Java Database Connectivity and Structured Query Language

ECE 155B – Spring 2012
Discussion Section 6 & 7

JDBC, SQL and Relational Database URLs

Basic Tutorial on JDBC
http://java.sun.com/docs/books/tutorial/jdbc/

Getting Started with the JDBC API
http://java.sun.com/j2se/1.4.1/docs/guide/jdbc/getstart/GettingStartedTOC.fm.html

JDBC API in a Nutshell
http://www.cs.unc.edu/Courses/wwwp-s98/members/thornett/jdbc/183.html

Database Programming with JDBC and Java: Chapter 4. Database Access Through JDBC
http://oreilly.com/catalog/javadata/chapter/ch04.html

A Gentle Introduction to SQL
http://sqlzoo.net

MySQL Reference Manual
http://dev.mysql.com/doc/mysql/en/

15 Seconds: Introduction to Relational Databases
http://www.15seconds.com/issue/020522.htm

XML Representation of a Relational Database
http://www.w3.org/XML/RDB.html

Java Database Connectivity (JDBC)

- JDBC provides APIs for interacting with many different database systems
- JDBC requires suitable drivers for specific database systems

Structured Query Language (SQL)

- SQL is a standard (relational) database language
- MySQL is a popular SQL database system

How do I use JDBC?

In your source files you must `import java.sql.*;`
Now you can use the necessary objects for creating and executing database queries. However, before you can run your application, you must install a suitable driver for your target database system. In our case we will be using MySQL.

Where do I find a suitable JDBC driver for MySQL?

You will need to download the MySQL Connector/J driver from:
http://dev.mysql.com/downloads/connector/j/3.0.html.

To install the connector:
1) Unpack the archive
2) Put the org and com directories somewhere safe
3) Make sure your CLASSPATH includes the path to these directories

How do I setup my database?

Please contact Firat as soon as you know you will be doing this assignment. He will perform the initial setup for your particular database. Your database will be accessible by your last name.

After the database is created, you will need to create the tables for your database. A database can contain as many tables as you like. However, for this assignment, you shouldn't have more than three, in particular, one for Patient records, one for Diagnosis records and one for Information records. A Diagnosis record refers to a Patient record by the social security number of that patient and, similarly, for an Information record.

You setup your database by:
Use mysql admin application.

Server Host: *ece155b.ece.ucsb.edu*
User name: Your first name
Password: Your first name
Database: Your first name

Send me email to get username/password that you want to use.

What are the MySQL commands?

CREATE TABLE – used to create a new table in your database

```
CREATE TABLE tableName (
    id INTEGER PRIMARY KEY,
```

```
    fName VARCHAR(30),
    lName VARCHAR(30)
      );
```

DROP TABLE – used to delete tables from your database

DROP TABLE tableName;

TRUNCATE TABLE – used to empty a table

TRUNCATE TABLE tableName;

INSERT TABLE – used to insert values into a table

INSERT INTO tableName VALUES (23, “Joe”, “Shmoe”);

SELECT QUERY – used for retrieving matching database rows

SELECT * FROM tableName WHERE fName=”Joe”;

UPDATE QUERY – used to update rows in your database

UPDATE tableName SET id=0 WHERE id=23;

DELETE TABLE – deletes rows from a table

DELETE FROM tableName WHERE id=0;

How do I talk to a MySQL database from my Java program?

You talk to a MySQL database from your Java program by creating a statement on a connection, and invoking an execute method on that statement. The execute method has a String parameter that corresponds to the SQL command that you wish to execute.

For example, once you have a connection object conn, you create a statement that will be sent over that connection to the database.

Statement stmt = conn.createStatement();

Then you execute an update or a query on that statement with a String parameter str that corresponds to the SQL command that you wish to execute.

stmt.executeUpdate(str);

stmt.executeQuery(str);

See the example on the next page.

```
import java.sql.*;
public class testQuery {
  public static void main(String[] args) {
    Connection conn = null;

    try {
    Class.forName("com.mysql.jdbc.Driver").newInstance();

    } catch (Exception ex) {
         // handle the error
     }

    try {
      conn = DriverManager.getConnection(
           "jdbc:mysql://ece155b.ece.ucsb.edu:3306/myTable",
           "myName", "myPasswd");
      Statement stmt = null;
      ResultSet rs = null;

      try {
        stmt = conn.createStatement();
        rs = stmt.executeQuery("SELECT * FROM tableName");

        while(rs.next()) {
          System.out.println(rs.getInt(1));
          System.out.println(rs.getString(2));
          System.out.println(rs.getString(3));
        }
      }
      finally {
        // it is a good idea to release
        // resources in a finally{} block
        // in reverse-order of their creation
        // if they are no-longer needed

        if (rs != null) {
          try {
            rs.close();
          } catch (SQLException sqlEx) {  }

          rs = null;
        }

        if (stmt != null) {
          try {
            stmt.close();
          } catch (SQLException sqlEx) {  }

          stmt = null;
        }
      }
     } catch (SQLException ex) {
        // handle any errors
        System.out.println("SQLException: " + ex.getMessage());
        System.out.println("SQLState: " + ex.getSQLState());
        System.out.println("VendorError: " + ex.getErrorCode());
     }
  }
}
```

By default, JDBC is designed to commit each individual operation automatically. For a transaction, you need to ensure that all of the operations in the transaction are completed successfully before committing. To do that, you say “setAutoCommit(false)” at the beginning of the transaction. At the end of the transaction when you are sure everthing is ok, you invoke the “commit()” method and then say “setAutoCommit(true)”.

Example:

```
MyTransaction()
{
  setAutoCommit(false);
  operation_1 (if error happens return)
  operation_2 (if error happens return)
  operation_3 (if error happens return)
  commit();
  setAutoCommit(true);
}
```

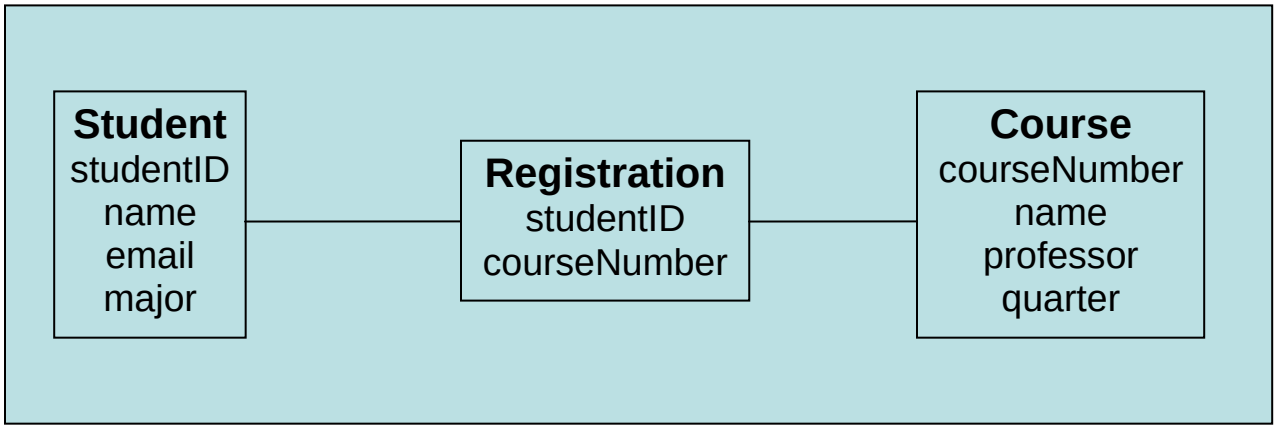
What should you not do?

Do not under any circumstances modify the mysql setup or any database other than your own.

Example MySQL Database

Design

In this example we will build a simple three table database. The database has three tables: Student, Registration and Course tables. The following Entity Relationship (ER) diagram shows the basic design of the database:



The records in the Student table have fields (attributes) for studentID, name, email and major. The studentID is the Primary Key, which uniquely identifies each record in the table. The records in the Course table have fields for courseNumber, name, professor, quarter. The courseNumber is the Primary Key. The Registration table links the records in the Student table with the records in the Course table. A student can register for many courses and a course can have many students. The relationship between course and student is stored in the records in the Registration table, which have fields studentID and courseNumber that correspond to the Primary Keys of the related tables. Because a student cannot be registered for a course twice, both fields (studentID and courseNumber) are the joint Primary Key for the records in the Registration table.

SQL Data Definition

To create the database in MySQL we need to convert the above design into SQL statements. To create the three tables in the database, we use the CREATE TABLE statement as shown below:

```
CREATE TABLE Student (
  ID int NOT NULL AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(30) NOT NULL,
  email VARCHAR(45),
  major VARCHAR(15)
);
```

```
CREATE TABLE Course (
  courseNumber INT NOT NULL PRIMARY KEY,
  name VARCHAR(25) NOT NULL,
  professor VARCHAR(30),
  quarter VARCHAR(10) NOT NULL
);

CREATE TABLE Registration (
  studentID INT NOT NULL,
  courseNumber INT NOT NULL,
  PRIMARY KEY (studentID, courseNumber)
);
```

Note: You should take advantage of the AUTO_INCREMENT command (used in the Student table above) because it will enable you to assign a PRIMARY key without worrying about it. No counter is needed.

Data Types

Each field (column) must have a data type defined for it. As with most database systems, MySQL provides a number of data types. The most common are INT (an integer), FLOAT (a floating point number), DOUBLE (a double precision floating point number), DATE, DATETIME, CHAR (a fixed-length string), and VARCHAR (a variable-length string). We used INT and VARCHAR in the example database. See the MySQL Reference Manual for more information about the data types that MySQL supports.

NOT NULL, AUTO_INCREMENT, PRIMARY KEY Keywords

Besides the data types in the above example, we used the following three key words or phrases.

- NOT NULL
- AUTO_INCREMENT
- PRIMARY KEY

NOT NULL means that the field must have a value for every record.

AUTO_INCREMENT means that the field automatically gets the largest value plus one for each insert where the column value is 0 or NULL. Note that not all SQL database systems use this syntax for auto-numbering.

PRIMARY KEY indicates that the field (attribute) is the primary key for that table. Note that in MySQL, for tables where the Primary Key is made up of more than one attribute, you cannot use the keyword after each attribute name. Instead, you must define the Primary Key at the end of the SQL statement with the attributes in parenthesis (see the CREATE TABLE statement for the Registration table above).

SHOW, DESCRIBE, DROP Commands

Some other useful MySQL commands are SHOW, DESCRIBE, DROP.

The SHOW command creates a list of tables in the database. The following is an example of how you use the SHOW command in MySQL:

```
mysql> show tables;
+------------------+
| Tables in CRS  |
+------------------+
| Course          |
| Registration    |
| Student         |
+------------------+
```

The name of the database is CRS. It consists of three tables: Course, Registration, and Student.

To obtain a description of a table type: SHOW tableName. For example:

```
mysql> describe Course;
+-----------------+--------------+-------+-------+-----------+------------------+
| Field           | Type         | Null  | Key   | Default |     Extra        |
+-----------------+--------------+-------+-------+-----------+------------------+
| courseNumber| int(8)       |       | PRI  | 0         |                  |
| name           | varchar(25)|       |       |           |                  |
| professor      | varchar(30)| YES  |       | NULL    |                  |
| quarter        | varchar(10)|       |       |           |                  |
+-----------------+--------------+--------+-------+-----------+------------------+

mysql> describe Registration;
+-----------------+---------------+---------+-------+-----------+-----------------+
| Field           | Type          | Null    | Key   | Default |     Extra       |
+-----------------+---------------+---------+-------+-----------+-----------------+
| studentID      | int(9)        |         | PRI   |    0      |                 |
| courseNumber| int(8)        |         | PRI   |    0      |                 |
+-----------------+---------------+---------+-------+-----------+-----------------+

mysql> describe Student;
+----------------+---------------+----------+-------+-----------+--------------------+
| Field          | Type          | Null     | Key   | Default |     Extra         |
+----------------+---------------+----------+-------+-----------+--------------------+
| studentID     | int(9)        |          | PRI   |     0     | auto_increment |
| name          | varchar(30) |          |       |           |                    |
| email         | varchar(45) | YES     |       | NULL    |                    |
| major         | varchar(15) | YES     |       | NULL    |                    |
+----------------+------------ ---+----------+-------+-----------+--------------------+
```

Finally, you might want to delete a table. You can do this with the DROP TABLE command. If tableName is the name of the table you want to delete, you would say:

```
DROP TABLE tableName;
```

Inserting Data

Adding data to a table is done with the INSERT command. It looks like this:

```
INSERT INTO Course VALUES
  (100, 'ECE 155B Network Programming', 'John Smith', 'Spring 2005')
);
```

Creating a Simple Query

The SELECT command is used for retrieving data. The syntax of SELECT is as follows:

```
SELECT column(s) (or '*' for all)
FROM table(s)
[WHERE constraint(s)];
```

An example of a simple query is:

```
mysql> SELECT * FROM Student;
+-------------+-----------------+---------------------------+---------+
| studentID | name            | email                     | major |
+-------------+-----------------+---------------------------+---------+
|123456789 | Bill Lindgren   | lindgren@cs.ucsb.edu |   CE    |
|987654321 | Rajiv Alur      | alur@ece.ucsb.edu    |   CS    |
|654321987 | Yin Yang        | yang@ece.ucsb.edu    |  ECE    |
+--------------+-----------------+---------------------------+---------+
```

The optional WHERE clause allows you to retrieve a limited data set. For example:

```
mysql> SELECT * FROM Course
    -> WHERE quarter = 'Spring 2011';
+------------------+-----------------------------------+---------------+----------------+
| courseNumber | name                              | professor     | quarter      |
+------------------+-----------------------------------+---------------+----------------+
|  ECE 155B       | Network Computing             | Moser         | Spring 2005 |
|  ECE 154        | Computer Architecture         | Butner        | Spring 2005 |
|  CS 174A        | Database Systems              | Su            | Spring 2005 |
+------------------+-----------------------------------+---------------+----------------+
```