CS348 - Project 2

Overview

In this project, you'll create a simple Oracle database & perform some common database operations. The schema of the database is almost the same as that you used in the first project.

Information about getting your Oracle account and general initial configuration is available at: http://www.cs.purdue.edu/oracle. The department has already setup Oracle accounts for you. You can get your password by logging into the CS portal (top right corner of http://www.cs.purdue.edu) and clicking on “My Accounts”. As soon as you start working on the project, check that your account exists and you are able to run the *sqlplus* command in a terminal (make sure you follow the setup instructions at http://www.cs.purdue.edu/oracle to get *sqlplus* to work). Keep in mind you will need to use “x@csora” as the username when logging into your Oracle account (after entering the *sqlplus* command), where “x” is your Purdue username. If you have problems with your account, please email oracle@cs.purdue.edu or sciencehelp@cs.purdue.edu as appropriate to get the problem fixed ASAP. If you would like to work on the project remotely, you can *ssh* to any of the lab machines (the lab machines are named sslab***nn***.cs.purdue.edu, e.g. sslab11.cs.purdue.edu).

Step 1: Create the tables

The schema for the database is as follows:

*1. STUDENT(*snum: integer, sname: string, deptid: integer, slevel: string, age: integer)*
*2. CLASS(*cname: string, meets_at: date, room: string, fid: integer)*
*3. ENROLLED(*snum:integer, *cname: string)*
*4. FACULTY(*fid: integer, fname: string, deptid: integer)*
*5. DEPARTMENT (*deptid: integer, dname: string, location:string)*

The fields marked with '*' are primary key.

The meaning of these relations is straightforward:
STUDENT contains one record per student identified by snum;
CLASS contains one record per class uniquely defined by its name; the fid field of the class gives the instructor of the class;
ENROLLED contains a record for each student enrolled in each course;
FACULTY contains one record per faculty member uniquely identified by the fid;
DEPARTMENT contains one record per department identified by deptid.

Create ***all the key and referential integrity constraints necessary to model the application***. Make sure that your field & table names correspond to those listed above. For this project, we’re not asking you to create domain constraints or indices on any of the tables (although you will not lose any points for creating them).

Your task: Create a file called tables.sql, which contains five create ... statements corresponding to the tables listed above.
If you're in the directory containing tables.sql, you can create your database tables as follows:

```
$ sqlplus
...
SQL> @ tables.sql
```

Remember the '@' operator forces SQL to execute commands from a file.

Step 2: Read data files
You are given a sample data file (data.sql) that you can use to populate your database for your own tests.
Your task: Check that all data insertion runs without any problem.
If you're in the directory containing *data.sql*, you can insert the given data as follows:

```
$ sqlplus
...
SQL> @ data.sql
```

Step 3: Query your database

Queries:
Write SQL queries that answer the questions below (one query per question) and run them on the Oracle system. The query answers should be duplicate-free, but you should use **distinct** only when necessary. If you are making any assumptions, state them clearly, and document your queries. We will run your queries on a different dataset from the one provided with the assignment, so be careful not to hardcode values to produce correct answers ?

1. For each department, print the department id, department name and number of faculty affiliated with that department. Print 0 as the number of faculty if a department has no faculty.
2. Print the faculty id, name and department id of the faculty teaching the maximum number of classes
3. Print the name, department id and age of the student enrolled in the maximum number of classes taught by faculty affiliated with the Computer Sciences department (i.e. dname is Computer Sciences)
4. Print the name and department id of the student(s) who take classes in all rooms that a class is taught OR are younger than 20.
5. Print the ids (snum) of the students who have more than 3 (distinct) classmates in total.

6. Print the names of (distinct) faculty who teach 2 or more classes in the same room
7. For each department except Management (i.e. dname = Management), print the department id and the average age of students in that department. If a department has no students, print 0 for the average age.
8. Print the department id, name and age of the youngest student not enrolled in any classes
9. Print the ids (snum) of the students majoring in a department whose faculty do not teach any classes
10. What is the name of the faculty teaching the class with the greatest number of students enrolled? What is the number of students in that class?

Hints:
Query 2: The "ALL" operator will help here.
Query 3: Compare counts from the repetition of the same query.
Query 4: Comparing counts will help in this query too and obviously you will need to union results from two separate queries.
Query 5: Remember the set of classmates of a student does not include the student himself. You also need to aggregate "distinct" classmates over all classes that the student is taking.
Query 6: Remember "2 or more" really means more than 1. You don't need to use COUNT for this query. Of course if you have a way using COUNT, that's acceptable too.
Query 8: You first need to find the students not enrolled in any classes and then the youngest in that set.
Query 9: You first need to identify the list of faculty teaching some class, find the ones not in that list, and then find their departments.
Query 10: In this query, you need to print both the faculty name and the number of students enrolled in the largest class. This is similar to Query 2.

General Hints:
- When you're using aggregate functions like COUNT for a specific list of records, the GROUP BY operator should be used in the query.
- For comparison with a string such as Computer Sciences, you can use the LIKE operator followed by the string in single quotes.
- There might be multiple ways to form a query and all of them will be acceptable as long as they result in the same record list.

Your task: Create a file called queries.sql, which contains the queries listed above, **in the order they are listed (jumbling the order of queries may cost you points)**. Before each query *i*, please put the following comment: *rem Query i*. If you are not able to provide a specific query, just type "rem Query i", where *i is the query number*. So, your file should look something like this:

```
rem Query 1
select ...

rem Query 2
select ....
...
```

Don't forget to add a semicolon at the end of each query in the file, which is what actually runs them.

Step 4: Views
Here, you'll create some simple views.
Create two views (Please name them VIEWA and VIEWB) and print their contents.

A. A view that shows the faculty name followed by the number of classes taught by that faculty, ordered by faculty name. Print 0 if the number of classes is 0.

B. A view that shows the names of people (students and faculty) that are expected to be present in a room each time that a relevant class is taught there. This should be one view with both faculty and student names. A student enrolled in a class is "expected" to be present for each session of the class, and a faculty member teaching a class is "expected" to be present for each session of the class. Hence, the view should contain three fields: name, room, and time.

Your task: Create a file called views.sql, which contains SQL commands (create view FOO as ...) to create the views listed above and the SELECT statements that list all the data of both views, i.e. your file should contain two view creation statements followed by two query statements (all of which are ended with semicolons).

Evaluation:
Your project will mostly be evaluated based on the correctness of your output for the queries and views. Some points will also be allocated for the proper creation of the database tables. Make sure you comply with the database schema provided above when creating the tables and writing the queries. If your table creation fails somehow, we will test your queries on a database populated with the correct tables and the table field names in your queries have to match those given in the assignment to produce correct output.

Submission Instructions:

Please create a README file that contains identifying information. For example:

CS348 - Project 2

Author: John Doe
Login: jdoe
Email: jdoe@cs.purdue.edu

Include here anything you might want us to know when grading your project.

To turn in your project, ssh to ***data.cs.purdue.edu***, create a folder named ***project2*** in your home directory and copy your .sql files and your README.txt to that folder.

After copying your files (tables.sql, queries.sql, views.sql) in the folder project2, execute the following command in your home directory:

turnin -c cs348 -p project2 project2

To verify the contents of your submission, execute the following command right after submission:

turnin -c cs348 -p project2 –v