Database Programming

Loading Driver

Before connecting to a database, you must first

- **load the driver**

```
Class.forName("JDBCDriverClass");
```

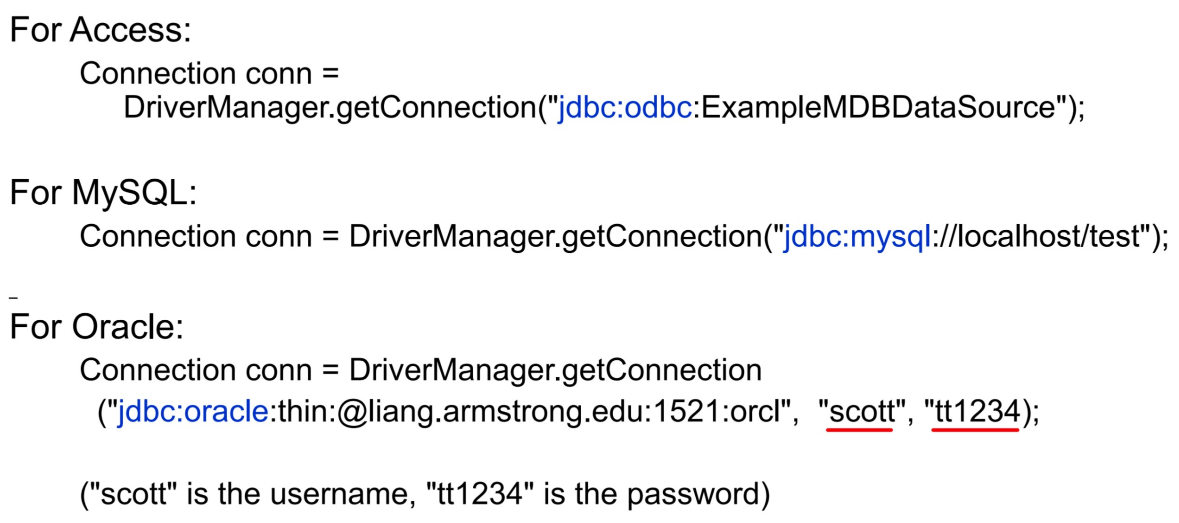
Establishing DB Connection

To establish a connection use this method

`DriverManager.getConnection()`

Example:

```
Connection conn = DriverManager.getConnection(databaseURL); // either relative
or remote
```



For Access:

    Connection conn =
        DriverManager.getConnection("jdbc:odbc:ExampleMDBDataSource");

For MySQL:

    Connection conn = DriverManager.getConnection("jdbc:mysql://localhost/test");

For Oracle:

    Connection conn = DriverManager.getConnection
      ("jdbc:oracle:thin:@liang.armstrong.edu:1521:orcl", "scott", "tt1234");

    ("scott" is the username, "tt1234" is the password)

Creating and Executing Statements

We use the methods provided in the `Connection` class

```
Statement st = conn.createStatement();

// Creating a table
st.executeUpdate("create table <tableName>(<attr1> <datatype>, ... primary
key(attr1))");

// Inserting data into the table
st.executeUpdate("insert into <tableName> values(<init. list here>)");

// Updating data
st.executeUpdate("update <tableName> set <attrName> = <newValue> where
<attrName> = <newValue>");

// To query results (Important)
ResultSet rs = st.executeQuery("select * from <tableName>"); // Stores in
ResultSet (row by row)

// Deleting records
st.executeUpdate("Delete from <tableName> where <attrName> = <value>")
```

ResultSet Class (Displaying Query Results)

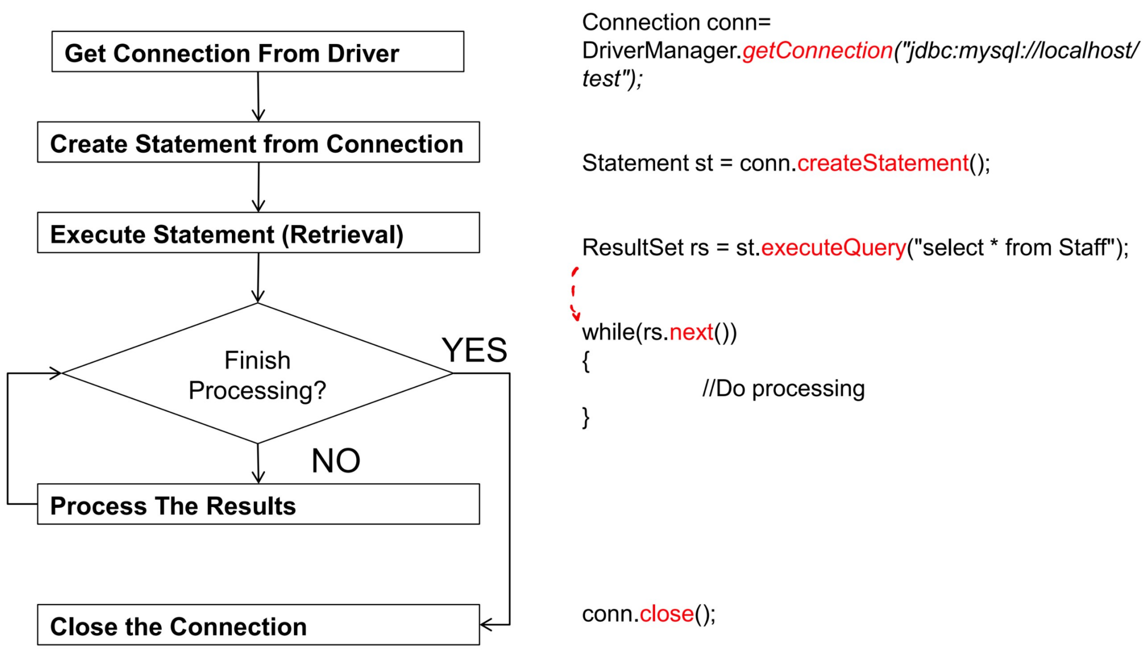
`next()` methods serves 2 purposes

- moves the cursor to the next row
- acts as a boolean flag to check if there are more results

To display the query results we use

```
int nc = 4 // no. of cols

while(rs.next()) { // EOF Flag

    for (int i = 1; i <= nc; i++) {
        System.out.print(rs.getString(i) + "\t");
    }

    System.out.println();
}

// The while loop controls the rows, while the for loop controls the columns

// Close the DB connection when done

conn.close();
```

General Pipeline For DB Programming


Get Connection From Driver
Create Statement from Connection
Execute Statement (Retrieval)
Finish Processing?
YES
NO
Process The Results
Close the Connection

```
Connection conn=
DriverManager.getConnection("jdbc:mysql://localhost/
test");

Statement st = conn.createStatement();

ResultSet rs = st.executeQuery("select * from Staff");

while(rs.next())
{
        //Do processing
}

conn.close();
```

Note: Code for Access Connection

```
Connection conn = DriverManager.getConnection(".../db1.accdb");

Statement statement = conn.createStatement();

statement.executeUpdate("<SQL code here>");
```

The above pieces of code will throw a checked exception (`SQLException`)

PreparedStatement

`Statement` class is used to execute static statements

`PreparedStatement` is to

- Execute precompiled SQL statement **dynamically**
- Optional Parameters

```
PreparedStatement pstmt = conn.prepareStatement("insert into <Table> (<attr(s)>)
values (?, ?, ...)");
```

The `?` acts as placeholders (tokens)

To set the values of the params:

- `pstmt.setInt(1, 4);`
- `pstmt.setString(2, "Jack");`
- `pstmt.setString(3, "Johor");`
- `pstmt.setString(4, "FBF");`

General pattern is `<PreparedStatementObj>.set<DataType>(<PlaceHolderPos>, <Value>)`

Execute `PreparedStatement` using `pstmt.executeUpdate();`

DB Metadata

Metadata describes the table

```
DatabaseMetaData dbMetaData = conn.getMetaData();
```

DatabaseMetaData Methods

Method	Info
`getURL()`	DB URL
`getUserName()`	DB Username
`getDatabaseProductName()`	DB Product Name
`getDriverName()`	JDBC Driver Name
`getDriverVersion()`	JDBC Driver Version
`getMaxConnections()`	Max # of connections
`getTables()`	Refer snippet below

```
ResultSet rsTables = dbMetaData.getTables(null, null, null, new String[]
{"Table"});

System.out.println("User Tables: ");

while (rsTables.next())
    System.out.println(rsTables.getString("TABLE_NAME") + " ");
```

Applying DatabaseMetaData

```
// To avoid hard coding this
// int nc = 4 // no. of cols

// We make use of MetaData

ResultSetMetaData rsMetaData = rs.getMetaData();

// Then do this

int nc = rsMetaData.getColumnCount();

while(rs.next()) { // EOF Flag

    for (int i = 1; i <= nc; i++) {
        System.out.print(rs.getObject(i) + "\t");
    }

    System.out.println();
}

// The while loop controls the rows, while the for loop controls the columns
// Close the DB connection when done

conn.close();
```