University of Waterloo
School of Computer Science
CS348

Fall 2016   **Assignment 4**   **Due: Nov. 14, 2016**

Consider the following relational schema.

Student(<u>SNO,</u> SNAME)
Course(<u>CNO,</u> CNAME**)**
Prerequisite(<u>CNO, PREREQ</u>)
Professor(<u>EID,</u>PNAME, OFFICE, DEPT)
Enrollment(<u>SNO, CNO, TERM</u>, SECTION, MARK)
Schedule(<u>CNO, TERM, SECTION, DAY,</u> TIME, ROOM)
Class(<u>CNO, TERM, SECTION</u>, INSTRUCTOR)

The relation Professor stores information on professors. EID is the employee number of the professor. PNAME and OFFICE are the name of a professor and his/her office, respectively. DEPT indicates the department with which a professor is affiliated.

The relation Prerequisite stores the information on prerequisites of a course. For example a ("CS335", "CS240") tuple in Prerequisite indicates that CS240 is one of the prerequisites of CS335.

The Enrollment relation records who are the students enrolled in a class and the grade a student obtained for the course.

The Schedule relation stores the time and place a class is held while the Class relation records the instructor of a class.

All other relations should be self-explanatory.

For each relation scheme, the underlined attributes form the primary key.

You may assume the domain of DAY is {M, T, W, R, F}. The domain of TERM consists of strings of length 3, where the first character representing the term while the last two digits indicating the year, e.g. "F99" for Fall term 1999. The domain of MARK is the set of integers from 0 to 100. The domain of SECTION is a single integer from 1 to 9. The TIME attribute is defined on SQL Time domain. The ROOM domain are strings beginning with the building name. The domains of SNO, EID and INSTRUCTOR are integers. The domains of other attributes are character strings.

You are required to test your answers (ISQL statements) with an in-memory dbms called SQLite. **If an ISQL statement is not tested or the statement cannot be compiled successfully, a zero MARK is automatically assigned to the query.** The instructions on how to access SQLite and how to submit your assignment are given at the end of this assignment (**Additional Assignment Information**).

Answer each of the following queries, **without any view**, with an ISQL statement. Briefly explain the strategy behind the formulation of your ISQL statements. State clearly all (if any) reasonable assumptions you made in formulating your answer. In order to increase the readability of your queries use **indentation** where applicable. **You are required to test your answers with the main-memory DBMS SQLite**. The output should have no duplication.

10% (i)   Find rooms in Schedule relation that locate in the building MC and were not booked at 12:30pm in term F11. You may assume that all rooms in Schedule are available in every term. You may use `time` function in SQLITE.

10% (ii)   A course is said to be *advanced* if there exists a prerequisite for the course. Find those advanced courses (CNO) which were taught by a Computer Science professor but some section of a prerequisite was taught by a non-Computer Science instructor. A professor taught a course exactly when the professor taught a class (a section of a course) in some term.

15% (iii)   List department name (DEPT) and its departmental teaching average. The output is sorted on departmental teaching average. Given a department, the department teaching average is the average mark of all classes that were taught by a member with the department. We are only interested in department which has at least one faculty teaching a class in the Enrollment relation.

15% (iv)   For each student enrolled in the class of CS348 in the year 2010, list the student name, the term and he difference of his/her mark from the average mark of the course. Note that the difference could be negative if the student's mark is less than the average. Given a course, the average mark of the course is the average of all marks for students who enrolled in the course in the year 2010. The output should be sorted by SNO.

20% (v)   Find those courses (CNO) in Enrollment such that whenever a class of the course was offered in the year 2009, the class average was above 80%. Note that we are only interested in courses that were offered in the year 2009.

30% (vi)   A student is considered to be *good* if whenever the student took a course with class (section) size greater than 20, the mark obtained by the student is greater than or equal to 85. Moreover, a good student must take at least 3 courses with class size greater than 20. Find students (SNO) and their average mark for those good students. The average mark is computed over all courses a student took. A student is said to take a course if the student enrolled in a section of the course with a non-null mark. We are only interested in those good students who took more than 10 courses (classes) with marks greater than 70.

Additional Assignment Information

How to login to a Linux machine

In order to test your queries you need to login to linux.student.cs.uwaterloo.ca using ssh.

To use ssh from Linux and Mac you can simply type the following command:
ssh uid@linux.student.cs.uwaterloo.ca
in which uid is your WatIAM user id. You will be prompted to enter a password. This is your corresponding WatIAM password.

If you are using your own Windows machine you can download the required software for using ssh from http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe
If you need more information about how to use ssh, please read the instructions at:
http://www.cs.uwaterloo.ca/cscf/howto/ssh/

➢ If you have login problem you can reset your student.cs password using the following link: https://www.student.cs.uwaterloo.ca/password
➢ You **may not be able to** ssh from locations outside campus to linux.student.cs.uwterloo.ca. If you have problems, you should contact cscf for assistance. You are encouraged to use machines in MC labs to connect to the server. Most of these machines already have ssh installed.

How to test your queries

In order to test your queries you need to use **SQLite**. SQLite is one of the most popular open source relational main-memory database management systems. You can run your queries via the Linux server or you can download the interpreter executable to your own environment from www.sqlite.org. **Since not every constructs are supported by SQLite, please refer to the system documentation in www.sqlite.org for more information on the language constructs.** To run the queries in your own environment, you also need to download two more files (*createschema.sql* and *populate.sql*) from the course website. The interpreter executable and the two files should be in the same directory. To know more about SQLite or download the command prompt interpreter executable to your desktop or laptop, please go to www.sqlite.org.

The following assume you access the DBMS via the Linux server. After you logged in successfully into the server, type **sqlite3** and then press Enter. By using this command the command prompt interpreter of SQLite will run and you will see something like the following messages:

```
SQLite version 3.7.2
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite>
```

Here you can type your "SQLite commands" and see the results.

SQLite commands start with a single **dot** while SQL statements don't. To see a list of SQLite commands type
.help
and then press Enter.

SQLite is a main-memory DBMS. Once you exit SQLite, all work done in the session will be lost.
Before starting, we need to create database schema and populate tables with test data.
In order to create the database schema run the following command:
.read /u1/cs348/public/sqlite/createschema.sql
This command reads all SQL statements inside the file *createschema.sql* and executes them one by one. Inside *createschema.sql* there are a number of CREATE TABLE SQL statements.
As a result of this command all required database tables for this assignment will be created.

By running the following command you can see the list of all created tables:
.tables

To populate the created tables with test data run the following command:
.read /u1/cs348/public/sqlite/populate.sql
This command loads all SQL statements inside the file *populate.sql* and executes them. Inside this file there are a bunch of INSERT INTO ISQL statements.

Now, the database is created and populated and ready for trying your answers.
To start, type the following simple ISQL statement and see the results:
select * from course;
Do not forget to put a semicolon at the end of each ISQL statement.
The SQLite interpreter considers semicolon as an ISQL statement separator. If you press enter before adding a semicolon at the end of the statement, your ISQL statement will not be executed. In this case simply type a semicolon and then press the Enter. You can type several ISQL statements and separate them by semicolon. After pressing Enter all of them will be executed one by one.

To exit from SQLite command line interpreter type:
.exit
Another method to exit from SQLite is to press **CTRL+ C**.

You can type your ISQL statements inside a file using a text editor and save the file with an arbitrary name (say test.sql). When you are in SQLite command line interpreter, you can read and execute all ISQL statements inside the file you created by typing the following command:
.read test.sql

You may want to set echo on by issuing *.echo on* and set output to a file by issuing *.output <file>*. In this way, the query and its result will be placed in *<file>*.

By typing **.help** a list of all possible SQLite commands will be appeared.
The only instructions you need to know for this assignment are **.read**, **.tables**, **and .exit**.

If you are interested to learn more about SQLite you can take a look at the following tutorials:
http://zetcode.com/databases/sqlitetutorial/tool/
http://zetcode.com/databases/sqlitetutorial/
http://www.shokhirev.com/nikolai/abc/sql/sql.html

How to create a query file

To create and edit a text file in Linux you can use **pico or vi** editor.

For example for creating 1.sql you need to type:

```
pico 1.sql
```

After typing your answer you need to press **CTRL+X** to save it.

pico asks you the following question:

"save modified buffer (answering "No" will destroy changes)?"

Type "**y**" to answr YES.

Then pico asks you to choose a filename. Default name is 1.sql (the file that you opened/created by pico). You may leave the default name as is and just press the Enter. Your file is created and ready for use.

To test your file type:

```
sqlite3
```

Then create the database schema and populate the tables by test data by typing the following commands:

```
.read /u1/cs348/public/sqlite/createschema.sql
.read /u1/cs348/public/sqlite/populate.sql
```

To direct the output to a text file called *submit*:

```
.echo on
.output submit
```

You can then test your query by typing:

```
.read 1.sql
```

If the results are not as you expected or if you received an error message about your query, you can exit SQLite by typing:

```
.exit
```

and open the file again by typing:

```
pico 1.sql
```

After correcting your SQL statement and saving the file you can test it again.

You need to repeat this process until you make sure that your file is ready for submission.

A more convenient method is to open two different sessions, one for pico editor and one for sqlite3. In pico editor session you can edit your file and in sqlite3 session you can test the results. In this way you don't need to go back and forth between sqlite3 and pico editor in the same session.

Tip: To save your modifications in pico without closing the editor you can press **CTRL+O** and then ENTER.

What to submit in this assignment

You need to submit both a **soft-copy** and a **hard-copy** of your assignment. The soft-copy **is a zip file** containing all your isql queries, naming as x.sql, where x=1,..n. The zip file also contains the test output *submit* file. Softcopy is for verifying that you have in fact tested the queries with SQLite. The TA may wish to test your queries with your soft-copy submission. **Please note that an untested query will automatically receive a zero mark.**

For the hardcopy, it contains the content of each query. Below, as an example, is the **content** of a hypothetical query:

```
--    Question: 6
--    The strategy behind the formulation of my answer: …
--    …
--    …
--    Reasonable assumptions (if any): …
--    …
--    …

SELECT    s.sno, s.sname, e.mark
FROM      student s, enrollment e
WHERE     s.sno = e.sno
          AND e.mark < 50
ORDER BY  e.mark ;
```

<Followed by the output of the query>

For each query, a content is required in the hard-copy. The contents of these queries are ordered according to their question numbers. Since the hard-copy is for the marking purpose, it is important for you to explain clearly the strategy behind the formulation of your queries.

How to submit your assignment

For **softcopy** submission, put it into A4 dropbox under you LEARN account. Follow the instructions given in course outline for softcopy submission. You need to submit a **hard-copy** of your assignment as well. The hardcopy is used by TAs for marking. Drop the hard-copy into boxes on the 4th floor in MC labelled CS348. Please make sure that you place the assignments into the right boxes. Both copies **must be submitted on or before the deadline**.