CSC 440
Database Management Systems
JDBC

This presentation uses slides and lecture notes available from
http://www-db.stanford.edu/~ullman/dscb.html#slides

The Project: What You Will Need
- DBMS
- SQL (DDL and DML)
- Host languages (Java, C/C++, Perl, …)
- Web application servers (optional)
- SQL editors (optional) – e.g., Toad
- Tools for user interface (optional): forms, reports, etc.

Course DBMS
- Oracle
- Information about accessing the course DBMS:
  http://www.csc.ncsu.edu/techsupport/technotes/oracle.php
SQL

- A data-definition and data-manipulation language
- Can be used for ad-hoc queries on (relational) databases
  - Generic SQL interface: users sit at terminals and ask queries on database
- Can be used in programs in some host language
  - Programs access (relational) database by “calls” to SQL statements

Connecting SQL to Host Language

- Embedded SQL
  - Special SQL statements (not part of host language)
  - Preprocessor transforms SQL statements into host-language code
- Call-level interfaces:
  - SQL/CLI (adaptation of ODBC)
  - JDBC: links Java programs to databases

JDBC Basics

- Read the tutorial at http://java.sun.com/docs/books/tutorial/jdbc/basics/
Two-Tier Model

Application

JDBC Driver

Data source

Steps to Use JDBC

- Loading a driver for our db system
  - Creates a DriverManager object
- Establishing a connection to database
  - Creates instance of a Connection object
- Using the connection to:
  - Create statement objects
  - Place SQL statements “in” these objects
  - Bind values to SQL statement parameters
  - Execute the SQL statements
  - Examine results tuple-at-a-time

DBMS Driver

- Specific information you need to know: see the sample JDBC program and the project FAQ on:
  - Driver for the course DBMS
  - Using the driver (add to classpath)
  - Driver specifics for your programs
```java
// Loading the driver:
Class.forName("oracle.jdbc.driver.OracleDriver");

// Establishing a connection:
Connection conn =
DriverManager.getConnection(jdbcURL, user, passwd);
```

**Statements**

Two JDBC classes:
- `Statement`: object that can accept and execute a string that is a SQL statement
- `PreparedStatement`: object that has an associated SQL statement ready to execute

**Using Statements in JDBC**

- Creating statements: using methods in the `Connection` class
- Executing statements:
  - `executeUpdate`: for database modifications
  - `executeQuery`: for database queries
firstExample.java

```java
// Create a statement object that will be sending your
// SQL statements to the DBMS:
Statement stmt = conn.createStatement();
// Create the COFFEES table:
stmt.executeUpdate("CREATE TABLE COFFEES " +
                    "("COF_NAME VARCHAR(32), SUP_ID INTEGER," +
                    " PRICE FLOAT, SALES INTEGER, TOTAL INTEGER")");
// Populate the COFFEES table:
stmt.executeUpdate("INSERT INTO COFFEES " +
                    "VALUES ('Colombian', 101, 7.99, 0, 0)");
// Get data from the COFFEES table:
ResultSet rs = stmt.executeQuery("SELECT COF_NAME,
                                    PRICE FROM COFFEES");
```

ResultSet

- An object of type ResultSet is like a cursor
- Method "next" advances cursor to next tuple:
  - The first time next() returns the first tuple
  - If no more tuples then next() returns FALSE
- Accessing components of tuples:
  - Method getX(name), where X is some type and
    name is an attribute name

firstExample.java

```java
// Now rs contains the rows of coffees and prices from
// the COFFEES table. To access the data, use the
// method
// NEXT to access all rows in rs, one row at a time
while (rs.next()) {
    String s = rs.getString("COF_NAME");
    float n = rs.getFloat("PRICE");
    System.out.println(s + " " + n);
}
```
The general form of a URL is:
```
jdbc:oracle:<drivertype>::<username>/<password>@<database>
```

The `<drivertype>` is one of:
- `thin`  
  - Use Java sockets  
  - Recommended for our class  
- `oci`  
  - Use Oracle OCI calls  
  - Works through SQL *Net  
- `kprb`  
  - Mainly for stored procedures

Notes:
- The `<username/password>` is either empty or of the form `<username>/<password>`
- A URL like:
  
  `jdbc:oracle:thin://mydatabase`

  has an empty username and password whereas this URL:
  
  `jdbc:oracle:thin://mydatabase`

  does not specify a username and password. When using this form the username and password must be provided some other way.

The `<database>` description depends on the driver type:
- (`thin` or `oci` driver and not bequeath) the database description is one of the following:
  - Service approach
    
    `//<host>:<port>/<service>`
  - SID approach
    
    `<host>:<port>:<SID>`
    
    Discouraged; Oracle will stop supporting this form.
  - `TNSName`
    
    TNS (Transparent Network Substrate)
    
    You need to define the parameters for the TNS name
Example

- In our class:
  ```java
  jdbc:oracle:thin:@orca.csc.ncsu.edu:1521/ORCL.WORLD
  ```
  Or
  ```java
  jdbc:oracle:thin:@orca.csc.ncsu.edu:1521:ORCL
  ```
  - Driver type:
  - Hostname:
  - Port:
  - Service:
  - SID:

JDBC Object Summary

- Basic JDBC objects:
  - DriverManager (DataSource is used instead in most applications)
  - Connection
    - Abstract representation of a DBMS session
  - Statement
    - Can be used to execute queries and update the database
  - ResultSet (= cursor)
    - Used to hold answers to database queries

In-Class Exercises - sqlplus

- login to `remote-linux.eos.ncsu.edu` using your unity account.
- add oracle10g
- run sqlplus
  - Username: `<unity ID>@orcl.world`
  - Password: 9 digit student ID
- check a few tables
  - `tabs`: all tables the user has
  - `session_privs`: privileges in this session
  - `session_roles`: roles in this session
- Useful SQL statement:
  - `SELECT * FROM <table-name>;`
In-Class Exercises - JDBC

- Preparation (see jdbc-prep.txt on course website)
  - Install JDK
  - Download JDBC driver
  - Configure CLASSPATH
  - Test running javac and java
  - Should be done before class

Ex. 1
- Open and read firstExample.java
- Recognize critical steps just discussed in class
- Edit the program to put in your username and password
- Compile and run
- Report the output

Ex. 2
- Use sqlplus to connect the Oracle server
  - Check tabs
  - Check table COFFEES

Useful SQL statements:
- SELECT * FROM <table-name>;
- DROP TABLE <table-name>;
In-Class Exercises - JDBC

- Ex. 3:
  - The current program uses the service format for URL.
  - Change it to SID format.
  - Redo Ex 1 & 2.

- Useful SQL statements:
  - SELECT * FROM <table-name>;
  - DROP TABLE <table-name>;

In-Class Exercises - JDBC

- Ex. 4:
  - It’s annoying to have to drop table COFFEES through sqlplus.
  - Modify the program to drop table COFFEE if it’s already there. Report the drop action if it happens.
  - Hint: Just drop the table and catch the SQLException.

In-Class Exercises - JDBC

- Ex. 5 (Optional)
  - Use PreparedStatement instead of Statement to update the COFFEEES table.
  - Put the values into arrays.
  - You need to set parameters of the PreparedStatement.
  - Redo Ex. 1 & Ex. 2.