DBMS

NAME: I.MUKIL

BATCH NO: 2022-9615

ENROLLMENT NO: EBEON0323765155

Database Management System (DBMS)

Database Management System(DBMS)is software for storing and retrieving users' data while considering appropriate security measures. It consists of a group of programs that manipulate the database. The DBMS accepts the request for data from an application and instructs the operating system to provide the specific data. In large systems, a DBMS helps users and other third-party software store and retrieve data.

Characteristics of DBMS

Here are the characteristics and properties of a Database Management System:

- Provides security and removes redundancy
- Self-describing nature of a database system
- Insulation between programs and data abstraction
- Support of multiple views of the data
- Sharing of data and multiuser transaction processing
- Database Management Software allows entities and relations among them to form tables.
- It follows the ACID concept (Atomicity, Consistency, Isolation, and Durability).
- DBMS supports a multi-user environment that allows users to access and manipulate data in parallel.

Advantages of DBMS

- DBMS offers a variety of techniques to store & retrieve data

- DBMS serves as an efficient handler to balance the needs of multiple applications using the same data
- Uniform administration procedures for data
- Application programmers are never exposed to details of data representation and storage.
- A DBMS uses various powerful functions to store and retrieve data efficiently.
- Offers Data Integrity and Security
- The DBMS implies integrity constraints to get a high level of protection against prohibited access to data.
- A DBMS schedules concurrent access to the data in such a manner that only one user can access the same data at a time
- Reduced Application Development Time

Disadvantage of DBMS

DBMS may offer plenty of advantages, but it has certain flaws-

- The cost of Hardware and Software of a DBMS is quite high, which increases the budget of your organization.
- Most database management systems are often complex, so training users to use the DBMS is required.
- In some organizations, all data is integrated into a single database that can be damaged because of electric failure or corruption in the storage media.
- Using the same program at a time by multiple users sometimes leads to data loss.
- DBMS can’t perform sophisticated calculations

SQL

SQL is the standard language for dealing with Relational Databases. SQL can be used to insert, search, update, and delete database records. SQL can do lots of other operations, including optimizing and maintenance of databases.

Types of SQL Statements

Here are five types of widely used SQL queries.

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
- Data Control Language (DCL)
- Transaction Control Language (TCL)
- Data Query Language (DQL)

1. Data Definition Language (DDL)

- DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.
- All the command of DDL are auto-committed that means it permanently save all the changes in the database.

Here are some commands that come under DDL:

- CREATE
- ALTER
- DROP
- TRUNCATE

a. **CREATE** It is used to create a new table in the database.

Syntax:

1. CREATE TABLE TABLE_NAME (COLUMN_NAME DATATYPES[,....]);

Example:

1. CREATE TABLE EMPLOYEE(Name VARCHAR2(20), Email VARCHAR2(100), DOB DATE);

b. DROP: It is used to delete both the structure and record stored in the table.

Syntax

1. DROP TABLE table_name;

Example

1. DROP TABLE EMPLOYEE;

c. ALTER: It is used to alter the structure of the database. This change could be either to modify the characteristics of an existing attribute or probably to add a new attribute.

Syntax:

To add a new column in the table

1. ALTER TABLE table_name ADD column_name COLUMN-definition;

To modify existing column in the table:

1. ALTER TABLE table_name MODIFY(column_definitions....);

EXAMPLE

1. ALTER TABLE STU_DETAILS ADD(ADDRESS VARCHAR2(20));
2. ALTER TABLE STU_DETAILS MODIFY (NAME VARCHAR2(20));

d. TRUNCATE: It is used to delete all the rows from the table and free the space containing the table.

Syntax:

1. TRUNCATE TABLE table_name;

Example:

1. TRUNCATE TABLE EMPLOYEE;

2. Data Manipulation Language

- DML commands are used to modify the database. It is responsible for all form of changes in the database.
- The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback.

Here are some commands that come under DML:

- INSERT
- UPDATE
- DELETE

a. INSERT: The INSERT statement is a SQL query. It is used to insert data into the row of a table.

Syntax:

```
1. INSERT INTO TABLE_NAME
2. (col1, col2, col3,.... col N)
3. VALUES (value1, value2, value3, .... valueN);
```

Or

```
1. INSERT INTO TABLE_NAME
2. VALUES (value1, value2, value3, .... valueN);
```

For example:

```
1. INSERT INTO javatpoint (Author, Subject) VALUES ("Sonoo", "DBMS");
```

b. UPDATE: This command is used to update or modify the value of a column in the table.

Syntax:

1. UPDATE table_name SET [column_name1= value1,...column_nameN = valueN] [WHERE CONDITION]

For example:

```
1. UPDATE students
2. SET User_Name = 'sonu'
3. WHERE Student_Id = '3'
```

c. DELETE: It is used to remove one or more row from a table.

Syntax:

```
1. DELETE FROM table_name [WHERE condition];
```

For example:

```
1. DELETE FROM javatpoint
2. WHERE Author="sonu";
```

3. Data Control Language

DCL commands are used to grant and take back authority from any database user.

Here are some commands that come under DCL:

- Grant
- Revoke

a. Grant: It is used to give user access privileges to a database.

Example

```
1. GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER, ANOTHER_US
   ER;
```

b. Revoke: It is used to take back permissions from the user.

Example

```
1. REVOKE SELECT, UPDATE ON MY_TABLE FROM USER1, USER2;
```

4. Transaction Control Language

TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.

These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.

Here are some commands that come under TCL:

- COMMIT
- ROLLBACK
- SAVEPOINT

a. Commit: Commit command is used to save all the transactions to the database.

Syntax:

```
1. COMMIT;
```

Example:

```
1. DELETE FROM CUSTOMERS
2. WHERE AGE = 25;
3. COMMIT;
```

b. Rollback: Rollback command is used to undo transactions that have not already been saved to the database.

Syntax:

```
1. ROLLBACK;
```

Example:

```
1. DELETE FROM CUSTOMERS
2. WHERE AGE = 25;
3. ROLLBACK;
```

c. SAVEPOINT: It is used to roll the transaction back to a certain point without rolling back the entire transaction.

Syntax:

```
1. SAVEPOINT SAVEPOINT_NAME;
```

5. Data Query Language

DQL is used to fetch the data from the database.

It uses only one command:

- SELECT

a. SELECT: This is the same as the projection operation of relational algebra. It is used to select the attribute based on the condition described by WHERE clause.

Syntax:

```
1. SELECT expressions
2. FROM TABLES
3. WHERE conditions;
```

For example:

```
1. SELECT emp_name
2. FROM employee
3. WHERE age > 20;
```

Different Types of SQL Functions

Aggregate Functions

Aggregate functions are used to perform calculations on a set of values and return a single result. Some of the most commonly used aggregate functions in SQL include −

- **COUNT()** - Returns the number of rows in a table or the number of non-NULL values in a column
- **SUM()** - Returns the sum of all non-NULL values in a column
- **AVG()** - Returns the average of all non-NULL values in a column
- **MIN()** - Returns the minimum value in a column
- **MAX()** - Returns the maximum value in a column

- Here's an example of using the COUNT() function to find the number of rows in a table called "orders" −

```
SELECT COUNT(*) FROM orders;
```

- And here's an example of using the SUM() function to find the total cost of all orders in the table −

```
SELECT SUM(total_cost) FROM orders;
```

Scalar Functions

Scalar functions are used to perform calculations on a single value and return a single result. Some examples of scalar functions in SQL include −

- **LENGTH()** - Returns the number of characters in a string
- **UPPER()** - Converts a string to uppercase
- **LOWER()** - Converts a string to lowercase
- **CONCAT()** - Concatenates two or more strings together
- **ROUND()** - Rounds a number to a specified number of decimal places

Here's an example of using the UPPER() function to display the names of all customers in uppercase −

```
SELECT UPPER(customer_name) FROM customers;
```

And here's an example of using the ROUND() function to round the total cost of an order to two decimal places −

```
SELECT ROUND(total_cost, 2) FROM orders;
```

Date and Time Functions

SQL also provides a number of functions for working with date and time values. Some examples of date and time functions in SQL include −

- **NOW()** - Returns the current date and time
- **CURRENT_DATE()** - Returns the current date
- **CURRENT_TIME()** - Returns the current time
- **YEAR()** - Returns the year of a date
- **MONTH()** - Returns the month of a date
- **DAY()** - Returns the day of a date

Here's an example of using the NOW() function to find the current date and time −

```
SELECT NOW();
```

And here's an example of using the MONTH() function to find the month of an order's date −

```
SELECT MONTH(order_date) FROM orders;
```

String Functions

SQL also provides a number of string manipulation function. Some examples of string functions in SQL include −

- **LTRIM()** - Removes the leading whitespace of the string
- **RTRIM()** - Removes the trailing whitespace of the string
- **TRIM()** - Removes both leading and trailing whitespace of the string
- **SUBSTRING()** - Extracts a specific portion of a string
- **REPLACE()** - Replaces all occurrences of a specified string with another string

SQL - ORDER BY Clause

The SQL **ORDER BY** clause is used to sort the data in either ascending or descending order, based on one or more columns. This clause can sort data by a single column or by multiple columns. Sorting by multiple columns can

be helpful when you need to sort data hierarchically, such as sorting by state, city, and then by the person's name.

ORDER BY is used with the SQL SELECT statement and is usually specified after the WHERE, HAVING, and GROUP BY clauses, if present in the query.

Note −

- Some databases sort the query results in an ascending order by default.
- To sort the data in ascending order, we use the keyword **ASC**.
- To sort the data in descending order, we use the keyword **DESC**.

SELECT * FROM CUSTOMERS ORDER BY NAME AS

SELECT * FROM CUSTOMERS ORDER BY NAME DESC;

SQL - Group By

The SQL **GROUP BY** clause is used in conjunction with the SELECT statement to arrange identical data into groups. This GROUP BY clause follows the WHERE clause in a SELECT statement and precedes the ORDER BY or HAVING clause (if they exist). It is often used with aggregate functions like SUM, COUNT, AVG, MAX, or MIN, which allows us to perform calculations on the grouped data.

Eg: SELECT NAME, SUM(SALARY) FROM emp

GROUP BY NAME;

HAVING Clause in GROUP BY Clause

We know that the WHERE clause is used to place conditions on columns but what if we want to place conditions on groups? This is where the HAVING clause comes into use. We can use the HAVING clause to place conditions to decide which group will be part of the final result set. Also, we can not use aggregate functions like SUM(), COUNT(), etc. with the WHERE clause. So we have to use the HAVING clause if we want to use any of these functions in the conditions.

Eg: SELECT NAME, SUM(sal) FROM Emp

GROUP BY name

HAVING SUM(sal)>3000;

SQL | Join (Inner, Left, Right and Full Joins)

SQL Join statement is used to combine data or rows from two or more tables based on a common field between them. Different types of Joins are as follows:

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL JOIN
- NATURAL JOIN

A. INNER JOIN

The INNER JOIN keyword selects all rows from both the tables as long as the condition is satisfied. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be the same.

- **Example Queries(INNER JOIN)**
- This query will show the names and age of students enrolled in different courses.
- SELECT StudentCourse.COURSE_ID, Student.NAME, Student.AGE FROM Student
- INNER JOIN StudentCourse
- ON Student.ROLL_NO = StudentCourse.ROLL_NO;

B. LEFT JOIN

This join returns all the rows of the table on the left side of the join and matches rows for the table on the right side of the join. For the rows for which there is no matching row on the right side, the result-set will contain *null*. LEFT JOIN is also known as LEFT OUTER JOIN.

Example Queries(LEFT JOIN):

SELECT Student.NAME,StudentCourse.COURSE_ID

FROM Student

LEFT JOIN StudentCourse

ON StudentCourse.ROLL_NO = Student.ROLL_NO;

C. RIGHT JOIN

RIGHT JOIN is similar to LEFT JOIN. This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join. For the rows for which there is no matching row on the left side, the result-set will contain *null*. RIGHT JOIN is also known as RIGHT OUTER JOIN.

Example Queries(RIGHT JOIN):

SELECT Student.NAME,StudentCourse.COURSE_ID

FROM Student

RIGHT JOIN StudentCourse

ON StudentCourse.ROLL_NO = Student.ROLL_NO;

D. FULL JOIN

FULL JOIN creates the result-set by combining results of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both tables. For the rows for which there is no matching, the result-set will contain *NULL* values.

Example Queries(FULL JOIN):

SELECT Student.NAME,StudentCourse.COURSE_ID

FROM Student

FULL JOIN StudentCourse

ON StudentCourse.ROLL_NO = Student.ROLL_NO;

E. Natural join (⋈)

Natural join can join tables based on the common columns in the tables being joined. A natural join returns all rows by matching values in common columns having same name and data type of columns and that column should be present in both tables.

Both table must have at list one common column with same column name and same data type.

The two table are joined using Cross join.

DBMS will look for a common column with same name and data type
Tuples having exactly same values in common columns are kept in result.