MODULE IV

DEVELOP APPLICATION USING DATABASE

DATABASE

A database is an organized collection of data, so that it can be easily accessed and managed. Database is collection of related data.

RELATIONAL DATABASE

A relational database is the most commonly used database. It contains several tables. Everything in a relational database is stored in the form of relations. The RDBMS database uses tables to store data. A table is a collection of related data entries and contains rows and columns to store data.

- A row of a table is also called a **record or tuple.**
- A column is a vertical entity in the table which contains all information associated with a specific field in a table is called **field or attribute**
- The **domain** refers to the possible values each attribute can contain.

CREATION OF DATABASES AND TABLES

A database is a structured collection of data that is stored in a computer system. They are used to store and retrieve the data efficiently. Databases can be created using different query languages, and SQL is one such language.

The **CREATE DATABASE** statement is a DDL (Data Definition Language) statement used to create a new database in SQL.

Syntax:

Following is the syntax to create a database in **SQL** −

```
CREATE DATABASE  DatabaseName;
```

SQL

- SQL stands for Structured Query Language. It is used for storing and managing data in relational database management system (RDBMS).
- It is a standard language for Relational Database System. It enables a user to create, read, update and delete relational databases and tables.
- All the RDBMS like MySQL, Oracle, and SQL Server use SQL as their standard database language.

SQL Commands

- **CREATE**
- **ALTER**
- **DROP**
- **INSERT**
- **DELETE**
- **SELECT**

(1) CREATE It is used to create a new table in the database.

Syntax:

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

Example:

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

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

Syntax

DROP TABLE table_name;

Example

DROP TABLE EMPLOYEE;

(3) 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:

(a) To add a new column in the table:

ALTER TABLE table_name ADD column_name COLUMN-definition;

EXAMPLE

ALTER TABLE STU_DETAILS ADD(ADDRESS VARCHAR2(20));

(b) To modify existing column in the table:

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

EXAMPLE

ALTER TABLE STU_DETAILS MODIFY (NAME VARCHAR2(20));

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

Syntax:

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

OR

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

example:

INSERT INTO STUDENT VALUES ("JOHN", "DBMS",23);

(5). DELETE: It is used to remove one or more row from a table.

Syntax:

DELETE FROM table_name [WHERE condition];

For example:

DELETE FROM students WHERE User_Name="ARUN";

(6) SELECT:

SELECT: . It is used to select the attribute based on the condition described by WHERE clause.

Syntax:

SELECT expressions
FROM TABLES
WHERE conditions;

example:

SELECT emp_name
FROM employee
WHERE age > 20;

DATABASE CONNECTIVITY

JDBC

JDBC stands for "**Java Database Connectivity"**. JDBC is a Java API (Application Program Interface) to connect and execute the query with the database. JDBC API uses JDBC drivers to connect with the database

We can use JDBC API to access tabular data stored in any relational database. By the help of JDBC API, we can save, update, delete and fetch data from the database.

The **java.sql** package contains **classes and interfaces** for JDBC API.

A list of popular ***interfaces*** of JDBC API are given below:

- Driver interface
- Connection interface
- Statement interface
- PreparedStatement interface
- ResultSet interface

A list of popular *classes* of JDBC API are given below:

- DriverManager class
- Types class

Java has defined its own API (JDBC API) that uses JDBC drivers (written in Java language).We can use JDBC API to handle database using Java program and can perform the following activities:

1. **Connect to the database**
2. **Execute queries and update statements to the database**
3. **Retrieve the result received from the database.**

API

API (Application programming interface) is a document that contains a description of all the features of a product or software. It represents classes and interfaces that software programs can follow to communicate with each other. An API can be created for applications, libraries, operating systems, etc.

JDBC Driver

JDBC Driver is a software component that enables java application to interact with the database.

Stages in JDBC

There are 5 steps to connect any java application with the database using JDBC. These steps are as follows:

- Register the Driver class
- Create connection
- Create statement
- Execute queries
- Close connection

1) Register the driver class

The **forName()** method of class **Class** is used to register the driver class. This method is used to dynamically load the driver class.

Example to register the Driver class

```
Class.forName("com.mysql.jdbc.Driver");
```

2) Create the connection object

The **getConnection()** method of DriverManager class is used to establish connection with the database.

Example to establish connection with the database

```
Connection con=DriverManager.getConnection(
    "jdbc:mysql://localhost:3306/college","root","root");

    //here college is database name, root is username and password
```

3) Create the Statement object

The createStatement() method of Connection interface is used to create Statement . The object of statement is responsible to execute queries with the database.

Example to create the statement object

```
Statement stmt=con.createStatement();
```

4) Execute the query

The executeQuery() method of Statement interface is used to execute queries To the database. This method returns the object of ResultSet that can be used to get all the records of a table.

Example to execute query

```
ResultSet rs=stmt.executeQuery("select * from student");
     while(rs.next())
    System.out.println(rs.getInt(1)+"  "+rs.getString(2));
        }
```

5) Close the connection object

By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection.

Example to close connection

```
           con.close();
```

INTERFACES IN JDBC

(1)Driver interface

This interface is the Base interface for every driver class i.e. If you want to create a JDBC Driver of your own you need to implement this interface. If you load a Driver class , it will create an instance of itself and register with the driver manager.

(2)Connection interface

It helps to establish a connection with the database.This interface represents the connection with a specific database. SQL statements are executed in the context of a connection. This interface provides methods such as close(), commit(), rollback(), createStatement(), prepareStatement() etc.

(3)PreparedStatement interface

This represents a precompiled SQL statement. An SQL statement is compiled and stored in a prepared statement and you can later execute this multiple times. You can get an object of this interface using the method of the **Connection interface** named **prepareStatement()**. This provides methods such as executeQuery(), executeUpdate(), and execute() to execute the prepared statements.

(4) ResultSet interface

This interface represents the database result set, a table which is generated by executing statements. This interface provides getter and update methods to retrieve and update its contents respectively.

DriverManager class

The DriverManager class is the component of JDBC API and also a member of the *java.sql* package. The DriverManager class acts as an interface between users and drivers. It keeps track of the drivers that are available and handles establishing a connection between a database and the appropriate driver. It contains all the appropriate methods to register and deregister the database driver class and to create a connection between a Java application and the database. Before interacting with a Database, it is a mandatory process to register the driver; otherwise, an exception is thrown.