Databases and JDBC

by Vlad Costel Ungureanu
for “Learn Stuff”

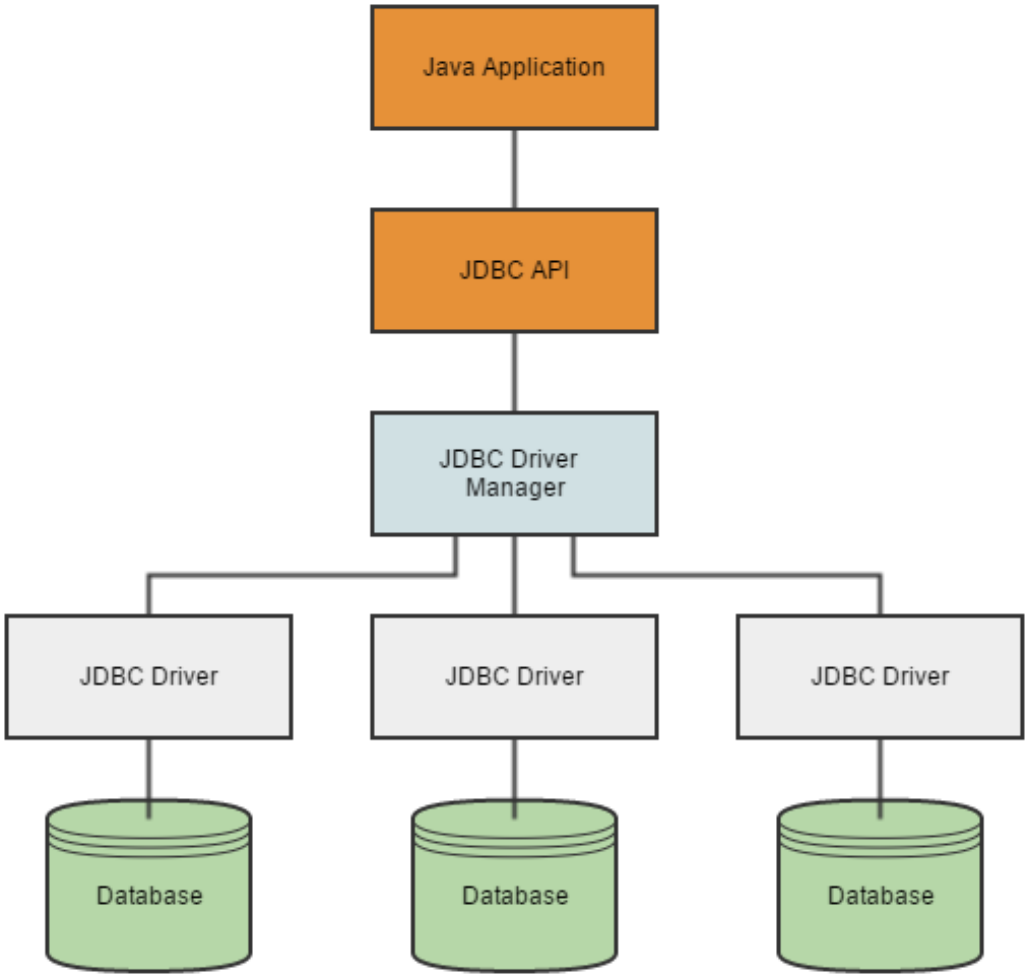
Working with Databases

- ✓ Create database using SQL scripts
- ✓ Connect to the database server using a driver
- ✓ Communicate with the database
  - ✓ Execute SQL statements and queries
  - ✓ Process results

LEARN stuff IO

JDBC

- ✓ JDBC (Java Database Connectivity) allows access to relational databases
- ✓ It is independent of the database type
- ✓ It uses a driver to ensure communication between client and database server
- ✓ In java there are two dedicated packages:
  - ✓ java.sql – JDBC core
  - ✓ javax.sql – J2EE extension over JDBC core

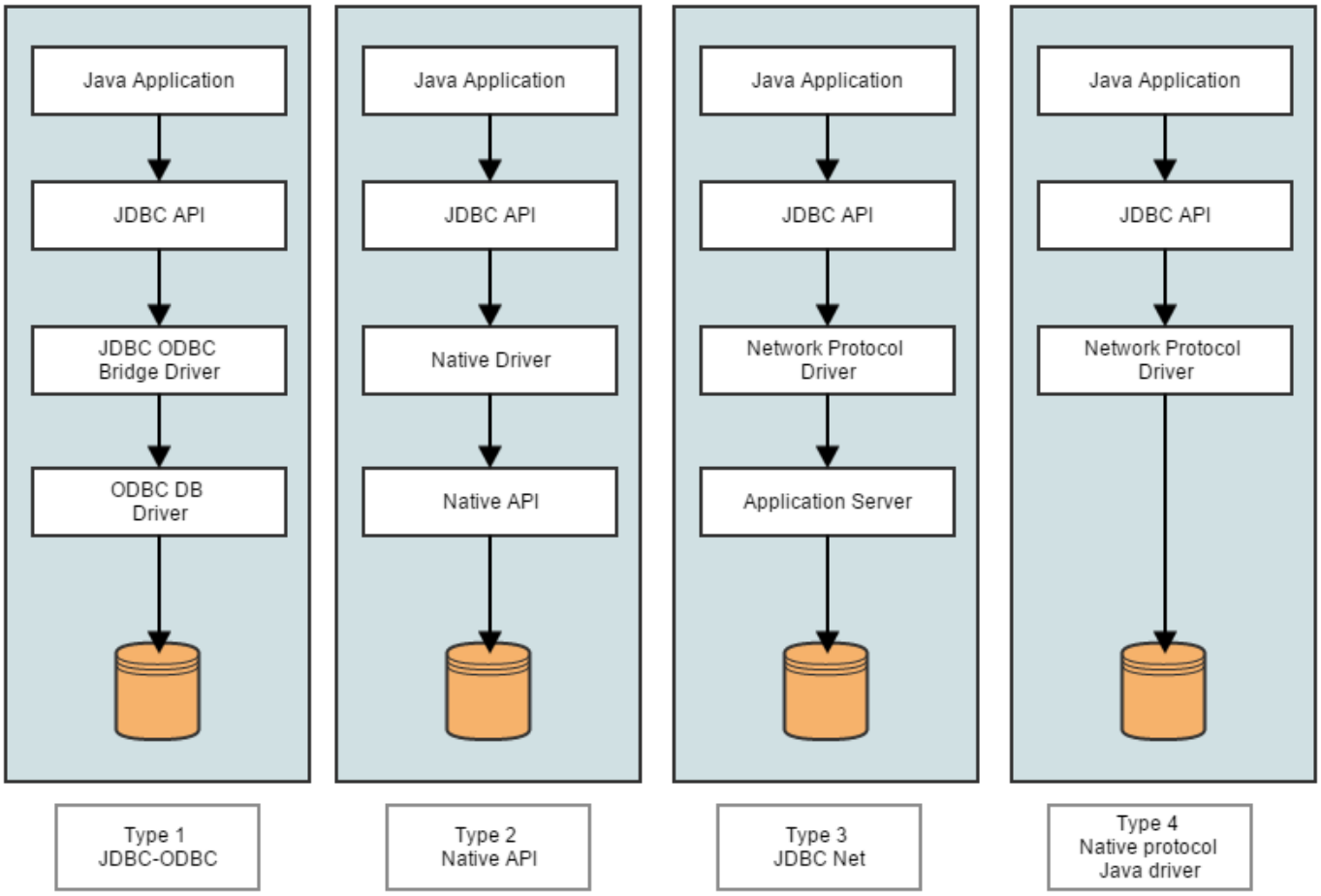

Java Application
JDBC API
JDBC Driver Manager
JDBC Driver
JDBC Driver
JDBC Driver
Database
Database
Database

Using JDBC Drivers

- ✓ DriverManager.registerDriver(new com.mysql.jdbc.Driver());
- ✓ Class.forName("com.mysql.jdbc.Driver").newInstance();
- ✓ System.setProperty("jdbc.drivers", "com.mysql.jdbc.Driver");

Using JDBC Drivers

- ✓ JDBC drivers ensure you can connect to a database
- ✓ JDBC drivers needs to know the URL of the database server
- ✓ URL addresses are specific to the database type we want to connect to:
  - ✓ jdbc:odbc:test
  - ✓ jdbc:mysql://localhost/test
  - ✓ jdbc:oracle:thin@persistentjava.com:1521:test
  - ✓ jdbc:sybase:test


Java Application
JDBC API
JDBC ODBC Bridge Driver
ODBC DB Driver
Type 1 JDBC-ODBC
Java Application
JDBC API
Native Driver
Native API
Type 2 Native API
Java Application
JDBC API
Network Protocol Driver
Application Server
Type 3 JDBC Net
Java Application
JDBC API
Network Protocol Driver
Type 4 Native protocol Java driver

Using JDBC Connections

- ✓ Connection conn = DriverManager.getConnection(url);
- ✓ Connection conn = DriverManager.getConnection(url, username, password);
- ✓ Connection conn = DriverManager.getConnection(url, dbproperties)

Using JDBC Connections

```
Connection con = null;
try {
        Class.forName("com.mysql.jdbc.Driver");
        Connection connection = DriverManager.getConnection
                                (url, “user_name", “user_password");
} catch(ClassNotFoundException e) {
        System.err.print("ClassNotFoundException: " + e) ;
} catch(SQLException e) {
        System.err.println("SQLException: " + e);
} finally {
        // database connections are heavy resources
        connection.close();
}
```

JDBC Statements

- ✓ Creation of a Statement

```
Connection con = DriverManager.getConnection(url);
Statement stmt = con.createStatement();
```

- ✓ Executing an Insert Statement

```
String sql = "SELECT * FROM persons";
ResultSet rs = stmt.executeQuery(sql);
```

- ✓ Executing an Update Statement

```
String sql = "DELETE FROM persons WHERE code > 100";
int lines = stmt.executeUpdate(sql);
```

- ✓ Executing a generic statement

```
stmt.execute("any correct SQL statement");
```

JDBC Prepared Statements

```
String sql = "UPDATE persons SET name=? WHERE code=?";
PreparedStatement pstmt = con.prepareStatement(sql);
pstmt.setString(1, “John");
pstmt.setInt(2, 100);
pstmt.executeUpdate();

pstmt.setString(1, “Doe");
pstmt.setInt(2, 200);
pstmt.executeUpdate();
```

JDBC Callable Statement

```
Connection con = DriverManager.getConnection(url);
CallableStatement cstmt = con.prepareCall("{call storedProcedure(?, ?)}");
cstmt.setString(1, “John");
cstmt.setInt(2, 100);
cstmt.executeQuery();
```

```
CallableStatement cstmt = con.prepareCall("{call anotherProcedure(?)}");
cstmt.registerOutParameter(1, java.sql.Types.FLOAT);
cstmt.executeQuery();
float average= cstmt.getDouble(1);
```

JDBC Data Types

- ✓ pstmt.setObject(1, "Ionescu", Types.CHAR);
- ✓ pstmt.setObject(2, 100, Types.INTEGER);
- ✓ pstmt.setObject(2, 100);
- ✓ pstmt.setNull(1, Types.CHAR);
- ✓ pstmt.setInt(2, null);

JDBC ResultSets

```
ResultSet rs = stmt.executeQuery(“sql statement”);

while (rs.next()) {
        int cod = rs.getInt("code");
        // any number of columns found in the result set
        String nuame = rs.getString("name");
        System.out.println(cod + ", " + name);
}
```

JDBC Meta Data

```
// database meta data
DatabaseMetaData dbmd = con.getMetaData();
ResultSet rs = dbmd.getTables (null, null, null, null);
while (rs.next ())
        System.out.println(rs.getString ("TABLE_NAME"));
        con . close ();
}

// result set meta datas
ResultSet rs = stmt.executeQuery("SELECT * FROM tabel");
ResultSetMetaData rsmd = rs.getMetaData();
int n = rsmd.getColumnCount();
```

JDBC Transaction Control

- ✓ COMMIT, ROLLBACK

```
// commit when ready
con.commit();
// rollback on fail
con.rollback();
```

- ✓ Savepoints

```
Savepoint save1 = con.setSavepoint();
// at some point in time, if errors occur
con.rollback(save1);
```

- ✓ Auto commit, for procedures

```
con.setAutoCommit(false);
```

LEARN stuff IO

Assignments

- ✓Create an application that allows connection to a database of your choosing. Implement, using JDBC and prepared statements, all CRUD operations (create, read, update, delete)

- ✓Modify the Swing UI application for event management to use JDBC instead of file serialization

THANK YOU!

Vlad Costel Ungureanu
ungureanu_vlad_costel@yahoo.com

This is a free course from LearnStuff.io
– not for commercial use –