theo các giá trị trong trường Sal. Do vậy, tất cả các bộ với Age=25 được phân vào cùng một nhóm.

Ngược lại, nếu chúng ta sử dụng chỉ mục hợp theo thứ tự <Sal,Age> thì thời gian thực hiện truy vấn trên lại khác hẳn so với trường hợp đầu. Bởi trong trường hợp này, các bộ đầu tiên được sắp xếp theo Sal, và với 2 bản ghi có cùng một giá trị trong trường Age (như trong trường hợp này ta có Age=25) thì chúng cách nhau khá xa, do Sal của chúng ta là được lựa chọn theo giá trị khoảng.

Tuy nhiên việc sử dụng các chỉ mục hợp cũng cần phải được cân nhắc kỹ, bởi trong quá trình sử dụng cơ sở dữ liệu, khi dữ liệu ở các quan hệ được thay đổi thì các chỉ mục tương ứng cũng được thay đổi theo, và việc cập nhật, sửa đổi các chỉ mục hợp sẽ phức tạp hơn cũng như tốn kém hơp so với các chỉ mục đơn.

6.6. RAID

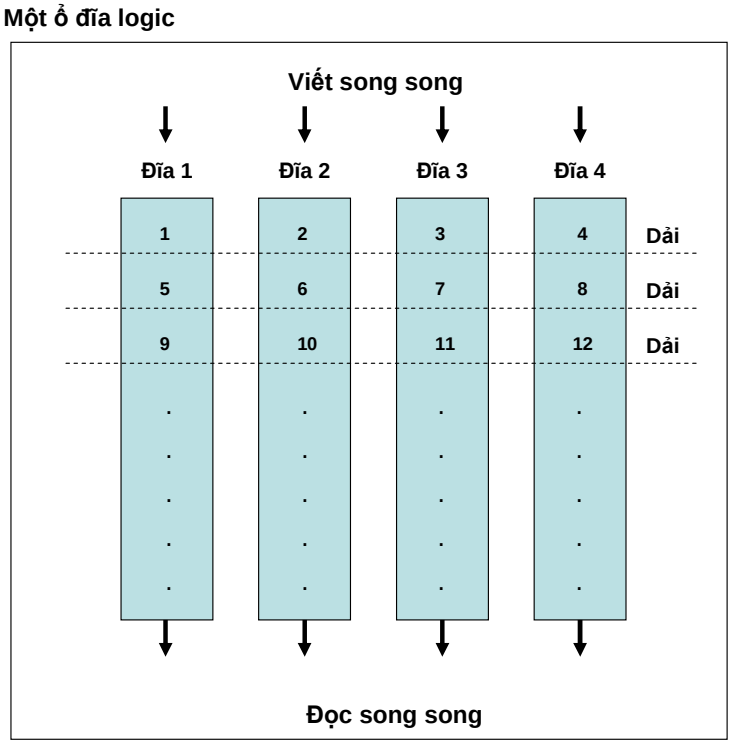
Như các phần trên chúng ta đã thấy việc phi chuẩn, tạo chỉ mục và phân cụm dữ liệu đã làm giảm thời gian truy xuất dữ liệu nếu dữ liệu được sắp xếp hợp lý. Với những bản ghi nhỏ, chúng ta có thể thiết kế để sắp xếp chúng liền nhau trong cùng một trang vật lý, tuy nhiên trên thực tế có rất nhiều những bản ghi lớn mà chúng ta phải sắp xếp chúng trên nhiều trang, do vậy các thao tác đọc đĩa tuần tự vào/ra chiếm khá nhiều thời gian để xử lý loại dữ liệu này. Do vậy các giải pháp phi chuẩn và phân cụm không đủ để giải quyết trong những trường hợp như thế này.

Hai xu hướng vẫn thường thấy trong công nghệ máy tính đó là giảm về giá và và giảm về kích thước. Chính hai xu hướng này đã cho phép người ta kết hợp nhiều thành phần của máy tính để có thể xử lý dữ liệu song song đặc biệt là với các thao tác vào ra của dữ liệu.

RAID, viết tắt của cụm từ Redundant Arrays of Inexpensive Disks, chúng ta có thể hiểu rằng đó là một tập ổ cứng được sử dụng đồng thời nhằm làm tăng tốc độ xử lý vào ra của dữ liệu, và cũng để đơn giản, chúng ta sử dụng cụm từ viết tắt này trong các phần tiếp theo. Điểm ưu việt của RAID đó là nó không làm thay đổi cấu trúc logic, vật lý của các chương trình ứng dụng cũng như các truy vấn trên cơ sở dữ liệu.

Đặc điểm quan trọng nhất của RAID đó là nâng cao hiệu năng truy xuất các tệp dữ liệu bằng cách xử lý song song.

Để có thể làm tăng tối đa tốc độ xử lý vào/ra của RAID, tất cả các ổ đĩa cứng cần phải liên tục được hoạt động. Ở đây chúng ta cần làm quen với một khái niệm mới, đó là dải (strip), một dải như trong hình sau là một tập các trang trên tất cả các ổ đĩa cứng tham gia vào RAID có cùng khoảng cách tương đối từ vùng đầu của đĩa. Như trong hình chúng ta thấy, tại một thời điểm máy tính có thể đọc 4 trang logic liên tiếp của dữ liệu theo cơ chế song song.

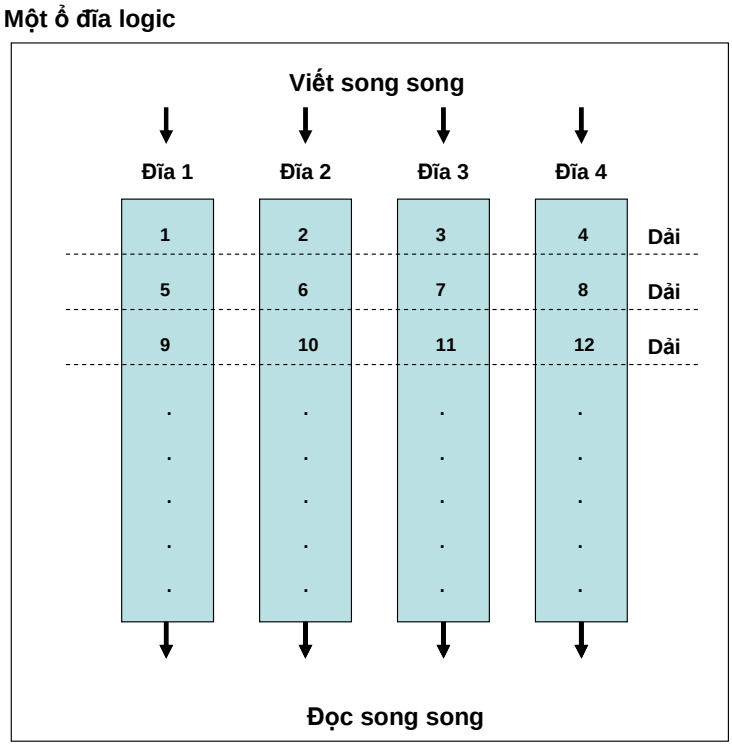


Hình 2.6.6.1 RAID với 4 ổ đĩa và các dải dữ liệu

Tuy nhiên, chúng ta cũng cần phải lưu ý một điểm quan trọng là khi sử dụng RAID sẽ làm tăng khả năng gặp lỗi đối với ổ cứng. Nếu trung bình, một ổ đĩa cứng có thể hỏng sau 1,2 triệu lần thao tác, thì đối với một RAID sử dụng 4 ổ đĩa cứng như vậy thì khả năng gặp lỗi sẽ tăng lên là 300.000 lần thao tác theo cơ chế song song. Để có thể khắc phục vấn đề này, nhiều loại công nghệ RAID được sử dụng để lưu trữ dữ liệu đã được đưa ra trong đó dữ liệu sẽ được lưu thêm một bản copy hoặc là sử dụng các mã sửa lỗi để có thể kịp thời khôi phục lại được dữ liệu trong trường hợp một ổ cứng bị hỏng. Dưới đây là một số mức RAID phổ biến:

- RAID-0: Đây là cách lưu trữ hiệu quả nhất cũng như có thời gian đọc/viết dữ liệu nhanh nhất đối với bất

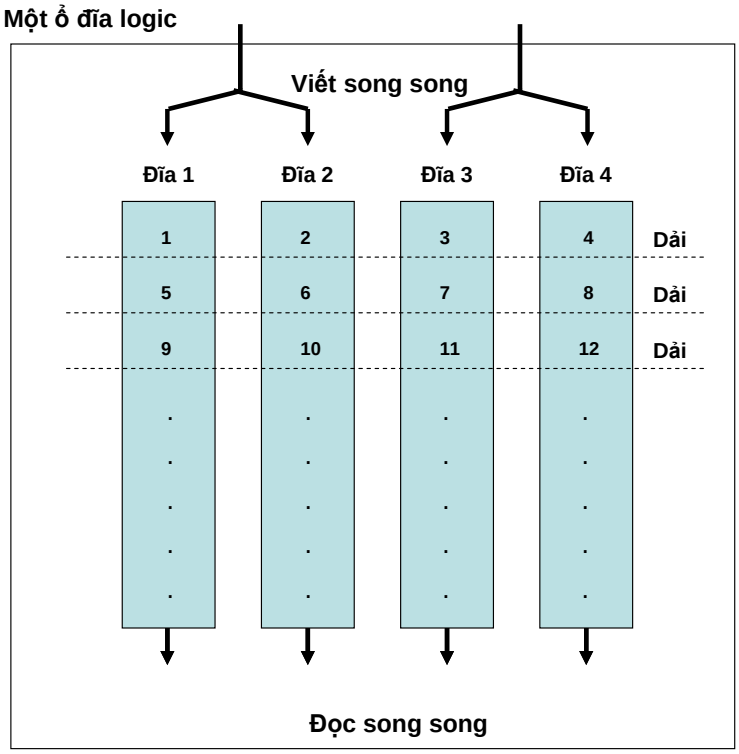
kỳ tập ổ đĩa cứng nào (hình 2.6.6.2). Trong RAID-0, mọi đĩa trong dãy đều được sử dụng để lưu trữ dữ liệu. RAID-0 thường được sử dụng khi tần suất đọc/viết dữ liệu cao cũng như nhu cầu truy suất các bản ghi dữ liệu lớn. Tuy nhiên, nhược điểm lớn nhất khiến RAID-0 ít được sử dụng đó là nó không có khả năng sửa lỗi cũng như khôi phục lại được dữ liệu khi một ổ đĩa cứng bị hỏng bởi tất cả ổ cứng đều được sử dụng tối đa, do vậy khả năng tin cậy không cao.



Hình 2.6.6.2 RAID - 0

- RAID-1: Mức RAID-1 này cho phép nâng cao khả năng phòng chống lỗi đó là lưu trữ dữ liệu dư thừa. Trong mức này, mỗi trang dữ liệu sẽ được lưu trữ thêm một bản trên một ổ cứng khác (hình 2.6.6.3). Số lượng đĩa cứng tham gia vào RAID sẽ được chia

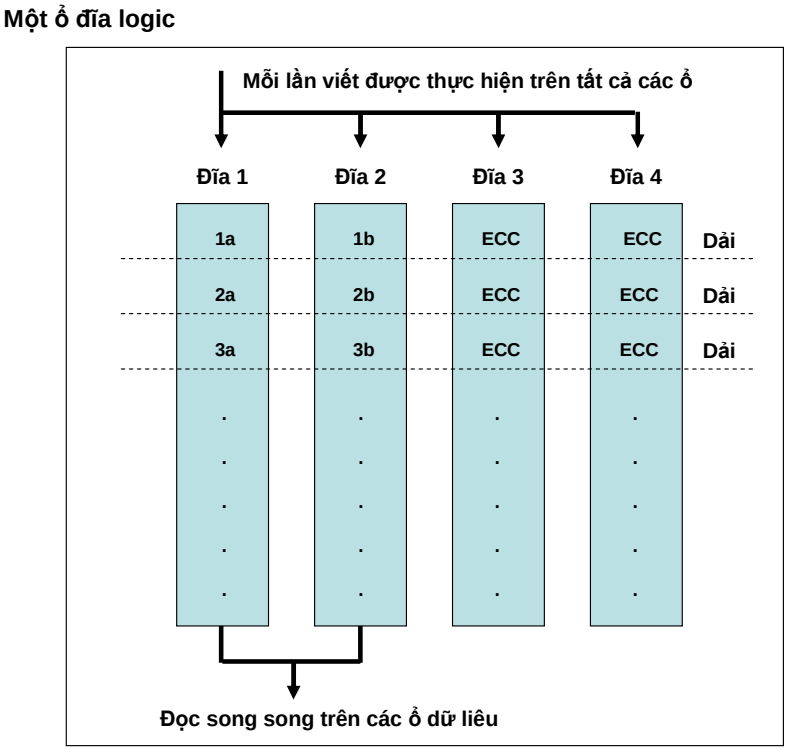
thành những cặp cùng nhau, một thao tác ghi một trang dữ liệu sẽ được tiến hành đồng thời trên mỗi cặp ổ cứng. Do vậy, với RAID-1 này, khả năng khắc phục lỗi rất cao, bởi các trường hợp để một trang dữ liệu được ghi trên 2 ổ cứng cùng xảy ra lỗi là rất ít. Trong RAID-1, các truy suất đọc sẽ hiệu quả hơn các truy suất viết bởi các truy suất viết phải được tiến hành song song vào từng cặp ổ đĩa, còn khi đọc thì có thể được thực hiện ở tất cả các ổ đĩa. RAID-1 này là một lựa chọn khá tốt cho các môi trường một cũng như đa người dùng.



Hình 2.6.6.3 RAID - 1

- RAID-2: trong mức này, người ta dành ra một số đĩa để lưu trữ các mã sửa lỗi. Chúng sẽ được sử dụng để tìm ra lỗi cũng như khôi phục lại trong các dải dữ liệu

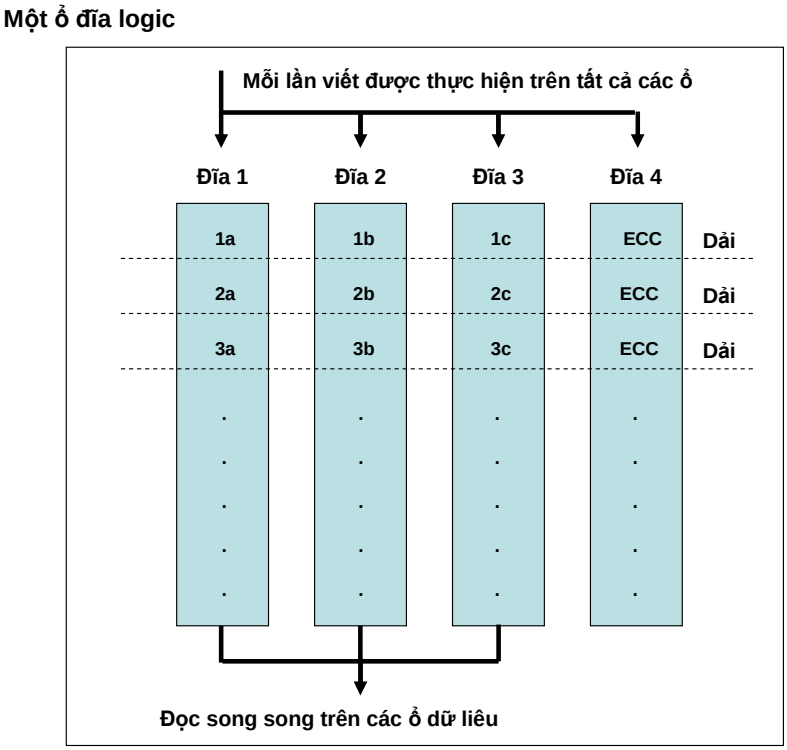
được lưu trong các ổ còn lại. Như vậy, nếu như với RAID-1 người ta sử dụng việc lưu trữ dư thừa dữ liệu để nhằm mục đích đảm bảo an toàn cho dữ liệu thì với RAID-2 (và các mức sau) này, người ta sử dụng những mã sửa lỗi hiệu quả hơn. Tuy nhiên, hầu hết các ổ đĩa cứng ngày nay đều đã được thiết kế sẵn các mã sửa lỗi trong mỗi trang dữ liệu vật lý nên RAID-2 ít được sử dụng hơn so với mức RAID-3 dưới đây.



Hình 2.6.6.4 RAID - 2

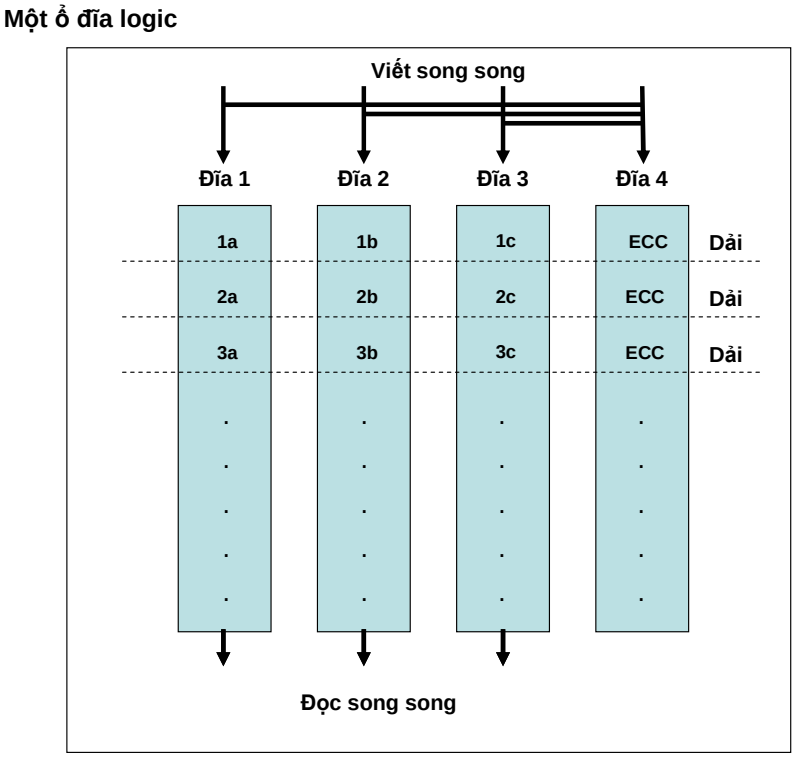
- RAID-3: Như với RAID-2, RAID-3 lưu dữ liệu trên tất cả các đĩa và chỉ dành duy nhất một đĩa để lưu trữ mã sửa lỗi (hình 2.6.6.5). Các mã sửa lỗi được tích hợp vào mỗi trang dữ liệu và được sử dụng để tìm ra các lỗi, nhưng mã sửa lỗi của trang không thể được sử dụng để sửa lỗi của chính trang đó. Giả sử rằng đĩa

số 3 bị lỗi, khi đó trang 3 trong mỗi dải dữ liệu có thể được khôi phục lại từ các trang dữ liệu còn lại kết hợp với trang sửa lỗi trong dải đó thông qua các phép toán logic OR. RAID-3 thường được sử dụng trong các cơ sở dữ liệu mà kích thước các bản ghi dữ liệu khá dài, tức là mỗi một bản ghi có thể được lưu trên một hoặc nhiều dải dữ liệu. Khi đó, mỗi một phép toán vào/ra logic sử dụng tất cả các ổ đĩa trong dãy RAID. Kết quả là nó không thể xử lý song song các yêu cầu từ nhiều người sử dụng vì mỗi một phép vào/ra logic sẽ sử dụng tất cả các ổ đĩa. Do vậy, RAID-3 thường được lựa chọn trong môi trường đơn người dùng với một chương trình duy nhất.



Hình 2.6.6.5 RAID - 3

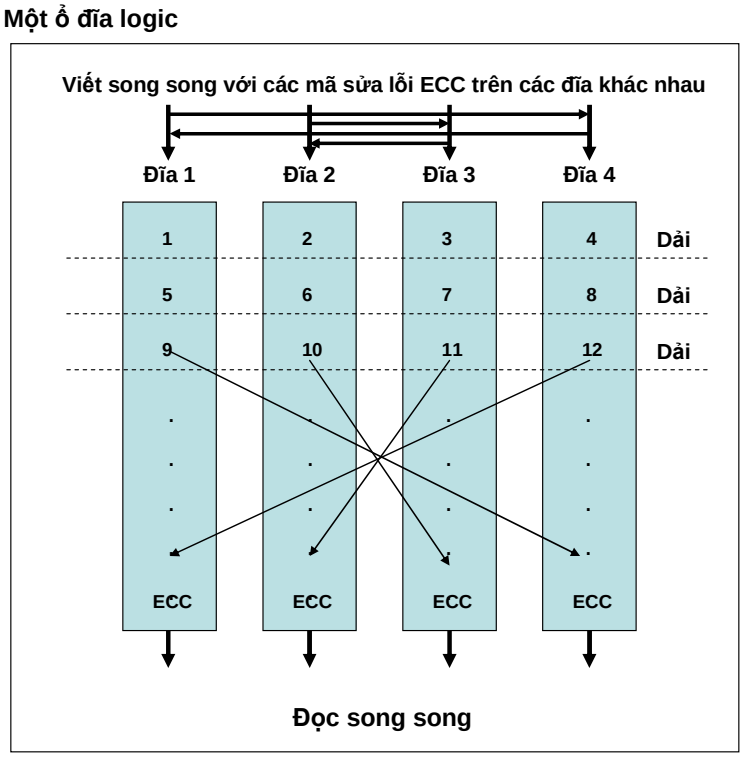
- RAID-4: Trong RAID-4, các mã sửa lỗi của các dải dữ liệu cũng được lưu duy nhất trên một ổ đĩa cứng như trong RAID-3, nhưng nó khác ở chỗ là nhiều bản ghi cùng được lưu trữ trên cùng một dải dữ liệu (hình 2.6.6.6). Do vậy các phép đọc dữ liệu trên các bản ghi khác nhau của cùng một chương trình hoặc nhiều chương trình khác nhau có thể được thực hiện song song qua các ổ đĩa. Tuy nhiên với RAID-4 này lại xảy ra trường hợp thắt cổ chai đối với các trang dữ liệu trên ổ lưu mã sửa lỗi, do khi mỗi bản ghi được cập nhật hệ thống cũng phải cập nhật các trang dữ liệu trên ổ đĩa chứa mã sửa lỗi này. Với những cơ sở dữ liệu liên tục có các phép cập nhật dữ liệu như các cơ sở dữ liệu xử lý giao dịch trực tuyến thì RAID-4 tỏ ra không hiệu quả do các phép cập nhật phải trờ đợi nhau trên ổ đĩa chứa mã sửa lỗi, và làm giảm quá trình xử lý song song. Vấn đề này được khắc phục bằng cách sử dụng RAID-5 như dưới đây.



Hình 2.6.6.6 RAID - 4

- RAID-5: Còn được gọi là mảng sửa lỗi quay vòng (rotating parity array). RAID-5 không sử dụng duy nhất một ổ đĩa làm nhiệm vụ lưu mã sửa lỗi, thay vào đó, mỗi một ổ đĩa đều chứa cả các trang dữ liệu cùng các trang chứa mã sửa lỗi. Do vậy các phép đọc có thể được thực hiện song song trên tất cả các ổ đĩa. Với phép viết, nó sẽ truy nhập đến một hoặc nhiều ổ đĩa để ghi dữ liệu cùng với một ổ đĩa để ghi mã sửa lỗi tương ứng cho trang dữ liệu đó. Các trang khác nhau sẽ lưu trữ các mã sửa lỗi của nó trên những ổ khác nhau. Do vậy, phép ghi dữ liệu cũng có thể được thực hiện song song nhưng khả năng bị trễ lại của nó đã được hạn chế hơn so với RAID-4 bởi việc cập nhật ngẫu nhiên các bản ghi cũng sẽ tạo ra các

phép cập nhật các trang sửa lỗi ngẫu nhiên trên các ổ cứng khác nhau.



Hình 2.6.6.7 RAID - 5

Phân tích hiệu quả của các mức RAID: Như chúng ta thấy, RAID-1,3 và 5 tỏ ra trội hơn so với các mức RAID khác về tính sửa lỗi. Trong đó, RAID-3 được xây dựng chủ yếu phục vụ cho các môi trường đơn người dùng hay cho các ứng dụng đặc biệt. Với hầu hết các ứng dụng cơ sở dữ liệu khác, người ta thường phải lựa chọn giữa RAID-1 và RAID-5. Đã có một số thử nghiệm được tiến hành để so sánh hai mức RAID này, kết quả cho thấy RAID-5 tỏ ra hiệu quả hơn trong việc lưu trữ dữ liệu do nó không lưu trữ dư thừa dữ liệu. Với các phép đọc dữ liệu, nó cũng tỏ ra hiệu quả hơn do sử dụng cơ chế song song trên tất cả các ổ đĩa cứng. Tuy nhiên, với phép ghi thì nó kém hơn từ 1.6

đến 3 lần so với RAID-1. Do vậy, so với RAID-1, RAID-5 ít được lựa chọn hơn trong những hệ thống sử dụng các tệp luôn luôn bị thay đổi. Ngoài ra các thử nghiệm cũng cho thấy rằng, để tăng tốc độ ghi dữ liệu lên gấp đôi so với phương pháp sử dụng một ổ đĩa, trong khi RAID-1 chỉ cần 4 ổ đĩa thì RAID-5 cần tới từ 7 cho tới 12 ổ đĩa.

Như vậy chúng ta có thể thấy rằng, RAID-1 là lựa chọn tốt nhất cho các cơ sở dữ liệu, đặc biệt với hệ thống chỉ có 2 ổ đĩa. Trong khi RAID-5 lại là lựa chọn tốt nhất cho các cơ sở dữ liệu đòi hỏi thời gian đọc dữ liệu nhanh và đỏi hỏi ít nhất từ 3 ổ đĩa trở lên (thông thường là 5).

6.7. Tối ưu hoá hiệu năng truy vấn

Tối ưu hoá hiệu năng xử lý của cơ sở dữ liệu cũng được xem là một mục đích quan trọng trong việc thiết kế cơ sở dữ liệu mức vật lý, đặc biệt đối với các cơ sở dữ liệu đòi hỏi phải đáp ứng một lượng lớn các dữ liệu tìm kiếm và trả về cho người sử dụng. Việc xử lý của cơ sở dữ liệu bao gồm các thao tác thêm, xoá và sửa đổi dữ liệu cũng như các phép tìm kiếm trả về dữ liệu.

Một trong những thay đổi đáng kể nhất trong những năm gần đây đó là việc tăng số lượng chip xử lý trong các máy chủ cơ sở dữ liệu. Các máy chủ cơ sở dữ liệu thường xuyên sử dụng công nghệ các chíp xử lý đối xứng (symmetric) hay các chip xử lý song song. Để có thể tận dụng được ưu điểm này, một số hệ quản trị cơ sở dữ liệu đã cho phép can thiệp xâu hơn vào hệ thống khi tiến hành thiết kế cơ sở dữ liệu như: các chiến lược chia nhỏ một câu lệnh truy vấn thành các phần khác nhau và mỗi phần được thực hiện song song bởi các chip khác nhau.

Một trong những cách tiếp cận thông dụng nhất đó là thực hiện câu lệnh truy vấn trên những phần khác nhau của cơ sở dữ liệu. Ví dụ với một bảng đã được chia thành phân vùng thì có thể sẽ được xử lý bởi các chip khác nhau trong cùng một câu lệnh truy vấn. Kết quả trả về sẽ được kết hợp lại để cung cấp cho người sử dụng, nhờ đó, thời gian thực hiện truy vấn giảm đi đáng kể.

Oracle là một trong những hệ quản trị cơ sở dữ liệu tiêu biểu nhất đã cho phép sử dụng cơ chế này. Ví dụ nếu bảng dữ liệu ORDER của chúng ta chứa khoảng vài triệu bản ghi, nếu được thiết kế bình thường thì khi có một truy vấn trên bảng này sẽ tỏ ra khá chậm. Tuy nhiên nếu máy chủ cơ sở dữ liệu có nhiều chip xử lý thì khi chúng ta thiết kế bảng đó, chúng ta có thể yêu cầu hệ thống sử dụng số lượng chip song song để thực hiện chẳng hạn như câu lệnh sau:

```
ALTER TABLE ORDER PARALLEL 4
```

Với các bảng khác nhau, chúng ta có thể điều chỉnh việc xử lý song song khác nhau.

Một lựa chọn khác đối với việc tận dụng được khả năng xử lý của các chip đó là khi viết câu lệnh truy vấn. Khi hệ quản trị cơ sở dữ liệu phân tích câu lệnh, nó sẽ quyết định lựa chọn kế hoạch thực hiện nào sẽ là tốt nhất dựa trên các dữ liệu thống kê về cơ sở dữ liệu, cấu trúc các bảng, số lượng các giá trị khác nhau của các trường,... để có thể lựa chọn sẽ sử dụng tệp chỉ mục nào cũng như bảng nào được thực hiện đầu tiên. Ngoài ra chúng ta có thể viết chú thích trong các câu lệnh đó để dẫn hướng cho bộ lựa chọn kế hoạch thực hiện câu lệnh đó một cách hợp lý nhất.

7. Thiết kế an toàn bảo mật

Các hệ quản trị cơ sở dữ liệu ngày càng được sử dụng rộng rãi để lưu trữ thông tin về tất cả các khía cạnh của một doanh nghiệp. Dữ liệu được lưu trữ trong một hệ quản trị cơ sở dữ liệu thường là tất yếu đối với các nghiệp vụ của doanh nghiệp và nó được xem là tài sản chung của doanh nghiệp. Thêm vào đó, để bảo vệ giá trị thực sự của dữ liệu, doanh nghiệp phải cân nhắc lựa chọn các giải pháp khác nhau nhằm đảm bảo tính bảo mật và kiểm soát được các truy nhập đến dữ liệu của các nhóm người sử dụng khác nhau với các lý do khác nhau.

Trong bước này chúng ta sẽ xem xét các vấn đề liên quan đến việc kiểm soát các truy nhập cũng như vấn đề đảm bảo an toàn trong một hệ quản trị cơ sở dữ liệu.

Khi chúng ta thiết kế an toàn bảo mật cho một cơ sở dữ liệu chúng ta cần phải cân nhắc 3 mục đích chính sau:

- Tính bí mật: Thông tin không thể được truy nhập tuỳ tiện. Một người sử dụng nếu không có thẩm quyền thì không thể truy nhập đến những phần mà không phân quyền cho anh ta. Ví dụ, một sinh viên không thể được phép truy nhập đến dữ liệu như về điểm của các sinh viên khác chẳng hạn.
- Tính toàn vẹn: Chỉ những người sử dụng có thẩm quyền mới được phép cập nhật dữ liệu. Ví dụ, các sinh viên được phép xem điểm của họ nhưng tất nhiên không được phép sửa đổi chúng.
- Tính sẵn có: Đối với những người sử dụng có thẩm quyền thì họ luôn luôn được phép truy nhập đến

những dữ liệu đã cấp quyền cho họ. Ví dụ, với giáo viên họ có thể được phép cập nhật điểm môn học cho các sinh viên mà họ đã dạy môn học đó.

Để có thể đạt được những mục đích này, một chính sách về an toàn bảo mật phải được xây dựng một cách rõ ràng và nhất quán. Cụ thể hơn, chúng ta phải xác định được những phần nào của dữ liệu cần phải được bảo vệ và những người sử dụng nào thì được phép truy nhập đến những phần dữ liệu nào. Tiếp theo, những cơ chế an toàn phải được triển khai theo những chính sách đã được xây dựng.

Chúng ta cũng cần phải nhấn mạnh đến mức độ an toàn ở những cấp độ khác nhau. Sự rò rỉ an toàn ở mức hệ điều hành hay kết nối mạng có thể phá vỡ các cơ chế đảm bảo an toàn cho cơ sở dữ liệu. Ví dụ, những rò rỉ này có thể cho phép những người sử dụng bình thường được phép đăng nhập vào máy chủ với quyền quản trị cao nhất mặc dù vai trò của họ không thể được như vậy. Ngoài ra các nhân tố liên quan đến con người cũng là một nguồn gây mất an toàn. Ví dụ như những người có thẩm quyền đặt mật khẩu quá ngắn hoặc quá dễ đoán. Trên thực tế những lỗi này chiếm một phần không nhỏ trong việc đảm bảo an toàn cho cơ sở dữ liệu, tuy nhiên ở đây chúng tôi sẽ không trình bày sâu, bởi nó đã được đề cập đến trong các giáo trình về hệ điều hành hay quản trị mạng.

Để dễ minh hoạ, trong phần này chúng tôi sẽ sử dụng những quan hệ có lược đồ như sau cho các phần trình bày tiếp theo:

SAILOR(Sid: integer, Sname: string, Rating: integer, Age:real)

BOAT(Bid: integer, Bname: string, Color: string)

RESERVER(Sname: string, Bid: integer, Day:date)

Trong đó:

- SAILOR là một quan hệ lưu danh sách các thuỷ thủ, Sid là thuộc tính mã hiệu, Sname là thuộc tính tên, Rating là thuộc tính cấp bậc và Age là thuộc tính tuổi.
- BOAT là một quan hệ lưu danh sách các tàu thuyền. Bid là thuộc tính mã hiệu, Bname là thuộc tính tên và Color là thuộc tính màu sắc.
- RESERVER: là một quan hệ đặt chỗ, cho biết thuỷ thủ nào (thuộc tính Sname) đã đặt chỗ trên chiếc tàu nào (Bid) vào ngày nào (Day). Chú ý rằng ở đây chúng tôi sử dụng thuộc tính Sname thay vì Sid trong quan hệ RESERVER để dễ dàng minh hoạ cho các câu lệnh tạo khung nhìn sau này.

7.1. Xác định các nhóm sử dụng và các quyền hạn

Như chúng ta đã biết, một cơ sở dữ liệu cho một doanh nghiệp thường chứa đựng một lượng lớn các thông tin cũng như các nhóm người sử dụng trên cơ sở dữ liệu đó. Hầu hết mỗi một người hoặc một nhóm người sử dụng chỉ làm việc trên một phạm vi dữ liệu nhỏ trong cơ sở dữ liệu đó để phục vụ cho công việc của mình. Do vậy nếu cho phép họ được phép truy nhập đến tất cả các dữ liệu trong cơ sở dữ liệu là điều không cần thiết. Chính vì vậy một hệ quản trị cơ sở dữ liệu thường cung cấp những cơ chế để có thể phân quyền và kiểm soát việc truy nhập dữ liệu này.

Với việc sử dụng các quyền do hệ quản trị cơ sở dữ liệu cung cấp, chúng ta có thể cấp phép các quyền truy nhập cũng như hình thức thao tác trên những đối tượng dữ liệu đến từng người hay nhóm người cụ thể.

Một hệ quản trị cơ sở dữ liệu sẽ lưu trữ đầy đủ thông tin về các quyền đã được cấp cho từng người sử dụng cụ thể, và cũng đảm bảo chắc chắn rằng tại bất kỳ thời điểm nào chỉ những người sử dụng được cấp phép thao tác trên những đối tượng dữ liệu đã được phân quyền cho họ.

SQL-92 hỗ trợ việc kiểm soát an toàn dựa trên các quyền thông qua các lệnh GRANT và REVOKE. Trong đó lệnh GRANT là lệnh cấp quyền cho người sử dụng, còn lệnh REVOKE là lệnh thu lại một quyền của người sử dụng. Và chỉ có những người có quyền hạn nhất định mới được phép thực hiện những lệnh này. Dưới đây chúng ta xem xét việc sử dụng các lệnh đó như thế nào đối với các đối tượng dữ liệu để phân quyền cho các đối tượng sử dụng khác nhau.

7.2. Xác định quyền hạn thao tác trên dữ liệu

SQL-92 hỗ trợ việc kiểm soát truy nhập thông qua các câu lệnh GRANT, REVOKE. Cấu trúc của chúng như sau:

```
GRANT privileges ON object TO users [WITH GRANT OPTION]
```

Trong đó

- object có thể là một bảng hay một khung nhìn, ngoài ra có thể là các đối tượng khác như hàm hay thủ tục.

- privileges là các quyền thao tác trên các đối tượng đó, với các đối tượng là các bảng, khung nhìn chúng có thể là:

  SELECT: là quyền được phép truy nhập (đọc) đến object. Ví dụ nếu object là một bảng, một khung nhìn thì quyền này cho phép truy nhập đến tất cả các cột của bảng, khung nhìn đó, thậm chí cả những cột mà sau này được thêm vào bảng, khung nhìn đó.

  INSERT (column_name): Quyền được phép chèn thêm số liệu vào các cột liệt kê của object, nếu không có các cột thì tương đương với tất cả các cột.

  UPDATE(column_name): Quyền được phép sửa đổi dữ liệu trên các cột liệt kê của object.

  DELETE: Quyền được phép xoá bỏ dữ liệu của object.

  REFERENCES (column_name): quyền được định nghĩa các khoá ngoại lai (trong các bảng khác) tham chiếu đến các column_name của object.

  Với các hàm hay thủ tục chúng có thể là quyền thực hiện EXCUTE

- users: là tên người sử dụng trong cơ sở dữ liệu.

Nếu một user được gán thêm tuỳ chọn WITH GRANT OPTION, thì anh ta có thể tiếp tục gán các quyền đó cho những người sử dụng khác. Do vậy chúng ta cần phải đặc biệt lưu ý khi sử dụng tùy chọn này, bởi có thể sẽ gặp phải

những lỗi không mong muốn do chúng ta đã trao quyền gán tiếp cho một người sử dụng khác.

Nếu một khung nhìn được tạo dựa trên các bảng và các khung nhìn khác, thì nó cũng sẽ thừa hưởng tất cả các quyền trên từng bảng, từng khung nhìn đó. Người tạo khung nhìn ít nhất phải có quyền SELECT trên các đối tượng dữ liệu mà họ muốn dựa trên chúng để tạo nên khung nhìn mới đó. Người tạo khung nhìn nếu muốn được gán khung nhìn vừa tạo cho những người sử dụng khác thì trước đó anh ta phải được gán tuỳ chọn WITH GRANT OPTION trên tất cả các đối tượng dữ liệu tạo lên khung nhìn đó.

Kết hợp với các lệnh GRANT và REVOKE, các khung nhìn được xem là một công cụ hữu hiệu, một thành phần quan trọng trong các cơ chế đảm bảo an toàn trong các hệ cơ sở dữ liệu quan hệ. Cơ chế khung nhìn được sử dụng để tạo ra một "cửa sổ" trên một tập các dữ liệu sao cho phù hợp với một hoặc một nhóm người cụ thể nào đó. Bằng cách định nghĩa các khung nhìn, chúng ta có thể giới hạn phạm vi truy nhập đối với từng người sử dụng.

Ví dụ với câu lệnh tạo khung nhìn sau:

```
CREATE VIEW ACTIVESAILOR(Sname, Age, Day)
AS SELECT S.Sname, S.Age, R.Day
FROM SAILOR S, RESERVER R
WHERE S.Sname = R.Sname AND S.Rating >6
```

chúng ta đã giới hạn việc truy nhập dữ liệu. Thông qua việc gán quyền thao tác trên khung nhìn này, chúng ta đã không cho phép người sử dụng truy nhập dữ liệu trực tiếp trên các

bảng SAILOR hay RESERVER. Với các thông tin trong ACTIVESAILOR, họ chỉ có thể biết được những thông tin như một số thủy thủ đã đặt chỗ trên những con tàu nào và vào những ngày nào mà không hề biết các thông tin như mã số của các con tàu, màu sắc,...

Dưới đây, chúng ta hãy xem xét các lệnh GRANT, REVOKE:

Giả sử Dần là người đã tạo ra các bảng BOAT, RESERVER và SAILOR trên. Anh ta thực hiện các lệnh gán sau cho các người sử dụng khác:

```
GRANT INSERT, DELETE ON RESERVER TO Ất WITH GRANT OPTION
GRANT SELECT ON RESERVER TO Mùi
GRANT SELECT ON SAILOR TO Mùi WITH GRANT OPTION
GRANT UPDATE (Rating) ON SAILOR TO Tân
GRANT REFERENCES(Bid) ON Boat TO Giáp
```

Với câu lệnh GRANT thứ nhất, Ất có thể chèn, xoá các bản ghi trên bảng RESERVER và thậm chí được phép gán tiếp các quyền này cho những người sử dụng khác, bởi anh ta đã được Dần gán thêm cho cả tuỳ chọn WITH GRANT OPTION.

Với câu lệnh thứ hai và ba, Mùi được phép truy vấn trên các bảng SAILOR và RESERVER. Ngoài ra với bảng SAILOR, anh ta còn được phép gán tiếp các quyền này cho những người sử dụng khác, nhưng với bảng RESERVER thì lại không. Với quyền SELECT, Mùi có thể tạo ra một số khung nhìn dựa trên cả 2 bảng SAILOR và RESERVER

(như khung nhìn ACTIVESAILOR trên chẳng hạn) nhưng anh ta không thể gán quyền SELECT trên khung nhìn đó cho các người sử dụng khác.

Tuy nhiên, giả sử Mùi tạo một khung nhìn lựa chọn những thuỷ thủ trẻ có tuổi dưới 18 như sau:

```
CREATE VIEW YOUNGSAILOR (Sname,Age,Rating)
AS SELECT S.Sname, S.Age, S.Rating
FROM SAILOR S
WHERE S.Age < 18
```

Khung nhìn này chỉ dựa trên duy nhất bảng SAILOR, mà Mùi lại có quyền SELECT với tuỳ chọn WITH GRANT OPTION, do vậy anh ta cũng sẽ có quyền gán quyền SELECT trên khung nhìn này, thậm chí với tuỳ chọn WITH GRANT OPTION cho các người sử dụng khác, ví dụ anh ta sẽ gán quyền SELECT trên khung nhìn này cho Hợi và Dần:

```
GRANT SELECT ON YOUNGSAILOR to Hợi, Quý
```

Khi đó Hợi và Quý có thể thực hiện các truy vấn trên khung nhìn YOUNGSAILOR, nhưng rõ ràng họ không thể truy vấn trực tiếp trên bảng SAILOR.

Mùi cũng có thể định nghĩa các ràng buộc dựa trên thông tin trong các bảng SAILOR và RESERVER. Ví dụ, anh ta có thể định nghĩa một bảng với ràng buộc kiểm tra như sau:

```
CREATE TABLE SNEAKY (Maxrating INTEGER,
CHECK (Maxrating >=
(SELECT MAX(S.Rating)
FROM SAILOR S)))
```

Bằng cách chèn dần các giá trị tăng dần vào bảng SNEAKY, Mùi có thể tìm ra được giá trị cấp bậc Rating lớn nhất trong bảng SAILOR. Điều này minh hoạ cho việc tại sao SQL lại đòi hỏi người tạo ràng buộc trên một bảng liên quan đến bảng SAILOR thì cũng phải có quyền SELECT trên bảng SAILOR.

Quay trở lại câu lệnh thứ tư với quyền UPDATE được gán cho Tân, với lệnh này cô ta chỉ có thể cập nhật các giá trị trong cột Rating của bảng SAILOR. Ví dụ, Tân được phép thực hiện lệnh cập nhật dữ liệu sau cho tất cả các thuỷ thủ:

```
UPDATE SAILOR S
SET S.Rating = 8
```

Tuy nhiên, cô ta lại không thể thực hiện được lệnh cập nhật dữ liệu tương tự nếu mệnh đề SET được thay đổi thành S.Age = 25, vì Tân đã không được gán quyền UPDATE trên trường Age. Một lưu ý khác mà cũng là lỗi rất hay gặp phải trong trường hợp này, đó là nếu Tân thực hiện lệnh sau:

```
UPDATE SAILOR S
SET S.Rating = S.Rating - 1
```

Lệnh này rõ ràng không thể được thực hiện thành công bởi nó đòi hỏi phải có quyền SELECT trên cột Rating mà Tân thì lại không được gán cho quyền này! Như vậy chúng ta thấy rằng giá trị dùng để cập nhật cũng phụ thuộc vào quyền được gán.

Với Giáp, anh ta có thể tham chiếu cột Bid của bảng Boat như là một khoá ngoại lai cho một bảng khác. Ví dụ, Giáp

có thể tạo bảng RESERVER của riêng mình thông qua lệnh sau:

```
CREATE TABLE RESERVER (Sname CHAR(10) NOT NULL,
                Bid  INTEGER,
                Day  DATE,
                PRIMARY KEY (Bid, Day),
                UNIQUE (Sname),
            FOREIGN KEY (Bid) REFERENCES BOAT)
```

Nếu Giáp không có quyền REFERENCES trên cột Bid của bảng Boat thì anh ta không thể thực hiện được câu lệnh tạo bảng trên bởi mệnh đề FOREIGN KEY đòi hỏi phải có quyền này.

Để phân biệt sự khác nhau giữa lệnh INSERT trên từng trường và trên toàn bảng SAILOR, chúng ta hãy xem tiếp:

Giả sử Dần chạy tiếp câu lệnh sau:

```
GRANT INSERT ON SAILOR TO Mùi
```

Khi đó, nếu có sự thay đổi các trường trên bảng này, như bổ sung thêm trường mới (thông qua câu lệnh ALTER TABLE) thì Mùi vẫn có quyền chèn thêm số liệu vào các trường này. Nhưng ngược lại, thay vì câu lệnh gán trên, Dần chỉ chạy câu lệnh gán như sau:

```
GRANT INSERT ON SAILOR(Sid), SAILOR(Sname), SAILOR (Rating) to Mùi
```

thì khi bảng SAILOR được bổ sung thêm các trường mới thì Mùi cũng chỉ được phép chèn dữ liệu vào 3 cột trên mà thôi.

Trái với lệnh GRANT là lệnh REVOKE, lệnh này rút lại quyền đã được cấp phát, có cấu trúc như sau:

```
REVOKE [GRANT OPTION FOR] privileges
ON object FROM users {RESTRICT | CASCADE}
```

Lệnh này nhằm loại bỏ một quyền hoặc tuỳ chọn WITH GRANT OPTION trên một đối tượng dữ liệu nào đó. Hai tuỳ chọn RESTRICT và CASCADE cũng phải được lựa chọn khi thực hiện câu lệnh này.

Tuỳ chọn CASCADE khác với RESTRICT ở chỗ, nếu khi thực hiện lệnh này thì tất cả những người sử dụng nào đã được gán tiếp bởi người sử dụng vừa bị thu lại quyền cũng sẽ bị thu lại quyền đó trừ khi anh ta được gán độc lập bởi một người sử dụng khác.

Để minh hoạ cho lệnh REVOKE này, chúng ta hãy xét tiếp ví dụ trên. Đầu tiên, hãy xem cái gì sẽ xảy ra nếu các câu lệnh sau lần lượt được thực hiện, trong đó Dần vẫn là người sở hữu (tạo ra) bảng SAILOR:

```
GRANT SELECT ON SAILOR TO Sửu WITH GRANT OPTION        (Dần chạy)
GRANT SELECT ON SAILOR TO Thìn WITH GRANT OPTION       (Sửu chạy)
REVOKE SELECT ON Salior FROM Sửu CASCADE
    (Dần chạy)
```

Trong 3 câu lệnh trên, câu lệnh thứ nhất được Dần chạy, sau câu lệnh này Sửu được quyền SELECT trên bảng SAILOR và anh ta cũng có quyền gán tiếp quyền SELECT này cho những người sử dụng khác. Và câu lệnh thứ 2 chính Sửu đã tận dụng quyền hạn này để thực hiện lệnh gán tiếp quyền SELECT trên bảng SAILOR cho Thìn và cũng với tuỳ chọn WITH GRANT OPTION. Như vậy sau câu lệnh thứ 2, Thìn có quyền SELECT trên bảng SAILOR như Sửu đã có quyền. Tuy nhiên anh ta được gán gián tiếp từ Sửu chứ không phải trực tiếp từ Dần là người đã tạo ra bảng đó và có quyền cao nhất. Do vậy, khi Dần thực hiện câu lệnh REVOKE thứ 3, anh ta sẽ loại bỏ quyền SELECT của Thìn, và với từ khoá CASCADE của câu lệnh này, nó cũng sẽ loại bỏ luôn cả quyền SELECT trên bảng SAILOR của Thìn.

Chúng ta hãy xem trường hợp khác với các câu lệnh sau:

GRANT SELECT ON SAILOR TO Sửu WITH GRANT OPTION (Dần thực hiện)

GRANT SELECT ON SAILOR TO Thìn WITH GRANT OPTION (Dần thực hiện)

GRANT SELECT ON SAILOR TO Thìn WITH GRANT OPTION (Sửu thực hiện)

REVOKE SELECT ON SAILOR FROM Sửu CASCADE (Dần thực hiện)

Với hai câu lệnh đầu, cả Sửu và Thìn đều được gán quyền SELECT trên bảng SAILOR trực tiếp từ Dần. Với câu lệnh thứ 3, Thìn nhận được quyền này gián tiếp từ Sửu. Nhưng với câu lệnh thứ 4, Sửu sẽ bị thu lại quyền SELECT do Dần đã gán. Tuy nhiên với Thìn thì sao? Thìn nhận được quyền

này từ cả Sửu, và trực tiếp từ cả Dần, do vậy Thìn vẫn có quyền này ngay cả khi Sửu không có.

Chúng ta xem tiếp các lệnh sau:

GRANT SELECT ON SAILOR TO Sửu WITH GRANT OPTION (Dần thực hiện)

GRANT SELECT ON SAILOR TO Sửu WITH GRANT OPTION (Dần thực hiện)

REVOKE SELECT ON SAILOR FROM Sửu CASCADE (Dần thực hiện)

Như trên chúng ta đã chạy 2 lần lệnh gán quyền SELECT cho Sửu nhưng lại chỉ sử dụng một lệnh thu, khi đó liệu Sửu có còn quyền đó không? Theo SQL-92 chuẩn, câu trả lời là không và ba câu lệnh trên chỉ tương đương với hai câu lệnh sau cùng.

Tiếp tục, nếu Dần thực hiện tiếp:

GRANT SELECT ON SAILOR TO Sửu WITH GRANT OPTION (Dần thực hiện)

REVOKE GRANT OPTION FOR SELECT ON SAILOR FROM Sửu CASCADE (Dần thực hiện)

Hai câu lệnh này minh hoạ cho việc gán và loại bỏ tuỳ chọn WITH GRANT OPTION cho Sửu, sau câu lệnh thứ hai, Sửu vẫn có quyền SELECT trên bảng SAILOR, nhưng anh ta không thể gán quyền này tiếp cho những người sử dụng khác.

Thông thường trong những hệ quản trị cơ sở dữ liệu, khi một lệnh GRANT được thực hiện, các thông tin về nó sẽ được lưu trong từ điển dữ liệu như các thông tin về người gán, người được gán, quyền đã gán (bao gồm cả tên của đối

tượng đã gán) và cả tuỳ chọn WITH GRANT OPTION. Còn khi người sử dụng tạo một đối tượng của riêng mình thì coi như người quản trị cơ sở dữ liệu System đã gán cho anh ta đầy đủ các quyền được phép trên đối tượng mới đó.

Khi tiến hành phân quyền, chúng ta nên sử dụng một sơ đồ trong đó mô tả đồ thị gán quyền. Trong đồ thị, các nút tương ứng với người sử dụng, các cung cho biết quyền được gán như thế nào, ở đây cần lưu ý rằng các cung là định hướng và nhãn trên cung thể hiện quyền được gán.

Sẽ tồn tại một cung từ người sử dụng 1 đến người sử dụng 2 nếu người sử dụng 1 thực hiện một lệnh GRANT gán quyền cho người sử dụng 2. Ví dụ với các lệnh gán sau, chúng ta sẽ xây dựng được một đồ thị như trong hình 2.7.2.1:

GRANT SELECT ON SAILOR TO Sửu WITH GRANT OPTION (Dần thực hiện)

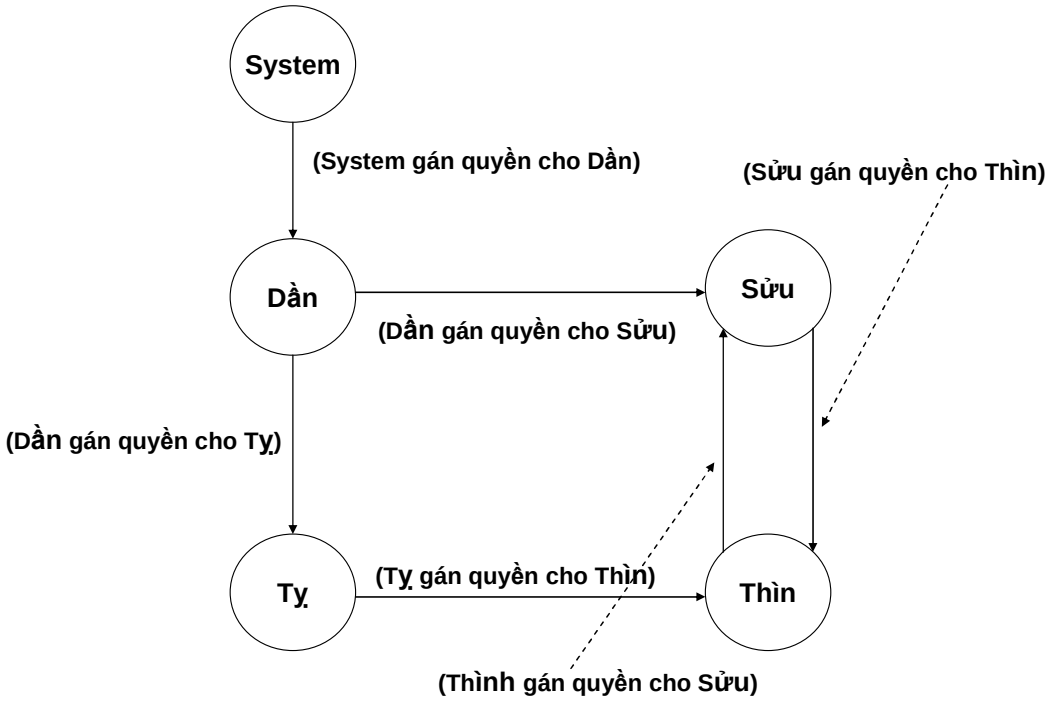
GRANT SELECT ON SAILOR TO Thìn WITH GRANT OPTION (Sửu thực hiện)

GRANT SELECT ON SAILOR TO Sửu WITH GRANT OPTION (Thìn thực hiện)

GRANT SELECT ON SAILOR TO Tỵ WITH GRANT OPTION (Dần thực hiện)

GRANT SELECT ON SAILOR TO Thìn WITH GRANT OPTION (Tỵ thực hiện)

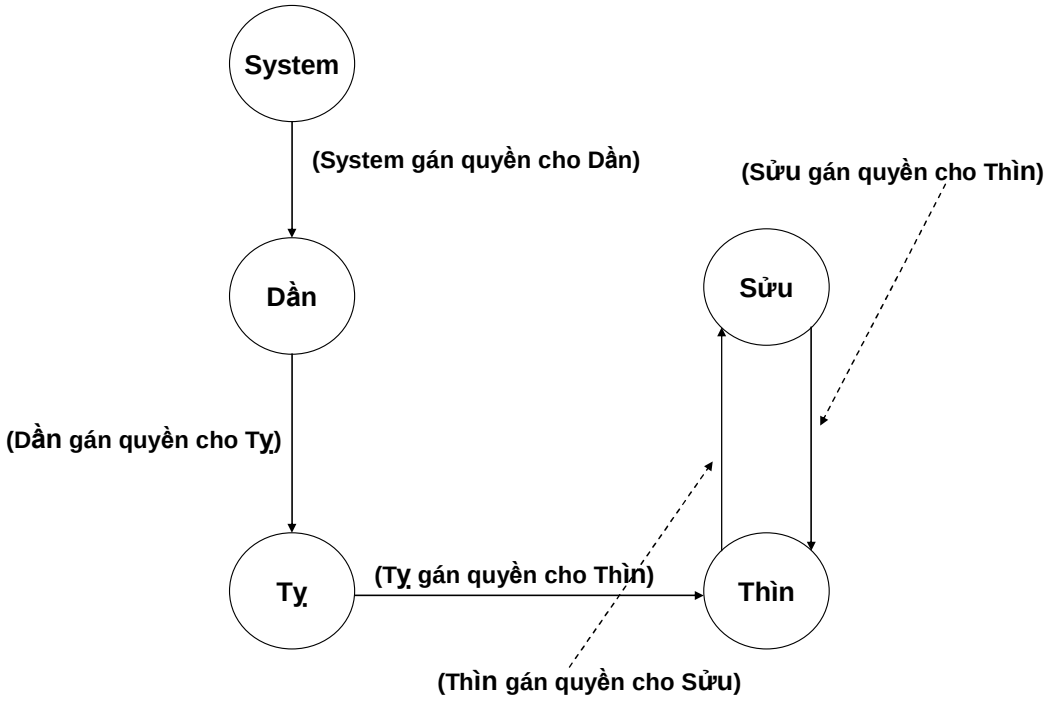
REVOKE SELECT ON SAILOR FROM Sửu CASCADE (Dần thực hiện)



Hình 2.7.2.1 Một đồ thị gán quyền

Giả thiết quyền được gán ở đây là quyền SELECT, như trong hình 2.7.2.1, do Dần sở hữu bảng SAILOR nên coi như anh ta được gán quyền bởi người quản trị cơ sở dữ liệu System. Từ đồ thị chúng ta thấy rằng, Sửu gán quyền cho Thìn, Thìn lại gán quyền lại cho Sửu tạo thành một chu trình. Ngoài ra Thìn cũng nhận được quyền từ Tỵ, mà Tỵ lại là người nhận quyền trực tiếp từ Dần.

Khi Dần quyết định loại bỏ quyền mà anh ta đã gán cho Sửu thì chúng ta sẽ thấy như sau: trên đồ thị cung nối từ Dần đến Sửu sẽ bị loại bỏ tương ứng với lệnh REVOKE. Tất cả các cung còn lại đều vẫn được giữ nguyên, lưu ý rằng ở đây mặc dù Sửu đã bị loại bỏ khỏi quyền SELECT từ Dần, nhưng Sửu vẫn nhận được quyền đó từ Thìn, và ngược lại nó lại gán quyền này cho Thìn, do đó cung từ Sửu đến Thìn vẫn tồn tại.



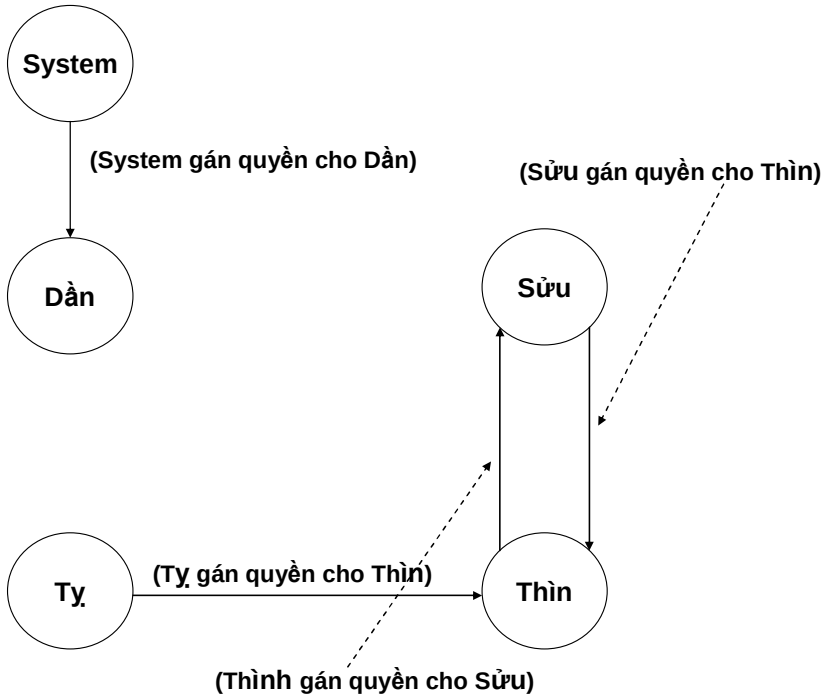
Hình 2.7.2.2 Dần loại bỏ quyền gán cho Sửu

Nói một cách tổng quát, khi loại bỏ một cung khỏi đồ thị, các cung còn lại sẽ được điều chỉnh dựa trên các đường từ nút System đến các nút.

Quay trở lại ví dụ trên, chúng ta thấy vấn đề này hơi phức tạp, do đã loại bỏ quyền SELECT của Sửu mà anh ta vẫn có quyền này!

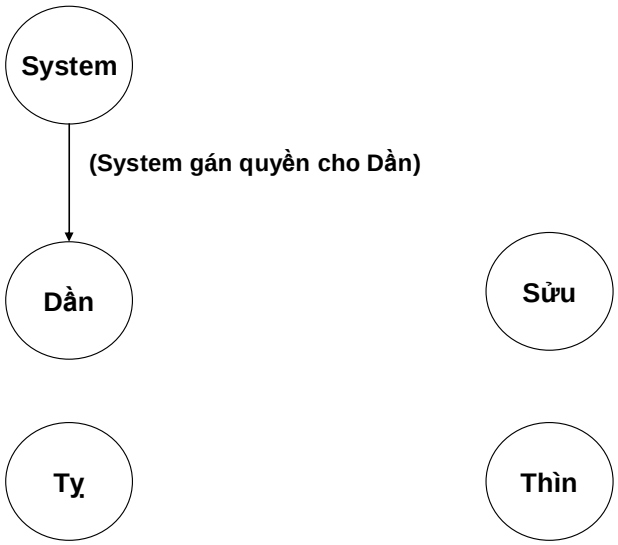
Do vậy khi thiết kế an toàn bảo mật thông qua việc gán các quyền trong cơ sở dữ liệu chúng ta phải đặc biệt lưu tâm đến vấn đề rơi vào vòng lặp loại như thế này.

Chúng ta hãy tiếp tục, nếu Dần thực hiện tiếp lệnh REVOKE để loại bỏ quyền SELECT của Tỵ, rõ ràng khi đó cũng nối từ Dần đến Tỵ sẽ bị loại bỏ và cung nối từ Tỵ đến Thìn cũng bị loại bỏ theo. Đến đây đồ thị chỉ còn hai nút nối với nhau là Sửu và Thìn



Hình 2.7.2.3 Dần loại bỏ tiếp quyền gán cho Tỵ

Tuy nhiên chúng ta nhận thấy rằng các cung này không được nối từ nút System, do đó theo qui định chúng cũng sẽ bị loại bỏ. Cuối cùng Dần đã thu lại toàn bộ các quyền đã gán cho mọi người.



Hình 2.7.2.4 Sơ đồ phân quyền cuối cùng

Dưới đây chúng ta hãy xem tiếp các lệnh GRANT và REVOKE trên các khung nhìn cùng với các ràng buộc toàn vẹn:

Như trên chúng ta đã thấy, việc tạo các khung nhìn phụ thuộc vào các bảng mà nó được xây dựng lên. Trên thực tế các lệnh gán trên các bảng cũng có thể bị thay đổi do đó các khung nhìn cũng sẽ bị thay đổi theo các lệnh gán mới. Dưới đây là một số lưu ý đối với những trường hợp này:

- Một khung nhìn sẽ bị loại bỏ nếu người tạo ra khung nhìn đó không có quyền SELECT trên những bảng hoặc khung nhìn đã được sử dụng để tạo nên khung nhìn đó.
- Nếu người tạo ra khung nhìn có thêm các quyền trên các bảng được sử dụng để tạo nên khung nhìn thì nó cũng tự động có thêm các quyền đó.

- Cần phải phân biệt rõ ràng giữa các quyền SELECT và REFERENCES

Dưới đây chúng ta xét tiếp các ví dụ để minh hoạ cho những điểm lưu ý trên.

Giả sử Dần là người đã tạo ra bảng SAILOR, anh ta gán quyền SELECT trên bảng này cho Mùi với tuỳ chọn WITH GRANT OPTION. Sau khi nhân được quyền này, Mùi tạo ra một khung nhìn mới có tên là YOUNGSAILOR dựa trên bảng SAILOR này và gán quyền SELECT trên khung nhìn này cho Hợi. Với Hợi, anh ta lại tạo một khung nhìn tiếp theo được đặt tên là FINEYOUNGSAILOR dựa trên khung nhìn YOUNGSAILOR do Mùi đã gán cho, các câu lệnh lần lượt như sau:

```
GRANT SELECT ON SAILOR TO Mùi WITH GRANT OPTION  (Dần thực hiện)

CREATE VIEW YOUNGSAILOR (Sname,Age,Rating)
AS SELECT S.Sname, S.Age, S.Rating
FROM SAILOR S
WHERE S.Age < 18                                (Mùi thực hiện)
GRANT SELECT ON YOUNGSAILOR TO Hợi
    (Mùi thực hiện)

CREATE VIEW FineYOUNGSAILOR (Sname,Age,Rating)
AS SELECT S.name, S.Age, S.Rating
FROM YOUNGSAILOR S
```

WHERE S.Rating > 6 (Hợi thực hiện)

Bây giờ chúng ta hãy xem điều gì sẽ xảy ra nếu Dần loại bỏ quyền SELECT trên bảng SAILOR từ Mùi. Khi đó Mùi sẽ không còn quyền SELECT trên bảng SAILOR, do đó khung nhìn YOUNGSAILOR do anh ta tạo ra sẽ không còn tác dụng, và theo hệ quả, khung nhìn FINEYOUNGSAILOR của Hợi cũng không còn tác dụng do không thể SELECT trên khung nhìn YOUNGSAILOR của Mùi. Các định nghĩa về hai khung nhìn này sẽ bị loại bỏ khỏi từ điển dữ liệu. Ngay cả khi sau đó Dần thực hiện lại quyền gán SELECT cho Mùi thì chúng ta vẫn cứ phải tiến hành lại các lệnh tạo khung nhìn trên.

Hãy xét một trường hợp khác, giả sử các câu lệnh gán và tạo khung nhìn trên vẫn được thực hiện, nếu Dần gán thêm quyền INSERT trên bảng SAILOR cho Mùi, khi đó, nếu khung nhìn do Mùi tạo ra có quyền cập nhật thì tự động quyền INSERT này cũng có tác dụng trên khung nhìn đó. Tuy nhiên nếu Mùi không gán tiếp quyền INSERT này cho Hợi thì khung nhìn do Hợi tạo ra vẫn chỉ có quyền SELECT mà không có quyền INSERT.

Tiếp theo chúng ta hãy xét đến ví dụ minh hoạ cho sự khác nhau giữa quyền SELECT và REFERENCES:

Trong ví dụ này chúng ta giả sử Dần vẫn là người tạo ra bảng BOAT, nhưng Hợi lại là người tạo ra bảng RESERVER, và anh ta muốn tham chiếu đến khoá chính Bid trong bảng BOAT của Dần. Để có thể thực hiện được điều này, Dần nhất thiết phải cấp quyền REFERENCES trên cột Bid của bảng BOAT cho Hợi.

Ngược lại, giả sử nếu Dần chỉ gán quyền SELECT trên bảng BOAT cho Hợi thì Hợi cũng không thể thực hiện được việc tham chiếu này cho dù anh ta có thể SELECT toàn bộ dữ liệu trên bảng BOAT của Dần.

Trong trường hợp Dần gán quyền REFERENCES trên cột Bid cho Hợi và giả thiết Hợi đã tạo xong tham chiếu của khoá ngoại lai trong bảng RESERVER của mình đến cột Bid của bảng BOAT. Sau một thời gian Dần lại muốn loại bỏ quyền REFERENCES này đi, khi đó bảng Reserve của Hợi vẫn tồn tại nhưng ràng buộc khoá ngoại lai đến trường Bid của bảng BOAT sẽ bị loại bỏ.

Để có thể hiểu được tại sao SQL-92 chuẩn lại phải đưa ra hai quyền REFERENCES và SELECT thay vì chỉ sử dụng mỗi quyền SELECT. Chúng ta hãy xem các trường hợp sau:

Nếu Dần chỉ gán quyền SELECT mà không gán quyền REFERENCES cho Hợi, và khi tạo bảng RESERVER Hợi không xác lập tuỳ chọn REFERENCEScho khoá ngoại lai Bid, khi đó Dần dễ dàng xoá bỏ các bản ghi trong bảng BOAT, và chính điều này sẽ gây ra sự dị thường trong bảng RESERVER của Hợi.

Trong trường hợp ngược lại, nếu Hợi được gán đúng quyền REFERENCES và anh ta sử dụng quyền này trong tuỳ chọn tham chiếu của khoá ngoại lai Bid trong bảng RESERVER của mình thì khi Dần xoá các bản ghi trong bảng Boat, nếu giá trị trong cột Bid đã được sử dụng trong bảng RESERVER của Hợi thì Dần không thể xoá được các bản ghi đó. Như vậy chúng ta thấy chỉ khi đó dữ liệu trong cơ sở dữ liệu mới được đảm bảo tính toàn vẹn và nhất quán.

7.3. Mã hoá dữ liệu

Một hệ quản trị cơ sở dữ liệu có thể sử dụng các cơ chế mã hoá để bảo vệ thông tin của mình trong những trường hợp mà các cơ chế an toàn thông thường không đáp ứng đủ. Ví dụ như các tin tặc có thể lấy cắp dữ liệu thông qua các thiết bị gắn vào đường truyền (tap), hay qua các thiết bị lưu trữ dữ liệu. Tuy nhiên nếu dữ liệu được mã hoá trong các công đoạn lưu trữ và truyền dẫn thì hệ quản trị cơ sở dữ liệu có thể đảm bảo rằng những dữ liệu đã bị đánh cắp trên đường truyền hay trong khi lưu trữ sẽ khó có thể sử dụng được hoặc người ta phải mất rất nhiều thời gian và công sức để giải mã nó.

Ý tưởng cơ bản đằng sau vấn đề mã hoá đó là việc áp dụng các thuật toán mã hoá. Việc áp dụng một thuật toán mã hoá phải đi kèm với một thuật toán giải mã cùng với khoá được giải. Dữ liệu sau khi qua thuật toán mã hoá sẽ trở thành dữ liệu được mã hoá và truyền tới người nhận, người nhận muốn hiển thị được dữ liệu thật phải áp dụng một thuật toán giải mã tương ứng thông qua khoá giải mã mà người đã mã hoá cung cấp cho. Cách tiếp cận này đã trở thành một chuẩn mã hoá dữ liệu (DES) xuất hiện từ năm 1977, với một thuật toán mã hoá chứa các ký tự thay thế và hoán vị. Điểm yếu nhất của cách tiếp cận này đó là khoá được sử dụng mã hoá có thể bị đánh cắp do người ta phải cung cấp nó cho những người muốn nhận được dữ liệu đã bị mã hóa.

Một cách tiếp cận khác đối với vấn đề mã hoá đó là sử dụng phương pháp mã hoá với khoá công khai, phương pháp này đã trở nên khá phổ biến trong những năm gần đây. Sơ đồ mã hoá đã được đề xuất bởi Rivest, Shamir và

Adleman, được gọi là RSA. Nguyên lý của cách tiếp cận này như sau, mỗi một người sử dụng đều có một khoá mã hoá, khoá này được công bố công khai đến tất cả mọi người và một khoá giải mã (được sử dụng trong thuật toán giải mã), khoá này thì chỉ mình người sử dụng đó được biết và anh ta sẽ bị lộ dữ liệu nếu để mất khóa giải mã này. Các thuật toán mã hoá và giải mã cũng có thể được công khai cho mọi người biết. Để dễ hình dung hơn đối với cách tiếp cận này, chúng ta hãy xem ví dụ sau:

Ví dụ: Giả sử có một người tên là Sam, bất kể người nào muốn gửi dữ liệu cho anh ta mà không muốn bị đánh cắp dữ liệu đều phải sử dụng khoá công khai cùng với thuật toán do Sam đã công bố. Dữ liệu sẽ được mã hoá và gửi đến Sam. Khi dữ liệu đến Sam, anh ta phải sử dụng khoá riêng của mình cùng với thuật toán giải mã để giải mã dữ liệu, do vậy ngay cả khi dữ liệu bị đánh cắp trên đường truyền thì cũng không thể giải mã được bởi khoá giải mã chỉ có mình Sam biết được. Cách tiếp này đã khắc phục được hạn chế của cách tiếp cận theo chuẩn mã hoá dữ liệu DES thông thường nên nó được sử dụng khá rộng rãi.

Vấn đề chính đối với cách tiếp cận mã hoá với khoá công khai ở chỗ đó là làm sao chọn được khoá mã hoá và khoá giải mã. Về kỹ thuật, các thuật toán mã hoá với khoá công khai sử dụng các hàm một chiều là các hàm mà việc xác định chiều ngược lại rất khó tìm hoặc hầu như không thể tính nổi. Ví dụ, một thuật toán RSA rất nhiều người biết đến đó là dựa trên các số nguyên tố, như chúng ta đã biết việc xác định xem một số có phải là số nguyên tố không thì rất dễ, tuy nhiên để tìm ra các thừa số nguyên tố của một số không phải là số nguyên tố thì lại là vô cùng khó. Người ta

đã chứng minh được rằng để tìm ra các thừa số nguyên tố của một số có trên 100 chữ số thì với một máy tính chạy nhanh nhất hiện nay cũng phải mất hàng năm.

Chúng ta hãy đi xâu tìm hiểu xem thuật toán RSA như thế nào. Giả sử dữ liệu cần được mã hoá là một số nguyên I, để lựa chọn khoá mã hoá và khoá giải mã, Sam đầu tiên lựa chọn một số nguyên rất lớn và đặt tên là limit, chúng ta giả thiết rằng số limit này lớn hơn số nguyên lớn nhất I muốn mã hoá. Sam đã lựa chọn số limit này là tích của 2 số nguyên tố p và q (p*q). Sau đó Sam lựa chọn một số nguyên tố khác e lớn hơn cả p và q làm khoá mã hoá. Cả limit và e đều được đưa ra công khai với mọi người và sẽ được sử dụng trong thuật toán mã hoá.

Bây giờ Sam phải lựa chọn cho mình khoá giải mã d theo một phương pháp đặc biệt dựa trên cả p,q và e (ví dụ, d được lựa chọn sao cho d*e = 1 mode((p-1)*(q-1))). Điểm thiết yếu ở đây đó là d rất dễ tính ra nếu biết cả p,q và e, nhưng lại rất khó tính nếu chỉ biết e và limit (bằng tích của p*q). Bởi như trên chúng ta thấy rằng rất khó để xác định các thừa số nguyên tố của limit.

Ví dụ một công ty buôn bán đã quý cho phép nhận các đơn đặt hàng trên mạng internet và lưu trữ chúng trong một cơ sở dữ liệu, các yêu cầu đặt ra sẽ là:

- Chỉ công ty mới có khả năng đọc được một đơn đặt hàng. Ngoài ra khách hàng (ví dụ Sam sẽ là khách hàng của công ty) đặt đơn hàng đó của công ty cũng muốn các đơn hàng của anh ta được giữ bí mật.

- Công ty có thể kiểm tra chính xác một đơn hàng có thực sự được đặt bởi khách hàng Sam hay không, hay bởi một người nào khác tự nhận là Sam.

Để đảm bảo các yêu cầu đó, công ty sẽ yêu cầu mỗi một khách hàng phải lựa chọn cho mình một khoá mã hoá công khai (giả sử Sam chọn khoá là eSam) và một khoá giải mã bí mật (dSam). Công ty cũng lựa chọn cho mình một khoá mã hoá công khai (eEm). Khoá giải mã của công ty dEm sẽ được giữ bí mật. Các khách hàng cũng được yêu cầu là họ phải giữ bí mật khoá giải mã của mình (như dSam).

Bây giờ chúng ta hãy xem hai yêu cầu trên được thoả mãn như thế nào. Để đặt một đơn hàng, Sam sử dụng khoá công khai eEm của công ty để mã hoá đơn hàng của mình và gửi cho công ty. Công ty khi nhận được đơn hàng này sẽ sử dụng khoá giải mã dEm để giải mã. Như vậy chúng ta thấy rằng giai đoạn này thoả mãn yêu cầu thứ nhất trên, bởi dEm chỉ duy nhất công ty kinh doanh có thể biết được. Tuy nhiên, ở đây chúng ta sẽ đặt câu hỏi là khoá eEm là công khai với tất cả mọi người, vậy thì một kẻ nào đó có thể mạo danh là Sam và gửi đến công ty một đơn hàng cũng được mã hoá bởi khoá eEm thì công ty làm sao biết được? Rõ ràng theo cách này thì công ty không thể biết được đơn hàng đó có đúng do Sam đặt không, họ có thể cung cấp cho mỗi một khách hàng một account để đăng nhập hệ thống và đặt đơn hàng, tuy nhiên khi số lượng khách hàng lớn (hàng nghìn chẳng hạn) thì cách cung cấp tài khoản đăng nhập cho khách hàng cũng tỏ ra không hiệu quả.

Để có thể khắc phục vấn đề trên, và để xác định chính xác đơn đặt hàng là do đúng khách hàng của công ty đặt, công ty đã yêu cầu các khách hàng của mình thực hiện các bước

sau khi đặt một đơn hàng. Thay vì sử dụng ngay lập tức khóa công khai eEm của công ty để mã hoá đơn hàng, khách hàng Sam đầu tiên áp dụng thuật toán giải mã, sử dụng mã riêng của mình dSam (công ty có thể không biêt khoá này) để mã hoá. Rõ ràng, sau bước này do Sam không sử dụng thuật toán mã hoá nên đơn hàng sẽ ở một dạng đặc biệt, tiếp theo Sam mới sử dụng khoá công khai eEm của công ty để mã hoá tiếp và gửi cho công ty.

Khi công ty nhận được đơn hàng này, đầu tiên họ sẽ giải mã nó sử dụng khoá riêng dEm của mình. Bước này đơn hàng được đưa về dạng đặc biệt do Sam đã mã hoá ở bước đầu. Tiếp theo công ty áp dụng tiếp thuật toán mã hoá đối với bản này và sử dụng khoá công khai eSam do Sam cung cấp và họ sẽ nhận được đơn hàng cuối cùng. Như vậy theo cách này công ty sẽ nhận được đơn hàng đảm bảo do chính khách hàng Sam gửi đến bởi cả hai khoá bí mật dSam và dEm đều được sử dụng. Như vậy nếu đơn hàng là giả mạo thì nó không thể biết khoá giải mã dSam của Sam, và kết quả mã hoá với khoá công khai eSam của Sam sẽ không thể đưa về đơn hàng nguyên thuỷ ban đầu. Ngoài ra, một ưu điểm nữa đó là công ty cũng không hề biết khoá dSam của khách hàng Sam do đó Sam có thể hoàn toàn tin tưởng rằng không ai có thể biết được khoá bí mật của mình trừ khi anh ta để lộ nó. Chúng ta có sơ đồ cho quá trình này như sau:

Sam → khoá riêng dSam → khoá công khai eEm → khoá riêng dEm → khóa công khai eSam → công ty.

Việc sử dụng mã hoá với khoá công khai không giới hạn trong các hệ quản trị cơ sở dữ liệu, nó được sử dụng rộng rãi ở khắp các ứng dụng khác, và nhờ có nó mà thương mại điện tử trên mạng Internet đã phát triển đáng kể.

7.4. Vai trò của người quản trị

Người quản trị cơ sở dữ liệu đóng một vai trò quan trọng trong việc đảm bảo các tính năng an toàn liên quan của cơ sở dữ liệu. Anh ta phải là người thực sự có kinh nghiệm và hiểu biết sâu sắc hệ quản trị cơ sở dữ liệu mà mình phải quản lý

Người quản trị cơ sở dữ liệu luôn là người sử dụng có quyền cao nhất mà chúng ta thường gọi là System account (như trong các ví dụ trên đã trình bày) và chịu trách nhiệm về toàn bộ các vấn đề an toàn của hệ thống cơ sở dữ liệu. Dưới đây là một số vai trò và trách nhiệm của người quản trị:

Các vai trò và trách nhiệm cơ bản:

- Cài đặt các phần mềm cho hệ thống và đặc biệt là hệ quản trị cơ sở dữ liệu.
- Tiến hành cấu hình cả phần cứng và phần mềm để cố gắng đảm bảo cho hệ quản trị cơ sở dữ liệu hoạt động hiệu quả nhất.
- Quản trị các vấn đề an toàn cho các đối tượng trong cơ sở dữ liệu.
- Kiểm soát các vấn đề an toàn cho cơ sở dữ liệu.
- Tham gia đóng góp vào việc thiết kế các chính sách an toàn bảo mật cho các đối tượng dữ liệu.
- Kiểm tra các vấn đề liên quan đến hiệu năng của hệ cơ sở dữ liệu và nếu cần thiết điều chỉnh lại các tham hệ thống.

- Thực hiện công việc sao lưu và khôi phục cho hệ cơ sở dữ liệu định kỳ.
- Tiến hành định kỳ các công việc bảo trì cho hệ cơ sở dữ liệu.

Ngoài ra người quản trị cơ sở dữ liệu cũng có thể thực hiện một số công việc sau:

- Tham gia phân tích dữ liệu..
- Tham gia thiết kế cơ sở dữ liệu.
- Hỗ trợ người phát triển chương trình.
- Tham gia vào quá trình đề xuất ra các chuẩn trong doanh nghiệp.
- Tham gia vào quá trình đánh giá và kiểm tra các phần mềm trong hệ thống.
- Lưu trữ lại các thông số về môi trường cho hệ quản trị cơ sở dữ liệu.
- Tham gia đánh giá các vấn đề về mua sắm nâng cấp các phần mềm cũng như phần cứng mới.
- Lên kế hoạch và đề xuất khi thấy hệ thống cần thiết phải nâng cấp để đáp ứng các nhu cầu tiếp theo của doanh nghiệp.

Chúng ta sẽ có dịp tìm hiểu cụ thể những vai trò và trách nhiệm trên của người quản trị hệ cơ sở dữ liệu trong chủ đề tiếp theo: Quản trị hệ thống cơ sở dữ liệu.

8. Kết luận

Việc chia quá trình xây dựng cơ sở dữ liệu thành các bước nên được xem như là việc phân nhóm các bước khác nhau trong thiết kế. Trên thực tế, mặc dù chúng ta có thể bắt đầu với sáu bước trình bày trên, việc thiết kế một cơ sở dữ liệu hoàn chỉnh có lẽ vấn đòi hỏi những khâu điều chỉnh thêm mà trong đó cả sáu bước thiết kế trên đều được xen vào và được lặp lại cho tới khi thoả mãn các yêu cầu thiết kế. Ngoài ra, ở đây chúng ta cũng không trình bày một số bước quan trọng khác như bước cài đặt thiết kế cơ sở dữ liệu, các bước xây dựng và cài đặt các lớp ứng dụng chạy trên hệ quản trị cơ sở dữ liệu. Tất nhiên, trên thực tế những bước này có thể dẫn tới việc xem xét lại quá trình thiết kế cơ sở dữ liệu.

Những khái niệm và công nghệ của hệ quản trị cơ sở dữ liệu quan hệ là rất hữu ích cho những người muốn cài đặt và bảo trì một hệ thống cơ sở dữ liệu. Đối với những người làm công tác quản trị, điều quan trọng là họ phải biết một hệ quản trị cơ sở dữ liệu hoạt động như thế nào. Nắm vững những thành phần bên trong của một hệ cơ sở dữ liệu sẽ giúp ích rất nhiều cho công việc thiết kế thành công một cơ sở dữ liệu, đặc biệt là trong các bước thiết kế vật lý và điều chỉnh cơ sở dữ liệu.

9. Bài tập

1. Vẽ sơ đồ cho mỗi liên kết sau:

- Giữa kiểu thực thể sinh viên STUDENT và khoá học COURSE (sinh viên đăng ký học các khoá học khác nhau)

- Giữa kiểu thực thể sách BOOK và các bản photo BOOK_COPY.
- Giữa kiểu thực thể giảng viên TEACHER và khoá học COURSE
- Giữa kiểu thực thể ngân hàng BANK và chi nhánh BRANCH (mỗi một chi nhánh luôn thuộc vào một ngân hàng nhất định).

2. Lấy một số ví dụ về kiểu thực thể mạnh và một số kiểu thực thể yếu đi cùng với nó.

3. Thế nào là thuộc tính khoá chính, thuộc tính khoá ngoại lai? Lấy các ví dụ minh hoạ

4. Thế nào là liên kết một-một, liên kết một-nhiều, liên kết nhiều-nhiều.? Lấy ví dụ minh hoạ trên thực tế cho từng loại liên kết đó.

5. Vẽ sơ đồ thực thể - liên kết cho những trường hợp sau (bạn có thể bổ sung thêm một số giả thiết nếu thấy cần):

- Một công ty có một số lượng nhân viên. Các thuộc tính của kiểu thực thể nhân viên EMPLOYEE bao gồm Emp_id (đồng thời là thuộc tính định danh), Name, Address, Birth_date. Đồng thời công ty cũng quản lý một số dự án, các thuộc tính của kiểu thực thể dự án PROJECT này bao gồm Proj_id (thuộc tính định danh), Proj_name, Start_date. Mỗi một nhân viên có thể tham gia vào một hoặc một số dự án, hoặc có thể không tham gia vào bất cứ dự án nào. Tuy nhiên, mỗi một dự án phải bao gồm ít nhất một nhân viên. Công ty có nhu cầu quản lý việc cấp kinh phí cho từng dự án đến từng nhân viên. Bạn hãy tạo thêm những kiểu thực thể mới để giải quyết nhu cầu đó, và

xác định các liên kết với hai kiểu thực thể EMPLOYEE và PROJECT đã có.

- Một phòng nghiên cứu bao gồm một số nhân viên EMPLOYEE cùng một số dự án, mỗi một dự án sẽ sử dụng một số thiết bị nào đó, chúng ta đặt tên cho kiểu thực thể thiết bị là EQUIPMENT bao gồm hai thuộc tính là Serial_no và Cost. Người ta muốn quản lý việc phân phối sử dụng các thiết bị cho từng nhân viên trong các dự án khác nhau thông qua một thuộc tính là ngày giao thiết bị Assign_date. Bạn hãy xây dựng một sơ đồ thực thể - liên kết cho nhu cầu đó.

6. Tìm hiểu bộ công cụ Oracle Designer2000 của Oracle. Thể hiện sơ đồ thực thể - liên kết trong hình 2.3.7.6 bằng tiện ích Entity Relationship Diagrammer của bộ công cụ này.

7. Vẽ sơ đồ thực thể - liên kết cho bài tập 6 trong chủ đề 1.

Chủ đề III: Quản trị hệ thống cơ sở dữ liệu

Mục tiêu:

Trong chủ đề này cần phải nắm được những vấn đề sau:

- Các mô hình tổ chức cơ sở dữ liệu, đặc biệt là mô hình tổ chức hệ cơ sở dữ liệu tập trung.
- Vai trò và nhiệm vụ chính của người quản trị cơ sở dữ liệu.
- Các vấn đề quản trị hệ thống cơ sở dữ liệu về mặt nội dung
- Các vấn đề quản trị hệ thống cơ sở dữ liệu về mặt kỹ thuật
- Các vấn đề quản trị hệ thống cơ sở dữ liệu về mặt an ninh, phân quyền
- Cách khắc phục một số lỗi thường gặp đối với hệ quản trị cơ sở dữ liệu.

1. Khái quát chung

Sau khi đã xây dựng xong một cơ sở dữ liệu và đưa vào hoạt động, công việc tiếp theo sẽ là vận hành và quản lý cơ sở dữ liệu đó hay nói một cách khác là quản trị hệ thống cơ sở dữ liệu đó.

Việc quản trị một hệ thống cơ sở dữ liệu bao gồm rất nhiều vấn đề và thông thường, với một hệ thống cơ sở dữ liệu lớn thì cần phải có một đội ngũ chuyên gia tin học không những chỉ giỏi về chuyên môn và có nhiều kinh nghiệm mà cần cả sự hiểu biết về các nghiệp vụ của ứng dụng vận hành trên cơ sở dữ liệu đã được xây dựng. Họ được gọi là những nhà quản trị hệ thống cơ sở dữ liệu.

Trong chủ đề này chúng ta sẽ đi sâu tìm hiểu vai trò và trách nhiệm đối với một người quản trị hệ thống cơ sở dữ liệu. Các vấn đề cần phải thực hiện khi quản trị một hệ thống cơ sở dữ liệu như quản trị hệ thống cơ sở dữ liệu về mặt nội dung, về mặt kỹ thuật, về các vấn đề liên quan đến đảm bảo an ninh, phân quyền dữ liệu. Ngoài ra, trong phần cuối của chủ đề này chúng tôi sẽ dành một mục để đề cập đến một số lỗi thường gặp trong quá trình quản trị một hệ thống cơ sở dữ liệu và hướng dẫn một số cách khắc phục chúng.

Trước tiên, chúng ta cùng xem xét một số mô hình tổ chức cơ sở dữ liệu.

2. Các mô hình tổ chức cơ sở dữ liệu

Các mô hình tổ chức cơ sở dữ liệu thường được phân chia thành hai loại, đó là các mô hình tổ chức cơ sở dữ liệu tập trung và các mô hình tổ chức cơ sở dữ liệu phân tán. Mặc dù mô hình tổ chức cơ sở dữ liệu tập trung có các ưu điểm vượt trội của nó, song trong một số trường hợp, người ta vẫn phải xây dựng những hệ thống cơ sở dữ liệu phân tán đặc biệt là khi hệ thống cần phải tiến hành thực hiện song song ở nhiều vị trí vật lý khác nhau và đặc biệt là khi có các yêu cầu về tốc độ cũng như đặc trưng của dữ liệu. Thường

những hệ thống phải sử dụng mô hình cơ sở dữ liệu phân tán là những hệ thống lớn mà vị trí địa lý của nó trải dài trên một vùng lãnh thổ như trong một quốc gia, một châu lục hay trên toàn thế giới.

Việc thiết kế cơ sở dữ liệu theo mô hình phân tán và giải quyết các vấn đề liên quan đến quản trị những hệ thống này là rất phức tạp, đòi hỏi khá nhiều thời gian và công sức. Bởi vậy, trong chủ đề này chúng tôi chỉ trình bày các vấn đề liên quan đến quản trị các hệ thống cơ sở dữ liệu theo mô hình tập trung.

2.1. Các hệ cơ sở dữ liệu tập trung

Với một hệ cơ sở dữ liệu tập trung, tất cả các dữ liệu được định vị tại một trạm đơn lẻ, những người sử dụng tại các trạm từ xa có thể truy nhập vào cơ sở dữ liệu thông qua các công cụ truy thông dữ liệu. Các hệ cơ sở dữ liệu tập trung đảm bảo sự kiểm soát tốt hơn trong việc truy nhập và cập nhật dữ liệu so với các hệ cơ sở dữ liệu phân tán, song lại có thể bị mắc lỗi nhiều hơn do chúng phụ thuộc vào tính sẵn sàng của các tài nguyên.

a. Hệ cơ sở dữ liệu cá nhân

Hệ cơ sở dữ liệu cá nhân là một kiểu hệ cơ sở dữ liệu rất quen thuộc đối với hầu hết mọi người. Các hệ cơ sở dữ liệu cá nhân thường do một người sử dụng đơn lẻ vừa thiết kế, tạo lập cơ sở dữ liệu, trực tiếp viết chương trình ứng dụng, đồng thời cũng là người sử dụng cuối của hệ. Thông thường, hệ cơ sở dữ liệu này hỗ trợ một ứng dụng hay một số các ứng dụng đơn lẻ.

Các ứng dụng phổ biến nhất của các hệ cơ sở dữ liệu cá nhân là các ứng dụng về các công việc kinh doanh và quản lý có quy mô nhỏ. Các ứng dụng điển hình bao gồm: quản lý thanh toán, kiểm kê hàng hoá và quản lý khách hàng. Việc phát triển và sử dụng các hệ cơ sở dữ liệu cá nhân là khá đơn giản và dễ dàng.

Các hệ cơ sở dữ liệu cá nhân được sử dụng trong các tổ chức có quy mô vừa và nhỏ, tuy nhiên, trong khi phát triển các ứng dụng như vậy, những người sử dụng có nguy cơ phải đối mặt với sự rủi ro khi tạo ra các ứng dụng mang tính đặc thù lớn. Mặt khác, dữ liệu khó có thể được chia sẻ cho nhiều ứng dụng khác nhau. Nếu có nhu cầu chia sẻ dữ liệu, cơ sở dữ liệu của hệ này phải được lưu trữ trên một máy chủ cơ sở dữ liệu hay trong một mạng cục bộ.

b. Hệ cơ sở dữ liệu trung tâm

Trong các tổ chức lớn, dữ liệu mà hầu hết các ứng dụng đều có thể truy nhập tới được lưu trữ trên một máy tính trung tâm. Trong nhiều hệ thống, những người sử dụng từ xa có thể truy nhập tới cơ sở dữ liệu này thông qua các thiết bị đầu cuối và các móc nối truyền thông dữ liệu. Tuỳ thuộc vào quy mô của tổ chức, máy tính trung tâm này thường lưu trữ các cơ sở dữ liệu tích hợp rất lớn và được nhiều người sử dụng truy nhập. Việc sử dụng thường có cường độ lớn, hàng trăm giao dịch trong một giây đang cần được xử lý (một số hệ thống có thể hỗ trợ hàng nghìn giao dịch trong một giây). Một trong các ứng dụng điển hình của hệ cơ sở dữ liệu trung tâm là hệ thống đăng ký giữ chỗ trên máy bay, các hệ thống thông tin của các cơ quan tài chính….

c. Hệ cơ sở dữ liệu khách/chủ

Các máy tính trung tâm lớn thường rất đắt so với các máy tính nhỏ và các máy trạm. Do vậy, nhiều tổ chức đã giảm kích thước các ứng dụng để có thể cài đặt trên các máy tính nhỏ với giá thành thấp và hiệu quả hơn.

Các máy tính được nối với nhau trong một mạng cục bộ sao cho chúng có thể chia sẻ các tài nguyên như các máy in, máy quét, các thiết bị lưu trữ,...

Mục đích chính của một kiến trúc khách/chủ là cho phép các ứng dụng máy khách truy nhập dữ liệu được quản lý bởi máy chủ. Giao diện người sử dụng và các yêu cầu đặt ra đối với ứng dụng được xử lý trên máy khách, còn việc xử lý dữ liệu được thực hiện trên máy chủ cơ sở dữ liệu.

Các hệ thống khách/chủ (và các mạng cục bộ kết hợp) thường được sử dụng để hỗ trợ các tính toán theo một nhóm công việc, có nghĩa là việc sử dụng các tài nguyên tính toán để hỗ trợ ra quyết định và các ứng dụng khác bởi một nhóm người sử dụng. Một vài mạng như vậy có thể được móc nối với nhau sao cho các nhóm công việc khác nhau có thể chia sẻ công việc. Các hệ thống khách/chủ đang nhanh chóng phát triển thành các modun chuẩn cho các hệ thống tính toán quy mô toàn xí nghiệp tại nhiều tổ chức.

2.2. Các hệ cơ sở dữ liệu phân tán

Ngày nay, nhiều tổ chức phân bố trên nhiều vị trí khác nhua như các thành phố hay thậm chí các quốc gia khác nhau. Trong những trường hợp như vậy, việc xây dựng các

hệ cơ sở dữ liệu tập trung đối với các tổ chức này thường là không thực tế và không kinh tế.

Một cơ sở dữ liệu phân tán là một cơ sở dữ liệu logic đơn lẻ được phân bố về mặt vật lý trên nhiều máy tính ở nhiều vị trí địa lý khác nhau. Có hai kiểu chung nhất của các hệ cơ sở dữ liệu phân tán, đó là: hệ cơ sở dữ liệu phân tán thuần nhất và hệ cơ sở dữ liệu phân tán không thuần nhất.

a. Hệ cơ sở dữ liệu phân tán thuần nhất

Khi áp dụng đối với các hệ cơ sở dữ liệu, thuật ngữ thuần nhất có nghĩa là công nghệ về cơ sở dữ liệu là như nhau (hay ít nhất là có thể tương thích) tại mỗi một vị trí địa lý và dữ liệu tại các vị trí địa lý khác nhau cũng có thể tương thích.

Ví dụ, một cơ sở dữ liệu của một hệ thống được phân tán trên 3 nút sẽ tương ứng với 3 vị trí địa lý khác nhau. Đối với các hệ cơ sở dữ liệu phân tán thuần nhất này, các vấn đề sau đây có thể xảy ra:

- Các hệ điều hành máy tính tại mỗi một vị trí địa lý là như nhau hay ít nhất chúng có khả năng tương thích cao.
- Các mô hình dữ liệu được sử dụng tại mỗi một vị trí địa lý là như nhau (mô hình quan hệ được sử dụng chung nhất đối với các hệ cơ sở dữ liệu phân tán ngày nay).
- Các hệ quản trị cơ sở dữ liệu được sử dụng tại mỗi một vị trí địa lý là như nhau hay ít nhất chúng có khả năng tương tích cao.

- Dữ liệu tại các vị trí khác nhau có các định nghĩa và khuôn dạng chung

Các hệ cơ sở dữ liệu phân tán thuần nhất đơn giản hoá việc chia sẻ dữ liệu giữa những người sử dụng khác nhau. Ví dụ, với mô hình 3 nút trên, một người sử dụng tại vị trí A có thể dễ dàng tìm kiếm hay cập nhật dữ liệu được lưu tại các vị trí B và C.

Các cơ sở dữ liệu phân tán thuần nhất thể hiện một mục đích thiết kế đối với các hệ cơ sở dữ liệu phân tán. Cụ thể, các cơ sở dữ liệu phân tán thuần nhất được thiết kế bằng cách chia nhỏ một cơ sở dữ liệu thành nhiều cơ sở dữ liệu địa phương, các cơ sở dữ liệu địa phương định vị trên các trạm làm việc khác nhau nhưng chúng được biểu diễn bởi cùng một mô hình dữ liệu và được quản trị bởi cùng một hệ quản trị dữ liệu địa phương.

b. Hệ cơ sở dữ liệu phân tán không thuần nhất

Trong hầu hết các doanh nghiệp, các hệ cơ sở dữ liệu liên quan đến một chu kỳ dài không được chỉ đạo và lập kế hoạch cẩn thận. Các máy tính khác nhau và các hệ điều hành khác nhau có thể được sử dụng tại mỗi một vị trí địa lý. Các mô hình dữ liệu khác nhau và các hệ quản trị cơ sở dữ liệu khác nhau cũng có thể được lựa chọn sử dụng. Ví dụ, một ví trí có thể sử dụng công nghệ cơ sở dữ liệu quan hệ mới nhất, trong khi một vị trí khác có thể lưu trữ dữ liệu sử dụng các tệp truyền thống hay các cơ sở dữ liệu mạng, phân cấp cũ hơn.

Phức tạp hơn nữa, dữ liệu trên các vị trí thường không tương tích. Các mâu thuẫn điển hình bao gồm các khác biệt về cú pháp (sự biểu diễn khác nhau của các khoản mục dữ

liệu tại hai vị trí) và các khác biệt về ngữ nghĩa (các ngữ nghĩa khác nhau đối với cùng một khoản mục dữ liệu tại các vị trí khác nhau).

Sớm hay muộn thì những người sử dụng tại các vị trí khác nhau sẽ phát hiện ra rằng họ cần chia sẻ dữ liệu cho dù có sự không tương thích. Để giải quyết vấn đề này, giải pháp tỏ ra tương đối hữu hiệu là phát triển một cơ sở dữ liệu mới hoàn toàn hợp nhất được tất cả các hệ đang tồn tại. Tuy nhiên, giải pháp này không dễ thực hiện về mặt kỹ thuật cũng như kinh tế. Thay vào đó, đôi khi các cơ sở dữ liệu được móc nối với nhau và kết quả là tạo ra một tập các cơ sở dữ liệu không thuần nhất. Một hệ thống như vậy sẽ hạn chế các kiểu xử lý mà những người sử dụng có thể thực hiện. Ví dụ, một người sử dụng tại một vị trí có thể đọc nhưng không thể cập nhật dữ liệu tại một vị trí khác.

3. Quản trị hệ thống cơ sở dữ liệu

Để có thể hiểu rõ nội dung này, trước hết người đọc cần hiểu được các công việc cơ bản cần làm khi thực hiện công việc quản trị các hệ thống cơ sở dữ liệu. Các khái niệm cơ bản về các hệ thống cơ sở dữ liệu đã được trình bày ở các mục trên, ở mục này, chúng ta sẽ chỉ tập trung vào việc xác định vai trò và trách nhiệm của người quản trị hệ thống cơ sở dữ liệu.

Nhìn chung, về cơ bản công việc quản trị hệ thống của một hệ cơ sở dữ liệu ở các doanh nghiệp là giống nhau. Nhưng điều này sẽ không còn đúng nếu đem so sánh vai trò, trách nhiệm của người quản trị trên các doanh nghiệp có quy mô khác nhau. Ví dụ, việc quản trị chỉ một hệ thống bán hàng tại một cửa hàng sách sẽ nhẹ nhàng hơn nhiều so với việc

quản trị toàn bộ hệ thống cơ sở dữ liệu bao gồm các hệ thống tại các cửa hàng bán sách nhỏ. Có thể coi yếu tố kinh nghiệm và những hiểu biết về các hệ quản trị cơ sở dữ liệu của một doanh nghiệp như là những người thực hiện công việc quản trị các hệ thống cơ sở dữ liệu này.

Trước hết, phải xác định được vai trò và trách nhiệm của người quản trị hệ thống để từ đó sẽ quyết định các kỹ năng cần thiết để thực hiện các vai trò này. Ở đây, vai trò và trách nhiệm sẽ được chia ra làm 2 phạm trù chính: cơ bản và các phần khác. Tuy vậy, người đọc cần lưu ý rằng, trên thực tế, ranh giới phân chia giữa các phạm trù này là không rõ ràng mà chỉ mang tính chất tương đối.

Dưới đây sẽ chỉ nêu ra các vai trò và trách nhiệm xét ở mức tổng quát nhất, bởi vì trên thực tế, người quản trị có thể không cần thực hiện tất cả các nội dung này hoặc có thể sẽ phải làm nhiều hơn:.

- Cài đặt các phần mềm:Cài đặt các phiên bản mới của các phần mềm ứng dụng, các phần mềm liên quan đến việc quản trị cơ sở dữ liệu.Các phần mềm này cần được kiểm tra cẩn thận trước khi tiến hành cài đặt và đưa vào môi trường sử dụng chính thức.
- Cấu hình phần cứng, phần mềm cùng với người quản trị hệ thống:Trong rất nhiều trường hợp, các hệ thống phần mềm chỉ có thể truy cập, cài đặt với vai trò của người quản trị hệ thống. Khi đó, người quản trị hệ thống cơ sở dữ liệu cần làm việc cùng với người quản trị hệ thống trong suốt quá trình thực hiện cài đặt phần mềm và thực hiện cấu hình phần cứng, phần mềm sao cho hệ thống mới sẽ là môi trường hoạt động tối ưu cho hệ quản trị cơ sở dữ liệu.

- Quản trị về an ninh:Nội dung của phần này sẽ được nêu rõ ràng hơn trong các mục sau, ở đây chỉ nêu ra vấn đề ở mức tổng quát nhất. Một trong những trách nhiệm chính của người quản trị hệ thống cơ sở dữ liệu là theo dõi và quản lý an ninh các hệ quản trị cơ sở dữ liệu. Công việc này bao gồm: thêm, loại bỏ người dùng, kiểm soát dung lượng đĩa cứng dùng cho hệ quản trị cơ sở dữ liệu, kiếm soát truy nhập và khi cần, thực hiện các thao tác xử lý sự cố về an ninh hệ thống.
- Điều chỉnh và kiểm soát hiệu năng hoạt động của hệ quản trị cơ sở dữ liệu: Người quản trị hệ thống cơ sở dữ liệu cần luôn thực hiện theo dõi hiệu năng hoạt động của hệ cơ sở dữ liệu và sẵn sàng điều chỉnh lại hệ thống khi cần. Công việc này cần được thực hiện ngay cả đối với các hệ thống đã được xây dựng và hiệu chỉnh tốt. Ví dụ như thay đổi các thông số hay thiết lập lại các tệp chỉ mục, cấu trúc lại các bảng dữ liệu.
- Sao lưu và phục hồi dữ liệu: Đây có lẽ là công việc quan trọng bậc nhất của người quản trị hệ thống, bởi lẽ họ có trách nhiệm duy trì và đảm bảo hệ thống cơ sở dữ liệu hoạt động một cách thông suốt tại mọi thời điểm, đảm bảo tính sẵn sàng của dữ liệu. Để có thể thực hiện được điều này, người quản trị cơ sở dữ liệu cần xây dựng một chiến lược sao lưu và khôi phục dữ liệu đủ tin cậy, tiến hành thực hiện công việc này một cách thường xuyên. Trách nhiệm chính của người quản trị hệ thống cơ sở dữ liệu là đảm bảo sự sẵn sàng và toàn vẹn của cơ sở dữ liệu, vì vậy, quá trình

sao lưu và phục hồi dữ liệu cần phải được thực hiện và kiểm tra thường xuyên.

- Bảo trì định kỳ: Trách nhiệm của người quản trị hệ thống cơ sở dữ liệu cũng là lập lịch bảo trì các hệ thống cơ sở dữ liệu và thực hiện theo lịch đó. Thông thường, việc bảo trì định kỳ được thực hiện vào sáng sớm hoặc vào các ngày cuối tuần. Khi đó sẽ tránh được sự ảnh hưởng đến việc sử dụng, truy cập các thông tin lưu tại cơ sở dữ liệu của người sử dụng.
- Xử lý sự cố: Trong trường hợp hệ thống hay hệ quản trị cơ sở dữ liệu không hoạt động do hỏng hóc hay do các nguyên nhân khác, người quản trị cần thực hiện xử lý sự cố hoặc hỗ trợ xử lý sự cố. Họ cũng cần tham gia hoặc chịu trách nhiệm chính để tìm ra và loại bỏ tận gốc các vấn đề gặp phải.
- Phục hồi sau sự cố: Trong trường hợp hệ thống ngừng hoạt động do hỏng thì người chịu trách nhiệm chính trong việc đưa hệ thống trở lại hoạt động bình thường là người quản trị hệ thống cơ sở dữ liệu. Giải pháp tốt nhất là người quản trị cần có kế hoạch xử lý cho các sự kiện ngừng hoạt động bất thường của hệ thống. Khi đó, hệ thống sẽ rất nhanh chóng sẵn sàng hoạt động trở lại sau khi đã được xử lý sự cố.

Ngoài ra, người quản trị hệ thống cơ sở dữ liệu có thể sẽ phải thực hiện hoặc tham gia thực hiện các công việc sau trong doanh nghiệp ngoài các trách nhiệm chính nêu trên:

- Tham gia phân tích dữ liệu: Người quản trị hệ thống cơ sở dữ liệu có thể sẽ được yêu cầu thực hiện phân tích các dữ liệu lưu trữ trong cơ sở dữ liệu và từ đó,

đưa ra các khuyến nghị liên quan đến hiệu năng của việc lưu trữ đó. Điều này có thể liên quan đến những việc như: nâng cao hiệu quả sử dụng các chỉ mục, hay là việc sử dụng các câu lệnh (tùy chọn) làm tăng tốc độ thực hiện truy vấn.

- Tham gia thiết kế cơ sở dữ liệu: Người quản trị cũng có thể tham gia vào giai đoạn đầu khi tiến hành thiết kế cơ sở dữ liệu, với kinh nghiệm và chức năng của mình, người quản trị có thể đóng góp rất nhiều ý kiến hữu ích cho người thiết kế cơ sở dữ liệu trong các vấn đề như tạo lập các đối tượng dữ liệu, lưu trữ đối tượng dữ liệu, tối ưu hoá hiệu năng truy vấn,...
- Tham gia vào việc mô hình hóa dữ liệu và tối ưu hóa: Thông qua mô hình hóa dữ liệu, người quản trị cơ sở dữ liệu có thể tối ưu hóa được cách thức tổ chức hệ thống sao cho tận dụng được những ưu điểm của hệ thống vào/ra dữ liệu, từ đó có thể nâng hiệu suất phục vụ của cơ sở dữ liệu đối với các chương trình ứng dụng.
- Trợ giúp người phát triển: Người quản trị hệ thống cơ sở dữ liệu cũng được xem như là một nguồn lực có thể trợ giúp những người phát triển hệ thống và người sử dụng trong việc hỗ trợ xây dựng các câu lệnh SQL, các thủ tục được cài đặt trong cơ sở dữ liệu.
- Tham gia vào quá trình đề xuất ra các chuẩn trong doanh nghiệp: Với một hệ thống được phát triển và triển khai bởi nhiều nhóm phần mềm khác nhau, vai trò của người quản trị trong vấn đề đưa ra các chuẩn trong quá trình phân tích, thiết kế cơ sở dữ liệu cũng

như xây dựng phần mềm ứng dụng phải được quan tâm. Các chuẩn này có thể là quy ước đặt tên, quy ước sử dụng dữ liệu, quy ước sử dụng tài nguyên hệ thống,...

- Tham gia vào quá trình đánh giá và kiểm tra các phần mềm trong hệ thống: với vai trò là người phải đảm bảo được tính sẵn sàng và tin cậy của hệ quản trị cơ sở dữ liệu, người quản trị hệ thống có thể tham gia vào các quá trình như kiểm tra các phần mềm sẽ được cài đặt trên hệ quản trị cơ sở dữ liệu, các phiên bản nâng cấp tiếp theo,...
- Lập các tài liệu về môi trường hệ thống: Người quản trị hệ thống cơ sở dữ liệu cần lập các tài liệu về tất cả các khía cạnh của hệ thống, bao gồm: cấu hình phần cứng, các thời điểm bảo trì, các lần cập nhật, nâng cấp phần mềm, các thay đổi về ứng dụng và hệ quản trị cơ sở dữ liệu, và tất cả các mục liên quan đến những thay đổi của hệ thống. Các tài liệu này cần có khả năng sẵn sàng để có thể cung cấp cho người quản trị hệ thống cơ sở dữ liệu, những nhà phát triển hệ thống, người sử dụng, và những người có liên quan khi có yêu cầu.
- Đánh giá phần mềm cần mua mới: Công vệc này bao gồm việc tư vấn cho lãnh đạo về các phần mềm mới, bao gồm các phần mềm xây dựng mới hoặc các phiên bản tiếp theo sắp được đưa ra thị trường. Việc đánh giá này dựa trên cơ sở đảm bảo tính ổn định và tin cậy của toàn hệ thống khi đưa vào sử dụng các phần mềm này.

- Tham gia đánh giá các vấn đề về mua sắm nâng cấp các phần mềm cũng như phần cứng mới, đặc biệt là trong vấn đề đánh giá các chức năng và tính tương thích của của chúng với những thiết bị và phần mềm đã có.
- Lên kế hoạch và đề xuất khi thấy hệ thống cần thiết phải nâng cấp để đáp ứng các nhu cầu tiếp theo của doanh nghiệp. Đưa ra các quyết định liệu có cần mua thêm các hệ thống phần cứng, phần mềm mới nhằm đáp ứng được các nhu cầu ngày càng gia tăng về dữ liệu là công việc của người quản trị hệ thống cơ sở dữ liệu nhằm nâng cao khả năng đáp ứng, chất lượng dịch vụ của hệ thống cho người sử dụng.

3.1. Quản trị nội dung cơ sở dữ liệu

Trên đây, chúng ta đã làm quen với các khái niệm về công việc của những người quản trị hệ thống cơ sở dữ liệu. Thực chất, các công việc có liên quan đến nội dung cơ sở dữ liệu nêu trên có thể được phân chia thành các nhóm công việc chính như sau:

- Theo dõi cơ sở dữ liệu
- Hiệu chỉnh cơ sở dữ liệu
- Sao lưu và phục hồi dữ liệu

a. Theo dõi cơ sở dữ liệu

Theo dõi cơ sở dữ liệu *(Database Monitoring)* là một trong các công việc hết sức quan trọng khi thực hiện quản trị hệ thống cơ sở dữ liệu. Một hệ thống cơ sở dữ liệu cần đảm bảo khả năng phục vụ ổn định, thời gian trả lời các truy vấn cũng như các thao tác cập nhật dữ liệu tối thiểu. Tuy vậy,

sau một thời gian dài hoạt động, nếu không có sự hiệu chỉnh thì khả năng đáp ứng dịch vụ của hệ cơ sở dữ liệu có thể giảm đi, và do đó, chất lượng phục vụ cũng giảm theo. Việc theo dõi cơ sở dữ liệu giúp chúng ta có thể giảm thiểu các rủi ro do việc giảm chất lượng phục vụ của các cơ sở dữ liệu tới người sử dụng.

Trong thực tế, theo dõi cơ sở dữ liệu liên quan trực tiếp đến việc trả lời các câu hỏi:

- Máy chủ cơ sở dữ liệu có hoạt động hay không?
- Cơ sở dữ liệu có thể truy cập được không?
- Có thể thực hiện phục hồi dữ liệu và đảm bảo hệ thống hoạt động thông suốt khi xảy ra các sự cố làm hệ thống ngừng hoạt động hay không?

Đó là những câu hỏi cơ bản nhất mà một người quản trị hệ thống cơ sở dữ liệu cần phải trả lời. Nói cách khác là họ phải đảm bảo rằng hệ thống cơ sở dữ liệu đang hoạt động tốt. Việc theo dõi các cơ sở dữ liệu thực chất là việc theo dõi về trạng thái, các thông số kỹ thuật của cơ sở dữ liệu và các thông số liên quan đến các thông số kỹ thuật này.

- Theo dõi trạng thái

Trong quá trình hoạt động, việc đáp ứng đầy đủ các nhu cầu về sử dụng dữ liệu là rất quan trọng của các doanh nghiệp hiện nay. Các hệ thống có thể luôn phải sẵn sàng 24 giờ một ngày, 7 ngày trong tuần. Nếu hệ thống không thể đáp ứng nhu cầu, người sử dụng có thể quay sang sử dụng một hệ thống khác, của một công ty cạnh tranh khác chẳng hạn. Chính vì vậy, theo dõi trạng thái của cơ sở dữ liệu là rất cần thiết và thiết thực đối với các tổ chức.

Trạng thái của cơ sở dữ liệu bao gồm: tình trạng của máy chủ cơ sở dữ liệu có hoạt động hay không, cơ sở dữ liệu có truy cập được không, …

- Theo dõi các thông số kỹ thuật

Các thông số kỹ thuật cần thực hiện theo dõi bao gồm: hiện trạng sử dụng bộ nhớ của cơ sở dữ liệu, hiệu suất sử dụng đường truyền, các thủ tục chạy ngầm (*job*) theo định kỳ, dung lượng của cơ sở dữ liệu, dung lượng của các tệp chỉ mục, lượng bộ nhớ sử dụng,…

- Các thông số khác.

b. Hiệu chỉnh cơ sở dữ liệu

Việc theo dõi các thông tin trạng thái của cơ sở dữ liệu là rất cần thiết bởi lý do đảm bảo tính sẵn sàng cao phục vụ cho các nhu cầu về dữ liệu của người sử dụng. Trong thực tế, có thể xảy ra trường hợp sự cố gây ra bởi các tác nhân khách quan như: dung lượng đường truyền, số lượng người sử dụng, dung lượng bộ nhớ vật lý, dung lượng lưu trữ có thể, tính đúng đắn của dữ liệu ... làm cho hệ thống chậm phục vụ hay thậm chí người sử dụng bị từ chối dịch vụ. Trong các trường hợp này, người quản trị hệ thống cơ sở dữ liệu cần tìm ra được nguyên nhân gây ra sự cố và từ đó có thể đưa ra các hiệu chỉnh cần thiết nhằm đưa hệ thống trở lại hoạt động bình thường.

Các hiệu chỉnh đối với cơ sở dữ liệu có thể là:

Điều chỉnh các tham số khởi tạo của các cơ sở dữ liệu, bởi vì các tham số khởi tạo này có thể liên quan đến dung lượng lưu trữ có thể sử dụng cho cơ sở dữ liệu, lượng bộ nhớ mà cơ sở dữ liệu có thể sử dụng, ... ảnh hưởng đến khả

năng phục vụ của cơ sở dữ liệu; tăng cường số máy chủ phục vụ, giúp cho hệ thống được phân tải đều cho các yêu cầu từ phía người sử dụng để từ đó giảm thời gian trả lời tới người sử dụng; tăng lượng bộ nhớ cho máy chủ phục vụ, giúp nâng cao khả năng phục vụ cho cơ sở dữ liệu; thiết lập hay tạo lại các tệp chỉ mục cho các bảng dữ liệu, giúp hệ thống tìm kiếm các dữ liệu mà người sử dụng yêu cầu với thời gian trả lời nhanh nhất; kiểm tra và đảm bảo các ràng buộc dữ liệu, giúp cho dữ liệu được đảm bảo đúng đắn ...

Trong các điều kiện cụ thể, theo các nguyên nhân do người quản trị hệ thống cơ sở dữ liệu phát hiện được thông qua theo dõi cơ sở dữ liệu, các phản hồi từ người sử dụng, các sự cố có thể được phát hiện sớm và có các giải pháp cụ thể xử lý triệt để cũng như nâng cao hiệu năng phục vụ cho toàn hệ thống. Tuy vậy, chiến lược đúng đắn nhất vẫn là lập kế hoạch chi tiết, cụ thể các giải pháp để có thể xử lý các sự cố thường gặp, theo dõi và phát hiện các nguy cơ tiềm tàng có thể dẫn đến các sai sót hay sự cố về dữ liệu. Điều này yêu cầu người quản trị hệ thống cơ sở dữ liệu phải thường xuyên cập nhật các thông tin về kỹ thuật và luôn theo dõi sát sao các thay đổi của hệ thống.

c. Sao lưu và phục hồi dữ liệu

Như đã nêu trên, người quản trị hệ thống cơ sở dữ liệu cần đảm bảo hoạt động của hệ thống cơ sở dữ liệu thông suốt, nói một cách khác là cơ sở dữ liệu luôn hoạt động nhằm đáp ứng các nhu cầu về dữ liệu trong quá trình tác nghiệp của cơ quan, tổ chức. Trong thực tế, các sự cố như: mất điện, quá tải, virus, hỏng đĩa cứng lưu trữ hay thậm chí là các xâm nhập hệ thống bất thường với mục tiêu phá hoại có thể làm hệ thống cơ sở dữ liệu bị đình trệ hay không thể

phục vụ nhu cầu của người sử dụng. Nguyên nhân của việc không thể phục vụ được tại một thời điểm nào đó có thể là do quá tải đường truyền hoặc dữ liệu bị hư hại do các tác nhân ngoài ý muốn nêu trên.

Để có thể khắc phục các sự cố gây ra tình trạng từ chối dịch vụ, người quản trị có thể thực hiện tăng cường các biện pháp an ninh, bảo mật hệ thống. Nhưng đối với trường hợp dữ liệu bị mất mát, hư hỏng thì trách nhiệm của người quản trị hệ thống cơ sở dữ liệu là cần có biện pháp phục hồi lại trạng thái hoạt động của hệ thống như tại thời điểm ngay trước khi hệ thống bị ngừng hoạt động. Để làm được điều này cần lập các bản sao đầy đủ của các cơ sở dữ liệu, để khi cần có thể nhanh chóng phục hồi lại hệ thống cơ sở dữ liệu, đảm bảo tính sẵn sàng cho hệ thống.

Các hệ quản trị cơ sở dữ liệu tiên tiến thường cung cấp các công cụ rất trực quan, tiện dụng để người quản trị hệ thống cơ sở dữ liệu có thể dễ dàng thực hiện các thao tác này. Vì vậy, một người quản trị hệ thống cơ sở dữ liệu cần lập cho doanh nghiệp của mình một chiến lược cụ thể và thực hiện kế hoạch này thường xuyên theo một định kỳ rõ ràng tạo lập các bản sao đầy đủ của các cơ sở dữ liệu để khi có sự cố có thể nhanh chóng đưa hệ thống cơ sở dữ liệu vào hoạt động trở lại.

3.2. Quản trị kỹ thuật

Vấn đề về quản trị kỹ thuật cũng là một chủ đề quan trọng khi nói đến các vấn đề về quản trị cơ sở dữ liệu vì một hệ thống là tổng thể hài hòa của các thành phần tạo nên chúng. Nhưng do đặc thù và tính quan trọng của dữ liệu luôn được đặt lên hàng đầu, đôi khi vấn đề về quản trị kỹ thuật có thể

bị xem nhẹ. Nói một cách đơn giản là hệ thống cơ sở dữ liệu cần hoạt động trên một môi trường ổn định, và có khả năng đáp ứng chính xác mọi nhu cầu xuất phát từ người sử dụng.

Các yếu tố kỹ thuật chính mà chúng ta cần quan tâm, đó là:

- Cài đặt, cấu hình hệ thống
- Đảm bảo về đường truyền, mạng
- Quản trị tài nguyên cơ sở dữ liệu

Chúng ta sẽ lần lượt đi vào từng nội dung nêu trên để hiểu được các công việc này thực chất là phải làm những gì.

a. Cài đặt, cấu hình hệ thống

Như đã nêu trong các mục trước đây, quản trị hệ thống cơ sở dữ liệu liên quan đến việc phải theo dõi sát sao quá trình cài đặt các phần cứng, phần mềm cùng với người quản trị hệ thống. Việc cấu hình hệ thống cơ sở dữ liệu cần có một hiểu biết sâu sắc về các hệ quản trị cơ sở dữ liệu, các cơ sở dữ liệu và các nhân tố ảnh hưởng đến khả năng hoạt động của các cơ sở dữ liệu.

Việc thiết lập các thông số cấu hình của các cơ sở dữ liệu cần tuân thủ sát sao các khuyến cáo từ tài liệu kỹ thuật do những người phát triển hệ thống cung cấp. Các thông số này có thể là các khuyến cáo về dung lượng bộ nhớ sử dụng, dung lượng lưu trữ tối thiểu, các thông số khởi tạo hệ thống cơ sở dữ liệu,... Trong quá trình đưa hệ thống cơ sở dữ liệu vào phục vụ tác nghiệp, người quản trị hệ thống cơ sở dữ liệu có thể cần thực hiện cài đặt hệ thống từ đầu dựa trên các tài liệu do người phát triển hệ thống cung cấp đảm bảo môi trường tối ưu cho khai thác, cập nhật dữ liệu.

b. Đảm bảo về đường truyền, mạng

Trong thực tế, công việc của người quản trị hệ thống cơ sở dữ liệu có thể không liên quan trực tiếp đến vấn đề về đường truyền, mạng. Vì vậy, người quản trị hệ thống cơ sở dữ liệu cần làm việc sát sao cùng với người quản trị mạng để có thể đảm bảo được dung lượng đương truyền có thể đáp ứng tối đa các nhu cầu dữ liệu của người sử dụng.

Nguyên nhân là do môi trường tính toán hiện tại thường được tổ chức theo mô hình từ hai cho đến ba lớp: client / server hay client / middle tier / server nên các hệ thống chắc chắn phải hoạt động trên môi trường mạng. Khi đó, việc đảm bảo các truy cập của người sử dụng đến dữ liệu của mình thông qua các hệ thống mạng là hết sức cần thiết. Đôi khi, người quản trị hệ thống cơ sở dữ liệu cần phải trang bị cho mình các kiến thức cơ bản về đường truyền và mạng nhằm phát hiện, giải quyết các sự cố có thể xảy ra trong quá trình sử dụng cơ sở dữ liệu trong môi trường tác nghiệp.

c. Quản trị tài nguyên cơ sở dữ liệu

Các hệ thống cơ sở dữ liệu trong thực tế là một trong những tài sản sở hữu bởi các cơ quan, tổ chức. Nó không chỉ bao gồm các thông tin về khách hàng mà chủ yếu còn về các dữ liệu tác nghiệp trong cơ quan hoặc tổ chức đó. Bảo đảm tài nguyên này luôn sẵn sàng cho người sử dụng hệ thống là một công việc khá nặng nề đối với những người quản trị hệ thống cơ sở dữ liệu.

3.3. Đảm bảo an ninh, phân quyền truy nhập

Như đã nêu, nếu hệ thống bị ngừng trệ hoặc tốc độ phục vụ quá chậm sẽ dẫn đến hậu quả không tốt cho hoạt động tác nghiệp của các cơ quan, tổ chức. Trong thực tế, có thể có các truy cập không hợp lệ nhằm các ý đồ xấu, phá hoại dẫn đến các hậu quả nghiêm trọng như: mất cắp thông tin, mất, hỏng dữ liệu... Để có thể tránh được tình trạng nêu trên, người quản trị hệ thống cơ sở dữ liệu cần có các biện pháp bảo vệ dữ liệu, loại bỏ và từ chối các truy cập trái phép từ đó đảm bảo tính sẵn sàng và tính an toàn cho hệ thống cơ sở dữ liệu.

Các sự cố có thể xảy ra bao gồm việc mất mát dữ liệu, dữ liệu bị làm sai lệch so với nguyên bản, người sử dụng truy cập các chức năng, các dữ liệu không được phép có thể gây hậu quả nghiêm trọng. Biện pháp để giải quyết các vấn đề này là:

- Thiết lập cơ chế kiểm soát các truy nhập vào cơ sở dữ liệu của người sử dụng trong nội bộ cơ quan, tổ chức
- Phân quyền sử dụng các đối tượng dữ liệu cho từng người dùng khác nhau
- Có thể hạn chế quyền truy cập cơ sở dữ liệu của người sử dụng đến tận từng nội dung dữ liệu cụ thể
- Kết hợp với quản trị mạng để có cơ chế kiểm soát người sử dụng đăng nhập hệ thống của các nhân viên trong nội bộ cơ quan
- Cùng người quản trị hệ thống thiết lập các biện pháp bảo vệ mạng bằng cách sử dụng các hệ thống kiểm soát các truy nhập trái phép từ bên ngoài hệ thống

như: tường lửa (firewall), định kỳ quét và kiểm tra virus, cập nhật các bản vá lỗi an ninh, bảo mật cho các hệ điều hành, ...

3.4. Một số lỗi thường gặp và cách khắc phục

Trong mục này, chúng tôi sẽ trình bày một số lỗi thường gặp trong quá trình vận hành và khai thác cơ sở dữ liệu, các nguyên nhân chính có thể dẫn đến gây lỗi và cách khắc phục:

a. Không thể truy cập được cơ sở dữ liệu

Lỗi này thông thường là do các nguyên nhân: hệ thống mạng không được kích hoạt tại máy khách, hệ thống mạng không hoạt động, hệ quản trị cơ sở dữ liệu không hoạt động, hệ thống bị quá tải, lỗi do chương trình ứng dụng, … Các nguyên nhân nêu trên có độ ưu tiên từ trái qua phải, người quản trị hệ thống cần thực hiện kiểm tra tất cả các lý do nêu trên có thể gây ra tình trạng cơ sở dữ liệu không thể truy cập được.

Nếu hệ thống mạng không được kích hoạt tại máy khách có nghĩa là tất cả các giao dịch khác của máy khách với các máy tính còn lại trong mạng sẽ không thể thực hiện được. Thông thường, để kiểm tra lỗi này, người quản trị có thể sử dụng lệnh ‘ping.exe’, một tiện ích có sẵn của Windows để kiểm tra kết nối mạng giữa máy khách và máy chủ cơ sở dữ liệu.

Nếu hệ thống mạng của toàn bộ cơ quan không hoạt động cũng có nghĩa là chúng ta không thể thực hiện một giao tác mạng nào. Trong trường hợp nghi ngờ, hãy cùng với quản trị mạng kiểm tra sự hoạt động của toàn bộ hệ thống. Đừng

bỏ qua những chi tiết nhỏ, chẳng hạn như nguồn cấp điện cho thiết bị kết nối tập trung: hub, switch.

Nếu hệ thống mạng hoạt động tốt và chúng ta có thể thực hiện các giao dịch mạng với máy chủ cơ sở dữ liệu thì lỗi có thể là do các dịch vụ (service) của các hệ quản trị cơ sở dữ liệu này không được cho phép hoạt động. Các hệ quản trị cơ sở dữ liệu chạy trên các máy chủ thường có các dịch vụ chạy ngầm đóng vai trò cầu nối giữa các máy trạm và cơ sở dữ liệu. Ví dụ điển hình là các hệ quản trị cơ sở dữ liệu Oracle, MS SQL Server. Nguyên nhân này có thể xuất hiện sau trường hợp bị mất điện bất ngờ làm cho hệ thống máy chủ không thể kích hoạt, khởi động các dịch vụ cho hệ quản trị cơ sở dữ liệu. Để khắc phục, dựa vào các tài liệu hướng dẫn sử dụng thực hiện các bước khởi động các dịch vụ cho hệ quản trị cơ sở dữ liệu và đừng quên kiểm tra lại sự hoạt động của hệ thống.

Trong trường hợp tất cả các dịch vụ của hệ quản trị cơ sở dữ liệu đều đã hoạt động, lý do không truy cập được cơ sở dữ liệu đã được loại bỏ, vấn đề còn lại có thể nằm ở các chương trình ứng dụng, các thông số cấu hình khác của cơ sở dữ liệu. Hãy yêu cầu đối tác hoặc nhóm phát triển các ứng dụng kiểm tra hệ thống của họ.

b. Hệ thống báo lỗi timeout (chậm quá thời gian cho phép)

Khi hệ thống báo lỗi ‘timeout’, có nghĩa là nó có dấu hiệu của sự quá tải, hoặc do lỗi của đường truyền. Thông thường, các giao tác mạng được thực hiện theo kiểu: Gửi – Chờ – Nhận. Có nghĩa là các tiến trình thường phải chờ một khoảng thời gian trước khi nhận dữ liệu. Nếu thời gian chờ

dài hơn giới hạn cho phép, hệ thống tự động báo lỗi quá hạn (timeout). Giải pháp cho trường hợp này là kiểm tra tình trạng cơ sở dữ liệu, kiểm tra thời gian thực hiện các giao tác mạng.

Kiểm tra tình trạng cơ sở dữ liệu cho phép người quản trị hệ thống cơ sở dữ liệu có thể nắm được các thông tin về bộ nhớ sử dụng, dung lượng đĩa cứng, số lượng các giao dịch và lượng bộ nhớ mà các giao dịch có thể đang chiếm giữ. Các thông tin này rất có ích trong việc kiểm tra khả năng đáp ứng các giao dịch đến cơ sở dữ liệu từ người sử dụng. Từ đó có thể đưa ra các giải pháp khắc phục bằng cách hiệu chỉnh các thông số cơ sở dữ liệu, giải phóng bớt bộ nhớ từ các chương trình không cần thiết, dọn dẹp đĩa cứng, tăng dung lượng vùng swap cho hệ điều hành,… Đồng thời, xem xét khả năng tối ưu tài nguyên sử dụng, tăng tốc độ truy vấn của các câu lệnh SQL mà người sử dụng thực hiện trên cơ sở dữ liệu.

c. Hệ thống báo lỗi Out of Memory (tràn bộ nhớ)

Trong các hệ thống máy chủ hiện đại, hiện tượng tràn bộ nhớ hiếm khi xảy ra. Tuy nhiên, đối với các hệ thống cũ có thể xảy ra tình trạng này do các hệ quản trị cơ sở dữ liệu đôi khi chiếm lượng bộ nhớ rất lớn khiến cho hệ thống không thể đáp ứng được. Giải pháp là nên tăng cường thêm RAM, xem xét mua thêm ổ đĩa cứng mới hoặc nghiên cứu khả năng nâng cấp toàn bộ hệ thống của máy chủ. Tuy nhiên việc tăng RAM, tăng dung lượng ổ đĩa cứng chỉ có thể là giải pháp tạm thời. Với lượng dữ liệu ngày càng lớn theo thời gian thì giải pháp triệt để nhất vẫn là nghiên cứu chuyển cơ sở dữ liệu sang một hệ thống mới ổn định và mạnh hơn.

d. Không thể cung cấp được dữ liệu

Nguyên nhân không thể cung cấp được dữ liệu phần lớn là do một trong các trường hợp nêu trên. Tuy nhiên, cũng có trường hợp các bảng dữ liệu, các khung nhìn (view) hoặc các thủ tục định nghĩa sẵn bởi người sử dụng bị mất do cài đặt lại hoặc do các bảng dữ liệu chính của cơ sở dữ liệu có dung lượng quá lớn gây ra hiện tượng chậm cung cấp dữ liệu. Khi đó, người quản trị hệ thống cơ sở dữ liệu cần thực hiện các biện pháp kiểm tra tổng hợp như đã nêu trong các phần trên, đồng thời thực hiện công việc bảo trì hệ thống cơ sở dữ liệu. Cần nhớ một nguyên tắc bất di bất dịch là trước khi thực hiện bất kỳ thao tác bảo trì nào cần thực hiện sao lưu đầy đủ hệ thống cơ sở dữ liệu hiện tại như đã nêu trong phần "Sao lưu và phục hồi dữ liệu".

e. Virus

Các virus và biến thể của chúng ngày nay đang là những mối nguy hiểm đe dọa sự an toàn và ổn định của hệ thống mạng cũng như các hệ thống cơ sở dữ liệu. Điều này đặc biệt nguy hiểm khi các hệ cơ sở dữ liệu ngày nay không chỉ phục vụ cho riêng một doanh nghiệp nào. Virus có thể gây ra các hiện tượng tắc nghẽn đường truyền, mất dữ liệu, dữ liệu bị phá hủy, … và do đó hệ thống cơ sở dữ liệu sẽ bị ảnh hưởng đến khả năng cung cấp dữ liệu của mình.

Do sự nguy hiểm ngày càng tăng của virus, người quản trị hệ thống cơ sở dữ liệu cần thực hiện thường xuyên các thao tác sao lưu toàn bộ cơ sở dữ liệu theo định kỳ một cách nghiêm ngặt, đồng thời thường xuyên cập nhật các bản vá lỗi của hệ điều hành, hệ quản trị cơ sở dữ liệu mỗi khi các hãng cung cấp các hệ quản trị cơ sở dữ liệu thông báo, cập

nhật các phiên bản phần mềm chống virus mới nhất. Điều này nêu ra để nhấn mạnh đến tầm quan trọng của việc sử dụng các phần mềm có bản quyền và do đó có thể nhận được các hỗ trợ về kỹ thuật từ các nhà cung cấp sản phẩm, cũng như luôn đảm bảo hệ thống hiện tại là mới nhất và có khả năng ứng phó kịp thời đối với các nguy cơ khiến cho hệ thống mất ổn định.

4. Kết luận

Trong chủ đề này, chúng tôi đã giới thiệu các vấn đề liên quan đến việc quản trị đối với một hệ thống cơ sở dữ liệu. Các vấn đề chính được đề cập đến trong chủ đề này, đó là: hiểu và nắm vững các vai trò và trách nhiệm đối với một người quản trị hệ thống cơ sở dữ liệu; hiểu và nắm được các vấn đề liên quan đến quản trị cơ sở dữ liệu về mặt nội dung, kỹ thuật, an toàn, bảo mật dữ liệu. Một số lỗi thường gặp đối với cơ sở dữ liệu chúng tôi cũng đã trình bày, giúp người quản trị đỡ bỡ ngỡ và có thêm kinh nghiệm khi phải đảm nhận vai trò quản trị hệ thống cơ sở dữ liệu.

Đối với những hệ quản trị cơ sở dữ liệu lớn, ngoài việc cung cấp các công cụ quản trị qua giao diện người sử dụng thì đối với một người quản trị nhiều kinh nghiệm, việc sử dụng các câu lệnh sẽ cho phép họ có thể can thiệp sâu hơn vào hệ thống.

Trong chủ đề tiếp theo, chúng tôi sẽ tập trung vào việc trình bày một số hệ quản trị cơ sở dữ liệu phổ biến ở Việt Nam hiện nay.

5. Bài tập

1. Liệt kê các vai trò và trách nhiệm đối với một người quản trị hệ thống cơ sở dữ liệu.

2. Tìm và đọc thêm tài liệu sau: System Administration for Microsoft SQL Server

3. Tìm và đọc thêm tài liệu sau: Oracle Administrator's Guide, Oracle Administrator's Guide for Windows.

4. Tìm hiểu công cụ Enterprise Manager của Microsoft SQL Server

5. Tìm hiểu công cụ Oracle Enterprise Manager của Oracle

Chủ đề IV: Một số hệ quản trị cơ sở dữ liệu phổ biến

Mục tiêu:

Trong chủ đề này, cần phải nắm được các vấn đề sau:

- Kiến trúc và các thành phần của một hệ quản trị cơ sở dữ liệu.
- Các hệ quản trị cơ sở dữ liệu cỡ vừa và nhỏ: Foxpro, Access
- Các hệ quản trị cơ sở dữ liệu cỡ lớn SQL Server, Oracle
- Lựa chọn hệ quản trị cơ sở dữ liệu cho phù hợp đối với những bài toán thực tế.
- Thực hành thiết kế sơ bộ một cơ sở dữ liệu đã được trình bày trong các chủ đề I và II trên hệ quản trị cơ sở dữ liệu SQL Server.

1. Giới thiệu chung

Như chúng ta đã biết, cơ sở dữ liệu là một tập các dữ liệu được tổ chức. Để thực hiện các thao tác như chèn, sửa, xoá và tìm kiếm dữ liệu, cơ sở dữ liệu cần phải được quản lý bởi một phần mềm quan trọng, phần mềm này thường được gọi là hệ quản trị cơ sở dữ liệu (Database Management System - DBMS). Mục đích chính của một hệ quản trị cơ sở dữ liệu đó là cho phép người sử dụng lưu trữ, cập nhật và tìm kiếm dữ liệu thông qua các thuật ngữ trừu tượng, do

đó có thể dễ dàng bảo trì và quản lý thông tin bằng cơ sở dữ liệu. Nhờ hệ quản trị cơ sở dữ liệu, người sử dụng không phải quan tâm đến các vấn đề liên quan đến việc biểu diễn dữ liệu ở mức vật lý cũng như các thuật toán lưu trữ, cập nhật và tìm kiếm và trả về dữ liệu.

Một hệ quản trị cơ sở dữ liệu thường là một gói phần mềm rất lớn thực hiện rất nhiều các công việc khác nhau như cung cấp các công cụ cho người sử dụng có thể truy nhập và sửa đổi thông tin trong cơ sở dữ liệu chẳng hạn. Một cơ sở dữ liệu là một kết nối trung gian giữa cơ sở dữ liệu vật lý, phần cứng, hệ điều hành, và với người sử dụng. Để đáp ứng các nhu cầu sử dụng của các loại người dùng khác nhau, ngoài việc tạo ra các công cụ sử dụng khác nhau, một hệ quản trị cơ sở dữ liệu thường cung cấp thêm một hoặc một vài ngôn ngữ lập trình chuyên dụng thường được gọi là ngôn ngữ cơ sở dữ liệu. Các hệ quản trị cơ sở dữ liệu khác nhau cung cấp các ngôn ngữ cơ sở dữ liệu khác nhau, mặc dù chúng hầu như đều dựa trên ngôn ngữ chuẩn SQL.

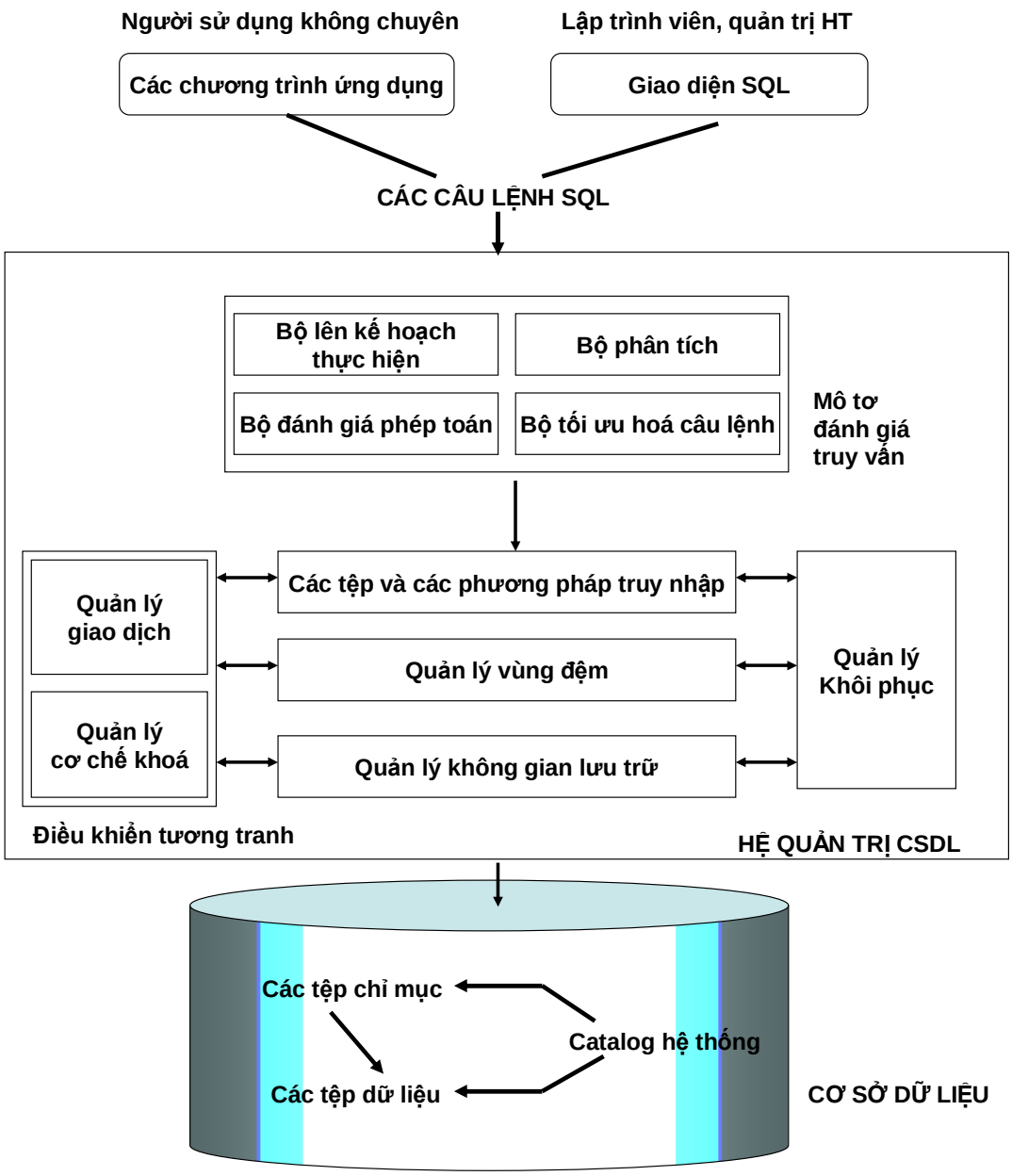
Ngày nay, trên thị trường các hệ quản trị cơ sở dữ liệu thương mại hầu hết là các hệ quản trị cơ sở dữ liệu quan hệ được xây dựng dựa trên mô hình dữ liệu quan hệ. Phiên bản đầu tiên của một hệ quản trị cơ sở dữ liệu quan hệ đó là hệ phần mềm lưu trữ dữ liệu quan hệ Multics, nó được tung ra thị trường lần đầu tiên vào năm 1978. Các hệ quản trị cơ sở dữ liệu quan hệ tiếp theo lần lượt được đưa ra, đó là Berkeley Ingres QUEL và IBM BS12.

Trong chủ đề này, chúng tôi sẽ trình bày những hệ quản trị cơ sở dữ liệu sau: Access, Visual Foxpro, SQL Server, Oracle, chúng đều là các hệ quản trị cơ sở dữ liệu quan hệ được thương mại hoá cao và rất phổ biến trên thị trường

Việt Nam cũng như trên thế giới. Tuy nhiên, trước tiên chúng ta hãy tìm hiểu kiến trúc đối với một hệ quản trị cơ sở dữ liệu.

2. Hệ quản trị cơ sở dữ liệu

2.1. Kiến trúc của một hệ quản trị cơ sở dữ liệu



Hình 4.2.1.1 Kiến trúc của một hệ quản trị CSDL

Hình 4.2.1.1 chỉ ra các thành phần chủ yếu của một hệ quản trị cơ sở dữ liệu. Tại đáy của kiến trúc, chúng ta thấy một

sự biểu diễn của các thiết bị lưu trữ cơ sở dữ liệu. Chú ý rằng thành phần này không chỉ chứa các dữ liệu được lưu trữ trong cơ sở dữ liệu mà còn bao gồm cả các siêu dữ liệu, tức là thông tin về cấu trúc của cơ sở dữ liệu cũng như các tệp chỉ mục dữ liệu.

Trong hình 4.2.1.1 chúng ta cũng thấy một bộ quản lý không gian lưu trữ mà nhiệm vụ của nó là lấy ra các thông tin được yêu cầu từ những thiết bị lưu trữ dữ liệu và thay đổi những thông tin này khi được yêu cầu bởi các mức trên nó trong hệ thống (chúng ta sẽ xem xét ký hơn thành phần này trong mục lưu trữ dữ liệu trong một hệ quản trị cơ sở dữ liệu dưới đây).

Thành phần tiếp theo là mô tơ đánh giá truy vấn hay bộ xử lý câu hỏi. Nhiệm vụ của nó là tìm ra phương án tốt nhất cho một thao tác được yêu cầu và phát ra các lệnh cho bộ quản lý lưu trữ để thực thi thao tác đó (chúng ta sẽ xem xét kỹ hơn thành phần này trong mục truy vấn trong một hệ quản trị cơ sở dữ liệu dưới đây).

Bộ quản trị giao dịch có trách nhiệm đảm bảo tính toàn vẹn của hệ thống. Kết hợp với bộ quản lý cơ chế khoá và quản lý khôi phục, đảm bảo các thao tác được thực hiện theo đúng thứ tự (quản lý tương tranh) và hệ thống sẽ không bị mất dữ liệu thậm chí khi có lỗi xảy ra. Bộ quản trị giao dịch cũng tương tác với bộ quản lý lưu trữ bởi việc xử lý giao dịch thường kéo theo việc lưu trữ nhật ký các thay đổi đối với dữ liệu để khi gặp lỗi, các thay đổi chưa được ghi vào đĩa có thể được thực hiện lại hoặc khôi phục lại.

Tại đỉnh của kiến trúc, chúng ta thấy các kiểu thao tác đối với hệ quản trị cơ sở dữ liệu:

- Thông qua một giao diện truy vấn chung: mỗi một hệ quản trị cơ sở dữ liệu thường cho phép người sử dụng gõ trực tiếp các truy vấn SQL thông qua một giao diện có sẵn. Thông thường giao diện này thường được thể hiện dưới dạng một số công cụ tích hợp trực tiếp vào hệ quản trị cơ sở dữ liệu và thường được dành cho những người sử dụng chuyên sâu, hiểu biết hệ thống.
- Thông qua các chương trình ứng dụng: Một hệ quản trị cơ sở dữ liệu điển hình cho phép những người lập trình viết các chương trình ứng dụng gọi đến hệ quản trị cơ sở dữ liệu này và truy vấn dữ liệu trong cơ sở dữ liệu. Các truy vấn được đưa ra thông qua một giao diện đặc biệt có thể chứa các hộp thoại. Một người sử dụng không thể đưa ra các câu truy vấn tuỳ ý thông qua giao diện này, nhưng nói chung sẽ dễ dàng hơn đối với họ khi sử dụng giao diện này so với việc phải viết một truy vấn trực tiếp bởi SQL.

Trên đây chúng ta đã có một cái nhìn tổng quát về kiến trúc của một hệ quản trị cơ sở dữ liệu chung. Phần tiếp theo chúng ta sẽ tìm hiểu xem một hệ quản trị cơ sở dữ liệu lưu trữ dữ liệu như thế nào cũng như các truy vấn được thực hiện ra sao.

2.2. Lưu trữ dữ liệu trong một hệ quản trị cơ sở dữ liệu

Các cơ sở dữ liệu thường đòi hỏi một khối lượng lớn không gian lưu trữ. Các cơ sở dữ liệu hợp nhất thường được đo theo các đơn vị gigabyte hay thậm chí terabyte. Do bộ nhớ

chính của máy tính không thể lưu trữ nhiều dữ liệu như vậy, nên các dữ liệu này thường được lưu trên các thiết bị nhớ ngoài như đĩa từ, băng từ, đĩa quang,…. Trong quá trình làm việc với cơ sở dữ liệu, dữ liệu được chuyển giữa thiết bị lưu trữ ngoài và bộ nhớ chính. Sự di chuyển dữ liệu này thường là khá chậm so với tốc độ của bộ xử lý trung tâm, do vậy các hệ quản trị cơ sở dữ liệu thường phải tổ chức lưu trữ dữ liệu vật lý sao cho tối thiểu hoá số yêu cầu chuyển đổi dữ liệu giữa thiết bị lưu trữ ngoài và bộ nhớ chính.

Như vậy, việc lưu trữ dữ liệu trong một hệ quản trị cơ sở dữ liệu được thực hiện thông qua bộ quản lý lưu trữ. Đây thực chất là một môđun chương trình cung cấp giao diện giữa các dữ liệu được lưu trữ trong cơ sở dữ liệu mức thấp và các chương trình ứng dụng hay các truy vấn được thực hiện bởi hệ thống. Bộ quản lý lưu trữ có trách nhiệm tương tác với các bộ quản lý tệp và các phương pháp truy nhập. Dữ liệu gốc được lưu trữ trên thiết bị lưu trữ ngoài sử dụng hệ thống xử lý tệp thường được cung cấp bởi một hệ điều hành nào đó. Bộ quản lý lưu trữ dịch các câu lệnh của ngôn ngữ SQL thành các lệnh hệ thống xử lý tệp ở mức thấp. Do vậy, bộ quản lý lưu trữ có trách nhiệm lưu trữ, tìm kiếm và cập nhật dữ liệu trong cơ sở dữ liệu.

2.3. Truy vấn trong một hệ quản trị cơ sở dữ liệu

Hệ quản trị cơ sở dữ liệu nhận các câu lệnh SQL thông qua các giao diện người dùng, lên kế hoạch đánh giá câu lệnh, thực hiện các câu lệnh này trên cơ sở dữ liệu và trả lại kết quả cho người sử dụng.

Khi một người sử dụng đưa ra một câu lệnh truy vấn, bộ tối ưu hoá truy vấn sẽ tiến hành phân tích câu lệnh bằng cách sử dụng các thông tin như dữ liệu được lưu như thế nào trong cơ sở dữ liệu, các chỉ mục nào nên được sử dụng, phần nào sẽ được thực hiện trước phần nào sẽ được thực hiện sau.,… để có thể đưa ra được một phương án thực hiện hiệu quả nhất trong khả năng có thể cho câu lệnh đó. Một phương án thực hiện là một kế hoạch cụ thể để đánh giá một câu hỏi, và nó thường được biểu diễn dưới dạng một cây các phép toán quan hệ. Thông qua phương án này, câu lệnh sẽ được chuyển tiếp xuống lớp tiếp theo để thực hiện. Sau khi được thực hiện xong, nếu dữ liệu được trả về, chúng sẽ được đưa trở lại cho người sử dụng đã đưa ra câu lệnh truy vấn đó.

3. Một số hệ thống quản trị cơ sở dữ liệu thông dụng

Trong phần này chúng tôi sẽ trình bày bốn hệ quản trị cơ sở dữ liệu phổ biến nhất hiện nay, xuất phát từ những hệ quản trị cơ sở dữ liệu cỡ vừa và nhỏ như Access, Foxpro đến những hệ quản trị cơ sở dữ liệu cớ lớn như SQL Server, Oracle. Trong mỗi hệ quản trị cơ sở dữ liệu, chúng tôi cố gắng trình bày các bước khi tiến hành thiết kế các đối tượng dữ liệu cho một cơ sở dữ liệu.

3.1. Các hệ quản trị cơ sở dữ liệu cỡ vừa và nhỏ

a. Hệ quản trị cơ sở dữ liệu Access

Microsoft Access là một hệ quản trị cơ sở dữ liệu quan hệ của hãng Microsoft chạy trên nền của hệ điều hành

Microsoft Windows, được đóng gói trong cùng bộ sản phẩm Microsoft Office. Nó được xây dựng chủ yếu nhằm phục vụ cho những người sử dụng không chuyên, không có nhiều kinh nghiệm lập trình nhưng vẫn có thể xây dựng được cho mình những ứng dụng client/server dựa trên cơ sở dữ liệu.

Một trong những lợi điểm chính của Access từ khía cạnh của những người thiết kế, lập trình đó là khả năng tương thích với các truy vấn SQL, câu lệnh SQL thân thiện, dễ dùng. Mặt khác, người sử dụng có thể sử dụng VBA để tạo lập các form màn hình nhập liệu cũng như khai thác cơ sở dữ liệu. Access cũng cho phép chúng ta tạo lập các báo cáo nghiệp vụ không quá phức tạp.

Access thường được sử dụng cho các doanh nghiệp nhỏ và những người lập trình nghiệp dư để tạo lập các hệ thống dễ thay đổi nhằm đáp ứng các công việc nhỏ, đơn giản. Ngoài ra nó cũng có khả năng truy nhập đến các cơ sở dữ liệu lớn hơn như Oracle, SQL Server.

Tuy nhiên, Access không phải là một hệ quản trị cơ sở dữ liệu đầy đủ bởi nó thiếu các phần như các kích hoạt (trigger), các thủ tục trong cơ sở dữ liệu (stored procedure). Cơ chế an toàn, bảo mật kém bởi nó không chia ra các user khác nhau trong cơ sở dữ liệu.

Dưới đây là một số hướng dẫn chính về hệ quản trị cơ sở dữ liệu Access:

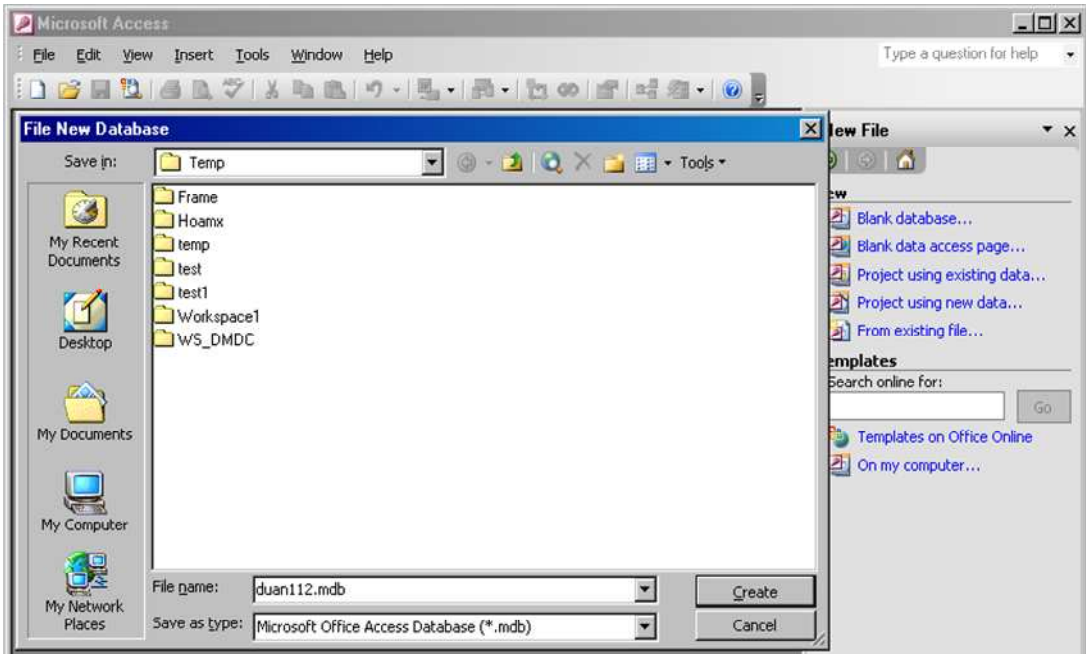
Access lưu trữ toàn bộ ứng dụng cơ sở dữ liệu trong một tệp đơn. Một tệp có đuôi .mdb của Access có thể chứa các đối tượng dữ liệu như các bảng, các chỉ mục, các truy vấn cũng như các đối tượng của ứng dụng như các màn hình

nhập liệu (forms), các báo cáo (reports), các lệnh macro và các mã visual basic.

Bước một: Kích hoạt Access

Để bắt đầu chúng ta hãy chọn START—>PROGRAMS—>Microsoft Access trên thanh công cụ của Windows. Access sẽ được kích hoạt. Chúng ta hãy lựa chọn để tạo một cơ sở dữ liệu mới.

Một cửa sổ khác sẽ xuất hiện yêu cầu chúng ta đặt tên tệp cho cơ sở dữ liệu mới đó. Chúng ta hay đưa vào tên mà chúng ta muốn đặt cho cơ sở dữ liệu đó cùng với vị trí thư mục và nhấn nút Create như màn hình 4.3.1.1 sau:



Hình 4.3.1.1 Tạo dựng một tệp cơ sở dữ liệu .mdb trong Access

Sau khi chúng ta nhấn nút create, Access sẽ đưa ra một cửa sổ với các lựa chọn sau:

- Tables: Dữ liệu trong cơ sở dữ liệu Access sẽ được lưu trong các bảng. Access cho phép tạo ra các bảng

theo các hình thức sau: tự khai báo các định dạng cho một bảng; tạo bảng từ các bảng khác đã có (wizard); và tạo bảng bằng cách đưa ngay dữ liệu vào bảng.

- Queries: các truy vấn, cho phép tạo ra các truy vấn lọc dữ liệu từ các bảng.
- Forms: Dùng để tạo ra các màn hình để nhập dữ liệu vào cơ sở dữ liệu cũng như các thao tác trình bày ứng dụng.
- Báo cáo: Dùng để tạo và định dạng dữ liệu đưa ra dưới dạng các biểu mẫu báo cáo.
- Pages: Dùng để tạo các trang web sử dụng ngôn ngữ đánh dấu siêu văn bản HTML nhằm đưa được dữ liệu lên trang web.
- Macros: được sử dụng để tự động thực hiện một loạt các thao tác tự động nào đó
- Modules: Được dùng chủ yếu trong khâu lập trình tạo ứng dụng, là nơi chứa các thủ tục, hàm của ứng dụng trên cơ sở dữ liệu Access.

Phần tiếp theo dưới đây sẽ chỉ tập trung vào việc sử dụng các bảng và truy vấn trong Access để có thể tạo dựng thành công một cơ sở dữ liệu quan hệ Access.

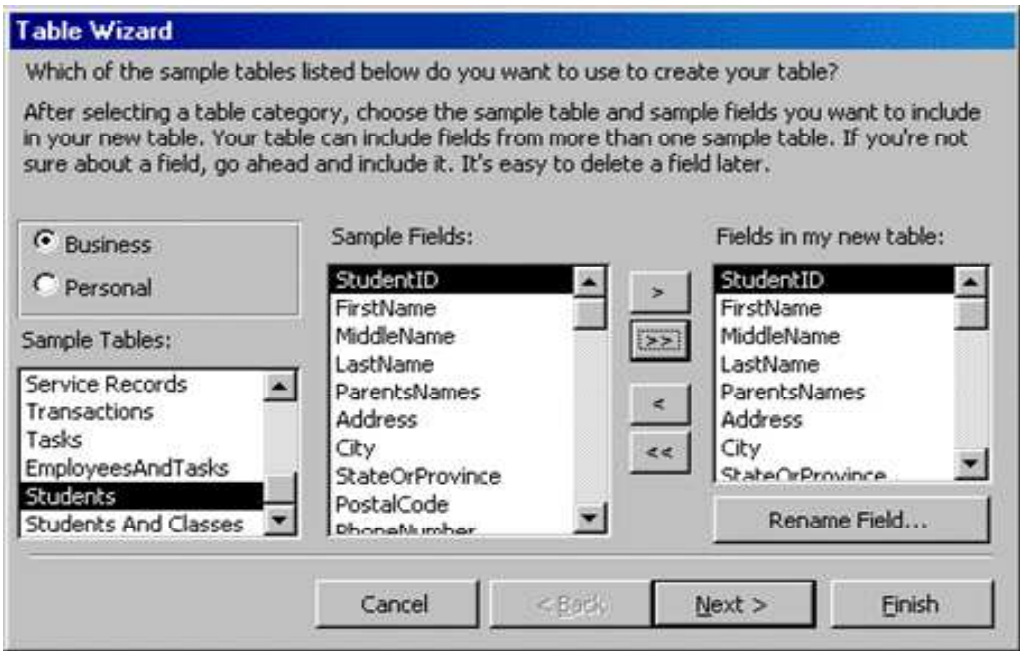
Bước hai: Thiết kế các bảng

Do trong Access không có công cụ cho phép thiết kế các đối tượng dữ liệu như kiểu thực thể, thuộc tính, liên kết nên chúng ta phải tự thực hiện bước thiết kế cơ sở dữ liệu mức logic này trên giấy hoặc bằng một số công cụ của các nhà cung cấp khác. Sau đó, dựa trên sơ đồ thực thể - liên kết

này, chúng ta tiến hành tạo dựng các bảng thông qua công cụ do Access cung cấp.

Bước ba: Tạo bảng trong Access

Trong cửa sổ chính, chọn chức năng tạo bảng “Create table”. Nếu muốn sử dụng cách tạo bảng từ các bảng đã có hãy chọn chức năng "Create table by using Wizard" màn hình sẽ xuất hiện như sau:



Hình 4.3.1.2 Tạo bảng bằng công cụ Wizard

Hãy lựa chọn bảng trong phần bên trái, các trường tương ứng của bảng đó sẽ xuất hiện ở giữa. Với những trường này, chúng ta có thể lựa chọn các trường cho bảng muốn xây dựng bằng cách chọn và nhấn nút mũi tên trỏ sang phải như hình trên. Sau khi lựa chọn xong các trường cho bảng mới nhấn nút Next.

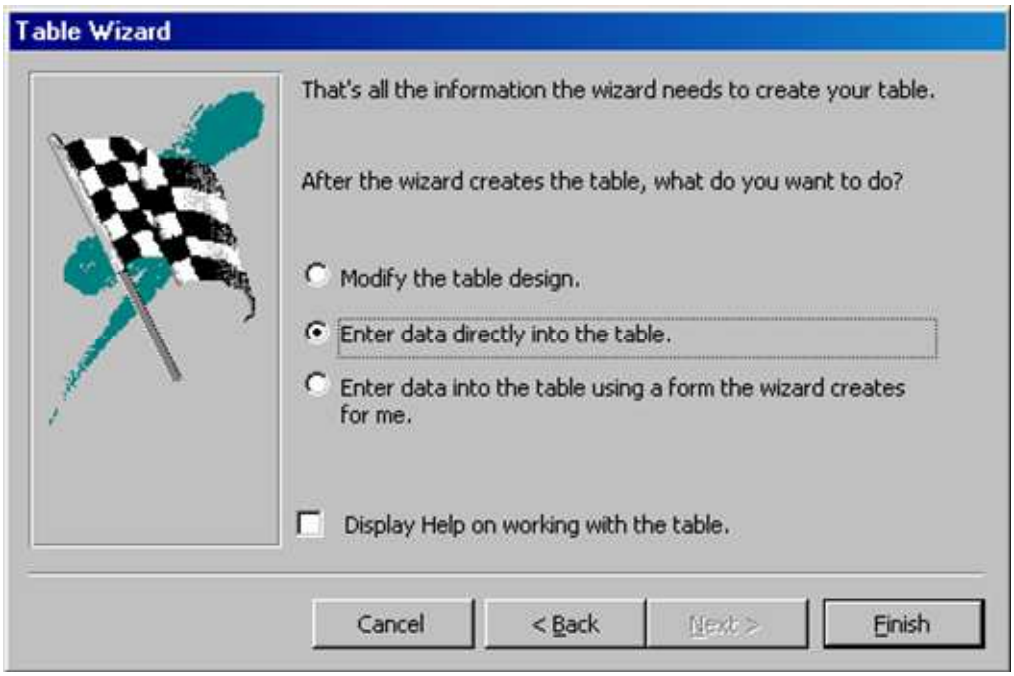
Hãy đưa vào tên cho bảng vừa tạo ở ô text trên cùng. Tiếp theo Access sẽ hỏi xem có muốn tạo khoá chính (primary key) cho bảng này không vì tùy chọn mặc định là các khoá

chính sẽ do Access tự tạo (thường là những trường có giá trị tự tăng).

Nếu muốn tạo khoá chính cần thực hiện theo hướng dẫn sau:

Hình 4.3.1.3 Tạo khoá chính cho bảng

Màn hình cuối cùng sẽ hỏi chúng ta có muốn sửa bảng tiếp, đưa dữ liệu vào bảng hay là tiếp tục sử dụng chức năng tạo màn hình nhập liệu tự động? Ở đây chúng ta hãy chọn tuỳ chọn thứ 2 là đưa dữ liệu vào bảng và nhấn phím Finish.

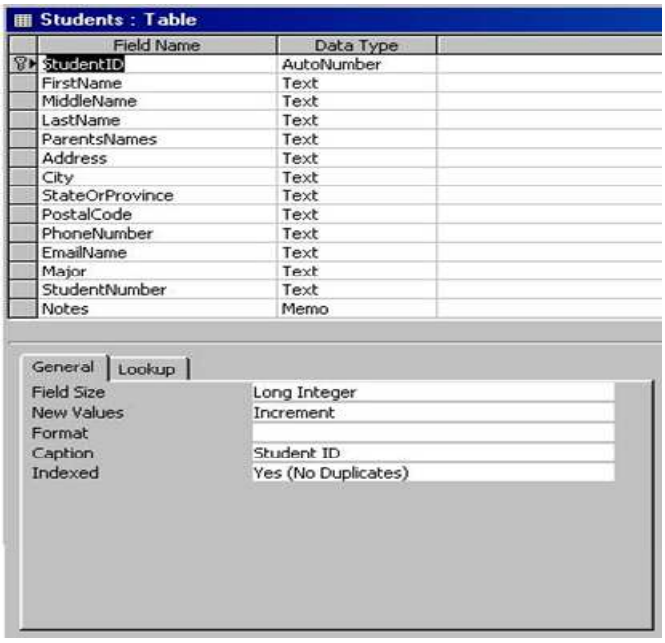


Hình 4.3.1.4 Kết thúc quá trình tạo một bảng dữ liệu

Bảng mà chúng ta vừa tạo sẽ xuất hiện trong khung datasheet. Khi thao tác với các bảng này, chúng ta sẽ thấy nó rất giống với các spreadsheet của Excel. Việc đưa các dữ liệu vào bảng sẽ được thực hiện bằng cách nháy kép vào bảng này và điền dữ liệu vào các trường tương ứng (như trong spreadsheet) hoặc có thể lấy (import) dữ liệu từ một số nguồn khác.

Dưới đây là màn hình của một bảng trong "Design View". Trong "Design View" chúng ta có thể tạo các trường, lựa chọn kiểu dữ liệu cho chúng cũng như các định dạng và khoá chính. Để lựa chọn trường nào làm khoá chính, hãy giữ Ctrl + nhấp chuột vào trường đó, sau đó di chuột đến cột bên trái ngoài cùng của bảng và nhấp chuột phải, lựa chọn chức năng Primary key, các trường vừa lựa chọn sẽ xuất hiện một biểu tượng chiếc chìa khoá bên trái trường đó. Khi chúng ta lựa chọn một trường trong lúc "Design View",

các tính chất đi kèm với thuộc tính đó sẽ xuất hiện trong màn hình “Field property”, tại màn hình này chúng ta có thể đặt các chức năng như giá trị mặc định, các luật ràng buộc khi đưa dữ liệu vào, các yêu cầu là bắt buộc phải nhập (required) hay trường sẽ được sử dụng làm chỉ mục,... Ví dụ như màn hình dưới đây:



Hình 4.3.1.5 Khung thiết kế cho một bảng dữ liệu

Màn hình trên là khung thiết kế cho bảng "Students" với thuộc tính đang chọn là StudentID, đây là thuộc tính khoá với kiểu giá trị lựa chọn là tự tăng "AutoNumber".

Bước bốn: Tạo liên kết giữa các bảng

Liên kết giữa các bảng luôn luôn đóng vai trò quan trọng trong các cơ sở dữ liệu quan hệ, nó cho phép các bản ghi (thực thể) kết nối thông qua các trường.

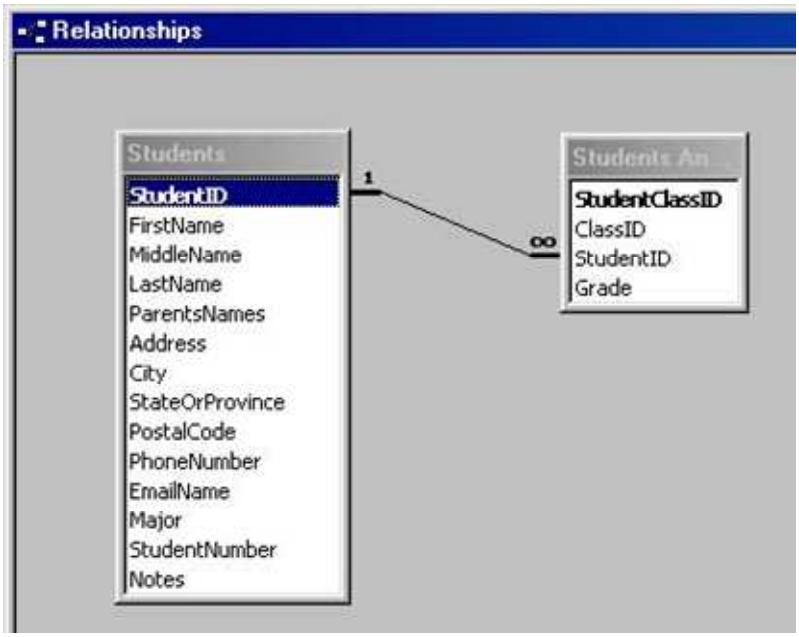
Hãy xem tiếp ví dụ sau:

Chúng ta tạo một bảng khác bằng chức năng “Create table by using wizard” và đặt tên cho bảng mới này là "Students and Classes". Tiếp theo, lặp lại các khâu như trong bước 3 cho đến khi chức năng wizard hỏi có muốn bảng mới liên kết với các bảng đã có hay không? Khi đó, hãy nhấn nút "Relationships" và chọn lựa chọn "The tables aren't related" rồi nhấn OK.

Đóng bảng vừa tạo lại và lựa chọn chức năng “Relationship” trên thực đơn công cụ “Tools”. Một màn hình sẽ hiện cho phép tạo dựng và chỉnh sửa các liên kết giữa các bảng. Nếu cả hai bảng cần kết nối không xuất hiện, hãy nhấn chuột phải vào màn hình hiện thời và lựa chọn chức năng “Show Table”. Hãy đánh dấu hai bảng cần tạo liên kết rồi nhất nút “Add”. Ở đây hai bảng sẽ là “Student” và “Students and Classes”. Access sẽ chèn những bảng này vào màn hình đang làm việc hiện thời. Nhấn nút Close để đóng cửa sổ lựa chọn bảng.

Đến đây trong màn hình đang làm việc xuất hiện hai bảng, hãy nhấn và kéo trường StudentID trong bảng “Students” sang trường StudentID trong bảng thứ hai “Students and Classes”. Một cửa sổ sẽ xuất hiện cho phép thay đổi kiểu của liên kết. Hãy đánh dấu vào ô có nhãn “"Enforce Referential Integrity" và nhấn nút "Create" để kết thúc việc tạo liên kết giữa hai bảng. Một đường kết nối sẽ cho biết liên kết đã được tạo. Các ký tự “1" and "∞" cho biết liên kết chúng ta vừa tạo là liên kết loại một - nhiều. Tức là một bản ghi ở phía bảng “1” sẽ có nhiều bản ghi kết nối với nó ở phía bảng “∞", nhưng ngược lại, một bản ghi ở phía bảng

"∞" chỉ có một bản ghi ở phía bảng "1" kết nối với nó mà thôi.



Hình 4.3.1.6 Một kết nối một nhiều trong Access

Như trong hình trên, liên kết một-nhiều đã tạo ra một toàn vẹn tham chiếu, các giá trị mà chúng ta có thể đưa vào thuộc tính StudentClassId bên bảng StudentsAndClasses không thể nằm ngoài các giá trị đã xuất hiện trong thuộc tính StudentId bên bảng Students. Và ngược lại, khi loại bỏ một bản ghi bên bảng Students thì trước đó chúng ta phải loại bỏ tất cả các bản ghi có giá trị trong thuộc tính StudentClassId bằng với giá trị trong thuộc tính StudentId của bản ghi chúng ta muốn xoá bên bảng Students.

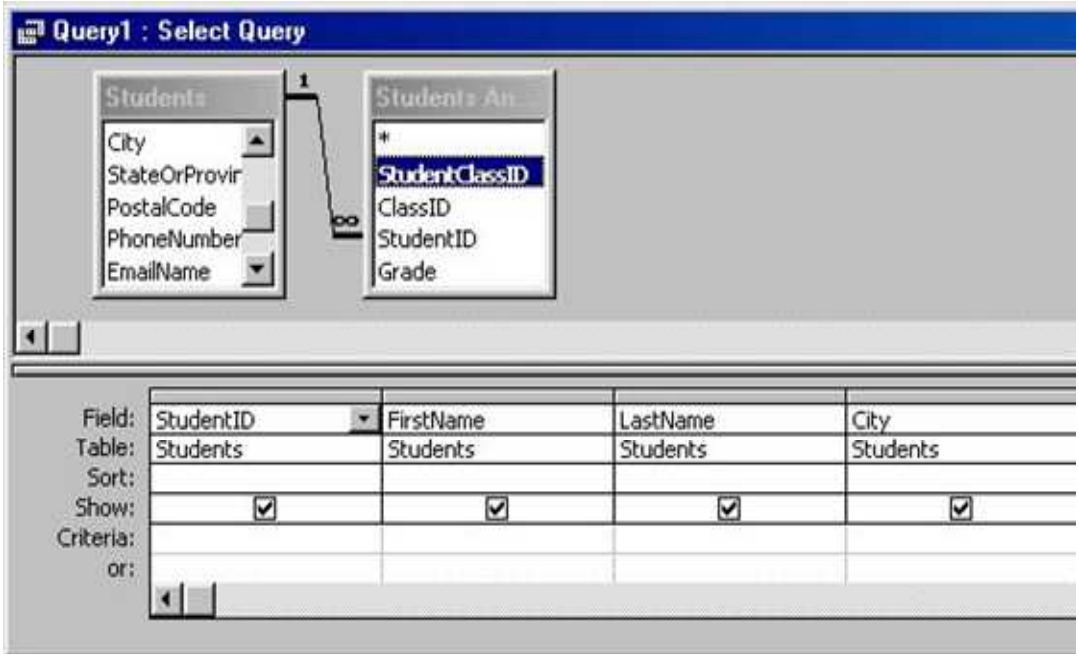
Hãy đóng cửa sổ tạo liên kết lại và nhấn "Yes" để lưu lại những thay đổi vừa tạo.

Bước năm: Truy vấn cơ sở dữ liệu

Sau khi tạo xong cơ sở dữ liệu gồm các bảng và các liên kết giữa chúng, chúng ta có thể đưa dữ liệu vào và tiến hành các truy vấn trên dữ liệu đó.

Microsoft Access cung cấp một công cụ gọi là Query-By-Example (QBE) để hỗ trợ cho việc xây dựng các câu lệnh SQL, giao diện của công cụ này rất thân thiện và chúng ta có thể dễ dàng sử dụng chúng. Khi một truy vấn được xây dựng xong thông qua QBE, chúng ta có thể xem được câu lệnh SQL mà Access đã tạo ra.

Để tạo các truy vấn, hãy nhấn kép vào chức năng "Create Query in Design View." Một cửa sổ sẽ xuất hiện và hỏi các bảng dữ liệu mà chúng ta muốn truy vấn. Hãy chọn các bảng bằng cách nhấn nút “Add”, sau khi chọn xong các bảng, hãy nhấn nút “Close”. Các bảng vừa chọn sẽ xuất hiện trên cửa sổ truy vấn phía trên. Access sẽ tự động hiển thỉ các liên kết giữa các bảng kết nối với nhau như hình sau:



Hình 4.3.1.7 Xây dựng một truy vấn trong Access

Để lựa chọn các trường cần truy vấn, hãy chọn trường đó và kéo xuống cửa sổ QBE phía dưới. Trong màn hình QBE, mỗi một cột thể hiện cho trường mà chúng ta đã chọn, hàng thứ nhất là tên trường, hàng thứ hai là bảng chứa trường đó, hàng thứ ba cho phép lựa chọn kiểu sắp xếp, hàng thứ tư cho phép lựa chọn trường đó có được hiển thị ra trong kết quả tìm kiếm hay không.

Vùng Criteria phía dưới cùng cho phép chúng ta định nghĩa các điều kiện tìm kiếm. Nếu trong cùng một hàng thì đó là điều kiện And, còn giữa các hàng khác nhau là điều kiện Or.

Ví dụ, để tạo một truy vấn tìm kiếm tất cả các sinh viên sống ở Hà nội hay Hải phòng và có chữ cái đầu tiên trong tên bắt đầu bằng "B" chúng ta đưa điều kiện vào vùng Criteria như sau:

Gõ xâu "B*" và ô tên riêng Lastname tại vùng Criteria (chú ý bao gồm cả ký tự ")

Gõ xâu "Hà nội" tại ô CityField tại vùng Criteria

Gõ xâu "Hải phòng" tại ô CityField tại vùng Criteria nhưng ở hàng Or.

Sau đó, hãy chọn chức năng "SQL View" trong thực đơn View. Một cửa sổ hiển thị câu lệnh SQL tương ứng với truy vấn đó sẽ được đưa ra

b. Hệ quản trị cơ sở dữ liệu Visual Foxpro

FoxPro là hệ quản trị cơ sở dữ liệu khá thông dụng trong những năm 90 của thế kỷ 20. Mặc dù sự lớn mạnh của các hệ quản trị cơ sở dữ liệu như Oracle, SQL Server, DB2, ... FoxPro vẫn tiếp tục được hãng Microsoft phát triển

và tiếp tục hoàn thiện cho các bài toán cỡ nhỏ. Các công cụ mà hãng Microsoft cung cấp cho người sử dụng được thực hiện thông qua các Wizard: Application Wizard, Form Wizard, Report Wizard,... Ngoài ra, các công cụ SQL Upsizing Wizard và Oracle Upsizing Wizard cho phép người sử dụng có thể thực hiện chuyển cơ sở dữ liệu của mình sang các hệ quản trị cơ sở dữ liệu nói trên đem lại khả năng phát triển tiềm tàng cho FoxPro.

Sau đây, chúng ta sẽ tìm hiểu công cụ chính của hệ quản trị cơ sở dữ liệu Visual FoxPro, Project Manager, để từ đó có thể có được cái nhìn toàn diện về cách thức xây dựng và phát triển các hệ thống cơ sở dữ liệu sử dụng công cụ này. Để thông tin đưa ra là cập nhật, chúng tôi sử dụng bộ công cụ Microsoft Visual FoxPro 6.0 để minh họa.

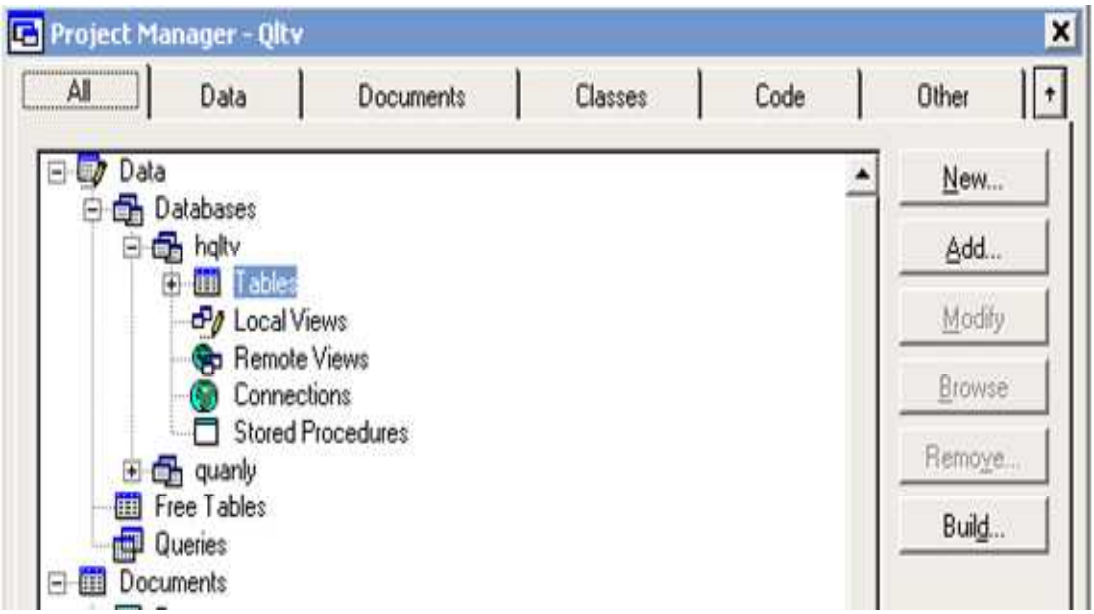
Công cụ quản lý các dự án (Project Manager)

Công cụ này có 2 mục đích chính:

1. Giúp việc tổ chức các file dựa trên các kiểu file
2. Cung cấp khung quản lý các thành phần của một ứng dụng để có thể dịch chúng sang dạng APP hay EXE.

Hình 4.3.1.8 Cửa sổ Project Manager

Project Manager cho phép người sử dụng quản lý tất cả các loại file trong cơ sở dữ liệu, các mục chính bao gồm: dữ liệu (data), tài liệu (documents), thư viện (class libraries), mã (code), và các mục khác. Mục dữ liệu chứa các mục nhỏ hơn là: các cơ sở dữ liệu (Databases), các bảng tự do (Free Tables), và các truy vấn (Queries). Như trong hình trên, mục dữ liệu chứa 2 cơ sở dữ liệu có tên tên lần lượt là: HQLTV và QUANLY.



Hình 4.3.1.9 Thành phần của một cơ sở dữ liệu

Thành phần chính của một cơ sở dữ liệu bao gồm: Các bảng (Tables), các khung nhìn (Local Views),... Các bảng tự do là các bảng không thuộc cơ sở dữ liệu nào, thường được sử dụng như các bảng tạm trong quá trình xử lý dữ liệu hay các bảng chứa các tham số của hệ thống. Các thủ tục lưu trong cơ sở dữ liệu và các khung nhìn từ xa thông qua một kết nối có thể liên kết đến một cơ sở dữ liệu khác (có thể không phải là Visual Foxpro).

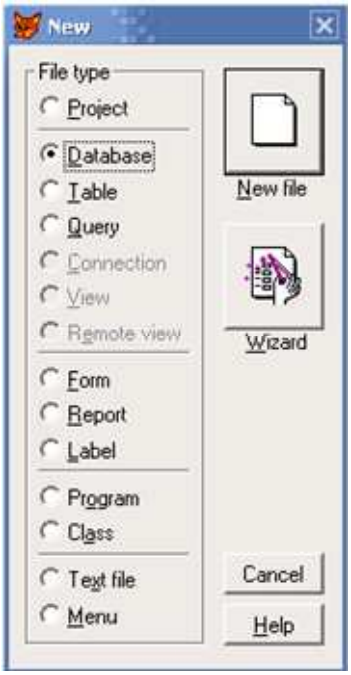
Mục Documents chứa các biểu mẫu: Forms và Reports: Reports sử dụng cho một chương trình ứng dụng chạy trên các cơ sở dữ liệu. Class Libraries chứa các lớp do người sử dụng tự định nghĩa. Mục Code chứa các file mã chương trình dạng .PRG, các hàm thư viện API, các file thực thi dưới dạng .EXE. Mục Others chứa các file khác mà Dự án sử dụng chẳng hạn như: các file biểu tượng, ảnh tĩnh, các file tham số, ...

Chúng ta sẽ thực hiện tạo cơ sở dữ liệu, và các bảng dữ liệu theo quy trình của Visual FoxPro. Lưu ý rằng, các bảng dữ liệu được lưu dưới dạng các file riêng rẽ có phần mở rộng .DBF, trong khi cơ sở dữ liệu được lưu dưới dạng .DBC (Database Container). Để có thêm thông tin về các dạng file của Visual FoxPro, người sử dụng có thể tra cứu trong phần Hướng dẫn sử dụng của chương trình hay các cuốn sách về nó.

Để tạo một cơ sở dữ liệu mới,cần thực hiện theo các bước sau:

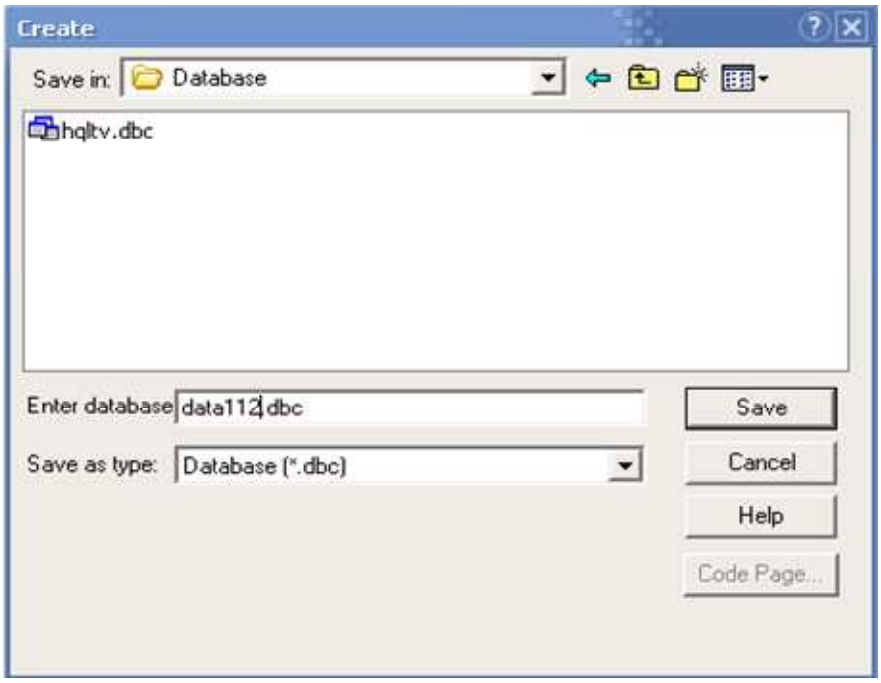
Bước 1: Tạo một Database mới bằng cách chọn trên thực đơn chương trình: ***File\New...***

Visual FoxPro sẽ hỏi xem muốn tạo file loại gì trong hộp thoại sau đây:



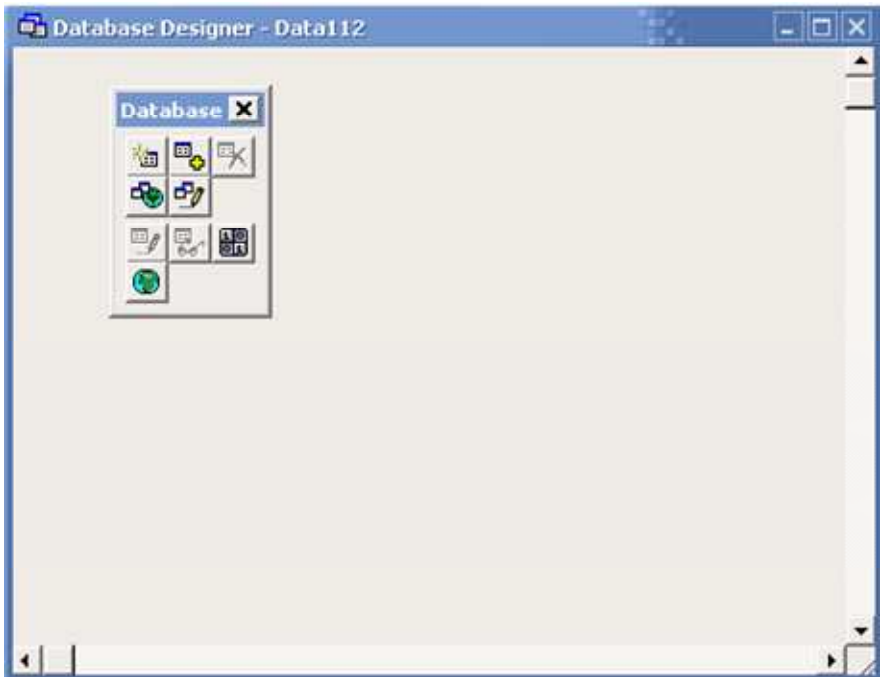
Hình 4.3.1.10 thoại chọn tạo mới một cơ sở dữ liệu

Hãy chọn Database như trong hình trên và nhấn nút lệnh "New file". Sau đó nhập đường dẫn và tên của cơ sở dữ liệu vào hộp thoại kế tiếp như sau:



Hình 4.3.1.11 Nhập đường dẫn và tên của cơ sở dữ liệu cần tạo

Cần lưu ý rằng các file cơ sở dữ liệu của Visual FoxPro có phần mở rộng là DBC. Đây chính là điểm khác biệt so với các phiên bản FoxPro for DOS, đó là sự phân biệt rõ ràng giữa các bảng dữ liệu và cơ sở dữ liệu. Sau khi nhập đường dẫn và tên file cơ sở dữ liệu, nhấn nút lệnh Save để lưu lại. Visual FoxPro sẽ mở cửa sổ thiết kế cơ sở dữ liệu để người sử dụng có thể thực hiện bổ sung, cập nhật các thành phần khác của cơ sở dữ liệu.



Hình 4.3.1.12 Cửa sổ Database Designer

Bước 2: Tạo các bảng dữ liệu

Để tạo mới một bảng dữ liệu, chúng ta sử dụng thanh công cụ Database, chọn nút lệnh **New Table**.

Hình 4.3.1.13 Nút tạo dựng bảng dữ liệu

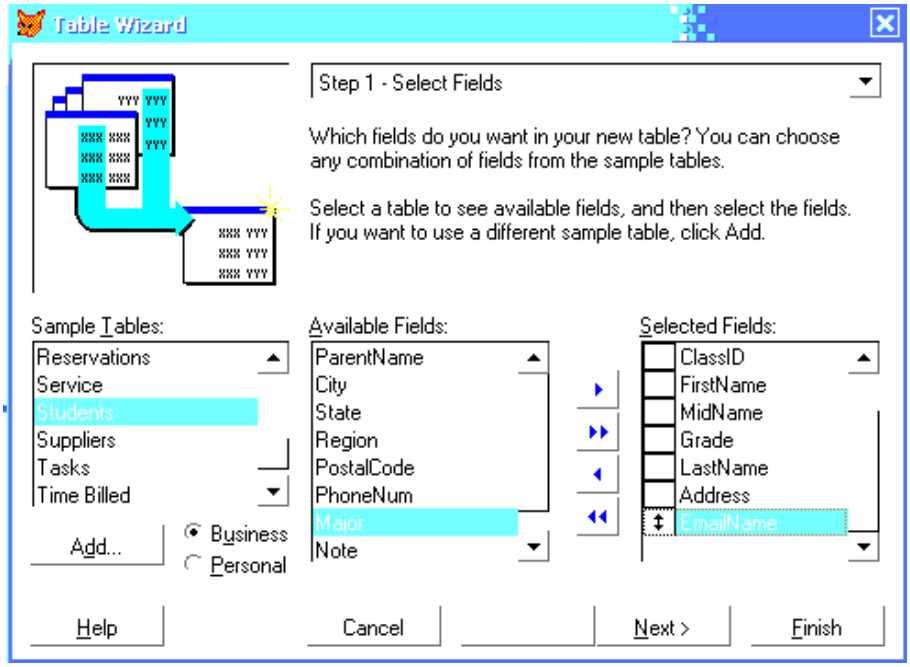
Người sử dụng có thể chọn một trong hai cách tạo bảng:

- Sử dụng Wizard

- Tạo bảng thủ công

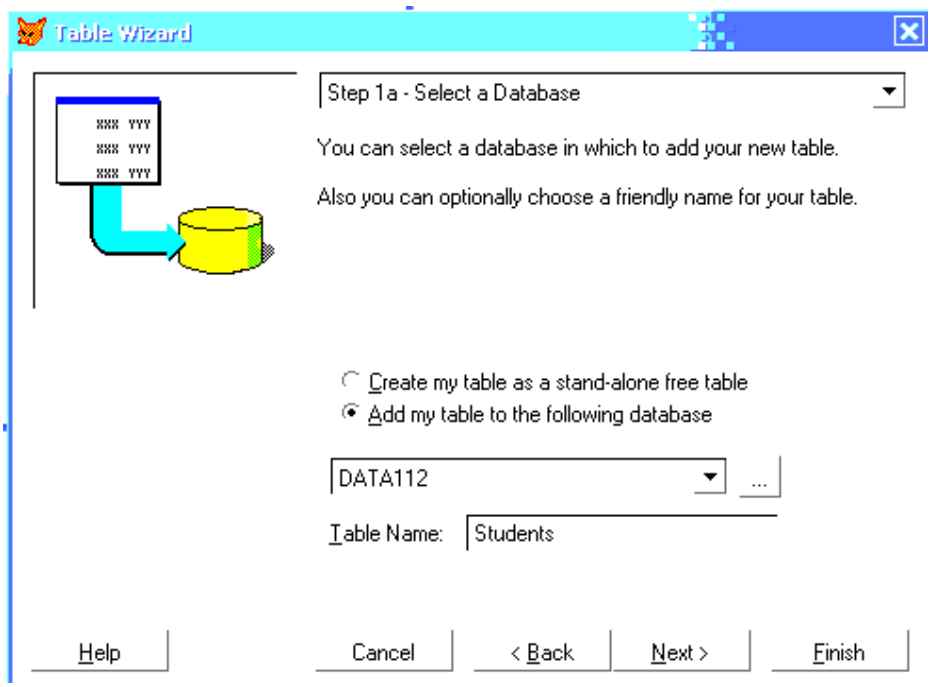
Hình 4.3.1.14 Hộp thoại lựa chọn cách tạo bảng

Nếu chọn Table Wizard, chương trình đưa ra các bước để người sử dụng có thể chọn:



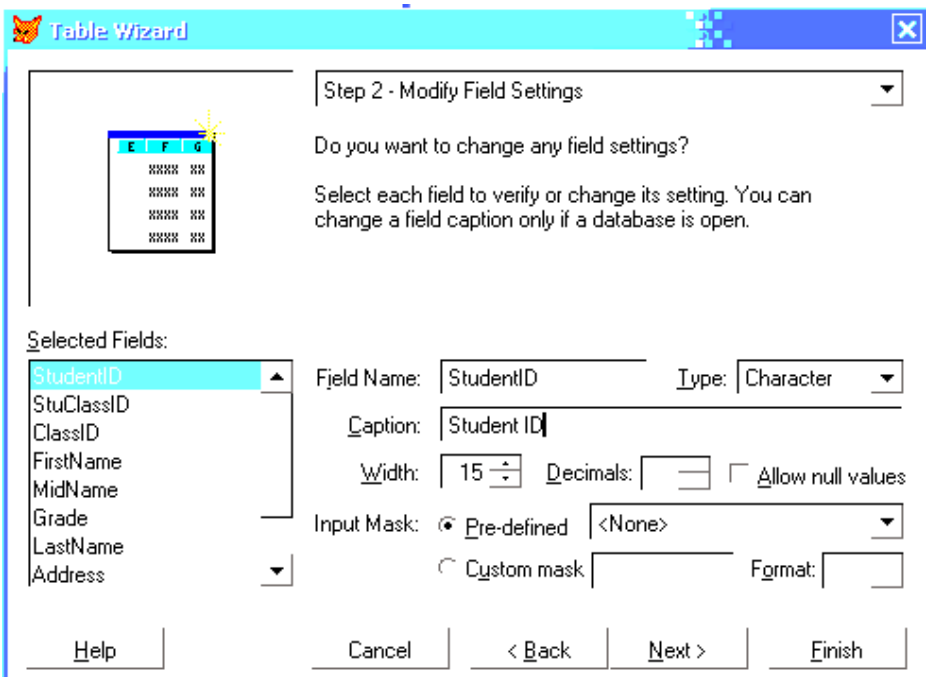
Hình 4.3.1.15 Chọn bảng mẫu, các trường dữ liệu cần thiết để tạo bảng

Nhấn **Next** để sang bước tiếp theo, chọn lựa cơ sở dữ liệu mà bảng sẽ tạo thuộc về:



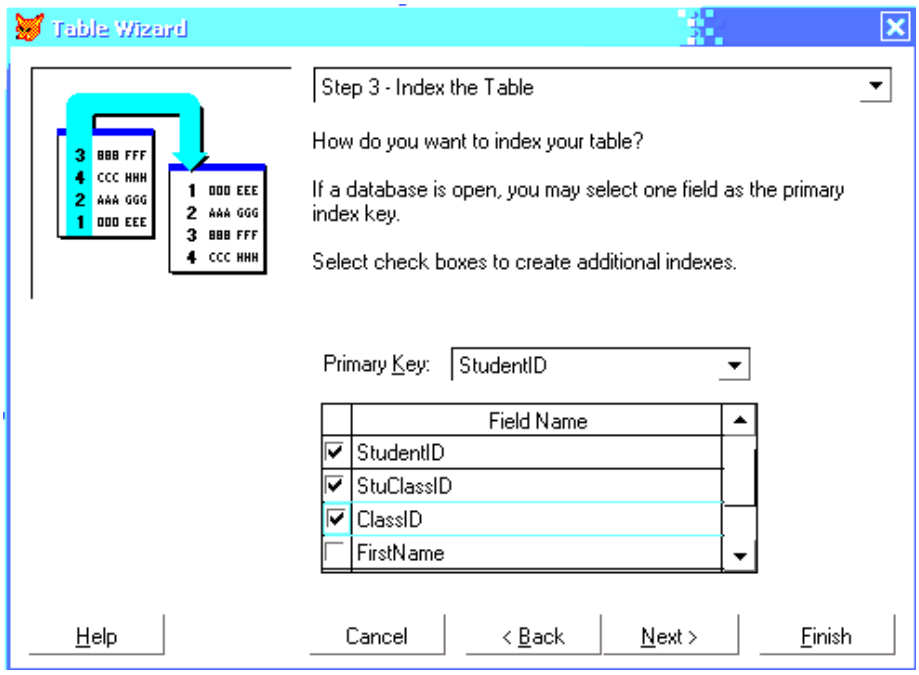
Hình 4.3.1.16 Chọn cơ sở dữ liệu

Nếu muốn tạo bảng mới như một bảng tự do, chọn "Create my table as a stand-alone free table", ngược lại, chọn cơ sở dữ liệu mà chúng ta vừa tạo ra. Nhấn **Next** để sang bước kế tiếp tạo lập cấu trúc cụ thể cho bảng:



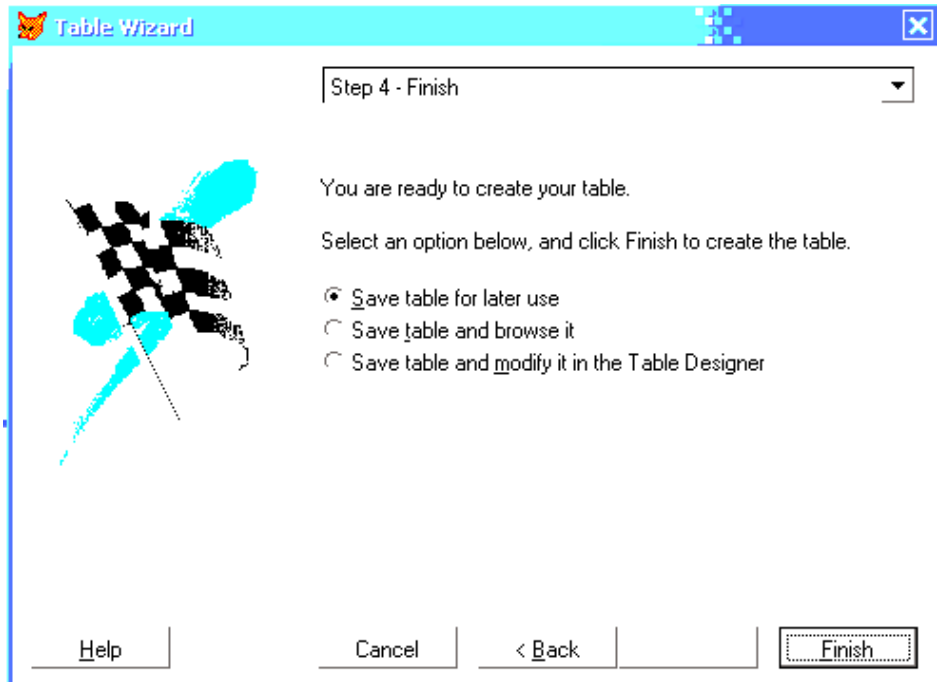
Hình 4.3.1.17 Sửa đổi cấu trúc các trường dữ liệu cho bảng

Người sử dụng cần xác định kiểu dữ liệu, mô tả độ rộng,... cho các trường dữ liệu. Khi hoàn tất, nhấn **Next** để chuyển sang bước kế tiếp tạo lập chỉ mục cho bảng



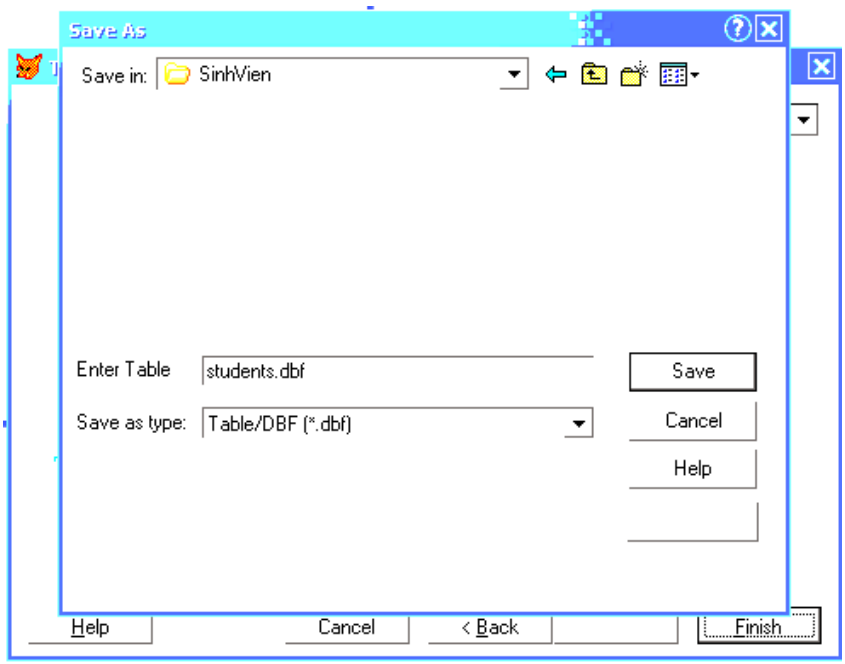
Hình 4.3.1.18 Tạo lập chỉ mục

Mục Primary Key cho phép người sử dụng xác định trường đóng vai trò khóa chính cho bảng dữ liệu. Nếu muốn tạo thêm các chỉ mục cho bảng, chọn các ô check phía bên dưới. Nhấn **Next** để sang bước kế tiếp tạo lập và lưu lại bảng đó



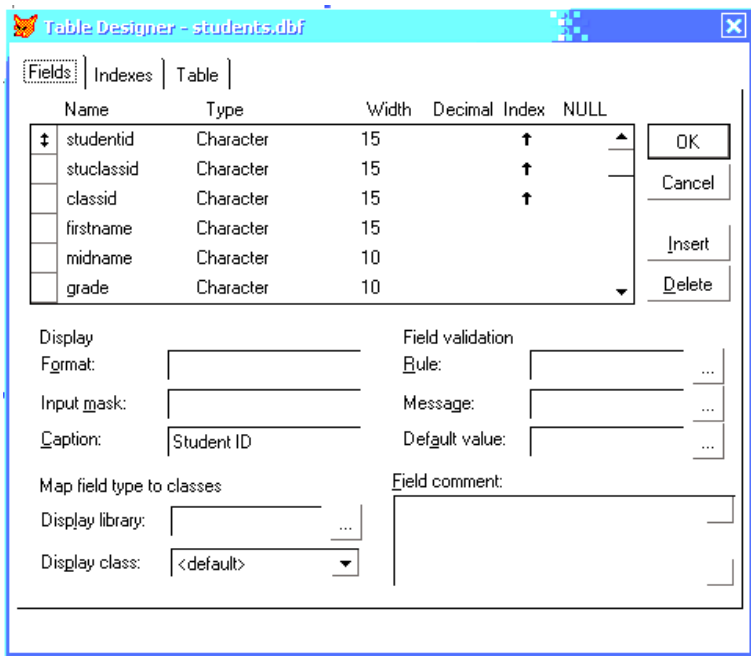
Hình 4.3.1.19 Kết thúc và lưu giữ bảng

Nhấn **Finish** để tạo và lưu lại hoặc xem lại cấu trúc của bảng (Chọn Save table and modify ...). Sau đó xác định đường dẫn và tên file chứa bảng dữ liệu này.



Hình 4.3.1.20 Xác định thư mục lưu bảng

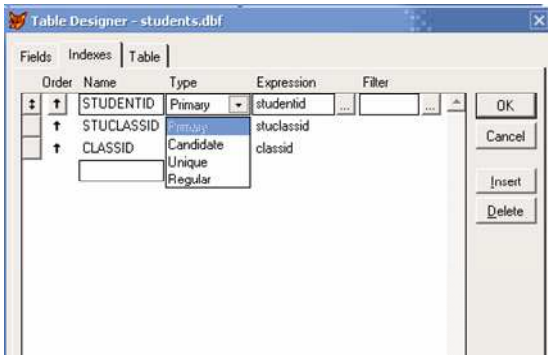
Và nhấn **Save** để lưu lại.



Hình 4.3.1.21 Sửa đổi, xem lại cấu trúc bảng

Nếu chọn New Table, để tạo lập bảng mới bằng tay, cửa sổ để người sử dụng tạo bảng tương tự như trong hình trên. Để tạo các chỉ mục, chọn Tab **Indexes** như trong hình trên, và

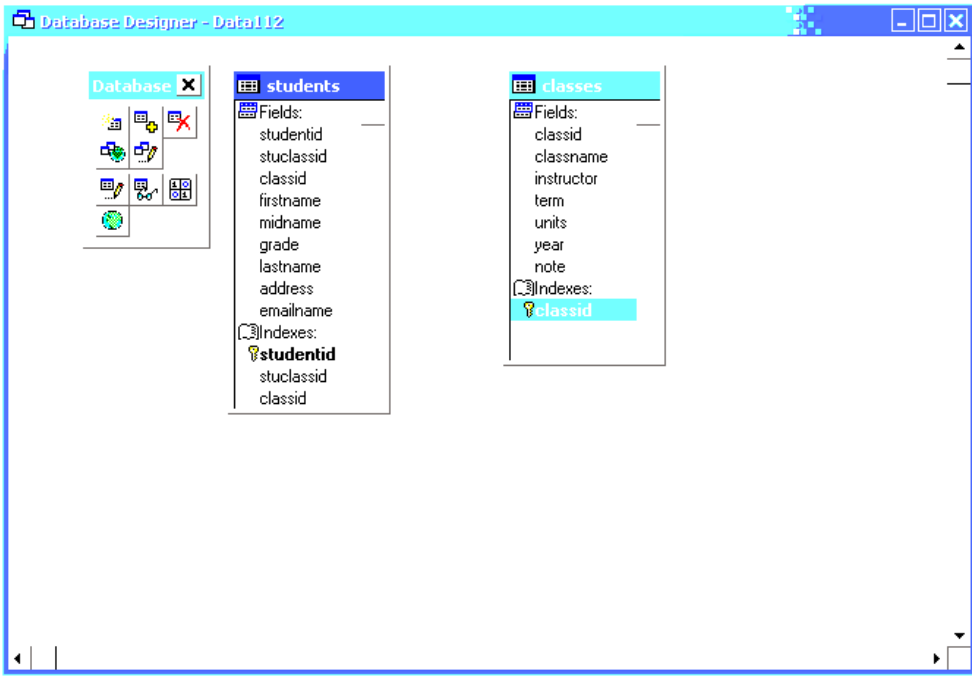
thiết lập các chỉ mục bằng cách chọn trường dữ liệu và cách sắp xếp của chỉ mục.



Hình 4.3.1.22 Tạo chỉ mục

Có thể chọn 1 trong 4 kiểu chỉ mục: Khóa chính (Primary), Ứng cử (candidate), Duy nhất (Unique), hay thông thường (Regular). Cột Expression cho phép tạo các chỉ mục bằng cách ghép các trường hoặc sử dụng các hàm do người sử dụng tự tạo.

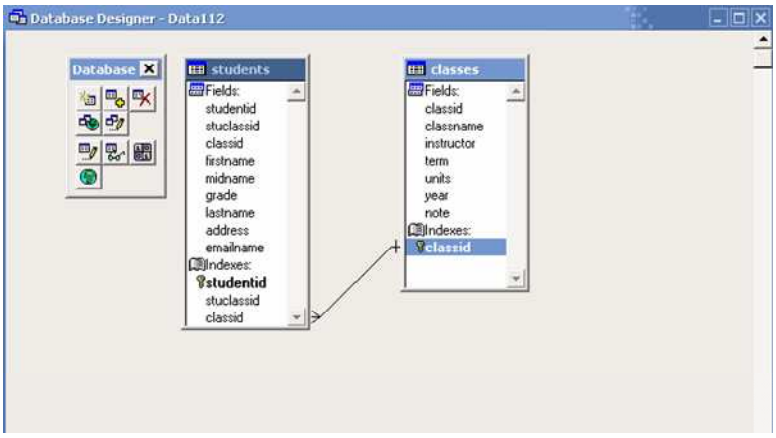
Lặp lại các thao tác trên, người sử dụng có thể tạo ra các bảng dữ liệu khác cho mình.



Hình 4.3.1.23 Các bảng trong cơ sở dữ liệu

Bước 3: Thiết lập các liên kết trên các bảng đã tạo

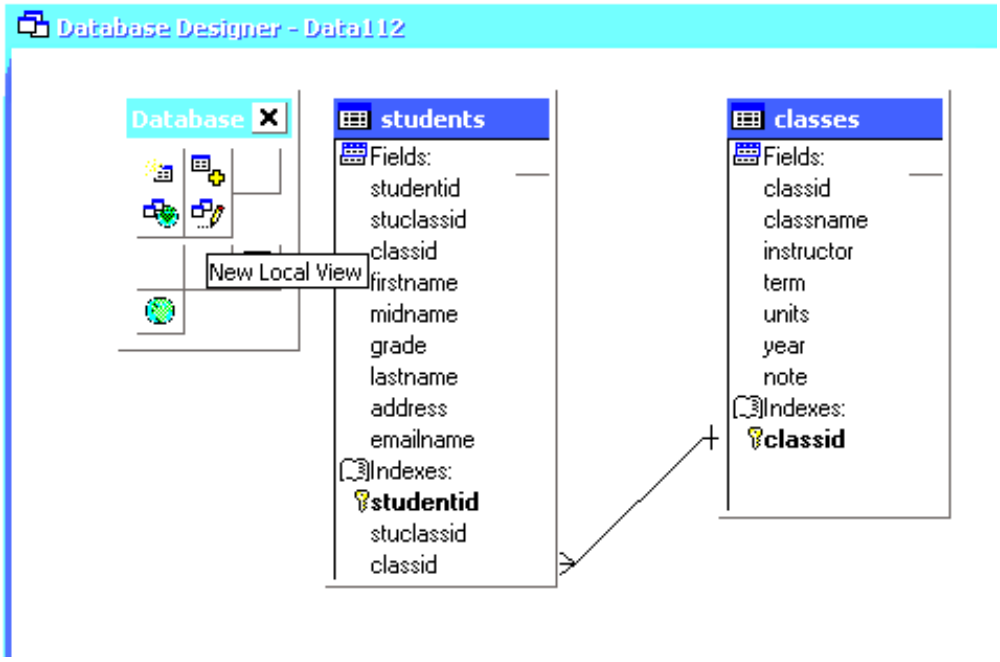
Cần chú ý rằng, để có thể tạo ra các liên kết giữa các bảng, cần thực hiện định nghĩa và tạo các chỉ mục trên các trường mà thông qua đó, chúng liên kết được với nhau. Chẳng hạn, trong hình vẽ trên, chúng ta có 2 bảng: Students và Classes. Liên kết được xác định giữa chúng là: Mỗi lớp (class) sẽ có 0 hoặc nhiều sinh viên (student). Chúng ta đã thực hiện tạo chỉ mục cho bảng Students trên trường *classid*, việc tạo liên kết giữa 2 bảng này được thực hiện bằng cách thông qua cửa sổ Database Designer. Giữ chuột vào trường **classid** đã được định nghĩa là khóa chính của bảng Classes, kéo và thả tại chỉ mục *classid* bên bảng Students. Khi đó Visual FoxPro sẽ tạo liên kết giữa 2 bảng này dựa trên trường dữ liệu chung: *classid*.



Hình 4.3.1.24 Tạo liên kết giữa 2 bảng dữ liệu

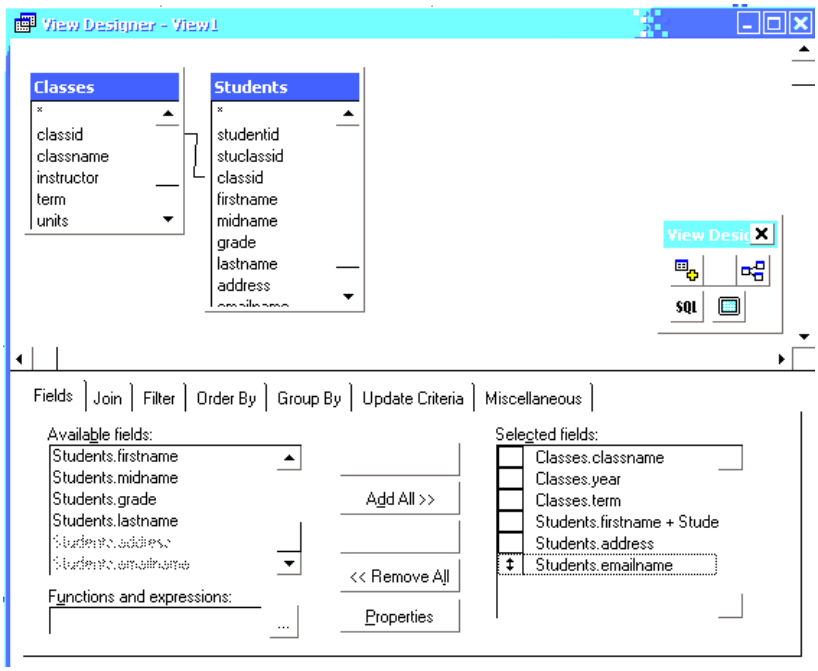
Bước 4: Tạo lập các khung nhìn trên các bảng đã có

Nhấn nút lệnh **New Local View** để bắt đầu thực hiện chức năng tạo khung nhìn.



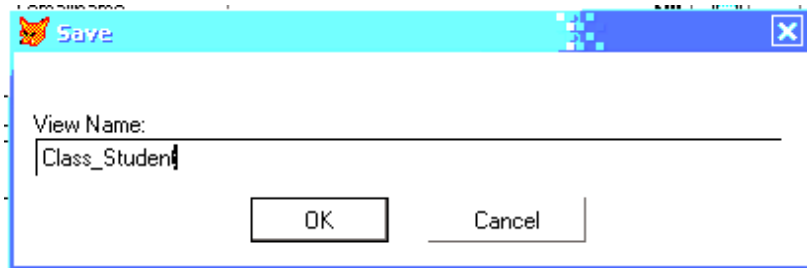
Hình 4.3.1.25 Tạo một khung nhìn nội bộ

Màn hình tiếp theo đó trợ giúp việc tạo các khung nhìn một cách trực quan hơn. Các bảng được chọn sẽ tự động hiển thị các liên kết giữa chúng như trong hình sau đây.



Hình 4.3.1.26 Xác định các bảng và trường dữ liệu

Hãy xác định các trường dữ liệu trong mục **Fields**, xác nhận và thêm vào các điều kiện kết nối nếu cần,... sau đó nhấn **Save** trên thanh công cụ chuẩn. Khi đó, khung nhìn mới định nghĩa sẽ được tạo với tên mà chúng ta nhập vào trong màn hình tiếp sau.



Hình 4.3.1.27 Nhập tên khung nhìn mới

3.2. Các hệ quản trị cơ sở dữ liệu cỡ lớn

a. Hệ quản trị cơ sở dữ liệu SQL Server

Microsoft SQL Server là một hệ quản trị cơ sở dữ liệu do công ty Microsoft xây dựng. Là một trong những hệ quản trị cơ sở dữ liệu phục vụ cho các hệ thống nhỏ lúc ban đầu, ngày nay, Microsoft SQL Server không ngừng được đổi mới,hoàn thiện và ngày càng đáp ứng được những yêu cầu khắt khe về tính tiện dụng, khả năng quản lý cơ sở dữ liệu,...

Phiên bản rút gọn của Microsoft SQL Server là MSDE (Microsoft SQL Server Desktop Engine) được đưa ra phục vụ cho các sản phẩm khác của Microsoft như: Visual FoxPro, MS Access, MS Web Matrix, và các sản phẩm khác. Hiện tại, MSDE bị giới hạn ở mức 2 GB bộ nhớ, 8 kết nối đồng thời, và không có công cụ để quản trị.

Microsoft SQL Server sử dụng một biến thể của ngôn ngữ SQL, Transact-SQL, một tập bao của ngôn ngữ chuẩn SQL-92 (Tiêu chuẩn ISO cho SQL, được chứng nhận vào năm 1992). T-SQL được gọi là tập bao của SQL-92 vì nó có hỗ trợ thêm các cú pháp sử dụng trong việc viết các thủ tục lưu trực tiếp trong cơ sở dữ liệu (stored procedures), và thực hiện các giao dịch (transactions support). Theo số liệu thống kê của hãng Gartner, vào tháng 09 năm 2001, Microsoft SQL Server chiếm khoảng 14% thị trường cơ sở dữ liệu thương mại.

Một điểm khác biệt của Microsoft SQL Server so với các hệ quản trị cơ sở dữ liệu khác đó là không thực hiện khóa dữ liệu ở mức bản ghi một cách chính tắc. Thay vào đó, Microsoft SQL Server sử dụng một ứng dụng khác để đảm bảo rằng không có 2 người sử dụng cùng thực hiện truy vấn

tại một thời điểm. Nếu tình trạng này xảy ra có thể gây ra hệ thống bị treo.

Các phiên bản của Microsoft SQL Server:

Việc hiểu rõ về khả năng và hạn chế của các phiên bản sẽ giúp chúng ta có được sự lựa chọn đúng đắn nhất khi mua sản phẩm này. Một điều cần lưu ý là tất cả các phiên bản này đều sử dụng cùng cấu trúc tệp và tập các lệnh của Transact-SQL (T-SQL). Điều đó có nghĩa là các file cơ sở dữ liệu tạo bởi một phiên bản có thể được sử dụng bởi các phiên bản khác của Microsoft SQL Server.

Standard Edition

Đây là phiên bản có lẽ được sử dụng rộng rãi nhất của Microsoft SQL Server. Phiên bản này hỗ trợ từ 1-4 bộ xử lý (CPU) và 2GB RAM, có thể đáp ứng phần lớn các nhu cầu của các cơ quan, xí nghiệp cỡ nhỏ. Phiên bản này hỗ trợ tất cả các tính năng mà Microsoft SQL Server đã nổi tiếng:

- Data Transformation Services (DTS), Các dịch vụ hỗ trợ chuyển đổi dữ liệu, cung cấp tính năng import, export, và chuyển đổi dữ liệu giữa nhiều nguồn khác nhau.
- Cho phép thực hiện tạo bản sao (Replication) giữa Microsoft SQL Server và các sản phẩm của các nhà cung cấp khác
- Có khả năng thực hiện các truy vấn gần ngôn ngữ Tiếng Anh (English Query)
- Hỗ trợ XML (eXtensive Mark-up Language)
- Có khả năng tìm kiếm toàn văn (Full-text searches)

Mặc dù vậy, phiên bản này không hỗ trợ khả năng phân cụm (clustering) và một vài chức năng nâng cao cho việc phân tích dữ liệu trực tuyến (OLAP). Phiên bản này chỉ có thể cài đặt trên hệ thống Windows 2000 Server hay cao hơn.

Personal Edition

Phiên bản này cũng có các chức năng giống như phiên bản Standard. Ràng buộc chính của nó là về bản quyền và các giới hạn về hệ điều hành.

Phiên bản dùng cho cá nhân này không thể sử dụng nhiều hơn 2 bộ vi xử lý, và nó có một bộ điều chỉnh tốc độ truy vấn. Bộ điều chỉnh truy vấn này sẽ làm ảnh hưởng đến hiệu năng nếu như bị quá tải. Đồng thời, phiên bản này cũng không thực hiện được việc tạo bản sao theo mô hình giao dịch. Mặc dù vậy nó có thể là một subscriber trong mô hình tạo bản sao nêu trên.

Phiên bản này có thể cài đặt trên Windows 98 hoặc cao hơn. Phiên bản này có kèm theo chức năng tìm kiếm toàn văn (full-text search) và dịch vụ phân tích dữ liệu (SQL Server 2000 Analysis Services), nhưng các tính năng này không thể cài đặt trên Windows 98 hay Windows ME. Một giới hạn nữa là các hệ điều hành như Windows 98 không hỗ trợ các dịch vụ như Windows 2000, XP, do vậy, một số các chức năng như SQL Agent sẽ không chạy khi chúng ta chưa đăng nhập vào Windows..

Microsoft Data Engine (MSDE)

Microsoft Data Engine (MSDE) cũng là một phiên bản được sử dụng rộng rãi của SQL Server. Bộ cài đặt của phần mềm này có thể tìm thấy trong bộ công cụ Visual Studio Development. Cũng có thể mua nó riêng lẽ hoặc trong bộ

Office Professional. MSDE thay thế JET với vai trò như cơ chế cung cấp dữ liệu trong các ứng dụng phân tán.

MSDE có cùng các đặc trưng cơ bản như phiên bản Standard, và cũng là một công cụ tốt cho việc phát triển các ứng dụng vì có thể chuyển đổi việc sử dụng nó sang các phiên bản Microsoft SQL Server khác do có cùng cấu trúc file như đã nêu trong các phần trên.

Rất nhiều các máy chủ web sử dụng phiên bản Microsoft SQL Server này bởi lẽ việc phân phối nó trong các bộ cài đặt là miễn phí. Nói một cách khác, nếu sử dụng các công cụ như: SQL Server 2000 (Developer, Standard, and Enterprise Editions), Visual Studio .NET (Architect, Developer, and Professional Editions), ASP.NET Web Matrix Tool, Office XP Developer Edition, MSDN Universal and Enterprise subscriptions thì chúng ta có thể cung cấp MSDE cài đặt cùng với ứng dụng của mình. Tất nhiên là chúng ta phải mua bản quyền của công cụ mà chúng ta sử dụng để phát triển.

Có 2 hạn chế chủ yếu của MSDE, đó là:

- Nó có một bộ điều chỉnh tốc độ truy vấn, giới hạn hiệu năng của các ứng dụng có lớn hơn 5 user, và tất nhiên, điều đó phụ thuộc vào cách viết chương trình. Không có các công cụ trực quan đi kèm để quản trị và các sách hướng dẫn trực tuyến.
- Các cơ sở dữ liệu bị giới hạn ở mức dưới 2GB, và MSDE không thể dùng SQL Mail.

Enterprise Edition

Phiên bản Enterprise cung cấp tất cả các chức năng mà các phiên bản khác có và hơn thế nó có thể sử dụng đến 32 bộ

vi xử lý và 64GB RAM có khả năng đem lại hiệu năng tốt nhất cho người sử dụng. Khả năng này đặc biệt rõ khi thực hiện các truy vấn song song do máy chủ cơ sở dữ liệu sử dụng nhiều bộ vi xử lý để thực hiện các truy vấn trong cùng một lúc.

Một lợi thế khác của phiên bản này là khả năng mở rộng. Nếu cài đặt trên hệ điều hành Windows 2000 Advanced Server, chúng ta có thể thực hiện chuyển hướng các yêu cầu dữ liệu sang một máy chủ cơ sở dữ liệu khác nhằm đảm bảo hệ thống hoạt động thông suốt.

Khi sử dụng phiên bản này, chúng ta có thể định nghĩa các phân đoạn OLAP (quá trình phân tích trực tuyến) và tạo các khối (cubes) chứa các dữ liệu đa chiều. Phiên bản này cũng hỗ trợ thực hiện các phân tích trực tuyến gần như trong thời gian thực.

Một điểm mạnh nữa là người phát triển ứng dụng có thể tạo các khung nhìn được đánh chỉ số, góp phần làm tăng tốc độ các báo cáo. Tuy nhiên, để đạt được điều này thì yêu cầu về cấu hình của máy chủ chắc chắn sẽ cao hơn so với các phiên bản khác.

Developer Edition

Với phiên bản này, chúng ta có thể sử dụng để phát triển các ứng dụng chạy trên nền Hệ quản trị cơ sở dữ liệu SQL Server. Phiên bản này có tất cả các đặc trưng của phiên bản Enterprise ngoại trừ bản quyền phát triển không cho phép triển khai phiên bản này. Nó có thể cài trên Windows 2000 Professional và cao hơn.

Với phiên bản này, chúng ta có quyền cài đặt SQL Server 2000 phiên bản cho Windows CE và triển khai các ứng

dụng đó miễn phí, mặc dù các thiết bị CE cần có bản quyền cho phiên bản SQL Server chạy trên CE.

Một đặc điểm khác là nó có thể nâng cấp lên phiên bản Standard hoặc Enterprise.

Các phiên bản dùng thử (Evaluation)

Các phiên bản này thường có đầy đủ các chức năng nhưng người sử dụng chỉ có thể xem và test các chức năng đó trong vòng 120 ngày. Các phiên bản này không thể dùng để phát triển hay triển khai mà chỉ để phục vụ cho mục đích có đi đến quyết định mua sản phẩm hay không.

CE Edition

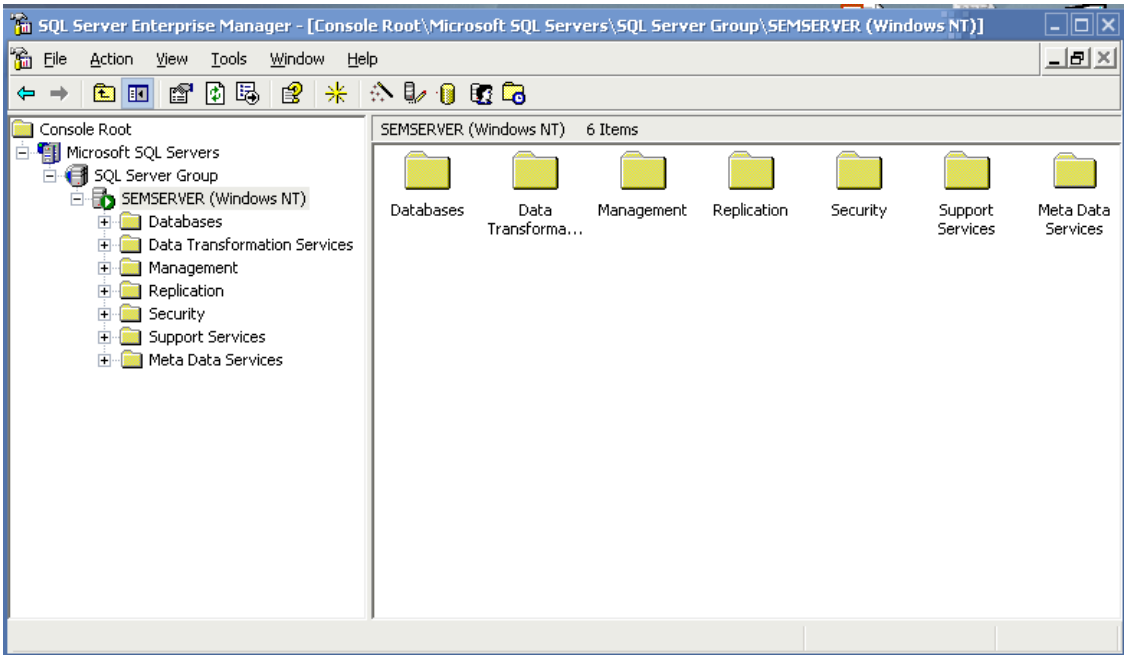
Phiên bản CE của Microsoft SQL Server có cùng các công cụ và cú pháp SQL mà người phát triển có thể dùng trong các ứng dụng dựa trên nền cơ sở dữ liệu SQL Server. Cơ sở dữ liệu này ở dạng quan hệ và có thể chạy trên các thiết bị bỏ túi. Mục đích chính của phiên bản này là dùng cho các ứng dụng phân tán. Điều này được thực hiện thông qua giao thức mạng HTTP, có hỗ trợ mã hóa. Hệ điều hành là Windows CE 2.11 hoặc cao hơn, và có thể sử dụng với cơ sở dữ liệu đến 2 GB tuy rằng, hiện tại chưa có thiết bị CE nào có đến 2 GB RAM.

Sau đây, chúng ta sẽ tìm hiểu cách thức tạo các cơ sở dữ liệu, các bảng, các quan hệ, ...

Công cụ Enterprise Manager

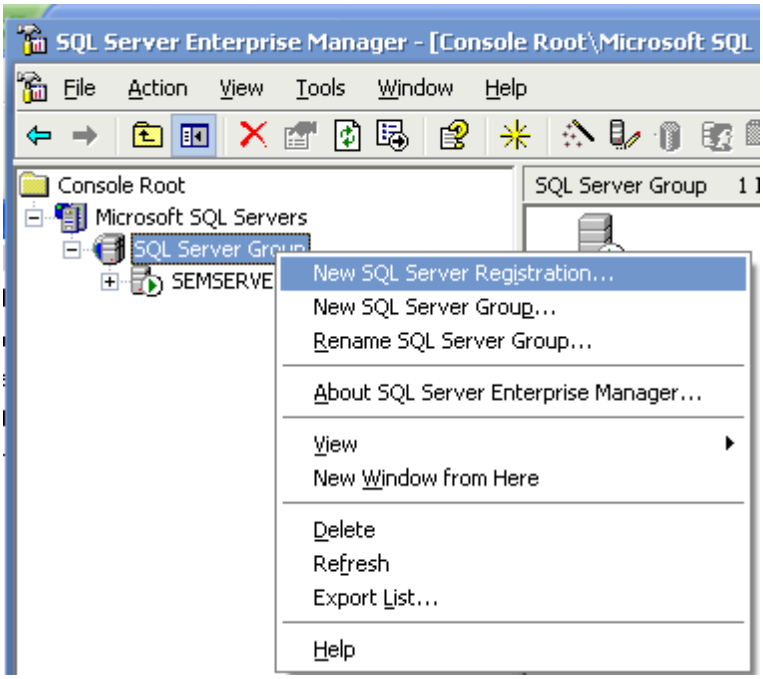
Như đã nêu trên, công cụ quản trị trực quan cơ sở dữ liệu được bán kèm trong các phiên bản như Standard, Enterprise, Developer. Công cụ này có tên là Enterprise Manager, giúp người sử dụng có thể thực hiện các thao tác

quản trị cơ sở dữ liệu một cách trực quan và dễ dàng. Màn hình chính của công cụ này như sau:



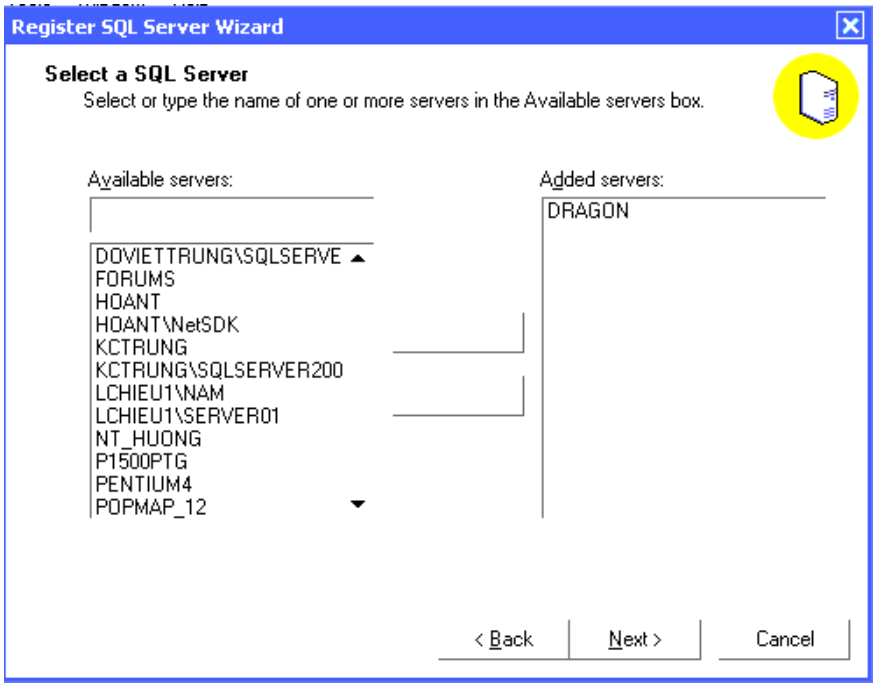
Hình 4.3.2.1 Màn hình chính công cụ Enterprise Manager

Công cụ này giúp chúng ta thực hiện các công việc liên quan đến cơ sở dữ liệu một cách dễ dàng. Điều đầu tiên là thực hiện đăng ký một mục cho máy chủ SQL Server mà chúng ta cần thực hiện quản trị. Để làm được điều này, nhấn chuột phải vào mục SQL Server Group, chọn **New SQL Server Registration** để đăng ký một mục mới như trong hình sau.



Hình 4.3.2.2 Đăng ký một máy chủ SQL Server mới

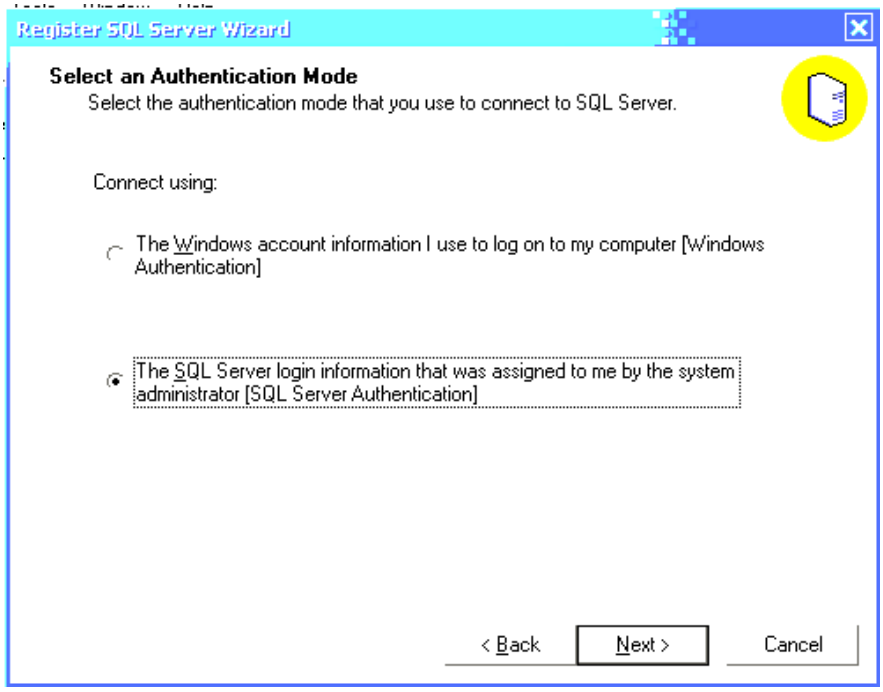
Nhấn **Next** ở màn hình Welcome đầu tiên, chương trình sẽ tự động tìm kiếm các máy chủ SQL Server đang hoạt động trên mạng nội bộ. Chọn đúng tên máy chủ SQL Server mà chúng ta cần thực hiện quản trị và nhấn **Next**:



Hình 4.3.2.3 Chọn máy chủ SQL Server cần quản trị

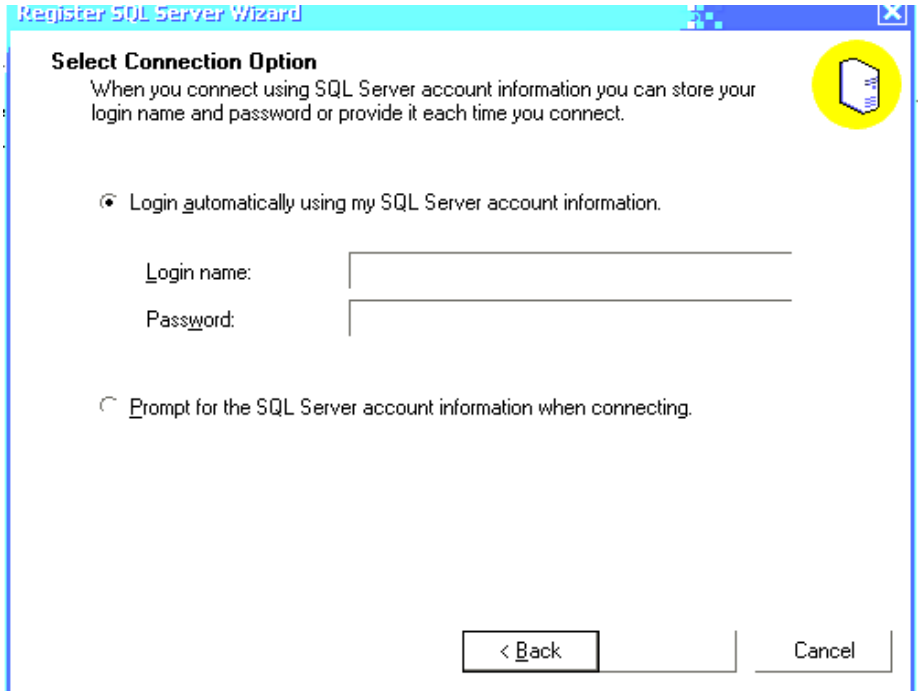
Màn hình tiếp theo cho phép chúng ta lựa chọn cách thức đăng nhập SQL Server thông qua tài khoản truy nhập mạng hoặc thông qua người sử dụng (được gán quyền)của SQL Server đó. Đối với trường hợp thứ nhất, chúng ta không cần phải nhập thông tin đăng nhập của SQL Server vì chúng ta đã thực hiện đăng nhập hệ thống thông qua tài khoản truy nhập mạng (do máy chủ SQL Server cung cấp). Chúng ta sẽ đi vào trường hợp thứ hai.

Chọn SQL Server Authentication như trong hình sau và nhấn **Next**:



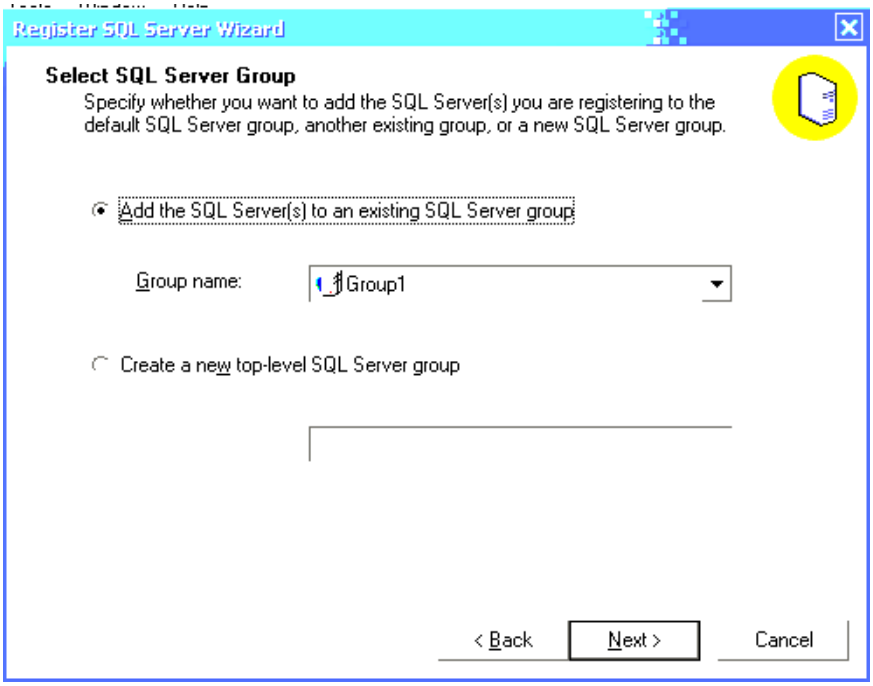
Hình 4.3.2.4 Chọn SQL Server Authentication

Trong màn hình tiếp theo, nhập các thông tin đăng nhập vào SQL Server, thông thường, các thông tin này được người quản trị hệ thống cơ sở dữ liệu cung cấp.



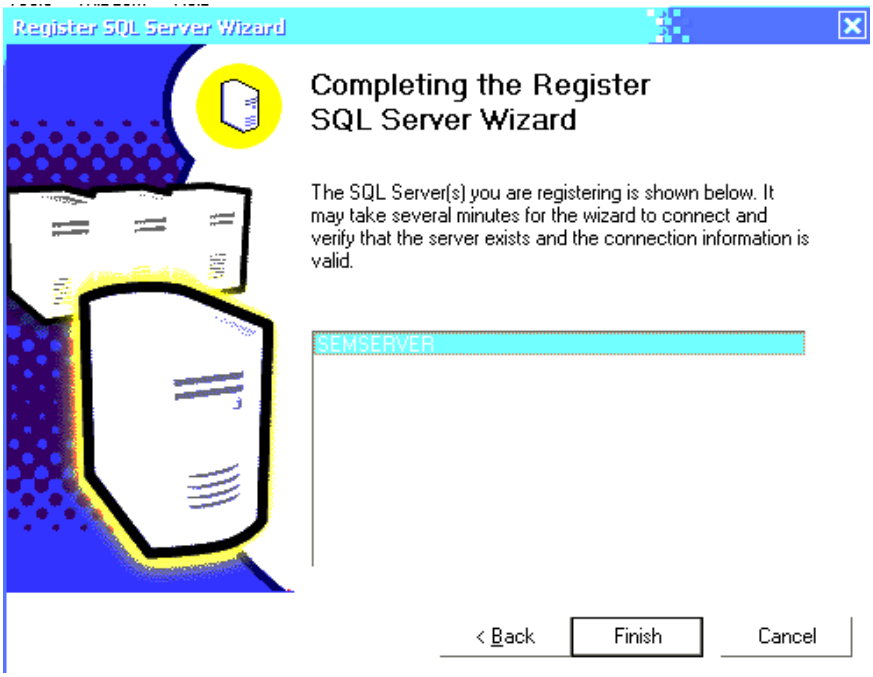
Hình 4.3.2.5 Nhập các thông tin đăng nhập

Sau khi nhập các thông tin đầy đủ, nhấn **Next** để tiếp tục và chọn nhóm mà chúng ta muốn đưa máy chủ đó vào:



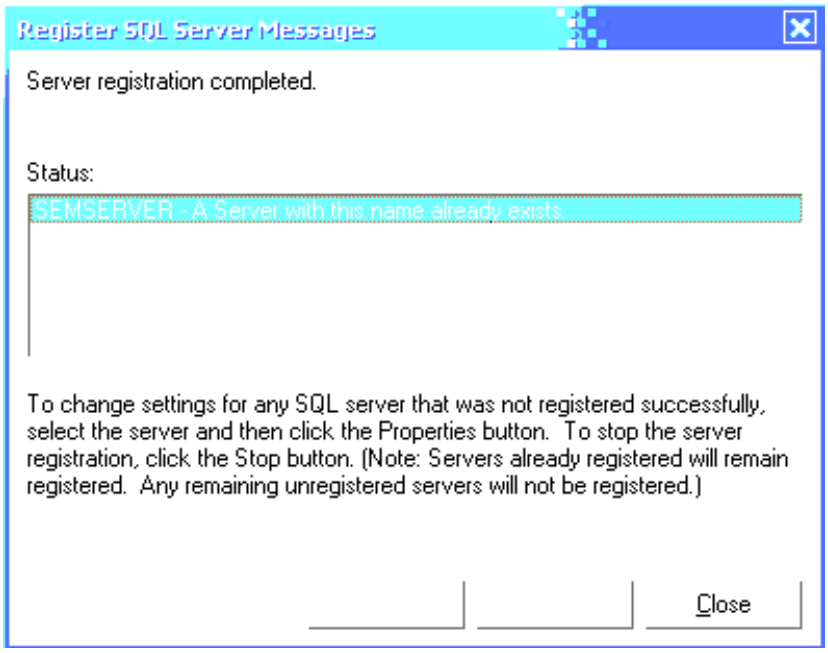
Hình 4.3.2.6 **Chọn nhóm cho máy chủ SQL Server**

Nhấn **Next**, màn hình tiếp theo cho phép kiểm tra việc đăng ký có thành công hay không. Nhấn **Finish** để thực hiện kiểm tra.



Hình 4.3.2.7 Kết thúc bước đăng ký

Trong trường hợp gặp lỗi, ví dụ như trong hình sau:



Hình 4.3.2.8 Lỗi máy chủ đã được đăng ký

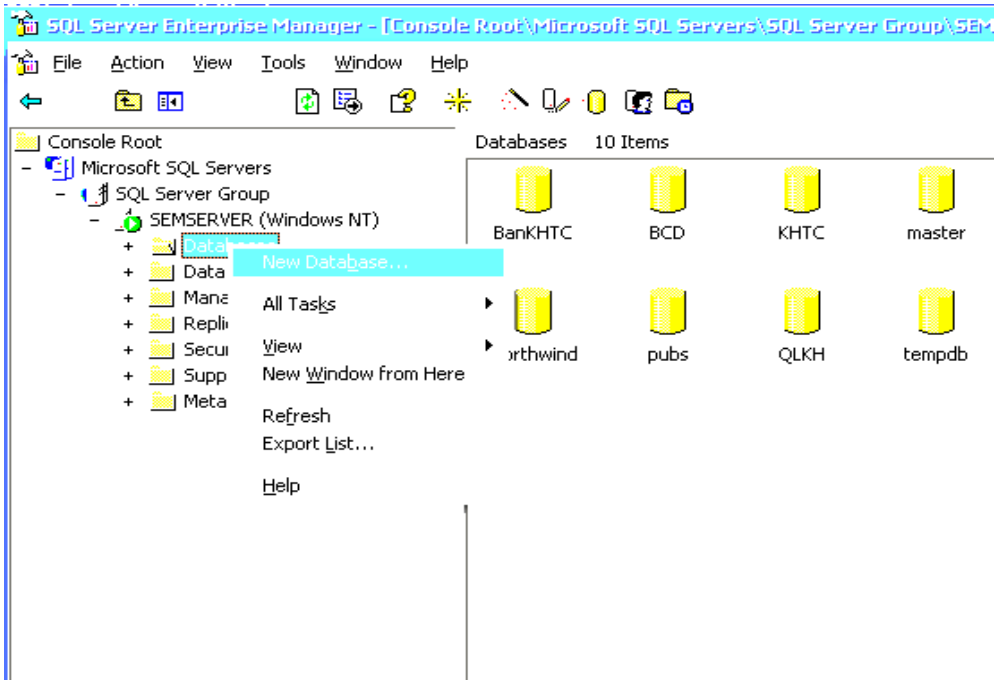
Khi gặp lỗi trên, nghĩa là chúng ta đã đăng ký máy chủ SQL Server đó từ trước rồi, vì vậy có thể bắt đầu các thao

tác khác. Nếu các thông số đều chính xác, chương trình sẽ thông báo thành công như sau:

Hình 4.3.2.9 Đăng ký thành công

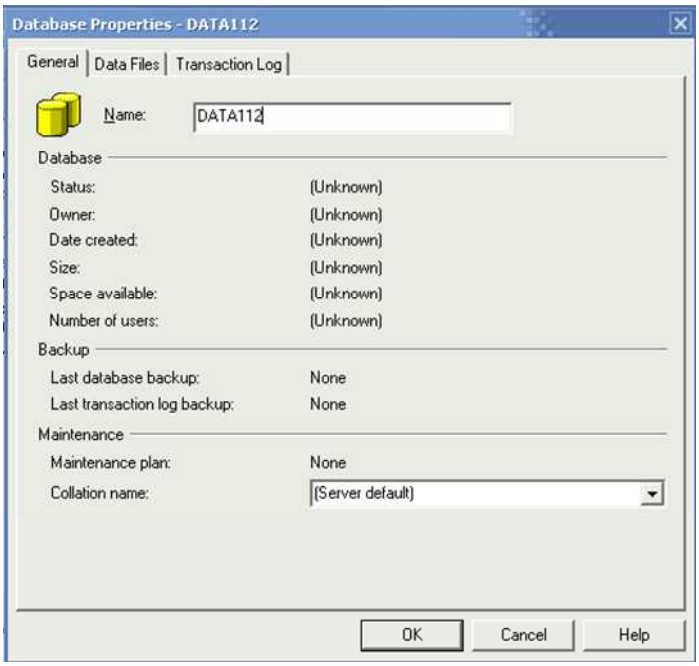
Tạo cơ sở dữ liệu

Sau khi đăng ký, chúng ta có thể thực hiện tạo lập cơ sở dữ liệu và các nội dung cho cơ sở dữ liệu. Nhấn chuột phải vào mục **Databases**, chọn mục **New Database**.



Hình 4.3.2.10 Lựa chọn tạo mới một cơ sở dữ liệu

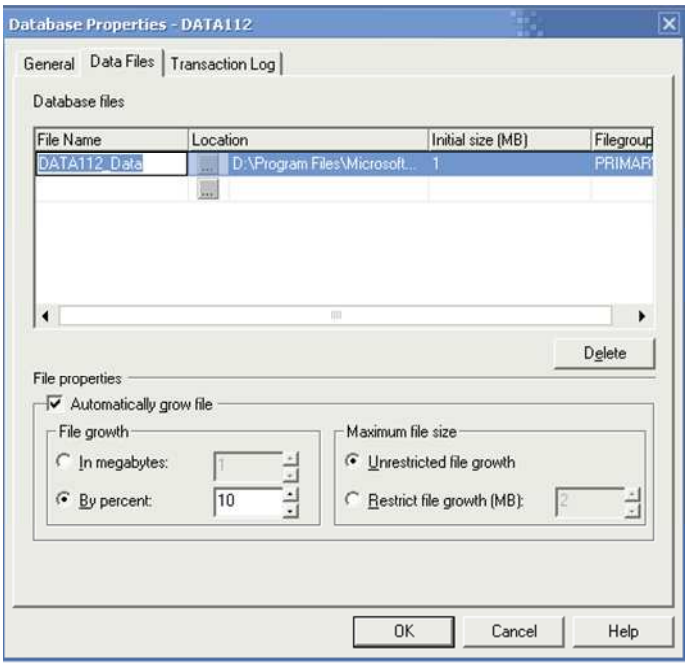
Enterprise Manager sẽ đưa ra màn hình mới để có thể tạo cơ sở dữ liệu.



Hình 4.3.2.11 Đặt tên cho một CSDL

Nhập tên cơ sở dữ liệu vào mục **Name**, đây sẽ là tên chính thức sử dụng để truy nhập cơ sở dữ liệu này. Mục **Collation name** cho phép chúng ta đặt thuộc tính của cơ sở dữ liệu, quy định cơ sở dữ liệu sẽ sử dụng cách thức sắp xếp theo bảng ký tự nào.

Ở tab **Data Files**, người sử dụng có thể thiết lập các thông số khác của cơ sở dữ liệu: tên file chứa dữ liệu cùng đường dẫn đến file đó như trong hình sau:



Hình 4.3.2.12 **Xác định đường dẫn và tên tệp của cơ sở dữ liệu**

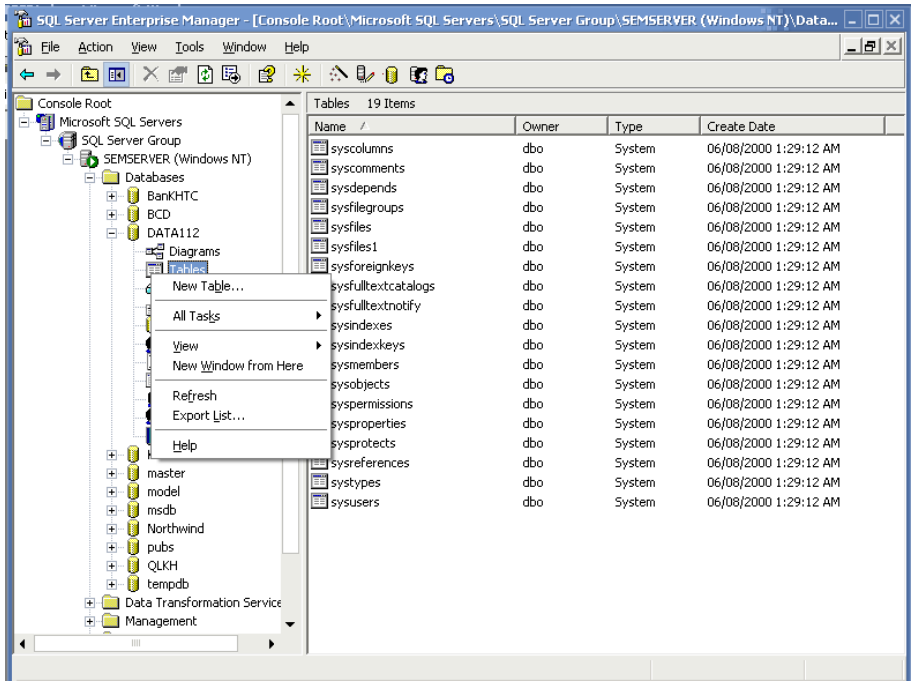
Tại đây, chúng ta có thể chấp nhận các thông số mặc định của SQL Server khuyến nghị sử dụng. Trong trường hợp chúng ta muốn thay đổi, do lượng đĩa cứng chỉ có hạn, do kích thước dữ liệu nhỏ, có thể sửa đổi các thông số này cho phù hợp với nhu cầu.

Nhấn **OK** để tạo cơ sở dữ liệu và kết thúc thao tác này. Kiểm tra lại trong cửa sổ Enterprise Manager cơ sở dữ liệu vừa tạo. Nếu chúng ta thấy cơ sở dữ liệu này hiển thị trong mục **Databases** có nghĩa là thao tác tạo cơ sở dữ liệu đã thành công.

Tạo các bảng dữ liệu và các chỉ số, khóa chính:

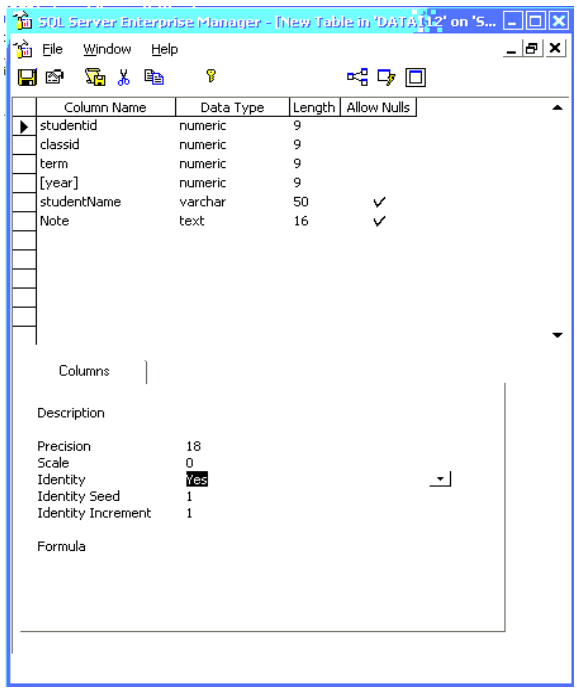
Tạo bảng dữ liệu

Sau khi tạo xong cơ sở dữ liệu, đặt tên là DATA112, chúng ta có thể thực hiện tạo các bảng dữ liệu cho cơ sở dữ liệu này. Trên khung bên trái, nhấn vào dấu + bên cạnh cơ sở dữ liệu chúng ta vừa tạo. Chọn mục **Tables** bằng cách nhấn chuột phải vào mục đó, sau đó chọn **New Table**… để tạo một bảng dữ liệu mới.



Hình 4.3.2.13 Các bảng dữ liệu ngầm định

Chương trình sẽ đưa ra cửa sổ để chúng ta có thể thiết kế bảng. Giả sử chúng ta tạo một bảng sinh viên như sau:

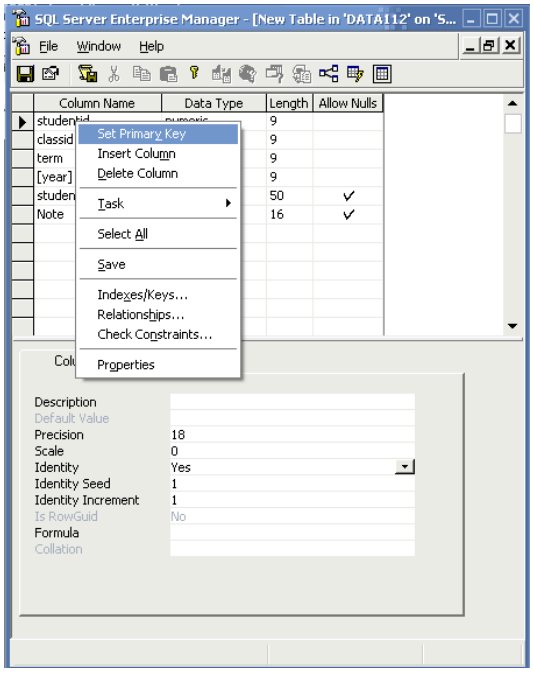


Hình 4.3.2.14 **Thiết kế bảng dữ liệu**

Nhập tên trường dữ liệu vào cột **Column Name**, kiểu dữ liệu vào cột **Data Type**, và các thông số khác như độ dài vào cột **Length**, cột **Allow Nulls** cho phép xác định cột dữ liệu có được phép để trống hay không khi nhập dữ liệu.

Tạo khóa chính

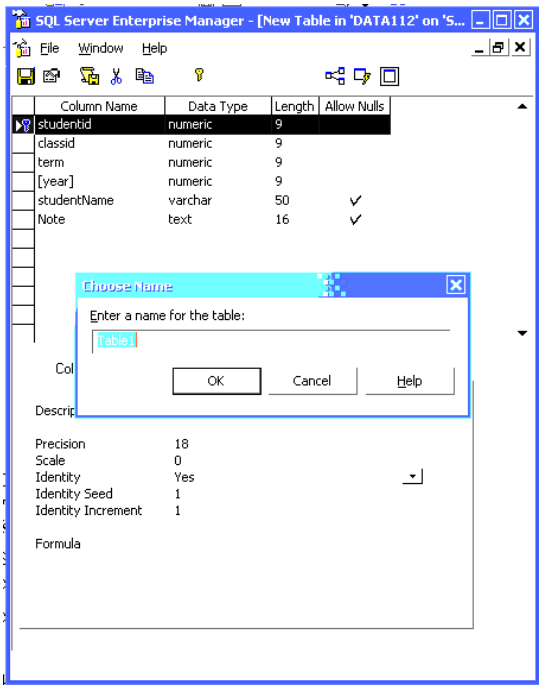
Sau khi đã nhập đủ các trường cho bảng sinh viên, có thể thiết lập khóa chính cho bảng luôn. Chọn trường cần thiết lập làm khóa chính bằng cách nhấn chuột phải vào trường đó và chọn **Set Primary Key**, như trong hình sau:



Hình 4.3.2.15 Thiết lập khóa chính cho bảng

Trong trường hợp muốn tạo khóa chính là nhiều trường, giữ phím **Control** và nhấn chuột vào vùng phía bên trái tên các trường. Sau đó cũng nhấn chuột phải và chọn **Set Primary Key**.

Sau khi chọn xong trường khóa, nhấn nút lệnh **Save** trên thanh công cụ chuẩn của SQL Server để lưu lại.

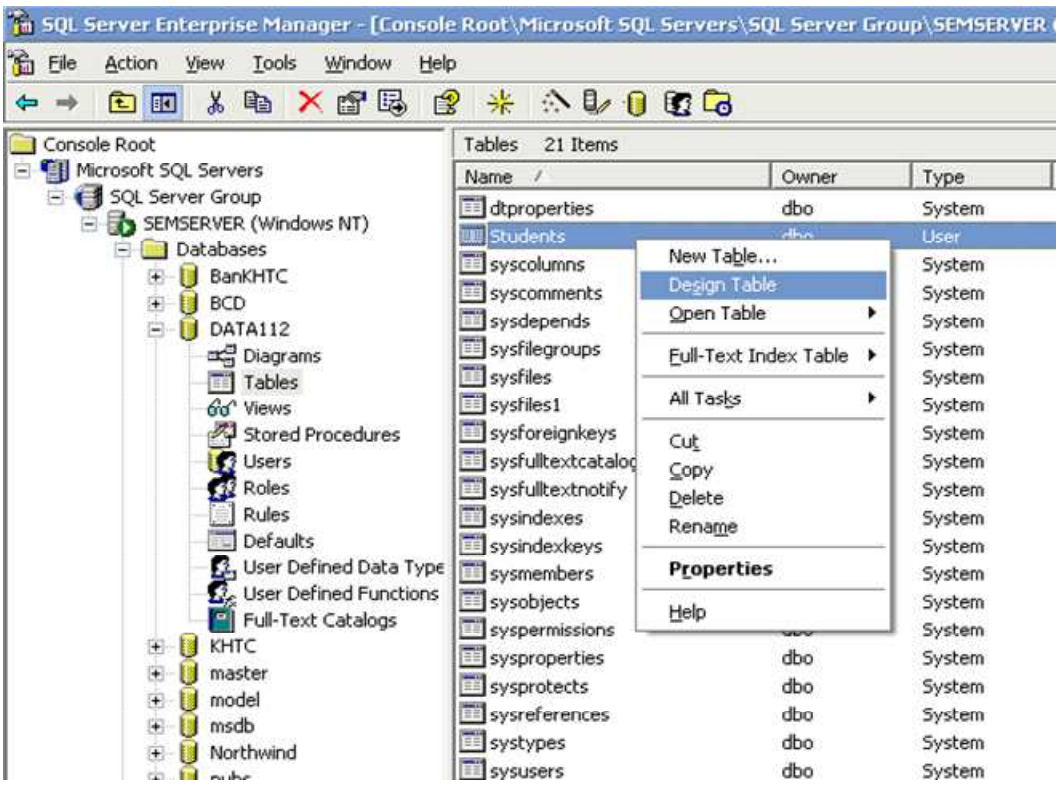


Hình 4.3.2.16 **Nhập tên cho bảng dữ liệu**

Sau đó, nhập tên cho bảng dữ liệu mới là Student, nhấn OK và đóng cửa sổ thiết kế bảng lại và quay trở lại Enterprise Manager.

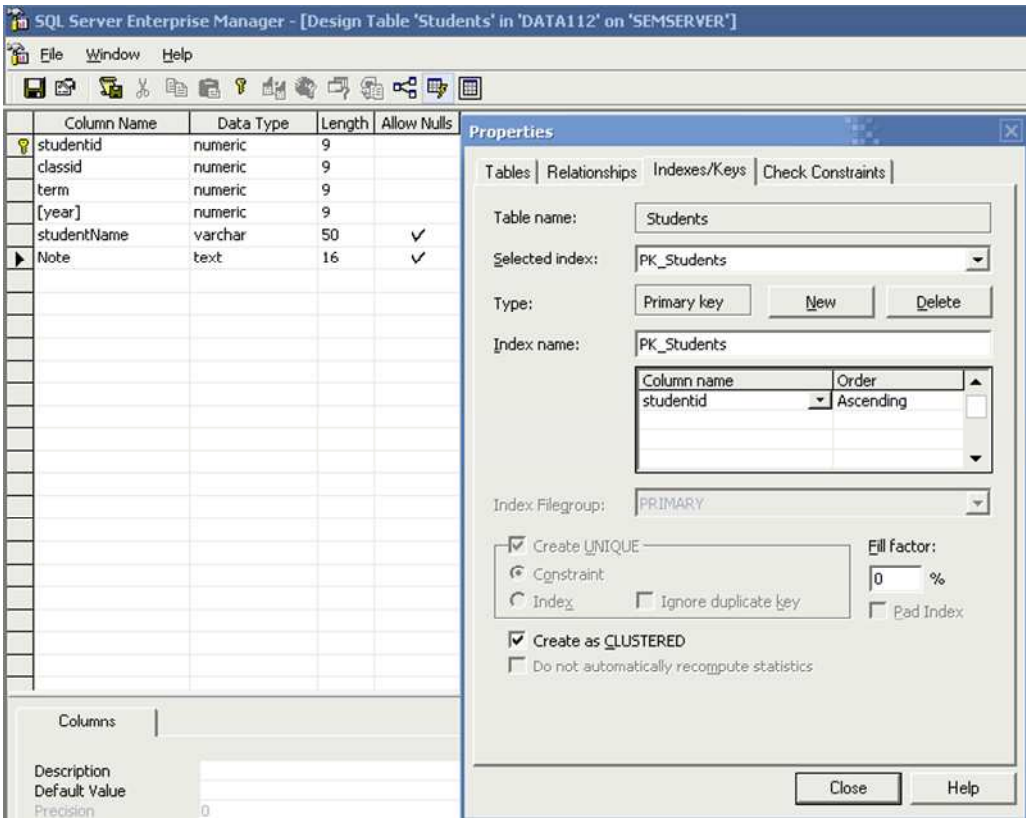
Tạo các chỉ mục

Sau khi tạo bảng và khóa chính, chúng ta tiếp tục thực hiện tạo các chỉ mục cho bảng dữ liệu này. Trong cửa sổ bên trái, tìm cơ sở dữ liệu chúng ta vừa tạo và vào mục **Tables**. Bảng **Students** mà chúng ta vừa tạo sẽ được hiển thị. Nhấn chuột phải vào bảng và chọn: **Design Table**



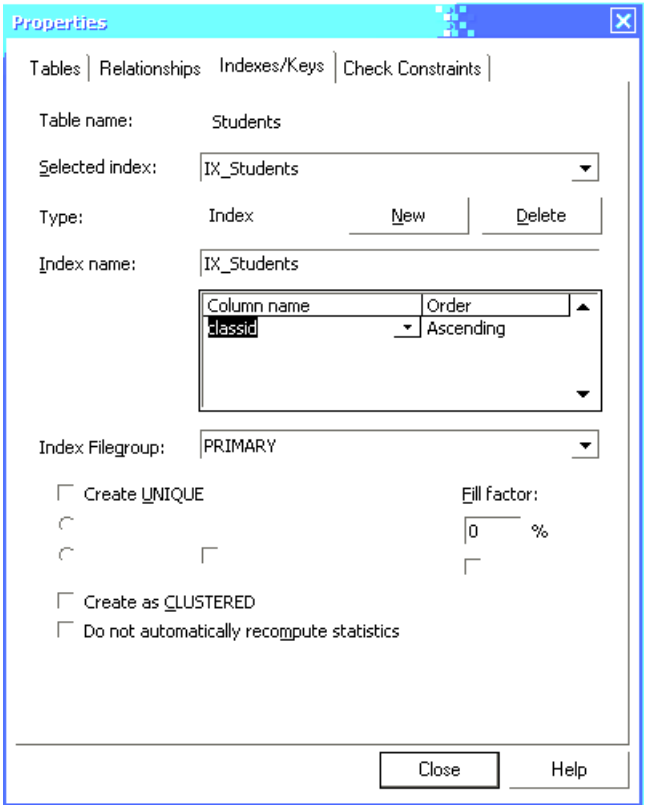
Hình 4.3.2.17 Chọn Design Table

SQL Server sẽ mở màn hình thiết kế bảng một lần nữa. Tại đây, chọn nút lệnh **Manage Indexes / Keys** trên thanh công cụ chuẩn của Enterprise Manager.



Hình 4.3.2.18 Chọn nút Manage Indexes / Keys

SQL Server sẽ mở cửa sổ mới để chúng ta thực hiện tạo, cập nhật các chỉ mục và khóa chính. Chọn nút lệnh **New** trên màn hình này:

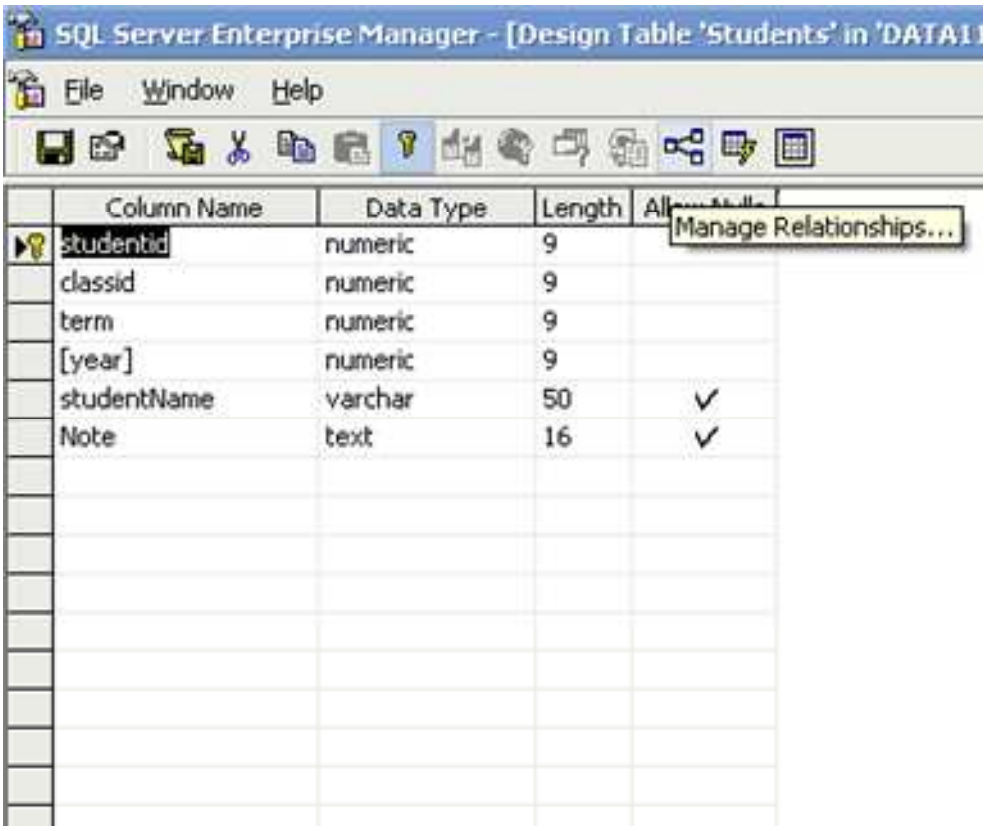


Hình 4.3.2.19 Tạo mới chỉ mục

Trong mục **Index name,** gõ tên của chỉ mục mới, sau đó, chọn cột dữ liệu mà chúng ta muốn thực hiện tạo chỉ mục trong cột **Column name**. Chọn cách thức sắp xếp của chỉ mục này bằng cách chọn trong danh sách thả xuống tại cột **Order**. Sau khi tạo xong, nhấn **Close** để đóng cửa sổ này lại.

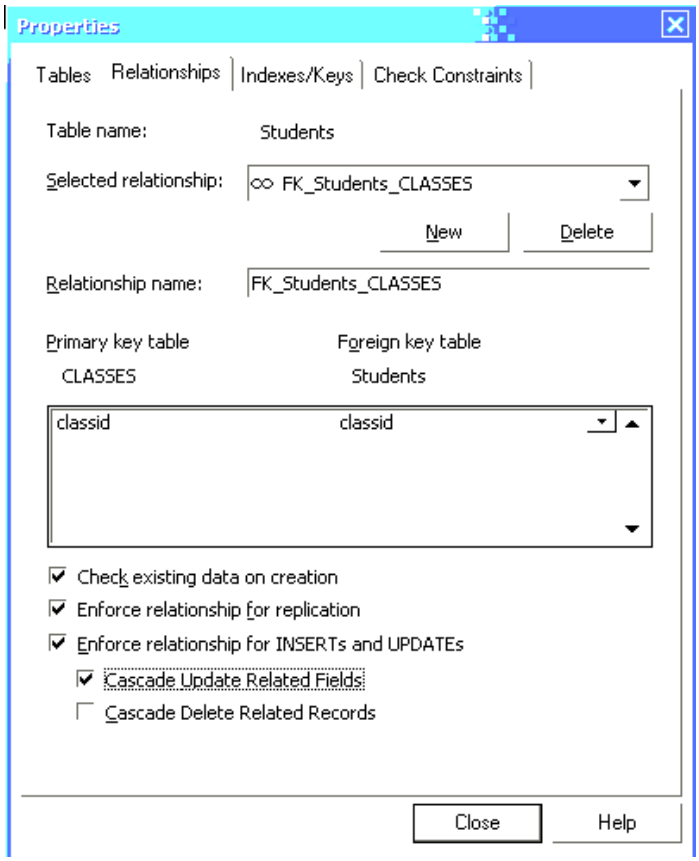
Tạo các liên kết giữa các bảng dữ liệu

Giả sử chúng ta muốn tạo liên kết giữa bảng STUDENTS với bảng có sẵn CLASSES, chúng ta chọn bảng **Students**, nhấn chuột phải và chọn Design Table.



Hình 4.3.2.20 **Nút lệnh Manage Relationships**

Sau đó, chọn nút lệnh **Manage Relationships** trên thanh công cụ chuẩn. Enterprise Manager sẽ đưa ra màn hình để quản lý, cập nhật các liên kết trên bảng mà chúng ta vừa chọn như sau.



Hình 4.3.2.21 **Tạo quan hệ giữa các bảng**

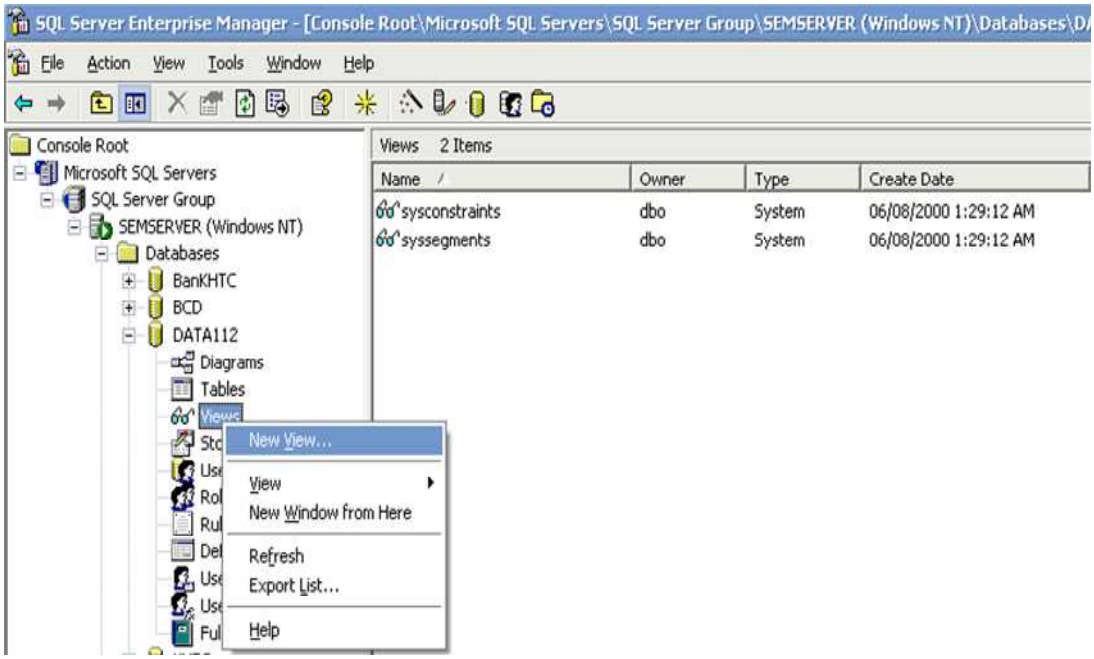
Nhấn nút lệnh **New** để tạo liên kết mới. Ở mục **Relationship Name** nhập tên của liên kết, sau đó, xác định bảng, trường ở bảng chính và xác định trường ở bảng phụ tạo ra liên kết. Các mục phía dưới cho phép chúng ta thực hiện các kiểm tra các dữ liệu ở hai bảng liên quan đến liên kết.

Tạo các khung nhìn (View)

Khung nhìn (View) thực chất là kết quả của việc lấy dữ liệu của một bảng hoặc kết nối các bảng dữ liệu với nhau. Thông thường, để tăng tốc xử lý, chúng ta có thể tạo ra các khung nhìn là kết quả của việc lấy dữ liệu trên một hoặc

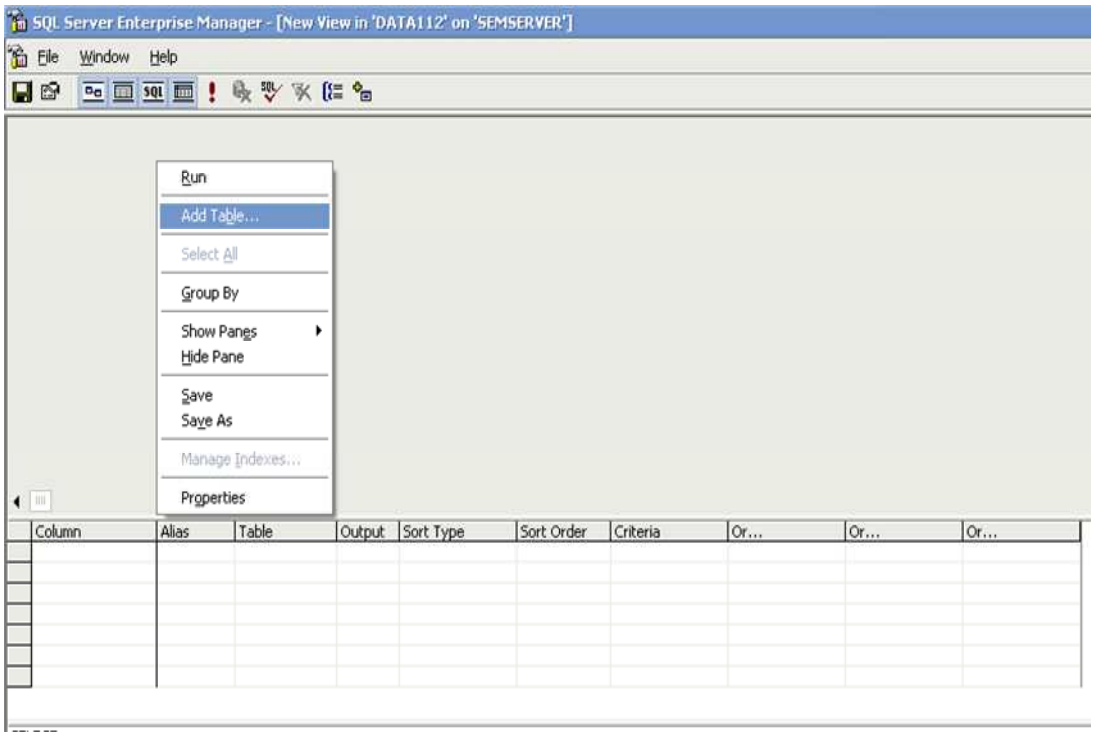
nhiều bảng trước khi thực hiện các thao tác như: hiển thị báo cáo, thiết kế các báo cáo, …

Để tạo một khung nhìn mới, trong danh sách các mục con của cơ sở dữ liệu, nhấn chuột phải vào **Views** và chọn **New View…**



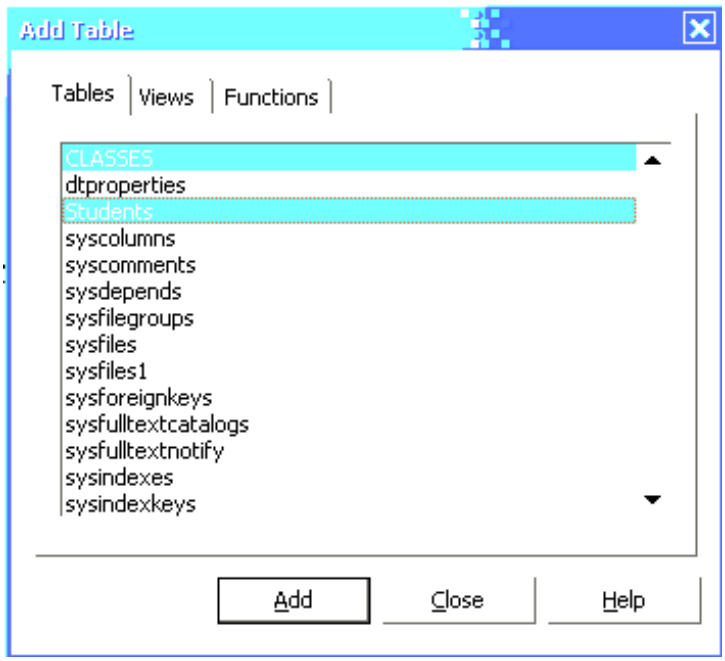
Hình 4.3.2.22 Tạo một khung nhìn mới

Bộ thiết kế khung nhìn sẽ được kích hoạt trợ giúp việc tạo mới. Để thực hiện tạo một khung nhìn, nhấn chuột phải vào vùng trống phía trên và chọn **Add Table…**



Hình 4.3.2.23 Thao tác tạo mới một khung nhìn

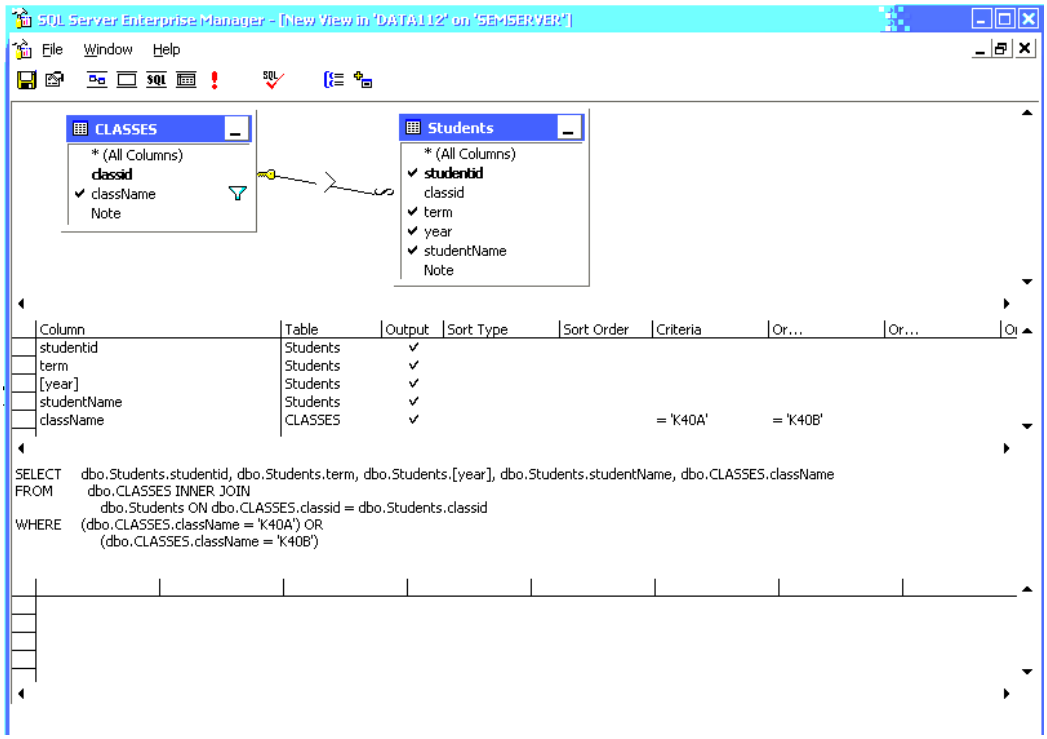
Chương trình sẽ hiển thị danh sách các bảng dữ liệu để chúng ta có thể chọn theo mục tiêu của chúng ta. Kích chuột vào tên bảng cần chọn, nếu muốn chọn nhiều bảng một lúc có thể giữ phím **Ctrl** và kích chuột vào các tên bảng.



Hình 4.3.2.24 Chọn các bảng thành phần tạo khung nhìn

Sau đó nhấn **Add** để chọn các bảng này. Sau khi chọn xong, nhấn **Close** để tiếp tục. Các bảng đã chọn sẽ hiển thị trong phần phía trên cửa sổ thiết kế khung nhìn. ở đây liên kết của các bảng cũng sẽ được hiển thị.

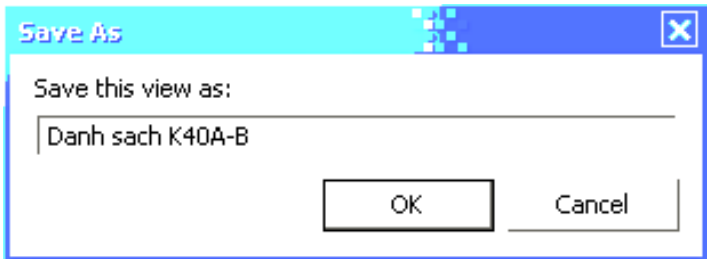
Ví dụ, chúng ta muốn lấy danh sách các sinh viên của lớp K40A và K40B, các bước cần thực hiện như trong hình sau:



Hình 4.3.2.25 Thiết kế khung nhìn

Như trong hình 4.3.2.25 trên, chúng ta cần thực hiện việc xác định các trường dữ liệu cần đưa ra khung nhìn bằng cách check vào các ô bên trái của các cột hoặc chọn trong danh sách (mục **Column**) trong phần cửa sổ thứ 2. Phần cửa sổ thứ3 hiển thị câu lệnh SQL thực tế SQL Server sẽ phải thực hiện để lấy dữ liệu. Nhấn nút lệnh ! trên thanh công cụ để xem trước kết quả.

Sau khi đã đồng ý với kết quả, chúng ta nhấn **Save** và nhập tên của khung nhìn mới.

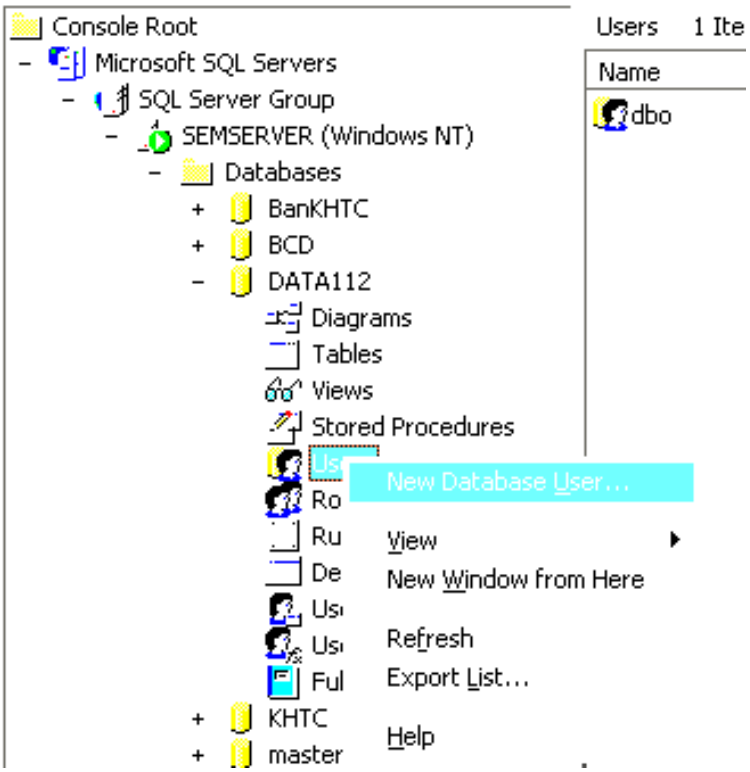


Hình 4.3.2.26 Nhập tên khung nhìn mới

Chúng ta có thể xem lại kết quả hoặc thiết kế lại các khung nhìn này vào bất kỳ thời điểm nào khi cần trong mục **Views** của cơ sở dữ liệu.

Phân quyền sử dụng các đối tượng dữ liệu cho người sử dụng

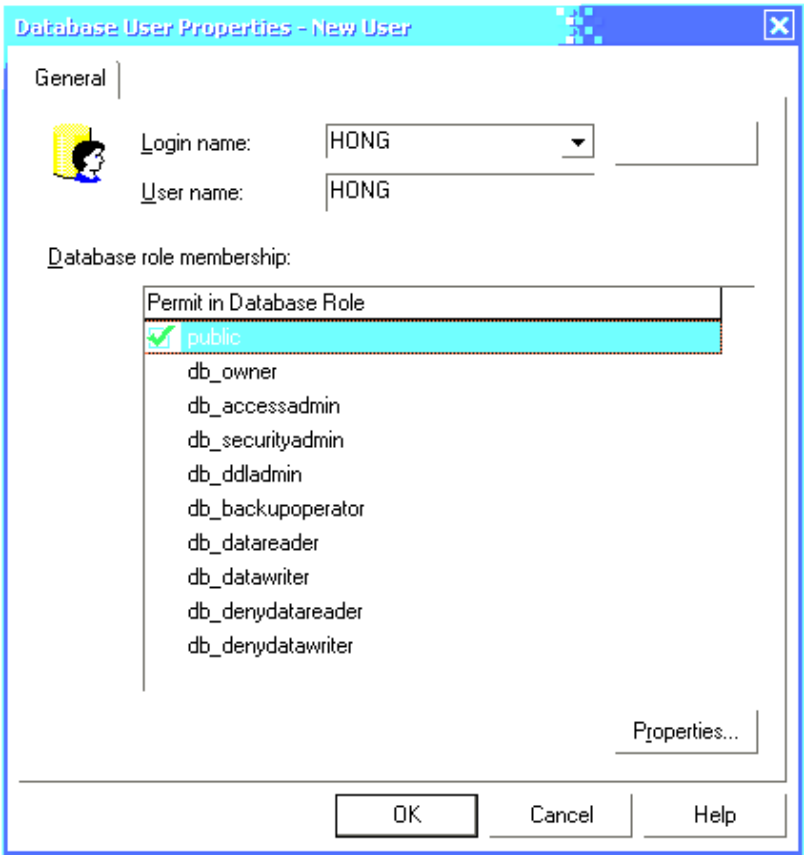
Sau khi tạo xong cơ sở dữ liệu, có thể thực hiện phân quyền sử dụng cho các người sử dụng khác trong hệ thống. Nhấn chuột phải vào mục **Users** và chọn **New User**.



Hình 4.3.2.27 **Phân quyền mới cho người sử dụng khác**

Tại màn hình kế tiếp, thực hiện chọn người sử dụng và các quyền hạn mà người sử dụng đó có thể thực hiện trên cơ sở dữ liệu. Cần chú ý rằng, danh sách người sử dụng là do người quản trị hệ thống tạo ra, và chúng ta không có quyền

thì không tạo được người sử dụng mới. Nếu có nhu cầu, hãy liên hệ với quản trị hệ thống cơ sở dữ liệu để tạo người sử dụng này. Màn hình để thực hiện việc này như hình 4.3.2.28 sau:



Hình 4.3.2.28 Gán quyền sử dụng cơ sở dữ liệu cho người sử dụng khác

Chọn người sử dụng trong mục **Login Name** và gán các quyền cho người sử dụng này tại mục **Database role membership**. Sau khi chọn xong, nhấn **OK** để kết thúc. Người sử dụng được gán quyền sẽ có thể truy cập và thực hiện các thao tác mà người tạo cơ sở dữ liệu (db_owner: chủ cơ sở dữ liệu) đã gán cho.

b. Hệ quản trị cơ sở dữ liệu Oracle

Một hệ quản trị cơ sở dữ liệu có thể nói là lớn nhất, chặt chẽ nhất và mạnh nhất trên thị trường phần mềm hệ quản

trị cơ sở dữ liệu hiện nay chính là hệ quản trị cơ sở dữ liệu ORACLE, do hãng Oracle cung cấp và phát triển. Hệ này hỗ trợ phát triển các hệ thống cơ sở dữ liệu cũng như các chương trình ứng dụng chạy trên hầu hết các hệ điều hành máy tính hiện nay như Windows, Unix, Linux, …

Trước tiên, chúng ta sẽ đi vào tìm hiểu về cấu trúc một cơ sở dữ liệu Oracle. Về tổng thể, một cơ sở dữ liệu Oracle có một tầng logic và một tầng vật lý. Tầng vật lý bao gồm các file nằm trên đĩa, còn tầng logic ánh xạ dữ liệu tới các thành phần vật lý nêu trên.

Tầng vật lý

Tầng vật lý của một cơ sở dữ liệu bao gồm 3 loại file:

Một hoặc nhiều file chứa dữ liệu (datafiles) chứa các thông tin được lưu trữ trong cơ sở dữ liệu. Chúng ta có thể có từ một đến hàng trăm file dữ liệu. Thông tin trong một bảng dữ liệu có thể được mở rộng và chứa trong nhiều file dữ liệu và tương tự, nhiều bảng dữ liệu có thể chứa trong cùng một file. Việc thực hiện trải các không gian bảng (tablespaces) trên nhiều file dữ liệu có thể mang lại các ảnh hưởng tích cực đến hiệu năng của toàn bộ hệ thống. Số file dữ liệu có thể thực hiện cấu hình được quy định bằng tham số MAXDATAFILES.

Hai hay nhiều các file redo (redo log files) chứa các thông tin sử dụng cho việc phục hồi trong trường hợp hệ thống hỏng hóc. Các file này lưu vết của tất cả các thay đổi được thực hiện trên cơ sở dữ liệu. Các vết này được sử dụng để phục hồi lại các thông tin thay đổi và lưu lại mà chưa được ghi vào các file dữ liệu. Vì vậy, nếu mất các file này, hệ

thống sẽ không thể được phục hồi nếu như hệ thống cơ sở dữ liệu bị hỏng.

Một hay nhiều các tệp điều khiển chứa các thông tin sử dụng cho việc khởi động một thể hiện (instance), ví dụ như: vị trí lưu các file dữ liệu và các file redo; Oracle cần các thông tin này để khởi động thể hiện của cơ sở dữ liệu. Các file điều khiển (Control files) cần phải được bảo vệ, do vậy Oracle cung cấp giải pháp là lưu nhiều bản sao của các file điều khiển này.

Tầng Logic

Tầng logic bao gồm các thành phần sau:

Một hoặc nhiều các không gian bảng (tablespaces).

Các lược đồ cơ sở dữ liệu (database schema), chứa các mục như: các bảng (tables), các phân đoạn (clusters), các chỉ mục (indexes), các khung nhìn (views), các thủ tục (stored procedures), các thủ tục kích hoạt cho cơ sở dữ liệu (database triggers), các dãy tự tăng (sequences), ...

Sau đây, chúng ta sẽ đi sâu nghiên cứu về các khái niệm, tác dụng và chức năng của các thành phần trong tầng logic, từ đó có được cái nhìn tổng thể và bao quát hơn về cơ sở dữ liệu Oracle.

Tablespaces và Datafiles

Cơ sở dữ liệu được chia thành một hoặc nhiều các phần logic, các phần này chính là các không gian bảng (tablespaces). Một tablespace được sử dụng để nhóm các dữ liệu vào với nhau một cách logic. Ví dụ: chúng ta có thể tạo một tablespace riêng cho tác nghiệp và một tablespace cho điều hành. Việc phân chia này sẽ đơn giản hóa việc

quản trị các nhóm này. Mỗi tablespace chứa một hoặc nhiều file dữ liệu (datafiles). Cách sử dụng nhiều file dữ liệu cho phép trải dữ liệu lưu trữ trên nhiều đĩa khác nhau để phân tải các thao tác vào ra dữ liệu có thể giúp tăng hiệu năng của hệ thống.

Trong quá trình tạo mới cơ sở dữ liệu, Oracle tự động tạo tablespace tên là SYSTEM. Mặc dù, đối với một hệ cơ sở dữ liệu nhỏ chúng ta có thể sử dụng ngay tablespace này, tuy nhiên, vẫn nên tạo thêm một tablespace riêng cho các dữ liệu người dùng bởi lẽ, tablespace SYSTEM được sử dụng cho các từ điển dữ liệu bao gồm: các bảng, các chỉ mục, …

Các file dữ liệu (datafiles) có thể là các file thuộc hệ điều hành hoặc trong một số hệ điều hành, nó là các RAW device, thiết bị giả. Vì vậy, việc truy nhập vào các file dữ liệu cũng phụ thuộc vào các hệ điều hành cụ thể.

Lược đồ cơ sở dữ liệu (Database Schema)

Lược đồ cơ sở dữ liệu là một tập hợp các đối tượng, hay còn gọi là các đối tượng lược đồ, thuộc tầng logic cho chúng ta hình dung một cơ sở dữ liệu bao gồm những dữ liệu kiểu gì. Các đối tượng này gồm các cấu trúc như: các bảng, các phân đoạn, các chỉ mục, các khung nhìn, các thủ tục, các kích hoạt cơ sở dữ liệu, và các dãy tự tăng.

Bảng dữ liệu

Một bảng được định nghĩa bao gồm tên, các hàng và các cột dữ liệu. Bảng là đơn vị lưu trữ dữ liệu logic cơ bản trong cơ sở dữ liệu Oracle. Các cột được định nghĩa với tên và kiểu dữ liệu. Thường thì một bảng được lưu trong một tablespace và có nhiều bảng cùng sử dụng một tablespace.

Một phân cụm – cluster – là một tập các bảng được lưu trữ trên thiết bị vật lý cùng nhau mà chia sẻ một cột dữ liệu chung. Nếu dữ liệu ở 2 hoặc nhiều bảng thường được lấy ra cùng nhau dựa trên cột dùng chung, khi đó, sử dụng các phân cụm là khá hiệu quả. Các bảng có thể được truy nhập riêng rẽ mặc dù chúng thuộc một bảng đã được phân cụm. Do cấu trúc như vậy nên các dữ liệu có liên quan sẽ yêu cầu ít hơn thời gian thực hiện các thao tác vào/ra nếu được truy cập đồng thời.

Chỉ mục (Index) – Một chỉ mục được tạo lập nhằm tăng cường khả năng lấy dữ liệu trên các bảng. Chỉ mục được định nghĩa dựa trên một hoặc một tập các cột. Khi chúng ta thực hiện các truy vấn dựa trên giá trị của cột đã được đánh chỉ mục (dùng trong mệnh đề WHERE của SQL) thì chỉ mục sẽ được sử dụng để xác định dữ liệu trong bảng.

Khung nhìn (View) – Trong Oracle thì khung nhìn là cửa sổ nhìn vào trong dữ liệu của một hoặc nhiều bảng. Một khung nhìn có thể truy cập và thực hiện truy vấn, cập nhật và xóa dữ liệu như một bảng mà không có sự ràng buộc nào. Thực chất, các khung nhìn được sử dụng như một cách làm đơn giản hóa nhận thức của người sử dụng về việc truy cập dữ liệu bằng cách cung cấp các thông tin giới hạn từ một bảng hay một tập các thông tin từ vài bảng. Khung nhìn cũng có thể được sử dụng để ngăn chặn các dữ liệu mà chúng ta không muốn người sử dụng truy cập hay tạo các liên kết giữa nhiều bảng dữ liệu.

Các thủ tục (Stored procedure)—Có thể hiểu các stored procedure như một câu truy vấn được định nghĩa trước lưu trong từ điển dữ liệu. Chúng được thiết kế nhằm tăng cường khả năng truy vấn dữ liệu do giảm được lượng thông

tin phải trao đổi giữa các máy khách và hệ quản trị cơ sở dữ liệu.

Kích hoạt cơ sở dữ liệu (Database trigger)—Đây là một dạng thủ tục được kích hoạt tự động khi một sự kiện nào đó xảy ra, chẳng hạn như: thêm mới, cập nhật hay thậm chí là thực hiện truy vấn dữ liệu trên một bảng nào đó.

Dãy tự tăng (Sequence)—Bộ tạo dãy tự tăng, Oracle sequence generator, được sử dụng để tự động tạo một dãy số tự tăng trong bộ đệm, giúp tránh việc tạo bằng tay rất tốn thời gian: khóa bảng – lấy giá trị cuối dãy – tạo giá trị mới – bỏ khóa bảng.

Phân đoạn, phần mở rộng, và các khối dữ liệu (Segments, Extents, and Data Blocks)

Bên trong Oracle, không gian dùng lưu trữ dữ liệu được kiểm soát bằng cách sử dụng các cấu trúc logic bao gồm:

Các khối dữ liệu (Data blocks) - Một khối là đơn vị lưu trữ nhỏ nhất trong một cơ sở dữ liệu Oracle. Các khối này chứa các thông tin trong phần đầu liên quan đến chính nó và dữ liệu chứa trong đó. Nói đúng nhất, các khối này được lưu trực tiếp trên các đĩa cứng nhưng chúng ta hoàn toàn có thể điều chỉnh kích thước các khối dữ liệu nhằm tăng hiệu quả truy cập tùy thuộc vào ứng dụng và hệ điều hành. (Mặc định là 2KB trong phần lớn các hệ thống)

Phần mở rộng (Extents) - chứa các khối dữ liệu. Một phần mở rộng được sử dụng để giảm thiểu các lãng phí lưu trữ vật lý (các vùng dữ liệu rỗng). Khi dữ liệu được nhập ngày càng nhiều vào cơ sở dữ liệu, các phần mở rộng này có thể được tăng trưởng (hoặc thu gọn lại khi cần thiết). Theo cách như vậy, nhiều tablespace có thể chia sẻ cùng một

không gian lưu trữ mà không cần phải xác định trước các phân khu giữa các tablespace này.

Trong quá trình tạo tablespace, chúng ta có thể xác định số tối thiểu các phần mở rộng cần thiết lập cũng như số các phần mở rộng này cần thêm khi trật tự này được thiết lập. Cách sắp xếp này cho phép quản lý hiệu quả không gian sử dụng trong cơ sở dữ liệu.

Phân đoạn (Segments)- Một phân đoạn là một tập của các phần mở rộng sử dụng để lưu một kiểu dữ liệu đặc thù.

Một cơ sở dữ liệu Oracle có thể sử dụng 4 kiểu phân đoạn:

Phân đoạn dữ liệu – Lưu dữ liệu người dùng trong cơ sở dữ liệu.

Phân đoạn chỉ mục - Lưu các chỉ mục.

Phân đoạn quay lui (Rollback segment) – Lưu thông tin sử dụng khi cần phục hồi lại thông tin trước khi thay đổi.

Phân đoạn tạm thời (Temporary segment) – Được tạo ra khi một câu lệnh SQL cần một vùng làm việc tạm thời. Các phân đoạn tạm thời sẽ được hủy bỏ khi câu lệnh SQL đã kết thúc. Các phân đoạn này thường dùng cho các hoạt động của cơ sở dữ liệu, chẳng hạn như: sắp xếp, làm các phép hợp, giao các tập dữ liệu,...

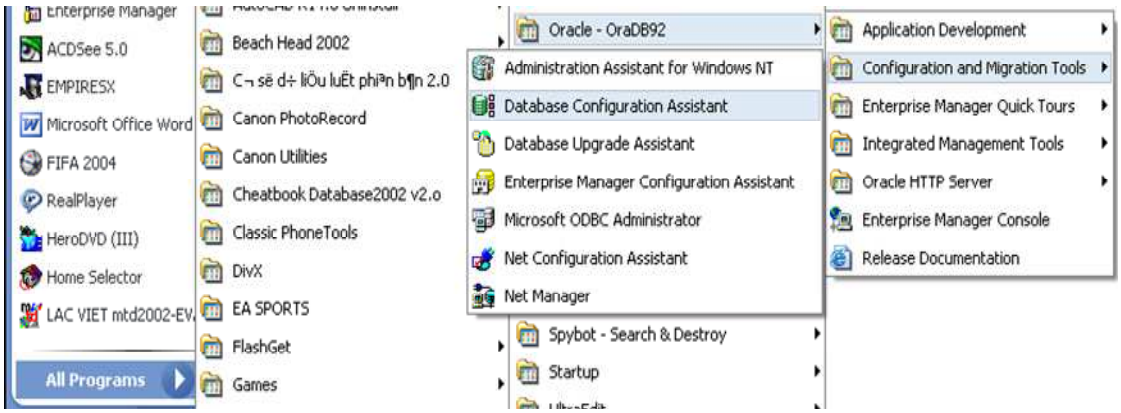
Các thao tác cơ bản

Chúng ta vừa xem xét các khái niệm cơ bản của một cơ sở dữ liệu Oracle. Cũng cần lưu ý rằng, riêng thao tác tạo lập cơ sở dữ liệu mới thường được người quản trị hệ thống thực hiện nên người sử dụng không nên tùy tiện thực hiện mặc dù các thao tác này cũng khá dễ thực hiện. Các thao

tác cơ bản như tạo bảng, tạo chỉ mục…sẽ được bàn chi tiết về cách thức thực hiện dưới đây:

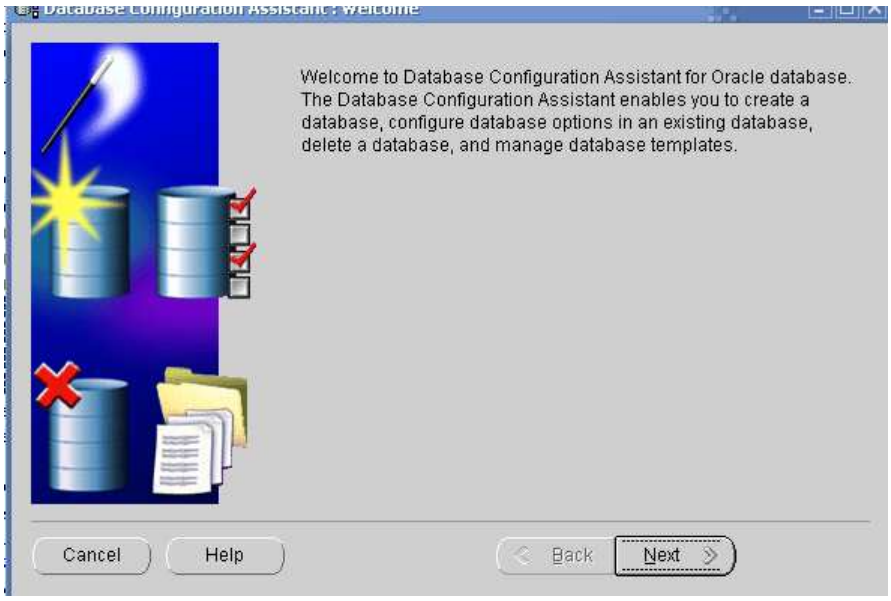
Tạo mới cơ sở dữ liệu

Để bắt đầu tạo mới một cơ sở dữ liệu, chúng ta sử dụng công cụ **Database Configuration Assistant** bán kèm theo bộ **Oracle Database**. Công cụ này cho phép thiết lập, tạo mới cơ sở dữ liệu Oracle. Cần lưu ý rằng, trước khi sử dụng công cụ này, cần đặt Oracle Home về đúng Oracle Home của Database Oracle.



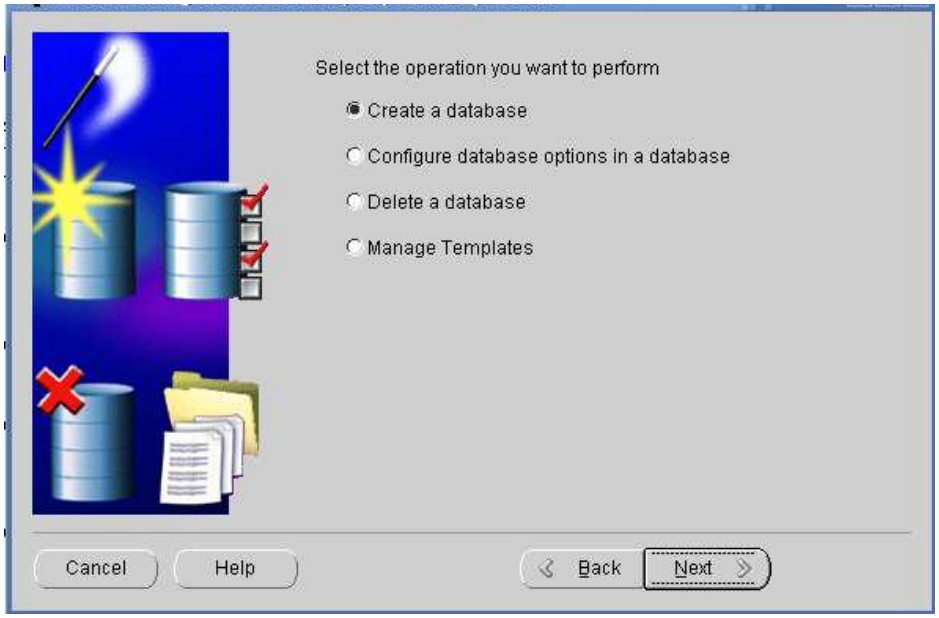
Hình 4.3.2.29 Kích hoạt Database Configuration Assistant

Màn hình bắt đầu của công cụ này như sau:



Hình 4.3.2.30 Màn hình khởi đầu của Database Configuration Assistant

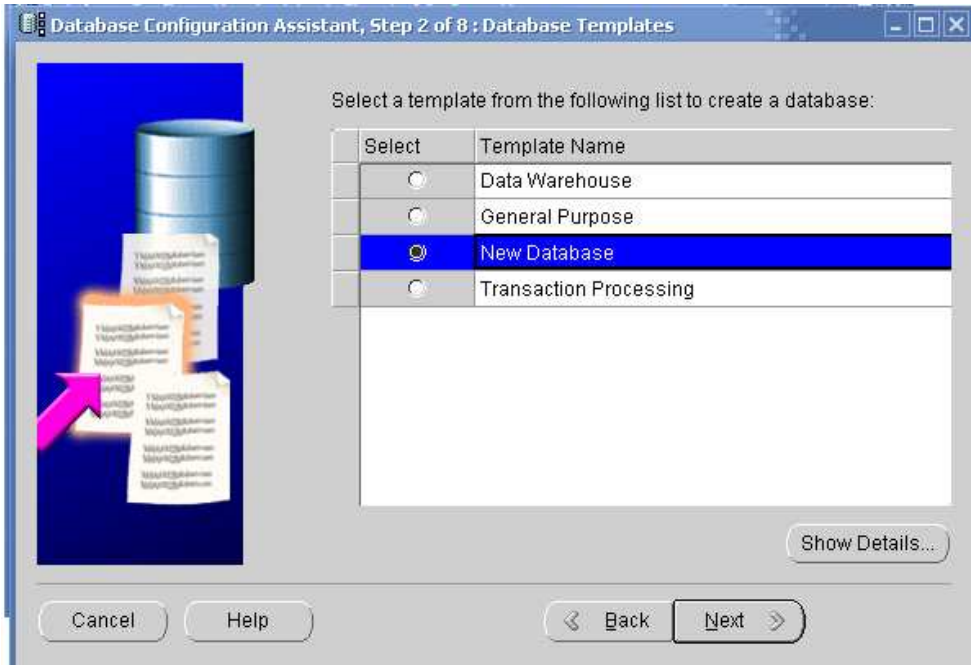
Nhấn **Next** để bắt đầu thực hiện thao tác tạo một cơ sở dữ liệu mới.



Hình 4.3.2.31 Các lựa chọn của công cụ Database Configuration Assistant

Chọn mục **Create Database** và nhấn **Next** để tiếp tục. Màn hình tiếp theo cho phép chọn một mẫu sẵn có để tạo cơ sở

dữ liệu. Chúng ta có thể chọn một trong các mục như trong hình sau:

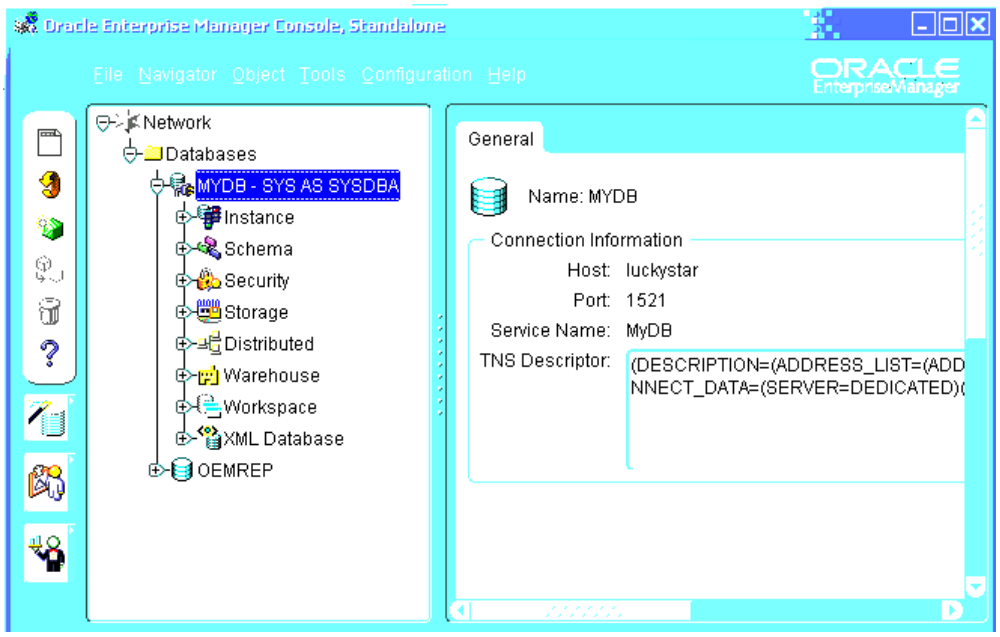


Hình 4.3.2.32 Các mẫu chọn để tạo cơ sở dữ liệu mới

Các bước tiếp theo, làm theo chỉ dẫn của công cụ để tạo cơ sở dữ liệu.

Tạo các bảng

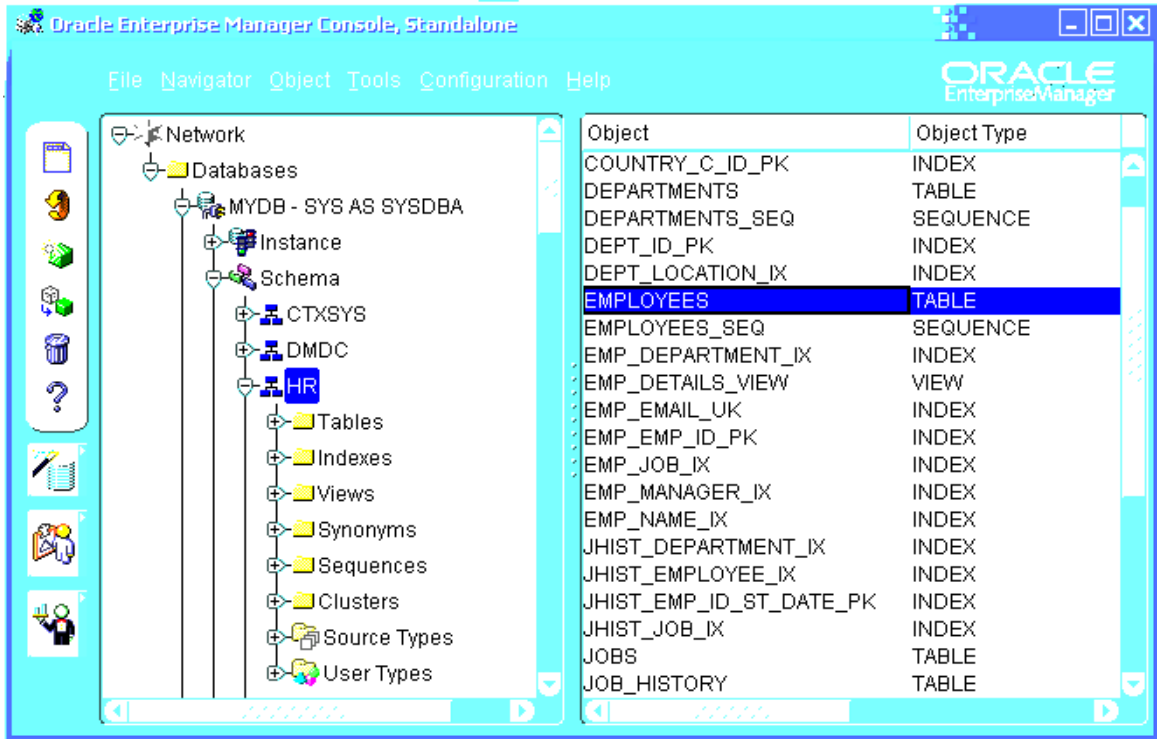
Chúng ta đã làm quen với các khái niệm như: Lược đồ cơ sở dữ liệu, bảng, chỉ mục, ... trong phần trên. Trong mục này chúng ta sẽ tìm hiểu cách thức tạo một bảng dữ liệu trong cơ sở dữ liệu Oracle. Để thực hiện điều này, chúng ta sẽ làm quen với bộ công cụ **Oracle Enterprise Manager**, một công cụ tiện ích giúp chúng ta thực hiện phần lớn các thao tác trên cơ sở dữ liệu.



Hình 4.3.2.33 Công cụ Oracle Enterprise Manager

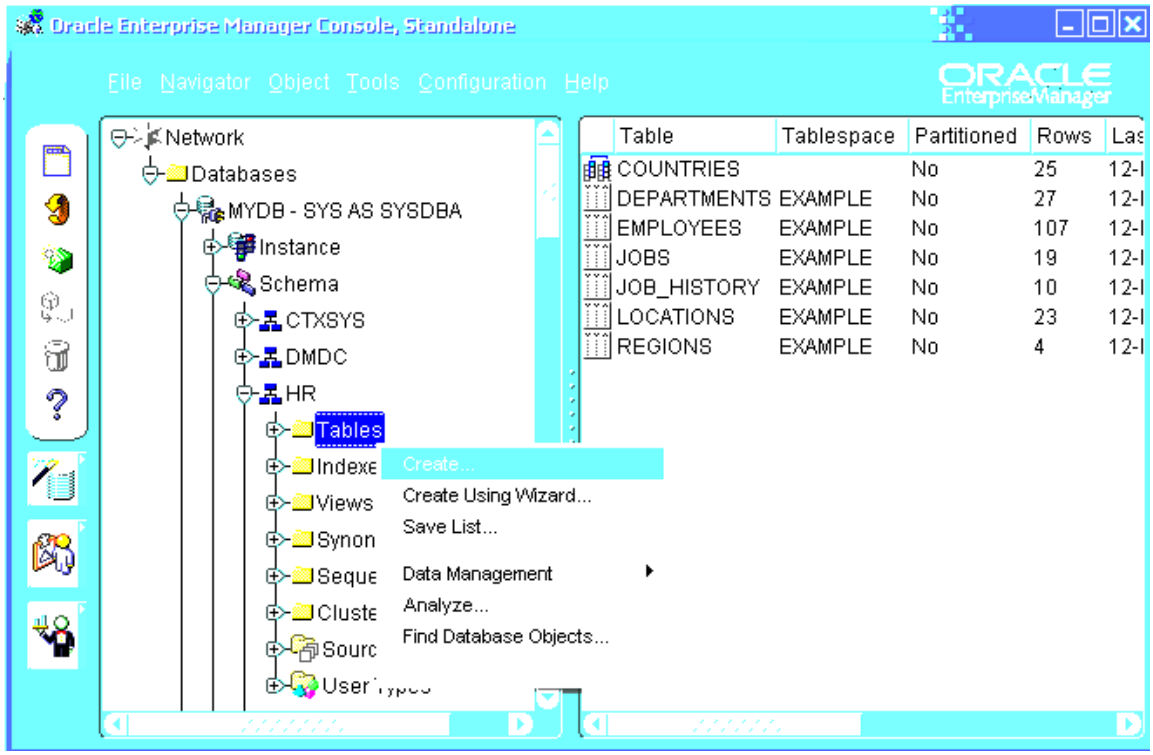
Để có thể thực hiện các thao tác với cơ sở dữ liệu, chúng ta cần thiết lập kết nối đến cơ sở dữ liệu. Hãy đọc thêm các tài liệu của Oracle để biết rõ hơn cách thực hiện thao tác này. Trong màn hình trên, **MYDB** là tên cơ sở dữ liệu (SID), phần sau dấu **–** là tên người sử dụng và vai trò của người sử dụng này trên cơ sở dữ liệu đã chọn.

Nhấn chuột vào mục **Schema** để xem thông tin của các lược đồ hiện có của cơ sở dữ liệu. Danh sách hiện ra là các lược đồ của cơ sở dữ liệu. Thực chất, các lược đồ này thuộc về từng người sử dụng.



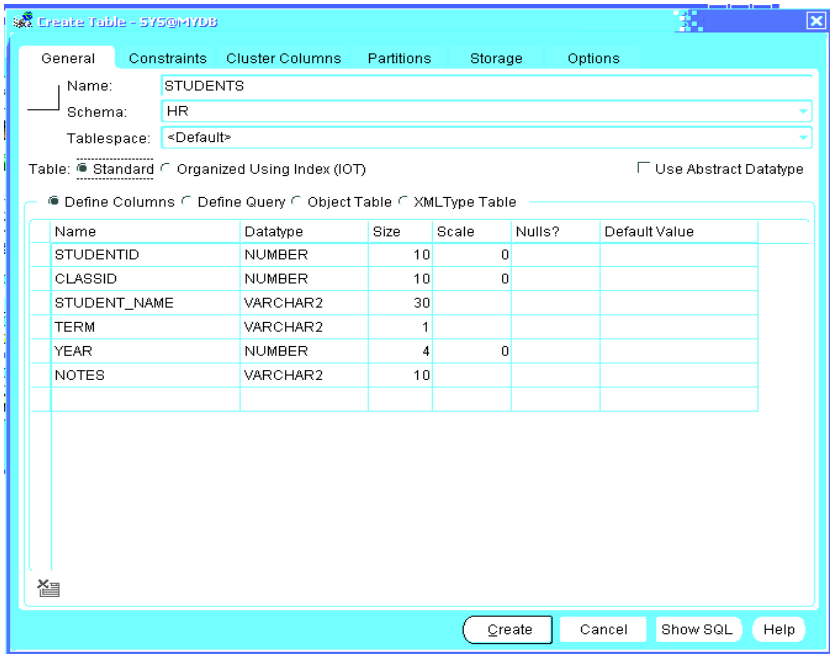
Hình 4.3.2.34 Xem các thành phần của một lược đồ

Để thực hiện tạo bảng, nhấn chuột phải vào mục **Tables** và chọn **Create**



Hình 4.3.2.35 Tạo bảng dữ liệu – bước 1

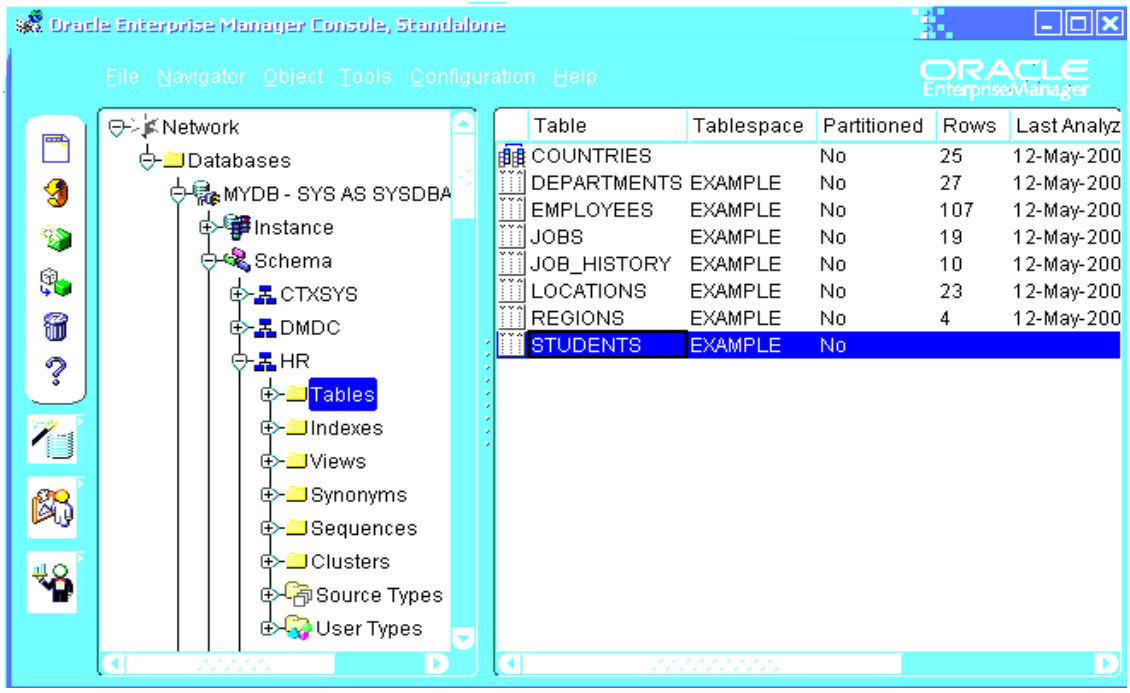
Chúng ta cũng có thể thực hiện tạo các bảng dữ liệu bằng cách sử dụng Wizard như trong cơ sở dữ liệu Access, Visual FoxPro hay SQL Server. Màn hình tiếp theo cho phép thực hiện định nghĩa cấu trúc của bảng.



Hình 4.3.2.36 Tạo bảng dữ liệu – Bước 2

Cũng tương tự như các cơ sở dữ liệu khác, chúng ta cần thực hiện nhập cấu trúc của bảng cần tạo, sau đó nhấn **Create**.

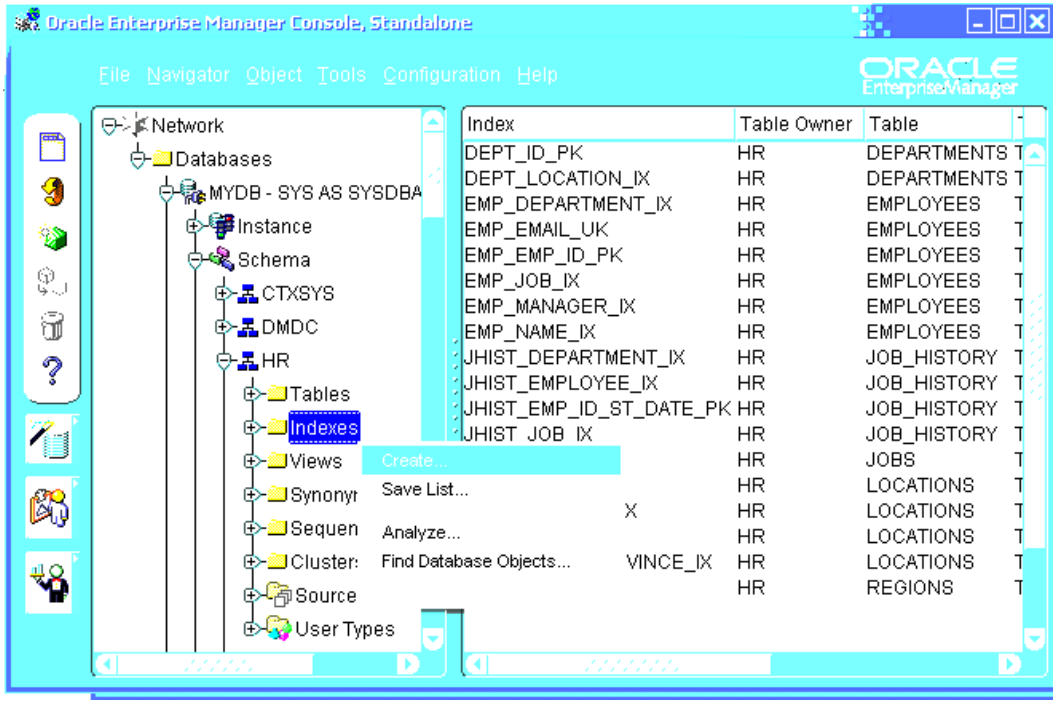
Sau khi tạo xong, chúng ta có thể kiểm tra trong màn hình Oracle Enterprise Manager



Hình 4.3.2.37 Danh sách các bảng dữ liệu

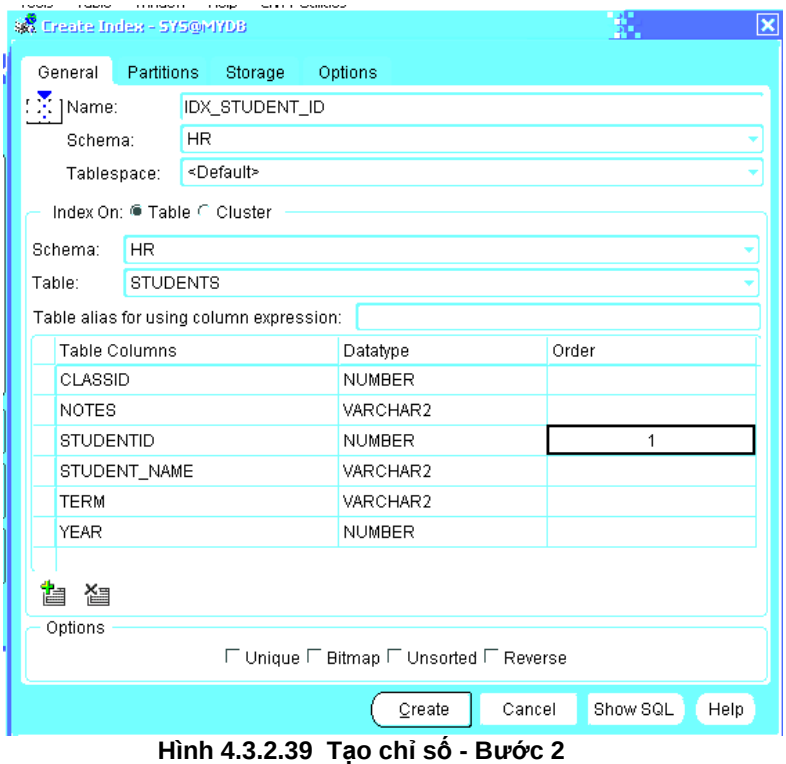
Tạo các chỉ mục

Để thực hiện tạo chỉ mục, nhấn chuột phải vào mục **Indexes** trong lược đồ đã chọn và cũng chọn **Create**.



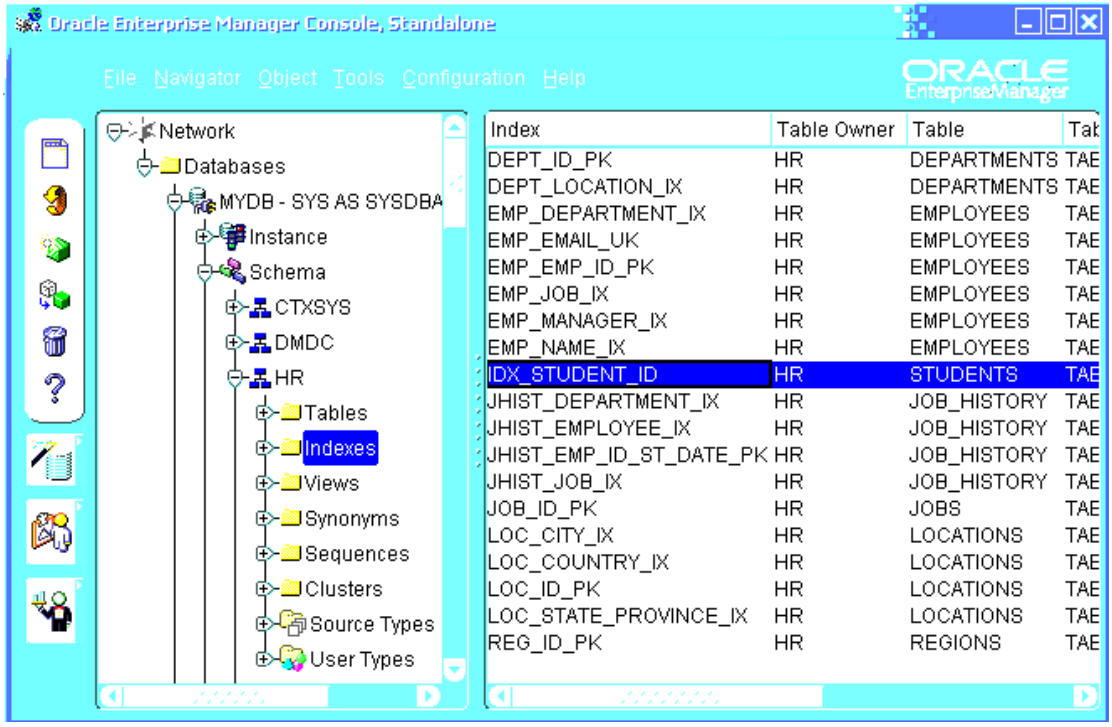
Hình 4.3.2.38 Tạo chỉ mục - Bước 1

Màn hình tiếp theo cho phép thực hiện định nghĩa chỉ mục



Hình 4.3.2.39 Tạo chỉ số - Bước 2

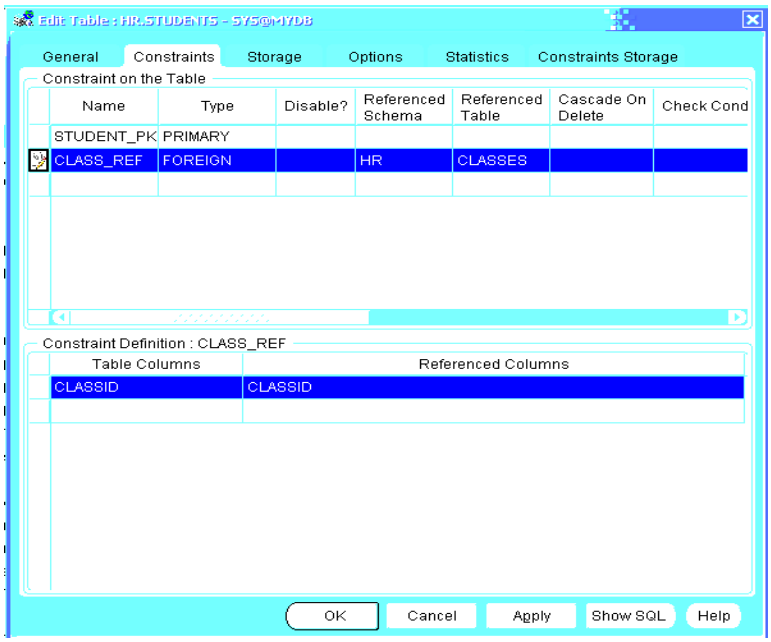
Xác định tên chỉ mục, lược đồ, và tên bảng mà chúng ta muốn tạo chỉ mục sau đó nhấn **Create**. Kiểm tra lại trong danh sách các chỉ mục của lược đồ, chúng ta sẽ thấy chỉ mục mà chúng ta vừa tạo.



Hình 4.3.2.40 Danh sách các chỉ mục

Tạo các liên kết

Để thực hiện tạo liên kết giữa các bảng dữ liệu, chọn bảng ở chiều nhiều, chọn tab **Constraints**.



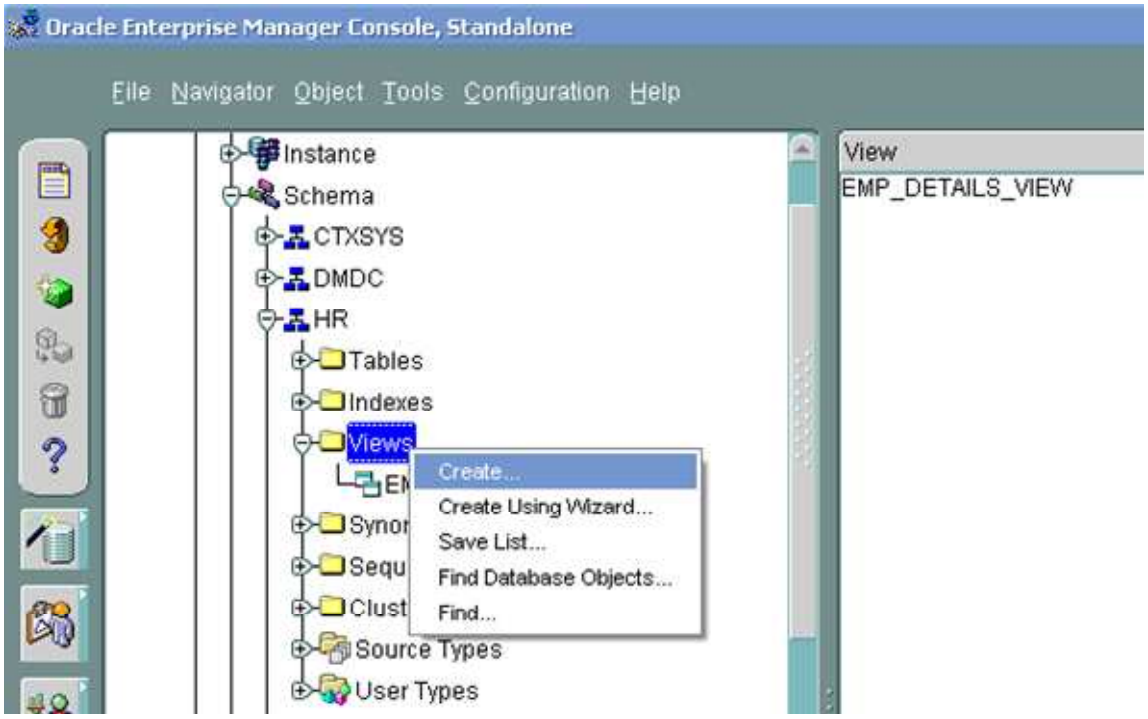
Hình 4.3.2.41 Tạo liên kết

Nhập mới một tên vào mục **Name** trong ô lưới phía trên, đặt kiểu (**Type**) là **FOREIGN** (khóa ngoại lai) vào lược đồ, bảng mà chúng ta sẽ đặt liên kết.

Trong ô lưới phía dưới, chọn bảng tham chiếu và cột trong bảng tham chiếu mà chúng ta sẽ đặt liên kết. Nhấn **OK** để lưu lại liên kết này.

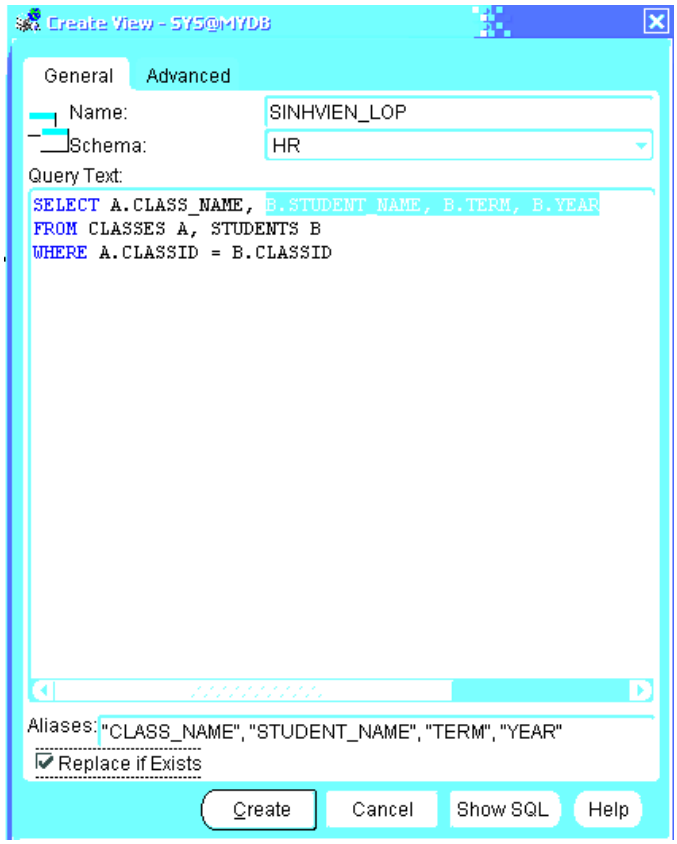
Tạo các khung nhìn

Chọn mục **Views** trong lược đồ hiện tại, nhấn chuột phải và chọn **Create**



Hình 4.3.2.42 **Tạo khung nhìn – Bước 1**

Màn hình tiếp sau đó cho phép thực hiện tạo khung nhìn mới.



Hình 4.3.2.43 Câu lệnh tạo khung nhìn

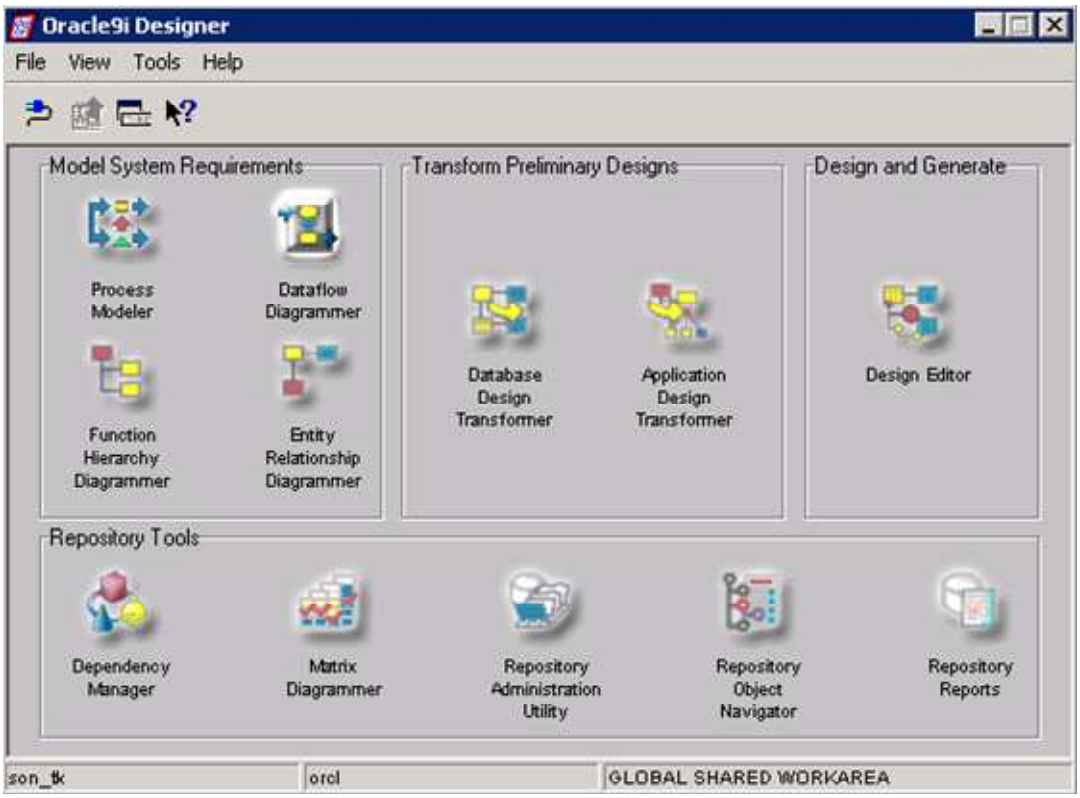
Để có thể sử dụng được các câu lệnh tạo khung nhìn, người sử dụng cần có những hiểu biết nhất định về cú pháp SQL để có thể tạo khung nhìn, bởi lẽ Oracle không cung cấp công cụ trực quan để thực hiện tạo các đối tượng kiểu này. Thay vào đó, việc nhập dòng lệnh SQL là để định nghĩa khung nhìn cho riêng chúng ta. Sau khi nhập đủ thông tin tại các mục, nhấn **Create** để tạo khung nhìn. Chú ý mục **Aliases** cần nhập tên các cột mà chúng ta muốn có trong khung nhìn này.

Sau khi tạo xong khung nhìn, hãy kiểm tra sự tồn tại của khung nhìn trong cơ sở dữ liệu, Đồng thời thực hiện kiểm tra dữ liệu của khung nhìn xem đã đúng ý chưa. Nếu chưa

thỏa mãn, hãy quay lại màn hình thiết kế khung nhìn để bổ sung các điểm cần thiết

Trên đây chúng ta đã xem xét các bước để tạo được các đối tượng dữ liệu trong cơ sở dữ liệu Oracle bằng công cụ quản trị hệ cơ sở dữ liệu Enterprise Manager. Đi kèm với hệ quản trị cơ sở dữ liệu, Oracle cũng đã cung cấp thêm các gói phần mềm hỗ trợ quá trình phân tích thiết kế cơ sở dữ liệu cũng như xây dựng ứng dụng được đánh giá là khá mạnh.

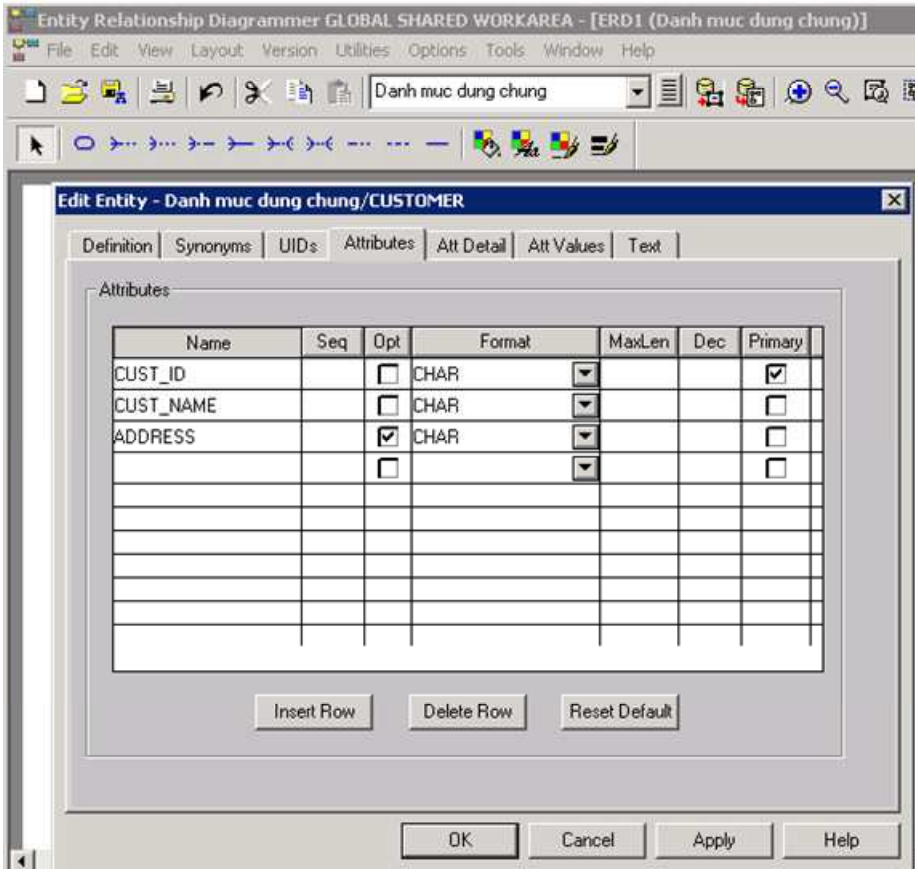
Một công cụ phân tích thiết kế cơ sở dữ liệu được biết nhiều nhất của Oracle đó là Oracle Designer. Ngoài ra, với các cơ sở dữ liệu được xây dựng theo mô hình kho dữ liệu hãng cũng cung cấp thêm một công cụ phân tích thiết kế mới đó là Oracle Warehouse Builder trong các phiên bản từ 9i trở đi. Những bộ công cụ này khá dễ sử dụng và tiện lợi, dưới đây chúng tôi giới thiệu sơ qua về công cụ phân tích, thiết kế cơ sở dữ liệu Oracle9i Designer.



Hình 4.3.2.44 Các công cụ của Oracle9i Designer

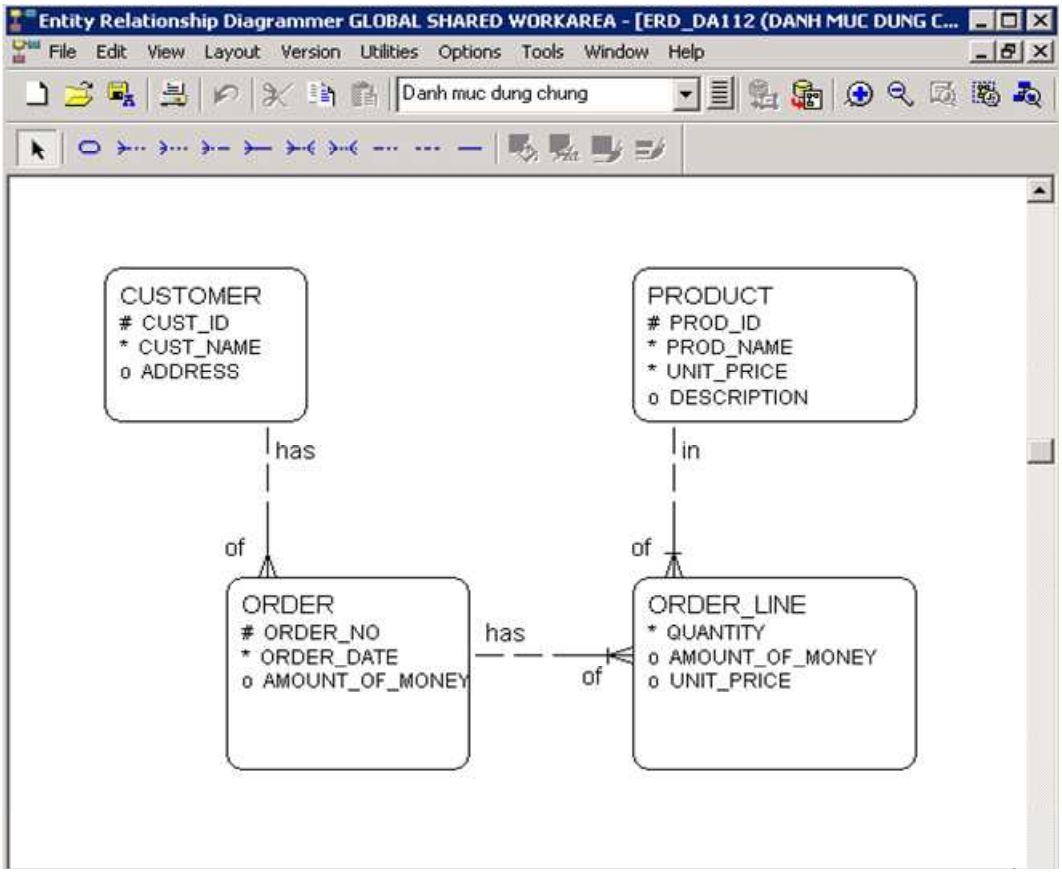
Màn hình trên hiển thị các chức năng trong bộ công cụ Oracle9i Designer. Các công cụ Entity Relationship Diagram, Database Design Transformer và Design Editor thường được sử dụng trong quá trình phân tích thiết kế cơ sở dữ liệu.

Công cụ Entity Relationship Diagram cho phép người sử dụng tạo dựng một sơ đồ thực thể - liên kết theo mô hình dữ liệu quan hệ cho cơ sở dữ liệu. Màn hình tiếp theo dưới đây minh hoạ cho việc tạo dựng một kiểu thực thể và các thuộc tính tương ứng của nó.



Hình 4.3.2.45 Tạo dựng một kiểu thực thể và các thuộc tính của nó

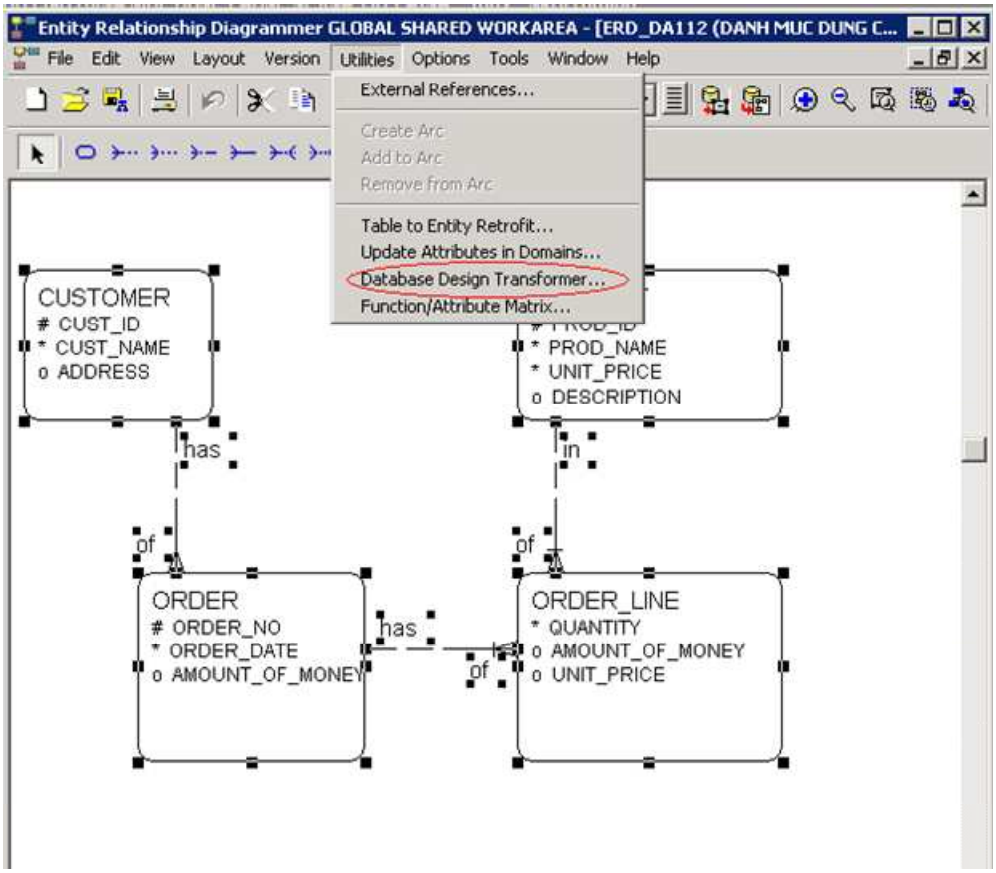
Sau khi tạo xong các kiểu thực thể, chúng ta sẽ có một sơ đồ thực thể - liên kết cho cơ sở dữ liệu, như màn hình sau minh hoạ cho bốn kiểu thực thể đã được tạo trong ví dụ chung trong giáo trình của chúng ta là CUSTOMER, PRODUCT, ORDER, ORDER_LINE:



Hình 4.3.2.46 Một sơ đồ thực thể liên kết trong Oracle

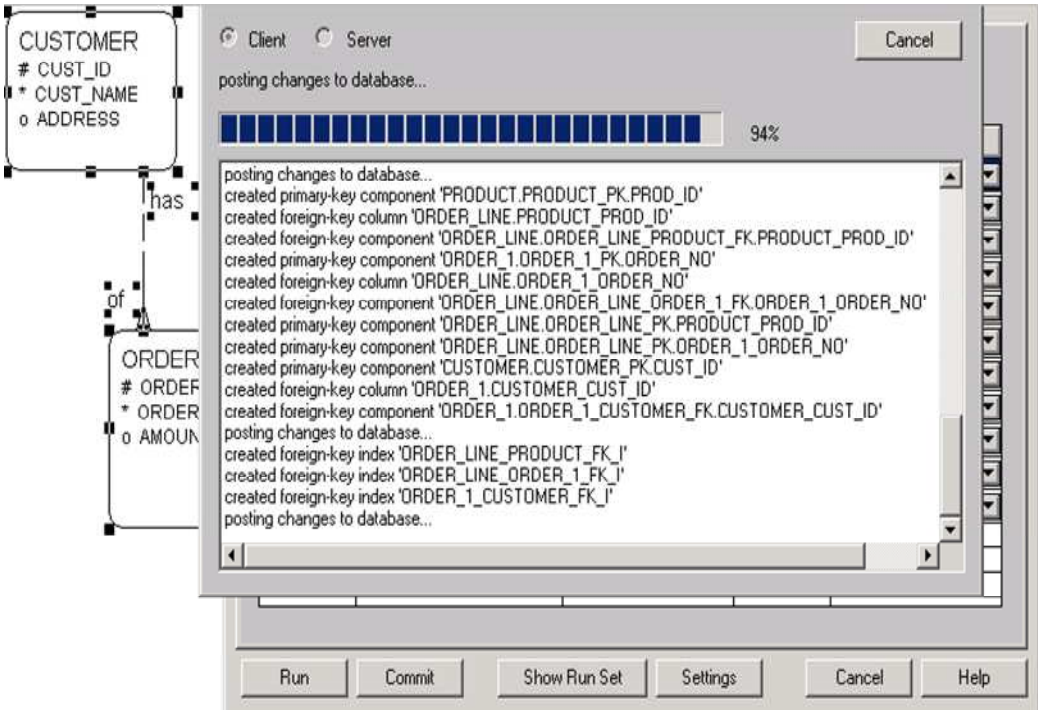
Trong hình trên, các kiểu thực thể được ký hiệu bằng các hình chữ nhật, các thuộc tính mô tả được đặt trong kiểu thực thể. Thuộc tính khoá được ký hiệu bằng một dấu thăng (#) phía trước, thuộc tính có giá trị NOT NULL được ký hiệu bằng một dấu sao (*) phía trước, các thuộc tính thông thường được ký hiệu bằng một dấu tròn (o) phía trước. Ký hiệu các liên kết giữa các kiểu thực thể vẫn được giữ nguyên. Ngoài ra, chúng ta nhận thấy rằng, nếu trong liên kết một nhiều, phía đầu liên kết nhiều có thể có thêm một nét gạch ngang phía trước, ký hiệu này cho biết các thuộc tính khoá chính của kiểu thực thể phía liên kết một sẽ được chuyển sang kiểu thực thể phía liên kết nhiều và tham gia vào làm khoá chính.

Sau khi mô tả xong sơ đồ thực thể - liên kết, chúng ta có thể tiến hành chuyển đổi chúng thành sơ đồ dữ liệu mức logic thông qua chức năng Database Design Transformer như hình dưới đây:



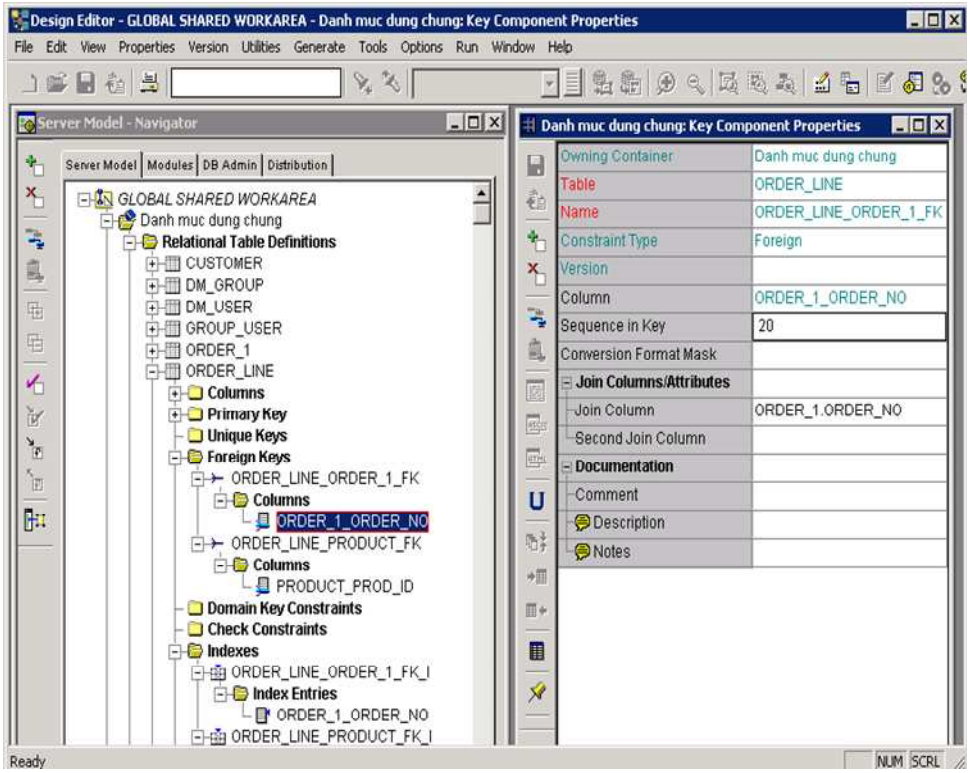
Hình 4.3.2.47 Chuyển đổi sơ đồ thực thể liên kết thành sơ đồ dữ liệu mức logic

Oracle Designer sẽ tự động sản sinh ra các quan hệ (hay bảng) cùng các khoá ngoại lai dựa trên các kiểu thực thể và các liên kết giữa chúng trong sơ đồ thực thể - liên kết trên. Quá trình chuyển đổi được thực hiện như hình sau:



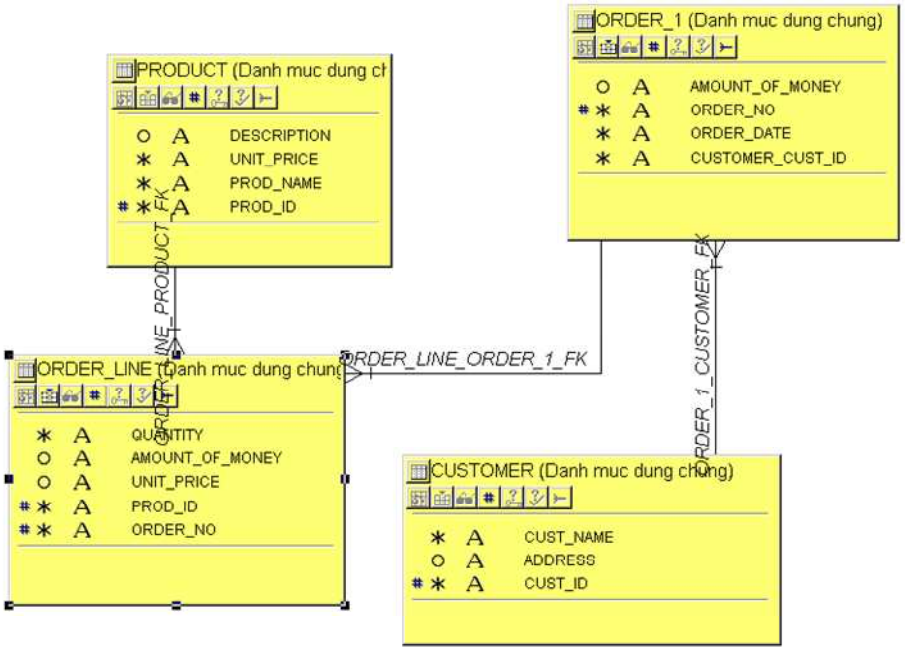
Hình 4.3.2.48 Quá trình chuyển đổi thành các quan hệ, khoá ngoại lai,...

Sau quá trình chuyển đổi trên, chúng ta sẽ có các quan hệ cùng các thuộc tính tương ứng từ các kiểu thực thể và các liên kết, tức là có sơ đồ cơ sở dữ liệu mức logic. Thông qua công cụ Desiger Editor như hình sau, chúng ta có thể tiến hành bổ sung các chỉ mục, các ràng buộc toàn vẹn cho quan hệ. Có một lưu ý là với các khoá ngoại lai được sinh từ liên kết giữa các kiểu thực thể xuất hiện trong quan hệ sẽ có một chỉ mục được tạo ra tương ứng với nó (Oracle Designer đã tự động thực hiện động tác này), như trong hình sau:



Hình 4.3.2.49 Các quan hệ tương ứng được tạo ra

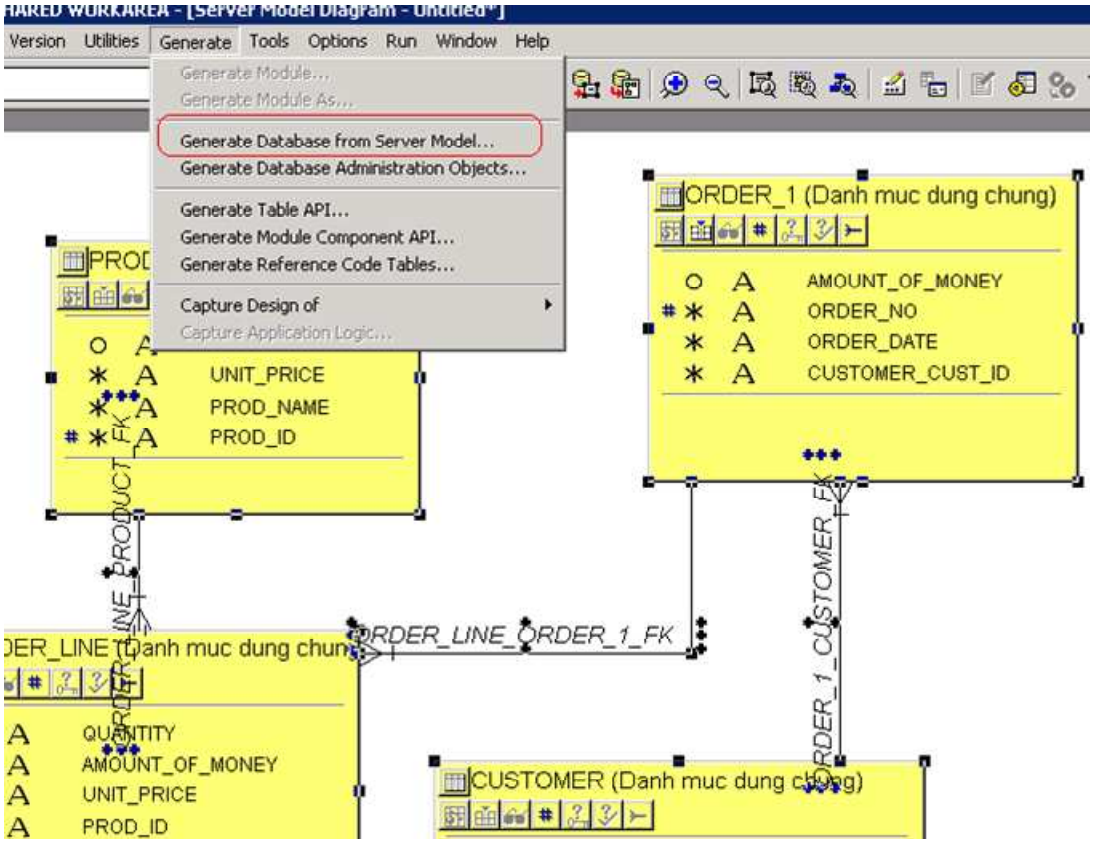
Chúng ta có thể hiển thị các quan hệ này cùng các mối quan hệ giữa chúng thông qua sơ đồ cơ sở dữ liệu mức logic:



Hình 4.3.2.50 Một sơ đồ cơ sở dữ liệu mức logic

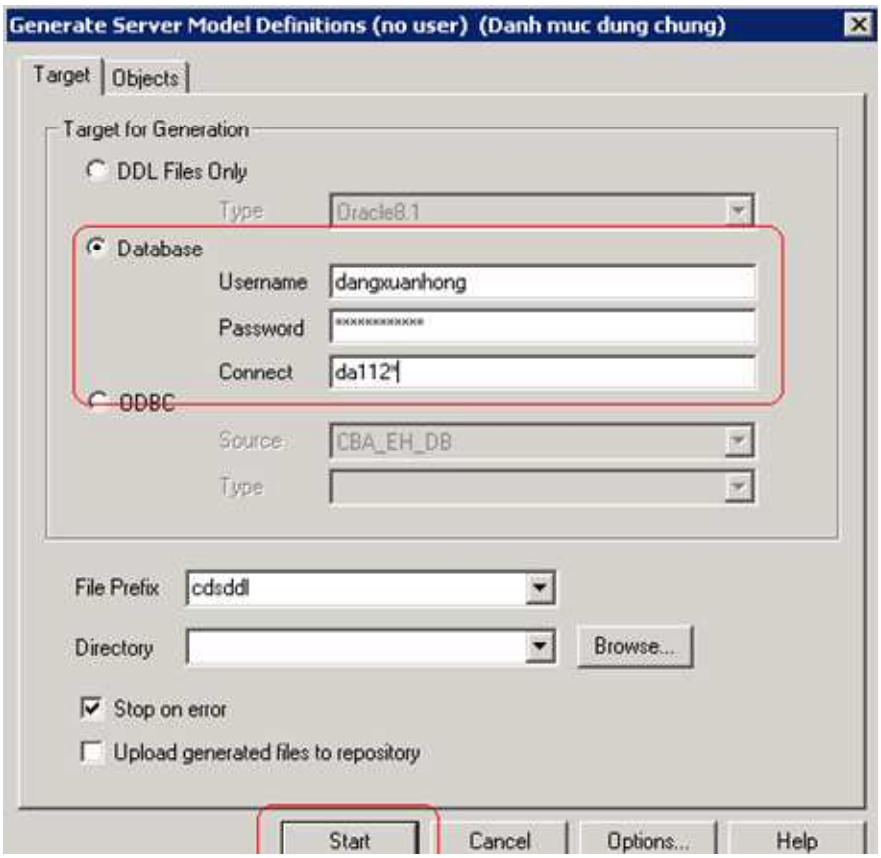
Chúng ta thấy sơ đồ này cũng tương tự như sơ đồ thực thể - liên kết. Tuy nhiên, các thuộc tính trong mỗi quan hệ đã được hình thành đầy đủ theo thiết kế. Và trong sơ đồ này, chúng ta vẫn có thể tiếp tục xây dựng các chỉ mục, phân tách quan hệ, tạo lập các ràng buộc dữ liệu,... Đây chính là một phần của quá trình thiết kế cơ sở dữ liệu mức vật lý.

Sau khi hoàn thành, Oracle sẽ cho phép chúng ta tạo dựng các câu lệnh tạo các đối tượng dữ liệu cho cơ sở dữ liệu thông qua chức năng Generate Database from Server Model như hình sau:



Hình 4.3.2.51 Câu lệnh tạo khung nhìn

Chúng ta lựa chọn cơ sở dữ liệu để cài đặt các đối tượng, cũng cần lưu ý là quá trình thiết kế cơ sở dữ liệu mức vật lý có thể được can thiệp vào bước này với việc chỉnh sửa thêm các câu lệnh sinh các đối tượng cho cơ sở dữ liệu.



Hình 4.3.2.52 Câu lệnh tạo khung nhìn

Sau khi thực hiện xong bước trên, chúng ta đã có một cơ sở dữ liệu hoàn chỉnh.

3.3. Hệ quản trị cơ sở dữ liệu mã nguồn mở

Ngày nay với xu thế sử dụng mã nguồn mở, trên thị trường cũng đã xuất hiện những hệ quản trị cơ sở dữ liệu đi theo xu hướng này, điển hình là hai hệ quản trị cơ sở dữ liệu MySQL và Postgre. Ưu điểm của những hệ quản trị cơ sở dữ liệu này là bạn hầu như không phải quan tâm đến vấn đề bản quyền và quan trọng hơn, bạn có thể thay đổi hay bổ sung thêm các chức năng cho chúng (nếu bạn là người am hiểu hệ thống và lập trình tốt).

Với hệ quản trị cơ sở dữ liệu MySQL, do công ty MySQL AB phát triển, phân phối và hỗ trợ, hiện nay có bốn phiên bản chủ yếu sau:

- MySQL Standard: phiên bản này được xây dựng nhắm tới những người dùng muốn có một cơ sở dữ liệu đủ mạnh, có hiệu suất tính toán cao.
- MySQL Max: phiên bản này nhắm tới những người dùng muốn sử dụng những tính năng mới không có ở các phiên bản khác như bộ xử lý lưu trữ Berkeley database, cho phép các bảng dữ liệu có thể được chia nhỏ thành các tệp, tránh được hạn chế về giới hạn kích thước ở một số hệ điều hành.
- MySQL Pro là một phiên bản thương mại của MySQL Standard.
- MySQL Classic: khác với các phiên bản Standard và Pro, đây là một phiên bản chỉ bao gồm một số chức năng chính của MySQL.

Hệ quản trị cơ sở dữ liệu MySQL cũng có hệ thống bảo đảm an toàn bảo mật cao, bao gồm cả việc hỗ trợ mã hoá tầng vận chuyển theo SSL (Security Socket Layer). Từ các phiên bản 4.0, hệ quản trị cơ sở dữ liệu này còn cho phép giới hạn tài nguyên đến từng người dùng. Ngoài ra với cơ chế query caching đã làm tăng đáng kể hiệu năng thực hiện các câu lệnh truy vấn thông thường.

Hệ quản trị cơ sở dữ liệu này cũng cho phép thực hiện các chức năng tạo bản sao dữ liệu (data replication) cho các máy server phụ (slave), giúp cho hệ thống phân tải tốt hơn cũng như hỗ trợ cơ chế phân tán dữ liệu.

Cũng như MySQL, Postgre cũng là một hệ quản trị cơ sở dữ liệu mã nguồn mở bắt nguồn từ trường đại học Berkely, hiện nay đã có đến phiên bản 7.3 với ngôn ngữ hỏi đáp dữ liệu cũng tuân theo chuẩn SQL99. Một ưu điểm đối với hệ quản trị cơ sở dữ liệu này đó là nó cho phép khôi phục dữ liệu ở mức thời điểm (point-in-time), tức là khi có sự cố, người quản trị không phải khôi phục lại toàn bộ cơ sở dữ liệu.

Các hệ quản trị cơ sở dữ liệu mã nguồn mở trên đều có thể tải về từ mạng internet với cả những phiên bản đã được thương mại hoá và có hỗ trợ kỹ thuật. Chúng ta có thể tham khảo chúng thêm ở những địa chỉ internet sau:

- www.MySQL.com
- www.PostgreSQL.org

3.4. Nhận xét chung

Các hệ quản trị cơ sở dữ liệu như Access hay Visual Fox rất thích hợp cho những cơ sở dữ liệu cỡ vừa và nhỏ (khoảng từ một vài Gigabyte trở lại). Ưu điểm chung của chúng là dễ cài đặt, giao diện thân thiện, khi vận hành không chiếm nhiều không gian nhớ. Nhưng nhược điểm lớn nhất của chúng đó là tính bảo mật không cao, tốc độ giảm đi đáng kể khi phải đáp ứng nhiều kết nối đến cơ sở dữ liệu. Trong các hệ quản trị cơ sở dữ liệu này không phân chia ra các người sử dụng khác nhau nên thường các ứng dụng dựa trên cơ sở dữ liệu phải tự xây dựng các môđun để thực hiện công việc này. Bởi vậy, chúng ta chỉ nên chọn những hệ quản trị cơ sở dữ liệu này khi cơ sở dữ liệu cần xây dựng chỉ phải đáp ứng một lượng người dùng nhỏ

(khoảng vài chục người), yêu cầu bảo mật với dữ liệu không đòi hỏi quá khắt khe.

Đối với các hệ quản trị cơ sở dữ liệu như SQL Server: ưu điểm chính vẫn là tính dễ sử dụng, giao diện thân thiện gần gũi với người sử dụng đã quen với hệ điều hành Windows. Với các phiên bản Enterprise, đã có thể đáp ứng được những cơ sở dữ liệu tầm cỡ doanh nghiệp có thể bao gồm vài trăm thậm chí vài nghìn người kết nối làm việc cùng một lúc. Tính bảo mật trong hệ quản trị cơ sở dữ liệu này cũng đã được chú trọng đáng kể, trong đó các tài khoản người dùng có thể được tích hợp với hệ điều hành WindowsNT hoặc Windows2000. Trong cùng một hệ quản trị cơ sở dữ liệu có thể tạo ra một số cơ sở dữ liệu khác nhau. Mỗi một người sử dụng có thể sở hữu cũng như có thể có quyền truy nhập đến một số cơ sở dữ liệu. Các cơ sở dữ liệu có thể có kích cỡ lớn, trong đó mỗi bảng có thể chứa đến vài Gigabyte dữ liệu hoặc hơn. Một số phiên bản còn hỗ trợ các ứng dụng theo mô hình kho dữ liệu. Ngoài ra, các bước cài đặt cho hệ quản trị này cũng không quá phức tạp. Nhược điểm chính của hệ quản trị cơ sở dữ liệu này chính là nó chỉ có thể chạy trên các hệ điều hành máy chủ Windows mà thôi. Tuy nhiên, với những ưu điểm đã trình bày, SQL Server rất phù hợp với các cơ quan, tổ chức, doanh nghiệp ở Việt Nam hiện nay, do đó nó thường được xem là lựa chọn đầu tiên cho những cơ sở dữ liệu tầm cỡ.

Đối với hệ quản trị cơ sở dữ liệu Oracle, đây được xem là một hệ quản trị cơ sở dữ liệu chiếm thị phần lớn nhất. Các ưu điểm chính là có thể đáp ứng được những cơ sở dữ liệu tầm cỡ khá lớn. Cũng như hệ quản trị cơ sở dữ liệu SQL Server, những phiên bản Enterprise có thể có kích thước

đến hàng Gigabyte thậm chí hàng Tetrabyte dữ liệu. Các bảng dữ liệu lớn có thể được chia ra thành các phân đoạn (partition) khác nhau và cũng có thể được lưu trữ trên những vị trí vật lý (ổ đĩa) khác nhau. Các cơ chế phân quyền và bảo mật cao, tuy nhiên so với hệ quản trị cơ sở dữ liệu SQL Server, việc quản lý giữa người sử dụng và cơ sở dữ liệu trong hệ quản trị cơ sở dữ liệu Oracle được tổ chức theo mô hình khác. Đó là một hệ quản trị cơ sở dữ liệu cũng có thể tạo dựng nhiều cơ sở dữ liệu khác nhau, tuy nhiên mỗi một người sử dụng phải thuộc vào một cơ sở dữ liệu nhất định. Việc gán quyền sử dụng trên các đối tượng dữ liệu được thực hiện giữa các người sử dụng trong cùng một cơ sở dữ liệu. Một ưu điểm nữa của hệ quản trị cơ sở dữ liệu này là có thể được cài đặt trên nhiều hệ điều hành khác nhau như Unix, Windows, Linux,... Chúng ta chỉ nên chọn hệ quản trị cơ sở dữ liệu này cho những hệ thống thực sự lớn, cơ sở dữ liệu chứa hàng Gigabyte trở lên, lượng người truy nhập nhiều, chỉ có như vậy mới có thể tận dụng được các ưu điểm do hệ quản trị cơ sở dữ liệu này mang lại. Tuy nhiên, cũng cần lưu ý rằng, việc quản trị một hệ thống cơ sở dữ liệu Oracle là khá phức tạp, tốn nhiều thời gian để tiếp cận và phải đòi hỏi những người quản trị thật chuyên sâu.

Hai hệ quản trị cơ sở dữ liệu mã nguồn mở MySQL và Postgre cũng còn tương đối mới mẻ đối với những nhà thiết kế cơ sở dữ liệu Việt Nam, trên thị trường hiện nay mới chỉ xuất hiện một số ứng dụng web viết bằng ngôn ngữ PHP kết nối với hệ quản trị cơ sở dữ liệu MySQL. Tuy nhiên có thể trong những năm tới các hệ quản trị cơ sở dữ liệu mã nguồn mở sẽ được để ý đến nhiều hơn và có thể chúng sẽ là một xu hướng mới trong ngành công nghiệp cơ sở dữ liệu.

4. Minh hoạ đối với một hệ quản trị cơ sở dữ liệu cụ thể

Trong các phần trình bày trên, chúng tôi đã trình bày khá kỹ các bước tạo dựng các đối tượng dữ liệu trong mỗi một hệ quản trị cơ sở dữ liệu. Do vậy trong phần này, các bạn có thể lựa chọn một trong bốn hệ quản trị cơ sở dữ liệu trên để có thể thực hành.

5. Kết luận

Trong chủ đề này chúng ta đã được làm quen với kiến trúc của một hệ quản trị cơ sở dữ liệu, bao gồm các thành phần như khối nhận câu lệnh truy vấn SQL từ người sử dụng, khối phân tích xử lý câu lệnh, khối quản lý các giao dịch, điều khiển tương tranh, khối quản lý không gian lưu trữ. Giữa các khối có tính độc lập tương đối nhưng cũng phối hợp chặt chẽ với nhau.

Ngoài ra, cũng trong chủ đề này, chúng ta đã được làm quen với bốn hệ quản trị cơ sở dữ liệu phổ biến nhất hiện nay trên thị trường, đó là hệ quản trị cơ sở dữ liệu Access, hệ quản trị cơ sở dữ liệu Visual Foxpro, hệ quản trị cơ sở dữ liệu SQL Server và hệ quản trị cơ sở dữ liệu Oracle. Trong mỗi một hệ quản trị cơ sở dữ liệu chúng ta cũng đã xem xét các đối tượng dữ liệu được tạo dựng như thế nào cũng như các bước để xây dựng lên một cơ sở dữ liệu hoàn chỉnh. Các ưu nhược điểm của mỗi một hệ quản trị cơ sở dữ liệu cũng đã được xem xét đến, giúp chúng ta có được những lựa chọn chính xác trước các hệ thống cụ thể.

6. Bài tập

1. Tiến hành cài đặt các hệ quản trị cơ sở dữ liệu sau:

- Access
- Visual Foxpro
- Microsoft SQL Server 2000
- Oracle Workgroup hoặc Oracle Enterprise Edition

2. Tiến hành thực hiện các bước đã nêu trong mục 3 của chủ đề đối với mỗi hệ quản trị cơ sở dữ liệu đã được cài đặt ở bài tập 1.

3. Với mỗi hệ quản trị cơ sở dữ liệu trong bài tập 1, dựa trên sơ đồ thiết kế cơ sở dữ liệu mức logic 2.4.11.7 trong chủ đề 2, tiến hành xây dựng các bảng (quan hệ), cùng các mối liên kết giữa chúng.

4. Với sơ đồ thực thể - liên kết đã xây dựng được trong bài tập 7 chủ đề 2, tiến hành xây dựng chúng với mỗi hệ quản trị cơ sở dữ liệu đã nêu trong bài tập 1 trên.

5. Tiếp tục tìm hiểu công cụ Oracle Designer2000, sử dụng chức năng Database Design Transformer để chuyển đổi sơ đồ thực thể - liên kết có được trong bài tập 7 chủ đề 2 thành sơ đồ dữ liệu và sử dụng chức năng Generate Database from Server Model để sinh mã cho cơ sở dữ liệu.

6. Lựa chọn hệ quản trị cơ sở dữ liệu SQL Server hoặc Oracle để tiến hành thiết kế, cài đặt cho cơ sở dữ liệu được trình bày trong phụ lục B của giáo trình.

Phụ lục A: Các hệ thống CSDL sẽ được xây dựng trong khuôn khổ dự án 112

Mục tiêu:

Trong phụ lục này chúng tôi muốn giới thiệu với các bạn một số nét chính về hệ thống các cơ sở dữ liệu mang tính chất định hướng sẽ được xây dựng trong khuôn khổ dự án 112.

1. Mô hình tổ chức hệ thống CSDL các cấp

Hệ thống thông tin (HTTT) phục vụ điều hành và quản lý hành chính nhà nước (đề án 112) bao gồm các hệ thống sau:

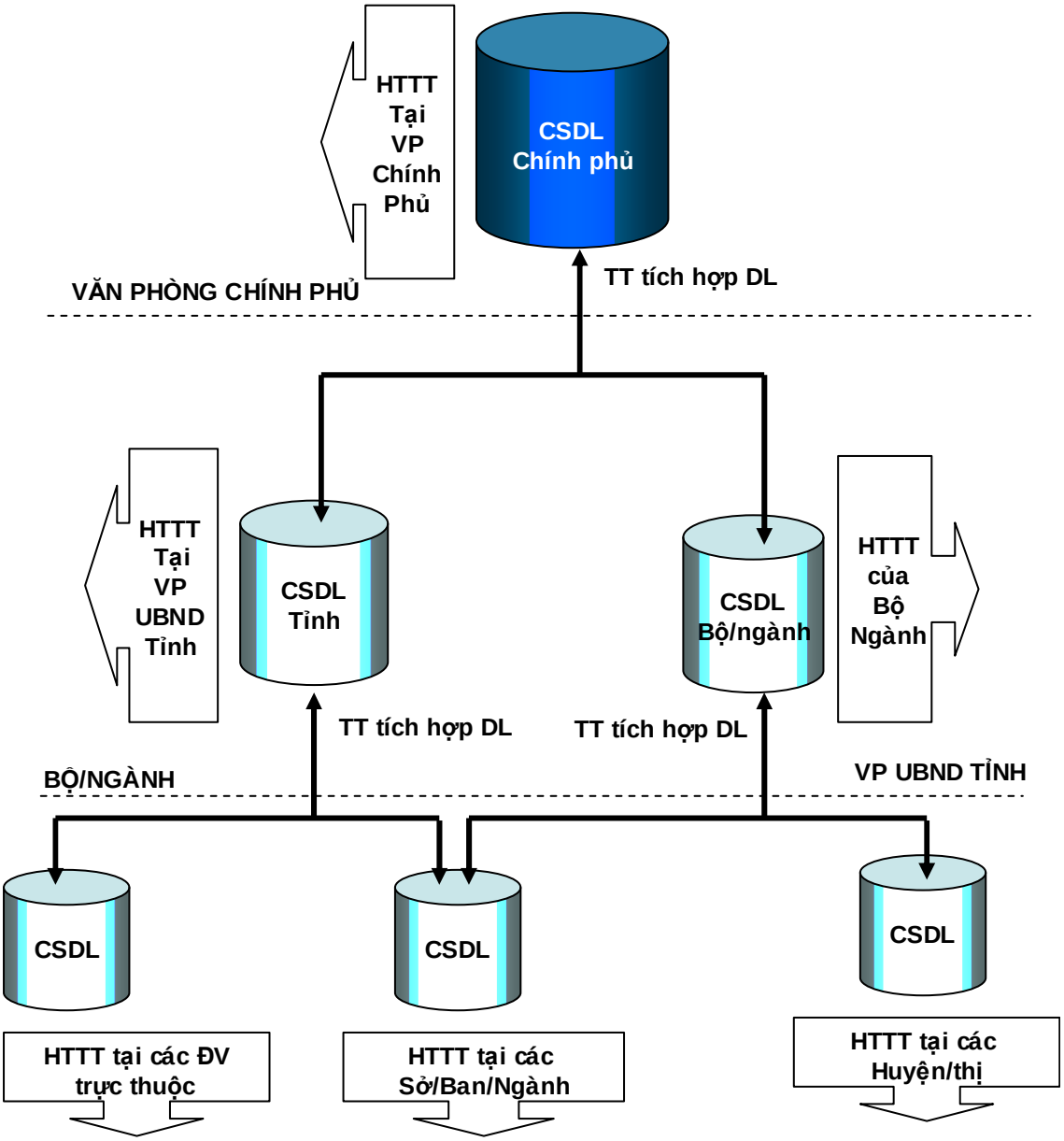
- Hệ thống thông tin Chính phủ
- Hệ thống thông tin tại các Bộ/Ngành
- Hệ thống thông tin tại các Tỉnh thành phố trực thuộc Trung ương
- Hệ thống các cơ sở dữ liệu Quốc gia.

Trong mỗi hệ thống trên bao gồm nhiều hệ thống con và hạt nhân của mỗi hệ thống là các cơ sở dữ liệu. Giữa chúng có các quan hệ liên kết, trao đổi, chia sẻ thông tin theo chiều ngang và chiều dọc. Quan hệ ngang là quan hệ trao đổi, chia sẻ thông tin có tính chất chuyên môn giữa các đơn vị ngang cấp, còn quan hệ theo chiều dọc là quan hệ dựa

trên cấu trúc phân cấp thẩm quyền chức năng quản lý nhà nước trong hệ thống các cơ quan hành chính nhà nước.

Tại mỗi hệ thống, bên cạnh việc xây dựng các HTTT sẽ hình thành các trung tâm dữ liệu, đóng vai trò là nơi liên kết, tích hợp các cơ sở dữ liệu của các đơn vị trong hệ thống và có chức năng cung cấp, chia sẻ thông tin, truyền các báo cáo lên cấp trên, truyền các mệnh lệnh quản lý xuống các cấp dưới.

Chúng ta có mô hình tổ chức các cơ sở dữ liệu đi kèm với HTTT ở các cấp như sau:



Hình 5.1.1 Mô hình tổ chức CSDL và HTTT ở các cấp

2. Các cơ sở dữ liệu phục vụ điều hành, quản lý

a. Cơ sở dữ liệu phục vụ hệ thống thông tin quản lý đơn thư khiếu nại tố cáo

Tại mỗi cơ quan phải giải quyết đơn thư khiếu nại tố cáo sẽ xây dựng một cơ sở dữ liệu quản lý đơn thư khiếu nại tố

cáo. Cơ sở dữ liệu này sẽ kết nối với HTTT cung cấp dịch vụ công để tiếp nhận và trả lời kết quả giải quyết đơn thư khiếu nại tố cáo cho dân.

Các cơ sở dữ liệu khiếu nại tố cáo của các cơ quan phải thoả mãn một số chuẩn chung và được kết nối với nhau nhằm trao đổi thông tin, chuyển đơn thư khiếu nại tố cáo cho nhau trong các trường hợp phải giải quyết liên ngành, liên cấp

Quá trình xây dựng và triển khai hệ cơ sở dữ liệu khiếu nại tố cáo bao gồm:

- Xây dựng, cài đặt và triển khai cơ sở dữ liệu khiếu nại tố cáo tại các cơ quan liên quan trên địa bàn.
- Kết nối với HTTT cung cấp dịch vụ công
- Liên kết các cơ sở dữ liệu khiếu nại tố cáo của các cơ quan nhà nước với nhau.

b. Cơ sở dữ liệu phục vụ HTTT luật và văn bản qui phạm pháp luật

Xây dựng một cơ sở dữ liệu trung tâm về văn bản quy phạm pháp luật tại trung tâm tích hợp dữ liệu, liên kết cơ sở dữ liệu trung tâm với hệ điều hành tác nghiệp tại văn phòng UBND và cơ sở dữ liệu quốc gia về luật để thường xuyên bổ sung nguồn dữ liệu cho cơ sở dữ liệu trung tâm.

Tại mỗi cơ quan xây dựng một cơ sở dữ liệu văn bản quy phạm pháp luật và kết nối với cơ sở dữ liệu trung tâm để cập nhật dữ liệu thuộc các lĩnh vực do cơ quan phụ trách và khai thác nguồn dữ liệu văn bản quy phạm pháp luật từ cơ sở dữ liệu trung tâm.

c. Các cơ sở dữ liệu ứng dụng chuyên ngành

Tuỳ thuộc vào các lĩnh vực chuyên môn nghiệp vụ của cơ quan, tại mỗi cơ quan nhà nước cần tiến hành xây dựng các cơ sở dữ liệu và chương trình ứng dụng chuyên ngành. Mỗi cơ sở dữ liệu chuyên ngành gắn với một lĩnh vực chuyên môn nghiệp vụ và có nhiệm vụ cập nhật, xử lý và lưu trữ các dữ liệu liên quan. Cụ thể, các cơ sở dữ liệu chuyên ngành có nhiệm vụ:

- Phục vụ trực tiếp các hoạt động chuyên môn theo lĩnh vực nghiệp vụ.
- Cung cấp, tiếp nhận và trao đổi thông tin chuyên môn với các phòng ban cấp dưới
- Cung cấp, thu nhận và trao đổi các dữ liệu tổng hợp phục vụ điều hành với các cơ sở dữ liệu tổng hợp kinh tế - xã hội đặt tại cơ quan.
- Một số cơ sở dữ liệu chuyên ngành sẽ kết nối với các HTTT phục vụ dịch vụ công để cung cấp, thu nhận, trao đổi thông tin.
- Cung cấp, thu nhận và trao đổi thông tin với các cơ sở dữ liệu Quốc gia cùng lĩnh vực.

Như vậy, các cơ sở dữ liệu chuyên ngành được liên kết với nhau hình thành nên một hệ thống các cơ sở dữ liệu chuyên ngành, đóng vai trò là kho dữ liệu nền phục vụ đa mục tiêu, có mối liên kết với các HTTT khác trên địa bàn.

Các cơ sở dữ liệu chuyên ngành bao gồm:

- Cơ sở dữ liệu chuyên ngành tại VP UBND Tỉnh

- Cơ sở dữ liệu chuyên ngành tại các Sở, được kết nối với các cơ sở dữ liệu liên quan như: cơ sở dữ liệu tổng hợp kinh tế - xã hội đặt tại Sở; cơ sở dữ liệu quốc gia liên quan; HTTT phục vụ dịch vụ công; cơ sở dữ liệu tại các phòng/ ban của UBND các huyện/thị
- Cơ sở dữ liệu chuyên ngành tại UBND huyện/thị: bao gồm các cơ sở dữ liệu chuyên ngành cấp cơ sở tại các phòng/ban, được kết nối với cơ sở dữ liệu chuyên ngành tại các Sở, với cơ sở dữ liệu tổng hợp kinh tế - xã hội tại UBND huyện/thị và với các HTTT phục vụ dịch vụ công trên địa bàn cấp huyện

Việc xây dựng và triển khai các cơ sở dữ liệu chuyên ngành tại các cơ quan nhà nước gồm các công việc sau:

- Phân tích, thiết kế và xây dựng các cơ sở dữ liệu chuyên ngành
- Kết nối các cơ sở dữ liệu chuyên ngành với cơ sở dữ liệu tổng hợp kinh tế - xã hội phục vụ điều hành tại cơ quan
- Kết nối các cơ sở dữ liệu chuyên ngành với các cơ sở dữ liệu quốc gia liên quan
- Tổ chức việc thu thập, phân loại, xử lý dữ liệu từ nhiều nguồn để cập nhật vào các cơ sở dữ liệu chuyên ngành (tạo nguồn dữ liệu cho cơ sở dữ liệu hoạt động).

3. Các cơ sở dữ liệu phục vụ dịch vụ công

Hệ thống các HTTT phục vụ dịch vụ công nhằm mục tiêu:

- Nâng cao tính hiệu quả, rút ngắn thời gian, tự động hoá, đơn giản hoá và làm gọn nhẹ các qui trình giải quyết, xử lý các dịch vụ công của các cơ quan nhà nước
- Thực hiện việc phục vụ cung cấp các dịch vụ công, nhất là các dịch vụ công mang tính liên ngành theo chế độ một cửa.
- Cung cấp các thông tin công cộng về luật pháp, về các lĩnh vực dịch vụ công cho mọi người dân, nâng cao hiểu biết của nhân dân.

Để xây dựng và triển khai các HTTT phục vụ dịch vụ công cần phải:

- Xây dựng các cơ sở dữ liệu và giao diện xử lý
- Xây dựng các Web site kết nối đến các cơ sở dữ liệu.
- Tạo nguồn dữ liệu cho các cơ sở dữ liệu.

Các cơ sở dữ liệu cần xây dựng có thể là:

- Cơ sở dữ liệu phụ vụ cấp giấy chứng nhận quyền sử dụng đất
- Cơ sở dữ liệu phục vụ cấp phép xây dựng
- Cơ sở dữ liệu phục vụ cấp phép đầu tư
- Cơ sở dữ liệu phụ vụ cấp đăng ký và quản lý kinh doanh
- Cơ sở dữ liệu phục vụ đăng ký hộ khẩu và dân cư
- Cơ sở dữ liệu phụ vụ tiếp nhận và giải quyết đơn thư khiếu nại, tố cáo

4. Các cơ sở dữ liệu quốc gia

Trong quá trình xây dựng các hệ thống thông tin chuyên ngành, có một số cơ sở dữ liệu quan trọng mang tính liên ngành, nhiều đơn vị sử dụng chung, đóng vai trò then chốt cho việc cung cấp thông tin kịp thời, đủ tin cậy phục vụ cho công tác quản lý nhà nước và các đối tượng khác nhau, đó là các cơ sở dữ liệu Quốc gia. Phù hợp với các cơ sở dữ liệu Quốc gia này, tại địa bàn Tỉnh cần xây dựng các cơ sở dữ liệu là thành phần của các cơ sở dữ liệu quốc gia trên địa bàn tỉnh. Việc xây dựng các cơ sở dữ liệu quốc gia trên địa bàn tỉnh phải tuân theo nội dung của các cơ sở dữ liệu quốc gia đang được triển khai xây dựng trong phạm vi toàn quốc.

Đó là các cơ sở dữ liệu sau đây:

- Cơ sở dữ liệu quốc gia về thống kê kinh tế xã hội
- Cơ sở dữ liệu quốc gia về tài chính, ngân sách
- Cơ sở dữ liệu quốc gia về văn bản quy phạm pháp luật
- Cơ sở dữ liệu quốc gia về cán bộ, công chức
- Cơ sở dữ liệu quốc gia về dân cư
- Cơ sở dữ liệu quốc gia về tài nguyên đất
- …..

Phụ lục B: Phân tích, thiết kế cơ sở dữ liệu cho một cửa hàng sách

Mục tiêu:

Trong phụ lục này, chúng ta sẽ xem xét một ví dụ minh hoạ cho các bước phân tích, thiết kế và xây dựng một cơ sở dữ liệu như đã trình bày trong giáo trình. Nội dung bài toán là Xây dựng một cơ sở dữ liệu cho một cửa hàng sách. Các bạn có thể lựa chọn một trong số bốn hệ quản trị cơ sở dữ liệu đã trình bày trong chủ đề IV để thiết kế cho cơ sở dữ liệu này..

1. Phân tích yêu cầu

Người đại diện cho cửa hàng sách đã suy nghĩ về vấn đề này và trình bày tương đối rõ ràng trong tóm tắt sau:

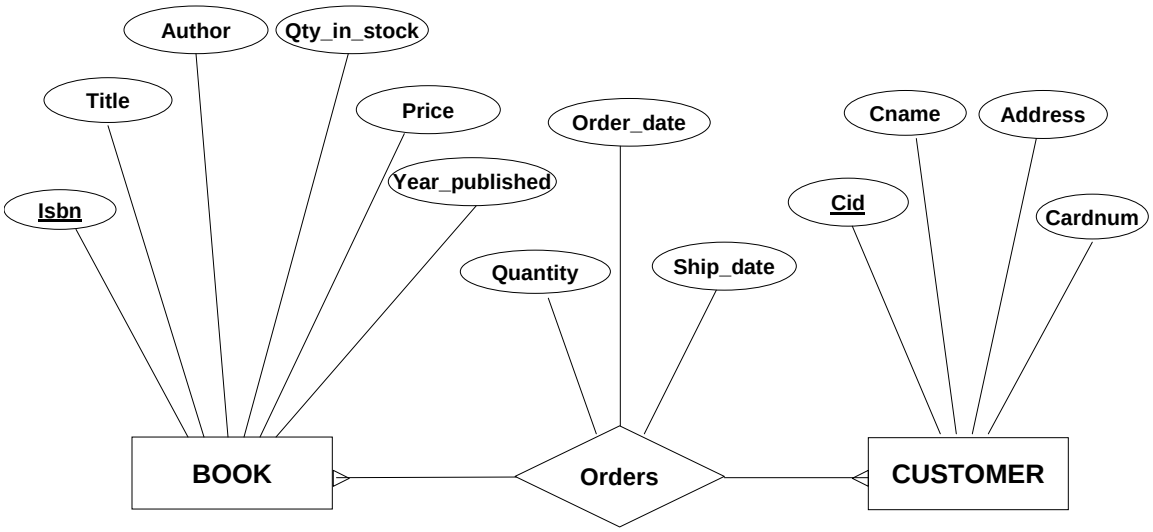
"Tôi muốn khách hàng có thể tìm kiếm các danh mục sách trong cửa hàng và có thể đặt hàng ngay trên mạng internet. Hiện nay, các công việc đặt mua sách này vẫn đang được thực hiện qua điện thoại. Các khách hàng gọi điện đến cửa hàng và nói với tôi số hiệu sách ISBN cùng số lượng đối với mỗi loại mà họ muốn mua. Nếu tôi không có đủ số lượng sách đối với một đầu sách nào đó, tôi sẽ phải đặt hàng thêm và trì hoãn lại ngày giao sách cho tới khi số lượng sách mới tới cửa hàng. Tôi muốn giao toàn bộ sách trong yêu cầu đặt hàng cho khách hàng cùng một lúc với bảng kê bao gồm tất cả các sách mà tôi bán. Với mỗi đầu sách sẽ được lưu trong bảng kê như sau: số hiệu ISBN của

sách, tiêu đề, tác giả, giá mua, giá bán và năm xuất bản. Hầu hết các khách hàng đều là khách quen và tôi đã ghi lại danh sách của họ bao gồm: tên, địa chỉ, số thẻ tín dụng. Đối với một khách mới của cửa hàng thì họ phải liên hệ trước để tôi tạo cho họ một tài khoản trước khi họ có thể sử dụng trang web của tôi để có thể tìm kiếm và đặt hàng mua sách trực tuyến trên trang web".

Với trình bày trên đây, chúng ta sẽ phải tiến hành phân tích chi tiết các thông tin. Bước tiếp theo chúng ta phải làm đó là:

2. Thiết kế cơ sở dữ liệu mức khái niệm

Trong bước này chúng ta sẽ phải mô tả dữ liệu dưới dạng mô hình thực thể liên kết. Bản thiết kế đầu tiên như trong hình sau. Sách BOOK và khách hàng CUSTOMER là những kiểu thực thể chúng ta có thể xác định được đầu tiên, hai kiểu thực thể này kết nối với nhau thông qua liên kết đơn đặt hàng "Orders". Với mỗi đơn đặt hàng, những thông tin sau cần phải được lưu lại: số lượng Quantity, ngày đặt hàng Order_date và ngày giao hàng Ship_date. Khi ngày giao hàng Ship_date mang giá trị rỗng, tương ứng với thông tin cho biết là đơn đặt hàng chưa được giao. Ngay sau khi đơn đặt hàng được giao, giá trị trong ngày giao hàng Ship_date sẽ được đặt lại giá trị và nó cho biết đơn giao hàng đó đã được giao.



Hình 6.2.1 Sơ đồ thực thể - liên kết giữa BOOK và CUSTOMER

Đến đây chúng ta đã có một bản thiết kế đầu tiên và chúng ta cần phải thảo luận thêm trong nhóm thiết kế cơ sở dữ liệu về một số trường hợp có thể xảy ra trên thực tế và cách giải quyết ở bước này như thế nào?

CH (câu hỏi): Nếu một khách hàng đặt 2 đơn đặt hàng cho cùng một quyển sách trong cùng một ngày thì sẽ phải xử lý như thế nào?

TL (trả lời): Đối với lần đặt hàng thứ nhất, nó sẽ được tạo ra một đơn đặt hàng trong liên kết đơn hàng “Orders”, đối với lần đặt hàng thứ 2 nó sẽ được xử lý bằng cách cập nhật lại thuộc tính số lượng Quantity trong liên kết “Orders” đã tạo.

CH: Nếu một khách hàng đặt 2 đơn đặt hàng cho 2 quyển sách khách nhau trong cùng một ngày thì sẽ được xử lý như thế nào:

TL: Khi đó ta coi mỗi lần đặt sách là một đơn đặt hàng khác nhau như trong các trường hợp thông thường. Tức là mỗi một thể hiện của liên kết "Orders" liên quan đến khách hàng cho mỗi quyển sách đó.

CH: Nếu khách hàng đặt hàng cùng một quyển sách nhưng ở 2 ngày khác nhau thì sẽ được xử lý như thế nào?

TL: Cũng tương tự như trong trường hợp trên, ta sẽ phải lưu 2 thể hiện cho 2 đơn đặt hàng đó và chúng sẽ có giá trị tại thuộc tính ngày đặt hàng Order_date khác nhau.

CH: Nếu xử lý như vậy thì không thể được, bởi các giá trị kết nối giữa hai thuộc tính mã hiệu khách hàng Cid và số hiệu sách Isbn khi được liên kết với nhau không thể xác định ra duy nhất một đơn hàng trong "Orders". Như vậy bản thiết kế ban đầu này chưa cho phép khách hàng đặt các đơn đặt hàng khác nhau cho cùng một cuốn sách trong những ngày khác nhau.

TL: Đúng như vậy, có lẽ khách hàng chưa quan tâm đến trường hợp này, chúng ta cần phải tìm hiểu thêm.

Chúng ta hãy chuyển sang bước thiết kế tiếp theo và xem xét vấn đề trên sẽ được xử lý tiếp như thế nào

3. Thiết kế cơ sở dữ liệu mức logic

Ở bước này chúng ta ánh xạ sơ đồ thực thể liên kết trong hình 6.2.1 sang mô hình quan hệ, sinh ra các quan hệ như sau:

```
Create table BOOK  (Isbn  char(10),
                    Title char(80),
                    Author char(80),
```

```
                    Qty_in_stock integer,
                    Price real,
                    Year_published integer,
                    primary key (Isbn))
Create table CUSTOMER (Cid integer,
                    Cname char(80),
                    Address char(200),
                    Cardnum char(16),
                    primary key (Cid),
                    unique (Cardnum))
Create table ORDER (Isbn char(10),
                    Cid integer,
                    Quantity integer,
                    Order_date date,
                    Ship_date date,
                    primary key (Isbn, Cid),
                    foreign key (Isbn) references BOOK,
                    foreign key (Cid) references CUSTOMER)
```

Chúng ta nhận thấy rằng liên kết nhiều-nhiều "Orders" đã được chuyển thành một quan hệ (bảng).

Quay trở lại vấn đề phát sinh trong phần trên, đến đây chúng ta nhận thấy rằng, quan hệ ORDER chứa một trường Order_date và khoá của bàng bao gồm 2 thuộc tính Cid và Isbn. Chính vì lý do này mà khách hàng không thể đặt 2 đơn hàng cho cùng một cuốn sách ở các ngày khác nhau,

hạn chế này chúng ta hoàn toàn không muốn và trên thực tế có những trường hợp như vậy xảy ra. Khi đó để loại bỏ hạn chế này, chúng ta chỉ cần chèn thêm thuộc tính Order_date vào khoá của quan hệ ORDER, và có quan hệ mới như sau:

```
Create table ORDER (Isbn char(10),
        Cid integer,
        Quantity integer,
        Order_date date,
        Ship_date date,
        primary key (Isbn, Cid, Order_date),
        foreign key (Isbn) references BOOK,
        foreign key (Cid) references CUSTOMER)
```

Tuy nhiên đây có vẻ như không hợp lý với sơ đồ thực thể liên kết, và chúng ta quyết định phải gặp lại khách hàng.

Khách hàng lại cho chúng ta biết thêm một số yêu cầu mà chưa được đề cập đến ở lần trước đó như sau: "Khách hàng có thể đặt mua một số đầu sách trong cùng một đơn đặt hàng. Ví dụ, nếu khách hàng muốn mua 3 quyển sách Văn học và 2 quyển Luật học thì họ có thể yêu cầu trong cùng một đơn đặt hàng". Và chúng ta có thể hỏi khi đó nó sẽ ảnh hưởng như thế nào đến quá trình giao hàng nếu cửa hàng không đủ số lượng cho hai đầu sách đó? Cửa hàng sẽ giao đầy đủ tất cả các cuốn sách theo đơn hàng hay có thể giao từng loại? Và khách hàng cho chúng ta biết như sau: "Ngay sau khi cửa hàng có đủ số lượng sách của một đầu sách như yêu cầu trong đơn hàng, cửa hàng sẽ tiến hành giao hàng ngay mặc dù trong đơn đặt hàng có nhiều đầu sách. Tức là có thể xảy ra trường hợp 3 cuốn sách Văn học sẽ được giao

ngày hôm nay bởi trong kho của cửa hàng có 5 cuốn loại này, tuy nhiên sách Luật học có thể ngày mai mới được giao bởi trong kho chỉ có một cuốn, và ngày mai số lượng sách đó mới về tới kho của cửa hàng. Thêm vào đó, khách hàng có thể đặt nhiều đơn hàng trong một ngày và họ cũng muốn biết thông tin về các đơn đặt hàng mà họ đã đặt".

Như vậy, chúng ta nhận thấy rằng đã có 2 yêu cầu mới phát sinh trong lần gặp gỡ khách hàng này: thứ nhất, hệ thống phải đáp ứng được khả năng đặt nhiều đầu sách khác nhau trong cùng một đơn đặt hàng duy nhất. Thứ 2, một khách hàng có thể đặt các đơn hàng khác nhau trong cùng một ngày. Để có thể giải quyết được vấn đề này, chúng ta cần đưa thêm một thuộc tính mới vào quan hệ ORDER, đó là thuộc tính số hiệu đơn hàng Order_id, sẽ xác định duy nhất một đơn hàng do khách hàng đặt. Tuy nhiên, vì khách mua có thể đặt nhiều đầu sách khác nhau trong cùng một đơn hàng nên giá trị trong hai thuộc tính Order_id và Isbn kết hợp với nhau mới có thể xác định được các giá trị trong thuộc tính Quantity và Ship_date trong quan hệ ORDER.

Các đơn hàng sẽ được gán bởi các số hiệu đơn hàng khác nhau theo thứ tự tuần tự, và đơn hàng sau sẽ có giá trị số hiệu đơn hàng lớn hơn đơn hàng trước. Nếu có nhiều đơn hàng của cùng một khách trong cùng một ngày thì chúng cũng được phân biệt với nhau do các số hiệu đơn hàng là khác nhau. Câu lệnh SQL để tạo quan hệ ORDER này sẽ như sau:

```
Create table ORDER (Order_id integer,
                    Isbn char(10),
                    Cid integer,
```

Quantity integer,

Order_date date,

Ship_date date,

primary key (Order_id, Isbn),

foreign key (Isbn) references BOOK,

foreign key (Cid) references CUSTOMER)

4. Tinh chỉnh lược đồ

Tại bước này, chúng ta phải phân tích tập các quan hệ có thể gây ra sự dư thừa số liệu. Trong quan hệ BOOK chỉ có duy nhất một trường khoá đó là Isbn, và không có phụ thuộc hàm nào khác thoả trên quan hệ này. Do vậy quan hệ BOOK là đã ở dạng chuẩn 3NF. Quan hệ CUSTOMER cũng có duy nhất một thuộc tính khoá đó là Cid, bởi số hiệu thẻ tín dụng là duy nhất nên phụ thuộc hàm Cardnum → Cid cũng thoả. Không còn phụ thuộc hàm nào khác nên CUSTOMER cũng ở dạng chuẩn 3NF.

Như trên đã xác định cặp (Order_id, Isbn) là khoá của quan hệ ORDER. Thêm vào đó, mỗi đơn hàng được đặt bởi một khách mua của một ngày cụ thể nên 2 phụ thuộc hàm sau là thoả:

Order_id → Cid, và Order_id → Order_date

Do vậy chúng ta có thể kết luận rằng quan hệ ORDER không ở dạng chuẩn 3NF. Nên chúng ta phải tiến hành phân rã ORDER thành 2 quan hệ sau:

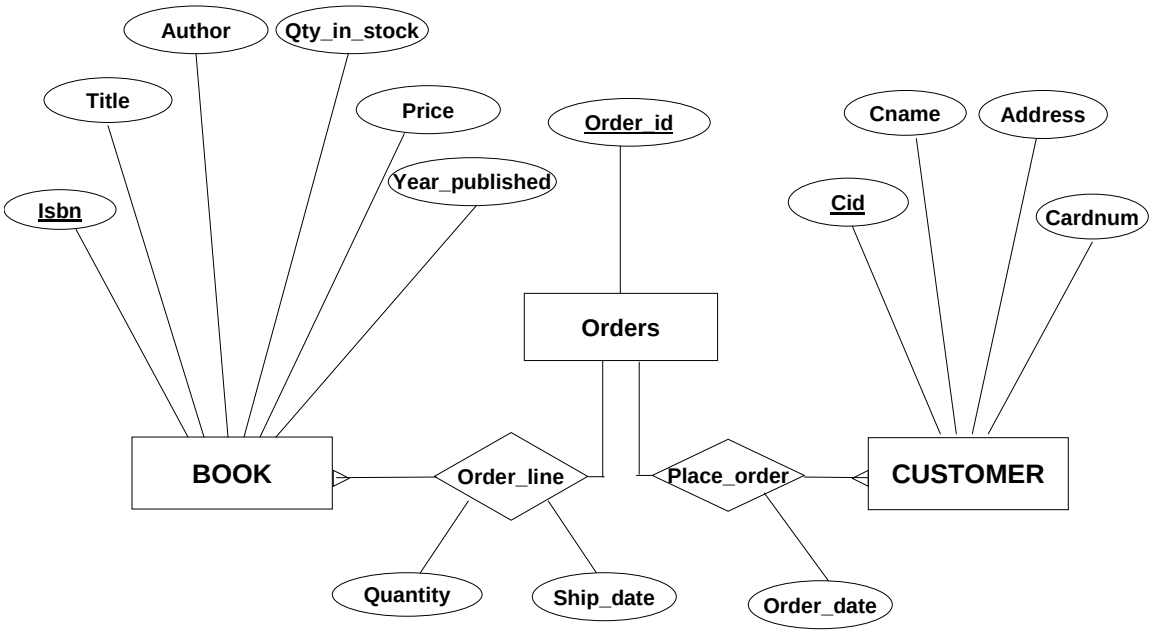
ORDER(Order_id, Cid, Order_date) và

ORDER_LINE(Order_id, Isbn, so_luong, Ship_date)

Như vậy cả 2 quan hệ trên đều ở dạng chuẩn 3NF. Và đến đây chúng ta có các câu lệnh SQL tạo quan hệ như sau:

```
Create table ORDER (Order_id integer,
        Cid integer,
        Order_date date
        primary key (Order_id),
        foreign key (Cid) references CUSTOMER)
Create table  ORDER_LINE (Order_id integer,
        Isbn char(10),
        Quantity integer,
        Ship_date date
        primary key (Order_id, Isbn),
        foreign key (Order_id) references ORDER,
        foreign key (Isbn) references BOOK)
```

Hình 6.4.1 là sơ đồ thực thể liên kết được sửa đổi theo cách thiết kế mới như trên. Chú ý rằng, chúng ta sẽ có sơ đồ này ngay nếu ngay từ đầu chúng ta coi ORDER là một kiểu thực thể thay vì là một liên kết. Mặc dù vậy, việc chúng ta chưa thể hiểu được đầy đủ các nghiệp vụ ngay từ lần gặp gỡ đầu tiên với khách hàng là một điều hết sức bình thường. Quá trình tinh chỉnh lược đồ này có thể được lặp đi lặp lại nhiều lần, đặc biệt đối với những bài toán lớn có nghiệp vụ phức tạp và đó thực sự là một qui trình phát sinh trong thực tế bởi hầu hết các thiết kế ban đầu đều được làm mịn dần trong quá trình xây dựng dự án.



Hình 6.4.1 Sơ đồ thực thể - liên kết được tinh chỉnh

Sau giai đoạn này chúng ta sẽ chuyển sang việc thiết kế cơ sở dữ liệu vật lý

5. Thiết kế cơ sở dữ liệu vật lý

Chủ cửa hàng sách mong muốn rằng khách hàng của họ sẽ tìm kiếm sách thông qua số ISBN trước khi đặt hàng mua sách. Việc đặt đơn hàng mua sách bao gồm: chèn 1 bản ghi vào quan hệ ORDER và một hoặc nhiều bản ghi vào quan hệ ORDER_LINE. Nếu số lượng sách yêu cầu có đủ trong cửa hàng, công việc giao hàng được chuẩn bị và Ship_date trong quan hệ ORDER_LINE được đặt giá trị tương ứng. Thêm vào đó, số lượng sách trong cửa hàng sẽ thay đổi

theo thời gian, tăng lên khi cửa hàng nhập thêm sách và giảm đi khi khách hàng đặt mua.

Như vậy chúng ta phải bắt đầu bằng việc xem xét đến vấn đề tìm kiếm sách thông qua giá trị ISBN. Vì Isbn là trường khoá nên việc tìm kiếm trên nó sẽ trả lại nhiều nhất là 1 bản ghi. Như vậy để tăng tốc độ tìm kiếm khi khách mua của cửa hàng tìm sách, chúng ta sẽ phải tạo một chỉ số băm không phân cụm (unclustered hash index) trên trường Isbn.

Tiếp theo chúng ta phải cân nhắc đến vấn đề cập nhật số lượng sách. Để cập nhật giá trị Qty_in_stock cho mỗi một đầu sách, đầu tiên phải tìm đến bản ghi đó thông qua giá trị ISBN, tệp chỉ số trên sẽ giúp công việc tìm kiếm nhanh hơn.

Ngoài ra, do khách mua cũng sẽ thường xuyên tìm kiếm sách thông qua tiêu đề hay tên tác giả, nên chúng ta quyết định xây dựng thêm các tệp chỉ số trên 2 trường Author và Title, hơn nữa việc bảo trì các tệp chỉ số này không phức tạp lắm do danh mục đầu sách ít khi thay đổi (chỉ khi có thêm đầu sách mới), mặc dù số lượng sách trong cửa hàng liên tục thay đổi.

Với quan hệ CUSTOMER, mỗi một khách mua được xác định bởi một giá trị định danh duy nhất. Do hầu hết các tìm kiếm trên CUSTOMER là dựa trên các giá trị này nên chúng ta quyết định tạo một tệp chỉ số băm phân cụm trên trường Cid để tăng tốc độ tìm kiếm.

Sang đến quan hệ ORDER, chúng ta nhận thấy rằng có 2 loại truy vấn: chèn thêm các đơn hàng mới và trả về các đơn hàng đã đặt. Cả 2 truy vấn này đều sử dụng thuộc tính Order_id như là khoá tìm kiếm, do vậy chúng ta quyết định xây dựng một tệp chỉ số trên nó. Tuy nhiên phải lựa chọn

giữa hai loại tệp chỉ số là B-tree hay băm. Trong trường hợp này giá trị các đơn hàng tăng theo thời gian, tương ứng với các giá trị trong thuộc tính Order_date, sắp xếp theo Order_id cũng hiệu quả như sắp xếp theo ngày đặt, do vậy chúng ta quyết định lựa chọn loại B-tree phân cụm làm tệp chỉ mục cho trường Order_id.

Trong quan hệ ORDER_LINE, hầu hết các thao tác trên quan hệ này là phép chèn, một số phép cập nhật trên thuộc tính Ship_date, và phép liệt kê danh mục các sách đặt mua của một đơn hàng đã cho. Nếu quan hệ này được sắp xếp theo Order_id, khi đó tất cả các phép chèn đơn giản chỉ là việc thêm vào cuối quan hệ và do đó rất hiệu quả. Một tệp chỉ mục phân cụm B-tree nên được xây dựng cho giá trị này, tệp chỉ số này cũng làm tăng tốc độ cho câu lệnh tìm kiếm liệt kê các sách đã đặt mua của một đơn hàng nào đó. Với phép cập nhật giá trị trong thuộc tính Ship_date, chúng ta cần phải tìm kiếm trên giá trị của 2 trường là Order_id và Isbn. Tệp chỉ số trên Order_id xây dựng phía trên cũng hữu ích cho phép tìm kiếm này, tuy nhiên nếu chúng ta có một tệp chỉ số trên cả 2 thuộc tính thì vẫn sẽ nhanh hơn.

Một số điều chỉnh đối với cơ sở dữ liệu:

Giả thiết sau khi cơ sở dữ liệu được đưa vào sử dụng và khách hàng bắt đầu đặt mua sách qua trang web, sau vài tháng, chúng ta nhận được một số phàn nàn của cửa hàng rằng khách mua của họ thấy việc xử lý các đơn đặt hàng của họ ngày càng chậm. Mặc dù các thao tác vẫn được thực hiện thành công, nhưng do số lượng bản ghi trong các quan hệ ORDER và ORDER_LINE ngày càng nhiều và các quan hệ này ngày càng lớn so với ban đầu nên ảnh hưởng đến tốc độ xử lý đơn hàng? Nếu điều này xảy ra, chúng ta phải xem

xét lại bản thiết kế. Chúng ta nhận thấy rằng, ở đây, có 2 loại đơn hàng, một loại đơn hàng đã được thực hiện xong tương ứng với các đầu sách trong đơn đã được giao và một loại còn chưa hoặc đang được thực hiện tương ứng với các đầu sách trong đơn hàng chưa hoặc đang được giao. Và hầu hết các khách mua có nhu cầu tìm kiếm trên loại đơn hàng thứ 2 (chưa hoặc đang được giao). Ngoài ra, cũng nhận thấy rằng, đối với loại đơn hàng này thì số lượng rất nhỏ so với loại đơn hàng đã được thực hiện xong. Do vậy chúng ta quyết định tiến hành điều chỉnh thêm đối với 2 quan hệ này, chúng ta tiến hành phân vùng (partition) ngang 2 quan hệ này thành 4 quan hệ mới đó là NEWORDER, OLDORDER, NEWORDER_LINE và OLDORDER_LINE. Rõ ràng, một đơn hàng luôn đi thành từng cặp hoặc cũ hoặc mới tương ứng với nhau. Đến đây chúng ta đã khắc phục được hiện tượng tìm kiếm trên những đơn hàng còn đang được thực hiện giao.

Tuy nhiên, đối với những truy vấn như tìm tất cả các đơn hàng của một khách hàng nào đó thì vẫn chậm, bởi khi đó hệ thống vẫn phải tìm kiếm trên cả 2 loại quan hệ cũ và mới trên, nhưng việc tìm kiếm này là có thể chấp nhận được và nó ít khi xảy ra, thường là các báo cáo tháng, quý, năm của cửa hàng.

6. Vấn đề an toàn bảo mật

Chúng ta đã hoàn tất việc thiết kế vật lý cho cơ sở dữ liệu. Tiếp theo chúng ta phải xem xét đến vấn đề an toàn dữ liệu.

Thông thường, có 3 nhóm người dùng có nhu cầu truy cập cơ sở dữ liệu, đó là: khách mua, nhân viên cửa hàng, và chủ cửa hàng. Ngoài ra, cũng cần có người quản trị hệ thống,

chịu trách nhiệm đối với các hoạt động của cơ sở dữ liệu và có quyền truy nhập vào toàn bộ cơ sở dữ liệu.

- Với chủ cửa hàng, anh ta sẽ có đầy đủ các quyền trên tất cả các quan hệ, tuy nhiên anh ta không được phép tạo dựng thêm các đối tượng dữ liệu mới. Do vậy người quản trị sẽ phân quyền thao tác đầy đủ trên các đối tượng dữ liệu trong cơ sở dữ liệu cho anh ta.
- Với khách mua, họ chỉ được phép tìm kiếm trên quan hệ BOOK và có thể đặt hàng trực tuyến, nhưng họ không được phép truy nhập đến các bản ghi của các khách hàng khác cũng như các đơn đặt hàng khác. Do vậy ở đây chúng ta phải giới hạn theo 2 cách. Cách thứ nhất, phải sử dụng các đặc trưng an toàn của cơ sở dữ liệu để giới hạn phạm vi truy suất đến các dữ liệu nhạy cảm. Ví dụ khi tạo lập một khoản mục mới cho khách hàng, người quản trị cơ sở dữ liệu sẽ tạo dựng luôn những khung nhìn chỉ cho phép anh ta xem các dữ liệu của riêng mình thông qua các khung nhìn đó. Cách thứ 2, chúng ta có thể giới hạn thông qua việc tạo các giao diện trong ứng dụng trên Web. Và mọi truy vấn đều được gắn thêm giá trị Cid của khách mua hiện thời.
- Với các nhân viên của cửa hàng, tùy từng đối tượng mà người quản trị sẽ phân cho họ các quyền thao tác trên từng đối tượng dữ liệu cụ thể. Ví dụ với người quản lý kho, anh ta chỉ được phép cập nhật trên quan hệ BOOK. Với người quản lý các thông tin về khách hàng thì chỉ có thể thao tác trên quan hệ CUSTOMER mà thôi. Còn với những người quản lý các đơn hàng, họ có các quyền cập nhật trên hai quan

hệ NEWORDER và NEWORDERLINE, nhưng với các quan hệ còn lại như BOOK, CUSTOMER thì chỉ có quyền xem mà thôi.

Cụ thể, để hạn chế quyền truy nhập của khách mua, chúng ta có thể tạo một tài khoản có tên là khachmua và gán cho tài khoản này các quyền sau:

Select on BOOK, NEWORDER, OLDORDER, NEWORDER_LINE, OLDORDER_LINE

Insert on NEWORDER, NEWORDER_LINE

Với các nhóm nhân viên, như trên đã phân tích, họ có thể có quyền chèn thêm một đầu sách mới, cập nhật lại số lượng sách trong cửa hàng và sửa lại thông tin đặt hàng của khách trong một số trường hợp, cập nhật lại thông tin liên quan đến khách hàng ngoại trừ giá trị tại trường Cardnum. Trên thực tế, nhóm này còn không được phép biết đến các giá trị tại thuộc tính này. Do vậy chúng ta phải tạo thêm các khung nhìn sau:

Create view CUSTOMERINFO (Cid, Cname, Address) as

Select Cid, Cname, Address

From CUSTOMER

Và cấp các quyền sau cho nhóm nhân viên này:

Select on CUSTOMERINFO, BOOK, NEWORDER, OLDORDER, NEWORDER_LINE, OLDORDER_LINE

Insert on CUSTOMERINFO, BOOK, NEWORDER, OLDORDER, NEWORDER_LINE, OLDORDER_LINE

Update on CUSTOMERINFO, BOOK, NEWORDER, OLDORDER, NEWORDER_LINE, OLDORDER_LINE

Delete on BOOK, NEWORDER, OLDORDER, NEWORDER_LINE, OLDORDER_LINE

Ngoài ra, do sử dụng trang web trên mạng, nên việc gửi nhận thông tin trên mạng có thể cần được mã hoá và nên sử dụng giao thức SSL (Security Socket Layer).

Tài liệu tham khảo

[1] C.Goble - J.Keane, *Advances in Databases*, Springer , **1997**.

[2] H.David, *Data Analysis for Database Design. Third ed*, Butterworth, **2003**.

[3] J.Teorey, Database Modeling & Design , Morgan Kauf, **1999**.

[4] R.McFadden and al., *Modern Database Management. (5th. Ed.)*, Addison-Wesley, **2000**.

[5] R.Raghu, G.Jahannes, *Database Management Systems*, McGraw-hill, **2000**.

[6] C.J.Date, *Nhập môn các hệ cơ sở dữ liệu* (Hồ Thuần dịch), Nhà xuất bản thống kê, **1986**.

[7] Hồ Thuần, *Đề cương chi tiết giáo trình cơ sở dữ liệu nâng cao*, Bộ GD&ĐT, **1998**.

[8] Nguyễn Kim Anh, *Nguyên lý của các hệ cơ sở dữ liệu*, Nhà xuất bản Đại học quốc gia Hà Nội, **2004**.

[9] Nguyễn Văn Ba, *Phân tích và thiết kế hệ thống thông tin*, Nhà xuất bản Đại học quốc gia Hà Nội, **2002**.

[10] Trần Thành Trai, *Phân tích và thiết kế hệ thống thông tin quản lý*, Nhà xuất bản thống kê, **1994**.

[11] Vũ Đức Thi, *Cơ sở dữ liệu kiến thức và thực hành* , Nhà xuất bản thống kê, **1997**.

[12] http://www.oracle.com

[13] http://metalink.oracle.com

[14] http://www.microsoft.com

[15] http://www.microsoft.com/sql/evaluation/overview/default.asp

[16] http://docs.rinet.ru/

[17] http://www.arch.su.edu.au/kcdc/teaching

Một số khái niệm đầu tiên về các hệ thống cơ sở dữ liệu

Cơ sở dữ liệu

- Định nghĩa
  - Là tập các dữ liệu bền vững, có thể chia sẻ.
- Lí do cần thiết dùng cơ sở dữ liệu
  - Công việc xử lí dữ liệu hàng ngày, thông tin về sách, về ngân hàng, đặt hàng...
  - Dữ liệu thể hiện các sự kiện thường ngày
  - Cần thiết tổ chức dữ liệu để truy cập dễ dàng

Đặc tính của cơ sở dữ liệu

- **Bền vững** – tức dữ liệu được đặt trên thiết bị lưu trữ ổn định, cho phép sử dụng nhiều lần
- **Chia sẻ** – tức cơ sở dữ liệu cho phép nhiều người dùng, nhiều công việc.
  - Cơ sở dữ liệu cá nhân
  - Cơ sở dữ liệu nhóm
  - Cơ sở dữ liệu xí nghiệp
- **Liên kết** – tức dữ liệu được lưu tại nhiều nơi, có liên kết, như bức tranh tổng thể

Thí dụ cơ sở dữ liệu về đại học

Đăng kí

Ghi lại bậc học

<u>Entities</u>:
Students, faculty, courses
Offerings, enrollments

<u>Relationships:</u>
Faculty teach offerings,
Students enroll in
Offerings, offerings made
Of courses

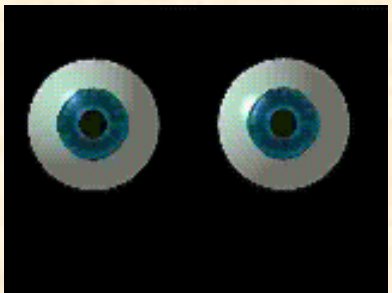
Gắn với khoa

Thời khóa biểu

Cho phép biết người đứng lớp

Cho biết tên sinh viên, với lớp học

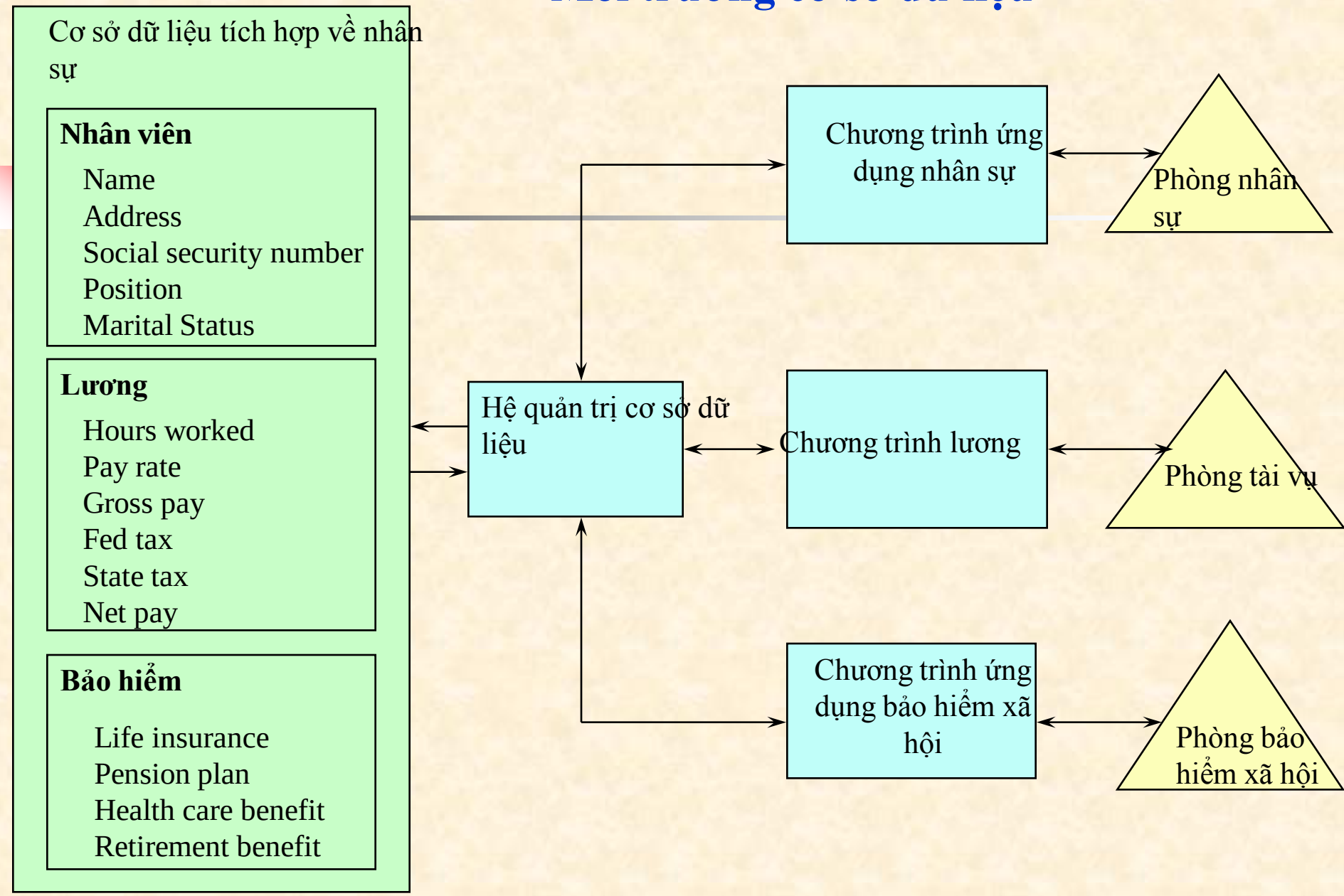
Biết được khoa, bậc học



Hệ quản trị cơ sở dữ liệu

- Hệ quản trị cơ sở dữ liệu (database management system - DBMS) là tập các phần mềm cho phép tạo, sử dụng, bảo trì cơ sở dữ liệu
- Trước tiên, DBMSs đảm bảo lưu trữ và tìm kiếm hiệu quả
- Hệ quản trị là phần mềm thương mại
- Theo thị trường, hệ quản trị có các khía cạnh : thu thập dữ liệu, lưu trữ, bảo trì, lập báo cáo...

Môi trường cơ sở dữ liệu



Khía cạnh thương mại của hệ quản trị cơ sở dữ liệu

- **Xác định cơ sở dữ liệu** – các công cụ ngôn ngữ và đồ họa cho phép xác định thực thể, mối quan hệ, điều kiện ràng buộc, quyền truy cập...
- **Truy cập phi thủ tục** – các công cụ ngôn ngữ và đồ họa cho phép truy cập dữ liệu mà không phải viết chương trình phức tạp
- **Phát triển ứng dụng** – công cụ đồ họa để phát triển thực đơn, khuôn dạng nhập dữ liệu, báo cáo...
- **Giao diện ngôn ngữ phi thủ tục** – ngôn ngữ kết hợp truy cập phi thủ tục với các khả năng của ngôn ngữ lập trình đầy đủ
- **Xử lí giao tác** – cơ chế điều khiển để tránh xung đột dữ liệu và khôi phục sai sót
- **Tinh chỉnh dữ liệu** – công cụ giám sát và nâng cao hiệu năng hệ thống

Hai khung nhìn cơ sở dữ liệu

- **Khung nhìn vật lí** : mô tả nơi lưu dữ liệu
  - Thiết bị, đĩa, rãnh, bề mặt, từ quạt, bản ghi...
  - Băng từ, khối dữ liệu, số các bản ghi...
- **Khung nhìn logic**: mô tả ứng dụng cần đến dữ liệu
  - Sự kiện cần thiết của xí nghiệp
  - Tên, độ dài bản ghi, kiểu dữ liệu

- DBMS cho phép người dùng hay người lập trình không phải quan tâm đến nơi, cách thức lưu trữ dữ liệu

Tiến hóa của công nghệ cơ sở dữ liệu

Kỉ nguyên	Thế hệ	Định hướng	Nét chính
1960s	thế hệ 1	File	cấu trúc file và giao diện chương trình
1970s	thế hệ 2	mạng	mạng, phân cấp các bản ghi chương trình chuẩn
1980s		quan hệ	quan hệ tối ưu, xử lí giao tác
1990s		đối tượng	đa phương tiện cơ sở dữ liệu động, xử lí phân tán

Chia sẻ thị trường về hệ quản trị cơ sở dữ liệu xí nghiệp

Sản phẩm	Chiếm thị trường	Bình luận
IBM DB2	33%	Dominates the MVS and AS/400 environments
Oracle	29%	Dominates the Unix environment (61%), leader in NT environment (46%)
Microsoft SQL server	10%	30% market share in NT environment; no presence in other environments
Informix	4%	13% share of Unix market
Sybase SQL server	4%	
Khác	20%	Includes CA, NCR, Progress Software, NEC etc.

Xác định cơ sở dữ liệu

- Để xác định cơ sở dữ liệu, cần xác định thực thể và mối quan hệ
- DBMS dùng các bảng để lưu tập các thực thể. Mối quan hệ nhằm vào các liên kết giữa các bảng
- Ngôn ngữ mô tả dữ liệu (DDL) xác định mỗi phần tử dữ liệu như là bản ghi trong bảng, trước khi phần tử dữ liệu được chuyển sang dạng dùng cho người lập trình
- Ngôn ngữ xử lí dữ liệu (DML) thông dụng là SQL

- Người dùng hay dùng SQL, *Structured Query Language*

- Cho phép tìm kiếm phức tạp, với điều kiện

  SELECT tên FROM sinh viên WHERE toán > 7 and tuổi < 30

Ngôn ngữ khác : QUEL, QBE (Query by Example, QBE)

Ngôn ngữ SQL

Tổ chức dữ liệu

- 4 tiếp cận
  - Mô hình cơ sở dữ liệu quan hệ
  - Mô hình cơ sở dữ liệu phân cấp
  - Mô hình cơ sở dữ liệu mạng

  - Mô hình cơ sở dữ liệu đa chiều

Mô hình cơ sở dữ liệu quan hệ

- Sử dụng bảng 2 chiều
- Bảng được gọi là quan hệ
- Dựa trên lí thuyết tập
- Dòng dữ liệu = bản ghi = bộ (tuple)
- Cột dữ liệu = trường = thuộc tính
- Dùng tập các bảng thay vì một bảng, để tạo nên cơ sở dữ liệu

Thí dụ mô hình quan hệ

East Coast Mgrs.

NAME	TITLE	AGE	DIVISION
Smith, A.	Dir., Accntg	43	China
Jones, W.	Dir., TQM	32	Stemware

Thực thể, thuộc tính, giá trị

- Thực thể : là người, đối tượng hay sự kiện mà người ta đang quản lí thông tin về chúng
- Thuộc tính : tính chất, chất lượng, mô tả thực thể cụ thể
- Trường khóa : thông tin xác định duy nhất bản ghi, để có thể tìm kiếm, lưu trữ và sắp x

Các bước thiết kế cơ sở dữ liệu

Yêu cầu của người dùng

Thiết kế khái niệm

(có thể là mô hình thực thể quan hệ)

Thiết kế logic

(có thể mô hình quan hệ)

thiết kế vật lí

(tối ưu về hiệu năng)

Yêu cầu của người dùng

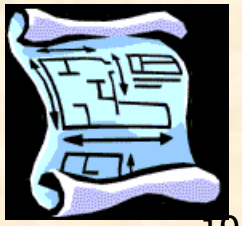
- **yêu cầu cơ sở dữ liệu từ phía người dùng**

- **Khung nhìn người dùng : là phần cơ sở dữ liệu, là quan trọng đối với người dùng**

- **Một số khó khăn**

Thiết kế khái niệm
Mô hình thực thể quan hệ

- Thực thể
  - Là cái đang quan tâm
    - Có thể là trực tiếp, hay gián tiếp.
- Thuộc tính – tính chất, điều vốn có của thực thể
- Khóa
  - Mỗi thực thể cần được xác định duy nhất bằng thuộc tính, hay nhóm các thuộc tính, gọi là khóa



Thực thể Đặt hàng

Thuộc tính

Số đặt hàng	Ngày đặt hàng	Mã số hàng	Số lượng	Tiền
4340	02/08/94	1583	2	1740

Trường khóa

Bản ghi này mô tả thực thể ORDER và các thuộc tính. Các giá trị riêng đối với yêu cầu đặt hàng là các giá trị thuộc tính. Trường khóa là *Order number* do mỗi đặt hàng gắn với con số duy nhất.

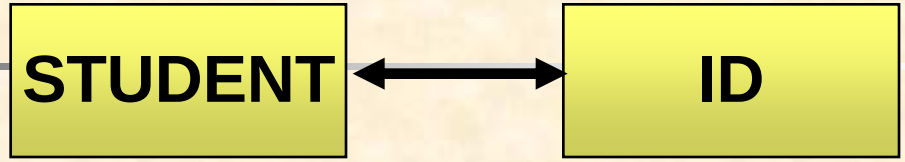
Order : [<u>Order number</u> , order date, item number, quantity, amount]

Mối quan hệ

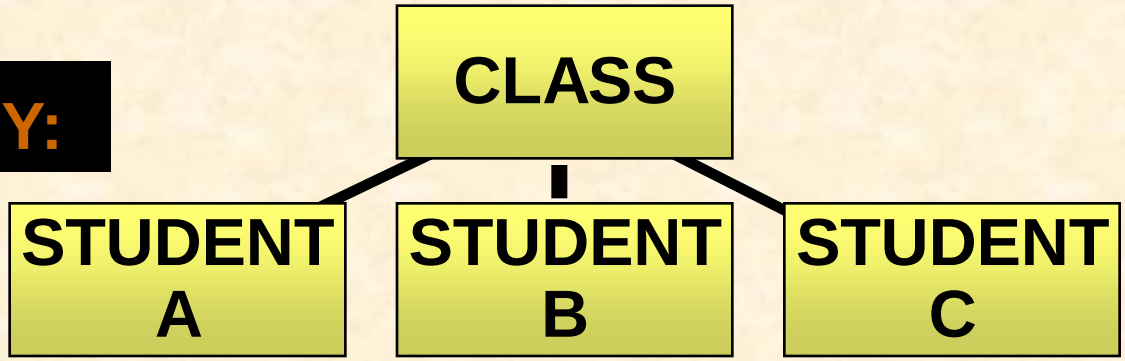
- Là liên kết giữa các thực thể.
  - Có dạng 1 ngôi (quản lí), 2 ngôi, nhiều ngôi
- Bậc quan hệ chỉ số của mỗi thực thể tham gia trong mối quan hệ :
  - Một một (1:1)
  - Một nhiều (1:N)
  - Nhiều nhiều (N:M)
- Mối quan hệ M:N có thể có thuộc tính

Kiểu quan hệ

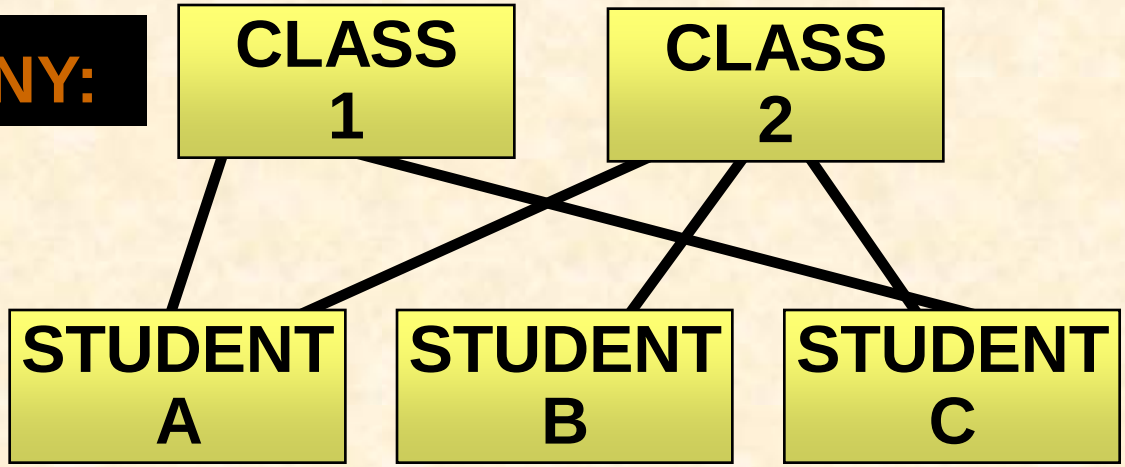
ONE-TO-ONE:



ONE-TO-MANY:



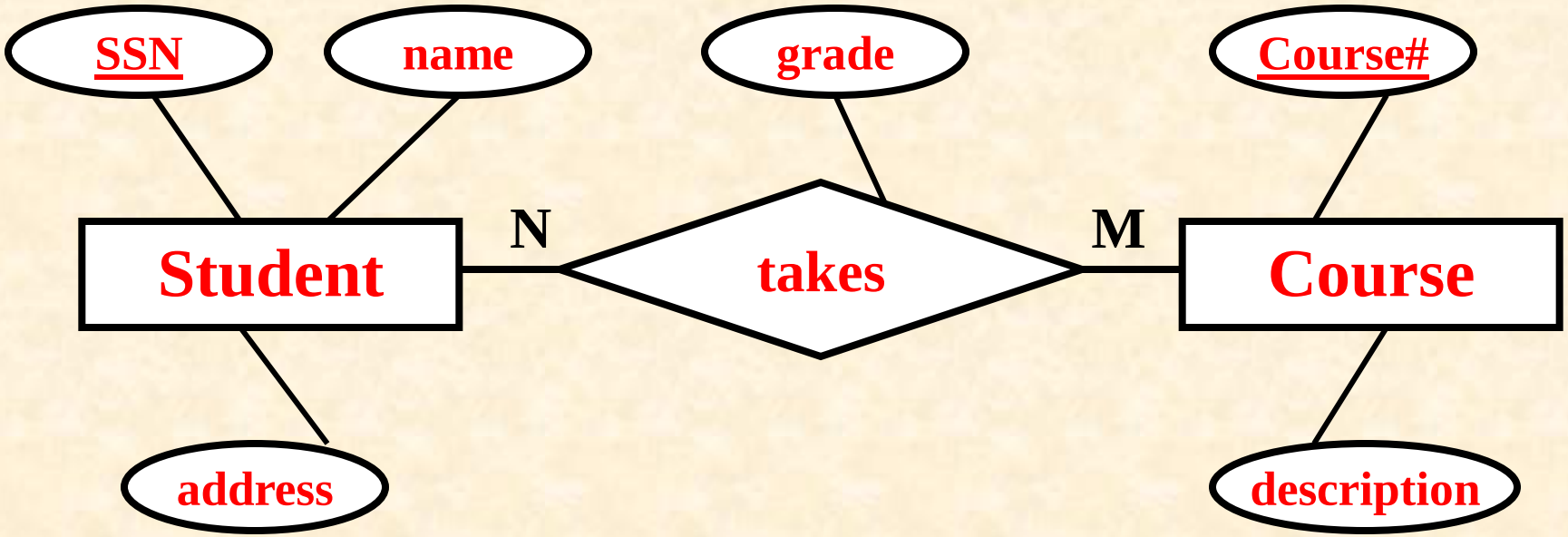
MANY-TO-MANY:



Sơ đồ thực thể quan hệ

- Mỗi thực thể được thể hiện bằng hình chữ nhật.
- Mối quan hệ được thể hiện qua hình thoi.
- Các thuộc tính được thể hiện qua hình elip.
- Các bậc viết kề bên thực thể

Sơ đồ E-R (diagram)



Chuyển mô hình ER sang mô hình quan hệ

Dùng quan hệ để thể hiện thực thể

Quan hệ N:M được thể hiện bằng quan hệ tách biệt

- Khóa là nối kết các khóa thực thể.
- Các thuộc tính của mối quan hệ không là khóa.

Khóa ngoài được xác định “thuộc tính của một bảng quan hệ, là khóa chính của quan hệ khác”

Thí dụ Student-Course

Thiết kế cơ sở dữ liệu để theo dõi sinh viên với bài giảng, và kết quả học nhận được

Thực thể : Student: [SSN, name, address]
Course: [Course-Id, description]

Mối quan hệ Student takes Course: [grade]
N : M

Thí dụ về SQL

- Tìm tên và số bảo hiểm của sinh viên.

```
SELECT Name, SSN
FROM Student;
```

SELECT	cột
FROM	bảng
WHERE	điều kiện

Name	SSN
M. Tomkins	111-22-3333
L. Richardo	444-71-2222
H. McEnroe	795-44-1111

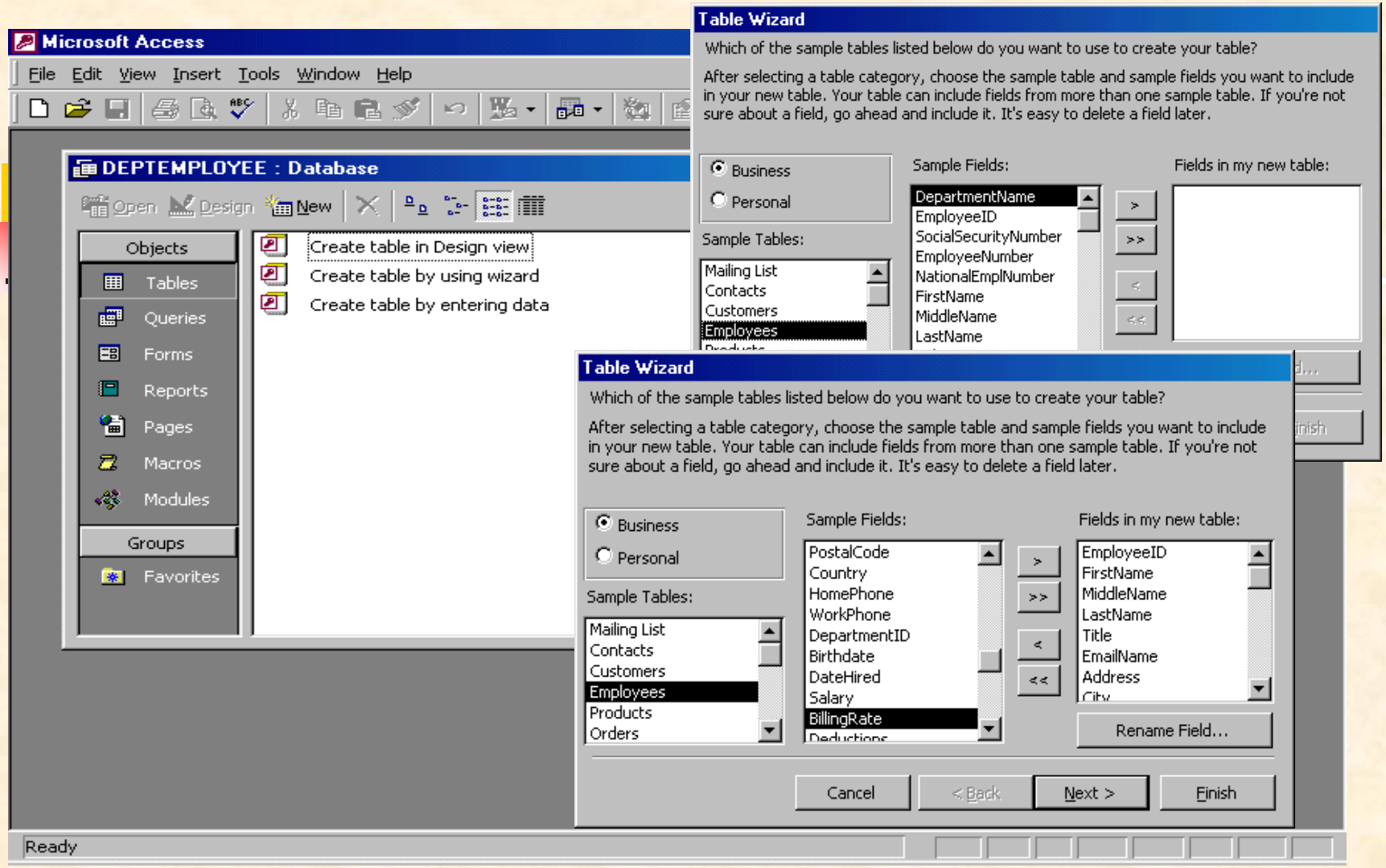
Điều kiện và nối trong SQL

- Tìm SSN của sinh viên đạt kết quả B.

```
SELECT  SSN
FROM    Takes
WHERE   Grade = B;
```

- Có thể nối hai bảng và rút ra thông tin. Thí dụ tìm tên các sinh viên kết quả A.

```
SELECT  Student.Name
FROM    Student, Takes
WHERE   Takes.grade = A AND Takes.SSN = Student.SSN
```



MS Access cho phép thể hiện thực thể như các bảng. Trong bảng có thuộc tính. Người ta có thể tạo bảng theo nhiều cách, như trong hình, theo Wizard...

Lí do cần mô hình hóa dữ liệu

- Một cơ sở dữ liệu cần thể hiện thế giới thực
- Chỉ mô hình hóa mới thể hiện được thế giới thực
- Mô hình hóa nhấn mạnh thể hiện thực tế, sự phức tạp của kinh doanh
- Thể hiện đồ họa tốt cho thực tế và cả dữ liệu trong cơ sở dữ liệu
- Đích của công việc là định tên sự kiện trong cơ sở dữ liệu

Phát triển hệ thống thông tin dựa trên dữ liệu

Lĩnh vực bài toán

Thiết kế khái niệm

Conceptual Schema, e.g., ER Model

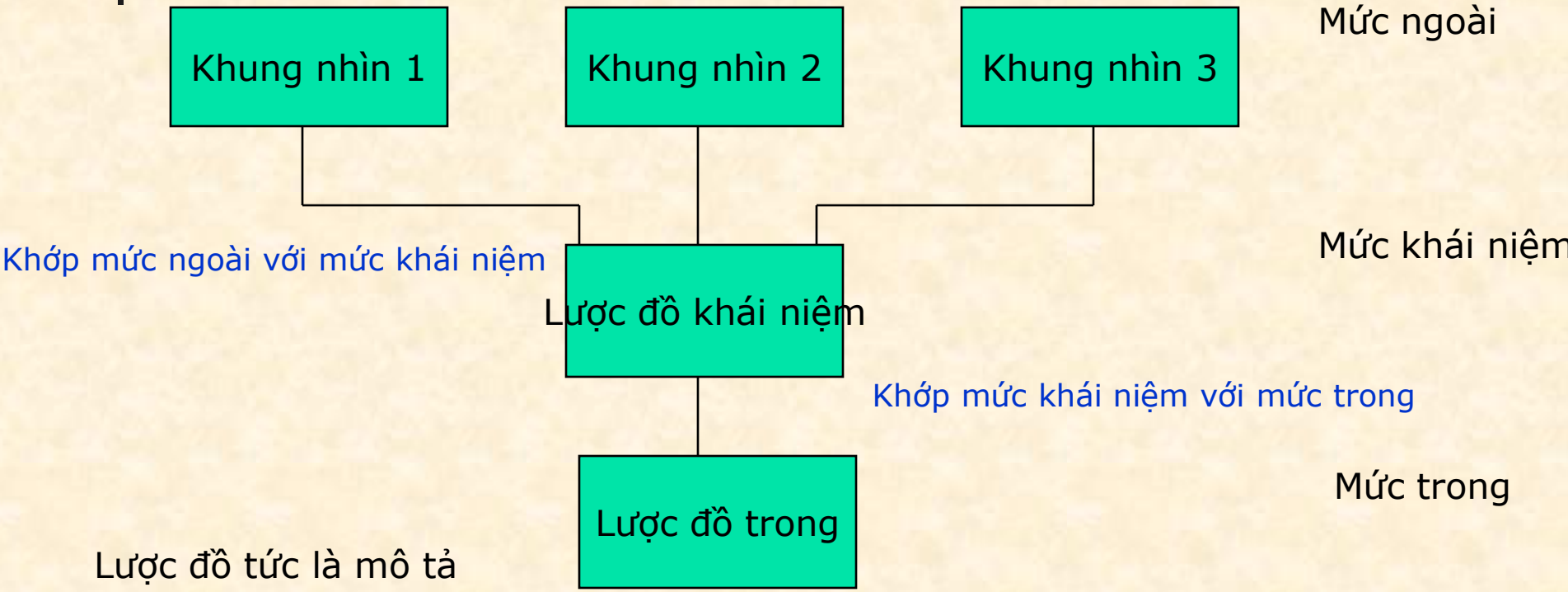
thiết kế logic

Logical Schema, e.g., Relational Model

thiết kế vật lí

Physical Schema, e.g., Access implementation

Ba kiến trúc lược đồ



Lược đồ tức là mô tả

Độc lập dữ liệu có nghĩa cơ sở dữ liệu cần được xác định độc lập với chương trình ứng dụng mà nó sử dụng, như báo cáo, khuôn dạng...

Quản lí dữ liệu

vấn đề kinh doanh thế giới thực

- Một số khó khăn
  - **Khối lượng dữ liệu tăng nhanh, mà cần giữ dữ liệu**
  - **Nhiều truy cập, sử dụng các nguồn, thiết bị khác nhau**
  - **Chỉ phần nhỏ dữ liệu trong tổ chức được dùng trong trợ giúp quyết định**
  - **Dữ liệu ngoài cũng cần thiết cho quyết định**

Quản lí dữ liệu ...

- Dữ liệu thô có trong nhiều hệ thống hợp pháp
- Yêu cầu mang tính pháp luật đối với dữ liệu khác nhau ở các nước
- Có nhiều công cụ quản trị dữ liệu
- Cần có an toàn, toàn vẹn dữ liệu

- Nhận xét chung, dữ liệu cần :
  - Có tính thời sự
  - Chính xác

Quản lí dữ liệu ...

- Về lịch sử, dữ liệu được tổ chức phân cấp để quản lí các giao tác
- Phân cấp là hiệu quả đối với xử lí tác nghiệp, số lượng lớn các dữ liệu
- Mô hình trước (mạng, phân cấp) không tiện cho quản trị, cho hỏi dữ liệu
- Cơ sở dữ liệu quan hệ có nhiều chức năng

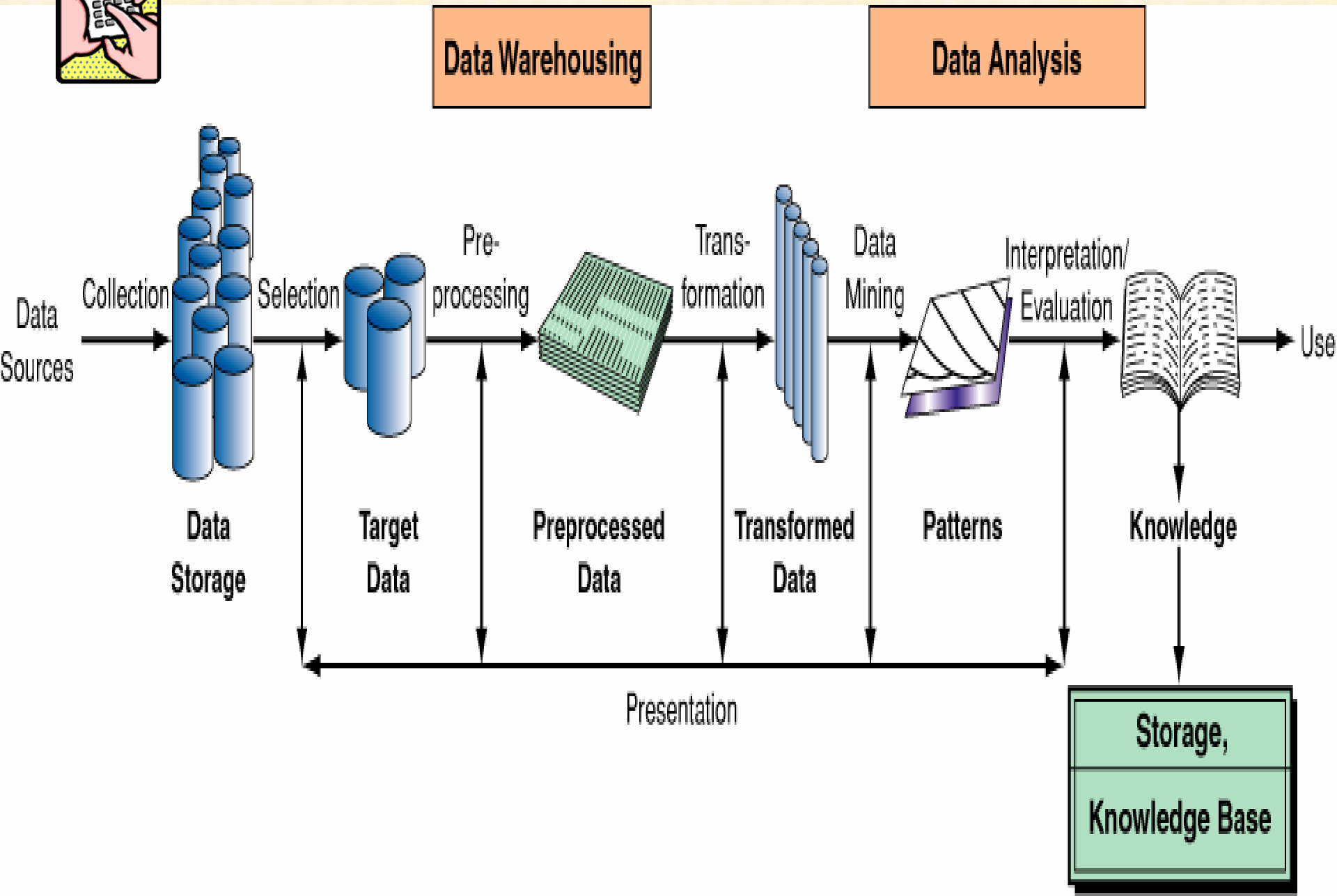
Quản lí dữ liệu ...

- Cơ sở dữ liệu quan hệ
  - Tiện lợi cho tính toán người dùng qua định nghĩa đơn giản, các câu hỏi, khuôn dạng, báo cáo...
  - Có trợ giúp quyết định
- Đối với kiến trúc khách/ chủ, cơ sở dữ liệu trở nên phân tán
- Cơ sở dữ liệu nhiều chiều và nhiều khối dữ đến kiến thức về kho dữ liệu

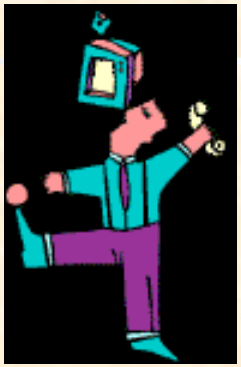
Chu kì vòng đời dữ liệu và phát hiện tri thức

- Y tưởng chính:
  - Chuyển hóa dữ liệu, thông tin và tri thức
  - Quá trình :
    - Thu thập dữ liệu từ nhiều nguồn
    - Lưu trong cơ sở dữ liệu
    - Làm sạch dữ liệu và lưu trong kho dữ liệu
    - Thực hiện khai phá dữ liệu, thu được tri thức
    - Lưu kết quả trong cơ sở tri thức

Chuyển dữ liệu sang tri thức



Nguồn dữ liệu và thu thập dữ liệu



- Dữ liệu gồm
  - Tư liệu
  - Hình ảnh
  - Bản đồ
  - Âm thanh
  - Hoạt hình
  - Khái niệm, ý nghĩ, ý kiến

- Dữ liệu có thể là thô, hay đã được tổng hợp
- Dữ liệu có thể là từ trong, bên ngoài hay của cá nhân

Kiểu dữ liệu

- Dữ liệu trong
  - Con người, sản phẩm, quá trình,
- Cá nhân
  - Các đánh giá chủ đề, ý kiến, qui tắc. Một số dữ liệu là "ẩn", số khác là hiện.
- Dữ liệu ngoài
  - Các báo cáo, cơ sở dữ liệu ngoài, hình ảnh

Thu thập dữ liệu

- Vấn đề :
  - Chất lượng dữ liệu
  - Toàn vẹn dữ liệu, tức thay đổi dữ liệu tại một nơi sẽ truyền khắp các nơi
- Phương pháp thu thập
  - Thủ công
  - Có công cụ, đầu dò, thu
  - Quét hay tải tự động

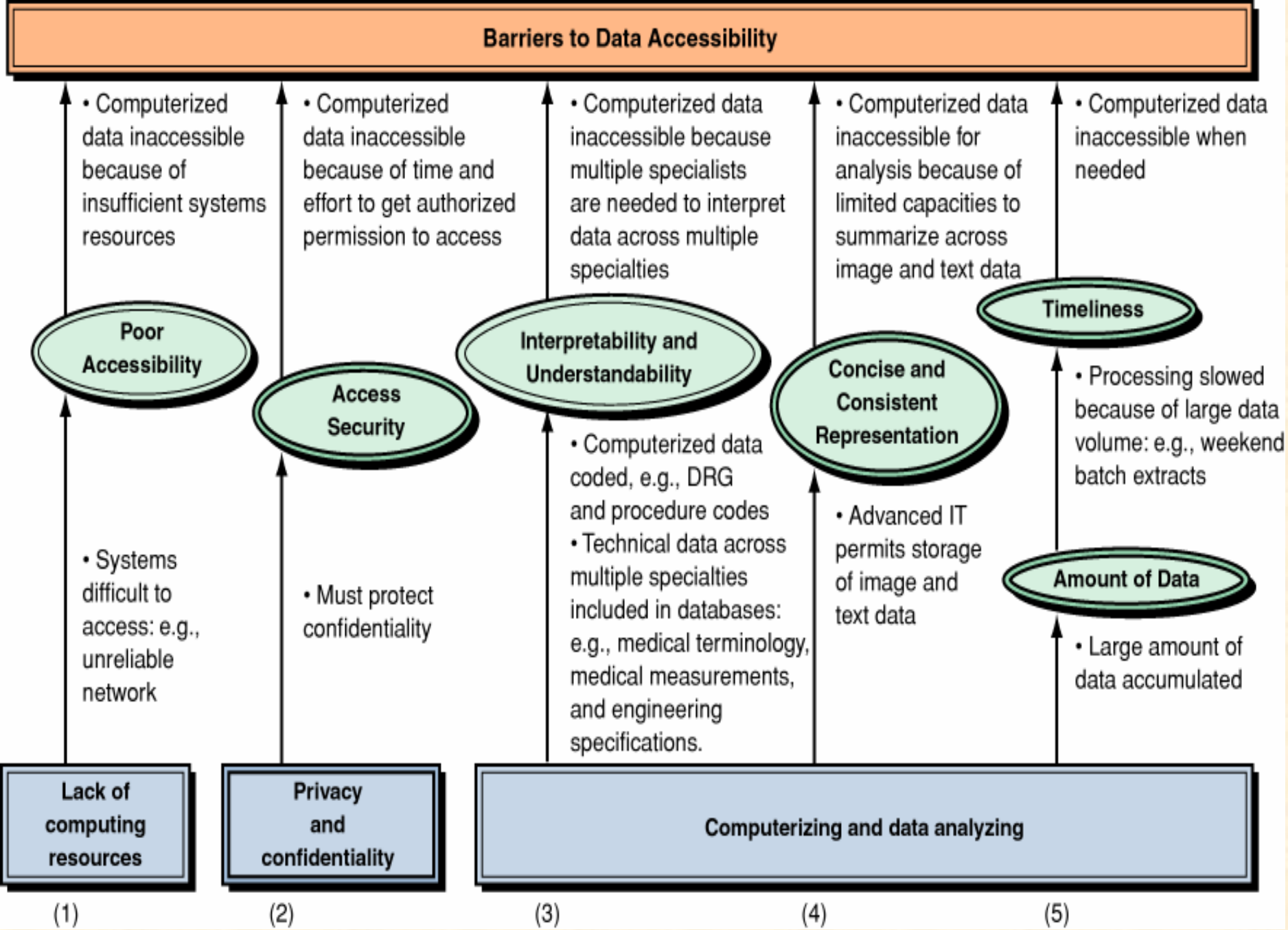
- Quản lí luồng dữ liệu (Data flow manager DFM)
  - Trợ giúp thu thập dữ liệu từ nhiều nguồn
  - Có hệ thống DSS, bộ xử lí dữ liệu trung tâm, bộ toàn vẹn dữ liệu, nối với nguồn ngoài

Chất lượng dữ liệu

- Chất lượng dữ liệu quyết định tính sử dụng được
- Các sai sót tiềm năng
  - Dữ liệu không chính xác
  - Dữ liệu mơ hồ, mờ
  - Dữ liệu không được chỉ số hóa đúng
  - Không có dữ liệu đang cần

Tham số về chất lượng dữ liệu :

Về tính chất	Chính xác, đối tượng, tin cậy, uy tín
Về truy cập	Truy cập được, an toàn truy cập
Về ngữ cảnh	Thể hiện, có giá trị, đầy đủ
Về thể hiện	Dễ hiểu, diễn tả, chi tiết



o cản truy cập dữ liệu

Xử lí phân tích so với xử lí giao tác

- Xử lí phân tích yêu cầu xử lí hàng ngày các giao tác về tổ chức như thanh taón, đặt hàng...
- Các cơ sở dữ liệu và hệ thống xử lí cần thiết được gọi là hệ thống tác nghiệp
- Nhu cầu thị trường đòi hỏi có xử lí phân tích

- Xử lí phân tích trực tuyến (OLAP) cho phép người dùng
  - Truy cập dữ liệu dễ dàng
  - Ra quyết định nhanh
  - Thao tác chính xác và hiệu quả
  - Mềm dẻo
- OLAP thường đi với DSS, EIS và các hoạt động hướng người dùng khác.

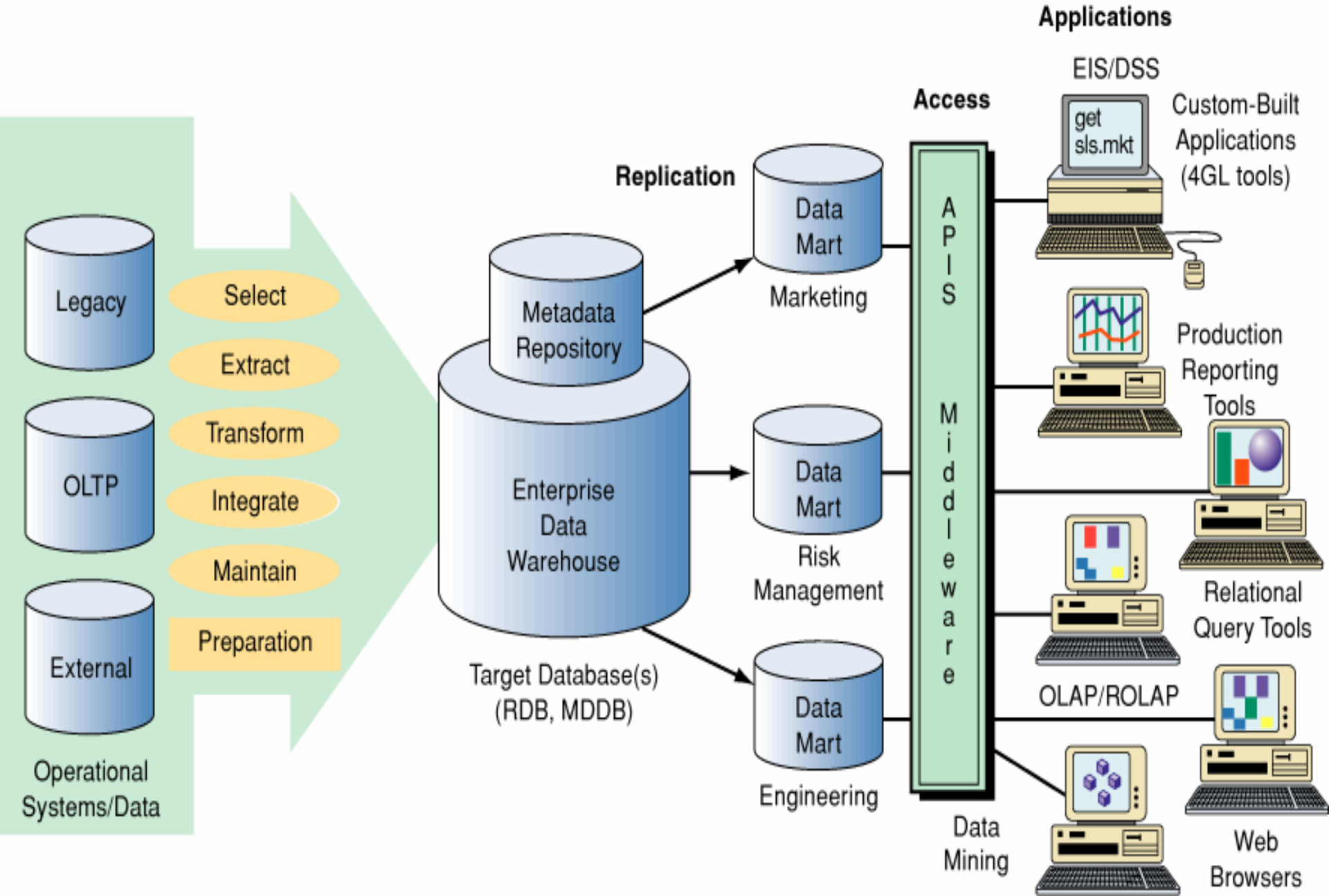
Xử lí phân tích

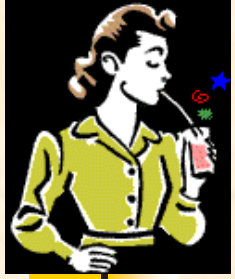
- OLAP cần đến ba khái niệm:
- **Thể hiện kinh doanh về dữ liệu người dùng**
- **Môi trường khách/ chủ cho phép người dùng hỏi về dữ liệu và ra báo cáo**
- **Thư mục trên máy tính chủ, kho dữ liệu... Cho phép an toàn dữ liệu và điều khiển tập trung dữ liệu**

Kho dữ liệu

- Định nghĩa:
  - Tập dữ liệu hướng chủ đề, tích hợp, thay đổi theo thời gian, không mất đi, để ra quyết định
- Xử lí phân tích trực tuyến (OLAP)
- Khai phá dữ liệu
- Tiếp thị cơ sở dữ liệu
- Khớp nhu cầu về dữ liệu và nhu cầu h thống thông tin điều hành

Kho dữ liệu


Applications
EIS/DSS
Access
get
sls.mkt
Custom-Built
Applications
(4GL tools)
Replication
Data
Mart
Marketing
Legacy
Select
Extract
Transform
Integrate
Maintain
Preparation
OLTP
External
Operational
Systems/Data
Metadata
Repository
Enterprise
Data
Warehouse
Target Database(s)
(RDB, MDDB)
Data
Mart
Risk
Management
Data
Mart
Engineering
API S
Middleware
Production
Reporting
Tools
Relational
Query Tools
OLAP/ROLAP
Data
Mining
Web
Browsers



Các tính chất

- Cơ sở dữ liệu tách biệt
- Dữ liệu sạch
- Thông tin tổng hợp để quản lí
- Dữ liệu meta
- Dữ liệu từ hệ thống hợp pháp
- Thời gian là yếu tố dữ liệu

Khai phá dữ liệu

- Kiểu dữ liệu
  - Liên kết
  - Xâu, chuỗi
  - Phân loại, tức các luật
  - Phân cụm, tức tạo nhóm
  - Dự báo, theo chuỗi thời gian
- Phát hiện giả thuyết, thay vì thử giả thuyết

Cám ơn sự theo dõi

THIẾT KẾ CƠ SỞ DỮ LIỆU

1. Cơ sở lý thuyết
2. Hệ thống quản lý thư viện
3. Hệ thống quản lý câu hỏi và đề thi
4. Hệ thống quản lý đào tạo theo phương thức tích luỹ tín chỉ

THIẾT KẾ CƠ SỞ DỮ LIỆU

1. Lựa chọn mô hình (Mô hình dữ liệu quan hệ)
2. Thiết kế bảng dữ liệu (danh sách các thuộc tính)
3. Thiết kế thuộc tính
4. Thiết kế bảng dữ liệu vật lý (tên bảng, khởi tạo bảng, lưu trữ bảng)

HỆ THỐNG QUẢN LÝ THƯ VIỆN

HỆ THỐNG QUẢN LÝ THƯ VIỆN

Quản lý bạn đọc

1. Sau khi nhập học, sinh viên được yêu cầu làm thẻ bạn đọc. Thông tin về bạn đọc - sinh viên được lưu trữ trong kho dữ liệu nhằm phục vụ cho công tác quản lý.
2. Sau mỗi năm học, ban quản lý xét gia hạn sử dụng thẻ bạn đọc, trên cơ sở xem xét việc sinh viên phải thực hiện đúng các quy định của thư viện là trả sách đúng hạn, nếu làm hỏng hoặc mất sách thì đền sách theo quy định. Trong trường hợp bạn đọc không yêu cầu gia hạn thì huỷ thẻ bạn đọc hoặc vi phạm quy định của thư viện sẽ bị phạt trong 1 năm không được xét cấp thẻ.
3. Việc tra cứu, tìm kiếm thông tin bạn đọc phục vụ cho ban quản lý, thủ thư và bạn đọc.

HỆ THỐNG QUẢN LÝ THƯ VIỆN

Quản lý sách

1. Hàng năm thư viện phải lập kế hoạch bổ sung sách mới, dựa trên yêu cầu của các khoa và thống kê những sách có nhiều bạn đọc yêu cầu. Sách mới bổ sung được thư viện được phân loại, ghi số cá biệt, từ khoá, tên sách, tên tác giả, nhà xuất bản, tóm tắt nội dung và được cập nhật vào kho thông tin.
2. Thống kê, tìm kiếm sách theo số cá biệt, tên sách hoặc theo tên tác giả, theo từ khoá để phục vụ bạn đọc mượn sách và thủ thư theo dõi sách.
3. Cập nhật thông tin về xử lý sách hỏng, sách cũ cần thanh lý, sách bị mất.
4. Hỗ trợ bạn đọc tra cứu, tìm kiếm thông tin về sách trước khi đến làm thủ tục mượn.

HỆ THỐNG QUẢN LÝ THƯ VIỆN

Theo dõi hoạt động mượn trả

Mượn sách:

1. **Bạn đọc điền vào phiếu mượn tên sách, số cá biệt, tác giả những quyển sách cần mượn, số thẻ, ngày mượn và nộp phiếu mượn cho thủ thư. Bạn đọc có thể đặt mượn qua mạng.**
2. **Sau khi thủ thư kiểm tra thông tin về bạn đọc (đúng chủ số thẻ, còn hạn sử dụng, số lượng sách đang mượn ít hơn số cho phép, không có sách mượn quá hạn) thì kiểm tra tiếp thông tin về sách mà bạn đọc yêu cầu. Nếu mọi yêu cầu đều được đáp ứng bạn đọc sẽ được thông báo đợi để mượn sách.**
3. **Tại kho, căn cứ vào phiếu mượn được thủ thư chuyển xuống, bộ phận quản lý kho giao sách cho bạn đọc và xác nhận thông tin mượn vào *Sổ theo dõi bạn đọc* và thay đổi trạng thái sách trong kho sách.**

HỆ THỐNG QUẢN LÝ THƯ VIỆN

Theo dõi hoạt động mượn trả

Trả sách:

Thủ thư ghi ngày trả, xác nhận vào *Sổ theo dõi bạn đọc* nếu sách không bị hư hại. Trong trường hợp sách bị hư hỏng hoặc mất thì làm thủ tục phạt hoặc đền sách.

Hàng tháng, thư viện xem xét những quyển sách bị mượn quá hạn và lập phiếu đòi sách quá hạn. Những quyển nào đã đòi quá ba lần thì được coi như đã mất.

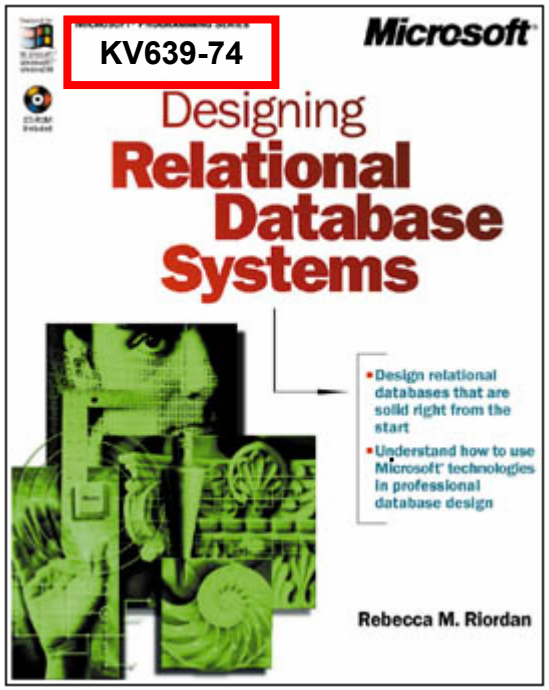
Thống kê, tìm kiếm phục vụ thủ thư và bạn đọc:

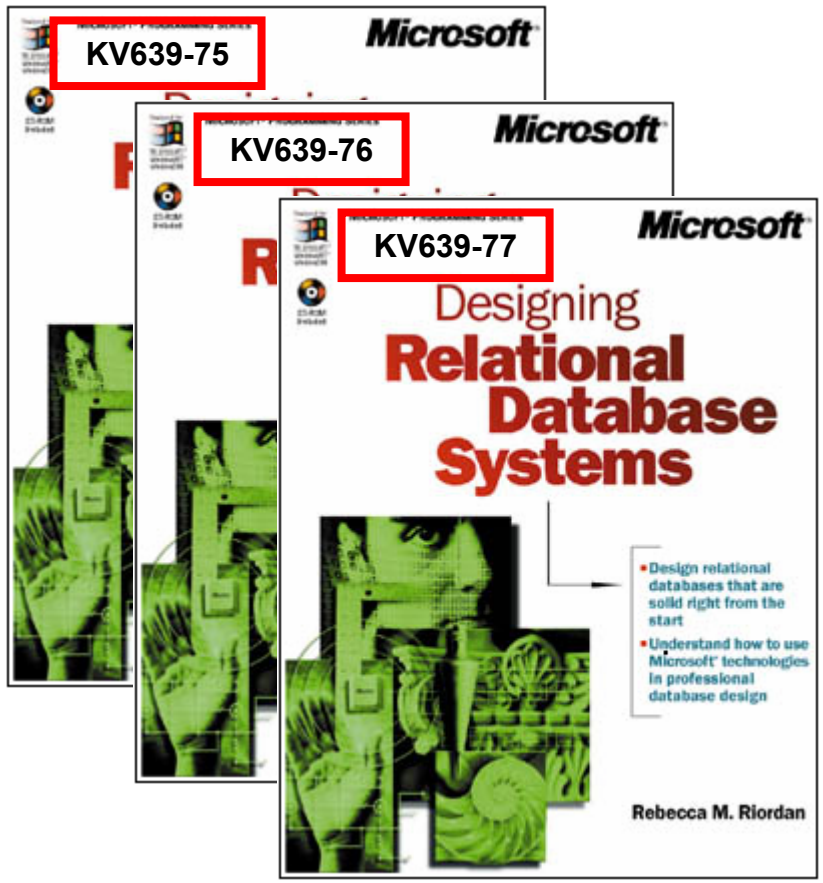
- Tìm bạn đọc mượn sách lần sau cùng hoặc đã từng mượn quyển sách;
- Tìm những sách quá hạn chưa trả và in phiếu báo đòi;
- Thống kê những sách có nhiều người mượn.

Thống kê những sách đang được mượn.

HỆ THỐNG QUẢN LÝ THƯ VIỆN

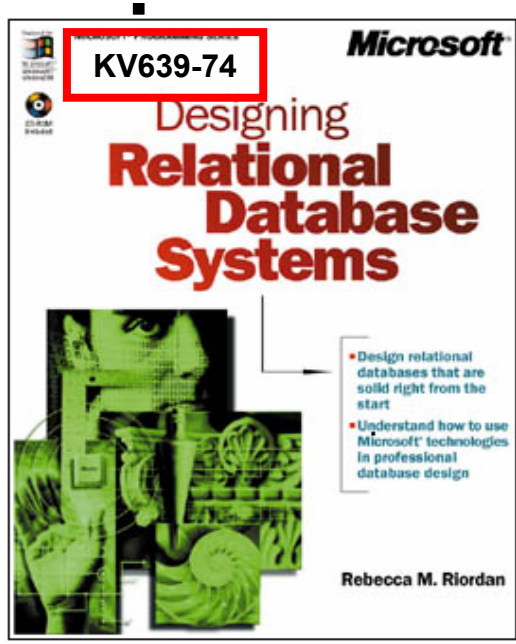
Quản lý sách - Mô hình 1





HỆ THỐNG QUẢN LÝ THƯ VIỆN

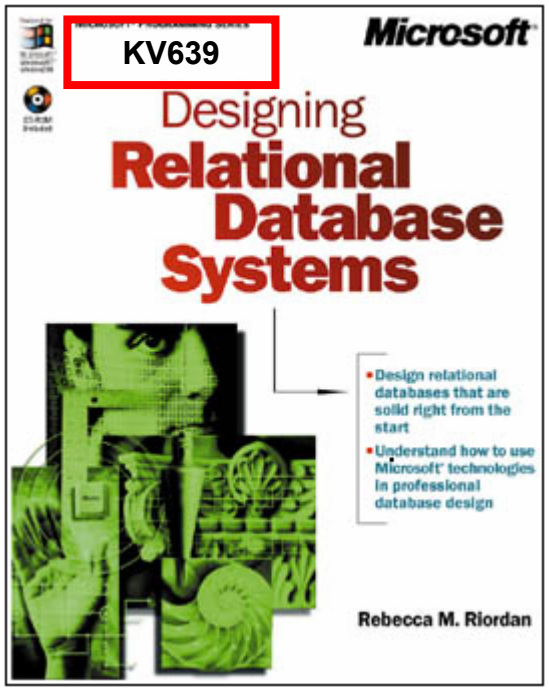
Quản lý sách - Mô hình 1

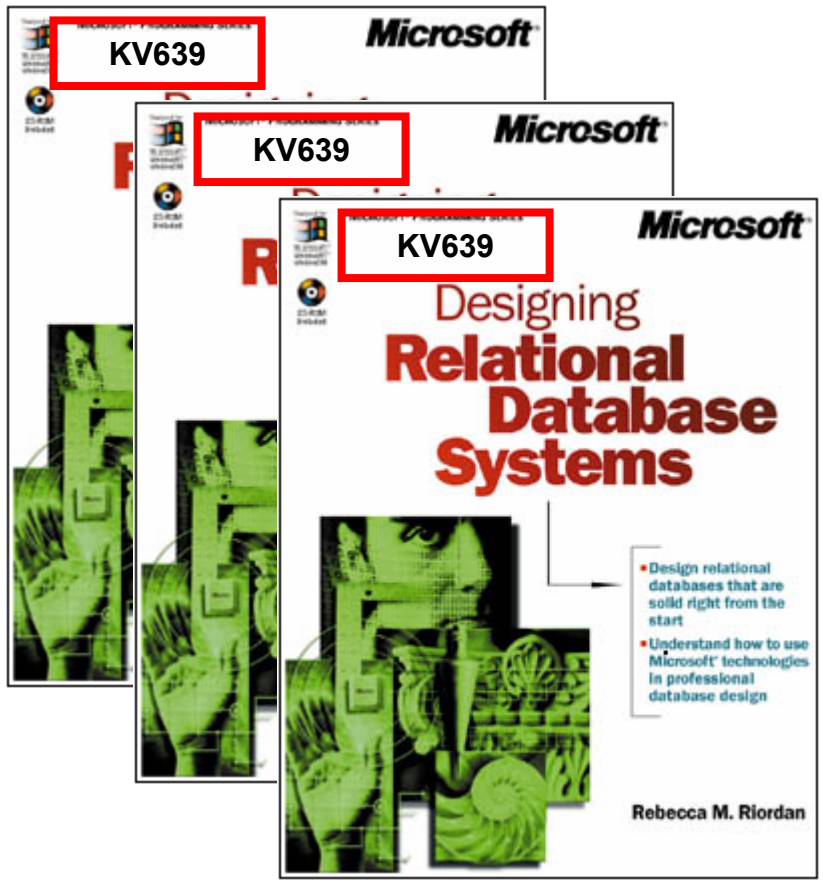


ID	Tên sách	Tác giả	Nhà xuất bản
KV639-74	Designing Relational Database Systems	Rebecca M. Riordan	Microsoft
KV639-75	Designing Relational Database Systems	Rebecca M. Riordan	Microsoft
KV639-76	Designing Relational Database Systems	Rebecca M. Riordan	Microsoft
KV639-77	Designing Relational Database Systems	Rebecca M. Riordan	Microsoft

HỆ THỐNG QUẢN LÝ THƯ VIỆN

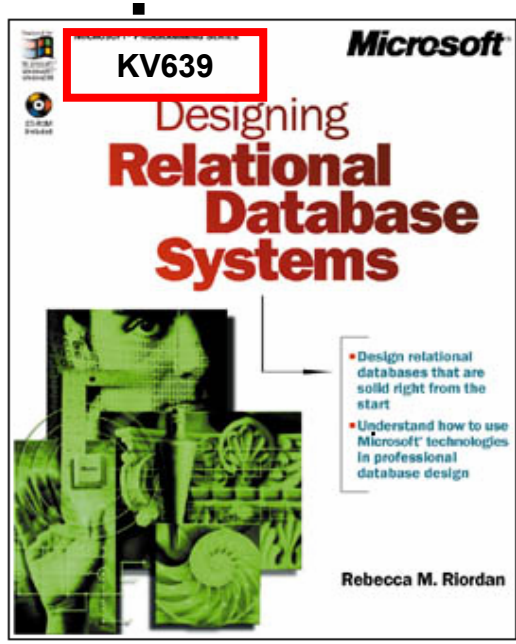
Quản lý sách - Mô hình 2





HỆ THỐNG QUẢN LÝ THƯ VIỆN

Quản lý sách - Mô hình 2



ID	Tên sách	Tác giả	Nhà xuất bản	Số lượng
KV639	Designing Relational Database Systems	Rebecca M. Riordan	Microsoft	10
KV640	Introduction to Relational Database Design	Fernando Lozano	Internet Edition	2
KV641	An Introduction to Database Systems	Date C.J.	Addison-Wesley Publishing Company	5
KV642	Systems Analysis and Design	Hawryszkiewyez I.T.	Univ. of Technology Sydney, Prentice Hall Astralia	1

HỆ THỐNG QUẢN LÝ THƯ VIỆN

Hoạt động mượn trả - Mô hình 1

ID	Tên sách	Tác giả	Nhà xuất bản
KV639-74	Designing Relational Database Systems	Rebecca M. Riordan	Microsoft
KV639-75	Designing Relational Database Systems	Rebecca M. Riordan	Microsoft
KV639-76	Designing Relational Database Systems	Rebecca M. Riordan	Microsoft
KV639-77	Designing Relational Database Systems	Rebecca M. Riordan	Microsoft

Bạn đọc tra cứu thông tin về sách và điền phiếu mượn

HỆ THỐNG QUẢN LÝ THƯ VIỆN

Hoạt động mượn trả - Mô hình 1

Nộp phiếu mượn và đợi thông tin phản hồi

Các tình huống không được mượn:

1. Số thẻ không đúng;
2. Thẻ hết hạn sử dụng;
3. Nợ sách quá hạn chưa trả;
4. Không còn sách trong kho;

HỆ THỐNG QUẢN LÝ THƯ VIỆN

Hoạt động mượn trả - Mô hình 1

Nộp phiếu mượn và đợi thông tin phản hồi

Nếu được mượn, bạn đọc làm thủ tục mượn tại kho

HỆ THỐNG QUẢN LÝ THƯ VIỆN

Hoạt động mượn trả - Mô hình 2

ID	Tên sách	Tác giả	Nhà xuất bản
KV639-74	Designing Relational Database Systems	Rebecca M. Riordan	Microsoft
KV639-75	Designing Relational Database Systems	Rebecca M. Riordan	Microsoft
KV639-76	Designing Relational Database Systems	Rebecca M. Riordan	Microsoft
KV639-77	Designing Relational Database Systems	Rebecca M. Riordan	Microsoft

Bạn đọc vào kho, tự tìm sách cần thiết

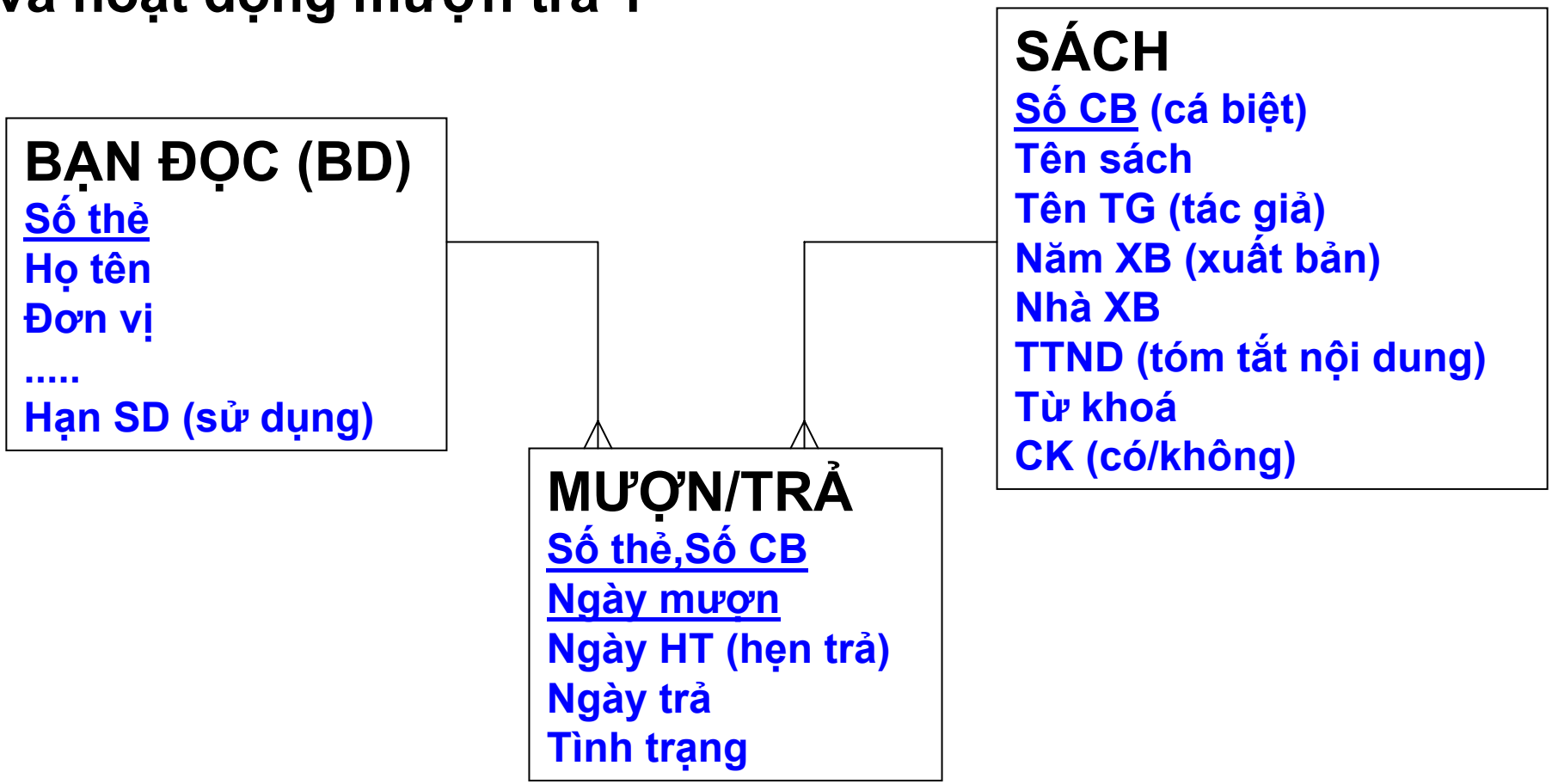
HỆ THỐNG QUẢN LÝ THƯ VIỆN

Hoạt động mượn trả - Mô hình 2

Sau đó làm
thủ tục mượn

HỆ THỐNG QUẢN LÝ THƯ VIỆN

Xét mô hình quản lý sách 1 và hoạt động mượn trả 1



HỆ THỐNG QUẢN LÝ THƯ VIỆN

Ví dụ 1 bản ghi SÁCH, BẠN ĐỌC và MƯỢN/TRẢ

Số CB	**KV639-74**
Tên sách	Designing Relational Database Systems
Tên TG	Rebecca M. Riordan
Năm XB	1999
Nhà XB	Microsoft
TTND	The book is divided into three sections. Part 1, Relational Database Theory, covers the fundamental principles of the relational model. This is where the really ugly, theoretical stuff is. But don't worry; it will get easier. Part 2, Designing Relational Database Systems, examines the analysis and design process—what you should do to get from the real world to a reliable database system design. Part 3 discusses the most important aspect of a database system from a user's point of view: the user interface.
Từ khoá	relational databases, database systems, database designing, relational model, design process, user interface, normal form
CK	1

Số thẻ	**12_321**
Họ tên	Hoàng Văn Trọng
Đơn vị	TH_7A
Hạn SD	31/12/2012

Số thẻ	**12_321**
Số CB	**KV639-74**
Ngày mượn	23/08/2010
Ngày HT	23/09/2010
Ngày trả	15/09/2010
Tình trạng	Tốt

HỆ THỐNG QUẢN LÝ THƯ VIỆN

BẠN ĐỌC (BD)
- Số thẻ
- Họ tên
- Đơn vị
- .....
- Hạn SD (sử dụng)

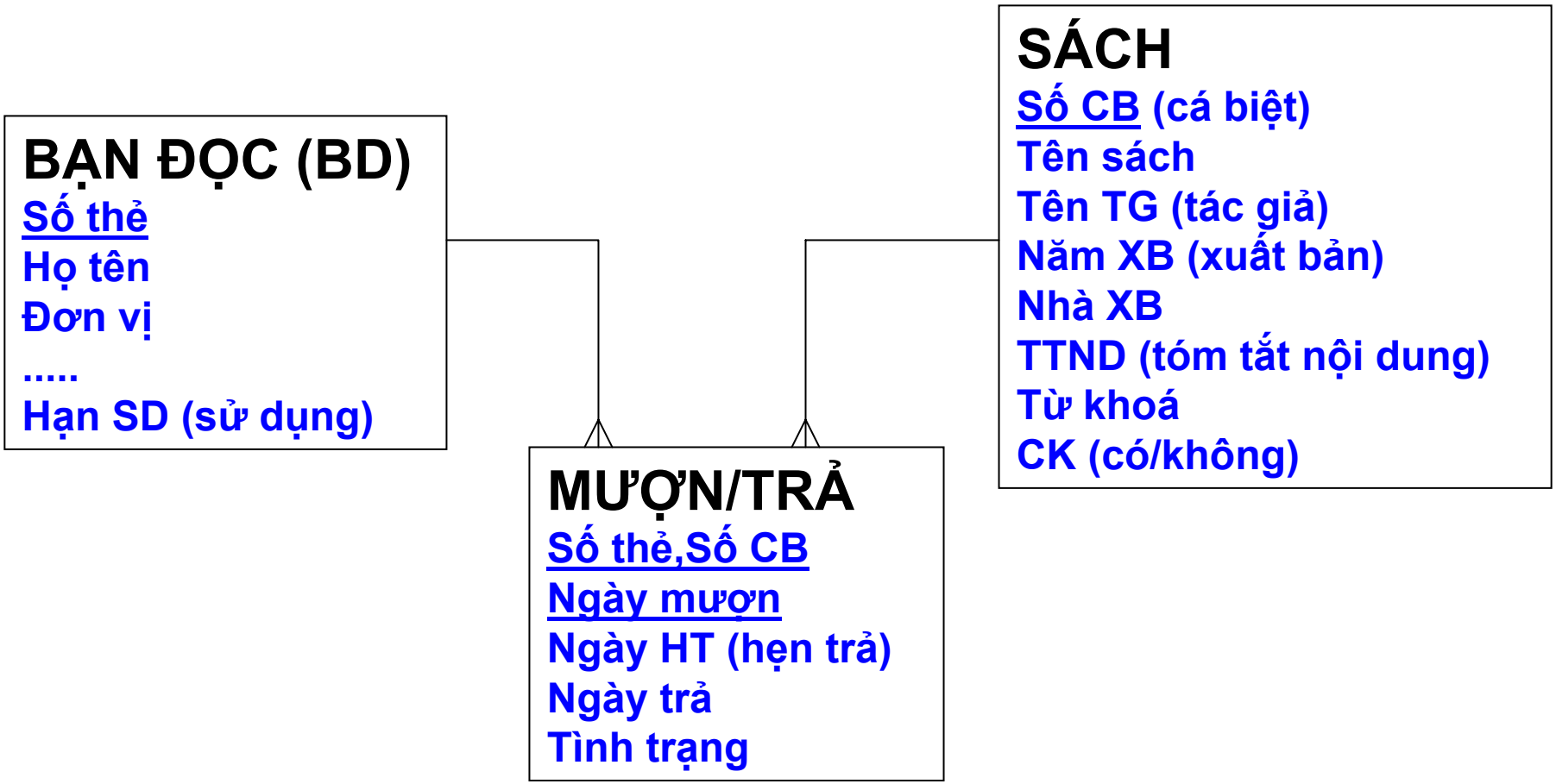
MƯỢN/TRẢ
- Số thẻ,Số CB
- Ngày mượn
- Ngày HT (hẹn trả)
- Ngày trả
- Tình trạng

SÁCH
- Số CB (cá biệt)
- Tên sách
- Tên TG (tác giả)
- Năm XB (xuất bản)
- Nhà XB
- TTND (tóm tắt nội dung)
- Từ khoá
- CK (có/không)

HỆ THỐNG QUẢN LÝ THƯ VIỆN

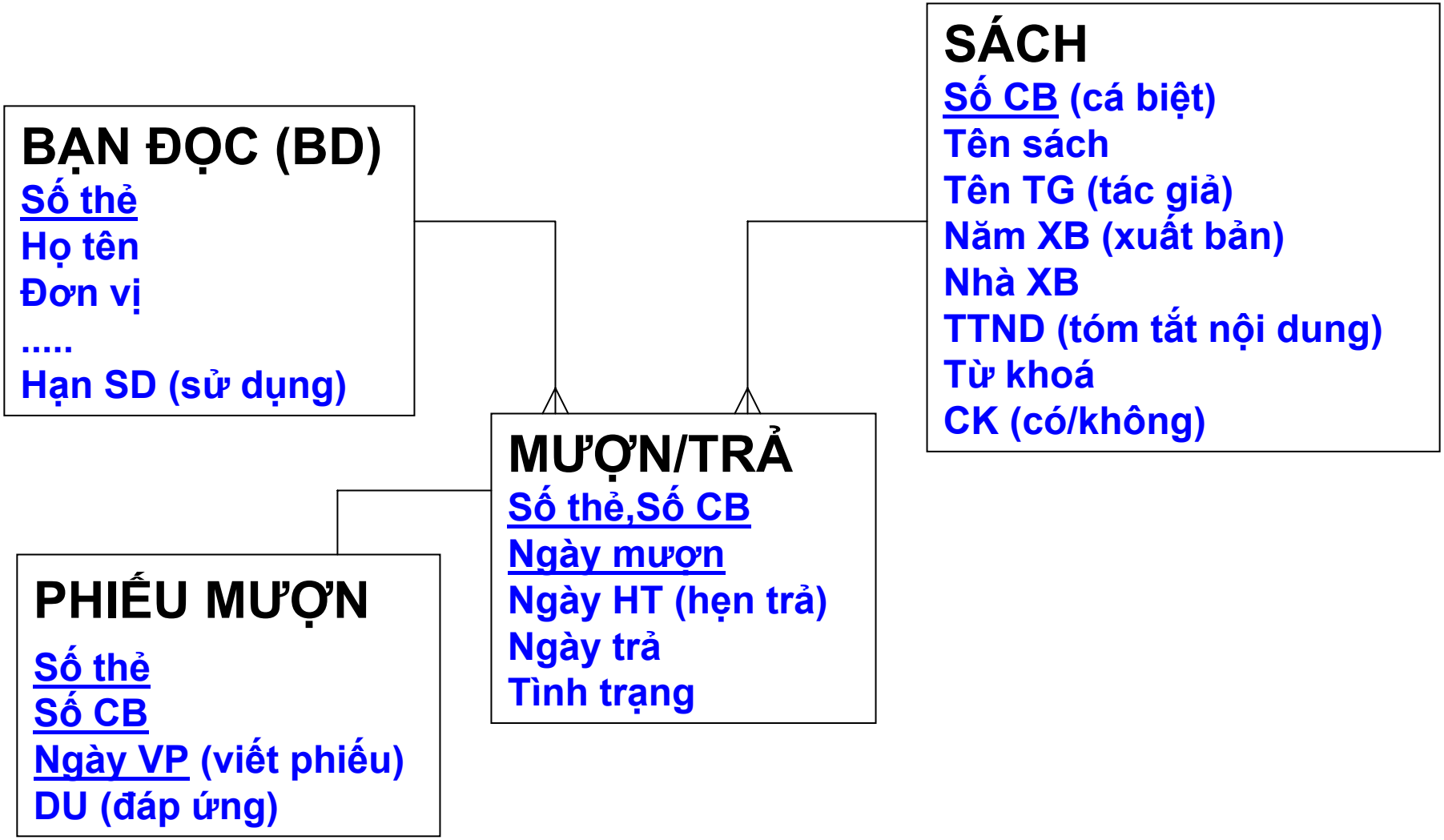
Đủ lưu trữ thông tin cho các tình huống xảy ra hay chưa?

HỆ THỐNG QUẢN LÝ THƯ VIỆN



Hệ thống quản lý: Cần có thông tin để thống kê yêu cầu mượn không được đáp ứng ⇒ Cần thông tin về PHIẾU MƯỢN.

HỆ THỐNG QUẢN LÝ THƯ VIỆN



HỆ THỐNG QUẢN LÝ THƯ VIỆN

BẠN ĐỌC (BD)
Số thẻ
Họ tên
Đơn vị
.....
Hạn SD (sử dụng)

SÁCH
Số CB (cá biệt)
Tên sách
Tên TG (tác giả)
Năm XB (xuất bản)
Nhà XB
TTND (tóm tắt nội dung)
Từ khoá
CK (có/không)

MƯỢN/TRẢ
Số thẻ,Số CB
Ngày mượn
Ngày HT (hẹn trả)
Ngày trả
Tình trạng

PHIẾU MƯỢN
Số thẻ
Số CB
Ngày VP
DU (đáp ứng)

Xét chi tiết từng bảng:

Đáp ứng các yêu cầu quản lý?

HỆ THỐNG QUẢN LÝ THƯ VIỆN

SÁCH
Số CB (cá biệt)
Tên sách
Tên TG (tác giả)
Năm XB (xuất bản)
Nhà XB
TTND (tóm tắt nội dung)
Từ khoá
CK (có/không)

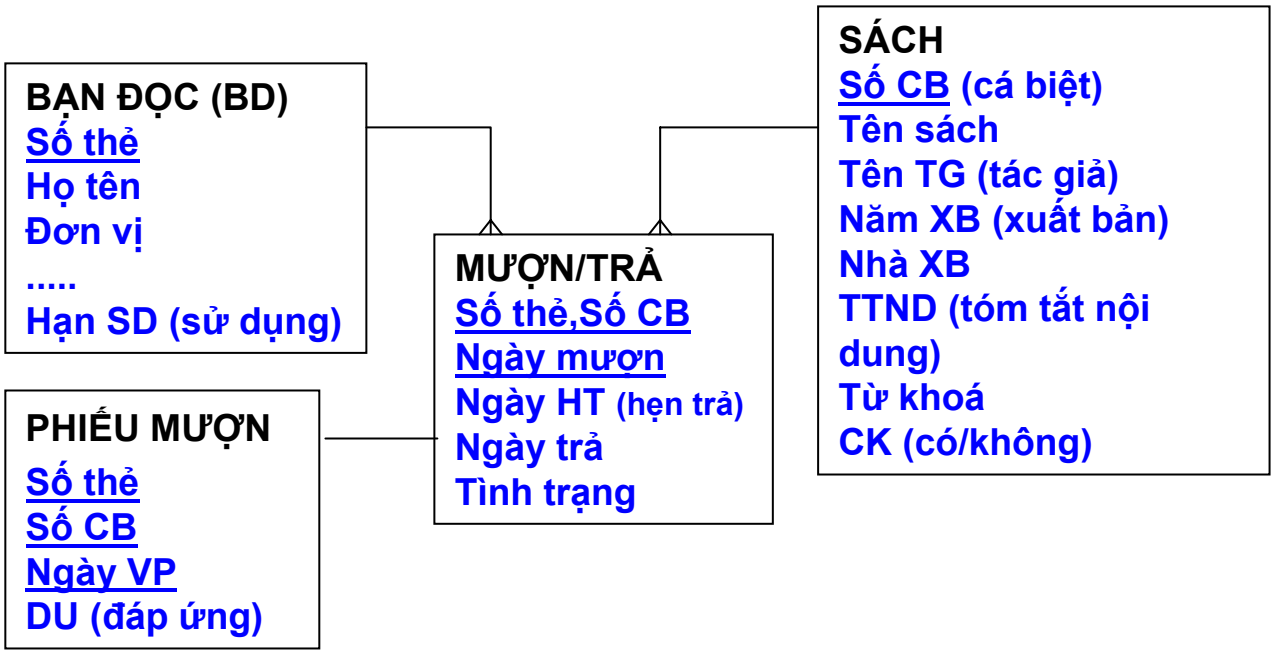
BẠN ĐỌC (BD)
Số thẻ
Họ tên
Đơn vị
.....
Hạn SD (sử dụng)

1) Bổ sung bạn đọc mới
2) Xét gia hạn sử dụng

MƯỢN/TRẢ
Số thẻ,Số CB
Ngày mượn
Ngày HT (hẹn trả)
Ngày trả
Tình trạng

PHIẾU MƯỢN
Số thẻ
Số CB
Ngày VP
DU (đáp ứng)

HỆ THỐNG QUẢN LÝ THƯ VIỆN



Xác định nhu cầu mua sắm:

Input: Số CB_Xét

select on PHIẾU MƯỢN for (Ngày mượn ∈Khoảng TG) & (DU=0) & (Số CB= Số CB_Xét)

HỆ THỐNG QUẢN LÝ THƯ VIỆN

BẠN ĐỌC (BD)
Số thẻ
Họ tên
Đơn vị
.....
Hạn SD (sử dụng)

PHIẾU MƯỢN (PM)
Số thẻ
Số CB
NgàyVP(viết phiếu)
DU (đáp ứng)

MƯỢN/TRẢ (MT)
Số thẻ,Số CB
Ngày mượn
Ngày HT (hẹn trả)
Ngày trả
Tình trạng

SÁCH (S)
Số CB (cá biệt)
Tên sách
Tên TG (tác giả)
Năm XB (xuất bản)
Nhà XB
TTND (tóm tắt nội dung)
Từ khoá
CK (có/không)

Giao sách

input Số thẻ, Số CB
if (Giao nhận)
    create record in MT (Ngày trả={})
    S.CK = 0

Thủ tục mượn sách:

1) input Số thẻ;
2) if !(Số thẻ là hợp lệ)
    {found BD.Số thẻ = Số thẻ;
     BD.Hạn SD< Ngày hiện thời
     số sách nợ quá hạn = 0}
    message M1; goto 1);
3) input Số CB
4) if !found(Số CB, SÁCH)
    message M2; goto 1);
5) Create record in PM (DU=0)
6) if (S.CK=0)
    message M3;
    goto 1);
7) PM.DU =1
8) goto 1)

HỆ THỐNG QUẢN LÝ THƯ VIỆN

BẠN ĐỌC (BD)
- Số thẻ
- Họ tên
- Đơn vị
- .....
- Hạn SD (sử dụng)

PHIẾU MƯỢN (PM)
- Số thẻ
- Số CB
- NgàyVP(viết phiếu)
- DU (đáp ứng)

MƯỢN/TRẢ (MT)
- Số thẻ,Số CB
- Ngày mượn
- Ngày HT (hẹn trả)
- Ngày trả
- Tình trạng

SÁCH (S)
- Số CB (cá biệt)
- Tên sách
- Tên TG (tác giả)
- Năm XB (xuất bản)
- Nhà XB
- TTND (tóm tắt nội dung)
- Từ khoá
- CK (có/không)

Tình huống:

Bạn đọc A làm thủ tục mượn quyển X, được mượn, và làm thủ tục giao nhận

Bạn đọc B có yêu cầu mượn quyển X?

B

PM: B, X, 20120207, ?

A

PM: A, X, 20120207, 1

KHO

X

THIẾT KẾ CƠ SỞ DỮ LIỆU

GVHD: ThS. Trần Sơn Hải

http://fit.hcmup.edu.vn/~haits

Email: transonhai@hcmup.edu.vn

Nội dung

Chương 1 : Tổng quan các hệ thông tin và cơ sở dữ liệu

Chương 2 : Hệ quản trị cơ sở dữ liệu SQL Server

Chương 3 : Tổng quan quá trình thiết kế CSDL

Chương 4 : Lý thuyết mô hình CSDL quan hệ

Chương 5 : Thiết kế CSDL ở mức quan niệm

Chương 6 : Thiết kế CSDL ở mức lô-gíc

Chương 7 : Thiết kế CSDL ở mức vật lý

Báo Cáo và Ôn Tập

1. Tài Liệu Tham Khảo

Thomas M. Connolly, Carolyn E. Begg, A Practical Approach to Design, Implementation, and Management, Addion-Wesley, 1999.

J. Ullman, Database and knowledge base systems,Vol 1. Computer Science Press, 1988

Ramez Elmasri, Shamkant B. Navathe , Fundamentals of database systems, Addison Wesley - 4th edition, 2003

Joseph M. Hellerstein, Michael Stonebraker, Readings in Database Systems. Fourth Edition, MIT Press, 2005

❖ Chương 1 : Tổng quan các hệ thông tin và cơ sở dữ liệu

3. Cơ sở dữ liệu (1)

- Định nghĩa:

  Cơ sở dữ liệu là một hệ thống các thông tin có cấu trúc, được lưu trữ trên các thiết bị lưu trữ nhằm thỏa mãn yêu cầu khai thác thông tin đồng thời của nhiều người sử dụng hay nhiều chương trình ứng dụng với những mục đích khác nhau

3. Cơ sở dữ liệu (2)

- Ưu điểm:
  - Giảm trùng lắp thông tin xuống mức thấp nhất, đảm bảo tính nhất quán và toàn vẹn dữ liệu.
  - Đảm bảo dữ liệu được truy xuất theo nhiều cách khác nhau.
  - Khả năng chia sẽ thông tin cho nhiều người, nhiều ứng dụng khác nhau.

3. Cơ sở dữ liệu

- **Những vấn đề cần giải quyết:**
  - Tính chủ quyền dữ liệu.
  - Tính bảo mật và quyền khai thác thông tin của người sử dụng
  - Tranh chấp dữ liệu
  - Đảm bảo dữ liệu khi có sự cố

5. Hệ quản trị cơ sở dữ liệu (1)

- **Hệ quản trị cơ sở dữ liệu (DBMS – DataBase Management System) là hệ thống các phần mềm hỗ trợ tích cực cho các nhà phân tích, thiết kế và khai thác CSDL.**
- **Các DBMS thông dụng: Visual FoxPro, Microsoft Access, SQL Server, DB2, Oracle … hầu hết các DBMS hiện nay đều dựa trên mô hình quan hệ.**

5. Hệ quản trị cơ sở dữ liệu (2)

- **Một DBMS phải có:**
  - Ngôn ngữ giao tiếp giữa người sử dụng và CSDL
  - Từ điển dữ liệu (Data Dictionary)
  - Có biện pháp bảo mật khi có yêu cầu
  - Cơ chế giải quyết tranh chấp dữ liệu
  - Có cơ chế sao lưu (backup), phục hồi (restore)
  - Đảm bảo tính độc lập giữa dữ liệu và chương trình

5. Hệ quản trị cơ sở dữ liệu (3)

Ngôn ngữ giao tiếp:

- **Ngôn ngữ mô tả dữ liệu (*DDL – Data Definition Language*): cho phép khai báo cấu trúc CSDL, các mối liên hệ của dữ liệu, các quy định, ràng buộc dữ liệu.**
- **Ngôn ngữ thao tác dữ liệu (*DML – Data Manipulation Language*): cho phép thực hiện thao tác thêm, xóa, sửa dữ liệu.**
- **Ngôn ngữ truy vấn có cấu trúc (SQL – *Structured Query Language*): cho phép người khai thác sử dụng để truy vấn thông tin cần thiết.**
- **Ngôn ngữ quản lý dữ liệu (DCL – *Data Control Language*) cho phép thay đổi cấu trúc bảng, khai báo bảo mật, cấp quyền cho người sử dụng.**

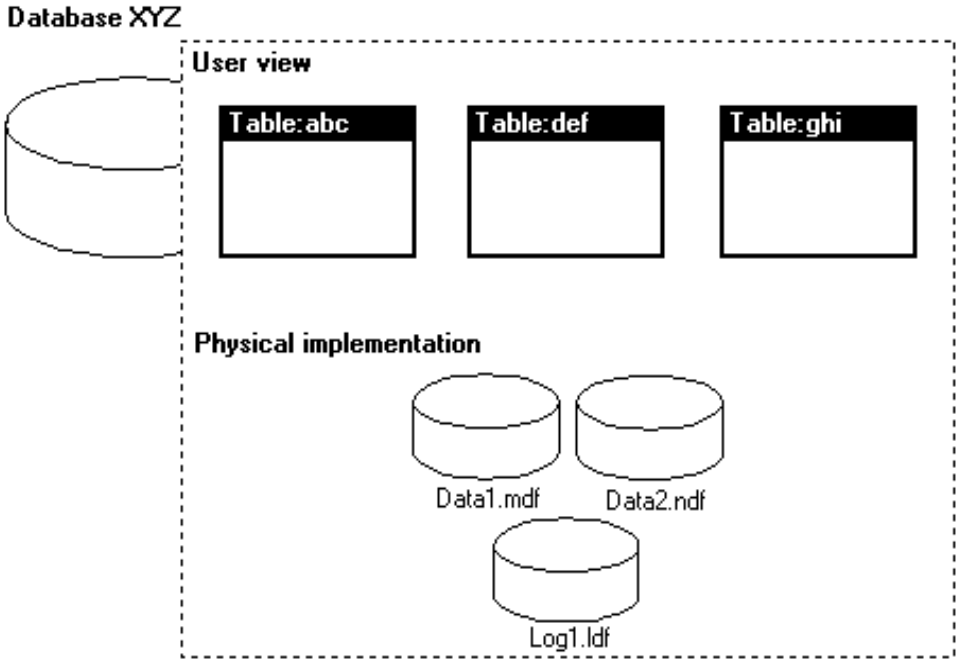
Các Thành Phần Liên Quan

- **Bộ phận quản lý tập tin**
  - .mdf : meta data file
  - .ldf : log data file
  - .bak : bakup data file
- **Bộ phận quản lý đĩa**
- **Bộ phận quản lý dữ liệu vật lý**

Tổ Chức Vật Lý trên SQL Server

- **Dữ liệu trong CSDL được tổ chức thành các thành phần (Component) logic cho user sử dụng như: Table, View….**
- **Dữ liệu vật lý có thể lưu trên nhiều file hay thậm chí nhiều ổ đĩa.**
- **Người dùng (trừ các DBA) chỉ làm việc trên các thành phần logic của SQL Server.**

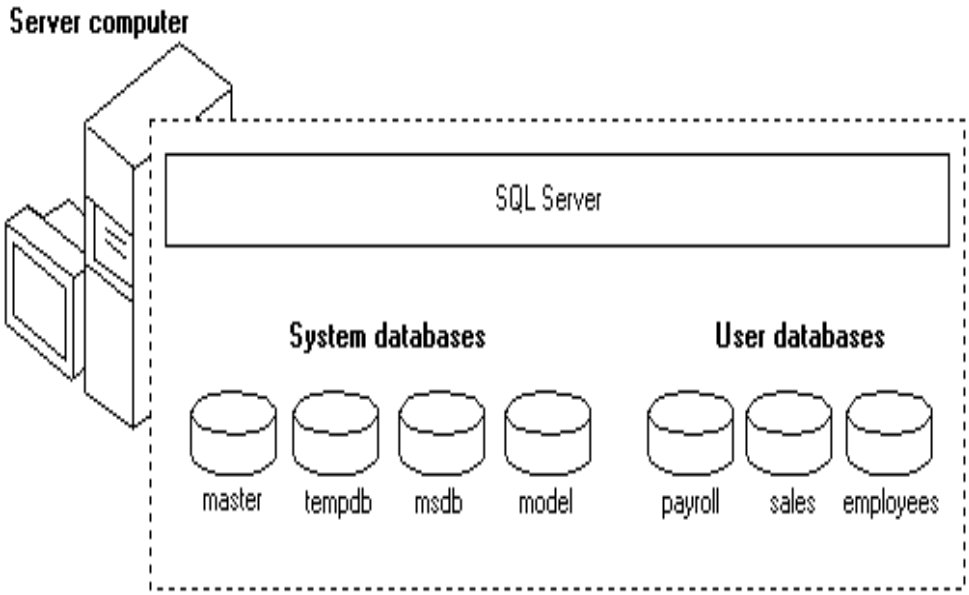
Kiến trúc của SQL Server



Kiến trúc của SQL Server

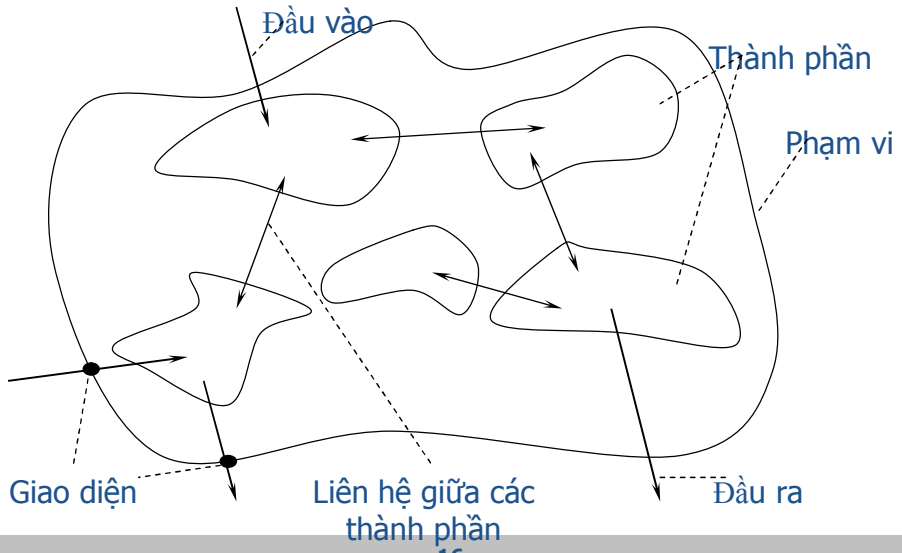
- Các database hệ thống của SQL Server
  - master Database chứa thông tin về cấu hình, phần quyền của hệ thống
  - tempdb Database chứa dữ liệu tạm cần lưu trữ
  - model Database chứa database mẫu để tạo một database mới
  - msdb Database để hỗ trợ SQL Server Agent

Kiến trúc của SQL Server



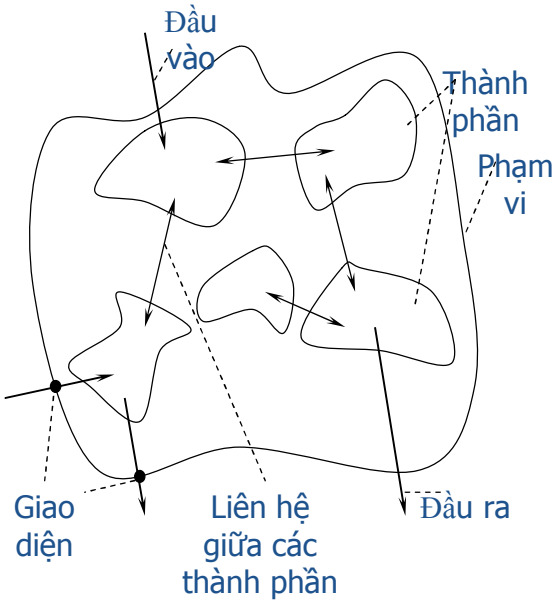
Hệ thống

- **Một tập hợp các thành phần liên kết với nhau, thể hiện qua một phạm vi (boundary) xác định, hoạt động kết hợp với nhau nhằm đạt đến những mục đích xác định.**



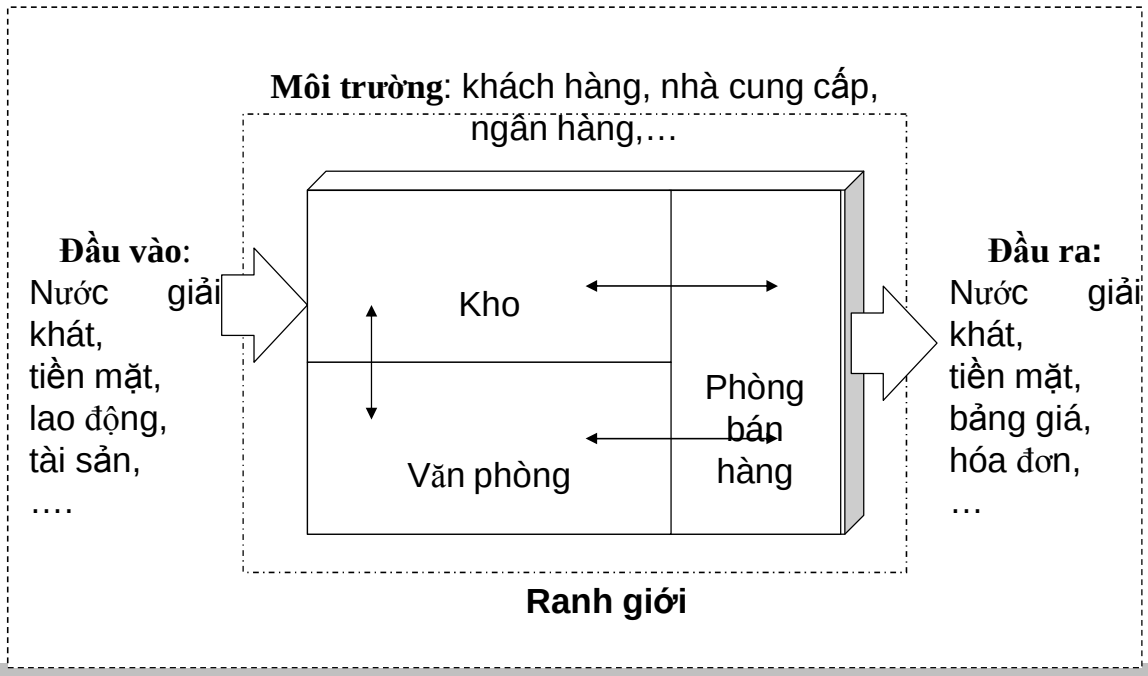
Các đặc điểm của hệ thống

- Thành phần (component)
- Liên hệ giữa các thành phần
- Ranh giới (boundary)
- Mục đích (purpose)
- Môi trường (environment)
- Giao diện (interface)
- Đầu vào (input)
- Đầu ra (output)
- Ràng buộc (constraints)t

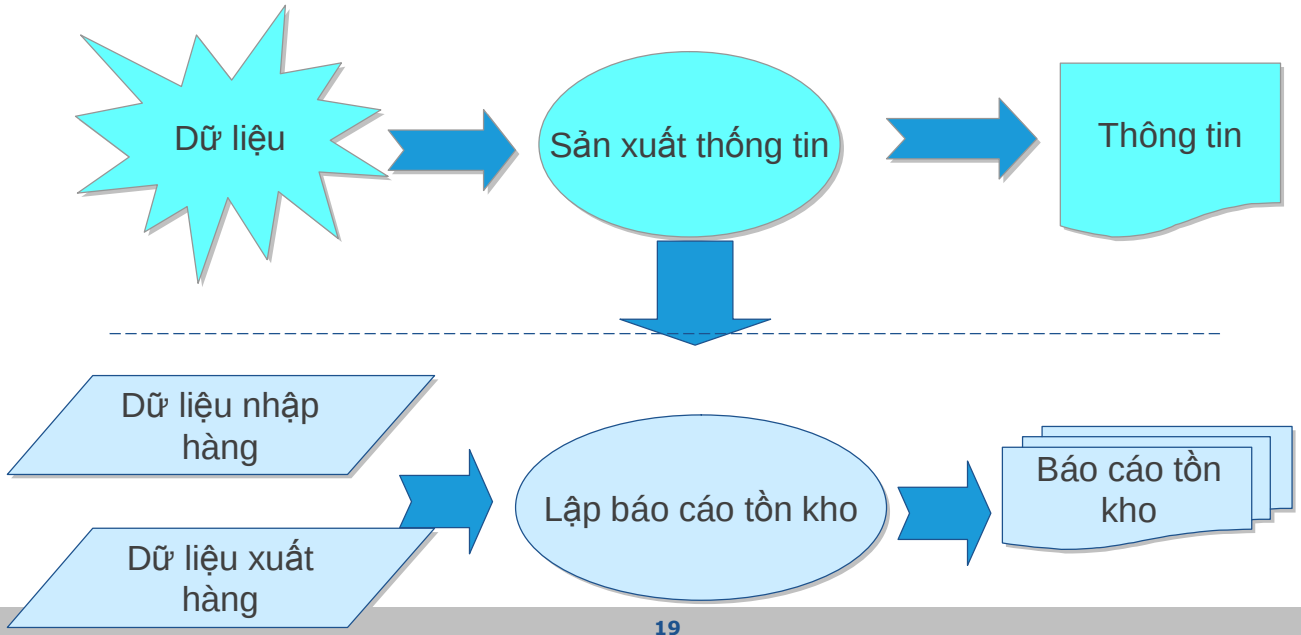


Hệ thống – Ví dụ

Xem cửa hàng bán nước giải khát như một hệ thống:



- **Thông tin: khái niệm phản ánh sự hiểu biết của con người về một đối tượng**



- **Hệ thống gắn liền với một tổ chức cụ thể nhằm điều hành tổ chức một cách tốt nhất. Hai yếu tố quan tâm:**
  - ***Tổ chức***: cơ quan hành chính, trường học, doanh nghiệp,…
  - ***Môi trường của tổ chức***:
    - Các cơ quan hành chính khác mà cơ quan cần làm việc chung hoặc hệ thống luật phải tuân theo.
    - Tổ chức kinh tế: ngân hàng, khách hàng, nhà cung ứng, hệ thống luật kinh doanh,…

Câu hỏi thảo luận

- **Hệ thống thông tin là gì?**
- **Vai trò của CSDL trong hệ thống thông tin.**
- **Phân biệt dữ liệu, thông tin và tri thức?**
- **Đề xuất một quy trình thiết kế cơ sở dữ liệu.**

Tóm lại

- **Khái niệm cơ sở dữ liệu**
- **Mục tiêu của các hệ CSDL**
- **Kiến trúc và chức năng của một hệ CSDL**
- **Vai trò cơ sở dữ liệu trong các hệ thống thông tin**

CẢM ƠN

Chương 2 : Hệ quản trị cơ sở dữ liệu SQL Server

I. Tổng quan :

1. SQL SerVer là một hệ quản trị CSDL quan hệ

A. Cơ Sở Dữ Liệu Quan Hệ :

- Dữ liệu được tổ chức lưu trữ thành từng bảng
- Một bảng dùng để lưu trữ thông tin của một đối tượng ngoài thế giới thực
- Một bảng được tạo từ dòng và cột

Tổng Quan : (tt)

- Cột :Dùng để lưu trữ thuộc tính của đối tượng
- Một bảng phải có tối thiểu một cột,Thứ tự cột khơng quan trọng
- Một cột có ba thuộc tính cơ bản
  - ✓ Column Name :Tênn cột,trong một bảng không thể có hai cột trùng tên
  - ✓ Data Type :Kiểu dữ liệu
  - ✓ Length :chiều dài của dữ liệu

Tổng Quan(tt)

- **Dòng :tập hợp dữ liệu của nhiều cột**
- **Trong một bảng không thể có hai dòng trùng thông tin**
- **Một dòng cho biết thông tin cụ thể của một đối tượng lưu trữ trong bảng**
- **Một bảng có thể không có dòng nào**

Tổng Quan(tt)

- **Khoá Chính (Primary Key) :**
  - ✓ **Tập hợp của một hay nhiều cột trong bảng**
  - ✓ **Dữ liệu tại cột này khác rỗng**
  - ✓ **Dữ liệu tại cột này là duy nhất trong bảng**

Tổng Quan (tt)

- **Khoá ngoại (Foreign Key) :**
  - ✓ **Tập hợp của một hay nhiều cột trong bảng,mà các cột này là khoá chính của bảng khác**
  - ✓ **Dữ liệu tại các cột này hoặc rỗng,hoặc phải lấy từ bảng mà cột làm khoá chính**

Tổng Quan(tt)

B.Hệ Quản Trị CSDL Quan Hệ :

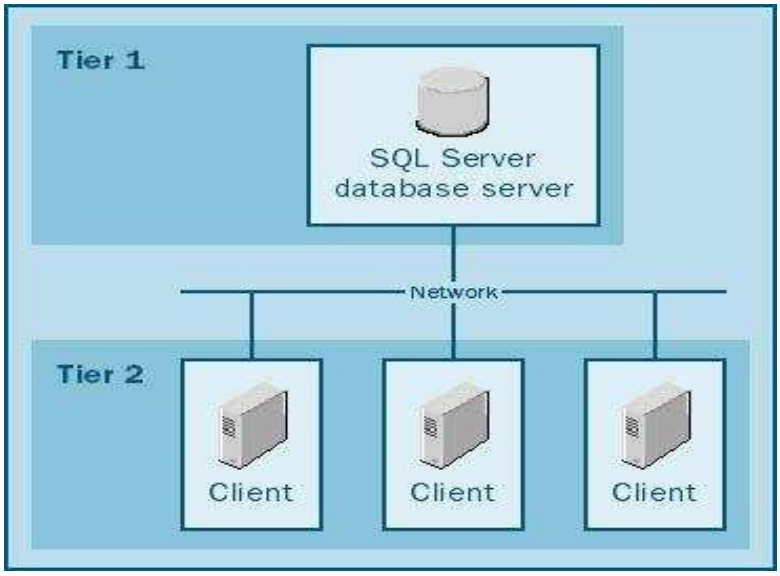
- Là Là một hệ thống các chương trình có các chức năng
  - ✓ **Tạo và quản lý một CSDL Quan hệ**
  - ✓ **khai thác Một CSDL Quan hệ**
  - ✓ **Bảo mật Một CSDL Quan hệ**
  - ✓ **Tạo bảng sao (Backup) Một CSDL Quan hệ**
  - ✓ **Phục hồI(Restore) Một CSDL Quan hệ**
- **SQL Server là một hệ quản trị CSDL Quan hệ**

Các hệ thống SQL Server

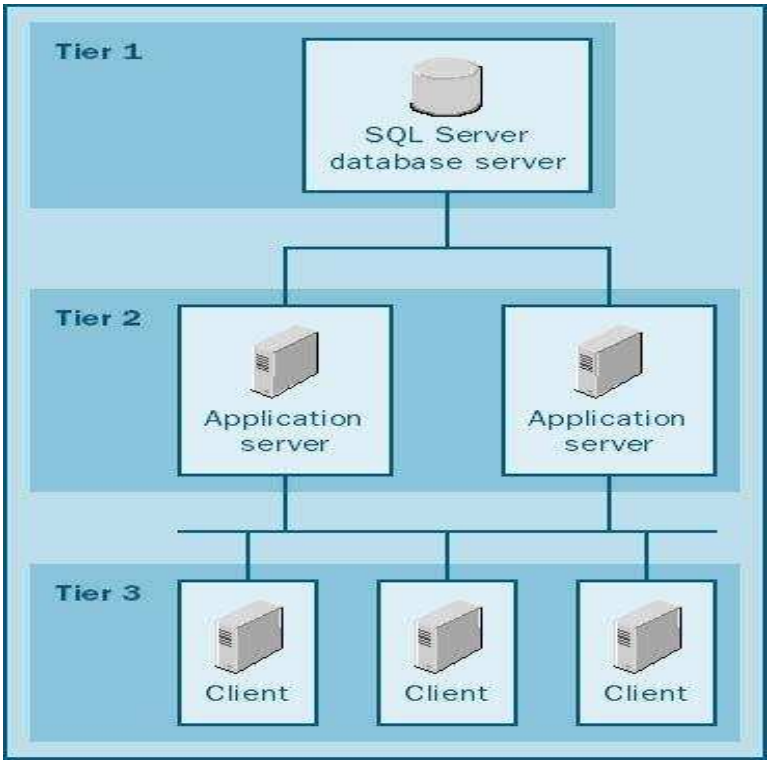
- **Hệ Thống SQL Server có thể triển khai cho một hệ thống kiểu Client-Server hoặc triển khai cho một hệ thống máy đơn**
- **Hệ thống được triển khai phụ thuộc vào số người truy cập đồng thời đến CSDL để thực hiện một công việc**

Các hệ thống SQL Server

- **Trong mô hình Client-Server ta có thể triển khai theo mô hình 2 lớp hoặc 3 lớp**



Các hệ thống SQL Server



Cài Đặt SQL Server

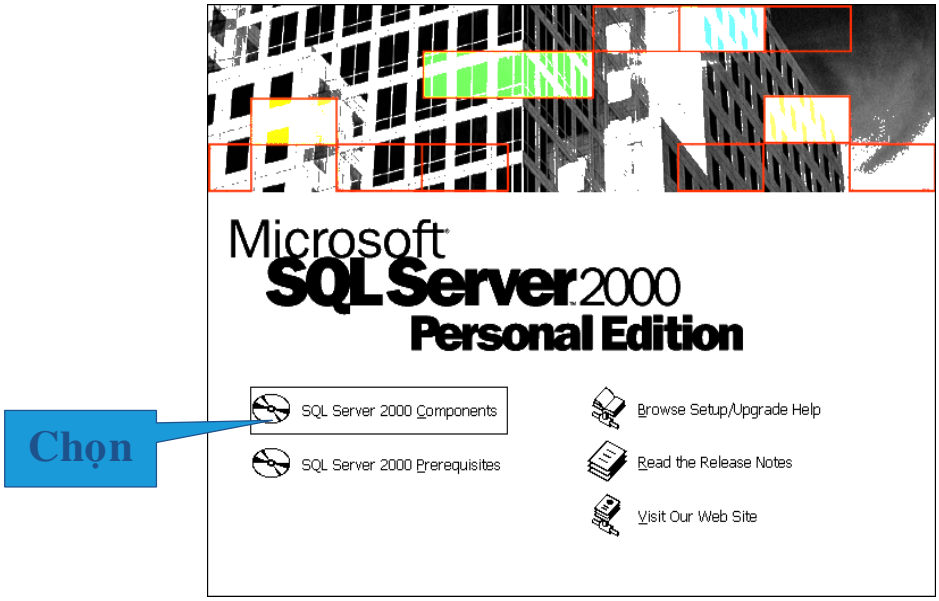
A.Yêu cầu phần cứng :

Phần cứng	Yêu cầu
CPU	Từ Pentium 166 MHz trở lên
RAM	Từ 64MB trở lên
Đĩa cứng	Còn trống 100MB trở lên
Màn hình	VGA có độ phân giải 800X600 trở lên

A.Yêu cầu phần Mềm:

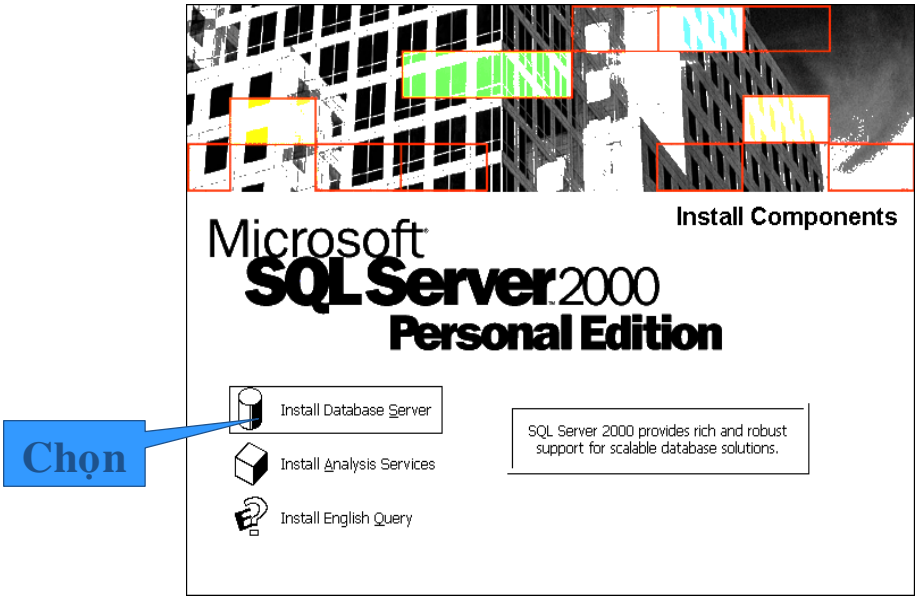
Phĩn bản SQL Server	Hệ điều Hình
➢Enterprise Edition ➢Standard Edition	Microsoft Windows Server(nt,2000,Xp Server)
➢Personal Edition	Có thể chạy trên tất cả các hệ điều hành

1. Đưa đĩa CD chương trình vào ,màn hình sau xuất hiện



Cài đặt Personal Edition SQL Server(tt)

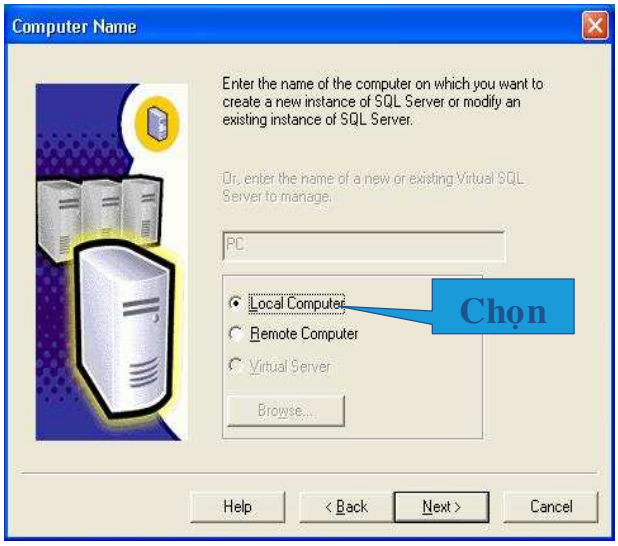
2.Màn hình sau xuất hiện



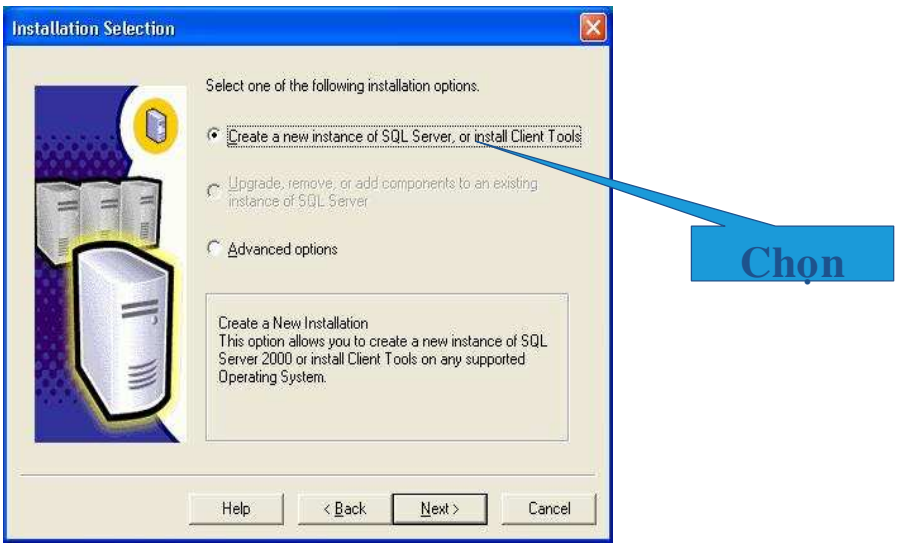
Cài đặt Personal Edition SQL Server(tt)

3.Màn hình sau xuất hiện

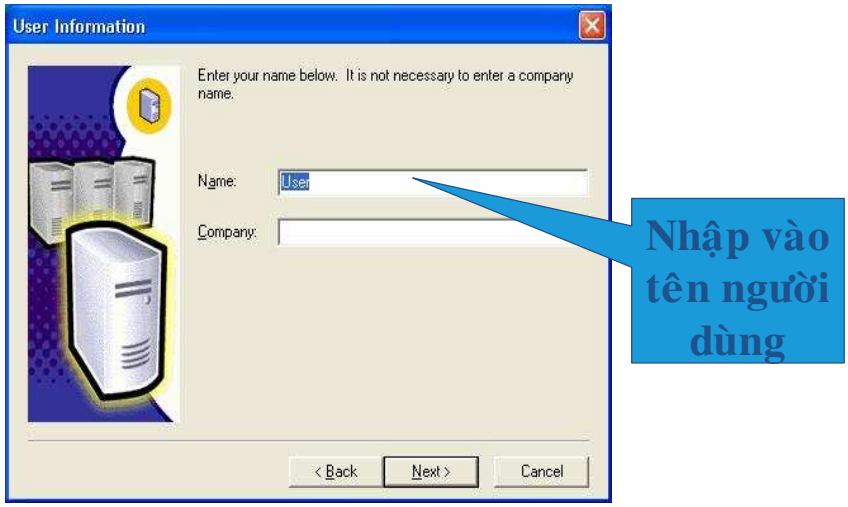
4.Màn hình sau xuất hiện



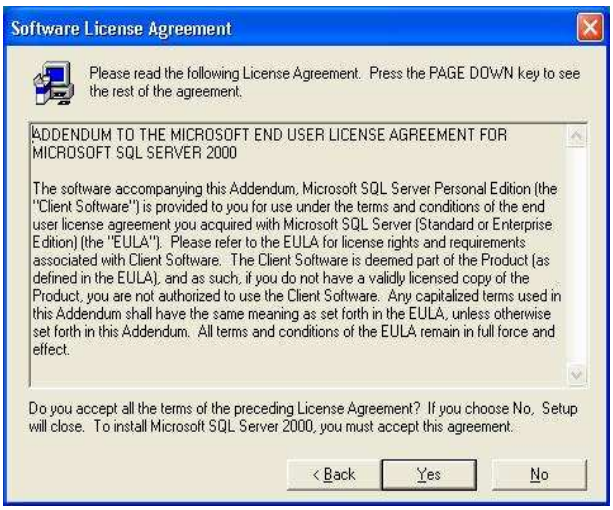
5.Màn hình sau xuất hiện



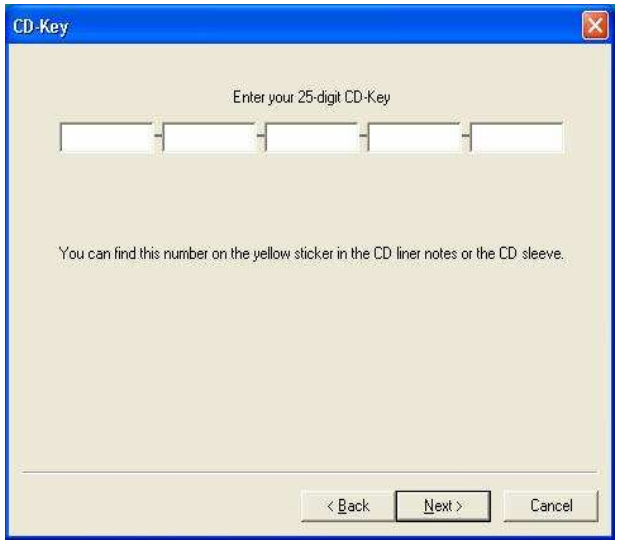
6.Màn hình sau xuất hiện



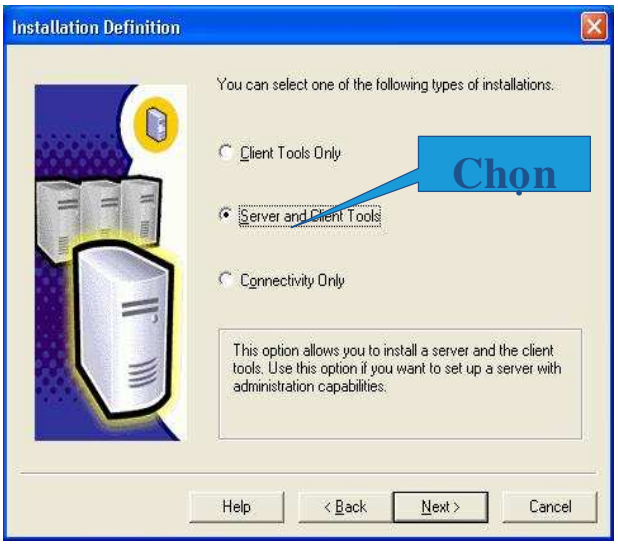
7.Màn hình cảnh báo xuất hiện,chọn yes



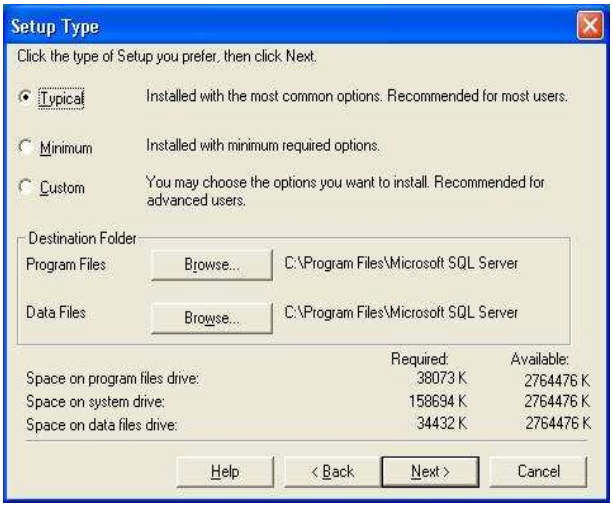
8.Màn hình đăng nhập xuất hiện ,nhập vào CD Key



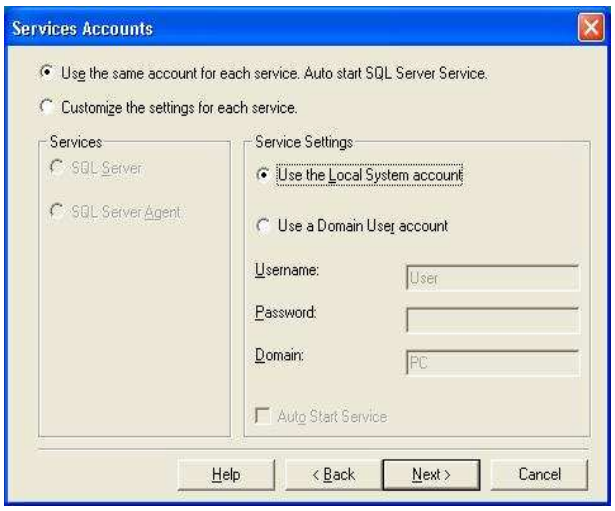
9.Chọn kiểu cài đặt



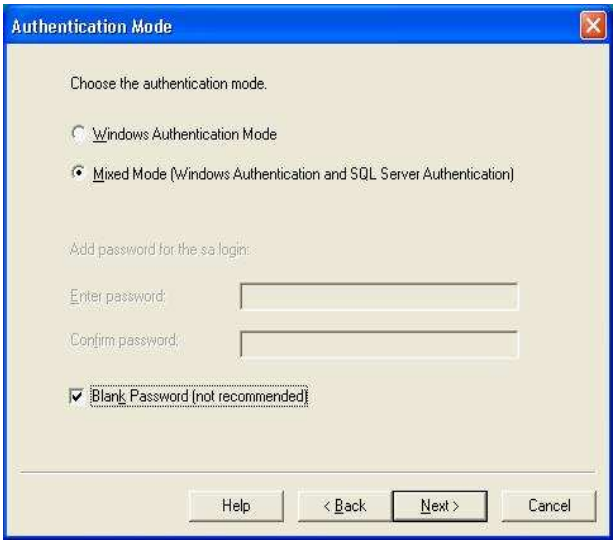
10.Chọn Dạng cài đặt



11.Chọn kiểu đăng nhập vào SQL Server

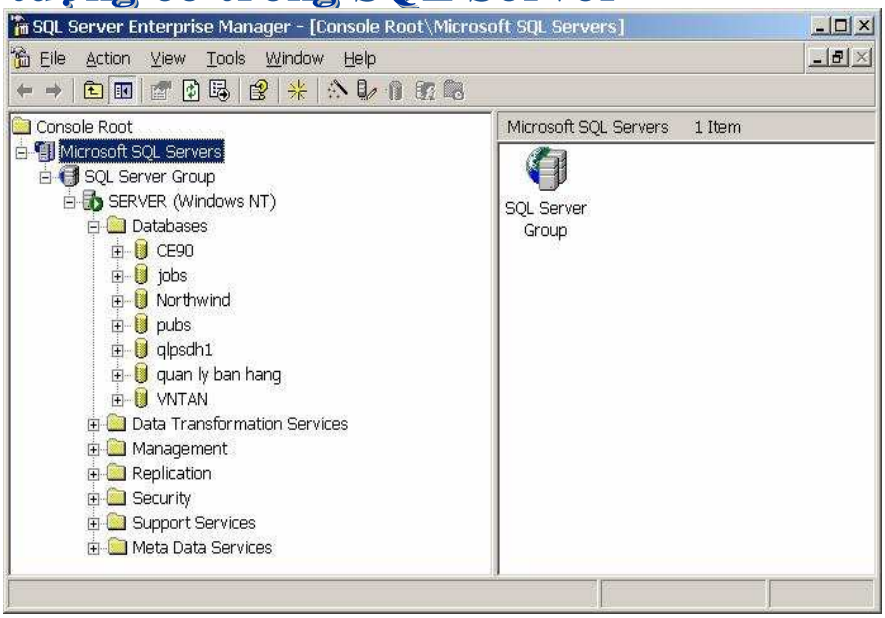


12.Chọn kiểu đăng nhập vào SQL Server



- Nhấp Next để chương trình cài đặt vào máy
- Sau khi cài đặt xong,khởi động lại máy, sau khi khởi động xong,thấy xuất hiện biểu tượng Trên thanh Status của Win vs

1.Tiện ích EnterPriseManager:Màn hình quản lý các đối tượng có trong SQL Server



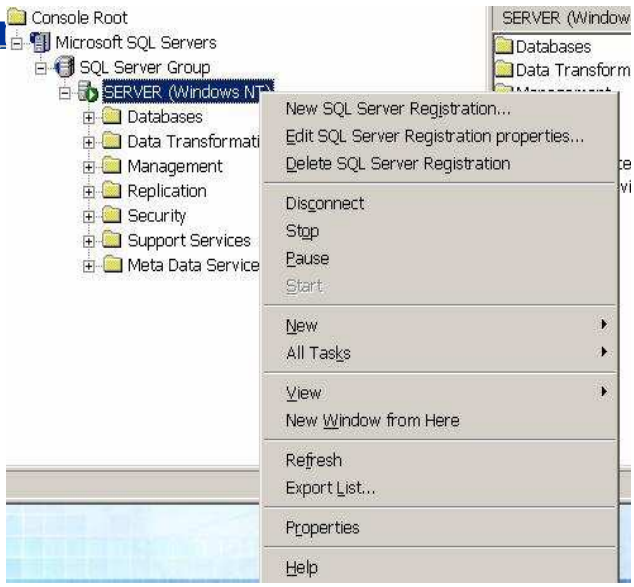
A> Microsoft SQL Server:Chứa các nhóm Server

- Tạo mới một nhóm Server ta làm như sau :
  - Click chuột phải vào Microsoft SQL Server,chọn New SQL Server Group

Màn hình sau xuất hiện

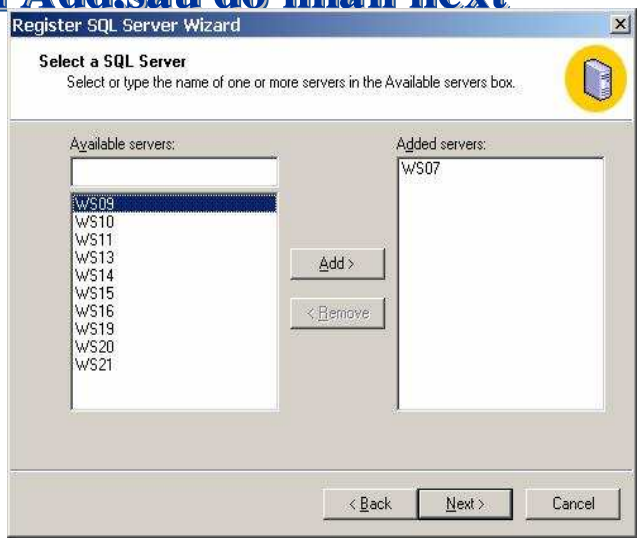
- Chọn Top Level Group : Tạo một nhóm gốc
- Chọn Sub-group of: tạo một nhóm là con của một nhóm khác
- Name:tên nhóm

B>Đăng ký một Server vào hệ thống :Click chuột phải vào SQL Server Group,chọn New SQL Server Registr

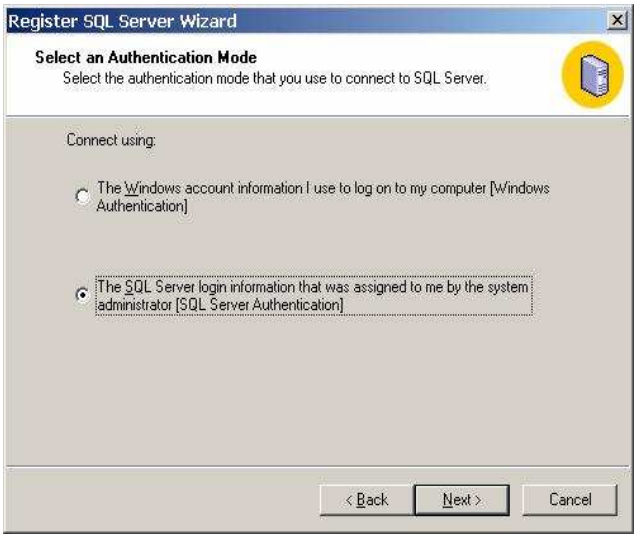


Màn hình sau xuất hiện : Nhấn Next

Màn hình sau xuất hiện : chọn Server cần đăng ký,nhấn Add.sau đó nhấn next

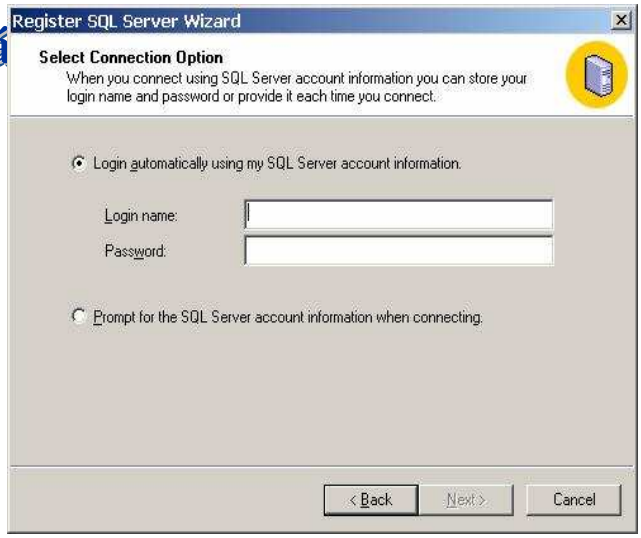


Màn hình sau xuất hiện :Màn hình sau xuất hiện chọn dạng kết nối ,chọn lựa chọn 1:



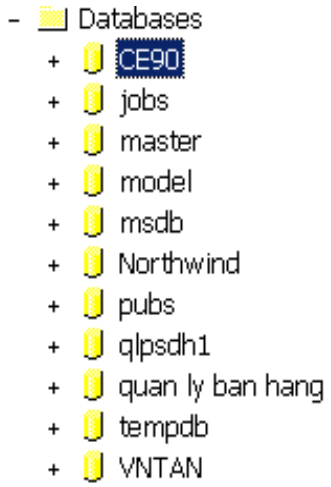
Màn hình sau xuất hiện :Nhập Login name,PassWord để đăng nhập vào SQL Server

- **Chú ý : Nếu quá trình cài đặt ở bước 11 và 12 chọn như trên thì :**
- **Login name :Sa**
- **Password :**

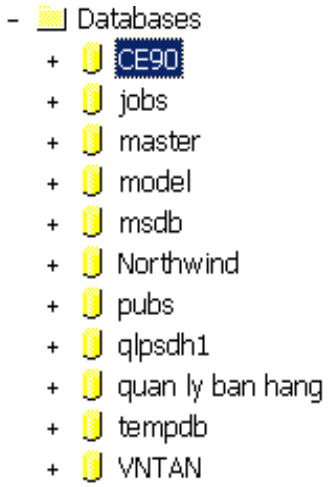


C>DataBases:Chứa các CSDL Trong SQL Server, có 7 CSDL có sẵn ,chia làm 2 loại

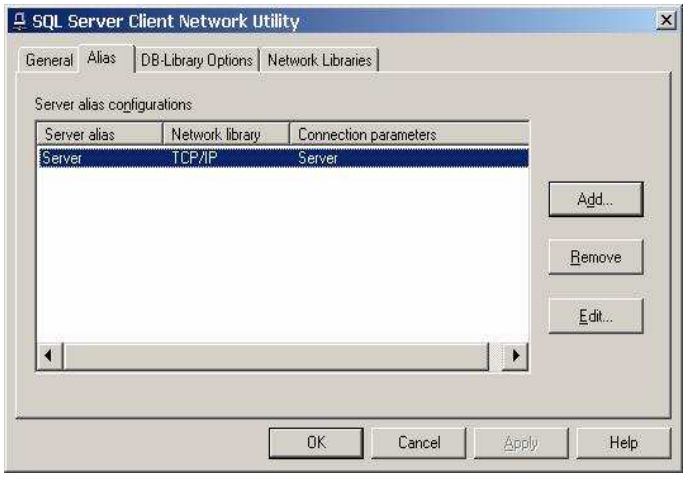
CSDL Hệ thống :không được xoá,sửa	
Tên CSDL	Ý nghĩa
Master	Là CSDL quản lý thông tin của các CSDL còn lại
Mode	Là CSDL mẩu
Msdb	Ề CSDL phục vụ cho dịch vụ SQL Server Agent



Tempdb	Là CSDL chứa các bảng tạm được phát sinh trong quá trình xữ lý
CSDL mẫu	
Tên CSDL	Ý nghĩa
Jobs	
Northwind	
pubs	

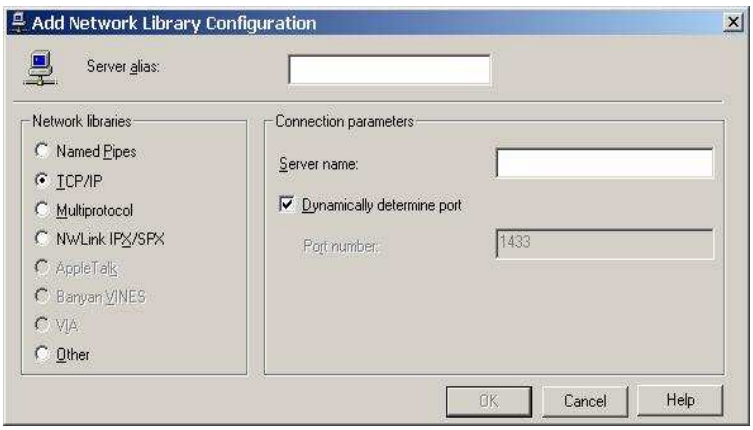


C>Client Network Utility : dùng để đăng ký một Server cho hệ thống



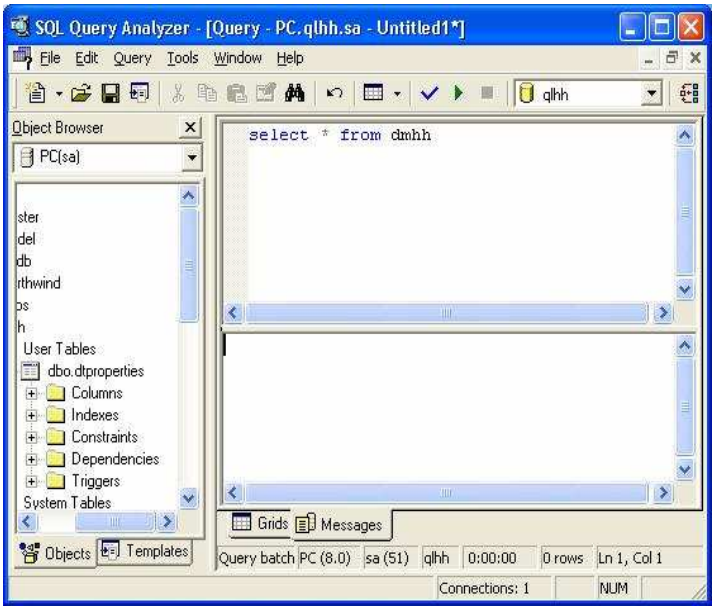
Cách đăng ký : chọn Tab Alias nhấn nút Add,màn hình sau xuất hiện

- Chọn TCP/IP
- Server Alias :nhập vào nhãn
- Server name : nhập vào địa chỉ IP của Server



Chú ý: Tiện ích này chỉ sử dụng trong hệ thống mạng

Tiện ích Query Enalyzer :là màn hình viết lệnh của SQL Server

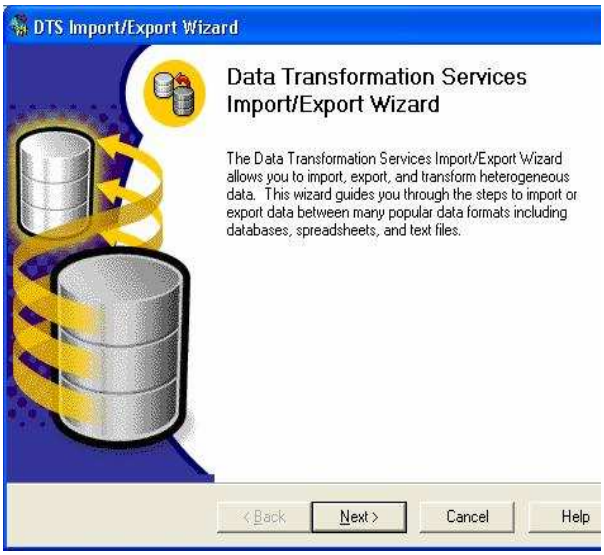


Tiện ích Book Online :Tài liệu hướng dẫn cách sử dụng SQL Server

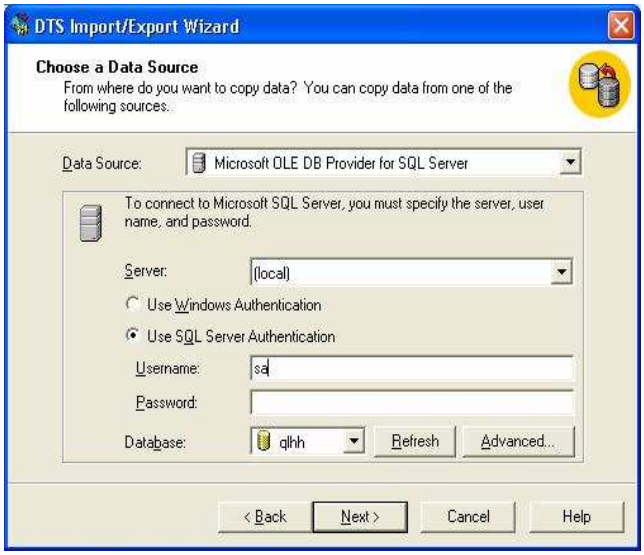
Tiện ích Service Manager :quản lý các dịch vụ liên quan đến SQL Server

Muốn sử dụng SQL Server phải khởi động dịch vụ này

Các dịch vụ này khởi động bằng cách : Start -> Programs->Mcrosoft SQL Server->Dịch vụ



Các tiện ích của SQL Server



Các tiện ích của SQL Server

Các tiện ích của SQL Server

Xây dựng một CSDL trong SQL Server

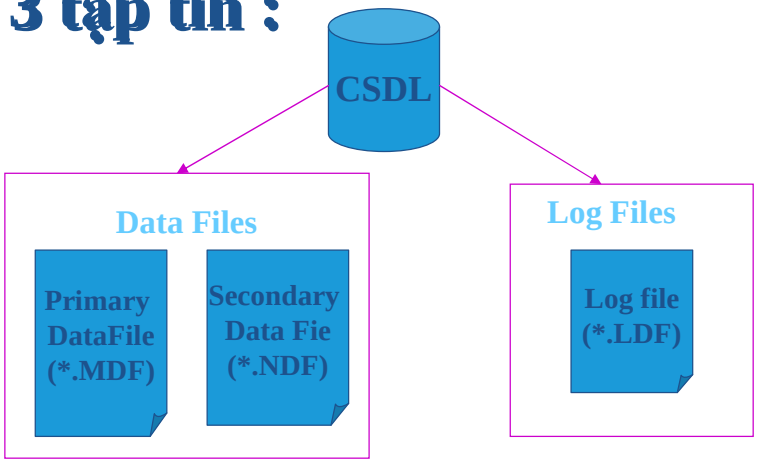
- **Một (CSDL) của SQL Server là tập hợp các đối tượng**
  - ✓ **bảng (Table):Chứa dữ liệu**
  - ✓ **bảng ảo (View):chứa dữ liệu lấy từ Bảng**
  - ✓ **thủ tục nội tại (Stored Procedure): Chứa các lệnh thao tác với Dữ liệu**

- ✓ Triger :Chứa các lệnh thao tác với Dữ liệu,kiểm tra ràng buộc tự động
- ✓ Constraint :Chứa các ràng buộc dữ liệu

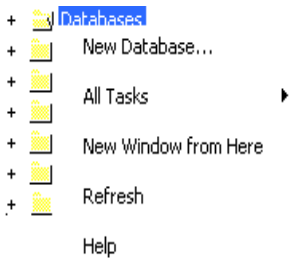
- ➢ Một CSDL do một hoặc nhiều người tạo nhưng có thể được nhiều người truy cập
- ➢ Trong SQL Server có thể tạo được 32767 CSDL khác nhau

- ➢ Cấu trúc lưu trữ của một CSDL : Một CSDL trong SQL Server tối thiểu sẽ dùng hai tập tin vật lý để lưu trữ dữ liệu:
  - ✓ Một dùng lưu trữ dữ liệu (Data file)
  - ✓ Một dùng để lưu trữ các vết giao tác (Transaction log file)

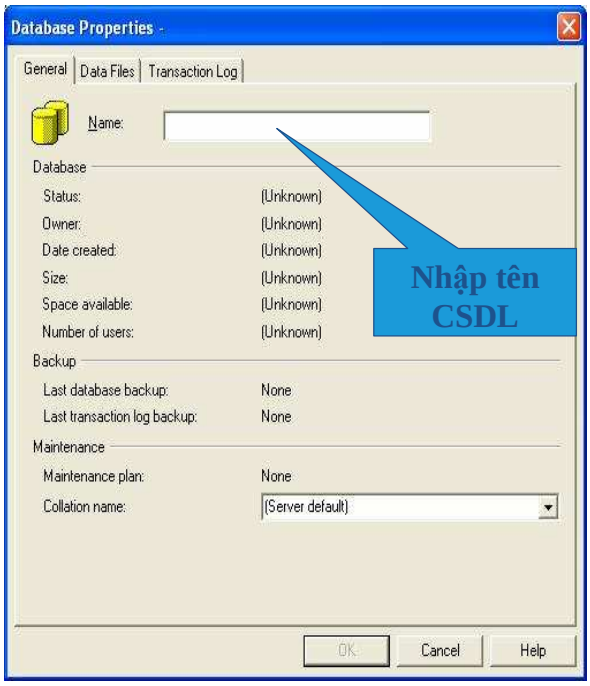
Thường được tổ chức lưu trữ trong 3 tập tin :



- **Tạo mới một CSDL trong SQL Server**
  - **Cách 1: Sử dụng tiện ích EnterPriseManager :Click chuộc phải vào đối tượng Databases chọn New Database**



❖ Xuất hiện hộp thoại :



❖ Chọn Tab Data Files:

- Filename :Tên tập tin Data Files
- Location: Vị trí lưu trữ
- Initial size(Mb) :Kích thước ban đầu
- Automatically grow file: Tự động tăng kích thước
  - In megabytes : tăng bao nhiêu mB
  - By percent :Tăng theo phần trăm
- Maximun file size : kích thước tối đa của CSDL
  - Unrestricted file growth :Không giới hạn kích thước
  - Restrict file growth(MB) :giới hạn

❖ Tương tự cho Transaction Log

❖ Tạo CSDL bằng cách viết lệnh trong Query Enalyzer

```
CREATE DATABASE  QLBANHANG
ON PRIMARY
( NAME= QLBANHANG_DATA,
FILENAME='C:\MSSQL\DATA\ QLBANHANG_DATA.MDF',
SIZE=50MB,
MAXSIZE=200MB,
FILEGROWTH=10%)
LOG ON
(
NAME= QLBANHANG_LOG,
FILENAME='C:\MSSQL\DATA\ QLBANHANG_LOG.LDF',
SIZE=10MB,
MAXSIZE=UNLIMITED,
FILEGROWTH=5MB
)
```


❖ Các kiểu dữ liệu trong SQL Server

Kiểu dữ liệu	Kích thước	Miền giá trị dữ liệu lưu trữ
> Các kiểu dữ liệu dạng số nguyên		
Int	4 bytes	từ-2,147,483,648đến +2,147,483,647
SmallInt	2 bytes	từ -32768 đến +32767
TinyInt	1 byte	từ 0 đến 255
Bit	1 byte	0, 1 hoặc Null
> Các kiểu dữ liệu dạng số thập phân		
Decimal, Numeric	17bytes	từ -10^{38} đến $+10^{38}$
> Các kiểu dữ liệu dạng số thực		
Float	8 bytes	từ -1.79E+308 đến +1.79E+308
Real	4 bytes	từ -3.40E+38 đến +3.40E+38

❖ Các kiểu dữ liệu trong SQL Server

> Các kiểu dữ liệu dạng chuỗi có độ dài cố định		
Char	N bytes	từ 1 đến 8000 ký tự, mỗi ký tự là một byte
> Các kiểu dữ liệu dạng chuỗi có độ dài biến đổi		
VarChar	N bytes	từ 1 đến 8000 ký tự, mỗi ký tự là 1 byte
Text	N bytes	từ 1 đến 2,147,483,647 ký tự, mỗi ký tự là 1 byte
> Các kiểu dữ liệu dạng chuỗi dùng font chữ Unicode		
NChar	2*N bytes	từ 1 đến 4000 ký tự, mỗi ký tự là 2 bytes
NVarChar	2*N bytes	từ 1 đến 4000 ký tự, mỗi ký tự là 2 bytes
NText	2*N bytes	từ 1 đến 1,073,741,823 ký tự, mỗi ký tự là 2 bytes

❖ Các kiểu dữ liệu trong SQL Server

> Các kiểu dữ liệu dạng tiền tệ		
Money	8 bytes	từ -922,337,203,685,477.5808 đến +922,337,203,685,477.5807
SmallMoney	4 bytes	từ -214,748.3648 đến + 214,748.3647
> Các kiểu dữ liệu dạng ngày và giờ		
DateTime	8 bytes	từ01/01/1753đến31/12/9999
SmallDateTime	4 bytes	từ01/01/1900đến06/06/2079
> Các kiểu dữ liệu dạng chuỗi nhị phân (Binary String)		
Binary	N bytes	từ 1 đến 8000 bytes
VarBinary	N bytes	từ 1 đến 8000 bytes
Image	N bytes	từ 1 đến 2,147,483,647 bytes

❖ Các thuộc tính cơ bản của Một Bảng

- **Tên bảng** (Table Name) Độ dài<= 128 ký tự. Tên bảng phải duy nhất bên **trong phạm vi** của người đã tạo ra nó trong một CSDL
- **Tên cột** (Column Name): là tên của các cột bên trong bảng, tên của các cột bên trong bảng phải duy nhất.

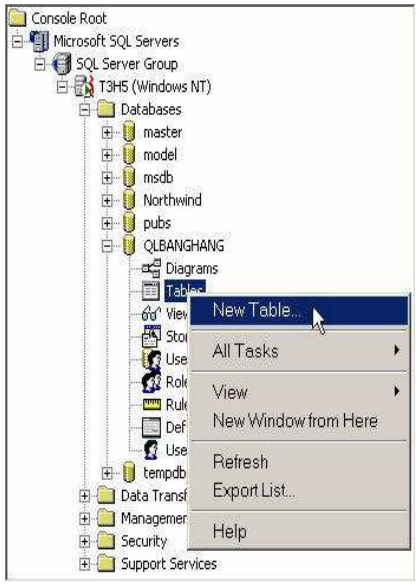
- **Độ dài dữ liệu** (Data Length): dùng để qui định độ dài dữ liệu mà cột sẽ lưu trữ đối với các kiểu dữ liệu dạng chuỗi, số.
- **Số ký số lưu trữ** (Precision): là số ký số tối đa mà các kiểu dữ liệu dạng số có thể lưu trữ được.
- **Số lẻ lưu trữ** (Scale): là số lẻ tối đa mà các kiểu dữ liệu dạng số thập phân dùng để lưu trữ.

- **Cột định danh** (Identity): Giá trị của cột định danh là số nguyên không trùng lắp do hệ thống Microsoft SQL Server tự động cấp phát.
  - Các kiểu dữ liệu của cột định danh : Int, SmallInt, TinyInt, Decimal , Numeric
  - Thuộc tính **Identity Seed :Số đầu tiên**
  - Thuộc tính **Identity**

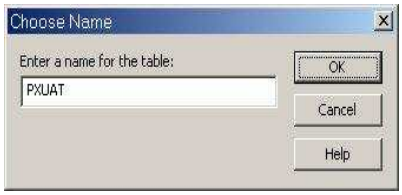
Chương II: Xây dựng một CSDL trong SQL Server

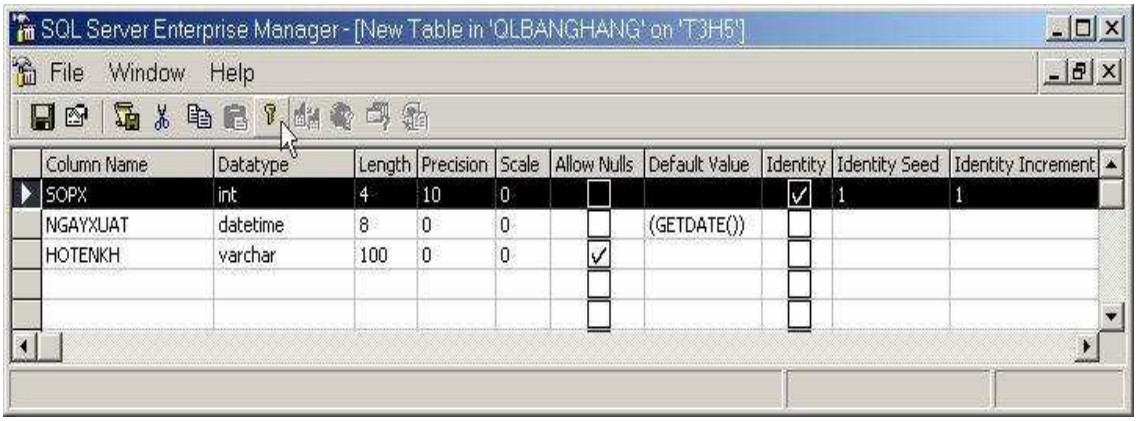
- **Cho phép để trống dữ liệu** (Allow Null): qui định dữ liệu có thể được phép để trống hay là phải nhập
- **Giá trị mặc định** (Default Value): là giá trị mặc nhiên sẽ được gán vào cột dữ liệu khi người sử dụng thêm mới một mẫu tin nhưng lại để trống giá trị tại cột dữ liệu đó.

- **Tạo Bảng trong Sql Server**
  - Cách 1: Sử dụng tiện ích **Enterprise Manager**
    - Mở CSDL Click chuột phải trên đối tượng **Tables**, chọn chức năng **New Table...** trong thực đơn tắt.



- **Màn hình sau xuất hiện : Nhập tên bảng,nhấp OK**
- **Màn hình thiết kế bảng xuất hiện :nhập thông tin và chọn cột làm khoá chính**





❖ Tạo bảng bằng lệnh trong màn hình Query Elyzer

Tạo cấu trúc bảng đơn giản

Cú pháp:

```
CREATE TABLE Tên_bảng
(
Tên_cột_1 Kiểu_dữ_liệu [NOT NULL] ,
Tên_cột_2 Kiểu_dữ_liệu [NOT NULL] [ , ...]
)
```

Trong đó:

+ Tên bảng, tên cột và kiểu dữ liệu đã được mô tả ở trên

+ Từ khóa NOT NULL chỉ định không cho phép dữ liệu tại cột đó được phép bỏ trống.

❖ Tạo bảng bằng lệnh trong màn hình Query Elyzer

Tạo cấu trúc bảng có giá trị mặc định

Cú pháp:

```
CREATE TABLE Tên_bảng
(
Tên_cột_1 Kiểu_dữ_liệu DEFAULT Giá_trị|Hàm ,
Tên_cột_2 Kiểu_dữ_liệu [NOT NULL] [ ,...]
)
```

Trong đó:

+ Giá trị: là giá trị cụ thể sẽ được gán vào cột.

+ Hàm: là tên hàm cùng với các tham số (nếu có), tuy nhiên kiểu dữ liệu mà hàm trả về phải là cùng với kiểu dữ liệu mà cột sẽ lưu trữ.

Tạo bảng bằng lệnh trong màn hình Query Elyzer

Ví dụ 1

```
CREATE TABLE VATTU
(
MAVTU CHAR(4) NOT NULL,
TENVTU VARCHAR(100) NOT NULL,
DVTINH VARCHAR(10) NOT NULL,
PHANTRAM REAL DEFAULT 20
)
```

Ví dụ 2

```
CREATE TABLE DONDH
(
SODH CHAR(4) NOT NULL,
NGAYDH DATETIME DEFAULT GETDATE(),
MANHACC CHAR(3) NOT NULL
)
```


Tạo cấu trúc bảng có cột định danh

Cú pháp:

```
CREATE TABLE Tên_bảng
(
Tên_cột_1 Kiểu_dữ_liệu_số IDENTITY [(Số_bắt_đầu,
Chỉ_số_tăng)] ,
Tên_cột_2 Kiểu_dữ_liệu [NOT NULL] [ ,...]
)
```

- Trong một bảng chỉ có tối đa một cột được chỉ định làm cột định danh.
- Số bắt đầu: là số mà SQL Server sử dụng để cấp phát cho mẫu tin đầu tiên. Mặc định là 1.
- Chỉ số tăng: là chỉ số mà SQL Server cộng lên để cấp phát cho từng mẫu tin kết tiếp. Mặc định là 1.

- **Tính toàn vẹn dữ liệu trong cơ sở dữ liệu : các ràng buộc trong SQL Server được quản lý bởI hai đốI tượng :CONSTRAINT và TRIGGER**
- **Constraint : quản lý các ràng buộc như :**
  - Ràng buộc dữ liệu duy nhất
  - Ràng buộc dữ liệu tồn tại
  - Ràng buộc miền giá trị
- **Trigger :Quản lý các ràng buộc phức tạp,liên quan đến dữ liệu trên nhiều bảng**

- **Xây dựng các Constraint**
- **A>Kiểm tra duy nhất dữ liệu :có hai loại**
  - Primary Key (khoá chính) :dữ liệu khác rỗng và duy nhất
  - Unique : dữ liệu có thể rỗng và dữ liệu tồn tại duy nhất

Định nghĩa ràng buộc khoá chính :

```
CREATE TABLE Tên_bảng
(
Tên_cột_1 Kiểu_dữ_liệu DEFAULT Giá_trị|Hàm
  ,
Tên_cột_2 Kiểu_dữ_liệu [NOT NULL] [ ,...]
[CONSTRAINT PK_Tên_bảng_TenCot
  ] PRIMARY
    KEY (Danh_sách_cột_khóa_chính)
)
```


Định nghĩa ràng buộc dữ liệu duy nhất :

```
CREATE TABLE Tên_bảng
(
Tên_cột_1 Kiểu_dữ_liệu DEFAULT Giá_trị|Hàm
  ,
Tên_cột_2 Kiểu_dữ_liệu [NOT NULL] [ ,...]
[CONSTRAINT
  UQ_Tên_bảng_Tên_cột]
  UNIQUE (Danh_sách_các_cột)
)
```

Trong một bảng chỉ định nghĩa một ràng buộc khoá chính nhưng có thể định nghĩa nhiều ràng buộc dữ liệu duy nhất

Xây dựng một CSDL trong SQL Server

Ví dụ :

```
CREATE  TABLE VATTU
(
MAVTU varchar(4) not null,
 TenVtu varchar(100),
Constraint PK_Vattu_Mavtu_tenvtu primary
  key(Mavtu,tenvtu),
constraint UQ_Vattu_tenvtu unique(tenvtu)
)
```


Xây dựng một CSDL trong SQL Server

B>Kiểm tra tồn tại Dữ liệu :Định nghĩa khoá ngoại

```
CREATE TABLE Tên_bảng
(
Tên_cột_1 Kiểu_dữ_liệu DEFAULT Giá_trị|Hàm
  ,
Tên_cột_2 Kiểu_dữ_liệu [NOT NULL] [ ,...]
[CONSTRAINT
  FK_Tên_bảng_Tên_cột]
FOREIGN                                KEY
  (Danh_sách_cột_khóa_ngoại)
REFERENCES   Tên_bảng_tham_chiếu
  (Danh_sách_cột_tham_chiếu)
)
```

B>Ví dụ :

```
CREATE TABLE CTHOADON
(
SOHD Varchar(3) not null ,
MAVTU Varchar(3) not null,
Soluong int,
Dongia float,
CONSTRAINT FK_CTHOADON_SOHD
FOREIGN KEY (SOHD)
REFERENCES HOADON(SOHD),
CONSTRAINT FK_CTHOADON_MAVTU
FOREIGN KEY (MAVTU)
REFERENCES VATTU(MAVTU)
)
```


C>Kiểm tra ràng buộc miền giá trị :

Cú pháp :

```
CREATE TABLE Tên_bảng
(
Tên_cột_1 Kiểu_dữ_liệu DEFAULT Giá_trị|Hàm
  ,
Tên_cột_2 Kiểu_dữ_liệu [NOT NULL] [ ,...]
[CONSTRAINT
  CK_Tên_bảng_Tên_cột]
CHECK (Biểu_thức_luận_lý)
)
```

Xây dựng một CSDL trong SQL Server

B>Ví dụ :

```
CREATE TABLE CTHOADON
(
SOHD Varchar(3) not null ,
MAVTU Varchar(3) not null,
Soluong int,
Dongia float,
CONSTRAINT
  CK_CTHOADON_SOLUONG
CHECK (Soluong>0),
CONSTRAINT CK_CTHOADON_DonGia
CHECK (Dongia>0)
)
```


Xây dựng một CSDL trong SQL Server

B>Thêm Costraint vào bảng đã có

Cú pháp chung:

```
ALTER TABLE Tên_bảng
ADD [ CONSTRAINT Tên_Constraint ]
LOẠI Các_tham_số [ , ...]
```

Cú pháp chi tiết từng loại constraint

PRIMARY KEY

ALTER TABLE Tên_bảng

ADD [CONSTRAINT PK_Tên_bảng]

PRIMARY KEY (Danh_sách_cột_khóa_chính)

UNIQUE

ALTER TABLE Tên_bảng ADD [CONSTRAINT UQ_Tên_bảng_Tên_cột]

UNIQUE (Danh_sách_các_cột)

FOREIGN KEY

ALTER TABLE Tên_bảng

ADD [CONSTRAINT FK_Tên_bảng_Tên_cột]

FOREIGN KEY (Danh_sách_cột_khóa_ngoại)

REFERENCES Tên_Bảng (Danh_sách_cột_tham_chiếu)

Kiểm tra miền giá trị :

ALTER TABLE Tên_bảng

ADD [CONSTRAINT CK_Tên_bảng_Tên_cột]

CHECK (Biểu_thức_luận_lý)

Thêm giá trị mặc định :

ALTER TABLE Tên_bảng

ADD [CONSTRAINT DF_Tên_bảng_Tên_cột]

DEFAULT Giá_trị_mặc_định FOR Tên_cột

Hủy một Constraint :

ALTER TABLE Tên_bảng

DROP CONSTRAINT Tên_constraint [, ...]

Tắt các Constraint :

ALTER TABLE Tên_bảng

NOCHECK CONSTRAINT ALL| Tên_constraint [, ...]

Bật các Constraint :

ALTER TABLE Tên_bảng

CHECK CONSTRAINT ALL| Tên_constraint [, ...]

Xóa cấu trúc bảng

DROP TABLE Danh_sách_tên_các_bảng

Thay đổi cấu trúc bảng

a. Thêm một cột mới trong bảng

ALTER TABLE Tên_bảng

ADD Tên_cột Kiểu_dữ_liệu [, ...]

b. Hủy bỏ cột hiện có bên trong bảng

ALTER TABLE Tên_bảng

DROP COLUMN Tên_cột [, ...]

Sửa đổi kiểu dữ liệu của cột

ALTER TABLE Tên_bảng

ALTER COLUMN Tên_cột Kiểu_dữ_liệu_mới

Đổi tên cột

EXEC SP_Rename

"Tên_bảng.Tên_cột","Tên_mới","COLUMN"

Đổi tên bảng

EXEC sp_rename "Tên_bảng","Tên_mới"

Chú ý :Muốn thay đổi cấu trúc của cột phải xoá tất cả các Constraint liên quan

BÀI 3: BÀNG Ảo(view)

- **1>Định Nghĩa :**
  - Bàng ào là một đối tượng chứa câu lệnh Select lấy dữ liệu

  Từ các bảng trong csdl

  Khi truy cập vào CSLD ,người sử dụng không phân biệt được Bảng ảo và Bảng Thật

  2> Tạo View

  Create view Ten_view

  As

  Câu lệnh Select

Cấu trúc câu lệnh Select

- **2>Cấu trúc câu lệnh Select**

  Select[Ten_Bang.]Ten_Cot[,…]

  From Ten_Bang1 <Liên kết > Ten_Bang2

  Where <Điều kiện>

  Group by <các cột gom nhóm>

  HaVing <Điều kiện lọc nhóm>

  Order by [Ten_Bang.]Ten_cot [asc/desc,…]

  Compute Hàm thống kê [By Ten_cot]

Cấu trúc câu lệnh Select

- **Mệnh đề Select :chứa các cột,các hàm muốn hiển thị**
- **Nếu cột tồn tại trong hai bảng trong mệnh đề From ta phải ghi rõ :Tên_Bảng.Tên_Cột**

Trong mệnh đề Select ta có thể dùng cấu trúc lệnh

Case
When điều kiện 1 Then Giá trị 1
When Điều kiện 2 then Giá trị 2
…
Else
Giá tri n
End
Để hiển thị dữ liệu

- **Mệnh đề from : cho biết dữ liệu trên câu truy vấn được lấy từ các bảng nào**
- **nếu dữ liệu lấy từ nhiều bảng thì các bảng này phải có quan hệ với nhau:**

Cấu trúc câu lệnh Select

Bảng 1 **Inner** / **Left** / **Right** / **full** **Join Bảng 2 on Bảng 1.tên cột= bảng 2.tên cột**

- **Mệnh đề Where : mệnh đề lọc dữ liệu trả về hoặc để gom nhóm**
- **Trong mệnh đề Where có thể sử dụng các toán tử :**
  - Like tìm chuỗi gần đúng
    - % Thay thế 1 chuỗi
    - _(Gạch dưới) thay thế 1 ký tụ
  - Toán tử BETWEEN Gt1 AND GT2

Cấu trúc câu lệnh Select

- **<u>Mệnh đề Group by</u> : mệnh đề gom nhóm dữ liệu**
  - Mệnh đề này phải có khi trong câu truy vấn có sử dụng hàm thống kê và trên mệnh đề Select có lấy các giá trị không sử dụng hàm thống kê
  - Cú pháp : Group By Danh sách cột,Biểu thức
  - Với Danh sách cột,Biểu thức :được lấy trên mệnh đề Select không sử dụng hàm thống kê
- **Các Hàm thống kê:**
  - Sum(Ten_Cot) :tính tổng với cột có kiểu số
  - Count(Ten_Cot/*) :tính tổng số dòng
  - AVG(Ten_Cot) :tính giá trị trung bình
  - Max(Ten_cot) : giá trị lớn nhất
  - Min(Ten_Cot) :giá trị nhỏ nhất)

Cấu trúc câu lệnh Select

- **<u>Mệnh đề HaVing</u> : mệnh đề Lọc nhóm dữ liệu**
  - **Mệnh đề này chỉ tồn tại khi trong câu truy vấn có mệnh đề Group By và điều kiện lọc dữ liệu có hàm thống kê**
  - **Chú ý : trong câu truy vấn có hai mệnh đề lọc dữ liêu là Where và havinh**
    - **Where :lọc dữ liệu để tạo nhóm (có điều kiện không sữ dụng hàm thống kê)**
    - **Having :lọc nhóm(các điều kiện có thống kê)**
- **Mệnh đề OrDer By : Mệnh đề sắp xếp dữ liệu**
  - Cú pháp Order by [Ten_bang.]Ten_Cot [asc/Desc] [,…]
- **Mệnh đề Compute /Compute .. By : là mệnh đề tạo giá trị thống kê cuối kết quả hoặc thống kê theo cột**
  - Cú pháp
    - Compute Hàm thống kê : tạo thống kê cuối kết quả
    - Compute Hàm thống kê By [Ten_Bang.]Ten_cot :thống kê theo cột,dữ liệu phải được sắp xếp theo cột này

Cấu trúc câu lệnh Select

- Các toán tử dùng trên mệnh đề Select
  - Cú pháp : Select Toán tử …
  - Với các toán tử :
    - Distict : loại bỏ các dòng dữ liệu trùng
    - Top n : lấy n dòng đầu tiên
    - Top n with ties : lấy những dòng thuộc n cấp đầu tiên
    - Top n PERCENT : lấy N % dòng đầu tiên

Cấu trúc câu lệnh InSert into

- **Insert into** : dùng thêm dữ liệu vào bảng,có 2 dạng
  - **Thêm một dòng**
  - Cú pháp : Inser into Ten_bang(Ten_cot1,Ten_cot2[,…]) Values(Gia_Tri1,…)

    với giá trị :
    - ✓ kiểu chuỗi : nằm trong hai dấu ‘’ ví dụ ‘Trần thanh nhật’
    - ✓ Kiểu Ngày : ‘yyyy-mm-dd’ ví dụ :’1982-02-29’
  - **Thêm nhiều dòng**
  - Cú pháp : Inser into Ten_bang_Dich(Ten_cot1,Ten_cot2[,…])

    Select Ten_bang_nguon.ten_cot ,…

    from …

Cấu trúc câu lệnh UpDate

- **UpDate** : dùng cập nhật dữ liệu của một bảng
  - Dữ liệu Update chỉ trên một Bảng :
    - Cú pháp :

      UpDate Ten_Bang
      Set Ten_Cot= Giá trị/Biểu thức
      [Where Điều kiện]
  - Dữ liệu Update trên nhiều Bảng :
    - Cú pháp :

      UpDate Ten_Bang
      Set Ten_Cot= Giá trị/Biểu thức
      From …
      [Where Điều kiện]

      Chú ý : dữ liệu cập nhật có thể là một giá trị thống kê trả về từ câu truy vấn con

      Có thể dùng cấu trúc Case trong lệnh cập nhật

      Trong câu truy vấn update không sử dụng hàm thống kê

Cấu trúc câu lệnh Delete

- **Delete** : dùng xóa dữ liệu của một bảng
  - Dữ liệu xóa chỉ trên một Bảng :
    - Cú pháp :

      delete Ten_Bang
      [Where Điều kiện]
  - Dữ liệu delete trên nhiều Bảng :
    - Cú pháp :

      delete Ten_Bang
      From …
      [Where Điều kiện]

      Chú ý : khi xóa dữ liệu trên bảng quan hệ 1 mà dữ liệu này đã có bên bảng quan hệ nhiều -> có lỗi

Cấu trúc câu lệnh Select .. into

- **Select .. into** : dùng tạo một bảng có cấu trúc và dữ liệu lấy từ các bảng có trong CSDL
  - Cú pháp :

    Select [Ten_Bang.]tenCot [,…]

    From ..

Câu truy vấn Con (Sub Query)

- Sub Query :là câu truy vấn mà kết quả trả về là điều kiện để so sánh,gán giá trị cho câu truy vấn khác
- Đặt điểm :
  - Chỉ trả về 1 cột dữ liệu
  - Nếu trả về 1 dòng : điều kiện so sánh giữa câu truy vấn chính và câu truy vấn con là các toán tử :=,>,<,<>,>=,<=
  - Nếu trả về nhiều dòng : điều kiện so sách : in (nằm trong) hoặc Not in(không nằm trong)

Bài 4: Lập trình với SQL SerVer

- **I>Biến và các cấu trúc lệnh trong SQL Server**
- **A>Biến :**

  1> Biến cục bộ : là biến do người lập trình khai báo,Biến có thể được khai báo trong thủ tục nội tại,hàm, trong một lô lệnh

  - Lô Lệnh : là tập hợp các lệnh được kết thúc bằng từ **Go**
  - Cú pháp khai báo biến : **Declare @Tên_Biên Kiểu_Dữ_Liệu**
    - ✓ Tên Biến :Luôn bắt đầu bằng ký tự @,không có khoảng trắng,ký tự đặt biệt,ký tự số đứng đầu
  - Gán giá trị cho biến:

    Gán bằng 1 giá trị :

    Set @tên_biến=giátrị/hàm

    hoặc **Select @tên_biến=Giátrị/hàm**

  Gán bằng kết quả trả về của câu truy vấn

  Select @tên_Biến=Tênsốt/Hàm ... from

Bài 4: Lập trình với SQL SerVer

- **1> Biến hệ thống** (t155) : là biến do SQL Server cung cấp,Cho biết trạng thái của hệ thống,Biến hệ thống luôn bắt đầu bằng hai ký tự @@,Chỉ đọc
- Một số các biến thường dùng

Tên Biến	Ý nghĩa
@@Error	Cho biết các lệnh trước đó có lỗi hay không,nếu có lỗi @@Error <>0
@@Rowcount	Cho biết số dòng bị tác động bởi câu truy vấn gần nhất
@@Fetch_Status	Trả về <>0 nếu lệnh duyệt CurSor gây lỗi

- **B>Các cấu trúc lênh**
  - **1>lệnh điều khiển IF .. Else**
  - **Cú pháp :**

```
If   Điều kiện
        Begin
                <tập Lệnh 1>
        end
else
        Begin
                <tập Lệnh 2>
        End
```

- **Nếu điều kiện đúng : tập Lệnh 1 thực hiện, ngược lại tập lệnh 2 thực hiện**
- **Nếu tập hợp lệnh 1 ,lệnh 2 chỉ có 1 lệnh thì không cần Begin .. End**

- **B>Các cấu trúc lênh**
  - **2>Cấu trúc lặp :While**

```
While Biểu thức Điều kiện
    Begin
      <Tập lệnh>
    end
```

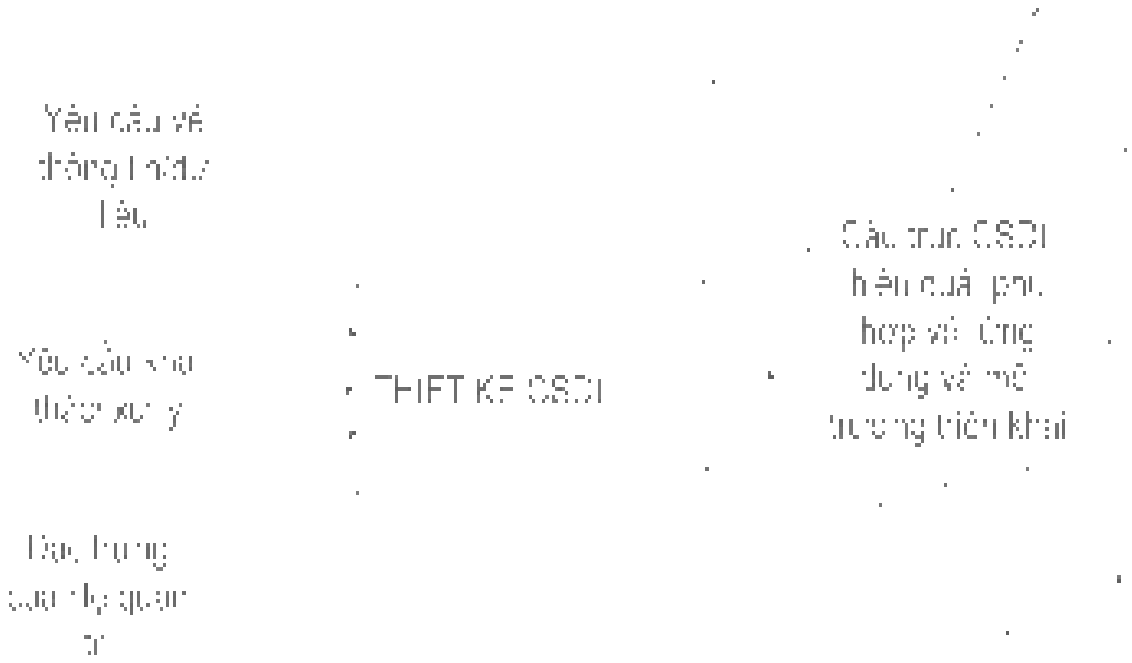
- Tập lệnh sẽ được thực hiện đến khi biểu thức điều kiện trả về False
- -có thể dùng lệnh **Break** để thoát khỏi vòng lặp

Bài 4: Lập trình với SQL SerVer

- **2>Các hàm cơ bản trong SQL SerVer**
- **A>Các hàm chuyển đổi kiểu dữ liệu :thường dùng để chuyển dữ liệu từ số,ngày sang chuỗi**
- **Hàm Cast : chuyển một kiểu dữ liệu sang kiểu bất kỳ**
  - Cú pháp :***Cast(Biểu_Thức as Kiểu_Dữ Liệu)***
    - Ví dụ : Cast (@tong as varchar(10))
- **Hàm CONVERT : chuyển từ ngày sang chuỗi**
  - Cú pháp :***Convert(Kiểu_dữ_Liệu,Biểu_Thức[,Định_Dạng])***

Chương 3: Tổng quan quá trình thiết kế CSDL

Giới Thiệu



Các tiêu chí thiết kế

- Tính uyển chuyển của cấu trúc CSDL: dễ dàng thêm, bớt, chỉnh sửa trong tương lai
- Chi phí lưu trữ và chi phí truy xuất và chi phí cập nhật
- Nhu cầu của các nhóm người sử dụng khác nhau
- Phạm vi ứng dụng

Chu kỳ phát triển một CSDL

- **Trải qua các giai đoạn:**
  - Phân tích nhu cầu NSD
  - Thiết Kế:
    - Mức quan niệm
    - Mức logic
    - Mức vật lý
  - Cài đặt và cập nhật dữ liệu ban đầu
  - Khai thác
  - Bảo hành

Chu kỳ phát triển một CSDL

Giai đoạn phân tích yêu cầu của NSD

- **Mục tiêu: làm việc với NSD để xác định một số thông tin sau:**
  - Mục tiêu và yêu cầu chung về hiệu suất của hệ thống.
  - Các thông tin cần thiết của CSDL trong hệ thống và ý định khai thác. (lưu thông tin gì, chung chung, chi tiết hay chỉ để tham khảo…)
  - Khối lượng và tần suất truy xuất.
  - Quy định của tô chức
  - Yêu cầu về an toàn và đề phòng sự cố.

Giai đoạn phân tích yêu cầu của NSD

- **Cách thực hiện:**
  - Xác định những người sử dụng và lập bảng ưu tiên.
  - Phỏng vấn (interviewing), thảo luận (discussion), làm bài khảo sát (quiz), theo dõi công việc hằng ngày (shadowing)
  - Vẽ sơ đồ luồng dữ liệu Data Flow Diagram (DFD) để có cái nhìn tổng thể.
  - Xác định các nút trọng điểm trong sơ đồ (CSDL phải phục vụ tốt cho các nút trọng điểm.
  - Lập sưu liệu thể hiện kết quả phân tích.

Giai đoạn phân tích yêu cầu của NSD

- Xác định những người sử dụng và lập bảng ưu tiên cho csdl quản lý kho hàng. (bảng)
- Chọn phương pháp thu thập thông tin thích hợp.(tên phương pháp)
- Vẽ sơ đồ luồng dữ liệu Data Flow Diagram (DFD) và xác định các nút trọng điểm trong sơ đồ cho bài toán quản lý kho.(lược đồ và danh sách nút)
- Lập sưu liệu thể hiện kết quả phân tích.(tài liệu tổng hợp)

Giai đoạn Thiết kế CSDL

- **Qua trình thiết kế phải thực hiện ở 3 mức:**
  - Mức quan niệm.
  - Mức Logic
  - Mức vật lý.

Mức quan niệm

- **Quan tâm đến nội dung dữ liệu (what)**
- **Chọn một mô hìnhvà diễn đạt dữ liệu theo mô hình đó.**
- **Input: kết quả của giai đoạn phân tích và phát thảo cấu trúc CSDL.**
- **Output: lược đồ CSDL mức quan niệm (bảo đảm lưu vừa đủ thông tin, tốt hơn và phù hợp hơn với môi trường phát triển ứng dụng.**

Mức quan niệm

- **Quan tâm đến nội dung dữ liệu (what)**
- **Chọn một mô hìnhvà diễn đạt dữ liệu theo mô hình đó.**
- **Input: kết quả của giai đoạn phân tích và phát thảo cấu trúc CSDL.**
- **Output: lược đồ CSDL mức quan niệm (bảo đảm lưu vừa đủ thông tin, tốt hơn và phù hợp hơn với môi trường phát triển ứng dụng.**

Mức logic

- **Trung gian giữa mức quan niệm và vật lý**
- **Quan tâm đến nhu cầu khai thác**
  - Tần suất truy xuất
  - Các con đường truy xuất
- **Input: lược đồ mức quan niệm và nhu cầu khai thác.**
- **Output: cấu trúc CSDL phù hợp với ứng dụng; đồ thị quan hệ và các con đường truy xuất.**

Giới Thiệu

Music Inventory Data Dictionary

Database is MIToeTappin written in Oracle 10i

Table: Artist

Indexes: *ByCode* *Artist_CD*
ByName *Artist_NM*

Use: This table contains all of the music artists.

Field	*Description*
ARTIST_CD	Unique code identifying record
ARTISTS_NM	Artist name

Table: Inventory

Indexes: *ByCode* *Media _CD*
ByType *Media_Type*

Use: This table contains all of the music items found in the store's inventory.

Field	*Description*
MEDIA _CD	Unique code identifying record
MEDIA_TYPE	Media type (CD, tape, album, etc.)
ON_HAND	Quantity on hand
MRP	Minimum re-order point
COST	Our cost
PRICE	Retail sales price

Mức vật lý

- **Cài đặt trên một hệ quản trị CSDL cụ thể**
- **Quan tâm đến hiệu quả và tốc độ xử lý**
- **Input:**
  - Kết quả thiết kế ở quan niệm và logic.
  - Thông tin về hệ quản trị CSDL.
- **Output: cài đặt hoàn chỉnh, khai báo khóa, ràng buộc và thông số hệ thống.**

Chương 4: Lý thuyết mô hình CSDL quan hệ

1. Giới thiệu

- Mô hình Dữ liệu Quan hệ (*Relational Data Model*) dựa trên khái niệm quan hệ.
- *Quan hệ* là khái niệm toán học dựa trên nền tảng hình thức về lý thuyết tập hợp.
- Mô hình này do TS. E. F. Codd đưa ra năm 1970.

2.1 Thuộc tính (attribute)

- Thuộc tính:
  - ***Tên gọi***: dãy ký tự (gợi nhớ)
  - ***Kiểu dữ liệu***: Số, Chuỗi, Thời gian, Luận lý, OLE.
  - ***Miền giá trị***: tập giá trị mà thuộc tính có thể nhận. Ký hiệu miền giá trị của thuộc tính A là ***Dom***(A).
- **Ví dụ**:GIOITINH kiểu dữ liệu là Chuỗi,miền giá trị

  Dom(GIOITINH)=('Nam','Nu')
- Tại một thời điểm, một thuộc tính không có giá trị hoặc chưa xác định được giá trị => giá trị Null

2.2 Quan hệ (relation)

- **Định nghĩa**: quan hệ là một tập hữu hạn các thuộc tính.
  - **Ký hiệu**: $Q(A_1, A_2, ..., A_n)$
  - Trong đó Q là tên quan hệ, $Q^+ = \{A_1, A_2, ..., A_n\}$ là tập các thuộc tính của quan hệ Q
  - **Ví dụ**:

HOCVIEN (Mahv, Hoten, Ngsinh, Gioitinh, Noisinh, Malop)

LOP (Malop, Tenlop, Siso, Trglop, Khoa)

2.3 Bộ (tuple)

- **Định nghĩa:** Bộ là các thông tin của một đối tượng thuộc quan hệ, được gọi là mẫu tin (record), dòng.
- Quan hệ là một bảng (table) với các cột là các thuộc tính và mỗi dòng được gọi là bộ.
- Một bộ của quan hệ $Q(A_1, A_2, ..., A_n)$ là $q = (a_1, a_2, ..., a_n)$ với $\forall a_i \in Dom(A_i)$
- **Ví dụ**: HOCVIEN(Mahv, Hoten, Ngsinh, Noisinh) có q=(1003,Nguyen Van Lam, 1/1/1987,Dong Nai) nghĩa là học viên có mã số là 1003, họ tên là Nguyen Van Lam, sinh ngày 1/1/1987 ở Dong Nai

2.4 Thể hiện của quan hệ (instance)

- **Định nghĩa**: thể hiện của một quan hệ là tập hợp các bộ giá trị của quan hệ tại một thời điểm.
- **Ký hiệu:** thể hiện của quan hệ Q là T_Q
- **Ví dụ**: $T_{HOCVIEN}$ là thể hiện của quan hệ HOCVIEN tại thời điểm hiện tại gồm có các bộ như sau:

HOCVIEN				
Mahv	HoTen	Gioitinh	Noisinh	Malop
K1103	**Ha Duy Lap**	**Nam**	**Nghe An**	**K11**
K1102	**Tran Ngoc Han**	**Nu**	**Kien Giang**	**K11**
K1104	**Tran Ngoc Linh**	**Nu**	**Tay Ninh**	**K11**

2.5 Tân từ

- **Định nghĩa**: tân từ là một quy tắc dùng để mô tả một quan hệ.
- **Ký hiệu**: ||Q||
- **Ví dụ**: THI (Mahv, Mamh, Lanthi, Diem)
- ||THI||: mỗi học viên được phép thi một môn học nhiều lần, mỗi lần thi lưu trữ học viên nào thi môn gì? lần thi thứ mấy? và điểm là bao nhiêu?

2.6 Phép chiếu (1)

- **Phép chiếu** : Dùng để trích giá trị của một số thuộc tính trong danh sách các thuộc tính của quan hệ.
- **Ký hiệu**: phép chiếu của quan hệ R lên tập thuộc tính X là ***R[X]*** ***hoặc*** ***R.X***.
- **Ví dụ**:

HOCVIEN				
Mahv	HoTen	Gioitinh	Noisinh	Malop
K1103	**Ha Duy Lap**	**Nam**	**Nghe An**	**K11**
K1102	**Tran Ngoc Han**	**Nu**	**Kien Giang**	**K11**
K1104	**Tran Ngoc Linh**	**Nu**	**Tay Ninh**	**K11**

- hv_1=
- hv_2 =
- hv_3 =

2.6 Phép chiếu (2)

- Phép chiếu của quan hệ HOCVIEN lên thuộc tính NoiSinh của quan hệ HOCVIEN:

HOCVIEN[Noisinh] = {‘Nghe An’,’Kien Giang’,’Tay Ninh’}

HOCVIEN				
Mahv	HoTen	Gioitinh	Noisinh	Malop
K1103	**Ha Duy Lap**	**Nam**	**Nghe An**	**K11**
K1102	**Tran Ngoc Han**	**Nu**	**Kien Giang**	**K11**
K1104	**Tran Ngoc Linh**	**Nu**	**Tay Ninh**	**K11**

2.6 Phép chiếu (3)

- Phép chiếu lên 1 tập thuộc tính X={Hoten,Noisinh} của quan hệ HOCVIEN

HOCVIEN[Hoten, Noisinh] = {(‘Ha Duy Lap’, ‘Nghe An’),(‘Tran Ngoc Han’, ‘Kien Giang’),(‘Tran Ngoc Linh’,’Tay Ninh’)}

HOCVIEN				
Mahv	HoTen	Gioitinh	Noisinh	Malop
K1103	**Ha Duy Lap**	**Nam**	**Nghe An**	**K11**
K1102	**Tran Ngoc Han**	**Nu**	**Kien Giang**	**K11**
K1104	**Tran Ngoc Linh**	**Nu**	**Tay Ninh**	**K11**

2.6 Phép chiếu (4)

- **Chiếu của một bộ lên tập thuộc tính:** dùng để trích chọn các giá trị cụ thể của bộ giá trị đó theo các thuộc tính được chỉ ra trong danh sách thuộc tính của một quan hệ.
- **Ký hiệu**: chiếu của một bộ giá trị t lên tập thuộc tính X của quan hệ R là $t_R[X]$ hoặc $t[X]$. Nếu X có 1 thuộc tính $t_R.X$
- **Ví dụ**: cho quan hệ HOCVIEN với tập thuộc tính $HOCVIEN^+$={Mahv,Hoten,Gioitinh,Noisinh,Malop}, chứa 3 bộ giá trị hv_1,hv_2 và hv_3

2.6 Phép chiếu (5)

- Phép chiếu 1 bộ lên 1 thuộc tính
  - hv_1[Hoten] = ('Ha Duy Lap')

	HOCVIEN				
	Mahv	HoTen	Gioitinh	Noisinh	Malop
hv_1=	**K1103**	**Ha Duy Lap**	**Nam**	**Nghe An**	**K11**
hv_2=	**K1102**	**Tran Ngoc Han**	**Nu**	**Kien Giang**	**K11**
hv_3=	**K1104**	**Tran Ngoc Linh**	**Nu**	**Tay Ninh**	**K11**

2.6 Phép chiếu (6)

- Phép chiếu 1 bộ lên 1 tập thuộc tính
  - tập thuộc tính X={Hoten, Gioitinh}
  - $hv_2[X]$ = ('Tran Ngoc Han','Nu')

- hv_1 =
- hv_2 =
- hv_3 =

HOCVIEN				
Mahv	HoTen	Gioitinh	Noisinh	Malop
K1103	**Ha Duy Lap**	**Nam**	**Nghe An**	**K11**
K1102	**Tran Ngoc Han**	**Nu**	**Kien Giang**	**K11**
K1104	**Tran Ngoc Linh**	**Nu**	**Tay Ninh**	**K11**

Câu hỏi ghi nhớ

- **Trong mô hình quan hệ của Codd, quan hệ là gì, bộ là gì, tân từ và thể hiện. Biểu diễn quan hệ, bộ, thể hiện và tân từ cho trường hợp sau:**
- **Lớp học có mã lớp, tên lớp trưởng, sĩ số, ghi chú. Và hiện tại chỉ có 2 lớp cần quản lý: lớp thứ nhất mã M1, lớp trưởng tên A, sỉ số 61, ghi chú là đi học đều, lớp thứ hai có mã M2, lớp trưởng tên B, sỉ số 69 ghi chú là hay vắng.**

2.7 Khóa

2.7.1 Siêu khóa (super key)

2.7.2 Khóa (key)

2.7.3 Khóa chính (primary key)

2.7.4 Khóa tương đương

2.7.5 Khóa ngoại (foreign key)

2.7.1 Siêu khóa (super key) (1)

- ***Siêu khóa*** : là một tập con các thuộc tính của Q^+ mà giá trị của chúng có thể phân biệt 2 bộ khác nhau trong cùng một thể hiện $\boldsymbol{T}_Q$ bất kỳ.

  Nghĩa là: $\forall\ t_1, t_2 \in \boldsymbol{T}_Q,\ t_1[K] \neq t_2[K] \Leftrightarrow$ K là siêu khóa của Q.
- Một quan hệ có ít nhất một siêu khóa (Q^+) và có thể có nhiều siêu khóa.

2.7.1 Siêu khóa (super key) (2)

- Ví dụ: các siêu khóa của quan hệ HOCVIEN là: {Mahv};{Mahv,Hoten};{Hoten};{Noisinh,Hoten} …

HOCVIEN				
Mahv	HoTen	Gioitinh	Noisinh	Malop
K1103	**Ha Duy Lap**	**Nam**	**Nghe An**	**K11**
K1102	**Tran Ngoc Han**	**Nu**	**Kien Giang**	**K11**
K1104	**Tran Ngoc Linh**	**Nu**	**Tay Ninh**	**K11**
K1105	**Tran Minh Long**	**Nam**	**TpHCM**	**K11**
K1106	**Le Nhat Minh**	**Nam**	**TpHCM**	**K11**

2.7.2 Khóa (key) (1)

Khóa : K là khóa của quan hệ R, thỏa mãn 2 điều kiện:

- K là một siêu khóa.
- K là siêu khóa "*nhỏ nhất*" (chứa ít thuộc tính nhất và khác rỗng) nghĩa là

$$\neg\exists K_1 \subset K,\ K_1 \neq \varnothing \text{ sao cho } K_1 \text{ là siêu khóa.}$$

- Thuộc tính tham gia vào một khóa gọi là *thuộc tính khóa*, ngược lại là *thuộc tính không khóa*.

2.7.2 Khóa (key) (2)

- **Ví dụ**: các siêu khóa của quan hệ HOCVIEN là: {Mahv};{Mahv,Hoten};{Hoten};{Hoten,Gioitinh}; {Noisinh,Hoten};{Mahv,Hoten,Gioitinh,Noisinh}…

  => thì khóa của quan hệ HOCVIEN có thể là {Mahv}; {Hoten}
- **Ví dụ**: khóa của quan hệ GIANGDAY (Malop, Mamh, Magv, HocKy, Nam) là K={Malop,Mamh}. Thuộc tính khóa sẽ là: Mamh,Malop. Thuộc tính không khóa sẽ là Magv, HocKy, Nam.

2.7.3 Khóa chính (primary key)

- Khi cài đặt trên một DBMS cụ thể, nếu quan hệ có nhiều hơn một ***khóa***, ta chỉ được chọn một và gọi là ***khóa chính***
- **Ký hiệu**: các *thuộc tính nằm trong khóa chính* khi liệt kê trong quan hệ phải được <u>gạch dưới</u>.
- **Ví dụ**:
  - HOCVIEN (<u>Mahv</u>,Hoten,Gioitinh,Noisinh,Malop)
  - GIANGDAY(<u>Mamh,Malop</u>,Magv,Hocky,Nam)

2.7.4 Khóa tương đương

- Các khóa còn lại (không được chọn làm khóa chính) gọi là khóa tương đương.
- **Ví dụ:** trong hai khóa {Mahv},{Hoten} thì khóa chính là {Mahv}, khóa tương đương là {Hoten}

2.7.5 Khóa ngoại (1)

- Cho R(U), S(V). $K_1 \subseteq U$ là khóa chính của R, $K_2 \subseteq V$
- Ta nói **K_2 là khóa ngoại** của S *tham chiếu đến* khóa chính K_1 của R nếu thỏa các điều kiện sau:
  - K_1 và K_2 có cùng số lượng thuộc tính và ngữ nghĩa của các thuộc tính trong K_1 và K_2 cũng giống nhau.
  - Giữa R và S tồn tại mối quan hệ 1-n trên K_1 và K_2,
  - $\forall s \in S, \quad !\exists r \in R$ sao cho $r.K_1 = s.K_2$

2.7.5 Khóa ngoại (2)

- Ví dụ, cho 2 quan hệ

LOP (Malop,Tenlop,Siso,Khoahoc)

HOCVIEN (Mahv,Hoten,Gioitinh,Noisinh,Malop)

- Thuộc tính Malop trong quan hệ LOP là khóa chính của quan hệ LOP. Thuộc tính *Malop trong quan hệ HOCVIEN là khóa ngoại, tham chiếu đến Malop trong quan hệ LOP*

2.7.5 Khóa ngoại (3)

HOCVIEN				
Mahv	HoTen	Gioitinh	Noisinh	Malop
K1103	**Ha Duy Lap**	**Nam**	**Nghe An**	**K11**
K1102	**Tran Ngoc Han**	**Nu**	**Kien Giang**	**K11**
K1104	**Tran Ngoc Linh**	**Nu**	**Tay Ninh**	**K11**
K1105	**Tran Minh Long**	**Nam**	**TpHCM**	**K11**
K1106	**Le Nhat Minh**	**Nam**	**TpHCM**	**K11**

LOP				
Malop	Tenlop	Trglop	Siso	Magvcn
K11	**Lop 1 khoa 1**	**K1106**	**11**	**GV07**
K12	**Lop 2 khoa 1**	**K1205**	**12**	**GV09**
K13	**Lop 3 khoa 1**	**K1305**	**12**	**GV14**

Lược đồ CSDL QLSV

HOCVIEN (MAHV, HO, TEN, NGSINH, GIOITINH, NOISINH, MALOP)

Tân từ: mỗi học viên phân biệt với nhau bằng mã học viên, lưu trữ họ tên, ngày sinh, giới tính, nơi sinh, thuộc lớp nào.

LOP (MALOP, TENLOP, TRGLOP, SISO, MAGVCN)

Tân từ: mỗi lớp gồm có mã lớp, tên lớp, học viên làm lớp trưởng của lớp, sỉ số lớp và giáo viên chủ nhiệm.

KHOA (MAKHOA, TENKHOA, NGTLAP, TRGKHOA)

Tân từ: mỗi khoa cần lưu trữ mã khoa, tên khoa, ngày thành lập khoa và trưởng khoa (cũng là một giáo viên thuộc khoa).

MONHOC (MAMH, TENMH, TCLT, TCTH, MAKHOA)

Tân từ: mỗi môn học cần lưu trữ tên môn học, số tín chỉ lý thuyết, số tín chỉ thực hành và khoa nào phụ trách.

DIEUKIEN (MAMH, MAMH_TRUOC)

Tân từ: có những môn học học viên phải có kiến thức từ một số môn học trước.

Lược đồ CSDL quản lý thi học kỳ

GIAOVIEN(MAGV,HOTEN,HOCVI,HOCHAM,GIOITINH,NGSINH,NGVL, HESO, MUCLUONG, MAKHOA)

Tân từ: mã giáo viên để phân biệt giữa các giáo viên, cần lưu trữ họ tên, học vị, học hàm, giới tính, ngày sinh, ngày vào làm, hệ số, mức lương và thuộc một khoa.

GIANGDAY(MALOP,MAMH,MAGV,HOCKY, NAM,TUNGAY,DENNGAY)

Tân từ: mỗi học kỳ của năm học sẽ phân công giảng dạy: lớp nào học môn gì do giáo viên nào phụ trách.

KETQUATHI (MAHV, MAMH, LANTHI, NGTHI, DIEM, KQUA)

Tân từ: lưu trữ kết quả thi của học viên: học viên nào thi môn học gì, lần thi thứ mấy, ngày thi là ngày nào, điểm thi bao nhiêu và kết quả là đạt hay không đạt.

Câu hỏi và Ôn tập

- **Kết quả khảo sát hiện trạng hệ thống quản lý nhân sự tiền lương tại một doanh nghiệp thu thập được các thông tin sau:**
- **Mỗi** *nhân viên* **có một mã nhân viên duy nhất, và có họ tên, ngày tháng năm sinh và được lãnh lương theo hệ bậc lương của mình. Đồng thời một nhân viên thì chỉ thuộc biên chế của một và chỉ một phòng ban. Ví dụ: Nhân viên tên A có mã số NV1 thuộc biên chế phòng Tổng Hợp (với mã phòng TH) sinh năm 1986 lãnh lương chuyên viên chính.**
- **Thông tin cần quản lý về** *phòng ban* **bao gồm: tên phòng ban, mã phòng ban và số lương nhân viên và người trưởng phòng.Ví dụ: phòng Tổng Hợp có mã TH, số lượng nhân viên là 5 có trưởng phòng là Nhân Viên A.**
- *Mức lương* **gồm tên, số tiền và mô tả yêu câu công việc. Ví dụ: Mức lương chuyên viên chính là 2,5 triệu VNĐ, cần tối thiểu 5 năm kinh nghiệm.**

- **Vẽ Lược đồ CSDL Nhân Sự Tiền Lương nói trên**

Phụ thuộc hàm

- **Phụ thuộc hàm (PTH) thể hiện sự phụ thuộc của một tập thuộc tính (Y) đối với một tập thuộc tính khác(X)**
  - Định nghĩa dựa trên những ngữ nghĩa, qui tắc tìm hiểu được từ môi trường ứng dụng
  - Ký hiệu $X \rightarrow Y$
- **Cho quan hệ Q(X, Y, Z), với X, Y, Z là các tập thuộc tính, $X \neq \varnothing$, $Y \neq \varnothing$**
  - Một thể hiện TQ của Q *thỏa* PTH $X \rightarrow Y$ nếu:

    $\forall q,q' \in TQ,\ q.X = q'.X \Rightarrow q.Y = q'.Y$

Phụ thuộc hàm

- TQ *vi phạm* PTH X→Y nếu:

  $\exists q,q' \in TQ$: $q.X = q'.X$ và $q.Y \neq q'.Y$

- PTH X→Y được gọi là *định nghĩa trên Q* nếu ∀TQ là thể hiện của Q, TQ thỏa PTH này

- PTH X→Y gọi là *phụ thuộc hàm hiển nhiên* ⇔ $Y \subseteq X$

Phụ thuộc hàm

Xét lịch xếp lớp của một cơ sở giảng dạy trong một ngày, ta có các phụ thuộc hàm sau:

- (1) GV, Giờ → Lớp

  (nếu biết giảng viên và giờ dạy, ta sẽ biết được lớp mà giảng viên dạy vào giờ đó)

- (2) Giờ, Lớp → Phòng

  (Cho một giờ học và lớp học cụ thể, ta sẽ biết được lớp đang học phòng nào vào giờ đó)

⇒ *Nếu biết giảng viên và giờ dạy, ta sẽ biết Phòng mà giảng viên dạy vào giờ đó*

⇒ (3) GV,Giờ → Phòng

(3) Là *hệ quả* của (1) và (2)

Phụ thuộc hàm

Cho F là tập các PTH định nghĩa trên Q

- PTH f là *hệ quả* của F, ký hiệu $F \models f$ nếu f được thỏa trong tất cả các thể hiện TQ của Q
- Tập tất cả các phụ thuộc hàm hệ quả của F gọi là *bao đóng* (closure) của F, ký hiệu F^+ ($F \subseteq F^+$)

Làm sao xác định F^+?

Phụ thuộc hàm

- **f là một PTH được *suy dẫn* từ F, ký hiệu $F \vdash f$, nếu:**
  - Tồn tại một chuỗi phụ thuộc hàm $f_1, f_2, \ldots f_n$, với
    - $f_n = f$
    - $f_i \in F$ hoặc được suy từ những phụ thuộc hàm f_j, j=1..i-1 nhờ vào *luật dẫn*
  - F' là tập các PTH suy dẫn từ F nhờ vào tập luật dẫn R ($F \subseteq F'$)
  - Tập luật dẫn R là *hợp lệ và đầy đủ* nếu và chỉ nếu $F' = F^+$

Phụ thuộc hàm

- **Gồm 3 luật dẫn:**
  - (FD1) Luật phản xạ:

    $\forall Y \subseteq X,\ X \rightarrow Y$
  - (FD2) Luật cộng:

    Nếu $X \rightarrow Y$ và $Z \subseteq W$ thì $X,W \rightarrow Y,Z$
  - (FD3) Luật bắc cầu:

    Nếu $X \rightarrow Y$ và $Y \rightarrow Z$ thì $X \rightarrow Z$
- **Hệ tiên đề Amstrong là một tập luật dẫn hợp lệ và đầy đủ**
- **Là cơ sở để tính F^+**

Phụ thuộc hàm

Từ các luật dẫn trong tiên đề Amstrong ta có thể suy ra các luật dẫn khác, một số sau đây thường được sử dụng:

- Luật bắc cầu giả:

  Nếu $X \rightarrow Y$ và $Y,W \rightarrow Z$ thì $X,W \rightarrow Z$
- Luật hội:

  Nếu $X \rightarrow Y$ và $X \rightarrow Z$ thì $X \rightarrow Y,Z$
- Luật phân rã:

  Nếu $X \rightarrow Y$ và $Z \subseteq Y$ thì $X \rightarrow Z$

Ghi chú: X,Y hay XY có nghĩa là $X \cup Y$

Phụ thuộc hàm

- F^+ thường rất lớn
- Thực tế: cho một phụ thuộc hàm f: X→Y, xác định xem f có thuộc F^+(có thỏa với mọi thể hiện của quan hệ Q?)
- →Có cần phải xác định F^+ để trả lời câu hỏi này?

Phụ thuộc hàm

- Bao đóng của tập thuộc tính X nhờ vào tập phụ thuộc F, ký hiệu X^+_F, được định nghĩa:

$$X^+_F = \{Y \mid X \rightarrow Y \text{ được suy dẫn từ } F\}$$

$\rightarrow X \subseteq X^+_F$;

$\rightarrow X^+_F \subseteq Q^+$

- Khái niệm bao đóng của một tập thuộc tính được sử dụng để kiểm tra xem một phụ thuộc hàm f có được suy dẫn từ F hay không ($f \in F^+$?)

Phụ thuộc hàm

Vào:

- Q^+ : tập các thuộc tính của quan hệ Q (hữu hạn)
- F : tập các phụ thuộc hàm định nghĩa trên Q
- $X \subseteq Q^+$

Ra: X^+_F

Thuật toán:

1. $X^+_F := X$
2. Lặp

   { Nếu ($\exists$ f: U→V thuộc F | $U \subseteq X^+_F$)

   $X^+_F := X^+_F \cup V$

   } cho đến khi ($X^+_F = Q^+$ hoặc X^+_F không còn thay đổi nữa)

Phụ thuộc hàm

Bổ đề (2.1):

- Cho f: X→ Y

$$_F \vdash f \Leftrightarrow Y \subseteq X^+_F$$

- CM: Y là tập thuộc tính, ta viết Y = {A_1, A_2,…A_n}
  - Chiều thuận:

    Ta có X → Y được suy dẫn từ F

    Theo luật phân rã: X → A_i, ∀i =1..n

    ⇒ Ai ∈ X^+_F , ∀i =1..n

    ⇒ Y ⊆ X^+_F (1)
  - Chiều nghịch:

    Ta có Y ⊆ X^+_F , suy ra $_F \vdash f_i$: X → Ai, ∀i =1..n

    ⇒PTH X → Y được suy dẫn từ F (theo luật hội) (2)

  (1) và (2) : ĐPCM

Bài toán thành viên:

- Cho F là tập PTH định nghĩa trên Q
- f: X→ Y là một phụ thuộc hàm mới nhận diện

  Bài toán: $f \in F^+$?

  Theo bổ đề 2.1, để chứng minh $f \in F^+$, ta chỉ cần chứng minh $Y \subseteq X^+_F$

→ Để giải quyết bài toán thành viên, chỉ cần xác định X^+_F, không cần xác định F^+

Cho Q(ABCDEG)

F = { AB→C; D→EG; C→A; BE →C; BC →D

CG → BD; ACD→B; CE → AG }

Phụ thuộc hàm BD →A có đúng với mọi thể hiện của q không?

BD → ???

Theo luat D→ EG {B D E G}

Theo luat BE → C {B D E G C}

Theo luat BC → D, CG→BD, CE→AG {B D E G C A ~~G~~}

Phụ thuộc hàm

Tập phụ thuộc hàm tương đương

- Định nghĩa: Hai tập PTH F và F' gọi là tương đương, ký hiệu $F \equiv F'$ nếu $F^+ = F'^+$
- Bổ đề 2.2: $F \equiv F' \Leftrightarrow F \models F'$ và $F' \models F$

  $\Leftrightarrow F \vdash F'$ và $F' \vdash F$

Ví dụ 2.2: Xét tính tương đương của hai tập phụ thuộc hàm sau, định nghĩa trên Q(ABCDE)

$F = \{A \rightarrow BC;\ A \rightarrow D;\ CD \rightarrow E\}$

$F' = \{A \rightarrow BCE;\ A \rightarrow ABD;\ CD \rightarrow E\}$

Phụ thuộc hàm

- **Định nghĩa:**

  Một tập phụ thuộc hàm F' được gọi là phủ của F nếu $F' \equiv F$

- **Ví dụ:**

  Trong ví dụ 2.2, F' là một phủ của F.

Phụ thuộc hàm

- **Phụ thuộc hàm đầy đủ**
  - Định nghĩa:

    Cho F là tập các phụ thuộc hàm và $f: X \rightarrow Y \in F$

    f là một phụ thuộc hàm đầy đủ trong F nếu:

    $\neg\exists X' \subset X$ sao cho: $F \equiv (F - \{X \rightarrow Y\} \cup \{X' \rightarrow Y\})$,

    nghĩa là, từ F không suy dẫn ra được PTH $X' \rightarrow Y$
  - Ví dụ 2.3

    $F = \{ A \rightarrow BCD;\ BCD \rightarrow E;\ CD \rightarrow I;\ I \rightarrow E \}$

    $BCD \rightarrow E$ là phụ thuộc hàm không đầy đủ,

    vì $F \vdash CD \rightarrow E$

Phụ thuộc hàm

Định nghĩa:

Cho F là tập các PTH mà vế phải chỉ chứa một thuộc tính (dùng luật phân rã để biến đổi các PTH mà vế phải có nhiều hơn một thuộc tính)

G là một phủ tối thiểu của F nếu G là một phủ của F và thoả hai điều kiện:

(1) G chỉ chứa các PTH đầy đủ

(2) $\nexists f: X \rightarrow A \in G$ sao cho $G \equiv (G - \{X \rightarrow A\})$

Ký hiệu: G = PTT(F)

Thuật toán tìm phủ tối thiểu

- Vào: Tập phụ thuộc hàm F
- Ra : PTT(F)
- Thực hiện:
  1. Nếu ∃ f: X → Y ∈ F với card(Y) >1, phân rã f.
  2. Thay các PTH không đầy đủ bằng PTH đầy đủ
  3. Loại bỏ các phụ thuộc hàm thừa (có thể được suy dẫn từ các PTH còn lại trong F)

Ghi chú: Một tập PTH F có thể có nhiều hơn 1 phủ tối thiểu

Ví dụ 2.4:

Cho F = {A→BC; B →AC; C →A}

Tìm phủ tối thiểu của F?

Ví dụ 2.5:

- Cho F = { AB→C; A → B; B → A}

  Tìm phủ tối thiểu của F

Phụ thuộc hàm

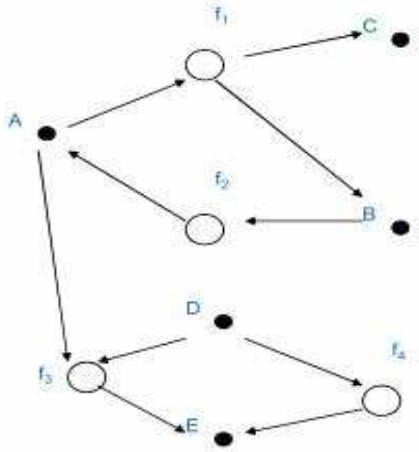
- **Đồ thị phụ thuộc hàm là một đồ thị vô hướng, với :**
  - Một tập nút tượng trưng cho tập PTH, ký hiệu O với tên PTH bên cạnh.
  - Một tập nút tượng trưng cho các thuộc tính, ký hiệu • với tên thuộc tính bên cạnh.
  - Một tập cung có hướng nối một nút PTH(thuộc tính) đến một nút thuộc tính (PTH).
  - Một cung xuất phát từ nút thuộc tính A đến một nút PTH f, cùng với một cung từ nút PTH f đến nút thuộc tính B, biểu diễn cho PTH A→B

Khi F có nhiều PTT, đồ thị của F có chứa chu trình.

Phụ thuộc hàm

Cho F = {f_1:A→BC; f_2: B →A; f_3: AD →E; f_4:BD →E }

Đồ thị của F:



Phụ thuộc hàm

- **Tính chiếu**

  f: X→Y định nghĩa trên Q

  Lấy Q' = Q[W] với W ⊇X và W ∩Y <>0

  Khi đó, trên Q' ta có PTH f':X → W ∩Y

- **Tính phản chiếu:**

  f: X →Y định nghĩa trên Q'[W]

  Lấy Q(V) sao cho V⊃ W

  Khi đó, PTH X→Y định nghĩa trên cả Q

Phụ thuộc hàm

Định nghĩa lại khóa theo khái niệm phụ thuộc hàm:

- Cho quan hệ Q và F là tập PTH định nghĩa trên Q
- $K \subseteq Q^+$ là một khóa của Q nếu:
  - i. $f: K \rightarrow Q^+ \in F^+$ (hay $K^+_F = Q^+$)
  - ii. $\neg\exists K' \subset K \mid K' \rightarrow Q^+$

Phụ thuộc hàm

Thuật toán xác định khóa của quan hệ:

1. Xây dựng các tổ hợp có thể có từ Q^+
2. Tìm tập S chứa tất cả các tổ hợp $K \subseteq Q^+$ thỏa điều kiện (i), mỗi tổ hợp K như vậy là một siêu khóa của Q.
3. $\forall K \in S$

   Nếu $\exists K' \mid K' \subset K$ thì loại K ra khỏi S

– *Thực tế, kết hợp bước 2 và bước 3: bắt đầu xét từ những tổ hợp có ít phần tử nhất, nếu tìm được một tổ hợp K_i thỏa điều kiện (i) thì loại bỏ ngay các tổ hợp có chứa K_i*

Vấn đề: Số tổ hợp có thể có từ Q^+ sẽ rất lớn nếu Q^+ lớn → Cần giới hạn số tổ hợp cần khảo sát

Phụ thuộc hàm

Giới hạn số lượng tổ hợp:

– Thuộc tính nguồn:
  - A là một thuộc tính nguồn nếu $\neg\exists f: X \rightarrow Y \in F \mid A \in Y$
  - Trên đồ thị PTH, thuộc tính nguồn không có cung vào
  - Nhận xét: mọi thuộc tính nguồn phải xuất hiện trong mọi khóa của Q

– Thuộc tính đích:
  - B là một thuộc tính đích nếu $\neg\exists f: X \rightarrow Y \in F \mid B \in X$
  - Trên đồ thị PTH, thuộc tính đích chỉ có cung vào, không có cung ra.
  - Nhận xét: thuộc tính đích không xuất hiện trong bất kỳ khóa nào của Q

Phụ thuộc hàm

- Cho Q(ABCDEG) với

  $F = \{f_1: AD \rightarrow B;\ f_2: EG \rightarrow A;\ f_3: BC \rightarrow G\}$

  Xác định các khóa của Q

Phụ thuộc hàm

- *Phụ thuộc đa trị (multi-value dependency)*

Phụ thuộc hàm

- **Định nghĩa 1:**

Cho Q(XYZ) với:

- $\emptyset \neq X \subseteq Q^+, \emptyset \neq Y \subseteq Q^+$
- $Z = Q^+ - (X \cup Y) \neq \emptyset$

$X \rightarrow\rightarrow Y$ là một *phụ thuộc đa trị được định nghĩa trên Q* nếu mỗi giá trị x của X xác định duy nhất một tập giá trị $\{y_1, y_2, \ldots\}$ của Y, không phụ thuộc vào các giá trị z của Z.

Phụ thuộc hàm

Định nghĩa 2:

$X \rightarrow\rightarrow Y$ là một *phụ thuộc đa trị được định nghĩa trên Q (XYZ)*, với $X \neq \emptyset$, $Y \neq \emptyset$ và $Z \neq \emptyset$ nếu:

$\forall q1, q2 \in TQ$ sao cho: $q1.X = q2.X$ và $q1.Z \neq q2.Z$ thì:

$\exists\, q3, q4 \in TQ$ sao cho: $q3.X = q4.X = q1.X = q2.X$

và $q3.Y = q1.Y$ và $q3.Z = q2.Z$

và $q4.Y = q2.Y$ và $q4.Z = q1.Z$

Phụ thuộc hàm

- **Ví dụ 2.8**

NHÂN_VIÊN(Mã_NV, HọTên, Con_NV, BậcLương)

Ta có phụ thuộc đa trị MãNV →> Con_NV

Mã_NV	HọTên	ConNV	Bậc lương
NV001	Nguyễn Văn A	Nguyễn Văn A1	2.34
NV001	Nguyễn Văn A	Nguyễn Văn A2	2.34
NV002	Nguyễn Văn B	Nguyễn Văn B1	2.67

Phụ thuộc hàm

- **Ví dụ 2.9**

GIANG_DAY(ChuyênĐề, Sách, GV, SV)

Phụ thuộc đa trị ChuyênĐề →> Sách được định nghĩa trên quan hệ GIANG_DAY

ChuyênĐề	Sách	GV	SV
CSDL	S1	Thủy	A
CSDL	S2	Hoa	B
CSDL	S1	Hoa	B
CSDL	S2	Thủy	A

Phụ thuộc hàm

❖ **X →> Y là một *phụ thuộc đa trị hiển nhiên* trong Q nếu**
$X \cup Y = Q^+ (Z = \varnothing)$

Ví dụ 2.10

Xét quan hệ CD_SACH(ChuyênĐề, Sách)

Phụ thuộc đa trị ChuyênĐề →> Sách, và
Sách →> ChuyênĐề là hiển nhiên trên CD_SACH

ChuyênĐề	Sách
CSDL	S1
CSDL	S2
TK CSDL	S2
TK CSDL	S3
TK CSDL	S4

Phụ thuộc hàm

- **Tương tự phụ thuộc hàm, ta cũng có khái niệm phụ thuộc đa trị hệ quả và phụ thuộc đa trị suy dẫn từ một *tập phụ thuộc D***
- **Ký hiệu**

$D^+ = \{d \mid D \models d\}$

$= \{d \mid D \vdash d$ nhờ tập luật dẫn hợp lệ và đầy đủ$\}$

Phụ thuộc hàm

Cho $Q(Q^+)$ và $X, Y, W, V \subseteq Q+$

- *(MVD1) Luật bổ sung:*

  Nếu $X \rightarrow\rightarrow Y$ thì $X \rightarrow\rightarrow Q^+ - (X \cup Y)$

- *(MVD2) Luật cộng:*

  Nếu $X \rightarrow\rightarrow Y$ và $V \subseteq W$ thì $WX \rightarrow\rightarrow VY$

- *(MVD3) Luật bắc cầu:*

  Nếu $X \rightarrow\rightarrow Y$ và $Y \rightarrow\rightarrow Z$ thì $X \rightarrow\rightarrow (Z-Y)$

Phụ thuộc hàm

- *(FD_MVD1)* **Nếu $X \rightarrow Y$ thì $X \rightarrow\rightarrow Y$**
- *(FD_MVD2)* **Nếu $X \rightarrow\rightarrow Y$ và $W \rightarrow Z$, với $Z \subseteq Y$, $W \cap Y = \varnothing$**

  thì $X \rightarrow Z$

- **Tập các luật dẫn trên, bổ sung 3 luật dẫn trong hệ tiên đề Amstrong là tập luật dẫn đầy đủ**

Phụ thuộc hàm

Luật bắc cầu giả:

Nếu $X \rightarrow\rightarrow Y$ và $WY \rightarrow\rightarrow Z$ thì $WX \rightarrow\rightarrow (Z - WY)$

Luật hội:

Nếu $X \rightarrow\rightarrow Y$ và $X \rightarrow\rightarrow Z$ thì $X \rightarrow\rightarrow (YZ)$

Luật phân rã

Nếu $X \rightarrow\rightarrow Y$ và $X \rightarrow\rightarrow Z$ thì :

$X \rightarrow\rightarrow (Y \cap Z)$

$X \rightarrow\rightarrow (Y-Z)$

$Z \rightarrow\rightarrow (Z-Y)$

Phụ thuộc hàm

- **Bài toán thành viên đối với phụ thuộc đa trị**

  Cho d: $X \rightarrow\rightarrow Y$ là một phụ thuộc đa trị, kiểm tra $d \in D$?

- **Khái niệm tương tự bao đóng của tập thuộc tính X:** ***Tập phụ thuộc của X dựa trên D, ký hiệu PT(X)$_D$***

$$PT(X)_D = \{Y \subseteq Q \mid D \models (X \rightarrow\rightarrow Y)\}$$

- **$PT(X)_D$ là cơ sở để giải quyết bài toán thành viên đối với phụ thuộc đa trị.**

Phụ thuộc hàm

Lưu ý:

- Khác với phụ thuộc hàm, ta không thể phân rã tùy ý vế phải của phụ thuộc đa trị, vì vậy, mỗi phần tử của $PT(X)_D$ là 1 tập thuộc tính
- Do đó, số lượng phần tử của $PT(X)_D$ có thể sẽ rất lớn. Thuật toán tìm $PT(X)_D$ phải chú ý đến việc cho ra tập $PT(X)_D$ *tối thiểu*

Phụ thuộc hàm

Vào: $Q(Q^+)$, D, X ; Ra: $PT(X)_D$

0. $G_X := \varnothing$

1. Dùng luật bổ sung và luật cộng để xác định G_X:

$G_X := \{A\}$, $A \in X$

$\forall d : X \to Y' \in D$ với $X' \subseteq X$ thì: $G_X := G_X \cup \{Y', Q^+(X' \to Y')\}$

2. Tối thiểu G_X bằng cách áp dụng luật phân rã :

$\forall Y_1, Y_2 \in G_X$ thì:

Nếu $Y_1 \cap Y_2 \neq \varnothing$

Thì: thay Y_1, Y_2 trong G_X bằng $(Y_1 \cap Y_2)$, $(Y_1 - Y_2)$, $(Y_2 - Y_1)$

(chỉ thay vào các tập khác $\varnothing$)

Cuối Nếu

Cuối $\forall$

(Tập G_X sau bước 2 gọi là $CTT(G_X)$)

$PT(X)_D = CTT(G_X)$

Phụ thuộc hàm

3. Phân nhỏ các tập hợp trong $PT(X)_D$ bằng cách sử dụng luật bắc cầu:

$\forall d: W \rightarrow\rightarrow Z \in D$ thì:

nếu $\exists\{Y_1, Y_2, \ldots Y_n\} \subset PT(X)_D$ sao cho $W \cup Y = Y_1 \cup Y_2 \cup \ldots \cup Y_n$ thì

(Ta có: $X \rightarrow\rightarrow Y$ (do định nghĩa $PT(X)_D$)

$Y \rightarrow\rightarrow Z$ (luật cộng))

Nếu $\exists\{Y'_1, Y'_2, \ldots Y'_m\} \subset PT(X)_D$ sao cho $(Z-Y) = Y'_1 \cup Y'_2 \cup \ldots \cup Y'_m$

Thì $PT(X)_D := CTT(PT(X)_D \cup \{Z-Y\})$

Cuối Nếu

Cuối $\forall$

Phụ thuộc hàm

- **Tính chất chiếu của phụ thuộc đa trị**

  Cho $Q(Q^+)$ và $d: X \rightarrow\rightarrow Y$ định nghĩa trên Q

  Nếu có Z sao cho $X \subseteq Z \subseteq Q^+$

  Thì $X \rightarrow\rightarrow (Y \cap Z)$ là một phụ thuộc đa trị định nghĩa trên $Q(Z)$

- **Không có tính chất ngược lại (tính phản chiếu):**

  Nếu $d: X \rightarrow\rightarrow Y$ định nghĩa trên $Q'(Q'^+)$ với $Q'^+ \subseteq Q^+$ thì không nhất thiết d phải được định nghĩa trên Q.

Định nghĩa phụ thuộc đa trị tiềm tàng trong Q:

Cho d': X' →>Y' là một phụ thuộc đa trị tiềm tàng trong Q nếu:

1. d' được định nghĩa trên Q'(Q'$^+$) với Q'$^+$ ⊆ Q$^+$
2. ¬∃ d định nghĩa trên Q sao cho d' có được bằng cách chiếu d lên Q'$^+$

- **Ví dụ 2.11**

Cho quan hệ PHỤ_ĐẠO(ChuyênĐề, SV, TrợGiảng, SốGiờ)

D = {SV, TrợGiảng → SốGiờ; SV, TrợGiảng →> ChuyênĐề}

Thể hiện sau của PHỤ_ĐẠO thỏa D:

ChuyênĐề	SV	TrợGiảng	SốGiờ
LT_C	Hoa	Minh	0
LT_C	Hùng	Minh	12
LT_C	Lan	Minh	15
LT_C	Hoa	Nhân	0
LT_C	Hùng	Nhân	0
LT_C	Lan	Nhân	12
LT_Pas	Lan	Nhân	12
LT_Pas	Dung	Nhân	6

Ví dụ 2.11(tt)

Nhận xét:

Trong Q không có phụ thuộc đa trị ChuyênĐề →>SV và ChuyênĐề →>TrợGiảng

Tuy nhiên, hai phụ thuộc đa trị trên tồn tại trong quan hệ PHỤ_ĐẠO_2(ChuyênĐề, SV, TrợGiảng)

Sự tương ứng với mô hình ER

- **Mỗi quan hệ (bảng) tương ứng với một kiểu thực thể hoặc với một kiểu mối liên kết nhiều - nhiều.**
- **Mỗi hàng tương ứng với một thể hiện thực thể hoặc với một thể hiện mối liên kết nhiều - nhiều.**
- **Mỗi cột tương ứng với một thuộc tính.**
- **Từ *quan hệ* (*relation*) trong cơ sở dữ liệu quan hệ không có cùng nghĩa với từ *mối quan hệ* (*relationship*) trong mô hình *ER*.**

Lược đồ cơ sở dữ liệu

- **Lược đồ cơ sở dữ liệu**
  - *database schema*
  - *Lược đồ cơ sở dữ liệu* là một tập hợp các lược đồ quan hệ.
  - Trong một lược đồ cơ sở dữ liệu, các tên lược đồ quan hệ là duy nhất.

Lược đồ cơ sở dữ liệu:

Emp (***Empnum***, ***Name, Sal, Tax, Mgrnum, Deptnum***)
Dept (***Deptnum***, ***Name, Area, Mgrnum***)
Supplier (***Snum***, ***Name, City***)
Supply (***Snum, Pnum***, ***Deptnum, Quan***)

Ràng buộc toàn vẹn

- **Ràng buộc toàn vẹn**
  - *integrity constraint*
  - *Ràng buộc toàn vẹn* là một qui tắc mà tất cả các dữ liệu trong CSDL phải thỏa mãn qui tắc này.
- **Ràng buộc miền trị**
  - *domain constraint*
  - Các giá trị cho phép của một thuộc tính.
- **Toàn vẹn thực thể**
  - *entity integrity*
  - Thuộc tính khóa chính không có giá trị rỗng (*null value*).

Ràng buộc toàn vẹn

- **Qui tắc hoạt động**
  - *action assertion*
  - Các qui tắc nghiệp vụ (*business rule*).

Ràng buộc toàn vẹn

Attribute	*Domain Name*	*Description*	*Domain*
Customer_ID	Customer_IDs	Set of all possible customer IDs	character: size 5
Customer_Name	Customer_Names	Set of all possible customer names	character: size 25
Customer_Address	Customer_Addresses	Set of all possible customer addresses	character: size 30
City	Cities	Set of all possible cities	character: size 20
State	States	Set of all possible states	character: size 2
Postal_Code	Postal_Codes	Set of all possible postal zip codes	character: size 10
Order_ID	Order_IDs	Set of all possible order IDs	character: size 5
Order_Date	Order_Dates	Set of all possible order dates	date format mm/dd/yy
Product_ID	Product_IDs	Set of all possible product IDs	character: size 5
Product_Description	Product_Descriptions	Set of all possible product descriptions	character size 25
Product_Finish	Product_Finishes	Set of all possible product finishes	character: size 15
Standard_Price	Unit_Prices	Set of all possible unit prices	monetary: 6 digits
Product_Line_ID	Product_Line_IDs	Set of all possible product line IDs	integer: 3 digits
Ordered_Quantity	Quantities	Set of all possible ordered quantities	integer: 3 digits

Định nghĩa miền trị cho các thuộc tính

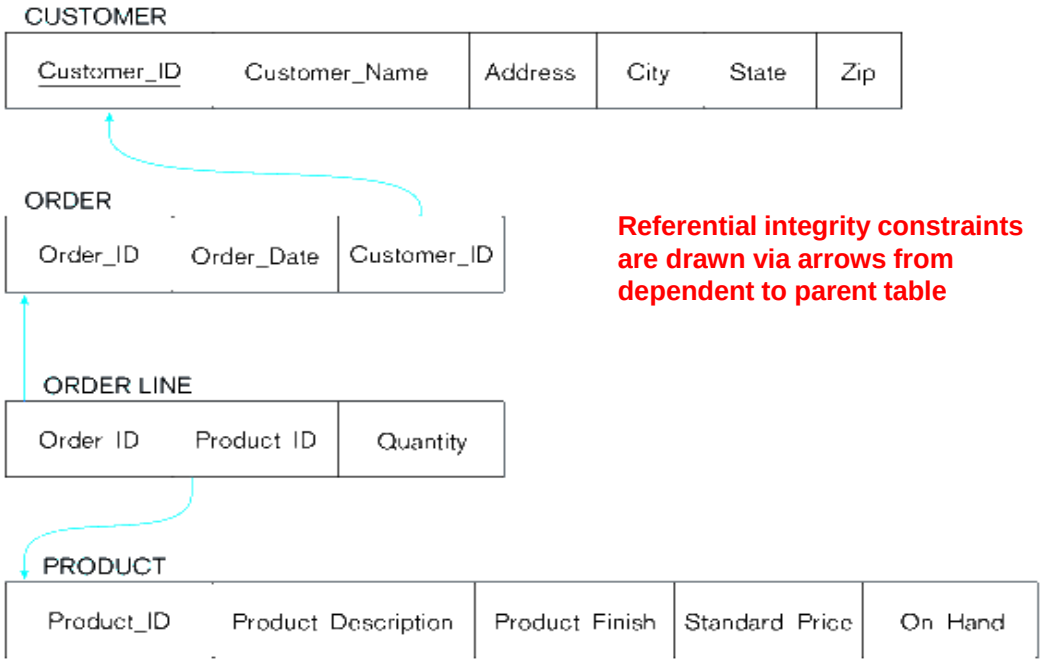
Ràng buộc toàn vẹn

- **Ràng buộc toàn vẹn tham chiếu**
  - *referential integrity constraint*
  - *Ràng buộc toàn vẹn tham chiếu* là một qui tắc mà tất cả các giá trị của khóa ngoại (nếu khác *null*) trong quan hệ bên phía *nhiều* phải có trong các giá trị của khóa chính trong quan hệ bên phía *một*.

Ràng buộc toàn vẹn

- **Ràng buộc toàn vẹn tham chiếu**
  - Qui tắc xóa các hàng dữ liệu
    - Hạn chế (*restrict*): không cho phép xóa các hàng bên phía cha (*parent*) nếu tồn tại các hàng liên quan bên phía phụ thuộc (*dependent*).
    - Tầng (*cascade*): tự động xóa các hàng bên phía phụ thuộc tương ứng với các hàng bên phía cha.
    - Gán null (*set-to-null*): gán *null* cho khóa ngoại của các hàng bên phía phụ thuộc tương ứng với các hàng bên phía cha. Không áp dụng cho các thực thể yếu.

Ràng buộc toàn vẹn



Referential integrity constraints are drawn via arrows from dependent to parent table

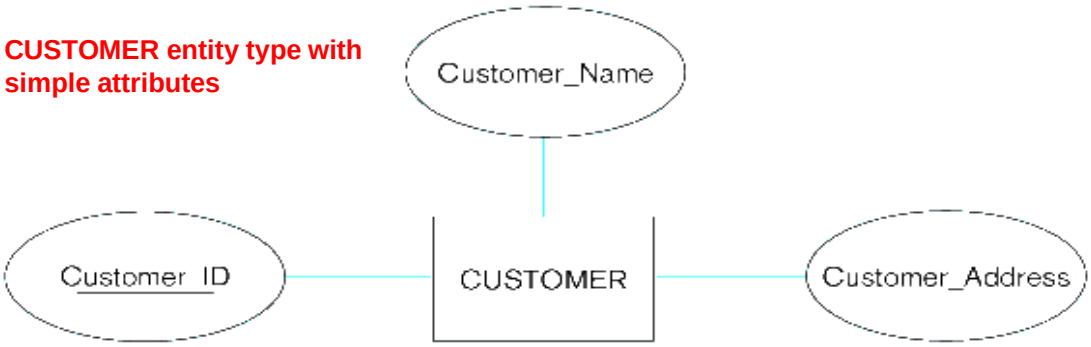
Ví dụ về ràng buộc toàn vẹn tham chiếu

Biến đổi ERD thành các quan hệ

- **Qui tắc 1: Biến đổi một kiểu thực thể thành một quan hệ.**
  - Đối với kiểu thực thể thông thường (*regular entity type*): khóa của quan hệ là khóa của kiểu thực thể.
  - Thuộc tính của quan hệ là thuộc tính của kiểu thực thể.
  - Quan hệ chỉ chứa các thuộc tính thành phần của thuộc tính phức hợp.
  - Quan hệ không chứa các thuộc tính đa trị.

Biến đổi ERD thành các quan hệ

CUSTOMER entity type with simple attributes



CUSTOMER relation

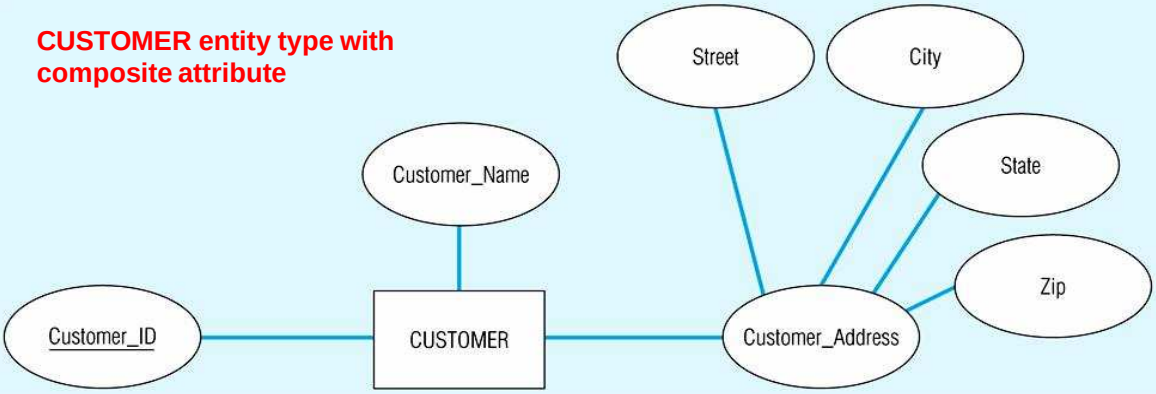
CUSTOMER

Customer_ID	Customer_Name	Customer_Address

Biến đổi kiểu thực thể thông thường

Biến đổi ERD thành các quan hệ

CUSTOMER entity type with composite attribute



CUSTOMER relation with address detail

CUSTOMER

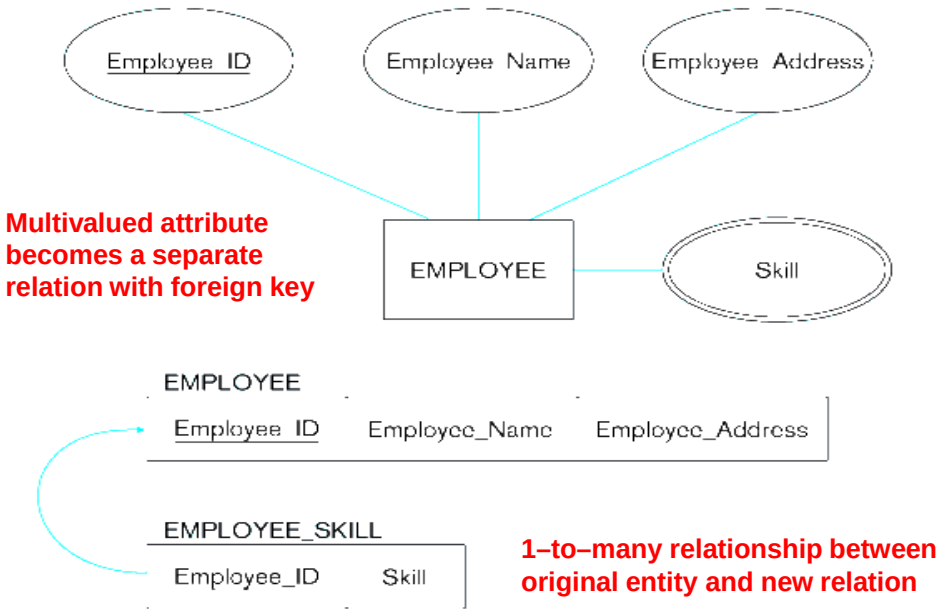
Customer_ID	Customer_Name	Street	City	State	Zip

Biến đổi thuộc tính phức hợp

Biến đổi ERD thành các quan hệ

- **Qui tắc 2: Biến đổi thuộc tính đa trị thành một quan hệ.**
  - Quan hệ chứa khóa của kiểu thực thể và thuộc tính đa trị.
  - Khóa của quan hệ gồm khóa của kiểu thực thể và thuộc tính đa trị.

Biến đổi ERD thành các quan hệ

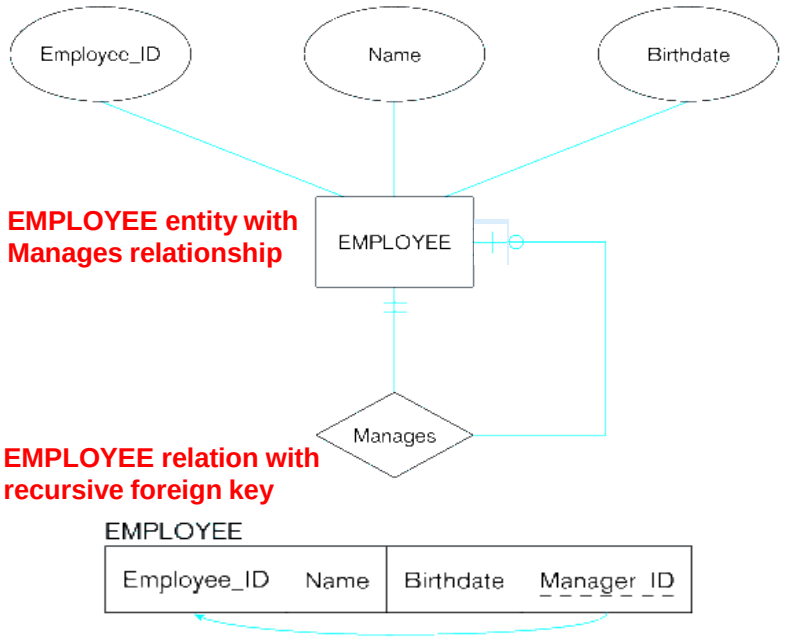


Biến đổi thuộc tính đa trị

Biến đổi ERD thành các quan hệ

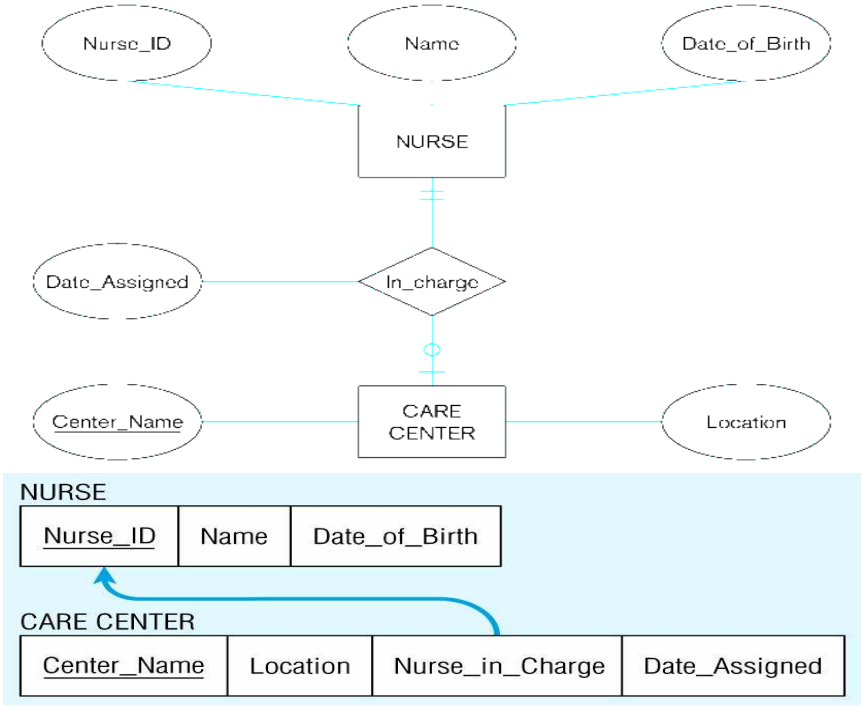
- **Qui tắc 3: Biểu diễn mối liên kết 1-ngôi hoặc 2-ngôi có lượng số một-một.**
  - Đặt khóa của kiểu thực thể bên phía bắt buộc và các thuộc tính của mối liên kết vào quan hệ của kiểu thực thể bên phía tùy chọn.

Biến đổi ERD thành các quan hệ



Biến đổi mối liên kết một ngôi có lượng số một - một

Biến đổi ERD thành các quan hệ

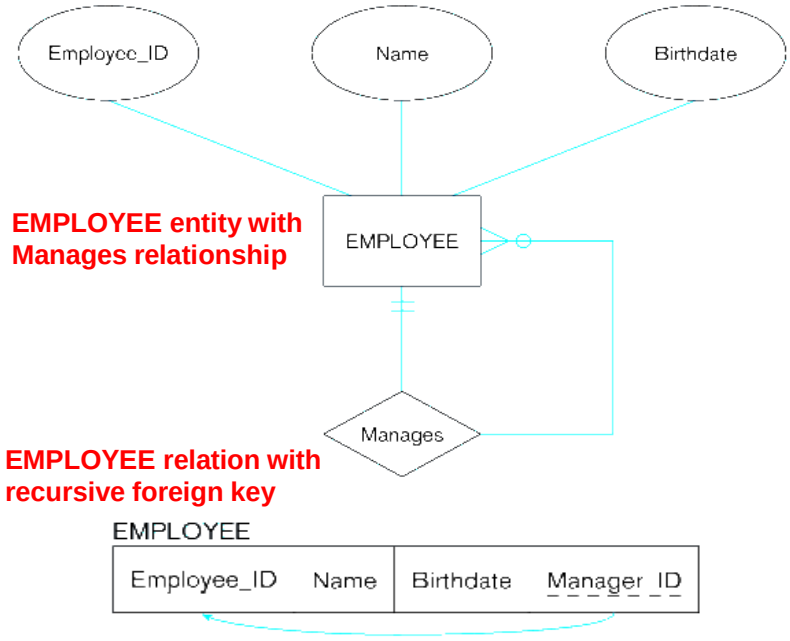


Biến đổi mối liên kết hai ngôi có lượng số một - một

Biến đổi ERD thành các quan hệ

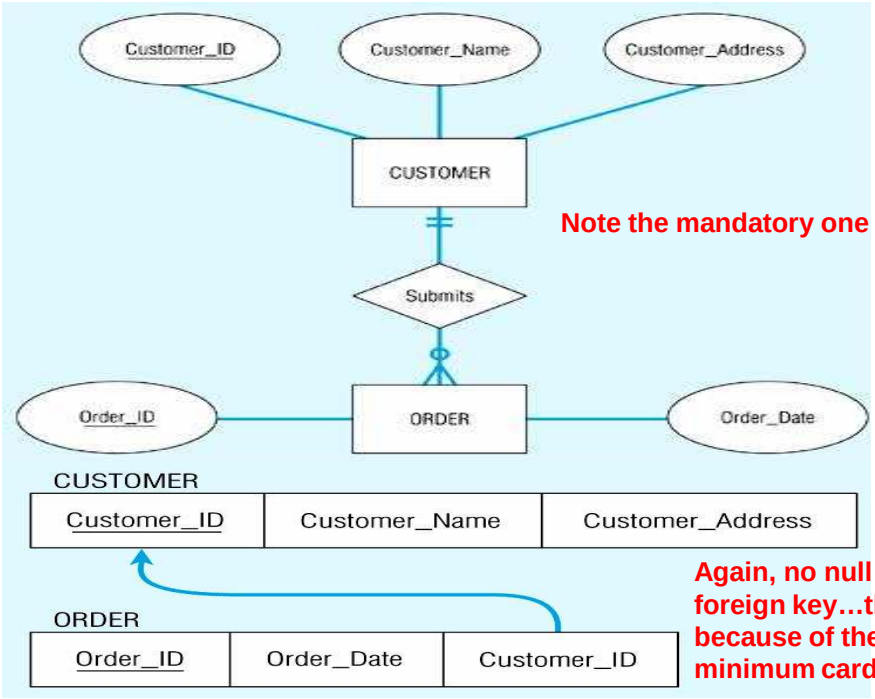
- **Qui tắc 4: Biểu diễn mối liên kết 1-ngôi hoặc 2-ngôi có lượng số một-nhiều.**
  - Đặt khóa của kiểu thực thể bên phía *một* và các thuộc tính của mối liên kết vào quan hệ của kiểu thực thể bên phía *nhiều*.

Biến đổi ERD thành các quan hệ



Biến đổi mối liên kết một ngôi có lượng số một - nhiều

Biến đổi ERD thành các quan hệ

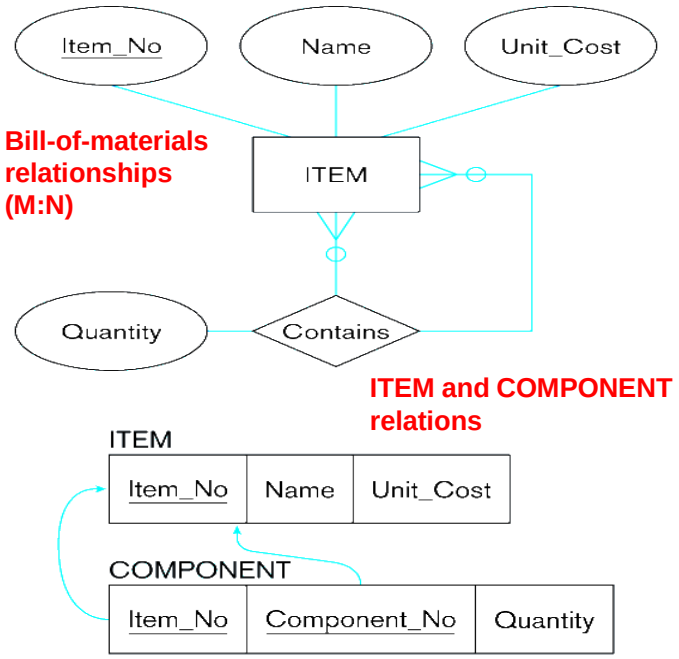


Biến đổi mối liên kết hai ngôi có lượng số một - nhiều

Biến đổi ERD thành các quan hệ

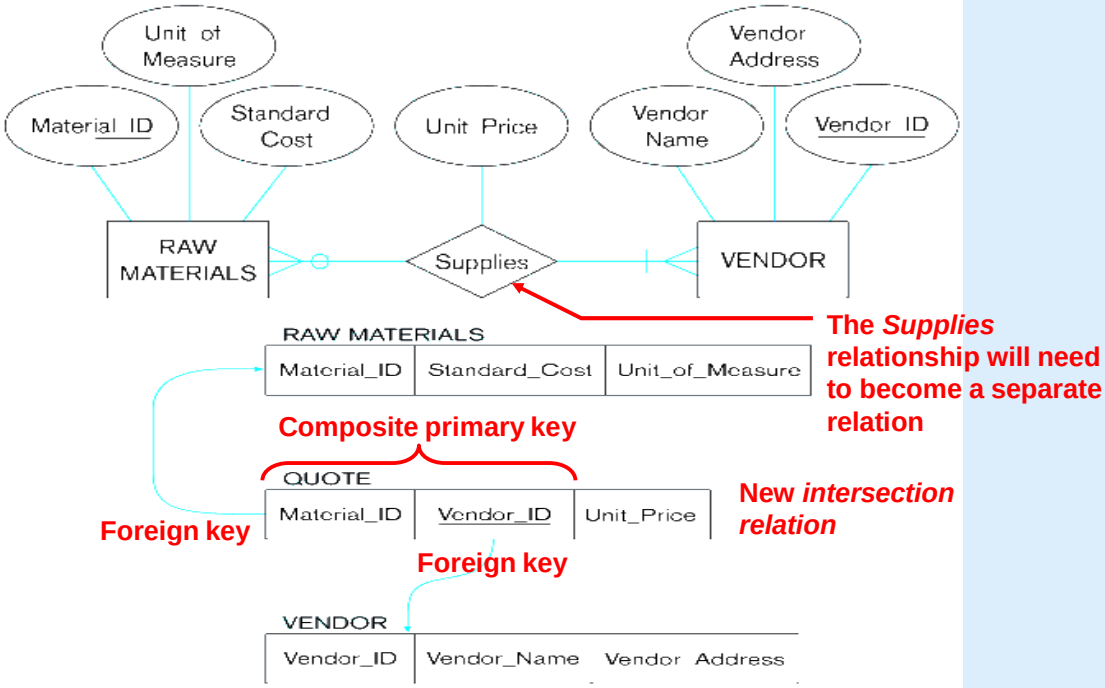
- **Qui tắc 5: Biến đổi mối liên kết 1-ngôi hoặc 2-ngôi có lượng số nhiều-nhiều thành một quan hệ.**
  - Quan hệ chứa các khóa của các kiểu thực thể tham gia vào mối liên kết.
  - Khóa của quan hệ gồm cả hai khóa của hai kiểu thực thể.
  - Thuộc tính của quan hệ là thuộc tính của mối liên kết.

Biến đổi ERD thành các quan hệ



Biến đổi mối liên kết một ngôi có lượng số nhiều - nhiều

Biến đổi ERD thành các quan hệ

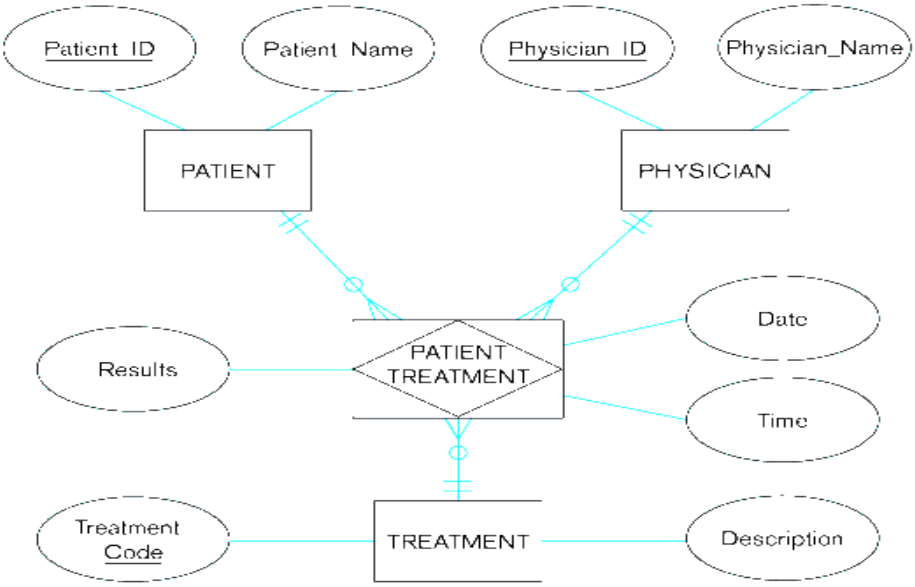


Biến đổi mối liên kết hai ngôi có lượng số nhiều - nhiều

Biến đổi ERD thành các quan hệ

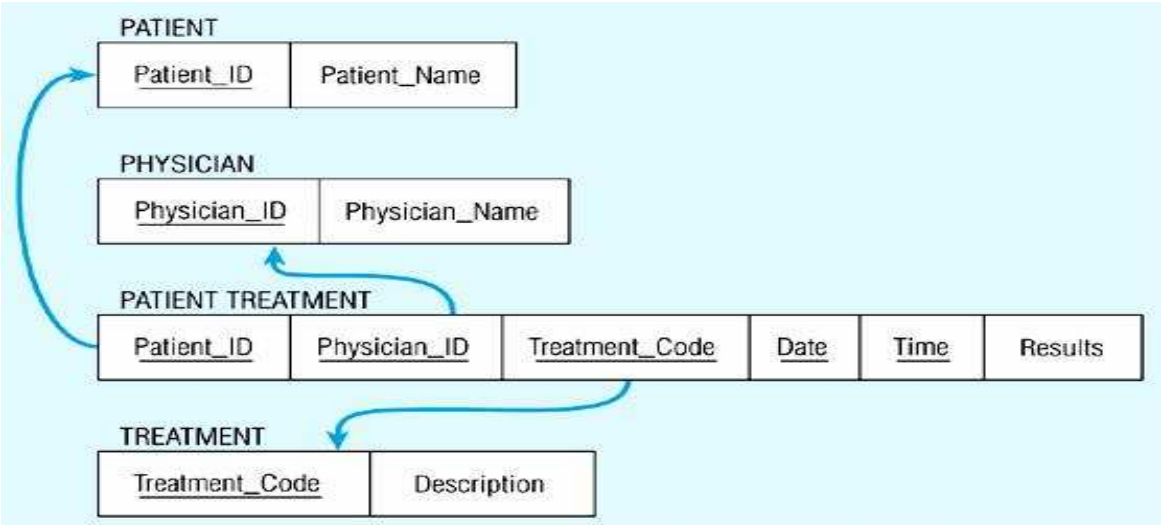
- **Qui tắc 6: Biến đổi mối liên kết 3-ngôi thành một quan hệ.**
  - Quan hệ chứa ba khóa của ba kiểu thực thể tham gia vào mối liên kết.
  - Mối liên kết có bao nhiêu kiểu thực thể bên phía *một* thì quan hệ có bấy nhiêu khóa: đối với một kiểu thực thể bên phía *một* thì khóa của quan hệ gồm cả hai khóa của hai kiểu thực thể còn lại. Nếu không có kiểu thực thể bên phía *một* thì khóa của quan hệ bao gồm cả ba khóa của ba kiểu thực thể.
  - Thuộc tính của quan hệ là thuộc tính của mối liên kết.

Biến đổi ERD thành các quan hệ



Biến đổi mối liên kết ba ngôi

Biến đổi ERD thành các quan hệ



Biến đổi mối liên kết ba ngôi

Chuẩn hóa dữ liệu

- **Chuẩn hóa dữ liệu**
  - *data normalization*
  - *Chuẩn hóa dữ liệu* là một quá trình thuận nghịch từng bước để thay thế tập hợp các quan hệ cho trước thành các quan hệ *có cấu trúc đơn giản hơn* và *chuẩn hơn*.
  - Chuẩn hóa dữ liệu nhằm để cải tiến một thiết kế CSDL *thỏa mãn các ràng buộc toàn vẹn* và *tránh dữ liệu bị lặp lại không cần thiết*.

Chuẩn hóa dữ liệu

- **Mục đích của chuẩn hóa dữ liệu**
  - Loại bỏ các bất thường (*anomaly*) của một quan hệ để có được các quan hệ *có cấu trúc tốt hơn, nhỏ hơn*.
  - *Quan hệ có cấu trúc tốt* (*well-structured relation*)
    - Có sự dư thừa dữ liệu là tối thiểu.
    - Cho phép người sử dụng thêm vào, cập nhật và xóa bỏ dữ liệu mà không gây ra sự mâu thuẫn dữ liệu.

Phụ thuộc hàm

- **Phụ thuộc hàm**
  - FD – *Functional Dependency*
  - Cho r là một quan hệ, X và Y là hai tập thuộc tính của r.
  - Chúng ta nói "*X xác định hàm Y*" hoặc "*Y phụ thuộc hàm vào X*", ký hiệu là $X \rightarrow Y$ và được gọi là *phụ thuộc hàm* nếu với mỗi giá trị của X trong r chỉ tương ứng với một giá trị của Y.
  - Khóa của một quan hệ xác định hàm các thuộc tính không khóa của quan hệ này.

Phụ thuộc hàm

- **Phụ thuộc hàm riêng phần**
  - *partial functional dependency*
  - $X \rightarrow A$ được gọi là *phụ thuộc hàm riêng phần* nếu tồn tại $Y \subset X$ để cho $Y \rightarrow A$.
- **Phụ thuộc hàm đầy đủ**
  - *full functional dependency*
  - $X \rightarrow A$ được gọi là *phụ thuộc hàm đầy đủ* nếu không tồn tại $Y \subset X$ để cho $Y \rightarrow A$.
- **Phụ thuộc bắc cầu**
  - *transitive dependency*
  - $X \rightarrow A$ được gọi là *phụ thuộc bắc cầu* nếu tồn tại Y để cho $X \rightarrow Y$, $Y \rightarrow A$, $Y \not\rightarrow X$ và $A \notin XY$.

Phụ thuộc hàm

R

Mãsv	Họtên	Mãlớp	Tênlớp	Mônhọc	Điểm
S1	Tiến	L1	MT01	M1	9
S1	Tiến	L1	MT01	M2	7
S1	Tiến	L1	MT01	M3	8
S2	Trúc	L1	MT01	M1	9
S2	Trúc	L1	MT01	M2	8
S3	Hiền	L2	MT02	M1	5

Các phụ thuộc hàm:
Mãsv → {Họtên, Mãlớp}
Mãlớp → Tênlớp
{Mãsv, Mônhọc} → Điểm

Khóa của R: {Mãsv, Mônhọc}

{Mãsv, Môn học} → Họtên là phụ thuộc hàm riêng phần
{Mãsv, Mônhọc} → Điểm là phụ thuộc hàm đầy đủ

Dạng chuẩn 1

- **Định nghĩa**

  Quan hệ *R* ở *dạng chuẩn 1* (1NF - *First Normal Form*) nếu mọi thuộc tính của *R* đều chứa các *giá trị nguyên tố* (*atomic value*), giá trị này không là *một danh sách các giá trị* hoặc *các giá trị phức hợp* (*composite value*).

- **Các thuộc tính của quan hệ *R***
  - Không là thuộc tính đa trị (*multivalued attribute*).
  - Không là thuộc tính phức hợp (*composite attribute*).

Dạng chuẩn 1

R

Mãsv	Họtên	Mãlớp	Tênlớp	Điểmthi	
S1	Tiến	L1	MT01	M1	9
S1	Tiến	L1	MT01	M2	7
S1	Tiến	L1	MT01	M3	8
S2	Trúc	L1	MT01	M1	9
S2	Trúc	L1	MT01	M2	8
S3	Hiền	L2	MT02	M1	5

Quan hệ *R* không ở dạng chuẩn 1 vì thuộc tính *Điểmthi* là thuộc tính phức hợp.

Dạng chuẩn 1

R

Mãsv	Họtên	Mãlớp	Tênlớp	Mônhọc	Điểm
S1	Tiến	L1	MT01	M1	9
S1	Tiến	L1	MT01	M2	7
S1	Tiến	L1	MT01	M3	8
S2	Trúc	L1	MT01	M1	9
S2	Trúc	L1	MT01	M2	8
S3	Hiền	L2	MT02	M1	5

Quan hệ *R* ở 1NF vì các thuộc tính của *R* không là thuộc tính đa trị, không là thuộc tính phức hợp.

Dạng chuẩn 1

- **Các bất thường của quan hệ ở 1NF**
  - Thêm vào
    - Không thể thêm thông tin của sinh viên mới có mã là *S4*, tên là *Thành*, thuộc lớp có mã là *L1* nếu sinh viên này chưa đăng ký học môn học nào cả.
  - Cập nhật
    - Sửa tên của sinh viên có tên là *Tiến* với tên mới là *Thành* sẽ phải sửa tất cả các hàng của sinh viên này.
  - Xóa bỏ
    - Xóa thông tin sinh viên *S3* đăng ký môn học *M1* sẽ làm mất thông tin của sinh viên này.
  - Nguyên nhân
    - Tồn tại thuộc tính không khóa phụ thuộc hàm riêng phần vào khóa.

Dạng chuẩn 2

- **Định nghĩa**

  Quan hệ *R* ở *dạng chuẩn 2* (2NF - *Second Normal Form*) nếu *R* ở dạng chuẩn 1 và mọi thuộc tính không khóa đều phụ thuộc hàm đầy đủ vào mọi khóa của *R*.

Dạng chuẩn 2

R

Mãsv	Họtên	Mãlớp	Tênlớp	Mônhọc	Điểm
S1	Tiến	L1	MT01	M1	9
S1	Tiến	L1	MT01	M2	7
S1	Tiến	L1	MT01	M3	8
S2	Trúc	L1	MT01	M1	9
S2	Trúc	L1	MT01	M2	8
S3	Hiền	L2	MT02	M1	5

Các phụ thuộc hàm:
Mãsv → {Họtên, Mãlớp}
Mãlớp → Tênlớp
{Mãsv, Môhọc} → Điểm

Khóa của R: {Mãsv, Môhọc}

Lược đồ quan hệ *R* không ở 2NF vì thuộc tính không khóa *Họtên* phụ thuộc hàm riêng phần vào khóa {*Mãsv*, *Môhọc*}.

Dạng chuẩn 2

R_1

Mãsv	Họtên	Mãlớp	Tênlớp
S1	Tiến	L1	MT01
S2	Trúc	L1	MT01
S3	Hiền	L2	MT02

Khóa của R_1: Mãsv

R_2

Mãsv	Mônhọc	Điểm
S1	M1	9
S1	M2	7
S1	M3	8
S2	M1	9
S2	M2	8
S3	M1	5

Khóa của R_2: {Mãsv, Môhọc}

Lược đồ quan hệ R_1 và R_2 đều ở 2NF vì các thuộc tính không khóa đều phụ thuộc hàm đầy đủ vào khóa.

Dạng chuẩn 2

- **Các bất thường của quan hệ ở 2NF**
  - Thêm vào
    - Không thể thêm thông tin của lớp *L3* có tên là *MT03* nếu chưa có sinh viên nào học lớp này.
  - Cập nhật
    - Sửa tên của lớp có mã *L1* với tên mới là *MT_1* sẽ phải sửa tất cả các hàng của lớp này.
  - Xóa bỏ
    - Xóa thông tin của sinh viên có mã *S3* sẽ làm mất thông tin của lớp *L2*.
  - Nguyên nhân
    - Tồn tại thuộc tính không khóa phụ thuộc bắc cầu vào khóa.

Dạng chuẩn 3

- **Định nghĩa**

  Quan hệ *R* ở *dạng chuẩn 3* (3NF- *Third Normal Form*) nếu *R* ở dạng chuẩn 1 và mọi thuộc tính không khóa đều không phụ thuộc bắc cầu vào một khóa của *R*.

Dạng chuẩn 3

R_1

Mãsv	Họtên	Mãlớp	Tênlớp
S1	Tiến	L1	MT01
S2	Trúc	L1	MT01
S3	Hiền	L2	MT02

Lược đồ quan hệ R_1 không ở 3NF vì thuộc tính không khóa *Tênlớp* phụ thuộc bắc cầu vào khóa *Mãsv*.

Mãsv → Mãlớp *Mãlớp → Tênlớp*

Mãlớp –I→ Mãsv *Tênlớp ∉ {Mãsv, Mãlớp}*

Dạng chuẩn 3

R_{11}

Mãlớp	Tênlớp
L1	MT01
L2	MT02

Khóa của R_{11}: Mãlớp

R_{12}

Mãsv	Họtên	Mãlớp
S1	Tiến	L1
S2	Trúc	L1
S3	Hiền	L2

Khóa của R_{12}: Mãsv

Lược đồ quan hệ R_{11} và R_{12} đều ở 3NF vì các thuộc tính không khóa đều không phụ thuộc bắc cầu vào khóa.

Ngôn ngữ *SQL*

- *Ngôn ngữ truy vấn có cấu trúc* (SQL -*Structured Query Language*) là một ngôn ngữ chuẩn được dùng để tạo lập và truy vấn các cơ sở dữ liệu quan hệ.
- *SQL* là một ngôn ngữ chuẩn cho các hệ quản trị CSDL quan hệ (RDBMS - *Relational* DBMS).

Chương 5: Thiết Kế CSDL Mức Quan Niệm

Nội Dung Chính

- Mục tiêu thiết kế mức quan niệm
- Cơ chế kiểm tra ràng buộc
- Các dạng chuẩn
- Cách tiếp cận phân rã và tổng hợp

Thiết kế Mức Quan Niệm

- Input: Sơ phác cấu trúc về CSDL (biểu diễn bằng một mô hình dữ liệu như: mô hình quan hệ, mô hình thực thể kết hợp, mô hình đối tượng…
- Output: Cấu trúc CSDL phù hợp hơn với môi trường ứng dụng.
- Processing: dựa trên một số tiêu chuẩn thiết kế.

Quan hệ phổ quát

- Quan hệ phổ quát: nền tảng của nghiên cứu mô hình quan hệ và phụ thuộc dữ liệu vào thiết kế CSDL
- Khái niệm:
  - Mỗi CSDL ứng dụng đều có thể mô hình bằng một quan hệ duy nhất, gọi là quan hệ phổ quát ký hiệu U.
  - Quan hệ phổ quát chứa tất cả thuộc tính của các quan hệ trong csdl ứng dụng.
- Nếu csdl có Q_i quan hệ với i=1..n thì
  - $U^+ = Q_1^+ \cup Q_2^+ .. \cup Q_n^+$
  - $Q_i = U[Q_i^+]$

Quan hệ phổ quát

- Hệ quả:
  - Mỗi thuộc tính có một ý nghĩa duy nhất dù thuộc tính này nằm trong quan hệ con hay quan hệ phổ quát.
  - Giữa hai thuộc tính chỉ có một mối kết hợp mang ý nghĩa duy nhất.

Ví dụ Quan hệ phổ quát

Ví dụ 3.1: Cho môi trường ứng dụng với 3 quan hệ con như sau:

ĐẶT_HÀNG (Mã_ĐH, Ngày), $F_{ĐH}$ = {Mã_ĐH → Ngày}

THỐNG_KÊ_HÀNG_GIAO(MãHG, Ngày, Lượng)

F_{TKHG} = {MãHG, Ngày → Lượng }

CHI_TIẾT_ĐẶT_HÀNG (Mã_ĐH, Mã_HG, Lượng)

$F_{CTĐH}$ = {Mã_ĐH, Mã_HG → Lượng }

→ *Các thuộc tính Ngày và Lượng có ý nghĩa khác nhau trong các quan hệ khác nhau.*

Ví dụ Quan hệ phổ quát

Ví dụ 3.1 (tt):

→ *Gây ra tình trạng nhập nhằng về ngữ nghĩa, dẫn đến kết quả sai trong quá trình thiết kế lại cấu trúc CSDL khi áp dụng luật dẫn cho phụ thuộc dữ liệu trên toàn cục*

- Từ (Mã_ĐH → Ngày) và (MãHG, Ngày → Lượng) ta suy dẫn được PTH (MãHG, Mã_ĐH → Lượng), dù PTH này không đúng trên thực tế.

→ *Phải đặt tên lại cho các thuộc tính:*

- Ngày trong ĐẶT_HÀNG thành Ngày_ĐH
- Ngày trong THỐNG_KÊ_HÀNG_GIAO thành Ngày_TK
- ...

U(Ma_DH,Ngay_DH,Ngay_TK,Luong_Dat,Luong_Giao,Ma_HG)

Ví dụ Quan hệ phổ quát

Ví dụ 3.2:

Cho quan hệ NHÂN_VIÊN(Mã_NV, Tên_NV, Ngày)

Ý nghĩa mối kết hợp giữa Mã_NV và Ngày . Ngày sinh của nhân viên

→ Mối kết hợp này là duy nhất : nếu cần thêm thông tin về ngày tuyển dụng, ta phải dùng thêm một thuộc tính Ngày_TD, không thể thêm một ý nghĩa khác cho mối kết hợp giữa Mã_NV và Ngày.

Cơ chế kiểm tra phụ thuộc hàm

Phụ thuộc hàm: một loại ràng buộc toàn vẹn quan trọng trong CSDL quan hệ

Cần chọn cấu trúc CSDL sao cho việc kiểm tra PTH hiệu quả nhất.

Xem xét:

- Cơ chế kiểm tra PTH nào được hệ quản trị CSDL hỗ trợ.
- Thời điểm cần kiểm tra PTH trong quá trình khai thác CSDL?

Cơ chế khóa kiểm tra phụ thuộc hàm

Định nghĩa:

Cho một quan hệ Q, một khóa K của Q

Cơ chế khóa là một cơ chế có sẵn của Hệ quản trị CSDL nhằm đảm bảo:

$$\forall q, q' \in Q \text{ sao cho } q \neq q' , q.K \neq q'.K$$

Hệ thống chỉ mục với giá trị duy nhất là một ví dụ của cơ chế khóa

Cơ chế khóa đảm bảo kiểm tra hiệu quả các phụ thuộc hàm dạng $K \rightarrow A$, với $A \in Q^+$ và K là một khóa của Q (Không cần duyệt tuần tự qua tất cả các bộ trong TQ)

Cơ chế kiểm tra phụ thuộc hàm

Cho một quan hệ Q(ABC) với khóa là A, và các phụ thuộc hàm được định nghĩa trên Q:

$F = \{B \rightarrow C, A \rightarrow BC\}$

Cho một tình trạng của Q:

A	B	C
a_1	b_1	c_1
a_2	b_2	c_1
a_3	b_3	c_3
a_4	b_2	c_1

Tình trạng của Q được hợp lệ hóa đối với các PTH định nghĩa trên Q nếu TQ thỏa *tất cả* các PTH này

Cơ chế kiểm tra phụ thuộc hàm

Kiểm tra B→ C, đảm bảo:

$\forall q, q' \in Q$, nếu $q.B = q'.B$ thì $q.C = q'.C$

→Cần duyệt qua tất cả các bộ của TQ để kiểm tra →tốn kém và mất thời gian khi TQ lớn

Kiểm tra A→ BC: nhờ cơ chế khóa của HQT CSDL, việc kiểm tra PTH này sẽ hiệu quả hơn.

Mục tiêu của Thiết Kế CSDL

Hai tiêu chuẩn quan trọng cần đạt được trong quá trình thiết kế CSDL mức quan niệm:

- Cấu trúc CSDL kết quả (đầu ra của giai đoạn thiết kế mức quan niệm) cần đạt **dạng chuẩn** cao nhất
- Cấu trúc CSDL kết quả phải **tương đương** với cấu trúc ban đầu

Mục tiêu của Thiết Kế CSDL

Tiêu chuẩn về dạng chuẩn:

- Giảm tối đa sự trùng lắp thông tin, từ đó tránh được một số bất tiện khi cập nhật CSDL
- Các phụ thuộc dữ liệu được kiểm tra dễ dàng nhất, đơn giản và tương đối ít tốn kém nhất

Tiêu chuẩn tương đương:

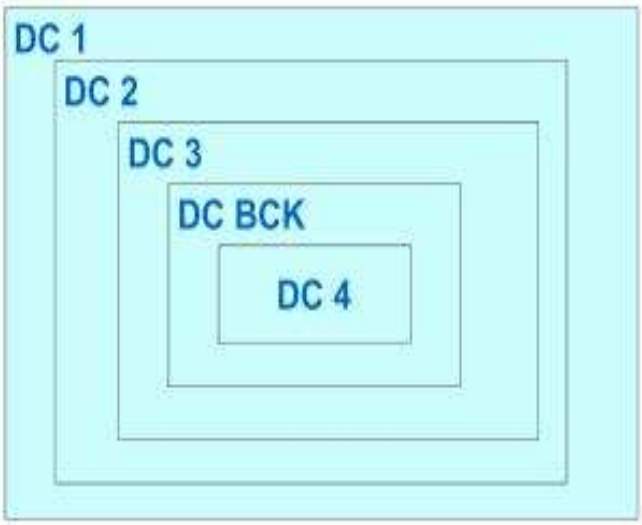
- Đảm bảo các thông tin trong quan hệ phổ quát sẽ được tìm thấy đầy đủ trong CSDL của cấu trúc kết quả.

Mục tiêu của Thiết Kế CSDL

Một CSDL thỏa mãn tiêu chuẩn về dạng chuẩn và tính tương đương đảm bảo cho việc khai thác nó được thuận lợi trên cả ba phương diện:

- *Truy vấn:* tiêu chuẩn tương đương đảm bảo các thông tin được truy xuất từ CSDL là những thông tin đã được phân tích.
- *Cập nhật:* Tiêu chuẩn dạng chuẩn giảm bớt các tình huống thông tin mâu thuẫn sau khi cập nhật.
- *Kiểm tra ràng buộc toàn vẹn:* cả hai tiêu chuẩn đều hướng đến mục tiêu là kiểm tra RBTV dạng phụ thuộc dữ liệu được thuận lợi.

Các dạng chuẩn



Các dạng chuẩn

- Dạng chuẩn 1: cấu trúc csdl đạt dạng chuẩn một khi hết đa trị hay trường lặp (cấu trúc phẳng).
- Dạng chuẩn 1: chưa chú ý đến giảm trùng lắp thông tin → cập nhật khó.

Các dạng chuẩn

Quan hệ CHUAN1

Emp_ID	Emp_Name	Post_Code	Post_Name	Prj_ID	Prj_name	Time	Prj_Type
E1	A	058	NT	P1	P_A	2008	IT
E2	B	058	NT	P1	P_A	2008	IT
E1	A	058	NT	P2	P_B	2009	FA
E3	C	08	HCM	P3	P_C	2009	HR

Nhân xét sự trùng lắp thông tin và khó khăn khi cập nhật dữ liệu.

Các dạng chuẩn

- Khi ở dạng chuẩn 1 cần tách nhỏ quan hệ theo các tiêu chuẩn nghiêm ngặt hơn để giảm trùng lắp thông tin và csdl hoạt động hiệu quả hơn.

Các dạng chuẩn

- Dạng chuẩn 2: một quan hệ ở dạng chuẩn 2 khi các thuộc tính không khóa phụ thuộc hoàn toàn (đầy đủ) vào khóa.
- Xét quan hệ CHUAN1 ở slide trước:
  - Khóa: Emp_ID và Prj_ID
  - Thuộc tính không khóa: Emp_Name, Post_Code, Post_Name, Prj_Name, Time, Prj_Type
  - Prj_ID → Prj_Name. Vậy Prj_Name chỉ phụ thuộc một phần vào khóa.
  - Vì vậy quan hệ này chưa đạt chuẩn 2

Các dạng chuẩn

Emp_ID	Emp_Name	Post_Code	Post_Name
E1	A	058	NT
E2	B	058	NT
E3	C	08	HCM

Emp_ID	Pro_ID
E1	P1
E2	P1
E1	P2
E3	P3

Prj_ID	Prj_name	Time	Prj_Type
P1	P_A	2008	IT
P1	P_A	2008	IT
P2	P_B	2009	FA
P3	P_C	2009	HR

Các dạng chuẩn

- Dạng chuẩn 3: một quan hệ ở dạng chuẩn 3 khi các thuộc tính không khóa phụ thuộc hoàn toàn (đầy đủ) và không phụ thuộc bắc cầu vào khóa.
- Xét quan hệ CHUAN2 ở slide trước:
  - Thuộc tính không khóa Post_Name phụ thuộc bắc cầu vào khóa Emp_ID:

    Emp_ID → Post_Code

    Post_Code → Post_Name

  Vậy chưa đạt chuẩn 3, phải tiếp tục tách bảng để giảm trùng lắp thông tin.

Các dạng chuẩn

Emp_ID	Emp_Name	Post_Code
E1	A	058
E2	B	058
E3	C	08

Emp_ID	Pro_ID
E1	P1
E2	P1
E1	P2
E3	P3

Prj_ID	Prj_name	Time	Prj_Type
P1	P_A	2008	IT
P1	P_A	2008	IT
P2	P_B	2009	FA
P3	P_C	2009	HR

Post_Code	Post_Name
058	NT
08	HCM

Thiết kế ở mức logic

Giai đoạn thiết kế logic là bước trung gian để giai đoạn thiết vật lý được thực hiện dễ dàng.

- Biểu diễn lại mô hình CSDL mức quan niệm thành mô hình dữ liệu phù hợp với hệ quản trị sẽ dùng để cài đặt sau này.
  - Độc lập với các đặc trưng khác của hệ quản trị.
- Chọn một biểu diễn ở dạng đồ thị phù hợp, làm cơ sở cho việc lựa chọn chỉ mục ở giai đoạn thiết kế vật lý.

Thiết kế ở mức logic

Kết quả của giai đoạn thiết kế vật lý phải hoàn toàn "trung thực" với cấu trúc quan niệm:

- Bảo toàn nội dung CSDL
- Bảo toàn sự truy xuất trực tiếp đến các quan hệ của cấu trúc quan niệm.
  - Khi một quan hệ con Q_i tồn tại trong cấu trúc quan niệm C, một bộ của Q_i có thể được truy xuất trực tiếp (không cần thông qua một quan hệ nào khác). Cấu trúc logic phải bảo toàn khả năng này.
- Bảo toàn tính hiệu quả trong việc kiểm tra ràng buộc toàn vẹn.

Thiết kế ở mức logic

Ví dụ 4.1

Cho cấu trúc mức quan niệm:

NhânViên(Ma_NV, HoTen_NV, Ma_P)

Phòng (Ma_P, Ten_P)

ĐềÁn(Ma_DA, Ten_DA, Ma_P)

PhânCông (Ma_NV, Ma_DA), *ràng buộc: Một nhân viên được phân công vào tất cả các đề án do phòng ban mà nhân viên đó trực thuộc phụ trách)*

Thiết kế ở mức logic

– Xét cấu trúc logic sau:

NhânViên(Ma_NV, HoTen_NV, Ma_P)

Phòng (Ma_P, Ten_P)

ĐềÁn_1 (Ma_DA, Ten_DA)

PhụTrách(Ma_DA, Ma_P)

→Không có quan hệ *PhânCông*: có bảo toàn nội dung không?

→Có bảo toàn sự truy xuất trực tiếp đến quan hệ *PhânCông* không?

Thiết kế ở mức logic

Định nghĩa:

– Đồ thị con đường truy xuất là một đồ thị có hướng với:

- ✓ N: Tập các nút của đồ thị
- ✓ $C \subseteq (N \times N)$: Tập các cung (có hướng)
- ✓ Q_i: tập các quan hệ Q_i.
- ✓ Cđ: Tập các con đường truy xuất

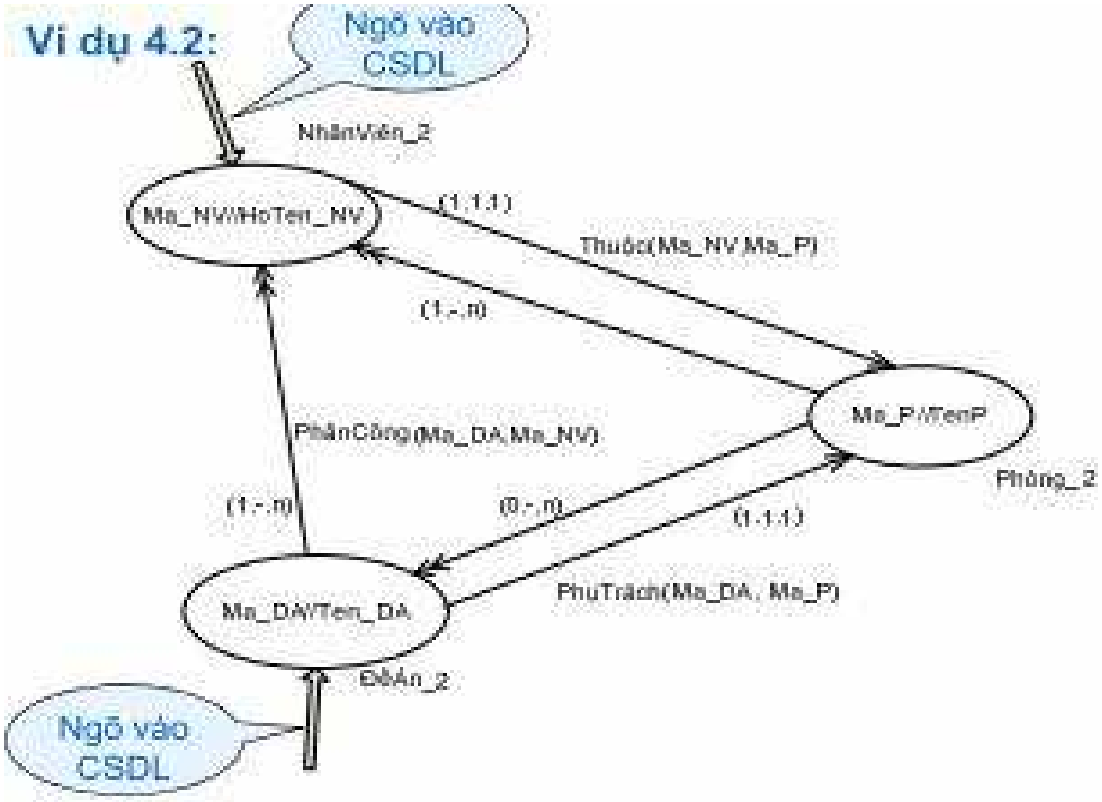
Thiết kế ở mức logic

– Mỗi cung trên đồ thị tương ứng với một con đường truy xuất

– Một quan hệ $Q_i \in Q_i$ có thể là quan hệ nút (nếu nó tương ứng với một nút trên đồ thị) hoặc quan hệ cung (nếu nó ứng với một cung trên đồ thị).

– Mỗi quan hệ cung Q_C có thể tương ứng với tối đa hai cung ngược chiều nhau trên đồ thị, nút đến của cung này là nút đi của cung kia và ngược lại

Thiết kế ở mức logic



Thiết kế ở mức logic

Diễn giải:

- Có hai ngõ vào CSDL: NhânViên_2 và ĐềÁn_2, nghĩa là cung cấp một giá trị Ma_NV (Ma_DA), ta có thể truy xuất ngay một bộ tương ứng trong quan hệ NhânViên_2 (ĐềÁn_2)
- Từ một bộ NhânViên_2, ta có thể truy xuất *trực tiếp* một bộ của Phòng_2 mà nhân viên trực thuộc, thông qua con đường truy xuất NhânViên_2 → Phòng_2
- Từ một bộ của Phòng_2, ta có thể truy xuất trực tiếp danh sách các nhân viên của phòng thông qua con đường truy xuất Phòng_2 → NhânViên_2.
- Từ một bộ của NhânViên, ta không thể truy xuất trực tiếp danh sách các đề án mà nhân viên được phân công, do không có con đường truy xuất NhânViên_2 → ĐềÁn_2

Thiết kế ở mức vật lý

- **Cơ sở cân nhắc:**
  - Ý đồ khai thác
  - Đặc trưng kỹ thuật của hệ quản trị csdl

Thiết kế ở mức vật lý

Đào một quan hệ nút – Chọn chỉ mục cho một nút

Ví dụ:

NV(Mã_NV, Tên_NV,Mã_P)

Có nhu cầu truy xuất các bộ của nhân viên dựa trên Mã_P → Khai báo Mã_P là chỉ mục của quan hệ NV.

→ Đảo quan hệ NV theo Mã_P

Chia nhỏ một nút:

Để nguyên hay tách một nút?

– *Tách dọc:*

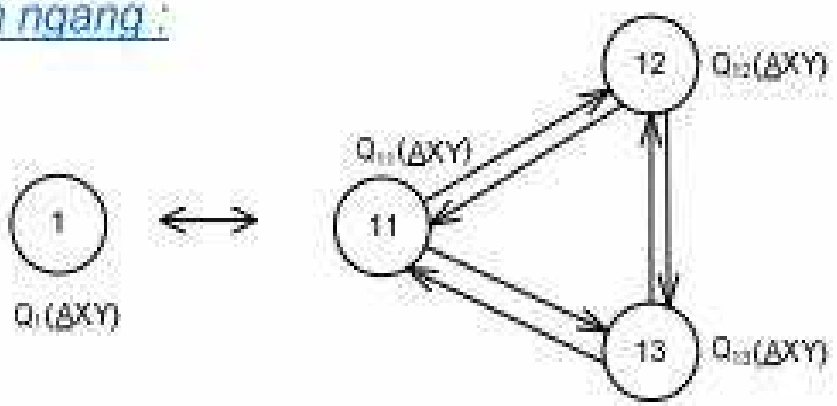
Xét quan hệ Q(AXY), trong đó Y có thể là một tập thuộc tính lớn, và trong quá trình khai thác ta thường xuyên quan tâm tới AX, ít nhắc tới Y

→ tách Q thành hai nút Q_1(AX) và Q_2(AY)

Thiết kế ở mức vật lý

Chia nhỏ một nút (tt):

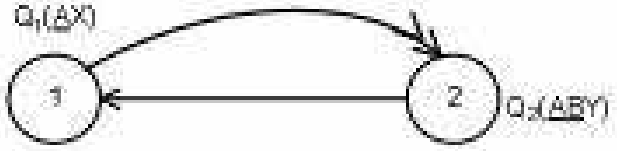
– *Tách ngang :*



Ví dụ: Sổ thu chi trong năm từ 1/1 đến 31/12 có dữ liệu rất lớn → chia ra thành 12 quan hệ con, mỗi quan hệ chứa thông tin của từng kỳ

Gộp nút – Đưa thêm thuộc tính vào một quan hệ nút

– Ví dụ 1:



→ Từ một bộ của Q_1, rất thường xuyên có nhu cầu truy xuất các bộ của Q_2 tương ứng. Do đó gộp hai quan hệ này lại thành một để truy xuất cho nhanh. Khi đó khai báo vật lý:

Q_{12} (ABXY), khoá chính (AB), chỉ mục (A)

→ Q_{12} chỉ đạt DC1, ta chấp nhận trùng lắp thông tin để truy xuất nhanh

Thiết kế ở mức vật lý

Cài đặt nút vào:

– Cài đặt nút vào cho phép ta truy xuất nhanh đến một bộ của quan hệ từ giá trị khóa mà không cần duyệt tuần tự qua các bộ trong thể hiện.

– Tùy vào hỗ trợ của hệ quản trị, ta chọn cách cài đặt nút vào:

- ✓ Cài đặt bằng một chỉ mục có giá trị duy nhất , xem chỉ mục đặc biệt này là khóa của nút
- ✓ Cài đặt bằng hàm băm,…
- ✓ Hiện nay đa số các hệ quản trị CSDL đều có hỗ trợ cơ chế khóa chính.

CÂU 1

Cho biết đối tượng user nào thực hiện các chức năng sau trong Hệ thống chi trả của 1 công ty:

a) Viết 1 chương trình để tạo và in các hóa đơn

b) Thay đổi địa chỉ trong CSDL

c) Tạo 1 tài khoản cho user mới

CÂU 2

Một công ty, trong đó Tên và Địa chỉ khách hàng được lưu trong 1 CSDL của Bộ phận bán hàng và 1 CSDL khác của Bộ phận hóa đơn. Hỏi sự dư thừa dữ liệu này vẫn đến vấn đề thiếu nhất quán như thế nào?

CÂU 3

User sẽ dùng ngôn ngữ nào (mô tả, thao tác, truy vấn, quản lý) để thực hiện:

a) Điều chỉnh địa chỉ khách hàng

b) Định nghĩa Bảng Hàng tồn kho

c) Nhập thông tin cho nhân viên mới

CÂU 4

User sẽ dùng ngôn ngữ nào (mô tả, thao tác, truy vấn, quản lý) để thực hiện:

a) Cập nhật điểm trung bình cho sinh viên

b) Định nghĩa Bảng Danh sách lớp

c) Thêm 1 cột mới vào Bảng

CÂU 1

- Cho quan hệ Q (ABC)

A	B	C
1	4	2
3	5	6
3	4	6
7	3	8
9	1	0

<u>Hỏi Q có thỏa các pth sau không</u>:

$A \rightarrow B$, $A \rightarrow C$, $AB \rightarrow C$,

$C \rightarrow A$, $BC \rightarrow A$, $AC \rightarrow B$

CÂU 2

Cho tập phụ thuộc hàm

F = { A → B, C → X, BX → Z }

CMR: AC → Z

CÂU 3

Cho quan hệ Q (ABCD) với tập phụ thuộc hàm F = { A → B, BC → D }

<u>Hỏi các phụ thuộc hàm sau có được suy diễn từ F không (chứng minh):</u>

a) **AC → D**

b) **B → D**

c) **AD → B**

CÂU 4

- Cho quan hệ Q (ABC)

A	B	C
f	e	e
d	e	e
b	c	e
a	c	d
a	b	c

Tìm tập phụ thuộc hàm F

CÂU 5

Cho quan hệ Q(ABCDEGH)

Cho F = { $AB \rightarrow D$, $B \rightarrow C$, $DC \rightarrow E$,

$DE \rightarrow GH$, $G \rightarrow A$ }

CMR: $AB \rightarrow Q^+ \in F^+$

CÂU 6

Cho quan hệ Q(ABCD)

Cho F = { AB → C, C → D, D → A }

Tìm khóa

CÂU 7

Cho quan hệ Q(ABC)

Cho F = { AB → C, B → D, D → B }

Tìm khóa

CÂU 8

Cho quan hệ Q(XYZWP)

Cho F = { Y → Z, Z → Y, Z → W, Y → P }

Tìm khóa

CÂU 9

Cho quan hệ Q(ABCDEG)

Cho F = { EC → B, AB → C, EB → D,

BG → A, AE → G }

Tìm khóa

CÂU 10

Cho quan hệ Q(ABCDEH)

Cho F = { AB → C, CD → E, EC → A,

CD → H, H → B }

Tìm khóa

CÂU 11

Cho tập phụ thuộc hàm

F = { X → YZ, ZW → P, P → Z, W → XPQ,

XYQ → YW, WQ → YZ }

Tìm phủ tối thiểu

CÂU 12

Cho tập phụ thuộc hàm

F = { X → Y, Y → X, Y → Z,

Z → Y, X → Z, Z → X }

<u>Tìm phủ tối thiểu</u>

CÂU 13

Cho tập phụ thuộc hàm

F = { X → YW, XW → Z, Z → Y, XY → Z }

<u>Tìm phủ tối thiểu</u>

Đại học Sư Phạm Tp HCM
Khoa Toán Tin Học

Công hòa xã hội chủ nghĩa Việt Nam
Độc lập - Tự do - Hạnh phúc

ĐỀ THI MÔN THIẾT KẾ CƠ SỞ DỮ LIỆU

Thời gian : 90 phút -

(Sinh viên được quyền tham khảo tài liệu
cán bộ coi thi không giải thích gì thêm)

Giảng Viên ra đề: Trần Sơn Hải

Câu 1: Trình bày các bước thiết kế cơ sở dữ liệu?

Câu 2: Trình bày các kỹ thuật để thu nhận yêu cầu của người sử dụng cơ sở dữ liệu.

Câu 3: **Cho quan hệ** GIẢNG_DẠY (MS_CBGD, MS_MÔNHỌC, TÊN_CBGD, HỌCHÀM, MÃLỚP, SỐSV)

F = { MS_CBGD → TÊN_CBGD,
MS_CBGD → HỌCHÀM,
MS_CBGD, MS_MÔNH ỌC → SỐSV,
MS_MÔNHỌC → HỌCHÀM, MÃLỚP }

Tìm khóa ?

Câu 4: Cho quan hệ Q(ABCDEH)

Cho F = { AB → C, CD → E, EC → A,
CD → H, H → B }

Tìm khóa ?

Câu 5: Cho quan hệ Q (ABCD) với tập phụ thuộc hàm F = { A → B, BC → D }
Hỏi các phụ thuộc hàm sau có được suy diễn từ F (chứng minh):

a) **AC → D**
b) **B → D**
c) **AD → B**

----------hết----------

CÂU 14

Cho quan hệ

GIẢNG_DẠY (MS_CBGD, MS_MÔNHỌC, TÊN_CBGD, HỌCHÀM, MÃLỚP, SỐSV)

F = { MS_CBGD → TÊN_CBGD,
MS_CBGD → HỌCHÀM,
MS_CBGD, MS_MÔNHỌC → SỐSV,
MS_MÔNHỌC → HỌCHÀM, MÃLỚP }

Tìm khóa

THIẾT KẾ CƠ SỞ DỮ LIỆU QUAN HỆ (Relational Database Designing)

Phần II – NGÔN NGỮ TRUY VẤN SQL

(Structured Query Language = ngôn ngữ truy vấn có cấu trúc)

SQL = Structured Query Language

- Là ngôn ngữ dùng để truy vấn dữ liệu
- Ngôn ngữ = cú pháp (cấu trúc ngữ pháp) + các từ khóa (từ vựng) + hàm lập sẵn.
- Là 1 công cụ giao tiếp của Hệ Quản Trị CSDL
- Là cầu nối giữa :
  - Nhà phát triển (Lập trình viên) và Hệ quản trị CSDL
  - Người dùng cuối (End-user) và Hệ quản trị CSDL

SQL = Structured Query Language

- Ngôn ngữ SQL là một <u>chuẩn chung tương đối</u> giữa các Hệ quản trị CSDL khác nhau.
- 1 trong các cú pháp của SQL :

SELECT <tên các thuộc tính>

FROM <tên các quan hệ>

WHERE <điều kiện chọn>

…

Cú pháp SQL – Kiểu Dữ liệu

(data type)

- Chuỗi (String) : được đặt trong dấu nháy kép hoặc đơn.
  - Ví dụ :

```
SELECT *
FROM SINHVIEN
WHERE MASV = “SV01”
```

dữ liệu chuỗi

Cú pháp SQL – Kiểu Dữ liệu (t.t)

(data type)

- Số (number)
  - Ví dụ : 1024 ; 4.5 ; …
- Ngày tháng (date/time) : được đặt trong cặp dấu #, giữa ngày – tháng – năm là dấu phân cách "-" hoặc "/", tên tháng có thể là số (1-12) hoặc viết tắt 3 chữ cái đầu.
  - Ví dụ : #12/2/2001# ; #1-Jan-94# ; …

Cú pháp SQL – Các toán tử số học

(Arithmetic Operations)

Toán tử	Ý nghĩa	Ví dụ	Kết quả
+	Cộng	5 + 2 #28/08/01# + 4	7 #01/09/01#
-	Trừ	#02/09/01# - 3	#30/08/01#
*	Nhân	5 * 2	10
/	Chia	5 / 2	2.5
\	Chia nguyên	5 \ 2	2
^	Lũy thừa	5 ^ 2	25
Mod	Chia dư	5 Mod 2	1

Cú pháp SQL – Các toán tử so sánh

(Comparative Operations)

Toán tử	Ý nghĩa	Ví dụ	Kết quả
<	Nhỏ hơn	3 < 5	True
<=	Nhỏ hơn hay bằng	2 <= 5	True
>	Lớn hơn	2 > 5	False
>=	Lớn hơn hay bằng	2 >= 5	False
=	Bằng nhau	2 = 5	False
<>	Khác nhau	2 <> 5	True

Cú pháp SQL – Các toán tử luận lý

(Logical Operations)

Toán tử	Ý nghĩa	Ví dụ	Kết quả
Not	Luật phủ định	Not (5 > 2) Not (2>5)	False True
And	Luật và	(5>2) And (2>5) (5>2) And (5>4)	False True
Or	Luật hay	(5>2) Or (2>5) (2>5) Or (4>5)	True False

Ví dụ

1. *SELECT* HO,TEN
   FROM SINHVIEN
   WHERE NOT(MASV = ‘SV01’)
2. *SELECT* MASV,HO,TEN
   FROM SINHVIEN
   WHERE (DIEMTB >= 5) *AND* (DIEMTB<=6.5)

Cú pháp SQL – Các toán tử *BETWEEN…AND*

Cú pháp :

value1 *Between* value2 *and* value3

Ý nghĩa : trả về True nếu value1 nằm giữa value2 và value3 ⇔value2<=value1<=value3

Ví dụ :

SELECT *

FROM SINHVIEN

WHERE DIEMTB *BETWEEN* 5 *AND* 6.5

Cú pháp SQL – Các toán tử *LIKE*

Cú pháp :

value1 *LIKE* <khuôn mẫu giá trị>

Ý nghĩa :

_Trả về các value1 có dạng thức giống như <khuôn mẫu giá trị>

Các ký tự đại diện dùng trong khuôn mẫu :

* : đại diện cho tất cả ký tự bất kỳ

? : đại diện cho một ký tự bất kỳ

: đại diện cho 1 ký tự số

[A_1,A_2,…] : đại diện cho 1 ký tự thuộc tập {A_1, A_2, …}

[A_1 – A_2] : đại diện cho 1 ký tự thuộc khoảng ký tự từ A_1 đến A_2

Các toán tử *LIKE* – Ví dụ

SELECT *

FROM SINHVIEN

WHERE TEN *LIKE* '*Hoa'

Chọn tất cả các cột có trong quan hệ

<u>Ý nghĩa</u> : tìm tất cả sinh viên có từ Hoa trong phần cuối của tên, ví dụ : 'Ngọc Thoa', 'Đào Hoa', …

Các toán tử *LIKE* – Ví dụ (t.t)

SELECT *

FROM SINHVIEN

WHERE MASV *LIKE* ‘SV0[1-4]’

Ý nghĩa : tìm tất cả sinh viên có mã sinh viên là SV01, SV02, SV03 hoặc SV04

➔ Toán tử LIKE được sử dụng nhiều trong các truy vấn tìm kiếm dữ liệu

Cú pháp SQL – Các hàm lập sẵn

Cú pháp chung : <tênHàm>(Danh sách đối số)

Hàm **IIf**

Cú pháp : **IIf**(điều kiện,giá trị 1,giá trị 2)

Ý nghĩa : Trả về giá trị 1 nếu điều kiện đúng, ngược lại, trả về giá trị 2.

Ví dụ :

SELECT *

FROM SINHVIEN

WHERE DIEMTB >= *IIF*(GIOITINH='Nam',6.5,6)

Hàm Date

Cú pháp : **Date**()

Ý nghĩa : Trả về giá trị ngày giờ hiện tại của hệ thống.

Ví dụ :

SELECT *

FROM HOADON

WHERE NGAYLAP >= (*DATE()*-5)

Hàm Sum

Cú pháp : **Sum**(<tên thuộc tính>)

Ý nghĩa : Trả về tổng của các giá trị tương ứng với <tên thuộc tính> của tất cả các bộ có trong quan hệ thỏa điều kiện WHERE.

Ví dụ :

SELECT	Sum(GIATRI)
FROM	HOADON
WHERE	NGAYLAP >= (*DATE()*-5)

Ý nghĩa : Trả về tổng giá trị của các hóa đơn có ngày lập trong vòng 6 ngày gần đây.

Hàm Max

Cú pháp : **Max**(<tên thuộc tính>)

Ý nghĩa : Trả về giá trị lớn nhất trong các giá trị tương ứng với <tên thuộc tính> của các bộ có trong quan hệ thỏa điều kiện WHERE.

Ví dụ :

SELECT	Max(GIATRI)
FROM	HOADON
WHERE	NGAYLAP >= (*DATE()*-5)

Ý nghĩa : Trả về giá trị lớn nhất trong các hóa đơn có ngày lập trong vòng 6 ngày gần đây.

Một số hàm khác

- ***Day***(<biểu thức ngày>) : trả về chỉ số của ngày trong <biểu thức ngày>.
  - Ví dụ : ***Day***(#12/2/2005#) → 12
- ***Month***(<biểu thức ngày>) : trả về chỉ số của tháng trong <biểu thức ngày>.
- ***Year***(<biểu thức ngày>) : trả về chỉ số của năm trong <biểu thức ngày>.
- ***Len***(<giá trị chuỗi>) : trả về độ dài của chuỗi

Một số hàm khác (t.t)

– Ví dụ :

SELECT *

FROM SINHVIEN

WHERE LEN(TEN) > 4

- ***Chr***(<mã ASCII>) : trả về ký tự có mã ASCII tương ứng.
- ***InStr***(start,s1,s2) : trả về vị trí của chuỗi s2 trong chuỗi s1 kể từ vị trí start.
- ***LCase***(s) : trả về giá trị chuỗi in thường của chuỗi s
- ***UCase***(s) : trả về giá trị chuỗi in hoa của chuỗi s

Một số hàm khác (t.t)

- ***Left***(s,n) : trả về chuỗi gồm n ký tự bên trái của chuỗi s.
- ***Right***(s,n) : trả về chuỗi gồm n ký tự bên phải của chuỗi s.
- ***Mid***(s,i,n) : trả về chuỗi con của chuỗi s gồm n ký tự kể từ vị trí i.
- ***Nz***(v1,v2) : trả về giá trị v1 nếu v1 khác Null, ngược lại trả về giá trị v2.

Một số hàm khác (t.t)

- ***Min***(<tên thuộc tính>) : trả về giá trị nhỏ nhất trong các giá trị tương ứng với <tên thuộc tính> của các bộ thỏa điều kiện WHERE có trong quan hệ.
- ***Avg***(<tên thuộc tính>) : trả về giá trị trung bình cộng của các giá trị tương ứng với <tên thuộc tính> của các bộ thỏa điều kiện WHERE có trong quan hệ.
- ***Count***(<tên thuộc tính>) : trả về <u>số lượng</u> các giá trị tương ứng với <tên thuộc tính> của các bộ thỏa điều kiện WHERE và khác Null có trong quan hệ.

Các Loại Truy Vấn SQL

1. **Truy vấn chọn** **(Select query)**
   - Là các truy vấn bắt đầu bằng từ khóa *SELECT*
   - Trả về 1 giá trị hoặc 1 tập các bộ
2. **Truy vấn định nghĩa dữ liệu** **(Data Definition Query)**
   - Là các truy vấn bắt đầu bằng từ khóa *CREATE, DELETE, INSERT, ALTER, …*
   - Sử dụng để tạo,thêm,xóa,sửa các bảng (quan hệ), bộ, ràng buộc, … trong CSDL
3. **Truy vấn cập nhật dữ liệu** **(Data Modification Query)**

Truy vấn định nghĩa dữ liệu – Tạo lược đồ quan hệ

Ví dụ 1 :

CREATE TABLE SINHVIEN(

MASV *Text*(10) *CONSTRAINT* k1 *PRIMARY KEY*,

HOTEN *Text*(30), NGAYSINH *Date*, MALOP *Text*(10),

DIEMTB *Double*)

Ghi chú : _ Từ in nghiêng là từ khóa của SQL

_ *Text, Date, Double, ...* : các kiểu dữ liệu (của thuộc tính)

_ *Text*(10) : kiểu dữ liệu Text, có độ dài 10 ký tự

Tạo lược đồ quan hệ (t.t)

_ MASV *Text*(10) *CONSTRAINT* k1 *PRIMARY KEY* :

Khai báo thuộc tính MASV là khóa chính với *quy tắc ràng buộc* tên là k1

<u>Ví dụ 2</u> :

CREATE TABLE BANGDIEM(

MASV *Text*(10), MAMH *Text*(10), DIEM Double, *CONSTRAINT* k2 *PRIMARY KEY* (MASV, MAMH)

)

Thêm,xóa,sửa thuộc tính (cột)

Thêm thuộc tính và quan hệ

Ví dụ :

ALTER TABLE SINHVIEN *ADD COLUMN* GIOITINH *TEXT*(10)

Sửa kiểu dữ liệu của thuộc tính :

ALTER TABLE SINHVIEN *ALTER COLUMN* GIOITINH *BOOLEAN*

Xóa thuộc tính

Ví dụ :

ALTER TABLE SINHVIEN *DROP COLUMN* GIOITINH

Xóa,thêm các ràng buộc

Xóa ràng buộc khóa chính

Ví dụ :

ALTER TABLE SINHVIEN *DROP CONSTRAINT* k1

Thêm ràng buộc khóa chính

Ví dụ :

ALTER TABLE SINHVIEN *ADD CONSTRAINT* k1 *PRIMARY KEY* (MASV)

Thêm ràng buộc miền giá trị lên thuộc tính

Ví dụ :

ALTER TABLE SINHVIEN *ADD CONSTRAINT* k3 *CHECK* (DIEMTB>=0 AND DIEMTB<=10)

Truy vấn chọn

Ví dụ 1 : Chọn tất cả sinh viên có điểm trung bình >= 6.5

SELECT * *FROM* SINHVIEN *WHERE* DIEMTB >= 6.5;

Ví dụ 2 : Chọn 10 sinh viên có điểm trung bình cao nhất

SELECT TOP 10 *FROM* SINHVIEN;

Ví dụ 3 : Chọn 10% sinh viên có điểm trung bình cao nhất

SELECT TOP 10% *FROM* SINHVIEN;

Ví dụ 4 : Chọn có loại bỏ các bộ trùng : chọn các mức điểm khác nhau mà các sinh viên đã đạt được

SELECT DISTINCT DIEMTB *FROM* SINHVIEN;

Lưu ý : Dấu ; cho biết đã kết thúc câu lệnh SQL

Truy vấn chọn từ nhiều bảng

<u>Ví dụ 1</u> : Tìm tất cả các tên học phần mà sinh viên mang mã số SV01 đã đăng ký.

SELECT HOCPHAN.TENHP

FROM SINHVIEN,DANGKY_HOCPHAN,HOCPHAN

WHERE SINHVIEN.MASV = ‘SV01’ *AND*

SINHVIEN.MASV = DANGKY_HOCPHAN.MASV *AND*

DANGKY_HOCPHAN.MAHP = HOCPHAN.MAHP;

<u>Lưu ý</u> : *FROM* $Q_1,Q_2,\ldots,Q_n \Leftrightarrow$

FROM $Q_1 x Q_2 x \ldots x Q_n$ (Tích Descartes)

Truy vấn chọn có kết

Ví dụ 1 : Tìm tất cả các tên học phần mà sinh viên mang mã số SV01 đã đăng ký.

```
SELECT    HOCPHAN.TENHP
FROM      (SINHVIEN INNER JOIN DANGKY_HOCPHAN ON
          SINHVIEN.MASV = DANGKY_HOCPHAN.MASV)
          INNER JOIN HOCPHAN ON
          DANGKY_HOCPHAN.MAHP = HOCPHAN.MAHP
WHERE     MASV = 'SV01';
```

Lưu ý : Phép kết chính là phép chọn có điều kiện từ tích Descartes.

Truy vấn chọn có sắp thứ tự kết quả trả về

Ví dụ 1 : Tìm tất cả các tên sinh viên đã đăng ký học phần có mã là CSDL, sắp thứ tự kết quả trả về theo tên tăng dần, họ tăng dần và mã sinh viên giảm dần.

SELECT MASV,HO,TEN

FROM (SINHVIEN *INNER JOIN* DANGKY_HOCPHAN *ON*

SINHVIEN.MASV = DANGKY_HOCPHAN.MASV

WHERE MAHP = ‘CSDL’

ORDER BY TEN *ASC*, HO *ASC*, MASV *DESC*;

Lưu ý : Khi thuộc tính giữa các bảng được truy vấn sau từ khóa From không trùng tên thì ta có thể ghi tường minh tên thuộc tính, mà không cần phải ghi :

<Tên bảng>.<Tên thuộc tính>

Truy vấn chọn có sắp các kết quả trả về theo nhóm (group by)

<u>Ví dụ 1</u> : Tìm tất cả các tên sinh viên đã đăng ký học phần ít nhất 3 học phần trở lên.

SELECT SINHVIEN.MASV, SINHVIEN.HOTEN

FROM DANGKY_HOCPHAN *INNER JOIN* SINHVIEN *ON* DANGKY_HOCPHAN.MASV=SINHVIEN.MASV

GROUP BY SINHVIEN.MASV,SINHVIEN.HOTEN

HAVING *COUNT*(DANGKY_HOCPHAN.MAHP)>=3

Truy vấn chọn lồng nhau (nested/sub query)

- Là câu lệnh truy vấn khi mà trong biểu thức điều kiện của WHERE hoặc HAVING là một câu truy vấn khác.

Ví dụ : Lấy về thông tin của sinh viên có điểm trung bình cao nhất.

SELECT MASV,HOTEN

FROM SINHVIEN

WHERE DIEMTB >=

ALL(*SELECT* DIEMTB *FROM* SINHVIEN)

Các từ khóa trong truy vấn lồng nhau

- ***ANY***, ***SOME*** : Kết quả các bộ trả về của query cha so sánh với 1 trong (bất kỳ) các bộ của query con.
- ***ALL*** : Kết quả các bộ trả về của query cha so sánh với tất cả các bộ của query con.
- ***IN*** : Kết quả các bộ trả về của query cha bằng với 1 trong (bất kỳ) các bộ của query con.
- ***NOT IN*** : Kết quả các bộ trả về của query cha không bằng với bất kỳ bộ nào của query con.
- ***EXISTS*** / ***NOT EXISTS*** : Kết quả các bộ trả về của query cha được thỏa khi query con có tồn tại ít nhất 1 bộ / không tồn tại bộ nào.

Truy vấn lồng nhau – Ví dụ

<u>Ví dụ</u> : Lấy về thông tin của các sinh viên có đăng ký môn học CSDL.

SELECT MASV,HOTEN

FROM SINHVIEN

WHERE MASV ***IN***

(*SELECT* MASV *FROM* DANGKY_HOCPHAN *WHERE* MAHP='CSDL')

Truy vấn lồng nhau – Ví dụ

<u>Ví dụ</u> : Lấy về thông tin của các sinh viên không có đăng ký môn học CSDL.

SELECT MASV,HOTEN

FROM SINHVIEN

WHERE MASV ***NOT IN***

(*SELECT* MASV *FROM* DANGKY_HOCPHAN *WHERE* MAHP='CSDL')

Truy vấn lồng nhau – Ví dụ

<u>Ví dụ</u> : Trả về điểm trung bình cộng của các sinh viên nếu như có ít nhất 1 sinh viên có điểm trung bình >= 5.

SELECT *AVG*(DIEMTB)

FROM SINHVIEN

WHERE ***EXISTS***(*SELECT* DIEMTB *FROM* SINHVIEN *WHERE* DIEMTB>=5)

Truy vấn cập nhật dữ liệu – Cập nhật các bộ

<u>Cú pháp</u> :

UPDATE <TÊN BẢNG> *SET*

<TÊN THUỘC TÍNH 1> = <GIÁ TRỊ 1>,

<TÊN THUỘC TÍNH 2> = <GIÁ TRỊ 2>,

…

<TÊN THUỘC TÍNH n> = <GIÁ TRỊ n>

WHERE <ĐIỀU KIỆN>

Cập nhật các bộ (t.t)

Ví dụ : Cộng thêm 1 điểm cho các sinh viên có điểm trung bình >= 4 và <5

UPDATE SINHVIEN *SET* DIEMTB=DIEMTB+1

WHERE DIEMTB>=4 *AND* DIEMTB<5

Xóa các bộ

Cú pháp :

DELETE FROM <TÊN BẢNG>

WHERE <ĐIỀU KIỆN>

Ví dụ : Xóa các học sinh không có điểm trung bình

DELETE FROM SINHVIEN

WHERE DIEMTB = Null

Thêm các bộ vào quan hệ (bảng)

Cú pháp :

INSERT INTO <TÊN BẢNG>(
<TÊNTHUỘC TÍNH1>,<TÊNTHUỘCTÍNH2>,…)
VALUES(<GIÁ TRỊ 1>, <GIÁ TRỊ 2>, …)

Lưu ý : Các giá trị trong VALUES(…) phải tương ứng với các thuộc tính trong <TÊN BẢNG>(…)

Nếu có thuộc tính nào trong lược đồ quan hệ <TÊN BẢNG> không được khai báo trong <TÊN BẢNG>(…) và VALUES(…) thì giá trị của bộ mới được thêm vào ứng với thuộc tính đó sẽ được đặt bằng Null

THIẾT KẾ CƠ SỞ DỮ LIỆU QUAN HỆ (Relational Database Designing)

Phần II – NGÔN NGỮ TRUY VẤN SQL

(Structured Query Language = ngôn ngữ truy vấn có cấu trúc)

SQL = Structured Query Language

- Là ngôn ngữ dùng để truy vấn dữ liệu
- Ngôn ngữ = cú pháp (cấu trúc ngữ pháp) + các từ khóa (từ vựng) + hàm lập sẵn.
- Là 1 công cụ giao tiếp của Hệ Quản Trị CSDL
- Là cầu nối giữa :
  - Nhà phát triển (Lập trình viên) và Hệ quản trị CSDL
  - Người dùng cuối (End-user) và Hệ quản trị CSDL

SQL = Structured Query Language

- Ngôn ngữ SQL là một <u>chuẩn chung tương đối</u> giữa các Hệ quản trị CSDL khác nhau.
- 1 trong các cú pháp của SQL :

SELECT <tên các thuộc tính>

FROM <tên các quan hệ>

WHERE <điều kiện chọn>

…

Cú pháp SQL – Kiểu Dữ liệu

(data type)

- Chuỗi (String) : được đặt trong dấu nháy kép hoặc đơn.
  - Ví dụ :

```
SELECT *
FROM SINHVIEN
WHERE MASV = “SV01”
```

dữ liệu chuỗi

Cú pháp SQL – Kiểu Dữ liệu (t.t)

(data type)

- Số (number)
  - Ví dụ : 1024 ; 4.5 ; …
- Ngày tháng (date/time) : được đặt trong cặp dấu #, giữa ngày – tháng – năm là dấu phân cách "-" hoặc "/", tên tháng có thể là số (1-12) hoặc viết tắt 3 chữ cái đầu.
  - Ví dụ : #12/2/2001# ; #1-Jan-94# ; …

Cú pháp SQL – Các toán tử số học

(Arithmetic Operations)

Toán tử	Ý nghĩa	Ví dụ	Kết quả
+	Cộng	5 + 2 #28/08/01# + 4	7 #01/09/01#
-	Trừ	#02/09/01# - 3	#30/08/01#
*	Nhân	5 * 2	10
/	Chia	5 / 2	2.5
\	Chia nguyên	5 \ 2	2
^	Lũy thừa	5 ^ 2	25
Mod	Chia dư	5 Mod 2	1

Cú pháp SQL – Các toán tử so sánh

(Comparative Operations)

Toán tử	Ý nghĩa	Ví dụ	Kết quả
<	Nhỏ hơn	3 < 5	True
<=	Nhỏ hơn hay bằng	2 <= 5	True
>	Lớn hơn	2 > 5	False
>=	Lớn hơn hay bằng	2 >= 5	False
=	Bằng nhau	2 = 5	False
<>	Khác nhau	2 <> 5	True

Cú pháp SQL – Các toán tử luận lý

(Logical Operations)

Toán tử	Ý nghĩa	Ví dụ	Kết quả
Not	Luật phủ định	Not (5 > 2) Not (2>5)	False True
And	Luật và	(5>2) And (2>5) (5>2) And (5>4)	False True
Or	Luật hay	(5>2) Or (2>5) (2>5) Or (4>5)	True False

Ví dụ

1. *SELECT* HO,TEN
   FROM SINHVIEN
   WHERE NOT(MASV = ‘SV01’)
2. *SELECT* MASV,HO,TEN
   FROM SINHVIEN
   WHERE (DIEMTB >= 5) *AND* (DIEMTB<=6.5)

Cú pháp SQL – Các toán tử *BETWEEN…AND*

Cú pháp :

value1 *Between* value2 *and* value3

Ý nghĩa : trả về True nếu value1 nằm giữa value2 và value3 ⇔value2<=value1<=value3

Ví dụ :

SELECT *

FROM SINHVIEN

WHERE DIEMTB *BETWEEN* 5 *AND* 6.5

Cú pháp SQL – Các toán tử *LIKE*

Cú pháp :

value1 *LIKE* <khuôn mẫu giá trị>

Ý nghĩa :

_Trả về các value1 có dạng thức giống như <khuôn mẫu giá trị>

Các ký tự đại diện dùng trong khuôn mẫu :

* : đại diện cho tất cả ký tự bất kỳ

? : đại diện cho một ký tự bất kỳ

: đại diện cho 1 ký tự số

[A_1,A_2,…] : đại diện cho 1 ký tự thuộc tập {A_1, A_2, …}

[A_1 – A_2] : đại diện cho 1 ký tự thuộc khoảng ký tự từ A_1 đến A_2

Các toán tử *LIKE* – Ví dụ

SELECT *

FROM SINHVIEN

WHERE TEN *LIKE* '*Hoa'

Chọn tất cả các cột có trong quan hệ

<u>Ý nghĩa</u> : tìm tất cả sinh viên có từ Hoa trong phần cuối của tên, ví dụ : 'Ngọc Thoa', 'Đào Hoa', …

Các toán tử *LIKE* – Ví dụ (t.t)

SELECT *

FROM SINHVIEN

WHERE MASV *LIKE* ‘SV0[1-4]’

Ý nghĩa : tìm tất cả sinh viên có mã sinh viên là SV01, SV02, SV03 hoặc SV04

➔ Toán tử LIKE được sử dụng nhiều trong các truy vấn tìm kiếm dữ liệu

Cú pháp SQL – Các hàm lập sẵn

Cú pháp chung : <tênHàm>(Danh sách đối số)

Hàm **IIf**

Cú pháp : **IIf**(điều kiện,giá trị 1,giá trị 2)

Ý nghĩa : Trả về giá trị 1 nếu điều kiện đúng, ngược lại, trả về giá trị 2.

Ví dụ :

SELECT *

FROM SINHVIEN

WHERE DIEMTB >= *IIF*(GIOITINH='Nam',6.5,6)

Hàm Date

Cú pháp : **Date**()

Ý nghĩa : Trả về giá trị ngày giờ hiện tại của hệ thống.

Ví dụ :

SELECT *

FROM HOADON

WHERE NGAYLAP >= (*DATE()*-5)

Hàm Sum

Cú pháp : **Sum**(<tên thuộc tính>)

Ý nghĩa : Trả về tổng của các giá trị tương ứng với <tên thuộc tính> của tất cả các bộ có trong quan hệ thỏa điều kiện WHERE.

Ví dụ :

SELECT	Sum(GIATRI)
FROM	HOADON
WHERE	NGAYLAP >= (*DATE()*-5)

Ý nghĩa : Trả về tổng giá trị của các hóa đơn có ngày lập trong vòng 6 ngày gần đây.

Hàm Max

Cú pháp : **Max**(<tên thuộc tính>)

Ý nghĩa : Trả về giá trị lớn nhất trong các giá trị tương ứng với <tên thuộc tính> của các bộ có trong quan hệ thỏa điều kiện WHERE.

Ví dụ :

SELECT	Max(GIATRI)
FROM	HOADON
WHERE	NGAYLAP >= (*DATE()*-5)

Ý nghĩa : Trả về giá trị lớn nhất trong các hóa đơn có ngày lập trong vòng 6 ngày gần đây.

Một số hàm khác

- ***Day***(<biểu thức ngày>) : trả về chỉ số của ngày trong <biểu thức ngày>.
  - Ví dụ : ***Day***(#12/2/2005#) → 12
- ***Month***(<biểu thức ngày>) : trả về chỉ số của tháng trong <biểu thức ngày>.
- ***Year***(<biểu thức ngày>) : trả về chỉ số của năm trong <biểu thức ngày>.
- ***Len***(<giá trị chuỗi>) : trả về độ dài của chuỗi

Một số hàm khác (t.t)

– Ví dụ :

SELECT *

FROM SINHVIEN

WHERE LEN(TEN) > 4

- ***Chr***(<mã ASCII>) : trả về ký tự có mã ASCII tương ứng.
- ***InStr***(start,s1,s2) : trả về vị trí của chuỗi s2 trong chuỗi s1 kể từ vị trí start.
- ***LCase***(s) : trả về giá trị chuỗi in thường của chuỗi s
- ***UCase***(s) : trả về giá trị chuỗi in hoa của chuỗi s

Một số hàm khác (t.t)

- ***Left***(s,n) : trả về chuỗi gồm n ký tự bên trái của chuỗi s.
- ***Right***(s,n) : trả về chuỗi gồm n ký tự bên phải của chuỗi s.
- ***Mid***(s,i,n) : trả về chuỗi con của chuỗi s gồm n ký tự kể từ vị trí i.
- ***Nz***(v1,v2) : trả về giá trị v1 nếu v1 khác Null, ngược lại trả về giá trị v2.

Một số hàm khác (t.t)

- ***Min***(<tên thuộc tính>) : trả về giá trị nhỏ nhất trong các giá trị tương ứng với <tên thuộc tính> của các bộ thỏa điều kiện WHERE có trong quan hệ.
- ***Avg***(<tên thuộc tính>) : trả về giá trị trung bình cộng của các giá trị tương ứng với <tên thuộc tính> của các bộ thỏa điều kiện WHERE có trong quan hệ.
- ***Count***(<tên thuộc tính>) : trả về <u>số lượng</u> các giá trị tương ứng với <tên thuộc tính> của các bộ thỏa điều kiện WHERE và khác Null có trong quan hệ.

Các Loại Truy Vấn SQL

1. **Truy vấn chọn** **(Select query)**
   - Là các truy vấn bắt đầu bằng từ khóa *SELECT*
   - Trả về 1 giá trị hoặc 1 tập các bộ
2. **Truy vấn định nghĩa dữ liệu** **(Data Definition Query)**
   - Là các truy vấn bắt đầu bằng từ khóa *CREATE, DELETE, INSERT, ALTER, …*
   - Sử dụng để tạo,thêm,xóa,sửa các bảng (quan hệ), bộ, ràng buộc, … trong CSDL
3. **Truy vấn cập nhật dữ liệu** **(Data Modification Query)**

Truy vấn định nghĩa dữ liệu – Tạo lược đồ quan hệ

Ví dụ 1 :

CREATE TABLE SINHVIEN(

MASV *Text*(10) *CONSTRAINT* k1 *PRIMARY KEY*,

HOTEN *Text*(30), NGAYSINH *Date*, MALOP *Text*(10),

DIEMTB *Double*)

Ghi chú : _ Từ in nghiêng là từ khóa của SQL

_ *Text, Date, Double, ...* : các kiểu dữ liệu (của thuộc tính)

_ *Text*(10) : kiểu dữ liệu Text, có độ dài 10 ký tự

Tạo lược đồ quan hệ (t.t)

_ MASV *Text*(10) *CONSTRAINT* k1 *PRIMARY KEY* :

Khai báo thuộc tính MASV là khóa chính với *quy tắc ràng buộc* tên là k1

<u>Ví dụ 2</u> :

CREATE TABLE BANGDIEM(

MASV *Text*(10), MAMH *Text*(10), DIEM Double, *CONSTRAINT* k2 *PRIMARY KEY* (MASV, MAMH)

)

Thêm,xóa,sửa thuộc tính (cột)

Thêm thuộc tính và quan hệ

Ví dụ :

ALTER TABLE SINHVIEN *ADD COLUMN* GIOITINH *TEXT*(10)

Sửa kiểu dữ liệu của thuộc tính :

ALTER TABLE SINHVIEN *ALTER COLUMN* GIOITINH *BOOLEAN*

Xóa thuộc tính

Ví dụ :

ALTER TABLE SINHVIEN *DROP COLUMN* GIOITINH

Xóa,thêm các ràng buộc

Xóa ràng buộc khóa chính

Ví dụ :

ALTER TABLE SINHVIEN *DROP CONSTRAINT* k1

Thêm ràng buộc khóa chính

Ví dụ :

ALTER TABLE SINHVIEN *ADD CONSTRAINT* k1 *PRIMARY KEY* (MASV)

Thêm ràng buộc miền giá trị lên thuộc tính

Ví dụ :

ALTER TABLE SINHVIEN *ADD CONSTRAINT* k3 *CHECK* (DIEMTB>=0 AND DIEMTB<=10)

Truy vấn chọn

Ví dụ 1 : Chọn tất cả sinh viên có điểm trung bình >= 6.5

SELECT * *FROM* SINHVIEN *WHERE* DIEMTB >= 6.5;

Ví dụ 2 : Chọn 10 sinh viên có điểm trung bình cao nhất

SELECT TOP 10 *FROM* SINHVIEN;

Ví dụ 3 : Chọn 10% sinh viên có điểm trung bình cao nhất

SELECT TOP 10% *FROM* SINHVIEN;

Ví dụ 4 : Chọn có loại bỏ các bộ trùng : chọn các mức điểm khác nhau mà các sinh viên đã đạt được

SELECT DISTINCT DIEMTB *FROM* SINHVIEN;

Lưu ý : Dấu ; cho biết đã kết thúc câu lệnh SQL

Truy vấn chọn từ nhiều bảng

<u>Ví dụ 1</u> : Tìm tất cả các tên học phần mà sinh viên mang mã số SV01 đã đăng ký.

SELECT HOCPHAN.TENHP

FROM SINHVIEN,DANGKY_HOCPHAN,HOCPHAN

WHERE SINHVIEN.MASV = ‘SV01’ *AND*

SINHVIEN.MASV = DANGKY_HOCPHAN.MASV *AND*

DANGKY_HOCPHAN.MAHP = HOCPHAN.MAHP;

<u>Lưu ý</u> : *FROM* $Q_1,Q_2,\ldots,Q_n \Leftrightarrow$

FROM $Q_1 x Q_2 x \ldots x Q_n$ (Tích Descartes)

Truy vấn chọn có kết

Ví dụ 1 : Tìm tất cả các tên học phần mà sinh viên mang mã số SV01 đã đăng ký.

```
SELECT    HOCPHAN.TENHP
FROM      (SINHVIEN INNER JOIN DANGKY_HOCPHAN ON
          SINHVIEN.MASV = DANGKY_HOCPHAN.MASV)
          INNER JOIN HOCPHAN ON
          DANGKY_HOCPHAN.MAHP = HOCPHAN.MAHP
WHERE     MASV = 'SV01';
```

Lưu ý : Phép kết chính là phép chọn có điều kiện từ tích Descartes.

Truy vấn chọn có sắp thứ tự kết quả trả về

Ví dụ 1 : Tìm tất cả các tên sinh viên đã đăng ký học phần có mã là CSDL, sắp thứ tự kết quả trả về theo tên tăng dần, họ tăng dần và mã sinh viên giảm dần.

SELECT MASV,HO,TEN

FROM (SINHVIEN *INNER JOIN* DANGKY_HOCPHAN *ON*

SINHVIEN.MASV = DANGKY_HOCPHAN.MASV

WHERE MAHP = ‘CSDL’

ORDER BY TEN *ASC*, HO *ASC*, MASV *DESC*;

Lưu ý : Khi thuộc tính giữa các bảng được truy vấn sau từ khóa From không trùng tên thì ta có thể ghi tường minh tên thuộc tính, mà không cần phải ghi :

<Tên bảng>.<Tên thuộc tính>

Truy vấn chọn có sắp các kết quả trả về theo nhóm (group by)

<u>Ví dụ 1</u> : Tìm tất cả các tên sinh viên đã đăng ký học phần ít nhất 3 học phần trở lên.

SELECT SINHVIEN.MASV, SINHVIEN.HOTEN

FROM DANGKY_HOCPHAN *INNER JOIN* SINHVIEN *ON* DANGKY_HOCPHAN.MASV=SINHVIEN.MASV

GROUP BY SINHVIEN.MASV,SINHVIEN.HOTEN

HAVING *COUNT*(DANGKY_HOCPHAN.MAHP)>=3

Truy vấn chọn lồng nhau (nested/sub query)

- Là câu lệnh truy vấn khi mà trong biểu thức điều kiện của WHERE hoặc HAVING là một câu truy vấn khác.

Ví dụ : Lấy về thông tin của sinh viên có điểm trung bình cao nhất.

SELECT MASV,HOTEN

FROM SINHVIEN

WHERE DIEMTB >=

ALL(*SELECT* DIEMTB *FROM* SINHVIEN)

Các từ khóa trong truy vấn lồng nhau

- ***ANY***, ***SOME*** : Kết quả các bộ trả về của query cha so sánh với 1 trong (bất kỳ) các bộ của query con.
- ***ALL*** : Kết quả các bộ trả về của query cha so sánh với tất cả các bộ của query con.
- ***IN*** : Kết quả các bộ trả về của query cha bằng với 1 trong (bất kỳ) các bộ của query con.
- ***NOT IN*** : Kết quả các bộ trả về của query cha không bằng với bất kỳ bộ nào của query con.
- ***EXISTS*** / ***NOT EXISTS*** : Kết quả các bộ trả về của query cha được thỏa khi query con có tồn tại ít nhất 1 bộ / không tồn tại bộ nào.

Truy vấn lồng nhau – Ví dụ

<u>Ví dụ</u> : Lấy về thông tin của các sinh viên có đăng ký môn học CSDL.

SELECT MASV,HOTEN

FROM SINHVIEN

WHERE MASV ***IN***

(*SELECT* MASV *FROM* DANGKY_HOCPHAN *WHERE* MAHP='CSDL')

Truy vấn lồng nhau – Ví dụ

<u>Ví dụ</u> : Lấy về thông tin của các sinh viên không có đăng ký môn học CSDL.

SELECT MASV,HOTEN

FROM SINHVIEN

WHERE MASV ***NOT IN***

(*SELECT* MASV *FROM* DANGKY_HOCPHAN *WHERE* MAHP='CSDL')

Truy vấn lồng nhau – Ví dụ

<u>Ví dụ</u> : Trả về điểm trung bình cộng của các sinh viên nếu như có ít nhất 1 sinh viên có điểm trung bình >= 5.

SELECT *AVG*(DIEMTB)

FROM SINHVIEN

WHERE ***EXISTS***(*SELECT* DIEMTB *FROM* SINHVIEN *WHERE* DIEMTB>=5)

Truy vấn cập nhật dữ liệu – Cập nhật các bộ

<u>Cú pháp</u> :

UPDATE <TÊN BẢNG> *SET*

<TÊN THUỘC TÍNH 1> = <GIÁ TRỊ 1>,

<TÊN THUỘC TÍNH 2> = <GIÁ TRỊ 2>,

…

<TÊN THUỘC TÍNH n> = <GIÁ TRỊ n>

WHERE <ĐIỀU KIỆN>

Cập nhật các bộ (t.t)

Ví dụ : Cộng thêm 1 điểm cho các sinh viên có điểm trung bình >= 4 và <5

UPDATE SINHVIEN *SET* DIEMTB=DIEMTB+1

WHERE DIEMTB>=4 *AND* DIEMTB<5

Xóa các bộ

Cú pháp :

DELETE FROM <TÊN BẢNG>

WHERE <ĐIỀU KIỆN>

Ví dụ : Xóa các học sinh không có điểm trung bình

DELETE FROM SINHVIEN

WHERE DIEMTB = Null

Thêm các bộ vào quan hệ (bảng)

Cú pháp :

INSERT INTO <TÊN BẢNG>(
<TÊNTHUỘC TÍNH1>,<TÊNTHUỘCTÍNH2>,…)
VALUES(<GIÁ TRỊ 1>, <GIÁ TRỊ 2>, …)

Lưu ý : Các giá trị trong VALUES(…) phải tương ứng với các thuộc tính trong <TÊN BẢNG>(…)

Nếu có thuộc tính nào trong lược đồ quan hệ <TÊN BẢNG> không được khai báo trong <TÊN BẢNG>(…) và VALUES(…) thì giá trị của bộ mới được thêm vào ứng với thuộc tính đó sẽ được đặt bằng Null

Tổng quan về Acess và SQL - Giới thiệu chung về hệ quản trị cơ sở dữ liệu

Chương 1: Giới thiệu chung về hệ quản trị cơ sở dữ liệu

1.1 Thế nào là một cơ sở dữ liệu?

1.2 Thế nào là một hệ quản trị cơ sở dữ liệu?

1.2.1 Định nghĩa: Một HQTCSDL là một tập hợp chương trình giúp cho người sử dụng tạo ra, duy trì và khai thác một cơ sở dữ liệu: Nó là một hệ thống phần mềm phổ dụng, làm dễ quá trình định nghĩa, xây dựng và thao tác cơ sở dữ liệu cho các ứng dụng khác nhau

Chương 1: Giới thiệu chung về hệ quản trị cơ sở dữ liệu

1.2.2 Một số chức năng quan trọng của HQTCSDL

1. Lưu trữ các định nghĩa, các mối liên kết dữ liệu. Các chương trình truy cập đến cơ sở dữ liệu làm việc thông qua HQTCSDL
2. Tạo ra một hệ thống bảo mật và áp đặt tính bảo mật và riêng tư trong cơ sở dữ liệu.
3. Tạo ra các cấu trúc phức tạp cho phép nhiều người sử dụng truy cập đến cơ sở dữ liệu.
4. Cung cấp các thủ tục sao lưu và phục hồi dữ liệu để đảm bảo sự an toàn và toàn vẹn dữ liệu.
5. Cung cấp việc truy cập dữ liệu thông qua một ngôn ngữ truy vấn.

Chương 1: Giới thiệu chung về hệ quản trị cơ sở dữ liệu

1.3 Giới thiệu một số loại HQTCSDL

Access, Foxpro, DB2, Sysbase (Sysbase System), Oracle, MySQL, SQL Server (6.0, 6.5, 7.0, dòng 2000)

1.4 Ngôn ngữ HQTCSDL

1.5. Ví dụ

1.6 Tổng kết chương: Sinh viên hiểu thế nào là một CSDL và thế nào là một HQTCSDL?

Chương 2: Giới thiệu hệ quản trị cơ sở dữ liệu SQL Server

2.1 Giới thiệu về SQL Server

2.1.1 Giới thiệu SQL Server

- Là sản phẩm về phương tiện cơ sở dữ liệu hàng đầu của Microsoft.
- Chiếm thị phần nhiều nhất.
- Dùng được cho các hệ điều hành Window NT 4.0 (từ Service Pack 5 trở đi), Dòng Windows 2000, Window 98 hay ME.
- Tham khảo http://www.microsoft.com/sql/.

Chương 2: Giới thiệu hệ quản trị cơ sở dữ liệu SQL Server

2.1.2 Lịch sử của SQL Server

- Microsoft cùng với Sysbase phát triển phiên bản SQL Server đầu tiên sử dụng trên hệ điều hành IBM OS/2.
- SQL Server phiên bản 4.2 sau đó là 4.2.1 được phát triển cho hệ điều hành Window NT.
- Sysbase tách khỏi Microsoft phát triển các hệ Sysbase (mới nhất là Adaptive Server Enterprise).
- Microsoft phát triển tiếp các hệ SQL Server 6.0, 6.5 chạy trên Window NT. SQL Server phiên bản 7.0 chạy trên các hệ điều hành dòng Windows 2000, Window 98 hay ME. Sau này là các dòng SQL Server 2000.

Chương 2: Giới thiệu hệ quản trị cơ sở dữ liệu SQL Server

2.1.3 Các ấn bản và yêu cầu phần cứng

- Personal Edition
- Standard Edition
- Enterprise Edition

Yêu cầu phần cứng: Tối thiểu Pentium II, 32 MB

Chương 2: Giới thiệu hệ quản trị cơ sở dữ liệu SQL Server

2.2 Cài đặt và khởi động với SQL Server

Ví dụ về cài đặt và các thiết lập tùy chọn trong quá trình cài đặt

2.3 Tổng kết chương: Sinh viên hiểu sơ lược về hệ quản trị SQL Server, biết cách cài đặt với những tùy chọn.

Chương 3: Quản trị CSDL trong SQL Server

3.1 Các đối tượng và thuộc tính của một cơ sở dữ liệu

- Table (bảng).
- Field (trường) hay Column (cột): Phải được gán một kiểu dữ liệu và phải có tên duy nhất.
- Data Type (Kiểu dữ liệu): Có nhiều kiểu dữ liệu khác nhau, mỗi một kiểu được gán cho một trường nào đó bên trong bảng.
- Primary Key (Khóa chính): Là duy nhất trong các hàng, cung cấp cách xác định duy nhất một bản ghi.
- Foreign Key (Khóa ngoại): Là các cột tham chiếu đến khóa chính hay các ràng buộc duy nhất của một bảng khác.
- Constrains (Ràng buộc): Ràng buộc là cơ chế nhằm đảm bảo tính toàn vẹn dữ liệu, phụ thuộc vào hệ thống và Server.

Chương 3: Quản trị CSDL trong SQL Server

- Defaults (Mặc định): Một mặc định có thể xác lập trên một trường sao cho nếu không có dữ liệu nào được đưa vào thì giá trị mặc định sẽ được dùng đến.
- View (Khung nhìn hay bảng ảo): Về cơ bản là các truy vấn lưu trữ trong cơ sở dữ liệu để có thể tham chiếu tới một hay nhiều bảng. Có thể cho phép nhìn thấy hay không nhìn thấy một số cột trong một hay nhiều bảng nào đó. Cũng có thể dùng View làm kỹ thuật an toàn.
- Stored procedure (Thủ tục lưu trữ): Giống như các Macro, trong đó mã lệnh SQL có thể được viết và lưu trữ mang một tên.
- Triggers (Các bẫy lỗi – còn dịch là bộ kích khởi): Là các thủ tục lưu trữ kích hoạt trước hoặc sau khi bổ xung, sữa chữa hoặc xóa dữ liệu ra khỏi cơ sở dữ liệu.

Chương 3: Quản trị CSDL trong SQL Server

3.2 Khái niệm về quản trị và trình quản trị SQL Server Enterprise Manager

a. Các cách khởi động:

- Start\Program\Microsoft SQL Server\Enterprise Manager.
- Chạy Start\Run\mmc.exe khởi động Console, từ trình đơn File chọn SQL Server Enterprise Manager.MSC.
- Khi quản trị nếu chưa có máy chủ (mặc định là (LOCAL) (Windows NT)) thì cần phải tạo một máy chủ bằng cách: bấm chuột phải vào SQL Server Group chọn New SQL Server Registration, sau đó nhập tên Server và chọn chế độ xác thực. Có thể dùng nhiều máy chủ, tùy từng phiên bản của SQL Server mà có thể đăng ký được bao nhiêu máy chủ.

b. Chức năng: Quản trị cơ sở dữ liệu bằng giao diện đồ họa.

c. Ví dụ

Chương 3: Quản trị CSDL trong SQL Server

3.3 Giới thiệu SQL Query Analyzer

a. Khởi động:

- Ba cách khởi động thứ nhất chọn Tool\SQL Query Analyzer trong SQL Server Enterprise
- Thứ hai Start\Program\Microsoft SQL Server\Query Analyzer.
- Thứ ba chọn Start\Run\isqlw.exe
- Trong cả ba cách chọn thì trình SQL Query Analyzer đều yêu cầu một kết nối tới máy chủ SQL Server.

b. Chức năng:

- Là giao diện chính cho việc chạy các truy vấn SQL và các thủ tục lưu trữ
- Quản trị cơ sở dữ liệu bằng các lệnh SQL, còn SQL Server Enterprise Manager quản trị bằng giao diện.

c. Ví dụ

Chương 3: Quản trị CSDL trong SQL Server

3.4 Ví dụ về tạo, mở, đóng, đổi tên, xóa một cơ sở dữ liệu

a. Tạo: Hai cách hoặc dùng SQL Server Enterprise Manager bằng cách bấm chuột phải vào Databases chọn New Database…

Hoặc dùng cửa sổ SQL Query Analyzer bằng cách gõ tập lệnh:

```
Use master
go
Create Database Vidu on Primary (Name = ViduData,
Filename = 'C:\Program Files\Microsoft SQL
    Server\MSSQL\Data\ViduData.mdf',
Size = 4MB,
Maxsize = 10MB,
Filegrowth = 2MB)
Log ON
(Name = ViduLog,
FileName='C:\Program Files\Microsoft SQL Server\MSSQL\Data\ViduLog.ldf',
size = 2MB,
maxsize = 6MB,
Filegrowth = 1MB)
Go
```

Chương 3: Quản trị CSDL trong SQL Server

b. Xóa:

- Hai cách thứ nhất dùng SQL Server Enterprise Manager bấm chuột phải vào CSDL cần xóa chọn delete
- Thứ hai dùng lệnh SQL trong SQL Query Analyzer

Use master

Go

Drop Database name1, name2,…

Go

c. Muốn đổi tên dùng tập lệnh

use Ten_CSDL_muon_doi_ten

Go

exec sp_dboption Ten_CSDL_muon_doi_ten, 'Single User',true

exec sp_renamedb 'ten_cu', 'ten_moi'

exec sp_dboption ten_moi, 'Single User', false

Go

Trong SQL Query Analyzer.

Chương 3: Quản trị CSDL trong SQL Server

3.2 Bảo mật và Quyền trong SQL Server

3.2.1 Mô hình đăng nhập vào SQL Server

Mô hình ba mức :

Chương 3: Quản trị CSDL trong SQL Server

3.2.2. Mức thứ nhất Sự chứng thực của hệ điều hành Windows (Window 2000 Network Authentication)

- SQL Server được chứa trên một máy chủ cài hệ điều hành Windows, vì vậy trước tiên muốn truy cập đến SQL Server thì phải được phép truy cập vào máy chủ, tức là phải qua được bước chứng thực của hệ điều hành Windows cài trên máy chủ với một phép chứng thực hợp lệ nào đó.
- Ví dụ muốn vào nhà trước tiên phải đi qua cửa.

Chương 3: Quản trị CSDL trong SQL Server

3.2.3. Mức thứ hai là chứng thực đăng nhập vào SQL Server (SQL Server Login Authentication)

Có hai loại thứ nhất là

- Chế độ tích hợp Windows Authentication Mode (chỉ dùng trong hệ điều hành Windows 2000, NT), tức là dùng chính tài khoản đăng nhập vào Windows để đăng nhập vào SQL Server với sự "cho phép" của người quản trị SQL Server, lúc này SQL Server "nhờ" Windows chứng thực tài khoản hộ.
- Ví dụ: Dùng Computer Management\Local User and Groups tạo một tài khoản (hoặc nhóm), đăng nhập SQL Server với tư cách là nhà quản trị sau đó "cấp phép" cho tài khoản (hoặc nhóm) này được đăng nhập vào SQL Server dưới chế độ Windows Authentication.

Chương 3: Quản trị CSDL trong SQL Server

"Cấp phép" bằng hai cách dùng giao diện hoặc bằng câu lệnh SQL:

- *Dùng giao diện:* bằng cách mở cửa sổ Enterprise Manager chọn thư mục Security, bấm chuột phải vào Login chọn new login, bấm chuột vào nút bấm có dấu "…" chọn tài khoản (hoặc nhóm) cần cấp phép và bấm nút Add, sau đó bấm OK, ở trường hợp này như trên ta chọn chế độ Windows Authentication, cuối cùng là bấm OK.
- *Dùng câu lệnh:* Gõ lệnh SQL trong cửa sổ SQL Query Analyzer như sau:

  Use Tên một cơ sở dữ liệu nào đó (mặc định là master)

  Exec sp_grantlogin 'Domain(Server)\tài khoản'

Trong đó *'Domain(Server)\tài khoản'* có thể là một tài khoản hoặc nhóm.

Chương 3: Quản trị CSDL trong SQL Server

- Để gở bỏ quyền trong giao diện đồ họa chỉ đơn giản là bấm chuột phải vào tài khoản (hoặc nhóm) cần xóa và chọn delete.
- Dùng câu lệnh ta gõ lệnh:

Exec sp_revokelogin 'Domain(Server)\tài khoản'

hoặc

Exec sp_denylogin 'Domain(Server)\tài khoản'

Lưu ý: nếu trong trường hợp *'Domain(Server)\tài khoản'* là một nhóm trong Windows thì muốn gỡ bỏ quyền đăng nhập của một người trong nhóm đó nhất thiết phải dùng

Exec sp_denylogin 'Domain(Server)\tài khoản' với *'Domain(Server)\tài khoản'* là tài khoản của người cần gỡ bỏ.

Chương 3: Quản trị CSDL trong SQL Server

Thứ hai là

- Chế độ hỗn hợp Mixed Mode bao gồm chế độ tích hợp trên cộng với chế độ chứng thực SQL Server, chế độ chứng thực SQL Server là chế độ trong đó SQL Server chấp nhận một ID đăng nhập và mật khẩu từ một người sử dụng, và kiểm tra các ủy nhiệm hợp lệ mà không cần bất kỳ một sự giúp đỡ nào từ hệ điều hành Windows. Xảy ra trường hợp này khi trên mạng có những máy không cài hệ điều hành Windows (Unix chẳng hạn) muốn truy cập vào SQL Server.
- Ví dụ: Dùng Computer Management\Local User and Groups tạo một tài khoản, đăng nhập SQL Server với tư cách là nhà quản trị sau đó "cấp phép" cho tài khoản này được đăng nhập vào SQL Server với chế độ SQL Server Authentication với một password nào đó.

Chương 3: Quản trị CSDL trong SQL Server

"Cấp phép" bằng giao diện thì như trên tuy nhiên ở trường hợp này ta chọn chế độ SQL Server Authentication với một password nào đó sau đó cuối cùng là bấm OK.

Dùng câu lệnh thì gõ:

Use Tên một cơ sở dữ liệu nào đó (mặc định là master)

Exec sp_addlogin 'username','password'

Để gỡ bỏ dùng giao diện đồ họa hoặc dùng câu lệnh:

Exec sp_droplogin 'username'

Thay đổi password có thể dùng giao diện đồ họa hoặc câu lệnh

Exec sp_password 'old', 'new', 'username'

Tất nhiên để làm được tất cả các điều này người cấp phép phải đăng nhập SQL Server với tư cách là người quản trị.

Chương 3: Quản trị CSDL trong SQL Server

3.2.4. Yêu cầu chứng thực như một người dùng cơ sở dữ liệu (SQL Server Database User Authentication)

a. Người dùng cơ sở dữ liệu

- Việc có tài khoản truy cập được vào SQL Server không có nghĩa là đã có quyền truy cập vào được một cơ sở dữ liệu cụ thể nào đó. Muốn làm việc được với một cơ sở dữ liệu cụ thể nào đó bạn phải có một tài khoản trong cơ sở dữ liệu đó, tài khoản này nằm trong bảng sysusers của cơ sở dữ liệu đó. Tài khoản này ta còn gọi là người dùng CSDL.
- Mỗi một CSDL tạo ra thì luôn có một người dùng mặc định là chủ của cơ sở dữ liệu đó (dbo), cũng chính là người đã tạo ra CSDL đó, người dùng này hiển nhiên là có toàn quyền đối với CSDL đó.
- Các CSDL mặc định của SQL Server (như master, pubs,…) còn có một người dùng mặc định khác là Guest.

Chương 3: Quản trị CSDL trong SQL Server

- Để tạo một người dùng trong một CSDL cụ thể thông thường ta dùng chính tài khoản đã đăng nhập vào SQL Server luôn (có thể dùng một tên khác) để dễ dàng cho việc nhớ và quản lý.

b. Tạo người dùng trong CSDL

Hai cách

- Thứ nhất dùng giao diện: Chọn CSDL cụ thể mà ta muốn tạo một người dùng, bấm chuột phải vào Users sau đó chọn New Database User, trong mục login name chọn tài khoản mà ta cho phép dùng CSDL này với user name mặc định ta chọn trùng với login name cho dễ quản lý, cuối cùng là bấm OK.

Chương 3: Quản trị CSDL trong SQL Server

- Thứ hai dùng câu lệnh:

Use Tên CSDL

Exec sp_grantdbaccess 'Domain(Server)\tài khoản', 'tên người dùng'

Ví dụ:

Exec sp_grantdbaccess 'TRINHCUONG\test'

c. Gở bỏ

- *Dùng giao diện chỉ đơn giản là bấm chuột phải vào người dùng đó và chọn delete.*
- *Dùng câu lệnh thì gõ lệnh:*

Exec sp_revokedbaccess 'tên người dùng'

Ví dụ: Exec sp_revokedbaccess 'TRINHCUONG\test'

Chương 3: Quản trị CSDL trong SQL Server

3.2.5 Các vai trò (Role)

- Vai trò có nghĩa là tập hợp các quyền được làm một việc gì đấy trong CSDL, có hai mức là vai trò ở mức hệ quản trị SQL Server (ta hay gọi là mức Server) và mức CSDL (mức CSDL).

3.2.5.1 Vai trò ở mức hệ quản trị SQL Server (Serverwide)

a. Định nghĩa các vai trò Serverwide

- Có tám vai trò, không thể thêm hay xóa các vai trò này, mỗi người dùng khi có tài khoản đăng nhập vào SQL Server đều có thể được gán một trong những vai trò này (ta có thể hiểu là các tài khoản đăng nhập có thể được gán một quyền nào đó).

Chương 3: Quản trị CSDL trong SQL Server

- Sysadmin: Thành viên của vai trò này có thể làm bất cứ điều gì trong SQL Server. Chúng xuất hiện là dbo của mỗi CSDL. Về cơ bản chúng ghi đè lên các hệ thống quyền hạn và bảo mật.
- Serveradmin: Thành viên của vai trò này có thể thiết lập các tùy chọn cấu hình và có thể ngừng hoạt động máy chủ.
- Setupadmin: Thành viên của vai trò này có thể cài đặt và định cấu hình liên kết với các máy chủ và đánh dấu một thủ tục để chạy khi khởi động.
- Securityadmin: Thành viên của vai trò này có thể tạo và điều khiển các đăng nhập của máy chủ cũng như các quyền hạn để tạo các CSDL. Có thể xác lập lại các mật khẩu của các đăng nhập SQL Server (ngoại trừ các thành viên của quyền sysadmin).

Chương 3: Quản trị CSDL trong SQL Server

- Processadmin: Thành viên của vai trò này có thể điều khiển quá trình chạy trên CSDL của máy chủ, có thể hủy các truy vấn và hướng dẫn người dùng những việc phải làm khi thực hiện các truy vấn.
- Dbcreator: Thành viên của vai trò này có thể tạo, xóa và sữa các CSDL trên máy chủ, họ cũng có thể khôi phục bất kỳ cơ sở dữ liệu nào trên máy chủ.
- Bulkadmin: Thành viên của vai trò này có thể chạy phát biểu BULK INSERT.
- Diskadmin: Thành viên của vai trò này có thể quản lý các tập tin và sự tăng trưởng của các tập tin, tuy nhiên thường dùng trong SQL Server 6.5 (nên ít được quan tâm ở đây).

Chương 3: Quản trị CSDL trong SQL Server

b. Gán một đăng nhập vào một Server Role

Có hai cách dùng giao diện hoặc dòng lệnh

- Thứ nhất dùng giao diện: Chọn Login trong Security, sau đó bấm chuột phải vào đăng nhập cần gán chọn Properties, chọn mục Server Roles, sau đó tích vào những vai trò mà mình muốn cấp quyền cho đăng nhập. Ta cũng có thể làm tương tự như vậy khi tạo mới một đăng nhập.
- Thứ hai dùng dòng lệnh thì ta gõ lệnh:

Exec sp_addsrvrolemember 'login', 'role'

Ví dụ: Exec sp_addsrvrolemember 'TRINHCUONG\test', 'dbcreator'

Chương 3: Quản trị CSDL trong SQL Server

c. Gỡ bỏ

- Dùng giao diện thì chỉ đơn giản là lặp lại bước trên và bỏ dấu tích vào vai trò mà ta muốn gỡ bỏ.
- Dùng dòng lệnh thì gõ lệnh:

Exec sp_dropsrvrolemember 'login', 'role'

Ví dụ: Exec sp_dropsrvrolemember 'TRINHCUONG\test', 'dbcreator'

3.2.5.2 Vai trò ở mức CSDL

- Mỗi CSDL đều chứa các vai trò, có khoảng 9 vai trò là cố định mà bất kỳ CSDL nào cũng đều có, không thể xóa bất kỳ một vai trò nào trong số này, một số vai trò trong đó là có thể sữa được. Ngoài ra người dùng cũng có thể tự định nghĩa các vai trò cho riêng mình. Mỗi vai trò chỉ ảnh hưởng trong một CSDL cụ thể.

Chương 3: Quản trị CSDL trong SQL Server

a. Các vai trò CSDL cố định

- Db_owner : Thành viên của vai trò này có thể thực hiện bất kỳ điều gì họ muốn, tất nhiên là chỉ ở CSDL mà ta đang xét đến, quyền giống hệt như người dùng dbo ngoại trừ việc không thực hiện được câu lệnh khôi phục CSDL.
- Db_accessadmin : Thành viên của vai trò này có thể thêm hoặc gỡ bỏ quyền truy cập vào CSDL của người dùng (ví dụ chạy thủ tục sp_grantdbaccess).
- Db_securityadmin : Thành viên của vai trò này có thể điều khiển các quyền hạn, các vai trò, các thành viên của vai trò và các chủ nhân của các đối tượng trong CSDL.
- Db_ddladmin : Thành viên của vai trò này có thể tạo, sửa đổi và xóa tất cả các đối tượng của CSDL nhưng họ không thể đưa ra các câu lệnh liên quan đến bảo mật (grant, revoke, deny).

Chương 3: Quản trị CSDL trong SQL Server

- Db_backupoperator : Thành viên của vai trò này có quyền đưa ra các điểm kiểm tra cũng như các câu lệnh sao lưu.
- Db_datareader : Thành viên của vai trò này có quyền select trên bất kỳ bảng, bảng nhìn, hoặc hàm nào trong CSDL.
- Db_datawriter : Thành viên của vai trò này có các quyền chèn, cập nhật, và xóa trên bất kỳ bảng, bảng nhìn trong CSDL.
- Db_denydatareader : Thành viên của vai trò không thể select trên bất kỳ bảng, bảng nhìn hoặc hàm nào trong CSDL.
- Db_denydatawriter : Thành viên của vai trò không thể sữa đổi bất kỳ dữ liệu nào trong CSDL bằng các phát biểu insert, update, hoặc delete trên bất kỳ bảng hoặc bảng nhìn nào.

Chương 3: Quản trị CSDL trong SQL Server

b. Các vai trò CSDL do người dùng tự định nghĩa

- Người dùng có thể thêm các vai trò do mình tự định nghĩa, các vai trò có thể chứa các vai trò khác là thành viên, và tất nhiên một user có thể là thành viên của bất kỳ vai trò nào.

Để tạo một vai trò có hai cách

- Thứ nhất dùng giao diện: Chọn CSDL cần tạo vai trò, bấm chuột phải vào Roles, chọn New Database Role, đặt tên vai trò mà bạn muốn vào hộp name, trong mục Add bạn có thể thêm vào các thành viên của vai trò mà bạn muốn, thành viên có thể là người dùng cũng có thể là các vai trò khác, để gở bỏ thành viên của vai trò chỉ đơn giản bạn chọn thành viên cần gở bỏ và bấm remove, cuối cùng bấm OK. Để xóa một vai trò bạn phải xóa tất cả các thành viên của vai trò sau đó bấm chuột phải vào vai trò cần xóa và chọn delete.

Chương 3: Quản trị CSDL trong SQL Server

- Thứ hai dùng câu lệnh: Để tạo vai trò mới dùng câu lệnh:

Use Tên CSDL cần tạo vai trò

Exec sp_addrole 'tên vai trò', 'tên người dùng là chủ vai trò (nếu không có thì mặc định là dbo)'

Tên vai trò phải là duy nhất trong một CSDL

Để gở bỏ dùng câu lệnh:

Use Tên CSDL cần gở bỏ vai trò

Exec sp_droprole 'tên vai trò'

Để thêm một thành viên vào vai trò dùng câu lệnh:

Use Tên CSDL

Exec sp_addrolemember 'tên vai trò', 'database_account'

Trong đó database_account là tên người dùng, tên nhóm hoặc tên vai trò mà bạn muốn thêm vào vai trò này.

Chương 3: Quản trị CSDL trong SQL Server

Để gở bỏ một thành viên từ một vai trò dùng câu lệnh:

Use Tên CSDL

Exec sp_droprolemember 'tên vai trò', 'database_account'

Database_account giống như trên.

- Ví dụ: Mã lệnh sau trình bày cách thêm vai trò mới trong CSDL pubs, và rồi gán người dùng, vai trò vào nó:

Use Pubs

Exec sp_addrole '5b'

Exec sp_addrole '5c'

Exec sp_addrolemember '5b', 'TRINHCUONG\test'

Exec sp_addrolemember '5c', '5b'

Exec sp_addrolemember '5c', 'Bob'

Chương 3: Quản trị CSDL trong SQL Server

- Ở đây TRINHCUONG\test và Bob đều là tên người dùng CSDL, TRINHCUONG\test là trường hợp mà tên đăng nhập trùng luôn với tên người dùng CSDL (thông thường là như vậy). Lưu ý nếu tài khoản đăng nhập có tên domain (hoặc Server) đàng trước\tên đăng nhập đàng sau ví dụ như dạng TRINHCUONG\test thì đây là dạng đăng nhập Windows Authentication.
- Để xóa Vai trò thì trước hết vai trò phải không chứa thành viên nào. Muốn xóa vai trò 5c ta phải thực hiện như sau:

Use Pubs

Exec sp_droprolemember ‘5c’, ‘5b’

Exec sp_droprolemember ‘5c’, ‘Bob’

Exec sp_droprole ‘5c’

Chương 3: Quản trị CSDL trong SQL Server

3.2.6 Thiết lập việc cấp quyền truy cập CSDL

- Trong các phần trước chúng ta đã biết nếu có một đăng nhập vào SQL Server thì cũng chưa nói lên điều gì về khả năng làm việc với một CSDL cụ thể, thậm chí khi có một tài khoản người dùng trong một CSDL cụ thể thì cũng không có nghĩa là đã được quyền làm mọi việc với CSDL đó, một tài khoản người dùng CSDL có khả năng làm được gì trong CSDL phải qua bước cấp quyền.

Chương 3: Quản trị CSDL trong SQL Server

- Cấp quyền là một cơ chế bảo mật tốt của SQL Server, giúp cho người quản trị quản lý tốt được CSDL, phòng chống được những hư hại có thể xảy ra với CSDL hơn là phải đi tìm cách khắc phục những hư hại có thể xảy ra, người quản trị có thể khống chế được người dùng CSDL bằng cách cấp quyền cho phép họ có thể làm được và không làm được việc gì trong một CSDL cụ thể.
- Có hai loại cấp quyền truy cập CSDL thứ nhất là Quyền phát biểu (Statement Permissions) và thứ hai là Quyền về đối tượng (Object Permissions).

Chương 3: Quản trị CSDL trong SQL Server

3.2.6.1 Thiết lập việc cấp các quyền phát biểu

Có hai cách:

a. Thứ nhất là dùng giao diện

- Bấm chuột phải vào CSDL mà ta muốn cấp quyền chọn mục Properties, chọn tiếp mục Permissions, hiện ra một bảng danh sách các quyền và danh sách các người dùng và vai trò, để cấp quyền ta chỉ đơn giản là tích vào để hiện dấu tích màu xanh, để vô hiệu (deny) một quyền ta tích làm cho dấu đỏ xuất hiện, để gở bỏ ta để trống ô đấy. Lưu ý rằng quyền tạo CSDL (create database) chỉ có trong CSDL master.

Chương 3: Quản trị CSDL trong SQL Server

b. Thứ hai là dùng câu lệnh

- Cấp quyền:

Use Tên CSDL

Grant All hoặc danh sách các quyền To Account

Trong đó Account có thể là tên người dùng CSDL hoặc vai trò.

Ví dụ: *Grant Create table, create view to role1, role2*

Câu lệnh trên gán quyền tạo bảng và tạo view cho vai trò role1 và role2

Grant All to [TRINHCUONG\test]

Câu lệnh này gán tất cả các quyền trong một CSDL cho TRINHCUONG\test

Chương 3: Quản trị CSDL trong SQL Server

- Gở bỏ:

Use Tên CSDL

Revoke All hoặc danh sách các câu lệnh To Account

Trong đó Account tương tự như trên

Ví dụ: *Revoke Create table, create view to role1, role2*

Câu lệnh này gở bỏ quyền tạo bảng và tạo view của role1 và role2

Revoke All to [TRINHCUONG\test]

Câu lệnh này gở bỏ tất cả các quyền của TRINHCUONG\test trong CSDL

Chương 3: Quản trị CSDL trong SQL Server

- Vô hiệu:

Use Tên CSDL

Deny All hoặc danh sách các câu lệnh To Account

Trong đó Account tương tự như trên

Ví dụ: *deny Create table, create view to role1, role2*

Giả sử ta có TRINHCUONG\test là một thành viên của vai trò Role3

Câu lệnh: *Grant create table, create view to Role3*

Lúc này hiển nhiên là TRINHCUONG\test có hai quyền create table và create view, nếu ta không muốn TRINHCUONG\test có quyền tạo bảng ta dùng câu lệnh

deny create table to TRINHCUONG\test

Chương 3: Quản trị CSDL trong SQL Server

3.2.6.2 Thiết lập việc cấp các quyền về đối tượng

a. Người sở hữu CSDL (dbo)

- Người dùng dbo có tất cả các quyền của các thành viên của vai trò db_owner. Mỗi CSDL chỉ có một người dbo
- Được xem là người dbo của CSDL nếu bạn là người tạo lập CSDL, người đăng nhập để tạo CSDL là dbo, hoặc bạn được giao nhiệm vụ sở hữu CSDL thông qua thủ tục lưu trữ hệ thống sp_changedbowner, hoặc bạn kết nối với SQL Server như là thành viên điều hành hệ thống (sysadmin), hay tài khoản sa.

Chương 3: Quản trị CSDL trong SQL Server

b. Người sở hữu đối tượng (dboo)

- Mặc nhiên người dùng nào tạo ra đối tượng CSDL thì là dboo của đối tượng đó, các thành viên của db_owner và db_ddladmin có thể tạo ra các đối tượng và định rõ tên người sở hữu đối tượng. Nếu bạn đăng nhập với quyền sysadmin hoặc dbowner khi tạo đối tượng thì mặc nhiên người sở hữu đối tượng là dbo.
- Để thuận tiện cho việc sử dụng và cấp quyền trong CSDL tốt nhất nên để các đối tượng được sở hữu duy nhất bởi người sơ hữu đối tượng dbo, lý do là vấn đề liên hệ sở hữu đơn và liên hệ sở hữu đứt.

Chương 3: Quản trị CSDL trong SQL Server

c. Gán quyền về đối tượng

Có hai cách dùng giao diện hoặc câu lệnh

- Dùng giao diện: Mở CSDL mà bạn muốn xem hoặc hiệu chỉnh, nhấp chuột phải vào đối tượng mà bạn muốn hiệu chỉnh và chọn mục Properties, sau đó kích vào nút Permissions, lúc này bạn có thể cấp quyền cho một người dùng nào đó sử dụng một quyền nào đó (select, update, …) bằng cách tích dấu xanh vào ô tương ứng, gở bỏ hay vô hiệu hoàn toàn tương tự như cách trước đây. Nếu bạn muốn cấp quyền ở mức cột thì kích vào nút columns và tích vào cột mà bạn muốn cấp quyền, cuối cùng chọn OK.

Chương 3: Quản trị CSDL trong SQL Server

- Dùng dòng lệnh:
  - Lệnh cấp quyền Grant cho phép bất kì ai một hoặc nhiều quyền về đối tượng. Nó cũng đè lên quyền vô hiệu hóa:

Grant All hoặc Tập lệnh [danh sách cột nếu có] On Đối tượng To Account [With Grant Option].

Ví dụ: *Grant select on Chucdanh (tenchucdanhvt, tenchucdanhdd) to TRINHCUONG\test*

Grant select (tenchucdanhvt, tenchucdanhdd) on Chucdanh to TRINHCUONG\test

Grant select, insert on Chucdanh to TRINHCUONG\test with grant option.

Chương 3: Quản trị CSDL trong SQL Server

- Lệnh Gở bỏ Revoke sẽ thu hồi một hay nhiều quyền được gán trước đây:

Revoke All hoặc Tập lệnh [danh sách cột nếu có] On Đối tượng From Account [Cascade].

Ví dụ: *Revoke select (tenchucdanhvt, tenchucdanhdd) on Chucdanh from TRINHCUONG\test Cascade.*

- Lệnh Vô hiệu Deny sẽ vô hiệu hóa những quyền của một đối tượng cụ thể

Deny All hoặc Tập lệnh [danh sách cột nếu có] On Đối tượng To Account [Cascade].

Ví dụ: *Deny select, insert on chucdanh to TRINHCUONG\test*

Chương 3: Quản trị CSDL trong SQL Server

3.2.6.3 Phát sinh kịch bản Script

- Dùng giao diện, kích chuột phải vào CSDL, hoặc đối tượng cần phát sinh kịch bản chọn nút All Tasks , chọn Generate SQL Script, chọn Show All ……

Chương 3: Quản trị CSDL trong SQL Server

3.3 Bài tập

Bài 1: Cơ chế bảo mật trong SQL Server có mấy mức?Hãy trình bày tổng quan từng mức?

Bài 2: Sự khác nhau giữa chế độ đăng nhập một tài khoản (integrated) và chế độ hai tài khoản (SQL Server – authentication)? Khi nào phải dùng chế độ hai tài khoản?

Bài 3: Tôi có một tài khoản người dùng trong Windows, tôi đã được quyền đăng nhập vào SQL Server chưa?để có thể sử dụng được CSDL pubs tôi cần thêm ít nhất là quyền gì?tại sao?

Chương 3: Quản trị CSDL trong SQL Server

Bài 4: Dùng câu lệnh trong SQL Query Analyzer tạo một CSDL tên là Tink5 gồm hai file Tink5_Data.mdf và Tink5_Log.ldf chứa trong thư mục C:\database\fanmem. Kích thước ban đầu của hai file lần lượt là 2MB và 1MB, tốc độ tăng là 1MB, kích thước tối đa của file Tink5_Data.mdf và Tink5_Log.ldf lần lượt là 20MB và 10MB.

- Sau đó dùng câu lệnh trong SQL Query Analyzer đổi tên CSDL trên thành Tin5b_5c.
- Cuối cùng dùng câu lệnh trong SQL Query Analyzer xóa CSDL trên.

Chương 3: Quản trị CSDL trong SQL Server

Bài 5: Có ba đăng nhập trong Windows là 5b, 5c, 5d, Tên máy chủ là HONGDUC. Dùng câu lệnh trong SQL Query Analyzer để tạo ba đăng nhập vào SQL Server dùng chính các tài khoản trong Windows này luôn (chế độ một tài khoản). Dùng tiếp câu lệnh gở bỏ đăng nhập 5d.

- Giả sử đã có CSDL Tink5, viết câu lệnh trong SQL Query Analyzer cho phép hai đăng nhập 5b, 5c ở trên dùng được CSDL này.
- Dùng tiếp câu lệnh gở bỏ đăng nhập 5b không cho phép dùng CSDL Tink5 nữa.

Chương 3: Quản trị CSDL trong SQL Server

Bài 6: Có một nhóm người dùng tên là M trong Windows gồm ba thành viên A, B, C. Tên máy chủ là HONGDUC.

- Dùng câu lệnh cấp phép cho nhóm M được quyền đăng nhập vào SQL Server.
- Dùng tiếp câu lệnh gở bỏ quyền đăng nhập của A vào SQL Server nhưng vẫn giữ nguyên quyền đăng nhập vào SQL Server của nhóm M.

Chương 3: Quản trị CSDL trong SQL Server

Bài 7: Dùng câu lệnh trong SQL Query Analyzer để tạo hai đăng nhập vào SQL Server ở chế độ hai tài khoản (SQL Server – authentication) với cặp username và password lần lượt là (5b, bb) và (5c, cc)

Sau đó đổi password của 5c thành 5ccc. Tiếp theo cấp phép cho 5c được dùng CSDL Tink5.

Dùng tiếp câu lệnh gở bỏ đăng nhập của 5b.

Chương 3: Quản trị CSDL trong SQL Server

Bài 8: Bạn hiểu vai trò (role) là gì? Có mấy mức, là những mức nào? Tại sao lại nên có vai trò?

Bài 9: Giả sử ta có hai tài khoản đăng nhập trong Windows là test1 và test2, tên máy chủ là HONGDUC. Dùng câu lệnh hãy cấp phép cho hai tài khoản này đăng nhập vào SQL Server ở chế độ một tài khoản, sau đó gán cho đăng nhập test1 là thành viên của vai trò dbcreator, test2 là thành viên của vai trò SecurityAdmin, và dbcreator. Sau đó gở bỏ test2 khỏi thành viên của vai trò dbcreator.

Chương 3: Quản trị CSDL trong SQL Server

Bài 10: Thế nào là vai trò ở mức CSDL?

- Có mấy vai trò cố định ở mức CSDL?
- Có thể xóa sửa được các vai trò này không?
- Kể tên và tác dụng của một số vai trò trong số đó?

Chương 3: Quản trị CSDL trong SQL Server

Bài 11: Trong CSDL Tink5, dùng câu lệnh hãy tạo vai trò tên là K5, sau đó dùng tiếp câu lệnh gán cho vai trò này hai quyền tạo bảng và tạo view.

Giả sử có hai người dùng trong CSDL Tink5 là 5b và 5c.

a. Dùng câu lệnh để gán hai người dùng này là thành viên của vai trò K5.

b. Dùng câu lệnh vô hiệu quyền tạo bảng của 5b, nhưng giữ nguyên quyền tạo bảng và tạo view của vai trò K5, và 5b vẫn là thành viên của vai trò K5.

c. Xóa vai trò K5.

Chương 3: Quản trị CSDL trong SQL Server

Bài 12: Giả sử có hai người dùng trong CSDL Tink5 là test1 và test2. Dùng câu lệnh hãy cấp quyền tạo bảng và tạo view cho hai tài khoản người dùng này. Dùng tiếp câu lệnh gở bỏ quyền tạo view của người dùng test1.

Bài 13: Bạn hiểu thế nào là quyền phát biểu (Statement Permissions) và quyền về đối tượng (Object Permissions)?

Chương 3: Quản trị CSDL trong SQL Server

Bài 14: Bạn hiểu thế nào là người sở hữu CSDL dbo?

- Thế nào là người sở hữu đối tượng dboo?
- Tại sao trong một CSDL lại nên để các đối tượng được sở hữu duy nhất bởi người dbo?

Chương 3: Quản trị CSDL trong SQL Server

Bài 15: Giả sử có hai người dùng trong CSDL Tink5 là test1 và test2.

- Hãy dùng câu lệnh cấp cho người dùng test1 quyền select và insert trên bảng Canbo.
- Giả sử bảng Canbo có các trường là (Hoten, Tuoi, Gioitinh, Quequan, Chucvu, Luong), dùng câu lệnh cấp cho test2 quyền select trên các cột (Hoten, Tuoi, Gioitinh, Quequan, Chucvu), và quyền Update trên các cột (Hoten, Tuoi, Gioitinh, Quequan) và cho phép test2 quyền được cấp phép cho các người dùng khác quyền Update này.
- Gở bỏ quyền Update trên bảng Canbo của test2 và tất cả những người đã được test2 cấp cho quyền này.
- Vô hiệu hóa quyền select và insert trên bảng Canbo của test1.

Chương 3: Quản trị CSDL trong SQL Server

3.3.2 Lời giải

Bài 4:

- Tạo CSDL Tink5

Use master

go

Create Database Tink5 on Primary (Name = Tink5_Data,

Filename = 'C:\database\fanmem\Tink5_Data.mdf',

Size = 2MB,

Maxsize = 20MB,

Filegrowth = 1MB)

Chương 3: Quản trị CSDL trong SQL Server

```
Log ON
(Name = Tink5_Log,
FileName='C:\database\fanmem\Tink5_Data.ldf',
size = 1MB,
maxsize = 10MB,
Filegrowth = 1MB)
Go
```

Chương 3: Quản trị CSDL trong SQL Server

- Đổi tên CSDL Tink5 thành Tin5b_5c:

use Tink5

Go

exec sp_dboption Tink5, 'Single User',true

exec sp_renamedb 'Tink5', 'Tin5b_5c'

exec sp_dboption Tin5b_5c, 'Single User',false

Go

Xóa CSDL Tin5b_5c:

Use master

Go

Drop Database Tin5b_5c

Go

Chương 3: Quản trị CSDL trong SQL Server

Bài 5: Cấp phép cho ba đăng nhập 5b, 5c, 5d (chế độ một tài khoản)

Exec sp_grantlogin 'HONGDUC\5b'

Exec sp_grantlogin 'HONGDUC\5c'

Exec sp_grantlogin 'HONGDUC\5d'

Gở bỏ đăng nhập 5d:

Exec sp_revokelogin 'HONGDUC\5d'

Chương 3: Quản trị CSDL trong SQL Server

Cấp phép dùng CSDL Tink5 cho hai đăng nhập 5b và 5c:

Use Tink5

Exec sp_grantdbaccess 'HONGDUC\5b'

Exec sp_grantdbaccess 'HONGDUC\5c'

Gở bỏ người dùng 5b khỏi CSDL Tink5:

Use Tink5

Exec sp_revokedbaccess 'HONGDUC\5b'

Chương 3: Quản trị CSDL trong SQL Server

Bài 6: Cấp phép cho nhóm M được đăng nhập vào SQL Server

Exec sp_grantlogin 'HONGDUC\M'

Gở bỏ quyền đăng nhập của A

Exec sp_denylogin 'HONGDUC\A'

Bài 7: Tạo hai đăng nhập:

Exec sp_addlogin '5b','bb'

Exec sp_addlogin '5c','cc'

Chương 3: Quản trị CSDL trong SQL Server

Đổi password của 5c

Exec sp_password 'cc', '5ccc', 'username'

Cấp phép cho 5c được dùng CSDL Tink5

User Tink5

Exec sp_grantdbaccess '5c'

Gở bỏ đăng nhập 5b

Exec sp_droplogin '5b'

Chương 3: Quản trị CSDL trong SQL Server

Bài 9: Cấp phép cho test1 và test2 đăng nhập vào SQL Server ở chế độ một tài khoản:

Exec sp_grantlogin 'HONGDUC\test1'

Exec sp_grantlogin 'HONGDUC\test2'

Gán vai trò cho test1 và test2

Exec sp_addsrvrolemember 'HONGDUC\test1', 'dbcreator'

Exec sp_addsrvrolemember 'HONGDUC\test2', 'dbcreator'

Exec sp_addsrvrolemember 'HONGDUC\test2', 'SecurityAdmin'

Chương 3: Quản trị CSDL trong SQL Server

Gở bỏ test2 khỏi vai trò dbcreator

Exec sp_dropsrvrolemember 'HONGDUC\test2', 'dbcreator'

Bài 11: Tạo vai trò K5 trong CSDL Tink5:

Use Tink5

Exec sp_addrole 'K5'

Gán quyền cho vai trò K5

Use Tink5

Grant Create table, create view to K5

Chương 3: Quản trị CSDL trong SQL Server

Gán 5b và 5c là thành viên của K5

Use Tink5

Exec sp_addrolemember 'K5', '5b'

Exec sp_addrolemember 'K5', '5c'

Vô hiệu quyền tạo bảng của 5b

Use Tink5

deny Create table to 5b.

Xóa vai trò K5

Use Tink5

Exec sp_droprole 'K5'

Chương 3: Quản trị CSDL trong SQL Server

Bài 12:

Cấp quyền cho hai người dùng test1 và test2

Use Tink5

Grant Create table, create view to test1, test2

Gở bỏ quyền tạo view của người dùng test1

Use Tink5

Revoke Create view to test1

Chương 3: Quản trị CSDL trong SQL Server

Bài 15: Cấp cho người dùng test1 quyền select và insert trên bảng Canbo:

Use Tink5

Grant select, insert on Chucdanh to test1.

Cấp quyền select (Hoten, Tuoi, Gioitinh, Quequan, Chucvu), Update (Hoten, Tuoi, Gioitinh, Quequan) cho test2 và cho phép test2 quyền được cấp phép cho các người dùng khác quyền Update này.

Grant Select on Canbo (Hoten, Tuoi, Gioitinh, Quequan, Chucvu) to test2

Grant Update on Canbo (Hoten, Tuoi, Gioitinh, Quequan) to test2 With Grant Option

Chương 3: Quản trị CSDL trong SQL Server

Gở bỏ quyền Update trên bảng Canbo của test2 và tất cả những người đã được test2 cấp cho quyền này

Revoke Update on Canbo (Hoten, Tuoi, Gioitinh, Quequan) from test2 Cascade.

Vô hiệu hóa quyền select và insert trên bảng Canbo của test1

Deny select, insert on Canbo to test1

Chương 4: Bảng (Table) và Lược đồ (Diagram)

4.1 Khái niệm về bảng

- SQL Server làm việc trên mô hình CSDL quan hệ, nên bảng là nơi chứa dữ liệu thực sự của CSDL.
- Bảng có các thành phần chính là trường (cột-column), bản ghi (dòng dữ liệu), khóa chính có thể có hoặc không, mỗi trường thì có một kiểu dữ liệu nào đó tùy thuộc từng HQTCSDL.
- Một CSDL có thể chứa đến 2 tỉ bảng, mỗi bảng có thể chứa tới 1024 cột, các cột phải được định danh duy nhất trong một bảng theo một quy tắc đặt tên (không dài quá 128 kí tự, ….).

Chương 4: Bảng (Table) và Lược đồ (Diagram)

4.2 Các kiểu dữ liệu

- **Loại String:** Bao gồm các kiểu dữ liệu như char(n), varchar(n), text, nchar, nvarchar, ntext. Trong đó nchar, nvarchar, ntext lưu trữ dữ liệu theo dạng Unicode (mỗi dữ liệu được lưu trữ trong 2 byte).

  Nếu có thông số đàng sau thì chỉ rõ độ dài lớn nhất của chuỗi đó, thiếu thì thêm ký tự trằng vào, thừa thì cắt đi. Nếu không chỉ rõ thì chiều dài của chuỗi là tự tăng khi các ký tự trong chuỗi tăng lên, nhưng điều này trả giá là tốc độ xử lý chậm.

Chương 4: Bảng (Table) và Lược đồ (Diagram)

Ngoài ra nếu chiều dài của chuỗi mà vượt quá 8000 ký tự thì phải dùng kiểu text.

Nếu dùng kiểu text thông thường trong thực tế khi lập trình người ta thường lưu trữ thành một file đặt ở một nơi nào đó trên ổ cứng vật lý còn trong CSDL người ta lưu trữ đường dẫn đến file đó.

Chương 4: Bảng (Table) và Lược đồ (Diagram)

- **Kiểu dữ liệu số chính xác:** Bao gồm các kiểu dữ liệu như bit, tinyint, smallint, int, bigint trong đó bit có chiều dài là 1 byte nhận hai giá trị là 0 hoặc 1, tinyint có chiều dài là 1 byte (0 – 255), smallint chiều dài là 2 bytes (+ – 32767), int có độ dài là 4 bytes (+ – 214783657), bigint có chiều dài là 8 bytes (+ – 2^63).

  Ngoài ra còn hai loại nữa là decimal và numeric, khi khai báo thì phải khai báo độ chính xác của phía phải dấu chấm thập phân (scale), mặc định ban đầu là 0.

Chương 4: Bảng (Table) và Lược đồ (Diagram)

- **Kiểu dữ liệu số gần đúng:** Bao gồm các kiểu dữ liệu real, float, trong đó real yêu cầu 4 bytes nên lưu trữ 7 ký số, còn float 8 bytes nên có thể lưu trữ lên tới 38 ký số.
- **Kiểu dữ liệu Date và Time:** Bao gồm hai loại Datetime và Smalldatetime. Trong đó Datetime gồm 8 bytes lưu trữ phạm vi từ 1/1/1753 – 12/31/9999, Smalldatetime gồm 4 bytes lưu trữ phạm vi từ 1/1/1900 đến 6/6/2079.

  Trong Datetime giờ được lưu trữ đến miliseconds, trong smalldatetime được lưu trữ đến từng phút.

Chương 4: Bảng (Table) và Lược đồ (Diagram)

Lưu ý: Nếu chỉ cần làm việc đến năm, tháng, ngày nên dùng kiểu dữ liệu là số thì sẽ dễ dàng hơn trong quá trình lập trình. Ví dụ để lưu trữ năm ta dùng kiểu dữ liệu smallint, ngày trong tháng dùng tinyint…

- **Kiểu tiền tệ:** Bao gồm hai loại là money và smallmoney, cả hai đều lưu trữ bốn chữ số ở bên phải dấu thập phân, khi nhập dữ liệu tiền tệ phải nhập ở trước dấu $. Trong đó money gồm 8 byte lưu trữ phạm vi từ + - 922337203685447.5808, smallmoney gồm 4 byte lưu trữ phạm vi từ + - 214748.3647.

Chương 4: Bảng (Table) và Lược đồ (Diagram)

- **Kiểu dữ liệu người dùng:** Người dùng có thể tự tạo kiểu dữ liệu do mình muốn dựa vào những dữ liệu do hệ thống cung cấp. Sử dụng thủ tục lưu trữ hệ thống sp_addtype, ví dụ:

Use Pubs

Exec sp_addtype test, 'char(9)','null'

Câu lệnh này thêm kiểu dữ liệu test vào CSDL Pubs, kiểu dữ liệu này có chính là kiểu char(9), và cho phép nhận giá trị null nếu không có giá trị mặc định.

Chương 4: Bảng (Table) và Lược đồ (Diagram)

Có thể tạo bằng giao diện bằng cách bấm chuột phải vào User defined Data Types trong một CSDL nào đó mà mình muốn tạo kiểu dữ liệu mới sau đó chọn New User defined Data Types , các bước sau làm theo hướng dẫn của giao diện.

Để xóa một kiểu dữ liệu người dùng trong giao diện chỉ đơn giản là vào mục User defined Data Types chọn kiểu dữ liệu cần xóa bấm chuột phải chọn delete, bằng câu lệnh thì chạy lệnh:

Use Tên CSDL

Exec sp_droptype tên kiểu dữ liệu cần xóa.

Lưu ý rằng để xóa được kiểu dữ liệu người dùng nào đó thì yêu cầu kiểu DL đó phải không được dùng ở bất kỳ đâu.

Chương 4: Bảng (Table) và Lược đồ (Diagram)

4.3 Làm việc với bảng

4.3.1 Tạo và sửa bảng

Hai cách dùng câu lệnh hoặc giao diện

- Dùng giao diện: Bấm chuột phải vào Table chọn new Table, trong cửa sổ giao diện, sau đó tạo mới các trường và chọn kiểu dữ liệu tương ứng cần thiết. Trong nửa dưới màn hình bạn có thể thiết lập các thuộc tính, chẳng hạn như các diễn giải, các giá trị mặc định, thuộc tính identity, … Khi hoàn tất click vào nút Save và đặt tên cho bảng.

Chương 4: Bảng (Table) và Lược đồ (Diagram)

- Để sửa bảng chỉ đơn giản là bấm chuột phải vào tên bảng cần sửa và chọn Design Table, thực hiện công việc sửa và chọn Save.
- Dùng câu lệnh: *Create Table Tên bảng*

  (Tên trường kiểu dữ liệu Các thuộc tính,

  ...

)

Ví dụ: Create Table Test

(id tinyint Identity Not Null,

Hoten nvarchar(30) Null)

Chương 4: Bảng (Table) và Lược đồ (Diagram)

4.3.2 Xóa bảng:

Hai cách dùng câu lệnh hoặc giao diện

- Dùng giao diện: Bấm chuột phải vào Table chọn delete.
- Dùng câu lệnh: *Drop Table tên bảng*
- Chú ý: Các bảng hệ thống không thể bị xóa, các bảng mà còn đang bị ràng buộc cũng không thể bị xóa.

Chương 4: Bảng (Table) và Lược đồ (Diagram)

4.4 Các loại toàn vẹn

Có ba loại toàn vẹn:

- Toàn vẹn trên miền: Các loại dữ liệu giúp xác định các giá trị gì là hợp lệ cho một cột riêng biệt nào đó điều này được gọi là toàn vẹn miền (domain integrity). Miền đơn giản là một tập các giá trị hợp lệ cho một cột riêng biệt.
- Toàn vẹn tham chiếu: Toàn vẹn tham chiếu liên quan đến việc quản lý các mối quan hệ giữa các dòng dữ liệu trong nhiều bảng.
- Toàn vẹn thực thể: Có nghĩa là bạn có thể nhận diện duy nhất mỗi dòng dữ liệu trong một bảng, đây chính là ràng buộc khóa chính.

Chương 4: Bảng (Table) và Lược đồ (Diagram)

4.5 Các phương pháp toàn vẹn

4.5.1 Các phương pháp truyền thống

4.5.1.1 Dùng các kiểu dữ liệu do người dùng định nghĩa

- Chọn một kiểu dữ liệu cụ thể cho một trường của bảng, dữ liệu được nhập vào trường này sẽ nằm trong miền giới hạn của kiểu dữ liệu đó.
- Ví dụ tạo kiểu dữ liệu mới *test*

Use Pubs

Exec sp_addtype test, 'char(9)','null'

- Trong CSDL Pubs tạo mới một bảng và chọn kiểu dữ liệu của trường trong bảng đó là test, lúc này dữ liệu được nhập vào trường này sẽ có giới hạn là char(9).

Chương 4: Bảng (Table) và Lược đồ (Diagram)

4.5.1.2 Các mặc định (Default Value)

- Giá trị mặc định mà một cột của trường sẽ nhận được khi dữ liệu của cột ấy không được thêm vào ở câu lệnh Insert.
- Để thực hiện điều này mở bảng ở chế độ Design Table, chọn trường cần thiết lập mặc định, sau đó ở nửa dưới màn hình chọn mục Default Value và nhập giá trị mặc định.
- Lưu ý giá trị mặc định phải nằm trong miền giá trị cho phép của kiểu dữ liệu của trường mà bạn muốn thiết lập mặc định.

Chương 4: Bảng (Table) và Lược đồ (Diagram)

4.5.1.3 Các quy tắc (Rules)

- Các quy tắc cũng cố thêm tính toàn vẹn miền bằng cách cung cấp nhiều hơn việc kiểm tra các giá trị hợp lệ. Các quy tắc được sử dụng đảm bảo các giá trị, dùng các biểu thức của ngôn ngữ SQL.
- Ví dụ để tạo mới một quy tắc (Rule), mở CSDL mà ta muốn tạo Rule, bấm chuột phải vào Rules chọn New Rule, trong hộp Name gõ tên Rule mà bạn muốn tạo, ở hộp text bạn gõ cấu trúc: *@tên biểu thức SQL*. Sau đó bấm OK.

Chương 4: Bảng (Table) và Lược đồ (Diagram)

- Ví dụ muốn tạo một Rule tên là myRule mà có quy tắc là giá trị bé hơn 100:

ở hộp Name ta gõ: myRule

ở hộp text ta gõ: @tt [redacted]100

- Nếu ta muốn tạo một Rule có tên là Tink5 mà có quy tắc là những ký tự đầu tiên phải bắt đầu bằng Tink5:

ở hộp Name ta gõ: Tink5

ở hộp text ta gõ: @kk Like 'Tink5%'

Chương 4: Bảng (Table) và Lược đồ (Diagram)

- Để gán một cột nào đó nhận một quy tắc nào đó ta làm như sau: Trong cửa sổ Rules bên phải, bấm chuột phải vào quy tắc cần gán, chọn Properties, ở cửa sổ Rule Properties chọn Bind Columns, ở mục Table, chọn bảng mà chứa cột cần gán, sau đó chọn cột cần gán ở phía trái và bấm Add, tương tự muốn bỏ gán tới một cột nào đó ta chọn cột đó và bấm Remove.
- Lưu ý là kiểu dữ liệu của cột cần gán và biểu thức của quy tắc (Rule) là phải phù hợp với nhau.

Chương 4: Bảng (Table) và Lược đồ (Diagram)

- Ví dụ bảng Vidu2 có cột tên là gg có kiểu tinyint, ta muốn yêu cầu dữ liệu nhập vào cột này phải nhỏ hơn 100, tức là ta gán cho cột này nhận quy tắc myRule ta làm như sau:
- Bấm chuột phải vào myRule, chọn Properties, chọn Bind Columns, ở mục Table chọn bảng Vidu2, chọn cột gg bên trái và bấm Add, cuối cùng là bấm OK.

Chương 4: Bảng (Table) và Lược đồ (Diagram)

4.5.2 Lược đồ và sự toàn vẹn

Ta sẽ thực hiện các loại toàn vẹn bằng giao diện:

4.5.2.1 Thực hiện toàn vẹn thực thể: Mở bảng ở chế độ Design Table, để thiết lập khóa chính cho một bảng chọn cột hoặc các cột mà bạn muốn thiết lập khóa chính rồi click vào biểu tượng chìa khóa màu vàng trên thanh công cụ để thiết lập khóa chính. Yêu cầu khóa chính xác định duy nhất một dòng dữ liệu trong bảng.

Chương 4: Bảng (Table) và Lược đồ (Diagram)

- Nếu không có khóa chính tự nhiên ta có thể dùng thuộc tính Identity (khóa chính tự tăng).
- Nếu dùng Identity có hai thuộc tính là seed và Increment, seed là giá trị tăng bắt đầu, Increment là bước tăng. Ví dụ nếu seed = 2 và Increment = 1 có nghĩa là bắt đầu tăng từ 2 và mỗi lần tăng lên 1.
- Để chuyển giá trị của Identity về lại giá trị seed ban đầu ta dùng câu lệnh

Truncate Table Tên bảng

- Câu lệnh này sẽ xóa toàn bộ các dòng dữ liệu trong bảng và chuyển giá trị của trường khóa Identity về giá trị seed ban đầu.

Chương 4: Bảng (Table) và Lược đồ (Diagram)

4.5.2.2 Thực hiện toàn vẹn miền: Chọn một kiểu dữ liệu cụ thể cho một trường của bảng, dữ liệu được nhập vào trường này sẽ nằm trong miền giới hạn của kiểu dữ liệu đó.

4.5.2.3 Thực hiện toàn vẹn tham chiếu: Ta dùng lược đồ (Diagram), mở CSDL mà bạn muốn làm việc, bấm chuột phải vào Diagrams chọn New Database Diagram, bấm tiếp Next, chọn những bảng cần thực hiện toàn vẹn tham chiếu và bấm Add, bấm tiếp Next và cuối cùng bấm Finish.

Chương 4: Bảng (Table) và Lược đồ (Diagram)

- Ở màn hình làm việc của Diagrams chọn cột của bảng muốn làm quan hệ khóa chính giữ chuột và kéo tới bảng mà ta muốn có quan hệ (cột của bảng này sẽ xem như là khóa ngoại), lúc này sẽ xuất hiện màn hình Create Relationship, chọn các cột tương ứng ở hai bảng, bảng bên trái (ở mục Primary Key Table) sẽ chứa cột là khóa chính, còn bên phải chứa cột là khóa ngoại.

Chương 4: Bảng (Table) và Lược đồ (Diagram)

- Nếu tích vào ô Cascade Update Related Fields có nghĩa là cập nhật ở bảng chính thì cũng cập nhật liên hoàn ở các bảng phụ (các bảng chứa khóa ngoại), còn nếu chọn Cascade Delete Related Records, thì có nghĩa là nếu xóa ở bảng chính thì cũng sẽ xóa tương ứng ở các bảng phụ.
- Lưu ý là các cột khóa chính, khóa ngoại phải có kiểu dữ liệu phù hợp với nhau.
- Để xóa ràng buộc chỉ đơn giản là bấm chuột phải vào ràng buộc và chọn Delete RelationShip From Database.

Chương 4: Bảng (Table) và Lược đồ (Diagram)

4.6 Bài tập

Bài 1: Có cần sử dụng các kiểu dữ liệu do người dung định nghĩa không?Một số quy tắc thông thường phải tuân theo khi thao tác với giá trị số?Một số quy tắc thông thường phải tuân theo khi thao tác với dữ liệu kiểu ký tự? Điểm quan trọng nhất phải nhớ khi tạo các bảng là gì?

Bài 2: Tạo một bảng chứa thông tin về cửa hang, dữ liệu nên lưu trữ bao gồm định danh cửa hang, tên, địa chỉ, thành phố, bang, ZIP, chủ nhân, người liên hệ, số FAX và Email.

Chương 4: Bảng (Table) và Lược đồ (Diagram)

Bài 3: Tạo một bảng chứa thông tin về việc bán hang, dữ liệu nên lưu trữ bao gồm định danh cửa hang, ngày bán, tổng số đã bán, tổng số tiền phải trả và số tiền đặt cọc.

Bài 4: Bạn sẽ tạo các kiểu dữ liệu do người dùng định nghĩa sau như thế nào?

- Zip_Code
- Phone_Number
- Store_id
- Fax
- Email

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

5.1 Làm việc với truy vấn

- Mục tiêu của truy vấn là truy xuất thông tin từ cơ sở dữ liệu, có một số môi trường làm việc như MSQuery, OSQL, SQL English Query nhưng ta quan tâm ở đây là môi trường SQL Query Analyzer.
- Câu lệnh cơ bản của một câu truy vấn là:

Select <column_list>

From <table_list>

Where <search_criteria>

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

- Bạn cũng có thể thay đổi tiêu đề cột cho bằng cách đặt bí danh cho nó, ví dụ như:

Select maChucdanh as Mã_chức_danh, TenChucdanhDD as Tên_Chức_danh_đầy_đủ

From Chucdanh

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

- Bạn cũng có thể dùng Literals để cho ra kết quả dễ đọc hơn, Literal là một chuỗi ký tự được bao quanh những (‘ ‘) trong column_list và được hiển thị như một cột khác trong kết quả truy vấn. Kết quả xuất hiện có một mác cạnh cột thông tin. Ví dụ:

Select maChucdanh, ‘ Loai chuc danh tong quat:’, maloaichucdanh

From Chucdanh

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

5.2 Các hàm và toán tử trong truy vấn

5.2.1 Các hàm toán học

- Các hàm toán học cho phép bạn thực hiện các phép toán cần thiết thông thường trên dữ liệu toán học. Bạn có thể trả về dữ liệu toán học bằng cách sử dụng cụ pháp sau:
- Cú pháp: Select Function_name (Parameters)
- Bảng sau liệt kê một số các hàm cơ bản:

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

Hàm	Kết Quả
ABS(numric_expr)	Giá trị tuyệt đối
ACOS | ASIN | ATAN (Float_expr)	Góc theo radian đối số của hàm cosine, sine hoặc tanget là một giá trị theo kiểu dữ liệu float
COS | SIN | COT | TAN (float_expr)	Là số cosine, sin, hoặc tangent của góc (theo radian)
CEILING (numeric_expr)	Số nguyên nhỏ nhất lớn hơn hoặc bằng với giá trị xác định
DEGREES (numeric_expr)	Chuyển đổi từ radian sang độ

EXP (float_expr)	Giá trị mũ của giá trị xác định
FLOOR (numeric_expr)	Số nguyên lớn nhất nhỏ hơn hoặc bằng giá trị xác định
LOG(float_expr)	Số log tự nhiên
LOG10 (float_expr)	Log cơ số 10
PI ( )	Hằng số 3.141592653589793
POWER (numeric_expr, y)	Giá trị của numeric_expr luỹ thừa y
RADIAN (numeric_expr)	Chuyển đổi từ độ sang radian
RAND([seed])	Số float ngẫu nhiên từ 0 đến 1
ROUND (numeric_expr, len)	Số numeric_expr làm tròn đến chiều dài xác định trong một số nguyên
SIGN (numeric_expr)	Số dương, âm hoặc bằng 0
SQUARE (float_expr)	Bình phương giá trị xác định
SQRT (float_expr)	Căn của giá trị xác định

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

Ví dụ:

Phát biểu	Kết quả
SELECT SQRT (9)	3.0
SELECT ROUND (1234.56, 0)	1235
SELECT ROUND (123.56, 1)	123.60
SELECT ROUND ($1234.56, 1)	1,234.60
SELECT POWER (2, 8)	256.0
SELECT ABS (- 365)	365
SELECT SQUARE (3)	9

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

- Khi bạn dùng kiểu dữ liệu float có thể bạn nhận những kết quả không mong đợi. Nên nói chung bạn nên tránh dùng kiểu dữ liệu float.

5.2.2 Các hàm chuỗi

- Khi bạn thao tác với thông tin ký tự, bạn có thể sử dụng các hàm chuỗi khác nhau để thao tác dữ liệu.
- Cú pháp: Select function_name (parameters)

Hàm	Kết quả
‘Expression’ + ‘Expression’	Kết nối hai hay nhiều chuỗi ký tự.
ASSCII (char_expr)	Trả vê giá trị mã ASCII của ký tự sát bên trái.
CHAR (integer_expr)	Trả về ký tự tương ứng với giá trị mã ASCII
DIFFERENCE (char_expr1, char_expr2)	So sánh hai chuỗi và đánh giá sự giống nhau; trả về một giá trị từ 0 đến 4, 4 là giống nhau nhất
LEFT (char_expr, integer_expr)	Trả về một chuỗi ký tự bắt đầu từ bên trái và đứng trước integer_expr ký tự
LOWER (char_expr)	Chuyển thành chữ thường
LTRIM (char_expr)	Trả về dữ liệu không có khoảng trắng đường trước
NCHAR (integer_expr)	Trả về ký tự theo mã Unicode tương ứng với integer_expr

PATINDEX ('%pattern%',Expression)	Trả về vị trí bắt đầu của lần xuất hiện đầu tiên pattern trong expression
REPLACE ('string1', 'string2', 'string3')	Thay thế tất cả lần xuất hiện string2 trong string1 bằng string3
REPLICATE (char_expr, integer_expr)	Lặp lại chuỗi char_expr với integer_expr lần
REVERSE (char_expr)	Trả về chuỗi đảo ngược của char_expr
RIGHT (char_expr, integer_expr)	Trả về một chuỗi ký tự bắt đầu từ integer_expr ký tự tính từ bên phải
RTRIM (char_expr)	Trả về dữ liệu không có khoảng trắng ở đằng sau
SOUNDEX (char_expr)	Trả về mã bốn chữ số để tính mức tương đương của hai chuỗi ký tự
SPACE (integer_expr)	Trả về một chuỗi lập lại integer_expr khoảng trắng
STUFF (char_expr1,start,length, char_expr2)	Xoá length ký tự khỏi char_expr1 ở vị trí đầu và chèn char_expr2 vào đó
SUBSTRING (expression, start, length)	Trả về một phần chuỗi ký tự hoặc chuỗi nhị phân
UNICODE ('nchar_string')	Trả về giá trị số theo mã Unicode của ký tự đầu tiên trong 'nchar_expr'
UPPER (char_expr)	Chuyển thành chữ hoa

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

Ví dụ:

Phát biểu	Kết quả
SELECT ASCII (‘G’)	71
SELECT PATINDEX (‘%BC%’, ‘ABCDE’)	2
SELECT RIGHT (‘ABCDE’, 3)	CDE
SELECT REVERSE (‘ABCDE’)	EDCBA
SELECT LOWER (‘ABCDE’)	abcde

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

5.2.3 Các hàm ngày tháng

- Bạn có thể biến đổi các giá trị thuộc kiểu dữ liệu datetime bằng cách sử dụng các hàm ngày tháng.
- **Cú pháp:** Select date_function (parameters)
- Bạn phải đặt các giá trị thuộc kiểu dữ liệu datetime truyền vào cho các tham số giữa các dấu nháy đơn hoặc dấu nháy đôi. Một số hàm sử dụng một thông số gọi là datepart. Bảng sau liệt kê các giá trị của datepart và chữ viết tắt của chúng.

Datepart	Viết tắt	Giá trị
Day	dd	1 – 31
Day of Year	dy	1 – 366
Hour	Hh	0 – 23
Millisecond	Ms	0 – 999
Minute	Mi	0 – 59
Month	Mm	1 – 12
Quarter	Qq	1 – 4
Second	Ss	0 – 59
Week	Wk	0 – 53
Weekday	Dw	1 – 7 (Sun – Sat)
Year	Yy	1753 – 9999

Các hàm ngày tháng:

Hàm	Kết quả
DATEDD (datepart, number, date)	Thêm số datepart vào date
DATEDIFF (datepart, date1, date2)	Trả về số datepart giữa hai ngày
DATENAME (datepart, date)	Trả về giá trị ASCII cho giá trị của datepart của ngày date
DATEPART (datepart, date)	Trả về giá trị integer cho giá trị của datepart của ngày date
DAY (date)	Trả về một giá trị số nguyên thể hiện ngày
GETDATE ( )	Trả về ngày và giờ hiện hành
MONTH (date)	Trả về một giá trị số nguyên thể hiện tháng
YEAR (date)	Trả về một giá trị số nguyên thể hiện năm.

Ví dụ:

Hàm	Kết quả
SELECT DATEDIFF (mm, ‘1/1/00’, ‘12/31/02’)	35 – (tháng)
SELECT GETDATE ( )	Apr 27, 2004 2:32 AM
SELECT DATEDD (mm, 6, ‘1/1/04’)	Jul 1. 2004 2:32 AM
SELECT DATEDD (mm, -5, ‘10/6/04’)	May 6, 2004 2:32 AM

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

5.2.4 Các hàm tập hợp

Hàm	Kết quả
AVG ([ALL | DISTINCT] column_name)	Trả về trị trung bình của các giá trị trong biểu thức số, toàn bộ hoặc riêng biệt
COUNT (*)	Trả về số dòng được chọn
COUNT ([ALL | DISTINCT] column_name)	Trả về số các giá trị trong biểu thức, tất cả hoặc riêng biệt
MAX (column_name)	Trả về giá trị lớn nhất trong biểu thức

MIN (column_name)	Trả về giá trị nhỏ nhất trong biểu thức
STDEV (column_name)	Trả về độ lệch chuẩn thống kê của tất cả các giá trị trong tên cột hoặc biểu thức
STDEVP (column_name)	Trả về độ lệch chuẩn thống kê đối với mẫu của tất cả các giá trị trong tên cột hoặc biểu thức
SUM ([ALL | DISTINCT] column_name)	Trả về tổng số các giá trị trong biểu thức số, tất cả hoặc riêng biệt
TOP n [PERCENT]	Trả về n giá trị đầu hoặc n% giá trị trong tập kết quả của bạn
VAR (column_name)	Trả về sự dao động của các giá trị liệt kê trong tên cột hoặc biểu thức

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

5.3 Xây dựng điều kiện trong truy vấn

- Cú pháp: Select column_list
  From table_list
  Where search_conditions

5.3.1 Các phép toán so sánh

- Bạn có thể thực hiện điều kiện bằng các phép toán so sánh như ‘=’, ‘>’, ‘<’, ‘!=’, …
- Ví dụ: Select * from chucdanh where machucdanh = 1

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

5.3.2 Phạm vi

- Bạn có thể trích xuất các dòng dựa trên phạm vi của các giá trị bằng cách sử dụng từ khoá BETWEEN.
- Cú pháp: Select column_list

  From table_list

  Where column_name [NOT] BETWEEN expression AND expression
- Ví dụ: select * from sinhvien where ngaysinh between '1/1/1976' and '2/2/2000'

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

5.3.3 Danh sách

- Bạn có thể trích xuất các dòng với các giá trị mà so khớp chúng trong một danh sách bằng cách sử dụng từ khoá IN.
- Cú pháp: Select column_list

  From table_list

  Where column_name [NOT] IN (value_list)
- Ví dụ: select * from hocnganh2

  where maNganh in (

  select maNganh from nganh where maDonvi=1)

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

5.3.4 Chuỗi ký tự

- Bạn có thể trích xuất các dòng dựa trên các phần của các chuỗi ký tự bằng cách sử dụng từ khoá LIKE.
- Cú pháp: Select column_list

  From table_list

  Where column_name [NOT] LIKE 'string'
- Các ký tự thay thế là:

  % : Chuỗi không có ký tự hay nhiều ký tự

  - : Một ký tự.

  [] : Một ký tự bên trong phạm vi xác định.

  [^] : Một ký tự không nằm bên trong phạm vi xác định.
- Ví dụ: Select * from sinhvien where hosinhvien Like 'Lê%'

  Select * from sinhvien where hosinhvien Like '[VT]%'

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

5.3.5 Các giá trị không xác định

- Một giá trị NULL không giống với các giá trị 0 cũng như khoảng trắng, giá trị NULL có nghĩa là unknown. Trong các trường hợp như vậy một giá trị NULL sẽ không thể tham gia vào các phép so sánh ở trên. Lúc này ta làm việc với giá trị NULL bằng cách kiểm tra IS NULL hoặc IS NOT NULL.
- Ví dụ: Select * from sinhvien Where anhcanhan IS NULL

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

5.3.6 Giới hạn thông tin trùng

- Bạn có thể bỏ thông tin trùng lặp bằng cách dùng DISTINCT
- Ví dụ: Select DISTINCT(gioitinh) from sinhvien

5.3.7 Sắp xếp dữ liệu bằng cách sử dụng mệnh đề ORDER BY

- Bạn có thể sắp xếp kết quả truy vấn của bạn bằng cách sử dụng mệnh đề ORDER BY trong phát biểu Select của bạn
- Cú pháp: Select column_list

  From table_list

  [ORDER BY column_name | column_list [ASC|DESC]
- Ví dụ: Select * from sinhvien ORDER BY tensinhvien ASC

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

5.3.8 Compute

- Mệnh đề Compute sinh ra dòng dữ liệu tổng hợp mới, chúng sử dụng các hàm tập hợp được chỉ rõ trước đây.
- Cú pháp: *Select column_list*

  From table_list

  Where search_criterial

  [Compute] aggregate_expression(s)
- Ví dụ: *Select * from canbo*

  compute Sum(mahocvi)

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

5.4 Các loại truy vấn và Làm việc với cácTruy vấn con

- Một phát biểu Select lồng bên trong phát biểu Select, Insert, Update hoặc Delete khác thường được xem là một truy vấn con (subquery).
- Nếu truy vấn con trả về một giá trị, chẳng hạn như một tổng số, nó có thể sử dụng ở bất kỳ nơi nào mà một giá trị đơn có thể được sử dụng.
- Nếu truy vấn con trả về một danh sách, chẳng hạn như một cột đơn của nhiều giá trị, nó chỉ có thể được sử dụng trong mệnh đề Where.

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

- Truy vấn con luôn luôn được đặt bên trong các dấu ngoặc đơn, ngoại trừ bạn đang thực hiện một truy vấn liên kết, kết thúc trước khi truy vấn ngoài được thực hiện.
- Một truy vấn con có thể chứa các truy vấn con khác, không giới hạn số lượng các truy vấn con.

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

5.4.1 Các loại truy vấn con

- Có hai loại, loại thứ nhất trả về một giá trị đơn, ta có thể dùng với các phép toán như: ‘=’, ‘<’, ‘>’, ‘<>’,….
- Loại thứ hai trả về một cột đơn hoặc nhiều giá trị ta có thể dùng phép toán so sánh của danh sách như IN và EXITS trong mệnh đề Where.
- **Ví dụ:**

1. *select * from hocnganh2*

   where maNganh in (

   select maNganh from nganh where maDonvi=1)

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

2. *select * from hocnganh2*

where maNganh < (select maLoaiChucdanh from chucdanh where maChucdanh = 29)

3. *select * from hocnganh2*

where EXISTS

*(select * from Nganh where maNganh = 1)*

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

5.4.2 Truy vấn con tương quan

- Một truy vấn con tương quan (correlated subquery) tham chiếu tới một bảng từ một truy vấn ngoài và ước lượng mỗi dòng đối với truy vấn ngoài.
- Trong khía cạnh này, một truy vấn tương quan biến đổi từ một truy vấn con bình thường bởi vì truy vấn con phụ thuộc vào các giá trị từ truy vấn bên ngoài.
- Một truy vấn bình thường được thực thi không lệ thuộc vào truy vấn bên ngoài.

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

- **Ví dụ:** Lấy tất cả những sinh viên sống cùng huyện với nhau và có cùng giới tính.

*Select * from sinhvien sv1*

Where gioitinh IN

(select gioitinh from sinhvien sv2

where sv1.maHuyen = sv2.maHuyen

And sv1.maSinhvien <> sv2.maSinhvien)

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

5.4.3 SELECT INTO

- Phát biểu Select Into cho phép bạn tạo một bảng mới dựa trên các kết quả truy vấn.
- Bảng mới bạn tạo dựa trên các cột bạn xác định trong danh sách chọn lựa, các bảng bạn nêu trong mệnh đề from và các dòng bạn chọn trong mệnh đề Where.
- Bạn có thể tạo hai loại bảng với phát biểu Select Into: thường trực và tạm thời.

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

- **Cú pháp:** Select column_list
  INTO new_table_name
  From table_list
  Where search_criteria
- **Chú ý:** Có hai loại bảng tạm, thứ nhất là bảng tạm chỉ tồn tại và có giá trị trong phiên làm việc của người dùng hiện hành, nếu phiên làm việc của người dùng kết thúc bảng tạm sẽ bị xoá, ta tạo bảng tạm như thế bằng cách đặt # trước tên bảng.

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

- Loại thứ hai là tồn tại toàn cục có giá trị trong tất cả các phiên làm việc hiện hành và sẽ bị xoá bỏ khi phiên làm việc của người dùng cuối cùng kết thúc, ta tạo bằng cách đặt ## trước tên bảng.
- Nếu tạo bảng thường trực bạn phải thiết lập tuỳ chọn SELECT INTO/ BULKCOPY của cơ sở dữ liệu, tên bảng định nghĩa phải là duy nhất trong CSDL và tuân theo quy tắc.

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

- **Ví dụ:**

1. *Select hosinhvien, tensinhvien*
   Into #sv1
   From sinhvien
2. *Use DaoTaoHongDuc*
   Go
   Exec sp_dboption 'DaoTaoHongDuc', 'Select into/bulkcopy', true
   go
   *Select * into tmptest from chucdanh*

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

5.4.4 Toán tử Union

- Bạn có thể kết nối kết quả của hai hay nhiều truy vấn thành một tập kết quả đơn bằng cách sử dụng toán tử Union.
- Mặc nhiên hai dòng trùng bị giới hạn, tuy nhiên việc sử dụng Union với từ khoá All trả về tất cả các dòng bao gồm cả các dòng trùng.
- Tất cả các column_list phải có cùng số cột, cùng trật tự cột và có cùng loại dữ liệu.

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

- **Cú pháp:**

Select column_list From table_list Where clause

…

Union [All]

Select column_list From table_list Where clause

…

- **Ví dụ:**

Select tenhuyen, matinh from huyen

union

select null, matinh from tinh

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

5.5 Hiệu chỉnh dữ liệu

5.5.1 Chèn dữ liệu

- Phát biểu Insert cơ bản là một thời điểm bạn có thể thêm một dòng vào trong một bảng, bằng cách biến thể ta cũng có thể thêm nhiều dòng vào bảng bằng cách chọn dữ liệu từ các bảng khác.
- Ví dụ:

insert into uutiengiamtiet(maCanbo, maTruonghopUutienGT, namhoc, hocky)

values(36, 1, 2006, 1)

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

- **Chú ý:** Ta có thể không cần chỉ rõ tất cả các cột, nếu không chỉ rõ giá trị null (nếu cho phép null) hoặc giá trị mặc định (Default Value) sẽ được tự động thêm vào, nếu cả hai dạng này đều không có phát biểu sẽ sinh lỗi.
- Ta cũng có thể thêm cùng lúc nhiều dòng bằng cách dùng truy vấn con Select, nhưng phát biểu Select ở đây khác là không có dấu ngoặc đơn như trên
- **Ví dụ:**

Insert into tmptest(tenchucdanhVT, tenchucdanhDD, maloaichucdanh)

select tenchucdanhVT, tenchucdanhDD, maloaichucdanh from chucdanh

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

5.5.2 Xoá dữ liệu

- Phát biểu delete đơn giản là xoá một hoặc nhiều dòng từ một bảng.

Ví dụ: *Delete from chucdanh where machucdanh = 1*

- Ngoài ra ta cũng có thể thực hiện phát biểu Delete bằng cách dùng bảng dò tìm (hay truy vấn con)

Ví dụ: *Delete from chucdanh*

Where maloaichucdanh in

(Select maloaichucdanh from loaichucdanh)

- Ngoài ra ta cũng có thể sử dụng lệnh Truncate Table.

Ví dụ: *Truncate Table chucdanh.*

Sẽ gở bỏ tất cả dữ liệu từ một bảng chỉ giử lại cấu trúc của bảng.

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

5.5.3 Cập nhật dữ liệu

- Sử dụng phát biểu Update đơn giản là cập nhật lại dữ liệu trên một hoặc nhiều cột ở một hoặc nhiều hàng trong một bảng.
- Ví dụ: *update Chucdanh set*

  tenChucdanhVT='HT',

  tenChucdanhDD='Hieu Truong',

  maLoaiChucdanh=1

  where maChucdanh=30

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

- Ngoài ra cũng như trên ta cũng có thể sử dụng phát biểu Update bằng cách dùng bảng dò tìm hoặc truy vấn con.
- Ví dụ: *Update Chucdanh*

  Set maloaichucdanh =

  (select maloaichucdanh from loaichucdanh where tenloaichucdanh = 'dang')

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

5.6 Sử dụng các phần mở rộng của T-SQL

5.6.1 Các bó (Batch)

- Đơn giản một bó là một tập các phát biểu T-SQL được biên dịch đồng thời bởi SQL Server, phát biểu GO đánh dấu kết thúc của bó.
- Ví dụ: *Use Daotaohongduc*

  *select * from chucdanh*

  *select * from loaichucdanh*

  go
- Chú ý các phát biểu Create phải tạo thành các bó riêng không được liên kết với các phát biểu khác

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

5.6.2 Kịch bản (Script)

- Một kịch bản là một tập của một hoặc nhiều bó được lưu vào một nhóm
- Ví dụ: *Use Daotaohongduc*

*select * from chucdanh*

*select * from loaichucdanh*

go

*Select * from canbo*

Go

- Cách tạo kịch bản đã được đề cập đến trong chương 3 mục 3.2.6.3

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

5.6.3 Chuyển tác (Transaction)

- Một chuyển tác là một đơn vị làm việc trong đó tất cả các phát biểu phải được xử lý thành công, hoặc là không có phát biểu nào được xử lý, đây là vấn đề rất quan trọng trong CSDL phân tán và Hệ phân tán,
- Chúng ta sẽ được nghiên cứu kỹ hơn về chuyển tác trong môn học CSDL nâng cao và Hệ phân tán.

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

5.6.4 Các thành phần của ngôn ngữ Control-of-flow

5.6.4.1 Khối Begin … End

- Đơn giản như trong ngôn ngữ lập trình nếu bạn muốn thực hiện một tập các phát biểu được thực thi với nhau bạn đặt chúng giửa tập lệnh Begin … End.

5.6.4.2 Phát biểu Print

- Từ trước đến giờ ta chỉ biết đến phát biểu Select dùng để trả về kết quả, Print cũng là một phát biểu dạng đó nhưng có sự giới hạn hơn.
- Ví dụ : Print 'Hello'

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

5.6.4.3 Khối If … Else

- Cũng như ngôn ngữ lập trình một khối If…Else là một phát biểu được thực thi tuỳ theo điều kiện.
- Ví dụ: *If (datename(dw, getdate())='Friday')*

```
    Begin
        Print 'Thu 6'
        Print ' Have a good weekend'
    End
Else
    Begin
        Print 'Have a good day'
    End
```

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

5.6.4.4 Biểu thức Case

- Ý nghĩa cũng tương tự như trong ngôn ngữ lập trình.
- Ví dụ: *Select macanbo , day = CASE*

  When khoiluong < 10 Then 'It'

  When khoiluong Between 10 and 50 Then 'Trung binh'

  Else 'Nhieu Qua'

  End

  from thuchiencvkhacday

Chương 5: Truy vấn và sử dụng các phần mở rộng của T-SQL

5.6.4.5 Sử dụng câu lệnh While

- Ý nghĩa cũng tương tự như trong ngôn ngữ lập trình.
- Ví dụ: ***while (select avg(maloaichucdanh) from tmptest)< 9***

  Begin

  Update tmptest set maloaichucdanh maloaichucdanh * 1.1

  Print 'Tang len 10%'

  End
- **Bài tập:** Chương này có tính chất ôn lại về câu lệnh SQL nên yêu cầu sinh viên làm lại các bài tập về các câu lệnh T-SQL đã học trong môn CSDL quan hệ, có áp dụng thêm các phần mở rộng của T-SQL.

Chương 6: View, Trigger, Stored procedure và các function

6.1 View

6.1.1 Giới thiệu

- Các View (hay còn gọi là các bảng nhìn) cho phép bạn chia ngang hoặc chia dọc thông tin từ một hoặc nhiều bảng trong một CSDL.
- Ví dụ: Create View test as

  Select ho, ten from canbo

Chương 6: View, Trigger, Stored procedure và các function

- Về cơ bản các View có các lợi ích thứ nhất là có thể làm đơn giản hoá giao diện người dùng, người dùng lúc này có thể thực hiện những câu truy vấn đơn giản thay vì phải cung cấp các thông số mỗi khi truy vấn được chạy.
- Thứ hai với View bạn có thể thiết lập chế độ an toàn cho phép người dùng được nhìn thấy những gì cần thiết theo ý bạn muốn, nó tối ưu và đơn giản hơn việc bạn phải cấp quyền trên từng cột cho người dùng.

Chương 6: View, Trigger, Stored procedure và các function

6.1.2 Các thao tác

- Tạo View cũng như tạo bảng có hai cách thứ nhất dùng câu lệnh SQL và thứ hai dùng giao diện.
- Dùng câu lệnh cú pháp cơ bản như sau:

Create View tên_view as select_statement.

- Ví dụ: *Create View test as*

  Select ho, ten, machucdanh from canbo,chucdanhcanbo

  where canbo.macanbo=chucdanhcanbo.macanbo

Chương 6: View, Trigger, Stored procedure và các function

- Chỉnh sửa view ta làm như với bảng với lưu ý rằng chỉnh sửa view tức là bạn đang chỉnh sửa trong bảng cơ sở của nó, và không thể chỉnh sửa cùng lúc nhiều hơn một bảng cơ sở.
- Xoá View cũng tương tự như xoá bảng.
- Nếu dùng giao diện bạn bấm chuột phải vào View chọn New View, sau đó bấm chuột phải vào dấu cộng trên thanh công cụ (Add Table), chọn những bảng mà cần dùng để tạo View, bấm add lần lượt từng bảng, cuối cùng bấm close. Từ đây bằng thao tác kéo thả ta có thể thiết kế View theo ý muốn.

Chương 6: View, Trigger, Stored procedure và các function

6.1.3 Các loại và Mục đích sử dụng

- Về cơ bản là có hai loại View, thứ nhất là View projection đơn giản là một tập các cột trong một bảng.
- Thứ hai là View joint bao gồm các cột trong các bảng khác nhau được kết nối lại theo một điều kiện nhất định

6.1.4 Ví dụ

- Thứ nhất là View projection

Create View test as

Select ho, ten from canbo

Chương 6: View, Trigger, Stored procedure và các function

- Thứ hai là View joint

Create View test as

Select ho, ten, machucdanh from canbo,chucdanhcanbo

where canbo.macanbo=chucdanhcanbo.macanbo

Create View test2 as

Select kk1.ho, kk1.ten, kk2.que from kk1,kk2

where kk1.id=kk2.id

Update test2

Set ho ='aa',ten='ss', que =’gg’ báo lỗi vì cập nhật đến nhiều hơn một bảng cơ sở.

Chương 6: View, Trigger, Stored procedure và các function

6.2 Trigger

6.2.1 Giới thiệu

Trigger là một loại thủ tục lưu trữ đặc biệt tự động thực hiện khi bạn hiệu chỉnh, xoá , chèn dữ liệu.

6.2.2 Các thao tác

- Cú pháp tạo mới một Trigger như sau:

Create Trigger tên_Trigger

On table_name (hoặc View_name)

[For ..Alter | INSTEAD OF] {Insert | Update | Delete}

As sql_statement

Chương 6: View, Trigger, Stored procedure và các function

- Ví dụ:

Create Trigger test

On kk1

For Delete as

Delete kk2 From deleted where deleted.id = kk2.id

- Lúc này khi xoá trên bảng kk1, thì Trigger test sẽ tự động được thực hiện và xoá tất cả những record tương ứng trên bảng kk2.

6.2.3 Bảo đảm toàn vẹn dữ liệu bằng Trigger

- Về cơ bản Trigger có tác dụng để đảm bảo toàn vẹn dữ liệu, nhưng từ phiên bản 7.0 của SQL Server việc đảm bảo toàn vẹn dữ liệu đã được thực hiện bằng DRI nên Trigger dần ít được sử dụng

Chương 6: View, Trigger, Stored procedure và các function

- Ví dụ bây giờ ta muốn thêm một record vào bảng kk1, với điều kiện là id ở bảng kk1 đã phải tồn tại ở bảng kk2 (dạng toàn vẹn tham chiếu tương tự như bảng đơn hàng và bảng khách hàng), ta tạo Trigger như sau:

```
Create Trigger hh on kk1
for insert as
if (Select Count(*) from kk2,inserted where kk2.id = inserted.id)=0
Begin
Print 'Khong ton tai id nay'
RollBack Transaction
End
```

- Ngoài ra bạn cũng có thể xem thông tin về Trigger trong một bảng bằng cách chạy thủ tục sau:
- Exec sp_helptrigger tên_bảng.

Chương 6: View, Trigger, Stored procedure và các function

6.3 Stored procedure

6.3.1 Giới thiệu

- Thủ tục lưu trữ hệ thống được xây dựng sẵn bằng các phát biểu T-SQL, được lưu trữ trong một CSDL của SQL Server.
- Lợi ích cơ bản nhất của Stored procedure là khả năng dùng lại và tốc độ thực hiện, nếu trong một phiên làm việc bạn đã triệu gọi một lần thì nó sẽ được lưu lại ở vùng đệm, nến lần triệu gọi sau nó sẽ được gọi trực tiếp từ vùng đệm.
- Ngoài ra bạn cũng có thể gán quyền cho người sử dụng trên một Stored procedure mà không nhất thiết phải gán quyền trên bảng cơ sở

Chương 6: View, Trigger, Stored procedure và các function

6.2.3 Các thao tác

- Tạo Stored procedure bạn bấm chuột phải vào mục Stored Procedures và chọn new Stored Procedures, soạn thảo lệnh tạo có dạng như sau:

Create Procedure Tên_Stored Procedure

@parameter(nếu có) Kiểu dữ liệu của tham số

As

Sql_Statement

Go

Chương 6: View, Trigger, Stored procedure và các function

- **Ví dụ:** ***Create Procedure vidu***

as Select * from chucdanh

Go

- Sau đó chạy thủ tục bằng câu lệnh **Exec vidu** ta sẽ nhận được kết quả là tất cả các cột trên bảng chucdanh
- Bạn cũng có thể tạo một nhóm các thủ tục giống như một bó, phân biệt các thủ tục trong nhóm với thông số **;number**.
- Ví dụ tạo một nhóm thủ tục tên là sp_vidu gồm ba thủ tục ta làm như sau:

Chương 6: View, Trigger, Stored procedure và các function

```
Create Procedure  sp_vidu;1
As   Select * from chucdanh
GO
Create Procedure  sp_vidu;2
As   Select * from canbo
GO
Create Procedure  sp_vidu;3
As   Select * from machucdanh
GO
```

- Nếu bạn muốn chạy một thủ tục nào trong bó bạn chỉ cần thực hiện

Exec sp_vidu;2 lúc này ta sẽ được một danh sách tất cả các cột trong bảng canbo.

- Lưu ý khi xoá bạn không thể xoá từng thủ tục một mà phải xoá cả nhóm.

Chương 6: View, Trigger, Stored procedure và các function

6.3.3 Sử dụng các thông số và tùy chọn

- Các thông số cho phép bạn tạo các thủ tục lưu trữ và truyền giá trị vào cho nó.
- Lúc này thủ tục lưu trữ hoạt động gần giống như chương trình con trong ngôn ngữ lập trình.

Chương 6: View, Trigger, Stored procedure và các function

- **Ví dụ:** Ta tạo thủ tục DTB như sau:

```
Create Procedure DTB
@gt1 smallint,
@gt2 smallint,
@gt3 smallint,
@gt4 smallint,
@gttb smallint output
as
Select @gttb = (@gt1+@gt2+@gt3+@gt4)/4
```

Chương 6: View, Trigger, Stored procedure và các function

- Sau đó ta chạy lệnh

Declare @kk smallint

Exec DTB 10,6, 7, 9, @kk output

Select 'Gia tri TB la:',@kk

Go

- Ngoài ra bạn cũng có thể đặt thêm tuỳ chọn With Recompile vào câu lệnh tạo thủ tục hoặc chạy thủ tục, khi đặt tuỳ chọn này thì mỗi khi thủ tục được triệu gọi thì nó lại được biên dịch lại từ đầu.

Chương 6: View, Trigger, Stored procedure và các function

- **Ví dụ**

Create Procedure vidu With Recompile

*as Select * from chucdanh*

Go

- Muốn buộc tất cả các thủ tục lưu trữ hay các Trigger mà tham chiếu tới một bảng đặc biệt nào đó phải biên dịch lại ở phần chạy kế tiếp ta dùng thủ tục
- Exec sp_recompile Tên bảng.

Chương 6: View, Trigger, Stored procedure và các function

6.3. 4 Ví dụ

```
CREATE PROCEDURE kk1_XOA
   @ma int
 AS
   delete from kk1 where id=@ma
GO
```

- Lúc này ta chạy dòng lệnh

```
exec kk1_Xoa 1
```

- Ta đã truyền tham biến là @ma = 1, thủ tục sẽ xoá trong bảng kk1 record có id = 1.

Chương 6: View, Trigger, Stored procedure và các function

6.4 Các Function

- Một tính năng mạnh của SQL Server 2000 là cho phép người dùng tự tạo ra các hàm do chính mình định nghĩa, chứ không nhất thiết phải phụ thuộc vào các hàm mà hệ thống cung cấp. Việc này rất lợi ích khi làm việc với những ứng dụng lớn có những yêu cầu phức tạp mang tính chuyên nghiệp cao.
- Giá trị trả về của các hàm do người dùng định nghĩa có thể là giá trị vô hướng (giá trị đơn) có thể là một bảng.

Chương 6: View, Trigger, Stored procedure và các function

6.4.1 Các hàm vô hướng

- Là các hàm mà trả về giá trị đơn
- Cú pháp cơ bản như sau:

Create Function Tên hàm(danh sách biến Kiểu dữ liệu tương ứng)

Returns Kiểu dữ liệu trả về

AS

Begin

Thân chương chình…

Return Giá trị trả về

End

Chương 6: View, Trigger, Stored procedure và các function

- Ví dụ 1: Xây dựng một hàm tính tổng của ba số nguyên cho trước.

```
CREATE FUNCTION F_Tong (@x int, @y int, @z int)
RETURNS int
 AS
BEGIN
Declare @gt1   int
Declare @gt2   int
Declare @gt3   int
Declare @tong  int
SET  @gt1 =@x
Set @gt2 = @y
Set @gt3 = @z
Set @tong = @gt1+@gt2+@gt3
Return @tong
END
```

Chương 6: View, Trigger, Stored procedure và các function

- Để gọi hàm ta dùng câu lệnh Select Tên hàm(ds biến)
- Như ví dụ trên ta dùng câu lệnh Select dbo.F_Tong(10, 11, 12) ta sẽ nhận được kết quả là 33.
- Ví dụ 2: Tìm số lớn nhất trong ba số nguyên cho trước.

CREATE FUNCTION F_Max (@x int, @y int, @z int)

RETURNS int

AS

Chương 6: View, Trigger, Stored procedure và các function

```
BEGIN
Declare @gt1 int
Declare @gt2 int
Declare @gt3 int
Declare @max int
Set  @gt1 =@x
Set @gt2 = @y
Set @gt3 = @z
Set @max = @gt1
If @max < @gt2  Set @max = @gt2
If @max < @gt3  Set  @max = @gt3
Return @max
END
```

Chương 6: View, Trigger, Stored procedure và các function

- Lưu ý rằng các câu lệnh và hàm sử dụng trong hàm là các câu lệnh và hàm của SQL.

6.4.2 Các hàm Inline Table – Valued

- Tương tự như hàm trên nhưng giá trị trả về của hàm này là một bảng và không có cặp lệnh Begin…end.
- Cú pháp:

Create Function Tên hàm(danh sách biến Kiểu dữ liệu)

Returns Table

AS

Return Select_Statement

Chương 6: View, Trigger, Stored procedure và các function

- Ví dụ: Trả về danh sách những cán bộ có những ký tự bắt đầu cho trước.

CREATE FUNCTION F_Table1(@bien nvarchar(10))

RETURNS Table

AS

Return (Select * from canbo where ten like @bien+'%')

- Để gọi hàm tương tự như trên ta dùng câu lệnh Select.
- Ví dụ ta muốn lấy về tên những cán bộ có chữ cái bắt đầu bằng ‘t’ ta dùng câu lệnh

Select ten from dbo.F_Table1(‘t’)

Chương 6: View, Trigger, Stored procedure và các function

6.4.3 Các hàm Multistatement Table – Valued

- Tương tự như hàm trên là trả về giá trị là một bảng nhưng ở đây cú pháp có khác và phức tạp hơn, tuy nhiên đổi lại ở đây ta có khả năng tuỳ biến dữ liệu mạnh hơn

Chương 6: View, Trigger, Stored procedure và các function

- Cú pháp:

Create Function Tên hàm(ds biến Kiểu dữ liệu tương ứng)

Returns @Tên bảng Table(danh sách các cột và kiểu dữ liệu tương ứng của bảng được trả về)

AS

Begin

Thân chương trình

Return [( ] select_statement [ )]

End

Chương 6: View, Trigger, Stored procedure và các function

- Ví dụ:

Viết hàm lấy về một bảng chứa các cột ho, ten trong bảng canbo(macanbo, ho ,ten, …) và cột tenchucdanhdd trong bảng chucdanh(machucdanh, tenchucdanhdd , …), liên hệ giữa hai bảng này có bảng ChucdanhCanbo(macanbo, machucdanh, idAo).

CREATE FUNCTION F_Table2()

RETURNS @ketqua Table(ho nvarchar(30), ten nvarchar(20), tenChucdanhDD nvarchar(50))

AS

Chương 6: View, Trigger, Stored procedure và các function

```
Begin
Declare  @temptb Table (ho nvarchar(30), ten nvarchar(20),
    machucdanh smallint)
Insert @temptb
Select ho, ten, machucdanh from canbo, chucdanhcanbo
where canbo.macanbo = chucdanhcanbo.macanbo
Insert @ketqua
Select ho, ten, tenchucdanhdd from @temptb tp, chucdanh cd
where tp.machucdanh=cd.machucdanh
Return
End
```

Chương 6: View, Trigger, Stored procedure và các function

- Ở đây ta thấy khả năng tuỳ biến và xử lý dữ liệu cao hơn hẳn hàm ở mục trước, ta khai báo một bảng tạm @temptb chứa 3 cột ho, ten, machudanh.
- Sau đó mới dùng bảng tạm này kết nối với bảng chucdanh để đưa kết quả vào bảng @ketqua của ta.
- Lưu ý khi dùng bảng tạm thì ta phải đặt bí danh cho nó, ví dụ như trên ta đặt bí danh là tp.
- Gọi hàm thì cũng tương tự như trên ta có thể dùng câu lệnh Select * from dbo.F_Table2().

Chương 6: View, Trigger, Stored procedure và các function

6.5 Bài tập

Bài 1: Bạn tự thiết kế một CSDL sau đó tạo các Trigger, SP, Function, View theo ý bạn tự nghĩ ra.

Bài 2: Bài 2: Có CSDL có ba bảng KhachHang(kh_id, ten, tuoi, quequan, diachi, sdt), bảng DonHang(dh_id, kh_id, ma_hang, soluong), và bảng Hang(ma_hang, ten_hang, gia).

1. Hãy tạo View tên là TongHop chứa các trường (kh_id, ten, ten_hang, soluong, gia, tonggia) với nội dung là liệt kê ra tên tất cả các khách hàng đã mua những mặt hàng gì và với số lượng là bao nhiêu, và có tổng giá là bao nhiêu.

Chương 6: View, Trigger, Stored procedure và các function

2. Hãy tạo một SP với tác dụng như trên và đặt tên là SP_TongHop, và một SP tên là SP_TimKiem lấy ra tất cả những khách hàng (gồm kh_id và tên khách hàng) mà mua hàng với tổng giá trị các mặt hàng lớn hơn một giá trị nhập vào từ bàn phím .
3. Hãy tạo Trigger mà thực hiện toàn vẹn tham chiếu giữa hai bảng (nghĩa là khi thêm mới một bản ghi vào bảng DonHang thì yêu cầu kh_id phải tồn tại trong bảng KhachHang, và khi xoá khách hàng trong bảng KhachHang thì các bản ghi tương ứng trong bảng DonHang cũng phải tự động được xoá).

Chương 6: View, Trigger, Stored procedure và các function

4. Xoá các Trigger trên, và tạo ra Trigger trên hai bảng trên sao cho không cho phép người dùng xoá, sữa gì trên hai bảng đó.
5. Hãy tạo ra UDF (hàm tự định nghĩa) tính giá trung bình của những mặt hàng đã được bán, và tạo ra UDF trả về danh sách giá những mặt hàng đã được bán .

Bài 3: Bạn hãy tạo ra các Function tính min của bốn số nguyên cho trước, Tính trung bình cộng của 5 số cho trước, ….

Chương 6: View, Trigger, Stored procedure và các function

Bài 1: Tự làm

Bài 2:

1. Tạo View TongHop:

- Dùng giao diện sau đó add ba bảng vào và tích vào những trường có trong View, tạo thêm một cột là tonggia = gia*soluong bằng cách ở cột column gõ gia*soluong và ở cột Alias gõ tonggia.
- Câu lệnh tương ứng sẽ như sau:

Chương 6: View, Trigger, Stored procedure và các function

```
CREATE VIEW dbo.TongHop
AS
SELECT     dbo.KhachHang.kh_id, dbo.KhachHang.ten,
   dbo.Hang.ten_hang, dbo.DonHang.soluong, dbo.Hang.gia,
                    dbo.Hang.gia * dbo.DonHang.soluong AS
   tonggia
FROM        dbo.DonHang INNER JOIN
                    dbo.Hang ON dbo.DonHang.ma_hang =
   dbo.Hang.ma_hang INNER JOIN
                    dbo.KhachHang ON dbo.DonHang.kh_id =
   dbo.KhachHang.kh_id
GO
```

Chương 6: View, Trigger, Stored procedure và các function

2. *CREATE PROCEDURE SP_TongHop AS*

SELECT dbo.KhachHang.kh_id, dbo.KhachHang.ten, dbo.Hang.ten_hang, dbo.DonHang.soluong, dbo.Hang.gia,

*dbo.Hang.gia * dbo.DonHang.soluong AS tonggia*

FROM dbo.DonHang INNER JOIN

dbo.Hang ON dbo.DonHang.ma_hang = dbo.Hang.ma_hang INNER JOIN

dbo.KhachHang ON dbo.DonHang.kh_id = dbo.KhachHang.kh_id

GO

Chương 6: View, Trigger, Stored procedure và các function

```
CREATE PROCEDURE SP_TimKiem
@bien int
AS
SELECT     kh_id,ten,Sum(tonggia) as cost
FROM       TongHop
GROUP BY kh_id,ten
HAVING     (SUM(tonggia) > @bien)
GO
```

Chương 6: View, Trigger, Stored procedure và các function

3. *Create Trigger TG_KiemTra on DonHang*

for insert as

if (Select Count(*) from KhachHang,inserted where KhachHang.kh_id = inserted.kh_id)=0

Begin

Print 'Khong ton tai kh_id nay'

RollBack Transaction

End

Create Trigger TG_Xoa

On KhachHang

For Delete as

Delete DonHang From deleted where deleted.kh_id = DonHang.kh_id

Chương 6: View, Trigger, Stored procedure và các function

4. *Drop Trigger TG_KiemTra*

   Drop Trigger TG_Xoa

Create Trigger TG_Cam on KhachHang

for delete, update as

Begin

RollBack Transaction

End

Create Trigger TG_Cam on DonHang

for delete, update as

Begin

RollBack Transaction

End

Chương 6: View, Trigger, Stored procedure và các function

5. ***CREATE VIEW dbo.View_temp***

AS

SELECT dbo.Hang.ma_hang, dbo.Hang.gia

FROM dbo.DonHang INNER JOIN

dbo.Hang ON dbo.DonHang.ma_hang = dbo.Hang.ma_hang

Go

CREATE FUNCTION F_GiaTrungbinh ()

RETURNS smallmoney

AS

BEGIN

Return (select avg(gia) from View_temp)

END

Chương 6: View, Trigger, Stored procedure và các function

```
CREATE FUNCTION F_CacGia()
RETURNS Table
 AS
Return (SELECT     Distinct(gia)
FROM          View_Temp)
```

Chương 7: Sao lưu, Khôi phục và Nhân bản

7.1 Sao lưu

7.1.1 Giới thiệu và ý nghĩa của việc sao lưu

- Với một thế giới đầy hiểm hoạ như ngày nay, và một cuộc sống không có gì là không thể xảy ra thì ta luôn luôn phải lường trước hết được mọi việc kể cả việc toàn bộ CSDL của ta bị hỏng.
- Vì vậy việc sao lưu CSDL là vô cùng quan trọng, dù rằng việc này đem lại sự nhàm chán, và đôi khi bị xem là có tính cẩn thận đàn bà. Nhưng với một hệ thống chuyên nghiệp việc sao lưu CSDL luôn là một phần của hệ thống, và thậm chí người quản trị CSDL phải có hẳn một chiến lược sao lưu làm sao cho hợp lý nhất.

Chương 7: Sao lưu, Khôi phục và Nhân bản

7.1.2 Các thao tác

- Lệnh Sao lưu toàn bộ CSDL.
- Bấm chuột phải vào CSDL cần sao lưu, chọn All tasks, chọn tiếp Backup Database. Ở màn hình SQL Server Backup bạn bấm vào Add, lúc này xuất hiện màn hình Select Backup Destination, ở mục File name bạn chọn đường dẫn đến vị trí sẽ chứa file backup CSDL của bạn. Sau đó bấm OK hai lần.

Chương 7: Sao lưu, Khôi phục và Nhân bản

Lệnh sao lưu một phần CSDL

- Ta có thể thực hiện điều này khi ta đã có một bản sao lưu toàn bộ CSDL.
- Lúc này ở các lần tiếp theo của việc sao lưu bạn không cần phải sao lưu lại toàn bộ CSDL (nếu CSDL lớn thì việc này chiếm rất nhiều thời gian và công sức) mà chỉ cần sao lưu những thay đổi so với bản sao lưu toàn bộ bằng cách như trên nhưng ở màn hình SQL Server Backup lúc này bạn tích vào Database – Differential.
- Lúc này bạn sẽ có bản sao lưu CSDL tại thời điểm mà bạn tạo backup.
- Bạn có thể đặt một tên khác cho bản sao lưu này hay đơn giản bạn dùng luôn tên của bản sao lưu toàn bộ nhưng hệ thống sẽ tự động phân biệt cho bạn.

Chương 7: Sao lưu, Khôi phục và Nhân bản

- Ngoài ra bạn cũng có thể sao lưu bằng cách copy các file mdf và ldf của CSDL cần sao lưu vào một thiết bị sao lưu nào đó.
- Lúc khôi phục lại CSDL nếu tên CSDL vẫn đang còn bạn bấm chuột phải vào CSDL đó chọn All task, chọn tiếp Detach Database, bấm OK (nếu như còn một chương trình nào đó đang dùng CSDL này bạn bấm vào Clear).

Chương 7: Sao lưu, Khôi phục và Nhân bản

- Sau đó bạn bấm chuột phải vào mục Databases, chọn All task, chọn tiếp Attach Database, bấm vào dấu … chọn đường dẫn đến nơi chứa file mdf của CSDL mà bạn cần khôi phục, bấm OK.
- Lưu ý: Nếu bạn Detach Database thì các file mdf và ldf không mất đi, khác với khi bạn chọn delete CSDL các file mdf và ldf cũng bị xoá luôn

Chương 7: Sao lưu, Khôi phục và Nhân bản

7.1.3 Ví dụ

- Tạo một CSDL tên là test, tạo một bảng tên là test1(ten, tuoi) nhập dữ liệu cho bảng đó. Sau đó thao tác y hệt như trên, ở mục File name đặt tên file là test_backup.

Chương 7: Sao lưu, Khôi phục và Nhân bản

7.2 Khôi phục

7.2.1 Giới thiệu và những trường hợp của khôi phục

- Tất nhiên đã có sao lưu thì phải có quá trình khôi phục nếu không thì sao lưu chẳng mang một ý nghĩa gì, nếu có hỏng hóc của CSDL thì ta có thể khôi phục bất kỳ phiên bản nào mà ta muốn, và khôi phục bất kỳ CSDL nào mà ta muốn (tất nhiên là phải có bản sao lưu của CSDL đó).

Chương 7: Sao lưu, Khôi phục và Nhân bản

7.2.2 Các thao tác

Khôi phục toàn bộ CSDL:

- Nếu CSDL của bạn bị hỏng hóc bạn muốn sao lưu lại thì bấm chuột phải vào CSDL cần sao lưu chọn All Tasks, chọn tiếp Restore Database, ở màn hình Restore Database tích vào From device, sau đó bấm vào Select Devices.
- Ở màn hình Choose Restore Devices bấm vào Add, ở màn hình Choose Restore Destination ở mục File name ta trỏ đường dẫn đến file backup của ta sau đó bấm OK liên tiếp là ta đã hoàn tất quá trình sao lưu.

Chương 7: Sao lưu, Khôi phục và Nhân bản

- Lưu ý: Nếu CSDL của ta bị xoá hẳn, bạn phải tạo mới một CSDL với tên như cũ sau đó lặp lại quá trình trên.
- Và nếu CSDL của bạn đã được Restore một lần thì ở các lần Restore tiếp theo bạn không cần lặp lại quá trình trên mà ở màn hình Restore Database hệ thống sẽ hiện ra bản backup CSDL toàn bộ cuối cùng của bạn và bạn chỉ đơn giản là bấm Ok.

Khôi phục một phần CSDL:

- Tương tự như trên nhưng ở màn hình Restore Database lúc này sẽ hiện ra một danh sách các lần sao lưu một phần CSDL và bạn chỉ việc tích vào bản sao lưu nào cần khôi phục mà bạn muốn.

Chương 7: Sao lưu, Khôi phục và Nhân bản

7.2.3 Ví dụ

- Như ví dụ trên bạn chỉ cần thêm mới, xoá, sữa các bảng rồi ở các thời điểm đó bạn tạo các bản sao lưu một phần sau đó khôi phục lại bạn sẽ thấy tác dụng của lệnh sao lưu một phần.

7.3 Nhân bản

7.4 Bài tập

- Bạn tự thực hành tạo ra một CSDL sau đó thay đổi CSDL đó theo ý của bạn và lưu lại các bản sao lưu sao cho bạn có thể thực hiện được các thao tác ở trên như sao lưu toàn bộ CSDL và sao lưu một phần CSDL.

Chương 8: Xây dựng phần mềm dùng HQT SQL Server

8.1. Khái niệm Application

8.1.1. Application có CSDL và Application không có CSDL

Application hay ta còn hiểu là một phần mềm tin học thực hiện một công việc về một lĩnh vực nào đó.

Về cơ bản có các dạng phần mềm sau:

- Phần mềm hệ thống
- Phần mềm thời gian thực
- Phần mềm nghiệp vụ
- Phần mềm khoa học kỹ thuật
- Phần mềm nhúng
- Phần mềm máy tính cá nhân
- Phần mềm trí tuệ nhân tạo
- Phần mềm công cụ

Chương 8: Xây dựng phần mềm dùng HQT SQL Server

- Các loại phần mềm này có thể có CSDL hoặc có thể không có CSDL, CSDL rất đa dạng có thể là một định dạng đặc biệt nào đó như SQL, Oracle, …, cũng có thể chỉ đơn giản là một file text.

8.1.2. Vai trò, vị trí của HQT SQL Server nằm ở đâu trong Application

- Nếu phần mềm có CSDL thì lúc này Phần mềm = Code + CSDL, CSDL như ta nói có lịch sử phát triển và rất đa dạng, và để tăng tốc độ truy cập từ chương trình vào CSDL và khả năng lưu giữ dữ liệu các định dạng đặc biệt của CSDL ra đời, SQL Server là một trong số những định dạng đó.

Chương 8: Xây dựng phần mềm dùng HQT SQL Server

8.2 Kiến trúc và mô hình kết nối từ Application vào CSDL

- Ta đã biết CSDL rất đa dạng, chúng có các định dạng đặc biệt khác nhau, từ đơn giản như file text đến các file định dạng đặc biệt như SQL Server, Oracle, ….
- Vì thế cho nên khó khăn lớn nhất khi lập trình truy xuất CSDL đó là cơ chế kết nối, đọc và xử lý dữ liệu từ các file của những HQTCSDL. Chúng có cấu trúc khác nhau và có cách xử lý riêng biệt.

Chương 8: Xây dựng phần mềm dùng HQT SQL Server

- Điểm chung nhất ta đã biết giữa các HQTCSDL đó là ngôn ngữ SQL, nhưng để sử dụng được câu lệnh SQL tác động vào CSDL, lập trình viên cần sử dụng các trình điều khiển do mỗi HQTCSDL cung cấp.
- Nhưng học cách sử dụng các hàm của trình điều khiển là rất khó khăn đối với các lập trình viên.
- Microsoft đã giải quyết vấn đề này bằng cách xây dựng một tập các hàm giao tiếp tổng quát (API) dành cho kết nối và truy xuất dữ liệu, ta gọi đó là cầu nối ODBC.
- Các chương trình Windows muốn truy xuất dữ liệu lúc này chỉ đơn giản là đăng ký trình điều khiển của HQTCSDL mà mình muốn truy xuất với bộ quản lý ODBC.

Chương 8: Xây dựng phần mềm dùng HQT SQL Server

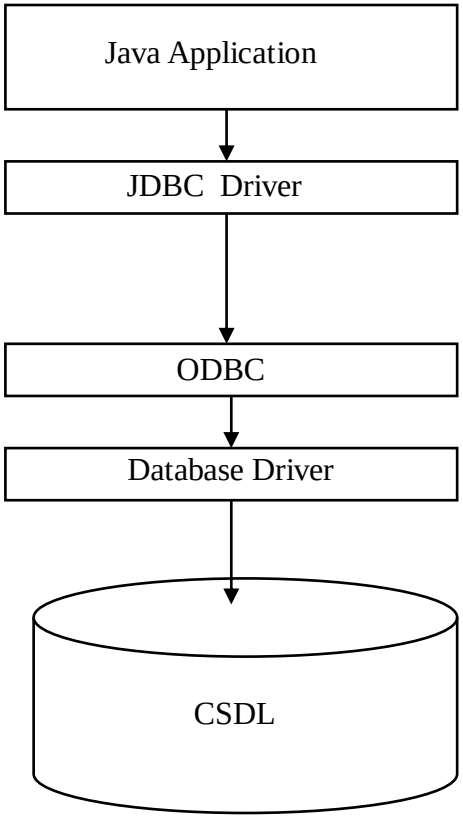
8.3. Giới thiệu các loại Ngôn ngữ lập trình và các Mô hình kết nối cụ thể vào HQT CSDL SQL Server

8.3.1 Java và mô hình ODBC

- Ngôn ngữ lập trình Java do Sun phát triển, là ngôn ngữ hướng đối tượng, mạnh cho các ứng dụng mạng, cung cấp nhiều công cụ free bởi các hãng thứ ba, được dùng nhiều hiện nay.
- Trình điều khiển JDBC do Java cung cấp, lập trình viên lúc này có khả năng sử dụng JDBC để gọi ODBC và bằng cách này lập trình viên có thể kiểm soát rất nhiều HQTCSDL khác nhau.
- Sun cung cấp sẵn thư viện kết nối JDBC – ODBC kèm theo bộ JDK và dùng miễn phí.

Chương 8: Xây dựng phần mềm dùng HQT SQL Server

- Kiến trúc:



Chương 8: Xây dựng phần mềm dùng HQT SQL Server

- **Câu lệnh kết nối:**

```
String drivername="sun.jdbc.odbc.JdbcOdbcDriver";
    String connectionURL="jdbc:odbc:bkshop";
    String username="";
    String password="";
    Connection con = null;
    Statement stmt = null;
    ResultSet rs = null;
        Class.forName(drivername).newInstance();
con = DriverManager.getConnection(connectionURL,username,password);
    stmt = con.createStatement();
String SqL="select  *  from SellAgent" ;
            rs = stmt.executeQuery(SqL);
```

Chương 8: Xây dựng phần mềm dùng HQT SQL Server

- Ở đây bkshop là tên ODBC Data source mà bạn đăng ký với ODBC, bước đăng ký như sau:
- Vào Administrative Tools, click đúp chuột vào Data sources (ODBC), chọn mục System DSN, chọn tiếp Add, sau đó chọn trình điều khiển tương ứng với HQTCSDL bạn dùng, ở đây ta dùng SQL Server nên ta chọn SQL Server, click Finish, tiếp theo ở mục Name ta gõ tên ODBC Data source, trong trường hợp này ta gõ là bkshop, ở mục Server ta gõ tên Server của HQTCSDL, ở màn hình tiếp theo bạn phải chọn chế độ đăng nhập vào HQTCSDL

Chương 8: Xây dựng phần mềm dùng HQT SQL Server

- Ở đây ta có thể chọn chế độ một tài khoản, hoặc chế độ đăng nhập bằng User và password.
- Nếu chọn chế độ một tài khoản thì không cần quan tâm đến user và password, nhưng nếu chọn chế độ đăng nhập bằng User và password thì bạn phải điền đúng user và password.
- Click next ở màn hình tiếp theo bạn tích vào mục *change the default database to* để chọn CSDL mà bạn muốn truy cập tới, Click next liên tục và bấm OK.

Chương 8: Xây dựng phần mềm dùng HQT SQL Server

- Ngoài ra có thể bạn dùng JDBC kết nối trực tiếp với các trình điều khiển của CSDL mà không phải thông qua cầu nối ODBC của Win.
- Tất nhiên lúc này các kết nối này nhanh và hiệu quả hơn so với sử dụng cầu nối ODBC nhưng nhược điểm đó là bạn phải mua các trình điều khiển CSDL đặc thù do những nhà phát triển CSDL hay bên thứ 3 (third – party) cung cấp.

Chương 8: Xây dựng phần mềm dùng HQT SQL Server

- Một cách nữa cũng được dùng nhiều hiện nay đó là JDBC kết nối thông qua các ứng dụng mạng trung gian.
- Theo mô hình lập trình phân tán, ứng dụng trung gian (middle – ware) trên một máy chủ sẽ chịu trách nhiệm kết nối với CSDL.
- Nhiệm vụ của trình khách là sử dụng những trình điều khiển JDBC gọn nhẹ, có khả năng giao tiếp với ứng dụng trung gian qua Internet để truy vấn dữ liệu.

Chương 8: Xây dựng phần mềm dùng HQT SQL Server

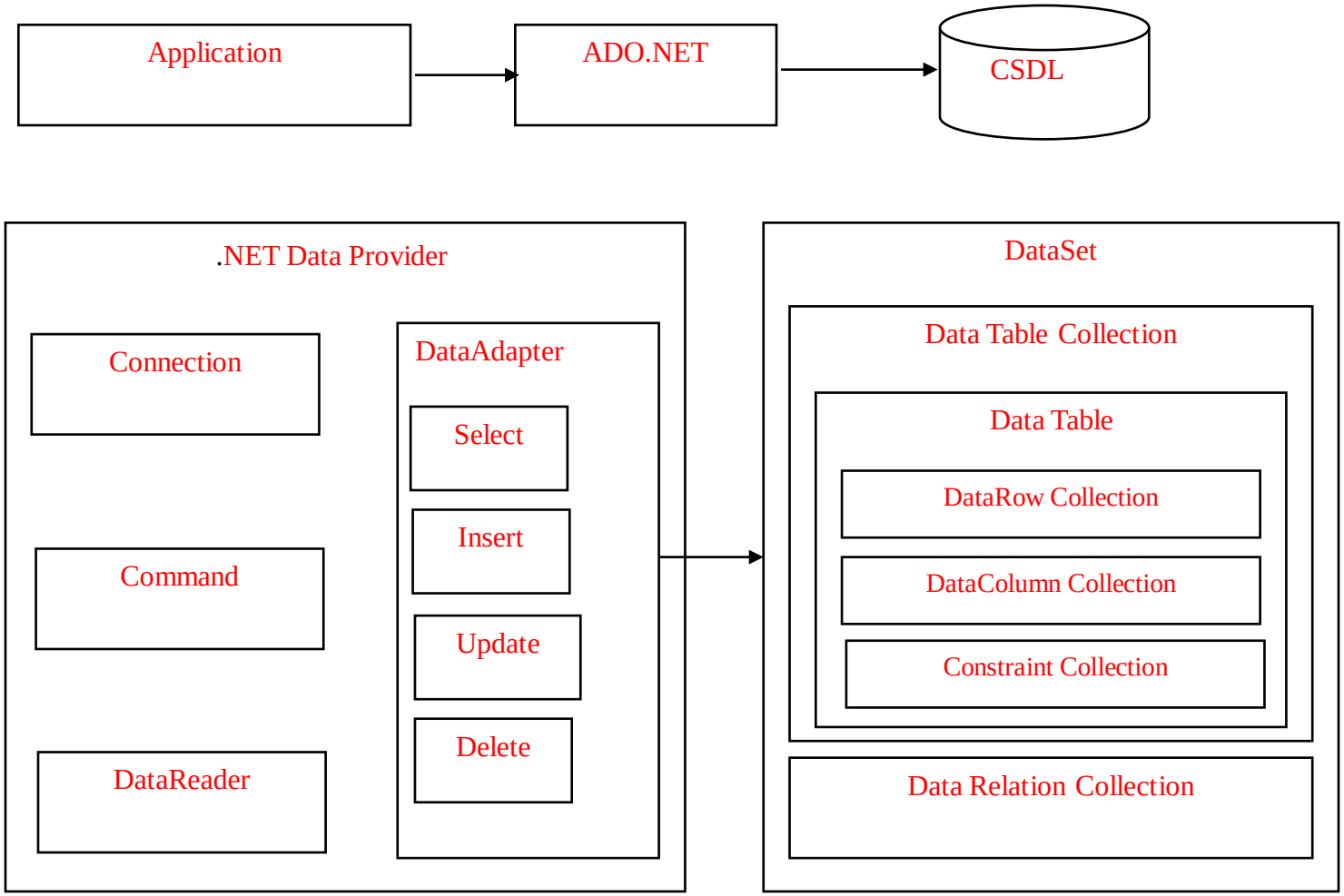
8.3.2. .NET và mô hình ADO.NET

- .NET là bộ công cụ, ngôn ngữ lập trình do Microsoft phát triển, có một số tính năng như Java (mục tiêu là để cạnh tranh), có khả năng tích hợp một số ngôn ngữ, không như Java là ngôn ngữ mở, .NET là ngôn ngữ đóng chỉ do mình hãng Microsoft phát triển.
- .NET cũng như Java được dùng nhiều hiện nay.

Chương 8: Xây dựng phần mềm dùng HQT SQL Server

- Kiến trúc tổng quát:
- .NET kết nối với CSDL không cần thông qua cầu nối ODBC, Microsoft đã xây dựng một cầu nối và tập các hàm truy xuất dữ liệu ở dạng tổng quát ở dạng đối tượng (COM) hoạt động gắn với hệ điều hành cho phép truy xuất đến mọi nguồn dữ liệu.

Chương 8: Xây dựng phần mềm dùng HQT SQL Server



Chương 8: Xây dựng phần mềm dùng HQT SQL Server

- **Câu lệnh kết nối:**
- ***string constring = "server=TRINHCUONG;database=DaoTaoHongDuc;uid=sa;pwd=trinh cuong;"***
- ***SqlConnection m_Conn1 = null;***
- ***SqlCommand m_Cmd;***
- ***DataSet m_DataSet;***
- ***DataTable m_Lophoc;***
- ***m_Cmd = new SqlCommand();***
- ***m_DataSet = new DataSet();***
- ***m_Lophoc = new DataTable();***
- ***m_Conn1 = new SqlConnection(constring);***
- ***m_Conn1.Open();***
- ***m_Cmd.Connection= m_Conn1;***

Chương 8: Xây dựng phần mềm dùng HQT SQL Server

- *m_Cmd.Parameters.Clear();*
- *m_Cmd.CommandText= "LOPHOC_MOLAYCATHEONAM";*
- *m_Cmd.CommandType= System.Data.CommandType.StoredProcedure;*
- *m_Cmd.Parameters.Add(new SqlParameter("@namhoc",SqlDbType.SmallInt));*
- *m_Cmd.Parameters["@namhoc"].Value=”2006”;*
- *SqlDataAdapter da= new SqlDataAdapter(m_Cmd);*

- *m_DataSet.Clear();*
- *da.Fill(m_DataSet, "Lophoc");*
- *m_Lophoc = m_DataSet.Tables[0];*
- *……*
- *m_Conn1.Close();*

Chương 8: Xây dựng phần mềm dùng HQT SQL Server

- Trong đó TRINHCUONG là tên Server của HQTCSDL, DaoTaoHongDuc là tên CSDL, sa và trinhcuong lần lượt là username và password của HQTCSDL.
- Khi bạn muốn kết nối và làm việc với CSDL thì trước hết bạn tạo một chuỗi kết nối và một đối tượng Connection để tạo kết nối. Ở trên chuỗi kết nối chính là *constring*, đối tượng kết nối chính là *m_Conn1* .

Chương 8: Xây dựng phần mềm dùng HQT SQL Server

- Sau đó bạn tạo đối tượng Command thực hiện các câu lệnh SQL tương tác với CSDL, sử dụng đối tượng Command bạn có thể truyền tham số cho câu lệnh SQL nếu bạn sử dụng SP, hoặc bạn có thể sử dụng trực tiếp dòng text không sử dụng SP lúc đó ta thay dòng lệnh *m_Cmd.CommandType= System.Data.CommandType.StoredProcedure;* bằng dòng lệnh *m_Cmd.CommandType= System.Data.CommandType.Text;*

Chương 8: Xây dựng phần mềm dùng HQT SQL Server

- Nếu là câu lệnh SQL có trả về kết quả (select) thì bạn tạo tiếp một đối tượng DataAdapter, DataSet, DataTable như trên để chứa kết quả trả về. Nếu không thì đơn giản chỉ gọi dòng lệnh *m_Cmd.ExecuteNonQuery();* ngay sau khi truyền xong tham số cho SP.
- Cuối cùng khi thực hiện xong bạn đóng kết nối *m_Conn1.Close();*
- Ở ví dụ trên ta dùng SP tên là LOPHOC_MOLAYCATHEONAM và truyền vào một tham số cho SP tên là @namhoc có giá trị là 2006.

Chương 8: Xây dựng phần mềm dùng HQT SQL Server

- Chú ý: Đối với CSDL đặc biệt SQL Server, .NET có hẳn các đối tượng kết nối đặc thù là SqlConnection, SqlCommand, SqlDataAdapter như trong ví dụ trên, các đối tượng này được thiết kế riêng cho CSDL SQL Server nhằm tăng tốc độ và hiệu quả khi làm việc.

Đề kiểm tra:

Câu 1: Cơ chế bảo mật trong SQL Server có mấy mức?Hãy trình bày tổng quan từng mức?

Câu 2: Bạn hiểu thế nào là quyền phát biểu (Statement Permissions) và quyền về đối tượng (Object Permissions)?

Câu 3: Trong CSDL HongDuc, dùng câu lệnh hãy tạo vai trò tên là ĐHTin, sau đó dùng tiếp câu lệnh gán cho vai trò này hai quyền tạo bảng và tạo view.

Giả sử có hai người dùng trong CSDL HongDuc là TinK5 và TinK6

a. Dùng câu lệnh để gán hai người dùng này là thành viên của vai trò ĐHTin.

b. Dùng câu lệnh gở bỏ quyền tạo bảng của TinK5 , nhưng giữ nguyên quyền tạo bảng và tạo view của vai trò ĐHTin, và TinK6 vẫn là thành viên của vai trò ĐHTin.

c. Xóa vai trò ĐHTin.

Câu 4: Giả sử có hai người dùng trong CSDL HongDuc là nd1 và nd2, hãy dùng câu lệnh cấp cho người dùng nd1 quyền select và update trên bảng Canbo.

Giả sử bảng Canbo có các trường là (Hoten, Tuoi, Gioitinh, Quequan, Chucvu, Luong), dùng câu lệnh cấp cho nd2 quyền select trên các cột (Hoten, Tuoi, Gioitinh, Chucvu), và quyền Update trên các cột (Hoten, Tuoi, Gioitinh) và cho phép nd2 quyền được cấp phép cho các người dùng khác quyền Update này.

Gở bỏ quyền Update trên bảng Canbo của nd2 và tất cả những người đã được nd2 cấp cho quyền này.

Vô hiệu hóa quyền select và update trên bảng Canbo của nd1.