JDBC

- JDBC supports querying and updating data
- JDBC also supports metadata retrieval
  - data about relations, names and types of attributes.
- steps:
  - Open a connection
  - Create a “statement” object
  - Execute queries using the Statement object and fetch results
  - Exception mechanism to handle errors

JDBC Code

```
public static void JDBCexample(String userid, String passwd)  {
    try {
         Class.forName ("com.mysql.jdbc.Driver");
         Connection conn = DriverManager.getConnection(
               "jdbc:mysql://localhost:3306/<database_name>", userid, passwd);
        Statement stmt = conn.createStatement();
           … Do Actual Work ….
        stmt.close();
        conn.close();
    }
    catch (SQLException sqle) {
        System.out.println("SQLException : " + sqle);
    }
}
```

JDBC Code (Cont.)

- Update to database

```
try {
   stmt.executeUpdate(
       "insert into instructor values(’77987’, ’Kim’,
       ’Physics’, 98000)");
} catch (SQLException sqle)
  {
  System.out.println("Could not insert tuple. " +
       sqle);
  }
```

- Execute query and fetch and print results

```
    ResultSet rset = stmt.executeQuery(
                     "select dept_name, avg (salary)
                      from instructor
           group by dept_name");

      while (rset.next()) {
             System.out.println(rset.getString
                                ("dept_name") + " ”
                                 rset.getFloat(2));
      }
```

Note : result metadata is available

JDBC Code Details

- Dealing with Null values

```
int a = rs.getInt(“a”);
if (rs.wasNull()) Systems.out.println(“Got
        null value”);
```

Prepared Statement : provide for parameters

```
PreparedStatement pStmt = conn.prepareStatement(
          "insert into instructor values(?,?,?,?)");

pStmt.setString(1, "88877");
pStmt.setString(2, "Perry");
pStmt.setString(3, "Finance");
pStmt.setInt(4, 125000);
pStmt.executeUpdate();
 …..
pStmt.setString(1, "88878"); …..
pStmt.executeUpdate();
```

- For queries, use pStmt.executeQuery(), which returns a ResultSet

- WARNING: always use prepared statements when taking an input from the user and adding it to a query
  - NEVER create a query by concatenating strings which you get as inputs

```
"insert into instructor values(’ " + ID + " ’, ’ " +          name + " ’, "  +" ’ +
   dept name + " ’, " ’      balance + ")“
```

  - What if name is “D’Souza”?

SQL Injection defect

- Suppose query is constructed using

"select * from instructor where name = ’" + name + "’"

- User may enter wrong values
  - X’ or ’Y’ = ’Y
- then the resulting statement becomes:

```
select * from instructor where
    name = ’X’ or ’Y’ = ’Y’     --- will get full table !
```

- User could have even used

X’; update instructor set salary = salary + 10000;

- Always use prepared statements, with user inputs as parameters
- To avoid ‘sql injection’ error

Metadata Features

- ResultSet metadata – get column names and types after executing query to get a ResultSet rs:

```
ResultSet rs = …
 ResultSetMetaData rsmd = rs.getMetaData();

 for(int i = 1; i <= rsmd.getColumnCount(); i++) {
      System.out.println(rsmd.getColumnName(i));
     System.out.println
           (rsmd.getColumnTypeName(i));
    }
```

Metadata (Cont)

- Database metadata : table, column names, ….

```
DatabaseMetaData dbmd = conn.getMetaData();
ResultSet rs = dbmd.getColumns(null, "univdb", "department",
"%");
// Arguments to getColumns are Catalog, Schema-pattern, Table-
name,  and Column-name
// Returns: One row for each column containing
COLUMN_NAME, TYPE_NAME
```

Transaction Control in JDBC

- By default, each SQL statement is treated as a separate transaction that is committed automatically
- Can turn off automatic commit on a connection
  - conn.setAutoCommit(false);
- Transactions must then be committed or rolled back explicitly
  - conn.commit();   or
  - conn.rollback();
- conn.setAutoCommit(true) turns on automatic commit.