DBMS/SQL

Data Query Language (The Select Statement)

Capgemini

Lesson Objectives

To understand the following topics:

- The SELECT statement
- The WHERE clause
- The Mathematical, Comparison and Logical operators
- The DISTINCT clause
- The ORDER BY clause
- Tips and Tricks in SELECT Statement

3.1: The SELECT Statement

The Select Statement and Syntax

The SELECT command is used to retrieve rows from a single table or multiple Tables or Views.

- A query may retrieve information from specified columns or from all of the columns in the Table.
- It helps to select the required data from the table.

```
SELECT [ALL | DISTINCT] { * | col_name,...}
FROM table_name alias,...
     [ WHERE expr1 ]
     [ CONNECT BY expr2 [ START WITH expr3 ] ]
     [ GROUP BY expr4 ] [ HAVING expr5 ]
     [ UNION | INTERSECT | MINUS SELECT ... ]
     [ ORDER BY expr | ASC | DESC ];
```

3.1: The SELECT Statement

Selecting Columns

Displays all the columns from the student_master table

```
SELECT  *
        FROM student_master;
```

Displays selected columns from the student_master table

```
SELECT student_code, student_name
        FROM student_master;
```

3.2: SELECT statement Clauses

The WHERE clause

The WHERE clause is used to specify the criteria for selection.

- For example: displays the selected columns from the student_master table based on the condition being satisfied

```
SELECT  student_code,  student_name, student_dob
            FROM student_master
        WHERE dept_code = 10;
```

3.2: SELECT statement Clauses

The AS clause

The AS clause is used to specify an alternate colum heading.

- For example: displays the selected columns from the student_master table based on the condition being satisfied. Observe the column heading

```
SELECT student_dob as "Date of Birth"
            FROM student_master
        WHERE dept_code = 10;
 -- quotes are required when the column heading contains a
space
```

```
SELECT student_dob   "Date of Birth"
            FROM student_master
        WHERE dept_code = 10;

-- AS keyword is optional
```

3.2: SELECT statement Clauses

Character Strings and Dates

Are enclosed in single quotation marks
Character values are case sensitive
Date values are format sensitive

```
SELECT  student_code, student_dob
        FROM  student_master
        WHERE student_name = 'Sunil' ;
```

3.3: SELECT statement Clauses

Mathematical, Comparison & Logical Operators

Mathematical Operators:
- Examples: +, -, *, /

Comparison Operators:

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or Equal to
<	Less than
<=	Less than or Equal to
<>, !=, or ^=	Not Equal to

Logical Operators:
- Examples: AND, OR, NOT

3.3: SELECT statement Clauses

Other Comparison Operators

Other Comparison operators	Description
[NOT] BETWEEN x AND y	Allows user to express a range. For example: Searching for numbers BETWEEN 5 and 10. The optional NOT would be used when searching for numbers that are NOT BETWEEN 5 AND 10.
[NOT] IN(x,y,…)	Is similar to the OR logical operator. Can search for records which meet at least one condition contained within the parentheses. For example: Pubid IN (1, 4, 5), only books with a publisher id of 1, 4, or 5 will be returned. The optional NOT keyword instructs Oracle to return books not published by Publisher 1, 4, or 5.

3.3: SELECT statement Clauses

Other Comparison Operators

Other Comparison operators	Description
[NOT] LIKE	Can be used when searching for patterns if you are not certain how something is spelt. For example: title LIKE 'TH%'. Using the optional NOT indicates that records that do contain the specified pattern should not be included in the results.
IS[NOT]NULL	Allows user to search for records which do not have an entry in the specified field. For example: Shipdate IS NULL. If you include the optional NOT, it would find the records that do not have an entry in the field. For example: Shipdate IS NOT NULL.

3.3: SELECT statement Clauses

BETWEEN … AND Operator

The BETWEEN … AND operator finds values in a specified range:

```
SELECT staff_code,staff_name
     FROM  staff_master
     WHERE  staff_dob  BETWEEN '01-Jan-1980'
          AND  '31-Jan-1980';
```

3.3: SELECT statement Clauses

IN Operator

The IN operator matches a value in a specified list.
- The List must be in parentheses.
- The Values must be separated by commas.

```
SELECT dept_code
     FROM  department_master
     WHERE  dept_name IN ( 'Computer Science', 'Mechanics');
```

3.3: SELECT statement Clauses

LIKE Operator

The LIKE operator performs pattern searches.
- The LIKE operator is used with wildcard characters.
- Underscore (_) for exactly one character in the indicated position
- Percent sign (%) to represent any number of characters

```
SELECT book_code,book_name
        FROM  book_master
        WHERE  book_pub_author LIKE '%Kanetkar%' ;
```

3.3: SELECT statement Clauses

||Operator (Concatenation)

The || operator performs concatenation.
- between a string literal and a column name.
- between two column names
- between string literal and a pseudocolumn

```
SELECT 'Hello' ||  student_name
    FROM  student_master

-- only single quotes not double
```

```
SELECT student_code || '  ' ||  student_name
    FROM  student_master
```

```
SELECT 'Today is ' ||  sysdate
    FROM  dual
```

3.3: SELECT statement Clauses

Logical Operators

Logical operators are used to combine conditions.
- Logical operators are NOT, AND, OR.
  - NOT reverses meaning.
  - AND both conditions must be true.
  - OR at least one condition must be true.
- Use of AND operator

```
SELECT staff_code,staff_name,staff_sal
    FROM  staff_master
    WHERE  dept_code = 10
    AND  staff_dob > '01-Jan-1945';
```

3.3: SELECT statement Clauses

Using AND or OR Clause

Use of OR operator:

```
SELECT  book_code
    FROM  book_master
    WHERE  book_pub_author LIKE '%Kanetkar%'
    OR  book_name LIKE '%Pointers%';
```

3.3: SELECT statement Clauses

Using NOT Clause

The NOT operator finds rows that do not satisfy a condition.
- For example: List staff members working in depts other than 10 & 20.

```
SELECT staff_code,staff_name
    FROM  staff_master
    WHERE  dept_code NOT IN ( 10,20 );
```

3.3: SELECT statement Clauses

Treatment of NULL Values

NULL is the absence of data.
Treatment of this scenario requires use of IS NULL operator.

```
SQL>SELECT student_code
            FROM student_master
            WHERE dept_code IS NULL;
```

Operator Precedence

Operator precedence is decided in the following order:

Levels	Operators
1	* (Multiply), / (Division), % (Modulo)
2	+ (Positive), - (Negative), + (Add), (+ Concatenate), - (Subtract), & (Bitwise AND)
3	=, >, <, >=, <=, <>, !=, !>, !< (Comparison operators)
4	NOT
5	OR
6	AND
7	ALL, ANY, BETWEEN, IN, LIKE, OR, SOME
8	= (Assignment)

3.4: SELECT statement Clauses

The DISTINCT clause

The SQL DISTINCT clause is used to eliminate duplicate rows.

- For example: Displays student codes from student_marks tables. the student codes are displayed without duplication

```
SELECT DISTINCT student_code
    FROM student_marks;
```

3.5: SELECT statement Clauses

The ORDER BY clause

The ORDER BY clause presents data in a sorted order.

- It uses an "ascending order" by default.
- You can use the DESC keyword to change the default sort order.
- It can process a maximum of 255 columns.

In an ascending order, the values will be listed in the following sequence:

- Numeric values
- Character values
- NULL values

In a descending order, the sequence is reversed.

3.5: SELECT statement Clauses

Sorting Data

The output of the SELECT statement can be sorted using ORDER BY clause

- ASC : Ascending order, default
- DESC : Descending order

Display student details from student_master table sorted on student_code in descending order.

```
SELECT Student_Code,Student_Name,Dept_Code,
Student_dob
    FROM Student_Master
    ORDER BY Student_Code DESC ;
```

3.6: Tips and Tricks in SELECT Statements

Quick Guidelines

It is necessary to always include a WHERE clause in your SELECT statemen to narrow the number of rows returned.

- If you do not use a WHERE clause, then Oracle will perform a table scan of your table, and return all the rows.
- By returning data you do not need, you cause the SQL engine to perform I/O it doe not need to perform, thus wasting SQL engine resources.

Quick Guidelines

- In addition, the above scenario increases network traffic, which can also lead to reduced performance.
- And if the table is very large, a table scan will lock the table during the time-consuming scan, preventing other users from accessing it, and will hurt concurrency.

In your queries, do not return column data that is not required.

- For example:
  - You should not use SELECT * to return all the columns from a table if all the data from each column is not required.
  - In addition, using SELECT * prevents the use of covered indexes, further potentially decreasing the query performance.

Quick Guidelines

Carefully evaluate whether the SELECT query requires the DISTINCT clause or not.

- The DISTINCT clause should only be used in SELECT statements.
  - This is mandatory if you know that "duplicate" returned rows are a possibility, and that having duplicate rows in the result set would cause problems with your application.
- The DISTINCT clause creates a lot of extra work for SQL Server.
  - The extra load reduces the "physical resources" that other SQL statements have at their disposal.
- Hence, use the DISTINCT clause only if it is necessary.

Quick Guidelines

In a WHERE clause, the various "operators" that are used, directly affect the query performance.

- Given below are the key operators used in the WHERE clause, ordered by their performance. The operators at the top produce faster results, than those listed at the bottom.

  =

  >, >=, <, <=

  LIKE

  <>
- Use "=" as much as possible, and "<>" as least as possible.

Quick Guidelines

If you use LIKE in your WHERE clause, try to use one or more leading character in the clause, if at all possible.

- For example: Use LIKE 'm%' not LIKE '%m'

Certain operators in the WHERE clause prevents the query optimizer from using an Index to perform a search.

- For example: "IS NULL", "<>", "!=", "!>", "!<", "NOT", "NOT EXISTS", "NOT IN", "NOT LIKE", and "LIKE '%500'"

Quick Guidelines

Suppose you have a choice of using the IN or the BETWEEN clauses. In such a case use the BETWEEN clause, as it is much more efficient.

- For example: The first code is much less efficient than the second code given below.

```
SELECT customer_number, customer_name
      FROM customer
      WHERE customer_number in (1000, 1001, 1002, 1003, 1004)
```

```
SELECT customer_number, customer_name
      FROM customer
      WHERE customer_number BETWEEN 1000 and 1004
```

Quick Guidelines

Do not use ORDER BY in your SELECT statements unless you really need to use it.

- Whenever SQL engine has to perform a sorting operation, additional resources have to be used to perform this task.

Summary

In this lesson, you have learnt:

- What is SELECT statement?
- Usage of the following:
  - The WHERE clause
  - The Mathematical, Comparison, and Logical operators
  - The AND or OR clause
  - The NOT clause
  - The DISTINCT clause
  - The ORDER BY clause

Review – Questions

Question 1: The ___ table consists of exactly one column, whose name is "dummy".

Question 2: The LIKE operator comes under the ___ category.

- Option 1: mathematical
- Option 2: comparison
- Option 3: logical

Question 3: The ___ specifies the order in which the operators should be evaluated.

Review – Questions

Question 4: The NOT NULL operator finds rows that do not satisfy a condition.

- True / False

Question 5: More than one column can also be used in the ORDER BY clause.

- True / False