

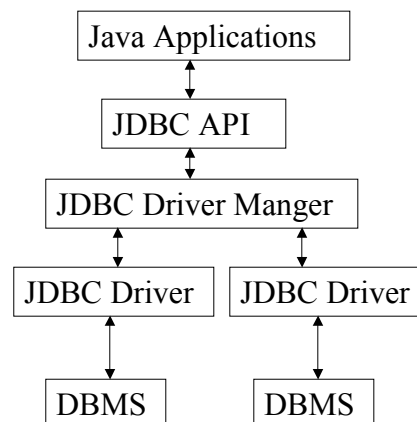
## What is JDBC ?

- > Developed in 1996.
- > Essentially a trademark, but meaning of „Java Database Connectivity“.
- > Its tasks:
  - + establish connection
  - + send SQL statements
  - + process the results
- > It is a part of JDK 1.4

## JDBC Structure

### ◆ Two major sets of interfaces:

- Lower level JDBC driver
- JDBC API

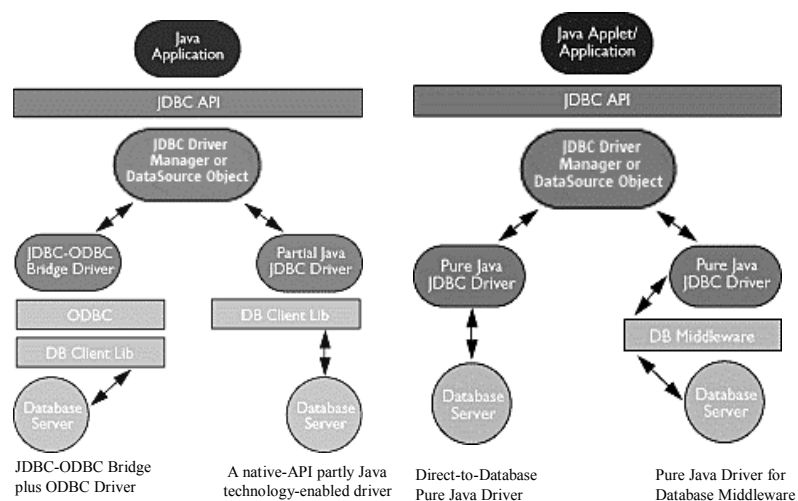


## JDBC Structure (cont)

### ◆ Four drivers:

- JDBC-ODBC Bridge plus ODBC Driver
- A native-API partly Java technology-enabled driver
- Pure Java Driver for Database Middleware
- Direct-to-Database Pure Java Driver.

## Four Drivers



<http://java.sun.com/products/jdbc/datasheet.html>

## Features and Benefits

### ◆ Benefits:

- Flexibility to businesses.
- Easy and economical.
- No configuration for client side is required.

### ◆ Features:

- No Complex Installation.
- Complete access to metadata.
- URL Database Connection.

## JDBC API

### ◆ Data Types

### ◆ Simplified API

- Loading Driver
- Connecting
- Statements
- Results

### ◆ API in Detail

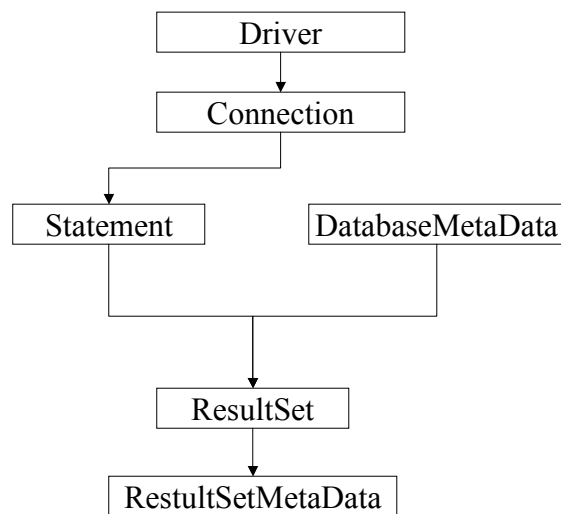
- Callable and Prepared
- Exceptions
- Transactions
- Applets

# Data Types

Java to SQL Mapping. JDBC defines a set of generic SQL type identifiers in the class `java.sql.Types`

- ◆ bit
- ◆ tinyint
- ◆ smallint
- ◆ integer
- ◆ bigint
- ◆ float
- ◆ real
- ◆ double
- ◆ numeric
- ◆ decimal
- ◆ char
- ◆ varchar
- ◆ longvarchar
- ◆ date
- ◆ time
- ◆ timestamp

# JDBC Interface (simplified)



## Establishing a Connection

- ◆ The first thing you need to do is establish a connection with the DBMS you want to use. This involves two steps:
  - loading the driver and
  - making the connection.

## Loading Drivers

- ◆ Loading the driver or drivers you want to use is very simple. For example, if you want to use the JDBC-ODBC Bridge driver, the following code will load it:

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

- ◆ Your driver documentation will give you the class name to use. For instance, if the class name is `exgwe.sql.gweMySQLDriver`, you would load the driver with the following line of code:

```
Class.forName("exgwe.sql.gweMySQLDriver");
```

## Making a Connection

- ◆ The second step is to have the appropriate driver connect to the DBMS.

```
Connection con =  
    DriverManager.getConnection("jdbc:mysql://cannings  
        .org:3306/test", "myLogin", "myPassword");
```

- ◆ The URL consists of the protocol, type of database, hostname, port, and name of the database. In this case we are using JDBC to connect to a mysql database, on host cannings.org, port 3306, to database test.

## Making a Connection in AKI LU

- ◆ In the laboratory you can easily establish a connection by using:

- ```
Connection con =  
    DBConnection.getConnection(request);
```

- You don't need to load a driver! It's done automatically.

- Just import "`at.ac.tuwien.ict.aki.DBConnection`" and provide your `HttpRequest` object. It automatically looks up a suitable connection from a connection pool.

## Creating Statements, Updates

- ◆ A Statement object is what sends your SQL statement to the DBMS. You simply create a Statement object and then execute it.
- ◆ Statements can be used to update the database:

```
Statement stmt = con.createStatement();  
stmt.executeUpdate("CREATE TABLE COFFEES " +  
    "(COF_NAME VARCHAR(32), SUP_ID INTEGER, PRICE FLOAT, " +  
    "SALES INTEGER, TOTAL INTEGER)");
```

## Creating Statements, Queries

- ◆ Statements are also used to Query the database:

```
ResultSet rs = stmt.executeQuery(  
    "SELECT COF_NAME, PRICE FROM COFFEES");
```

## Result Sets

- ◆ JDBC returns results in a `ResultSet` object, so we need to declare an instance of the class `ResultSet` to hold our results.

```
String query = "SELECT COF_NAME, PRICE FROM COFFEES";
ResultSet rs = stmt.executeQuery(query);
while (rs.next()) {
    String s = rs.getString("COF_NAME");
    float n = rs.getFloat("PRICE");
    System.out.println(s + "    " + n);
}
```

## Result Sets, methods

- ◆ The `getXXX` method is used to retrieve data by column name:

```
String s = rs.getString("COF_NAME");
```

- ◆ We may also use a column index to access the data:

```
String s = rs.getString(1);
float n = rs.getFloat(2);
```

- ◆ The `next()` method is used to access the next row in the resultset:

```
while (rs.next()) { /* get current row */ }
```

## More JDBC

◆ We have now been introduced to the following interfaces:

- `java.sql.drivermanager`
- `java.sql.connection`
- `java.sql.statement`
- `java.sql.resultset`
- `java.sql.types`

◆ Now lets look at the rest of the API.

## Callable and Prepared Statements

◆ Prepared Statement

- pre-compiled query
- reduces execution time
- given sql query when it is created
- used for queries that are used many times

◆ Callable Statement

- a stored procedure inside the DBMS
- can be created and accessed by JDBC
- used to encapsulate a set of operations or queries to execute on a database server

## Prepared Statement Example

```
PreparedStatement updateSales =
    con.prepareStatement(
        "UPDATE COFFEE SET SALES = ? WHERE
        COF_NAME LIKE ? ");
updateSales.setInt(1, 75);
updateSales.setString(2, "Columbian");
updateSales.executeUpdate();
```

## Callable Statement Example

◆ Syntax for defining a stored procedure is different for each DBMS

```
String createProcedure = "create procedure SHOW_SUPPLIERS " +
    "as " +
    "select SUPPLIERS.SUP_NAME, COFFEES.COF_NAME " +
    "from SUPPLIERS, COFFEES " +
    "where SUPPLIERS.SUP_ID = COFFEES.SUP_ID " +
    "order by SUP_NAME";
Statement stmt = con.createStatement();
stmt.executeUpdate(createProcedure);

CallableStatement cs = con.prepareCall("{call SHOW_SUPPLIERS}");
ResultSet rs = cs.executeQuery();
```

## Main Exceptions

### ◆ SQLException

- An exception that provides information on a database access error or other errors.

### ◆ SQLWarning

- An exception that provides information on database access warnings. Warnings are silently chained to the object whose method caused it to be reported.

## Summary

### ◆ JDBC's goal is:

- DBMS-independent interface.
- Can access any data source without recoding.

### ◆ JDBC Drivers:

- 1)JDBC-ODBC Bridge
- 2) Partial Java driver
- 3)Pure Java driver for database middleware
- 4)direct to database pure Java driver.

### ◆ JDBC supports basic SQL functionality

### ◆ JDBC is flexible, easy to manage and inexpensive.