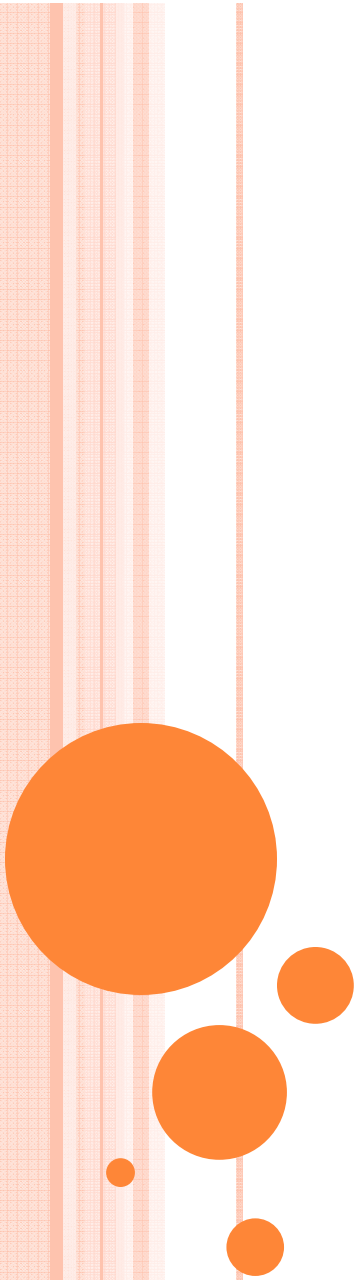

JDBC BASICS

CS 561 Database Management Systems 1

WHAT IS JDBC

Stands For: **J**ava **D**ata **B**ase **C**onnectivity

- Intermediate between the Java Application and database.
- Consists of set of java APIs that helps the java application to interact with all sorts of relational database management systems.
- Makes it possible to write a single database application that can run on different platforms and interact with different DBMSs.

JDBC Drivers

- Set of classes that enables the Java application to communicate with databases.
- Application will talk to the JDBC driver to talk to the respective database.
- Different Drivers for Different Databases
- Four Types of JDBC Drivers

TYPE 1 DRIVER - JDBC-ODBC BRIDGE

- A database driver implementation that employs the ODBC driver to connect to the database.
- The driver converts JDBC method calls into ODBC function calls.
- Platform dependent, ODBC Installation Required, Overhead of extra layer
- Recommended only for experimental use or when no other alternative is available.

Type 2 Driver - Native-API Driver

- A database driver implementation that uses the client-side libraries of the database.
- The driver converts JDBC method calls into native calls of the database API.
- Native API installation required, Database specific (Change in DB causes change in API)
- Mostly obsolete now

TYPE 3 DRIVER - NETWORK-PROTOCOL DRIVER

- A database driver implementation which makes use of a middleware server between the calling program and the database.
- The middleware server converts JDBC calls directly or indirectly into the vendor-specific database protocol
- Platform Independent, No installation required on client
- A single driver can handle any database, provided the middleware supports it.

TYPE 4 DRIVER - NATIVE-PROTOCOL DRIVER

- A database driver implementation that converts JDBC calls directly into a vendor-specific database protocol.
- Does not have the overhead of conversion of calls into ODBC or database API calls
- Platform Independent, Direct connection to the database improves the performance
- Different Drivers for Different Databases

How to Load The Drivers

FOR WINDOWS

- **Integreted Development Environment:**
  - **Eclipse:** After creating a new project -> Right click on Project name> Properties -> Select Java Buildpath-> Add External Jars-> Enter the downloaded jar file path -> Ok
  - **Netbeans:** Go to Services Tab, Right Click on Drivers->New Driver -> Specify the source.
- **Text Editor:**

  Setting the Class Path initially: Go to Environment Variables -> System Variables -> Edit CLASSPATH -> ; + Append the jar file path to it

How to Load The Drivers (Cont.)

FOR LINUX

Integreted Development Environment:

- **Eclipse:** After creating a new project -> Right click on Project name> Properties -> Select Java Buildpath-> Add External Jars-> Enter the downloaded jar file path -> Ok

Text Editor:

Javac **ProgName**.java

Java –cp .:***postgresql-8.4-701.jdbc4.jar*** **ProgName**

How to use JDBC?

1) Importing JDBC:

```
import java.sql.*;
```

2) Loading the Driver:

```
Class.forName("org.postgresql.Driver");
```

3) Connecting to the Database:

```
Connection db = DriverManager.getConnection(url, username, password);
```

url = jdbc:postgresql://localhost: 5432/*database*
username = the postgresql database account username
Password = the postgresql database account password

4) Closing the Connection

```
db.close();
```

SAMPLE CODE

```
public class TestDB {
Class.forName("org.postgresql.Driver");
Connection con = DriverManager.getConnection(
   "jdbc:postgresql://localhost: 5432/DBNAME", "postgres",
   "Password");
Statement st = con.createStatement();
rs = st.executeQuery("select * from sales");
}
```