Escuela Politécnica Superior, UAM

Estructura de Datos 2016-2017

Lab 2. Databases and ODBC.

Deadline for this lab: week of November 13.

Each group will turn in the material on the day of class, 2 hours before the beginning of class.

The labs will be turned in using moodle. Delay will be penalized with one point for each calendar day past the deadline.

Useful Advice

1. Reports are important: they are not just so that your lecturer may have some idea of what you have done, it also gives important indications on the kind of reasoning you have followed and why you did things the way you did. This will help us better evaluate your work. Besides, writing a good report is a crucial skill for those who, like you, will work in a technical field: let your boss, your colleagues, and your customers understand your work will be one of the most important aspects of your profession. You will have to write good reports, eventually. You might as well start now.
2. This might seem trivial, but... put the name of ALL members of your group in ALL files! You have no idea in how many different ways is it possible to lose track of who did what.
3. Turn in all documents in pdf format. Turning in files in a editable format is a very bad practice, even if you are using such a common format as MS Word. Cruel and unusual punishment will be applied to those who upload documents in format other than pdf. The only exception is plain ASCII text files: those can be uploaded in text format.

Objectives

These days, people seldom interact directly with a data base. More often than not, they do it through some suitable program that interacts with the user, builds

queries based on the interaction, and uses them to query the data base. This happens, for example, each time we interact with a web page of reasonable size that contains a lot of data subject to frequent updates.

In this lab, we shall experiment with accessing (querying and managing) data bases from C programs. In particular, we shall use the ODBC library (if you are curious, it stands for Object Data Base Connectivity) to create a bunch of programs that will interact with the bookstore database that we created in the first lab.

What must you upload

All your files must be tar-ed or zip-ed in a single file in .tgz or .zip format with name EDAT1718_p1_XXX_AAA_BBB.tgz (or .zip) where XXX is the number of your lab group (if you are reading this in English it will probably be 1251 or 1291), and AAA and BBB are the last names of the members of the group (don't be overly Spanish: one last name is enough). The file must be uploaded using moodle and the ink that will be published in due time. Only one of the members of the group must upload the file. Please avoid double uploads: don't you worry; we are going to grade both of you!

The file must contain three directories called database, queries and odbc, each one including the material relative to one of the three parts of this assignment.

Part I: Fill it up!

An empty database is useful to pass the first assignment but, apart from that, it doesn't help much. So, we are going to put some data into it. For this, we shall face a very common problem, something that almost everybody has to solve before populating an existing database: the schema of the data as it appears in the text files that contain them does not correspond to the tables that we have designed. It will be necessary to "massage" the data to coax them into getting into the tables. The details of how this is done are clearly different for each group, as each group will have its individual design, different from everybody else's. The general principles are, however, pretty much the same for everybody, and we can divide the procedure in five steps:

1. we create a collection of temporary tables, one table for each data file. Each

table will have a number of columns equal to the number of fields in the file. The data type of he columns will generally be "string" (that is, VARCHAR(A LOT)): strings will accept pretty much everything, and it is better to read the data into the database first and worry about conversions later;

2. we use the COPY instruction to insert the data into the temporary tables;

3. we create the tables that compose our relational design;

4. we transform the data: for each table of out relational design, we create a query that, starting with the data in the temporary tables, fills in a table of thh relational design: the results are inserted into the final table using an INSERT INTO instruction;

5. we do a DROP TABLE to get rid of the temporary tables.

What to do

a. Create a database in PosgresSQL called "books" that implements teh relational model of the first assignment. It might be necessary to change slightly your design to adapt it to the data: do whatever modifications are necessary and document them.
b. Load the data into the data base using the procedure specified above. The text files with the data are available in moodle.

NOTE: The database must be called "books". Do not complicate my life, please...

What to turn in

- A file with the backup of the data base; use "pgdump".

- A short report, called "adaptation.pdf" in which you describe and justify the changes that you have made to your relational design in order to adapt it to the data.

- A file "populate.txt" in which you include all the commands that you have executed in order to create and populate the database.

Part II. Queries

What to do

Implement using SQL and PosgresSQL the queries proposed in the first assignment. You can do it using pgadmin, quite useful for debugging your queries.

What to turn in

- A file "sql_queries.txt" with the SQL text of the queries.
- A series of files "result_1.txt", "result_2.txt", etc. containing the results of the executions of each of the queries.

Part III. Wake up and smell the C!

In this part we shall use the C programming language to create a series of programs that shall execute certain operations on the bookstore database. In class, we shall see the basics of ODBC and work out a couple of examples of interaction with data bases. You are, of course, required to do some of the legwork and get documented on your own. The "manuals" sections of the web site http://www.unixodbc.org contains useful documentation. Especially useful:

- "Programming Manual tutorial", a good start to get your first program going and working out the basics;
- "ODBC from C Tutorial Part 1", has good explanations of teh basic functions: useful complement to the tutorial.

You shall create the following programs (note: all programs work from the command line; I don't want to see interactive stuff, menus or anything like that: when you receive a specification it is important that your programs should work exactly as requested):

sale

Thi program allows you to put books on sale. In order to create a sale one shall execute:

sale <discount> <from> <to> <isbn> <isbn> .... <isbn>

where "discount" is an integer between 0 (no discount) and 100 (you get the book for free). The isbn list has arbitrary length, and it represents the books that we put on sale (always check your limit case: does your program crash if I give a list of zero isbns?).

buy

This program performs two functions, depending on the value of the first parameter: I can add a sale for a registered user (receiving in return an invoice number for that sale), or I can delete all the sales associated to a given invoice number. You shall execute:

buy add <screen_name> <isbn> <isbn> .... <isbn>

buy del <invoice_id>

If the parameter add is specified, the program will print on the screen and return as a return value the number of the new invoice relative to the sale (again, careful with the limit cases: what do you do if I specify zero books? Will you create an empty invoice or do nothing? Justify your decision.). The program will also print on the screen the total price to be paid.

Example:

```
$ buy add jack 213456 8976752 2175643
invoice: 1786
total: 56 E
$ _
```

fill_sales

This program fills the "sales" table with data from an input file with the same format as "ventas.txt".

fill_sales <input_file>

Compare the data that you insert in the "sales" table using the method in part 2 with those that you get by filling the same table with the same text file using "fill_sale". Do you get the same result? Yes? No? Why?

What to do

- Implement the program described above using C and ODBC;
- You don't have to check the input data: assume that, as format goes, they are correct (of course you have to check things like the existence of users and so on).
- Do error checking on all the ODBC calls: this is not only a requirement but also something that will save you a lot of trouble.
- For the connection to the "book" data base you shall use the same username and password that you have used to log in in the first assignment.
- Check your code with valgrind.

What to turn in

- Source code and Makefile (one makefile must generate all programs).
- A report discussing your work.