

Khoa Hệ thống thông tin quản lý

CƠ SỞ DỮ LIỆU II

NGÔN NGỮ SQL

V.0.1

Biên soạn: Cao Thị Nhâm

VERSION LOG

Version	P.I.C		Date
0.1	NhamCT	First created	31/01/2015

MỤC LỤC

PHẦN I. NGÔN NGỮ SQL.....	5
1.1. Giới thiệu chung.....	5
1.1.1. Ngôn ngữ SQL	5
1.1.2. Các khái niệm cơ bản trong cơ sở dữ liệu	6
1.1.3. Các nhóm lệnh	6
1.1.4. Cơ sở dữ liệu sử dụng trong tài liệu	7
1.2. Các lệnh DDL	9
1.2.1. Các kiểu dữ liệu cơ bản	9
1.2.2. Lệnh tạo bảng (CREATE TABLE).....	10
1.2.3. Lệnh sửa bảng (ALTER TABLE).....	14
1.2.4. Các lệnh DDL khác	16
1.3. Các lệnh DML	16
1.3.1. Thêm dữ liệu	16
1.3.2. Sửa dữ liệu	17
1.3.3. Xóa dữ liệu.....	17
1.4. Lệnh DQL (truy vấn dữ liệu).....	17
1.4.1. Quy tắc viết lệnh	17
1.4.2. Cú pháp tổng quát	18
1.4.3. Các phép toán.....	18
1.4.3.7. Phép toán logic	23
1.4.4. Sắp xếp dữ liệu.....	23
1.4.5. Một số hàm cơ bản	24
1.4.6. Nối bảng.....	31
1.4.7. Truy vấn lồng (subquery)	34
1.6. View	38
1.7. Index.....	39
1.7.1. Giới thiệu	39
1.7.2. Tạo, xóa index.....	39

1.8. Sequence.....	40
1.8.1. Giới thiệu	40
1.8.2. Tạo và sử dụng sequence	40
1.8.3. Sửa và xóa sequence.....	41
1.9. Synonym.....	41

PHẦN I. NGÔN NGỮ SQL

1.1. Giới thiệu chung

1.1.1. Ngôn ngữ SQL

SQL (Structured Query Language - ngôn ngữ truy vấn mang tính cấu trúc) là một loại ngôn ngữ máy tính phổ biến để tạo, sửa, và lấy dữ liệu từ một hệ quản trị cơ sở dữ liệu quan hệ.

Giữa những năm 1970, một nhóm các nhà phát triển tại trung tâm nghiên cứu của IBM tại San Jose phát triển hệ thống cơ sở dữ liệu "Hệ thống R" dựa trên mô hình của Codd. Structured English Query Language, viết tắt là "SEQUEL" (tạm dịch là "Ngôn ngữ truy vấn tiếng Anh có cấu trúc"), được thiết kế để quản lý và truy lục dữ liệu được lưu trữ trong Hệ thống R. Sau này, tên viết tắt SEQUEL được rút gọn thành SQL để tránh việc tranh chấp nhãn hiệu (từ SEQUEL đã được một công ty máy bay của UK là Hawker-Siddeley đăng ký). Mặc dù SQL bị ảnh hưởng bởi công trình của tiến sĩ Codd nhưng nó không do tiến sĩ Codd thiết kế ra. Ngôn ngữ SEQUEL được thiết kế bởi Donald D. Chamberlin và Raymond F. Boyce tại IBM, và khái niệm của họ được phổ biến để tăng sự chú ý về SQL.

SQL được thừa nhận là tiêu chuẩn của ANSI (American National Standards Institute) vào năm 1986 và ISO (International Organization for Standardization) năm 1987. ANSI đã công bố cách phát âm chính thức của SQL là "ess kyoo ell", nhưng rất nhiều các chuyên gia cơ sở dữ liệu nói tiếng Anh vẫn gọi nó là sequel.

Tiêu chuẩn SQL đã trải qua một số phiên bản:

Năm	Tên	Tên khác	Chú giải
1986	SQL-86	SQL-87	Được công bố đầu tiên bởi ANSI. Được phê chuẩn bởi ISO năm 1987.
1989	SQL-89		Thay đổi nhỏ.
1992	SQL-92	SQL2	Thay đổi lớn.
1999	SQL:1999	SQL3	
2003	SQL:2003		

1.1.2. Các khái niệm cơ bản trong cơ sở dữ liệu

1.1.3. Các nhóm lệnh

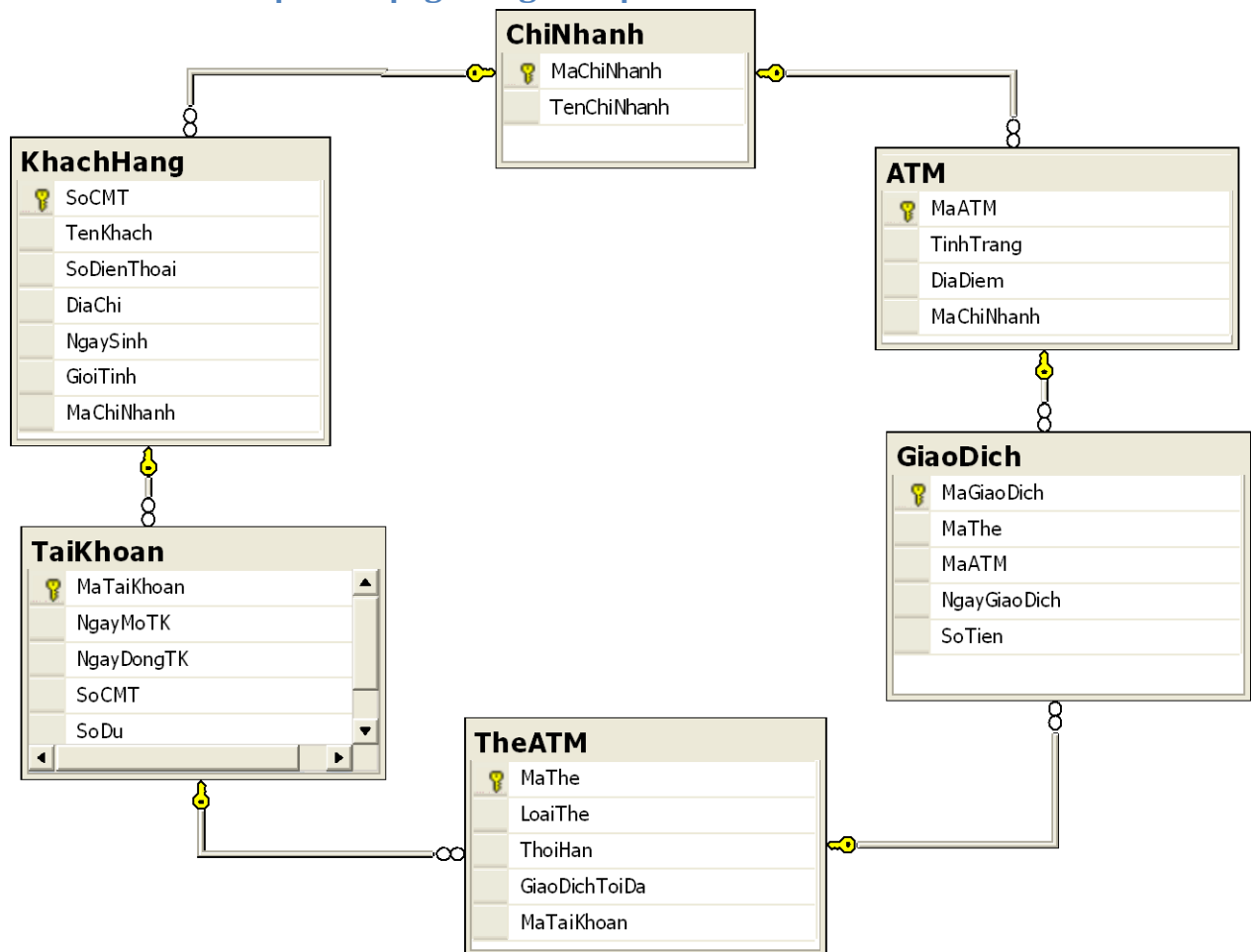
Nhóm lệnh	Lệnh	Chú giải
DQL (Data Query Language)	SELECT	Đây là lệnh phổ dụng nhất. Dùng để lấy dữ liệu trong cơ sở dữ liệu.
DML (Data Manipulation Language)	INSERT UPDATE DELETE	Là những lệnh dùng để thay đổi dữ liệu có trong các bảng của cơ sở dữ liệu
DDL (Data Definition Language)	CREATE ALTER DROP	Là những lệnh dùng để tạo ra, thay đổi hoặc xóa bỏ cấu trúc dữ liệu (bảng)
Transaction Control	COMMIT ROLLBACK SAVE POINT	Dùng để quản lý các giao dịch trong cơ sở dữ liệu.
DCL (Data Control Language)	GRANT REVOKE	Cấp quyền hoặc hủy quyền của người dùng trên đối tượng của cơ sở dữ liệu.

Trong nội dung tài liệu này chỉ xin đề cập tới DQL, DML và DDL. Ngôn ngữ DCL tham khảo thêm ở phần Quản lý người dùng.

Một số chú ý khi viết lệnh SQL:

- KHÔNG phân biệt chữ HOA, chữ thường trong cú pháp viết lệnh SQL và tên bảng.
- CÓ phân biệt HOA thường trong các phép so sánh chuỗi

1.1.4. Cơ sở dữ liệu sử dụng trong tài liệu



Thông tin chi tiết về các bảng như sau:

Bảng CHINHANH

Name	Null	Type
MACHINHANH	NOT NULL	CHAR(10)
TENCHINHANH		NVARCHAR2(30)

Bảng KHACHHANG

Name	Null	Type
SOCMT	NOT NULL	CHAR(9)
TENKHACH		NVARCHAR2(50)
SODIENTHOAI		VARCHAR2(15)
DIACHI		NVARCHAR2(100)
NGAYSINH		DATE
GIOITINH		NUMBER(1)
MACHINHANH		CHAR(10)

Bảng TAIKHOAN

Name	Null	Type
MATAIKHOAN	NOT NULL	CHAR(13)
NGAYMOTK		DATE
NGAYDONGTK		DATE
SOCMT		CHAR(9)
SODU		NUMBER(12)

Bảng ATM

Name	Null	Type
MAATM	NOT NULL	CHAR(10)
TINHTRANG		CHAR(1)
DIADIEM		NVARCHAR2(50)
MACHINHANH		CHAR(10)

Bảng THEATM

Name	Null	Type
MATHE	NOT NULL	CHAR(10)
LOAITHE		CHAR(1)
THOIHAN		DATE
GIAODICHTOIDA		NUMBER(10)
MATAIKHOAN		CHAR(13)

Bảng GIAODICH

Name	Null	Type
MAGIAODICH	NOT NULL	CHAR(10)
MATHE		CHAR(10)
MAATM		CHAR(10)
NGAYGIAODICH		DATE
SOTIEN		NUMBER(10)

1.2. Các lệnh DDL

1.2.1. Các kiểu dữ liệu cơ bản

Cấu trúc lưu trữ logic cơ bản của Oracle 11g là bảng. Mỗi bảng có các cột và các hàng. Mỗi cột có kiểu dữ liệu khác nhau. Oracle cung cấp một số kiểu dữ liệu có sẵn. Tài liệu này chỉ đề cập đến những kiểu dữ liệu thường dùng.

Loại	Kiểu dữ liệu
Chữ	CHAR, NCHAR, VARCHAR2, NVARCHAR2
Số	NUMBER, FLOAT, BINARY_FLOAT, BINARY_DOUBLE
Thời gian	DATE, TIMESTAMP, TIMESTAMP WITH TIMEZONE, TIMESTAMP WITH LOCAL TIMEZONE, INTERVAL YEAR TO MONTH, INTERVAL DAY TO SECOND
Các loại khác	LONG, CLOB, ROWID, ...

Sau đây là mô tả chi tiết cho một số kiểu dữ liệu thường dùng.

CHAR(n), NCHAR(n)

Đây là kiểu dữ liệu dạng chuỗi có độ dài cố định. Với n là độ dài của chuỗi hay độ dài của cột, $n \in [1, 2000]$. Khi người dùng nhập vào chuỗi có độ dài nhỏ hơn độ dài đã khai báo thì Oracle tự động thêm dấu cách (khoảng trống) phía đuôi chuỗi để đảm bảo độ dài luôn luôn cố định. Trong trường hợp độ dài chuỗi người dùng nhập vào lớn hơn giá trị n , Oracle sẽ báo lỗi.

CHAR(n) dùng để lưu trữ kí tự dưới dạng ASCII, NCHAR(n) dùng để lưu kí tự dưới dạng UNICODE.

Giá trị mặc định của CHAR và NCHAR là 1.

VARCHAR2(n), NVARCHAR2(n)

Đây là kiểu dữ liệu dạng chuỗi có độ dài không cố định. Với n là độ dài của chuỗi hay độ dài của cột, $n \in [1, 4000]$. VARCHAR2 và NVARCHAR2 không có độ dài mặc định, chính vì vậy, khi người dùng quên khai báo độ dài thì Oracle sẽ báo lỗi.

VARCHAR2 dùng để lưu trữ kí tự dưới dạng ASCII, NVARCHAR2 dùng để lưu kí tự dưới dạng UNICODE.

Đối với kiểu dữ liệu CHAR/NCHAR thì mọi dấu cách(khoảng trống) ở cuối chuỗi đều bỏ qua trong các phép toán so sánh, còn VARCHAR2/NVARCHAR2 thì ngược lại. Ví dụ:

Kiểu dữ liệu	Phép so sánh
CHAR/NCHAR	'CSDL' = 'CSDL '
VARCHAR2/NVARCHAR2	'CSDL' <> 'CSDL '

NUMBER(n, m)

Đây là dữ liệu kiểu số. Trong cách định nghĩa kiểu NUMBER, n là số các chữ số có nghĩa, m là số các chữ số sau dấu thập phân, $n \in [1, 38]$, $m \in [-84, 127]$. Nếu người dùng không xác định các giá trị n, m (ví dụ: chỉ khai báo `TOTAL NUMBER`) khi khai báo thì Oracle cung cấp độ rộng tối đa. Bảng dưới đây là một số ví dụ về cách khai báo.

Kiểu dữ liệu	Format	Giá trị input	Giá trị lưu trữ	Giải thích
NUMBER(6,2)	xxxx.xx	1234.9876	1234.99	Oracle làm tròn chỉ lưu 2 chữ số sau dấu thập phân
NUMBER(3,2)	x.xx	123.45	Lỗi	

DATE

Dùng để lưu dữ liệu liên quan đến ngày tháng và thời gian. Dữ liệu DATE bao gồm các thành phần:

- Thế kỷ
- Năm
- Tháng
- Ngày
- Giờ
- Phút
- Giây

1.2.2. Lệnh tạo bảng (CREATE TABLE)

1.2.2.1. Quy tắc đặt tên bảng

- Tên bảng dài 1-30 kí tự
- Tên bảng bắt đầu bằng chữ cái
- Tên bảng bao gồm chữ cái, số, _, #, \$ (hạn chế dùng \$)

- Tên bảng/cột không dùng những cụm từ đã có sẵn trong Oracle (ví dụ: NUBER)
- Tên cột phải duy nhất trong bảng
- Tên bảng phải duy nhất trong tablespace

1.2.2.2. Cú pháp tạo bảng

Cú pháp tổng quát để tạo bảng như sau:

```
CREATE TABLE tên_bảng(  
    Tên_Cột_1 Kiểu_Dữ_Liệu,  
    Tên_Cột_2 Kiểu_Dữ_liệu,  
    .....  
    primary key (Tên_Cột_X, Tên_Cột_Y, ...),  
    foreign key (Tên_Cột_Z) references Tên_Bảng_Liên_Kết,  
    .....  
);
```

Ví dụ tạo bảng TAIKHOAN như sau:

```
CREATE TABLE TaiKhoan(  
    MaTaiKhoan    char(13),  
    NgayMoTK      date,  
    NgayDongTK    date,  
    SoCMT         char(9),  
    primary key (MaTaiKhoan),  
    foreign key (SoCMT) references KhachHang  
);
```

1.2.1.3. Ràng buộc dữ liệu

Các ràng buộc dữ liệu dùng để kiểm tra sự đúng đắn và tính toàn vẹn của dữ liệu. Ràng buộc gồm các loại sau:

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK

Ràng buộc NOT NULL

Là ràng buộc không cho phép cột chứa giá trị NULL (để trống).

Trong ví dụ dưới đây, cột **TenChiNhanh** không được phép để trống.

```
CREATE TABLE ChiNhanh(  
    MaChiNhanh    char(10) primary key,  
    TenChiNhanh   nvarchar2(30) NOT NULL  
);
```

Chú ý: trong trường hợp khóa chính chỉ có 1 cột, có thể khai báo từ khóa PRIMARY KEY ngay ở dòng khai báo cột như ví dụ trên.

Ràng buộc UNIQUE

Ràng buộc này quy định giá trị của một cột (hoặc một số cột) phải là duy nhất (không cho phép giá trị trùng lặp).

Ví dụ 1:

```
CREATE TABLE KhachHang(  
    SoCMT          char(9) primary key,  
    TenKhach       nvarchar2(50),  
    SoDienThoai    varchar2(15),  
    DiaChi         nvarchar2(100),  
    NgaySinh       date,  
    GioiTinh       number(1,0),  
    MaChiNhanh     char(10),  
    foreign key(MaChiNhanh) references ChiNhanh,  
    CONSTRAINT UNQ_SDT UNIQUE (SoDienThoai)  
);
```

Trong ví dụ này, cột **SoDienThoai** có giá trị duy nhất.

Ví dụ 2:

```
CREATE TABLE KhachHang (
    SoCMT          char(9) primary key,
    TenKhach       nvarchar2(50),
    SoDienThoai    varchar2(15),
    DiaChi         nvarchar2(100),
    NgaySinh       date,
    GioiTinh       number(1,0),
    MaChiNhanh     char(10),
    foreign key(MaChiNhanh) references ChiNhanh,
    CONSTRAINT UNQ_SDT UNIQUE (SoDienThoai, TenKhach)
);
```

Trong ví dụ này, tổ hợp cột **SoDienThoai** và **TenKhach** có giá trị duy nhất.

Ràng buộc PRIMARY KEY (khóa chính)

Cũng đưa ra ràng buộc về tính duy nhất của giá trị trong cột giống như UNIQUE nhưng cao cấp hơn. Mỗi bảng chỉ có một khóa chính. Khóa chính được tạo thành từ một hoặc một số cột. Những cột đóng vai trò làm khóa chính không được phép chứa giá trị NULL.

Ví dụ: (xem ở mục Cú pháp tạo bảng)

Ràng buộc FOREIGN KEY (khóa ngoại)

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

Ví dụ: (xem ở mục Cú pháp tạo bảng)

Chú ý: Không được phép tạo khóa ngoại cho những cột có kiểu dữ liệu: CLOB, NCLOB, BLOB, LONG, LONG RAW, TIMESTAMP WITH TIMEZONE

Ràng buộc CHECK

Ràng buộc này dùng để kiểm tra miền giá trị của một cột xác định.

Ví dụ:

```
CREATE TABLE TaiKhoan (
```


Ví dụ:

```
ALTER TABLE ChiNhanh MODIFY DiaChi nvarchar2(50) DEFAULT 'HN';
```

1.2.3.3. Thay đổi kiểu dữ liệu của cột

Cú pháp 1 (dùng để sửa 1 cột):

```
ALTER TABLE Tên_bảng MODIFY Tên_cột Kiểu_Dữ_Liệu_mới;
```

Cú pháp 2 (dùng để sửa nhiều cột):

```
ALTER TABLE Tên_bảng MODIFY (  
    Tên_cột_1 Kiểu_Dữ_Liệu_mới,  
    Tên_cột_2 Kiểu_Dữ_Liệu_mới,  
    .....  
);
```

1.2.3.4. Xóa cột

Cú pháp 1 (dùng để xóa 1 cột):

```
ALTER TABLE Tên_bảng DROP COLUMN tên_cột;
```

Cú pháp 2 (dùng để xóa nhiều cột):

```
ALTER TABLE Tên_bảng DROP COLUMN (tên_cột_1, tên_cột_2, ...);
```

1.2.3.5. Thêm ràng buộc

Cú pháp:

```
ALTER TABLE Tên_bảng ADD CONSTRAINT tên_ràng_buộc  
loại_ràng_buộc...;
```

Ví dụ:

```
ALTER TABLE taikhoan ADD CONSTRAINT ck_D CHECK(NgayMoTK <  
NgayDongTK);
```

1.2.3.6. Xóa ràng buộc

Cú pháp:

```
ALTER TABLE Tên_bảng DROP CONSTRAINT tên_ràng_buộc;
```

1.2.4. Các lệnh DDL khác

1.2.4.1. Xóa bảng

Cú pháp:

```
DROP TABLE tên_bảng [PURGE/CASCADE CONSTRAINTS];
```

Một số chú ý:

- Khi DROP TABLE thì xóa cấu trúc và toàn bộ dữ liệu của bảng.
- Tùy chọn PURGE: xóa bảng không cho phép flashback
- Tùy chọn CASCADE CONSTRAINTS: xóa bảng và các ràng buộc liên quan
- Chỉ có người tạo ra bảng hoặc người có quyền DBA thì mới được phép xóa bảng.

1.2.4.2. Đổi tên bảng

Cú pháp:

```
RENAME tên_bảng TO tên_bảng_mới;
```

Hoặc:

```
ALTER TABLE tên_bảng RENAME TO tên_bảng_mới;
```

1.3. Các lệnh DML

1.3.1. Thêm dữ liệu

Để thêm một bản ghi cho bảng ta dùng lệnh **INSERT**.

Cú pháp 1 (dùng để thêm giá trị cho tất cả các cột trong bảng):

```
INSERT INTO Tên_bảng VALUES (gt1, gt2, ...);
```

Chú ý: Thứ tự của **gt1**, **gt2** phải tương ứng với thứ tự của các cột trong bảng.

Cú pháp 2 (dùng để thêm giá trị cho một số cột trong bảng):

```
INSERT INTO Tên_bảng (cột1, cột2, ...) VALUES (gt1, gt2, ...);
```

Chú ý: Thứ tự của **gt1**, **gt2** phải tương ứng với **cột1**, **cột2**.

Cú pháp 3 (lấy giá trị từ bảng khác để thêm mới cho bảng).

```
INSERT INTO Tên_bảng (cột1, cột2, ...)
```

```
SELECT cotx, coty, ...
```

```
FROM...WHERE...
```


Chú ý: thứ tự của **cotx**, **cotx** phải tương ứng với thứ tự của **cột1**, **cột2**.

1.3.2. Sửa dữ liệu

Cú pháp:

```
UPDATE Tên_bảng  
SET cột1 = giá_trị1, ..., cộtn = giá_trị_n  
[WHERE điều_kiện];
```

Ví dụ:

```
UPDATE khachhang  
SET diachi = 'Cau Giay - Ha Noi'  
WHERE socmt = '151413097';
```

1.3.3. Xóa dữ liệu

Xóa từng bản ghi (row) dùng lệnh DELETE. Cú pháp như sau:

```
DELETE Tên_bảng WHERE [điều_kiện];
```

Ví dụ: xóa những khách hàng có tên bắt đầu bằng chữ T.

```
DELETE KhachHang WHERE TenKhach LIKE 'T%';
```

Xóa toàn bộ dữ liệu trong bảng dùng lệnh TRUNCATE. Cú pháp như sau:

```
TRUNCATE TABLE Tên_bảng;
```

Chú ý: Trong trường hợp cần xóa toàn bộ dữ liệu trong bảng, hãy dùng lệnh TRUNCATE để có tốc độ thực hiện nhanh hơn.

1.4. Lệnh DQL (truy vấn dữ liệu)

1.4.1. Quy tắc viết lệnh

- Không phân biệt chữ HOA, thường trong câu lệnh truy vấn
- Câu truy vấn có thể viết trên nhiều dòng
- Các từ khóa không được viết tắt hay phân cách trên nhiều dòng
- Câu truy vấn kết thúc bằng dấu ;

1.4.2. Cú pháp tổng quát

```
SELECT [DISTINCT] danh_sách_cột
FROM {table_name | view_name}
[WHERE điều_kiện]
[GROUP BY danh_sách_cột_1]
[HAVING điều_kiện_lọc]
[ORDER BY danh_sách_cột_2 [ASC | DESC]];
```

Một câu truy vấn **bắt buộc** phải có mệnh đề SELECT...FROM. Các mệnh đề khác là tùy chọn.

Ví dụ: Hiển thị danh sách khách hàng.

```
select socmt, tenkhach, sodienthoai, diachi
from khachhang;
```

Kết quả trả về như sau:

#	SOCMT	TENKHACH	SODIENTHOAI	DIACHI
1	151413097	Ly Nhan Dao	0988333666	Duc Tho - Ha Tinh
2	151011097	Tran Duc Nghia	0988333663	Ly Nhan - Ha Nam
3	252433097	Nguyen Thi Hang	0988333623	Giao Thuy - Nam Dinh
4	353423196	Le Hong Nam	0978453676	Tinh Gia - Thanh Hoa
5	251718097	Dinh Hong Nam	0912338766	Dong Da - Ha Noi
6	971063137	Ho Viet Duan	0932333098	Thanh Xuan - Ha Noi
7	123456789	Dinh Chi Long	0955356666	Vinh Yen - Vinh Phuc
8	856382748	Tran Hong Nam	0901098612	Quan 1 - HCM
9	104875748	Nguyen Ba Dao	0902334554	Luc Ngan - Bac Giang
10	184656387	Ngo The Vinh	0915123123	Luc Ngan - Bac Giang

1.4.3. Các phép toán

1.4.3.1. Các phép toán học

Các phép toán học bao gồm: +, -, *, /. Các phép toán này thường được dùng ở mệnh đề SELECT, WHERE, HAVING.

Ví dụ:

```
select socmt, sodu, sodu + 50000
from taikhoan;
```

Kết quả trả về như sau:

	SOCMT	SODU	SODU+50000
1	151413097	12000000	12050000
2	151011097	4450000	4500000
3	252433097	120000	170000
4	353423196	50000	100000
5	251718097	120000	170000
6	971063137	0	50000
7	123456789	400000	450000
8	856382748	50000	100000
9	104875748	0	50000
10	184656387	510000	560000

Chú ý: các phép toán toán học chỉ thay đổi kết quả hiển thị, còn giá trị thực trong bảng vẫn không thay đổi.

1.4.3.2. Đặt lại tên cột

Thông thường kết quả trả về của các câu truy vấn sẽ hiển thị tên cột giống như tên cột của các bảng trong cơ sở dữ liệu. Nếu muốn đặt lại tên cột cho rõ nghĩa hơn có thể dùng dùng pháp sau:

```
SELECT tên_cột_1 "Tên cột 1 mới", tên_cột_2 "Tên cột 2 mới"
FROM...
WHERE...
...
```

Tên cột mới đặt trong dấu nháy kép (“ ”) và có thể đặt tiếng Việt có dấu, có khoảng cách.

Ví dụ:

```
select      socmt, tenkhach "Tên khách hàng",
           sodienthoai "Số điện thoại", diachi "Địa chỉ"
from khachhang;
```

Kết quả trả về như sau:

	SOCMT	Tên khách hàng	Số điện thoại	Địa chỉ
1	151413097	Ly Nhan Dao	0988333666	Duc Tho - Ha Tinh
2	151011097	Tran Duc Nghia	0988333663	Ly Nhan - Ha Nam
3	252433097	Nguyen Thi Hang	0988333623	Giao Thuy - Nam Dinh
4	353423196	Le Hong Nam	0978453676	Tinh Gia - Thanh Hoa
5	251718097	Dinh Hong Nam	0912338766	Dong Da - Ha Noi
6	971063137	Ho Viet Duan	0932333098	Thanh Xuan - Ha Noi
7	123456789	Dinh Chi Long	0955356666	Vinh Yen - Vinh Phuc
8	856382748	Tran Hong Nam	0901098612	Quan 1 - HCM
9	104875748	Nguyen Ba Dao	0902334554	Luc Ngan - Bac Giang
10	184656387	Ngo The Vinh	0915123123	Luc Ngan - Bac Giang

1.4.3.3. Phép so sánh

Phép so sánh gồm: >, >=, <, <=, = và <>(!=). Các phép toán này thường được dùng ở mệnh đề WHERE hoặc HAVING, dùng để giới hạn kết quả trả về.

Cú pháp:

Tên_Cột <phép_so_sánh> Giá_trị

Ví dụ: Hiển thị những tài khoản có số dư nhiều hơn 10.000.000

```
select socmt, sodu
from taikhoan
where sodu > 10000000;
```

Kết quả trả về như sau:

	SOCMT	SODU
1	151413097	12000000
2	684756584	150000000
3	067463836	1234000000
4	104875748	19077700

1.4.3.4. Các phép toán liên quan đến chuỗi

LIKE/NOT LIKE

Phép toán LIKE dùng để tìm kiếm các giá trị kiểu chuỗi một cách gần đúng. Thường được dùng ở mệnh đề WHERE.

Phép toán LIKE dùng 2 ký tự đại diện:

- %: đại diện cho mọi chuỗi ký tự (kể cả chuỗi NULL)
- _: đại diện cho một ký tự bất kì (khác NULL)

Cú pháp:

Tên_cột LIKE/NOT LIKE 'mẫu ký tự'

Chú ý: Oracle có phân biệt HOA thường trong phép tìm kiếm gần đúng.

Ví dụ: Tìm những khách hàng có tên bắt đầu bằng từ *T* và kết thúc bằng chữ *m*.

```
select tenkhach
from khachhang
where tenkhach like 'T%m';
```

Kết quả trả về như sau:

	TENKHACH
1	Tran Hong Nam

Phép ghép chuỗi (||)

Phép toán này dùng để nối hai chuỗi hoặc giá trị của 2 cột với nhau.

Ví dụ:

```
select 'Ông/bà ' || tenkhach || ' có ngày sinh là: ' || ngaysinh
from khachhang;
```

Kết quả:

	'ÔNG/BÀ ' TENKHACH 'CÓ NGÀY SINH LÀ: ' NGAYSINH
1	Ông/bà Ly Nhan Dao có ngày sinh là: 12-JAN-78
2	Ông/bà Tran Duc Nghia có ngày sinh là: 17-MAY-90
3	Ông/bà Nguyen Thi Hang có ngày sinh là: 02-JAN-76
4	Ông/bà Le Hong Nam có ngày sinh là: 08-SEP-66
5	Ông/bà Dinh Hong Nam có ngày sinh là: 19-DEC-60
6	Ông/bà Ho Viet Duan có ngày sinh là: 30-DEC-84
7	Ông/bà Dinh Chi Long có ngày sinh là: 01-JAN-84
8	Ông/bà Tran Hong Nam có ngày sinh là: 12-AUG-83

1.4.3.5. Phép toán IN/NOT IN

Phép toán này dùng để kiểm tra xem cột nào đó có giá trị nằm trong danh sách giá trị cho trước hay không. Thường được dùng ở mệnh đề WHERE hoặc HAVING. Dùng cho mọi kiểu dữ liệu (số, chuỗi, date, ...).

Cú pháp:

```
Tên_cột IN/NOT IN (giá_trị_1, giá_trị_2, ...)
```

Ví dụ: Tìm khách hàng thuộc chi nhánh HN00000001, HN00000003, HN00000007.

```
select tenkhach, machinhanh
from khachhang
where machinhanh in ('HN00000001', 'HN00000003', 'HN00000007');
```

Kết quả:

	TENKHACH	MACHINHANH
1	Ly Nhan Dao	HN00000001
2	Tran Duc Nghia	HN00000001
3	Nguyen Thi Hang	HN00000003
4	Le Hong Nam	HN00000003
5	Dinh Hong Nam	HN00000003
6	Ho Viet Duan	HN00000001
7	Dinh Chi Long	HN00000001
8	Tran Hong Nam	HN00000001
9	La Quang Vinh	HN00000003
10	Hoang A Na	HN00000001

1.4.3.5. Phép toán BETWEEN/NOT BETWEEN

Phép toán này dùng để kiểm tra xem giá trị của một cột có thuộc vào một đoạn cho trước hay không. Thường được dùng ở mệnh đề WHERE hoặc HAVING.

Cú pháp:

Tên_cột **BETWEEN/NOT BETWEEN** giá_trị_1 **AND** giá_trị_2

Ví dụ: Tìm những khách hàng sinh từ năm 1978 đến năm 1982.

```
select tenkhach, ngaysinh
from khachhang
where ngaysinh between '1-JAN-1978' and '31-DEC-1982';
```

Kết quả:

	TENKHACH	NGAYSINH
1	Ly Nhan Dao	12-JAN-78
2	Nguyen Ba Dao	29-APR-82
3	Ngo The Vinh	05-JUN-80
4	La Quang Vinh	12-JAN-80
5	Ma Van Kháng	10-OCT-78

1.4.3.6. Phép toán IS NULL/IS NOT NULL

Phép toán này dùng để kiểm tra xem một ô nào đó đã nhập giá trị hay chưa. Những ô chưa nhập giá trị sẽ có giá trị NULL.

Ví dụ: Tìm những tài khoản chưa đóng.

```
select socmt, mataikhoan, ngaydongtk
from taikhoan
where ngaydongtk is null;
```

Kết quả:

	SOCMT	MATAIKHOAN	NGAYDONGTK
1	151413097	151413097000	(null)
2	151011097	151011097210	(null)
3	353423196	353423196011	(null)
4	251718097	251718097123	(null)
5	971063137	971063137987	(null)
6	123456789	123456789098	(null)
7	856382748	856382748596	(null)
8	184656387	184656387564	(null)

Chú ý: Không bao giờ được dùng phép toán = và <> đối với giá trị NULL.

1.4.3.7. Phép toán logic

Dùng để kết hợp các mệnh đề điều kiện. Các phép toán logic bao gồm: AND, OR, NOT

- AND: chỉ trả về kết quả khi và chỉ khi 2 mệnh đề đúng
- OR: trả về kết quả khi một trong 2 mệnh đề đúng.
- NOT: phủ định

Cú pháp:

<mệnh_đề_1> AND/OR [[NOT]mệnh_đề_2] AND/OR [[NOT]mệnh_đề_3]...

Ví dụ: Tìm những khách hàng không sống ở Hà Nội và sinh trước năm 1980.

```
select tenkhach, ngaysinh, diachi
from khachhang
where diachi not like '%Ha Noi' and ngaysinh < '1-JAN-1980';
```

Kết quả:

	TENKHACH	NGAYSINH	DIACHI
1	Ly Nhan Dao	12-JAN-78	Duc Tho - Ha Tinh
2	Nguyen Thi Hang	02-JAN-76	Giao Thuy - Nam Dinh
3	Le Hong Nam	08-SEP-66	Tinh Gia - Thanh Hoa
4	Hoang A Na	01-MAY-71	Can Loc - Ha Tinh
5	Tran Thi Lan	20-APR-77	Hai Hau - Nam Dinh

1.4.4. Sắp xếp dữ liệu

Dùng mệnh đề ORDER BY để sắp xếp dữ liệu trả về của câu truy vấn. Mệnh đề ORDER BY luôn luôn đứng ở vị trí cuối cùng trong câu truy vấn. Có thể sắp xếp theo nhiều cột, cột nào gần mệnh đề ORDER BY hơn sẽ có mức độ ưu tiên khi sắp xếp cao hơn. Chỉ định cách sắp xếp bằng từ ASC (tăng dần) hoặc DESC (giảm dần).

Cú pháp:

```
SELECT...

FROM...

...

ORDER BY tên_cột_1 ASC/DESC, tên_cột_2 ASC/DESC, ...
```

Ví dụ: Hiển thị danh sách khách hàng theo chiều tăng dần của mã chi nhánh và giảm dần của tên.

```
select machinhanh, tenkhach
from khachhang
order by machinhanh, tenkhach desc;
```

Kết quả:

	MACHINHANH	TENKHACH
1	HN00000001	Tran Hong Nam
2	HN00000001	Tran Duc Nghia
3	HN00000001	Ly Nhan Dao
4	HN00000001	Hoang A Na
5	HN00000001	Ho Viet Duan
6	HN00000001	Dinh Chi Long
7	HN00000002	Ngo The Vinh
8	HN00000002	Ma Van Kháng
9	HN00000003	Nguyen Thi Hang
10	HN00000003	Le Hong Nam

Chú ý: nếu không có từ khóa ASC/DESC đặt sau tên cột thì Oracle mặc định sắp xếp tăng dần.

1.4.5. Một số hàm cơ bản

1.4.5.1. Hàm xử lý chuỗi

Tên hàm	Giải thích
LENGTH(tên_cột)	Tính độ dài chuỗi
UPPER(tên_cột)	Chuyển thành chữ HOA
LOWER(tên_cột)	Chuyển thành chữ thường
INITCAP(tên_cột)	Các kí tự đầu tiên của một từ chuyển thành chữ HOA
LTRIM(tên_cột)	Cắt các dấu cách bên trái chuỗi (đầu chuỗi)
RTRIM(tên_cột)	Cắt các dấu cách bên phải chuỗi (cuối chuỗi)
TRIM(tên_cột)	Cắt các dấu cách ở đầu và cuối chuỗi
LPAD(a, n, [k])	Hiển thị cột a có độ dài n. Nếu a có độ dài nhỏ hơn n thì chèn thêm kí tự k vào đầu chuỗi cho đủ n. Nếu a có độ dài lớn hơn n thì cắt đi, chỉ giữ lại n kí tự bên trái.

RPAD(a, n, [k])	Giống LPAD nhưng căn phải
SUBSTR(tên_cột, vị_trí_bắt_đầu, độ_dài_chuỗi_con)	Cắt chuỗi con từ một cột cho trước
INSTR(tên_cột, chuỗi)	Tìm vị trí xuất hiện của chuỗi trong tên_cột

Ví dụ: hiển thị 5 kí tự đầu tiên trong địa chỉ của khách hàng và viết HOA.

```
select tenkhach, upper(substr(diachi, 1, 5))
from khachhang;
```

Kết quả:

TENKHACH	UPPER(SUBSTR(DIACHI,1,5))
1 Ly Nhan Dao	DUC T
2 Tran Duc Nghia	LY NH
3 Nguyen Thi Hang	GIAO
4 Le Hong Nam	TINH
5 Dinh Hong Nam	DONG
6 Ho Viet Duan	THANH
7 Dinh Chi Long	VINH
8 Tran Hong Nam	QUAN
9 Nguyen Ba Dao	LUC N
10 Ngo The Vinh	LUC N

1.4.5.2. Hàm xử lý số

Tên hàm	Giải thích
CEIL(n)	Lấy số nguyên nhỏ nhất lớn hơn hoặc bằng n
FLOOR(n)	Lấy số nguyên lớn nhất nhỏ hơn hoặc bằng n
MOD(m, n)	Phép chia lấy phần dư
ROUND(n, m)	Làm tròn số n đến m đơn vị (m: số các chữ số sau dấu thập phân)
SQRT(n)	Tính căn bậc 2 của n
POWER(m, n)	Tính m^n
EXP(n)	Tính e^n

1.4.5.3. Hàm xử lý ngày tháng

Tên hàm	Giải thích
MONTHS_BETWEEN(date1, date2)	Tính số tháng giữa date1 và date2
ADD_MONTHS(date, n)	Cộng n tháng vào date

SYSDATE	Trả về thời gian hiện tại
EXTRACT(YEAR/MONTH/DAY FROM date)	Lấy ra năm/tháng/ngày từ date
LAST_DAY(date)	Tìm ngày cuối cùng trong tháng của date

Ví dụ: lấy ra năm hiện tại.

```
select extract(year from sysdate) from dual;
```

Kết quả:

EXTRACT(YEARFROMSYSDATE)
1 2015

1.4.5.4. Hàm rẽ nhánh

Dùng để xử lý các rẽ nhánh trong lệnh truy vấn.

CASE...WHEN

Dạng 1:

```
CASE tên_cột    WHEN giá_trị_1 THEN kết_quả_1
                WHEN giá_trị_2 THEN kết_quả_2
                ...
                ELSE kết_quả_ngoại_lệ
END
```

Dạng 2:

```
CASE WHEN <so_sánh_1> THEN kết_quả_1
      WHEN <so_sánh_2> THEN kết_quả_2
      ...
      ELSE kết_quả_ngoại_lệ
END
```

Ví dụ: Hiển thị danh sách khách hàng và giới tính (Nam, nữ).

```
select tenkhach, case gioitinh when 1 then 'Nam'
                        when 0 then 'Nu'
                        else 'Khong xac dinh' end
from khachhang;
```

Kết quả:

TENKHACH	CASEGIOITINHWHEN1 THEN'NAM'WHEN0 THEN'NU'ELSE'KHONGXACDINH'END
1 Ly Nhan Dao	Nam
2 Tran Duc Nghia	Nam
3 Nguyen Thi Hang	Nu
4 Le Hong Nam	Nu
5 Dinh Hong Nam	Nam
6 Ho Viet Duan	Nam
7 Dinh Chi Long	Nam
8 Tran Hong Nam	Nam

DECODE

Cú pháp:

```
DECODE(tên_cột, giá_trị_1, kết_quả_1, ..., kết_quả_ngoại_lệ)
```

Ví dụ:

```
select tenkhach, decode(gioitinh, 1, 'Nam', 0, 'Nu', 'KXD')
from khachhang;
```

Kết quả: tương tự như ví dụ trên.

1.4.5.5. Hàm gộp/nhóm

Tên hàm	Giải thích
SUM(tên_cột)	Tính tổng. Chỉ áp dụng cho cột có kiểu dữ liệu số
COUNT(*)/COUNT(tên_cột)	COUNT(*): đếm số bản ghi COUNT(tên_cột): đếm các giá trị khác NULL
AVG(tên_cột)	Tính giá trị trung bình. Chỉ áp dụng cho cột có kiểu số
MIN(tên_cột)	Đối với dữ liệu số, date: tìm giá trị nhỏ nhất Đối với dữ liệu chuỗi: tìm giá trị nhỏ nhất theo thứ tự trong từ điển.
MAX(tên_cột)	Ngược lại với hàm MIN


Có thể dùng các hàm này để tác động lên toàn bộ dữ liệu hoặc các nhóm dữ liệu. Các hàm này chỉ dùng ở mệnh đề **SELECT** hoặc **HAVING**.

Tác động lên toàn bộ dữ liệu

Ví dụ 1: Tính tổng số dư của các tài khoản.

```
select sum(sodu) from taikhoan;
```

Kết quả:

	 SUM(SODU)
1	1447046900

Ví dụ 2: Tính tuổi trung bình của các khách hàng quê ở Hà Nội.

```
select avg(extract(year from sysdate) - extract(year from ngaysinh))
from khachhang
where diachi like '%Ha Noi';
```

Kết quả:

	AVG(EXTRACT(YEARFROMSYSDATE)-EXTRACT(YEARFROMNGAYSINH))
1	36.5

Tác động lên các nhóm dữ liệu

Muốn dùng hàm gộp cho từng nhóm dữ liệu thì dùng thêm mệnh đề **GROUP BY**. Cú pháp như sau:

```
SELECT hàm_gộp(tên_cột), danh_sách_cột
FROM ...
WHERE...
GROUP BY danh_sách_cột_1;
```

Với, `danh_sách_cột_1` và `danh_sách_cột` có các cột giống nhau nhưng thứ tự có thể khác nhau.

Chú ý: khi có `danh_sách_cột` xuất hiện cùng với các hàm gộp thì bắt buộc phải dùng mệnh đề **GROUP BY**.

Ví dụ: Thống kê số lượng khách hàng theo từng chi nhánh.

```
select count(socmt), machinhanh
from khachhang
group by machinhanh;
```

Kết quả:

	COUNT(SOCMT)	MACHINHANH
1	6	HN00000001
2	3	HN00000004
3	2	HN00000002
4	2	HN00000005
5	4	HN00000003

Lọc dữ liệu sau khi dùng hàm gộp

Sau khi dùng hàm gộp, muốn lọc dữ liệu thì dùng mệnh đề **HAVING** sau **GROUP BY**. Cú pháp như sau:

```
SELECT hàm_gộp(tên_cột), danh_sách_cột
FROM ...
WHERE...
GROUP BY danh_sách_cột_1
HAVING <điều_kiện_lọc>;
```


Format	Giải thích
SS	Giây
MI	Phút
HH, HH12, HH24	Giờ trong ngày
D	Ngày trong tuần (dạng số)
DAY	Ngày trong tuần (dạng chữ đầy đủ, ví dụ: Monday)
DY	Ngày trong tuần (dạng chữ viết tắt, ví dụ: MON)
DD	Ngày trong tháng (dạng số)
DDD	Ngày trong năm
W	Tuần trong tháng
WW	Tuần trong năm
MM	Tháng (dạng số)
MONTH	Tháng (dạng chữ đầy đủ, ví dụ: January)
MON	Tháng (dạng chữ viết tắt, ví dụ: JAN)
Q	Quý
Y, YY, YYYY, YYYY	Năm
DL	Ngày hiển thị theo kiểu đầy đủ (ví dụ: 01/12/2015)
DS	Ngày hiển thị theo kiểu ngắn gọn (ví dụ: 01/12/15)

Ví dụ 1: Hôm nay là thứ mấy?

```
select to_char(sysdate, 'DAY')
from dual;
```

Kết quả:

TO_CHAR(SYSDATE,'DAY')
1 THURSDAY

Ví dụ 2: Trần Thị Lan sinh vào tháng mấy?

```
select tenkhach, to_char(ngaysinh, 'MONTH')
from khachhang
where tenkhach = 'Tran Thi Lan';
```

Kết quả:

TENKHACH	TO_CHAR(NGAYSINH,'MONTH')
1 Tran Thi Lan	APRIL

Ví dụ 3: Thêm 1 bản ghi vào bảng TAIKHOAN, hãy định dạng ngày tháng sao cho có thể chạy trên các máy tính khác nhau.

```
insert into TaiKhoan values ('151413097000',
                             to_date('12/01/2012','DD/MM/YYYY'),
                             null,
                             '151413097',
                             12000000);
```

1.4.6. Nối bảng

Muốn lấy dữ liệu từ nhiều bảng, hãy sử dụng đến các phép nối (JOIN). Dưới đây là một số loại phép nối thường dùng.

1.4.6.1. Nối trong

Phép nối trong là phép nối chỉ trả về những bản ghi thỏa mãn điều kiện nối.

Cú pháp:

```
SELECT ...
FROM bảng_1 JOIN bảng_2 ON <điều_kiện_nối>
WHERE...
...
```

Ví dụ: Hiển thị danh sách khách hàng và số lượng tài khoản tương ứng của họ.

```
select tenkhach, count(mataikhoan)
from khachhang join taikhoan on khachhang.socmt = taikhoan.socmt
group by tenkhach;
```

Kết quả:

TENKHACH	COUNT(MATAIKHOAN)
1 Tran Thi Lan	1
2 Dinh Hong Nam	1
3 Ho Viet Duan	1
4 La Quang Vinh	2
5 Ly Nhan Dao	2
6 Tran Hong Nam	2
7 Nguyen Ba Dao	3
8 Ma Van Kháng	1
9 Ngo The Vinh	1
10 Tong Van Tinh	2

NATURAL JOIN

Là trường hợp đặc biệt của nối trong. Phép nối trong mà cột dùng để nối 2 bảng có tên giống nhau thì gọi là nối tự nhiên (NATURAL JOIN). Như vậy, chỉ sử dụng cú pháp của nối tự nhiên khi 2 bảng tồn tại cột giống nhau.

Cú pháp 1:

```
SELECT ...  
FROM bảng_1 NATURAL JOIN bảng_2  
WHERE...
```

...

Cú pháp 2:

```
SELECT ...  
FROM bảng_1 JOIN bảng_2 USING(tên_cột_giống_nhau)  
WHERE...
```

...

Ví dụ: giống như ví dụ trên nhưng cách viết theo kiểu nối tự nhiên như sau:

```
select tenkhach, count(mataikhoan)  
  
from khachhang natural join taikhoan  
  
group by tenkhach;
```

Hoặc:

```
select tenkhach, count(mataikhoan)  
  
from khachhang join taikhoan using(socmt)  
  
group by tenkhach;
```

1.4.6.2. Nối ngoài

Phép nối ngoài trả về các bản ghi thỏa mãn điều kiện nối và không thỏa mãn điều kiện nối. Những bản ghi không thỏa mãn điều kiện nối sẽ để giá trị NULL.

Có 3 loại nối ngoài:

Nối trái (LEFT OUTER JOIN)

Phép nối trái lấy các bản ghi của bảng bên trái nối với các bản ghi ở bảng bên phải. Những bản ghi thỏa mãn điều kiện nối sẽ giữ giá trị như bình thường. Những bản ghi không thỏa mãn điều kiện nối sẽ để giá trị NULL.

Minh họa cho phép nối trái giữa bảng NHANVIEN và PHONGBAN

MaNV	TenNV	MaPB
01	Thành	01
02	An	02
03	Phát	03

(NHANVIEN)

MaPB	TenPB
01	Marketing
03	Sales
04	HR
05	IT

(PHONGBAN)

Kết quả:

MaNV	TenNV	MaPB	MaPB	TenPB
01	Thành	01	01	Marketing
02	An	02	NULL	NULL
03	Phát	03	03	Sales

Cú pháp:

```
SELECT ...
FROM bảng_1 LEFT OUTER JOIN bảng_2 ON <điều_kiện_nối>
WHERE...
...
```

Nối phải (RIGHT OUTER JOIN)

Phép nối phải lấy các bản ghi của bảng bên phải nối với các bản ghi ở bảng bên trái. Những bản ghi thỏa mãn điều kiện nối sẽ giá về giá trị như bình thường. Những bản ghi không thỏa mãn điều kiện nối sẽ để giá trị NULL.

Minh họa cho phép nối phải giữa bảng NHANVIEN và PHONGBAN

MaNV	TenNV	MaPB
01	Thành	01
02	An	02
03	Phát	03

(NHANVIEN)

MaPB	TenPB
01	Marketing
03	Sales
04	HR
05	IT

(PHONGBAN)

Kết quả:

MaPB	TenPB	MaNV	TenNV	MaPB
01	Marketing	01	Thành	01
03	Sales	03	Phát	03
04	HR	NULL	NULL	NULL
05	IT	NULL	NULL	NULL

Cú pháp:

```

SELECT ...
FROM bảng_1 RIGHT OUTER JOIN bảng_2 ON <điều_kiện_nối>
WHERE...
...

```

Nối đầy đủ (FULL OUTER JOIN)

Là hợp của phép nối trái và nối phải (những bản ghi giống nhau chỉ hiển thị một lần).

Minh họa cho phép nối phải giữa bảng NHANVIEN và PHONGBAN

Kết quả:

MaNV	TenNV	MaPB	MaPB	TenPB
01	Thành	01	01	Marketing
02	An	02	NULL	NULL
03	Phát	03	03	Sales
NULL	NULL	NULL	04	HR
NULL	NULL	NULL	05	IT

Cú pháp:

```

SELECT ...
FROM bảng_1 FULL OUTER JOIN bảng_2 ON <điều_kiện_nối>
WHERE...
...

```

1.4.7. Truy vấn lồng (subquery)

Đặt một câu truy vấn bên trong câu truy vấn khác gọi là truy vấn lồng. Truy vấn con thường được đặt ở các mệnh đề: SELECT, FROM, WHERE, HAVING,... Có thể phân truy vấn lồng thành các loại như sau:

- *Truy vấn con đặt ở mệnh đề WHERE.* Với loại này, có thể lồng đến 255 mức.
- *Truy vấn chéo:* Cột ở truy vấn cha được sử dụng trong truy vấn con.
- *Truy vấn vô hướng:* Giá trị trả về của truy vấn con là một đại lượng vô hướng.
- *Loại khác:* truy vấn con đặt ở mệnh đề FROM, HAVING...

1.4.7.1. Truy vấn con đặt ở mệnh đề WHERE

Đây là loại truy vấn lồng thường gặp nhất. Truy vấn này phân làm 2 loại: truy vấn con trả về 1 bản ghi và truy vấn con trả về nhiều bản ghi.

Truy vấn con trả về 1 bản ghi

Đối với những truy vấn lồng này, sử dụng các phép toán so sánh như các truy vấn thông thường.

Ví dụ 1: Tìm khách hàng cùng tuổi với Đinh Chí Long.

```
select tenkhach, ngaysinh
from khachhang
where extract(year from ngaysinh) =
      (select extract(year from ngaysinh)
       from khachhang
       where tenkhach = 'Đinh Chí Long')
and tenkhach <> 'Đinh Chí Long';
```

Kết quả:

R	TENKHACH	R	NGAYSINH
1	Ho Viet Duan	30-DEC-84	

Ví dụ 2: Tìm những khách hàng nhiều hơn tuổi của Hồ Viết Duẩn và chi nhánh với Nguyễn Thị Hằng.

```
select tenkhach, ngaysinh, machinhanh
from khachhang
where extract(year from ngaysinh) <
      (select extract(year from ngaysinh)
       from khachhang
       where tenkhach = 'Ho Viet Duan')
and machinhanh = (select machinhanh
                  from khachhang
                  where tenkhach = 'Nguyễn Thị Hằng')
and tenkhach not in ('Nguyễn Thị Hằng', 'Ho Viet Duan');
```

Kết quả:

R	TENKHACH	R	NGAYSINH	R	MACHINHANH
1	Le Hong Nam	08-SEP-66	HN00000003		
2	Đinh Hong Nam	19-DEC-60	HN00000003		
3	La Quang Vinh	12-JAN-80	HN00000003		

1.4.7.2. Truy vấn con trả về nhiều bản ghi

Vì truy vấn con trả về nhiều bản ghi, do vậy không thể dùng các phép toán so sánh như truy vấn thông thường. Khi đó cần sử dụng thêm từ khóa SOME/ANY, ALL hoặc sử dụng phép toán IN/NOT IN.

Ý nghĩa của các từ khóa:

Phép toán và từ khóa	Ý nghĩa
< ANY	Nhỏ hơn giá trị lớn nhất
<= ANY	Nhỏ hơn hoặc bằng giá trị lớn nhất
> ANY	Lớn hơn giá trị nhỏ nhất
= ANY	Tương tự như phép toán IN
< ALL	Nhỏ hơn giá trị nhỏ nhất
> ALL	Lớn hơn giá trị lớn nhất
<> ALL	Tương tự phép toán NOT IN

Ví dụ 1: Tìm những khách hàng lớn hơn tuổi của Trần Bá Đạo, Hoàng A Na và Trần Hồng Nam.

```
select tenkhach, ngaysinh
from khachhang
where extract(year from ngaysinh) <
      all(select extract(year from ngaysinh)
         from khachhang
         where tenkhach in ('Tran Ba Dao',
                           'Hoang A Na',
                           'Tran Hong Nam'));
```

Kết quả:

	TENKHACH	NGAYSINH
1	Le Hong Nam	08-SEP-66
2	Dinh Hong Nam	19-DEC-60

1.4.7.3. Truy vấn chéo (correlated subquery)

Truy vấn con sử dụng một hoặc một số cột của truy vấn cha gọi là truy vấn chéo.

Ví dụ: Tìm khách hàng trẻ nhất ở mỗi chi nhánh.

```
select tenkhach, extract(year from ngaysinh), machinhanh
from khachhang k1
where extract(year from ngaysinh) =
      (select max(extract(year from ngaysinh))
       from khachhang k2
       where k1.machinhanh = k2.machinhanh);
```

Kết quả:

	TENKHACH	EXTRACT(YEARFROMNGAYSINH)	MACHINHANH
1	Tran Duc Nghia	1990	HN00000001
2	Ngo The Vinh	1980	HN00000002
3	La Quang Vinh	1980	HN00000003
4	Tong Van Tinh	1992	HN00000004
5	Bui An Duy	1983	HN00000005

1.4.7.4. Truy vấn con vô hướng

Giá trị trả về của truy vấn con là một giá trị xác định. Truy vấn con có thể đặt ở mệnh đề SELECT, ORDER BY, WHERE, VALUES (của lệnh INSERT), nhưng không được phép đặt ở mệnh đề GROUP BY, HAVING.

Ví dụ: Hiển thị tên khách và số lượng tài khoản của họ.

```
select tenkhach, (select count(socmt)
                  from taikhoan
                  where taikhoan.socmt = kháchhang.socmt) "SoTK"
from kháchhang;
```

Kết quả:

R	TENKHACH	R	So TK
1	Ly Nhan Dao	2	
2	Tran Duc Nghia	1	
3	Nguyen Thi Hang	2	
4	Le Hong Nam	1	
5	Dinh Hong Nam	1	
6	Ho Viet Duan	1	
7	Dinh Chi Long	2	
8	Tran Hong Nam	2	
9	Nguyen Ba Dao	3	
10	Ngo The Vinh	1	

1.4.7.5. Truy lồng vấn khác

Ngoài ra, truy vấn con có thể đặt ở mệnh đề FROM, HAVING, ...

Ví dụ 1: hiển thị danh sách 4 khách hàng trẻ nhất.

```
select tenkhach, tuoi
from (select tenkhach,
             extract(year from sysdate) - extract(year from ngaysinh) Tuoi
      from kháchhang
      order by Tuoi)
where rownum <= 4;
```

Kết quả:

R	TENKHACH	R	TUOI
1	Tong Van Tinh	23	
2	Tran Duc Nghia	25	
3	Ho Viet Duan	31	
4	Dinh Chi Long	31	

Chú ý: Trong Oracle không có mệnh đề SELECT TOP, do vậy để lấy ra *n* giá trị lớn/hoặc nhỏ nhất thì phải dùng đến truy vấn lồng.

Ví dụ 2: Hiển thị những chi nhánh có nhiều khách hơn chi nhánh Cầu Giấy.

```
select chinhanh.tenchinhanh, count(khachhang.socmt) "SoKhach"
from khachhang natural join chinhanh
group by chinhanh.tenchinhanh
having count(khachhang.socmt) >
        (select count(socmt)
         from khachhang natural join chinhanh
         where chinhanh.tenchinhanh = 'Dong Da');
```

Kết quả:

TENCHINHANH	SoKhach
1 Cau Giay	6

1.6. View

View là một bảng logic hay còn gọi là bảng ảo dùng để hiển thị dữ liệu từ một hoặc nhiều bảng khác. View không phải là nơi lưu trữ dữ liệu. Dữ liệu của view lấy từ các bảng vật lý. Mọi tác động lên view đều gây ảnh hưởng đến bảng gốc và ngược lại.

Cú pháp tạo view:

```
CREATE [OR REPLACE] [FORCE] VIEW tên_view[cột1, cột2,...]
AS câu_lệnh_truy_vấn
[WITH CHECK OPTION];
```

Với:

- OR REPLACE: nếu đã tồn tại view cùng tên thì thay thế bằng view mới.
- FORCE: tạo view trong cả trường hợp tên bảng trong câu truy vấn không tồn tại.
- Cột1, cột2, ...: tên cột của view
- WITH CHECK OPTION:

Người dùng có thể SELECT dữ liệu trên view giống như trên bảng. Các thao tác INSERT, UPDATE, DELETE thì chỉ một số trường hợp mới thực hiện được. Sau đây là một số lưu ý:

- Không thể INSERT, UPDATE trên view khi câu truy vấn của view chứa các toán tử JOIN, SET, DISTINCT, GROUP BY.
- Không thể INSERT, UPDATE trên view nếu view dùng WITH CHECK OPTION.
- Không thể dùng insert trên view nếu trong câu truy vấn của view có dùng lệnh DECODE.

1.7. Index

1.7.1. Giới thiệu

Index là cấu trúc dữ liệu tùy chọn, xây dựng cho một bảng nhất định dùng để tăng tốc độ thực hiện các câu truy vấn.

Sử dụng index trong một số trường hợp sau:

- Tăng tốc độ tìm kiếm cho một hoặc một số trường nào đó trong bảng.
- Dùng cho những cột để nối giữa các bảng

Không nên dùng index trong một số trường hợp:

- Bảng có ít dữ liệu
- Số lượng dữ liệu lấy ra của một câu truy vấn lớn hơn 15% dữ liệu của toàn bảng

Khi khai báo ràng buộc `UNIQUE` hoặc khóa chính cho một hoặc một số cột thì Oracle tự động tạo index cho các cột đó.

Không giới hạn số lượng index cho một bảng. Tuy nhiên, không nên tạo quá nhiều vì sẽ gây ảnh hưởng đến tốc độ `INSERT`, `UPDATE`, `DELETE` dữ liệu.

Có 2 loại index:

- B-tree index (*mặc định*)
 - Sắp xếp giá trị khóa & ROWID dưới dạng B-tree
 - Phù hợp với những cột có ít giá trị giống nhau
 - Hỗ trợ “*row locking*”
 - Có 2 loại: unique và non-unique
- Bitmap
 - Khóa và ROWID lưu dưới dạng bitmap
 - Phù hợp với những cột có nhiều giá trị giống nhau
 - Không hỗ trợ “*row-locking*”

1.7.2. Tạo, xóa index

Cú pháp tạo B-tree index:

```
CREATE [UNIQUE] INDEX tên_index ON tên_bảng(cột_1, cột_2, ...);
```

Cú pháp tạo Bitmap index:

```
CREATE BITMAP INDEX tên_index ON tên_bảng(cột_1, cột_2, ...);
```

Xóa index:

```
DROP INDEX tên_index;
```

1.8. Sequence

1.8.1. Giới thiệu

Là đối tượng dùng để tự động sinh ra các số nguyên theo thứ tự nào đó. Sequence thường được dùng để sinh ra giá trị tự động cho khóa chính.

Đặc điểm chính của sequence:

- Mỗi sequence có 1 tên xác định
- Không gắn với 1 cột hay 1 bảng nào
- Có thể tạo ra số nguyên theo thứ tự tăng hoặc giảm dần đều
- Khoảng cách giữa 2 số nguyên do người dùng tùy đặt
- Có thể quay vòng nếu giá trị sinh ra đã đạt ngưỡng
- START WITH và INCREMENT BY là 2 thành phần bắt buộc phải có trong khai báo sequence, các thành phần còn lại là tùy chọn.

1.8.2. Tạo và sử dụng sequence

Cú pháp tạo sequence:

```
CREATE SEQUENCE tên_dãy_số START WITH n
                        INCREMENT BY m
                        MAXVALUE s /NOMAXVALUE
                        MINVALUE p /NOMINVALUE
                        CYCLE/NOCYCLE
                        CACHE t/NOCACHE;
```

Với:

- START WITH: là giá trị bắt đầu của dãy số
- INCREMENT BY: khoảng cách giữa 2 số

- MAXVALUE: giá trị lớn nhất của dãy số, dùng NOMAXVALUE nếu muốn dãy số vô hạn
- MINVALUE: giá trị nhỏ nhất của dãy số, dùng NOMINVALUE nếu muốn dãy số vô hạn
- CYCLE: cho phép dãy số quay vòng nếu đạt ngưỡng (lớn hoặc nhỏ nhất). NOCYCLE thì ngược lại
- CACHE: số lượng giá trị thường trực trong bộ nhớ đệm. NOCACHE: không lưu giá trị nào của dãy số trong bộ nhớ đệm.

Để làm việc với sequence, dùng 2 thuộc tính sau:

- tên_dãy_số.CURVAL: giá trị hiện tại của dãy
- tên_dãy_số.NEXTVAL: giá trị tiếp theo của dãy

Ví dụ tạo dãy số có tên PKValue như sau:

```
CREATE SEQUENCE PKValue
START WITH 1
INCREMENT BY 2
MAXVALUE 10000
MINVALUE 1;
```

Sử dụng sequence trong câu lệnh insert như sau:

```
insert into machinhanh values(PKValue.Nextval, 'Hoàng Mai');
```

Hiển thị giá trị của sequence như sau:

```
select PKValue.Nextval from dual;
```

1.8.3. Sửa và xóa sequence

Có thể sửa mọi thuộc tính, trừ START WITH. Cú pháp sửa sequence như sau:

```
ALTER SEQUENCE tên_dãy_số ...
```

Ví dụ:

```
alter sequence PKValue increment by 1;
```

Xóa sequence dùng cú pháp sau:

```
DROP SEQUENCE tên_dãy_số;
```

1.9. Synonym

Là tên viết tắt cho mọi đối tượng trong Oracle. Cú pháp tạo synonym:

```
CREATE PUBLIC SYNONYM tên_viết_tắt FOR [schema.]tên_đối_tượng;
```

Dùng synonym có lợi sau:

- Không tốn thêm dung lượng lưu trữ
- Dùng tên các đối tượng một cách ngắn gọn
- Tăng tính bảo mật