Databases

JDBC - Practice n. 4

Homework n. 4

The purpose of this practice is to write the missing parts of a simple Java application that makes use of the Java DataBase Connectivity (JDBC) interface to access a database.

Attention. The code developed during the practice, and eventually completed at home, must be compressed into a zip file and uploaded on the didactic portal (Portale della didattica) as it corresponds to the homework n.4. The deadline for the upload is on the website of the course.

Preliminary steps

Database creation:
You have to create the database executing a script from the commands shell of SQLite.

[TO DO ONLY THE FIRST TIME]

- Download **practiceJDBCSQLite.zip**: http://dbdmg.polito.it/wordpress/teaching/databases/
- Unzip the zip file on your PC
- Download and install the commands shell of SQLite for the own operating system from: http://www.sqlite.org/download.html
- Unzip the file
- From terminal, move to the directory where was unzipped SQLite and run the following command (let assume that the practiceejdbcSQLite.sql file is in the same directory of the SQLite executable):

```
sqlite3 databases.db < practicejdbcSQLite.sql
```

In this way you create in the same directory a database into databases.db file, with table creates and initializes by the practicejdbcSQLite.sql file.

Query of the created database:
You can run SQL queries from the commands shell:

- From terminal, move to the SQLite directory (and where is the database) and run the following command:

```
sqlite3  databases.db
```

- Now, you can run the SQL queries. For example, if you digit on the terminal the following command: `select * from course;` (remember the semi-colon, ";" at the end of the query!!!) you will obtaine the courses list:

```
1|DataBase|Avanzato|100|4
2|Java|Base|50|10
3|DotNet|Intermedio|75|20
4|SistemiOperativi|Base|200|1
```

1. Description of the Computer Science Courses database

The *Computer Science Courses* database contains information about the courses of a small company that manages computer science classes. The database stores information about available courses, clients and enrollment of the clients to courses. The E-R model in Figure 1 represents the conceptual model of this database.

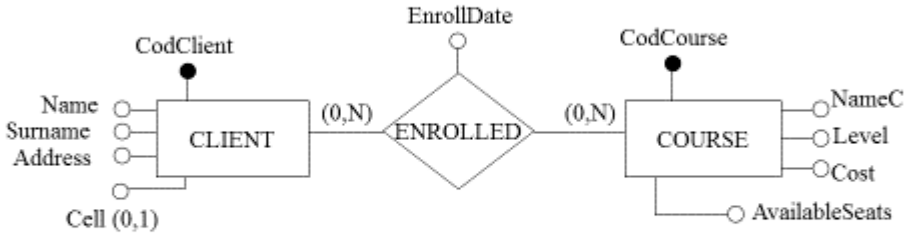

Figure 1. E-R schema of the database

The logical schema of the database is the following (primary keys are underlined); the * symbol means that attribute is optional:

CLIENT (CodClient, Name, Surname, Address, Cell*)

COURSE (CodCourse, NameC, Level, AvailableSeats)

ENROLLED (CodClient, CodCourse, EnrollDate)

Each client is identified by a code and characterized by name, surname, address and mobile number (if available).

The courses are identified by a code and are characterized by a name, the level (an integer between 1 and 4) and the number of the available seats. **Note.** The AvailableSeats attribute contains information about the number of seats still available. Its value is updated after each new registration.

For each client, the courses which is enrolled and the corresponding dates of the enrollment in the different courses are stored. Each client may be enrolled in more than one course. **Attention.** A client can enroll for a course only if there is at least one available seat (AvailableSeats greater than or equal to 1).

The **practicejdbcSQLite.sql** script creates tables above described and load initial data within them.

2. Notes on compilation and execution of Java program.

The class implementing the Oracle JDBC driver to use to connect to the database is **org.sqlite.JDBC**. This class is included in the Java library **sqlite-jdbc-3.7.2.jar** that

is available on the course website. This library should be included among the library in your Eclipse project.

- To create the connection to the database, the *getConnection(String url, String username, String password)* method must be executed using these parameter:
  - url: “"jdbc:sqlite:databases.db"”

As you note in the URL there is the name of the file that contains the database. This file must be placed in the directory root of the Eclipse project.

3. Application to be implemented

The application consists in a simple graphical interface that allows you to manage the following operations:

- Display information of a client and the list of the courses where the he/she is enrolled
- Enrollment of clients in courses present in the database

The purpose of this practice is to understand the operation of Java methods that allows access to a database. For this reason, we provide you as a starting point an Eclipse project that already contains all the necessary class for the implementation of the application, in particular those for the windows management. The Eclipse project is in **practiceJDBCSQLite.zip** file, you can download it from the website course.

You have to implement some methods, currently empty or containing the “static” code, which are used to access the database. You have to implement methods in the class it.polito.db.DB (source file DB.java). All other classes of the project do not require any change.

Before starting to modify the code, you should import the Eclipse project in the working environment (select Eclipse 3.4.1 between the different versions available on LABINF PCs), compile the project and try to run it.

To import the project into Eclipse:

- Open Eclipse
- Select from menu bar File → Import
- Now select “General → Existing Projects into Workspace” and click on “Next”.
- In the following window select as “Select root directory”, the directory where you unzipped the practiceJDBCSQLite.zip file and click on “Finish”.

At the end of the above operations, in Eclipse you get a project called **practiceJDBCSQLite**. This project is your starting point.

You can find the main java class in the it.polito.Main class. Try to run the project (as Java Application) to verify the correctness of the importation. The code creates a window containing two buttons ("Client Information" and "Enrollment"), associated with the two operations mentioned at the beginning of the section. Pressing the first button ("Client Info") it will open a second window that allows you to specify the code of a client (CodClient textbox) and, pressing “Info”, displays client data and the

related courses. The current version of the code displays the same information regardless of the code provided in input because there is not access to the database. Pressing “Enrollment” in the main window, it will open another window that allows enrolling clients for courses. You select the course of interest from a drop-down menu, then you put the code of the client who wants to enroll in the text box, finally pressing the “Enroll” button you sign up the client to the selected course. This part for now is also “static”, i.e. the program will always say that the enrollment was successful even if nothing happened, because the current program cannot access to the database.

You have to implement the following methods of the it.polito.db.DB (source file DB.java) class in order to make sure that the program accesses to the database. For each method there is a brief description of what it has to do.

- *public DB()* – constructor of the it.polito.db.DB class

In the constructor of the DB class you must instantiate / register your JDBC driver used to access the database. For Oracle the JDBC driver is **org.sqlite.JDBC**

- *public boolean OpenConnection()*

This method creates the connection to the database and stores that connection in a variable of the *conn* class

- *public String getDataClient(long cod_client)*

This method receives as an input parameter the code of a client and returns the data of this client as a string. The client information must be obtained by querying in a suitable way the database. The returned string by the method is generated by concatenating the attribute names of the client and the values of those attributes for the client with the same identification code specified in *cod_client*. For example, if the selected client has name “Poul”, surname “White”, address “Street Inesistente 24, Turin” and mobile number “3933570222” then the method returns the string: “Name: Poul\nSurname: White\nAddress: Street Inesistente 24, Turin\nCell: 3933570222”. Look at the code for more detail.

If there is not client with the identification code equal to *cod_client*, the method returns the string “Client does not exist”.

- *public List<String> getCourseClient(long cod_client)*

This method receives as an input parameter code of a client and returns the names of the courses where the client is enrolled using a list of strings. Each string is the name of one of the courses.

- *public List<String> getCodCourses()*

This method returns a list of courses for which there is at least one empty seat (courses in the *Course* table with the *AvailableSeats* attribute greater than or equal to 1). The list is a list of strings. Each string corresponds to one of the courses selected

by the query and contains code and name of the course. Each string has the following format: “cod_course - name_course”. Look at the code for more detail.

- *public boolean addEnroll(long codCourse, long codClient)*

This method receives as input parameters the code of a course and the code of a client and it enrolls the client in the course (insert a new tuple in the database).Enrollment is the entering of the appropriate tuple in the *Enrolled* table. Then, the method has to modify (decreasing it by one) the value of the available seats for the course with the *codCourse* (update *Course.AvailableSeats* for the course *codCourse*). The method returns true if the enrollment was successful, false otherwise.*public void CloseConnection()*

This method closes the connection to the database.