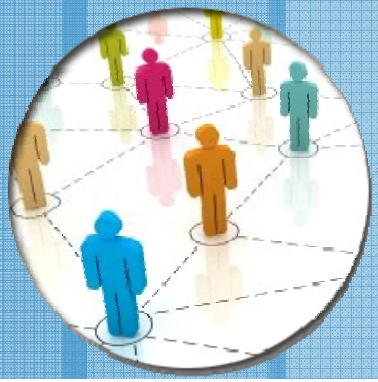
Database Management System

Lab 02

Basic data retrieval operations in SQL



Capabilities of select statement

1. **SELECTION (σ)**

   The selection capability can be used to choose rows in a table depending on the criteria to selectively restrict the rows.

Examples

a. Selecting all employees whose salary is between 3500 and 5000 and who were hired after 31st July, 1981.

```
SELECT * FROM EMP
WHERE (SAL BETWEEN 3500 AND 5000)
AND HIREDATE > TO_DATE('31-JUL-1981', 'DD-MON-YYYY');
```

OR

```
SELECT * FROM EMP
WHERE (SAL BETWEEN 3500 AND 5000)
AND HIREDATE > '31-JUL-1981'
```

Capabilities of select statement

b. Selecting all employees whose job is either clerk or analyst and were hired between 23rd July, 1981 and 14th May, 1982

```
SELECT * FROM EMP
WHERE (JOB = 'CLERK' OR JOB = 'ANALYST') AND HIREDATE
BETWEEN TO_DATE('23-JUL-1981', 'DD-MON-YYYY') AND
TO_DATE('14-MAY-1982', 'DD-MON-YYYY');
```

OR

```
SELECT * FROM EMP
WHERE (JOB = 'CLERK' OR JOB = 'ANALYST') AND HIREDATE
>='23-JUL-1981' AND HIREDATE <='14-MAY-1982'
```

Capabilities of select statement

2. **Projection (π)**

When we don't need all columns of a table we chose few columns this is called projection.

Examples

a. Selecting employee number, name and their job.

```
SELECT EMPNO, ENAME, JOB
FROM EMP;
```

b. Selecting employee number, name and their salary who do not earn commission

```
SELECT EMPNO, ENAME, SAL
FROM EMP
WHERE COMM IS NULL;
```

Capabilities of select statement

2. **Join (θ)**

   To bring together data that is stored in different tables by creating a link through a column that both the tables share.

Examples

To retrieve the employee name, their job and department name, we need to extract data from two tables, EMP and DEPT. This type of join is called *Equi join* that is values in the DEPTNO column must be equal in both tables. *Equi* join is also called *simple join* or *inner join*.
(Joins will be discussed in Lab 03)

```
SELECT E.ENAME, E.JOB, D.DNAME
FROM EMP E,DEPT D
WHERE E.DEPTNO=D.DEPTNO;
```

Comparison Operators

Comparison operators are used in conditions that compare one expression to another. They are used in the *WHERE* or *HAVING* clause of the *SELECT* statement.

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to

Besides basic comparison operators above, Oracle SQL also supports following comparison operators

Operator	Meaning
BETWEEN … AND …	Between two values (inclusive)
IN (list)	Match any of a list of values
LIKE	Match a character pattern
IS NULL	Is a null value

Comparison Operators

Examples

a. To display record of employees who are not managers.

```
SELECT * FROM EMP
WHERE JOB <> 'MANAGER';
```

b. To display the employee number, name, salary and the manager's employee number of all the employees whose manager's employee number is 7902, 7566, or 7788.

```
SELECT EMPNO, ENAME, SAL, MGR
FROM EMP
WHERE MGR IN (7902, 7566, 7788);
```

c. To display the names of all employees with names starting with **S:**

```
SELECT ENAME
FROM EMP
WHERE ENAME LIKE 'S%';
```

OR

```
SELECT ENAME
FROM EMP
WHERE ENAME = 'S%';
```

Note: Above query performs wildcard searches using LIKE operator. Here % symbol represents any sequence of zero or more characters.

Comparison Operators

d. To display the names of all employees with second character of name as **A,**

```
SELECT ENAME
FROM EMP
WHERE ENAME LIKE '_A%';
```

Note: "_" character represents any single character

Logical Operators

Operator	Meaning
AND	Returns TRUE if both component conditions are TRUE
OR	Returns TRUE if either component condition is TRUE
NOT	Returns TRUE if the following condition is FALSE

Examples

a. To display record of all clerks who earn more than 1100

```
SELECT empno, ename, job, sal
FROM emp
WHERE job = 'CLERK'
AND Sal> 1100
```

b. To display record of all employees who are either clerks or earn more than 1100.

difference

```
SELECT empno, ename, job, sal
FROM emp
WHERE job = 'CLERK'
OR Sal> 1100
```

Logical Operators

c. To display name and job title of all the employees who are not Clerk, Manager or Analyst

```
SELECT ename, job
FROM emp
WHERE job NOT IN ('CLERK', 'MANAGER', 'ANALYST');
```

Rules of Precedence

Order Evaluated	Operator
1	All comparison operators
2	NOT
3	AND
4	OR

We will use Underlines to understand precedence concept

Let take a query Example:

```
SELECT ename, job, sal FROM emp
WHERE job = 'SALESMAN'
OR job = 'PRESIDENT'
AND sal > 1500;
```

As per precedence Comparison=

AND Comparison=

OR Comparison=

Rules of Precedence

In order to force the OR operator to be evaluated before AND, use parentheses as follows

```
SELECT ename, job, sal FROM emp
WHERE (job = ‘SALESMAN’
OR job = ‘PRESIDENT’)
AND sal > 1500;
```

Ordering Data

The ***ORDER BY*** clause can be used to sort the rows.
ORDER BY DESC specifies descending order whereas ASC is the default order.

Examples

a. To select data in the increasing order of hiredate

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

b. To select data in the decreasing order of hiredate

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

Ordering Data

c. To sort by multiple columns

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

Note: The DESC applies only to SAL column. The DEPTNO appears in ascending order.

d. To select list of names and jobs of all employees hired in 1987 in the alphabetical order of name

```
SELECT UPPER(ENAME) “EMP NAME”,JOB
FROM EMP
WHERE TO_CHAR(HIREDATE, ‘YYYY’) = ‘1987’
ORDER BY ENAME;
```

Ordering Data

Before part (e) lets take few examples

- Example: To print employee number, name, job, annual salary of all employees and suppose employees take commission on monthly basis

  We must know

Logical Operator Precedence	
*, /	multiplication, division
+, -, ||	addition, subtraction, concatenation

Annual Salary = 12 (salary + Commission)
for monthly commision
Annual Salary = 12 (salary) + Commission
for Annual commision

Ordering Data

Note: The **NVL**( ) function is used to replace NULL value with another value. It is similar to the IFNULL Function in MySQL and the ISNULL Function in **SQL** Server.

Solution

```
SELECT EMPNO, ENAME, JOB, 12*(SAL + NVL(COMM, 0))
ANNUAL_SALARY
FROM EMP
```

In case of “ALLEN” ANNUAL SALARY would be 228220

- Example: To print employee number, name, job, annual salary of all employees and suppose employees take commission on Annual basis

```
SELECT EMPNO, ENAME, JOB, 12*SAL + NVL(COMM, 0)
ANNUAL_SALARY
FROM EMP
```

In case of “ALLEN” ANNUAL SALARY would be 19500

Ordering Data

e. To print employee number, name, job, annual salary of all managers and clerks whose monthly salary is between 3000 and 5500 in descending order of annual salary. Assume Annual Commissions

Solution

```
SELECT EMPNO, ENAME, JOB, 12*SAL + NVL(COMM, 0)
ANNUAL_SALARY
FROM EMP
WHERE JOB = 'MANAGER' OR JOB = 'CLERK'
AND SAL BETWEEN 3000 AND 5500
ORDER BY ANNUAL_SALARY DESC;
```