Data Retrieval Language

Presented by

Sagar

Java Consultant

Getting Specific Column Data

- **SELECT** statement is used to retrieve data from the table
- It can be used to retrieve a column or multiple columns from a table. Example:
- To get the names of all the departments

```
SELECT DNAME FROM DEPT;
```

- List the names of the columns separated by a comma to retrieve multiple columns

```
SELECT DNAME, LOC FROM  DEPT;
```

- * is used to retrieve all columns from the table

```
SELECT * from DEPT;
```

Using ALL

- **ALL** is used with **SELECT** statement to retrieve all the dept names. The **ALL** keyword is optional.
  - Example: To retrieve all dept name from table DEPT

```
SELECT ALL DName from DEPT;
```

  - By-default all the values will be retrieved from the column though **ALL** is not specified

Using DISTINCT

- **DISTINCT** keyword is used with **SELECT** statement to retrieve unique values from a column
  – Example :

  To retrieve unique location from the table DEPT

```
SELECT DISTINCT LOC FROM DEPT;
```

Filtering Rows

- It is common to filter the results to get only the information required than selecting all the records in a table.
- We can use WHERE clause with SELECT statements to filter results to get only the data which is required.

Syntax :

```
SELECT  COL1,COL2,.........FROM
TABLE_NAME WHERE  < SEARCH CONDITION>;
```

Example:

```
SELECT EMPNO,ENAME,DEPTNO FROM
EMP WHERE JOB='ANALYST';
```

Sample output

EMPNO	ENAME	SAL	DEPTNO
7788	SCOTT	3000	20
7902	FORD	3000	20

Operators in WHERE Clause

- Operators can be used with WHERE clause

Operator	Description
<,>,<=,>=,=,<> or !=	It is used to compare values
IN	It selects from a list of values
BETWEEN	Selects from a range of values
NULL	Compares with Null values
LIKE	Compares with the pattern

Comparison Operators

- Comparison operators are used to compare the values from the left hand side of a relational operator with that on the right hand side
- Relational Operators are: **= , < , > , <= , >= , != or < >**

Example: To retrieve all the EMPNO,ENAME columns which belong to the DEPTNO 10 :

```
SELECT   EMPNO,ENAME
FROM      EMP
WHERE   DEPTNO = 10;
```

Sample output

```
EMPNO      ENAME
====================
              7782
CLARK
  7839        KING
  7934        MILLER
```

Using IN Operator

- The **IN** operator is generally used to retrieve values from a list of values called as inclusive list.
- Inclusive list is a list which contains multiple values to be retrieved for the same column.
- The IN operator checks the database to see if the specified column matches one or more of the values listed inside the list.

Example : To get the list of Employees working as MANAGER and ANALYST:

```
SELECT  EMPNO,ENAME FROM
EMP
WHERE  JOB  IN
('MANAGER' ,'ANALYST');
```

Sample Output

```
EMPNO ENAME
===============
7566   JONES
7698   BLAKE
7782   CLARK
7788   SCOTT
7902   FORD
```

Using BETWEEN Operator

- The **BETWEEN** operator allows to specify a range of values.
- Where the range consists of lower range and a higher range.
- It eliminates the use of >= and <= Relational operators
- BETWEEN operator can be modified using AND

Example: To get EMPNO,ENAME,JOB,SAL where salary is in the range 1000 to 2000 (inclusive) :

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

Sample output

EMPNO	ENAME	JOB	SAL
7499	ALLEN	SALESMAN	1600
7521	WARD	SALESMAN	1250
7654	MARTIN	SALESMAN	1250
7844	TURNER	SALESMAN	1500
7876	ADAMS	CLERK	1100
7934	MILLER	CLERK	1300

Using LIKE Operator

- The LIKE operator allows you to use **wildcard characters** when searching a character field.

- A wildcard character matches any one or more characters in a string.

- 2 Types of Wildcards
  - % : This wildcard matches multiple characters.
  - _ : This wildcard matches exactly one character.

Using LIKE Operator (Contd…)

- Examples :

  - A% → indicates matching data starting with A and having any number of characters.
  - %S → indicates matching data ending with S and having any number of characters.
  - A_R%S → indicates matching data starting with A, having r as the third character ending with S.
  - __S_A → indicates matching data having S as the third character and A as the fifth character. This data should be of 5 characters only.

Using LIKE Operator (Contd…)

Example: To get details of employees whose name begin with letter S :

```
SELECT EMPNO,
ENAME, JOB,SAL
FROM EMP WHERE
ENAME LIKE 'S%';
```

EMPNO	ENAME	JOB	SAL
7369	SMITH	CLERK	800
7788	SCOTT	ANALYST	3000

Example: To get details of employees whose name starts with J, and the fourth letter as E and the remaining can be anything.

```
SELECT EMPNO, ENAME,
JOB,SAL FROM EMP WHERE
ENAME LIKE 'J__E%'
```

EMPNO	ENAME	JOB	SAL
7566	JONES	MANAGER	2975
7900	JAMES	CLERK	950

Using Logical operators

- In WHERE statement AND and OR logical operators is used to test more than one condition

Operator	Description
AND	Both the condition on the left-hand side of the operator and the condition on the right-hand side must evaluate to true
OR	Either condition must evaluate to true. If the condition on the left-hand side of the operator evaluates to true, the next condition is not evaluated.
NOT	Negates the condition. It can be used with IN, BETWEEN and LIKE operator

Using Logical operators (Contd…)

Example: To get EMPNO, ENAME, JOB, SAL of all employees having salary more than 2000 and working in DEPTNO 20.

```
SELECT EMPNO,
ENAME, SAL, DEPTNO
FROM EMP WHERE
SAL >2000 AND
DEPTNO=20
```

EMPNO	ENAME	SAL	DEPTNO
7566	JONES	2975	20
7788	SCOTT	3000	20
7902	FORD	3000	20

Example: To get EMPNO, ENAME, JOB of all employees working as SALESMAN or those employees whose names start with A.

```
SELECT EMPNO,ENAME,
JOB FROM EMP WHERE
JOB= 'SALESMAN'
OR ENAME LIKE 'A%'
```

EMPNO	ENAME	JOB
7499	ALLEN	SALESMAN
7521	WARD	SALESMAN
7654	MARTIN	SALESMAN
7844	TURNER	SALESMAN
7876	ADAMS	CLERK

Using Logical operators (Contd…)

Example: To get EMPNO, ENAME, JOB, SAL of all employees whose names do not start with A.

```
SELECT EMPNO,
ENAME,JOB
FROM EMP WHERE
ENAME NOT LIKE 'A%'
```

EMPNO	ENAME	JOB
7369	SMITH	CLERK
7521	WARD	SALESMAN
7566	JONES	MANAGER
7654	MARTIN	SALESMAN
7698	BLAKE	MANAGER
7782	CLARK	MANAGER
7788	SCOTT	ANALYST
7839	KING	PRESIDENT
7844	TURNER	SALESMAN
7900	JAMES	CLERK
7902	FORD	ANALYST
7934	MILLER	CLERK

Using NULL Operator

- NULL values are also known as unknown values.
- In RDBMS, inapplicable or missing information is known as NULL.
- The IS NULL condition tests for nulls.

Example: To get EMPNO, ENAME and SAL from EMP who do not receive a commission.

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

Sample output

EMPNO	ENAME	SAL	COMM
7369	SMITH	800	
7566	JONES	2975	
7698	BLAKE	2850	
7782	CLARK	2450	
7788	SCOTT	3000	
7839	KING	5000	
7876	ADAMS	1100	
7900	JAMES	950	
7902	FORD	3000	
7934	MILLER	1300	

Using NOT NULL Operator

- NOT NULL is used to check whether a particular column contains a value
- We can use NOT NULL to select a column which contain values

Example: To get EMPNO, ENAME and SAL from EMP who do not receive a commission.

```
SELECT EMPNO, ENAME,
SAL FROM EMP
WHERE COMM IS
NOT NULL;
```

Sample output

EMPNO	ENAME	SAL	COMM
7499	ALLEN	1600	300
7521	WARD	1250	500
7654	MARTIN	1250	1400
7844	TURNER	1500	0

Using Arithmetic Operators

- Arithmetic operators along with the column names can be used to retrieve values
- Arithmetic operations can be performed on a single column or multiple columns
- The Arithmetic operators normally used are: +, - , / and *

Example: To get EMPNO and TOTALSAL of all employees

```
SELECT
EMPNO,SAL+ COMM
TOTALSAL FROM
EMP WHERE COMM
IS NOT NULL;
```

Sample output :

EMPNO	TOTALSAL
7499	1900
7521	1750
7654	2650
7844	1500

Using Column Aliases

- An alias is another name for a collection of data like a column or a table.
- Aliases increases the readability of the column name or a table name.
- Aliases are effective when we use complicated joins or sub queries

Example demonstrating the use of aliases:

```
SELECT EMPNO EMPLOYEENO,ENAME "EMP NAME",
SAL SALARY FROM EMP WHERE SAL <1000
```

Sample output :

```
EMPLOYEENO EMP NAME      SALARY
==========================
7369          SMITH          800
7900          JAMES          950
```

Using Order by Clause

- **ORDER BY** Clause allows to list the contents in
  - Ascending or
  - Descending order
- The default order is ASCENDING
- ORDER BY can be used either with a column name or with a column position.

Syntax :

```
SELECT COL1,COL2,.......
      FROM TABLE_NAME
      WHERE <SEARCH CONDITION>
      ORDER BY COL-NAME [DESC|ASC];
```

Example – ORDER BY clause

Example: To get EMPNO and ENAME of employees who are working as ANALYST in descending order of names.

```
SELECT  EMPNO, ENAME
FROM  EMP WHERE
JOB='ANALYST'
ORDER BY ENAME DESC;
```

EMPNO	ENAME
7788	SCOTT
7902	FORD

Example: To get ENAME, JOB of employees who are working in deptno 10 and 20 in order of job and sal in descending order.

```
SELECT ENAME, JOB FROM
EMP WHERE DEPTNO
IN(10,20) ORDER BY 2,1
DESC;
```

ENAME	JOB
SCOTT	ANALYST
FORD	ANALYST
SMITH	CLERK
.....	.....

NVL Function

- Converts a null value to a non-null value.

Syntax :

```
NVL(arg1, arg2)
```

- If argument1 is not null, then it displays its value.
- If argument 1 is null , then it displays arg2.

Example:

```
SELECT NVL(COMM,0) FROM EMP
```

NVL Function (Contd...)

```
SELECT NVL(COMM,0)
AS COMMISION FROM EMP
```

```
COMMISION
===========
        0
      300
      500
        0
     1400
        0
        0
        0
        0
        0
        0
        0
        0
        0
        0
```

Sample Output

```
SELECT NVL(TO_CHAR
(COMM),'NO COMMISION')
AS COMMISION FROM EMP
```

```
COMMISION
===========
NO COMMISION
300
500
NO COMMISION
1400
NO COMMISION
NO COMMISION
NO COMMISION
NO COMMISION
0
NO COMMISION
NO COMMISION
NO COMMISION
NO COMMISION
```

Assignment

1. Display the STUDENT ID, STUDENT FIRST NAME from the STUDENT_INFO table whose name begins with letter “A”.
2. Display only the STUDENT ID of all the students from STUDENT_INFO.
3. Display the FIRST NAME of all the students who live in “New York” OR who live in “Mumbai”.
   - Using the OR operator
   - Using the IN operator

Assignment …

4. Display the STUDENT ID and STUDENT FIRST NAME from STUDENT_INFO table who's STUDENT FIRST NAME has 'e' as the last letter.
5. Display the details of the instructor "Wong" from the INSTRUCTOR_INFO table.
6. Display the STUDENT ID between 11433 and 11435 from STUDENT_INFO table.
7. Display all the details of the student from STUDENT_INFO table whose lives in "New York" and whose STUDENT ID is greater than 11434.
8. Display all the records of STUDENT_INFO in the increasing order of STUDENT_ID column.

Assignment …

9. Display the first name and last name of the instructor who has not updated his city details in the table INSTRUCTOR_INFO
10. Display the details of all the instructors who live in “washington” (search should not be case sensitive)
11. Display the records from INSTRUCTOR_INFO where the Instructor id ends with ‘09’.
12. Write the query to display the details from INSTRUCTOR_INFO as shown below:

```
1109--Kenta--Chicago
109--Leia--Washington
120--Tony--St.Charles
130--Joe--
```

Assignment …

13. Write the query to display the details from INSTRUCTOR_INFO as shown below

Sap Id	Name
1109	Kenta Wong
109	Leia Chow
120	Tony Adams
130	Joe Black

13. Display the student details who live in “Delhi” and Last name is ”Joshi”

Assignment …

14. Assuming INSTRUCTOR_INFO has the following details:

INSTRUCTOR_ID	INSTRUCTOR_FIRST_NAME	INSTRUCTOR_LAST_NAME	CITY
1109	Kenta	Wong	Chicago
109	Leia	Chow	Washington
120	Tony	Adams	St.Charles
130	Joe	Black	

The expected outcome is as shown below – notice the circled area.

1109	Kenta	Wong	Chicago
109	Leia	Chow	Washington
120	Tony	Adams	St.Charles
130	Joe	Black	Not updated


Thank you!