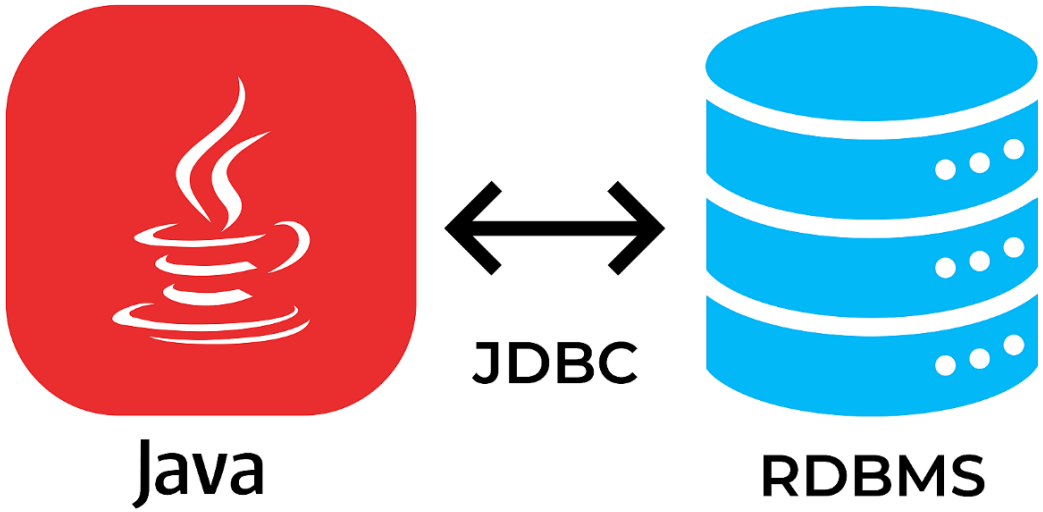
Introduction to JDBC

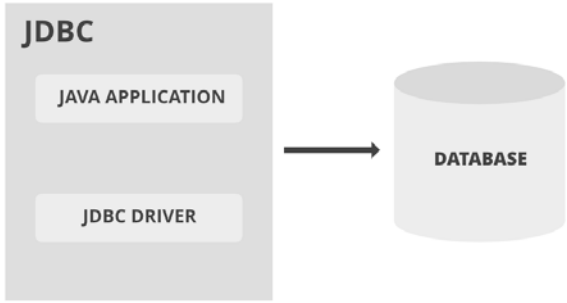


Agenda

- What is JDBC?
- JDBC Driver
- Connection
- Statement
- ResultSet
- Transaction

JDBC

- What is JDBC?
  - JDBC is a Java API for executing SQL statement
  - Easy to send SQL statement to virtually any relational database
  - What does JDBC do?
    - Establish a connection with a database
    - Send SQL statement
    - Process the results



JDBC

- Basic Example

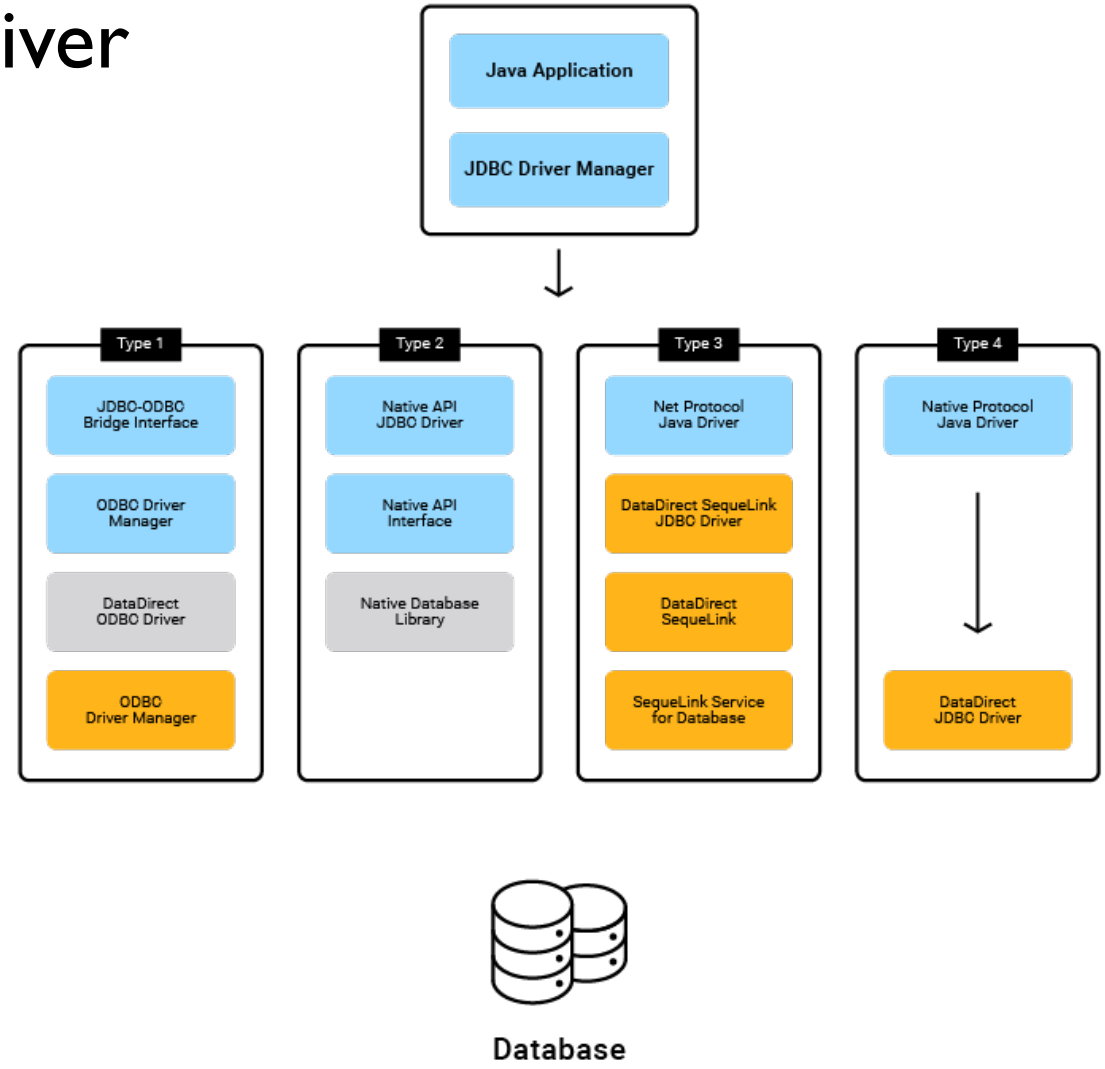
```
import java.sql.*;
public class BasicJDBC {
        public static void main(String[] args) {
                try {
                        Class.forName("com.mysql.jdbc.Driver");
                        Connection con =
DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/mydb",“user",“password");
                        Statement stm = con.createStatement();
                        ResultSet rs = stm.executeQuery("select
journalname,amount from tjournal");
                        while(rs.next()) {
                                String name = rs.getString("journalname");
                                java.math.BigDecimal amt =
rs.getBigDecimal("amount");
                                System.out.println(name+"="+amt);
                        }
                } catch(Exception ex) {
                        ex.printStackTrace();
                }
        }
}
```

JDBC

- JDBC Driver
  - JDBC-ODBC bridge driver
    - ODBC driver
  - Native API partly Java Driver
    - Native code to java
  - JDBC-Net pure Java driver
    - Network protocol
  - Native protocol pure Java driver
    - Thin drivers

JDBC

- JDBC Driver



JDBC

- Connection
  - A Connection object represents a connection with a database
  - Opening a Connection
    - DriverManager.getConnection
  - URLs in General Use
    - jdbc:<subprotocol>:<subname>

```
String url = "jdbc:mysql://127.0.0.1:3306/mydb“;

Connection con = DriverManager.getConnection(url,“user",“password");
```

JDBC

- DriverManager
  - The DriverManager class is the management layer of JDBC working between the user and the drivers
  - Loading the driver manager
    - Class.forName("com.mysql.jdbc.Driver");
  - Establishing a Connection

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

String url = "jdbc:mysql://127.0.0.1:3306/mydb“;

Connection con = DriverManager.getConnection(url,“user",“password");
```

JDBC

- DBMS Driver & URL

DB	Driver	URL
MySQL	com.mysql.jdbc.Driver com.mysql.cj.jdbc.Driver	jdbc:mysql://host:port/dbname ex. jdbc:mysql://127.0.0.1:3306/mydb
SQLServer	com.microsoft.sqlserver.jdbc.SQLServerDriver	jdbc:sqlserver://host:port;DatabaseName=dbname ex. jdbc:sqlserver://127.0.0.1:1433;DatabaseName=mydb
Informix	com.informix.jdbc.IfxDriver	jdbc:informix-sqli://host:port/dbname:INFORMIXSERVER=online ex. jdbc:informix-sqli://127.0.0.1:9088/mydb:INFORMIXSERVER=ol_svr_custom
Oracle	oracle.jdbc.driver.OracleDriver	jdbc:oracle:thin:@host:port:dbname ex. jdbc:oracle:thin:@127.0.0.1:1521:mydb
DB2	com.ibm.db2.jcc.DB2Driver	jdbc:db2://host:port/dbname ex. jdbc:db2://127.0.0.1:50000/mydb
PostgreSQL	org.postgresql.Driver	jdbc:postgresql://host:port/dbname ex. jdbc:postgresql://localhost:5432/mydb

JDBC

- Statement
  - A Statement object is used to send SQL statement to a database
  - 3 kind of statement : Statement, PreparedStatement and CallableStatement
  - Creating Statement Objects

```
String url = "jdbc:mysql://127.0.0.1:3306/mydb“;
Connection con = DriverManager.getConnection(url,“user",“password");
Statement stm = con.createStatement();
```

  - Executing Statement Objects

```
ResultSet rs = stm.executeQuery("select journalname,amount from tjournal");
```

JDBC

- ResultSet
  - A ResultSet contains all of the rows which satisfied the conditions in an SQL statement and it provides access to the data in those rows

```
ResultSet rs = stm.executeQuery("select journalname,amount from tjournal");
while(rs.next()) {
        String name = rs.getString("journalname");
        java.math.BigDecimal amt = rs.getBigDecimal("amount");
        System.out.println(name+"="+amt);
}
```

JDBC

- ResultSet

Method	Description
public boolean next():	is used to move the cursor to the one row next from the current position.
public boolean previous():	is used to move the cursor to the one row previous from the current position.
public boolean first():	is used to move the cursor to the first row in result set object.
public boolean last():	is used to move the cursor to the last row in result set object.
public int getInt(int columnIndex): public int getInt(String columnName):	is used to return the data of specified column index / column name of the current row as int.
public String getString(int columnIndex): public String getString(String columnName):	is used to return the data of specified column index / column name of the current row as String.
public BigDecimal getBigDecimal(int columnIndex): public BigDecimal getBigDecimal(String columnName):	is used to return the data of specified column index / column name of the current row as java.math.BigDecimal.
public Date getDate(int columnIndex): public Date getDate(String columnName):	is used to return the data of specified column index / column name of the current row as java.sql.Date.
public Time getTime(int columnIndex): public Time getTime(String columnName):	is used to return the data of specified column index / column name of the current row as java.sql.Time.
public Timestamp getTimestamp(int columnIndex): public Timestamp getTimestamp(String columnName):	is used to return the data of specified column index / column name of the current row as java.sql.Timestamp.

JDBC

- ResultSetMetaData
  - An object that can be used to get information about the types and properties of the columns in a ResultSet object

Method	Description
public int getColumnCount() throws SQLException	it returns the total number of columns in the ResultSet object.
public String getColumnName(int index) throws SQLException	it returns the column name of the specified column index.
public String getColumnTypeName(int index) throws SQLException	it returns the column type name for the specified index.
public String getTableName(int index) throws SQLException	it returns the table name for the specified column index.

JDBC

- ResultSetMetaData

```
import java.sql.*;
public class BasicResultSetMetaData {
        public static void main(String[] args) {
                try {
                        Class.forName("com.mysql.jdbc.Driver");
                        Connection con =
DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/mydb",“user",“password");
                        Statement stm = con.createStatement();
                        ResultSet rs = stm.executeQuery("select * from tjournal");
                        java.sql.ResultSetMetaData met = rs.getMetaData();
                        int colcount = met.getColumnCount();
                        while(rs.next()) {
                                for(int i=1;i<=colcount;i++) {
                                        String name =
met.getColumnName(i);

        System.out.println(name+"="+rs.getString(name));
                                }
                        }
                } catch(Exception ex) {
                        ex.printStackTrace();
                }
        }
}
```

JDBC

- PreparedStatement
  - PreparedStatement interface inherits from Statement
  - Contain an SQL statement that has already been compiled
  - May have one or more parameters
  - A question mark (“?”) as a placeholder for each parameter

JDBC

- PreparedStatement

```
import java.sql.*;

public class BasicPrepared {
		public static void main(String[] args) {
				try {
						Class.forName("com.mysql.jdbc.Driver");
						Connection con =
DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/mydb",“user",“password");
						String sql = "insert into
tjournal(journalid,journalname,amount) values(?,?,?)";
						PreparedStatement ps = con.prepareStatement(sql);
						ps.setString(1,"3001");
						ps.setString(2,"Fee");
						ps.setBigDecimal(3,new java.math.BigDecimal(100));
						int rows = ps.executeUpdate();
						System.out.println("effected "+rows+" rows");
				} catch(Exception ex) {
						ex.printStackTrace();
				}
		}
}
```

JDBC

- CallableStatement
  - A CallableStatement object provides a way to call stored procedures in a standard way for all DBMSs.
  - Stored Procedures are group of statements that compile in the database for some task

JDBC

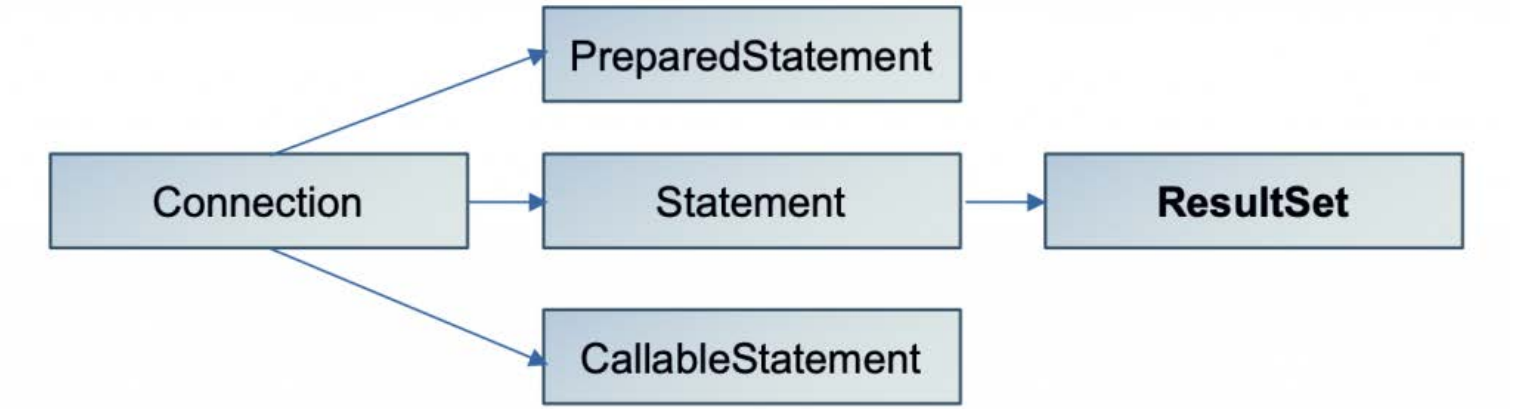
- CallableStatement

```
import java.sql.*;
public class BasicCallable {
        public static void main(String[] args) {
                try {
                        Class.forName("com.mysql.jdbc.Driver");
                        Connection con =
DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/mydb",“user",“password");
                        String sql = "{call GetJournal(?)}";
                        CallableStatement cs = con.prepareCall(sql);
                        cs.setString(1,"1");
                        ResultSet rs = cs.executeQuery();
                        while(rs.next()) {
                                String id = rs.getString(1);
                                String name = rs.getString(2);
                                java.math.BigDecimal amt =
rs.getBigDecimal(3);

        System.out.println("id="+id+",name="+name+",amt="+amt);
                        }
                } catch(Exception ex) {
                        ex.printStackTrace();
                }
        }
}
```

JDBC

- JDBC Hierarchy



JDBC

- Transaction
  - Transaction is a logical unit of work
  - Transaction support allows applications to ensure the following :
    - All the steps to complete a logical unit of work are followed
    - When one of the steps to the unit of work files fails, all the work done as part of that logical unit of work can be undone and the database can return to its previous state before the transaction began

JDBC

- Transaction
  - The default behavior of a Connection is auto commit
    - public abstract void setAutoCommit(boolean) throws SQLException;
    - public abstract void commit() throws SQLException;
    - public abstract void rollback() throws SQLException;

JDBC

- Transaction

```
connect to database

try {
        set AutoCommit(false)

        execute query1
        execute query2
        ...
        execute queryN

        commit
} catch(Exception ex) {
        rollback
}

close connection
```

JDBC

- Transaction

```
import java.sql.*;
public class BasicTransaction {
        public static void main(String[] args) {
                try {
                        Class.forName("com.mysql.jdbc.Driver");
                        Connection con =
DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/mydb",“user",“password");
                        try {
                        con.setAutoCommit(false);
                        String sql = "update tjournal set amount = ? where journalid
= ?";

                        PreparedStatement ps = con.prepareStatement(sql);
                        ps.setBigDecimal(1,new java.math.BigDecimal(150));
                        ps.setString(2,"3001");
                        int rows = ps.executeUpdate();
                        System.out.println("effected "+rows+" rows");
                        con.commit();
                        } catch(SQLException ex) {
                                ex.printStackTrace();
                                con.rollback();
                        }
                } catch(Exception ex) { ex.printStackTrace(); }
        }
}
```

Reference

- https://en.wikipedia.org/wiki/JDBC_driver
- https://www.progress.com/faqs/datadirect-jdbc-faqs/what-are-the-types-of-jdbc-drivers
- https://www.geeksforgeeks.org/jdbc-type-3-driver/
- https://www.tutorialspoint.com/jdbc/jdbc-driver-types.htm
- https://www.javatpoint.com/jdbc-driver
- https://www.tutorialspoint.com/jdbc/jdbc-result-sets.htm
- https://www.baeldung.com/java-transactions

Q & A