Java Database Connectivity (JDBC)

Instructor: Louise Moser
TA: Yung-Ting Chuang
Quarter: Spring 2012

Project3 announcement…

- No office hour next Wednesday
- So for students who haven't demo project3 to me, please demo to me next Friday
- Project4 is due on May 23 (Wed). But if you wish to demo to me on May 25 (Fri), that's fine ☺

Project Goal: Moving data from all your XML files into MySQL database

- XML vs Database
  - In XML we load all information toward a vector initially, and save this vector back to XML file at the end
  - But in database, we don't want to load all information in the beginning
  - Instead, we retrieve and update only the information we need
    - Example: When you load patient list, you don't need to load all patient appointments, prescriptions, notes etc.

JDBC

- **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

JDBC Project

- JDBC Servers
  - ece155b.ece.ucsb.edu:3306
- If you need your local copy of database
  - Download MySQL & MySQL Administrator from http://dev.mysql.com website…
  - Email me for more detail if you want to know more
- Example template project available at project website

Supporting Software

- MySql Query Browser
  - Send & View Queries
  - Add/Delete/Modify your data
  - See “MySQL Installation Notes” from class website to download Query Browser
- MySQL Database (Optional)
  - To have your local database in your computer
- MySql Administrator (Optional)
  - Access your account
  - Create/Update/Drop tables

SQL

- SQL Data Definition Language (DDL)
  - *CREATE TABLE - creates a new database table*
  - *ALTER TABLE - alters (changes) a database table //Optional*
  - *DROP TABLE - deletes a database table*
  - *CREATE INDEX - creates an index (search key) //Optional*
  - *DROP INDEX - deletes an index* //Optional
- SQL Data Manipulation Language (DML)
  - *SELECT - extracts data from a database table*
  - *UPDATE - updates data in a database table*
  - *DELETE - deletes data from a database table*
  - *INSERT INTO - inserts new data into a database table*

SQL Commands

- Create a Database
  - To create a database:
  - CREATE DATABASE database_name

SQL Commands, cont.

- **Create a Table**
  - To create a table in a database:
  - CREATE TABLE table_name (
    column_name1 data_type,
    column_name2 data_type,
)
- Example
  - CREATE TABLE Person (
    LastName varchar(45),
    FirstName varchar(45),
    Address text,
    Age int
)

SQL Commands, cont.

- **Delete a Table or Database**
  - To delete a table (the table structure, attributes, and indexes will also be deleted):
  - DROP TABLE table_name
- **To delete a database**
  - DROP DATABASE database_name
- **Truncate a Table**
  - What if we only want to clear the data inside a table?
  - Use the TRUNCATE TABLE command (deletes only the data inside the table)

SQL Commands, cont.

- ALTER TABLE
  - The ALTER TABLE statement is used to add or drop columns in an existing table.
  - ALTER TABLE table_name
    ADD column_name datatype
  - ALTER TABLE table_name
    DROP COLUMN column_name
  - Note: Some database systems don't allow the dropping of a column in a database table (DROP COLUMN column_name).

SQL Queries

- SELECT
  - SELECT * FROM Supply
  - **ORDER BY keyword is used to sort the result**

ID	Name	Brand	Price
ID1	IPOD Nano	Apple	149.99
ID2	Doritos Nacho Cheese	Doritos	4.00
ID3	POP Tarts	Kellogg's	5.00

Where Clause

Operator	Description
=	Equal
<>	Not equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern

The LIKE Condition

- The LIKE condition is used to specify a search for a pattern in a column.
- Syntax
  - SELECT column FROM table
    WHERE column LIKE pattern
- A "%" sign can be used to define wildcards (missing letters in the pattern) both before and after the pattern.

The LIKE Condition, cont.

- Select * from Supply where ID like '%ab%',
  - returns supplies whose ID includes 'ab'
- Select * from Supply where ID like 'ab%',
  - returns supplies whose ID starts with 'ab'
- Select * from Supply where ID like '%ab',
  - returns supplies whose ID ends with 'ab'
- Select * from Supply where ID = 'ab',
  - returns supplies whose ID IS 'ab'

Insert Into

- **The INSERT INTO Statement**
  - The INSERT INTO statement is used to insert new rows into a table.
  - Syntax
    - INSERT INTO table_nameVALUES (value1, value2,....)
  - Ex
    - INSERT INTO Patient(Firstname, Lastname, SSN) Values (“Yung-Ting”, “Chuang”, “123456789”)

Update

- **The Update Statement**
  - The UPDATE statement is used to modify the data in a table.
  - Syntax
    - UPDATE table_name
      SET column_name = new_value
      WHERE column_name = some_value
  - Ex
    - UPDATE Patient SET Firstname="YT" WHERE SSN="123456789"

Delete

- **The Delete Statement**
  - The DELETE statement is used to delete rows in a table.
  - Syntax
    - DELETE FROM table_name
      WHERE column_name = some_value
  - Ex
    - DELETE FROM Patient WHERE SSN='123456789'

Aggregate functions (Optional…FYI)

Function	Description
AVG(column)	Returns the average value of a column
COUNT(column)	Returns the number of rows (without a NULL value) of a column
COUNT(*)	Returns the number of selected rows
COUNT(DISTINCT column)	Returns the number of distinct results
FIRST(column)	Returns the value of the first record in the specified field (not supported in SQLServer2K)
LAST(column)	Returns the value of the last record in the specified field (not supported in SQLServer2K)
MAX(column)	Returns the highest value of a column
MIN(column)	Returns the lowest value of a column
SUM(column)	Returns the total sum of a column

Technical Information

- Our application makes a call to userExists method to authenticate users of the system
  - boolean userExists (String username)
  - We use the following SQL command. If the user exists, returns true, otherwise false
- You can use either *Statement* or *Prepared Statement*.
  - ***But I prefer you to use Prepared Statement instead of Statement***

Java Code

- >> Select Name from Users
  - **Name**
    firat
    moser
- Using *Statement*
  - ResultSet rs = conn.createStatement().executeQuery(
    "Select * from users where name='"+input+"'")
- Using Prepared Statement
  - PreparedStatement ps = conn.prepareStatement(
    "Select * from users where name=?");
    ps.setString(1,**input**);
    ResultSet rs = ps.executeQuery();

Before you start project 4, think how you would structure your tables

- Avoid same information storing in the different tables.
  - Example: Same patient might link to multiple doctors. It will require lots of memory spaces if you have multiple doctors to store the same patient over and over.
  - Solution: Have a patientDB table with all patient information. Every doctor only save patientID (primary ID from the database) in his/her database.

Layout of your tables for project 4…

- **In project4, you will need to create…**
  - **Doctor table** (stores all doctors' information)
  - **Patient table** (stores all patients' information)
  - **DoctorPatient table** *(*relational table, referencing both Patient and Doctor)
  - **Appointment table** (relational table referencing DoctorPatient)
  - **Prescription table** (relational table referencing Appointment)
  - ***Notes table** (relational table referencing Appointment)
- Please refer to my diagram shown on the board

Project IV (Instructions of getting started…)

1. First install MySQL browser
2. Try to login into your database from MySQL browser
   - Server Host: ece155b.ece.ucsb.edu
   - Port: 3306
   - Username: your first name
   - Password: your first name
   - Default schema: your first name

   (Please email me if it reports any error. Then I will check if it's password or database issue…)

Project IV
(Instructions of getting started…)

3. Design your Tables on the paper
   - Think about how you construct your XML files…
4. Open DoctorDB.java from project_template4\src\ece155b\doctor\db\, then change:
   - private String USER = “your_firstname";
   - private String PASSW = “your_firstname";
   - private String DB = “your_firstname";
5. Save file, and then try to execute “ant db”.
   (Remember to change 4. before you call ant db)
   You should now see a table with some entries inserted from your MySQL Browser.

Project IV
(Instructions of getting started…)

6. FIRST get familiar with MySQL statements
   - From your MySQL browser, try to insert/update/delete some entries from there. (This is very important before you start project4!!!)
7. Follow the similar method as addPatient(), you do updatePatient(), and deletePatient().
8. Try to call insertPatient(), updatePatient() and deletePatient() in src\ece155b\doctor\data\Doctor.java.

Project IV (Instructions of getting started…)

9. Open PatientGUI.java and do the similar call as line 41:
   - Patient p = Application.doctor.getPatient(ssn);
   (So instead of calling getPatient(ssn), you will call removePatient(ssn) or updatePatient(ssn) methods)
10. Repeat above steps for your patientDB and pharmacyDB.
    - That means you will have a folder called “db” from both pharmacy and patient folders.
    - You will call these insert, delete, update, list(select) methods for both patientDB and pharmacyDB.

Last Reminder…

- Always use MySQL Query Browser to test your query before you actually insert it to the program. In this way it's easier to debug ☺

Questions?

- Start this project as soon as possible…