Using JDBC

JDBC (Java Database Connectivity) is an API that describes how application developers can connect to relational databases and execute SQL statements from a Java program. JDBC is supported by most DBMS such as Oracle, mySQL, SQL Server, DB2 etc, by providing drivers that implement the JDBC API.

Check the JDBC API from java.sun.com/j2se/1.4.2/docs/api ; check for the package java.sql
Also find introduction to JDBC with Oracle from the Stanford DB 1 class at
http://www-db.stanford.edu/~ullman/fcdb/oracle/or-jdbc.html

The first step in using JDBC with any RDBMS is to install a suitable driver. If running from CCC machines, the Oracle JDBC driver is already installed. You need to only set the classpath as shown below. However, if you are using your own machine for development, you may want to grab the Oracle JDBC driver and include it in your build path. For mySQL, you have to install the driver (even if you are using CCC machines).

JDBC with Oracle

Check details of Oracle JDBC driver at $ORACLE_HOME/jdbc/Readme.txt Include the JDBC driver in your classpath as (in .cshrc), if you are using CCC unix machines for application development.

setenv CLASSPATH ${CLASSPATH}:${ORACLE_HOME}/jdbc/lib/ojdbc6.jar

To use JDBC in your code, you have to do the following steps:
Load the driver as: Class.forName ("oracle.jdbc.driver.OracleDriver");
Connect to the database: Connection conn = DriverManager.getConnection
("jdbc:oracle:thin:@oracle.wpi.edu:1521:CS", "<uname>", "<passwd>");

Replace <uname> and <passwd> with your Oracle user name and password.
Here oracle.wpi.edu is our oracle DB server machine, 1521 is the port where Oracle server listens to for JDBC connections, and WPIDBR2 is the database instance where we create our tables.

Now you are all set to prepare and execute statements.
Create a statement by Statement stmt = conn.createStatement ();

To execute a query, we use either
stmt.executeQuery (sql) (or)
stmt.executeUpdate (sql)

Here sql is a string representing the SQL query. We use executeQuery for SELECT statements (statements that return a set of rows), and executeUpdate for DDL statements (creating tables etc), and for data modification (inserting/deleting/updating tuples).

Let us look at a full example, which will create a table a (a1 int, a2 int), insert two rows in it, and select those rows and display them.

```
import java.sql.*;
import java.util.*;
import java.io.*;

public class Test {

    public static void main (String args []) {
      Connection conn = null;
      Statement stmt = null;
      try {
```

```
            Class.forName ("oracle.jdbc.driver.OracleDriver");
                                                    // Load Driver
            conn = DriverManager.getConnection
              ("jdbc:oracle:thin:@oracle.wpi.edu:1521:WPIDBR2",
                          "<userName>", "<passwd>"); // Connect to DB
            stmt = conn.createStatement (); // Create statement
        } catch (Exception ex) {
            System.out.println ("Could not create Statement:" +
                                ex.toString ());
          }
        String sql = "CREATE TABLE a (a1 int, a2 int)";
        try {
            stmt.executeUpdate (sql); // create the table
        } catch (SQLException sqlEx) {
            System.out.println ("Could not create table:" +
                                sqlEx.toString ());
        }
        String ins1 = "INSERT INTO a values (1, 1)";
        String ins2 = "INSERT INTO a values (2, 2)";
        try {
            stmt.executeUpdate (ins1); // insert the values
            stmt.executeUpdate (ins2);
        } catch (SQLException sqlEx) {
              System.out.println ("Could not create table:"
                                + sqlEx.toString ());
          }

        sql = "SELECT * from a";
        try {
            ResultSet r = stmt.executeQuery (sql); // execute the query
            while (r.next ()) {
                System.out.println ("a1 = " + r.getInt ("a1") +
                                ": a2 = " + r.getInt ("a2"));
            }
        } catch (SQLException sqlEx) {
            System.out.println ("Could not get results:" +
                                      sqlEx.toString ());
        }
        try {
            stmt.close (); // Close the statement
            conn.close (); // Close the connection
        } catch (SQLException sqlEx) {
            System.out.println ("Could not close connection:" +
                                      sqlEx.toString ());
        }
    }
}
```

JDBC with mySQL

First, we need to download the JDBC driver, called Connector/J from www.mysql.com
Set your classpath to include the driver by
setenv CLASSPATH ${CLASSPATH}:*<dir>*/mysql-connector-java-3.0.10-stable-bin.jar
Replace *<dir>* with the directory where you have installed the JDBC driver.

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

```
import java.util.*;
import java.io.*;

public class TestMySQL {

    public static void main (String args []) {
      Connection conn = null;
      Statement stmt = null;
      try {
          Class.forName ("com.mysql.jdbc.Driver"); // Load Driver
          conn = DriverManager.getConnection
            ("jdbc:mysql://mysql.wpi.edu/research",
                      "<userName>", "<passwd>"); // Connect to DB
          stmt = conn.createStatement (); // Create statement
      } catch (Exception ex) {
          System.out.println ("Could not create Statement:" +
                                    ex.toString ());
        }
      String sql = "CREATE TABLE a (a1 int, a2 int)";
      try {
          stmt.executeUpdate (sql); // create the table
      } catch (SQLException sqlEx) {
          System.out.println ("Could not create table:" +
                               sqlEx.toString ());
      }
      String ins1 = "INSERT INTO a values (1, 1)";
      String ins2 = "INSERT INTO a values (2, 2)";
      try {
          stmt.executeUpdate (ins1); // insert the values
          stmt.executeUpdate (ins2);
      } catch (SQLException sqlEx) {
            System.out.println ("Could not create table:" +
                                    sqlEx.toString ());
        }

      sql = "SELECT * from a";
      try {
          ResultSet r = stmt.executeQuery (sql); // execute the query
          while (r.next ()) {
              System.out.println ("a1 = " + r.getInt ("a1") +
                               ": a2 = " + r.getInt ("a2"));
          }
      } catch (SQLException sqlEx) {
          System.out.println ("Could not get results:" +
                                    sqlEx.toString ());
      }
      try {
          stmt.close ();
          conn.close ();
      } catch (SQLException sqlEx) {
          System.out.println ("Could not close connection:" +
                                    sqlEx.toString ());
      }
    }
}
```