**查询DATABASE属性**

REM: List of Tables //会显示在输出上这句话

select table\_name

from user\_tables

查询所有表名

REM: List of Table Columns

select table\_name, column\_name,column\_id

from user\_tab\_columns

where table\_name like 'AP%'

order by table\_name,column\_id;

查询所有表里所有列

REM: List of Table Column Constraiints

select table\_name,constraint\_name,constraint\_type,search\_condition,index\_name,r\_constraint\_name,delete\_rule

from user\_constraints

order by table\_name;

查询限制

REM: List of Table Column Comments

select table\_name,column\_name,comments

from user\_col\_comments

order by table\_name;

查询comment

**Adding CHECK constraints to enforce business rules**

ALTER TABLE AP\_for

adding constraint, modifying column sizes, setting NULL/NOT NULL, DEFAULT values

增加限制条件，格式：

ALTER TABLE TABLE名 ADD CONSTRAINT 自定义代表限制的名字 CHECK ( 可以是多条限制，用 AND 连接 )

alter table AP\_EMP add constraint C\_EMP\_EMPNO check (EMPNO between 1000 and 9999);

//1000 to 9999 input only

alter table AP\_EMP add constraint C\_EMP\_ENAME check (ENAME=upper(ENAME));

// only upper case input allowed

* Test **constraint** **enforcements**

insert into AP\_EMP values (10001, 'Robert', 'TEMP',7369,sysdate-40,1000,null,10);

**//insert 是单独指令，结束要加分号，不然不能批量运行insert**

insert into AP\_EMP values (9999, 'Robert', 'TEMP',7369,sysdate-40,1000,null,10);

//now empno must be in 1000-9999, ename must be upper case

**MODIFY**:

* **Modify** column to add DEFAULT value

alter table AP\_PROEMP modify ( HOURS default 0 );

* **Modify** column to allow for NULL

alter table AP\_EMP modify job NULL;

* **Modify** column to change size and set NOT NULL

alter table AP\_EMP modify job varchar2(20) NOT NULL;

// if any job columns are assigned with value more than 10 size, we can reduce size to 10 again.

**Add FOREIGN KEY CONSTRAINT**

* Adding **foreign key constrain**

alter table AP\_EMP add constraint FK\_EMP\_MGR foreign key (MGR) references AP\_EMP(EMPNO)

**ALTER TABLE AP\_to add/remove column**

alter table AP\_EMP **add** BIRTHDATE date;

alter table AP\_EMP add ADDRESS varchar2(50);

alter table AP\_EMP **drop** (BIRTHDATE, ADDRESS);

**Creating table from other table**

CREATE TABLE AP\_EMPTEST **AS SELECT \* FROM AP\_EMP** -- CREATE A TABLE FROM EXISITNG TABLE

SELECT **\*** FROM AP\_EMPTEST -- RETRIVING/SELECTING ALL RECORDS OF A TABLE

**TRUNCATE** TABLE AP\_EMPTEST -- REMOVING/DELETING ALL RECORDS OF A TABLE

SELECT \* FROM AP\_EMPTEST-- RETRIVING/SELECTING ALL RECORDS OF A TABLE

**DROP** TABLE AP\_EMPTEST -- DROPPING A TABLE

//once dropped, it is not easy to recover

//database need back up

SELECT **\*** FROM AP\_EMPTEST -- RETRIVING/SELECTING ALL RECORDS OF A TABLE

foSQL PRIMARY KEY Constraint

The PRIMARY KEY constraint uniquely identifies each record in a table.

Primary keys must contain UNIQUE values, and cannot contain NULL values.

A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).

SQL PRIMARY KEY on CREATE TABLE

The following SQL creates a PRIMARY KEY on the "ID" column when the "Persons" table is created:

**MySQL:**

CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int,  
    PRIMARY KEY (ID)  
);

**SQL Server / Oracle / MS Access:**

CREATE TABLE Persons (  
    ID int NOT NULL PRIMARY KEY,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int  
);

To allow naming of a PRIMARY KEY constraint, and for defining a PRIMARY KEY constraint on multiple columns, use the following SQL syntax:

**MySQL / SQL Server / Oracle / MS Access:**

CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int,  
    CONSTRAINT PK\_Person PRIMARY KEY (ID,LastName)  
);

**Note:** In the example above there is only ONE PRIMARY KEY (PK\_Person). However, the VALUE of the primary key is made up of TWO COLUMNS (ID + LastName).

SQL PRIMARY KEY on ALTER TABLE

To create a PRIMARY KEY constraint on the "ID" column when the table is already created, use the following SQL:

**MySQL / SQL Server / Oracle / MS Access:**

**ALTER TABLE Persons  
ADD PRIMARY KEY (ID);**

To allow naming of a PRIMARY KEY constraint, and for defining a PRIMARY KEY constraint on multiple columns, use the following SQL syntax:

**MySQL / SQL Server / Oracle / MS Access:**

**ALTER TABLE Persons  
ADD CONSTRAINT PK\_Person PRIMARY KEY (ID,LastName);**

**Note:** If you use the ALTER TABLE statement to add a primary key, the primary key column(s) must already have been declared to not contain NULL values (when the table was first created).

DROP a PRIMARY KEY Constraint

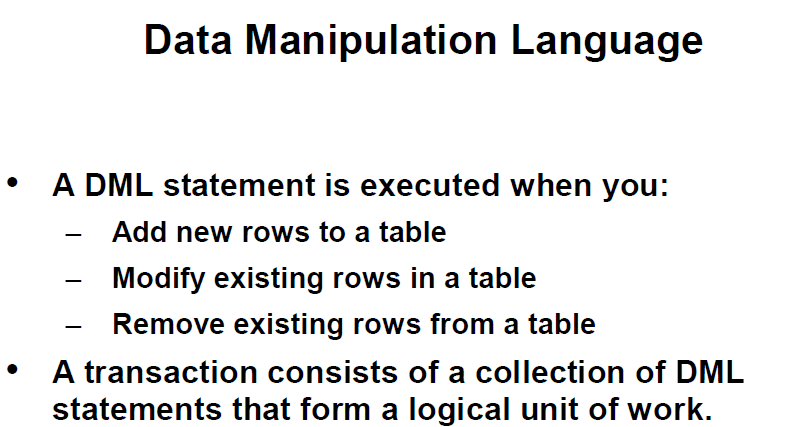
To drop a PRIMARY KEY constraint, use the following SQL

**MySQL:**

ALTER TABLE Persons  
DROP PRIMARY KEY;

**SQL Server / Oracle / MS Access:**

ALTER TABLE Persons  
DROP CONSTRAINT PK\_Person;



– **CREATING A EMPTY TABLE FROM ANOTHER TABLE;**

**(故意给提一个错误的condition)**

CREATE TABLE AP\_DEPTTEST AS SELECT \* FROM AP\_DEPT WHERE 1=2;

CREATE TABLE AP\_EMPTEST AS SELECT \* FROM AP\_EMP **WHERE 1=2**;

//create empty table with false condition

--POPULATING TABLE FROM ANOTHER TABLE;

INSERT INTO AP\_DEPTTEST SELECT \* FROM AP\_DEPT;

INSERT INTO AP\_EMPTEST SELECT \* FROM AP\_EMP;

COMMIT;

//commit apply to all statements before

INSERT INTO AP\_DEPTTEST (deptno,dname,loc) VALUES ('TRAIING','AUSTIN');

// will not success, not enough value inserted as AP\_DEPTTEST (deptno,dname,loc) asked

INSERT INTO AP\_DEPTTEST (deptno,dname,loc) VALUES (NULL,'TRAINING','AUSTIN');

//will not success because PK should not be null

INSERT INTO AP\_DEPTTEST (deptno,dname,loc) VALUES (400,'TRAINING','AUSTIN');

//fail, because deptno is set not to exceed 2 digits

INSERT INTO AP\_DEPTTEST VALUES (50,'TRAINING','AUSTIN');

**\*//if there is no deptno 50 in table, it will success, but it will fails vice versa.不同PK相同attributes**

**//If there is a 4th column, it is allowed to be NULL, success(no repeat PK)**

SELECT \* FROM AP\_DEPTTEST;

COMMIT;

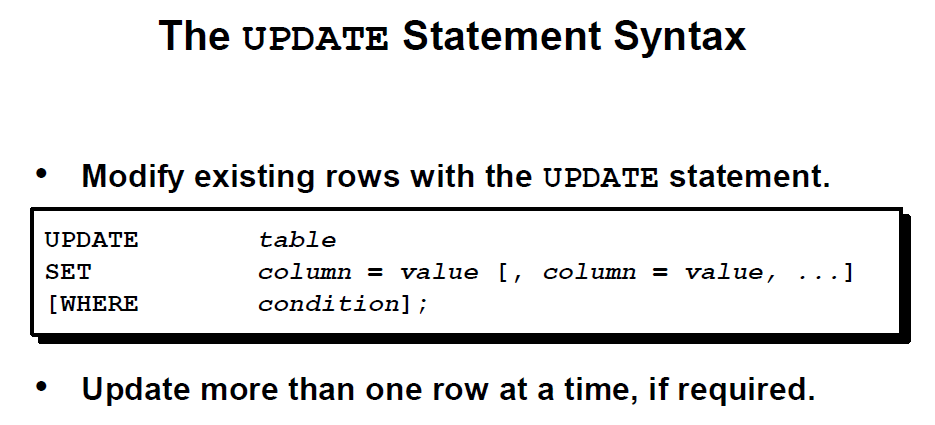
INSERT INTO AP\_DEPTTEST VALUES (60,'LEGAL',NULL);

// if third column allows NULL, it success

**//in exam, be thoughtful, think about every condition**

SELECT \* FROM AP\_DEPTTEST;

SELECT \* FROM AP\_DEPTTEST;



SELECT deptno

FROM ap\_emp

WHERE empno=7369;

UPDATE ap\_emptest

SET deptno=30

WHERE empno=7369;

//if there no empno constraints, every record will be updated

SELECT empno,deptno

FROM ap\_emptest;

UPDATE ap\_emptest

SET sal=sal+100

WHERE job='CLERK';

UPDATE ap\_emptest

SET comm=comm +100;

**// if anyone’s comm is null, it won’t update because anything with null will gets null after math operation**

UPDATE ap\_emptest

SET comm=**nvl(comm**,0)+100 , ename=INITCAP(ename);

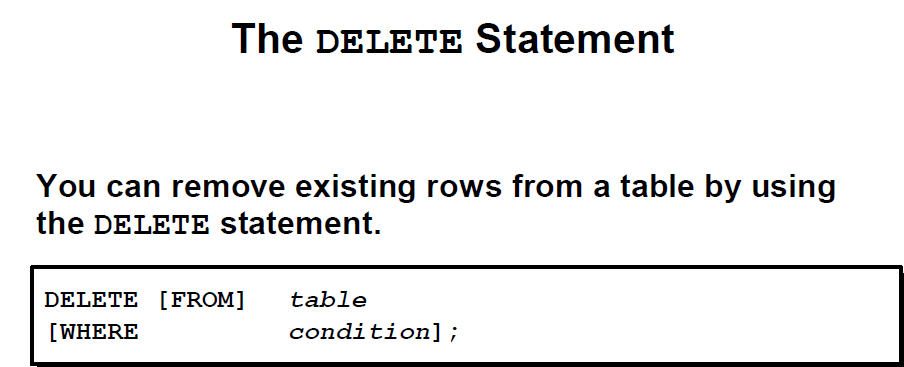
// update multiple columns together

//**INITCAP** sets first letter to uppercase and rest to be lower case

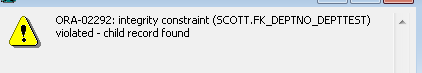
//**nvl**: if comm is null, take it as 0, so comm will updated

SELECT \* FROM ap\_emptest;

commit;



DELETE FROM depttest;



DELETE FROM ap\_emptest

WHERE deptno=30;

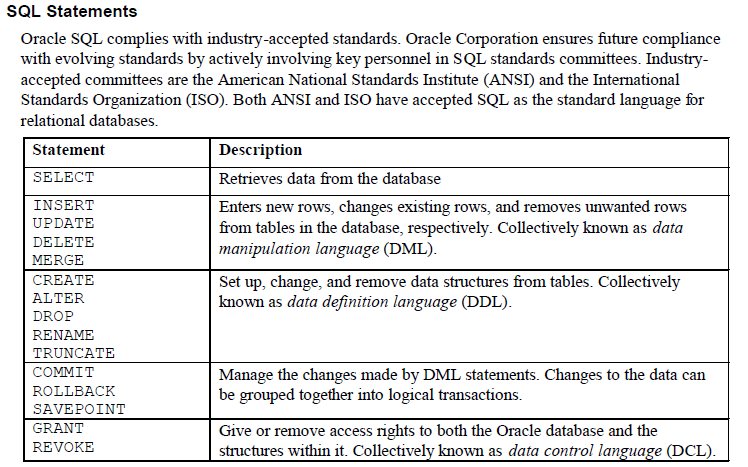
DELETE FROM ap\_emptest;

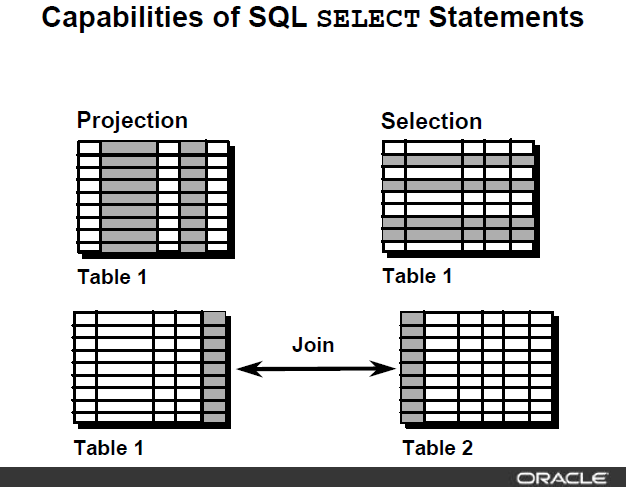
SELECT COUNT(\*) FROM ap\_emptest;enter

**Difference between DELETE and TRUNCATE**:

Truncate can’t be recover but delete can

Delete can be set with condition, truncate removes all records





**Basic SELECT structure**

**SELECT < column list, expressions, literals>**

**FROM <table list>**

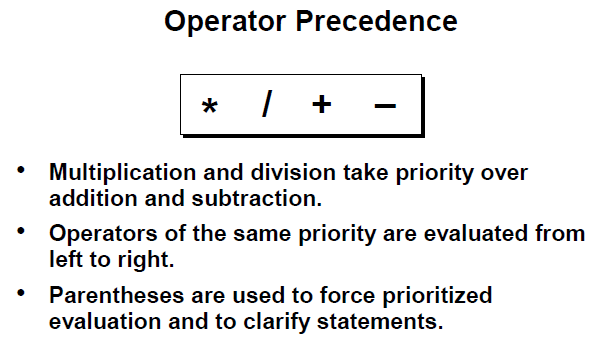
**WHERE <filter conditions with AND/OR/NOT logical operators)**

**GROUP BY <column list for aggregate functions COUNT/SUM/MIN/MAX/AVG etc.>**

**HAVING <filter conditions for grouping results>**

**ORDER BY <column list for sorting result set>**

**(SELECT and FROM clause are mandatory, all other clauses are optional and to be used as required by query result)**



SELECT \*

FROM ap\_emp;

//select all

SELECT empno, ename, job, sal

FROM ap\_emp;

//projection

SELECT empno (as is optional but recommended) "Employee Number", ename as name , job, sal "Monthly Salary USD"

FROM ap\_emp;

//display things as alias别名, only for presentation, will not change table

SELECT ename||' was hired on '||hiredate||' and current monthly salary is USD'||sal

FROM ap\_emp;

//between || || is statement out of data, for display

SELECT A.ename, A.sal, A.sal+100, A.comm, A.comm+10

from ap\_emp A;

SELECT A.ename, A.sal, A.sal+100, A.comm, nvl(A.comm,0)+10

FROM ap\_emp A;

SELECT DISTINCT job

FROM ap\_emp;

//select distinct data

SELECT DISTINCT job,deptno

FROM ap\_emp;

SELECT DISTINCT job,deptno

FROM ap\_emp

ORDER BY job, deptno;

SELECT DISTINCT job,deptno

FROM ap\_emp

ORDER BY deptno, job;

**//order 对应顺序不能反?可以**

SELECT DISTINCT job,deptno

FROM ap\_emp

ORDER BY deptno, job desc;

SELECT ename, sal

FROM ap\_emp

ORDER BY 2;

**//position 2 is sal so ordered by sal**

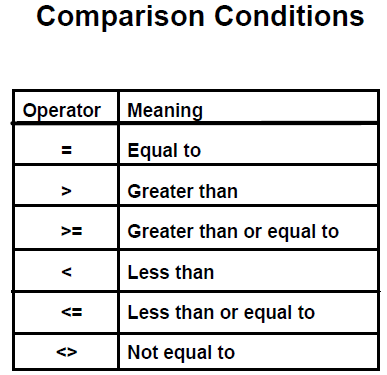
SELECT ename, sal

FROM ap\_emp

ORDER BY 2 desc, 1;

**//对于sal降序，ename升序**

WHERE clause (SELECTION of results - Limiting result sets based upon the conditions)



SELECT \*

FROM ap\_emp

WHERE sal>9000;

SELECT \*

FROM ap\_emp

WHERE sal<9000;

SELECT ename, sal AS salary

FROM ap\_emp

WHERE sal>=9000

ORDER BY salary;

SELECT ename, deptno

FROM ap\_emp

WHERE deptno=10

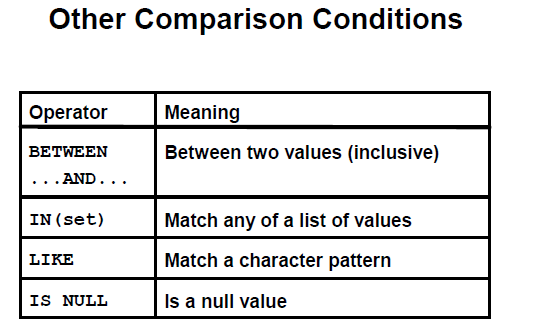
ORDER BY 1;

SELECT ename, deptno

FROM ap\_emp

WHERE deptno<>10//not equal

ORDER BY 2, 1;



SELECT ename, sal

FROM ap\_emp

WHERE sal BETWEEN 4500 AND 9000

ORDER BY sal;

SELECT ename, deptno

FROM ap\_emp

WHERE deptno IN (10,30)

ORDER BY 2;

SELECT ename, deptno

FROM ap\_emp

WHERE ename **like 'A%'**

ORDER BY 1;

**//start with A(case sensitive**)

SELECT ename, deptno

FROM ap\_emp

WHERE ename like '%N'

ORDER BY 1;

SELECT ename, deptno

FROM ap\_emp

WHERE ename like '%A%'

ORDER BY 1;

**//have A, no matter position**

SELECT ename, sal, comm

FROM ap\_emp

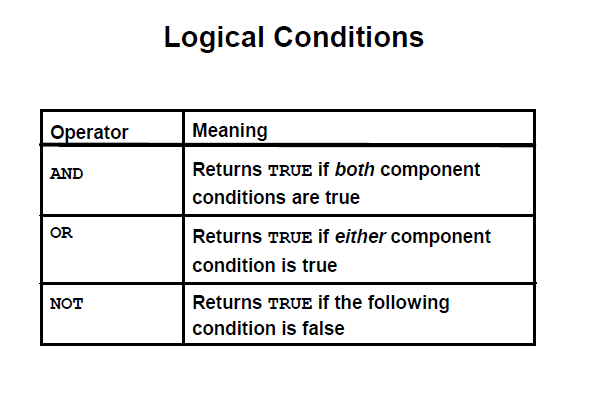
WHERE comm **IS NULL**;

//判断NULL 不能用=，用 is null

SELECT ename, sal, comm

FROM ap\_emp

WHERE comm IS NOT NULL;



SELECT ename, sal, deptno

FROM ap\_emp

WHERE sal>5000 AND deptno=10;

SELECT ename,sal, deptno

FROM ap\_emp

WHERE sal>5000 OR deptno=10;

SELECT ename,sal, deptno, job

FROM ap\_emp

WHERE sal<5000 OR deptno=10 AND job='ANALYST';

**// and优先级大于or**

SELECT ename,sal, deptno, job

FROM ap\_emp

WHERE (sal<5000 OR deptno=10) AND job='ANALYST';

SELECT ename,sal, deptno, job

FROM ap\_emp

WHERE sal<5000 OR deptno=10 OR job='ANALYST';

SELECT ename, sal

FROM ap\_emp

**//between 语句是 inclusive 的**

WHERE sal NOT BETWEEN 4500 AND 9000 //between

ORDER BY sal;

SELECT ename,deptno

FROM ap\_emp

WHERE deptno NOT IN (10,30)

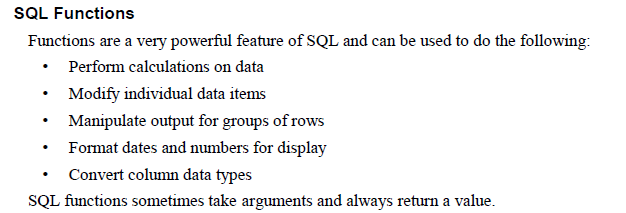
ORDER BY 2;

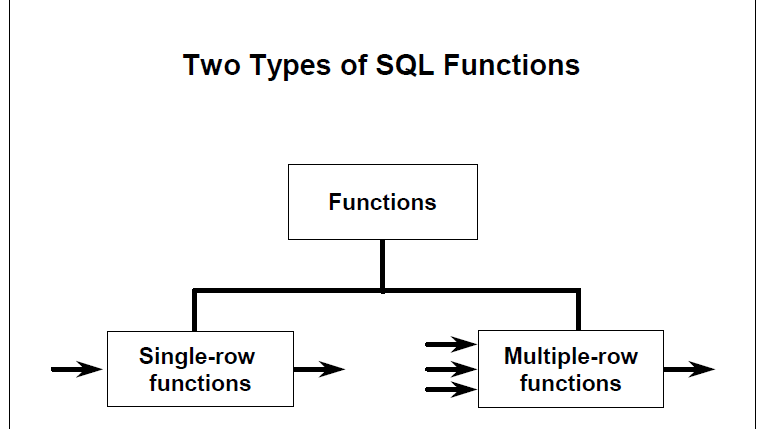
SELECT ename, deptno

FROM ap\_emp

WHERE ename NOT like 'A%'

ORDER BY 1;





**ORACLE SQL FUCTIONS**

[**https://docs.oracle.com/database/121/SQLRF/functions.htm#SQLRF006**](https://docs.oracle.com/database/121/SQLRF/functions.htm#SQLRF006)

SELECT ename,

lower(ename),

initcap(ename),

substr(ename, 1,2), **substr(column name, starting position, length)**

substr(ename, -3,2),

substr(ename, -2), //starting from -2 position , go to the end(by default

lower(substr( ename, 1,2))//function within the function

FROM ap\_emp

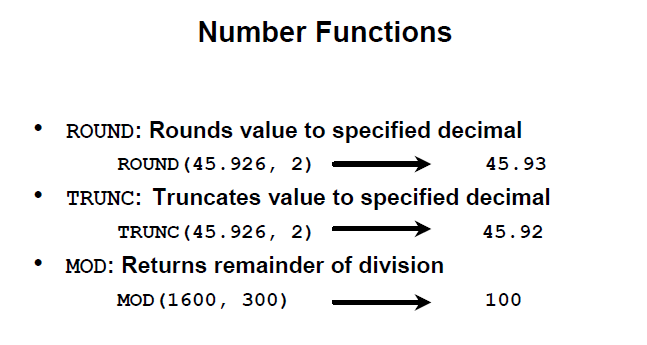
WHERE UPPER(JOB)='CLERK';

SELECT ename, LENGTH(ename), //ename’s length for letters

sal, LPAD(sal, 10,0) //length ten padding 0 to the left

FROM ap\_emp

WHERE UPPER(JOB)='CLERK';

0

SELECT **ROUND**(45.926,2), ROUND( 45.926) FROM dual;

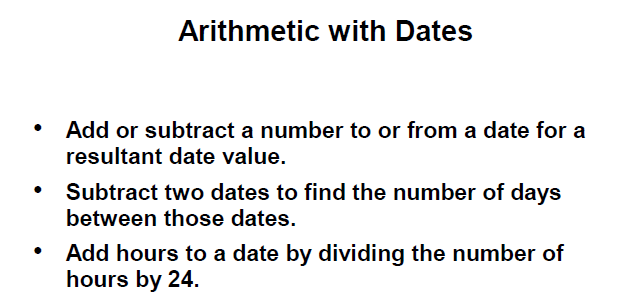
**按位数保留小数位数（四舍五入，默认保留整数**

SELECT **TRUNC**(45.926,2), TRUNC( 45.926) FROM dual;

**//直接往下降为整数**

SELECT ename, sal, **MOD**(sal, 1000)//**返回余数**

FROM ap\_emp;



SELECT sysdate //系统今日日期

FROM DUAL;//系统table？？

SELECT sysdate+14//from todays date， add two days

FROM DUAL;

SELECT SYSDATE-30

FROM DUAL;

SELECT ename, hiredate, sysdate-hiredate "No of days at work"

FROM ap\_emp;

SELECT ename, hiredate, (sysdate-hiredate)/365 "No of years at work"

FROM ap\_emp;

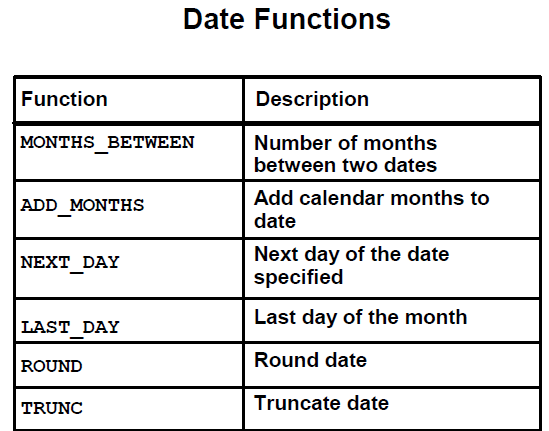
**//不做truncate，会有很多小数，trunc是直接往下降取整**

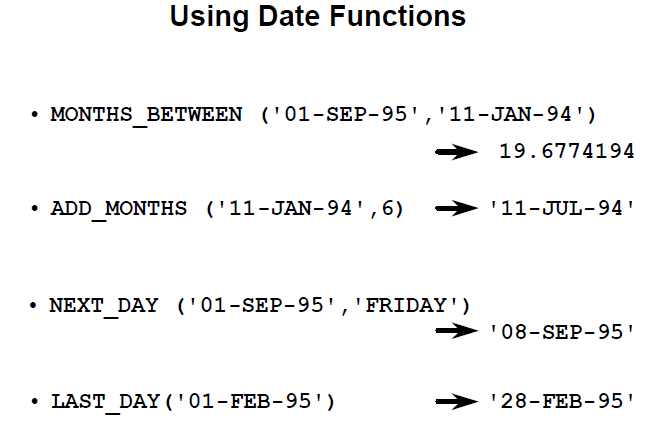
SELECT ename, hiredate, trunc( (sysdate-hiredate)/365 ) "No of years at work"

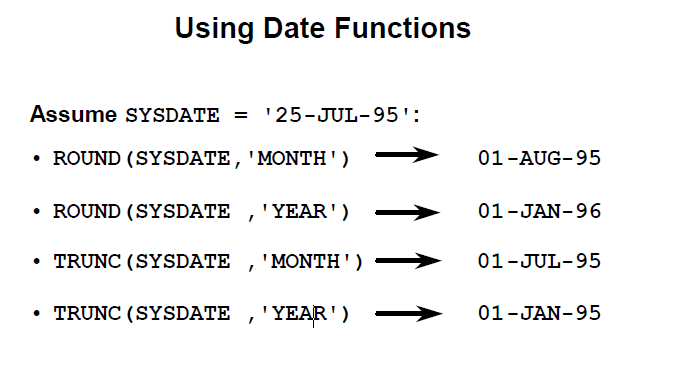
FROM ap\_emp;

SELECT ename, hiredate, round( (sysdate-hiredate)/7) "No of weeks at work"

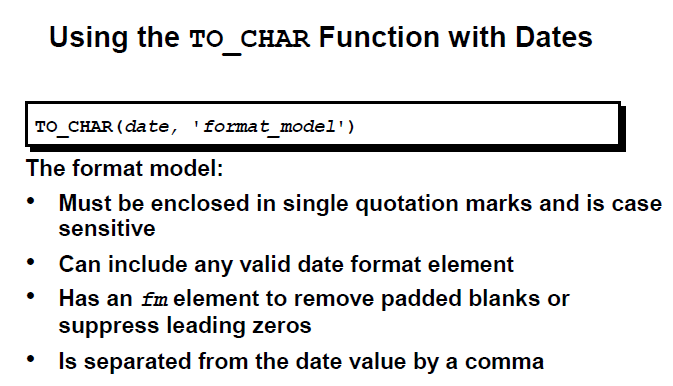
FROM ap\_emp;

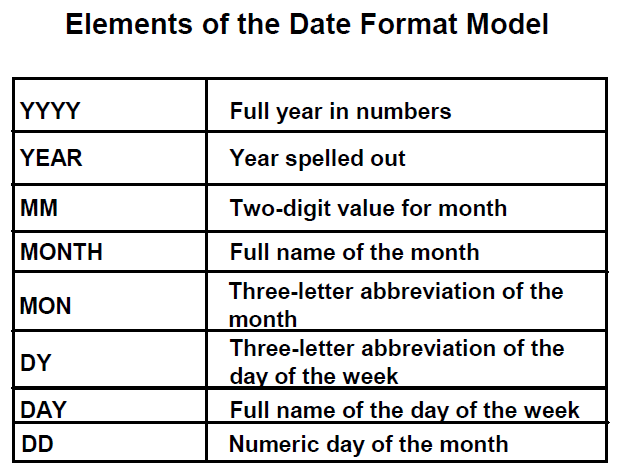


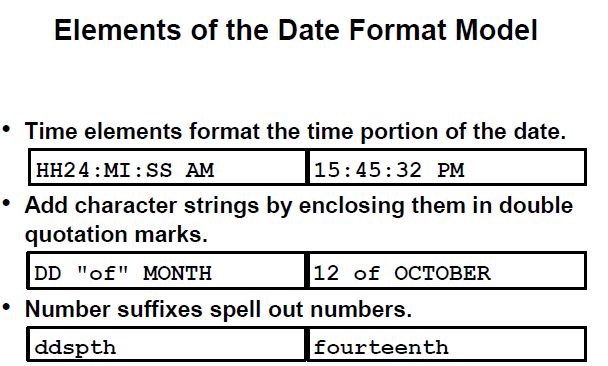




1. First date of the next month 按月round进了
2. First date of next year 按年round直接加一年
3. Return first day of this month 按月trunc直接回到本月第一天
4. return first day of this year 按年trunc直接回到本年第一天







**大小写控制显示的对应部分的大小写，AM 控制计时显示（如果是PM会自动显示PM**

SELECT to\_char(sysdate, 'dd-mon-yy hh24:mi:ss ')

FROM dual;

SELECT to\_char(sysdate, 'dd-mon-yy hh:mi:ss AM')

FROM dual;

SELECT to\_char(sysdate, 'dd-MON-yyyy hh:mi:ss AM')

FROM dual;

SELECT to\_char(sysdate, 'dd-Month,yyyy hh:mi:ss AM')

FROM dual;

SELECT to\_char(sysdate, 'ddspth -Month,yyyy hh:mi:ss AM')

FROM dual;

SELECT to\_char(sysdate, 'Ddspth -Month,yyyy hh:mi:ss AM')

FROM dual;

//spell day and month in English , FIRST LETTER FOR DAY IS UPPER CASE

SELECT to\_char(sysdate, 'Ddspth -Month "of" Year hh:mi:ss AM')

FROM dual;

//中间插入字符

SELECT to\_char (sysdate, 'DDD') // Day of year (1-366).

FROM dual;

**一年里的第几天**

SELECT to\_char (sysdate, 'DD') // Day of month (1-31).

FROM dual;

**一月里的第几天**

SELECT to\_char (sysdate, 'D') // Day of week (1-7).

FROM dual;

**一周里的第几天**

SELECT to\_char (sysdate, 'DY') // Abbreviated name of day.

FROM dual;

//show satur （selling model

SELECT to\_char (sysdate, 'Day') // Name of day.

FROM dual;

//Saturday

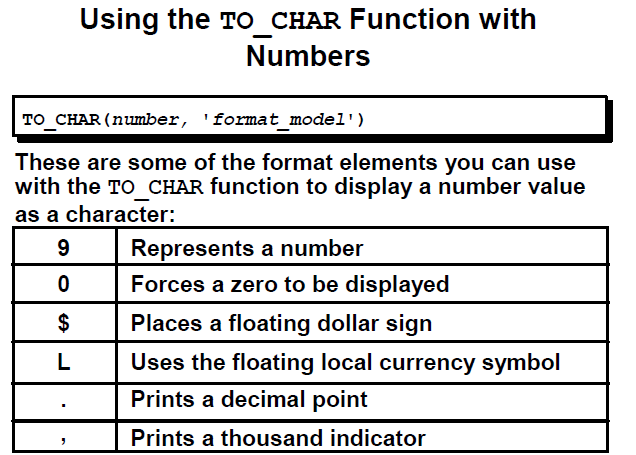
SELECT to\_char (sysdate, 'WW') // Week of year (1-53)

FROM dual;

SELECT to\_char (sysdate, 'W') // Week of month (1-5)

FROM dual;

|  |  |
| --- | --- |
| D | Day of week (1-7). |
| DAY | Name of day. |
| DD | Day of month (1-31). |
| DDD | Day of year (1-366). |
| DY | Abbreviated name of day. |
| WW | Week of year (1-53) where week 1 starts on the first day of the year and continues to the seventh day of the year. |
| W | Week of month (1-5) where week 1 starts on the first day of the month and ends on the seventh. |
| IW | Week of year (1-52 or 1-53) based on the ISO standard. |



SELECT ename, sal,

to\_char(sal, '99999'),

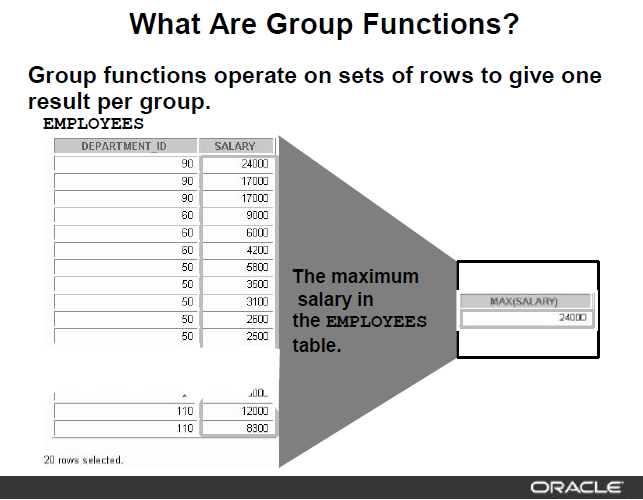
to\_char(sal, '09999.99'),

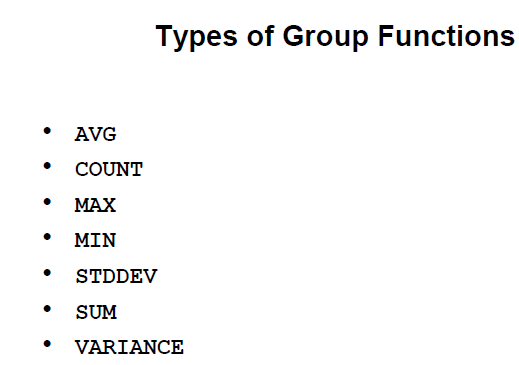
to\_char(sal, '$99999'),

to\_char(sal, '$99,999'),

to\_char(sal, 'L99999')

FROM ap\_emp;





SELECT COUNT(\*), MIN(sal), MAX(sal), AVG(sal), SUM(sal)

FROM ap\_emp

WHERE deptno=10;

SELECT MIN(hiredate), MAX(hiredate)

FROM ap\_emp

WHERE deptno=20;

SELECT COUNT( DISTINCT job)

FROM ap\_emp

WHERE deptno=20;

SELECT AVG(comm)

FROM ap\_emp;

SELECT AVG(NVL(comm,0))

from ap\_emp;

SELECT deptno, SUM(sal)

FROM ap\_emp

GROUP BY deptno

ORDER BY 2;

SELECT deptno, job, AVG(sal)

FROM ap\_emp

GROUP BY deptno, job

ORDER BY 1,2;

SELECT deptno, job, AVG(sal)

FROM ap\_emp

GROUP BY deptno, job

HAVING AVG(sal)>3000

**//因为计算出的value不在DB里，是对分组过滤后的结果再次过滤，所以不能用where，要用having，**

ORDER BY 1,2;

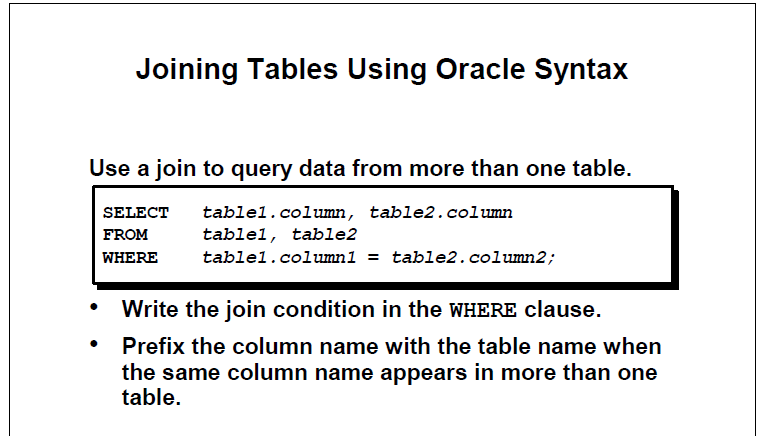
SELECT SUM(QUANTITY \* WEIGHT ) AS TOTAL\_WEIGHT, SHIPMENT\_ID,SHIPMENT\_DT,EXPECTED\_ARRIVAL\_DT,ORIGIN,DESTINATION,SHIP\_NUMBER,CAPTAIN\_ID

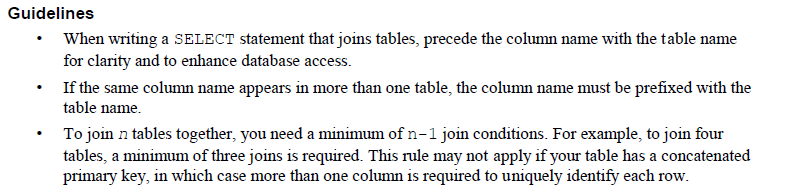
FROM SHIPMENT NATURAL JOIN SHIPMENT\_ITEM NATURAL JOIN ITEM

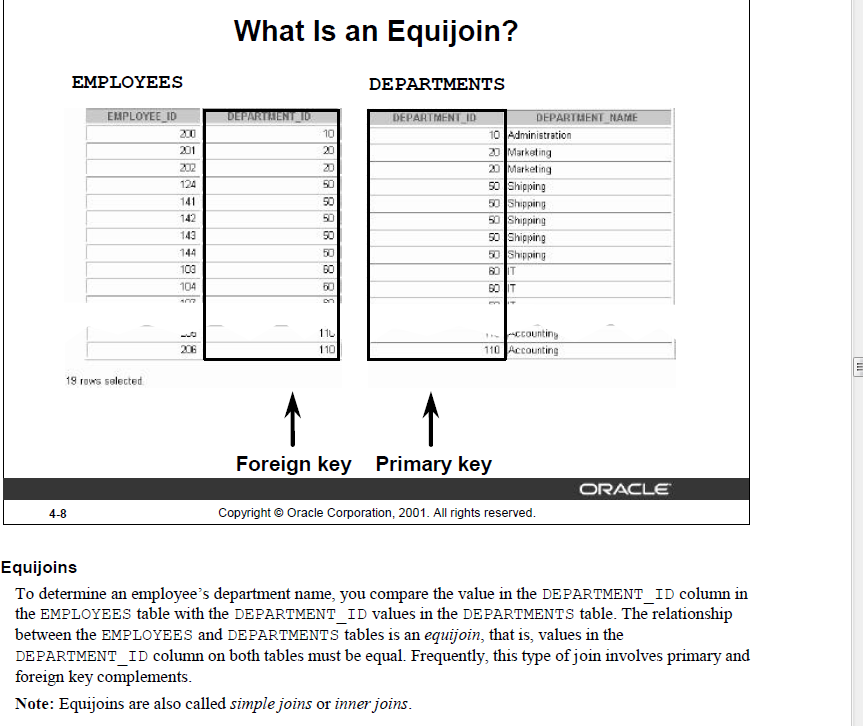
GROUP BY SHIPMENT\_ID,SHIPMENT\_DT,EXPECTED\_ARRIVAL\_DT,ORIGIN,DESTINATION,SHIP\_NUMBER,CAPTAIN\_ID

HAVING SUM(QUANTITY \* WEIGHT ) >= 1000;

**//对于HAVING，它对group by结果处理，并且在select之前执行，所以having里面不能用别名**







别名EQUI JOIN = INNER JOIN

SELECT a.empno, a.ename,a.deptno, b.dname

FROM ap\_emp a INNER JOIN ap\_dept b ON a.deptno=b.deptno

order by 3;

//inner join base on FK, PK，matching corressponding record

//**inner join and join functions the same**

SELECT a.empno, a.ename,a.deptno, b.dname

FROM ap\_emp a JOIN ap\_dept b ON a.deptno=b.deptno

order by 3;

SELECT a.empno, a.projid,b.pname,a.hours

FROM ap\_proemp a JOIN ap\_project b ON a.projid=b.projid

WHERE a.hours>50

ORDER BY 1;

SELECT a.empno, a.ename,a.deptno, b.dname,b.loc,c.hours, d.pname

FROM ap\_emp a JOIN ap\_dept b ON a.deptno=b.deptno JOIN ap\_proemp c ON a.empno=c.empno JOIN ap\_project d ON c.projid=d.projid

WHERE b.loc IN ('NEW YORK', 'DALLAS') AND c.hours>40

ORDER BY 1;

//join multiple tables, a as intersection

**NATURAL JOIN: when FK and PK column of same name, do not need to write join condition.**

SELECT empno, ename,deptno, dname – column alias on common column will return error.

FROM ap\_emp NATURAL JOIN ap\_dept;

**不允许表的别名，因为natural join 合并了重名同值column**

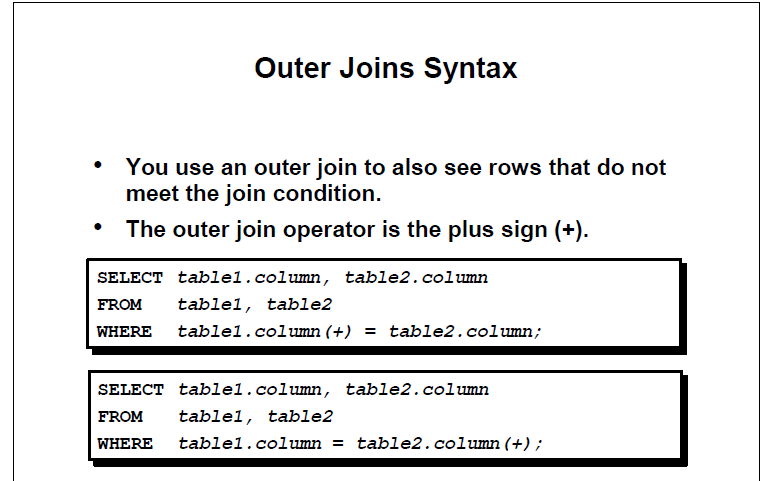
**NON-EQUIJOIN: Join based on other than equality operator: Cartesian Prodecut**

SELECT a.empno, a.ename,a.sal,b.grade, b.losal, b.hisal

FROM ap\_emp a , ap\_salgrade b

WHERE a.sal between b.losal AND b.hisal

ORDER BY b.grade;



Here are the different types of the JOINs in SQL:

* **(INNER) JOIN**: Returns records that have matching values in both tables
* **LEFT (OUTER) JOIN**: Returns all records from the left table, and the matched records from the right table
* **RIGHT (OUTER) JOIN**: Returns all records from the right table, and the matched records from the left table
* **FULL (OUTER) JOIN**: Returns all records when there is a match in either left or right table

SELECT a.empno, a.ename,b.deptno,b.dname

FROM ap\_emp a RIGHT OUTER JOIN ap\_dept b ON a.deptno=b.deptno

ORDER BY a.deptno;

SELECT b.empno, b.ename,a.deptno,a.dname

FROM ap\_dept a LEFT OUTER JOIN ap\_emp b ON a.deptno=b.deptno

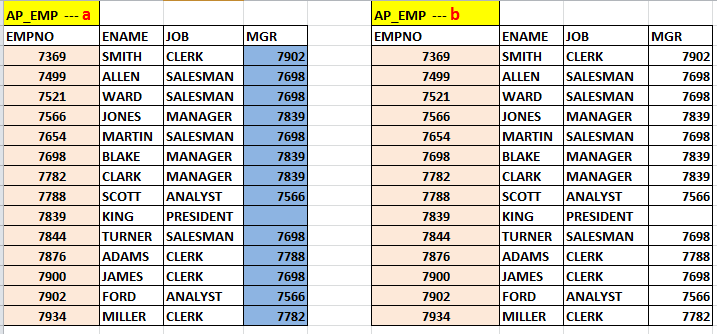
ORDER BY b.deptno;

SELECT b.empno, b.ename,a.deptno,a.dname

FROM ap\_dept a FULL OUTER JOIN ap\_emp b ON a.deptno=b.deptno

ORDER BY b.deptno;

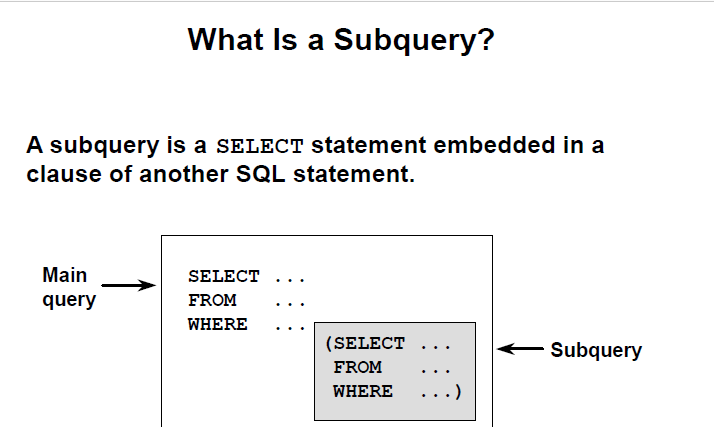
**SELF JOIN**



SELECT a.empno,a.ename,a.mgr, b.empno " Manager ID", b.ename "Manager Name"

FROM ap\_emp a JOIN ap\_emp b ON a.mgr=b.empno;

**USING SUBQUERY**



SELECT ename, sal

FROM ap\_emp

WHERE sal>( SELECT sal FROM ap\_emp WHERE ename='ALLEN');

Subquery的使用只能让其返回一行

//sql里where语句里不能直接使用聚合函数，having可以，所以有时候在subquery里用聚合返回一个值

SELECT ename, job

FROM ap\_emp

WHERE deptno=(SELECT deptno FROM ap\_dept WHERE loc='NEW YORK');

SELECT ename, job,sal

FROM ap\_emp

WHERE sal>(SELECT avg(sal) FROM ap\_emp);

SELECT ename, job,sal

FROM ap\_emp

WHERE sal>(SELECT avg(sal) FROM ap\_emp) AND

deptno= (SELECT deptno FROM ap\_dept where loc ='NEW YORK')

SELECT job, AVG(sal)

FROM ap\_emp

GROUP BY job

HAVING AVG(sal)=(SELECT MIN(AVG(sal)) FROM ap\_emp GROUP BY job);

**Ensure only one value return! 为啥不能 = min(avg(sal)) ！！！**

**记住最好用subquery**

SELECT ename, sal

FROM ap\_emp

WHERE sal > ALL ( SELECT avg(sal) FROM ap\_emp GROUP BY deptno);

**ALL MEANS MAX**

SELECT ename, sal

FROM ap\_emp

WHERE sal > ANY ( SELECT avg(sal) FROM ap\_emp GROUP BY deptno);

**ANY MEANS MIN**

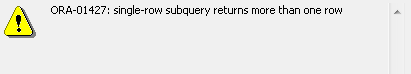
SELECT ename, sal

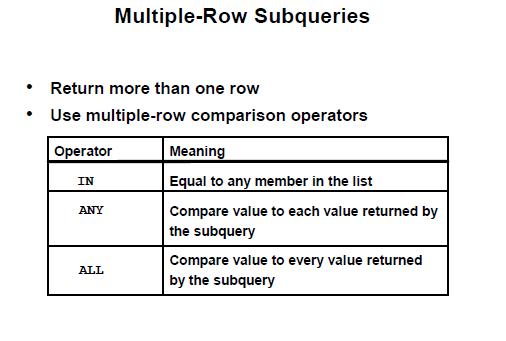
FROM ap\_emp

WHERE sal IN ( SELECT avg(sal) FROM ap\_emp GROUP BY deptno);

**IN MEANS IN THAT LIST**

**USE ALL, ANY, IN** TO SOLVE BELOW PROBLEM





SELECT ename, sal

FROM ap\_emp

WHERE sal > ALL ( SELECT avg(sal) FROM ap\_emp GROUP BY deptno);

SELECT ename, sal

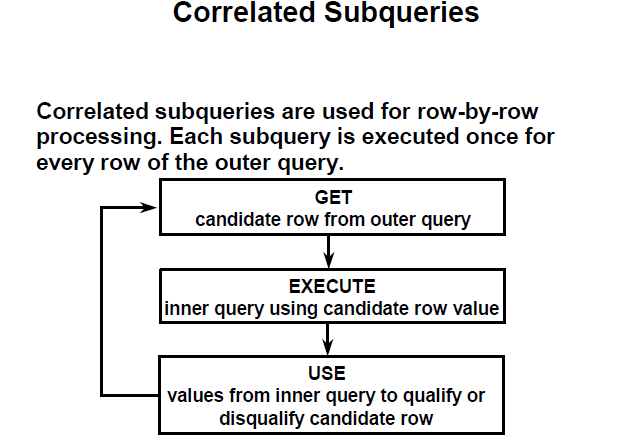
FROM ap\_emp

WHERE sal > ANY ( SELECT avg(sal) FROM ap\_emp GROUP BY deptno);

SELECT ename, sal

FROM ap\_emp

WHERE sal IN ( SELECT avg(sal) FROM ap\_emp GROUP BY deptno);



**List employees whose salary is higher than their respective department’s average salary**

SELECT a.empno,a.deptno, a.sal

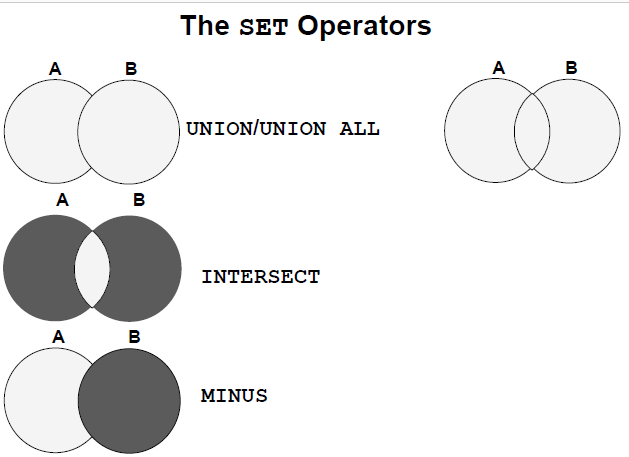
FROM ap\_emp a

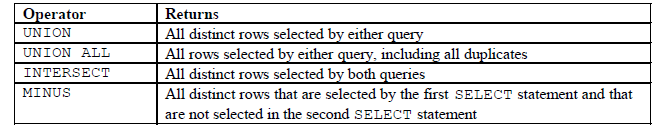
WHERE a.sal > (SELECT AVG(sal) deptavg FROM ap\_emp b WHERE a.deptno=b.deptno);

//大于自己部门的平均值

**CORRELATED SUBQUERY**:

1. START WITH OUTER QUERY WILL PASS INPUT INTO INNER QUERY
2. INNER QUERY WILL RUN FOR VALUE PASSED BY OUTER QUERY
3. RESULT OF INNER QUERY WILL PASS TO OUTER QUERY
4. OUTER QUERY WILL RUN TO SATISFY THE CONDITION





SELECT empno, ename, deptno,job

FROM ap\_emp

WHERE deptno in (20,30)

**UNION**

SELECT empno, ename,deptno, job

FROM ap\_emp

WHERE job='ANALYST'

ORDER BY deptno,job;

SELECT empno, ename, deptno,job

FROM ap\_emp

WHERE deptno in (20,30)

**UNION ALL**

SELECT empno, ename,deptno, job

FROM ap\_emp

WHERE job='ANALYST'

ORDER BY deptno,job;

**Union 和 union all 区别：union 全是distinct， union all 不处理重复**

SELECT empno, ename, deptno,job

FROM ap\_emp

WHERE deptno in (20,30)

**INTERSECT**

SELECT empno, ename,deptno, job

FROM ap\_emp

WHERE job='ANALYST'

ORDER BY deptno,job;

SELECT empno, ename, deptno,job

FROM ap\_emp

WHERE deptno in (20,30)

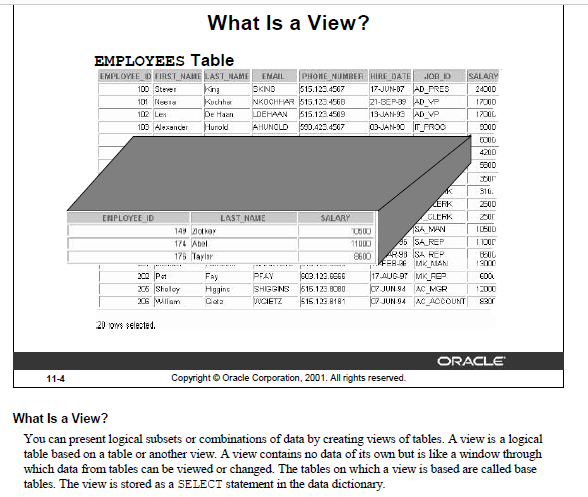
**MINUS**

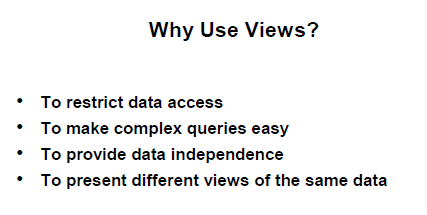
SELECT empno, ename,deptno, job

FROM ap\_emp

WHERE job='ANALYST'

ORDER BY deptno,job;





View的本质是query的别名，自身没有data

**== SIMPLE, SINGLE TABLE VIEW**

create or replace view testview

as

select \* from ap\_emp

where deptno=30

;

SELECT \* FROM testview;

update testview

set sal=sal+100

where empno=7499;

SELECT \* FROM testview;

**== READ ONLY VIEW**

create or replace view testview

as

select \* from ap\_emp

where deptno=30

with READ ONLY;

SELECT \* FROM testview;

update testview

set sal=sal+100

where empno=7499;

会失败，只能读数据，不能添加写数据操作

**== VIEW WITH CHECK OPTION**

create or replace view testview

as

select \* from ap\_emp

where deptno=30

with CHECK OPTION;

SELECT \* FROM testview;

update testview

set sal=sal+100

where empno=7499; //对应deptno=30那一行，只是改变值，改变后仍然可见success

UPDATE TESTVIEW

SET DEPTNO=20 //改变的话会使view里原来可见的行变为不可见，fail

WHERE EMPNO=7499;

不能成功。以防止插入 select\_statement 中 WHERE 子句不为真的行。它还会阻止可见行更新为不可见的行）。增删都必须使试图显示范围内！

== COMPLEX, MULTI TABLES / AGGREGATE functions

CREATE OR REPLACE VIEW dept30\_V

AS

SELECT a.empno "Employee Number", a.hiredate "Hire Date", a. sal "Monthly Salary", b.dname "Department Name", b. loc "Location"

FROM ap\_emp a, ap\_dept b

WHERE a.deptno=b.deptno AND

a.deptno=30 ;

SELECT \* FROM dept30\_V;

SELECT \* FROM USER\_VIEWS

WHERE VIEW\_NAME='DEPT30\_V';

CREATE OR REPLACE VIEW emp\_proj\_detail\_V

AS

SELECT a.empno AS Employee, a.ename AS name, a.hiredate AS HireDate, a. sal AS Salary, b.dname AS Department, c.pname AS Project, d.hours AS Hours

FROM ap\_emp a, ap\_dept b, ap\_project c, ap\_proemp d

WHERE a.deptno=b.deptno AND

a.empno=d.empno AND

d.projid=c.projid

ORDER BY 4;

SELECT \* FROM emp\_proj\_detail\_V;

SELECT name,sum(hours)

FROM ap\_emp\_proj\_detail\_V

GROUP BY name

ORDER by 2;

CREATE OR REPLACE VIEW dept\_summary\_V

(name,location,minsal,maxsal,avgsal)

AS

SELECT dname , loc, MIN(sal), MAX(sal) , AVG(sal)

FROM ap\_emp a, ap\_dept b

WHERE a.deptno=b.deptno

GROUP BY dname,loc;

SELECT \* FROM dept\_summary\_V;

**==== SUBQUERY in FROM CLAUSE**

* **List employee number, name, department, salary along with their respective department’s total number of employee, total salary, average salary, minimum, and maximum salary**

**SELECT a.empno, a.ename, a.deptno,sal, b.deptempcnt , b.depttotsal , trunc(b.deptavgsal) deptavgsal ,**

**b.deptminsal, b.deptmaxsal**

**FROM ap\_emp a, ( select deptno, count(\*) deptempcnt, sum(sal) depttotsal, avg(sal) deptavgsal, min(sal)**

**deptminsal, max(sal) deptmaxsal from ap\_emp group by deptno ) b**

**WHERE a.deptno=b.deptno ;**

**==== WITH CLAUSE [just like a temporary table that can be used multiple times in a query as needed**

**WITH DEPTVAL AS //创建临时表**

**( select deptno, count(\*) deptempcnt, sum(sal) depttotsal, avg(sal) deptavgsal, min(sal) deptminsal, max(sal) deptmaxsal from ap\_emp group by deptno)**

**SELECT a.empno, a.ename, a.deptno,sal, b.deptempcnt , b.depttotsal , trunc(b.deptavgsal) deptavgsal, b.deptminsal, b.deptmaxsal**

**FROM ap\_emp a, DEPTVAL b**

**WHERE a.deptno=b.deptno ;**

**== CASE statement [Conditional processing]**

**Increase employee’s salary by $500 if their salary is more than $5000, increase by $300 if their salary is over $3000, otherwise increase by $100.**

**select sal, case**

**when sal>5000 then sal+500**

**when sal>3000 then sal+300**

**else**

**sal+100**

**end “Salary Raise”**

**from ap\_emp;**

**update ap\_emp**

**set sal = case**

**when sal>5000 then sal+500**

**when sal>3000 then sal+300**

**else**

**sal+100**

**end ;**

**== RANK function for TOP-N queries**

* **List employee number and name of the employee who worked on highest number of projects**

**WITH PROJCNTRANK**

**AS**

**( select empno,count(projid) cnt,**

**rank () over (order by count(projid) desc) as myrank**

**from ap\_proemp**

**group by empno )**

**SELECT A.EMPNO, A.ENAME, B.CNT, B.MYRANK**

**FROM AP\_EMP A JOIN PROJCNTRANK B ON A.EMPNO=B.EMPNO AND**

**MYRANK<2;**

**-- Subquery in SELECT clause**

**-- List of depatment name, and total number of employyees in those department**

**select dname name,**

**(select count(\*)**

**from ap\_emp B**

**where A.deptno= B.deptno)**

**as Num\_Of\_Employees**

**from ap\_dept A ;**

**-- Subquery in HAVING clause**

**-- Department wise Total Salary for those departments which has total salary more than that of department 30**

**select deptno,sum(sal) Total\_Salary**

**from ap\_emp**

**group by deptno**

**having sum(sal) >= (select sum(sal) from ap\_emp where deptno=30);**