CƠ SỞ DỮ LIỆU

CƠ SỞ DỮ LIỆU

◊ ◊

PHẦN

II

NGÔN NGỮ SQL

♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦

- Giới thiệu
- Ngôn ngữ định nghĩa dữ liệu
- Ngôn ngữ quản trị dữ liệu
- Ngôn ngữ điều khiển dữ liệu
- Ngôn ngữ truy vấn dữ liệu
- Bài tập tổng hợp

GV. Phạm Thị Hoàng Nhung
Bộ môn Công nghệ phần mềm
Đại học Thủy lợi

1 Chương 1. GIỚI THIỆU

1.1 Lịch sử phát triển

SQL (Structured Query Language, đọc là "sequel") là tập lệnh truy xuất CSDL quan hệ. Ngôn ngữ SQL được IBM sử dụng đầu tiên trong hệ quản trị CSDL System R vào giữa những năm 70, hệ ngôn ngữ SQL đầu tiên (SEQUEL2) được IBM công bố vào tháng 11 năm 1976. Năm 1979, tập đoàn ORACLE giới thiệu thương phẩm đầu tiên của SQL, SQL cũng được cài đặt trong các hệ quản trị CSDL như DB2 của IBM và SQL/DS.

Ngày nay, SQL được sử dụng rộng rãi và được xem là ngôn ngữ chuẩn để truy cập CSDL quan hệ.

1.2 Chuẩn SQL

Năm 1989, viện tiêu chuẩn quốc gia Hoa kỳ (ANSI) công nhận SQL là ngôn ngữ chuẩn để truy cập CSDL quan hệ trong văn bản ANSI SQL89.

Năm 1989, tổ chức tiêu chuẩn quốc tế (ISO) công nhận SQL ngôn ngữ chuẩn để truy cập CSDL quan hệ trong văn bản ISO 9075-1989.

Tất cả các hệ quản trị CSDL lớn trên thế giới cho phép truy cập bằng SQL và hầu hết theo chuẩn ANSI.

1.3 Đặc điểm của SQL

- Ngôn ngữ gần với ngôn ngữ tự nhiên (tiếng Anh)
- SQL là ngôn ngữ phi cấu trúc, tức là trong các lệnh của SQL người sử dụng CHỈ CẦN đưa ra yêu cầu hệ thống CÁI GÌ chứ không cần chỉ ra phải làm THẾ NÀO.
- **Ví dụ:** Cho cấu trúc dữ liệu để quản lý học sinh như sau HOCSINH(MaHS, TenHS, ĐTB, Xeploai). Đưa ra TenHS, ĐTB của các học sinh có ĐTB>=8.0.

Pascal	**SQL**

Type Hocsinh=Record
MaHS: String[5]; TenHS: String[30];
ĐTB: Real; Xeploai: String[30];
End;
HS: Array[1..100] Of Hocsinh;
Begin
….
For i:=1 to n do
If HS[i].ĐTB>=8.0 then
Writeln(HS[i].TenHS, HS[i].DTB);
….
End.
	- Tạo bảng HOCSINH
	- Sử dụng lệnh SELECT
	Select TenHS, ĐTB
	From HOCSINH
	Where ĐTB>=5.0;

- Ngôn ngữ SQL được sử dụng rất rộng rãi trong các Hệ quản trị cơ sở dữ liệu.
- SQL được chia 2 loại: SQL (ngôn ngữ hỏi) và PL/SQL (ngôn ngữ lập trình)

1.4 Các loại lệnh của SQL

Chia làm các nhóm chính:

- Cho phép truy vấn cơ sở dữ liệu để đưa ra các thông tin cần thiết (**SELECT**).
- Các lệnh định nghĩa dữ liệu (**DDL_ Data Definition Language**): Tạo và thay đổi cấu trúc các đối tượng trong cơ sở dữ liệu (CREATE, ALTER…)
- Các lệnh thực hiện trên dữ liệu (**DML_ Data Manipulation Language**): Cho phép thêm, sửa, xóa dữ liệu (INSERT, UPDATE, DELETE…).
- Các lệnh điều khiển dữ liệu **(DCL_Data Control Language):** Cho phép gán hoặc huỷ các quyền truy cập dữ liệu (GRANT, REVOTE)

2 Chương 2. CÁC LỆNH ĐỊNH NGHĨA DỮ LIỆU (DDL)

2.1 Tạo một cơ sở dữ liệu

Cú pháp:

```
Create Database <Tên CSDL>
```

Ví dụ: Tạo một cơ sở dữ liệu có tên là QLTV _ Quản lý thư viện

```
Create Database QLTV;
```

2.2 Tạo một bảng

2.2.1 Cú pháp

```
CREATE TABLE <Ten bang> (Tên_thuộc_tính1
Kiểu_tt1  [NOT NULL],
Tên_thuộc_tính2  Kiểu_tt2 [NOT NULL],
…
Tên_thuộc_tínhn  Kiểu_ttn  [NOT NULL]
[, CONSTRAINT mệnh đề ])
```

Trong đó, mệnh đề CONSTRAINT cho phép ta khai báo các ràng buộc dữ liệu(chi tiết sẽ được trình bày ở phần sau).

Ví dụ:

Tạo bảng DOCGIA, có các thuộc tính:

```
CREATE TABLE DOCGIA(
    MaDG Text(10)  NOT NULL PRIMARY KEY,
    TenDG Text(30) NOT NULL,
    DiaChi  Text(50) NOT NULL,
    Tuoi NUMBER)
```

Bảng này sẽ được nhận một tên gọi và một cấu trúc (danh sách tên các thuộc tính và một vài đặc trưng). Khi mới được tạo, bảng chưa có dữ liệu, chỉ là một cấu trúc lôgic có thể tiếp nhận các dữ liệu.

2.2.2 Tên của bảng

Tên của bảng được xác định ngay sau lệnh CREATE TABLE.

Mỗi HQTCSDL có một quy tắc đặt tên riêng. Ví dụ :

- Tên bảng phải bắt đầu bằng một chữ cái, có dưới 30 kí tự (chữ cái, chữ số, và dấu ‘_’).

- Tên bảng phải khác tên gọi khác của bảng hay của khung nhìn và với tên gọi đã dành riêng của SQL.
- Không phân biệt hoa, thường.

2.2.3 Xác định các thuộc tính

Trong lệnh tạo bảng ta phải xác định cấu trúc của bảng. Cần phải xác định mỗi thuộc tính của một định nghĩa kết thúc bằng dấu ‘,’ và gồm:

- Tên thuộc tính
- Loại dữ liệu và độ dài
- Các ràng buộc có liên quan.

2.3 Các loại dữ liệu

Các loại dữ liệu được sử dụng còn tùy theo HQTCSDL.

2.3.1 Các loại dữ liệu được sử dụng trong MS Access

Kiểu dữ liệu	Miêu tả	Kích cỡ
Text	Sử dụng ký tự hoặc kết hợp giữa ký tự và số, như địa chỉ, hoặc những số không yêu cầu tính toán, như số điện thoại, mã nước, mã vùng…	Khả năng lưu trữ tối đa (FieldSize)là 255 ký tự.
Memo	Sử dụng khi bạn cần lưu trữ một lượng thông tin lớn, ví dụ như trường thông tin ghi chú về một cán bộ.	Khả năng lưu trữ tối đa là 65.536 ký tự.
Number	Number: Sử dụng cho những dữ liệu cần tính toán (loại trừ tính tiền, sử dụng Currency Type).	Khả năng lưu trữ có thể là 1, 2, 4, 8 tuỳ thuộc bào kiểu dữ liệu ta chọn (byte, integer, long integer, single, douple, decimal), riêng đối với kiểu dữ liệu ReplicationID (GUI) thi khả năng lưu trữ là 16 byte.
Date/Time	Lưu trữ thông tin về thời gian.	Sử dụng 8 byte để lưu trữ.
Currency	Sử dụng Currency cho các dữ liệu cần tính toán. Phần thập phân có thể có từ 1 đến 4 số.	Khả năng lưu trữ là 8 byte.
AutoNumber	Đây là kiểu số tự động tăng với bước tăng là 1. Ta không thể cập nhật lại	Sử dụng 4 byte để lưu trữ. Nếu chọn kiểu dữ liệu là ReplicationID

	được trường này.	thì khả năng lưu trữ có thể lên tới 16 byte.
Yes/No	Kiểu dữ liệu YES/NO chỉ chứa một trong 2 giá trị (Yes/No, True/False, On/ Off)Y	Sử dụng 1 bite để lưu trữ.
OLE Object	Đối tượng (như là một văn bản trong Microsoft Word, dữ liệu đồ hoạ, âm thanh, hoặc một kiểu dữ liệu nhị phân…)	Sử dụng 1 GB để lưu trữ (tuỳ thuộc vào dung lượng của đĩa).

Ngoài ra còn 2 loại dữ liệu khác như Hyperlink, Lookup Wizard.

Đối với kiểu dữ liệu Number, ta còn có thể lựa chọ chi tiết:

Kiểu dữ liệu	**Miêu tả**	**Độ chính xác thập phân**	**Kích cỡ**
Byte	Lưu trữ số từ 0 đến 255 (không có phân số)	Không	1 byte
Decimal	Lưu trữ tối đa 10^38-1	28	12bytes
Integer	Lưu trữ số từ -32,768 to 32,767 (không có phân số).	Không	2 bytes
Long Integer	Lưu trữ số từ -2,147,483,648 tới 2,147,483,647 (không có phân số).	None	4 bytes
Single	Lưu trữ số từ -3.402823E38 to -1.401298E-45 cho giá trị âm và từ 1.401298E-45 to 3.402823E38 giá trị dương.	7	4 bytes
Double	Lưu trữ số từ -1.79769313486231E308 tới -4.94065645841247E-324 cho giá trị âm và từ 4.94065645841247E–324 to 1.79769313486231E308 giá trị dương.	15	8 bytes

2.3.2 Các loại dữ liệu được sử dụng trong Oracle:

1. CHAR

Kiểu CHAR dùng để khai báo một chuỗi có chiều dài cố định, khi khai báo biến hoặc cột kiểu CHAR với chiều dài chỉ định thì tất cả các mục tin của biến hay cột này đều có cùng chiều dài được chỉ định. Các mục tin ngắn hơn ORACLE sẽ tự động thêm vào các khoảng trống cho đủ chiều dài. ORACLE không cho phép

gán mục tin dài hơn chiều dài chỉ định đối với kiểu CHAR. Chiều dài tối đa cho phép của kiểu CHAR là 255 byte

2. VARCHAR2

Kiểu VARCHAR2 dùng để khai báo chuỗi ký tự với chiều dài thay đổi. Khi khai báo một biến hoặc cột kiểu VARCHAR2 phải chỉ ra chiều dài tối đa, các mục tin chứa trong biến hay cột kiểu VARCHAR2 có chiều dài thực sự là chiều dài của mục tin. ORACLE không cho phép gán mục tin dài hơn chiều dài tối đa chỉ định đối với kiểu VARCHAR2. Chiều dài tối đa kiểu VARCHAR2 là 2000 byte

3. VARCHAR

Hiện tại ORACLE xem kiểu VARCHAR2 và VARCHAR là như nhau, tuy nhiên ORACLE khuyên nên dùng VARCHAR2. ORACLE dự định trong tương lai dùng kiểu VARCHAR để chứa các chuỗi với chiều dài biến đổi, nhưng trong phép so sánh sẽ được chỉ định theo nhiều ngữ nghĩa khác nhau.

4. NUMBER

Kiểu số của ORACLE dùng để chứa các mục tin dạng số dương, số âm, số với dấu chấm động.

NUMBER(p, s)

Trong đó:

p: số chữ số trước dấu chấm thập phân (precision), p từ 1 đến 38 chữ số

s: số các chữ số tính từ dấu chấm thập phân về bên phải (scale), s từ -84 đến 127

NUMBER(p) số có dấu chấm thập phân cố định với precision bằng p và scale bằng 0

NUMBER số với dấu chấm động với precision bằng 38. Nhớ rằng scale không được áp dụng cho số với dấu chấm động.

Ví dụ sau cho thấy cách thức ORACLE lưu trữ dữ liệu kiểu số tùy theo cách định precision và scale khác nhau:

Dữ liệu thực	Kiểu	Lưu trữ
7456123.89	NUMBER	7456123.89
7456123.89	NUMBER(9)	7456123
7456123.89	NUMBER(9,2)	7456123.89
7456123.89	NUMBER(9,1)	7456123.8
7456123.89	NUMBER(6)	Không hợp lệ

7456123.8	NUMBER(15,1)	7456123.8
7456123.89	NUMBER(7,-2)	7456100
7456123.89	NUMBER(-7,2)	Không hợp lệ

5. FLOAT

Dùng để khai báo kiểu số dấu chấm động, với độ chính xác thập phân 38 hay độ chính xác nhị phân là 126.

FLOAT(b) Khai báo kiểu dấu chấm động với độ chính xác nhị phân là b, b từ 1 đến 126. Có thể chuyển từ độ chính xác nhị phân sang độ chính xác thập phân bằng cách nhân độ chính xác nhị phân với 0.30103.

6. LONG

Dùng để khai báo kiểu chuỗi ký tự với độ dài biến đổi, chiều dài tối đa của kiểu LONG là 2 gigabyte. Kiểu LONG thường được dùng để chứa các văn bản.

Có một số hạn chế khi dùng kiểu LONG:

- Một table không thể chứa nhiều hơn một cột kiểu LONG.
- Dữ liệu kiểu LONG không thể tham gia vào các ràng buộc toàn vẹn, ngoại trừ kiểm tra NULL và khác NULL.
- Không thể index một cột kiểu LONG.
- Không thể truyền tham số kiểu LONG cho hàm hoặc thủ tục.
- Các hàm không thể trả về dữ liệu kiểu LONG.
- Trong câu lệnh SQL có truy cập các cột kiểu LONG, thì việc cập nhật hoặc khóa các bảng chỉ cho phép trong cùng một CSDL

Ngoài ra, các cột kiểu LONG không được tham gia trong các thành phần sau của câu lệnh SQL:

- Các mệnh đề WHERE, GROUP BY, ORDER BY, CONNECT BY hoặc với tác tử DISTINCT trong câu lệnh SELECT.
- Các hàm sử dụng trong câu lệnh SQL như SUBSTR, INSTR.
- Trong danh sách lựa chọn của câu lệnh SELECT có sử dụng mệnh đề GROUP BY.
- Trong danh sách lựa chọn của câu hỏi con, câu hỏi có sử dụng các toán tử tập hợp.
- Trong danh sách lựa chọn của câu lệnh CREATE TABLE AS SELECT

7. DATE

Dùng để chứa dữ liệu ngày và thời gian. Mặc dù kiểu ngày và thời gian có thể được chứa trong kiểu CHAR và NUMBER.

Với giá trị kiểu DATE, những thông tin được lưu trữ gồm thế kỷ, năm, tháng, ngày, giờ, phút, giây. ORACLE không cho phép gán giá trị kiểu ngày trực tiếp, để gán giá trị kiểu ngày, bạn phải dùng TO_DATE để chuyển giá trị kiểu chuỗi ký tự hoặc kiểu số.

Nếu gán một giá trị kiểu ngày mà không chỉ thời gian thì thời gian mặc định là 12 giờ đêm, Nếu gán giá trị kiểu ngày mà không chỉ ra ngày, thì ngày mặc định là ngày đầu của tháng. Hàm SYSDATE cho biết ngày và thời gian hệ thống.

Tính toán đối với kiểu ngày:

Đối với dữ liệu kiểu ngày, bạn có thể thực hiện các phép toán cộng và trừ.

Ví dụ:

- SYSDATE+1 ngày hôm sau
- SYSDATE-7 cách đây một tuần
- SYSDATE+(10/1440) mười phút sau
- Ngày Julian: Là giá trị số cho biết số ngày kể từ ngày 1 tháng giêng năm 4712 trước công nguyên.**Ví dụ:**

```
SELECT TO_CHAR (TO_DATE('01-01-1992', 'MM-DD-YYYY'),
'J') JULIAN FROM DUAL
```

Cho kết quả:

```
JULIAN
-----------------------------------------------
2448623
```

8. RAW và LONG RAW

Kiểu RAW và LONG RAW dùng để chứa các chuỗi byte, các dữ liệu nhị phân như hình ảnh, âm thanh. Các dữ liệu kiểu RAW chỉ có thể gán hoặc truy cập chứ không được thực hiện các thao tác như đối với chuỗi ký tự.

Kiểu RAW giống như kiểu VARCHAR2 và kiểu LONG RAW giống kiểu LONG, chỉ khác nhau ở chỗ ORACLE tự động chuyển đổi các giá trị kiểu CHAR, VARCHAR2 và LONG giữa tập hợp ký tự của CSDL và tập ký tự của các ứng dụng.

9. ROWID

Mỗi mẫu tin trong CSDL có một địa chỉ có kiểu ROWID. ROWID gồm block.row.file, trong đó:

block : chuỗi hệ hexa cho biết block chứa row

row : chuỗi hệ hexa cho biết row trong block

file : chuỗi hệ hexa cho biết database file chứa block

Ví dụ:

0000000F.0000.0002

Row đầu tiên trong block 15 của data file thứ hai.

10. MLSLABEL

Kiểu MLSLABEL dùng để chứa label dạng nhị phân mà ORACLE dùng để đảm bảo hoạt động của bản thân hệ thống.

2.3.3 Các loại dữ liệu sử dụng trong SQL SERVER

Phần này sẽ được trình bày trong phần sau, khi học về SQL SERVER 2000.

2.4 Các loại ràng buộc trong bảng dữ liệu .

Các dạng constraint gồm:

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY (Referential)
- CHECK

2.4.1 NOT NULL- Không rỗng

- Khi có mệnh đề NOT NULL có trong định nghĩa của một cột thì ta bắt buộc thuộc tính này phải có giá trị. Nếu ta không chỉ thị gì trong định nghĩa của thuộc tính thì nó có thể có hoặc không có giá trị.

```
CREATE TABLE NHANVIEN(
 MaNV NUMBER(10) NOT NULL,
 TenNV CHAR(30))
```

2.4.2 UNIQUE-Duy nhất

- Chỉ ra ràng buộc duy nhất, các giá trị của cột chỉ trong mệnh đề UNIQUE trong các row của table phải có giá trị khác biệt. Giá trị null là cho phép nêu UNIQUE dựa trên một cột. **Ví dụ:**

```
CREATE TABLE NHANVIEN (
 MaNV NUMBER(10) NOT NULL,
 TenNV CHAR(30),
 DiachiNV CHAR(50)
 CONSTRAINT UNQ_Ten_Diachi UNIQUE(Ten,Diachi))
```

2.4.3 PRIMARY KEY- Khoá chính

- Chỉ ra ràng buộc duy nhất (giống UNIQUE), tuy nhiên khoá là dạng khoá UNIQUE cấp cao nhất. Một table chỉ có thể có một PRIMARY KEY. Các giá trị trong PRIMARY KEY phải NOT NULL.

Cú pháp:

```
[CONSTRAINT constraint_name ]
PRIMARY KEY [CLUSTERED|NONCLUSTERED]
[( colname [,colname2 [...,colname16]])]
```

Ví dụ:

```
CREATE TABLE NHANVIEN
(
MaNV char(10) NOT NULL primary key,
TenNV char(30),
DiachiNV char(50),
)
```

Hoặc ta có thể viết câu lệnh sau:

```
CREATE TABLE NHANVIEN
(
MaNV char(10) NOT NULL,
TenNV char(30),
DiachiNV char(50),
CONSTRAINT NV_P_K PRIMARY KEY (MaNV))
```

2.4.4 FOREIGN KEY-Khoá ngoại

Chỉ ra mối liên hệ ràng buộc tham chiếu giữa bảng này với bảng khác.

Từ khoá ON DELETE CASCADE được chỉ định trong dạng khoá này để chỉ khi dữ liệu cha bị xoá thì dữ liệu con cũng tự động bị xoá theo.

Cú pháp:

```
[CONSTRAINT constraint_name ]
[FOREIGN KEY (colname [,colname2 [...,colname16]])]
REFERENCES reference_table
[(ref_colname[,ref_colname2[...,ref_colname 16]])]
```

Ví dụ: Hai bảng DONVI và bảng NHANVIEN có mối quan hệ cha – con (1_N). Thuộc tính MaDV trong bảng NHANVIEN(bảng con) là khoá ngoại, được tham chiếu từ thuộc tính MaDV của bảng DONVI(bảng cha)

Ta tạo 2 bảng như sau:

```
CREATE TABLE DONVI
(
MaDV char(2) primary key,
TenDV char(20) not null
)
```

```
CREATE TABLE NHANVIEN
(
MaNV char(10) primary key,
TenNV char(30) not null,
Diachi char(50),
madv char(2)
CONSTRAINT   k_n_madv   FOREIGN   KEY(madv)   REFERENCES
DONVI(MaDV)
)
```

2.4.5 CHECK- Ràng buộc kiểm tra giá trị

Ràng buộc CHECK được sử dụng để yêu cầu các giá trị trong cột, hoặc khuôn dạng dữ liệu trong cột phải theo một quy tắc nào đó. Trên một cột có thể có nhiều ràng buộc này. Để khai báo một rang buộc CHECK cho một cột nào đó ta dùng cú pháp sau.

Cú pháp:

```
[CONSTRAINT constraint_name]
CHECK (expression)
```

Trong đó, expression là một biểu thức logic. Sau khi có ràng buộc này, giá trị nhập vào cho cột phải thoả mãn điều kiện mới được chấp nhận.

Ví dụ:

```
CREATE TABLE NHANVIEN
     (MaNV CHAR(10) NOT NULL PRIMARY KEY,
      TenNV CHAR(30),
      Luong NUMBER(10,2)
CONSTRAINT CK_SAL CHECK(SAL>500))
```

2.4.6 DEFAULT-Mặc định

Ràng buộc DEFAULT được sử dụng để quy định giá trị mặc định cho một cột. Giá trị này sẽ tự động gán cho cột nếu người sử dụng không nhập vào khi bổ sung bản ghi.

Cú pháp:

```
[CONSTRAINT constraint_name]
DEFAULT {const_expression/.nonarguments_function/.NULL}
```

Ví dụ:

```
CREATE TABLE NHANVIEN
(
MaNV char(10) primary key,
TenNV char(30) not null,
Gioitinh char(3) DEFAULT 'Nam'
)
```

2.5 Sửa đổi cấu trúc

Có thể sửa đổi cấu trúc của bảng hiện đang tồn tại bằng lệnh ALTER. Chúng ta có thể thêm một thuộc tính (cột) mới, thay đổi cấu trúc của một thuộc tính (cột) đang có, bổ sung khoá, bổ sung ràng buộc.

Cú pháp tổng quát:

```
ALTER TABLE table_name
[ADD
{col_name  column_properties  [column_constraints]
[[,]table_constraint ] }
   [,{next_col_name|next_table_constraint}]...]
[DROP
[CONSTRAINT] constraint_name1
[, constraint_name2]...]
/ALTER
{col_name  column_properties  [column_constraints]
[[,]table_constraint ] }
  [,{next_col_name|next_table_constraint}]...]
```

Thêm một ràng buộc CHECK

```
ALTER TABLE DONVI
ADD CONSTRAINT check_madv
CHECK (MaDV LIKE '[0-9][0-9]')
```

Thêm một thuộc tính.

Cú pháp:

```
ALTER TABLE <Tên_bảng>
ADD COLUMN Tên_cột , Kiểu_cột[(size)] )
```

Ví dụ:

```
ALTER TABLE DONVI
```

```
ADD(GhiChu, VARCHAR(255))
```

Chú ý: Trong một số HQTCSDL ta cần phải thêm từ khoá COLUMN như sau:

Cú pháp:

```
ALTER TABLE <Tên_bảng>
ADD COLUMN Tên_cột , Kiểu_cột[(size)] )
```

Ví dụ:

```
ALTER TABLE NHANVIEN
ADD COLUMN GhiChu Text(50));
```

Thay đổi kiểu của một thuộc tính.

Cú pháp:

```
ALTER TABLE <Tên_bảng>
ALTER (Tên_cột, Kiểu_mới)
```

Ví dụ:

```
ALTER TABLE NHANVIEN
ALTER(HoTen, VARCHAR(40))
```

Chú ý: Trong một số HQTCSDL ta cần phải thêm từ khoá COLUMN như sau:

Cú pháp:

```
ALTER TABLE  <Tên_bảng>
ALTER COLUMN Tên_cột , Kiểu_cột_mới[(size)]
```

Ví dụ:

```
ALTER TABLE  NHANVIEN
ALTER COLUMN GhiChu Memo
```

Xóa một thuộc tính

Cú pháp:

```
ALTER TABLE <Tên_bảng>
DROP <Tên_thuộc_tính>
```

Ví dụ:

```
ALTER TABLE NHANVIEN
DROP GhiChu
```

Chú ý: Trong một số HQTCSDL ta cần phải thêm từ khoá COLUMN như sau:

Cú pháp:

```
ALTER TABLE  <Tên_bảng>
DROP COLUMN Tên_cột
```

Ví dụ:

```
ALTER TABLE  NHANVIEN
DROP COLUMN GhiChu
```

2.6 Xoá đối tượng

Cú pháp:

```
DROP <Object_name>
```

Ví dụ:

```
DROP TABLE SINHVIEN
```

3 Chương 3. CÁC LỆNH QUẢN TRỊ DỮ LIỆU

3.1 Thêm hàng (INSERT)

Cú pháp:

```
INSERT [INTO]<TableName> (Column1, Column2, …, Columnn)
VALUES (Values1, Values2,…., Valuesn)
```

Lệnh này được dùng để xen thêm một hoặc nhiều dòng (bản ghi) mới vào một bảng. Dạng đơn giản nhất của lệnh này là thêm mỗi lần 1 dòng. Nó đòi hỏi phải nên tên của bảng, tên các thuộc tính và giá trị cần gán cho chúng. Nếu không nêu tên các thuộc tính thì điều đó có nghĩa là tất cả các thuộc tính trong bảng đều cần được thêm giá trị theo thứ tự từ trái sang phải.

Ví dụ 1:

Giả sử ta đã có cấu trúc bảng NHANVIEN(MaNV, TenNV, Diachi, Tuoi)

- Thêm bản ghi mới có tất cả các trường cho bảng NHANVIEN. Vì tất cả các thuộc tính trong bảng đều được thêm giá trị nên ta không cần có danh sách các thuộc tính ngay sau tên bảng NHANVIEN.

```
INSERT INTO NHANVIEN
VALUES('DHTL05','Nguyễn Công Thành', 'KhoaCNTT',22 )
```

- Thêm bản ghi mới vào bảng, để tuổi không xác định:

```
INSERT INTO DOCGIA(MaDG,TenDG,DiaChi)
VALUES('DHTL06','Nguyễn Phương Lan', 'Khoa May' )
```

- Ngoài ra chúng ta còn có thêm dữ liệu cho bảng từ giá trị của bảng khác:

Cú pháp:

```
INSERT [INTO]<TableName> (Column1, Column2, …, Columnn)
SELECT Select_list FROM <Tables>
```

Ví dụ:

```
insert into NHANVIEN_tam (TenNV, Tuoi)
select TenNV, Tuoi from NHANVIEN where Tuoi > 20
```

3.2 Xóa hàng (DELETE)

Cú pháp:

```
DELETE FROM <Table_name> WHERE <Conditions>
```

Lệnh này gồm 1 mệnh đề DELETE FROM để chỉ ra tên gọi của bảng được xét, và một mệnh đề WHERE để chỉ ra các dòng cần phải xóa. Như vậy, ta có thể cùng lúc xóa được nhiều dòng nếu dòng đó thỏa mãn điều kiện. Muốn xóa mọi dòng của một bảng thì không cần đưa vào mệnh đề WHERE.

Ví dụ:

Xóa một bản ghi (dòng) có MaDG=’DHTL01’ trong bảng DOCGIA.

```
DELETE FROM DOCGIA WHERE MaDG=’DHTL01’
```

Xóa những độc giả có địa chỉ là: 41NC có trong bảng DOCGIA.

```
DELETE FROM DOCGIA WHERE Diachi=’41NC’
```

3.3 Sửa đổi giá trị của một hàng (UPDATE)

Cú pháp:

```
UPDATE <Table_name>
SET (Colunm_name= <new value>)
WHERE <Condition>
```

Ví dụ:

```
UPDATE DOCGIA
SET (Diachi= ‘Khoa Cong trinh’)
WHERE MaDG= ‘TD001’
```

4 Chương 4. NGÔN NGỮ ĐIỀU KHIỂN (DCL)

Ngôn ngữ điều khiển được sử dụng trong việc cấp phát hay huỷ bỏ quyền của người sử dụng.

4.1 Lệnh GRANT

Câu lệnh này dùng để cấp phát quyền cho người sử dụng trên đối tượng Cơ sở dữ liệu hoặc quyền thực thi các câu lệnh SQL SERVER. Cú pháp có 2 dạng như sau:

Dạng 1: Cấp quyền đối với câu lệnh SQL

```
GRANT ALL | statement [,...,statementN ]
TO account [,...,accountN]
```

Dạng 2: Cấp quyền đối với các đối tượng trong cơ sở dữ liệu

```
GRANT ALL | permission [,...,permissionN]
    ON table_name |view_name [(column1 [,...,columnN])]
    |ON stored_procedure
TO account [,...,accountN]
```

Trong đó:

- ALL: là từ khoá được sử dụng khi muốn cấp phát tất cả các quyền cho người sử dụng.
- Account: là tên tài khoản đăng nhận hệ thống
- Permission: là quyền cấp phát cho người sử dụng trên đối tượng cơ sở dữ liệu:
    - Các quyền có thể cấp phát trên một bảng hoặc một View: Select, Insert, Delete, Update.
    - Các quyền có thể cấp phát trên cột của bảng hoặc của View: Select, Update
    - Quyền có thể cấp phát với các thủ tục: EXCUTE(thực thi)
- Statement: Là câu lệnh được cấp phát cho người sử dụng Các câu lệnh có thể cấp phát là:
    - CREATE DATABASE
    - CREATE TABLE
    - CREATE VIEW
    - CREATE PROCEDURE

- CREATE RULE
- CREATE DEFAULT
- BACKUP DATABASE
- BACKUP LOG

Ví dụ 1: Câu lệnh sau sẽ 3 cấp quyền SELECT, UPDATE, INSERT trên các thuộc tính (TenNV,DiaChi, Tuoi) của bảng NHANVIEN cho 2 người sử dụng phnhung, htvan.

```
GRANT SELECT, UPDATE, INSERT
ON NHANVIEN(TenNV,DiaChi, Tuoi)
TO phnhung, htvan
```

Ví dụ 2:

Câu lệnh sau sẽ cấp quyền tạo bảng, tạo View và tạo thủ tục cho người dùng phnhung.

```
GRANT CREATE TABLE, CREATE VIEW, CREATE PROCEDURE
TO phnhung
```

4.2 Lệnh REVOKE

Lệnh REVOKE được dùng để huỷ bỏ quyền đã được cấp phát cho người sử dụng. Câu lệnh này cũng có 2 dạng tương tự như câu lệnh GRANT.

Dạng 1: Huỷ quyền thực hiện câu lệnh:

```
REVOKE ALL | statement [,...,statementN]
FROM account [,...,accountN]
```

Dạng 2: Huỷ quyền thực hiện các đối tượng:

```
REVOKE ALL | permission [,...,permissionN]}
ON table_name | view_name [(column [,...,columnN])]
| stored_procedure
FROM account [,...,accountN ]
```

5 Chương 5. TRUY VẤN DỮ LIỆU (SELECT)

- Mệnh đề SELECT cho phép chỉ ra các thuộc tính mà ta muốn tìm. Thứ tự các thuộc tính trong kết quả là thứ tự mà nó xuất hiện trong lệnh SELECT. Bằng cách đó cho phép ta thực hiện được phép chiếu của quan hệ.
- Như vậy, kết quả của câu lệnh SELECT là một bảng, bảng đó là kết quả của phép chiếu qua bảng xuất phát.
- SELECT có thể thực hiện trên 1 bảng hoặc trên nhiều bảng.
- SELECT có nhiều mệnh đề, mỗi mệnh đề đảm bảo một chức năng.

5.1 Cú pháp

```
SELECT [DISTINCT]|Columns_list|Expression_list|*
FROM <Tables_list>
WHERE <Conditions>
GROUP BY <Columns>
HAVING <Conditions_for_group>
ORDER BY [ACS| DESC]
```

Trong đó:

- Sau SELECT: Các thông tin cần đưa ra, đó chính là danh sách các thuộc tính
- Sau FROM: Danh sách các tên bảng, từ đó thông tin được lấy ra.
- Sau WHERE: Các biểu thức logic, chỉ ra thông tin được lấy ra từ hàm nào hoặc điều kiện nối giữa các bảng.
- Sau GROUP BY: Các cột mà trong đó được tính theo từng nhóm.
- Sau HAVING: Biểu thức logic chỉ ra thông tin được lấy ra từ nhóm nào.
- Sau ORDER BY: Chỉ ra các cột mà trong đó thông tin được sắp xếp theo thứ tự.
    - ASC: thông tin được sắp xếp theo chiều tăng dần (ASCendent)
    - DESC: thông tin được sắp xếp theo chiều giảm dần(DESCendent)

5.2 Ví dụ:

Xét bảng: **NHANVIEN**

NHANVIEN				
MaNV	**HoTen**	**CongViec**	**Luong**	**MaDV**
NV001	Phạm Thị Nhàn	Thư ký	500	0001
NV002	Hoàng Thanh Vân	Giáo viên	600	0001
NV003	Hoàng Thị Lan	Giáo viên	200	0002
NV004	Đỗ Trung Dũng	Thư ký	700	0003
...	...	....	...	...

5.3 Đưa ra các cột

5.3.1 Đưa tất cả các cột

Ví dụ: Đưa tất cả các thông tin về nhân viên

```
SELECT *
FROM NHANVIEN
```

Kết quả: Toàn bộ bảng trên.

5.3.2 Đưa một số các cột

Ví dụ: Đưa ra Hoten, Luong của các nhân viên

```
SELECT Hoten, Luong
FROM NHANVIEN
```

Kết quả:

sl_NV_some_col	
Hoten	**Luong**
Phạm Thị Nhàn	500
Hoàng Thanh Vân	600
Hoàng Thị Lan	200
Đỗ Trung Dũng	700

5.3.3 Tránh các giá trị trùng lặp (DISTINCT)

Ví dụ: Đưa ra các công việc khác nhau trong bảng NHANVIEN

```
SELECT DISTINCT Congviec
FROM NHANVIEN
```

Kết quả: - Nếu không có lệnh DISTINCT và có DISTINCT:

Congviec
Thư ký
Giáo viên
Giáo viên
Thư ký

CongViec
Giáo viên
Thư ký

5.3.4 Đưa ra các giá trị của các biểu thức

Ví dụ: Đưa ra Hoten, Luongnam (Lương *12) của tất cả các nhân viên

```
SELECT Hoten, Luong*12
FROM NHANVIEN
```

Kết quả:

sl_bieuthuc	
Hoten	**Expr1001**
Phạm Thị Nhàn	6000
Hoàng Thanh Vân	7200
Hoàng Thị Lan	2400
Đỗ Trung Dũng	8400

5.3.5 Sử dụng bí danh cột

```
SELECT Hoten, Luong*12 AS Luongnam
FROM NHANVIEN
```

Kết quả:

Hoten	LuongNam
Phạm Thị Nhàn	6000
Hoàng Thanh Vân	7200
Hoàng Thị Lan	2400
Đỗ Trung Dũng	8400

5.3.6 Sắp xếp thứ tự (ORDER BY)

Ví dụ: Đưa ra Hoten, Luong sắp xếp theo thứ tự tăng dần/ giảm dần của Luong.

```
SELECT Hoten, Luong
FROM NHANVIEN
ORDER BY Luong [ASC/ DESC]
```

Kết quả:

Hoten	Luong
Hoàng Thị Lan	200
Phạm Thị Nhàn	500
Hoàng Thanh Vân	600
Đỗ Trung Dũng	700

- Trong đó ASC(ascendent) là tăng dần, DESC(descendent) là giảm dần.
- Nếu để giá trị mặc định thì sẽ sắp xếp theo chiều tăng dần.

5.4 Đưa ra các hàng

Lệnh có dạng:

```
SELECT [DISTINCT]|Columns_list|Expression_list|*
FROM <Tables_list>
WHERE <Conditions>
```

Điều kiện sau mệnh để Where là một biểu thức lôgic, sử dụng các phép toán sau:

5.4.1 Sử dụng các phép so sánh

= : Toán tử bằng hay tương đương

!= : Toán tử khác hay không tương đương

> : Toán tử lớn hơn

< : Toán tử nhỏ hơn

>= : Toán tử lớn hơn hoặc bằng

<= : Toán tử nhỏ hơn hoặc bằng

Ví dụ: Đưa ra Hoten, Luong của các nhân viên có Luong>300

Hoten	Luong
Phạm Thị Nhàn	500
Hoàng Thanh Vân	600
Đỗ Trung Dũng	700

5.4.2 Sử dụng các phép logic: AND, OR, NOT

Ví dụ: Đưa ra Hoten, Luong của những nhân viên có công việc là Giáo viên và mức lương >300.

```
SELECT HoTen, Luong
FROM NHANVIEN
WHERE (Luong>300) AND (Congviec='Giáo viên')
```

Kết quả:

HoTen	Luong
Hoàng Thanh Vân	600

- Phân tích ví dụ sau:

```
SELECT HoTen, Luong
FROM NHANVIEN
WHERE (((Luong)>400) AND (Not(CongViec)=('Thư ký') )
      OR (Congviec=('Thư ký' ) ))
```

Kết quả:

HoTen	Luong
Phạm Thị Nhàn	500
Hoàng Thanh Vân	600
Đỗ Trung Dũng	700

5.4.3 Các toán tử của SQL

- **[NOT] BETWEEN x AND y:** [Không] nằm giữa giá trị X và Y
- **IN (danh sách):** thuộc bất kỳ giá trị nào trong danh sách
- **x [NOT] LIKE y**: Đúng nếu x [không] giống khung mẫu y. Các ký tự dùng trong khuôn mẫu:
  Dấu gạch dưới (_) : Chỉ một ký tự bất kỳ
  Dấu phần trăm (%) : Chỉ một nhóm ký tự bất kỳ
- **IS [NOT] NULL:** kiểm tra giá trị rỗng
- **EXISTS**: Trả về TRUE nếu có tồn tại.

Ví dụ:

- Phép BETWEEN … AND …

Ví dụ: Đưa ra những nhân viên có Lương trong khoảng 300 đến 600.

```
SELECT HoTen, Luong
FROM NHANVIEN
WHERE Luong BETWEEN 300 AND 600
```

Kết quả:

HoTen	Luong
Phạm Thị Nhàn	500
Hoàng Thanh Vân	600

- Phép IN (Một tập hợp);

Ví dụ: Đưa ra những nhân viên có lương hoặc 200, 300, 600.

```
SELECT HoTen, Luong
FROM NHANVIEN
WHERE Luong IN (200,500,600)
```

Kết quả:

HoTen	Luong
Phạm Thị Nhàn	500
Hoàng Thanh Vân	600
Hoàng Thị Lan	200

- Phép LIKE

- Ký tự thay thế '%' đại diện cho một nhóm các ký tự chưa biết (trong Access là: *).
- Ký tự thay thế '_' đại diện cho một ký tự chưa biết (trong Access là:?).
- **Ví dụ:** Đưa ra Hoten, Congviec của các nhân viên có Họ tên bắt đầu bằng chữ 'Hoàng'.

```
SELECT HoTen, Congviec
FROM NHANVIEN
WHERE Hoten LIKE 'Hoàng*'
```

Kết quả:

HoTen	Congviec
Hoàng Thanh Vân	Giáo viên
Hoàng Thị Lan	Giáo viên

Ví dụ:

```
SELECT HoTen, Congviec
FROM NHANVIEN
WHERE Hoten LIKE 'Hoàng Thanh Vân'
```

- Phép IS [NOT] NULL

Ví dụ:

```
SELECT * FROM NHANVIEN WHERE Diachi IS NULL
```

5.5 Sử dụng các hàm

Các HQTCSDL đưa ra các hàm khác nhau, vì thế khi làm việc với HQTCSDL nào chúng ta nên tìm hiểu các hàm và cách sử dụng chúng đối với HQTCSDL đó. Sau đây là một số các loại hàm thường dùng.

5.5.1 Hàm số học

Đầu vào và đầu ra là các giá trị kiểu số.

ROUND(n[,m]): Cho giá trị làm tròn của n (đến cấp m, mặc nhiên m=0)

TRUNC(n[,m]): Cho giá trị n lấy m chữ số tính từ chấm thập phân.

CEIL(n): Cho số nguyên nhỏ nhất lớn hơn hoặc bằng n.

FLOOR(n): Cho số nguyên lớn nhất bằng hoặc nhỏ hơn n.

POWER(m,n): Cho lũy thừa bậc n của m.

EXP(n): Cho giá trị của en

SQRT(n): Cho căn bậc 2 của n, n>=0

SIGN(n): Cho dấu của n.

n<0 có SIGN(n)= -1

n=0 có SIGN(n)= 0

n>0 có SIGN(n)= 1

ABS(n): Cho giá trị tuyệt đối

MOD(m,n): Cho phần dư của phép chia m cho n

5.5.2 Một số hàm kiểu số tham khảo khác:

LOG(m,n) cho logarit cơ số m của n

SIN(n) cosin của n (n tính bằng radian)

COS(n) cho cosin của n (n tính bằng radian)

TAN(n) cotang của n (n tính bằng radian)

Ví dụ hàm ROUND(n[,m]):

```
SELECT ROUND(4.923,1),
       ROUND(4.923),
       ROUND(4.923,-1),
       ROUND(4.923,2)
FROM DUMMY;

ROUND(4.923,1) ROUND(4.923) ROUND(4.923,-1) ROUND(4.923,2)
-------------- ------------ --------------- --------------
           4.9            5               0           4.92
```

Ví dụ hàm TRUNC(n[,m]):

```
SELECT TRUNC (4.923,1),
       TRUNC (4.923),
       TRUNC (4.923,-1),
       TRUNC (4.923,2)
FROM DUMMY;

TRUNC(4.923,1) TRUNC(4.923) TRUNC(4.923,-1) TRUNC(4.923,2)
-------------- ------------ --------------- --------------
           4.9            4               0           4.92
```

Ví dụ hàm CEIL(n)

```
SELECT CEIL (SAL), CEIL(99.9),CEIL(101.76), CEIL(-11.1)
FROM EMP
WHERE SAL BETWEEN 3000 AND 5000;

CEIL(SAL) CEIL(99.9) CEIL(101.76) CEIL(-11.1)
---------- ---------- ------------ -----------
      5000        100          102         -11
      3000        100          102         -11
      3000        100          102         -11
```

Ví dụ hàm FLOOR(n)

```
SELECT FLOOR (SAL), FLOOR (99.9), FLOOR (101.76), FLOOR (-11.1)
FROM EMP
WHERE SAL BETWEEN 3000 AND 5000;

FLOOR(SAL) FLOOR(99.9) FLOOR(101.76) FLOOR(-11.1)
---------- ----------- ------------- ------------
      5000          99           101          -12
      3000          99           101          -12
      3000          99           101          -12
```

Ví dụ hàm POWER(m,n)

```
SELECT SAL, POWER(SAL,2), POWER(SAL,3), POWER(50,5)
FROM EMP
WHERE DEPTNO =10;

       SAL POWER(SAL,2) POWER(SAL,3) POWER(50,5)
---------- ------------ ------------ -----------
      5000     25000000   1.2500E+11   312500000
```

```
      2450       6002500    1.4706E+10    312500000
      1300       1690000    2197000000    312500000
```

Ví dụ hàm EXP(n)

```
SELECT EXP(4) FROM DUMMY;

    EXP(4)
----------
  54.59815
```

Ví dụ hàm SQRT(n)

```
SELECT SAL, SQRT(SAL), SQRT(40), SQRT (COMM)
FROM EMP
WHERE DEPTNO =10;

       SAL  SQRT(SAL)    SQRT(40) SQRT(COMM)
---------- ---------- ---------- ----------
      5000 70.7106781 6.32455532
      2450 49.4974747 6.32455532
      1300 36.0555128 6.32455532
```

Ví dụ hàm SIGN(n)

```
SELECT SAL-NVL(COMM,0), SIGN(SAL-NVL(COMM,0)),
NVL(COMM,0)-SAL, SIGN(NVL(COMM,0)-SAL)
FROM EMP
WHERE DEPTNO =30

SAL-NVL(COMM,0)SIGN(SAL-NVL(COMM,0))NVL(COMM,0)-SAL
SIGN(NVL(COMM,0)-SAL)
--------------- ----------- --------------- ---------------------
           2850           1           -2850                    -1
           -150          -1             150                     1
           1300           1           -1300                    -1
           1500           1           -1500                    -1
            950           1            -950                    -1
            750           1            -750                    -1
```

5.5.3 Các hàm ký tự

- **CONCAT(char1, char2):** Cho kết hợp của 2 chuỗi ký tự, tương tự như sử dụng toán tử.
- **INITCAP(char):** Cho chuỗi với ký tự đầu các từ là ký tự hoa
- **LOWER(char):** Cho chuỗi ký tự viết thường (không viết hoa)
- **LPAD(char1, n [,char2]):** Chochuỗi ký tự có chiều dài bằng n. Nếu chuỗi char1 ngắn hơn n thì thêm vào bên trái chuỗi char2 cho đủ n ký tự. Nếu chuỗi char1 dài hơn n thì giữ lại n ký từ tính từ trái sang
- **LTRIM(char1, n [,char2]):** Bỏ các ký tự trống bên trái
- **NLS_INITCAP(char):** Cho chuỗi với ký tự đầu các từ là chữ hoa, các chữ còn lại là chữ thường

- **REPLACE(char,search_string[,replacement_string]):** Thay tất cả các chuỗi search_string có trong chuỗi char bằng chuỗi replacement_string.
- **RPAD(char1, n [,char2]):**Giống LPAD(char1, n [,char2]) nhưng căn phải.
- **RTRIM(char1, n [,char2]):** Bỏ các ký tự trống bên phải
- **SOUNDEX(char):** Cho chuỗi đồng âm của char.
- **SUBSTR(char, m [,n]):** Cho chuỗi con của chuỗi char lấy từ vị trí m về phải n ký tự, nếu không chỉ n thì lấy cho đến cuối chuỗi
- **TRANSLATE(char, from, to):** Cho chuỗi trong đó mỗi ký tự trong chuỗi from thay bằng ký tự tương ứng trong chuỗi to, những ký tự trong chuỗi from không có tương ứng trong chuỗi to sẽ bị loại bỏ.
- **UPPER(char):** Cho chuỗi chữ hoa của chuỗi char
- **ASCII(char):** Cho ký tự ASCII của byte đầu tiên của chuỗi char
- **INSTR(char1, char2 [,n[,m]]):** Tìm vị trí chuỗi char2 trong chuỗi char1 bắt đầu từ vị trí n, lần xuất hiện thứ m.
- **LENGTH(char):** Cho chiều dài của chuỗi char

Ví dụ hàm LOWER(char)

```
SELECT LOWER(DNAME), LOWER('SQL COURSE') FROM DEPT;

LOWER(DNAME)   LOWER('SQL
-------------- ----------
accounting     sql course
research       sql course
sales          sql course
operations     sql course
```

Ví dụ hàm UPPER(char)

```
SELECT ENAME FROM EMP WHERE ENAME = UPPER('Smith');

ENAME
----------
SMITH
```

Ví dụ hàm INITCAP(char)

```
SELECT INITCAP(DNAME), INITCAP(LOC) FROM DEPT;

INITCAP(DNAME) INITCAP(LOC)
-------------- -------------
Accounting     New York
Research       Dallas
Sales          Chicago
Operations     Boston
```

Ví dụ hàm CONCAT(char1, char2)

```
SELECT CONCAT(ENAME, JOB) JOB FROM EMP WHERE EMPNO = 7900;

JOB
------------------
JAMES     CLERK
```

Ví dụ hàm LPAD(char1, n [,char2])

```
SELECT LPAD(DNAME,20,'*'), LPAD(DNAME,20), LPAD(DEptno,20,' ')
FROM DEPT;

LPAD(DNAME,20,'*')   LPAD(DNAME,20)       LPAD(DEPTNO,20,'')
-------------------- -------------------- --------------------
******ACCOUNTING           ACCOUNTING                      10
******RESEARCH             RESEARCH                        20
******SALES                SALES                           30
******OPERATIONS           OPERATIONS                      40
```

Ví dụ hàm RPAD(char1, n [,char2])

```
SELECT RPAD(DNAME,20,'*'), RPAD(DNAME,20), RPAD(DEptno,20,' ')
FROM DEPT;

RPAD(DNAME,20,'*')   RPAD(DNAME,20)       RPAD(DEPTNO,20,'')
-------------------- -------------------- --------------------
ACCOUNTING    ****** ACCOUNTING           10
RESEARCH      ****** RESEARCH             20
SALES         ****** SALES                30
OPERATIONS    ****** OPERATIONS           40
```

Ví dụ hàm SUBSTR(char, m [,n])

```
SELECT SUBSTR('ORACLE',2,4), SUBSTR(DNAME,2), SUBSTR(DNAME,3,5)
FROM DEPT;

SUBS SUBSTR(DNAME, SUBST
---- ------------- -----
RACL CCOUNTING     COUNT
RACL ESEARCH       SEARC
RACL ALES          LES
RACL PERATIONS     ERATI
```

Ví dụ hàm INSTR(char1, char2 [,n[,m]])

```
SELECT DNAME, INSTR(DNAME, 'A'), INSTR(DNAME,'ES'),
INSTR(DNAME,'C',1,2)
FROM DEPT;

DNAME          INSTR(DNAME,'A') INSTR(DNAME,'ES')
INSTR(DNAME,'C',1,2)
-------------- ---------------- ----------------- -----------------
---
ACCOUNTING                    1                 0                 3
RESEARCH                      5                 2                 0
SALES                         2                 4                 0
OPERATIONS                    5                 0                 0
```

Ví dụ hàm LTRIM(char1, n [,char2])

```
SELECT DNAME, LTRIM(DNAME,'A'), LTRIM(DNAME,'AS'),
LTRIM(DNAME,'ASOP')
FROM DEPT;

DNAME          LTRIM(DNAME,'A LTRIM(DNAME,'A LTRIM(DNAME,'A
-------------- -------------- -------------- --------------
ACCOUNTING     CCOUNTING      CCOUNTING      CCOUNTING
RESEARCH       RESEARCH       RESEARCH       RESEARCH
SALES          SALES          LES            LES
OPERATIONS     OPERATIONS     OPERATIONS     ERATIONS
```

Ví dụ hàm RTRIM(char1, n [,char2])

```
SELECT DNAME, RTRIM(DNAME,'A'), RTRIM(DNAME,'AS'),
RTRIM(DNAME,'ASOP')
FROM DEPT;

DNAME          RTRIM(DNAME,'A RTRIM(DNAME,'A RTRIM(DNAME,'A
-------------- -------------- -------------- --------------
ACCOUNTING     ACCOUNTING     ACCOUNTING     ACCOUNTING
RESEARCH       RESEARCH       RESEARCH       RESEARCH
SALES          SALES          SALES          SALES
OPERATIONS     OPERATIONS     OPERATIONS     OPERATIONS
```

Ví dụ hàm SOUNDEX(char)

```
SELECT ENAME, SOUNDEX(ENAME)
FROM EMP
WHERE SOUNDEX(ENAME)= SOUNDEX('FRED');

ENAME      SOUN
---------- ----
FORD       F630
```

Ví dụ hàm LENGTH(char)

```
SELECT LENGTH('SQL COURSE'), LENGTH(DEPTNO), LENGTH(DNAME) FROM
DEPT;

LENGTH('SQLCOURSE') LENGTH(DEPTNO) LENGTH(DNAME)
------------------- -------------- -------------
                 10              2            14
                 10              2            14
                 10              2            14
                 10              2            14
```

Ví dụ hàm TRANSLATE(char, from, to)

```
SELECT ENAME, TRANSLATE(ENAME,'C','F'), JOB,
TRANSLATE(JOB,'AR','IT')
FROM EMP
WHERE DEPTNO = 10;

ENAME      TRANSLATE( JOB       TRANSLATE
---------- ---------- --------- ---------
KING       KING       PRESIDENT PTESIDENT
CLARK      FLARK      MANAGER   MINIGET
MILLER     MILLER     CLERK     CLETK
```

Ví dụ hàm REPLACE(char,search_string[,replacement_string])

```
SELECT JOB, REPLACE(JOB, 'SALESMAN', 'SALESPERSON'), ENAME,
REPLACE(ENAME, 'CO','PR')
FROM EMP
WHERE DEPTNO =30 OR DEPTNO =20;

JOB       REPLACE(JOB,'SALESMAN', ENAME      REPLACE(ENAME,'CO','
--------- ----------------------- ---------  ---------------
MANAGER   MANAGER                 BLAKE      BLAKE
MANAGER   MANAGER                 JONES      JONES
SALESMAN  SALESPERSON             MARTIN     MARTIN
SALESMAN  SALESPERSON             ALLEN      ALLEN
SALESMAN  SALESPERSON             TURNER     TURNER
CLERK     CLERK                   JAMES      JAMES
SALESMAN  SALESPERSON             WARD       WARD
ANALYST   ANALYST                 FORD       FORD
CLERK     CLERK                   SMITH      SMITH
ANALYST   ANALYST                 SCOTT      SPRTT
CLERK     CLERK                   ADAMS      ADAMS
```

Ví dụ các hàm lồng nhau:

```
SELECT DNAME, LENGHT(DNAME), LENGHT(TRANSLATE,DNAME, 'AS','A'))
FROM DEPT;

DNAME          LENGTH(DNAME) LENGTH(TRANSLATE(DNAME,'AS','A'))
-------------- ------------- ---------------------------------
ACCOUNTING                14                                14
RESEARCH                  14                                13
SALES                     14                                12
OPERATIONS                14                                13
```

5.5.4 Các hàm ngày

MONTH_BETWEEN(d1, d2): Cho biết só tháng giữa ngày d1 và d2.

ADD_MONTHS(d,n): Cho ngày d thêm n tháng.

NEXT_DAY(d, char): Cho ngày tiếp theo ngày d có thứ chỉ bởi char.

LAST_DAY(d): Cho ngày cuối cùng trong tháng chỉ bởi d.

Ví dụ hàm MONTH_BETWEEN(d1, d2)

```
SELECT MONTHS_BETWEEN( SYSDATE, HIREDATE),
MONTHS_BETWEEN('01-01-2000','05-10-2000')
FROM EMP
WHERE MONTHS_BETWEEN( SYSDATE,HIREDATE)>240;

MONTHS_BETWEEN(SYSDATE,HIREDATE) TWEEN('01-01-2000','05-10-2000')
-------------------------------- --------------------------------
                      241.271055                       -9.1290323
                      241.206539                       -9.1290323
                      243.367829                       -9.1290323
```

Ví dụ hàm ADD_MONTHS(d,n)

```
SELECT HIREDATE, ADD_MONTHS(HIRE,3), ADD_MONTHS(HIREDATE,-3)
FROM EMP
WHERE DEPTNO=20;

HIREDATE   ADD_MONTHS ADD_MONTHS
---------- ---------- ----------
02-04-1981 02-07-1981 02-01-1981
03-12-1981 03-03-1982 03-09-1981
17-12-1980 17-03-1981 17-09-1980
09-12-1982 09-03-1983 09-09-1982
12-01-1983 12-04-1983 12-10-1982
```

Ví dụ hàm NEXT_DAY(d, char)

```
SELECT HIREDATE, NEXT_DAY(HIREDATE,'FRIDAY'), NEXT_DAY(HIREDATE,6)
FROM EMP
WHERE DEPTNO = 10;

HIREDATE   NEXT_DAY(H NEXT_DAY(H
---------- ---------- ----------
17-11-1981 20-11-1981 20-11-1981
09-06-1981 12-06-1981 12-06-1981
23-01-1982 29-01-1982 29-01-1982
```

Ví dụ hàm LAST_DAY(d)

```
SELECT SYSDATE, LAST_DAY(SYSDATE), HIREDATE, LAST_DAY(HIREDATE),
LAST_DAY('15-01-2001')
FROM EMP
WHERE DEPTNO =20;

SYSDATE    LAST_DAY(S HIREDATE   LAST_DAY(H LAST_DAY('
---------- ---------- ---------- ---------- ----------
28-03-2001 31-03-2001 02-04-1981 30-04-1981 31-01-2001
28-03-2001 31-03-2001 03-12-1981 31-12-1981 31-01-2001
28-03-2001 31-03-2001 17-12-1980 31-12-1980 31-01-2001
28-03-2001 31-03-2001 09-12-1982 31-12-1982 31-01-2001
28-03-2001 31-03-2001 12-01-1983 31-01-1983 31-01-2001
```

Một số hàm khác có thể áp dụng cho kiểu ngày:

- **ROUND(date1):** Trả về ngày date 1 tại thời điểm giữa trưa 12:00 AM
- **ROUND(date1,'MONTH') :** Nếu date 1 nằm trong nửa tháng đầu trả về ngày đầu tiên của thàng, ngược lại sẽ trả về ngày đầu tiên của tháng sau.
- **ROUND(date1,'YEAR'):** Nếu date 1 nằm trong nửa năm đầu trả về ngày đầu tiên của thàng, ngược lại sẽ trả về ngày đầu tiên của năm sau.
- **TRUNC(date1, 'MONTH'):** Trả về ngày đầu tiên của tháng chứa date1.

- **TRUNC(date1, 'YEAR'):** Trả về ngày đầu tiên của năm chứa date1

5.5.5 Các hàm chuyển đổi kiểu

- **TO_CHAR(number|date, 'fmt'):** Chuyển kiểu số và ngày về kiểu ký tự.
- **TO_NUMBER(char):** Chuyển ký tự có nội dung số sang số
- **TO_DATE('chsr','fmt'):** Chuyển ký tự sang kiểu ngày với định dạng đặt trong fmt.
- **DECODE(EXPR, SEARCH1, RESULT1, SEARCH2, RESULT2, DEFAULT):** So sánh biểu thức expr với giá trị search nếu đúng trả về giá trị result nếu không trả về giá trị default.
- **NVL(COL|VALUE, VAL):** Chuyển giá trị COL|VALUE thành val nếu null.
- **Greatest(col|value1, col|value2):** Trả giá trị lớn nhất trong dãy giá trị.

Một số ví dụ:

```
SELECT To_char (sysdate, 'day, ddth month yyyy') from dummy;
SELECT EMPNO, ENAME, HIREDATE
FROM EMP
WHERE HIREDATE = TO_DATE ('June 4, 1984', 'month dd, yyyy');

INSERT INTO EMP (EMPNO, DEPTNO, HIREDATE
VALUES (777, 20, TO_DATE('19-08-2000', 'DD-MM-YYYY');

SELECT ENAME, JOB,
DECODE (JOB, 'CLERK','WWORKER','MANAGER','BOSS','UNDEFINED')
DECODẹD_JOB
FROM EMP;

SELECT GREATEST(1000,2000), GREATEST(SAL,COMM) FROM EMP
WHERE DEPTNO = 10;
```

Một số khuôn dạng ngày

SCC hoặc CC	thế kỷ; S chỉ ngày BC
YYYY hoặc SYYYY	năm; S chỉ ngày BC
YYY, YY, Y	Chỉ năm với 3,2,1 ký tự số
IYYY, IYY, IY, I	Chỉ năm theo chuẩn ISO
SYEAR, YEAR	Chỉ năm theo cách phát âm của người anh;
Q	Quý trong năm
MM	Giá trị tháng với 2 số (01-12)

MONTH	Tên đầy đủ của tháng theo tiếng anh, đọ dài 9
MON	Tháng với 3 ký tự viến tắt (JAN, FEB...)
WW, W	Tuần trong năm hoặc trong tháng
DDD, DD, D	Ngày trong năm, tháng hoặc tuần
DAY	Chỉ thứ trong tuần
DY	Chỉ thứ trong tuần với 3 ký tự viết tắt
J	Ngày Julian; bắt đầu từ ngày 31/12/4713 trước công nguyên
AM, PM	Chỉ định sáng, chiều
HH, HH12 HH24	Chỉ giờ trong ngày (1-12) hoặc (0-23)
MI	Phút (0-59)
SS	Giây (0-59)
SSSSS	Số giây đến nửa đêm (0-86399)
/ . , -	được tự động thêm khi đặt trong khuôn dạng
“char”	Đoạn ký tự đặt trong nháy đúp được tự động thêm khi đặt trong khuôn dạng
TH	Thêm phần thứ tự (1^{st}, 2^{nd}, 4^{th})
SP	Phát âm số (FOUR với DDSP)
SPTH, THSP	Phát âm và chuyển sang dạng thứ tự (First, second, ...)
RR	Ngày chuyển giao thiên niên kỷ với các năm <1999.

Một số khuôn dạng số

Ký tự	Mô tả	Ví dụ	Kết quả
9	Xác định hiển thị 1 số	999999	1234
0	Hiển thị cả số 0 ở đầu nếu độ dài khuôn dạng lớn hơn số hiện có	099999	001234
$	Thêm ký tự tiền tệ	$999999	$1234
L	Thêm ký tự tiền tệ bản địa	L999999	FF1234
.	Dấu thập phân	999999.99	1234.00
,	Dấu phân cách phần nghìn	999,999	1,234
MI	Dấu âm ở bên phải (với các giá trị âm)	999999MI	1234-
PR	Thêm ngoặc nhọn vào các giá trị âm	999999PR	<1234>
EEE	Chuyển sang hiển thị số E	99.9999RRRR	1.234E+03
V	Nhân với 10 n, n là số các số 9 đặt sau V	9999V99	123400
B	Hiển thị cả giá trị 0 nếu = 0.	B9999.99	1234.00

5.5.6 Hàm nhóm

- COUNT(): Đếm số lần xuất hiện của thuộc tính.
- SUM(colume): Tính tổng các giá trị của thuộc tính (thuộc loại số học)
- AVG(colume): Tính giá trị trung bình các giá trị của thuộc tính (thuộc loại số học)
- MAX(colume): Tìm giá trị cực đại của thuộc tính
- MIN(colume): Tìm giá trị cực tiểu của thuộc tính.

5.5.7 Sử dụng hàm nhóm

Đối số của các hàm nhóm là tên của thuộc tính mà hàm phải tính toán.

Ví dụ:

Đưa ra lương trung bình, lương lớn nhất, nhỏ nhất của tất cả các nhân viên trong bảng NHANVIEN.

```
SELECT Avg(Luong)  AS LuongTB,
      Max(Luong) AS LuongCN,
      Min(Luong) AS LuongTN,
      COUNT(MaNV) AS TongNV
FROM      NHANVIEN
```

Kết quả:

LuongTB	LuongCN	LuongTN	TongNV
500	700	200	4

5.5.8 Mệnh đề GROUP BY

Mệnh đề GROUP BY <các cột> cho phép đưa ra thông tin theo từng nhóm.

Ví dụ: Đưa ra Côngviệc, Lương trung bình của từng loại công việc.

```
SELECT CongViec, AVG(Luong) AS LuongTB
FROM NHANVIEN
GROUP BY CongViec
```

Kết quả:

CongViec	LuongTB
Giáo viên	400
Thư ký	600

Có thể thêm vào một mệnh đề WHERE để đưa vào một tiêu chuẩn chọn lựa các dòng. SQL thực hiện cùng một cách xử lý, đầu tiên là loại bỏ các dòng không đáp ứng tiêu chuẩn đã được xác định trong mệnh đề WHERE.

Ví dụ:

```
SELECT CongViec, AVG(Luong) AS LuongTB
FROM NHANVIEN
WHERE Luong>200
GROUP BY CongViec
```

Kết quả:

CongViec	LuongTB
Giáo viên	600
Thư ký	600

- Sử dụng mệnh đề GROUP BY để đưa ra các thông tin về các nhóm con trong các nhóm lớn.

Ví dụ: Đưa ra tổng lương của từng nhóm công việc trong từng đơn vị.

```
SELECT MaDV, CongViec, SUM(Luong) AS TongLuong
FROM NHANVIEN
GROUP BY MaDV, CongViec
```

Kết quả:

MaDV	CongViec	TongLuong
0001	Giáo viên	600
0001	Thư ký	500
0002	Giáo viên	200
0003	Thư ký	700

Chú ý: Nếu tên các cột ghi sau SELECT không phải là đối số của các hàm nhóm thì phải đưa vào mệnh đề GROUP BY.

Ví dụ:

TongLuong
1100
200
700

5.5.9 Mệnh đề HAVING

Muốn đưa ra các nhóm trên cơ sở thông tin nhóm thì điều kiện phải được viết trong mệnh đề HAVING (Không viết trong mệnh đề WHERE).

Ví dụ: Đưa ra những Congviec và trung bình lương của các công việc có trung bình lương >=300.

```
SELECT CongViec, Avg(Luong) AS TBLuong
FROM NHANVIEN
GROUP BY  CongViec
HAVING (Avg(Luong)>300)
```

Kết quả:

CongViec	TBLuong
Giáo viên	400
Thư ký	600

Ví dụ: Đưa ra những đơn vị và lương lớn nhất của các đơn vị có lương lớn nhất >=300.

```
SELECT MaDV,  Max(Luong) AS MaxLuong
FROM NHANVIEN
GROUP BY  MaDV
HAVING Max(Luong)>300
```

Kết quả:

MaDV	MaxLuong
0001	600
0003	700

Ghi chú: Mệnh để HAVING là mệnh đề tương đương với WHERE áp dụng cho các nhóm. Nói chung, mệnh đề này chỉ sử dụng nếu đã có chỉ thị một mệnh đề GROUP BY.

5.6 Lấy thông tin từ nhiều bảng

Muốn lấy thông tin từ nhiều bảng ta cần phải thực hiện nối các bảng, điều kiện nối phải được thiết đặt đầu tiên trong mệnh đề Where.

5.6.1 Nối bằng (Equi-Join)

Điều kiện nối là một đẳng thức.

Ví dụ: Đưa ra Hoten, Congviec, TenDV của tất cả nhân viên.

```
SELECT HoTen, CongViec, TenDV
FROM NHANVIEN, DONVI
WHERE NHANVIEN.MaDV= DONVI.MaDV
```

Kết quả:

HoTen	CongViec	TenDV
Phạm Thị Nhàn	Thư ký	KHTN

HoTen	CongViec	TenDV
Hoàng Thanh Vân	Giáo viên	KHTN
Hoàng Thị Lan	Giáo viên	DHTL
Đỗ Trung Dũng	Thư ký	DHQG

5.6.2 Bí danh bảng

Được viết ngay bên phải tên bảng trong mệnh đề FROM.

Ví dụ:

```
SELECT HoTen, CongViec, TenDV
FROM NHANVIEN NV, DONVI DV
WHERE NV.MaDV= DV.MaDV
```

5.6.3 Nối không bằng (Non Equi-Join)

Ví dụ: Đưa ra Hoten, Congviec, MaBac của tất cả nhân viên

```
SELECT HoTen, CongViec, MaBac
FROM NHANVIEN NV, BACLUONG BL
WHERE NV.Luong BETWEEN BL.BacThap AND BL.BacCao
```

Kết quả:

sl_non_equi

HoTen	CongViec	MaBac
Phạm Thị Nhàn	Thư ký	1
Hoàng Thanh Vân	Giáo viên	2
Đỗ Trung Dũng	Thư ký	3

Chú ý: Nếu ngoài các điều kiện nối còn có thêm các điều kiện khác thì điều kiện nối phải được viết trước.

Ví du: Đưa ra HoTen, Congviec, TenDV, Luong của những nhân viên có Luong>=500.

```
SELECT HoTen, CongViec, TenDV, Luong
FROM NHANVIEN AS NV, DONVI AS DV
WHERE (NV.MaDV=DV.MaDV) AND (Luong>=500);
```

Kết quả:

HoTen	CongViec	TenDV	Luong
Phạm Thị Nhàn	Thư ký	KHTN	500

HoTen	CongViec	TenDV	Luong
Hoàng Thanh Vân	Giáo viên	KHTN	600
Đỗ Trung Dũng	Thư ký	DHQG	700

5.6.4 Nối bảng với chính nó

Giả sử trong bảng NHANVIEN ta thêm 1 thuộc tính (cột) là MaPT (Mã phụ trách) để lưu mã của nhân viên phụ trách trực tiếp 1 nhân viên khác. Cụ thể như sau:

```
SELECT NV.MaNV, NV.Hoten, PT.MaNV, PT.Hoten
FROM NHANVIEN NV, NHANVIEN PT
WHERE (NV.MaNV=PT.MaPT) AND (NV.Luong>PT.Luong)
```

Kết quả:

NV.MaNV	NV.Hoten	PT.MaNV	PT.Hoten
NV002	Hoàng Thanh Vân	NV001	Phạm Thị Nhàn
NV002	Hoàng Thanh Vân	NV003	Hoàng Thị Lan

5.6.5 Thực hiện kết nối thông qua từ khóa Join

Ta có thể thực hiện lấy dữ liệu từ hai bảng thông qua từ khóa JOIN.

INNER JOIN (nối trong)

Cú pháp:

```
SELECT  field1, field2, field3
FROM  table1
INNER JOIN table2
ON  table1.keyfield=table2.foreign_keyfield;
```

Ví dụ: Giả sử có hai bảng:

KHACHHANG:

MaKH	TenKH
01	Hoàng Thanh Vân
02	Lê Thị Nhàn
03	Phan Thanh Hòa
04	Phạm Hồng Thanh

DONHANG:

MaSP	TenSP	MaKH
H102	Máy in	01
H106	Bàn	03
H301	Ghế	03

Yêu cầu: Đưa ra tên khách hàng và tên sản phẩm khách hàng đó mua.

```
SELECT KHACHHANG.TenKH, DONHANG.TenSP
FROM KHACHHANG
INNER JOIN DONHANG
ON KHACHHANG.MaKH=DONHANG.MaKH
```

Kết quả:

TenKH	TenSP
Hoàng Thanh Vân	Máy in
Phan Thanh Hòa	Bàn
Phan Thanh Hòa	Ghế

INNER JOIN trả về tất cả các dòng từ hai bảng thỏa mãn điều kiện. Nếu những dòng dữ liệu có bên table1 mà không có trong table2 thì sẽ không được hiển thị (khác với …)

LEFT JOIN

Cú pháp:

```
SELECT field1, field2, field3
FROM table1
LEFT JOIN table2
     ON table1.keyfield = table2.foreign_keyfield
```

Ví dụ:

```
SELECT KHACHHANG.TenKH, DONHANG.TenSP
FROM KHACHHANG
LEFT JOIN DONHANG
ON KHACHHANG.MaKH=DONHANG.MaKH
```

Kết quả:

TenKH	TenSP
Hoàng Thanh Vân	Máy in
Lê Thị Nhàn	
Phan Thanh Hòa	Bàn
Phan Thanh Hòa	Ghế
Phạm Hồng Thanh	

LEFT JOIN trả về tất cả các dòng có ở bảng thứ nhất, mặc dù ở bảng thứ hai không thỏa mãn phép toán. Nếu dữ liệu có ở bảng thứ nhất mà không có ở bảng thứ hai thì dữ liệu vẫn hiển thị.

RIGHT JOIN

Cú pháp

```
SELECT field1, field2, field3
FROM table1
RIGHT JOIN table2
ON            table1.keyfield           =
table2.foreign_keyfield
```

Ví dụ

```
SELECT KHACHHANG.TenKH, DONHANG.TenSP
FROM KHACHHANG
RIGHT JOIN DONHANG
ON KHACHHANG.MaKH=DONHANG.MaKH
```

Kết quả:

TenKH	TenSP
Hoàng Thanh Vân	Máy in
Phan Thanh Hòa	Bàn
Phan Thanh Hòa	Ghế

RIGHT JOIN trả về tất cả các dòng có ở bảng 2, mặc dù bảng 1 không thỏa mãn phép toán. Nếu dữ liệu có ở bảng 2 mà không có ở bảng 1 thì vẫn được hiển thị.

5.7 Thực hiện các phép toán trên tập hợp

Các phép toán trên tập hợp gồm: Hợp (UNION) hoặc UNION ALL, Giao (INTERSECT), Trừ (MINUS)

Điều kiện thực hiện các phép toán trên tập hợp: Các bảng tham gia vào phép toán phải có cùng số cột như nhau.

- **Phép UNION**.

Ví dụ: Đưa ra những công việc trong đơn vị 1 có MaDV là 0001 và đơn vị 2 có MaDV là 0002.

NHANVIEN					
MaNV	**HoTen**	**CongViec**	**Luong**	**MaDV**	**MaPT**
NV001	Phạm Thị Nhàn	Thư ký	500	0001	NV002
NV002	Hoàng Thanh Vân	Giáo viên	600	0001	NV003
NV003	Hoàng Thị Lan	Giáo viên	200	0002	NV002
NV004	Đỗ Trung Dũng	Thư ký	700	0003	NV002
NV005	Đỗ Văn Hải	Bảo vệ	100	0001	NV002
NV006	Nguyễn Nam Hải	Giám đốc	1000	0001	

```
SELECT CongViec
FROM NHANVIEN
WHERE MaDV='0001'
UNION
       SELECT CongViec
       FROM NHANVIEN
       WHERE MaDV='0002'
```

Kết quả:

CongViec
Bảo vệ
Giám đốc
Giáo viên
Thư ký

- Phép INTERSECT: Nếu thay UNION bằng INTERSECT thì kết quả sẽ đưa ra những công việc vừa có trong đơn vị 1, vừa có trong đơn vị 2.
- Phép MINUS: Nếu thay UNION bằng MINUS thì kết quả sẽ đưa ra những công việc chỉ có trong đơn vị 1, mà không có trong đơn vị 2.

5.8 Các câu hỏi lồng nhau

- Là các lệnh SELECT trong đó có chứa các lệnh SELECT khác.
- Các câu lệnh SELECT bên trong nằm sau mệnh đề WHERE hoặc HAVING của SELECT bên ngoài.
- **Cách thực hiện của câu lệnh SELECT lồng nhau:**
- Thực hiện lệnh SELECT bên trong.
- Sử dụng kết quả của lệnh SELECT bên trong để thực hiện lệnh SELECT bên ngoài.
- Số các lệnh SELECT lồng nhau được phép là 255.

5.8.1 Lệnh SELECT bên trong cho kết quả là 1 hàng

Xét bảng NHANVIEN trên.

Ví dụ: Đưa ra Hoten, TenDV, Congviec, Luong của những người có lương lớn hơn lương trung bình của toàn bộ nhân viên.

Đối với yêu cầu này ta cần làm những việc sau:

- Đưa ra trung bình lương của tất cả các nhân viên.
- Đưa ra những nhân viên thỏa mãn yêu cầu.

```
SELECT Hoten, TenDV, Congviec, Luong
FROM NHANVIEN AS NV, DONVI AS DV
WHERE (NV.MaDV= DV.MaDV)
                AND (Luong> ( SELECT AVG(Luong)
                              FROM NHANVIEN ))
```

Kết quả:

Hoten	TenDV	Congviec	Luong
Nguyễn Nam Hải	KHTN	Giám đốc	1000
Hoàng Thanh Vân	KHTN	Giáo viên	600
Đỗ Trung Dũng	DHQG	Thư ký	700

Ví dụ 2:

Đưa ra những nhân viên có lương lớn hơn người có lương lớn nhất trong đơn vị có tên là DHTL.

Công việc:

- Tìm MaDV có tên đơn vị là DHTL.
- Tìm mức lương lớn nhất trong đơn vị này.
- Tìm những nhân viên có lương thỏa mãn yêu cầu.

```
SELECT Hoten, TenDV, Congviec, Luong
FROM NHANVIEN AS NV, DONVI AS DV
WHERE (NV.MaDV= DV.MaDV)
            AND (Luong> ( SELECT MAX(Luong)
                          FROM NHANVIEN
                          WHERE MaDV =
                              SELECT MaDV
                              FROM DONVI
                              WHERE TenDV='DHTL')))
```

Kết quả:

Hoten	TenDV	Congviec	Luong
Nguyễn Nam Hải	KHTN	Giám đốc	1000
Phạm Thị Nhàn	KHTN	Thư ký	500
Hoàng Thanh Vân	KHTN	Giáo viên	600
Đỗ Trung Dũng	DHQG	Thư ký	700

5.8.2 Lệnh SELECT bên trong cho kết quả là nhiều hàng

Giả sử lệnh SELECT bên trong có dạng:

```
SELECT MaDV,MAX(Luong) AS LuongLN,MIN(Luong) AS LuongNN
FROM NHANVIEN
GROUP BY MaDV
```

Kết quả:

MaDV	LuongLN	LuongNN
0001	1000	100
0002	200	200
0003	700	700

Như vậy, kết quả của câu lệnh SELECT bên trong cho kết quả là một tập giá trị, thì ta phải sử dụng các phép toán so sánh với tập hợp, không sử dụng được các phép toán so sánh như (>, <, =, ….).

Toán tử SOME/ANY/ALL/NOT IN/EXITS

[NOT] IN : Không thuộc

ANY và SOME : So sánh một giá trị với mỗi giá trị trong một danh sách hay trong kết quả trả về của câu hỏi con, phải sau toán tử =

ALL : So sánh một giá trị với mọi giá trị trong danh sách hay trong kết quả trả về của câu hỏi con.

EXISTS : Trả về TRUE nếu có tồn tại.

- Phép toán IN:

Ta có biểu thức: <Giá trị> IN {Tập hợp} trả lại kết quả = TRUE nếu tập hợp các giá trị nằm trong tập hợp đứng sau IN.

Bảng NHANVIEN:

NHANVIEN					
MaNV	HoTen	CongViec	Luong	MaDV	MaPT
NV001	Phạm Thị Nhàn	Thư ký	500	0001	NV002
NV002	Hoàng Thanh Vân	Giáo viên	600	0001	NV003
NV003	Hoàng Thị Lan	Giáo viên	200	0002	NV002
NV004	Đỗ Trung Dũng	Thư ký	700	0003	NV002
NV005	Đỗ Văn Hải	Bảo vệ	100	0001	NV002
NV006	Nguyễn Nam Hải	Giám đốc	1000	0001	
NV007	Nguyễn Hoàng Lan	Giáo viên	500	0001	NV006
NV008	Nguyễn Thanh Ngọc	Giáo viên	700	0002	

Ví dụ 1: Đưa ra Hoten, MaDV, Luong của các nhân viên có Luong=Luong thấp nhất trong đơn vị của họ.

Công việc:

- Tính lương thấp nhất cho từng đơn vị
- So sánh (MaDV, Luong) của tất cả nhân viên với tập hợp đó.

```
SELECT Hoten, MaDV, Luong
FROM NHANVIEN
WHERE (MaDV, Luong) IN (Select MaDV, Min(Luong)
                        From NHANVIEN
                        Group by MaDV)
```

Đối với một vài HQTCSDL, tập hợp trong phép toán IN chỉ bao gồm 1 giá trị. Ví dụ không thể so sánh (MaDV, Luong), chỉ được phép so sánh MaDV hoặc Luong.

Ví dụ 2: Đưa ra Hoten, MaDV, Luong của các nhân viên có Luong=Luong thấp nhất trong một đơn vị nào đó.

```
SELECT NHANVIEN.MaNV, NHANVIEN.Hoten, NHANVIEN.Luong
FROM NHANVIEN
WHERE NHANVIEN.Luong IN (
      SELECT  Min(NHANVIEN.Luong) AS MinOfLuong
      FROM NHANVIEN
      GROUP BY NHANVIEN.MaDV)
```

Kết quả:

MaNV	Hoten	Luong
NV003	Hoàng Thị Lan	200
NV004	Đỗ Trung Dũng	700
NV005	Đỗ Văn Hải	100
NV008	Nguyễn Thanh Ngọc	700

Phép toán ALL

Kết hợp với các phép so sánh thông thường để so sánh một giá trị với 1 tập hợp.

Giá_trị > ALL{Tập hợp}: Biểu thức TRUE nếu giá trị so sánh > tất cả các giá trị trong tập hợp.

Ví dụ:

5> ALL(2,3,4): TRUE

5> ALL(2,4,6): FALSE

Phép toán ANY

Giá_trị> ANY{Tập hợp}: Biểu thức TRUE nếu giá trị so sánh > một giá trị nào đó trong tập hợp.

Ví dụ:

5> ANY(2,4,6): TRUE

Ví dụ: Đưa ra Hoten, Luong của các nhân viên có Luong lớn nhất của đơn vị có mã đơn vị là 0002.

```
SELECT Hoten, Luong
FROM NHANVIEN
WHERE Luong> ALL(
      Select Luong
      From NHANVIEN
      Where MaDV ='0002')
```

Kết quả select trong là:

Luong
200
700

Kết quả của cả câu lệnh:

Hoten	Luong
Nguyễn Nam Hải	1000

Nếu thay ALL = ANY thì kết quả:

Hoten	Luong
Phạm Thị Nhàn	500
Hoàng Thanh Vân	600
Đỗ Trung Dũng	700
Nguyễn Nam Hải	1000
Nguyễn Hoàng Lan	500
Nguyễn Thanh Ngọc	700

5.8.3 Mệnh đề HAVING trong SELECT lồng nhau.

Mệnh đề HAVING được sử dụng khi có điều kiện nhóm

Ví dụ: Đưa ra MaDV, AVG(Luong) của đơn vị có trung bình lương lớn hơn lương nhỏ nhất của đơn vị có mã đơn vị là 0003.

- Tính lương lớn nhất của đơn vị có mã đơn vị là 0003
- Đưa ra những đơn vị có TBLương > Lương nhỏ nhất vừa tính được

```
SELECT MaDV, Avg(Luong) AS AvgOfLuong
FROM NHANVIEN
GROUP BY NHANVIEN.MaDV
HAVING AVG(Luong)>
        Select Min(Luong)
        From NHANVIEN
        Where MaDV='0002')
```

5.8.4 Mệnh đề ORDER BY trong SELECT lồng nhau

Mỗi lệnh SELECT chỉ có 1 mệnh đề ORDER By duy nhất.

Một lệnh SELECT lồng nhau được coi là một lệnh SELECT. Vì vậy, nếu muốn sắp xếp dữ liệu thì mệnh đề ORDER BY phải là mệnh đề cuối cùng của lệnh SELECT ngoài cùng, các lệnh SELECT bên trong không có ORDER BY.

5.9 Các lệnh lồng nhau liên kết

Các lệnh liên kết cũng là các lệnh SELECT lồng nhau nhưng nó có cách thực hiện khác các lệnh lồng nhau thông thường.

Các bước thực hiện:

- Xét 1 hàng của bảng
- Sử dụng dữ liệu của hàng đó để thực hiện lệnh SELECT bên trong.
- Sử dụng kết quả của SELECT bên trong để thực hiện SELECT bên ngoài
- Lặp lại các bước trên cho đến khi hết các hàng được xét.

Ví dụ: Có bảng NHANVIEN

NHANVIEN

MaNV	HoTen	CongViec	Luong	MaDV	MaPT
NV001	Phạm Thị Nhàn	Thư ký	500	0001	NV002
NV002	Hoàng Thanh Vân	Giáo viên	600	0001	NV003
NV003	Hoàng Thị Lan	Giáo viên	200	0002	NV002
NV004	Đỗ Trung Dũng	Thư ký	700	0003	NV002
NV005	Đỗ Văn Hải	Bảo vệ	100	0001	NV002
NV006	Nguyễn Nam Hải	Giám đốc	1000	0001	
NV007	Nguyễn Hoàng Lan	Giáo viên	500	0001	NV006
NV008	Nguyễn Thanh Ngọc	Giáo viên	700	0002	

Đưa ra Hoten, MaDV, Luong của những nhân viên có Luong> LuongTB của đơn vị của họ.

```
SELECT NHANVIEN.HoTen, NHANVIEN.MaDV, NHANVIEN.Luong
FROM NHANVIEN
WHERE ((NHANVIEN.Luong)> (Select AVG(Luong)
                          From NHANVIEN NV1
                          Where NV1.MaDV= NHANVIEN.MaDV))
```

6 THỰC HÀNH TỔNG HỢP

- Phần thực hành được thực hiện trên hệ quản trị cơ sở dữ liệu Microsoft Aaccess.
- Tất cả các thao tác phải được thực hiện bằng ngôn ngữ SQL thông qua các Query.

6.1 Hướng dẫn thực hành

Mỗi yêu cầu được ghi vào 1 query. Các bước thao tác với Query như sau:

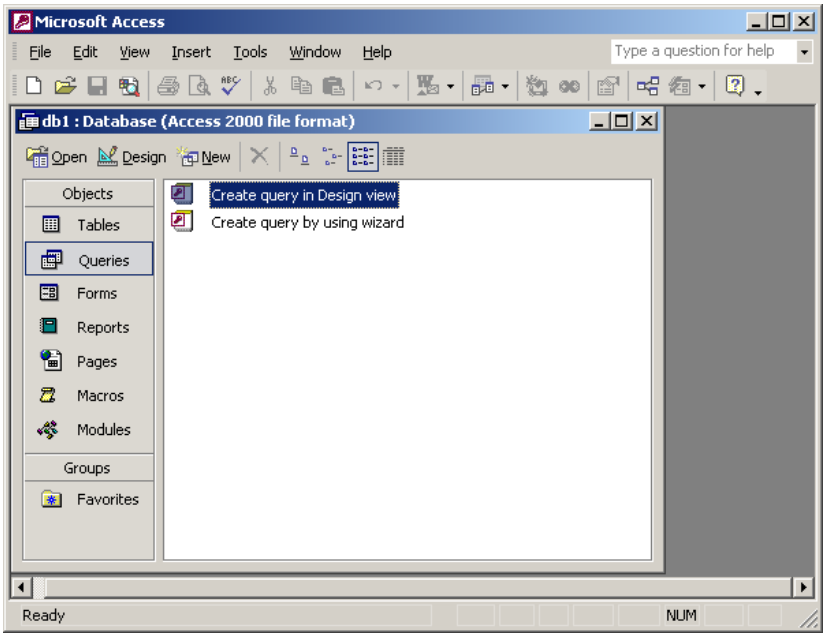
Bước 1: Mở Microsoft Access.

Bước 2: Tạo cơ sở dữ liệu (New/ Blank Database).

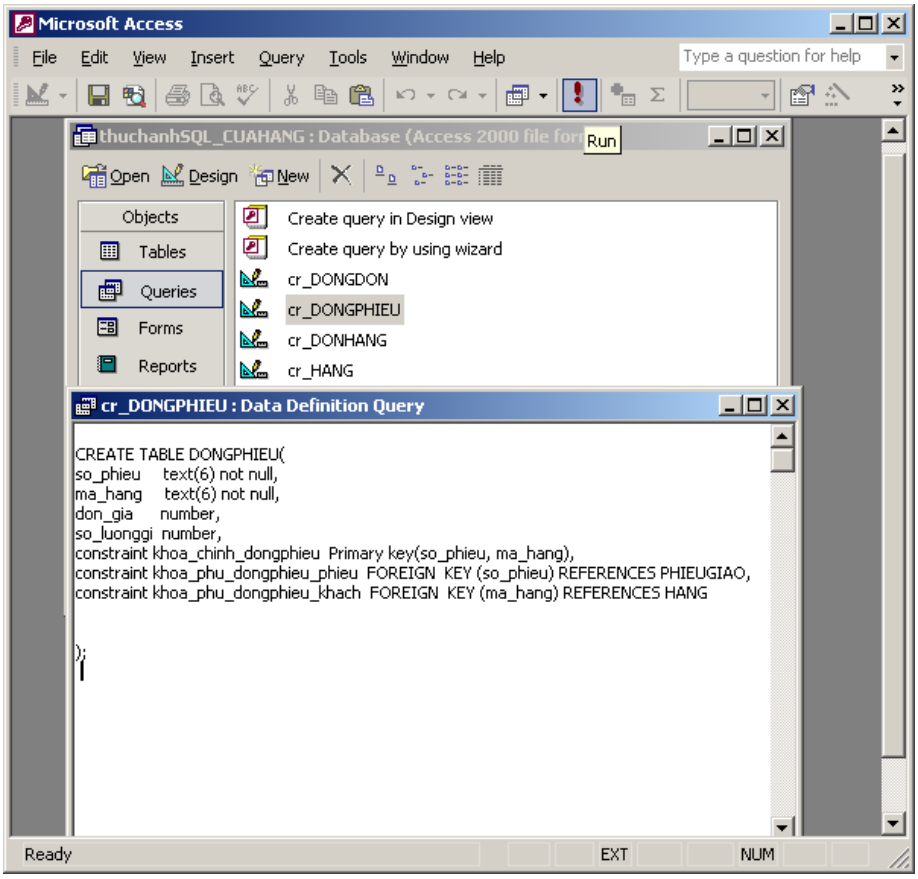
Nhập tên của cơ sở dữ liệu.

Bước 3: Tạo 1 truy vấn (Query) mới.

- Kích chuột vào Create query in Design view.



- Chọn SQL.
- Soạn thảo câu lệnh SQL.
- Ghi và đặt tên cho mỗi Query.
- Thực hiện câu lệnh bằng cách nhấn vào ! trên thanh công cụ.



- Chọn Save, nhập tên của Query.
- Chú ý:
  - Tất cả các yêu cầu của bài thực hành đều được thực hiện bằng lệnh của SQL thông qua Query.
  - Mỗi một yêu cầu được ghi lại trong một Query. Đặt tên Query theo tên của câu hỏi. Ví dụ: Cau1, Cau2,…

6.2 Bài số 1

1. Tạo một cơ sở dữ liệu có tên là Thuchanh.
2. Tạo một bảng có tên là DOCGIA, có các thuộc tính như sau:

```
CREATE TABLE DOCGIA(
        MaDG Text(10)  NOT NULL PRIMARY KEY,
        TenDG Text(30) NOT NULL,
        DiaChi  Text(50) NOT NULL,
        Tuoi NUMBER);
```

3. Thêm một thuộc tính mới có tên là Ghichu cho bảng DOCGIA.

```
        ALTER TABLE DOCGIA
```

```
ADD COLUMN GhiChu Text(50));
```

4. Thay đổi kiểu dữ liệu của thuộc tính Ghichu thành kiểu dữ liệu Memo.

```
ALTER TABLE  DOCGIA
ALTER COLUMN GhiChu Memo;
```

5. Xóa thuộc tính Ghichu trong bảng DOCGIA.

```
ALTER TABLE  DOCGIA
DROP COLUMN GhiChu;
```

6. Thực hiện các lệnh sau đây, mỗi lệnh này sẽ cho kết quả như thế nào? Nêu sự khác nhau giữa chúng.

Câu 1:

```
INSERT INTO DOCGIA
VALUES('DHTL05','Nguyễn Công Thành','Lớp 41NC',22 );
```

Câu 2:

```
INSERT INTO DOCGIA(MaDG, TenDG, DiaChi)
VALUES('DHTL06','Nguyễn Phương Lan','Lớp 41NC' );
```

7. Xóa một bản ghi có MaDG= 'DHTL01' trong bảng DOCGIA.

```
DELETE FROM DOCGIA WHERE MaDG='DHTL01';
```

8. Xóa những độc giả có địa chỉ là: 41NC trong bảng DOCGIA.

```
DELETE FROM DOCGIA WHERE Diachi='41NC';
```

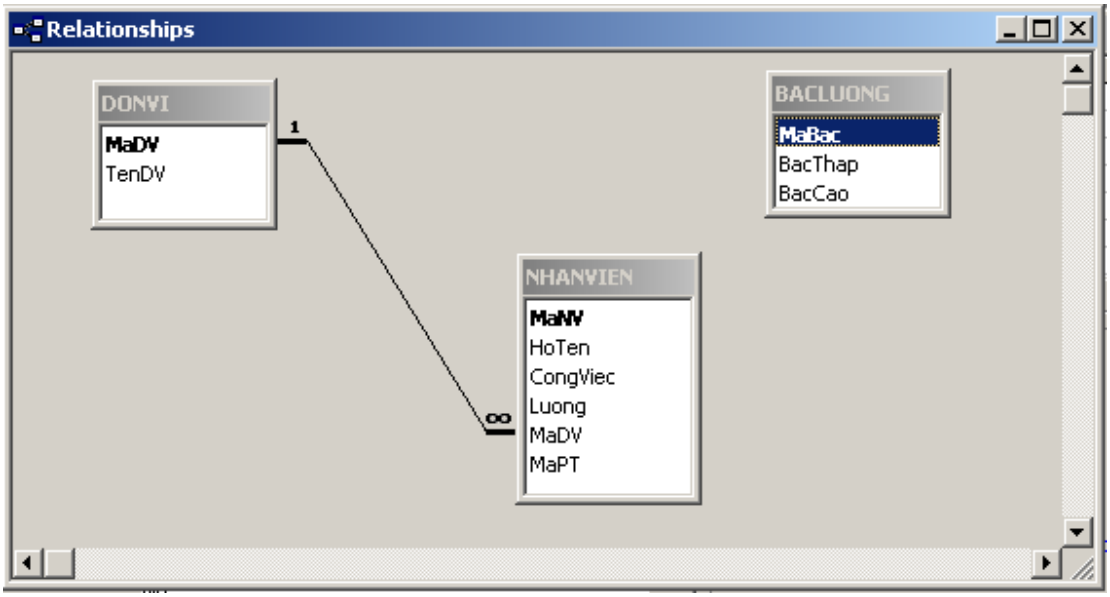
9. Sửa địa chỉ của độc giả có MaDG là TD001 thành địa chỉ mới là CVK3I.

```
UPDATE DOCGIA
SET (Diachi='CVK3I')
WHERE MaDG='TD001';
```

6.3 Bài số 2

1. Tạo một cơ sở dữ liệu (CSDL) dùng để quản lý nhân viên, lấy tên CSDL là QLNhanVien.

2. Cấu trúc của CSDL gồm các bảng với các quan hệ sau:



a. NHANVIEN

Tên thuộc tính	Kiểu dữ liệu	Kích cỡ	Khuôn dạng
MaNV	Ký tự	8	Chữ hoa +số
HoTen	Ký tự	30	
Congviec	Ký tự	50	
Luong	Số		
MaDV	Ký tự	4	
MaPT	Ký tự	8	Chữ hoa + số (Mã người phụ trách)

b. DONVI

Tên thuộc tính	Kiểu dữ liệu	Kích cỡ	Khuôn dạng
MaDV	Ký tự	4	Chữ hoa +số
TenDV	Ký tự	50	

c. BACLUONG

Tên thuộc tính	Kiểu dữ liệu	Kích cỡ	Khuôn dạng
MaBac	Ký tự	50	Chữ hoa +số
BacCao	Số		
BacThap	Số		

Chú ý: Mỗi câu lệnh tạo bảng được viết bằng 1 Query, đặt tên lần lượt là 21, 2b, 2c.

3. Thêm dữ liệu (bằng lệnh INERT) cho bảng NHANVIEN để có kết quả như sau:

NHANVIEN

MaNV	HoTen	CongViec	Luong	MaDV
NV001	Phạm Thị Nhàn	Thư ký	500	0001
NV002	Hoàng Thanh Vân	Giáo viên	600	0001
NV003	Hoàng Thị Lan	Giáo viên	200	0002
NV004	Đỗ Trung Dũng	Thư ký	700	0003

4. Thêm dữ liệu (bằng lệnh INERT) cho bảng DONVI để có kết quả như sau:

DONVI

MaDV	TenDV
0001	KHTN
0002	DHTL
0003	DHQG

5. Thêm dữ liệu (bằng lệnh INERT) cho bảng BACLUONG để có kết quả như sau:

BACLUONG

MaBac	BacThap	BacCao
1	400	500
2	501	600
3	601	800

6. Dùng câu lệnh truy vấn dữ liệu, đưa ra tất cả thông tin về nhân viên.
7. Đưa ra Hoten, Congviec, Luong của tất cả các nhân viên.

8. Đưa ra toàn bộ công việc của các nhân viên, các giá trị không trùng nhau.
9. Đưa ra Hoten, LuongQuy của tất cả các nhân viên, với LuongQuy= Luong*3.
10. Đưa ra Hoten, Luong sắp xếp theo thứ tự tăng dần/ giảm dần của Luong.
11. Đưa ra Hoten, Luong của các nhân viên có Luong>300.
12. Đưa ra Hoten, Luong của các nhân viên có Luong>300 và làm công việc là Giáo viên.
13. Đưa ra những nhân viên có lương hoặc 200, 300, 600.
14. Đưa ra những nhân viên có Lương trong khoảng 300 đến 600.
15. Đưa ra Hoten, Congviec của các nhân viên có Họ tên bắt đầu bằng chữ 'Hoàng'.
16. Đưa ra lương trung bình, lương lớn nhất, nhỏ nhất của tất cả các nhân viên trong bảng NHANVIEN.
17. Đưa ra Côngviệc, Lương trung bình của từng loại công việc.
18. Đưa ra Côngviệc, Lương trung bình của tất cả các nhân viên có Luong>200 theo từng loại công việc.
19. Đưa ra tổng lương của từng nhóm công việc trong từng đơn vị.
20. Đưa ra những Congviec và trung bình lương của các công việc có trung bình lương >=300
21. Đưa ra những đơn vị và lương lớn nhất của các đơn vị có lương lớn nhất >=300.
22. Đưa ra Hoten, Congviec, TenDV của tất cả nhân viên.
23. Câu lệnh sau cho kết quả như thế nào.

```
SELECT HoTen, CongViec, TenDV
FROM NHANVIEN NV, DONVI DV
WHERE NV.MaDV= DV.MaDV;
```

24. Đưa ra Hoten, Congviec, MaBac của tất cả nhân viên
25. Đưa ra HoTen, Congviec, TenDV, Luong của những nhân viên có Luong>=500.
26. Câu lệnh sau cho kết quả như thế nào.

```
SELECT NV.MaNV, NV.Hoten, PT.MaNV, PT.Hoten
FROM NHANVIEN NV, NHANVIEN PT
WHERE (NV.MaNV=PT.MaPT) AND (NV.Luong>PT.Luong);
```

27. Đưa ra những công việc trong đơn vị 1 có MaDV là 0001 và đơn vị 2 có MaDV là 0002.

28. Đưa ra Hoten, TenDV, Congviec, Luong của những người có lương lớn hơn lương trung bình của toàn bộ nhân viên.
29. Đưa ra những nhân viên có lương lớn hơn người có lương lớn nhất trong đơn vị có tên là DHTL.
30. Đưa ra Hoten, MaDV, Luong của các nhân viên có Luong=Luong thấp nhất trong đơn vị của họ.
31. Đưa ra Hoten, MaDV, Luong của các nhân viên có Luong=Luong thấp nhất trong một đơn vị nào đó.
32. Đưa ra Hoten, Luong của các nhân viên có Luong lớn nhất của đơn vị có mã đơn vị là 0002.
33. Đưa ra MaDV, AVG(Luong) của đơn vị có trung bình lương lớn hơn lương nhỏ nhất của đơn vị có mã đơn vị là 0003.

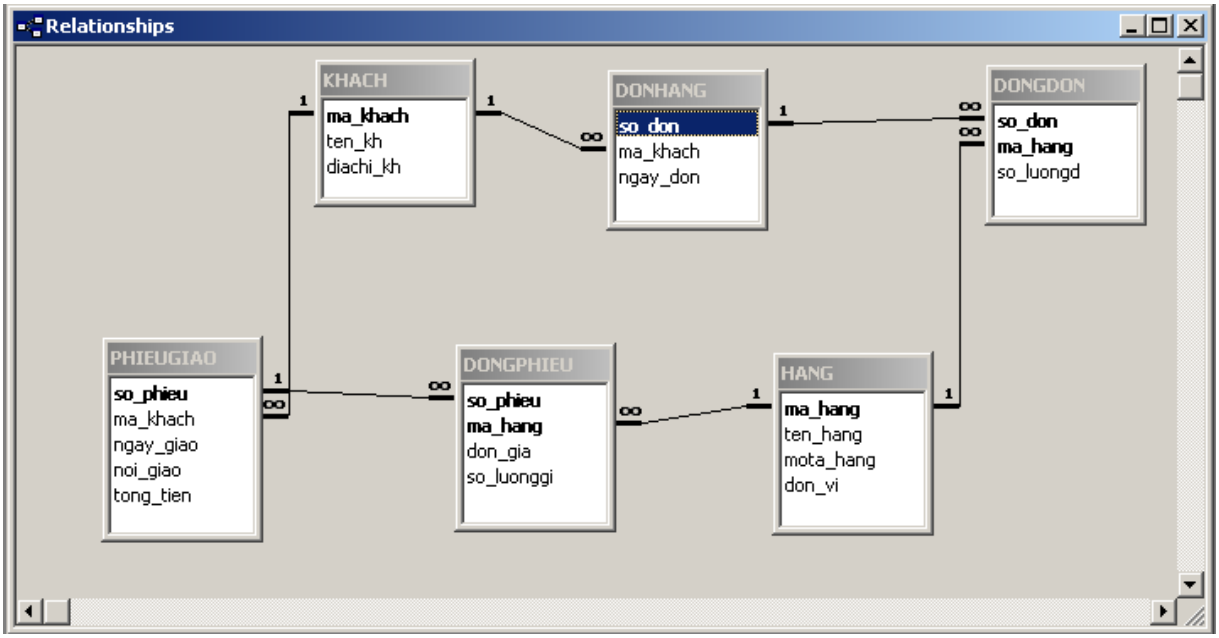
6.4 Bài số 3

Thực hiện các thao tác sau cho Cơ sở dữ liệu dùng để quản lý một cửa hàng kinh doanh.

I. ĐỊNH NGHĨA DỮ LIỆU

1. **Tạo cơ sở dữ liệu**

Tạo một cơ sở dữ liệu dùng để quản lý cửa hàng đặt tên là QLCH bao gồm các bảng có mối quan hệ như sau:



Cấu trúc của mỗi bảng như sau:

1. KHACH

Tên thuộc tính	Kiểu dữ liệu	Kích cỡ	Khuôn dạng
Ma_khach	Ký tự	6	Chữ hoa +số
Ten_kh	Ký tự	30	Chữ đầu viết hoa
Diachi_kh	Ký tự	30	Chữ đầu viết hoa

2. HANG

Tên thuộc tính	Kiểu dữ liệu	Kích cỡ	Khuôn dạng
Ma_hang	Ký tự	6	Chữ hoa+số
Ten_hang	Ký tự	15	Chữ đầu viết hoa
Mota_hang	Ký tự	30	Chữ đầu viết hoa

Don_vi	Ký tự	10	Chữ thường

3. DONHANG

Tên thuộc tính	**Kiểu dữ liệu**	**Kích cỡ**	**Khuôn dạng**
So_don	Ký tự	6	Chữ hoa +số
Ma_khach	Ký tự	15	Chữ hoa +số
Ngay_don	Ngày	8	Dd/mm/yy

4. DONGDON

Tên thuộc tính	**Kiểu dữ liệu**	**Kích cỡ**	**Khuôn dạng**
So_don	Ký tự	6	Chữ hoa +số
Ma_hang	Ký tự	6	Chữ hoa +số
So_luongd	Số	7	Số nguyên

5. PHIEUGIAO

Tên thuộc tính	**Kiểu dữ liệu**	**Kích cỡ**	**Khuôn dạng**
So_phieu	Ký tự	6	Chữ hoa +số
Ma_khach	Ký tự	6	Chữ hoa +số
Ngay_giao	Ngày	30	Dd/mm/yy
Noi_giao	Ký tự	30	
Tong_tien	Số	9	Số thực

6. DONGPHIEU

Tên thuộc tính	**Kiểu dữ liệu**	**Kích cỡ**	**Khuôn dạng**
So_phieu	Ký tự	6	Chữ hoa+ số
Ma_hang	Ký tự	6	Chữ hoa + số
Don_gia	Số	6	Số thực
So_luonggi	Số	7	Số thực

2. Sửa đổi cấu trúc:

1. Thêm một thuộc tính:

- Trong bảng **KHACH**, thêm một thuộc tính ***SoThich*** nhằm lưu vào bảng KHACH sở thích của khách hàng.

- Trong bảng **HANG**, thêm hai thuộc tính là ***NoiCungCap*** và ***NoiSanXuat*** nhằm lưu vào bảng HANG thông tin về nơi cung cấp hàng hóa đó và nơi sản xuất chúng.

2. Thay đổi kiểu của một thuộc tính.

- Trong bảng KHACH, thay đổi kiểu dữ liệu cho trường SoThich, từ kiểu Text thành kiểu Memo.
- Trong bang HANG, thay đổi kiểu dữ liệu cho trường NoiCungCap, từ kiểu Text thành kiểu Number.

3. Xóa một thuộc tính.

- Trong bảng HANG, hãy xóa 2 thuộc tính NoiCungCap và NoiSanXuat.
- Trong bảng KHACH, hãy xóa thuộc tính SoThich.

II. QUẢN TRỊ DỮ LIỆU

1. Thêm giá trị vào các dòng của bảng

- Thêm vào bảng KHACH các giá trị như sau:

KHACH		
Ma_khach	**Ten_kh**	**diachi_kh**
K001	Đào Minh Thư	Đại học Quốc Gia
K002	Nguyễn Liên Dung	BNC
K003	Phạm Hoàng Nhung	Đại học Thủy Lợi

- Thêm vào bảng DONHANG các giá trị như sau:

DONHANG		
so_don	**Ma_khach**	**ngay_don**
DH001	K001	3/22/2002
DH002	K001	7/13/2003
DH003	K002	12/24/2002

- Thêm vào tất cả các bảng, mỗi bảng một số bộ giá trị phù hợp.

Chú ý: Hãy thử trường hợp thêm ma_khach là K0010 vào bảng DONHANG thì kết quả thế nào? Tại sao?

2. Thêm giá trị cho một số thuộc tính

- Thêm giá trị vào bảng KHACH để được giá trị như sau:

KHACH		
Ma_khach	**Ten_kh**	**Diachi_kh**
K001	Đào Minh Thư	Đại học Quốc Gia
K002	Nguyễn Liên Dung	BNC
K003	Phạm Hoàng Nhung	Đại học Thủy Lợi
K004	Trịnh Hồng Cường	
K005	Nguyễn Văn Hải	Đại học Thủy Lợi

- Thêm giá trị vào bảng HANG để được giá trị như sau:

HANG			
ma_hang	**ten_hang**	**Mota_hang**	**don_vi**
H001	Hoa loa kèn	Hoa trắng, to	Bông
H002	Hoa hồng	Nhiều mau	Bông
H003	Hoa lan		Cành

3. Xóa hàng

- Xóa một hàng có ma_hang = 'H001' trong bảng HANG.
- Xóa tất cả các khách hàng có Diachi_kh = 'Đại học Thủy Lợi ' trong bảng KHACH.
- Xóa khách hàng có tên là Trịnh Hồng Cường.

4. Sửa đổi giá trị của một hàng.

- Sửa đổi địa chỉ của khách hàng thành địa chỉ mới là Đại học Quốc Gia với ma_khach là K002.
- Sửa đổi tất cả các tên hàng là "Hoa lan" thành "Hoa phong lan" và mô tả hàng là "Hàng nhập khẩu từ Đà Lạt"

III. CÁC LỆNH TRUY VẤN DỮ LIỆU

1. Xem toàn bộ nội dung của bảng KHACH
2. Xem toàn bộ nội dung của bảng HANG.
3. Xem toàn bộ nội dung của bảng DONHANG.
4. Đưa ra Tên và Địa chỉ của tất cả các Khách hàng trong bảng KHACH.
5. Đưa ra Tên hàng, Mô tả hàng và Đơn vị tính của tất cả các mặt hàng trong bảng HANG.
6. Đưa ra tất cả các đơn vị tính dùng để tính hàng hóa.
7. Đưa ra tất cả các tên hàng trong bảng HANG.
8. Đưa ra tất cả các địa chỉ của khách hàng.

9. Đưa ra so_phieu, ma_hang, don_gia, soluonggi và Thành tiền (don_gia*so_luong) của tất cả các hàng trong bảng PHIEU_GIAO.
10. Đưa ra so_phieu, ma_hang, don_gia, soluonggi và Thành tiền (don_gia*so_luong), sử dụng bí danh la thanh_tien của tất cả các hàng trong bảng PHIEU_GIAO .
11. Đưa ra tất cả giá trị của bảng HANG theo thứ tự giảm dần của ma_hang.
12. Đưa ra tất cả các giá trị của bảng PHIEUGIAO theo thứ tự tăng dần (giảm dần) của tổng tiền (tong_tien).
13. Đưa ra tất cả các khách hàng có địa chỉ là Đại học Quốc Gia trong bảng KHACH.
14. Đưa ra So_phieu, Ma_khach trong bảng PHIEUGIAO với điều kiện Tong_
    tien>=100.000, và sắp xếp theo điều kiện giảm dần của Tong_tien.
15. Đưa ra ten_hang, mota_hang của những hàng hóa có don_vi được tính theo bông trong bảng HANG.
16. Đưa ra danh sách các bản ghi bao gồm so_phieu, ma_khach, ngay_giao trong bảng PHIEUGIAO với điều kiện Noi_giao là Hà Nội và Tong_tien>50.000.
17. Đưa ra danh sách các bản ghi bao gồm Ma_khach, Ten_khach trong bảng KHACH với điều kiện địa chỉ của khách ở Đại học Quốc Gia hoặc Đại học Thủy Lợi.
18. Đưa ra những mặt hàng trong bảng HANG có đơn vị tính là Bông, Cành hoặc Bó.
19. Đưa ra danh sách những khách hàng (Ma_khach, Tong _tien) nằm trong bảng PHIEUGIAO có Noi_giao nằm trong số các địa điểm sau: Hà Nội, Hồ Chí Minh, Hải Phòng.
20. Đưa ra tất cả những thông tin trong bảng PHIEUGIAO với điều kiện tong_tien nằm trong khoảng từ 100.000 đến 500.000.
21. Đưa ra ma_kh, ten_kh trong bảng KHACH của những khách hàng có họ là Nguyễn.
22. Đếm số lần mua hàng của khách hàng có ma_kh là K001 trong bảng PHIEUGIAO. Thuộc tính mới này được đặt tên là SoLanMua.
23. Tính tổng tiền trung bình của mỗi PHIEUGIAO. Thuộc tính mới này được đặt tên là TrungBinhPG.
24. Tính tổng tiền trong bảng PHIEUGIAO của những khách hàng có ma_khach = ‘K002’.

25. Tính tổng số các khách hàng trong bảng KHACH có địa chỉ (diachi_KH) là Đại học Thủy Lợi.
26. Đưa ra ma_khach, ngay_giao, noi_giao trong bảng PHIEUGIAO có tong_tien thanh toán lớn nhất.
27. Đưa ra ma_khach, ngay_giao, noi_giao trong bảng PHIEUGIAO có tong_tien thanh toán nhỏ nhất.

TÀI LIỆU THAM KHẢO

1. Date, C.J., and Darwen, H.: *A Guide to the SQL Standard, 3rd ed*., Addison-Wesley.
2. *Tiện ích Book Online của SQL Server 2000.*
3. *Tiện ích Help của Microsoft Access 2000.*
4. *Tiện ích Help của Oracle 9i.*
5. Nguyễn Văn Vy, *SQL2*, NXB Thống kê.
6. Elmasri & Navathe: *Fundamentals of Database Systems*, International Edition.