JDBC (Java Database Connectivity)

Introduction

In Enterprise applications, it is convention to manage the details of the enterprises like Employee details, Customers details, Products Details, Services details, clients details To Manage these details we have to use Storage Areas in Enterprise applications.

There are two types of Storage areas in applications.

1. Temporary Storage Areas.

2. Permanent Storage Areas.

Temporary Storage Areas:

These are memory units; they will store data temporarily.

EX: Buffers, Java Objects

Permanent Storage Areas

These are memory units, they are able to store data permanently.

EX: File System

- DBMS [Database Management Systems]
- Datawarehouses

Q) To manage data permanently we have already File Systems then what is the requirement to use Database Management Systems?

Ans:

1. File Systems are managed by the local Operating Systems, it is platform dependent, it is not suitable for the platform Independent Programming Languages like Java.

DBMS are platform independent; they are very much suitable for the platform independent Programming languages like Java.

2. File Systems are able to store less data.

DBMS are able to store more data.

3. File Systems are able to provide less security.

DBMS are able to provide more security for the data.

4. File Systems are able to increase data redundancy.

DBMS are able to reduce Data Redundancy.

5. File Systems are not having Query Languages Support, so database operations are very much difficult.

DBMS is able to have Query Languages support, it is very simple to perform database operations.

Q) To manage the data, we have already DBMS then what is the requirement to use Datawarehouse's?

Ans:

1. DBMS are able to store less data when compared with Datawarehouses.

2. DBMS is very good while performing the operations like insert, update, delete,... , but, it is not good while performing retrieval operations.

To overcome these problems, we have to use "Datawarehouse's", Datawarehouse's will use fast retrieval mechanisms in the form of "Data Mining Techs".

Q) What are the differences between Database and Database Management Systems?

Ans:

1. Database is a memory Unit, it able to store data.

DBMS is software system, it able to manage the data by storing it in Databases, by retrieving it from Databases.

2. Database is the collection of inter related data as single unit.

DBMS is the collection of inter related data and a set of programs to perform database operations like insert, update, select, delete

DBMS = DB + Set of programs

In general, there are more no of database management systems, but, we will use the following three Database Management systems.

1. Relational Database Management Systems [RDBMS]
2. Object Oriented Database Management Systems [OODBMS]
3. Object Relational Database Management Systems [ORDBMS]

1. Relational Database Management Systems [RDBMS]:

- It able to manage the data in the form of tables.
- To perform operations in RDBMS we have to use a separate Query

Language that is "SQL"[Structered Query Language].

2. Object Oriented Database Management Systems [OODBMS]:

- It able to manage the data in the form of Objects as per Object

  Oriented Features like Objects, Polymorphism, Encapsulation, Abstraction,....

- To perform operations over data in OODBMS we have to use a

  separate Query Language that is "OQL"[Object Query Language].

3. Object Relational Database Management Systems [ORDBMS]:

- It able to manage the data in the from of Tables and Objects.
- To perform Database operations over the data we have to use a

separate Query Language that is "SQL3".

Note: SQL3 is the combination of SQL and OQL.

SQL3 = SQL + OQL

Query Processing System:

--> When we submit an sql query to Database, where at Database, there is a tool to execute sql queries that is Database Engine.

--> At Databases, Database Engines are able to execute sql queries by following the following phases.

1. Query Tokenization

2. Query Parsing

3. Query Optimization

4. Query Execution

1. Query Tokenization:

--> It able to devide the complete sql query into no of pieces, where each and every piece is called as a Token, finally, Query Tokenization phase is able to gete stream of Tokens as an output.

2. Query Parsing:

--> The main intention of Query Parsing Phase is to check whether they provided sql query contains errors or not.

--> Query Parsing phase will take stream of tokens as an input from Query Tokenization Phase, it is trying to construct a tree with the tokens called as "Query Tree".

--> If Query Tree is constructed Successfully then there are no errors in the provided sql query.

--> If Query is not constructed Successfully then there are some

syntax errors in the provided SQL query.

3. Query Optimization:

--> It will take Query Tree as an input, it will apply no of Query Optimization mechanisms, it will generate Optimized Query Tree.

--> The main intention of Query Optimization phase is to generate Optimized Query Tree in order to reduce execution time and in order to reduce memory utilization.

4. Query Execution:

--> It will take Optimized query tree as an input, it will execute that optimized Query Tree by having an interpreter internally and it will perform the required database operation and it will send the result of the Database operation to the respective Client, that is CMD.

JDBC[Java Database Connectivity]:

Def1: JDBC is the step by step process to interact with Database from Java applications inorder to perform database operations from Java applications.

Def2: JDBC is a Technology, not a Programming Language , it will provide very good environment to connect with database from Java applications inorder to perform database operations from Java applications.

Def3: JDBC is an API[Collection of Libraries, Collection of classes and interfaces], it can be used to interact with database from Java applications inorder to perfrom database operations from Java applications.

Def4: JDBC is an abstraction provided by SUN Microsystems and implemented by all the database vendors to connect with their respective databases from java applications inorder to perform database operations from Java applications.

In general, in Jdbc applications, we will provide Database logic in Java application, we will send the provided database logic from Java application to database, where at database, Database logic will be executed , database will send the result of the database logic execution to java application, At Java application we have to get results from database and we have to display that results in java applications.

In Jdbc applications, when we submit database logic from Java application database, at database, database Engine has to execute Database logic,p but, database Engine will not execute the provided database logic, because, database Engine is unable to understand Java representations, it able to understand Query Languages representations.

In the above context, to execute Database logic, we need a translator inbetween Java applications and Database, it has to convert Java representations to database representations and from Database representations to Java representations.

In the above context, what ever the translator we have taken to convert Java representations to Dabase representations and database representations to Java representations that convertor is called as "Driver".

"Driver" is an interface existed inbetween Java application and Database, it able to map JAVA API calls to DB API calls and from DB API calls to Java API calls.

To provide Driver, SUN Microsystems has provided an interface in the from of java.sql.Driver and it has given an option to all the Database vendors to prepare their own implementation classes for the Driver interface.

As per the above convention, all the Database vendors have provided their own implementation classes for the java.sql.Driver interface in their respective Databases.

---

1. Storage Areas in Applications

There are two types of storage areas in applications:

1.1 Temporary Storage Areas

- Stores data temporarily.
- Examples: Buffers, Java Objects.

1.2 Permanent Storage Areas

- Stores data permanently.
- Examples:
    - File System
    - DBMS (Database Management Systems)
    - Data Warehouses

---

2. Need for DBMS over File Systems

Feature	File System	MySQL DBMS
Platform Independence	No	Yes

Data Storage	Less	More
Security	Low	High
Query Language	No	Yes (SQL)
Data Redundancy	High	Low

---

3. Database vs. DBMS

Feature	Database	MySQL DBMS
Definition	Memory unit that stores data	Software system that manages data
Components	Collection of related data	Data + Programs to manage data

---

4. Types of Database Management Systems

1. **Relational Database Management Systems (RDBMS)**
    - Stores data in tables.
    - Uses SQL for operations.
    - Example: MySQL.
2. **Object-Oriented Database Management Systems (OODBMS)**
    - Stores data as objects following OOP principles.
    - Uses OQL (Object Query Language).
3. **Object-Relational Database Management Systems (ORDBMS)**
    - Hybrid of RDBMS and OODBMS.
    - Uses SQL3 (Combination of SQL and OQL).

---

5. Query Processing System

When an SQL query is submitted, the database engine processes it through the following phases:

1. **Query Tokenization** - Breaks the query into tokens.
2. **Query Parsing** - Checks for syntax errors.
3. **Query Optimization** - Optimizes execution time and memory.
4. **Query Execution** - Executes the query and returns results.

---

6. JDBC Overview

Definitions

- **JDBC is a step-by-step process** to interact with MySQL databases from Java applications.
- **JDBC is an API (Application Programming Interface)** that consists of libraries, classes, and interfaces.
- **JDBC is an abstraction provided by Sun Microsystems**, implemented by database vendors.

JDBC Workflow

1. **Java application sends a database request**.
2. **JDBC converts Java calls into database queries**.
3. **MySQL processes the request and returns results**.
4. **JDBC converts database responses into Java objects**.

---

7. Types of JDBC Drivers

Type	Name	Description
Type-1	JDBC-ODBC Bridge Driver	Uses ODBC Driver (Platform-dependent, slow)
Type-2	Native API Driver	Uses database vendor-specific native libraries
Type-3	Network Protocol Driver	Uses middleware to connect
Type-4	Thin Driver	Directly connects Java to MySQL

---

8. Steps to Create a JDBC Application

1. **Download and Install MySQL Database Software.**
2. **Download and Install IDE** (e.g., Eclipse, IntelliJ IDEA).
3. **Create Java Project and Add MySQL JDBC Library.**

JDBC Programming Steps

1. **Load and Register the Driver**

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

2. **Establish Connection**

```
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/mydataba
```

3. **Create Statement**

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

4. **Execute SQL Query**

```
ResultSet rs = stmt.executeQuery("SELECT * FROM employees");
```

5. **Process Results**

```
while(rs.next()) {
    System.out.println(rs.getInt("id") + " " + rs.getString("name"));
}
```

6. **Close the Connection**

```
rs.close();
stmt.close();
con.close();
```

---

9. Example JDBC Application for MySQL

```
import java.sql.*;

public class JdbcExample {
    public static void main(String[] args) {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM employees");
            while (rs.next()) {
                System.out.println(rs.getInt(1) + " " + rs.getString(2));
            }
            rs.close(); stmt.close(); con.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Q) What is the difference between Statement, Prepared Statement and Callable Statement?

In Jdbc applications, when we want to execute all the sql queries individually there we will use java.sql.Statement.

In Jdbc applications, when we have a requirement to execute the same sql query in the next sequence there to improve Jdbc application performance we have to use java.sql.PreparedStatement.

In Jdbc applications, when we have a requirement to access stored procedures and functions which are defined at databases from Java applications there we have to use java.sql.CallableStatement.

In basic JDBC applications we will use java.sql.Statement, to create Statement we will use the following method from java.sql.Connection.

public Statement createStatement()throws SQLException

EX: Statement st = con.createStatement();

Q) How does createStatement() create an object for java.sql.Statement?

Concept

- java.sql.Statement is an **interface** provided by **Sun Microsystems (now Oracle)** as part of the JDBC abstraction.
- JDBC itself is an **abstraction (collection of interfaces)** whose implementations are provided by database vendors.
- The actual **implementation classes** of Statement are provided by database vendors (e.g., Oracle, MySQL).

How createStatement() works:

- createStatement() **does not create an object** for the Statement interface directly.
- Instead, it **creates an object** for the **implementation class** of Statement, provided by the database vendor.

Methods in `java.sql.Statement` for executing SQL queries:

Method	Purpose
`executeQuery(String query)`	Used for **SELECT queries** (fetch data).
`executeUpdate(String query)`	Used for **non-SELECT queries** (INSERT, UPDATE, DELETE, etc.).
`execute(String query)`	Used for **both SELECT and non-SELECT queries**.

Difference between executeQuery(), executeUpdate(), and execute()

Types of SQL Queries in JDBC

1. **SELECT Queries** → Fetch data from the database.
2. **Non-SELECT Queries** → Perform modifications (INSERT, UPDATE, DELETE, etc.).

Method	Query Type	Return Type	When to Use
`executeQuery()`	`SELECT`	`ResultSet`	When fetching data from the database.
`executeUpdate()`	`INSERT`, `UPDATE`, `DELETE`, `ALTER`, `DROP`, etc.	`int` (row count)	When modifying data in the database.
`execute()`	Both `SELECT` and `Non-SELECT`	`boolean` (`true` for SELECT, `false` for others)	When the query type is **unknown in advance**.