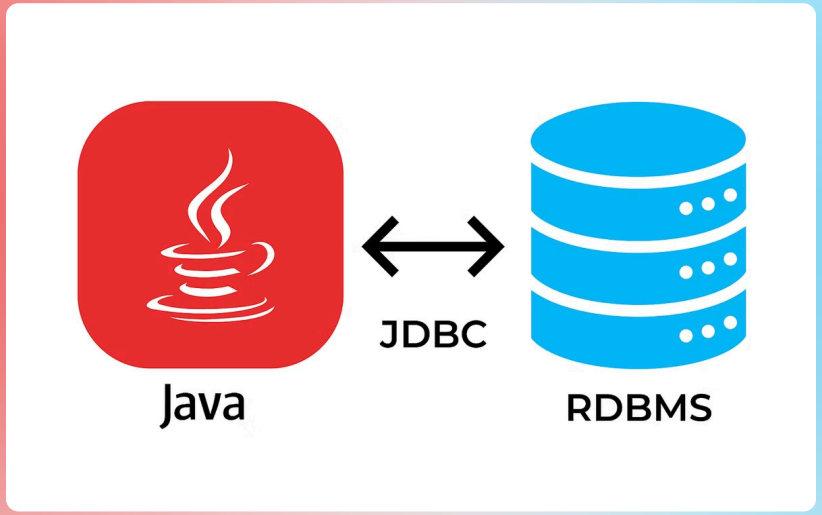

JDBC: Java Database Connectivity

JDBC connects Java applications to relational databases. It enables data-driven Java development. This presentation covers key concepts, drivers, API, and best practices for JDBC usage.

Types of JDBC Drivers

Type 1: JDBC-ODBC Bridge

Platform-dependent, uses ODBC driver, deprecated for production.

Type 2: Native-API Driver

Uses vendor libraries, better performance, platform-dependent.

Type 3: Network Protocol Driver

Middleware server based, platform-independent, good for web apps.

Type 4: Thin Driver

Direct database protocol, platform-independent, best performance.

Core JDBC API Overview

DriverManager

Manages all registered JDBC drivers.

Connection

Represents a database session.

Statements

Execute SQL queries and updates.

ResultSet

Holds query result data.

SQLException

Handles database errors.

Steps to Connect to a Database

- Load the driver class using Class.forName.
- Establish connection with DriverManager.getConnection using URL, user, and password.
- Create a Statement object to execute queries.
- Execute the query and get ResultSet.
- Process ResultSet data with looping.
- Close connection using try-with-resources for safety.

Statement Interfaces Explained

Statement

Executes static SQL; prone to SQL injection; basic methods include executeQuery.

PreparedStatement

Precompiled SQL with placeholders; prevents injection; efficient for repeated queries.

CallableStatement

Used to invoke stored procedures with input and output parameters.

Batch Processing & Transaction Management

Batch Processing

Combine multiple SQL commands using addBatch and executeBatch for efficiency.

Transaction Management

Control commit and rollback with setAutoCommit, ensuring ACID properties for data integrity.

Example: Secure fund transfers between accounts.

Metadata and CallableStatement Usage

DatabaseMetaData

Fetches details about database structure like tables and driver info.

ResultSetMetaData

Provides details about query results such as column names and types.

CallableStatement

Prepare and execute stored procedures; set parameters and retrieve outputs.

Best Practices and Conclusion

Use connection pooling to improve resource management.

Always handle exceptions with try-catch and close resources properly.

Choose the right JDBC driver for your platform needs.

Use PreparedStatement to safeguard against SQL injection.

Batch processing boosts performance for bulk SQL operations.

Manage transactions carefully to maintain data integrity.

JDBC bridges Java and databases, vital for enterprise apps.

Thank You

Feel free to reach out with any questions.