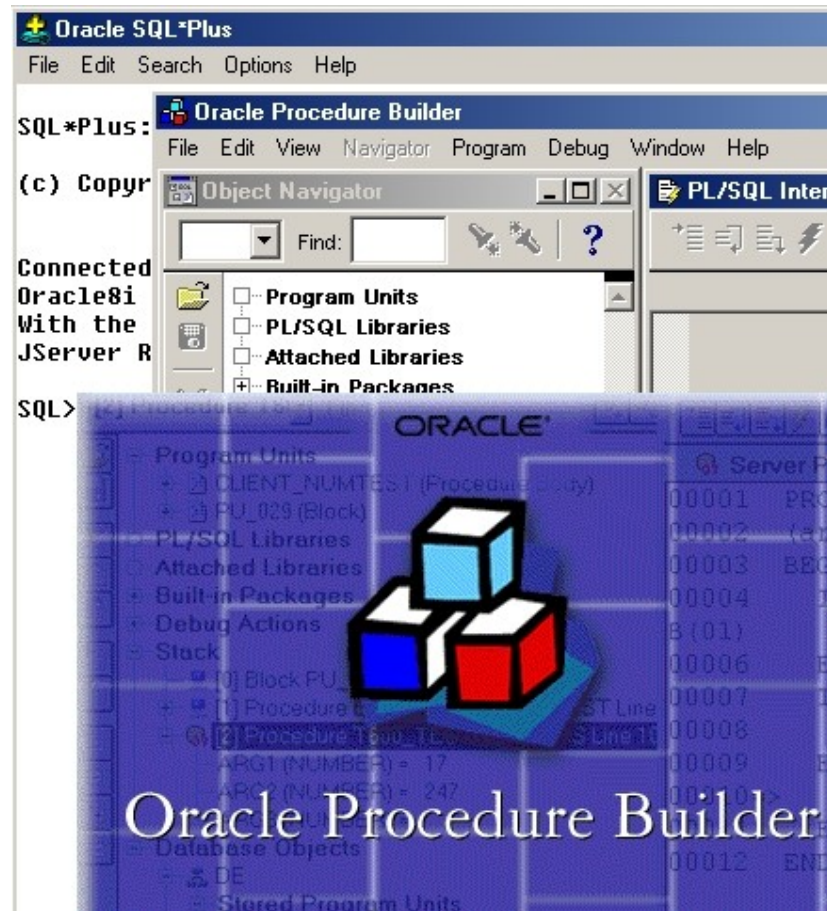


# SQL và PL/SQL

## Cơ bản



# MỤC LỤC

MỤC LỤC ...	...1
CHƯƠNG 1. GIỚI THIỆU CHUNG ...	...5
1.1. NGÔN NGỮ SQL ...	..5
1.1.1. Lịch sử phát triển của ngôn ngữ SQL ...	..5
1.1.2. Chuẩn SQL ...	..5
1.2. CÁC KHÁI NIỆM CƠ BẢN TRONG CƠ SỞ DỮ LIỆU ...	...5
1.2.1. Các thành phần logic trong database...	..5
1.2.2. Các đối tượng trong database ...	...6
1.2.3. Các nhóm lệnh SQL cơ bản ...	..6
1.3. CƠ SỞ DỮ LIỆU THỰC HÀNH...	..7
1.3.1. Mô hình dữ liệu ...	..7
1.3.2. Cấu trúc bảng dữ liệu ...	..7
CHƯƠNG 2. LỆNH TRUY VẤN CƠ BẢN ...	..9
2.1. CÂU LỆNH TRUY VẤN ...	..9
2.1.1. Quy tắc viết lệnh ...	...9
2.1.2. Câu lệnh truy vấn cơ bản ...	..9
2.1.3. Các thành phần khác của mệnh đề SELECT ...	..9
2.1.4. Phân biệt giá trị dữ liệu trả về ...	..10
2.1.5. Giá trị NULL ...	..11
2.2. SQL*PLUS, CÔNG CỤ TƯƠNG TÁC LỆNH SQL VỚI DATABASE ...	...11
2.2.1. Câu lệnh tương tác của SQL*Plus ...	..11
2.2.2. Phân nhóm câu lệnh trong SQL*Plus...	...12
2.2.3. Chi tiết các lệnh SQL*Plus cơ bản ...	..13
2.3. BÀI TẬP ...	..15
CHƯƠNG 3. TRUY VẤN DỮ LIỆU CÓ ĐIỀU KIỆN ...	..17
3.1. CÁC GIỚI HẠN TRONG TRUY VẤN DỮ LIỆU ...	...17
3.1.1. Mệnh đề WHERE...	...17
3.1.2. Các toán tử sử dụng trong mệnh đề WHERE ...	..18
3.1.3. Ví dụ sử dụng các toán tử điều kiện ...	..19
3.2. SẮP XẾP DỮ LIỆU TRẢ VỀ...	..20
3.2.1. Mệnh đề ORDER BY ...	...20
3.2.2. Sắp xếp nhiều cột dữ liệu trả về...	..20
3.3. BÀI TẬP ...	..21
CHƯƠNG 4. CÁC HÀM SQL ...	...23
4.1. TỔNG QUAN VỀ HÀM SQL...	..23
4.1.1. Cấu trúc hàm SQL ...	..23
4.1.2. Phân loại hàm SQL ...	...23
4.2. HÀM SQL THAO TÁC TRÊN TỪNG DÒNG DỮ LIỆU...	..24
4.2.1. Các hàm thao tác trên kiểu dữ liệu số...	..24
4.2.2. Các hàm thao tác trên kiểu dữ liệu ký tự...	..26
4.2.3. Các hàm thao tác trên kiểu dữ liệu thời gian...	..30
4.2.4. Các hàm chuyển đổi kiểu ...	...32
4.3. HÀM THAO TÁC TRÊN TẬP HỢP ...	...34
4.3.1. Các hàm tác động trên nhóm ...	...34
4.3.2. Mệnh đề GROUP BY ...	...35
4.4. MỘT SỐ HÀM MỚI BỔ SUNG TRONG Oracle9i ...	..36
4.4.1. Hàm NULLIF ...	..36
4.4.2. Hàm COALSCE ...	..36
4.4.3. Câu lệnh case ...	..36

4.5. BÀI TẬP ...	..36
4.5.1. Hàm trên từng dòng dữ liệu ...	..36
4.5.2. Hàm trên nhóm dữ liệu ...	...39
<b>CHƯƠNG 5. LỆNH TRUY VẤN DỮ LIỆU MỞ RỘNG...</b>	<b>..40</b>
5.1. KẾT HỢP DỮ LIỆU TỪ NHIỀU BẢNG ...	...40
5.1.1. Mối liên kết tương đương ...	..40
5.1.2. Mối liên kết không tương đương ...	..40
5.1.3. Mối liên kết cộng ...	..40
5.1.4. Liên kết của bảng với chính nó (tự thân)...	..41
5.1.5. Cách biểu diễn kết nối mới trong Oracle 9i ...	...41
5.1.6. Các toán tử tập hợp ...	..42
5.2. LỆNH TRUY VẤN LÔNG ...	...43
5.2.1. Câu lệnh SELECT lồng nhau. ..	...43
5.2.2. Toán tử SOME/ANY/ALL/NOT IN/EXISTS ...	..43
5.3. CẤU TRÚC HÌNH CÂY ...	..44
5.3.1. Cấu trúc hình cây trong 1 table ...	...44
5.3.2. Kỹ thuật thực hiện ...	..44
5.3.3. Mệnh đề WHERE trong cấu trúc hình cây...	..45
5.4. BÀI TẬP ...	..46
<b>CHƯƠNG 6. BIẾN RUNTIME ...</b>	<b>...50</b>
6.1. DỮ LIỆU THAY THẾ TRONG CÂU LỆNH ...	..50
6.2. LỆNH DEFINE ...	..50
6.3. LỆNH ACCEPT ...	...51
6.4. BÀI TẬP ...	..51
<b>CHƯƠNG 7. TABLE VÀ CÁC LỆNH SQL VỀ TABLE...</b>	<b>...52</b>
7.1. LỆNH TẠO TABLE...	52
7.1.1. Cú pháp tạo bảng ...	52
7.1.2. Tính toán kích thước table (tham khảo) ...	53
7.2. MỘT SỐ QUY TẮC KHI TẠO TABLE ...	54
7.2.1. Quy tắc đặt tên Object ...	54
7.2.2. Quy tắc khi tham chiếu đến Object ...	54
7.3. Các Kiểu dữ liệu cơ bản...	55
7.3.1. Kiểu CHAR ...	55
7.3.2. Kiểu VARCHAR2 ...	55
7.3.3. Kiểu VARCHAR ...	56
7.3.4. Kiểu NUMBER ...	56
7.3.5. Kiểu FLOAT ...	56
7.3.6. Kiểu LONG ...	56
7.3.7. Kiểu DATE ...	57
7.3.8. Kiểu RAW và kiểu LONG RAW...	58
7.3.9. Kiểu ROWID ...	58
7.3.10. Kiểu MLSLABEL ...	58
7.3.11. Chuyển đổi kiểu ...	58
7.4. RÀNG BUỘC DỮ LIỆU TRONG TABLE ...	59
7.4.1. NULL/NOT NULL ...	59
7.4.2. UNIQUE ...	59
7.4.3. PRIMARY KEY ...	59
7.4.4. FOREIGN KEY ( Referential ) ...	60
7.4.5. CHECK ...	60
7.5. LỆNH DDL CAN THIỆP TỚI TABLE ...	60
7.5.1. Chính sửa cấu trúc table ...	60
7.5.2. Các lệnh DDL khác ...	61
7.5.3. Chú dẫn cho table ...	61
7.5.4. Thay đổi tên object...	62
7.5.5. Xóa dữ liệu của table ...	62

7.6. THÔNG TIN VỀ TABLE TRONG TỪ ĐIỂN DỮ LIỆU...	...62
7.7. BÀI TẬP ...	..63
<b>CHƯƠNG 8. CÁC LỆNH THAO TÁC DỮ LIỆU...</b>	<b>..64</b>
8.1. THAO TÁC DỮ LIỆU TRONG TABLE ...	..64
8.1.1. Thêm mới dòng dữ liệu ...	...64
8.1.2. Cập nhật dòng dữ liệu ...	..65
8.1.3. Lệnh Merge...	...65
8.1.4. Xóa dòng dữ liệu...	..66
8.1.5. Lỗi ràng buộc dữ liệu ...	...66
8.2. LỆNH ĐIỀU KHIỂN GIAO DỊCH...	..66
8.3. BÀI TẬP ...	..67
<b>CHƯƠNG 9. SEQUENCE VÀ INDEX...</b>	<b>...68</b>
9.1. SEQUENCE...	...68
9.1.1. Tạo Sequence...	..68
9.1.2. Thay đổi và huỷ sequence ...	..69
9.2. INDEX...	...69
9.2.1. Tạo index ...	...69
9.2.2. Sử dụng index...	..69
9.3. BÀI TẬP ...	..70
<b>CHƯƠNG 10. VIEWS ...</b>	<b>...71</b>
10.1. VIEWS ...	...71
10.1.1. Tạo view ...	..71
10.1.2. Xóa các view ...	..71
10.2. BÀI TẬP ...	...72
<b>CHƯƠNG 11. QUYỀN VÀ BẢO MẬT ...</b>	<b>..73</b>
11.1. QUYỀN - PRIVILEGE ...	..73
11.2. ROLE...	..74
11.3. SYNONYM...	...74
<b>CHƯƠNG 12. GIỚI THIỆU NGÔN NGỮ PL/SQL...</b>	<b>..76</b>
12.1. TỔNG QUAN VỀ PL/SQL ...	...76
12.1.1. Cú pháp lệnh PL/SQL ...	...76
12.1.2. Khối lệnh PL/SQL ...	...76
12.2. LỆNH LẬP TRÌNH PL/SQL ĐƠN GIẢN ...	...77
12.2.1. Lệnh IF...	..77
12.2.2. Lệnh lặp LOOP không định trước ...	...78
12.2.3. Lệnh lặp LOOP có định trước ...	...78
12.2.4. Lệnh lặp WHILE ...	...78
12.2.5. Lệnh GOTO, nhảy vô điều kiện ...	..78
12.3. GIỚI THIỆU CURSOR ...	...79
12.4. CÁC KIỂU DỮ LIỆU THÔNG DỤNG...	..81
12.4.1. Kiểu dữ liệu Table ...	...81
12.4.2. Kiểu dữ liệu Record ...	...81
12.4.3. Sao kiểu dữ liệu một dòng ...	..82
12.4.4. Sao kiểu dữ liệu của một cột ...	..82
12.4.5. Lệnh SELECT... INTO...	...82
12.5. BÀI TẬP ...	...83
<b>CHƯƠNG 13. GIỚI THIỆU PROCEDURE BUILDER ...</b>	<b>..84</b>
13.1. CÁC THÀNH PHẦN TRONG PROCEDURE BUILDER ...	...84
13.1.1. Object Navigator ...	..84
13.1.2. Program Unit Editor...	..85
13.1.3. Store Program Unit Editor ...	...85

13.1.4. Database Trigger Edditor ...	.85
13.2. CÁC HÀM, THỦ TỤC ...	.86
13.2.1. Tạo hàm, thủ tục trên Client ...	...86
13.2.2. Tạo hàm, thủ tục trên Server ...	.86
13.2.3. Dò lỗi đối với các hàm, thủ tục ...	..87
<b>CHƯƠNG 14. GIỚI THIỆU CÁC THỦ TỤC, HÀM VÀ PACKAGE ...</b>	<b>...88</b>
14.1. THỦ TỤC ...	..88
14.1.1. Tạo thủ tục ...	..88
14.1.2. Huỷ bỏ thủ tục ...	...89
14.1.3. Các bước lưu giữ một thủ tục ...	...89
14.2. HÀM ...	...89
14.2.1. Tạo hàm...	...90
14.2.2. Thực hiện một hàm ...	..90
14.2.3. Lợi ích của việc sử dụng hàm ...	...91
14.2.4. Một số hạn chế khi sử dụng hàm trong câu lệnh SQL ...	..91
14.2.5. Huỷ bỏ hàm...	.91
14.2.6. Hàm và thủ tục ...	..92
14.3. PACKAGE...	...92
14.3.1. Cấu trúc của package ...	...92
14.3.2. Tạo package ...	..93
14.3.3. Huỷ package ...	..95
14.3.4. Lợi ích của việc sử dụng package ...	..95
14.3.5. Một số package chuẩn của Oracle ...	..96
<b>CHƯƠNG 15. DATABASE TRIGGER ...</b>	<b>.97</b>
15.1. TẠO TRIGGER ...	..97
15.1.1. Phân loại trigger ...	...97
15.1.2. Lệnh tạo trigger ...	..98
15.1.3. Sử dụng Procedure builder để tạo trigger ...	...99
15.2. QUẢN LÝ TRIGGER ...	...100
15.2.1. Phân biệt database trigger ...	...100
15.2.2. Thay đổi trạng thái của database trigger ...	...101
15.2.3. Huỷ bỏ trigger ...	..101
15.2.4. Lưu ý khi sử dụng trigger ...	..102
<b>PHỤ LỤC ...</b>	<b>...103</b>
A - TÀI LIỆU THAM KHẢO ...	..103
B - DANH MỤC CÁC HÌNH VẼ...	..103

## Chương 1. GIỚI THIỆU CHUNG

### 1.1. NGÔN NGỮ SQL

#### 1.1.1. Lịch sử phát triển của ngôn ngữ SQL

Mô hình cơ sở dữ liệu (CSDL) quan hệ - RDBMS, do E.F Codd đưa ra vào đầu thập kỷ 70. Từ đó đến nay, nó liên tục phát triển trở thành mô hình CSDL phổ biến bậc nhất. Mô hình quan hệ gồm các thành phần sau:

- ③ Tập hợp các đối tượng và / hoặc các mối quan hệ
- ③ Tập hợp các xử lý tác động tới các quan hệ
- ③ Ràng buộc dữ liệu đảm bảo tính chính xác và nhất quán.

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.

#### 1.1.2. 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.

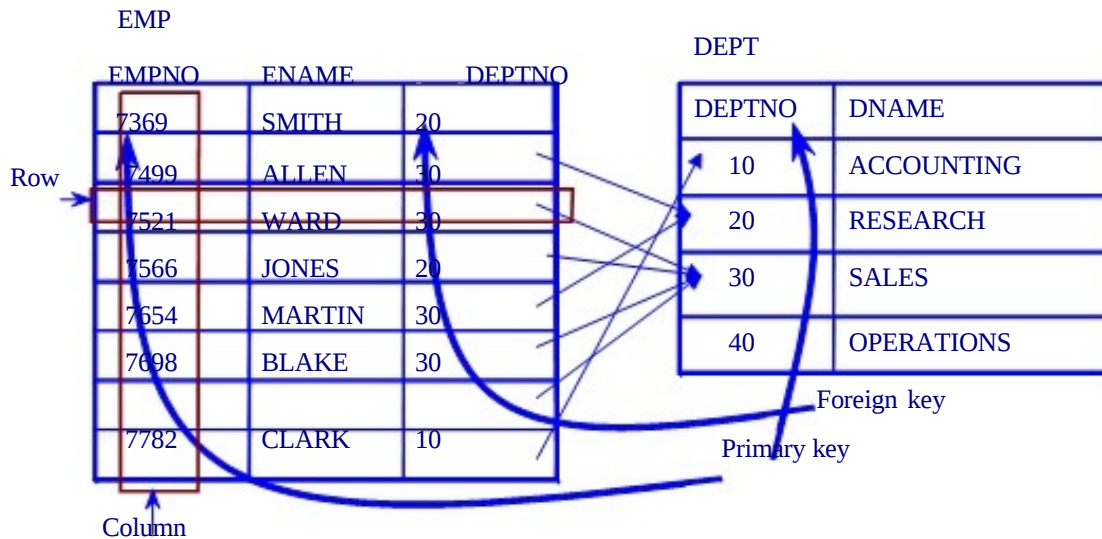
## 1.2. CÁC KHÁI NIỆM CƠ BẢN TRONG CƠ SỞ DỮ LIỆU

### 1.2.1. Các thành phần logic trong database

Thành phần	Diễn giải
Table	Cấu trúc lưu trữ cơ bản nhất trong CSDL quan hệ (RDBMS), nó bao gồm 1 hoặc nhiều columns (cột dữ liệu) với 0 hoặc nhiều rows (dòng dữ liệu).
Row	Tổ hợp những giá trị của Column trong bảng. Một row còn được gọi là 1 record (bản ghi).
Column	Quy định một loại dữ liệu trong bảng. Ví dụ: loại dữ liệu tên phòng ban có trong bảng phòng ban. Ta thể hiển thị column này thông qua tên column và có thể kèm theo một vài thông tin khác về column như kiểu dữ liệu, độ dài của dữ liệu.
Field	Giao của column và row. Field chính là nơi chứa dữ liệu. Nếu không có dữ liệu trong field ta nói field có giá trị là NULL.
Primary Key	Là một column hoặc một tập các column xác định tính duy nhất của các rows ở trong bảng. Ví dụ DEPTNO là Primary Key của bảng DEPT vì nó được dùng để xác định duy nhất một phòng ban trong bảng DEPT mà đại diện là một row dữ liệu.

	Primary Key nhất thiết phải có số liệu.
Foreign Key	Là một column hoặc một tập các columns có tham chiếu tới chính bảng đó hoặc một bảng khác. Foreign Key xác định mối quan hệ giữa các bảng.
Constraints	Là các ràng buộc đối với dữ liệu trong các bảng thuộc database. Ví dụ: Foreign Key, Primary Key...

Ví dụ: minh họa các thành phần logic trong database



Hình vẽ 1. Minh họa các thành phần logic trong database

### 1.2.2. Các đối tượng trong database

Đối tượng	Diễn giải
Table	Cấu trúc lưu trữ cơ bản nhất trong CSDL quan hệ (RDBMS), gồm row và column
View	Là cấu trúc logic hiển thị dữ liệu từ 1 hoặc nhiều bảng
Sequence	Lấy sinh giá trị cho các primary key
Index	Tăng tính thực thi cho câu lệnh truy vấn
Synonym	Tên tương đương của đối tượng
Program unit	Tập hợp các câu lệnh thực hiện được viết bởi ngôn ngữ SQL và PL/SQL, bao gồm Procedure, function, package...

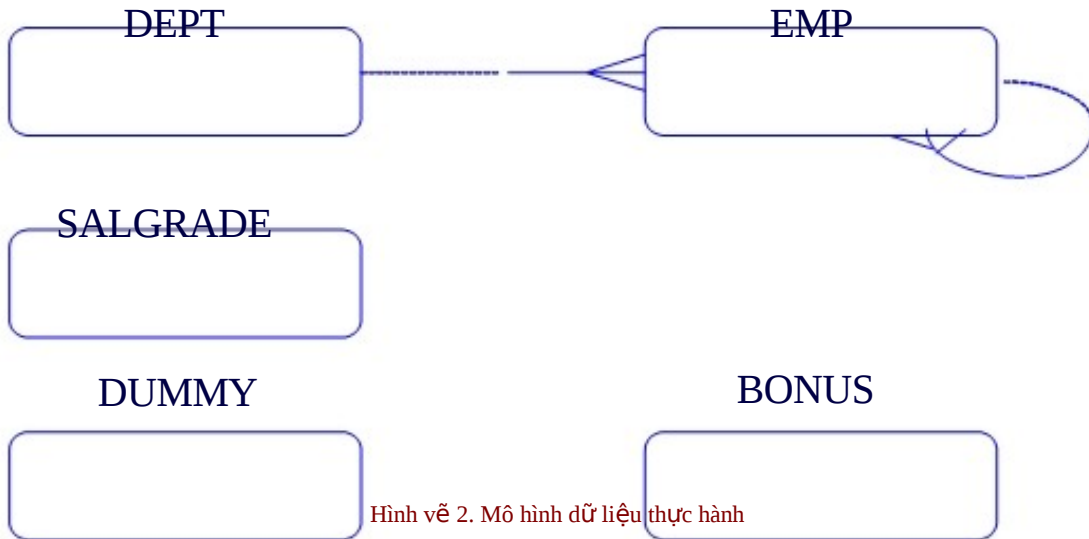
### 1.2.3. Các nhóm lệnh SQL cơ bản

Tên lệnh	Diễn giải
SELECT	Là lệnh thông dụng nhất, dùng để lấy, xem dữ liệu trong CSDL.
INSERT UPDATE DELETE	Là 3 lệnh dùng để nhập thêm những row mới, thay đổi nội dung dữ liệu trên các row hay xóa các row trong table. Những lệnh này được gọi là các lệnh thao tác dữ liệu DML (Data Manipulation Language)

CREATE ALTER DROP RENAME TRUNCATE	Là 3 lệnh dùng để thiết lập, thay đổi hay xóa bỏ cấu trúc dữ liệu như là table, view, index. Những lệnh này được gọi là các lệnh định nghĩa dữ liệu DDL (Data Definition Language)
COMMIT ROLLBACK SAVE POINT	Quản lý việc thay đổi dữ liệu bằng các lệnh DML. Việc thay đổi dữ liệu có thể được nhóm lại thành các transaction.
GRANT REVOKE	2 lệnh này dùng để gán hoặc huỷ các quyền truy nhập vào CSDL Oracle và các cấu trúc bên trong nó. Những lệnh này được gọi là các lệnh điều khiển dữ liệu DCL (Data Control Language)

### 1.3.CƠ SỞ DỮ LIỆU THỰC HÀNH

#### 1.3.1. Mô hình dữ liệu



#### 1.3.2. Cấu trúc bảng dữ liệu

Bảng DEPT

Tên cột	Kiểu	Điều kiện	Diễn giải
DEPTNO	NUMBER(2)	PRIMARY KEY	Mã phòng ban
DNAME	VARCHAR2(14)		Tên phòng ban
LOC	VARCHAR2(13)		Địa chỉ
Bảng SALGRADE			

Tên cột	Kiểu	Điều kiện	Diễn giải
GRADE	NUMBER	PRIMARY KEY	Mức lương
LOSAL	NUMBER		Giá trị thấp nhất
HISAL	NUMBER		Giá trị cao nhất



Bảng EMP

Tên cột	Kiểu	Điều kiện	Diễn giải
EMPNO	NUMBER(4)	PRIMARY KEY	Mã nhân viên
ENAME	VARCHAR2(10)		Tên nhân viên
JOB	VARCHAR2 (9)		Nghề nghiệp
MGR	NUMBER(4)	FOREIGN KEY (EMP.EMPNO)	Mã người quản lý
HIREDATE	DATE		Ngày gia nhập công ty
SAL	NUMBER(7,2)		Lương
COMM	NUMBER(7,2)		Thưởng
DEPTNO	NUMBER(2) NOT NULL,	FOREIGN KEY (DEPT.DEPTNO)	Mã phòng ban

## Chương 2. LỆNH TRUY VẤN CƠ BẢN

### 2.1. CÂU LỆNH TRUY VẤN

#### 2.1.1. Quy tắc viết lệnh

Các câu lệnh truy vấn được biểu diễn theo các quy tắc sau:

- ③ Các lệnh trong câu lệnh SQL thuộc loại không phân biệt chữ viết hoa hay thường.
- ③ Nội dung của một câu lệnh SQL có thể được trải dài trên nhiều dòng.
- ③ Các từ khoá không được phép viết tắt hay phân cách trên nhiều dòng
- ③ Các mệnh đề thông thường được đặt trên nhiều dòng khác nhau
- ③ Để rõ ràng trong việc thể hiện câu lệnh, ta nên sử dụng các dấu TAB khi viết lệnh
- ③ Ta có thể sử dụng các ký tự đặc biệt như: +, -, \, \*,... để biểu diễn giá trị trong câu lệnh.
- ③ Lệnh kết thúc bởi dấu chấm phẩy (;).

#### 2.1.2. Câu lệnh truy vấn cơ bản

Cú pháp:

```
SELECT          [DISTINCT ]      {*, column [alias],...}
FROM            table;
```

Với:

SELECT	Hiển thị nội dung của một hay nhiều cột
DISTINCT	Phân biệt nội dung giữa các dòng dữ liệu trả về
column	Lấy tất cả các cột trong bảng
alias	Tên cột dữ liệu cần trả về
FROM table	Phần tiêu đề của cột dữ liệu trả về
	Tên bảng chứa dữ liệu truy vấn

Ví dụ:

```
SELECT
FROM emp;
```

Cấu trúc của lệnh truy vấn gồm có hai phần:

- ③ Mệnh đề chọn lựa bao gồm Lệnh SELECT và tên cột dữ liệu trả về
- ③ Mệnh đề biểu diễn nơi chứa bao gồm FROM và tên bảng.

#### 2.1.3. Các thành phần khác của mệnh đề SELECT

Trong mệnh đề SELECT còn có thể đưa vào các thành phần khác:

- ③ Biểu thức toán học
- ③ Column alias
- ③ Các column được ghép chuỗi
- ③ Literal

Biểu thức toán học

Trong mệnh đề SELECT biểu thức toán học có thể các giá trị (column hoặc hàng số), các toán tử, các hàm. Các toán tử được dùng là (+), (-), (\*), (/). Độ ưu tiên của các toán tử giống trong phần số học.

Ví dụ:

```
SELECT ename, sal *12, comm FROM emp;  
SELECT ename, (sal+250)*12 FROM emp;
```

Tiêu đề của cột (column alias)

Trong mệnh đề SELECT, column alias là phần hiển thị của column khi lấy số liệu ra. Trong column alias không được có dấu cách và viết cách sau tên column một dấu cách. Column alias được chấp nhận có dấu cách khi được đặt trong dấu nháy kép (" ").

Ví dụ: (ANUAL chính là column alias)

```
SELECT ename, SAL*12 ANUAL, comm  
FROM emp;
```

Ghép tiếp các cột dữ liệu

Toán tử ghép tiếp chuỗi (||) cho phép ghép tiếp dữ liệu trong các cột khác nhau của cùng một dòng dữ liệu với nhau thành một chuỗi. Ta có thể có nhiều toán tử ghép chuỗi trong cùng một column alias.

Ví dụ:

```
SELECT empno||ename EMPLOYEE  
FROM emp;
```

Ghép tiếp chuỗi ký tự

Trong mệnh đề SELECT, ta có thể thực hiện ghép tiếp bất kỳ ký tự nào, biểu thức hay số nào mà không phải là column hoặc column alias.

Ví dụ:

```
SELECT empno || ename || ' WORK IN DEPARTMENT '  
|| deptno 'Employee Detail'  
FROM emp;
```

#### 2.1.4. Phân biệt giá trị dữ liệu trả về

Trong thực tế nhiều khi giá trị dữ liệu trên các dòng dữ liệu kết xuất trùng nhau. Gây nhiều bất tiện. Để có thể lấy được chỉ các dòng dữ liệu phân biệt với nhau. Ta sử dụng mệnh đề DISTINCT trong câu lệnh truy vấn.

Ví dụ:

```
SQL> SELECT deoptno FROM dept;  
DEPTNO  
  
10  
30  
10  
20  
  
14 rows selected.  
  
SQL> SELECT DISTINCT deoptno FROM dept;  
DEPTNO  
  
10  
30  
20  
3 rows selected.
```

### 2.1.5. Giá trị NULL

Cột có giá trị rỗng (NULL) là cột chưa được gán giá trị, nói cách khác nó chưa được khởi tạo giá trị. Các cột với bất cứ kiểu dữ liệu nào cũng có thể có trị NULL, trừ khi được nó là khóa hay có ràng buộc toàn vẹn NOT NULL. Trong biểu thức có bất kỳ giá trị NULL nào kết quả cũng là NULL.

Ví dụ:

```
SELECT ename, sal*12 + comm ANUAL_SAL
FROM emp;
```

NULL trong các hàm của SQL

Trong các hàm làm việc với từng cột hay hàm vô hướng (scalar function). Các hàm loại này trả về trị null khi có tham số NULL, trừ hàm NVL và TRANSLATE có thể trả về giá trị thực.

Cú pháp của hàm NVL:

```
NVL (DATECOLUMN, '01-01-2001')
NVL (NUMBERCOLUMN, 9)
NVL (CHARCOLUMN, 'STRING')
NVL(comm,0) trả về trị 0 khi comm là null

SELECT ename, sal*12 + NVL(comm,0) ANUAL_SAL FROM emp;
```

Trong các hàm làm việc với nhóm các cột (group function): Hầu hết các hàm làm việc trên nhóm bỏ qua trị null, ví dụ như khi sử dụng hàm AVG để tính trung bình cho một cột có các giá trị 1000, NULL, NULL, NULL, 2000. Khi đó trung bình được tính là  $(1000+2000)/2=1500$ , như vậy trị null bị bỏ qua chứ không phải xem là trị 0.

NULL trong các biểu thức so sánh, điều kiện

Để kiểm tra có phải NULL hay không dùng các toán tử IS NULL hoặc IS NOT NULL. Nếu trong biểu thức so sánh có trị null tham gia và kết quả của biểu thức phụ thuộc vào trị null thì kết quả là không xác định, tuy nhiên trong biểu thức DECODE, hai giá trị null được xem là bằng nhau trong phép so sánh.

Oracle xem các biểu thức với kết quả không xác định tương đương với FALSE. (Ví dụ: `comm = NULL`) có kết quả không xác định và do đó biểu thức so sánh xem như cho kết quả FALSE. Trong câu lệnh sau không có mẫu tin nào được chọn

```
SELECT * FROM emp WHERE comm=NULL;
```

Nếu muốn chọn các nhân viên có comm là NULL thì phải dùng toán tử IS NULL

```
SELECT * FROM emp WHERE comm IS NULL;
```

## 2.2. SQL\*PLUS, CÔNG CỤ TƯƠNG TÁC LỆNH SQL VỚI DATABASE

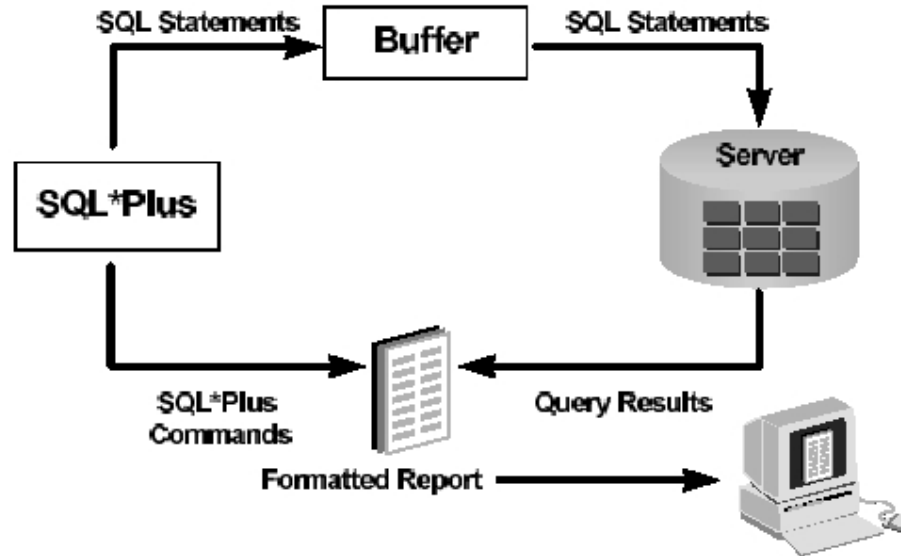
### 2.2.1. Câu lệnh tương tác của SQL\*Plus

Oracle hỗ trợ công cụ SQL\*Plus cho phép người sử dụng có thể tương tác trực tiếp với Oracle Server thông qua các câu lệnh SQL và PL/SQL.

Theo đó người sử dụng có thể tương tác với Oracle Server thông qua hai loại câu lệnh:

- ③ Câu lệnh SQL
- ③ Câu lệnh của bản thân chương trình SQL\*Plus

## SQL and SQL\*Plus Interaction



Hình vẽ 3. Câu lệnh của SQL\*Plus

Khác biệt giữa lệnh SQL và SQL\*Plus

SQL	SQL*Plus
Là ngôn ngữ để giao tiếp với Oracle Server trong việc truy xuất dữ liệu	Nhận dạng lệnh SQL và gửi lệnh lên Server
Câu lệnh dựa trên bộ ký tự chuẩn ASCII	Tùy thuộc vào từng phiên bản của Oracle Không
Thao tác trên các dữ liệu có trong các bảng đã được định nghĩa trong database	thao tác với dữ liệu trong database
Câu lệnh được nạp vào bộ nhớ đệm trên một hoặc nhiều dòng	Câu lệnh được tải trực tiếp không thông qua bộ đệm
Câu lệnh không được viết tắt	Câu lệnh có thể viết tắt
Có sử dụng ký tự kết thúc lệnh khi thực hiện Không	đòi hỏi phải có ký tự kết thúc lệnh
Sử dụng các hàm trong việc định dạng dữ liệu	Sử dụng các lệnh định dạng dữ liệu của chính SQL*Plus

### 2.2.2. Phân nhóm câu lệnh trong SQL\*Plus

Các lệnh SQL\*Plus có thể phân thành nhóm chính sau:

Nhóm lệnh	Diễn giải
Môi trường	Tác động và gây ảnh hưởng tới môi trường làm việc của SQL*Plus trong phiên làm việc hiện tại.
Định dạng dữ liệu	Định dạng lại dữ liệu trả về từ server
Thao tác file	Lưu giữ, nạp và chạy các file scrips
Thực hiện lệnh	Gửi các lệnh SQL có trong bộ đệm lên server
Soạn thảo	Sửa đổi lại lệnh SQL có trong bộ đệm

Tương tác	Cho phép người dùng có thể tạo các biến sử dụng trong câu lệnh SQL và thao tác với các biến đó như: nhập dữ liệu, kết xuất dữ liệu.
Các lệnh khác	Các lệnh khác cho phép kết nối tới cơ sở dữ liệu và hiển thị các cột dữ liệu theo như định dạng.

### 2.2.3. Chi tiết các lệnh SQL\*Plus cơ bản

Kết nối tới CSDL

Cú pháp:

Conn[ect] <user\_name>/<password>[@<database>];

Với:

user_name	Tên truy nhập
password	Mật khẩu truy nhập
database	Tên database truy nhập

Ví dụ:

Conn Tester/tester@DB1;

Hiển thị cấu trúc bảng dữ liệu

Cú pháp:

Desc[ribe] <table\_name>;

Với:

table_name	Tên bảng cần hiển thị cấu trúc
------------	--------------------------------

Ví dụ:

Desc Name	Dept;	Null?	Type
DEPTNO		NOT NULL	NUMBER(2)
DNAME			VARCHAR2(14)
LOC			VARCHAR2(13)

Lệnh soạn thảo

Tên lệnh	Diễn giải
A[PPEND] text	Đưa thêm đoạn text vào dòng hiện tại
C[HANGE] /old/new	Chuyển đoạn text cũ thành đoạn text mới trong dòng hiện tại
C[HANGE] /text/	Xoá đoạn text trong dòng hiện tại
CL[EAR] BUFF[ER]	Xoá tất cả các dòng trong SQL buffer
DEL	Xoá dòng hiện tại
DEL n	Xoá dòng n
DEL m n	Xoá dòng từ m đến n
I[NPUT]	Thêm một số dòng nhất định
I[NPUT] text	Thêm dòng có chứa text
L[IST]	Liệt kê toàn bộ các dòng trong SQL buffer

L[IST] n	Liệt kê dòng n
L[IST] m n	Liệt kê dòng m đến n
R[UN]	Hiển thị và chạy lệnh trong buffer
N	Nhảy đến dòng n
N text	Thay dòng n bởi đoạn text
Ø text	Chèn 1 dòng trước dòng 1

## Lệnh thao tác file

Tên lệnh	Diễn giải
SAVE filename [.ext] [REP[LACE] APP[END]]	Ghi nội dung bufer thành file. APPEND để ghi thêm vào file. REPLACE để chèn lên nội dung file cũ.
GET filename [.ext]	Ghi nội dung file vào buffer. Mặc định phần đuôi là .sql Chạy các
STA[RT] filename [.ext]	lệnh trong file
@ filename [.ext]	Giống lệnh Start
ED[IT]	Soạn thảo nội dung bufffer có tên là afiedt.buf Để chạy nội dung buffer dùng lệnh /
ED[IT]filename [.ext ]	Soạn thảo nội dung file
SPO[OL] filename [.ext ] [OFF OUT]	Cất kết quả hiển thị trên màn hình ra file. Vd: SPOOL result.sql  SPOOL OFF
EXIT	Thoát khỏi SQL*Plus

## Lệnh định dạng cột dữ liệu

Cú pháp:

```
COLUMN [{column | alias} [option]]
```

Tên lệnh	Diễn giải
CLE[AR]	Xoá định dạng của column
FOR[MAT] format	Chuyển định dạng của cột dữ liệu
HEA[DING] text	Đặt nhãn cho column
JUS[TIFY] align	Cán trái - left , phải - right, giữa - center cho nhãn
NOPRI[NT]	Ẩn column
NUL[L] text	Hiển thị text nếu giá trị của column là NULL
PRI[NT]	Hiển thị column
TRU[NCATED]	Xoá chuỗi tại cuối dòng đầu tiên khi hiển thị
WRA[PPED]	Phủ cuối chuỗi của dòng tiếp theo

Ví dụ 1: Chỉnh định dạng và nhãn của column

```
COLUMN ename HEADING 'Employee|Name' FORMAT A15
COLUMN sal JUSTIFY LEFT FORMAT $ 99,990.00 COLUMN
hiredate FORMAT A9 NULL ' Not hired'
```

Ví dụ 2: Hiển thị định dạng hiện tại của column

```
COLUMN
COLUMN ename
```

Ví dụ 3: Xoá định dạng hiện tại của column

```
COLUMN ename CLEAR
CLEAR COLUMN
```

Các loại định dạng

Định dạng	Diễn giải	Ví dụ	Kết quả
An	Hiển thị dài nhất n ký tự dùng cho các column dạng ký tự hoặc dạng ngày		
9	Hiển thị số, không bao gồm số 0	999999	1234
0	Hiển thị cả số 0	099999	01234
	\$Hiển thị \$	\$9999	\$1234
L	Hiển thị ký tự L	L9999	L1234
	Hiển thị dấu thập phân	9999.99	1234.00
,	Hiển thị dấu phân chia hàng nghìn	9,999	1,234

## 2.3.BÀI TẬP

1. Chọn toàn bộ thông tin trong bảng SALGRADE

```
GRADE      LOSAL      HISAL
-----
1           700       1200
2          1201       1400
3          1401       2000
4          2001       3000
5          3001       9999
```

2. Chọn toàn bộ thông tin trong bảng EMP

```
EMPNO ENAME      JOB          MGR HIREDATE      SAL COMM      DEPTNO
-----
7839 KING        PRESIDENT    17-11-1981  5000
7698 BLAKE      MANAGER      7839 01-05-1981  2850
7782 CLARK      MANAGER      7839 09-06-1981  2450
7566 JONES      MANAGER      7839 02-04-1981  2975
7654 MARTIN    SALESMAN     7698 28-09-1981  1250 1400
7499 ALLEN      SALESMAN     7698 20-02-1981  1600 300
7844 TURNER    SALESMAN     7698 08-09-1981  1500 0
7900 JAMES      CLERK        7698 03-12-1981  950
7521 WARD      SALESMAN     7698 22-02-1981  1250 500
7902 FORD      ANALYST      7566 03-12-1981  3000
7369 SMITH      CLERK        7902 17-12-1980  800
7788 SCOTT      ANALYST      7566 09-12-1982  3000
7876 ADAMS      CLERK        7788 12-01-1983  1100
7934 MILLER     CLERK        7782 23-01-1982  1300
```





## Chương 3. TRUY VẤN DỮ LIỆU CÓ ĐIỀU KIỆN

### 3.1. CÁC GIỚI HẠN TRONG TRUY VẤN DỮ LIỆU

Trong phần lớn các trường hợp lấy dữ liệu từ database, ta chỉ cần lấy một phần dữ liệu chứ không cần lấy tất cả. Để hạn chế các dữ liệu trả về không cần thiết, ta có thể sử dụng mệnh đề điều kiện trong câu lệnh truy vấn.

### Limiting Rows Using a Selection

**EMP**

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7698	BLAKE	MANAGER		30
7782	CLARK	MANAGER		10
7566	JONES	MANAGER		20
...				

"...retrieve all employees in department 10"

**EMP**

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7782	CLARK	MANAGER		10
7934	MILLER	CLERK		10

Hình vẽ 4. Hạn chế dữ liệu trả về

#### 3.1.1. Mệnh đề WHERE

Cú pháp:

```
SELECT [DISTINCT ] {*, column [alias],...}
FROM table
[WHERE condition (s)];
```

Với:

column	tên cột dữ liệu trả về
alias	tiêu đề của cột dữ liệu trả về
table	tên bảng truy vấn dữ liệu
condition	mệnh đề điều kiện để lọc dữ liệu trả về

Mệnh đề WHERE dùng để đặt điều kiện cho toàn bộ câu lệnh truy vấn. Trong mệnh đề WHERE có thể có các thành phần:

- ③ Tên column
- ③ Toán tử so sánh
- ③ Tên column, hằng số hoặc danh sách các giá trị

Ví dụ:

```
SELECT DEPTNO, JOB, ENAME, SAL
FROM EMP
WHERE SAL BETWEEN 1000 AND 2000 ;
```

Truy vấn dữ liệu với nhiều điều kiện

Mệnh đề WHERE cho phép ghép được nhiều điều kiện thông qua các toán tử logic AND/OR. Toán tử AND yêu cầu dữ liệu phải thỏa mãn cả 2 điều kiện. Toán tử OR cho phép dữ liệu thỏa mãn 1 trong 2 điều kiện.

Ví dụ:

```
SELECT DEPTNO, JOB, ENAME, SAL
FROM EMP
WHERE SAL BETWEEN 1000 AND 2000
AND JOB = 'MANAGER';
```

```
SELECT DEPTNO, JOB, ENAME, SAL
FROM EMP
WHERE SAL BETWEEN 1000 AND 2000
OR JOB = 'MANAGER';
```

```
SELECT DEPTNO, JOB, EMPNO, ENAME, SAL
FROM EMP
WHERE SAL > 1500
AND JOB = 'MANAGER'
OR JOB = 'SALESMAN';
```

```
SELECT DEPTNO, JOB, EMPNO, ENAME, SAL
FROM EMP
WHERE SAL > 1500
AND (JOB = 'MANAGER'
OR JOB = 'SALESMAN');
```

3.1.2. Các toán tử sử dụng trong mệnh đề WHERE Toán tử so sánh

Toán tử	Diễn giải
=	
!=, ^=, '+, <\>	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

Các toán tử của SQL

Toán tử	Diễn giải
[NOT] BETWEEN x AND y	[Không] lớn hơn hoặc bằng x và nhỏ hơn hoặc bằng y Thuộc bất kỳ giá
IN (danh sách):	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

## Các toán tử logic

Toán tử	Diễn giải
NOT	Phủ định mệnh đề
AND	Yêu cầu dữ liệu phải thỏa mãn cả 2 điều kiện
OR	Cho phép dữ liệu thỏa mãn 1 trong 2 điều kiện

Cấp độ ưu tiên khi thực hiện đối với các loại toán tử

Cấp độ ưu tiên	Toán tử
1	
2	Các toán tử so sánh
3	NOT
4	AND
	OR

## 3.1.3. Ví dụ sử dụng các toán tử điều kiện

[NOT] BETWEEN x AND y

Ví dụ chọn nhân viên có lương nằm trong khoảng 2000 và 3000

```
SELECT * FROM emp WHERE sal BETWEEN 2000 AND 3000;
```

IN (danh sách)

Chọn nhân viên có lương bằng một trong 2 giá trị 1400 hoặc 3000

```
SELECT * FROM emp WHERE sal IN (1400, 3000);
```

Tìm tên phòng ban nếu phòng đó có nhân viên làm việc.

```
SELECT dname FROM dept WHERE EXISTS
(SELECT * FROM emp WHERE dept.deptno = emp.deptno);
```

x [NOT] LIKE y

Tìm nhân viên có tên bắt đầu bằng chuỗi SMITH

```
SELECT * FROM emp WHERE
ename LIKE 'SMITH_';
```

Để chọn những nhân viên có tên bắt đầu bằng 'SM'

```
SELECT * FROM emp WHERE ename LIKE 'SM%';
```

Để tìm những nhân viên có tên có chuỗi 'A\_B'

```
SELECT ename FROM emp WHERE ename LIKE '%A\_B%'; ESCAPE '\'
```

Vì ký hiệu "\_" dùng để đại diện cho một ký tự bất kỳ nên nếu không có mệnh đề ESCAPE, câu lệnh trên sẽ tìm tất cả các nhân viên tên AAB, ABB, ACB, v.v...

Nếu muốn ký hiệu "\_" mang ý nghĩa nguyên thủy, tức là không còn đại diện cho ký tự bất kỳ nữa, ta đặt dấu "\" trước ký hiệu. Đồng thời khai báo thêm mệnh đề ESCAPE "\"

Ta cũng có thể dùng một ký tự bất kỳ thay cho "\". Chẳng hạn mệnh đề sau có cùng kết quả với mệnh đề trên

```
SELECT ename FROM emp WHERE ename LIKE '%A^_B%'; ESCAPE '^';
```

Ta gọi các ký tự như "\" hay "^" nói trên là các ký tự ESCAPE.

IS [NOT] NULL

Ví dụ:

```
SELECT * FROM emp WHERE comm IS NULL ;
```

## 3.2.SẮP XẾP DỮ LIỆU TRẢ VỀ

### 3.2.1. Mệnh đề ORDER BY

Cú pháp:

```
SELECT [DISTINCT ] {*, column [alias],...}
      FROM table;
      [WHERE condition]
      [ORDER BY expr/position [DESC/ASC]];
```

Mệnh đề ORDER BY dùng để sắp xếp số liệu được hiển thị và phải đặt ở vị trí sau cùng của câu lệnh truy vấn.

Ví dụ:

```
SELECT ENAME, JOB, SAL*12, DEPTNO
      FROM EMP
      ORDER BY ENAME;
```

Mệnh đề ORDER BY mặc định sắp xếp theo thứ tự tăng dần ASC[ENDING]

- ③ Số thấp trước
- ③ Ngày nhỏ trước
- ③ Ký tự theo bảng chữ cái

Để sắp xếp theo thứ tự ngược lại (giảm dần) đặt từ khoá DESC[ENDING] sau column cần sắp thứ tự.

Ví dụ:

```
SELECT ENAME, JOB, HIREDATE
      FROM EMP
      ORDER BY HIREDATE DESC ;
```

### 3.2.2. Sắp xếp nhiều cột dữ liệu trả về

Mệnh đề Order còn có thể sắp xếp nhiều column. Các column cần sắp xếp được viết thứ tự sau mệnh đề ORDER BY và cách bởi dấu phẩy (,). Column nào gần mệnh đề ORDER BY hơn có mức độ ưu tiên khi sắp xếp cao hơn. Chỉ định cách thức sắp xếp ASC/DESC được viết sau column cách bởi một dấu cách.

Ví dụ:

```
SELECT DEPTNO, JOB, ENAME, SAL
```

```
FROM EMP
ORDER BY DEPTNO, SAL DESC ;
```

Order giá trị NULL

Riêng đối với giá trị NULL, nếu sắp xếp theo thứ tự ASCENDING sẽ nằm ở các vị trí cuối cùng.

Chú ý: Có thể chỉ định sắp xếp theo thứ tự các column trong mệnh đề SELECT.

Ví dụ:

```
SELECT DEPTNO, JOB, ENAME, SAL
FROM EMP
ORDER BY 2;
```

### 3.3.BÀI TẬP

1. Chọn nhân viên trong bảng EMP có mức lương từ 1000 đến 2000 (chọn các trường ENAME, DEPTNO, SAL).

ENAME	DEPTNO	SAL
ALLEN	30	1600
WARD	30	1250
MARTIN	30	1250
TURNER	30	1500
ADAMS	20	1100
MILLER	10	1300

2. Hiển thị mã phòng ban, tên phòng ban, sắp xếp theo thứ tự tên phòng ban.

```
DEPTNO DNAME
```

```
10 ACCOUNTING
40 OPERATIONS
20 RESEARCH
30 SALES
```

3. Hiển thị danh sách những nhân viên làm tại phòng 10 và 20 theo thứ tự A,B,C

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7876	ADAMS	CLERK	7788	12-01-1983	1100		20
7782	CLARK	MANAGER	7839	09-06-1981	2450		10
7902	FORD	ANALYST	7566	03-12-1981	3000		20
7566	JONES	MANAGER	7839	02-04-1981	2975		20
7839	KING	PRESIDENT		17-11-1981	5000		10
7934	MILLER	CLERK	7782	23-01-1982	1300		10
7788	SCOTT	ANALYST	7566	09-12-1982	3000		20
7369	SMITH	CLERK	7902	17-12-1980	800		20

4. Hiển thị tên và nghề nghiệp những nhân viên làm nghề thư ký (clerk) tại phòng 20.

ENAME	JOB
SMITH	CLERK
ADAMS	CLERK

5. Hiển thị tất cả những nhân viên mà tên có các ký tự TH và LL.

ENAME

SMITH  
ALLEN  
MILLER

6. Hiển thị tên nhân viên, nghề nghiệp, lương của những nhân viên có giám đốc quản lý.

ENAME	JOB	SAL
SMITH	CLERK	800
ALLEN	SALESMAN	1600
WARD	SALESMAN	1250
JONES	MANAGER	2975
MARTIN	SALESMAN	1250
BLAKE	MANAGER	2850
CLARK	MANAGER	2450
SCOTT	ANALYST	3000
TURNER	SALESMAN	1500
ADAMS	CLERK	1100
JAMES	CLERK	950
FORD	ANALYST	3000
MILLER	CLERK	1300

13 rows selected.

7. Hiển thị tên nhân viên, mã phòng ban, ngày gia nhập công ty sao cho gia nhập công ty trong năm 1983.

ENAME	DEPTNO	HIREDATE
ADAMS	20	12-JAN-83

8. Hiển thị tên nhân viên, lương một năm (ANUAL\_SAL), thưởng sao cho lương lớn hơn thưởng và nghề nghiệp là SALEMAN, sắp theo thứ tự lương giảm dần và tên tăng dần.

ANUAL_SAL	COMM
19200	300
18000	0
15000	500

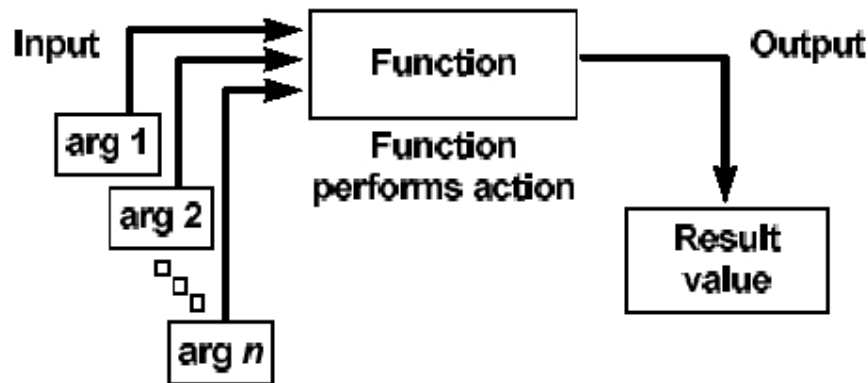
## Chương 4. CÁC HÀM SQL

### 4.1. TỔNG QUAN VỀ HÀM SQL

#### 4.1.1. Cấu trúc hàm SQL

Hàm SQL là một đặc điểm làm tăng khả năng sử dụng câu lệnh SQL. Hàm SQL có thể nhận nhiều tham số vào và trả về chỉ một giá trị.

## SQL Functions



Hình vẽ 5. Cấu trúc hàm SQL

Hàm SQL có một số đặc điểm sau:

- ③ Thực hiện việc tính toán ngay trên dữ liệu
- ③ Có thể thao tác, thay đổi ngay trên từng mục dữ liệu trả về
- ③ Hoặc cũng có thể thao tác trên nhóm các dữ liệu trả về
- ③ Có thể định dạng lại các dữ liệu trả về có kiểu số, hay kiểu thời gian
- ③ Có thể chuyển đổi kiểu dữ liệu trả về

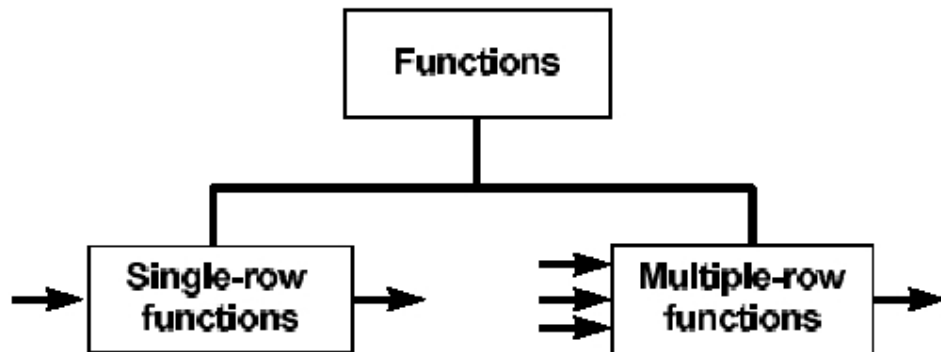
#### 4.1.2. Phân loại hàm SQL

Hàm SQL có thể phân ra làm hai loại:

- ③ Hàm tác động trên từng dòng dữ liệu: Giá trị trả về tương ứng với từng dữ liệu đầu vào tại mỗi dòng dữ liệu.
- ③ Hàm tác động trên nhóm các dòng dữ liệu: Giá trị trả về tương ứng với các phép thao tác trên nhóm dữ liệu trả về.



## Two Types of SQL Functions



Hình vẽ 6. Phân loại hàm SQL

### 4.2. HÀM SQL THAO TÁC TRÊN TỪNG DÒNG DỮ LIỆU

#### 4.2.1. Các hàm thao tác trên kiểu dữ liệu số

Đầu vào và đầu ra là các giá trị kiểu số

Một số hàm SQL hay dùng

Hàm SQL	Diễn giải
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 e <sup>n</sup>
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

Một số hàm kiểu số tham khảo khác

Hàm SQL	Diễn giải
LOG(m, n)	Cho logarit cơ số m của n
SIN(n)	Trả về cosin của n (n tính bằng radian)

COS(n)	Cho cosin của n (n tính bằng radian) Trả về
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
    
```

#### 4.2.2. Các hàm thao tác trên kiểu dữ liệu ký tự

Hàm SQL thao tác trên kiểu dữ liệu là ký tự

Hàm SQL	Diễn giải
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])	Cho chuỗi 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 tự 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	ACCOUNTING	ACCOUNTING	ACCOUNTING
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 SOUNDX(char)

```
SELECT ENAME, SOUNDX(ENAME)
FROM EMP
WHERE SOUNDX(ENAME)= SOUNDX('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
    
```

### 4.2.3. Các hàm thao tác trên kiểu dữ liệu thời gian Hàm SQL

thao tác trên kiểu dữ liệu là thời gian.

Hàm SQL	Diễn giải
MONTH_BETWEEN(d1, d2)	Cho biết số tháng giữa ngày d1 và d2. Cho ngày d thêm
ADD_MONTHS(d, n)	n tháng.
NEXT_DAY(d, char )	Cho ngày tiếp theo ngày d có thứ chỉ bởi char. Cho
LAST_DAY(d)	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

Hàm SQL	Diễn giải
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 tháng, 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 tháng, ngược lại sẽ trả về ngày đầu tiên của năm sau.
TRUNC(date1, 'MONTH')	Trả về ngày đầu tiên của tháng chứa date1 Trả về ngày đầu tiên của
TRUNC(date1, 'YEAR')	năm chứa date1

## 4.2.4. Các hàm chuyển đổi kiểu

Hàm SQL	Diễn giải
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('chr', 'fmt')	Chuyển ký tự sang kiểu ngày với định dạng đặt trong fmt.
DECODE(EXPR, SEARCH1, RESULT1, SEARCH2, RESULT2, DEFAULT):	Số 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. Trả giá trị
Greatest(col value1, col value2)	lớn nhất trong dãy giá trị.

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', 'WORKER', 'MANAGER', 'BOSS', 'UNDEFINED')
DECODED_JOB
FROM EMP;
```

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

Các khuôn dạng ngày	
Hàm SQL	Diễn giải
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, độ dài 9
MON	Tháng với 3 ký tự viết 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)
SSSS	Số giây đến nửa đêm (0-86399)
/ . , -	được tự động thêm khi đặt trong khuôn dạng
"char"	Đoạn ký tự đặt trong nháy kép được tự động thêm khi đặt trong khuôn dạng
TH	Thêm phần thứ tự (1st, 2nd, 4th )
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.

Năm		0-49	50-99
Năm hiện tại	0 - 49	thể kỷ hiện tại	Thế kỷ sau
	50 - 99	Thế kỷ trước	Thế kỷ hiện tại

## Một số khuôn dạng số

Ký tự	Diễn giải	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



## 4.3.HÀM THAO TÁC TRÊN TẬP HỢP

### 4.3.1. Các hàm tác động trên nhóm

Các hàm tác động trên nhóm các dòng dữ liệu hay tác động lên một tập hợp các các dòng dữ liệu bao gồm:

Hàm SQL	Diễn giải
AVG([DISTINCT/ALL] n)	Giá trị trung bình của n, không kể trị null Số
COUNT([DISTINCT/ALL] expr)	row có expr khác null
MAX([DISTINCT/ALL] expr)	Giá trị lớn nhất của expr
MIN([DISTINCT/ALL] expr)	Giá trị nhỏ nhất của expr
STDDVE([DISTINCT/ALL] n)	Phương sai của n không kể trị null Tổng của
SUM([DISTINCT/ALL] n)	của n không kể trị null Variance của n không
VARIANCE([DISTINCT/ALL] n)	kể trị null

Chú ý: Tất cả các hàm trên nhóm mẫu tin đều bỏ qua giá trị NULL trừ hàm COUNT. Dùng hàm NVL để chuyển đổi và tính giá trị NULL.

Có 2 cách để dùng các các hàm này

- ③ Tác động trên toàn bộ các dòng dữ liệu của câu lệnh truy vấn
- ③ Tác động trên một nhóm dữ liệu cùng tính chất của câu lệnh truy vấn. Cùng tính chất được chỉ bởi mệnh đề:
  - [GROUP BY expr]
  - [HAVING condition]

Ví dụ: Tác động trên toàn bộ các dòng dữ liệu của câu lệnh truy vấn:

- ③ Tính mức lương trung bình của toàn bộ nhân viên

```
Select AVG(SAL)
FROM EMP;
```

- ③ Tính mức lương thấp nhất của nhân viên làm nghề CLERK

```
Select MIN(SAL)
FROM EMP
WHERE JOB = 'CLERK';
```

Ví dụ: Tác động trên một nhóm dữ liệu cùng tính chất của câu lệnh truy vấn.

- ③ Tính mức lương trung bình của từng loại nghề nghiệp

```
SELECT JOB, AVG(SAL)
FROM EMP
GROUP BY JOB;
```

Chú ý: Chỉ được cùng đặt trong mệnh đề SELECT các hàm nhóm hoặc các column đã đặt trong mệnh đề GROUP BY.

Ví dụ:

```
Đúng:      SELECT MAX(SAL), JOB
           FROM EMP
           GROUP BY JOB;
```

Sai:               SELECT MAX(SAL), JOB  
                              FROM EMP;

#### 4.3.2. Mệnh đề GROUP BY

Cú pháp:

```
SELECT [DISTINCT ] {*, column [alias],...}
FROM table;
[WHERE condition]
[GROUP BY expr]
[HAVING condition]
[ORDER BY expr/position [DESC/ASC]]
```

Mệnh đề GROUP BY sẽ nhóm các dòng dữ liệu có cùng giá trị của expr. Ví dụ:

GROUP BY JOB nghĩa là sẽ nhóm các nghề giống nhau.

Mệnh đề HAVING là đặt điều kiện của nhóm dữ liệu. Mệnh đề này khác mệnh đề WHERE ở chỗ mệnh đề WHERE đặt điều kiện cho toàn bộ câu lệnh SELECT.

Ví dụ:

```
SELECT JOB, MAX(SAL)
FROM EMP
WHERE JOB != 'MANAGER'
GROUP BY JOB;
```

JOB	MAX(SAL)
ANALYST	3000
CLERK	1300
PRESIDENT	5000
SALESMAN	1600

```
SELECT JOB, MAX(SAL)
FROM EMP
GROUP BY JOB
HAVING COUNT(*)>3;
```

JOB	MAX(SAL)
CLERK	1300
SALESMAN	1600

```
SELECT JOB, MAX(SAL)
FROM EMP
HAVING MAX(SAL)>=3000
GROUP BY JOB;
```

JOB	MAX(SAL)
ANALYST	3000
PRESIDENT	5000

## 4.4.MỘT SỐ HÀM MỚI BỔ SUNG TRONG Oracle9i

### 4.4.1. Hàm NULLIF

Cú pháp:

```
NULLIF(expr1, expr2)
```

Hàm trả về giá trị NULL nếu biểu thức thứ nhất bằng biểu thức thứ 2. Trong trường hợp ngược lại, nó trả về giá trị của biểu thức thứ nhất.

### 4.4.2. Hàm COALSCE

Cú pháp:

```
COALESCE(expr1, expr2, expr3, ...)
```

Trả về giá trị của tham số đầu tiên khác null

### 4.4.3. Câu lệnh case

Ví dụ:

Case câu lệnh

```
SELECT ENAME, EXTRACT(YEAR FROM HIREDATE) AS YEAR_OF_HIRE,  
       (CASE EXTRACT(YEAR FROM HIREDATE)  
        WHEN 2002 THEN 'NEW HIRE'  
        WHEN 1997 THEN 'FIVE YEARS SERVICE'  
        WHEN 1992 THEN 'TEN YEARS SERVICE'  
        ELSE 'NO AWARD THIS YEAR'  
        END ) AS AWARD  
FROM EMP;
```

CASE biểu thức

```
SELECT ENAME, SAL,  
       (CASE  
        WHEN JOB = 'DBA' THEN SAL * 1.5  
        WHEN HIREDATE < SYSDATE - TO_YMINTERVAL('05-00') THEN SAL  
          * 1.25  
        WHEN DEPTNO IN (40,30,10) THEN SAL * 1.1  
        ELSE SAL * .9  
        END ) AS NEW_SAL  
FROM EMP;
```

## 4.5.BÀI TẬP

### 4.5.1. Hàm trên từng dòng dữ liệu

1. Liệt kê tên nhân viên, mã phòng ban và lương nhân viên được tăng 15% (PCTSAL).

DEPTNO	ENAME	PCTSAL
10	KING	5000
30	BLAKE	2850
10	CLARK	2450
20	JONES	2975

30	MARTIN	1250
30	ALLEN	1600
30	TURNER	1500
30	JAMES	950
30	WARD	1250
20	FORD	3000
20	SMITH	800
20	SCOTT	3000
20	ADAMS	1100
10	MILLER	1300

2. Viết câu lệnh hiển thị như sau:

EMPLOYEE\_AND\_JOB

KING***	*PRESIDENT
BLAKE***	**MANAGER
CLARK***	**MANAGER
JONES***	**MANAGER
MARTIN*****	*SALESMAN
ALLEN***	*SALESMAN
TURNER*****	*SALESMAN
JAMES***	*CLERK
WARD***	**SALESMAN
FORD***	***ANALYST
SMITH***	*CLERK
SCOTT***	**ANALYST
ADAMS***	*CLERK
MILLER***	***CLERK

3. Viết câu lệnh hiển thị như sau:

EMPLOYEE

KING	(President)
BLAKE	(Manager)
CLARK	(Manager)
JONES	(Manager)
MARTIN	(Salesman)
ALLEN	(Salesman)
TURNER	(Salesman)
JAMES	(Clerk)
WARD	(Salesman)
FORD	(Analyst)
SMITH	(Clerk)
SCOTT	(Analyst)
ADAMS	(Clerk)
MILLER	(Clerk)

4. Viết câu lệnh hiển thị như sau:

ENAME	DEPTNO	JOB
BLAKE	30	Manager
MARTIN	30	Salesperson
ALLEN	30	Salesperson
TURNER	30	Salesperson
JAMES	30	Clerk
WARD	30	Salesperson



5. Tìm ngày thứ 6 đầu tiên cách 2 tháng so với ngày hiện tại hiển thị ngày dưới dạng 09 February 1990.

6. Tìm thông tin về tên nhân viên, ngày gia nhập công ty của nhân viên phòng số 20, sao cho hiển thị như sau:

ENAME	DATE_HIRED
JONES	april,SECOND 1981
FORD	december,THIRD 1981
SMITH	december,SEVENTEENTH 1980
SCOTT	december,NINTH 1982
ADAMS	january,TWELFTH 1983

7. Hiển thị tên nhân viên, ngày gia nhập công ty, ngày xét nâng lương (sau ngày gia nhập công ty 1 năm), sắp xếp theo thứ tự ngày xét nâng lương.

ENAME	HIREDATE	REVIEW
SMITH	17-12-1980	17-12-1981
ALLEN	20-02-1981	20-02-1982
WARD	22-02-1981	22-02-1982
JONES	02-04-1981	02-04-1982
BLAKE	01-05-1981	01-05-1982
CLARK	09-06-1981	09-06-1982
TURNER	08-09-1981	08-09-1982
MARTIN	28-09-1981	28-09-1982
KING	17-11-1981	17-11-1982
JAMES	03-12-1981	03-12-1982
FORD	03-12-1981	03-12-1982
MILLER	23-01-1982	23-01-1983
SCOTT	09-12-1982	09-12-1983
ADAMS	12-01-1983	12-01-1984

8. Hiển thị tên nhân viên và lương dưới dạng

ENAME	SALARY
ADAMS	BELOW 1500
ALLEN	1600
BLAKE	2850
CLARK	2450
FORD	3000
JAMES	BELOW 1500
JONES	2975
KING	5000
MARTIN	BELOW 1500
MILLER	BELOW 1500
SCOTT	3000
SMITH	BELOW 1500
TURNER	On Target
WARD	BELOW 1500

9. Cho biết thứ của ngày hiện tại

10. Đưa chuỗi dưới dạng nn/nn, kiểm tra nếu không khuôn dạng trả lời là YES, ngược lại là NO. Kiểm tra với các chuỗi 12/34, 01/1a, 99\88

VALUE VALID?

12/34 YES

11. Hiển thị tên nhân viên, ngày gia nhập công ty, ngày lĩnh lương sao cho ngày lĩnh lương phải vào thứ 6, nhân viên chỉ được nhận lương sau ít nhất 15 ngày làm việc tại công ty, sắp xếp theo thứ tự ngày gia nhập công ty.

#### 4.5.2. Hàm trên nhóm dữ liệu

1. Tìm lương thấp nhất, lớn nhất và lương trung bình của tất cả các nhân viên
2. Tìm lương nhỏ nhất và lớn của mỗi loại nghề nghiệp
3. Tìm xem có bao nhiêu giám đốc trong danh sách nhân viên.
4. Tìm tất cả các phòng ban mà số nhân viên trong phòng >3
5. Tìm ra mức lương nhỏ nhất của mỗi nhân viên làm việc cho một giám đốc nào đó sắp xếp theo thứ tự tăng dần của mức lương.

## Chương 5. LỆNH TRUY VẤN DỮ LIỆU MỞ RỘNG

### 5.1.KẾT HỢP DỮ LIỆU TỪ NHIỀU BẢNG

#### 5.1.1. Mối liên kết tương đương

Mối liên kết tương đương được thể hiện trong mệnh đề WHERE.

Để liên kết trong mệnh đề WHERE phải chỉ rõ tên của các column và mệnh đề được đặt tương đương.

Ví dụ:

```
emp.deptno =dept.deptno
```

Các column trùng tên phải được chỉ rõ column đó nằm ở bảng nào thông qua tên hoặc qua alias. Tên trùng này có thể đặt trong các mệnh đề khác như SELECT, ORDER BY..

Ví dụ:

```
SELECT DEPT.DEPTNO, ENAME, JOB, DNAME
FROM EMP, DEPT
WHERE EMP.DEPTNO = DEPT.DEPTNO
ORDER BY DEPT.DEPTNO;
```

```
SELECT A.DEPTNO, A.ENAME, A.JOB, B.DNAME
FROM EMP A, DEPT B
WHERE A.DEPTNO = B.DEPTNO
ORDER BY A.DEPTNO;
```

#### 5.1.2. Mối liên kết không tương đương

Mối liên kết tương đương được thể hiện trong mệnh đề WHERE.

Để liên kết trong mệnh đề WHERE phải chỉ rõ tên của các column và mệnh đề được đặt không tương đương.

Ví dụ:

```
WHERE E.SAL BETWEEN S. LOSAL AND S.HISAL
```

Các column trùng tên phải được chỉ rõ column đó nằm ở bảng nào thông qua tên hoặc qua alias. Tên trùng này có thể đặt trong các mệnh đề khác như SELECT, ORDER BY..

Ví dụ:

```
SELECT E.ENAME, E.JOB, S.GRADE
FROM EMP E, SALGRADE S
WHERE E.SAL BETWEEN S. LOSAL AND S.HISAL;
```

Chú ý: Điều kiện liên kết đúng là số các bảng - 1 = số các điều kiện liên kết

#### 5.1.3. Mối liên kết cộng

Mối liên kết cộng trả về cả các giá trị NULL trong biểu thức điều kiện. Dấu (+) để ở vế nào tính thêm các giá trị NULL ở vế đó.

Một câu lệnh select chỉ đặt được 1 mối liên kết cộng, dấu (+) đặt ở bên phải column liên kết

Trong mệnh đề **WHERE** của mỗi liên kết cộng không được dùng toán tử **IN** hoặc **OR** để nối các điều kiện liên kết khác.

Ví dụ:

```
SELECT      E.ENAME, D.DEPTNO, D.DNAME
FROM        EMP E, DEPT D
WHERE       E.DEPTNO (+)=D.DEPTNO
AND         D.DEPTNO IN (30, 40);
```

ENAME	DEPTNO	DNAME
BLAKE	30	SALES
MARTIN	30	SALES
ALLEN	30	SALES
TURNER	30	SALES
JAMES	30	SALES
WARD	30	SALES
	40	OPERATIONS

#### 5.1.4. Liên kết của bảng với chính nó (tự thân)

Có thể liên kết bảng với chính nó bằng cách đặt alias.

Ví dụ:

```
Select e.ename emp_name, e.sal emp_sal,
       m.ename mgr_name, m.sal mgr_sal
from   emp e, emp m
where  e.mgr = m.empno
and e.sal <m.sal;
```

EMP_NAME	EMP_SAL	MGR_NAME	MGR_SAL
BLAKE	2850	KING	5000
CLARK	2450	KING	5000
JONES	2975	KING	5000
MARTIN	1250	BLAKE	2850
ALLEN	1600	BLAKE	2850
TURNER	1500	BLAKE	2850
JAMES	950	BLAKE	2850
WARD	1250	BLAKE	2850
SMITH	800	FORD	3000
ADAMS	1100	SCOTT	3000
MILLER	1300	CLARK	2450

#### 5.1.5. Cách biểu diễn kết nối mới trong Oracle 9i Tích đề-

các **CROSS JOIN** (Cartesian Product)

```
SELECT E.ENAME, D.DNAME
FROM EMP E CROSS JOIN DEPT D;
```

Kết nối tự nhiên **NATURAL JOIN** (Equijoin on All Identically Named Columns).

```
SELECT E.ENAME, D.DNAME
FROM EMP E NATURAL JOIN DEPT D;
```

Mệnh đề **USING** (Tương tự như Natural Join, nhưng cho phép chỉ rõ tên cột được sử dụng trong phép kết nối).

```
SELECT E.ENAME, D.DNAME
      FROM EMP E JOIN DEPT D USING (DEPTNO);
```

Mệnh đề ON (Chỉ rõ tên cột tham gia trong phép kết nối)

```
SELECT E.ENAME, D.DNAME
      FROM EMP E JOIN DEPT D ON (E.DEPTNO = D.DEPTNO);
```

Kết nối trái LEFT OUTER JOIN

```
SELECT E.ENAME, D.DNAME
      FROM EMP E LEFT OUTER JOIN DEPT D ON (E.DEPTNO = D.DEPTNO);
```

Kết nối phải RIGHT OUTER JOIN

```
SELECT E.ENAME, D.DNAME
      FROM EMP E RIGHT OUTER JOIN DEPT D ON (E.DEPTNO= D.DEPTNO);
```

Kết nối FULL OUTER JOIN (All records from both tables—Identical to a union of left outer join and right outer join)

```
SELECT E.ENAME, D.DNAME
      FROM EMP E FULL OUTER JOIN DEPT D ON (E.DEPTNO = D.DEPTNO);
```

### 5.1.6. Các toán tử tập hợp

Tên toán tử	Diễn giải
UNION	Kết hợp kết quả của nhiều câu hỏi với nhau, chỉ giữ lại một đại diện cho các mẫu tin trùng nhau.
UNION ALL	Kết hợp kết quả của nhiều câu hỏi với nhau, các mẫu tin trùng nhau cũng được lặp lại
INTERSET	Lấy phần giao các kết quả của nhiều câu hỏi
MINUS	Lấy kết quả có trong câu hỏi thứ nhất mà không có trong câu hỏi thứ hai (câu hỏi sau toán tử MINUS)

Ví dụ:

```
Select job from emp where deptno = 10
Union
Select job from emp where deptno = 30;

JOB

CLERK
MANAGER
PRESIDENT
SALESMAN
```

## 5.2.LỆNH TRUY VẤN LÔNG

### 5.2.1. Câu lệnh SELECT lồng nhau.

Trong mệnh đề WHERE

Tìm những nhân viên làm cùng nghề với BLAKE

```
select ename, job
from emp
where job = (select job from emp where ename = 'BLAKE');
```

ENAME	JOB
BLAKE	MANAGER
CLARK	MANAGER
JONES	MANAGER

Trong mệnh đề HAVING

Tìm những phòng có mức lương trung bình lớn hơn phòng 30

```
SELECT DEPTNO, AVG(SAL) FROM EMP
HAVING AVG(SAL) > (SELECT AVG(SAL) FROM EMP WHERE DEPTNO =30) GROUP
BY DEPTNO;
```

DEPTNO	AVG(SAL)
10	2916.66667
20	2175

### 5.2.2. Toán tử SOME/ANY/ALL/NOT IN/EXISTS

Tên toán tử	Diễn giải
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.

Ví dụ:

```
SELECT * FROM emp
WHERE sal = ANY (SELECT sal FROM emp WHERE deptno=30);
```

```
SELECT * FROM emp
WHERE sal >= ALL (select distinct sal
From emp
Where deptno =30)
Order by sal desc;
```

```
SELECT ENAME, SAL, JOB, DEPTNO
FROM EMP
WHERE SAL > SOME (SELECT DISTINCT SAL
FROM EMP
WHERE DEPTNO =30)
```

```
ORDER BY SAL DESC;
```

Tìm những người có nhân viên

```
SELECT EMPNO, ENAME, JOB, DEPTNO
FROM EMP E
WHERE EXISTS (SELECT EMPNO FROM EMP WHERE EMP.MGR = E.EMPNO);
```

## 5.3.CẤU TRÚC HÌNH CÂY

### 5.3.1. Cấu trúc hình cây trong 1 table

Trong một table của CSDL Oracle có thể hiện cấu trúc hình cây. Ví dụ trong bảng EMP cấu trúc thể hiện cấp độ quản lý.

- ③ Root node: là node cấp cao nhất
- ③ Child node: là node con hay không phải là root node
- ③ Parent node: là node có node con
- ③ Leaf node: là node không có node con

Level (cấp)

Level là một cột giả chứa cấp độ trong cấu trúc hình cây. Ví dụ.

```
SELECT LEVEL, DEPTNO, EMPNO, ENAME, JOB, SAL
FROM EMP
CONNECT BY PRIOR EMPNO = MGR
START WITH MGR is NULL;
```

LEVEL	DEPTNO	EMPNO	ENAME	JOB	SAL
1	10	7839	KING	PRESIDENT	5000
2	30	7698	BLAKE	MANAGER	2850
3	30	7654	MARTIN	SALESMAN	1250
3	30	7499	ALLEN	SALESMAN	1600
3	30	7844	TURNER	SALESMAN	1500
3	30	7900	JAMES	CLERK	950
3	30	7521	WARD	SALESMAN	1250
2	10	7782	CLARK	MANAGER	2450
3	10	7934	MILLER	CLERK	1300
2	20	7566	JONES	MANAGER	2975
3	20	7902	FORD	ANALYST	3000
4	20	7369	SMITH	CLERK	800
3	20	7788	SCOTT	SALEMAN	3300
4	20	7876	ADAMS	CLERK	1100

### 5.3.2. Kỹ thuật thực hiện

Có thể định nghĩa quan hệ thừa kế trong câu hỏi bằng mệnh đề STAR WITH và CONNECT BY trong câu lệnh SELECT, mỗi mẫu tin là một node trong cây phân cấp. Cột giả LEVEL cho biết cấp của mẫu tin hay cấp của node trong quan hệ thừa kế.

Cú pháp:

```
SELECT [DISTINCT/ALL] [expr [c_ias]]
FROM [table/view/snapshot] [t_alias]
[WHERE condition]
[START WITH condition CONNECT BY PRIOR condition]
[GROUP BY expr] [HAVING condition]
[UNION/UNION ALL/INTERSET/MINUS select command]
```

[ORDER BY expr/position [DESC/ASC]]

Với:

START WITH           Đặc tả điểm đầu của hình cây. Không thể để column giả level ở mệnh đề này.  
CONNECT BY           Chỉ column trong mối liên hệ tình cây.  
PRIOR                 Định hướng cấu trúc. Nếu prior xuất hiện trước mgr, Mgr sẽ được tìm trước sau đó đến empno, đây là hình cây hướng lên. Nếu prior xuất hiện trước empno, empno sẽ được tìm trước sau đó đến empno, đây là hình cây hướng xuống.

Ví dụ:

```
SELECT LEVEL, DEPTNO, EMPNO, ENAME, JOB, SAL
FROM EMP
CONNECT BY PRIOR MGR = EMPNO
START WITH empno = 7876;
```

LEVEL	DEPTNO	EMPNO	ENAME	JOB	SAL
1	20	7876	ADAMS	CLERK	1100
2	20	7788	SCOTT	SALEMAN	3300
3	20	7566	JONES	MANAGER	2975
4	10	7839	KING	PRESIDENT	5000

### 5.3.3. Mệnh đề WHERE trong cấu trúc hình cây

Mệnh đề WHERE và CONNECT BY có thể được dùng đồng thời trong cấu trúc hình cây. Nếu mệnh đề WHERE loại trừ một số row của cấu trúc hình cây thì chỉ những row đó được loại trừ. Nếu điều kiện đặt trong mệnh đề CONNECT BY thì toàn bộ nhánh của row đó bị loại trừ.

Ví dụ 1:

```
SELECT LEVEL, DEPTNO, EMPNO, ENAME, JOB, SAL
FROM EMP WHERE ENAME != 'SCOTT'
CONNECT BY PRIOR EMPNO = MGR
START WITH MGR IS NULL;
```

LEVEL	DEPTNO	EMPNO	ENAME	JOB	SAL
1	10	7839	KING	PRESIDENT	5000
2	30	7698	BLAKE	MANAGER	2850
3	30	7654	MARTIN	SALESMAN	1250
3	30	7499	ALLEN	SALESMAN	1600
3	30	7844	TURNER	SALESMAN	1500
3	30	7900	JAMES	CLERK	950
3	30	7521	WARD	SALESMAN	1250
2	10	7782	CLARK	MANAGER	2450
3	10	7934	MILLER	CLERK	1300
2	20	7566	JONES	MANAGER	2975
3	20	7902	FORD	ANALYST	3000
4	20	7369	SMITH	CLERK	800
4	20	7876	ADAMS	CLERK	1100

Ví dụ 2:

```
SELECT LEVEL, DEPTNO, EMPNO, ENAME, JOB, SAL
FROM EMP
CONNECT BY PRIOR EMPNO = MGR AND ENAME != 'SCOTT'
```



```
START WITH MGR IS NULL;
```

LEVEL	DEPTNO	EMPNO	ENAME	JOB	SAL
1	10	7839	KING	PRESIDENT	5000
2	30	7698	BLAKE	MANAGER	2850
3	30	7654	MARTIN	SALESMAN	1250
3	30	7499	ALLEN	SALESMAN	1600
3	30	7844	TURNER	SALESMAN	1500
3	30	7900	JAMES	CLERK	950
3	30	7521	WARD	SALESMAN	1250
2	10	7782	CLARK	MANAGER	2450
3	10	7934	MILLER	CLERK	1300
2	20	7566	JONES	MANAGER	2975
3	20	7902	FORD	ANALYST	3000
4	20	7369	SMITH	CLERK	800

## 5.4. BÀI TẬP

1. Hiển thị toàn bộ tên nhân viên và tên phòng ban làm việc sắp xếp theo tên phòng ban.

2. Hiển thị tên nhân viên, vị trí địa lý, tên phòng với điều kiện lương >1500.

ENAME	LOC	DNAME
KING	NEW YORK	ACCOUNTING
BLAKE	CHICAGO	SALES
CLARK	NEW YORK	ACCOUNTING
JONES	DALLAS	RESEARCH
ALLEN	CHICAGO	SALES
FORD	DALLAS	RESEARCH
SCOTT	DALLAS	RESEARCH

3. Hiển thị tên nhân viên, nghề nghiệp, lương và mức lương.

ENAME	JOB	SAL	GRADE
JAMES	CLERK	950	1
SMITH	CLERK	800	1
ADAMS	CLERK	1100	1
MARTIN	SALESMAN	1250	2
WARD	SALESMAN	1250	2
MILLER	CLERK	1300	2
ALLEN	SALESMAN	1600	3
TURNER	SALESMAN	1500	3
BLAKE	MANAGER	2850	4
CLARK	MANAGER	2450	4
JONES	MANAGER	2975	4
FORD	ANALYST	3000	4
SCOTT	ANALYST	3000	4
KING	PRESIDENT	5000	5

4. Hiển thị tên nhân viên, nghề nghiệp, lương và mức lương, với lương=3.

điều kiện mức

ENAME	JOB	SAL	GRADE
ALLEN	SALESMAN	1600	3
TURNER	SALESMAN	1500	3

## 5. Hiển thị những nhân viên tại DALLAS

ENAME	LOC	SAL
JONES	DALLAS	2975
FORD	DALLAS	3000
SMITH	DALLAS	800
SCOTT	DALLAS	3000
ADAMS	DALLAS	1100

## 6. Hiển thị tên nhân viên, nghề nghiệp, lương, mức lương, tên phòng làm việc trừ nhân viên có nghề là clerk và sắp xếp theo chiều giảm.

ENAME	JOB	SAL	GRADE	DNAME
MARTIN	SALESMAN	1250	2	SALES
WARD	SALESMAN	1250	2	SALES
ALLEN	SALESMAN	1600	3	SALES
TURNER	SALESMAN	1500	3	SALES
BLAKE	MANAGER	2850	4	SALES
CLARK	MANAGER	2450	4	ACCOUNTING
JONES	MANAGER	2975	4	RESEARCH
FORD	ANALYST	3000	4	RESEARCH
SCOTT	ANALYST	3000	4	RESEARCH
KING	PRESIDENT	5000	5	ACCOUNTING

## 7. Hiển thị chi tiết về những nhân viên kiếm được 36000 \$ 1 năm hoặc nghề là clerk. (gồm các trường tên, nghề, thu nhập, mã phòng, tên phòng, mức lương)

ENAME	JOB	ANUAL_SAL	DNAME	GRADE
JAMES	CLERK	11400	SALES	1
SMITH	CLERK	9600	RESEARCH	1
ADAMS	CLERK	13200	RESEARCH	1
MILLER	CLERK	15600	ACCOUNTING	2
FORD	ANALYST	36000	RESEARCH	4
SCOTT	ANALYST	36000	RESEARCH	4

## 8. Hiển thị những phòng không có nhân viên nào làm việc.

DEPTNO	DNAME	LOC
40	OPERATIONS	BOSTON

## 9. Hiển thị mã nhân viên, tên nhân viên, mã người quản lý, tên người quản lý

EMP_NAME	EMP_SAL	MGR_NAME	MGR_SAL
BLAKE	2850	KING	5000
CLARK	2450	KING	5000
JONES	2975	KING	5000
MARTIN	1250	BLAKE	2850
ALLEN	1600	BLAKE	2850
TURNER	1500	BLAKE	2850
JAMES	950	BLAKE	2850
WARD	1250	BLAKE	2850
FORD	3000	JONES	2975
SMITH	800	FORD	3000
SCOTT	3000	JONES	2975
ADAMS	1100	SCOTT	3000
MILLER	1300	CLARK	2450

10. Như câu 9 hiển thị thêm thông tin về ông KING.

EMP_NAME	EMP_SAL	MGR_NAME	MGR_SAL
KING	5000		
BLAKE	2850	KING	5000
CLARK	2450	KING	5000
JONES	2975	KING	5000
MARTIN	1250	BLAKE	2850
ALLEN	1600	BLAKE	2850
TURNER	1500	BLAKE	2850
JAMES	950	BLAKE	2850
WARD	1250	BLAKE	2850
FORD	3000	JONES	2975
SMITH	800	FORD	3000
SCOTT	3000	JONES	2975
ADAMS	1100	SCOTT	3000
MILLER	1300	CLARK	2450

11. Hiển thị nghề nghiệp được tuyển dụng vào năm 1981 và không được tuyển dụng vào năm 1994.

12. Tìm những nhân viên gia nhập công ty trước giám đốc của họ.

13. Tìm tất cả các nhân viên, ngày gia nhập công ty, tên nhân viên, tên người giám đốc và ngày gia nhập công ty của người giám đốc ấy.

EMP_NAME	EMP_SAL	MGR_NAME	MGR_SAL
BLAKE	2850	BLAKE	2850
MARTIN	1250	BLAKE	2850
ALLEN	1600	BLAKE	2850
TURNER	1500	BLAKE	2850
JAMES	950	BLAKE	2850
WARD	1250	BLAKE	2850
CLARK	2450	CLARK	2450
MILLER	1300	CLARK	2450
JONES	2975	JONES	2975
FORD	3000	JONES	2975
SMITH	800	JONES	2975
SCOTT	3300	JONES	2975
ADAMS	1100	JONES	2975

13 rows selected.

14. Tìm những nhân viên kiếm được lương cao nhất trong mỗi loại nghề nghiệp.

JOB	MAX(SAL)
ANALYST	3000
CLERK	1300
MANAGER	2975
PRESIDENT	5000
SALESMAN	1600

15. Tìm mức lương cao nhất trong mỗi phòng ban, sắp xếp theo thứ tự phòng ban.

ENAME	JOB	DEPTNO	SAL
KING	PRESIDENT	10	5000

SCOTT	SALEMAN	20	3300
BLAKE	MANAGER	30	2850

16. Tìm nhân viên gia nhập vào phòng ban sớm nhất

ENAME	HIREDATE	DEPTNO
CLARK	09-06-1981	10
SMITH	17-12-1980	20
ALLEN	20-02-1981	30

17. Hiển thị những nhân viên có mức lương lớn hơn lương TB của phòng ban mà họ làm việc.

EMPNO	ENAME	SAL	DEPTNO
7839	KING	5000	10
7566	JONES	2975	20
7902	FORD	3000	20
7788	SCOTT	3300	20
7698	BLAKE	2850	30
7499	ALLEN	1600	30

18. Hiển thị tên nhân viên, mã nhân viên, mã giám đốc, tên giám đốc, phòng ban làm việc của giám đốc, mức lương của giám đốc.

EMP_NUMBER	EMP_NAME	EMP_SAL	MGR_NUMBER	MGR_NAME	MGR_DEPT	MGR_GRADE
7698	BLAKE	2850	7698	BLAKE	30	4
7654	MARTIN	1250	7698	BLAKE	30	4
7499	ALLEN	1600	7698	BLAKE	30	4
7844	TURNER	1500	7698	BLAKE	30	4
7900	JAMES	950	7698	BLAKE	30	4
7521	WARD	1250	7698	BLAKE	30	4
7782	CLARK	2450	7782	CLARK	10	4
7934	MILLER	1300	7782	CLARK	10	4
7566	JONES	2975	7566	JONES	20	4
7902	FORD	3000	7566	JONES	20	4
7369	SMITH	800	7566	JONES	20	4
7788	SCOTT	3300	7566	JONES	20	4
7876	ADAMS	1100	7566	JONES	20	4

13 rows selected.

## Chương 6. BIẾN RUNTIME

### 6.1. DỮ LIỆU THAY THẾ TRONG CÂU LỆNH

Dùng (&) để chỉ phần thay thế trong câu lệnh.

Nếu dùng (&&) chỉ biến thay thế thì sau câu lệnh biến thay thế vẫn còn tồn tại Ví dụ 1:

```
SELECT * FROM emp
WHERE &Condition
```

```
Enter value for condition: sal > 1000
```

Khi này câu lệnh trên tương đương với:

```
SELECT * FROM emp
WHERE sal > 1000
```

Ví dụ 2:

```
Select ename, deptno, job
From emp
where deptno = &&deptno_please;
```

### 6.2. LỆNH DEFINE

Khai báo và gán trị cho các biến, ví dụ khai báo biến condition có giá trị 'sal > 1000'

```
DEFINE condition = 'sal > 1000'
```

Khi đó câu lệnh sau không yêu cầu nhập vào giá trị cho condition

```
SELECT * FROM emp
WHERE &Condition
```

Để loại bỏ biến ra khỏi bộ nhớ dùng lệnh UNDEFINE. Ví

dụ:

```
UNDEFINE condition
```

Để liệt kê các biến đã khai báo dùng lệnh DEFINE mà không chỉ biến, ví dụ

```
DEFINE
DEFINE CONDITION = 'SAL > 1000'
```

Ví dụ:

```
DEFINE REM='SAL*12+NVL(COMM,0)'
```

```
SELECT ENAME, JOB, &REM
FROM EMP ORDER BY & REM;
```

## 6.3.LỆNH ACCEPT

Khai báo và gán trị cho biến với dòng hiển thị

```
ACCEPT variable [NUMBER/CHAR] [PROMPT/NOPROMPT 'text'] HIDE
```

Ví dụ:

```
ACCEPT Salary NUMBER PROMPT 'Salary figure: '  
Salary figure : 3000
```

Từ khoá hide cho phép che chuỗi nhập liệu, hay dùng khi nhập password.

```
ACCEPT password CHAR PROMPT 'Enter password: ' HIDE  
Password : ***
```

## 6.4.BÀI TẬP

1. Hiển thị tên nhân viên, ngày gia nhập công ty với điều kiện ngày gia nhập công ty nằm trong khoảng hai biến runtime được nhập vào từ bàn phím (&first\_date, &last\_date).
2. Hiển thị tên nhân viên, nghề nghiệp, lương, mã giám đốc, mã phòng ban với điều kiện nghề nghiệp bằng một biến được nhập vào từ bàn phím. (&job)
3. Định nghĩa một biến tính thu nhập một năm của nhân viên. Dùng biến này để tìm những nhân viên có thu nhập lớn hơn hoặc bằng \$30000.
4. Định nghĩa một biến là khoảng thời gian nhân viên làm trong công ty. Hiển thị tên nhân viên và quãng thời gian nhân viên đó làm việc với điều kiện nhân viên là một biến được nhập vào từ bàn phím.

```
ENAME          LENGTH OF SERVICE  
  
KING           19 YEAR 4 MONTHS
```

## Chương 7. TABLE VÀ CÁC LỆNH SQL VỀ TABLE

### 7.1.LỆNH TẠO TABLE

#### 7.1.1. Cú pháp tạo bảng

Để tạo một bảng mới dùng lệnh CREATE TABLE. Cú

pháp:

```
CREATE TABLE tablename
  (column [datatype][DEFAULT expr][column_constraint]..)
  [PCTFREE integer][PCTUSED integer]
  [INITRANS integer][MAXTRANS integer]
  [TABLESPACE tablespace]
  [STORAGE storage_clause]
  [AS subquery]
```

Với:

tablename	Tên table cần tạo
column	Tên column trong table
[datatype]	Kiểu dữ liệu của column
[DEFAULT expr]	Giá trị mặc định của column trong trường hợp NULL là expr
[column_constraint]	Ràng buộc của bản thân column
[table_constraint]	Ràng buộc của toàn bảng
[PCTFREE integer]	Phần trăm không gian còn trống
[PCTUSED integer]	Phần trăm không gian đã sử dụng
[INITRANS integer]	Số bản ghi khởi tạo
[MAXTRANS integer]	Số bản ghi lớn nhất
[TABLESPACE tablespace]	Chỉ định TABLESPACE cho bảng
[STORAGE storage_clause]	Ghi mệnh đề lưu trữ, đơn vị mặc định là KB trong đó các các chọn lựa là: INITIAL - dung lượng khởi tạo; NEXT - dung lượng tăng tiếp theo; MINEXTENTS - % mở rộng nhỏ nhất; MAXEXTENTS- % mở rộng lớn nhất; PCTINCREASE - Tốc độ tăng hàng năm.
[AS subquery]	Tạo bảng có cấu trúc giống mệnh đề truy vấn

Ví dụ 1:

```
CREATE TABLE EMP
  EMPNO NUMBER NOT NULL CONSTRAINT PK_EMP PRIMARY KEY,
  ENAME VARCHAR2(10) CONSTRAINT NN_ENAME NOT NULL
  CONSTRAINT UPPER_ENAME CHECK (ENAME=UPPER(ENAME)),
  JOB VARCHAR2(9),
  MGR NUMBER CONSTRAINT FK_MGR REFERENCES
  SCOTT.EMP(EMPNO),
  HIREDATE DATE DEFAULT SYSDATE, SAL
  NUMBER(10,2) CONSTRAINT CK_SAL
  CHECK(SAL>500),
  COMM NUMBER(9,0) DEFAULT NULL,
  DEPTNO NUMBER(2) CONSTRAINT NN_DEPTNO NOT NULL
  CONSTRAINT FK_DEPTNO REFERENCES SCOTT.DEPT(DEPTNO))
  PCTFREE 5 PCTUSED 75
```

Ví dụ 2:

```
CREATE TABLE SALGRADE1
```

```
(GRADE NUMBER CONSTRAINT PK_SALGRADE PRIMARY KEY,  
LOSAL NUMBER,  
HISAL NUMBER)  
TABLESPACE USER  
STORAGE (INITIAL 6144 NEXT 6144  
MINEXTENTS 1 MAXEXTENTS 5 PCTINCREASE 5)
```

Ví dụ 3:

```
CREATE TABLE DEPT10  
AS  
SELECT EMPNO, ENAME, JOB, SAL  
FROM EMP WHERE DEPTNO =10;
```

Ví dụ 4:

```
CREATE TABLE EMP_SAL (NAME, SALARY, GRADE)AS  
SELECT ENAME, SAL, GRADE  
FROM EMP, SALGARDE  
WHERE EMP.SAL BETWEEN LOSAL AND HISAL ;
```

Để tạo một table mới, chúng ta cần phải chuẩn bị một số thông tin sau:

- ③ Table phải được chuẩn hóa.
- ③ Những column mà cho phép null nên định nghĩa sau để tiết kiệm nơi lưu trữ. ③ Gộp các table lại nếu có thể.
- ③ Chỉ định các thông số pcfree và pctused ③ Có thể chỉ định 2 thông số initstran, maxtrans ③ Có thể chỉ định tablespace cho table
- ③ Có thể ước lượng kích thước table, và các thông số cho storage.

### 7.1.2. Tính toán kích thước table (tham khảo)

1. Tính toán khoảng đĩa cần thiết cho data block header. Tính theo công thức sau:

$$\text{BLOCK HEADER} = (\text{FIXED HEADER} + \text{VARIABLE TRANSACTION HEADER}) + (\text{TABLE DIRECTORY} + \text{ROW DIRECTORY})$$

Trong đó:

```
fixed header = 57 bytes  
variable transaction header = 23*giá trị của thông số instrans  
table directory =4  
row directory = 2* số lượng row trong block.
```

2. Tính toán khoảng đĩa trống để chứa dữ liệu của data block. Tính theo công thức sau:  
Khoảng đĩa trống để chứa data =(block size -total block header) - (block size -(fixed header+ variable transaction header))\*(pctree/100)

Có thể biết block size bằng cách dùng lệnh

```
show parameters db_block_size.
```



3. Tính toán khoảng đĩa trống kết hợp bằng giá trị của mỗi row.
4. Tính toán kích thước trung bình của row:
  - Kích thước trung bình của row = row header +A+B+C
  - A = Tổng chiều dài của các cột <= 250 byte
  - B = Tổng chiều dài của các cột > 250 byte
  - C = Khoảng đĩa trống kết hợp
5. Quyết định số row trung bình cho một block:
  - avg rows /block = available space/average row size
6. Tính toán số lượng block
  - Block = số row / số row trung bình cho một block

## 7.2.MỘT SỐ QUY TẮC KHI TẠO TABLE

### 7.2.1. Quy tắc đặt tên Object

- ③ Tên dài từ 1 đến 30 ký tự, ngoại trừ tên CSDL không quá 8 ký tự và tên liên kết có thể dài đến 128 ký tự
- ③ Tên không chứa dấu nháy (")
- ③ Không phân biệt chữ hoa chữ thường
- ③ Tên phải bắt đầu bằng ký tự chữ trong bộ ký tự của CSDL
- ③ Tên chỉ có thể chứa ký tự số trong tập ký tự của CSDL. Có thể dùng các ký tự \_, \$, #. Oracle không khuyến khích dùng các ký tự \$ và #.
- ③ Tên không được trùng với các từ đã dùng bởi Oracle (xem phụ lục 1) ③ Tên không được cách khoảng trống
- ③ Tên có thể đặt trong cặp dấu nháy kép, khi đó tên có thể bao gồm các ký tự bất kỳ, có thể bao gồm khoảng trống, có thể dùng các từ khóa của Oracle, phân biệt chữ hoa chữ thường.
- ③ Tên phải duy nhất trong "không gian tên" nhất định. Các object thuộc cùng không gian tên phải có tên khác nhau.

Các bí danh của cột, bí danh bảng, tên người sử dụng, mật khẩu mặc dù không phải là các object hoặc các thành phần con của object nhưng cũng phải được đặt tên theo các quy tắc trên, ngoại trừ

Bí danh cột, bí danh bảng chỉ tồn tại khi thực hiện các lệnh SQL và không được lưu trữ trong CSDL, do vậy không áp dụng quy tắc 9 về không gian tên.

Mật khẩu không thuộc về không gian tên nào và do đó cũng không áp dụng quy tắc 9. Nên đặt tên theo một quy tắc đặt tên thống nhất

### 7.2.2. Quy tắc khi tham chiếu đến Object Cú pháp chung

khi tham chiếu đến các object

Sơ đồ chung khi tham chiếu các object hoặc thành phần của các object

Schema.Object.Part.@dblink

Trong đó:

object                      Tên object

schema	Schema chứa object
part	Thành phần của object
dblink	Tên CSDL chứa object

Oracle giải quyết việc tham chiếu các Object

Khi tham chiếu đến một object trong câu lệnh SQL, Oracle phân tích câu lệnh và xác định các object trong không gian tên. Sau khi xác định các object, Oracle thực hiện các thao tác mà câu lệnh quy định trên object. Nếu tên object truy cập không thuộc không gian tên thì câu lệnh không được thực hiện và có thông báo lỗi.

Câu lệnh sau thêm một mẫu tin vào bảng DEPT

```
INSERT INTO Dept VALUES (50, 'SUPPOR', 'PARIS')
```

Theo ngữ cảnh của câu lệnh, Oracle xác định bảng DEPT có thể là:

- ③ Một table trong schema của bạn
- ③ Một view trong schema của bạn
- ③ Đồng nghĩa riêng cho table hoặc view
- ③ Đồng nghĩa chung cho table hoặc view

Tham chiếu đến các object không thuộc quyền sở hữu

Để tham chiếu đến các object không thuộc schema hiện thời, phải chỉ ra tên của schema chứa object muốn truy cập

```
schema.object
```

Ví dụ: Để xóa table EMP trong schema SCOTT

```
DROP TABLE scott.emp
```

Tham chiếu các object từ xa

Để truy cập đến một CSDL ở xa, sau tên object phải chỉ ra tên liên kết CSDL (database link) của CSDL chứa object muốn truy cập. Database link là một schema object, Oracle dùng để thâm nhập và truy xuất CSDL từ xa.

## 7.3.Các Kiểu dữ liệu cơ bản

### 7.3.1. Kiểu 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

### 7.3.2. Kiểu 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.

### 7.3.3. Kiểu 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.

### 7.3.4. Kiểu NUMBER

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

NUMBER(p, s)  
 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	Giá trị 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

### 7.3.5. Kiểu 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

### 7.3.6. Kiểu 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

- ③ Một table không thể chứa nhiều hơn một cột kiểu LONG
- ③ Dữ liệu kiểu LONG không thể tham gia vào các ràng buộc toàn vẹn, ngoại trừ kiểm tra NULL và khác NULL
- ③ Không thể index một cột kiểu LONG
- ③ Không thể truyền tham số kiểu LONG cho hàm hoặc thủ tục ③ Các hàm không thể trả về dữ liệu kiểu LONG
- ③ 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

- ③ Các mệnh đề WHERE, GROUP BY, ORDER BY, CONNECT BY hoặc với tác tử DISTINCT trong câu lệnh SELECT
- ③ Các hàm sử dụng trong câu lệnh SQL như SUBSTR, INSTR
- ③ Trong danh sách lựa chọn của câu lệnh SELECT có sử dụng mệnh đề GROUP BY ③ Trong danh sách lựa chọn của câu hỏi con, câu hỏi có sử dụng các toán tử tập hợp ③ Trong danh sách lựa chọn của câu lệnh CREATE TABLE AS SELECT

### 7.3.7. Kiểu 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;
```

Kết quả:

```
JULIAN  
2448623
```

### 7.3.8. Kiểu RAW và kiểu 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.

### 7.3.9. Kiểu ROWID

Mỗi mẫu tin trong CSDL có một địa chỉ có kiểu ROWID. ROWID bao gồm các thành phần:

block.row.file.

Với

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.

### 7.3.10. Kiểu MLSLABEL

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

### 7.3.11. Chuyển đổi kiểu

Chuyển đổi mặc định

Nói chung một biểu thức không thể gồm các giá trị thuộc nhiều kiểu khác nhau, tuy nhiên Oracle cho phép chuyển đổi giữa các kiểu dữ liệu. Oracle tự động chuyển kiểu của dữ liệu trong một số trường hợp sau

- ③ Khi INSERT hoặc UPDATE gán giá trị cho cột có kiểu khác, Oracle sẽ tự động chuyển giá trị sang kiểu của cột.
- ③ Khi sử dụng các hàm hoặc các toán tử mà các tham số có kiểu không tương thích thì Oracle sẽ tự động chuyển kiểu.
- ③ Khi sử dụng toán tử so sánh mà các giá trị có các kiểu khác nhau, Oracle sẽ tự động chuyển kiểu.

Ví dụ 1:

```
SELECT ename FROM emp WHERE hiredate = '12-MAR-1993'
```

Oracle đã tự động chuyển chuỗi '12-MAR-1993' sang kiểu DATE trong phép so sánh

Ví dụ 2:

```
SELECT ename FROM emp WHERE ROWID = '00002514.0001.0001'
```

Oracle đã tự động chuyển chuỗi '00002514.0001.0001' sang kiểu ROWID trong phép so sánh

Người sử dụng tự chuyển đổi

Oracle cung cấp các hàm để chuyển đổi kiểu, ví dụ

- ③ TO\_NUMBER                      Chuyển sang kiểu số
- ③ TO\_CHAR                        Chuyển sang kiểu ký tự
- ③ TO\_DATE                        Chuyển sang kiểu ngày

(xem phần tra cứu các hàm và thủ tục)

## 7.4. RÀNG BUỘC DỮ LIỆU TRONG TABLE

Các dạng constraint gồm:

- ③ NULL/NOT NULL
- ③ UNIQUE
- ③ PRIMARY KEY
- ③ FOREIGN KEY (Referential Key)
- ③ CHECK

### 7.4.1. NULL/NOT NULL

Là ràng buộc column trống hoặc không trống.

Ví dụ mệnh đề ràng buộc:

```
CREATE TABLE DEPT (  
    DEPTNO                      NUMBER(2) NOT NULL,  
    DNAME                        CHAR(14),  
    LOC                         CHAR(13),  
    CONSTRAINT DEPT_PRIMARY_KEY PRIMARY KEY (DEPTNO));
```

### 7.4.2. UNIQUE

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

Ví dụ:

```
CREATE TABLE DEPT (  
    DEPTNO                      NUMBER(2),  
    DNAME                        CHAR(14),  
    LOC                         CHAR(13),  
    CONSTRAINT UNQ_DEPT_LOC UNIQUE(DNAME, LOC));
```

### 7.4.3. PRIMARY KEY

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 khi đặt CONSTRAINT ở mức TABLE  
[CONSTRAINT constraint\_name] PRIMARY KEY (column,  
Column..)

Cú pháp khi đặt CONSTRAINT ở mức COLUMN  
[CONSTRAINT constraint\_name] PRIMARY KEY

#### 7.4.4. FOREIGN KEY ( Referential )

Chỉ ra mối liên hệ ràng buộc tham chiếu giữa table này với table khác, hoặc trong chính 1 table. Nó chỉ ra mối liên hệ cha-con và chỉ ràng buộc giữa FOREIGN KEY bảng này với PRIMARY KEY hoặc UNIQUE Key của bảng khác. Ví dụ quan hệ giữa DEPT và EMP thông qua trường DEPTNO.

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

#### 7.4.5. CHECK

Ràng buộc kiểm tra giá trị.

Ví dụ:

```
CREATE TABLE EMP
(EMPNO NUMBER NOT NULL CONSTRAINT PK_EMP PRIMARY KEY,
 ENAME VARCHAR2(10) CONSTRAINT NN_ENAME NOT NULL
 CONSTRAINT UPPER_ENAME CHECK (ENAME =
UPPER(ENAME)),
 JOB VARCHAR2(9),
 MGR NUMBER CONSTRAINT FK_MGR REFERENCES
      SCOTT.EMP(EMPNO),
 HIREDATE DATE DEFAULT SYSDATE, SAL
 NUMBER(10,2) CONSTRAINT CK_SAL
      CHECK(SAL>500),
 COMM NUMBER(9,0) DEFAULT NULL,
 DEPTNO NUMBER(2) CONSTRAINT NN_DEPTNO NOT NULL
      CONSTRAINT FK_DEPTNO REFERENCES SCOTT.DEPT(DEPTNO);
```

## 7.5.LỆNH DDL CAN THIỆP TỚI TABLE

### 7.5.1. Chính sửa cấu trúc table

Dùng lệnh ALTER TABLE để chỉnh sửa cấu trúc bảng. Cú pháp:

```
ALTER TABLE tablename [ADD/MODIFY/DROP options
 ([column [column constraint) [ENABLE clause]
 [DISABLE clause]
```

Trong đó:

ADD	Thêm column hay constraint.
MODIFY	Sửa đổi kiểu các column
DROP	Bỏ constraint.
ENABLE/DISABLE	Che khuất hoặc đưa vào sử dụng các CONSTRAINT mà không xóa hẳn

Chú ý:

- ③ Khi dùng mệnh đề MODIFY không thể chuyển tính chất của COLUMN có nội dung là NULL chuyển thành NOT NULL;
- ③ Không thể đưa thêm một cột NOT NULL nếu table đã có số liệu. Phải thêm cột NULL, điền đầy số liệu, sau đó chuyển thành NOT NULL.
- ③ Không thể chuyển đổi kiểu khác nhau nếu column đã chứa số liệu
- ③ Không thể dùng mệnh đề MODIFY để định nghĩa các CONSTRAINT trừ ràng buộc NULL/NOT NULL. Muốn sửa CONSTRAINT cần xoá chúng sau đó ADD thêm vào.

Ví dụ 1:

```
ALTER TABLE emp ADD (spouse_name CHAR(10));
```

Ví dụ 2:

```
ALTER TABLE emp MODIFY (ename CHAR(25));
```

Ví dụ 3:

```
ALTER TABLE emp DROP CONSTRAINT emp_mgr;  
ALTER TABLE emp DROP PRIMARY KEY;
```

Ví dụ 4:

```
ALTER TABLE dept DISABLE CONSTRAINT dept_prim;
```

### 7.5.2. Các lệnh DDL khác

Xoá table

Dùng lệnh DROP TABLE để xoá bảng.

Cú pháp:

```
DROP TABLE table_name [CASCADE CONSTRAINTS]
```

Trong đó:

CASCADE CONSTRAINTS	xoá tất cả các ràng buộc toàn vẹn liên quan đến table bị xoá.
---------------------	---

Ví dụ:

```
DROP TABLE emp
```

Khi drop table thì:

- ③ Xoá tất cả dữ liệu
- ③ View và synonym liên quan vẫn còn nhưng không có giá trị
- ③ Các giao dịch chưa giải quyết xong sẽ được commit
- ③ Chỉ người tạo ra table hay DBA mới có thể xoá table

### 7.5.3. Chú dẫn cho table

Dùng lệnh COMMENT để chú thích.

Ví dụ:



```
COMMENT ON TABLE EMP IS ' THONG TIN NHAN VIEN'; COMMENT  
ON COLUMN EMP.EMPNO IS ' MA SO NHAN VIEN';
```

#### 7.5.4. Thay đổi tên object

Dùng lệnh RENAME để thay đổi tên object.

Cú pháp:

```
RENAME old TO new
```

Trong đó:

old	Tên cũ
new	Tên mới

Ví dụ:

```
RENAME emp TO employee
```

#### 7.5.5. Xóa dữ liệu của table

Dùng lệnh TRUNCATE TABLE để xóa dữ liệu của table, xóa tất cả các row trong table. Cú pháp:

```
TRUNCATE TABLE table_name [REUSE STORAGE]
```

Trong đó:

REUSE STORAGE giữ lại khung để chứa, chỉ xóa dữ liệu

## 7.6. THÔNG TIN VỀ TABLE TRONG TỪ ĐIỂN DỮ LIỆU

Trung tâm của cơ sở dữ liệu Oracle là data dictionary. Data dictionary tự động được tạo ra khi cơ sở dữ liệu Oracle được tạo. Oracle cập nhật lên data dictionary bằng các lệnh DDL (Data Define Language). Các table của từ điển dữ liệu được tạo ra bằng lệnh CREATE DATABASE và chỉ được tạo từ user SYS. Các view trong từ điển dữ liệu chức các thông tin dưới dạng dễ nhìn hơn bằng.

Có các dạng view là:

- ③ USER\_XXX: là những đối tượng thuộc user. Ví dụ các bảng được tạo bởi user
- ③ ALL\_XXX: là tất cả các đối tượng mà user có quyền truy nhập
- ③ DBA\_XXX: tất cả các đối tượng trong database
- ③ V\$: Các thực thi của Server.

Ngoài ra còn có các view quan trọng khác là:

- ③ DICTIONARY: Thông tin về toàn bộ các table, view, snapshot trong từ điển dữ liệu
- ③ TABLE\_PRIVILEGES: Thông tin về việc gán quyền trên các đối tượng
- ③ IND: đồng nghĩa của USER\_INDEX.

Muốn hiển thị toàn bộ thông tin về các table, view, snapshot trong từ điển dữ liệu dùng lệnh

```
SELECT * FROM DICTIONARY;
```

Hiển thị cấu của USER\_OBJECT

```
DESCRIBE USER_OBJECT;
```

Hiển thị tất cả các bảng mà user đó sở hữu:

```
SELECT OBJECT_NAME
       FROM USER_OBJECT
       WHERE OBJECT_TYPE = 'TABLE';

SELECT * FROM TAB;

SELECT TABLE_NAME FROM USER_TABLE;
```

Hiển thị tất cả các loại đối tượng trong từ điển dữ liệu:

```
SELECT DISTINCT OBJECT_TYPE
       FROM USER_OBJECTS;
```

## 7.7. BÀI TẬP

1. Tạo bảng PROJECT với các column được chỉ ra dưới đây, PROJID là primary key, và P\_END\_DATE > P\_START\_DATE.

Column name	Data Type	Size.
PROJID	NUMBER	4
P_DESC	VARCHAR2	20
P_START_DATE	DATE	
P_END_DATE	DATE	
BUDGET_AMOUNT	NUMBER	7, 2
MAX_NO_STAFF	NUMBER	2

2. Tạo bảng ASSIGNMENTS với các column được chỉ ra dưới đây, đồng thời cột PROJID là foreign key tới bảng PROJECT, cột EMPNO là foreign key tới bảng EMP.

Column name	Data Type	Size.	
PROJID	NUMBER	4	NOT NULL
EMPNO	NUMBER	4	NOT NULL
A_START_DATE	DATE		
A_END_DATE	DATE		
BILL_AMOUNT	NUMBER	4, 2	
ASSIGN_TYPE	VARCHAR2	2	

3. Thêm column COMMENTS kiểu LONG vào bảng PROJECTS. Thêm column HOURS kiểu NUMBER vào bảng ASSIGNMENTS.
4. Sử dụng view USER\_OBJECTS hiển thị tất cả các đối tượng user sở hữu.
5. Thêm ràng buộc duy nhất (UNIQUE) cho 2 column PROJECT\_ID và EMPNO của bảng ASSIGNMENTS.
6. Xem các thông tin về các ràng buộc trong USER\_CONSTRAINTS.
7. Xem trong USER hiện tại có tất cả bao nhiêu bảng.

## Chương 8. CÁC LỆNH THAO TÁC DỮ LIỆU

### 8.1. THAO TÁC DỮ LIỆU TRONG TABLE

#### 8.1.1. Thêm mới dòng dữ liệu

Để chèn một row vào table dùng lệnh INSERT. Cú

pháp:

```
INSERT INTO tablename ([column, column, ...])
VALUES (value, value ...);
```

Ví dụ:

```
INSERT INTO dept (deptno, dname, loc)
VALUES (50, 'MARKETING', 'SAN JOSE')
```

Chép dữ liệu từ table khác

```
INSERT INTO table [(column, column...)]
SELECT select_list
FROM table(s)
```

Ví dụ:

```
INSERT INTO emp_tmp (ename, sal)
SELECT ename, sal FROM emp WHERE sal > 1000
```

Bắt đầu từ phiên bản Oracle 9i, ta có thể thêm mới dòng dữ liệu và đặt giá trị mặc định thông qua từ khoá DEFAULT

Ví dụ:

```
INSERT INTO EMP (EMPNO, ENAME, DEPTNO)
VALUES (8000, 'MIKE', DEFAULT);
```

Oracle 9i còn cho phép thực hiện lệnh INSERT trên đồng thời nhiều table khác nhau, chỉ sử dụng một câu lệnh DML.

Ví dụ:

Lệnh INSERT không điều kiện (UNCONDITIONAL)

```
INSERT ALL
  INTO T1 (C1, C2, ...) VALUES (C1, C2, ...)
  INTO T2 (C1, C2, ...) VALUES (C1, C2, ...)
```

```
SELECT C1, C2, ... FROM T9;
```

Lệnh INSERT không điều kiện (CONDITIONAL)

```
INSERT [ALL|FIRST]
  WHEN c1 = 1 THEN INTO T1 (C1, C2, ...) VALUES (C1, C2, ...)
  WHEN c1 = 2 THEN INTO T2 (C1, C2, ...) VALUES (C1, C2, ...)
  WHEN c2 = 3 THEN INTO T3 (C1, C2, ...) VALUES (C1, C2, ...)
```

```
SELECT C1, C2, ... FROM T9;
```

FIRST: insert cho câu lệnh đầu tiên có giá trị điều kiện đúng  
ALL: insert cho mọi câu lệnh có giá trị điều kiện là đúng

### 8.1.2. Cập nhật dòng dữ liệu

Để chỉnh sửa dữ liệu dùng lệnh UPDATE.

Cú pháp:

```
UPDATE table [alias]
  SET column [,column...] = [expr, subquery]
  [WHERE condition]
```

Ví dụ 1:

```
UPDATE emp
  SET job = 'SALEMAN', hiredate = sysdate, sal = sal * 1.1
  WHERE ename = 'SCOTT';
```

Ví dụ 2:

```
UPDATE emp
  SET comm = (SELECT comm FROM commission C
              WHERE C.empno = emp.empno)
  WHERE empno IN (SELECT empno FROM commission);
```

Ví dụ 3:

```
UPDATE emp a
  SET deptno =
    (SELECT deptno FROM dept
     WHERE loc = 'BOSTON'),
    (sal, comm) = (SELECT
    1.1*AVG(sal), 1.5*AVG(comm)
     FROM emp b
     WHERE a.deptno = b.deptno)
  WHERE deptno IN
    (SELECT deptno FROM dept
     WHERE loc = 'DALLAS' OR loc = 'DETROIT');
```

Ta cũng có thể sử dụng mệnh đề DEFAULT trong câu lệnh cập nhật dữ liệu Ví dụ:

```
UPDATE EMP SET COMM = DEFAULT;
```

Chú thích:

- Cập nhật các nhân viên ở Dallas hoặc Detroit
- Thay DEPTNO của các nhân viên này bằng DEPTNO của Boston - Thay lương mỗi nhân viên bằng lương trung bình của bộ phận \* 1.1 - Thay commission của mỗi nhân viên bằng commission trung bình của bộ phận \* 1.5

### 8.1.3. Lệnh Merge

Lệnh MERGE là một đặc điểm rất hay của Oracle 9i. Nó còn được gọi là lệnh UPSERT, tức là có khả năng vừa thực hiện việc Update, vừa thực hiện lệnh Insert tùy vào bản ghi đích có tồn tại hay không.

Cú pháp:

```
MERGE INTO T1
```

```
USING T2 ON (T1.C9=T2.C9)
WHEN MATCHED THEN UPDATE SET T1.C1=T2.C2, T1.C2=T2.C2 ...
WHEN NOT MATCHED THEN INSERT (C1,C2, ...) VALUES (C1,C2,
...);
```

#### 8.1.4. Xóa dòng dữ liệu

Để xóa dòng dùng lệnh DELETE.

Cú pháp:

```
DELETE FROM table [WHERE condition]
```

Ví dụ:

```
DELETE FROM emp
WHERE deptno = 10;
```

#### 8.1.5. Lỗi ràng buộc dữ liệu

Thông thường khi thực hiện các lệnh thao tác dữ liệu hay gặp phải các lỗi ràng buộc toàn vẹn dữ liệu. Các lỗi này xuất hiện khi có các ràng buộc trước đó mà dữ liệu nhập vào, chỉnh sửa hay khi xoá đi không đảm bảo các điều kiện toàn vẹn. Mã lỗi: ORA\_02292: INTEGRITY CONSTRAINT. Sau đó báo tên của Constraint bị lỗi.

## 8.2. LỆNH ĐIỀU KHIỂN GIAO DỊCH

Một câu lệnh SQL có thể gồm

- ③ Lệnh DML thao tác dữ liệu
- ③ Lệnh DDL định nghĩa dữ liệu
- ③ Lệnh DCL điều khiển truy nhập dữ liệu

Một giao dịch bắt đầu khi một lệnh SQL được thực hiện Một giao dịch kết thúc một trong các trường hợp sau:

- ③ COMMIT hoặc ROLLBACK
- ③ Các lệnh DDL và DCL thực hiện (tự động commit)
- ③ Lỗi, thoát khỏi SQL\*Plus, hệ thống bị down.

Cú pháp:

Kết thúc giao dịch hiện tại, thực hiện các chuyển đổi dữ liệu

```
COMMIT
```

Xác định điểm savepoint của giao dịch

```
SAVEPOINT name
```

Quay lại dữ liệu ở điểm SAVEPOINT hoặc toàn bộ giao dịch.

```
ROLLBACK [TO SAVEPOINT name]
```

Tự động COMMIT khi thực hiện các lệnh Insert, update, delete.

```
SET AUTO[COMMIT] ON/OFF
```

Ví dụ:

```
INSERT INTO DEPT
VALUES (50, 'TESTING', 'LAS VEGAS');

SAVEPOINT INSERT_DONE;

UPDATE DEPT
SET DNAME = 'MARKETING';

ROLLBACK TO INSERT_DONE ;

UPDATE DEPT SET DNAME = 'MARKETING'
WHERE DNAME = 'SALES';

COMMIT;
```

## 8.3.BÀI TẬP

1. Thêm dữ liệu vào bảng PROJECTS.

PROJID	1	2
P_DESC	WRITE C030 COURSE	PROOF READ NOTES
P_START_DATE	02-JAN-88	01-JAN-89
P_END_DATE	07-JAN-88	10-JAN-89
BUDGET_AMOUNT	500	600
MAX_NO_STAFF	1	1

2. Thêm dữ liệu vào bảng ASSIGNMENTS.

PROJID	1	1	2
EMPNO	7369	7902	7844
A_START_DATE	01-JAN-88	04-JAN-88	01-JAN-89
A_END_DATE	03-JAN-88	07-JAN-88	10-JAN-89
BILL_RATE	50.00	55.00	45.50
ASSIGN_TYPE	WR	WR	PF
HOURS	15	20	30

3. Cập nhật trường ASIGNMENT\_TYPE từ WT thành WR.

4. Nhập thêm số liệu vào bảng ASSIGNMENTS.

## Chương 9. SEQUENCE VÀ INDEX

### 9.1. SEQUENCE

#### 9.1.1. Tạo Sequence

Sequence là danh sách tuần tự của con số, và được tạo bởi Oracle sever. Sequence dùng để tạo khóa chính một cách tự động cho dữ liệu.

Sequence thường dùng để tạo khóa chính trong sinh mã tự động. Có thể dùng chung cho nhiều đối tượng. Con số sequence này có chiều dài tối đa là 38 số.

Để tạo sequence, dùng lệnh CREATE SEQUENCE

Cú pháp:

```
CREATE SEQUENCE sequence_name
  INCREMENT BY integer
  START WITH integer
  [MAXVALUE integer]
  [MINVALUE integer]
  [CYCLE/NO CYCLE];
```

Với:

INCREMENT BY	Chỉ định khoảng cách của dãy số tuần tự
START WITH	Chỉ định số đầu tiên của dãy số tuần tự
MAXVALUE	Giá trị lớn nhất của dãy tuần tự
MINVALUE	Giá trị nhỏ nhất của dãy tuần tự
CYCLE/NO CYCLE	Dãy tuần tự có quay vòng khi đến điểm cuối. Mặc định là NO CYCLE

Ví dụ:

```
CREATE SEQUENCE sample_sequence
  INCREMENT 1
  START WITH 2
  MAXVALUE 100;
```

Để làm việc với các sequence, dùng lệnh SQL với các cột giả sau

CURRVAL	Cho giá trị hiện thời của sequence
NEXTVAL	Tăng giá trị hiện thời của sequence và cho giá trị sau khi tăng phải xác định tên sequence trước currval và nextval

```
sequence.CURRVAL
sequence.NEXTVAL
```

Để truy cập các sequence không thuộc schema hiện thời, thì phải chỉ ra tên schema

```
schema.sequence.CURRVAL
schema.sequence.NEXTVAL
```

Để truy cập các sequence từ xa, thì còn phải chỉ ra datalink

```
schema.sequence.CURRVAL@dblink
schema.sequence.NEXTVAL@dblink
```

## Sử dụng sequence

CURRVAL và NEXTVAL có thể được sử dụng trong các trường hợp sau:

- ③ Trong danh sách lựa chọn của câu lệnh SELECT
- ③ Trong mệnh đề VALUES của câu lệnh INSERT
- ③ Trong mệnh đề SET của câu lệnh UPDATE
- ③ Không được sử dụng CURRVAL và NEXTVAL trong các trường hợp sau
- ③ Trong câu hỏi con
- ③ Trong các view và snapshot
- ③ Trong câu lệnh SELECT có tác tử DISTINCT
- ③ Trong câu lệnh SELECT có sử dụng GROUP BY hay ORDER BY
- ③ Trong câu lệnh SELECT có sử dụng các phép toán tập hợp như UNION, INTERSET, MINUS
- ③ Trong mệnh đề WHERE của câu lệnh SELECT
- ③ Giá trị DEFAULT của cột trong câu lệnh CREATE TABLE hay ALTER TABLE
- ③ Trong điều kiện của ràng buộc CHECK

### 9.1.2. Thay đổi và huỷ sequence

Thay đổi sequence:

```
ALTER SEQUENCE sequence_name
  INCREMENT BY integer
  START WITH integer
  [MAXVALUE integer]
  [MINVALUE integer]
  [CYCLE/NO CYCLE];
```

Huỷ sequence:

```
DROP SEQUENCE sequence_name ;
```

## 9.2.INDEX

### 9.2.1. Tạo index

Index là một cấu trúc cơ sở dữ liệu, được server sử dụng để tìm một row trong bảng một cách nhanh chóng. Index bao gồm một key value (một cột (column) trong hàng (row)) và ROWID.

Cú pháp:

```
CREATE [UNIQUE]] INDEX index_name
  ON TABLE ( column [,column...]);
```

### 9.2.2. Sử dụng index

Ta sử dụng index trong một số trường hợp sau:

- ③ Dùng index để query cho nhanh.
- ③ Dùng Index khi mà việc lấy dữ liệu <15% số row trong bảng.
- ③ Index những column nào dùng để nối giữa các bảng lẫn nhau.
- ③ Không nên dùng Index cho các bảng nào chỉ có vài row.
- ③ Primary và unique key ( khóa chính và khóa duy nhất) tự động có index, nhưng nên có index cho foreign key( khóa ngoại).



Số lượng index cho một table là không giới hạn. Tuy nhiên nếu có quá nhiều index sẽ gây ảnh hưởng đến số liệu khi mà dữ liệu trong table bị thay đổi thứ tự theo index. Ví dụ: Thêm một row vào bảng tất cả các Index sẽ được update. Nên chọn lựa giữa yêu cầu query, và insert, update để có một index hợp lý. Đối với các khoá PRIMARY KEY và UNIQUE KEY từ khoá UNIQUE được tự động thêm khi tạo INDEX.

Ví dụ:

```
CREATE INDEX i-ENAME ON EMP (ENAME);
```

Xoá INDEX bằng lệnh:

```
DROP INDEX index_name ;
```

## 9.3.BÀI TẬP

1. Tạo Index trên cột PROJID cho bảng ASSIGNMENT.
2. Hiển thị danh sách của nhân viên thuộc sự quản lý của người có tên là 1 biến được nhập từ bàn phím

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7698	BLAKE	MANAGER	7839	01-05-1981	2850		30
7654	MARTIN	SALESMAN	7698	28-09-1981	1250	1400	30
7499	ALLEN	SALESMAN	7698	20-02-1981	1600	300	30
7844	TURNER	SALESMAN	7698	08-09-1981	1500	0	30
7900	JAMES	CLERK	7698	03-12-1981	950		30
7521	WARD	SALESMAN	7698	22-02-1981	1250	500	30

## Chương 10. VIEWS

### 10.1.VIEWS

#### 10.1.1. Tạo view

View là một table logic, view không phải là nơi lưu trữ dữ liệu ở mức vật lý. Các thành phần của view dựa trên table hoặc là trên view khác. Mọi tác động lên view đều gây ảnh hưởng tới table của view đó, và ngược lại. Để định nghĩa một view dùng query trên một bảng hay một view nào đó.

Cú pháp:

```
CREATE [OR REPLACE] [FORCE] VIEW view_name [(column, column,...)]
AS
SELECT statement
[WITH CHECK OPTION [CONSTRAINT constraint_name]];
```

Trong đó:

OR REPLACE	Để tạo view chèn lên view cùng tên
FORCE	Để tạo view cả khi table hay view nào đó không tồn tại trong câu lệnh SELECT.
column, column	Tên các column của view
WITH CHECK OPTION	Nếu có lệnh insert hoặc update lên view, ql sẽ kiểm tra điều kiện phù hợp trong mệnh đề where của view. Nếu không dữ liệu sẽ chỉ kiểm tra các ràng buộc toàn vẹn của bảng.
CONSTRAINT	Chỉ ra tên của điều kiện kiểm tra.

Ví dụ 1:

```
CREATE VIEW emp_view
AS
SELECT empno, ename, sal FROM emp WHERE deptno = 10;
```

Ví dụ 2:

```
CREATE VIEW dept_summary
(name, minsal, maxsal, avsal)
AS
SELECT dname, min(sal), max(sal), avg(sal) FROM emp, dept FROM
emp, dept
WHERE emp.deptno = dept.deptno
GROUP BY dname;
```

Ví dụ 3:

```
CREATE VIEW dept_view
AS
SELECT eame, sal*12 Annsal
FROM emp
WHERE deptno = 20
WITH CHECK OPTIION CONSTRAINT dept_check;
```

#### 10.1.2. Xóa các view

Chỉ những người tạo view mới có quyền DROP

```
DROP VIEW dept_view;
```

View có thể thực hiện các lệnh SQL sau:

- ③ SELECT
- ③ INSERT (insert trên view cũng ảnh hưởng lên table)
- ③ Update (ảnh hưởng lên table)
- ③ Comment

Tuy nhiên có những ràng buộc sau:

- ③ Không thể insert, update trên view, khi query của view chứa các toán tử join, set, distinct, group by, group.
- ③ Không thể nào insert, update trên view, nếu như trong view có dùng with check option.
- ③ Không thể nào insert trên view, trên table có những cột not Null mà không dùng default value (bởi vì trong trường hợp này view sẽ có ít column hơn table table. Nên insert 1 row vào view, thực chất là insert row đó vào table sẽ không hợp lệ).
- ③ Không thể nào insert trên view, nếu view này có dùng biểu thức decode.
- ③ Những query của view không thể nào tham khảo vào 2 column giả nextval, curval (nextval, curval dùng cho sequence).

## 10.2.BÀI TẬP

1. Tạo view có hiển thị như sau:

```
select * from aggregated;
```

DEPTNO	AVERAGE	MAXIMUM	MINIMUM	SUM	NO_SALS	NO_COMMS
10	2916.66667	5000	1300	8750	3	0
20	2235	3300	800	11175	5	0
30	1566.66667	2850	950	9400	6	4

2. Tạo view để nhập số liệu vào bảng ASSIGNMENT với các điều kiện sau:

```
PROJID <2000, P_START_DATE<P_END_DATE
```

Các giá trị có thể chấp nhận của assign\_type là PS, WT hoặc ED

```
EMPNO có giá trị NOT NULL
```

```
BILL_RATE < 50 Với ASSIGN_TYPE Là PS
```

```
BILL_RATE < 60 Với ASSIGN_TYPE Là WT
```

```
BILL_RATE < 70 Với ASSIGN_TYPE Là ED
```

3. Định nghĩa bảng MESSAGES có cấu trúc

```
Column name Data Type
```

NUMCOL1	NUMBER(9,2)
NUMCOL2	NUMBER(9,2)
CHARCOL1	VARCHAR2(60)
CHARCOL2	VARCHAR2(60)
DATECOL1	DATE
DATECOL2	DATE

## Chương 11. QUYỀN VÀ BẢO MẬT

### 11.1. QUYỀN - PRIVILEGE

Privileges là các quyền hạn được thực hiện các thao tác hoặc thực hiện việc truy nhập đến các đối tượng dữ liệu. Trong Oracle bạn sẽ không thể thực hiện được các thao tác mà không có các quyền tương ứng. Các quyền hạn này được gán cho User để có thể thực hiện các thao tác trên các đối tượng chỉ định. Việc gán quyền được thực hiện bởi người quản trị cơ sở dữ liệu.

Gán quyền hoặc loại bỏ: Để thực hiện gán quyền cho một đối tượng dùng lệnh Grant loại bỏ quyền hạn dùng Revoke (hoặc bằng các công cụ hỗ trợ khác như Oracle Enterprise manager)

Các quyền bao gồm:

- ③ Bảo mật CSDL
- ③ Bảo mật hệ thống
- ③ Bảo mật dữ liệu
- ③ Quyền hệ thống: Quyền truy nhập và CSDL
- ③ Quyền trên đối tượng: Thao tác nội dung của các đối tượng CSDL
- ③ Schema là tập hợp các đối tượng như tables, view...

CSDL: Khi cài đặt xong hệ quản trị CSDL Oracle mặc định đã có 2 user.

- ③ SYS: Có quyền cao nhất. Mật khẩu là change\_on\_install
- ③ SYSTEM: Có quyền thấp hơn SYS. Mật khẩu là MANAGER

Quyền hệ thống

Trong các quyền hệ thống quyền DBA là lớn nhất. DBA có quyền

- ③ CREATE USER : Tạo user mới
- ③ DROP USER : Xóa user
- ③ DROP ANY TABLE : Xóa table
- ③ BACKUP ANY TABLE : Tạo các backup table.

Lệnh tạo user của người có quyền DBA như sau:

```
CREATE USER user_name  
IDENTIFY BY password;
```

Quyền trên đối tượng:

- ③ CREATE SESSION: Truy nhập vào CSDL
- ③ CREATE TABLE: tạo bảng trong user đó
- ③ CREATE SEQUENCE: Tạo sequence
- ③ CREATE VIEW: Tạo view
- ③ CREATE PROCEDURE: Tạo procedure
- ③

Gán quyền

```
GRANT privilege[,privilege...] TO user [,user...]
```

Xoá quyền

```
REVOKE privilege[,privilege...] FROM user [,user...]
```

## 11.2.ROLE

Role là tên của một nhóm các quyền hạn. Nó được tạo để quản lý quyền hạn cho các ứng dụng hoặc nhóm các User. Việc dùng role cho phép quản lý thống nhất trên các đối tượng, tăng tính mềm dẻo trong quản trị, dễ dàng thay đổi. Ví dụ hai đối tượng X, Y có quyền trên role A tức là role A có quyền gì thì X, Y có quyền tương ứng khi role A bị thay đổi quyền hạn thì X, Y cũng bị thay đổi quyền hạn theo.

Lệnh tạo Role

Cú pháp:

```
CREATE ROLE role [IDENTIFY BY password];
```

Gán privilege cho Role

Gán Role có các đối tượng

Một số Role hay dùng:

- ③ CONNECT
- ③ RESOURCE

Lệnh gán và xoá Role giống như lệnh gán và xoá Privilege. Chi tiết xem trong phần quản trị Oracle.

## 11.3.SYNONYM

Synonyms là bí danh cho mọi đối tượng của Oracle. Các đối tượng của Oracle là table, view, snapshot, sequence, procedure, function, package và các synonym khác. cú pháp

```
CREATE PUBLIC SYNONYM synonym_name  
FROM [OWNER.]object_name;
```

Dùng Synonyms có những lợi điểm sau:

- ③ Không tốn thêm nơi lưu trữ khác bởi vì nó đã được cất trên từ điển dữ liệu.
- ③ Làm đơn giản đoạn chương trình SQL.
- ③ Tăng tính bảo mật cho database.
- ③ Có thể cho phép mọi người (public) truy xuất các đối tượng của Oracle.

Ví dụ: Chúng ta có một table EMPLOY trong schema emp\_01

Khi lập trình thì phải truy xuất theo emp\_01. EMPLOY, tên dài như vậy thì đoạn chương trình sẽ dài sẽ dễ lẫn lộn. Nên chúng ta phải dùng synonym

```
CREATE SYNONYM EMP FOR EMP_01.EMPLOY;
```

Có thể tạo một synonym cho phép mọi người có thể tham khảo tới

```
CREATE PUBLIC EMP FOR EMP_01.EMPLY;
```

Tính bảo mật là vì synonym là bí danh, nên người sử dụng dùng bí danh này sẽ không đoán được thêm thông tin gì.

## Chương 12. GIỚI THIỆU NGÔN NGỮ PL/SQL

### 12.1. TỔNG QUAN VỀ PL/SQL

#### 12.1.1. Cú pháp lệnh PL/SQL

- ③ Mỗi lệnh SQL kết thúc bằng dấu (;)
- ③ Lệnh định nghĩa CSDL (DDL) không được sử dụng trong PL/SQL
- ③ Lệnh SELECT trả về nhiều dòng có thể gây exception
- ③ Lệnh DML có thể tác động trên nhiều dòng

Ví dụ:

```
x := 1;
INSERT INTO emp (id, name)
VALUES (50, 'GARNOR');
BEGIN
    SELECT name FROM dept INTO :DEPT.NAME;
EXCEPTION
    WHEN others THEN
        Message(SQLERRM);
END;

UPDATE emp
    SET sal := sal*1.2
    WHERE dept_id = 10;
```

#### 12.1.2. Khối lệnh PL/SQL

Ngôn ngữ PL/SQL tổ chức các lệnh theo từng khối lệnh. Một khối lệnh PL/SQL cũng có thể có các khối lệnh con khác ở trong nó.

Cấu trúc đầy đủ của một khối lệnh PL/SQL bao gồm:

```
DECLARE /* Phần khai báo - Không bắt buộc */
    Khai báo các biến sử dụng trong phần thân
BEGIN /* Phần thân */
    Đoạn lệnh thực hiện;
EXCEPTION /* Phần xử lý lỗi - Không bắt buộc */
    Xử lý lỗi xảy ra;
END;
```

Ví dụ 1:

```
DECLARE
    empno NUMBER(4) := 7788;
BEGIN
    UPDATE emp
    SET sal = 9000
        WHERE empno = 0001;
END;
```

Ví dụ 2:

```
DECLARE
    v_deptno NUMBER(2);
    v_loc VARCHAR2(15);
BEGIN
```

```
        SELECT deptno, loc INTO v_deptno, v_loc
           FROM dept
           WHERE dname = 'SALES';
EXCEPTION
    WHEN others THEN
        Message(SQLERRM);
END;
```

## 12.2.LỆNH LẬP TRÌNH PL/SQL ĐƠN GIẢN

### 12.2.1. Lệnh IF

Thực hiện câu lệnh theo điều kiện.

Cú pháp:

```
IF <điều kiện 1> THEN
    Công việc 1;
[ELSIF <điều kiện 2> THEN
    Công việc 2;
]

[ELSE
    Công việc n + 1;
]
END IF;
```

Ví dụ 1:

```
IF ename = 'SCOTT' THEN
    beam_me_up := 'YES';
    COMMIT;
ELSE
    beam_me_up := 'NO';
    ROLLBACK;
END IF;
```

Ví dụ 2:

```
IF choice= 1 THEN
    action := 'Run payroll';
ELSIF choice=2 THEN
    action:='Run';
ELSIF choice=3 THEN
    action:='Backup';
ELSE
    action:='Invalid';
END IF;
```



### 12.2.2. Lệnh lặp LOOP không định trước

Trong lệnh lặp này, số lần lặp tùy thuộc vào điều kiện kết thúc vòng lặp và không xác định được ngay tại thời điểm bắt đầu vòng lặp.

Cú pháp:

```
LOOP
    Công việc;
    EXIT WHEN điều kiện;
END LOOP;
```

Ví dụ:

```
x := 0;
y := 1000;
LOOP
    x := x + 1;
    y := y - x;
    EXIT x > y;
END LOOP;
```

### 12.2.3. Lệnh lặp LOOP có định trước

Ngay khi bắt đầu vòng lặp, ta đã xác định được số lần lặp. Cú pháp:

```
LOOP Index IN Cận dưới .. Cận trên
    Công việc;
END LOOP;
```

Ví dụ:

```
x := 0;
LOOP Index IN 1 .. 100
    x := x + 1;
END LOOP;
```

### 12.2.4. Lệnh lặp WHILE

Cú pháp:

```
WHILE Điều kiện LOOP
    Công việc;
END LOOP;
```

Ví dụ:

```
WHILE length(:Address) < 50 LOOP
    :Address := :Address || ' ';
END LOOP;
```

### 12.2.5. Lệnh GOTO, nhảy vô điều kiện

Cú pháp:

```
GOTO Nhãn;
```

Ví dụ:

```
BEGIN
    <<Nhãn>>
    công việc;

    GOTO Nhãn;

END;
```

## 12.3.GIỚI THIỆU CURSOR

Cursor là kiểu biến có cấu trúc, cho phép ta xử lý dữ liệu gồm nhiều dòng. Số dòng phụ thuộc vào câu lệnh truy vấn dữ liệu sau nó. Trong quá trình xử lý, ta thao tác với cursor thông qua từng dòng dữ liệu. Dòng dữ liệu này được định vị bởi một con trỏ. Với việc dịch chuyển con trỏ, ta có thể lấy được toàn bộ dữ liệu trả về.

Các bước sử dụng biến cursor:

Khai báo --> mở cursor --> lấy dữ liệu để xử lý --> đóng cursor

Khai báo:

```
CURSOR Tên cursor( danh sách biến) IS Câu
lệnh truy vấn;
```

Ví dụ1:

```
CURSOR      c_Dept      IS
            SELECT deptno, dname
            FROM dept
            WHERE deptno>10;
```

Ví dụ2:

```
CURSOR      c_Dept(p_Deptno NUMBER)      IS
            SELECT deptno, dname
            FROM dept
            WHERE deptno>10;
```

Mở cursor:

```
OPEN Tên cursor | Tên cursor( danh sách biến);
```

Ví dụ1:

```
OPEN      c_Dept;
```

Ví dụ2:

```
OPEN      c_Dept(10);
```

Lấy dữ liệu:

```
FETCH Tên cursor INTO Tên biến;
```

Ví dụ:

```
FETCH c_Dept INTO v_Dept;
```

Đóng cursor:

```
CLOSE Tên cursor;
```

Ví dụ:

```
CLOSE c_Dept;
```

Các thuộc tính:

%isopen                    trả lại giá trị True nếu cursor đang mở

%notfound	trả lại giá trị True nếu lệnh fetch hiện thời trả lại không có row
%found	trả lại giá trị true cho đến khi fetch không còn row nào
%rowcount	trả lại số row đã được thực hiện bằng lệnh fetch

Ví dụ1:

```
DECLARE
    -- Khai báo cursor để truy vấn dữ liệu
    CURSOR c_Emp IS
SELECT *
FROM emp
WHERE dept_id = 10;

    -- Khai báo biến cursor tương ứng để chứa dòng dữ liệu
    v_Emp c_EMP%rowtype;

BEGIN
    -- Mở cursor
    OPEN c_Emp;
    LOOP
        -- Lấy dòng dữ liệu từ cursor
        FETCH c_Emp INTO v_Emp;

        -- Thoát khỏi vòng lặp nếu đã lấy hết dữ liệu trong cursor
        EXIT WHEN c_Emp%notfound;

        -- Bổ sung dữ liệu vào Emp_ext theo dữ liệu lấy được từ
        cursor
        INSERT INTO Emp_ext (empno, ename, job)
VALUES (v_Emp.empno, v_Emp.ename, v_Emp.job); END
    LOOP;
    -- Đóng cursor
    CLOSE c_Emp;
END;
```

Ví dụ 2:

```
DECLARE
    -- Khai báo cursor, có cho phép cập nhật dữ liệu
    CURSOR c_Dept IS
        SELECT dname, loc
        FROM dept FOR UPDATE OF loc;

    -- Khai báo biến lưu trữ dữ liệu
    v_Dept c_Dept%ROWTYPE;
    v_sales_count NUMBER:=0;
    v_non_sales NUMBER:=0;

BEGIN
    -- Mở cursor
    OPEN c_Dept;
    LOOP
        -- Lấy từng dòng dữ liệu của cursor để xử lý
        FETCH c_Dept INTO v_Dept;

        -- Thoát khỏi lệnh lặp nếu đã duyệt hết tất cả dữ liệu EXIT
        WHEN c_Dept %notfound;

        IF (v_Dept.dname = 'SALES')AND(v_Dept.loc!='DALLAS') THEN
            -- Cập nhật dữ liệu trên cursor
```

```
UPDATE Dept
    SET loc='DALLAS'
    WHERE CURRENT OF c_Dept;

-- Đếm số lượng bản ghi được cập nhật
v_sales_count := sales_count + 1;
ELSIF (v_dept.dname != 'SALES')AND(v_Dept.loc!='NEWYORK')
THEN
-- Cập nhật dữ liệu trên cursor
UPDATE Dept
    SET loc = 'NEWYORK'
    WHERE CURRENT OF c_Dept;

-- Đếm số lượng bản ghi được cập nhật
v_non_sales := v_non_sales + 1;
END IF;
END LOOP;
-- Đóng cursor
CLOSE c_Dept;

-- Lưu giữ các thông số vừa xác định vào bảng
INSERT INTO counts (sales_set, non_sales_set)
VALUES (v_sales_count, v_non_sales);

-- Ghi nhận các thay đổi dữ liệu ở trên
COMMIT;
END;
```

## 12.4.CÁC KIỂU DỮ LIỆU THÔNG DỤNG

### 12.4.1. Kiểu dữ liệu Table

Cú pháp:

```
TYPE Tên_kiểu_Table IS
    TABLE OF Tên kiểu dữ liệu [NOT NULL] INDEX BY BINARY_INTEGER;
```

Tên biến      Tên\_kiểu\_Table;

Ví dụ:

```
TYPE t_Name IS
    TABLE OF Emp.Ename%TYPE INDEX BY BINARY_INTEGER;
```

v\_First\_name      t\_Name;  
v\_Last\_name      t\_Name;

### 12.4.2. Kiểu dữ liệu Record

Cú pháp:

```
TYPE Tên_kiểu_Record IS
    RECORD OF (
        Col1              Tên kiểu [NOT NULL{:=|DEFAULT} biểu thức],
        Col2              Tên kiểu [NOT NULL{:=|DEFAULT} biểu
                        thức]...);
```

Tên biến      Tên\_kiểu\_Record;

Ví dụ:

```
TYPE t_Emp IS
    RECORD OF (
        empno            number(4) not null,
```

```
        ename      char(10),
        job        char(9),
        mgr        number(4),
        hiredate   date default sysdate,
        sal        number(7,2),
        comm       number(7,2),
        deptno    number(2) not null);

v_Emp_record      t_Emp;
```

### 12.4.3. Sao kiểu dữ liệu một dòng

Bản ghi trong PL/SQL. là một biến có thể giữ nhiều giá trị và là một tập hợp các biến tương ứng với các trường trong table.

Khai báo kiểu dữ liệu bản ghi.

```
Tên biến          Tên bảng%ROWTYPE;
```

Ví dụ:

```
v_Emp emp%ROWTYPE;
```

Truy nhập đến các trường trong dữ liệu bản ghi dùng giống như trong 1 dòng dữ liệu trả về. Ví dụ:

```
v_Emp.empno, v_Emp.sal, ...
```

### 12.4.4. Sao kiểu dữ liệu của một cột

Cú pháp:

```
Tên biến          Tên cột dữ liệu%TYPE;
```

Ví dụ:

```
v_Sal              Emp.sal%TYPE;
```

### 12.4.5. Lệnh SELECT... INTO

Cú pháp:

```
SELECT col1, col2...
      INTO var1, var2... [cursor_var]
      FROM table1, table2...
      [WHERE condition1, condition2... ]
      [GROUP BY col1, col2 ...]
      [HAVING condition1, condition2... ]
      [FOR UPDATE];
```

Với:

```
INTO var1, var2... [cursor_var]      Biến lưu giữ các giá trị trong
      table lấy từ lệnh select.
```

Ví dụ:

```
SELECT deptno, loc INTO v_deptno, v_loc
      FROM dept
      WHERE dname = 'SALES';
```

## 12.5.BÀI TẬP

1. Viết đoạn chương trình tìm kiếm các hàng trong bảng EMP với biến được đưa từ ngoài vào là &1 dạng JOB\_type(emp.job%type) và đưa ra thông báo thích hợp vào bảng MESSAGES.
  
2. Viết đoạn chương trình ghi dữ liệu vào bảng MESSAGES với cột NUMCOL1 mang giá trị là 1 nếu là row 1 được Insert, 2 nếu row 2 được Insert... . Không được Insert những row có giá trị là 6 hoặc 8, thoát khỏi vòng lặp insert sau giá trị 10. Commit sau vòng lặp.
  
3. Liệt kê các cột ENAME, HIREDATE, SAL Với điều kiện EMPNO bằng giá trị biến &EMPLOYEE\_NO được đưa vào, sau đó kiểm tra:
  - 1.1 Có phải mức lương lớn hơn 1200
  - 1.2 Tên nhân viên có phải có chứa chữ T
  - 1.3 ngày gia nhập cơ quan có phải là tháng 10 (DEC)và đưa giá trị kiểm tra này vào bảng message cột charcol1 (thử với các giá trị 7654, 7369, 7900, 7876)
  
4. Đưa vào vòng lặp v từ 1 đến 10 lệnh:

```
UPDATE messages
SET      numcol2=100
WHERE numcol1 = v;
```

nếu bất kỳ một lần update nào đó có số lượng row >1 thì exit khỏi vòng lặp.

## Chương 13. GIỚI THIỆU PROCEDURE BUILDER

Procedure builder là một thành phần được tích hợp vào môi trường phát triển ứng dụng của Oracle. Nó cho phép người sử dụng có thể soạn thảo, biên dịch, kiểm tra và dò lỗi đối với các hàm, thủ tục hay package viết bởi ngôn ngữ PL/SQL ở cả Client và Server.

### 13.1.CÁC THÀNH PHẦN TRONG PROCEDURE BUILDER

Thành phần	Diễn giải
Object Navigator	Điều khiển truy nhập các hàm, thủ tục PL/SQL. Thực hiện thao tác dò lỗi (debug) trên các khối lệnh SQL và PL/SQL.
PL/SQL Interpreter	Dò lỗi mã nguồn PL/SQL.
Program Unit Editor	Tạo và soạn thảo các mã nguồn khối lệnh PL/SQL.
Store Program Unit Editor	Tạo và soạn thảo các mã nguồn khối lệnh PL/SQL trong các Store Program thuộc Server.
Database Trigger Edditor	Tạo và soạn thảo các mã nguồn khối lệnh PL/SQL trong các Trigger thuộc Server.

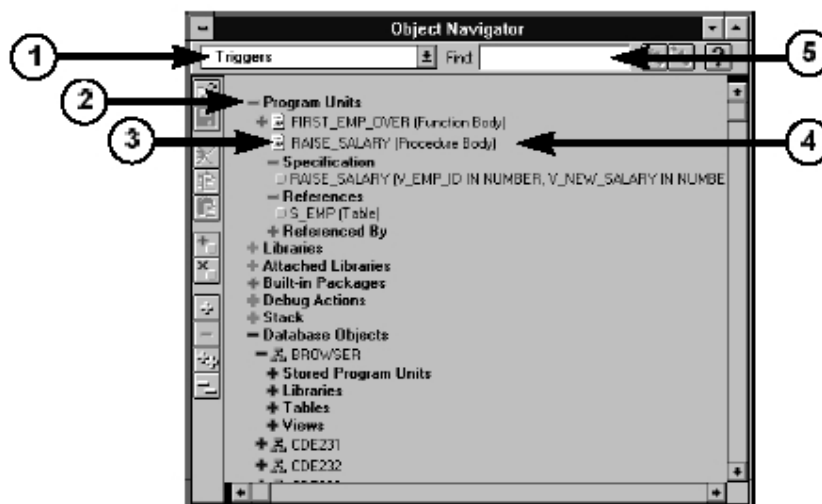
#### 13.1.1. Object Navigator

Object Navigator cho phép hiển thị các đối tượng trong database.

Ta có thể tạo, soạn thảo các thủ tục PL/SQL cũng như dò lỗi, nạp các thư viện thông qua Object Navigator.

Với Object Navigator, ta cũng có thể thực hiện sao chép các thủ tục, hàm thông qua các thao tác đơn giản như copy và paste.

### The Object Navigator



Hình vẽ 7. Cấu trúc của Object Navigator

Các thành phần chính của Object Navigator bao gồm:

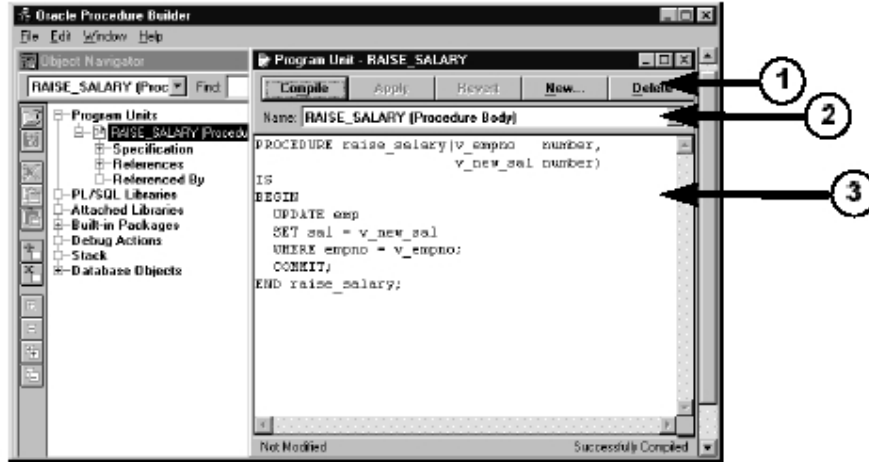
- ③ Navigator drop down list: Danh sách sổ xuống hiển thị tên các thủ tục
- ③ Subject indicator: Định vị các đối tượng cần soạn thảo
- ③ Type icon: Biểu tượng cho các loại

- ③ Object name: Tên các đối tượng
- ③ Find field: Tìm kiếm các đối tượng theo tên

### 13.1.2. Program Unit Editor

Là môi trường để tạo, soạn thảo, biên dịch và hiển thị lỗi biên dịch các hàm, thủ tục.

## The Program Unit Editor



Hình vẽ 8. Soạn thảo hàm, thủ tục phía Client

Các thành phần chính:

- ③ Các nút bấm thực hiện công việc: Compile, Apply, Revert, New, Delete, Close và Help
- ③ Danh sách tên các hàm, thủ tục khác
- ③ Nơi soạn thảo hàm, thủ tục

### 13.1.3. Store Program Unit Editor

Cũng tương tự như Program Unit Editor, Store Program Unit Editor được sử dụng cho việc tạo, soạn thảo các hàm, thủ tục trên server.

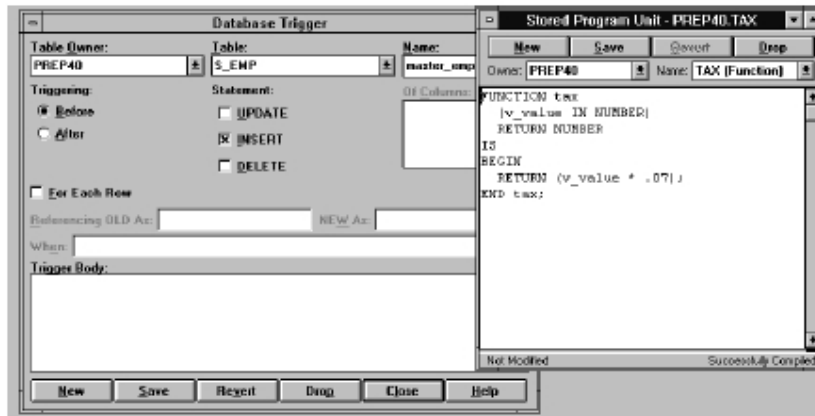
Các chức năng trong Store Program Unit Editor hoàn toàn tương tự như trong Program Unit Editor.

Ta chỉ gọi Store Program Unit Editor sau khi đã thực hiện kết nối tới một database cụ thể nào đó.

### 13.1.4. Database Trigger Editor

Là môi trường dùng để tạo và soạn thảo các trigger database trên server.





Hình vẽ 9. Soạn thảo hàm, thủ tục, trigger phía Server

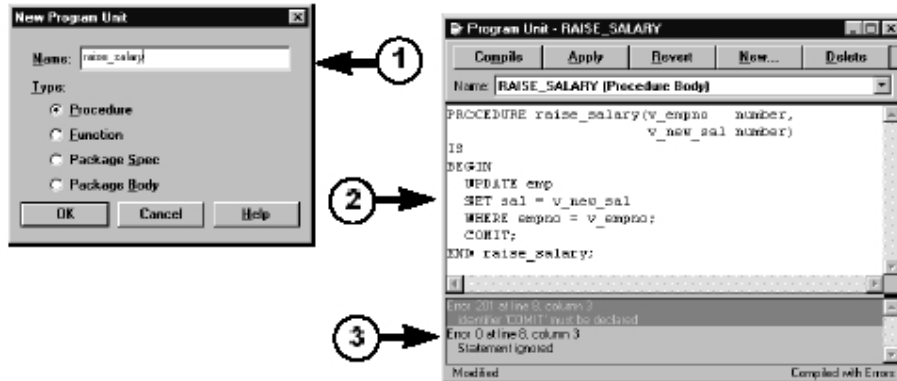
Trigger database được phân ra làm nhiều loại khác nhau và được thực hiện trước hoặc sau mỗi thao tác cụ thể trên từng bảng dữ liệu của database.

## 13.2.CÁC HÀM, THỦ TỤC

### 13.2.1. Tạo hàm, thủ tục trên Client

Đối với hàm, thủ tục hay package trên client, ta có thể tạo và biên dịch ngay chúng. Oracle Builder hỗ trợ trình thông dịch cho phép kiểm tra lỗi của đoạn chương trình vừa thực hiện.

## Creating a Program Unit on the Client Side



Hình vẽ 10. Tạo hàm, thủ tục tại Client

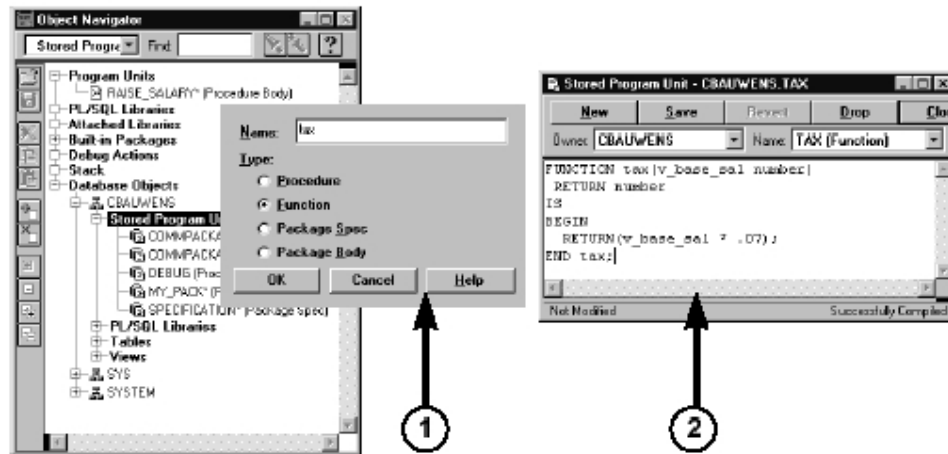
Việc tạo hàm, thủ tục được thực hiện theo ba bước:

- ③ Khai báo tên hàm hay thủ tục
- ③ Soạn thảo nội dung của hàm hay thủ tục
- ③ Biên dịch hàm hay thủ tục vừa tạo và xác định các lỗi nếu có.

### 13.2.2. Tạo hàm, thủ tục trên Server

Procedure Builder chỉ cho phép tạo mới, sửa chữa và lưu lại các thay đổi đối với các hàm và thủ tục trên Server, không hỗ trợ việc biên dịch và phát hiện lỗi.

## Creating a Program Unit on the Server Side



Hình vẽ 11. Tạo hàm, thủ tục tại Server

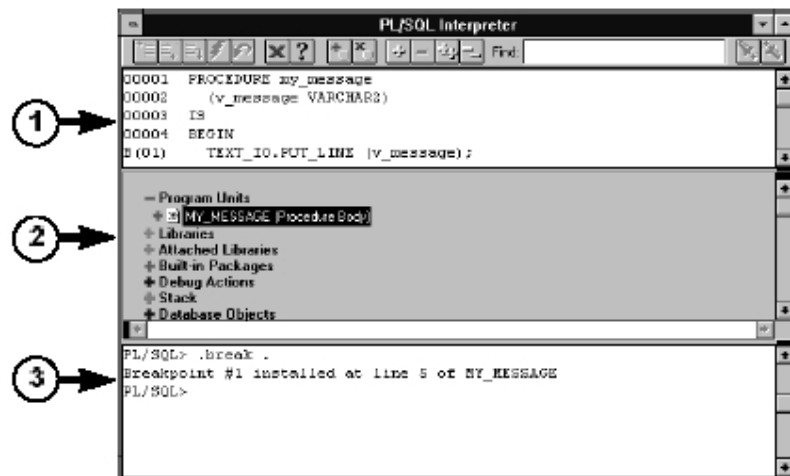
Ta thực hiện việc tạo hàm, thủ tục trên server theo hai bước:

- ③ Tạo hàm, thủ tục
- ③ Soạn thảo và ghi lại nội dung của hàm, thủ tục

### 13.2.3. Dò lỗi đối với các hàm, thủ tục

Với Procedure Builder, ta có thể thực hiện chạy các hàm, thủ tục theo từng bước. Qua đó, ta có thể phát hiện được các lỗi xảy ra trong chương trình, nếu có. Màn hình PL/SQL Interpreter cho phép ta thực hiện điều này:

## The PL/SQL Interpreter



Hình vẽ 12. Màn hình PL/SQL Interpreter

Cấu trúc của màn hình PL/SQL Interpreter được chia làm ba phần chính:

- ③ Phần mã nguồn hàm, thủ tục
- ③ Phần điều khiển
- ③ Phần tương tác trực tiếp với dữ liệu

## Chương 14. GIỚI THIỆU CÁC THỦ TỤC, HÀM VÀ PACKAGE

### 14.1. THỦ TỤC

Một nhóm các lệnh thực hiện chức năng nào đó có thể được gom lại trong một thủ tục (procedure) nhằm làm tăng khả năng xử lý, khả năng sử dụng chung, tăng tính bảo mật và an toàn dữ liệu, tiện ích trong phát triển.

Thủ tục có thể được lưu giữ ngay trong database như một đối tượng của database, sẵn sàng cho việc tái sử dụng. Thủ tục lúc này được gọi là Store procedure. Với các Store procedure, ngay khi lưu giữ Store procedure, chúng đã được biên dịch thành dạng p-code vì thế có thể nâng cao khả năng thực hiện.

#### 14.1.1. Tạo thủ tục

Ta có thể tạo thủ tục trực tiếp bằng dòng lệnh sau: Cú

pháp:

```
CREATE [OR REPLACE] PROCEDURE procedure_name
    [(argument1      [mode1]      datatype1,
      argument2      [mode2]      datatype2,
      ...)]
  IS | AS
BEGIN
    PL/SQL Block;
END;
```

Với:

procedure_name	Tên thủ tục
argument	Tên tham số
mode	Loại tham số: IN hoặc OUT hoặc IN OUT, mặc định là IN
datatype	Kiểu dữ liệu của tham số
PL/SQL Block	Nội dung khối lệnh SQL và PL/SQL trong thủ tục

Ví dụ:

```
CREATE OR REPLACE PROCEDURE change_sal
    (p_Percentage      IN number,
      p_Error           OUT varchar2,
    )
  IS
    v_User_exp         Exception;
  BEGIN
    IF p_Percentage < 0 THEN
      RAISE v_User_exp;
    END IF;

    UPDATE emp
      SET sal = sal*p_Percentage/100;
  EXCEPTION
```

```

WHEN v_User_exp THEN
    p_Error := 'Lỗi: Phần trăm nhỏ hơn 0';
    RETURN;
WHEN others THEN
    p_Error := 'Lỗi: ' || SQLERRM;
END;

```

Với việc tạo các thủ tục thông qua câu lệnh, ta có thể dễ dàng tạo các script chứa các thủ tục cần thiết khi tạo mới một database.

Một cách khác, ta có thể tạo mới hay sửa đổi thủ tục thông qua công cụ của Oracle. Trong chương trước, ta đã biết cách sử dụng Procedure Builder để tạo mới thủ tục.

#### 14.1.2. Huỷ bỏ thủ tục

Tương tự như việc tạo thủ tục, ta có thể huỷ bỏ thủ tục thông qua câu lệnh SQL. Cú pháp:

```
DROP PROCEDURE      Tên thủ tục;
```

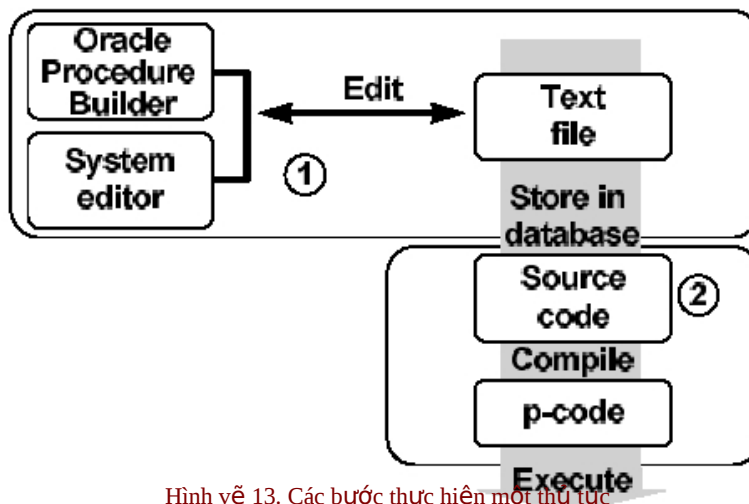
Ví dụ:

```
DROP PROCEDURE      change_sal;
```

#### 14.1.3. Các bước lưu giữ một thủ tục

Một thủ tục trong Oracle được thực hiện theo hai bước chính sau:

1. Nội dung của thủ tục được thiết lập và lưu giữ trong database dưới dạng văn bản (text)
2. Toàn bộ nội dung của thủ tục được biên dịch ra dạng mã p-code, tiện cho việc thực hiện thủ tục đó.



Hình vẽ 13. Các bước thực hiện một thủ tục

## 14.2.HÀM

Tương tự như thủ tục, hàm (function) cũng là nhóm các lệnh PL/SQL thực hiện chức năng nào đó. Khác với thủ tục, các hàm sẽ trả về một giá trị ngay tại lời gọi của nó.

Hàm cũng có thể được lưu giữ ngay trên database dưới dạng Store procedure.

### 14.2.1. Tạo hàm

Ta có thể tạo hàm trực tiếp bằng dòng lệnh sau: Cú

pháp:

```
CREATE [OR REPLACE] FUNCTION function_name
    [(argument1          [mode1]    datatype1,
      argument2          [mode2]    datatype2,
      ...)]
RETURN          datatype
IS | AS
BEGIN
    PL/SQL Block;
END;
```

Với:

function_name	Tên hàm
argument	Tên tham số
mode	Loại tham số: IN hoặc OUT hoặc IN OUT, mặc định là IN
datatype	Kiểu dữ liệu của tham số
PL/SQL Block	Nội dung khối lệnh SQL và PL/SQL trong thủ tục

Ví dụ:

```
CREATE OR REPLACE FUNCTION get_sal
    (p_Emp_id          IN number)
RETURN varchar2
IS
BEGIN
    SELECT sal
    FROM emp
    WHERE emp_id = p_Emp_id;

    RETURN null;
EXCEPTION
    WHEN others THEN
        RETURN 'Lỗi: ' || SQLERRM;
END;
```

### 14.2.2. Thực hiện một hàm

Quá trình lưu giữ và biên dịch một hàm cũng tương tự như đối với một thủ tục. Quá trình gọi và thực hiện một hàm được diễn ra theo ba bước:

1. Việc gọi hàm được thực hiện ngay khi tên hàm trong biểu thức được tham chiếu tới
2. Một biến host (host variable) được tự động tạo ra để lưu giữ giá trị trả về của hàm
3. Thực hiện nội dung trong phần thân hàm, lưu lại giá trị

Ví dụ:

```
SQL> VARIABLE v_Sal number ;

SQL> EXECUTE :v_SAL := get_sal(7934);
PL/SQL procedure successfully completed.

SQL> PRINT v_Sal;
v_Sal

1300
```

### 14.2.3. Lợi ích của việc sử dụng hàm

Với việc sử dụng hàm, trong một số trường hợp ta có thể thấy được các lợi điểm như sau:

- ③ Cho phép thực hiện các thao tác phức tạp (các phép tìm kiếm, so sánh phức tạp) ngay trong mệnh đề của câu lệnh SQL mà nếu không sử dụng hàm ta sẽ không thể nào thực hiện được
- ③ Tăng tính độc lập của dữ liệu do việc phân tích và xử lý dữ liệu được thực hiện ngay trên Server thay vì trả về dữ liệu trực tiếp cho ứng dụng dưới Client để chúng tiếp tục xử lý.
- ③ Tăng tính hiệu quả của câu lệnh truy vấn bằng việc gọi các hàm ngay trong câu lệnh SQL
- ③ Ta có thể sử dụng hàm để thao tác trên các kiểu dữ liệu tự tạo. ③ Cho phép thực hiện đồng thời các câu lệnh truy vấn

### 14.2.4. Một số hạn chế khi sử dụng hàm trong câu lệnh SQL

- ③ Chỉ các hàm do người dùng định nghĩa được lưu trên database mới có thể sử dụng được cho câu lệnh SQL.
- ③ Các hàm do người dùng định nghĩa chỉ được áp dụng cho điều kiện thực hiện trên các dòng dữ liệu (mệnh đề WHERE), không thể áp dụng cho các điều kiện thực hiện trên nhóm (mệnh đề GROUP).
- ③ Tham số sử dụng trong hàm chỉ có thể là loại IN, không chấp nhận giá trị OUT hay giá trị IN OUT.
- ③ Kiểu dữ liệu trả về của các hàm phải là kiểu dữ liệu DATE, NUMBER, NUMBER. Không cho phép hàm trả về kiểu dữ liệu như BOOLEAN, RECORD, TABLE. Kiểu dữ liệu trả về này phải tương thích với các kiểu dữ liệu bên trong Oracle Server.

### 14.2.5. Huỷ bỏ hàm

Tương tự như việc tạo hàm, ta có thể huỷ bỏ hàm thông qua câu lệnh SQL. Cú pháp:

```
DROP FUNCTION Tên hàm;
```

Ví dụ:

```
DROP FUNCTION get_sal;
```

### 14.2.6. Hàm và thủ tục

Ta tạo các thủ tục để lưu giữ một loạt các câu lệnh phục vụ cho nhiều lần gọi khác nhau. Thủ tục có thể không có, có một hoặc nhiều tham số. Tuy nhiên thủ tục không trả lại bất kỳ một kết quả nào.

Hàm cũng giống như thủ tục, nó cũng bao gồm một loạt các câu lệnh, có thể không có, có một hoặc nhiều tham số. Tuy nhiên khác với thủ tục, hàm bao giờ cũng trả về một kết quả. Vì vậy, ta sử dụng hàm trong các phép tính toán, gán giá trị. Khi đó, câu lệnh thực hiện sẽ dễ dàng và sáng sủa hơn.

So sánh giữa hàm và thủ tục

Thủ tục	Hàm
Thực hiện giống như thực hiện các câu lệnh PL/SQL	Có thể được gọi giống như một phần của biểu thức
Không có kiểu giá trị trả về	Có chứa giá trị trả về
Có thể trả về một hoặc nhiều giá trị (thông qua tham số OUT)	Trả về một giá trị

Lợi ích của việc sử dụng hàm, thủ tục

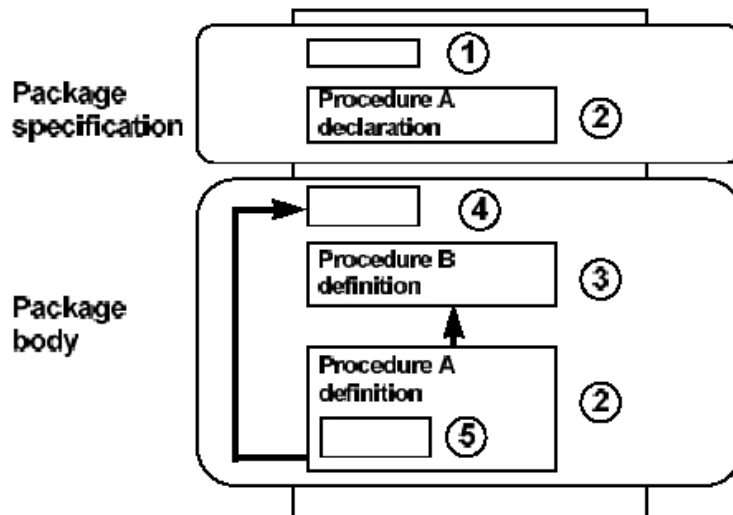
- ③ Nâng cao hiệu suất: Tránh việc tái sử dụng các câu lệnh nhiều lần bởi nhiều User khác nhau. Giảm thiểu thời gian biên dịch câu lệnh PL/SQL trong pha phân tích câu lệnh. Giảm thiểu số lần gọi lệnh thực hiện trên database, từ đó, làm giảm lưu lượng truyền thông trên mạng.
- ③ Nâng cao khả năng bảo trì: Ta có thể dễ dàng sửa nội dung bên trong các hàm, thủ tục mà không ảnh hưởng đến việc giao tiếp của chúng (các tham số và lời gọi vẫn y nguyên). Thay đổi nội dung của một hàm, hay thủ tục có thể ứng dụng được ngay cho nhiều user khác nhau.
- ③ Tăng tính bảo mật và toàn vẹn của dữ liệu: Với việc điều khiển truy nhập dữ liệu dán tiếp đối với các đối tượng trong database sẽ làm nâng cao tính bảo mật của dữ liệu. Quan hệ giữa các câu lệnh trong hàm, thủ tục luôn được đảm bảo.

## 14.3.PACKAGE

Package là một tập hợp các kiểu dữ liệu, biến lưu giữ giá trị và các thủ tục, hàm có cùng một mối liên hệ với nhau, được gộp chung lại. Đặc điểm nổi bật nhất của package là khi một phần tử trong package được gọi thì toàn bộ nội dung của package sẽ được nạp vào trong hệ thống. Do đó, việc gọi tới các phần tử khác trong package sau này sẽ không phải mất thời gian nạp vào hệ thống nữa. Từ đó, nâng cao tốc độ thực hiện lệnh của toàn bộ hàm, thủ tục có trong package.

### 14.3.1. Cấu trúc của package

Một package được cấu trúc làm hai phần. Phần mô tả (specification) định nghĩa các giao tiếp có thể có của package với bên ngoài. Phần thân (body) là các cài đặt cho các giao tiếp có trong phần mô tả ở trên.



Hình vẽ 14. Cấu trúc package

Trong cấu trúc của package bao gồm 05 thành phần:

1. Public variable (biến công cộng): là biến mà các ứng dụng bên ngoài có thể tham chiếu tới được.
2. Public procedure (thủ tục công cộng): bao gồm các hàm, thủ tục của package có thể triệu gọi từ các ứng dụng bên ngoài.
3. Private procedure (thủ tục riêng phần): là các hàm, thủ tục có trong package và chỉ có thể được triệu gọi bởi các hàm hay thủ tục khác trong package mà thôi.
4. Global variable (biến tổng thể): là biến được khai báo dùng trong toàn bộ package, ứng dụng bên ngoài tham chiếu được tới biến này.
5. Private variable (biến riêng phần): là biến được khai báo trong một hàm, thủ tục thuộc package. Nó chỉ có thể được tham chiếu đến trong bản thân hàm hay thủ tục đó.

### 14.3.2. Tạo package

Ta có thể tạo package trực tiếp bằng dòng lệnh sau: Cú pháp

khai báo phần mô tả package:

```
CREATE [OR REPLACE] PACKAGE package_name IS
| AS
    public type and các item declarations
    subprogram specifications
END package_name;
```

Với:

package_name	Tên package	
type and item declarations		Phần khai báo các biến, hằng, cursor, ngoại lệ và kiểu sử dụng trong toàn bộ package
subprogram specifications		Khai báo các hàm, thủ tục
PL/SQL		

Cú pháp khai báo phần thân package:

```
CREATE [OR REPLACE] PACKAGE BODY package_name
```



IS   AS	
private type and item declarations	Khai báo các kiểu chỉ sử dụng riêng trong package
subprogram bodies	Nội dung của package
END package_name;	

Với:

package_name	Tên package
type and item declarations	Phần khai báo các biến, hằng, cursor, ngoại lệ và kiểu
subprogram specifications	Khai báo các hàm, thủ tục PL/SQL

Ví dụ:

```
-- Phần khai báo của package
CREATE OR REPLACE PACKAGE comm_package
IS
    v_comm    number := 10; -- Khai báo biến có giá trị khởi tạo

    -- Khai báo thủ tục để giao tiếp với bên ngoài
    PROCEDURE reset_comm (p_comm IN number);
END comm_package;

-- Phần thân của package
CREATE OR REPLACE PACKAGE BODY comm_package
IS
    -- Hàm riêng phần chỉ sử dụng trong package
    FUNCTION    validate_comm
                (v_comm    IN number)
    RETURN BOOLEAN
    IS
        v_max_comm number;
    BEGIN
        SELECT max(comm) INTO v_max_comm
            FROM    emp;
        IF v_comm > v_max_comm THEN
            RETURN    FALSE;
        ELSE
            RETURN    TRUE;
        END IF;
    END validate_comm;

    -- Thủ tục giao tiếp với bên ngoài
    PROCEDURE reset_comm
                (p_comm    IN number)
    IS
```

```
        v_valid    BOOLEAN;
BEGIN
    v_valid := validate_comm(p_comm);
    IF v_valid = TRUE THEN
        v_comm := p_comm;
    ELSE
        RAISE_APPLICATION_ERROR(-20210, 'Invalid    comm');
    END IF;
END reset_comm;
END comm_package;
```

### 14.3.3. Huỷ package

Tương tự như việc tạo package, ta có thể huỷ bỏ hàm thông qua câu lệnh SQL. Cú pháp:

```
-- Huỷ phần package specification
DROP PACKAGE    Tên package;
-- Huỷ phần package body
DROP PACKAGE    BODY        Tên package;
```

Ví dụ:

```
DROP PACKAGE    comm_package;
DROP PACKAGE    BODY        comm_package;
```

### 14.3.4. Lợi ích của việc sử dụng package Tăng tính phân

nhỏ các thành phần (Modularity)

Ta có thể đóng gói các thành phần, cấu trúc có quan hệ logic với nhau trong cùng một module ứng với một package. Việc kế thừa giữa các package rất đơn giản, và được thực hiện một cách trong sáng.

Đơn giản trong việc thiết kế ứng dụng

Tất cả các thông tin cần thiết cho việc giao tiếp đều được đặt trong phần đặc tả của package (package specification). Nội dung phần này có thể được soạn thảo và biên dịch độc lập với phần thân của package (package body). Do đó, các hàm hay thủ tục có gọi tới các thành phần của package có thể được biên dịch tốt. Phần thân của package có thể được tiếp tục phát triển cho đến khi hoàn thành ứng dụng.

Ẩn dấu thông tin (hiding information)

Package cho phép sử dụng các thành phần bên trong dưới dạng public (công cộng) hay private (riêng tư). Tùy theo yêu cầu thiết kế, ta có thể cho phép truy nhập hay ẩn dấu thông tin. Từ đó, có thể bảo vệ được tính toàn vẹn dữ liệu.

Nâng cao hiệu suất sử dụng

Ngay khi gọi một hàm hay thủ tục bất kỳ trong package lần đầu tiên. Toàn bộ nội dung của package sẽ được nạp vào bộ nhớ. Do vậy, các hàm và thủ tục con trong package gọi đến sau này có thể thực hiện ngay mà không cần phải nạp lại vào bộ nhớ. Việc này làm giảm thiểu thao tác truy xuất vào ra (I/O access) nâng cao tốc độ.

Thực hiện quá tải (overloading)

Package cho phép thực hiện quá tải đối với các hàm và thủ tục trong nó. Theo đó, các hàm và thủ tục khác nhau có thể được phép đặt trùng tên. Việc này sẽ nâng cao tính mềm dẻo của việc sử dụng hàm, thủ tục trong package.

#### 14.3.5. Một số package chuẩn của Oracle

Thủ tục	Hàm
DBMS_ALERT	Cung cấp các sự kiện về các thông điệp của database
DBMS_APPLICATION_INFO	Thông tin về các hoạt động hiện thời đối với database
DBMS_DDL	Biên dịch lại các hàm, thủ tục và package. Phân tích các index, table, cluster,...
DBMS_DESCRIBE	Trả về các diễn giải cho các tham số của thủ tục, hàm
DBMS_JOB	Lên kế hoạch thực hiện các đoạn mã lệnh PL/SQL
DBMS_LOCK	Cung cấp các hàm cho phép yêu cầu, giải phóng, điều chỉnh các trạng thái khoá (lock) đối với từng đối tượng trên database.
DBMS_MAIL	Gửi các message từ Oracle Server tới Oracle*mail
DBMS_OUTPUT	Kết xuất các giá trị trả về từ các hàm, thủ tục, trigger,...
DBMS_PIPE	Cho phép xử lý gửi đồng thời các thông điệp
DBMS_SESSION	Cung cấp các phép truy nhập SQL thay vì các câu lệnh session
DBMS_SHARED_POOL	Cho phép lưu giữ các đối tượng trong vùng nhớ chia sẻ.
DBMS_SQL	Cho phép sử dụng lệnh SQL động để truy xuất database
DBMS_TRANSACTION	Điều khiển các giao dịch, cải thiện và nâng cao hiệu quả đối với các giao dịch nhỏ và không phân tán
DBMS_UTILITY	Phân tích các đối tượng trong từng schema.
UTL_FILE	Cho phép truy xuất tới file ngay với câu lệnh PL/SQL

## Chương 15. DATABASE TRIGGER

Database trigger là những thủ tục được thực hiện ngầm định ngay khi thực hiện lệnh SQL như INSERT, DELETE, UPDATE nhằm đảm bảo các quy tắc logic phức tạp của dữ liệu. Thiết kế các database trigger thỏa mãn các yêu cầu sau:

- ③ Sử dụng các database trigger nhằm đảm bảo thực hiện tất cả các thao tác có liên quan tới lệnh can thiệp dữ liệu được thực hiện.
- ③ Chỉ sử dụng database trigger đối với các thao tác trọng tâm.
- ③ Không sử dụng database trigger để thực hiện các ràng buộc sẵn có trong database Oracle. Ví dụ: dùng database trigger để thay thế cho các constrain.
- ③ Sử dụng database trigger có thể gây rối, khó khăn cho việc bảo trì và phát triển hệ thống lớn. Vì thế, ta chỉ sử dụng database trigger khi thật cần thiết.

### 15.1. TẠO TRIGGER

Khi tạo database trigger, ta cần lưu ý tới một số tiêu chí như:

- ③ Thời gian thực hiện: BEFORE, AFTER
- ③ Hành động thực hiện: INSERT, UPDATE, DELETE
- ③ Đối tượng tác động: bảng dữ liệu
- ③ Loại trigger thực hiện: trên dòng lệnh hay trên câu lệnh
- ③ Mệnh đề điều kiện thực hiện
- ③ Nội dung của trigger

#### 15.1.1. Phân loại trigger

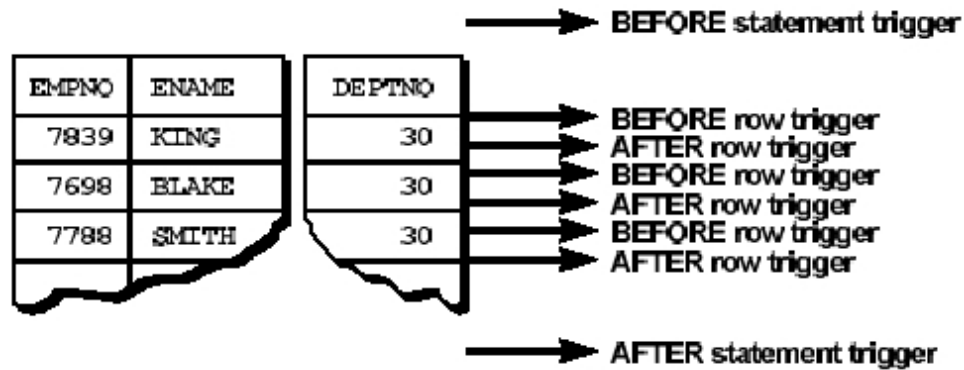
Ta có thể phân loại trigger theo thời gian thực hiện như: BEFORE và AFTER.

- ③ **BEFORE trigger:** Trigger được kích hoạt trước khi thực hiện câu lệnh. Việc này có thể cho phép ta loại bớt các phép xử lý không cần thiết, thậm chí có thể rollback dữ liệu trong trường hợp có thể gây ra các ngoại lệ (exception). Trigger thuộc loại này thường được sử dụng đối với các thao tác INSERT hoặc UPDATE.
- ③ **AFTER trigger:** Câu lệnh được thực hiện xong thì trigger mới được kích hoạt. Thực hiện các công việc thường phải làm sau khi đã thực hiện câu lệnh.
- ③ **INSTEAD OF trigger:** Loại trigger này cho phép người sử dụng có thể thay đổi một cách trong suốt dữ liệu của một số view mà không thể thực hiện thay đổi trực tiếp được. Với INSTEAD OF trigger, ta có thể thực hiện với cả ba thao tác: insert, update, delete.

Ta cũng có thể phân loại trigger theo loại câu lệnh kích hoạt như: INSERT, UPDATE, DELETE. Trong các trigger thuộc một trong ba loại lệnh: INSERT, UPDATE, DELETE. Trigger UPDATE cần phải chỉ rõ thêm tên cột dữ liệu kích hoạt trigger mỗi khi giá trị dữ liệu trong đó bị thay đổi. Bên trong Trigger có thể có chứa các lệnh thao tác dữ liệu. Do đó, cần phải tránh trường hợp lặp lại theo kiểu đệ quy.

Một cách khác ta cũng có thể phân loại trigger theo số lần kích hoạt. theo đó sẽ có 02 loại trigger:

- ③ **Trigger mức lệnh:** Trigger được kích hoạt mỗi khi thực hiện câu lệnh.
- ③ **Trigger mức dòng lệnh:** Trigger được kích nhiều lần ứng với mỗi dòng dữ liệu chịu ảnh hưởng bởi thao tác thực hiện lệnh.



Hình vẽ 15. Thứ tự thực hiện trigger

### 15.1.2. Lệnh tạo trigger

Ta có thể tạo trigger thông qua lệnh script.

Cú pháp lệnh tạo trigger mức câu lệnh:

```
CREATE [OR REPLACE] TRIGGER trigger_name
    timing event1 [OR event2 OR event3]
ON table_name
BEGIN
    PL/SQL Block;
END;
```

Cú pháp lệnh tạo trigger mức dòng dữ liệu:

```
CREATE [OR REPLACE] TRIGGER trigger_name
    timing event1 [OR event2 OR event3]
ON table_name
    [REFERENCING OLD AS old | NEW AS new]
FOR EACH ROW
    [WHEN condition]
BEGIN
    PL/SQL Block;
END;
```

Với:

trigger _name	Tên trigger
timing	Thời gian kích hoạt trigger
event	Loại câu lệnh kích hoạt trigger
referencing	Tên biến thay thế cho giá trị trước và sau thay đổi của dòng dữ liệu đang xử lý
FOR EACH ROW	Trigger thuộc loại tác động trên dòng dữ liệu
WHEN	Chỉ ra một số điều kiện ràng buộc thực hiện trigger
table_name	Tên bảng dữ liệu có gắn trigger trên đó
PL/SQL Block	Nội dung khối lệnh SQL và PL/SQL trong trigger

Ví dụ:

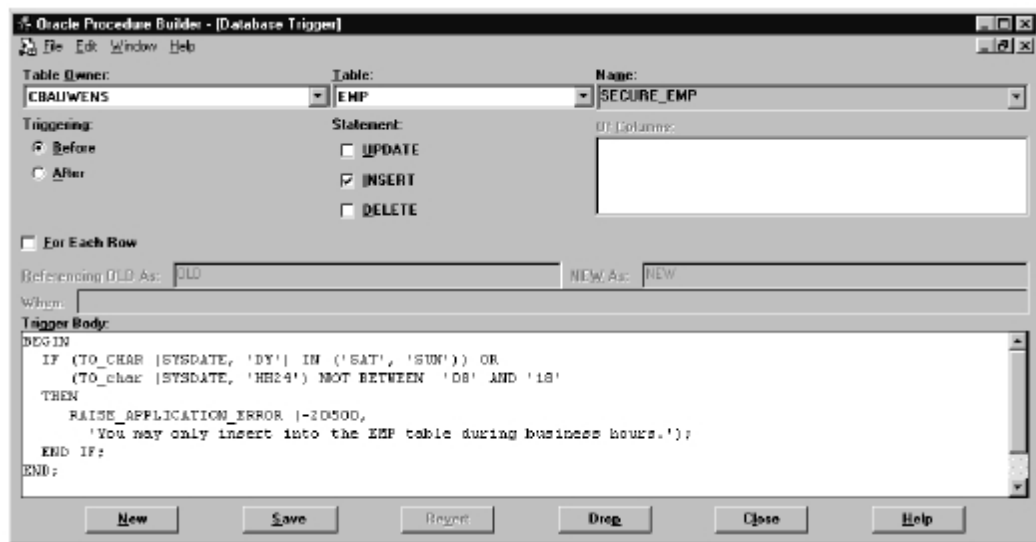
```
CREATE OR REPLACE TRIGGER secure_emp
BEFORE INSERT ON emp
BEGIN
    IF TO_CHAR(sysdate, 'DY') IN ('SAT', 'SUN')
        OR TO_CHAR(sysdate, 'HH24') NOT BETWEEN '08' AND '18'
    THEN
        RAISE_APPLICATION_ERROR (-20500,
            'Thời gian làm việc không phù hợp');
    END IF;
END;
```

```
CREATE OR REPLACE TRIGGER audit_emp_values
AFTER DELETE OR INSERT OR DELETE ON emp FOR
EACH ROW
BEGIN
    INSERT INTO audit_emp_values (user_name, timestamp, id,
        old_last_name, new_last_name, old_title,
        new_title, old_salary, new_salary)
    VALUES (USER, SYSDATE, :old.empno, :old.ename,
        :new.ename, :old.job, :new.job, :old.sal,
        :new.sal);
END;
```

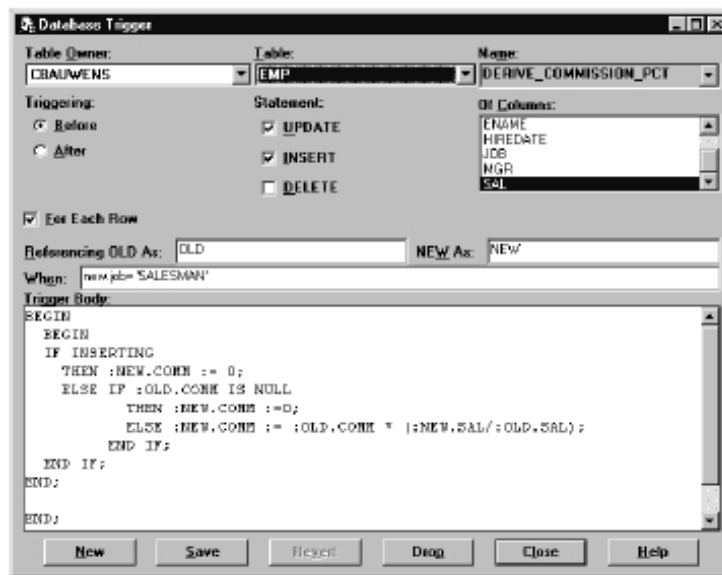
### 15.1.3. Sử dụng Procedure builder để tạo trigger

Ta cũng có thể tạo database trigger thông qua công cụ Procedure builder của Oracle. Ta lần lượt thực hiện theo các bước sau:

1. Kết nối tới database
2. Dịch chuyển tới đối tượng đặt trigger trong phần Object Navigator
3. Chuyển tới phần trigger rồi bấm nút New để tạo mới trigger.
4. Đặt các tùy chọn về thời gian, kiểu,.. cho trigger
5. Soạn thảo nội dung của trigger
6. Lưu giữ trigger



Hình vẽ 16. Tạo trigger bằng công cụ Procedure Builder



Hình vẽ 17. Trigger tác động trên dòng dữ liệu

## 15.2. QUẢN LÝ TRIGGER

### 15.2.1. Phân biệt database trigger

Trigger và thủ tục

Trigger	Thủ tục
Lệnh tạo CREATE TRIGGER	Lệnh tạo CREATE PROCEDURE
Lưu giữ trong Từ điển dữ liệu dưới dạng mã nguồn và dạng p-code	Lưu giữ trong Từ điển dữ liệu dưới dạng mã nguồn và dạng p-code
Được gọi ngầm định	Thực hiện theo lời gọi tường minh
Không cho phép dùng: COMMIT, ROLLBACK, SAVEPOINT	Cho phép dùng: COMMIT, ROLLBACK, SAVEPOINT.

## Database Trigger và Form Trigger

Database Trigger	Form Trigger
Được thực hiện khi có tác động lên database do ứng dụng hoặc do chính các công cụ của Oracle	Được thực hiện chỉ bởi các tác động ngay trên ứng dụng
Được kích hoạt bởi các lệnh SQL	Được kích bởi các sự kiện trên ứng dụng Không phân biệt
Phân biệt hai loại trigger trên câu lệnh và trên dòng dữ liệu	
Tuỳ theo lỗi xảy ra, trigger có thể gây ra rollback câu lệnh	Tuỳ theo lỗi xảy ra, có thể rollback toàn bộ giao dịch

Database Trigger được kích hoạt độc lập với các Form Trigger

### 15.2.2. Thay đổi trạng thái của database trigger

Cho phép/ không cho phép kích hoạt một database trigger Cú pháp:

```
ALTER TRIGGER trigger_name DISABLE | ENABLE;
```

Với:

```
trigger_name      Tên trigger;
```

Ví dụ:

```
-- Cho phép trigger được hoạt động
ALTER TRIGGER      check_sal ENABLE;
```

Cho phép/ không cho phép kích hoạt tất cả các database trigger của một bảng Cú pháp:

```
ALTER TABLE      table_name DISABLE | ENABLE ALL TRIGGERS;
```

Với:

```
table_name        Tên bảng;
```

Ví dụ:

```
-- Không cho phép các trigger ứng với bảng emp được hoạt động
ALTER TABLE      emp DISABLE ALL TRIGGERS;
```

Biên dịch lại database trigger

Cú pháp:

```
ALTER TRIGGER trigger_name      COMPILE;
```

Ví dụ:

```
-- Biên dịch lại trigger check_sal sau khi sửa đổi nội dung
ALTER TRIGGER check_sal COMPILE;
```

### 15.2.3. Huỷ bỏ trigger

Sử dụng câu lệnh SQL để huỷ bỏ trigger.



Cú pháp:

```
DROP TRIGGER      trigger_name;
```

Ví dụ:

```
DROP TRIGGER      check_sal;
```

#### 15.2.4. Lưu ý khi sử dụng trigger

Các trường hợp kiểm tra trigger

- ③ Kiểm tra trigger đúng với thao tác dữ liệu như dự định.
- ③ Kiểm tra thực hiện trigger theo đúng như mệnh đề When.
- ③ Kiểm tra ảnh hưởng của trigger đối với các trigger khác.
- ③ Kiểm tra ảnh hưởng của các trigger khác đối với trigger đang xem xét.

Thứ tự thực hiện trigger và các kiểm tra ràng buộc:

1. Thực hiện trigger BEFORE STATEMENT
2. Lặp trên nhiều dòng dữ liệu
  - a. Thực hiện trigger BEFORE ROW
  - b. Thực hiện câu lệnh thao tác dữ liệu và kiểm tra toàn vẹn dữ liệu trên dòng dữ liệu xem xét
  - c. Thực hiện trigger AFTER ROW
3. Thực hiện các phép kiểm tra ràng buộc
4. Thực hiện trigger AFTER STATEMENT.

Các quy tắc ràng buộc đối với trigger:

1. Không được phép sửa đổi dữ liệu trong cột dữ liệu có ràng buộc thuộc loại khoá chính (primary key), khoá ngoài (foreign key) hay duy nhất.
2. Không cho phép đọc dữ liệu từ các bảng đang thao tác.

## PHỤ LỤC

### A - DANH MỤC CÁC HÌNH VẼ

Hình vẽ 1. Minh họa các thành phần logic trong database ...	...6
Hình vẽ 2. Mô hình dữ liệu thực hành...	..7
Hình vẽ 3. Câu lệnh của SQL*Plus ...	.12
Hình vẽ 4. Hạn chế dữ liệu trả về ...	...17
Hình vẽ 5. Cấu trúc hàm SQL ...	...23
Hình vẽ 6. Phân loại hàm SQL ...	..24
Hình vẽ 7. Cấu trúc của Object Navigator...	...84
Hình vẽ 8. Soạn thảo hàm, thủ tục phía Client...	..85
Hình vẽ 9. Soạn thảo hàm, thủ tục, trigger phía Server ...	...86
Hình vẽ 10. Tạo hàm, thủ tục tại Client ...	..86
Hình vẽ 11. Tạo hàm, thủ tục tại Server ...	..87
Hình vẽ 12. Màn hình PL/SQL Interpreter ...	...87
Hình vẽ 13. Các bước thực hiện một thủ tục ...	..89
Hình vẽ 14. Cấu trúc package ...	..93
Hình vẽ 15. Thứ tự thực hiện trigger...	...98
Hình vẽ 16. Tạo trigger bằng công cụ Procedure Builder ...	..100
Hình vẽ 17. Trigger tác động trên dòng dữ liệu...	..100