

CƠ SỞ DỮ LIỆU NGÔN NGỮ SQL

NGUYỄN ĐỨC THUẦN

Bộ Môn Hệ Thống Thông Tin

NH NHA TRANG

Tài liệu tham khảo

- * SQL & PL/SQL Oracle®
- * RELATIONAL DATABASES and SQL PROGRAMMING Christopher Allen
Simon Chatwin
Catherine A. Creary
- * Database System Design/Building I JIPDEC
- * Giáo trình lý thuyết & thực hành ORACLE Trần Tiến Dũng (Chủ biên)

NGÔN NGỮ SQL (Oracle)

Giới thiệu SQL

Ngôn ngữ SQL ?

SQL (Structure Query Language) là ngôn ngữ truy vấn có cấu trúc. SQL là 1 ngôn ngữ chuẩn để thao tác trên dữ liệu lưu trữ trong CSDL quan hệ. SQL bao gồm 3 ngôn ngữ sau:

-**Ngôn ngữ định nghĩa dữ liệu (DDL)** *Data Definition Language*

Nònh nghóá 1 nñoái tööing CSDL nhö 1 teân Table, kieâu döö lieäu,..

-**Ngôn ngữ điều khiển dữ liệu (DCL)** *Data Control Language*

Nieàu khieân quyèän truy xuaát CSDL cuâa ngöööoi söü duäng

-**Ngôn ngữ thao tác dữ liệu (DML)** *Data Manipulation Language*

Thao taíc CSDL baèng caùch : Choiñ (tim kieám), Caäp nhaät, Xoaù döö lieäu ...

Ngôn ngữ SQL (tt)

SQL có thể phân thành 2 mức theo quyền truy xuất:

Mức người s.dụng	Kiểu SQL	Các lệnh mẫu
Quản trị CSDL	DDL	Create table <tên table> Drop table <tên table> Alter table <tên table>
	DCL	Grant <tên quyền> to <tên người sd>
Người sử dụng	DML	Select <tên trường> From <tên table> Insert into <tên table> <ds giá trị> Update <tên table> SET tên_trg =dliệu Delete from <tên table> Where <đk>

Các kiểu dữ liệu trong SQL (oracle)

Tên kiểu	Ý nghĩa
VARCHAR2(size)	Chứa chuỗi TEXT đến 2000 byte
CHAR(size)	Chứa chuỗi TEXT đến 255 byte
NUMBER(p[,s])	Chứa dữ liệu số
DATE	Chứa dữ liệu ngày ('dd-MMM-yy')
RAW	Chứa dữ liệu nhị phân đến 2000 byte
LONG	Chứa dữ liệu TEXT đến 2 giga byte
LONG RAW	Chứa dữ liệu nhị phân đến 2 giga byte
ROWID	Kiểu dữ liệu hỗ trợ mã hàng ảo kết hợp mỗi bảng

Chú ý:

- ☒ Char(n) ≠ varchar2(n)
- ☒ Number(n)
- ☒ Number(n,m)
- ☒ Date : khuôn dạng ‘dd-MMM-yy’, lưu giá trị ngày tháng dưới dạng số thập phân (tính từ ngày 31/12/4713 trước CN)
- ☒ Trong 1 số tài liệu còn kiểu:
 - CLOB: dữ liệu ký tự tối đa 4 GB
 - BLOB: dữ liệu nhị phân tối đa 4 GB
 - BFILE:dữ liệu nhị phân tối đa 4 GB, lưu trữ ở thiết bị nhớ ngoài



Cái phút ban đầu lưu luyến ấy
Nghìn năm chưa dễ mấy ai quên

Ngôn ngữ định nghĩa dữ liệu (DDL)

Tạo Table & Quản lý Table

* *Tạo cấu trúc Table*

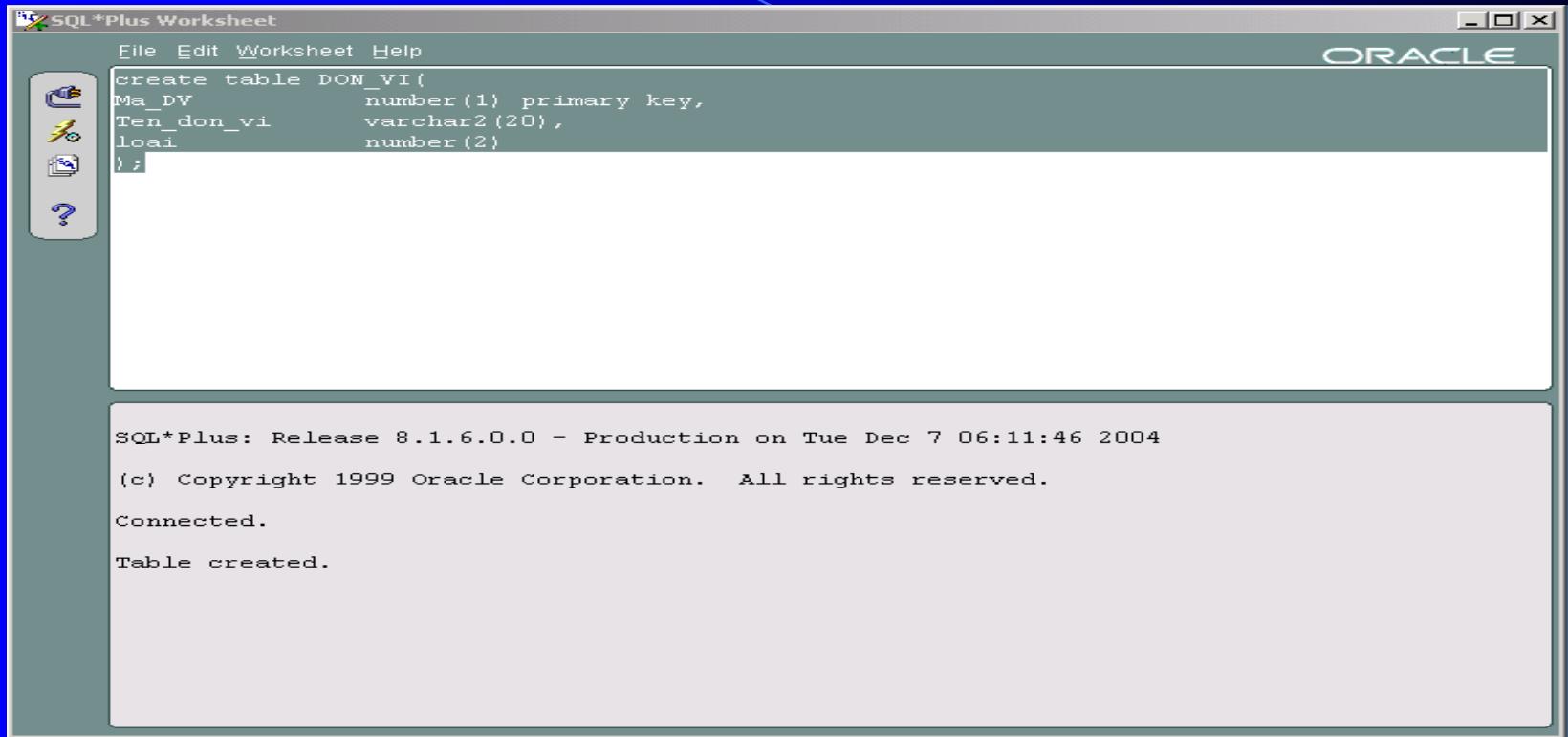
Tạo cấu trúc Table:

Create table <tên table> (<tên_cột_1
kiểu_dữ_lieu_1> [,<tên_cột_i
kiểu_dữ_lieu_i>]);

Ví dụ:

```
create table vidu (
    EMP_ID          number (5) PRIMARY KEY,
    Emp_Name        varchar2(30),
    Emp_Address     varchar2(60),
    Emp_birth       date);
```

* Tạo cấu trúc Table



The screenshot shows the Oracle SQL*Plus Worksheet interface. In the top-left corner, there's a toolbar with icons for Save, Undo, Redo, Print, and Help. The menu bar includes File, Edit, Worksheet, and Help. The title bar says "SQL*Plus Worksheet" and "ORACLE". The main workspace contains the following SQL code:

```
create table DON_VI(
Ma_DV          number(1) primary key,
Ten_don_vi     varchar2(20),
loai           number(2)
);
```

In the bottom pane, the output of the command is displayed:

```
SQL*Plus: Release 8.1.6.0.0 - Production on Tue Dec 7 06:11:46 2004
(c) Copyright 1999 Oracle Corporation. All rights reserved.

Connected.
Table created.
```

Xem các Table đã tạo lập:

*Select * from user_tables;*

* *Tạo cấu trúc Table*

- Qui định tên Table:

- *Phải bắt đầu bởi 1 chữ cái*
- *Chỉ chứa các ký tự : A-Z, a-z, 0-9, _, \$, #*
- *Không trùng với tên Table đã có*
- *Không là từ khoá của Oracle*

Tạo cấu trúc Table

Tạo cấu trúc Table bằng cách sử dụng
Subquery:

Create table <tên table> as SUBquery

Ví dụ:

```
create table vidu as
    select empno, ename, sal*12
    From Emp
    Where deptno = 30;
```

* Thêm 1 cột vào cấu trúc Table

Thêm 1 cột vào cấu trúc Table

Alter table <tên table> ADD (<tên_cột:Kiểu_dữ_liệu_cột>)

Ví dụ:

Alter table VIDU Add (Job VARCHAR2(30);

EMP_ID
Emp_Name
Emp_Address
Emp_birth
job

* *Hiệu chỉnh 1 cột*

Có thể thay đổi kiểu dữ liệu, kích thước, và giá trị mặc định 1 cột:

Alter table <tên table> MODIFY (<tên_cột
kiểu_dữ_liệu_cột>);

Ví dụ:

Alter table VIDU MODIFY (Emp_Name
VARCHAR2(15));

* Xoá 1 cột

Có thể xoá các cột:

Alter table <tên table> DROP (<tên_cột>);

Ví dụ:

Alter table VIDU DROP Emp_Name;

* Thiết lập mục chọn UNUSED

-  Sử dụng mục chọn SET UNUSED để đánh dấu hay nhiều cột không sử dụng.
-  Sử dụng mục chọn DROP UNUSED COLUMNS để xoá các cột đã đánh dấu bởi UNUSED

Alter table <tên table>

SET UNUSED (<tên_cột >);

Alter table <tên table>

DROP UNUSED COLUMNS;

* *Cắt (Truncating) bớt Table*

Lệnh TRUNCATE TABLE:

- Xoá tất cả các dòng từ 1 Table
- Xoá Không gian lưu trữ được dùng bởi Table

Truncate table <tên table> ;

* *Không thể roll back các dòng sau khi xoá*

* *Thể hiện các ràng buộc (constraints)*

- Các ràng buộc là gì?
 - Là các qui tắc có hiệu lực ở mức Table
 - Ràng buộc ngăn cản việc xoá 1 Table nếu có sự phụ thuộc
 - Các ràng buộc sau là có hiệu lực trong Oracle

Ràng buộc được tạo ra :

- *Khi tạo Table*
- *Sau khi Table được tạo*

* Định nghĩa các ràng buộc

```
CREATE TABLE <tên_table> (  
    <tên cột> <Kiểu dữ liệu> [gtrị mặc định]  
    [<ràng_buộc_cột>],  
    .....  
    [<ràng_buộc_Table>][...];
```

Ví dụ:

```
create table vidu (  
    EMP_ID          number (5),  
    Emp_Name        varchar2(30) NOT NULL,  
    Emp_Address     varchar2(60),  
    Emp_birth       date,  
    CONSTRAINT EMP_ID_PK PRIMARY KEY (EMP_ID));
```

* Định nghĩa các ràng buộc

- Có 2 mức

Mức ràng buộc cột:

<điều kiện> [CONSTRAINT constraint_name]
<kiểu ràng buộc>

Mức ràng buộc Table:

<điều kiện>,.....
[CONSTRAINT constraint_name
<kiểu ràng buộc> (<điều kiện>,.....)]

* ràng buộc NOT NULL

- Đảm bảo cột phải chứa giá trị, đn ở mức cột

Ví dụ :

```
create table vidu (
    EMP_ID          number (5),
    Emp_Name        varchar2(30) NOT NULL,
    Emp_Address     varchar2(60),
    Emp_birth       date);
```

* ràng buộc UNIQUE KEY



DEPTNO	DNAME	LOC
10	Accounting	New York
20	Research	Dallas
30	Sales	Chicago
40	Operations	Boston

Không cho phép

50	Sales	Detroit
60		Boston

← (Tên Sales đã có)

← Cho phép

* ràng buộc *UNIQUE KEY*

Định nghĩa có thể trên mức Table hoặc trên mức Cột

Ví dụ :

```
create table Dept(  
deptno          number (2),  
dname           varchar2(14),  
Loc              varchar2(13),  
CONSTRAINT dept_dname_uk UNIQUE (dname));
```

* ràng buộc PRIMARY KEY



DEPTNO	DNAME	LOC
10	Accounting	New York
20	Research	Dallas
30	Sales	Chicago
40	Operations	Boston

20	Sales	Detroit
60		Boston

Không cho phép

← (DEPTNO 20 đã có)

← Không Cho phép

(DEPTNO rỗng)

* ràng buộc PRIMARY KEY

Định nghĩa có thể trên mức Table hoặc trên mức Cột

Ví dụ :

```
create table Dept(  
deptno          number (2),  
dname           varchar2(14),  
Loc              varchar2(13),  
CONSTRAINT dept_dname_uk UNIQUE (dname),  
CONSTRAINT dept_deptno_pk PRIMARY KEY  
(deptno));
```

* ràng buộc FOREIGN KEY

PRIMARY KEY

DEPT



DEPTNO	DNAME	LOC
10	Accounting	New York
20	Research	Dallas
.....		

EMP

EMPNO	ENAME	JOB	...	COMM	DEPTNO
7839	KING	PRESIDENT			10
7698	BLAKE	MANAGER			30

2571	FORD	MANAGER		200	9
7571	FORD	MANAGER		200	20

FOREIGN KEY

Không cho phép
(DEPTNO 9
chưa có trong
DEPT)

Cho phép

* ràng buộc FOREIGN KEY

Định nghĩa có thể trên mức Table hoặc trên mức cột

Ví dụ :

```
create table emp(  
    empno          number (4),  
    ename          varchar2(10) NOT NULL,  
    ....  
    deptno         number(7,2) NOT NULL,  
    CONSTRAINT emp_deptno_fk FOREIGN KEY  
        (deptno) REFERENCES dept (deptno));
```

* ràng buộc FOREIGN KEY

- FOREIGN KEY : Định nghĩa trên cột tại Table con, nơi thiết lập mức ràng buộc
- REFERENCES : Định danh Table, cột trong Table mẹ.
- ON DELETE CASCADE : Khi xoá dữ liệu trong Table mẹ thì dữ liệu trong Table con tương ứng cũng bị xoá.

* ràng buộc CHECK

- Định nghĩa 1 điều kiện mà mỗi dòng phải thoả
- Các biểu thức không cho phép:
 - *Tham chiếu đến CURRVAL, NEXTVAL, LEVEL, và ROWNUM*
 - *Gọi đến SYSDATE, UID, USER và các hàm USERENV*
 - *Truy vấn tham khảo đến các giá trị trong các hàng khác*

... ,deptno NUMBER(2),

CONSTRAINT emp_deptno_ck

CHECK (DEPTNO BETWEEN 10 AND 99)...

* thêm 1 ràng buộc

```
ALTER TABLE <tên_Table>
ADD CONSTRAINT <ràng buộc> kiểu (tên_cột);
```

- ☛ Thêm hoặc xoá, nhưng không hiệu chỉnh được 1 ràng buộc
- ☛ Thêm 1 ràng buộc NOT NULL bằng cách sử dụng mệnh đề MODIFY

* xoá 1 ràng buộc

```
ALTER TABLE <tên_Table>
DROP CONSTRAINT <tên_ràng_buộc>;
```

Ví dụ :

```
ALTER TABLE emp
DROP CONSTRAINT emp_mgr_fk;
```

* *vô hiệu hoá ràng buộc*

ALTER TABLE <tên_Table>

DISABLE CONSTRAINT <tên_ràng_buộc>;

* *Có thể sử dụng mục chọn CASCADE để vô hiệu hóa các ràng buộc toàn vẹn phụ thuộc*

Ví dụ :

ALTER TABLE <tên_Table>

DISABLE CONSTRAINT emp_mgr_fk CASCADE;

* *kích hoạt ràng buộc*

- * *Kích hoạt các ràng buộc toàn vẹn đã vô hiệu hoá trước đó bằng DISABLE*

```
ALTER TABLE <tên_Table>
ENABLE CONSTRAINT <tên_ràng_buộc>;
```

- * *Một chỉ mục UNIQUE hay PRIMARY KEY tự động được tạo lập nếu kích hoạt ràng buộc UNIQUE hay PRIMARY KEY*

* ràng buộc thác nước

- ☞ Sử dụng mệnh đề CASCADE CONSTRAINT theo sau mệnh đề DROP COLUMN
- ☞ Mệnh đề CASCADE CONSTRAINT xoá tất cả các ràng buộc toàn vẹn tham chiếu liên quan đến cột bị xoá.

* Xem các ràng buộc

- ☞ Truy vấn Table USER_CONSTRAINTS để xem tất cả các định nghĩa và tên các ràng buộc.

```
SELECT constraint_name, constraint_type,  
       search_condition  
FROM USER_CONSTRAINTS;
```

* Xoá Table

Drop Table <tên Table>;

Ví dụ:

Drop table vidu;

Ngôn ngữ thao tác dữ liệu (DML)

Thao tác trên Table

* Chèn dữ liệu

INSERT INTO <Tên table> (<DS Cột>)
VALUES (< DS GIÁ TRỊ>);

* Chèn dòng dữ liệu

Ví dụ :

Insert into Don_vi (mdv, ten_dv) values ('01','Day la 1
vi du');

- Giá trị chuỗi ký tự hay ngày phải đặt trong 2 dấu nháy

* Chèn dữ liệu

Chèn các dòng với giá trị Null

* Phương pháp không tường minh: Bỏ qua cột trong danh sách cột

Ví dụ :

```
Insert into Don_vi (mdv) values ('01');
```

* Phương pháp tường minh: Sử dụng từ khoá NULL

Ví dụ :

```
Insert into Don_vi (mdv,Ten_don_vi) values ('01',  
null);
```

* *Chèn dữ liệu: Tạo 1 kịch bản với các dấu nhắc tùy biến*

- * ACCEPT lưu giá trị vào 1 biến
- * PROMPT hiển thị văn bản tùy ý của bạn

Ví dụ:

ACCEPT msnv_id PROMPT ' Dua vao ma so nhan vien: '

ACCEPT Ten_nv_id PROMPT ' Dua vao ten nhan vien: '

Insert into Nhan_vien (msnv,Ten_nv, ngayhd)
values (msnv_id, Ten_nv_id,SYSDATE);

* Sao chép các dòng dữ liệu từ 1 Table khác

Viết câu lệnh INSERT với Subquery

Ví dụ:

Insert into Nhan_vien (msnv,Ten_nv, ngayhd)

Select ms_nv,Ten_nv1, ngayhd

From NV

Where job = 'thay giao';

* Không dùng từ khoá VALUES

* *Thay đổi dữ liệu từ 1 Table*

Câu lệnh UPDATE

Tu sửa các dòng đã có

UPDATE <tên Table>

SET <cột> = <giá trị> [, <cột> = <giá trị>, ...]

[where <điều kiện>];

Ví dụ:

Update Nhan_vien

SET ngayhd= '12-DEC-04'

Where msnv = '12345';

* Cập nhật với subquery nhiều cột

Ví dụ:

Update Nhan_vien

SET (mpx,pcap) =

(Select mpx,pcap

From Nhan_vien

Where msnv ='7899')

Where msnv = '12345';

* Cập nhật các dòng dựa trên cơ sở
` 1 Table khác

* Sử dụng truy vấn con trong câu lệnh UPDATE để cập nhật các hàng trong 1 Table dựa trên cơ sở các giá trị từ 1 Table khác

```
UPDATE employee
SET deptno      =      (SELECT deptno
                          FROM emp
                          WHERE      empno = 7788)
WHERE      job      =      (SELECT      job
                          FROM emp
                          WHERE      empno = 7788);
```

Cập nhật các dòng Lỗi ràng buộc toàn vẹn

```
UPDATE      emp
SET          deptno = 55
WHERE        deptno = 10;
```

```
UPDATE      emp
           *
```

ERROR at line 1:

ORA-02291: integrity constraint (USR.EMP_DEPTNO_FK)
Violate – parent key not found

* Xoá các dòng từ 1 Table

Sử dụng câu lệnh DELETE

```
DELETE [FROM] <tên table>  
[where <điều kiện>];
```

Ví dụ:

```
Delete From Nhan_vien  
Where msnv ='7899');
```

- * Khi không có mệnh đề where, tất cả các dòng của Table được chỉ ra đều bị xoá.

* Xoá các dòng dựa trên 1 Table khác

- Dựa trên truy vấn con trong lệnh DELETE để xoá các dòng từ 1 Table dựa trên cơ sở 1 table khác

```
DELETE FROM employee
```

```
WHERE deptno =
```

```
(SELECT deptno  
FROM dept  
WHERE dname ='SALES');
```

Xoá các dòng : Lỗi ràng buộc toàn vẹn

```
DELETE FROM dept
```

```
WHERE deptno = 10;
```

```
DELETE FROM dept
```

```
*
```

ERROR at line 1:

ORA – 02292 : integrity constraint (USR.EMP_DEPTNO_FK)

Violated – child record found

**Các giao dịch Cơ sở dữ liệu*

- Bao gồm các lệnh sau:
 1. Các lệnh DML làm thay đổi dữ liệu
 2. Một lệnh DDL
 3. Một lệnh DCL

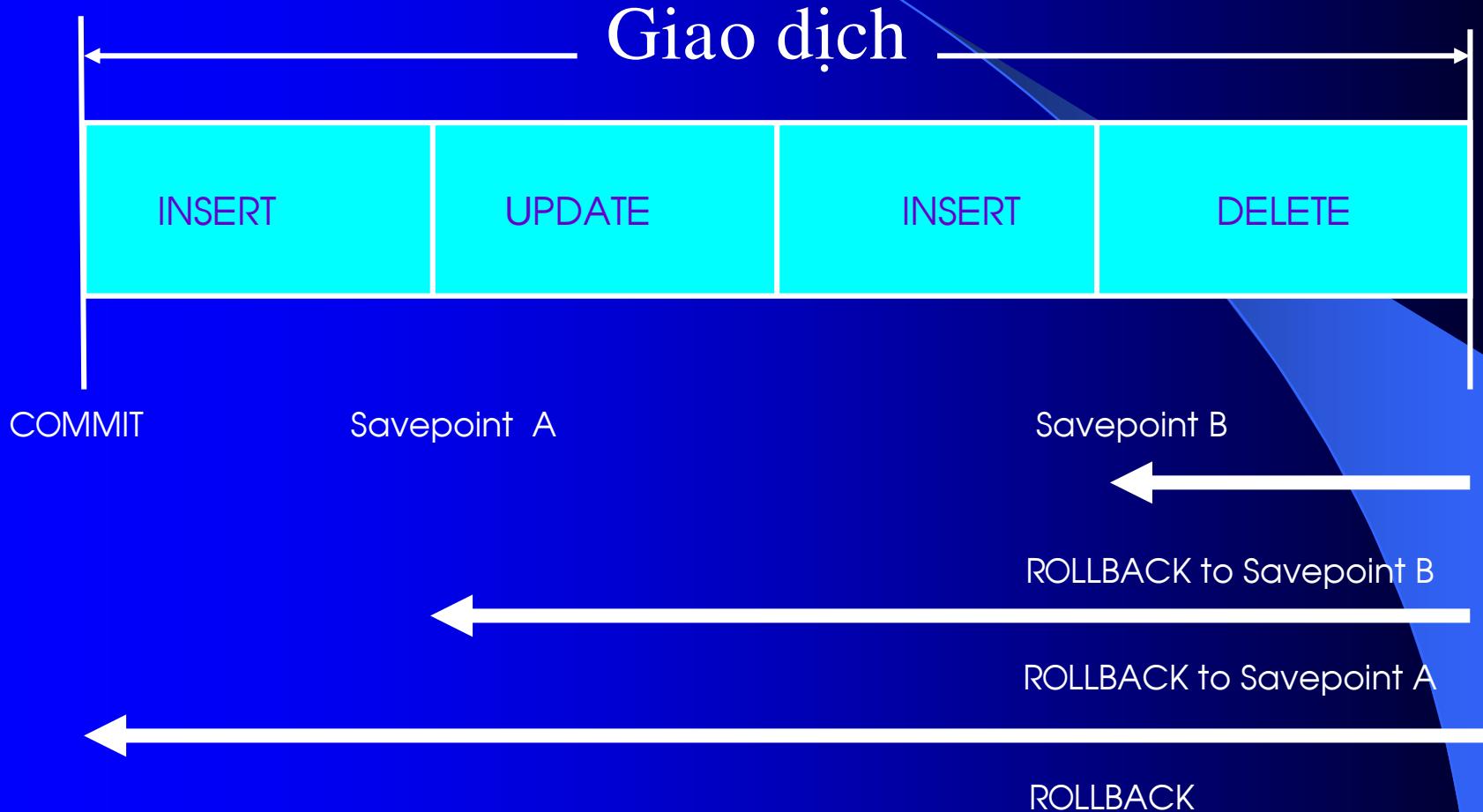
*Các giao dịch Cơ sở dữ liệu

- Bắt đầu khi lệnh thi hành được lần đầu tiên được thực hiện
- Kết thúc với 1 trong những sự kiện sau:
 - -COMMIT hay ROLLBACK được đưa ra
 - -Lệnh DDL hay DCL thực hiện (tự động commit)
 - -Người sử dụng thoát
 - -Hệ thống vỡ (crash)

*Các lệnh COMMIT và ROLLBACK nâng cao

- Đảm bảo ràng buộc toàn vẹn dữ liệu
- Xem xét các thay đổi trước khi thực hiện những thay đổi thường xuyên
- Các phép toán liên quan đến nhóm logic

*Các điều khiển giao dịch



**Cơ chế xử lý giao dịch*

1. Một COMMIT tự động sinh ra trong các tình huống sau:

- Một lệnh DDL được thi hành
- Một lệnh DCL được thi hành
- Thoát bình thường từ SQL*PLUS, không nhất thiết phải sử dụng COMMIT hay ROLLBACK

2. Một ROLLBACK tự động sinh ra khi kết thúc SQL*Plus không bình thường hay hệ thống hỏng.

Trạng thái của dữ liệu trước COMMIT hay ROLLBACK

- Trạng thái trước đó của dữ liệu được phục hồi
- Người sử dụng hiện hành có thể xem lại các kết quả của các thao tác DML bằng cách dùng lệnh SELECT
- Những người sử dụng khác không thể xem các kết quả các lệnh DML
- Những dòng giả lập bị khoá; những người sử dụng khác không thể thay đổi dữ liệu các dòng giả lập

Trạng thái của dữ liệu sau COMMIT

- Dữ liệu thay đổi được lưu thường xuyên trong CSDL
- Trạng thái trước đó của dữ liệu là bị mất
- Tất cả người sử dụng có thể xem các kết quả
- Các khoá trên các dòng giả lập (affected) bị xoá; những dòng này có khả năng những người sử dụng khác xử lý
- Tất cả các điểm lưu bị xoá bỏ

* COMMIT Data

- Thực hiện các thay đổi

UPDATE Emp

SET deptno = 10

WHERE empno = 7782

COMMIT các thay đổi

SQL> COMMIT

Commit complete

Trạng thái của dữ liệu sau ROLLBACK

- Loại bỏ những thay đổi chưa quyết định bằng cách dùng lệnh ROLLBACK
- Dữ liệu thay đổi bị xoá bỏ thay đổi
- Trạng thái trước đó của dữ liệu được phục hồi
- Các khoá trên các dòng giả lập (affected) được loại bỏ

`DELETE FROM Employee;`

`14 rows deleted`

`SQL> ROLLBACK`

`Rollback Complete`

ROLLING BACK thay đổi đến 1 đánh dấu

- Tạo lập 1 đánh dấu trong giao dịch hiện hành bằng cách dùng lệnh SAVEPOINT.
- ROLLBACK đến 1 đánh dấu sử dụng lệnh ROLLBACK TO SAVEPOINT

```
SQL> UPDATE  
SQL> SAVEPOINT update_done;  
Savepoint Created  
SQL> INSERT  
SQL> ROLLBACK TO update_done;  
Rollback complete
```

Các mức lệnh ROLLBACK

- Nếu một lệnh DML đơn lỗi trong quá trình thực hiện, chỉ 1 lệnh được ROLLBACK
- Máy chủ Oracle cài đặt 1 savepoint ẩn.
- Tất cả những thay đổi khác được giữ lại
- Người sử dụng nên kết thúc các giao dịch bằng cách thực hiện các lệnh COMMIT hay ROLLBACK

* *Nhất quán ĐỌC*

- Nhất quán ĐỌC đảm bảo 1 thống nhất dữ liệu tại mọi thời điểm.
- Thay đổi thực hiện bởi 1 người sử dụng không xung đột với những thay đổi của người sử dụng khác
- Nhất quán ĐỌC bảo đảm rằng trên các dữ liệu giống nhau:
 - Những người đọc không chờ những người ghi
 - Những người ghi không chờ những người đọc

* Cài đặt nhất quán ĐỌC



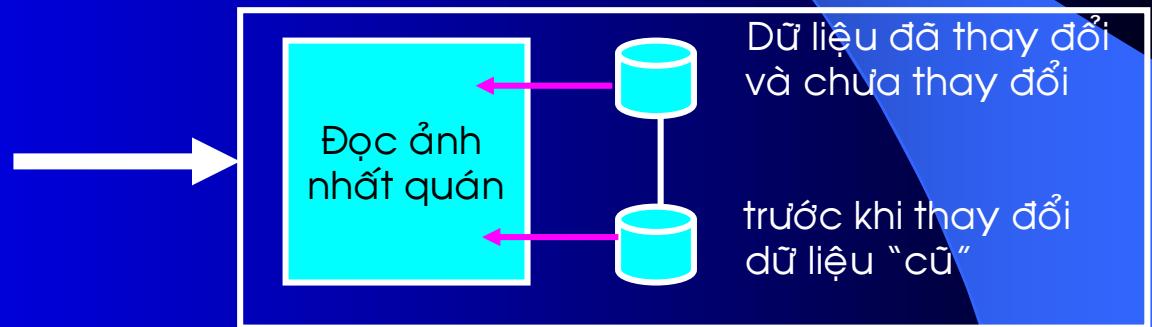
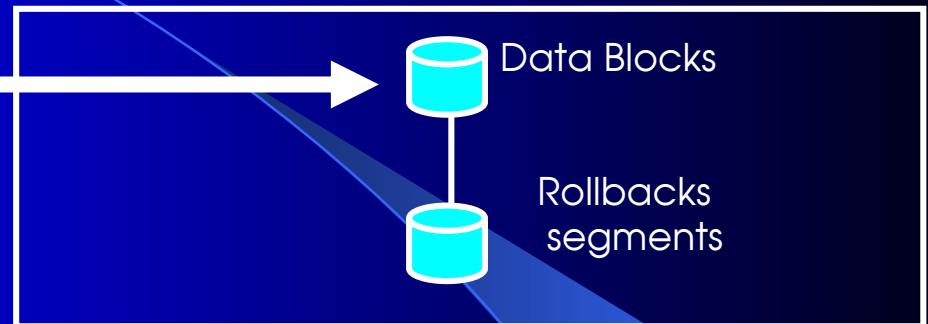
USER A

```
UPDATE emp  
SET sal = 2000  
WHERE ename = 'Scott';
```



USER B

```
SELECT *  
FROM emp;
```



* Khoá (Locking)

Oracle khoá :

- * Ngăn chặn tác động tiêu cực giữa những giao dịch tương tranh
- * Yêu cầu không có tác động người sử dụng
- * Tự động dùng mức thấp nhất của sự hạn chế
- * Nắm giữ khoảng thời gian của giao dịch
- * Có 2 mô hình cơ bản:
 - Loại trừ
 - Chia sẻ

* *Tóm lược*

Lệnh	Mô tả
INSERT	Thêm 1 dòng mới vào Table
UPDATE	Tu sửa các dòng đã có trong Table
DELETE	Xoá các dòng đã có
COMMIT	Thực hiện tất cả những thay đổi
SAVEPOINT	Cho phép 1 rollback đến 1 đánh dấu savepoint
ROLLBACK	Huỷ tất cả những thay đổi còn treo

* Chọn dữ liệu (**Select**)

Chọn cột hiển thị:

```
SELECT (<tên_cột_1, tên_cột2,...>)  
FROM <tên Table> );
```

Ví dụ:

```
Select EMP_ID, Emp_Name,  
FromVIDU;
```

* Biểu thức toán học

- Tạo các biểu thức trên dữ liệu NUMBER và DATE bằng cách sử dụng các phép toán

Phép toán	Ý nghĩa
+	Cộng
-	Trừ
*	Nhân
/	Chia

* Sử dụng các phép toán số học

Ví dụ :

```
Select ename, sal, sal+300  
From VIDU;
```

*Thứ tự ưu tiên các phép toán:

* / - +

```
Select ename, sal, 12*Sal+100  
From VIDU;
```

Sử dụng dấu ngoặc để thể hiện thứ tự ưu tiên

```
Select ename, sal, 12* (Sal+100)  
From VIDU;
```

* Định nghĩa bí danh (alias) 1 cột

- Định nghĩa lại tiêu đề
- Được dùng cho các biểu thức tính toán
- Được viết liền sau tên cột; hoặc sử dụng từ khoá AS giữa tên cột và bí danh
- Yêu cầu có dấu “ “ nếu chứa khoảng trắng, ký tự đặc biệt

* Sử dụng bí danh (alias) cột

① Select ename AS name, sal salary

From VIDU;

②Select ename “Name”, sal*12 “Annual
Salary”;

* *Toán tử ghép (concatenation operation)*

- Ghép các cột hay các chuỗi ký tự vào các cột khác
- Được biểu diễn bằng `||`
- Tạo kết quả là 1 biểu thức ký tự

Ví dụ:

*Select empname || job AS "Employee"
From emp;*

* Chuỗi chữ thêm cho các biểu thức hiển thị

- Một chữ (literal) là 1 ký tự, 1 số, hay 1 ngày tháng được chứa trong danh sách câu lệnh SELECT
- Giá trị chữ Ngày hay ký tự phải chứa trong dấu nháy kép
- Mỗi chuỗi ký tự được xuất 1 lần trong mỗi dòng

* Sử dụng chuỗi chữ thêm cho các biểu thức hiển thị

```
SELECT <tên cột> || <chuỗi chữ> ||  
<tên cột> [AS <bí danh>]
```

Ví dụ:

```
SELECT ename || “is a”||  
Job AS “Employee Detail”
```

* Các dòng trùng lắp

Hiển thị mặc định các truy vấn giá trị trùng nhau của các dòng vẫn hiển thị

Ví dụ : SELECT deptno From emp;

DEPTNO

10
30
10
20
....

* *Hạn chế các dòng trùng lắp*

Sử dụng từ khoá DISTINCT

Ví dụ : SELECT DISTINCT deptno From emp;

DEPTNO

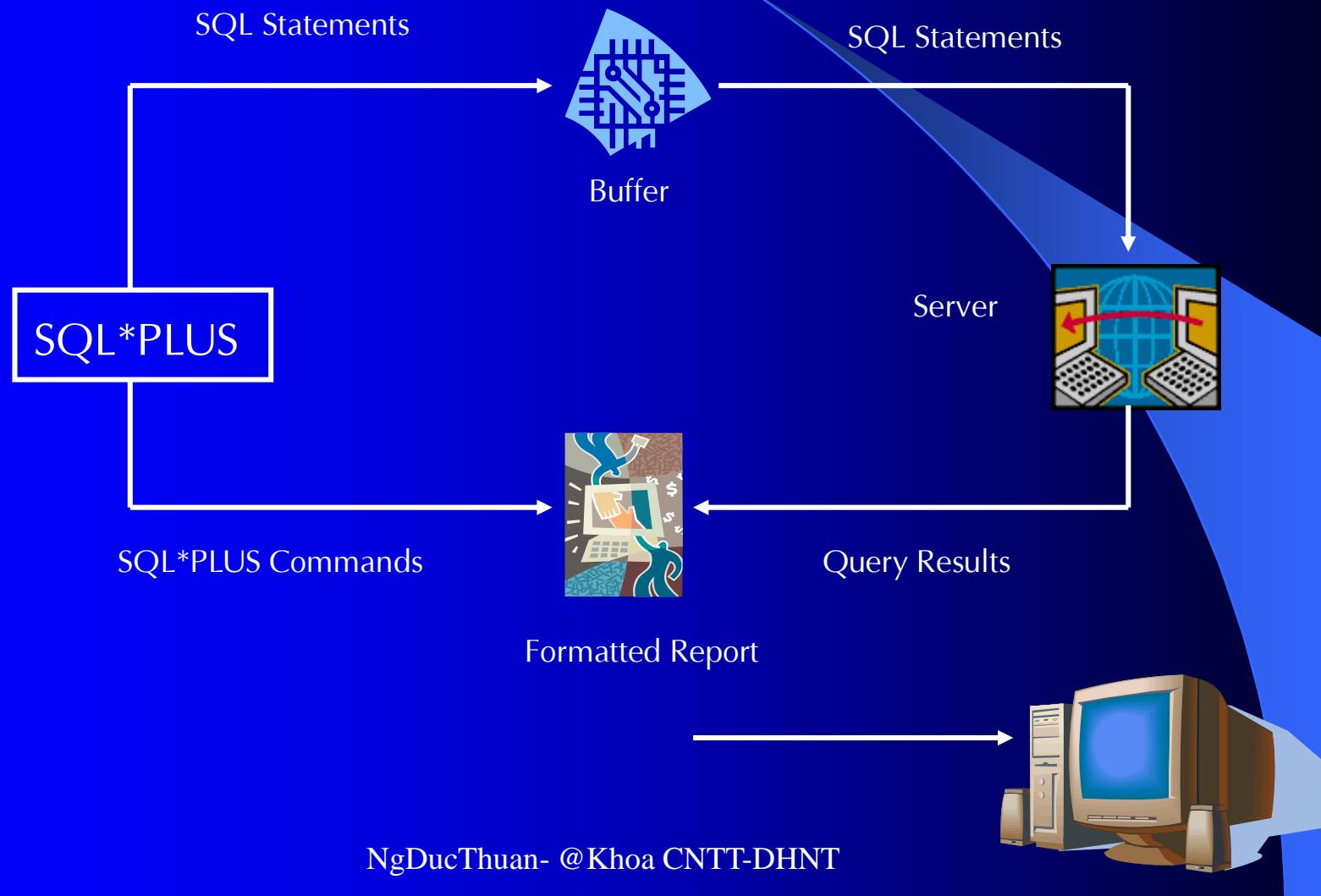
10

30

20

....

* *Tương tác giữa SQL & SQL Plus*



* *Tìm hiểu thêm SQL Plus*

- * Các lệnh SQL*PLUS soạn thảo
- * Các lệnh SQL*PLUS xử lý File:
 - . SAVE <tên File>
 - . GET <tên File>
 - . START <tên File>
 - . @ <tên File>
 - . EDIT <tên File>
 - . SPOOL <tên File>
 - . EXIT

* *Hiển thị cấu trúc Table*

DESC[RIBE] <tên_table>

* *Rút trích và sắp xếp dữ liệu*

SELECT [DISTINCT] (* | <tên_cột> [alias],....)

FROM <tên_Table>

[WHERE <điều kiện >]

*Sử dụng mệnh đề WHERE

- Giá trị chuỗi ký tự và ngày tháng phải để trong dấu nháy kép.
- Giá trị chuỗi phân biệt ký tự hoa, thường. Giá trị ngày tháng phải theo khuôn dạng.
- Giá trị mặc định có dạng DD-MON-YY

Ví dụ:

```
SELECT ename, job, deptno  
FROM emp  
WHERE ename = "THUAN"
```

*Sử dụng mệnh đề WHERE

Toán tử so sánh

Toán tử	Ý nghĩa
=	bằng
>	lớn hơn
>=	lớn hơn hoặc bằng
<	nhỏ hơn
<=	nhỏ hơn hoặc bằng
<>	không bằng

*Sử dụng mệnh đề WHERE

Các toán tử so sánh khác

Toán tử	Ý nghĩa
[NOT] Between ... and ...	Giữa 2 cận giá trị
[NOT] IN (danh sách)	Thuộc danh sách
[NOT] LIKE	Giống khuôn dạng
IS [NOT] NULL	Là 1 giá trị null

*Sử dụng mệnh đề WHERE

Sử dụng toán tử LIKE

- Dùng toán tử LIKE để thực hiện tìm kiếm giá trị theo khuôn dạng
- Điều kiện tìm kiếm có thể là dữ liệu ký tự hay số.
 - % thay thế cho zero hay bất kỳ ký tự nào
 - _ thay thế cho 1 ký tự

Ví dụ:

Select ename

From emp

Where ename LIKE 'S%'

Sử dụng mệnh đề WHERE

Các toán tử Logic

Toán tử	Ý nghĩa
AND	Trả về giá trị TRUE nếu 2 điều kiện thành phần đều TRUE
OR	Trả về giá trị TRUE nếu có 1 trong 2 điều kiện thành phần TRUE
NOT	Trả về giá trị TRUE nếu điều kiện False

Sử dụng mệnh đề WHERE

Thứ tự ưu tiên các phép toán

Thứ tự ưu tiên	Toán tử
1	Tất cả các phép toán so sánh
2	NOT
3	AND
4	OR

* *Ưu tiên các phép toán trong ngoặc trước*

Sắp xếp thứ tự

* Sắp xếp các hàng bằng mệnh đề ORDER BY:

-ASC : Sắp xếp theo thứ tự tăng dần (mặc định)

-DESC: Sắp xếp theo thứ tự giảm dần

Mệnh đề ORDER BY được viết cuối câu lệnh SELECT

Ví dụ : SELECT ename, job, deptno

From emp

Order by deptno DESC;

* Có thể sắp xếp bằng bí danh của cột, hay 1 biểu thức

Sắp xếp thứ tự

Ví dụ:

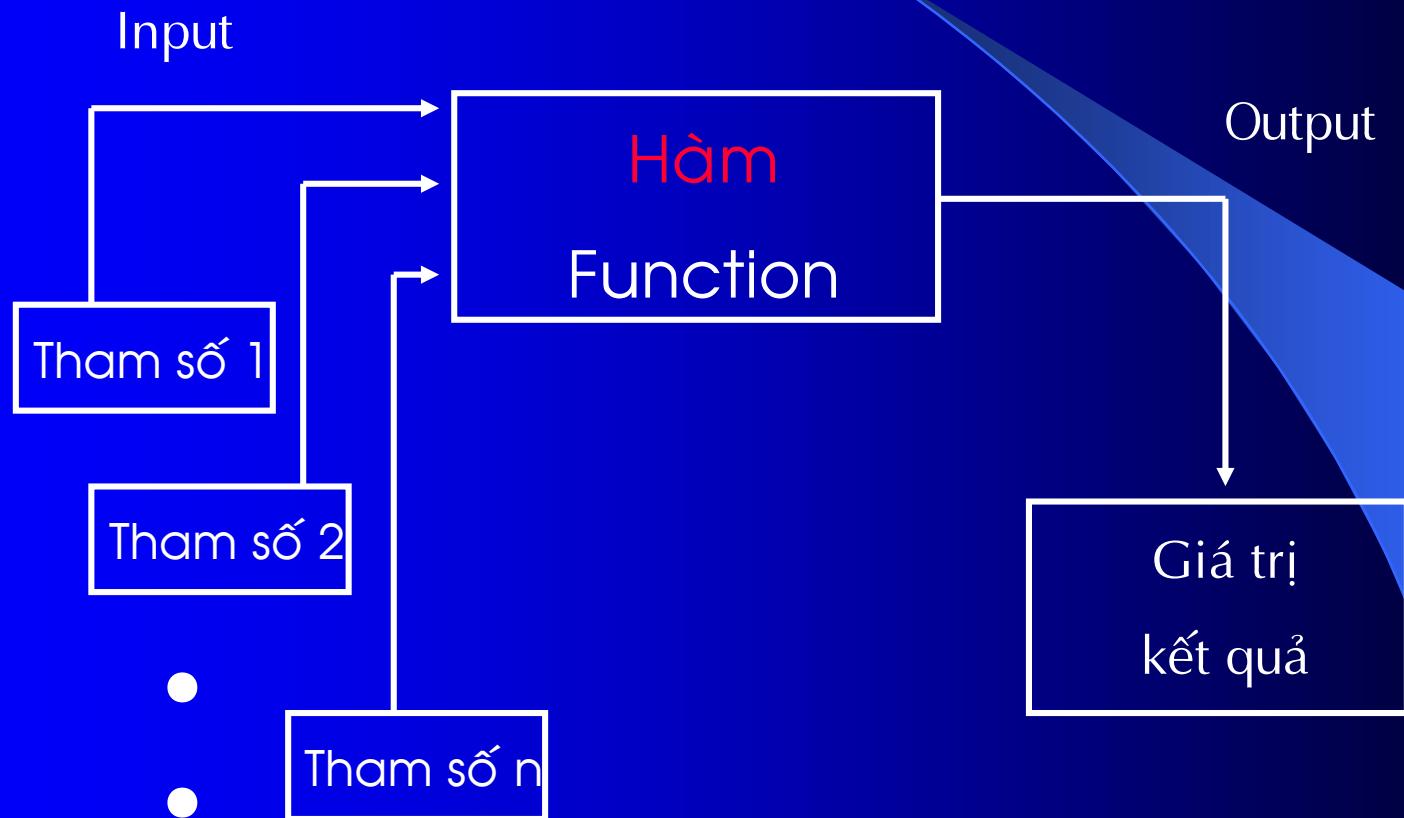
```
SELECT empno, ename, sal*12 annsal  
From emp  
Order by annsal;
```

Có thể sắp xếp thứ tự nhiều cột

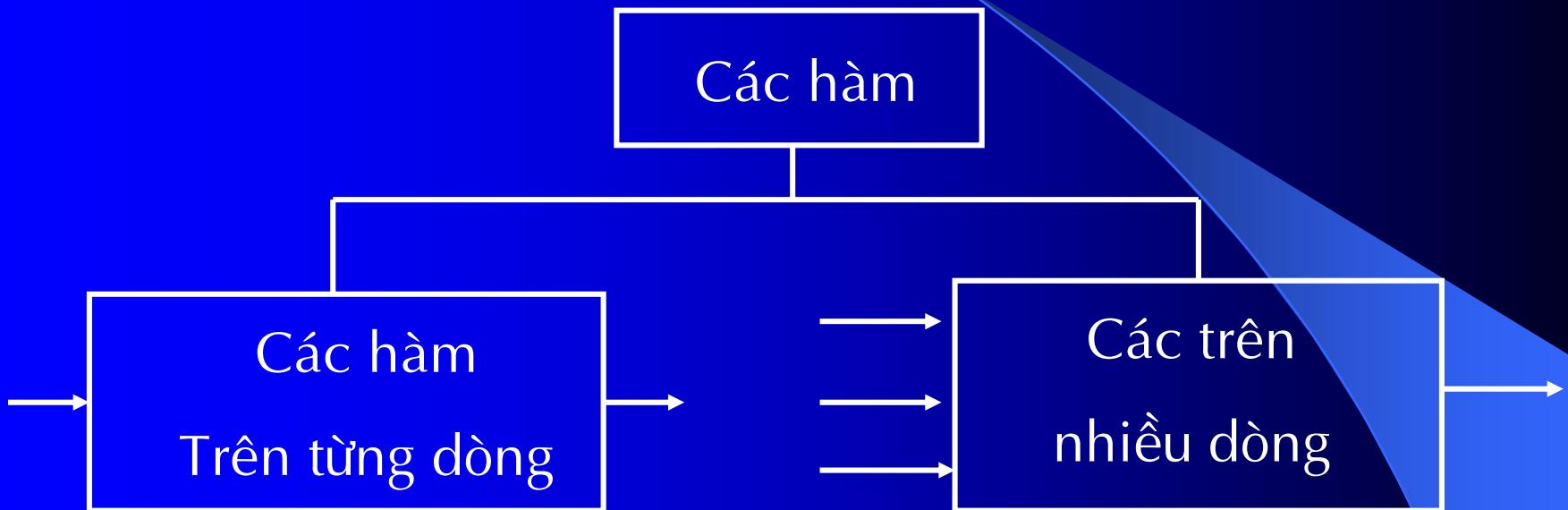
```
SELECT empno,deptno , sal  
From emp  
Order by deptno, sal DESC;
```

Các hàm trên từng dòng

* Các hàm SQL



* Hai loại hàm SQL

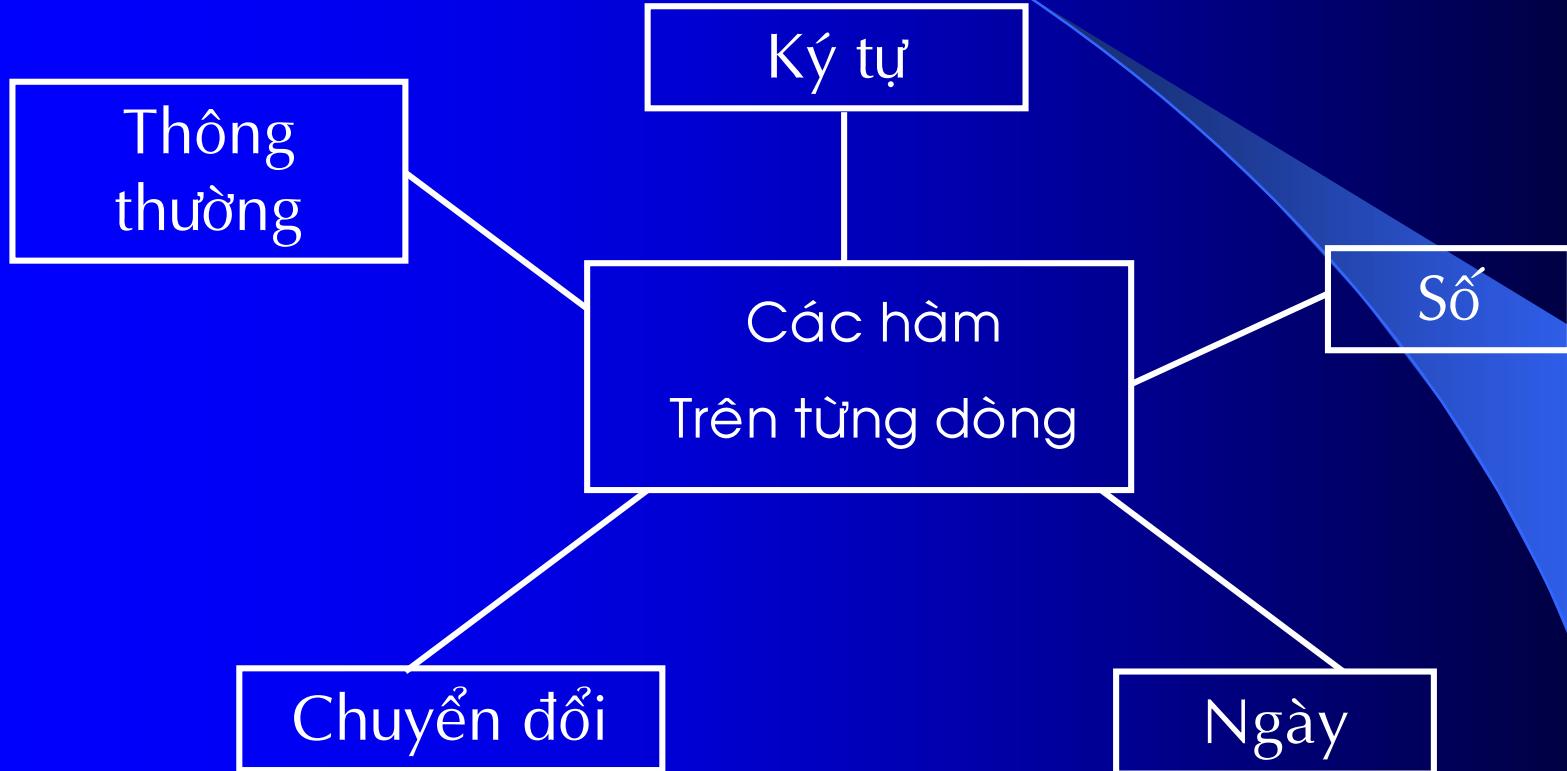


* Các hàm trên từng dòng

- ☞ Thao tác các mục chọn dữ liệu
- ☞ Chấp nhận các tham số và trả về 1 giá trị
- ☞ Trả về 1 giá trị trên mỗi dòng
- ☞ Có thể tu sửa kiểu dữ liệu
- ☞ Có thể lồng nhau

Tên_hàm (cột | biểu_thức, [thsố1], thsố2, ...])

* Các hàm trên từng dòng



* Các hàm ký tự



Các hàm chuyển đổi
ký tự

LOWER

UPPER

INITCAP

Các hàm thao tác
ký tự

CONCAT

SUBSTR

LENGTH

INSTR

LPAD

TRIM

* Các hàm chuyển đổi ký tự

Chuyển đổi chuỗi ký tự

Hàm	Kết quả
LOW('SQL Course')	sql course
UPPER ('SQL Course')	SQL COURSE
INITCAP ('SQL Course')	Sql Course

* Các hàm thao tác ký tự

Thao tác trên chuỗi ký tự

Hàm	Kết quả
CONCAT('Good','String')	GoodString
SUBSTR('String',1,3)	Str
LENGTH('String')	6
INSTR('String','r')	3
LPAD(sal,10,'*')	*****5000
TRIM('S' FROM 'SSMITH')	MITH

* Các hàm số

* ROUND: Làm tròn giá trị theo số thập phân được chỉ định

$$\text{ROUND}(45.926, 2) \longrightarrow 45.93$$

*TRUNC: Cắt giá trị theo số thập phân được chỉ định

$$\text{TRUNC}(45.926, 2) \longrightarrow 45.92$$

*MOD :Trả về phần dư của phép chia

$$\text{MOD}(1600, 300) \longrightarrow 100$$

*Sử dụng hàm ROUND

Ví dụ: SELECT ROUND(45.923,2), ROUND(45.923,0),
ROUND(45.923,-1)FROM DUAL

Kết quả:

ROUND(45.923,2)	45.92
ROUND(45.923,0)	46
ROUND(45.923,-1)	50

*Sử dụng hàm TRUNC

Ví dụ: SELECT TRUNC(45.923,2), TRUNC(45.923,0),
TRUNC(45.923,-1)FROM DUAL

Kết quả:

TRUNC(45.923,2)	45.92
TRUNC(45.923,0)	45
TRUNC(45.923,-1)	40

DUAL là 1 Table ảo chứa kết quả phép toán

**Làm việc với ngày tháng*

- *Oracle lưu trữ ngày tháng trong 1 dạng số (Julian): Thế kỷ, năm, tháng, ngày, giờ, phút, giây.
- *Định dạng ngày tháng mặc định là : DD-MON-YY
- *SYSDATE là hàm trả về ngày, giờ hệ thống
- *DUAL là 1 table giả (ảo) được dùng để xem SYSDATE

* Các phép tính số học với ngày tháng

- * Cộng hay trừ 1 số vào 1 ngày, cho kết quả là giá trị ngày
- * Trừ 2 dữ liệu ngày tháng cho kết quả là 1 số nguyên chỉ số ngày giữa các dữ liệu ngày tháng này.
- * Cộng giờ vào 1 ngày bằng cách chia số giờ cho 24.

* Các hàm ngày tháng

Hàm	Ý nghĩa
MONTHS_BETWEEN	Số tháng giữa 2 ngày tham số
ADD_MONTHS	Cộng số tháng vào ngày tham số
NEXT_DAY	Ngày kế tiếp của ngày chỉ định
LAST_DAY	Ngày trước của ngày chỉ định
ROUND	Làm tròn ngày
TRUNC	Làm tròn bằng cách cắt ngày

* Sử dụng các hàm ngày tháng

MONTHS_BETWEEN('01-SEP-95','11-JAN-94')

→ '19.6774194'

ADD_MONTHS('11-JAN-94',6) → '11-JUN-94'

NEXT_DAY('01-SEP-95', 'FRIDAY') → '08-SEP-95'

LAST_DAY ('01-SEP-95', 'FRIDAY') → '30-SEP-95'

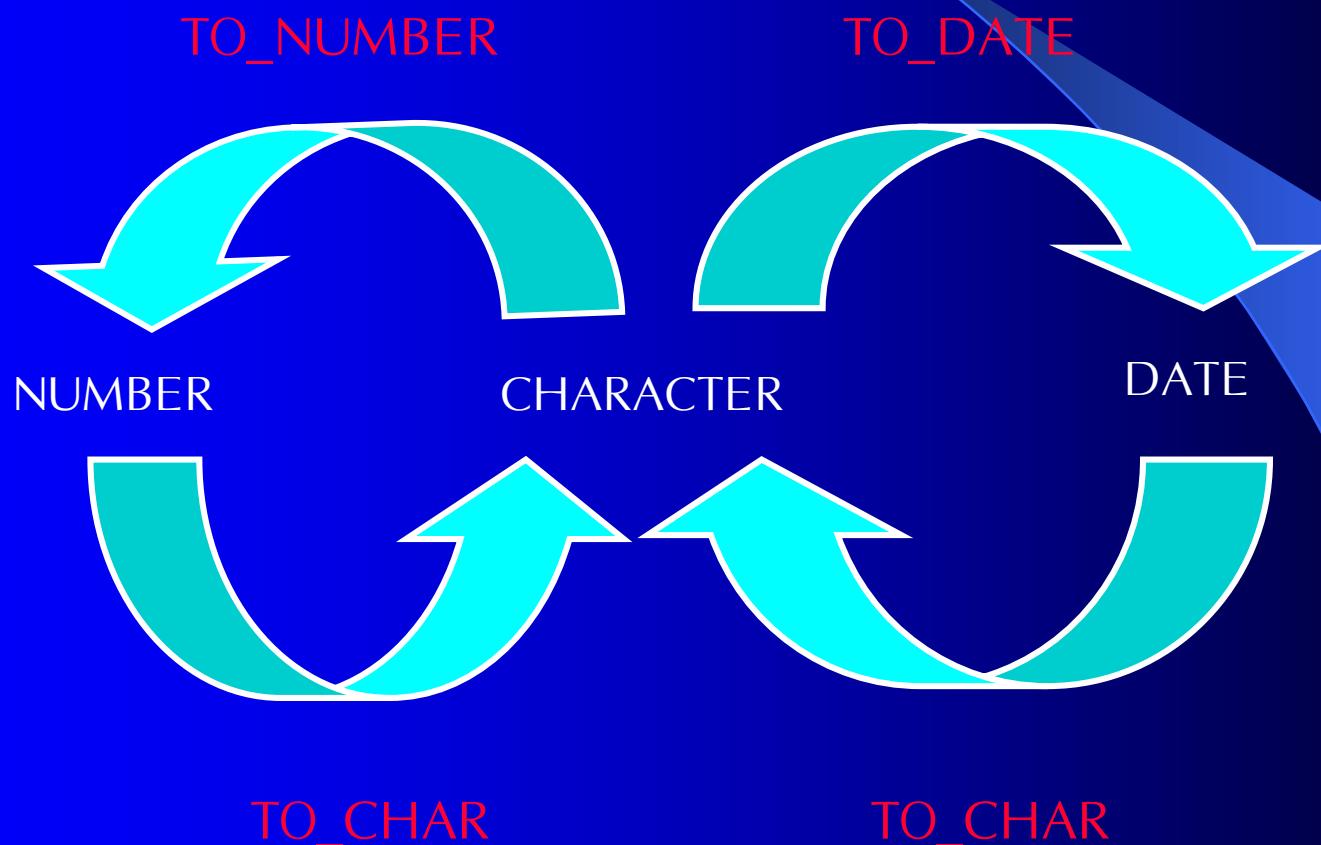
ROUND('25-JUL-95','MONTH') → 01-AUG-95

ROUND('25-JUL-95',' YEAR') → 01-JAN-96

TRUNC('25-JUL-95','MONTH') → 01-JUN-95

TRUNC('25-JUL-95',' YEAR') → 01-JAN-95

*Chuyển đổi kiểu dữ liệu tương minh



*Hàm TO_CHAR với dữ liệu ngày

TO_CHAR (date, 'fmt');

- Khuôn dạng phải viết trong dấu nháy đơn
- Nếu có tiếp đầu ngữ fm để xoá giá trị bắt đầu là 0
- Phân cách các giá trị ngày là 1 dấu ,

Ví dụ:

```
Select ename, TO_CHAR(birth_day, 'fmDD Month YYYY'  
From emp;
```

* Các phần tử khuôn dạng ngày

YYYY	Đầy đủ các ký số của năm
YEAR	Đành vẫn tên năm
MM	Hai ký số giá trị tháng
MONTH	Tên đầy đủ của tháng
DY	Ba ký tự tên ngày trong tuần
DAY	Tên đầy đủ của ngày

* Các phần tử khuôn dạng ngày

Ví dụ:

```
Select to_char(ngay, 'fm DD MM YEAR')  
from ABC;
```

To_char(ngay, fmDD MONTHYEAR)

15 12 TWO THOUSAND FOUR

* Các phần tử khuôn dạng ngày

- Các phần tử thời gian định dạng vị trí thời gian trong ngày

HH24:MI:SS AM 15:45:32 PM

- Thêm chuỗi ký tự bằng cách viết giữa 2 dấu nháy kép

DD "of" MONTH 12 of OCTOBER

* *Hàm TO_CHAR với số*

TO_CHAR(đến, 'fmt')

Hiển thị 1 giá trị số như 1 ký tự

9	Biểu diễn 1 số
0	Hiển thị số 0
\$	Hiển thị dấu \$
L	Hiển thị ký tự tiền tệ
.	In ra 1 dấu chấm thập phân
,	In ra 1 dấu phân cách hàng ngàn

* Hàm TO_CHAR với số

```
Select TO_CHAR(he_so_luong,'$0099.999')  
from ngach_bac_luong;
```

```
TO_CHAR(he_so_luong,'$0099.99')
```

\$003.35

\$004.12

\$002.55

* *Hàm NVL*

Hàm chuyển đổi giá trị Null thành giá trị
được chỉ ra

- Kiểu dữ liệu có thể là ngày, ký tự, và số
- Kiểu dữ liệu phải phù hợp

NVL(comm,0)

NVL(hiredate,'01-JAN-97')

NVL(job,'Giao vien')

* *Hàm DECODE*

Xử lý mệnh đề điều kiện, bằng cách sử dụng tương tự câu lệnh CASE hay IF-THEN-ELSE

DECODE (<cột1>/<bthức1>, <gtrị1>, <kquả 1>,
[<cột i>/<bthức i>, <gtrị i>, <kquả i>]
, <mặc định>)

Select job, sal, DECODE(job, 'Thay', sal*1.1, 'Quan ly, sal*1.15, sal)
Revised_salary from emp

JOB	SAL	Revised_salary
PRESIDENT	5000	5000
MANAGER	2850	3420
MANAGER	2450	2940

* Các hàm có thể lồng nhau

* Hiển thị dữ liệu từ nhiều Table

EMP	ENAME	DEPTNO
<hr/>			
7234	KING	10
7235	BLAKE	30
<hr/>			
7345	MILLER	10

DEPTNO	DNAME	LOC
<hr/>			
10	ACCOUNTING	NEWYORK
30	RESEARCH	DALLAS
<hr/>			
40	OPERATIONS	BOSTON

EMP	DEPTNO	LOC
<hr/>		
7234	10	NEWYORK
7235	30	DALLAS
<hr/>		
7345	10	NEWYORK



* Kết nối ?

```
SELECT <tên table1>.<tên cột>, <tên table2><tên cột>,...  
      From <tên table1>,<tên table2>  
 WHERE <tên table1>.<tên cột1>= <tên table2>.<tên cột 2>
```

- Viết điều kiện kết nối trong mệnh đề WHERE
- Trước tên các cột là tên Table
- Tích DESCASTER được tạo ra khi điều kiện kết nối không hợp lệ hay bị bỏ qua
- Nên sử dụng bí danh để thay thế tên Table
- Có thể kết nối nhiều Table

* Kết nối dữ liệu

EMPNO	ENAME	SAL
<hr/>		
7348	KING	5000
7698	BLAKE	2850
7782	CLARK	2450
7844	ALLEN	1600
7900	JACK	1500
7653	BIN	900
7888	LENA	1200

EMP

GRADE	LOSAL	HISAL
<hr/>		
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
.....		

SALGRADE

* Kết nối không bằng (Non – Equijoins)

```
SELECT e.name, e.sal, a.grade  
From emp e, salgrade a  
Where e.sal  
BETWEEN a.losal AND a.hisal
```

EMP	SAL	GRADE
7653	900	1
7888	1200	2
7900	1500	3
7844	1600	3
.....		

* Kết nối ngoài (Outer Joins)

EMP		DEPT	
ENAME	DEPTNO	DEPTNO	DNAME
KING	10	10	ACCOUNTING
BLAKE	30	20	RESEARCH
CLARK	10	30	SALES
JONES	20	40	OPERATIONS
.....			



Không có nhân viên nào thuộc phòng
OPERATIONS

* Kết nối ngoài (Outer Joins)

- Có thể sử dụng kết nối ngoài để thấy các dòng không thoả điều kiện kết nối
- Toán tử kết nối ngoài là dấu +

```
SELECT Table1.column, Table2.column
```

```
From Table1, Table2
```

```
Where Table1.column(+) = Table2.column;
```

```
SELECT Table1.column, Table2.column
```

```
From Table1, Table2
```

```
Where Table1.column = Table2.column(+);
```

* Kết nối ngoài (Outer Joins)

```
SELECT e.ename, d.deptno, d.name  
From emp e, dept d  
Where e. deptno(+) = d.deptno;  
Order by e.deptno
```

ENAME	DEPTNO	DNAME
KING	10	ACCOUNTING
CLARK	10	ACCOUNTING
.....		
	40	OPERATIONS

* Tự kết nối (Self Joins)

EMP (WORKER)

EMP (MANAGER)

EMPNO	ENAME	MGR	EMPNO	ENAME
7839	KING	7839	7839	KING
7698	BLAKE	7839	7839	KING
7698	CLARK	7839	7839	KING
7566	JONES	7839	7698	BLAKE
7644	MARTIN	7698	7698	BLAKE
7499	ALLEN	7698		

MGR trong bảng WORKER bằng EMPNO trong bảng MANAGER

* Tự kết nối (Self Joins)

```
SELECT worker.ename || 'works for ' ||  
       manager.ename  
  From emp worker, emp manager  
 Where worker.mgr = manager.empno;
```

WORKER.ENAME || 'works for ' || MANAGER

BLAKE	works for	KING
CLARE	works for	KING
JONES	works for	KING
MARTIN	works for	BLAKE

* Các toán tử tập hợp

TOÁN TỬ TẬP HỢP	KẾT QUẢ ĐƯỢC TẠO RA
UNION	Phép hợp – hợp các kết quả của 2 câu lệnh SELECT, những hàng trùng lặp chỉ hiển thị một hàng MINUS
UNION ALL	Phép hợp – tương tự phép hợp UNION, điểm khác : các hàng trùng lặp được hiển thị INTERSECT UNION UNION ALL
INTERSECT	Phép giao – Các hàng thuộc cả 2 câu lệnh SELECT
MINUS	Phép trừ – Các hàng được trả về câu lệnh SELECT thứ nhất không xuất hiện câu lệnh SELECT thứ hai

* Các toán tử tập hợp

Ví dụ :

Select ename

From empa

MINUS

UNION

INTERSECT

Select ename

From emp

Where ename LIKE 'S%' ;

* Gộp dữ liệu sử dụng hàm GROUP

Các hàm GROUP ?

Các hàm GROUP thao tác trên tập các dòng để cho ra kết quả trên mỗi nhóm

EMP	DEPTNO	SAL
	10	2450
	10	5000
	10	1300
	20	800
	20	1100
	20	3000
	20	3000
	20	2975
	30	1600
	30	2850
	30	1250
	30	950
	30	1500
	30	1250

Lương lớn nhất trong Table EMP

MAX(SAL)

5000

* Các hàm GROUP

AVG

COUNT

MAX

MIN

STDDEV

SUM

VARIANCE

* Sử dụng các hàm GROUP

```
SELECT [CỘT,] <hàm_GROUP> (cột)
FROM   <tên Table>
[WHERE  <điều kiện> ]
[GROUP BY <cột>]
[ORDER BY <cột>]
```

* Sử dụng hàm AVG, SUM, MIN, MAX

Có thể sử dụng

- hàm AVG và SUM cho dữ liệu số
- hàm MIN, MAX cho kiểu dữ liệu bất kỳ

```
SELECT AVG(SAL) , MAX(SAL), MIN(SAL), SUM(SAL)  
FROM emp  
WHERE Job LIKE 'SALES%';
```

AVG (SAL)	MAX(SAL)	MIN(SAL)	SUM(SAL)
1400	1600	1250	5600

* Sử dụng hàm COUNT

- Hàm COUNT(*) trả về số lượng dòng trong 1 Table (thoả điều kiện chỉ ra)
- Hàm COUNT(<bthức>) trả về số lượng dòng không rỗng (thoả điều kiện chỉ ra)

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

* *Hàm GROUP và giá trị NULL*

- * Các hàm GROUP bỏ qua giá trị NULL trong cột
Sử dụng hàm chuyển đổi giá trị NULL : NVL để
các hàm group bao gồm các giá trị NULL

```
SELECT AVG(NVL(SAL))
FROM emp
WHERE Job LIKE 'SALES%';
```

* Tạo nhóm dữ liệu

EMP

DEPTNO	SAL
10	2450
10	5000
10	1300
20	800
20	1100
20	3000
20	3000
20	2975
30	1600
30	2850
30	1250
30	950
30	1500
30	1250

2916.6667

2175

1566.6667

Mức lương trong
EMP
Theo mỗi
Phân xưởng

DEPTNO	AVG(SAL)
10	2916.6667
20	2175
30	1566.6667

Tạo nhóm dữ liệu: Mệnh đề GROUP BY

SELECT [CỘT,] <hàm_GROUP> (cột)

FROM <tên Table>

[WHERE <điều kiện>]

[GROUP BY <biểu thức>]

[ORDER BY <cột>]

Chia các hàng trong bảng thành các nhóm nhỏ bằng cách sử dụng mệnh đề GROUP BY

Tạo nhóm dữ liệu: Mệnh đề GROUP BY

- * Tất cả các cột trong danh sách SELECT mà không chứa trong các hàm gộp nhóm, thì phải ở trong mệnh đề GROUP BY

```
SELECT deptno, AVG(sal)  
      FROM emp  
  GROUP BY deptno;
```

DEPTNO	AVG(SAL)
--------	----------

10	2916.6667
20	2175
30	1566.6667

Gộp nhóm nhiều hơn 1 cột

EMP

DEPTNO	JOB	SAL
10	MANAGER	2450
10	PRESIDENT	5000
10	CLERK	1300
20	CLERK	800
20	CLERK	1100
20	ANALYST	3000
20	ANALYST	3000
20	MANAGER	2975
30	SALESMAN	1600
30	MANAGER	2850
30	SALESMAN	1250
30	CLERK	950
30	SALESMAN	1500
30	SALESMAN	1250

Tổng lương theo
Mỗi công việc
Nhóm theo
Phân xưởng

DEPTNO	JOB	SAL
10	CLERK	1300
10	MANAGER	2450
10	PRESIDENT	5000
20	ANALYST	6000
20	CLERK	1900
20	MANAGER	2975
30	CLERK	950
30	MANAGER	2850
30	SALESMAN	5600

Gộp nhóm nhiều hơn 1 cột

```
SELECT deptno, job, sum(sal)  
FROM emp  
GROUP BY deptno,job;
```

DEPTNO	JOB	SUM(SAL)
10	CLERK	1300
10	MANAGER	2450
10	PRESIDENT	5000
20	ANALYST	6000
20	CLERK	1900
20	MANAGER	2975
30	CLERK	950
30	MANAGER	2850
30	SALESMAN	5600

Truy vấn không hợp lệ sử dụng các hàm GROUP

Bất kỳ cột hay biểu thức trong danh sách SELECT mà không ở trong hàm gộp dữ liệu thì phải ở trong mệnh đề GROUP BY

```
SELECT deptno, COUNT(ename)  
FROM emp;
```

Column missing in the GROUP BY clause

```
SELECT deptno, COUNT(ename)  
*
```

ERROR at line 1:
ORA-00937: not a single-group group function

Truy vấn không hợp lệ sử dụng các hàm GROUP

Không thể sử dụng mệnh đề WHERE để hạn chế các nhóm

```
SELECT deptno, AVG(sal)
```

```
FROM emp
```

```
WHERE AVG(sal) > 2000;
```

```
GROUP BY deptno;
```

```
WHERE AVG(sal) > 2000
```

```
*
```

ERROR at line 3:

ORA-00934: group function is not allowed here

Hạn chế các kết quả nhóm

EMP

DEPTNO	SAL
10	2450
10	5000
10	1300
20	800
20	1100
20	3000
20	3000
20	2975
30	1600
30	2850
30	1250
30	950
30	1500
30	1250

2916.6667

2175

1566.6667

Mức lương tối đa
Trong mỗi phân
Xưởng lớn hơn
2900

DEPTNO	MAX(SAL)
10	5000
20	3000

Hạn chế các kết quả nhóm: mệnh đề HAVING

Sử dụng mệnh đề HAVING để hạn chế các nhóm:

Các dòng được gộp nhóm

Được sử dụng các hàm GROUP

Các nhóm thoả mệnh đề HAVING được thể hiện

Sử dụng mệnh đề HAVING

```
SELECT deptno, MAX(sal)
FROM emp
GROUP BY deptno
HAVING MAX(sal) > 2000;
```

DEPTNO	MAX(SAL)
-----	-----
10	5000
20	3000

Sử dụng mệnh đề HAVING

```
SELECT job, SUM(sal) PAYROLL  
FROM emp  
WHERE job NOT LIKE 'SALES%'  
GROUP BY job  
HAVING SUM(sal) > 5000  
ORDER BY SUM(sal);
```

JOB	PAYROLL
ANALYST	6000
MANAGER	8275

Các hàm GROUP lồng nhau

Hiển thị lương trung bình lớn nhất

```
SELECT MAX(AVG(sal))
```

```
FROM emp
```

```
GROUP BY deptno
```

MAX(AVG(sal))

2916.6667

Truy vấn con (Subqueries)

Sử dụng truy vấn con để giải quyết bài toán
‘ Ai có lương lớn hơn lương Jone ?’

Truy vấn chính



‘Những người nào có lương lớn hơn lương Jone ?’

Truy vấn con



Lương jone là bao nhiêu?



Truy vấn con (Subqueries)

```
SELECT <danh sách cột>
FROM   <tên Table>
WHERE  <biểu thức điều kiện> <toán tử>
        (SELECT <danh sách cột>
         FROM   <tên Table>);
```

- * Truy vấn con được thực hiện trước truy vấn chính
- * Kết quả của truy vấn con được sử dụng cho truy vấn chính (truy vấn ngoài)

* Sử dụng truy vấn con

```
SELECT ename  
FROM emp  
WHERE sal >  
      (SELECT sal  
       FROM emp  
      WHERE empno = 7566);
```

ENAME

KING
FORD
SCOTT

Một số hướng dẫn sử dụng truy vấn con

- *Truy vấn con đặt trong dấu ngoặc đơn
- *Đặt truy vấn con bên phải toán tử so sánh
- *Không sử dụng mệnh đề ORDER BY trong truy vấn con
- *Sử dụng các toán tử dòng đơn với các truy vấn con dòng đơn
- *Sử dụng các toán tử nhiều dòng với các truy vấn con nhiều dòng

Các kiểu truy vấn con

Truy vấn con dòng đơn



Truy vấn con nhiều dòng



Truy vấn con nhiều cột



Truy vấn con dòng đơn

- * Trả về chỉ 1 dòng
- * Sử dụng các toán tử so sánh dòng đơn

Toán tử	Ý nghĩa
=	Bằng
>	Lớn hơn
\geq	Lớn hơn hoặc bằng
<	Nhỏ hơn
\leq	Nhỏ hơn hoặc bằng
\neq	Khác

Thực hiện truy vấn con dòng đơn

```
SELECT ename, job  
FROM emp  
WHERE job =  
      (SELECT job  
       FROM emp  
      WHERE empno = 7566)  
AND    sal >  
      (SELECT sal  
       FROM emp  
      WHERE empno = 7876);
```

ENAME	JOB
MILLER	CLERK

Sử dụng hàm GROUP trong truy vấn con

```
SELECT ename, job,sal  
FROM emp  
WHERE sal =  
      (SELECT Min(sal)  
       FROM emp);
```

ENAME	JOB	SAL
MILLER	CLERK	800

Mệnh đề HAVING với truy vấn con

- * Oracle server thực hiện truy vấn con đầu tiên
- * Oracle server trả về kết quả cho mệnh đề HAVING của truy vấn chính

```
SELECT deptno, Min(sal)
```

```
FROM emp
```

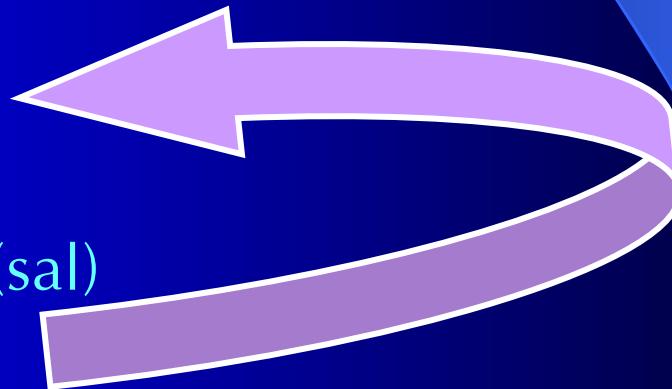
```
GROUP BY deptno
```

```
HAVING MIN(sal) >
```

```
(SELECT Min(sal)
```

```
FROM emp
```

```
WHERE deptno = 20);
```



Câu lệnh này sai gì ?

```
SELECT empno, ename  
FROM emp  
WHERE sal =  
      (SELECT Min(sal)  
       FROM emp  
       GROUP BY deptno );
```

Toán tử dòng đơn với
truy vấn dòng con nhiều dòng

ERROR:

ORA-01427: single-row subquery returns more than one row

Câu lệnh này sẽ làm việc gì ?

```
SELECT ename, job  
FROM emp  
WHERE job =  
      (SELECT job  
       FROM emp  
       WHERE ename = 'SMYTHE');
```

Truy vấn con không trả về giá trị

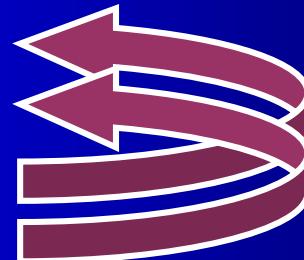
Truy vấn con nhiều dòng

- * Trả về nhiều hơn 1 dòng
- * Sử dụng các toán tử so sánh nhiều dòng

Toán tử	Ý nghĩa
IN	Bằng bất kỳ phần tử nào của danh sách
ANY	So sánh giá trị với mỗi giá trị trả về của truy vấn con
ALL	So sánh giá trị với tất cả giá trị trả về của truy vấn con

Sử dụng toán tử ANY trong truy vấn con nhiều dòng

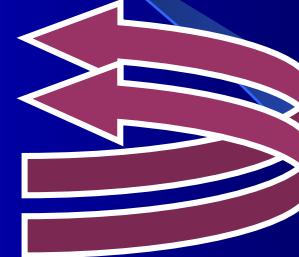
```
SELECT empno,ename, job  
FROM emp  
WHERE sal < ANY  
      (SELECT sal  
       FROM emp  
       WHERE job = 'CLERK')  
AND   job <> 'CLERK';
```



EMPNO	ENAME	JOB
7654	MARTIN	SALESMAN
7521	WARD	SALESMAN

Sử dụng toán tử ALL trong truy vấn con nhiều dòng

```
SELECT empno,ename, job  
FROM emp  
WHERE sal > ALL  
      (SELECT AVG(sal)  
       FROM emp  
       GROUP BY deptno);
```



EMPNO	ENAME	JOB
7839	KING	PRESIDENT
7566	JONÉ	MANAGER
7902	FORD	ANALYST
7788	SCOTT	ANALYST

** truy vấn con nhiều cột*

```
SELECT ordid, prodid, qty  
FROM item  
WHERE (prodid, qty) IN  
      (SELECT prodid, qty  
       FROM item  
       WHERE ordid = 605)  
AND    ordid <> 605;
```

Các giá trị NULL trong truy vấn nhiều cột

```
SELECT employee.ename  
FROM emp employee  
WHERE employee.empno NOT IN  
    (SELECT manager.mgr  
     FROM em manager);
```

* Toán tử EXISTS

- Trả về giá trị true nếu Subquery có kết quả khác rỗng, false trong trường hợp ngược lại.

Ví dụ:

```
SELECT distinct e.dept From emp e  
WHERE EXISTS  
(SELECT i.empd From invoice i  
WHERE (i.empid = e.empid) AND  
(i.pay_date > SYSDATE – 365));
```

Sử dụng 1 truy vấn con trong mệnh đề FROM

```
SELECT a.ename, a.sal, a.deptno, b.salavg
FROM emp a, (SELECT deptno,
                     AVG(sal) salavg
                  FROM emp
                 GROUP BY deptno) b
WHERE a.deptno = b.deptno
AND a.sal > b.salavg;
```

Kết xuất dữ liệu dễ đọc với SQL * PLUS

* Báo cáo tương giao



* Các biến thay thế

Dùng các biến thay thế để lưu các giá trị tạm thời

- Kí tự đơn &
- Kí hiệu kép &&
- Các câu lệnh DEFINE và ACCEPT

Chuyển đổi các giá trị biến giữa các câu lệnh SQL

Tự động thay đổi tiêu đề (header, footer)

* Sử dụng biến thay thế &

Dùng 1 biến có tiền tố là 1 ký tự & để nhắc người sử dụng cung cấp 1 giá trị

```
SELECT empno, ename, sal, deptno  
FROM   emp  
WHERE empno = &employee_num
```

Enter value for employee_num : 7369

EMPNO	ENAME	SAL	DEPTNO
7369	SMITH	800	20

* Sử dụng lệnh SET VERIFY

Chuyển đổi hiển thị văn bản của 1 câu lệnh trước và sau khi SQL*PLUS thay thế giá trị cho các biến

```
SET VERIFY ON
```

```
SELECT empno, ename, sal, deptno  
FROM emp  
WHERE empno = &employee_num;
```

Enter value for employee_num : 7369

Old 3: WHERE empno = &employee_num
New 3: WHERE empno = 7369

* Các giá trị ký tự và ngày với các biến thay thế

Sử dụng dấu nháy cho các giá trị ngày và ký tự

```
SELECT ename, deptno, sal*12
```

```
FROM emp
```

```
WHERE job = '&job_title';
```

Enter value for job_title : ANALYST

ENAME	DEPTNO	SAL*12
SCOTT	20	36000
FORD	20	36000

* *Chỉ định tên cột, biểu thức và văn bản
tại mỗi lần chạy*

Sử dụng các biến thay thế để bổ sung cho:

Điều kiện mệnh đề WHERE

Mệnh đề ORDER BY

Biểu thức cột

Tên Table

Dữ liệu vào câu lệnh statement

* *Chỉ định tên cột, biểu thức và văn bản
tại mỗi lần chạy*

```
SELECT empno, ename, job, &column_name  
FROM emp  
WHERE &condition  
ORDER BY &order_column;
```

Enter value for column_name : SAL

Enter value for condition: SAL >= 3000

Enter value for order_column : ename

.....

* Sử dụng biến thay thế &&

Dùng 2 ký hiệu &, nếu muốn sử dụng lại giá trị
biến đã có mà không hiển thị thông báo

```
SELECT empno, ename, job, &&column_name  
FROM   emp  
WHERE &column_name
```

Enter value for column_name : deptno

EMPNO	ENAME	JOB	DEPTNO
7369	KING	PRESIDENT	10
.....			

- * *Khai báo biến của người sử dụng*
 - * Bạn có thể định nghĩa lại các biến sử dụng 1 trong 2 câu lệnh SQL*PLUS:
 - **DEFINE**: Tạo 1 biến người sử dụng kiểu ký tự
 - **ACCEPT** : Đọc giá trị người sử dụng đưa vào và lưu vào 1 biến.
- Nếu bạn cần định nghĩa lại 1 biến mà có chứa ký tự trắng, bạn phải đóng giá trị trong dấu nháy đơn khi sử dụng câu lệnh **DEFINE**

* Câu lệnh ACCEPT

- *Tạo lập 1 thông báo của người sử dụng khi nhận giá trị của người sử dụng nhập từ bàn phím
- * Định nghĩa tường minh 1 biến kiểu số hay ngày
- * Làm ẩn giá trị nhập vào với lý do bảo mật

ACCEPT <biến> [kiểu dữ liệu] [format <khuôn dạng>]
[PROMPT <văn bản>] [HIDE]

* Sử dụng câu lệnh ACCEPT

ACCEPT dept PROMPT 'Provide the department name:'

```
SELECT *
```

```
FROM dept
```

```
WHERE dname = UPPER ('&dept');
```

Provide the department name: Sales

DEPTNO	DNAME	LOC
-----	-----	-----
10	SALES	CHICAGO
.....		

- * *Câu lệnh DEFINE và UNDEFINE*
- * Giá trị biến sẽ tồn tại cho đến khi hoặc:
 - Dùng lệnh UNDEFINE để xoá nó
 - Thoát SQL*Plus
- * Bạn có thể xem những thay đổi của bạn đối với lệnh DEFINE
- * Để định nghĩa các biến cho mọi lần thực hiện, tu sửa file login.sql (các biến sẽ được tạo lập mỗi lần khởi đầu)

* Sử dụng lệnh *DEFINE*

* Tạo 1 biến để chứa tên phân xưởng

DEFINE deptname = sales

DEFINE deptname

DEFINE DEPTNAME = “sales” (CHAR)

* Sử dụng các biến như bất kỳ các biến khác

SELECT *

FROM dept

WHERE dname = UPPER ('&deptname');

*Thiết lập môi trường SQL*PLUS*

- *Sử dụng các lệnh SET để điều khiển các phiên SET <tên biến hệ thống> <giá trị>
- *Xem những thiết lập bằng cách sử dụng lệnh SHOW

Ví dụ:

SQL>SET ECHO ON

SQL>SHOW ECHO

Echo ON

* Các biến lệnh SET

*ARRAYSIZE {20 | n}

*COLSEP {_ | <văn bản>}

*FEEDBACK {6 | n | OFF | ON }

*HEADING {OFF | ON}

*LINSIZE {80 | n}

*LONG {80 | n}

*PAGESIZE {24}

*PAUSE {ON | OFF | <Văn bản>}

*TERMOUT {OFF | ON}

* *Lưu các thiết lập vào File login.sql*

*File login.sql chứa các thiết lập chuẩn và các lệnh khác của SQL*Plus, được cài đặt sẵn tại mỗi lần login.

Có thể tu sửa login.sql để chứa các lệnh thiết lập (SET) mới.

* Các lệnh khuôn dạng SQL*PLUS

- *COLUMN [<giá trị chọn>]
- *TTITLE [<Văn bản> | OFF | ON]
- *BTITLE [<Văn bản> | OFF | ON]
- *BREAK [ON <phần tử báo cáo>]

* Lệnh COLUMN

Điều khiển hiển thị 1 cột

COL[UMN] [<cột> | <bí danh> [mục chọn]]

CLE[AR] : Xoá tất cả các khuôn dạng cột

FOR[MAT] <khuôn dạng>: Thay đổi hiển thị cột sử dụng mô hình khuôn dạng

HEA[DING] <văn bản> : Thiết lập tiêu đề cột

JUS[TIFY] {lề}: Căn lề tiêu đề cột : trái, giữa, phải

* Sử dụng lệnh COLUMN

*Tạo lập các tiêu đề cột

COLUMN ename HEADING 'Employee IName' FORMAT A15

COLUMN sal JUSTIFY LEFT FORMAT \$99,990.00

COLUMN mgr FORMAT 999999999 NULL ' No manager'

* Hiển thị thiết lập hiện hành của cột ENAME

COLUMN ename

* Xoá thiết lập hiện hành của cột ENAME

COLUMN ename CLEAR

* Các mô hình khuôn dạng COLUMN

Phần tử	Mô tả	Ví dụ	Kết quả
An	Thiết lập hiển thị độ rộng là n	N/A	N/A
9	Chặn các ký số zero đơn	999999	1234
0	Cho phép bắt đầu là 0	099999	01234
\$	Dấu \$ động	\$9999	\$1234
L	Tiền tệ địa phương	L9999	L1234
.	Vị trí dấu chấm thập phân	9999.99	1234.00
,	Phân cách hàng ngàn	9,999	1,234

* Sử dụng lệnh *BREAK*

* Ngăn các dòng trùng lắp và phân lớp dòng

* Ngăn sự trùng lắp

SQL> BREAK ON ename ON job

* Phân lớp dòng tại các giá trị phân cách

SQL> BREAK ON ename SKIP 4 ON job SKIP 2

* Sử dụng các lệnh TTITLE và BTITLE

*Hiển thị header và Footer

TTITLE [**< văn bản >** | ON | OFF]

*Thiết lập header báo cáo

TTITLE ‘Salary | Report’

*Thiết lập footer báo cáo

BTITLE ‘Confidential’

**Tạo lập 1 File kịch bản để chạy 1 báo cáo*

1. Tạo 1 lệnh SQL SELECT
2. Lưu lệnh SELECT vào 1 File kịch bản
3. Nạp File kịch bản vào 1 trình soạn thảo
4. Thêm các lệnh định dạng trước câu lệnh SELECT
5. Xem xét các ký tự kết thúc di theo sau lệnh SELECT
6. Xoá các câu lệnh định dạng sau câu lệnh SELECT
7. Lưu File kịch bản
8. Chạy File kịch bản : START <tên file>

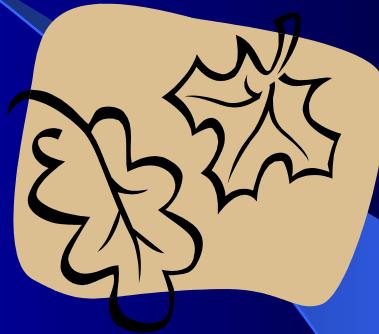
* Báo cáo mâu

Fri Oct 24

Page 1

Job Category	Employee	Salary
CLERK	ADAMS	\$1,100.00
	JAMES	\$950.00
	MILLER	\$1,300.00
MANAGER	SMITH	\$800.00
	BLAKE	\$2,850.00
	CLARK	\$2,450.00
SALESMAN	JONES	\$2,975.00
	ALLEN	\$1,600.00
	MARTIN	\$1,250.00
	TURNNER	\$1,500.00
	WARD	\$1,250.00

Confidential



Dường đi khó, không phải vì ngăn sông cách núi
Phường khó vì lòng người ngoài việc sông

Tạo lập khung nhìn (Creating Views)

* Mục tiêu

Sau khi hoàn thành bài học, bạn có thể thực hiện được các điều sau:

- *Mô tả 1 khung nhìn
- *Tạo 1 khung nhìn
- *Sửa đổi định nghĩa 1 khung nhìn
- *Chèn, tu sửa, và xoá dữ liệu từ 1 khung nhìn
- *Xoá 1 khung nhìn
- *Mô tả 1 dòng bên trong khung nhìn
- *Thực hiện 1 phân tích “Top-N”

*Các đối tượng Cơ sở dữ liệu

Đối tượng	Ý nghĩa
Table (Bảng)	Đơn vị cơ sở của sự lưu trữ; bao gồm các dòng, cột
View (Khung nhìn)	Hiển thị logic các tập con dữ liệu từ 1 hay nhiều Table
Sequence (Trình tự)	Sinh ra các giá trị của khoá chính
Index (Chỉ mục)	Nâng cao hiệu năng 1 số truy vấn
Synonym (Bí danh)	Gán tên cho 1 đối tượng

* Khung nhìn (view) là gì ?

EMP Table

EMPNO	ENAME	MGR	HIRE DATE	SAL	COMM
7839	KING	MANAGER	12/05/80	1200	1100

EMPVU10 View

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

* Khung nhìn đơn giản và khung nhìn phức hợp

Đặc trưng	Khung nhìn đơn giản	Khung nhìn phức hợp
Số lượng Table	Một	Một hay nhiều
Chứa Hàm	Không	Có
Chứa nhóm dữ liệu	Không	Có
DML thông qua khung nhìn	Có	Không

* Tạo lập một khung nhìn

- Có thể gắn 1 truy vấn con trong câu lệnh CREATE VIEW

```
CREATE (OR REPLACE) (FORCE | NOFORCE) VIEW view  
(( alias (, alias | ...))  
AS subquery  
(WITH CHECK OPTION (CONSTRAINT constraint) )  
(WITH READ ONLY);
```

- Truy vấn con có thể có cú pháp SELECT phức hợp
- Truy vấn con không thể chứa 1 mệnh đề ORDER BY

* Tạo lập một khung nhìn

- Tạo lập 1 khung nhìn, EMPVU10 chứa chi tiết các nhân viên trong phân xưởng 10

```
SQL> CREATE VIEW empvu10  
AS SELECT empno, ename, job  
FROM emp  
WHERE deptno = 10;
```

- Mô tả cấu trúc của khung nhìn bởi câu lệnh DESCRIBE của SQL*Plus

```
SQL> DESCRIBE empvu10
```

* Tạo lập một khung nhìn

- Tạo lập 1 khung nhìn sử dụng bí danh cột trong truy vấn con

```
SQL> CREATE VIEW salvu10
AS SELECT    empno EMPLOYEE_NUMBER, ename  NAME,
              Sal SALARY
FROM        emp
WHERE       deptno = 10;
```

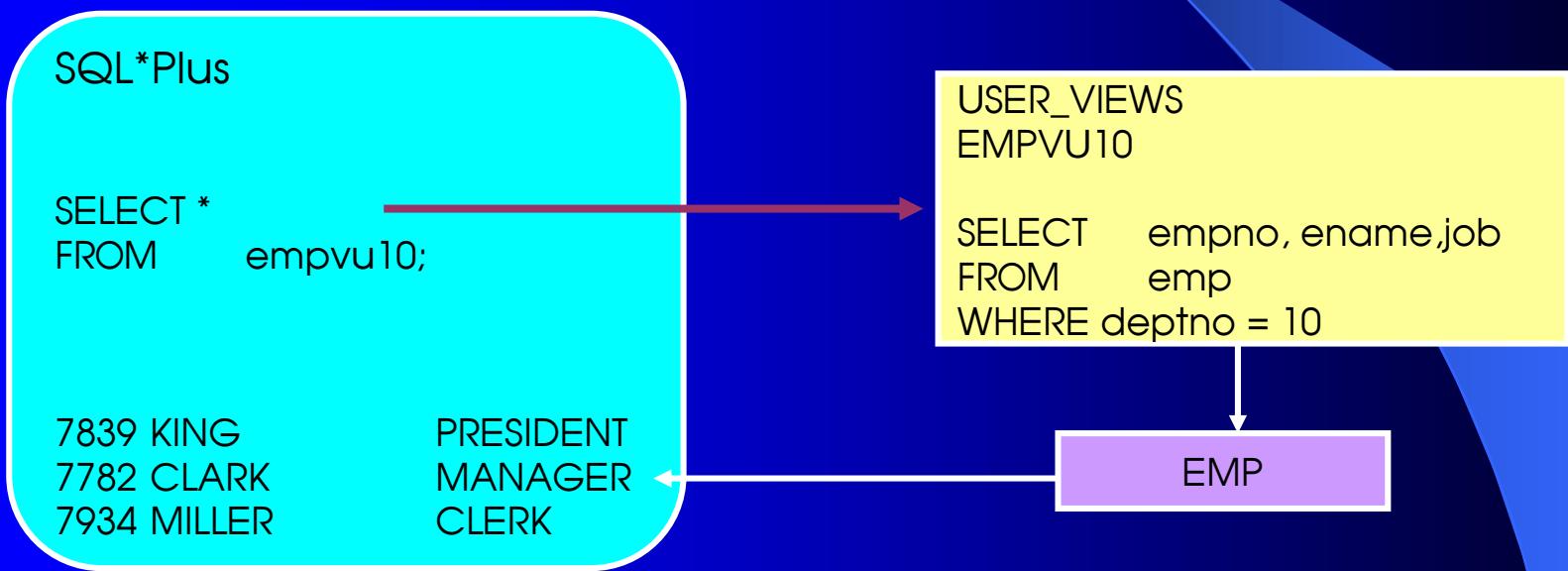
- Chọn các cột từ khung nhìn này bởi cách đưa ra tên bí danh

* Rút trích dữ liệu một khung nhìn

```
SQL> SELECT *  
FROM salvu30;
```

EMPLOYEE_NUMBER	NAME	SALARY
7698	BLAKE	2850
7654	MARTIN	1250
7499	ALLEN	1600
7844	TURNER	1500
7900	JAMES	950
7521	WARD	1250

* Truy vấn một khung nhìn



* Hiệu chỉnh một khung nhìn

- Hiệu chỉnh khung nhìn EMPVU10 bằng cách sử dụng **CREATE OR REPLACE VIEW**. Thêm bí danh cho mỗi tên cột.

```
SQL> CREATE OR REPLACE VIEW empvu10  
          (employee_number, employee_name, job_title)  
          AS SELECT  
          empno, ename, job  
          FROM emp  
          WHERE deptno = 10;
```

- Các bí danh cột trong mệnh đề **CREATE VIEW** được liệt kê cùng thứ tự trong truy vấn con

* Tạo lập một khung nhìn phức hợp

- Tạo lập 1 khung nhìn phức hợp chứa các hàm gộp nhóm để hiển thị các giá trị từ 2 table

```
SQL> CREATE VIEW  
AS SELECT  
FROM  
WHERE  
GROUP BY
```

```
dept_sum_vu  
(name, minsal, maxsal, avgsal)  
d.dname, MIN(e.sal), MAX(e.sal), AVG(e.sal)  
emp e, dept d  
e.deptno = d.deptno  
d.dname
```

* Các qui tắc để thực hiện các thao tác DML trên một khung nhìn

- Bạn có thể thực hiện các thao tác trên các khung nhìn đơn giản
- Bạn không thể xoá 1 dòng nếu khung nhìn chứa:
 - Hàm gộp nhóm
 - Một mệnh đề GROUP BY
 - Từ khoá DISTINCT
 - Từ khoá ROWNUM

* Các qui tắc để thực hiện các thao tác DML trên một khung nhìn

*Bạn không thể hiệu chỉnh dữ liệu nếu khung nhìn chứa:

- Bất kỳ những ghi chú trong Slide trước
- Các cột được định nghĩa bằng các biểu thức
- Cột giả ROWNUM

*Bạn không thể thêm dữ liệu nếu :

Khung nhìn chứa bất kỳ những gì đã nêu trên hoặc Slide trước đó

Có các cột NOT NULL trong các Table cơ sở mà không được chọn bởi khung nhìn

*Sử dụng mệnh đề WITH CHECK OPTION

- Bạn có thể đảm bảo rằng DML trên khung nhìn nằm trong miền khung nhìn bằng cách sử dụng mệnh đề WITH CHECK OPTION

```
SQL> CREATE OR REPLACE VIEW      empvu20
AS SELECT      *
FROM          emp
WHERE         deptno = 20
WITH CHECK OPTION CONSTRAINT empvu20_chk;
```

Bất kỳ cố gắng nào để thay đổi số của phân xưởng của bất kỳ dòng nào trong khung nhìn, bởi nó sẽ mâu thuẫn với ràng buộc WITH CHECK OPTION

* Từ chối các thao tác DML

- Bạn có thể đảm bảo rằng không có thao tác DML xảy ra bằng cách thêm vào WITH READ ONLY vào định nghĩa khung nhìn

```
SQL> CREATE OR REPLACE VIEW empvu10  
      (employee_number, employee_name, job_title)  
AS SELECT  
      empno, ename, job  
      FROM  
      emp  
      WHERE  
      deptno = 10  
      WITH READ ONLY;
```

- Mọi cố gắng thực hiện 1 DML trên bất kỳ dòng nào trong khung nhìn sẽ gây lỗi trên Oracle SERVER

* Xoá 1 khung nhìn

Xoá 1 khung nhìn không làm mất dữ liệu
bởi vì khung nhìn là được dựa trên cơ sở
các Table trong CSDL

```
DROP VIEW < Tên khung nhìn>
```

```
SQL> DROP VIEW empvu10;
```

* Các dòng trong khung nhìn

- Một dòng khung nhìn là 1 truy vấn con với 1 bí danh (quan hệ với tên) mà bạn sử dụng trong câu lệnh SQL.
- Một dòng khung nhìn là tương ứng với việc sử dụng 1 tên truy vấn con trong mệnh đề FROM của truy vấn chính.
- Một dòng khung nhìn không phải là 1 lược đồ

* Phân tích “Top –N”

- Các truy vấn Top-N yêu cầu n giá trị lớn nhất hay nhỏ nhất của 1 cột
 - Mười sản phẩm bán chạy nhất là gì?
 - Mười sản phẩm bán ít nhất là gì?

Cả hai tập các giá trị lớn nhất và nhỏ nhất là các truy vấn TOP-N

* Thực hiện Phân tích “Top –N”

- Cấu trúc mức cao của 1 truy vấn phân tích Top-N là :

```
SQL> SELECT (column_list), ROWNUM  
FROM (SELECT (colum_list) FROM table  
      ORDER BY Top-N_column)  
WHERE ROWNUM <= N
```

```
SQL> SELECT name, sal ROWNUM  
FROM    (SELECT ename, sal FROM emp  
        ORDER BY sal DESC)  
WHERE ROWNUM <= 3
```

Những đối tượng cơ sở dữ liệu khác

(Other Databases Objects)

* Các đối tượng

Sau khi hoàn tất bài học này, bạn có thể thực hiện:

Mô tả một số đối tượng CSDL và cách thức sử dụng chúng

Tạo lập, duy trì và sử dụng tuần tự

Tạo lập và duy trì chỉ mục

Tạo lập những bí danh riêng và chung

* Các đối tượng CSDL

Đối tượng	Ý nghĩa
Table	Đơn vị lưu trữ cơ sở; bao gồm các dòng và cột
Khung nhìn (View)	Biểu diễn logic tập con dữ liệu từ 1 hay nhiều Table
Tuần tự	Sinh ra các giá trị khoá chính
Chỉ mục	Tăng hiệu năng cho một số truy vấn
Bí danh	Gán tên cho một số đối tượng

* *Tuần tự (Sequence) là gì ?*

Tự động sinh ra các giá trị duy nhất

Là 1 đối tượng chia sẻ

Là kiểu được sử dụng để tạo ra giá trị khoá chính

Thay thế mã ứng dụng

Tăng tốc độ cho việc truy xuất các giá trị tuần tự

* Câu lệnh CREATE SEQUENCE

- Định nghĩa 1 tuần tự để tự động sinh ra các số tuần tự

```
CREATE SEQUENCE sequence  
  ( INCREMENT BY n)  
  (START WITH N)  
  ((MAXVALUE n | NOMAXVALUE))  
  ((MINVALUE n | NOMINVALUE))  
  ((CYCLE | NOCYCLE))  
  ((CACHE n | NOCACHE));
```

* *Tạo lập 1 tuần tự*

Tạo lập 1 tuần tự tên DEPT_DEPTNO được dùng làm khoá chính cho Table DEPT.

Không sử dụng mục chọn CYCLE

```
SQL> CREATE SEQUENCE dept_deptno
      INCREMENT BY 1
      START WITH 91
      MAXVALUE 100
      NOCACHE
      NOCYCLE
```

* Xác định tuần tự

- Xem các giá trị tuần tự trong Table tự điểuđữ liệu USER_SEQUENCES

```
SQL > SELECT sequence_name, min_value, max_value,  
          increment_by, last_number  
      FROM user_sequences;
```

- Cột LAST_NUMBER hiển thị số tuần tự kế tiếp

* Cột giả *NEXTVAL* và *CURRVAL*

- * *NEXTVAL* trả về giá trị kế tiếp
Nó trả về 1 giá trị duy nhất khi được tham khảo
- * *CURRVAL* chứa giá trị tuân tự hiện hành
NEXTVAL phải được lưu hành 1 giá trị trước
khi *CURRVAL* chứa 1 giá trị

* Sử dụng 1 tuần tự

- Chèn 1 phân xưởng mới tên “MARKETING” tại San Diego

```
SQL> INSERT INTO dept(deptno, dname, loc)
VALUES (dept_deptno, NEXTVAL,
'MARKETING', 'SAN DIEGO')
```

Xem giá trị hiện hành đối với tuần tự
DEPT_DEPTNO

```
SQL> SELECT dept_deptno, CURRVAL
FROM dual;
```

* Sử dụng 1 tuần tự

Các giá trị tuần tự trong bộ nhớ (cache) cho phép truy xuất nhanh hơn đến các giá trị này

Khoảng trống ngắt quãng trong các giá trị tuần tự có thể xảy ra khi:

- Một rollback xảy ra
- Hệ thống bị hỏng
- Một tuần tự được dùng trong 1 Table khác

Xem tuần tự kế tiếp, nếu nó được tạo lập bởi NOCACHE, bằng cách truy vấn Table USER_SEQUENCES

* *Hiệu chỉnh 1 tuần tự*

Thay đổi số gia, giá trị lớn nhất, giá trị nhỏ nhất,
tùy chọn **chu kỳ**, bộ đệm (cache)

```
SQL> ALTER SEQUENCE dept_deptno
      INCREMENT BY 1
      MAXVALUE 999999
      NOCACHE
      NOCYCLE
```

* Xoá 1 tuần tự

- Xoá 1 tuần tự từ 1 tự điển dữ liệu bằng cách sử dụng câu lệnh DROP SEQUENCE
- Sau khi xoá tuần tự không thể được tham khảo lại

```
SQL> DROP SEQUENCE dept_deptno;  
Sequence droppe
```

* Chỉ mục là gì ?

- Là 1 đối tượng lược đồ
- Là được sử dụng bởi Oracle Server để tăng tốc độ trích lọc các dòng bằng cách sử dụng 1 con trỏ
- Có thể rút ngắn các truy xuất vào ra đĩa bằng cách sử dụng thuật toán truy xuất đường đi tối ưu để định vị nhanh dữ liệu
- Là độc lập với Table mà nó chỉ mục
- Được sử dụng và duy trì tự động bằng Oracle Server

* Tạo lập 1 chỉ mục như thế nào?

Tự động : Một chỉ mục đơn (unique) được tạo lập tự động khi bạn định nghĩa 1 khoá chính (PRIMARY KEY) hay ràng buộc đơn trong 1 Table

Thủ công: Người sử dụng có thể tạo lập 1 chỉ mục không đơn trên 1 cột để nâng cao tốc độ triuy xuất trên các dòng

* Tạo lập 1 chỉ mục

- Tạo lập 1 chỉ mục trên 1 hay nhiều cột

```
CREATE INDEX index  
ON table (column [, column] . . . );
```

- Nâng cao tốc độ truy xuất trên cột ENAME trong table EMP

```
SQL > CREATE INDEX emp_ename_idx  
      ON emp(ename)
```

* Khi tạo lập 1 chỉ mục

- Cột được sử dụng thường nằm trong mệnh đề WHERE hay điều kiện liên kết
- Cột chứa dữ liệu lớn
- Cột chứa 1 số lớn hay nhiều giá trị rỗng
- Hai hay nhiều cột thường được sử dụng với nhau trong mệnh đề WHERE hay điều kiện kết nối
- Một Table lớn và hầu hết các truy vấn thường gọi ít hơn 2- 4% các dòng

* Khi không tạo lập 1 chỉ mục

- Table nhỏ
- Các cột thường ít được sử dụng như là 1 tự điển dữ liệu trong truy vấn
- Hầu hết các truy vấn là thường gọi nhiều hơn 2 – 4% các dòng
- Các table thường hay cập nhật

* Xác nhận 1 chỉ mục

- Khung nhìn từ điển dữ liệu USER_INDEXES chứa tên chỉ mục
- Khung nhìn USER_IND_COLUMNS chứa tên chỉ mục, tên Table, và tên cột

```
SQL> SELECT    ic.index_name, ic.column_name,  
           ic.column_position col_pos, ix.uniqueness  
      FROM    user_indexes ix, user_ind_columns ic  
     WHERE   ic.index_name = ix.index_name  
     AND    ic.table_name = 'EMP'
```

* *Hàm chỉ mục cơ sở*

- Một hàm chỉ mục cơ sở là 1 chỉ mục dựa trên 1 biểu thức
- Một biểu thức chỉ mục dựa trên các cột của Table, các hàm SQL, và các hàm do người sử dụng định nghĩa

```
SQL> CREATE TABLE test (col1 NUMBER);
```

```
SQL> SELECT col1+10 FROM test;
```

* Xoá 1 chỉ mục

Xoá 1 chỉ mục từ 1 tự điển dữ liệu

```
SQL> DROP INDEX index
```

Xoá chỉ mục EMP_ENAME_IDX từ tự điển dữ liệu

```
SQL> DROP INDEX emp_ename_idx;
```

* Đồng nghĩa (Synonyms)

- Truy xuất đơn giản đến các đối tượng bằng cách tạo ra 1 đồng nghĩa (một tên khác của 1 đối tượng)
- Tham khảo đến 1 Table bởi 1 người sử dụng khác
- Làm ngắn tên các đối tượng

```
CREATE (PUBLIC) SYNONYM synonym  
FOR object
```

* Thực hành

- Tạo lập các tuần tự
- Sử dụng tuần tự
- Tạo lập các chỉ mục không duy nhất
- Hiển thị thông tin tự điển dữ liệu về tuần tự và chỉ mục
- Xoá chỉ mục

* Tạo lập và xoá đồng nghĩa

- Tạo lập 1 tên ngắn cho khung nhìn
DEPR_SUM_VU

```
SQL> CREATE SYNONYM d_sum  
FOR      dept_sum_vu
```

- Xoá 1 đồng nghĩa

```
SQL> DROP SYNONYM d_sum;
```

ĐIỀU KHIỂN QUYỀN TRUY XUẤT người sử dụng

(Controlling User Access)

* *Mục tiêu*

Sau khi hoàn thành bài học , bạn có thể :

- * Tạo lập các người sử dụng (USERS)
- * Tạo lập các chức năng để thiết lập và duy trì dễ dàng mô hình bảo mật
- * Sử dụng lệnh GRANT và REVOKE để gán và huỷ bỏ quyền truy xuất

ĐIỀU KHIỂN QUYỀN TRUY XUẤT NGƯỜI SỬ DỤNG



* *Phân quyền*

* Bảo mật CSDL

- Bảo mật hệ thống
- Bảo mật dữ liệu

* Quyền hệ thống : Cho phép truy cập dữ liệu

* Quyền đối tượng : Xử lý nội dung các đối tượng dữ liệu

* Lược đồ : Chọn các đối tượng như Table, Khung nhìn và tuần tự

* *Quyền hệ thống*

- Có hơn 80 quyền
- DBA có quyền hệ thống mức cao:
 - Tạo người sử dụng mới
 - Xoá người sử dụng
 - Xoá Table
 - Cập nhật Table

* Tạo lập người sử dụng

- Chỉ có người quản trị hệ thống (DBA) mới tạo được người sử dụng, dùng lệnh CREATE USER

```
CREATE          USER    user  
IDENTIFIED      BY      password;
```

```
CREATE          USER    scott  
IDENTIFIER      BY      tiger;
```

* Quyền hệ thống người sử dụng

- Khi một người sử dụng được tạo lập, DBA có thể gán quyền hệ thống cho người sử dụng:

```
GRANT privilege      (, privilege. . .)
TO USER            (, user . . .);
```

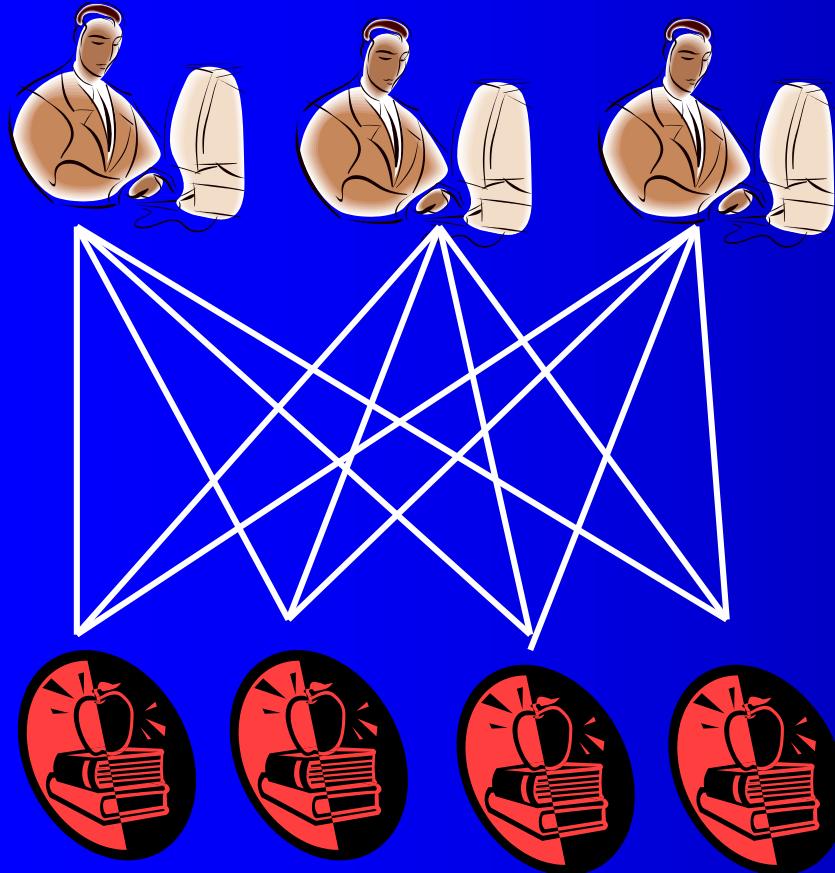
- Một phát triển ứng dụng có thể có các quyền hệ thống:
 - CREATE SESSION
 - CREATE TABLE
 - CREATE SEQUENCE
 - CREATE VIEW
 - CREATE PROCEDURE

* Gán quyền hệ thống

- Người quản trị hệ thống có thể gán cho người sử dụng các quyền hệ thống

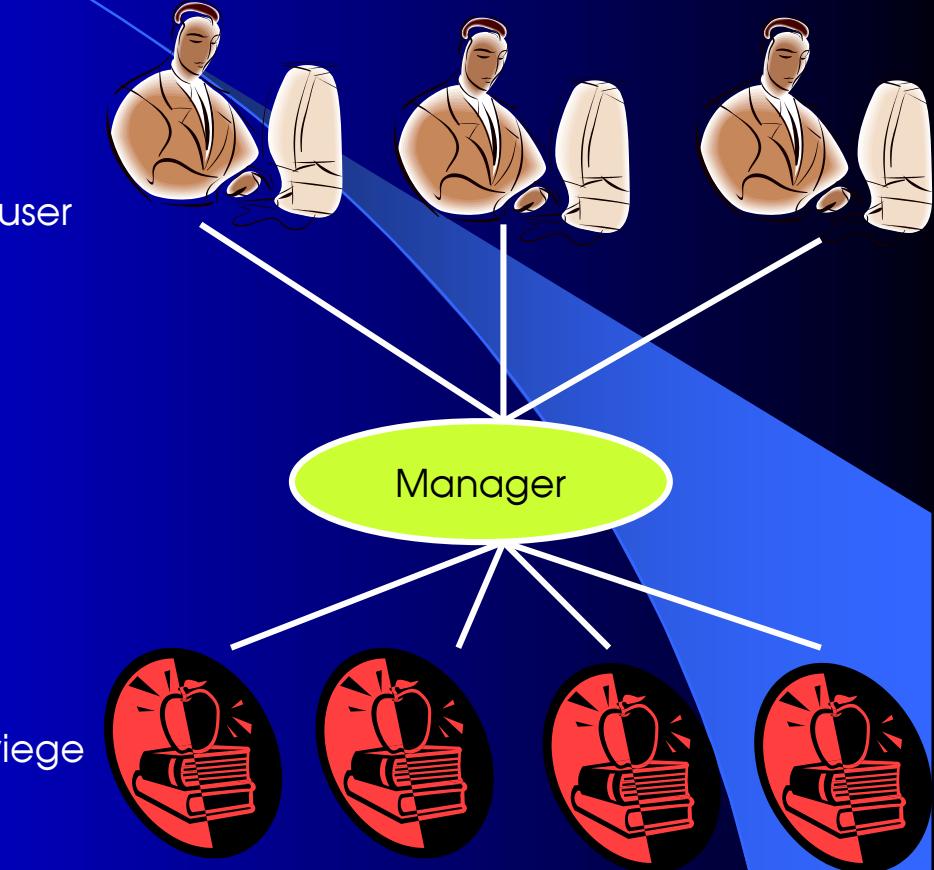
```
SQL> GRANT create table, create sequence, create view  
      TO      scott;
```

* Chức năng (role) là gì ?



Privilege

Không có chức năng phân chia dữ liệu



Manager

Có chức năng phân chia dữ liệu

* *Tạo lập và gán quyền cho chức năng (role)*

```
SQL> CREATE ROLE manager;
```

```
SQL> CREATE create table, create view  
      to manager;
```

```
SQL> GRANT manager to BALKE, CLARK;
```

* *Thay đổi Password*

- Người DBA tạo lập account và Password
- Bạn có thể thay đổi password bằng cách sử dụng lệnh ALTER USER

```
SQL> ALTER USER scott  
      IDENTIFIED BY lion;
```

* Các quyền đối tượng

Object Privilege	Table	View	Sequence	Procedure
ALTER	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>	
DELETE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
EXECUTE				<input checked="" type="checkbox"/>
INDEX	<input checked="" type="checkbox"/>			
INSERT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
REFERENCES	<input checked="" type="checkbox"/>			
SELECT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
UPDATE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		

* Các quyền đối tượng

- Quyền đối tượng biến đổi từ đối tượng đến đối tượng
- Một người sở hữu dữ liệu có mọi quyền đối tượng
- Một người sở hữu dữ liệu có thể chỉ định các quyền trên các đối tượng sở hữu

```
GRANT    object_priv [(column)]
ON      object
TO      (user | role | PUBLIC)
[WITH GRANT OPTION]
```

Xin cảm ơn !

