Static SQL queries: Queries which doesn’t have condition

Or

Condition values are hard Coded.

Dynamic SQL queries: There must be a condition plus one or more condition will be evaluated at the runtime. ( ? )

**Static SQL Queries**

* SQL queries
* “Without conditions” OR
* “with Hard Coded condition values” are called as “Static SQL Queries”

Example:

1. Select \* from tablename;
2. Create database DB\_NAME;
3. Select \* from ABC where X = 1;
4. Insert into ABC values (1, ‘Aatish’);

Note: ABC = Table Name

**Dynamic SQL Queries**

* SQL Queries which
* Must have conditions &
* One/More condition values get decided at runtime are known as “Dynamic SQL Queries”.

Examples:

1. Select \* from ABC where X = ? and Y = ?;
2. Select \* from ABC where X = 1 and Y = ?;
3. Insert into ABC values (?,”Praveen’);

Note:

1. ABC = Table Name
2. Dynamic SQL Query Must Contain One/More Question Marks.

**JDBC Statements**

* **JDBC**  Statements send SQL queries to RDBMS and retrieve the data from RDBMS application.
* **There are different typed of JDBC Statements**

1. java.sql.Statement
2. java.sql.PreparedStatement
3. java.sql.CallableStatement

* Once we create JDBC Statement Object (any of the above type) , then we MUST invoke any one of he below method to issue SQL queries to DB

1. Int executeUpdate() throws SQLExeception

* This method is used to execute “Other than SELECT “ SQL queries.
* This method return “NO. of Rows Affected Count” in the form of Integer.

1. ResultSet executeQuery() throws SQLException

* This method is used to execute “ONLY SELECT” SQL Queries
* This method returns “DB Results” in the form of “ResultSet” Object

1. Boolean execute() throws SQLException

* This method is used to execute “ANY SQL Query including SELECT”
* This method:
* Returns “true”,, if result is of type “DB Results”
* Returns “false” , if result is of type “integer Count”
* If we use this method then we must make use
* “getResultSet()”

OR

* “getUpdateCount()”

Methods to get the actual results

**Java.sql.Statement**

> Its an interface & an Object of Statement is used to execute “Static SQL Queries”

> Statement Object can be created by invoking “createStatement()” method on “Connection” Objects

Syntax:

Statement Connection.createStatement() throws SQLException

Statement stmt = con.createStatement();

Where “con” is the Object reference of “java.sql.Connection” Object

Q:Write a Java Program which deletes Reg. No. 6 data from “students\_info” table;

**Java.sql.PreparedStatement**

* It’s an interface & an object of PreparedStatement is used to execute “Dynamic SQL Queries”
* PreparedStatement Object can be created by invoking “prepareStatement()” method on “Connection” Object.

Syntax:

PreparedStatement Connection.prepareStatement(String query) throws SQLException

Example:

String query = “delete \* from students\_info where regno = ?”;

PreparedStatement pstmt = con.prepareStatement(query);

Where “con” is the object reference of “java.sql.Connection” Object

* PreparedStatements MUST be used with query parameters (?) & these query parameters need to be set using proper setXXX () method before executing the dynamic SQL query

Syntax:

Void setXXX(Position of ? as Int Value, Corresponding Runtime Value) throws SQLException where XXX = Java Data Type corresponding to DB Column Data Type.

* PreparedStatements are also Known as “precompiled Statements” & they helps us to achieve “high performance”

**Stored Procedures**

* Stored Procedures are group of SQL queries that perform a particular task ( functionality wise they are similar to Java Methods )
* As its name implies, they are stored at RDBMS Application / DB side
* Stored Procedures helps to achieve “Reusability”
* Query to get the list of Procedures available in MySql Database is:

**SHOW PROCEDURE STATUS WHERE DB = DATABASE();**

**Stored Procedure 1:-**

1. **delimiter &**
2. **CREATE PROCEDURE getAllStudents()**

**BEGIN**

**SELECT \* FROM students\_info;**

1. **END&**
2. **delimiter ;**

**Stored Procedure 2:-**

1. **delimiter $**
2. **CREATE PROCEDURE getStudentInfo(IN in\_regno INT)**

**BEGIN**

**SELECT \* FROM students\_info**

**WHERE REGNO = in\_regno;**

**END$**

1. **delimiter ;**
2. **call getStudentInfo (1);**

**java.sql.CallableStatement:**

* Its an interface & an Object of CallableStatement is used to execute “Stored Procedures”
* CallableStatement Object can be created by invoking “prepareCall()” method on “Connection” Object

Syntax:

CallableStatement Connection.prepareCall(String query) throws SQLException

Example:

String query = “call stroredProcedureNM()”;

CallableStatement ctmt = con.preparedCall(query);

Where “con” is an Object reference of “java.sql.Connection Object”

* While invoking the Procedure, which takes input arguments

-Either we can “hardcode the condition values”

Or

-These condition values may get decided at Runtime

> If condition values get decided at Runtime then we should have Question Mark(?) while constructing SQL Query

> Stored Procedures, by nature, reduces the number of DB calls

> Hence CallableStatements, which helps us to execute Stored Procedures, increases the “Performance of the Application”