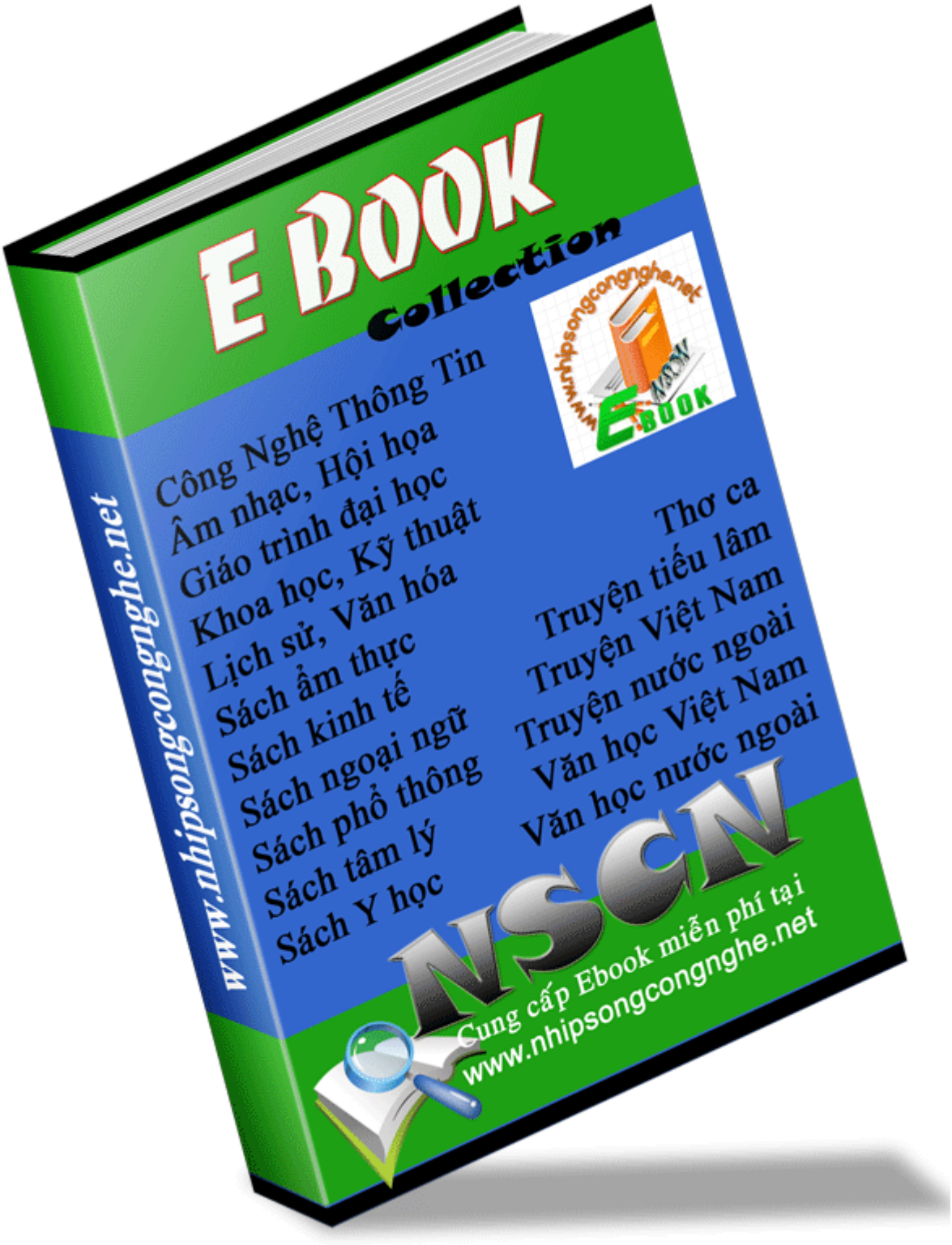


CÔNG TY CỔ PHẦN ĐẦU TƯ PHÁT TRIỂN CÔNG NGHỆ - FPT

Hà Nội, tháng 11 năm 2002.

Đào tạo Oracle cơ bản

Giáo trình SQL và PL/SQL

MỤC LỤC

1 GIỚI THIỆU

1.1 Mục tiêu khoá học

Kết thúc khoá học học viên phải nắm được

- Hiểu được phương pháp, các thành phần, thuật ngữ và thao tác trong CSDL quan hệ
- Tạo được các cấu trúc dữ liệu như table, view dùng SQL
- Ghi, đọc, và cập nhật dữ liệu trong CSDL
- Xây dựng các PL/SQL block dùng Procedure Builder

1.2 Khởi động và thoát khỏi Oracle

1.2.1 Tại Server (Window NT)

SQLDBA cung cấp các dịch vụ quản trị hệ thống, như: tạo lập CSDL, mở - đóng CSDL, tạo và quản lý các USER ... Các bước để khởi động tại Server như sau:

- Khởi động máy chủ
- Bật dịch vụ **OracleServiceXXX** (trong đó XXX là tên của CSDL) bằng cách nhấn vào Start -> Program -> Service -> **OracleServiceXXX** -> Nhấn chuột phải -> Nhấn Start. Chú ý chỉ bật dịch vụ này khi người cài đặt không để chế độ tự động hay khi dịch vụ này chưa được bật.
- Bật dịch vụ **OracleXXXTNSLístener** (trong đó XXX là tên của Database Home) bằng cách nhấn vào Start -> Program -> Service -> **OracleXXXTNSLístener** -> Nhấn chuột phải -> Nhấn Start. Chú ý chỉ bật dịch vụ này khi người cài đặt không để chế độ tự động hay khi dịch vụ này chưa được bật.
- Khi bật xong CSDL đã sẵn sàng để làm việc

Để đóng CSDL cần làm theo các bước ngược lại:

- Tắt dịch vụ **OracleXXXTNSLístener** (trong đó XXX là tên của Database Home) bằng cách nhấn vào Start -> Program -> Service -> **OracleXXXTNSLístener** -> Nhấn chuột phải -> Nhấn Stop.
- Tắt dịch vụ **OracleServiceXXX** (trong đó XXX là tên của CSDL) bằng cách nhấn vào Start -> Program -> Service -> **OracleServiceXXX** -> Nhấn chuột phải -> Nhấn Stop.
- Shutdown máy chủ.

1.2.2 Tại Client (Window 9x)

Các ứng dụng của oracle chạy trong môi trường Windows với giao diện graphic, các ứng dụng thường dùng có SQL*Plus, Oracle Form, Oracle Report, Oracle Designer ... Việc chạy các ứng dụng này hoàn toàn giống như việc chạy các ứng dụng thông thường trong môi trường windows.

Để làm việc với các ứng dụng truy cập CSDL Oracle, người sử dụng (NSD) phải connect vào CSDL. Có hai cách để connect.

Connect NSD/password, ví dụ NSD tên Scott có password là tiger thì

Connect Scott/tiger

Phát lệnh connect với tên NSD, khi đó Oracle sẽ hỏi password

Connect Scott

Enter password: *****

NSD có thể làm việc trong phạm vi cho phép của mình mà Oracle gọi là "khung cảnh" (Schema) của NSD. Mỗi khung cảnh chứa nhiều đối tượng các loại, NSD chỉ có thể tác động lên các đối tượng trong khung cảnh của mình.

Trong các ứng dụng đều có chức năng thoát và tự động disconnect.

Để thực hành phần SQL và PL/SQL gọi ứng dụng SQL* Plus.

1.3 Giới thiệu ngôn ngữ SQL

1.3.1 Lịch sử phát triển của ngôn ngữ SQL

Mô hình cơ sở dữ liệu (CSDL) quan hệ 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 (RDBMS). Mô hình quan hệ gồm các thành phần sau:

- Tập hợp các đối tượng và/hoặc các mối quan hệ
- Tập hợp các xử lý tác động tới các quan hệ
- 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.

1.3.2 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.

1.4 Các khái niệm trong CSDL

Table là cấu trúc lưu trữ cơ bản nhất trong CSDL quan hệ (RDBMS), nó bao gồm 1 hoặc nhiều column và 0 hoặc nhiều row.

Row là tổ hợp những giá trị của Column trong bảng. Một row còn có thể được gọi là 1 record.

Column hiển thị một loại dữ liệu trong bảng, ví dụ tên phòng ban trong bảng phòng ban. Người ta thể hiện nó thông qua tên column và giữ số liệu dưới các kiểu và kích cỡ nhất định.

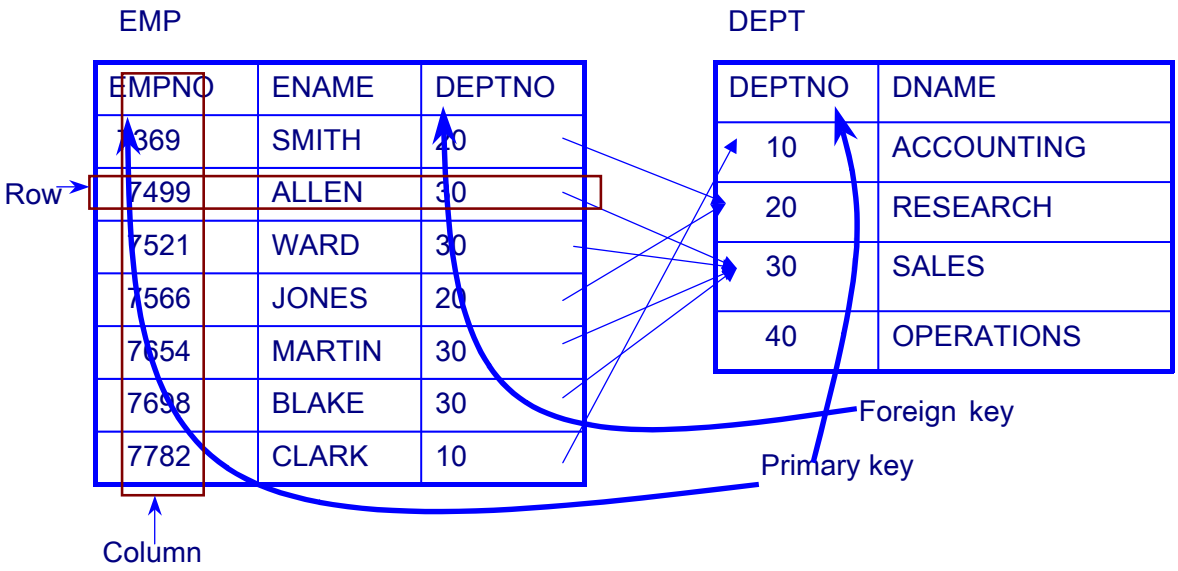
Field là giao của column và row. Field chính là nơi chứa dữ liệu. Nếu không có dữ liệu trong field người ta nói field có gia trị là null.

Primary Key là một column hoặc một tập các column xác định tính duy nhất của các row ở trong bảng. Ví dụ mã phòng ban. Primary Key nhất thiết phải có số liệu.

Foreign Key là một column hoặc một tập các column tham chiếu tới chính bảng đó hoặc một bảng khác. Foreign Key xác định mối quan hệ giữa các bảng.

Constraint là các ràng buộc dữ liệu, ví dụ Foreign Key, Primary Key...

Ví dụ:



1.5 Danh sách rút gọn các đối tượng CSDL

Table là 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 kết sinh giá trị cho các primary key

Index tăng tính thực thi của cáu truy vấn

Synonym tên tương đương của đối tượng

Program unit gồm Procedure, function, package...

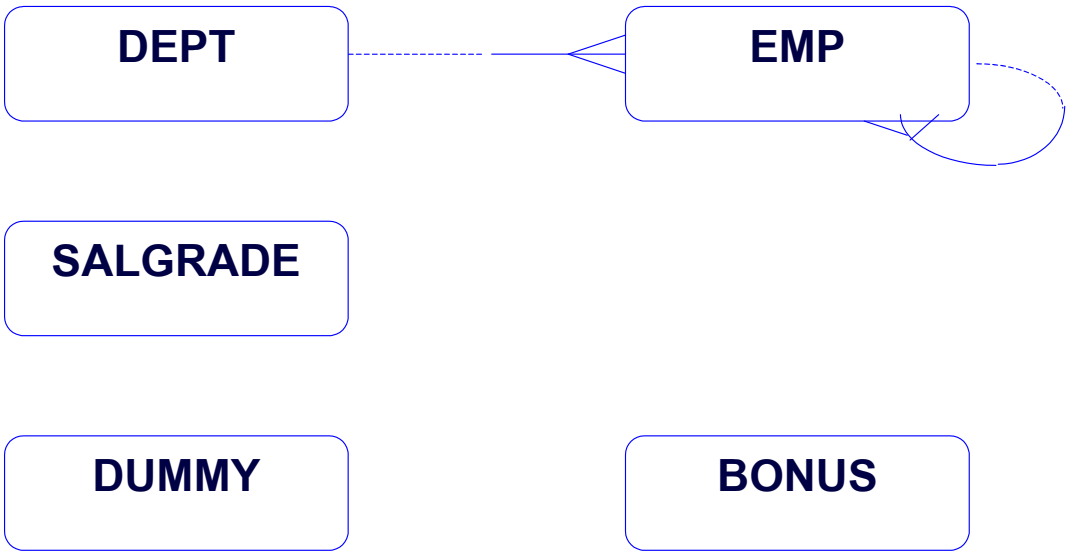
1.6 Các lệnh SQL

Lệnh	Mô tả
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 xoá 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 xoá 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)

1.7 Giới thiệu về ví dụ thực hành

1.7.1 Mô hình quan hệ dữ liệu



1.7.2 Mô tả dữ liệu

Tên	Kiểu	Khoá	Giải thích
DEPT			
DEPTNO	NUMBER(2) NOT NULL	PK	Mã phòng ban
DNAME	CHAR(14)		Tên phòng ban
LOC	CHAR(13)		Địa chỉ
SALGRADE			
GRADE	NUMBER	PK	Mức lơng
LOSAL	NUMBER		Giá trị thấp
HISAL	NUMBER		Giá trị cao
EMP			
EMPNO	NUMBER(4) NOT NULL,	PK	Mã nhân viên
ENAME	CHAR(10),		Tên nhân viên
JOB	CHAR(9),		Nghề nghiệp
MGR	NUMBER(4)	FK (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,	FK (DEPT.DEPTNO)	Mã phòng ban

2 LỆNH TRUY VẤN CƠ BẢN

2.1 Lệnh truy vấn cơ bản

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

- **SELECT** trả lời câu hỏi lấy dữ liệu nào? (column, biểu thức...), trong mệnh đề SELECT cần có ít nhất 1 column.
- **FROM** trả lời câu hỏi lấy dữ liệu ở đầu? (table, view...)
- **DISTINCT** chỉ định hiển thị 1 lần các dữ liệu trùng nhau.
- ***** thay cho việc chỉ tên tất cả các column
- **alias** đưa ra nhãn của column hiển thị dữ liệu.

Vd:

```
SELECT * FROM emp;
SELECT empno, ename, deptno, mgr FROM emp;
SELECT DISTINCT job  nghenghiep  FROM emp;
```

2.2 Các thành phần khác của mệnh đề SELECT

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

- Biểu thức toán học
- Column alias
- Các column được ghép chuỗi
- 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. Vd:

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

Column alias

Trong mệnh đề SELECT, column alias là phần nhãn 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 (□ □).

Vd: (ANUAL chính là column alias)

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

Các column được ghép chuỗi

Toán tử ghép chuỗi (||) cho phép các column được nối với nhau thành dạng chuỗi. Có thể có nhiều toán tử ghép chuỗi trong cùng một column alias. Vd:

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

SELECT ename || ' co luong la '  || (sal+250)*12 as "mieu ta"  FROM emp;

Chuỗi đặt trong nháy đơn, bí danh đặt trong nháy kép
```

Literal

Trong mệnh đề SELECT, literal là bất kỳ ký tự nào, biểu thức, hay số nàp mà không phải là column hoặc column alias. Vd:

```
SELECT empno||ename EMPLOYEE, 'WORK IN DEPARTMENT', deptno FROM emp;
```

2.3 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. Vd:

```
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')

Ví dụ: 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;

2.4 Lọc dữ liệu từ các row có cùng giá trị

Một câu lệnh truy vấn có thể trả về các row có cùng gía trị. Vd:

```
SELECT JOB FROM EMP;
SELECT DEPTNO FROM EMP;
SELECT JOB, DEPTNO FROM EMP;
```

Để lọc lấy một giá trị duy nhất người ta dùng mệnh đề **DISTINCT**. Mệnh đề này phải được đặt trước tất cả các column, sau mệnh đề SELECT. Vd:

```
SELECT DISTINCT JOB FROM EMP;
SELECT DISTINCT DEPTNO FROM EMP;
SELECT DISTINCT JOB, DEPTNO FROM EMP;
```

2.5 Hiển thị cấu trúc bảng

Cú pháp

DESC[RIBE] table_name

(lệnh này chỉ chạy được trên sqlplus, không chạy được trên PL/SQL Develop)

Ví dụ:

```
DESC emp;
Name                                  Null?    Type
 ----------------------------------- -------- ----
 EMPNO                                 NOT NULL NUMBER(4)
 ENAME                                          VARCHAR2(10)
 JOB                                            VARCHAR2(9)
 MGR                                            NUMBER(4)
 HIREDATE                                       DATE
 SAL                                            NUMBER(7,2)
 COMM                                           NUMBER(7,2)
 DEPTNO                                         NUMBER(2)
```

Kiểu dữ liệu NOT NULL nghĩa là column nhất định phải có giá trị.

2.6 Các lệnh của công cụ SQL*Plus

2.6.1 Các lệnh soạn thảo

Lệnh	Mô tả
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	Nhày đến dòng n
N text	Thay dòng n bởi đoạn text
0 text	Chèn 1 dòng trước dòng 1

2.6.2 Các lệnh về file

Lệnh	Mô tả
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
STA[RT] filename [.ext]	Chạy các 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

2.6.3 Các lệnh về column

Cú pháp

COLUMN [{column | alias} [option]]

Lệnh	Mô tả
CLE[AR]	Xoá định dạng của column
FOR[MAT] format	Chuyển định dạng của cột dữ liệu
HEA[DING] text	Đặt nhãn co 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
WOR[D_WAPPED]	Giống WAP nhưng từ không bị cắt

Lưu ý: các định dạng hiển thị này được lưu vào bộ nhớ và áp dụng cho mọi cột có tên như vậy, dù chúng ở trong bảng nào

Vd: 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'
```

Vd: Hiển thị định dạng hiện tại của column

```
COLUMN
COLUMN ename
```

Vd: Xoá định dạng hiện tại của column

```
COLUMN ename CLEAR
CLEAR COLUMN
```

Các loại định dạng

Định dạng	Mô tả	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 thi $	$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

2.7 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		10
7698	BLAKE	MANAGER	7839	01-05-1981	2850		30
7782	CLARK	MANAGER	7839	09-06-1981	2450		10
7566	JONES	MANAGER	7839	02-04-1981	2975		20

```
  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
  7902 FORD       ANALYST    7566 03-12-1981  3000                 20
  7369 SMITH      CLERK      7902 17-12-1980   800                 20
  7788 SCOTT      ANALYST    7566 09-12-1982  3000                 20
  7876 ADAMS      CLERK      7788 12-01-1983  1100                 20
  7934 MILLER     CLERK      7782 23-01-1982  1300                 10
```

3. Hiển thị mọi loại nghề nghiệp

```
JOB
---------
ANALYST
CLERK
MANAGER
PRESIDENT
SALESMAN
```

4. Hiển thị tên nhân viên và thu nhập trong một năm (REMUNERATION)

```
ENAME      REMUNERATION
---------- ------------
KING              60000
BLAKE             34200
CLARK             29400
JONES             35700
MARTIN            16400
ALLEN             19500
TURNER            18000
JAMES             11400
WARD              15500
FORD              36000
SMITH              9600
SCOTT             36000
ADAMS             13200
MILLER            15600
14 rows selected.
```

5. Hiển thị theo nội dung dưới đây

```
Who, what and when
--------------------------------------------------------------------------
KING   HAS HELP THE POSITION OF PRESIDENT IN DEPT 10 SINCE 17-11-1981
BLAKE  HAS HELP THE POSITION OF MANAGER   IN DEPT 30 SINCE 01-05-1981
CLARK  HAS HELP THE POSITION OF MANAGER   IN DEPT 10 SINCE 09-06-1981
JONES  HAS HELP THE POSITION OF MANAGER   IN DEPT 20 SINCE 02-04-1981
MARTIN HAS HELP THE POSITION OF SALESMAN  IN DEPT 30 SINCE 28-09-1981
ALLEN  HAS HELP THE POSITION OF SALESMAN  IN DEPT 30 SINCE 20-02-1981
TURNER HAS HELP THE POSITION OF SALESMAN  IN DEPT 30 SINCE 08-09-1981
JAMES  HAS HELP THE POSITION OF CLERK     IN DEPT 30 SINCE 03-12-1981
WARD   HAS HELP THE POSITION OF SALESMAN  IN DEPT 30 SINCE 22-02-1981
FORD   HAS HELP THE POSITION OF ANALYST   IN DEPT 20 SINCE 03-12-1981
SMITH  HAS HELP THE POSITION OF CLERK     IN DEPT 20 SINCE 17-12-1980
SCOTT  HAS HELP THE POSITION OF ANALYST   IN DEPT 20 SINCE 09-12-1982
ADAMS  HAS HELP THE POSITION OF CLERK     IN DEPT 20 SINCE 12-01-1983
MILLER HAS HELP THE POSITION OF CLERK     IN DEPT 10 SINCE 23-01-1982

14 rows selected.
```

6. Hiển thị cấu trúc bảng emp;

7. Thay đổi nhãn và định dạng hiển thị của cột sal và hiredate trong bảng emp;

3 TRUY VẤN DỮ LIỆU CÓ ĐIỀU KIỆN

3.1 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 để ORDER BY mặc định sắp xếp theo thứ tự tăng dần **ASC[ENDING]**

- Số thấp trước
- Ngày nhỏ trước
- 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. Vi dụ:

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

Order nhiều column

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 để 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;
```

3.2 Mệnh đề WHERE

Cú pháp

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

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:

- Tên column
- Toán tử so sánh
- 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 thoả mãn cả 2 điều kiện. Toán tử OR cho phép dữ liệu thoả 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');
```

3.3 Các toán tử

Toán tử so sánh

- **=** : Toán tử bằng hay tương đương
- **!=**, **^=**, **'+**, **<\>** : Toán tử khác hay không tương đương
- **>** : Toán tử lớn hơn
- **<** : Toán tử nhỏ hơn
- **>=** : Toán tử lớn hơn hoặc bằng
- **<=** : Toán tử nhỏ hơn hoặc bằng

Các toán tử logic

- **NOT** : Phủ định mệnh đề
- **AND** : yêu cầu dữ liệu phải thoả mãn cả 2 điều kiện
- **OR** : cho phép dữ liệu thoả mãn 1 trong 2 điều kiện

Các toán tử của SQL

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

[NOT] BETWEEN x AND y

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

```
SELECT * FROM emp WHERE sal BETEEN 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 không có mệnh đề ESCAPE '\')

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 du

SELECT * FROM emp WHERE comm IS NULL ;

3.4 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
```

Sal Between 1000 to 2000

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
```

Order by dname

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
```

Where deptno in (10,20) order by ename asc

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

```
ENAME      JOB
---------- ---------
SMITH      CLERK
ADAMS      CLERK
```

Where upper(job)= upper('clerk') **and** deptno = '20'; (lưu ý vấn đề chữ Hoa-thường)

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

```
ENAME
----------
SMITH
ALLEN
MILLER
```

Where ename like □%TH%□ or ename like □%LL%□

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
```

Where to_char(hiredate) like □%83□

Where hiredate like □%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
```

4 CÁC HÀM ÁP DỤNG CHO 1 DÒNG DỮ LIỆU

4.1 Các hàm số

Đầu vào và đầu ra là các giá trị kiểu số

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

TRUNC(n[,m]) cho giá trị n lấy m chữ số tính từ chấm thập phân = Format(dl, □99999,00□)

CEIL(n) cho số nguyên nhỏ nhất lớn hơn hoặc bằng n =

FLOOR(n) cho số nguyên lớn nhất bằng hoặc nhỏ hơn n =

POWER(m,n) cho lũy thừa bậc n của m =

EXP(n) cho giá trị của en =

SQRT(n) cho căn bậc 2 của n, n>=0 =

SIGN(n) cho dấu của n. =

n<0 có SIGN(n)= -1

n=0 có SIGN(n)= 0

n>0 có SIGN(n)= 1

ABS(n) cho giá trị tuyệt đối =

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

Một số hàm kiểu số tham khảo khác:

LOG(m,n) cho logarit cơ số m của n =

SIN(n)	cosin của n (n tính bằng radian)	=
COS(n)	cho cosin của n (n tính bằng radian)	=
TAN(n)	cotang của n (n tính bằng radian)	=

Ví dụ hàm ROUND(n[,m])

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

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

Ví dụ hàm TRUNC(n[,m])

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

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

Ví dụ hàm CEIL(n)

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

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

Ví dụ hàm FLOOR(n)

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

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

Ví dụ hàm POWER(m,n)

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

       SAL POWER(SAL,2) POWER(SAL,3) POWER(50,5)
---------- ------------ ------------ -----------
      5000     25000000   1.2500E+11   312500000
      2450      6002500   1.4706E+10   312500000
      1300      1690000   2197000000   312500000
```

Ví dụ hàm EXP(n)

```
SELECT EXP(4) FROM DUMMY;

    EXP(4)
```

```
----------
  54.59815
```

Ví dụ hàm SQRT(n)

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

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

Ví dụ hàm SIGN(n)

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

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

4.2 Các hàm ký tự

CONCAT(char1, char2) cho kết hợp của 2 chuỗi ký tự, tương tự như sử dụng toán tử ||

INITCAP(char) cho chuỗi với ký tự đầu các từ là ký tự hoa

LOWER(char) cho chuỗi ký tự viết thường (không viết hoa)

LPAD(char1, n [,char2]) 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 từ tính từ trái sang

LTRIM(char1, n [,char2]) bỏ các ký tự trống bên trái

NLS_INITCAP(char) cho chuỗi với ký tự đầu các từ là chữ hoa, các chữ còn lại là chữ thường

REPLACE(char,search_string[,replacement_string]) : thay tất cả các chuỗi search_string có trong chuỗi char bằng chuỗi replacement_string.

RPAD(char1, n [,char2]) Giống LPAD(char1, n [,char2]) nhưng căn phải

RTRIM(char1, n [,char2]) bỏ các ký tự trống bên phải

SOUNDEX(char) cho chuỗi đồng âm của char.

SUBSTR(char, m [,n]) cho chuỗi con của chuỗi char lấy từ vị trí m vế phải n ký tự, nếu không chỉ n thì lấy cho đến cuối chuỗi

TRANSLATE(char, from, to) cho chuỗi trong đó mỗi ký tự trong chuỗi from thay bằng ký tự tương ứng trong chuỗi to, những ký tự trong chuỗi from không có tương ứng trong chuỗi to sẽ bị loại bỏ.

UPPER(char) cho chuỗi chữ hoa của chuỗi char

ASCII(char)	cho ký tự ASCII của byte đầu tiên của chuỗi char
INSTR(char1, char2 [,n[,m]])	tìm vị trí chuỗi char2 trong chuỗi char1 bắt đầu từ vị trí n, lần xuất hiện thứ m.
LENGTH(char)	cho chiều dài của chuỗi char

Ví dụ hàm LOWER(char)

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

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

Ví dụ hàm UPPER(char)

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

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

Ví dụ hàm INITCAP(char)

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

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

Ví dụ hàm CONCAT(char1, char2)

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

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

Ví dụ hàm LPAD(char1, n [,char2])

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

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

Ví dụ hàm RPAD(char1, n [,char2])

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

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

Ví dụ hàm SUBSTR(char, m [,n])

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

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

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

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

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

Ví dụ hàm LTRIM(char1, n [,char2])

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

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

Ví dụ hàm RTRIM(char1, n [,char2])

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

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

Ví dụ hàm SOUNDEX(char)

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

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

Ví dụ hàm LENGTH(char)

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

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

Ví dụ hàm TRANSLATE(char, from, to)

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

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

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

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

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

Ví dụ các hàm lồng nhau:

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

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

4.3 Các hàm ngày

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

ADD_MONTHS(d,n) cho ngày d thêm n tháng.

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

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

Ví dụ hàm MONTH_BETWEEN(d1, d2)

```
SELECT MONTHS_BETWEEN( SYSDATE, HIREDATE),
MONTHS_BETWEEN('01-01-2000','05-10-2000')
```

```
FROM EMP
WHERE MONTHS_BETWEEN( SYSDATE,HIREDATE)>240;

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

Ví dụ hàm ADD_MONTHS(d,n)

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

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

Ví dụ hàm NEXT_DAY(d, char)

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

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

Ví dụ hàm LAST_DAY(d)

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

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

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

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

4.4 Các hàm chuyển đổi kiểu

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

Vd:

```
SELECT To_char (sysdate, 'day, ddth month yyyy') from dummy;

SELECT EMPNO, ENAME, HIREDATE
FROM EMP
WHERE HIREDATE = TO_DATE ('June 4, 1984', 'month dd, yyyy');

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

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

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

Một số khuôn dạng ngày

SCC hoặc CC	thế kỷ; S chỉ ngày BC
YYYY hoặc SYYYY	năm; S chỉ ngày BC
YYY, YY, Y	Chỉ năm với 3,2,1 ký tự số
IYYY, IYY, IY, I	Chỉ năm theo chuẩn ISO
SYEAR, YEAR	Chỉ năm theo cách phát âm của người anh;
Q	Quý trong năm
MM	Giá trị tháng với 2 số (01-12)
MONTH	Tên đầy đủ của tháng theo tiếng anh, đọ dài 9
MON	Tháng với 3 ký tự viến tắt (JAN, FEB...)
WW, W	Tuần trong năm hoặc trong tháng
DDD, DD, D	Ngày trong năm, tháng hoặc tuần
DAY	Chỉ thứ trong tuần
DY	Chỉ thứ trong tuần với 3 ký tự viết tắt
J	Ngày Julian; bắt đầu từ ngày 31/12/4713 trước công nguyên
AM, PM	Chỉ định sáng, chiều
HH, HH12 HH24	Chỉ giờ trong ngày (1-12) hoặc (0-23)
MI	Phút (0-59)
SS	Giây (0-59)
SSSSS	Số giây đến nửa đêm (0-86399)
/ . , -	được tự động thêm khi đặt trong khuôn dạng
"char"	Đoạn ký tự đặt trong nháy đúp được tự động thêm khi đặt trong khuôn dạng
TH	Thêm phần thứ tự (1^{st}, 2^{nd}, 4^{th})
SP	Phát âm số (FOUR với DDSP)

SPTH, THSP — Phát âm và chuyển sang dạng thứ tự (First, second, ...)

RR — Ngày chuyển giao thiên niên kỷ với các năm <1999.

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

Một số khuôn dạng số

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

4.5 Bài tập

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 itn 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 khúng 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.

5 BIẾN RUNTIME

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

```
SELECT * FROM emp
WHERE &Condition

Enter value for condition: sal > 1000

Khi ấy câu lệnh trên tương đương
```

SELECT * FROM emp

WHERE sal > 1000

Ví du 2:

```
Select ename, deptno, job
From emp
Where deptno = &&depno_please;
```

Lệnh Define

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

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

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

```
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 EKP ORDER BY & REM;
```

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 : ******
```

5.1 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
```

6 CÁC HÀM NHÓM ÁP DỤNG CHO LỚN HƠN HOẶC BẰNG 1 DÒNG DỮ LIỆU

6.1 Các hàm tác động trên nhóm

Các hàm tác động trên nhóm các dòng dữ liệu tác động lên một tập hợp các các dòng dữ liệu. Gồm các hàm:

AVG([DISTINCT/ALL] n)	Giá trị trung bình của n,không kể trị null
COUNT([DISTINCT/ALL] expr)	Số 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

SUM([DISTINCT/ALL] n) Tổng của của n không kể trị null

VARIANCE([DISTINCT/ALL] n) Variance của n không 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

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

```
Select AVG(SAL)
FROM EMP:
/Tính mức lương trung bình của toàn bộ nhân viên /

Select MIN(SAL)
FROM EMP
WHERE JOB ='CLERK':
/Tính mức lương thấp nhất của nhân viên làm nghề CLERK /
```

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

```
SELECT JOB, AVG(SAL)
FROM EMP
GROUP BY JOB;
/ Tính mức lương trung bình của từng loại nghề nghiệp/
```

Chú ý: Chỉ được cùng đặt trong mệnh để 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;
```

6.2 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
```

6.3 Bài tập

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.

7 HIỂN THỊ NỘI DUNG DỮ LIỆU TỪ NHIỀU BẢNG

7.1 Mối liên kết tương đương

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

  `Vd: emp.deptno =dept.deptno`
- Các column trùng tên phải được chỉ rõ column đó nằm ở bảng nảo thông qua tên hoặc qua alias. Tên trùng này có thể đặt trong các mệnh đề khác như SELECT, ORDER BY..

```
Vd:
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;
```

7.2 Mối liên kết không tương đương

- Mối liên kết tương đương được thể hiện trong mệnh để WHERE.
- Để liên kết trong mệnh để WHERE phải chỉ rõ tên của các column và mệnh đề được đặt KHÔNG tương đương.

  `Vd: 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..

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

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

7.3 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 mối liên kết cộng không được dùng toán tử IN hoặc OR để nối các điều kiện liên kết khác.

```
Vd:
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
```

7.4 Liên kết của bảng với chính nó

Có thể liên két bảng với chính nó bằng cách đặt alias. Ví du:

```
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
```

7.5 Các toán tử tập hợp

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)

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

```
JOB
---------
CLERK
MANAGER
PRESIDENT
SALESMAN
```

7.6 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 điều kiện mức lương = 3.

```
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 cleck 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 cleck. (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



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            3000 JONES            2975
ADAMS            1100 JONES            2975

13 rows selected.
```

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.

8 CÁC LỆNH TRUY VẤN LỒNG NHAU

8.1 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
```

Toán tử SOME/ANY/ALL/NOT IN/EXITS

NOT IN :	Không thuộc
ANY và SOME :	So sánh một giá trị với mỗi giá trị trong một danh sách hay trong kết quả trả về của câu hỏi con, phải sau toán tử =
ALL :	So sánh một giá trị với mọi giá trị trong danh sách hay trong kết quả trả về của câu hỏi con.
EXISTS :	Trả về TRUE nếu có tồn tại.

Ví dụ

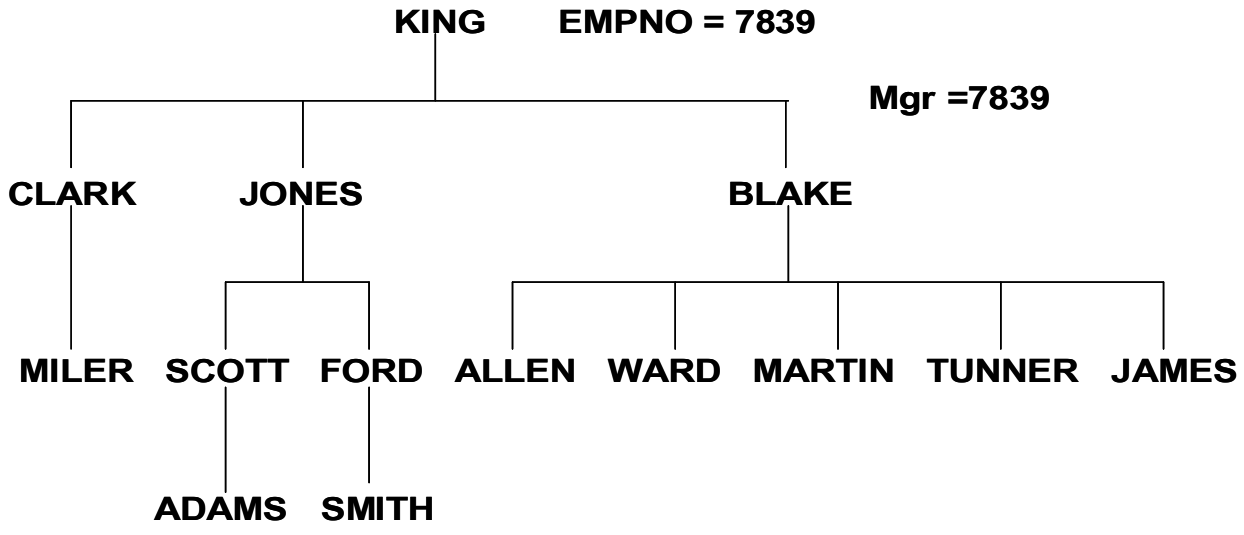
```
SELECT * FROM emp WHERE sal = ANY (SELECT sal FROM emp WHERE deptno=30);

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;

SELECT EMPNO, ENAME, JOB, DEPTNO
FROM EMP E
WHERE          EXISTS (     SELECT EMPNO
                            FROM EMP
                            WHERE EMP.MGR = E.EMPNO);
                            /Tìm những người có nhân viên/
```

8.2 Bài tập

9 CẤU TRÚC HÌNH CÂY

9.1 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.



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

Level

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

9.2 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 lênh SELECT, mỗi mầu 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]]
```

Trong đó:

- START WITH Đặc tả điểm đầu của hình cây. Không thể để column giả level ở mệnh để 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
```

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. Vì 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
```

```
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
```

9.3 Bài tập

1. 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.
```

2. 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
```

3. 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
```

4. 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
```

5. 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
```

6. 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.
```

10 TỔNG KẾT VỀ LỆNH SELECT

Cấu trúc lệnh

```
SELECT [DISTINCT/ALL] [expr [c_ias]]
FROM [table/view/snapshot] [t_alias]
[WHERE condition]
[START WITH condition CONNECT BY condition]
[GROUP BY expr] [HAVING condition]
[UNION/UNION ALL/INTERSET/MINUS select command]
[ORDER BY expr/position [DESC/ASC]]
[FOR UPDATE OF [column]]
[NOTWAIT]
```

Các thành phần trong câu lệnh SELECT

```
DISTINCT            Chỉ chọn 1 cho các row giống nhau trong kết quả
ALL                 Kết xuất cả các row giống nhau
*                   Chọn tất cả các column trong table, view... table.*,
                    view.*, snapshot.*
                    Chọn tất cả các cột trong table, view hay snapshot được
                    chỉ định
expr                Chọn các biểu thức
c_alias             Tên cột trong kết quả kết xuất. table, view, snapshot là
                     Tên table, view hay snapshot
subquery            Câu hỏi select con
t_alias             Tên cho các table
WHERE               Chọn các row thỏa điều kiện trong mệnh đề WHERE
START WITH,CONNECT BY
                    Chọn các dòng trong thứ tự thừa kế
GROUP BY            Nhóm các dòng có expr giống nhau
HAVING              Chọn những nhóm thỏa điều kiện mệnh đề HAVING
UNION, UNION ALL, INTERSET, MINUS    Cho kết quả kết hợp các toán tử tập
                    hợp
ORDER BY            Xếp thứ tư các row theo expr hay position (trong danh
                    sách select)
ASC, DESC           Trật tự xuôi (mặc nhiên) hay ngược
FOR UPDATE          Khóa những row được chọn, cho biết bạn có ý định xóa hay
                    cập nhật các row này
NOTWAIT             Trả quyền điều khiển nếu khi muốn lock row đã bị lock
                    bởi user khác.
```

11 TẠO TABLE

11.1 Lệnh tạo bảng

Để tạo một bảng mới dùng lệnh CREATE TABLE, Cú pháp như sau:

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

Trong đó:

```
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]       : % trống
[PCTUSED integer]       : % sử dụng
[INITRANS integer]      : Số bản ghi khởi tạo
[MAXTRANS integer]      : Số bản ghi lớn nhất
```

```
[TABLESPACE tablespace]  : Chỉ định TABLESAPCE 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 du 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:

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

Tính toán kích thước table (tham khảo):

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

```
BLOCK HEADER  = (FIXED HEADER + VARIABLE TRANSACTION HEADER) + ( TABLE
DIRECTORY + 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
```

11.2 Các quy tắc đặt tên object

- 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
- Tên không chứa dấu nháy (")
- Không phân biệt chữ hoa chữ thường
- Tên phải bắt đầu bằng ký tự chữ trong bộ ký tự của CSDL
- 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 #.
- Tên không được trùng với các từ đã dùng bởi ORACLE (xemphu lục 1)
- Tên không được cách khoảng trống
- 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.
- 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

11.3 Các 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 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 mẩu 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

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

11.4 Kiểu dữ liệu và điều kiện

11.4.1 CHAR

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

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

11.4.2 VARCHAR2

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

11.4.3 VARCHAR

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

11.4.4 NUMBER

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

NUMBER(p, s) trong đó

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

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

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

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

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

Dữ liệu thực	Kiểu	Lưu trữ
7456123.89	NUMBER	7456123.89
7456123.89	NUMBER(9)	7456123
7456123.89	NUMBER(9,2)	7456123.89
7456123.89	NUMBER(9,1)	7456123.8
7456123.89	NUMBER(6)	Không hợp lệ
7456123.8	NUMBER(15,1)	7456123.8
7456123.89	NUMBER(7,-2)	7456100
7456123.89	NUMBER(-7,2)	Không hợp lệ

11.4.5 FLOAT

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

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

11.4.6 LONG

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

Có một số hạn chế khi dùng kiểu LONG

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

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

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

11.4.7 DATE

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

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

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

Tính toán đối với kiểu ngày

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

Ví dụ

SYSDATE+1 ngày hôm sau

SYSDATE-7 cách đây một tuần

SYSDATE+(10/1440) mười phút sau

Ngày Julian: Là giá trị số cho biết số ngày kể từ ngày 1 tháng giêng năm 4712 trước công nguyên.

Ví dụ

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

Cho kết quả

JULIAN

2448623

11.4.8 RAW và LONG RAW

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

Kiểu RAW giống như kiểu VARCHAR2 và kiểu LONG RAW giống kiểu LONG, chỉ khác nhau ở chổ 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.

11.4.9 ROWID

Mỗi mẫu tin trong CSDL có một địa chỉ có kiểu ROWID. ROWID gồm

block.row.file, trong đó

block : chuỗi hệ hexa cho biết block chứa row

row : chuỗi hệ hexa cho biết row trong block

file : chuỗi hệ hexa cho biết database file chứa block

Ví dụ

```
0000000F.0000.0002
```

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

11.4.10 MLSLABEL

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

11.4.11 Chuyển đổi kiểu

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

- Khi INSERT hoặc UPDATE gán giá trị cho cột có kiểu khác, ORACLE sẽ tự động chuyển giá trị sang kiểu của cột.
- 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.
- 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

- TO_NUMBER Chuyển sang kiểu số
- TO_CHAR Chuyển sang kiểu ký tự
- TO_DATE Chuyển sang kiểu ngày

(xem phần tra cứu các hàm và thủ tục)

11.5 Constraint

Các dạng constraint gồm:

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

NULL/NOT NULL: ràng buộc column trống hoặc không trống, trong 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));
```

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 nêu UNIQUE dựa trên một cột. Vd:

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

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 NOTT 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
```

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).

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);
```

11.6 Bài tập

1. Tạo bảng PROJECT với các column được chỉ ra dưới đây, PROJID là promary 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
```

12 CÁC LỆNH DDL KHÁC VÀ DỮ LIỆU TRONG TỪ ĐIỂN DỮ LIỆU

12.1 Chỉnh sửa cấu trúc table

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

```
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:

- Khí dùng mệnh đề MODIFY không thể chuyển tính chất của COLUMN có nội dung là NULL chuyển thành NOT NULL;
- Không thể đưa thêm một cột NOT NUL nếu table đã có số liệu. Phải thêm cột NULL, điền đầy số liệu, sau đó chuyển thành NOT NULL.
- Không thể chuyển đổi kiểu khác nhau nếu column đã chứa số liệu
- 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 DROP PRIMARY KEY;
```

Ví dụ 4

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

12.2 Các lệnh DDL khác

12.2.1 Xóa table

Dùng lệnh DROP TABLE để xoá bảng. Cú pháp như sau:

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

Trong đó:

```
Option CASCADE để xóa tất cả các ràng buộc toàn vẹn liên quan đến table bị
xóa.
```

Ví dụ:

```
DROP TALE emp
```

Khi drop table thì:

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

12.2.2 Giải thích bảng

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';
```

12.2.3 Thay đổi tên object

Dùng lệnh RENAME để thay đổi tên object. Cú pháp như sau:

```
RENAME old TO new
```

Trong đó:

```
Old: Tên cũ
New: tên mới
```

Ví dụ

```
RENAME emp TO employee
```

12.2.4 Xóa dữ liệu của table

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

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

Trong đó:

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

12.3 Dữ liệu 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 bảng.

Có các dạng view là:

- USER_xxx: là những đối tượng thuộc user , ví dụ các bảng được tạo bởi user
- ALL_xxx: là tất cả các đối tượng mà user có quyền truy nhập
- DBA_xxx: tất cả các đối tượng trong database
- V$: Các thực thi của Server.

Ngoài ra còn có các view quan trọng khác là:

- DICTIONARY: Thông tin về toàn bộ các table, view, snapshot trong từ điển dữ liệu
- TABLE_PRIVILEGES: Thông tin về việc gán quyền trên các đối tượng
- 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 mã 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;
```

12.4 Bài tập

1. Thêm column COMMENTS kiểu LONG vào bảng PROJECTS. Thêm column HOURS kiểu NUMBER vào bảng ASSIGNMENTS.

2. Sử dụng view USER_OBJECTS hiển thị tất cả các đối tượng user sở hữu.

3. Thêm ràng buộc duy nhất (UNIQUE) cho 2 column PROJECT_ID và EMPNO của bảng ASSIGNMENTS.

4. Xem các thông tin về các ràng buộc trong USER_CONSTRAINTS.

5. Xem trong USER hiện tại có tất cả bao nhiêu bảng.

13 CÁC LỆNH THAO TÁC DỮ LIỆU KHÁC

13.1 Chèn một row vào table

Để chèn một row vào table dùng lệnh INSERT. Cú pháp như sau:

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

Ví dụ

```
INSERT INTO dept (depno, 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
```

13.2 Chỉnh sửa dữ liệu

Để chỉnh sửa dữ liệu dùng lệnh UPDATE. Cú pháp như sau :

```
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');

Chú thích:
- Cập nhật các nhân viên ở Dallas hoặc Detroit
- Thay DEPTNO của các nhân viên này bằng DEPTNO của Boston
- Thay lương của 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
```

13.3 Xóa dòng

Để xóa dòng dùng lệnh DELETE. Cú pháp như sau:

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

Ví dụ

DELETE FROM emp

```
WHERE deptno = 10;
```

13.4 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.
```

13.5 Lệnh điều khiển giao dịch

Một câu lệnh SQL có thể gồm

- Lệnh DML thao tác dữ liệu
- Lệnh DDL định nghĩa dữ liệu
- 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:

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

Cú pháp các lệnh điều khiển giao dịch:

COMMIT	Kết thúc giao dịch hiện tại, thực hiện các chuyển đổi dữ liệu
SAVEPOINT name	Xác định điểm savepoint của giao dịch
ROLLBACK [TO SAVEPOINT name]	Quay lại dữ liệu ở điểm SAVEPOINT hoặc toàn bộ giao dịch.
SET AUTO[COMMIT] ON/OFF	Tự động COMMIt khi thực hiện các lệnh Insert, update, delete.

```
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;
```

13.6 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.

14 SEQUENCE VÀ INDEX

14.1 Sequence

14.1.1 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 lệu.

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 như sau

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

Trong đó:
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 STRAT WITH 2 MAXVALUE 100;
```

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

CURRVAL Cho giá tri hiện thời của sequence

NEXTVAL Tăng giá tri 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:

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

14.1.2 Xoá và sửa sequence

Sửa bằng lệnh:

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

Xoá bằng lệnh:

```
DROP SEQUENCE sequence_name ;
```

14.2 Index

Index là một cấu trúc cơ sở dữ liệu, được sever 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...]);
```

- Dùng index để query cho nhanh.
- Dùng Index khi mà việc lấy dữ liệu <15% số row trong bảng.
- Index những column nào dùng để nối giữa các bảng lẫn nhau.
- Không nên dùng Index cho các bảng nào chỉ có vài row.
- Primaryvà 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 ;
```

14.3 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

15 TẠO VIEW

15.1 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 vieư, 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;
```

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

- Select
- Insert (insert trên view cũng ảnh hưởng lên table)
- Update (ảnh hưởng lên table)
- Comment

Tuy nhiên có những ràng buộc sau:

- Không thể insert, update trên view, khi query của view chứa các toán tử join, set, distinct, group by, group.
- Không thể nào insert, update trên view, nếu như trong view có dùng with check option.
- 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 colunm hơn table table. Nên insert 1 row vào view, thực chất là insert row đó vào table sẽ không hợp lệ).

- Không thể nào insert trên view, nếu view này có dùng biểu thức decode.
- Những query của view không thể nào tham khảo vào 2 column giả nextval, currval (nextval, currval dùng cho sequence).

15.2 Bài tập

1. Tạo view có hiển thị như sau:

```
select * from aggredates;

DEPTNO    AVERAGE MAXIMUN    MINIMUN        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 ASIGNMENT 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
```

2. 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
```

16 QUYỀN VÀ BẢO MẬT

16.1 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:

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

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

- SYS: Có quyền cao nhất. Mạt khẩu là change_on_install

- 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

- CREATE USER : Tạo user mới
- DROP USER :Xoá user
- DROP ANY TABLE :Xoá table
- 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:

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

Gán quyền

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

Xoá quyền

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

16.2 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:

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

Gán privilege cho Role

Gán Role có các đối tượng

Một số Role hay dùng:

- CONNECT
- RESOURCE

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

16.3 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:

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

Ví dụ: Chúng ta có một table EMPLY trong schema emp_01

Khi lập trình thì phải truy xuất theo emp_01. EMPLY, tên dài như vậy thì đoạn chương trình sẽ dài sẽ dễ lầm lẫn. Nên chúng ta phải dùng synonym

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

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.

17 TỔNG QUAN VỀ PL/SQL VÀ PROCEDURE BUILDER

17.1 Cú pháp lệnh PL/SQL

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

17.2 PL/SQL block

Khối lệnh PL/SQL gồm các thành phần

DECLARE /Không bắt buộc/

    Định nghĩa các biến

BEGIN

  Đoạn lệnh;

EXCEPTION /Không bắt buộc/

    Hành đồng nếu lỗi xuất hiện;

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';
END;
```

17.3 Giới thiệu Procedure builder

Trong Procedure Builder có thể xây dựng các đoạn chương trình PL/SQL như program units, libraries, và database triggers ở cả client-side và server-side. Procedure Builder có một số thành phần sau:

- Object Navigator là phần hiển thị mọi đối tợng trong Procedure Builder's
- Program Unit editor
- PL/SQL Interpreter
- Wizard

Object Navigator. Đặc tính

- Đóng (+), mở (-) các node để xem thông tin
- Có thể connect vào CSDL để xem thông tin về các đối tượng trong CSDL
- Kéo thả để copy đối tượng
- Tìm kiếm đối tượng

Program Unit editor

- Tác dụng: Dùng để soạn thảo đoạn chương trình PL/SQL dễ dàng
- Cách gọi:

  Nhấn đúp vào icon bên trái của program unit. Hoặc

  Nhấn đúp vào nút (+) để tạo Program unit mới
- Tiện ích

  Compile: Dịch

New: Tạo mới

Name: Tìm theo các program unit

Delete: Xoá

Close: Đóng

Help: Trợ giúp

Apply

Revert

PL/SQL Interpreter

- Khi chọn một program unit nào đó. Nội dung của program unit sẽ hiện lên cửa số Interpreter để debug.
- Cửa số interpreter còn có phần đánh lệnh PL/SQL sau dấu nhắc PL/SQL>

Wizard

- Cửa sổ Wizard hiển thị khi tạo một program unit mới bằng công cụ Wizard.
- Công cụ này giúp dễ dàng hơn trong việc xây dựng các program unit.

Tạo một program unit

- Tạo mới bằng cách nhấn vào nút (+) trên thanh toolbar.
- Soạn thảo

  Trợ giúp soạn thảo bằng menu edit.

  Có thể dùng import và export trên menu file để đưa thêm/loại bỏ đoạn text

  Chọn Syntax palete trong menu program để trợ giúp về cú pháp

  Compile để tìm lỗi, thông báo lỗi hiện lên tại dòng cuối của cửa số

Database Trigger

- Tạo mới: Nhấn vào nút (+), chọn loại database Trigger
- Soạn thảo:

  Giống như với Program unit

  Thêm các lựa chọn thuộc tính của trigger.

Tìm vết sửa lỗi các Program Unit

- Tìm vết, sửa lỗi các Program Unit trong PL/SQL interpreter gồm có:

  Toolbar

  Command line

- Tạo các breakpoint (nhấn đúp vào số dòng lệnh) để dừng đoạn chương trình, kiểm tra các biến runtime, dữ liệu...
- Các công cụ trong interpreter

  Step into: Thực hiện tiếp đến dòng lệnh tiếp theo (có thể ngoài program unit)

  Step over: Thực hiện tiếp đến dòng lệnh tiếp theo nhưng chỉ trong Program unit đó.

  Step out: Thực hiện phần còn lại của chương trình

  Go: Thực hiện đến cuối chương trình và dừng lại khi có breakpoint

Reset: Bỏ các breakpoint.

Tổ chức các PL/SQL Program Unit

- Các PL/SQL Program Unit thường được tổ chức lại trong các library (.PLL/.PPL).

  Create: Tạo một library mới

  Open: Mở library

  Save: Ghi lại thay đổi

- Attached library: Sử dụng các program unit trong các Attached library như các hàm mặc định.
- Stored Program Unit: Cất Program Unit thành các Stored Program Unit trong CSDL

18 CÚ PHÁP LẬP TRÌNH

18.1 IF

```
IF condition THEN actions [ELSIF condition THEN actions] [ELSE actions]
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;
```

18.2 LOOP và EXIT

```
LOOP
 actions;
 [EXIT loop_label [WHEN condition]]
END LOOP
```

Ví dụ 1:

```
LOOP
        counter:=counter-1
        INSERT INTO numbered_rows VALUES (counter);
        .....
        IF counter = 10 THEN
```

```
            COMMIT;
            EXIT;
        END IF;
END LOOP;
```

Ví dụ 2:

```
LOOP
        .....
        EXIT WHEN total_sals = 60000;
        ....
END LOOP;
```

18.3 FOR

```
FOR control_variable IN [REVERSE] low_value .. high_value
```

Ví dụ

```
FOR I IN 1..2000
LOOP
        INSERT INTO numbered_rows VALUES (i);
        preserve_i:=i;
        ....
END LOOP;
```

18.4 WHILE

```
WHILE condition
```

Ví dụ

```
WHILE Bill<250
LOOP
        actions;
END LOOP;
```

18.5 GOTO

```
GOTO label
```

Ví dụ

```
BEGIN
<<label1>>
.....
GOTO label1
```

19 CURSOR

19.1 Định nghĩa

Cursor: là kiểu biến có cấu trúc, cho phép ta xử lý dữ liệu gồm nhiều dòng. Số dòng phụ thuộc vào câu lệnh query sau nó. Trong quá trình xử lý cursor như là một con trỏ vị trí của row đang xử lý

Các bước sử dụng biến cursor: Khai báo -> mở -> lấy dữ liệu để xử lý -> đóng

Khai báo

```
cursor  <ten( danh sách biến)> is <câu lệnh select>: Mở vùng dữ liệu cất
trữ thông tin xử lý.
 vd: cursor x is select deptno from dept where deptno=10;
 vd: cursor x(b  number) is select * from dept where deptno>b;
```

Mở cursor

```
open <ten (trị của biến)>
vd: open x;
vd: open x(10);
```

Lấy dữ liệu

```
Fetch <tên cursor> into <tên biến>
Vd:  fetch x into b;
```

Đóng cursor

```
Close <tên cursor>
vd: Close x;
```

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 tri true cho đến khi fetch không còn row nào
%rowcount	: trả lại số row đã được thực hiện bằng lệnh fetch

Ví dụ1:

```
declare
       cursor v_a is select * from emp;
       m v_a%rowtype;
 begin
 open v_a;
    loop
 fetch v_a into m;
 insert into t_thu(empno, ename,job) values (m.empno,m.ename, m.job);
 exit when v_a%notfound;
     end loop;
close v_a;
 end;
```

ví dụ 2:

```
DECLARE
    CURSOR c1 IS SELECT dname, loc FROM dept FOR UPDATE OF loc;
    dept_rec      c1%ROWTYPE;
    sales_count   NUMBER:=0;
    non_sales     NUMBER:=0;
```

```
BEGIN
    OPEN c1;
    LOOP
        FETCH c1 INTO dept_rec;
        EXIT WHEN c1%NOTFOUND;
        IF dept_rec.dname = 'SALES' AND dept_rec.loc!='DALLAS'
        THEN
            UPDATE dept SET loc='DALLAS' WHERE CURRENT OF c1;
            sales_count:=sales_count+1;
        ELSIF
            dept rec.dname!='SALES' AND dept_rec.loc!='NEWYORK'
            THEN
            UPDATE dept SET loc ='NEWYORK' WHERE CURRENT OF c1;
            non_sales:=non_sales+1;
        END IF;
    END LOOP;
    CLOSE c1;
    INSERT INTO counts (sales_set, non_sales_set)
        VALUES (sales_count, non_sales);
    COMMIT;
END;
```

19.2 Kiểu dữ liệu Table và Record

Kiểu dữ liệu Table

Cú pháp:

```
TYPE type_name IS TABLE OF datatype
[NOT NULL] INDEX BY BINARY_INTEGER;
var    type_name;
```

Ví dụ

```
TYPE NAME IS TABLE OF EMP.ENAME%TYPE;
First_name   NAME;
Last_name    NAME;
```

Kiểu dữ liệu Record

Cú pháp

```
TYPE type_name IS RECORD OF
(Col1  datatype
[NOT NULL{:=|DEFAULT} expr],
(Col2  datatype
[NOT NULL{:=|DEFAULT} expr]...);

var    type_name;
```

Ví dụ

```
TYPE emp_rec 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 );

Emp_record emp_rec;
```

19.3 Sao kiểu dữ liệu

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.

Để định nghĩa kiểu dữ liệu bản ghi.

```
Var varref%ROWTYPE

Trong đó
Var    : biến bản ghi
Varref : Tên bảng
```

Ví dụ:

```
X emp%ROWTYPE;
```

Để truy nhập đến các trường trong dữ liệu bản ghi dùng giống như trong 1 row. Ví dụ

```
x.empno, x.sal...
```

Để sao kiểu dữ liệu của một biến nào đó.

```
X salgade%TYPE

X sẽ có kiểu dữ liệu giống biến salgrade.
```

19.4 Câu lệnh SELECT... INTO... trong PL/SQL

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];

trong đó:
INTO var1, var2... [cursor_var] để đưa giá trị trong table vào trong các
biến ( có thể là biến cursor ).
```

Ví dụ:

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

19.5 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

- Có phải mức lương lớn hơn 1200

- Tên nhân viên có phải có chứa chữ T

- 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;

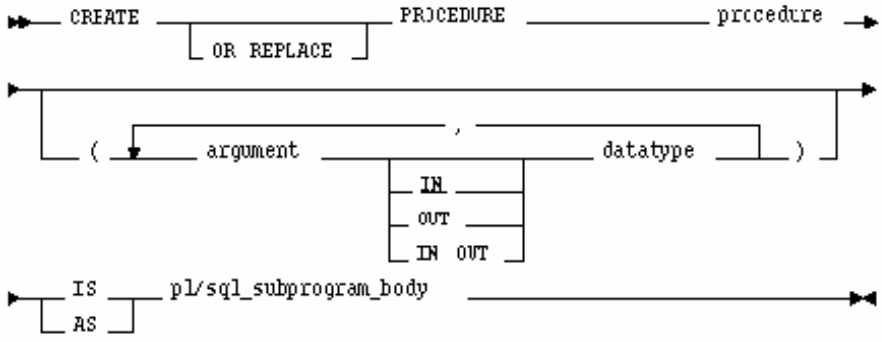
nếu bất kỳ một lần update nào đó có số lượng row >1 thì exit khỏi vòng lặp.

20 PROCEDURE VÀ FUNTION

20.1 Procedure

Là một nhóm các lệnh thực hiện chức năng nào đó nhằm tăng khả năng xử lý, khả năng sử dụng các thủ tục chung, tăng tính bảo mật và an toàn dữ liệu, tiện ích trong phát triển.

Cú pháp:



Procedure : Là tên của procedure được tạo.

Argument : Gồm tên của danh sách các biến và kiểu của nó.

IN : Chỉ định rằng bạn phải đưa trị khi gọi procedure.

OUT : Chỉ ra rằng Procedure sẽ trả lại trị cho biến tới môi trường gọi nó.

IN OUT : Chỉ ra rằng bạn phải gán trị cho argument khi gọi procedure và procedure sẽ trả lại trị argument tới môi trường gọi.

Nếu không ghi IN, OUT hoặc IN OUT thì ngầm định sẽ là IN

Datatype : Là kiểu của argument, ở đây chỉ được khai báo kiểu mà kgông được khai báo các chiều dài argument. Ví dụ không được khai báo argument là VARCHAR2(10) mà phải khai báo là VARCHAR2.

Pl/sql_subprogram_body: Là phần thân của procedure được viết bằng PL/SQL.

Ví dụ:

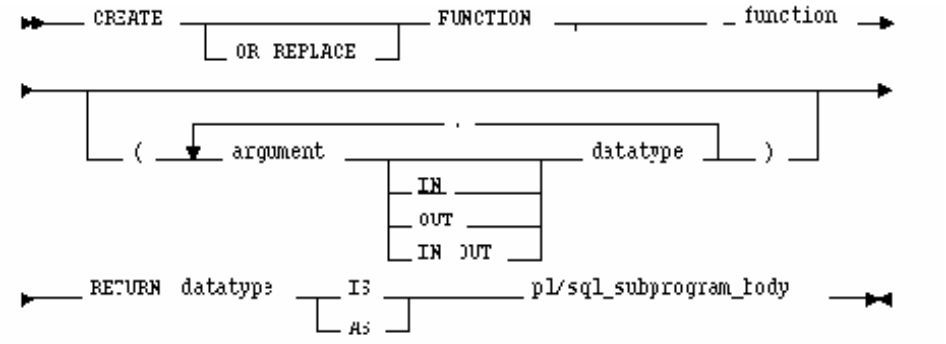
```
CREATE OR REPLACE PROCEDURE INS_DEPT(X NUMBER, Y VARCHAR2) IS
 BEGIN
     INSERT INTO DEPT(DEPTNO,DNAME) VALUES (X,Y);
 END;
```

Muốn thực hiện procedure tại SQL plus thự hiện dùng lệnh execute <ten(danh sách giá trị). Còn trong các thủ tục khác dùng lệnh gọi bình thường

```
SQL> execute ins_dept(55,' New Name');
```

20.2 Function

Cú pháp:



Các tham giống như procedure nhưng khác là sau khi gọi hàm trả lại trị

Ví dụ:

```
create or replace function get_dname( y number) return varchar2 is
 m char(14);
begin
     select dname into m from dept where deptno=y;
     if SQL%notfound then
 m:='Khong thay';
     end if;
    return(rtrim(m));
end;
```

Để gọi hàm get_dname ta gọi trực tiếp hoặc thông qua các phép gán.

Ví dụ:

```
SQL>  select * from dept where dname=get_dname(10);

   DEPTNO     DNAME              LOC
---------     --------------     -------------
       10     ACCOUNTING         NEW YORK

SQL> select get_dname(20) from dual;

GET_DNAME(20)
--------------------------------------------------------------
```

```
RESEARCH
```

20.3 Bài tập

1. Viết hàm lấy tên giám đốc theo biến empno được nhập vào,

2. Viết thủ tục nhập thông tin vào bảng message các trường

```
numcol1: Mã phòng
charcol1: tên phòng ban
numcol2: tổng lương của phòng
```

3. Viết thủ tục dùng cursor; lấy số liệu về n người (n là một biến được đưa vào từ man hình) có mức lương cao nhất trong bảng emp đưa vào bảng top_sal với các giá trị tương ứng trong num=empno, name = ename, salary = sal). Bảng top_sal có cấu trúc như sau:

```
NUM     NUMBER(4)
NAME    VARCHAR2(25)
SALARY  NUMBER(11,2)
```

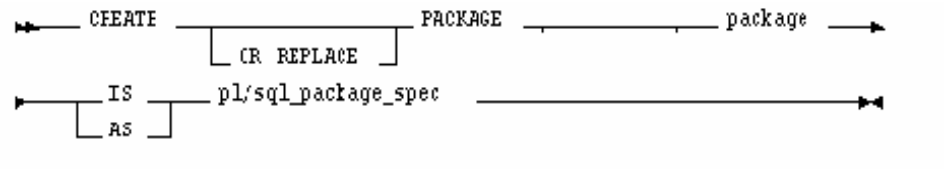
21 PAKAGE

21.1 Package

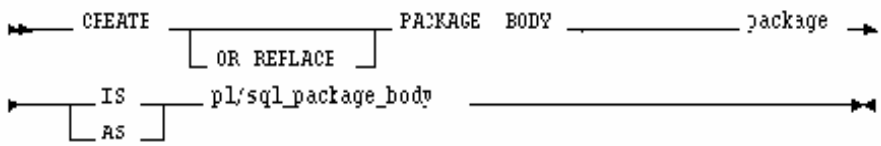
Là tập hợp của các đối tượng gồm các procedure, function, variable, constant, cursor và các exception.

Việc tạo các package cho phép tăng khả năng mềm dẻo, tăng tính bảo mật, tạo sự thuận lợi trong việc quản lý hệ thống đồng thời tăng hiệu suất xử lý của hệ thống.

Để tạo package thực hiện như sau:



Để tạo package body thực hiện như sau:



Với các releases trước đây của PL/SQL việc gọi các functions chỉ có thể được thực hiện bằng các lệnh của procedure, nhưng giờ thì các lời gọi này có thể xuất hiện trong câu lệnh SQL giống như lệnh procedure. Điều này có nghĩa là ta có thể sử dụng các functions giống như các built-in SQL functions. Bằng các mở rộng SQL ta có thể tập hợp phân tích ngay bên trong Oracle Server mà ta không cần lấy dữ liệu vào trong ứng dụng điều này làm tăng tính độc lập của cơ sở dữ liệu.

Tuy nhiên để có thể gọi được từ SQL thì các function phải đảm bảo chắc chắn việc kiểm soát kết quả. Với các standalone functions thì Oracle có thể thực hiện điều này bằng việc kiểm tra function body. Tuy nhiên với body của package là ẩn cho nên các packaged functions ta phải sử dụng pragma RESTRICT_REFERENCES để đảm bảo luật này.

```
Pragma Restrict_references(<Function/procedure Name>, <var1>, ...)

Trong đó:
WNDS: Write no database state
RNDS: Read no database state
WNPS: Write no package state
RNPS: Read no package state
```

Ví dụ:

```
create or replace package vidu is
    function  get_dname( y number) return varchar2;
    Pragma Resctrict_references(get_dname, WNDS, WNPS);
   Procedure ins_dept (x number, y varchar2);
end vidu;


create or replace package body vidu is
 function get_dname( y number) return varchar2 is
 m char(14);
 begin
             select dname into m from dept where deptno=y;
             if SQL%notfound then
       m:='Khong thay';
       end if;
       return(rtrim(m));
 end;
 procedure ins_dept(x number, y varchar2) is
 begin
             insert into dept(deptno,dname) values (x,y);
 end;
end vidu;
```

Để gọi ta thực hiện như sau:

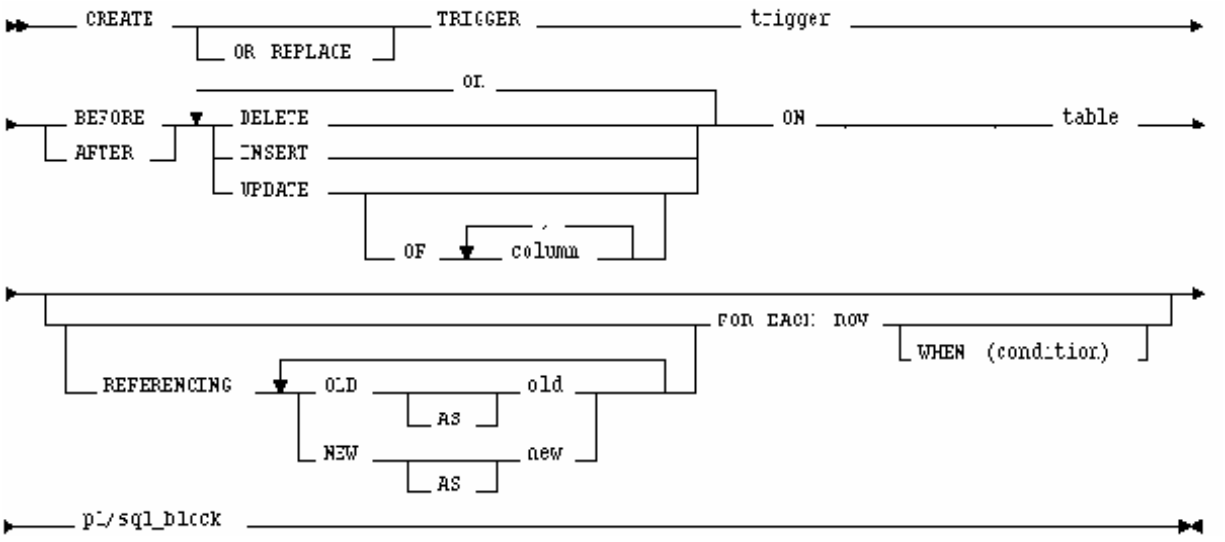
```
SQL> execute vidu.ins_dept(70,'Vi du');
```

22 DATABASE TRIGGER

22.1 Database Trigger

Một Database Trigger được tạo và lưu trữ trong PL/SQL block tương ứng với table. Nó được tự động gọi đến khi có sự truy nhập đến table tương ứng với các hành động định nghĩa.

Để tạo một triger ta gõ theo có pháp sau:



Create or replace : lệnh tạo hoặc tạo lại trigger nếu nó đã tồn tại.

Before : Chỉ ra rằng trigger sẽ được thực hiện trước khi thực hiện lệnh.

Affter : Chỉ ra rằng trigger sẽ được thực hiện sau lệnh gọi tới nó.

Delete, Insert, Update of <column> on <table>: trigger sẽ được gọi khi có các hành động tương ứng trên từng column của bảng.

Referencing : Chỉ tên quan hệ tới các trị cũ (OLD) và mới (NEW) của row .

For each row : Trigger thực hiện tương ứng với mỗi row có điều kiện ứng với mệnh đề trong WHEN.

Pl/sql_block : Là khối lệnh PL/SQL thực hiện các xử lý theo mong muốn.

Ví dụ:

```
create or replace trigger t_dname before insert or update of  dname on dept
for each row when (new.dname is null)
begin
 if (:new.dname is null) then
       :new.dname:='No Name';
 end if;
end ;
```

22.2 Bài tập

1. Viết trigger để khi nhập số liệu vào bảng emp thì nó cũng nhập số liệu vào bảng emp1 với điều kiện cấu trúc bảng emp1 có empno, ename, job, dname

2. Thêm 1 cột vào bảng DEP tên là SUMSAL. Viết trigger để cột SUMSAL luôn chứa tổng lương của phòng ban đó (dữ liệu lấy tương ứng từ bảng emp)

3. Lập 1 bảng tên là BACKUP có cấu trúc giống bảng EMP sao cho mỗi bản ghi trong emp bị xoá sẽ lưu sang backup

4. ứng với các theo tác insert, update, delete trên bảng emp, lưu lại các theo tác đó vào bảng message, với dữ liệu tương ứng charcol1 = tên thao tác, datecol2 = Ngày giờ thực hiện.

23 ERROR HANDING

Error handing là lỗi xuất hiện trong khối lệnh PL/SQL, tất cả các lỗi này sẽ chạy về phần EXCEPTION trong khối lệnh để xử lý.

Khối lệnh PL/SQL gồm các thành phần

DECLARE /Không bắt buộc/

Định nghĩa các biến

BEGIN

Đoạn lệnh;

EXCEPTION /Không bắt buộc/

Hành đồng nếu lỗi xuất hiện;

END;

Cú pháp thực hiện các EXCEPTION;

```
EXCEPTION
 WHEN exception1 [OR exception1. . .] THEN
 Xử lý;
 . . .
 [WHEN exception3 [OR exception4. . .] THEN
 Xử lý;
 . . .]
 [WHEN OTHERS THEN
 Xử lý;
 . . .]
Trong đó:
exception     : tên lỗi n
WHEN OTHERS   : dùng để xử lý các trường hợp lỗi khác
```

Điều kiện kích hoạt exception

Có 2 nhóm exception:

- Các exception của bản thân Oracle như: NO_DATA_FOUND, FOUND, TOO_MANY_ROW ...
- Các exception do người sử dụng khai báo

Các exception hệ thống tự động bị kích hoạt trong các trường hợp nhất định. Các exception người sử dụng định nghĩa phải tự kích hoạt, ví dụ

RAISE exception_identifier;

Một số exception hay dùng của bản thân Oracle:

Tên	Mã lỗi	Mô tả
NO_DATA_FOUND	ORA_01403	Câu lệnh SELECT INTO không trả về row nào
TOO_MANY_ROW	ORA_01422	Câu lệnh SELECT INTO không trả về lớn hơn 1 row
INVALID_CURSOR	ORA_01001	Lỗi xử lý CURSOR
ZERO_DIVIDE	ORA_01476	Lỗi chia cho 0

DUP_VAL_ON_INDEX	ORA_00001	Lỗi giá trị bị trùng lắp trong một UNI QUE INDEX

Ví dụ Xoá những nhân viên trong bảng emp nếu tại phòng nhân viên đó làm việc chỉ có một nhân viên; trong Procedure buider

```
PROCEDURE DELEMP
(V_EMP IN EMP.EMPNO%TYPE) IS
V_ID  EMP.EMPNO%TYPE;

BEGIN
 SELECT EMPNO
 INTO V_ID
 FROM EMP
 WHERE EMPNO = V_EMP;

 DELETE FROM EMP
 WHERE EMPNO = V_EMP;
 COMMIT;

EXCEPTION
 WHEN NO_DATA_FOUND THEN
 ROLLBACK;
 TEXT_IO.PUT_LINE(TO_CHAR(V_EMP)||'KHONG CO');

 WHEN TOO_MANY_ROWS THEN
 ROLLBACK;
 TEXT_IO.PUT_LINE('CO LOI DU LIEU TRONG BANG EMP');

 WHEN OTHERS THEN
 ROLLBACK;
 TEXT_IO.PUT_LINE('CO LOI KHAC TRONG BANG EMP');
END;
```

Gọi chạy delemp(7364);

Các exception do người sử dụng định nghĩa

Khai báo exception

```
identifier EXCEPTION;
```

Ví dụ:

```
DECLARE
        credit_exceeded EXCEPTION;
BEGIN
        IF stock_ordered > credit_limit THEN
                RAISE credit_exceeded;
        END IF
        ....
        EXCEPTION
                WHEN credit_exceeded THEN ....
END;
```

Đặt tên cho các exception hệ thống

Mỗi exception hệ thống được gán một số xác định, có thể đặt tên cho các exception để dễ sử dụng hơn.

```
PRAGMA EXCEPTION_INIT (exception_identifier, number)
```

Ví dụ

```
DECLARE
        fetch_failed EXCEPTION;
        PRAGMA EXCEPTION_INIT (fetch_failed, -1002);
BEGIN
        ....
        EXCEPTION
                WHEN fetch_failed THEN ....
END;
```

23.1 Bài tập

1. Dùng EXCEPTION bắt lỗi chặt hơn cho các bài tập từ phần 19-22.