CSC 370 — Database Systems
Summer 2004
Assignment No. 2

Note 1 **This assignment is to be done in teams of two people.**

Note 2 Except as indicated, working with other teams is strictly prohibited.

- Due date: July 5, 2003, at the beginning of the class.
- This assignment is worth 5% of your total course mark.
- Clearly mark your student number on all submissions.

Objectives

After completing this assignment, you will have experience in:

- Writing programs that interact with the database using embedded C, Java JDBC and Perl DBI;
- Using the system tables.

Preliminaries

System Tables

System tables are relations that store information used by the DBMS. The idea behind system tables is that everything that the DBMS knows about the database, its users, and its environment should be stored in tables, so this information can be manipulated using SQL. Unfortunately system tables are not standardized, and each DBMS uses different schemas for them and therefore stores similar information in very different ways.

In particular, a great deal of data is stored in *postgresql* system tables. The names of these tables begin with `pg_`. The tables contain information about data types, functions, operators, databases, users, and groups. The following table shows some of these tables:

Table	Contents
pg_database	databases
pg_class	tables
pg_attribute	columns
pg_users	users

Try doing a `select *` from each of these tables. You will find a detail description of each field in the Postgresql Administration Guide.

EXIF data in digital photos

Most digital cameras are able to generate JPEG files. These JPEG files contain information about the photo, such as the date/time it was taken, the aperture, exposure time, lens focal length, etc. It also contains information about the camera and its manufacturer. This information is stored in a format called EXIF.

Your task, should you choose to accept it

The programs you will write use a table. The catch is that you don't know the names of its columns, but you know the following information about each of the columns of the table.

Column number	data type	comment
1	timestamp	Date the photo is taken
2	char(40)	Camera model
3	char(40)	Camera maker
4	integer	Horizontal size in pixels
5	integer	Vertical size in pixels
6	float	Aperture
7	float	Exposure Time
8	integer	ISO speed

Table 1: Columns in photos table

The only field that is required is the date the photo is taken (which is the primary key of the relation). All the others can be NULL. Your programs will have to find out the names of the columns from the system tables. Your programs cannot assume anything.

This assignment is composed of 3 parts.

Part 1

```
loadphotos(1)

NAME

        loadphotos - load selected EXIF data from JPEG files into a
        given table.

SYNOPSIS

        loadphotos  HOST DB USERNAME TABLE [<jpegfile1> ...]

DESCRIPTION

        loadphotos connects to the postgresql database DB running in
        the computer HOST using as user USERNAME.

        If only these three parameters are specified, the program
        should print the names of the attributes of the table TABLE,
        one per line.

        <attributeName1>
        <attributeName2>
        ...
        <attributeNameN>


        where <N> is the number of attributes. The attributes should
        be printed in the same order as they were defined (hint, use
        field attnum of one of the system tables).

        When the optional file names are provided, it will inspect one
        file at a time, and for each file it will load the
        corresponding data into the photographs table (which will have the
        schema described previously).

        If the file is a valid JPEG photo, and it contains at least
        the date it was taken, then a new record is added to the
        database. This date is the primary key of the table, and
        therefore, you will not be able to insert the same file
        information twice. If one of the other EXIF attribute is not found
        in the JPEG then the attribute is filled with a NULL.

        The program should output one line per file, with the
```

```
        following legend:

        <filename>: <message>

        where the <message> can be


        error                the file does not exist, or it does not
                             contain a valid EXIF block or
                             it does not contain the
                             'date taken' field or an error occurred.

        ok                   the file's EXIF data was properly loaded
                             into the table

        already inserted     this file's data is already inserted

        For example, for 3 files, this is a potential output:

        <jpegfile1>: ok
        <jpegfile2>: already inserted
        <jpegfile3>: error


        At the end of its execution, it should print a message with
        the total number of photos loaded, for example:

        2 photo(s) loaded.


NOTES:

        The program reads the password needed to connect to the
        database from the standard input. You should prompt for the
        password, and then read one line from standard input and use
        this value as the password.

        Your program will be run as follows:

        loadphotos host user table file1 file2 ... < fileContainingPassword

        During a successful run, the program exits with errorcode 0.
        Any critical error is handled by printing some error message
        (such as "An error has occurred") to standard error and then
        exiting with error code 1.

EXAMPLES:

        Display the attributes of the table RIP in the
        databases csc370, in the host 'server' using username 'dmg.

        $ loadphotos server csc370 dmg RIP
        password:
        fieldname1
        fieldname2
        fieldname3


        Load all the photos in the current directory into the
        databases myphotos, table photos available in the host
        'localhost' using username 'test'.

        $ loadphotos  localhost myphotos test photos *.jpg  < passwd
        password:
        test1.jpg: ok
        test2.jpg: ok
        test3.jpg: ok
        who.jpg: error
        zapphoto.jpg: ok
        5 photo(s) loaded.



AUTHOR
       Written by you.

REPORTING BUGS
       Report bugs to <you@uvic.ca>.

COPYRIGHT
       Copyright  2002 Your Name
```

Write a programs that implements `loadphotos` in embedded SQL.

Part 2

```
reportphotos(1)

NAME

        reportphotos - report data from the photos table.

SYNOPSIS

        reportphotos  [options] HOST DB USERNAME TABLE

DESCRIPTION

        reportphotos connects to the postgresql database DB running in
        the computer HOST using as user USERNAME.

        The program should list all the photographs loaded in table TABLE
        as follows:

        Total: <n> photo(s)
        value1,value2,value3,value4,value5...,value8
        value1,value2,value3,value4,value5...,value8
        value1,value2,value3,value4,value5...,value8

        The report should be ordered by date (field 1 of the table).
        NULL values should be left empty.

        Strings should be printed without trailing spaces. The
        aperture should be printed using a "%3.1f" format, and the
        exposure time using "%7.4f" format , and integers should be
        printed without any spaces. Dates should be printed in the
        following format: yyyy/mm/dd HH:MM:SS, without trailing
        spaces.

OPTIONS:

        -sort <fieldnumber>   Order the report according to the field
                              number <fieldnumber>, and use the date taken
                              as the secundary ordering field.

        -group <fieldnumber>  Instead of the usual report, count the number
                              of photos that have the same field
                              value. For example, -group 2 will
                              result in a report of camera models,
                              followed by the number of photos of each
                              model, for example:

                              Total: 5 photo(s)
                              Canon EOS D60,3
                              Canon PowerShot S300,2

                              This report should be ordered by the
                              field to group by and it will always
                              contain only 2 columns.

NOTES:

        The program reads the password needed to connect to the
        database from the standard input. You should prompt for the
        password, and then read one line from standard input and use
        this value as the password.

        Your program will be run as follows:

        reportphotos host ... < fileContainingPassword

        During a successful run, the program exits with errorcode 0.
        Any critical error is handled by printing some error message
        (such as "An error has occurred") to standard error and then
        exiting with error code 1.

EXAMPLES:

        Load all the photos in the current directory into the
        databases myphotos, table photos available in the host
        'localhost' using username 'test'.

        $ reportphotos.pl  localhost myphotos test photos  < passwd
        password:
        Total: 5 photo(s)
        2002-11-20 11:23:35,CanonCanon PowerShot S300,1024,768,2.7, 0.0025,0
        2002-12-28 10:27:23,CanonCanon PowerShot S300,1024,768,2.7, 0.0100,0
        2004-06-04 17:01:03,CanonCanon EOS D60,160,120,2.0, 0.0100,800
        2004-06-06 11:18:09,CanonCanon EOS D60,160,120,4.5, 0.0016,100
        2004-06-06 11:43:00,CanonCanon EOS D60,160,120,4.0, 0.0100,100

        $ java -sort 2 reportphotos  localhost myphotos test photos  < passwd
```

```
        password:
        Total: 5 photo(s)
        2004-06-04 17:01:03,CanonCanon EOS D60,160,120,2.0, 0.0100,800
        2004-06-06 11:18:09,CanonCanon EOS D60,160,120,4.5, 0.0016,100
        2004-06-06 11:43:00,CanonCanon EOS D60,160,120,4.0, 0.0100,100
        2002-11-20 11:23:35,CanonCanon PowerShot S300,1024,768,2.7, 0.0025,0
        2002-12-28 10:27:23,CanonCanon PowerShot S300,1024,768,2.7, 0.0100,0



AUTHOR
       Written by you.

REPORTING BUGS
       Report bugs to <you@uvic.ca>.

COPYRIGHT
       Copyright  2004 Your Name
```

Write a programs that implements `reportphotos` in java and in perl. Both programs should be functionally equivalent.

Part 3

Write a programs that implements `aggregatephotos` in perl.

Implementation Notes

- Look at the examples in `~dmgerman/public/csc370/assign2`. There you will find:
  - an example of an embedded C program;
  - one of a Java JDBC program;
  - one of a Perl DBI;
  - an example of a program that reads EXIF data;
  - some sample images.
- Pay particular attention to the Makefiles and/or README files. Each program requires specific libraries to run.
- Make sure the format of your output is identical to the one specified. Otherwise you will lose marks.

What to submit

- Choose one of the CVS modules (each member of the team has one) to submit the assignment.
- Create a directory `assign2` to this module.
- Add to this directory any files you need, including a `Makefile`. After running `make`, there should be the following 3 executable files:
  - `loadphotos` (executable for embedded C),
  - `reportphotos.javac` (for Java JDBC),
  - `reportphotos.pl` (for Perl DBI),

- Print the results of the `cvs status` command by running it in the `assign2`.
- At the start of the class on the deadline, **hand in your solution to this assignment (including a hardcopy of the source code of each of your programs)**. Only one copy per team is required.
- Clearly mark the name of **your team**, the name of the CVS repository you used, and the name of each member of the team in your submission.

This assignment is not trivial. The amount of programming is small, but it will be time consuming. You will have a lot of questions about how to complete this assignment. You will need to use the Postgresql documentation a lot. I suggest you use google to find answers to your questions. Part of the assignment's objective is that you find the answers to those questions.

Start early. I will answer any questions about the project in the bulletin board of the course.