

CPSC 304 – Introduction to Relational Databases – Summer 2004

Programming Project

Given: Thursday, July 8, 2004

Due: July 21, 2004

Demo: Each group needs to sign up for a 15 minute demo, which will take place in CISR building on Wednesday, July 21, 2004. A sign-up sheet will be made available in class when the time comes.

Groups: For this project you can work in groups of two.

Objective: Introduction to Database Programming and embedded SQL (Pro * C + +, JDBC)

Background: You should go through the appropriate references for SQL * Plus and JDBC, Pro * C + + Embedded SQL from the course homepage as well as the references pointed in class. If you have a clear understanding of the contents of these references, you should do quite well on this assignment.

Start your assignment early: While this assignment is not particularly hard, you are recommended to start working on it immediately. Although we have a limited number of Oracle licenses for CPSC 304 this term, be advised that contention for Oracle resources may be heavy closer to the due date.

Oracle Tips: You are only permitted to have 2 concurrent Oracle connections; So, you cannot have more than two such windows/programs open (this includes SQL * Plus). If you see that the latest update is not included in your last SELECT statement, it may be the case that you have not committed your transaction in the other window.

1 PART I

1.1 Reading Requirements for the assignment

Carefully read the following.

1. Read and try the SQL * Plus tutorial (linked from the course homepage).
2. Read and try the Oracle tutorials for JDBC and Pro * C + +. The course homepage, in addition to tutorials on these topics, also include links to Oracle documentation (also refer to the Oracle Technology Network).
3. Make sure you try out enough examples to get a good working knowledge of the two topics.
4. You have the option of working with either Pro * C + + or JDBC for the embedded SQL part.

1.2 Problem for the first part

Create relational instances for the three relations given below.

1. Sailors(*sid:integer*, *sname:string*, *rating:integer*, *age:real*)
2. Boats(*bid:integer*, *bname:string*, *color:string*)
3. Reserves(*sid:integer*, *bid:integer*, *month:string*)

The instances of the three relations to be created must satisfy the following conditions.

1. The Sailors table must have at least 10 rows. At least two sailors should have the same name.
2. Every sailor must at least be 16 years of age. The maximum age of sailors should not exceed 65.
3. The boats relation must have at least 5 boats. At least two of the boats should have the same boat name.
4. There should be at least two boats with the color red and at least two boats with the color green.
5. The reserves relation should contain at least one reservation for every boat.
6. At least one sailor should have reserved all the boats.
7. There has to be at least one sailor who has not reserved any boat.

In a file, create the DDL (SQL statements) for the tables. Provide appropriate referential integrity constraints. Use files to create and save your input DDL and data to avoid re-typing everything. (Do not share any of your files with others). Run the DDL to create the tables. Populate the tables with the instances given in the tables.

2 PART II

Using your choice of

1. Pro * C + +, Oracle's embedded SQL language for C + +
2. Java/JDBC (independent language that can connect to a DBMS via API)

do the following Oracle related tasks:

1. Write code to allow all boat ids for a sailor to be displayed, given a sailor id (sid). Display the sailor's reservations in the non-decreasing order of boat ids.
2. Write code to DELETE sailor with a given sid from the Sailors table. All reservations of this sailor should automatically be deleted from the Reserves table.
3. Write code to insert a new reservation into the Reserves table. You should check to see whether the sailor and the boat exists. Do error checking with meaningful messages.
4. Use appropriate instances to test the insert, delete and update queries.

3 Notes

Save your SQL so that you can rerun it, if necessary. If you wish to reset your tables easily, you might want to create a set of SQL INSERT and DELETE statements in a file say "reset.sql" which you can execute within SQL*Plus by typing "@reset".

Be sure to have appropriate error checking in your code. A general rule is to test the return code (sqlcode) after every call. Some of this may be automated with error trapping. Provide meaningful messages for important events, like "Boat with boat id does not exist. Reservation denied."

Pro*C++ (and Java/JDBC) is very sensitive to syntax errors. Add functionality in small increments. You may need to comment out large chunks of your code to narrow down the syntax errors. For routine code, it is recommended that you use `//`. By doing so, you can use `/*` and `*/` for debugging purposes.

Pro*C++ users can use the C++ function `setw()` in the library `iomanip.h` to set the width of the field and format the output nicely.

The SQL statement: `select table_name from user_tables;` may be of value to you when you are trying to figure out what tables are currently defined in your Oracle account.