Lecture 10: Prepared Statement Solution

What is PreparedStatement?

PreparedStatement is a precompiled SQL statement that:

- Prevents SQL injection attacks
- Improves performance through query plan reuse
- Provides cleaner, more readable code
- Handles data type conversions automatically

Complete PreparedStatement Example:

```
import java.sql.*;

public class PreparedStatementDemo {
    public static void main(String[] args) {
        String url = "jdbc:postgresql://localhost:5432/demo";
        String username = "postgres";
        String password = "0000";

        Connection conn = null;
        PreparedStatement pstmt = null;

        try {
            Class.forName("org.postgresql.Driver");
            conn = DriverManager.getConnection(url, username, password);

            // SQL with placeholders (?)
            String sql = "INSERT INTO student VALUES (?, ?, ?)";

            // Create PreparedStatement
            pstmt = conn.prepareStatement(sql);

            // Set parameters (1-indexed)
            pstmt.setInt(1, 10);          // sid
            pstmt.setString(2, "Ananya");  // sname
            pstmt.setInt(3, 90);          // marks

            // Execute the statement
            int rowsAffected = pstmt.executeUpdate();
            System.out.println("Rows inserted: " + rowsAffected);

        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
            e.printStackTrace();
        } finally {
```

```
            try {
                if (pstmt != null) pstmt.close();
                if (conn != null) conn.close();
            } catch (SQLException e) {
                System.out.println("Error closing resources: " + e.getMessage());
            }
        }
    }
}
```

Understanding Placeholders (?):

- **?** represents a parameter placeholder
- Parameters are 1-indexed (first ? is index 1)
- Use appropriate setter methods based on data type:
    - `setInt(index, value)` for integers
    - `setString(index, value)` for strings
    - `setDouble(index, value)` for doubles
    - `setDate(index, value)` for dates
    - `setBoolean(index, value)` for booleans

PreparedStatement vs Statement Comparison:

Aspect	Statement	PreparedStatement
SQL Injection	Vulnerable	Safe
Performance	Slower (compiled each time)	Faster (precompiled)
Code Readability	Complex concatenation	Clean and simple
Parameter Handling	Manual string manipulation	Automatic type handling
Reusability	Limited	High (can reuse with different parameters)

Complete CRUD with PreparedStatement:

```
public class PreparedStatementCRUD {
    private static final String URL = "jdbc:postgresql://localhost:5432/demo";
    private static final String USERNAME = "postgres";
    private static final String PASSWORD = "0000";
```

```
    // CREATE
    public static void insertStudent(int sid, String sname, int marks) {
        String sql = "INSERT INTO student VALUES (?, ?, ?)";
        try (Connection conn = DriverManager.getConnection(URL, USERNAME,
PASSWORD);
             PreparedStatement pstmt = conn.prepareStatement(sql)) {

            pstmt.setInt(1, sid);
            pstmt.setString(2, sname);
            pstmt.setInt(3, marks);

            int rows = pstmt.executeUpdate();
            System.out.println("Inserted " + rows + " record(s)");

        } catch (SQLException e) {
            System.out.println("Insert error: " + e.getMessage());
        }
    }

    // READ
    public static void getStudent(int sid) {
        String sql = "SELECT * FROM student WHERE sid = ?";
        try (Connection conn = DriverManager.getConnection(URL, USERNAME,
PASSWORD);
             PreparedStatement pstmt = conn.prepareStatement(sql)) {

            pstmt.setInt(1, sid);
            ResultSet rs = pstmt.executeQuery();

            if (rs.next()) {
                System.out.println("ID: " + rs.getInt("sid") +
                                   ", Name: " + rs.getString("sname") +
                                   ", Marks: " + rs.getInt("marks"));
            } else {
                System.out.println("Student not found");
            }

        } catch (SQLException e) {
            System.out.println("Select error: " + e.getMessage());
        }
    }

    // UPDATE
    public static void updateStudent(int sid, String newName, int newMarks) {
        String sql = "UPDATE student SET sname = ?, marks = ? WHERE sid = ?";
        try (Connection conn = DriverManager.getConnection(URL, USERNAME,
PASSWORD);
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
```

```
            pstmt.setString(1, newName);
            pstmt.setInt(2, newMarks);
            pstmt.setInt(3, sid);

            int rows = pstmt.executeUpdate();
            System.out.println("Updated " + rows + " record(s)");

        } catch (SQLException e) {
            System.out.println("Update error: " + e.getMessage());
        }
    }

    // DELETE
    public static void deleteStudent(int sid) {
        String sql = "DELETE FROM student WHERE sid = ?";
        try (Connection conn = DriverManager.getConnection(URL, USERNAME,
PASSWORD);
             PreparedStatement pstmt = conn.prepareStatement(sql)) {

            pstmt.setInt(1, sid);

            int rows = pstmt.executeUpdate();
            System.out.println("Deleted " + rows + " record(s)");

        } catch (SQLException e) {
            System.out.println("Delete error: " + e.getMessage());
        }
    }
}
```