Lab 2: Database Creation

Due date: Thursday, April 17, **midnight**

Lab Assignment

Assignment Preparation

This is an individual lab. Each student has to complete all work required in the lab, and submit all required materials **exactly as specified** in this assignment.

For this assignment, you will be using your Oracle account. The accounts have been assigned in class on Thursday, April 10. If you have not recieved your Oracle account information, please contact the instructor.

You will be using Oracle's SQL*plus client and the Oracle DBMS server `ora10g`. Please refer to your SQL*plus handout for more information about starting SQL*plus and working with it.

For the purposes of this, and future assignments, an SQL script is a text file which contains a sequence of SQL statements and SQL*plus commands. Typically, these files should recieve a .sql extension. You will be required to prepare and submit a number of SQL scripts for this lab.

The Datasets

Starting with this lab, and for most of the remaining labs in the course you will be working with several datasets created specifically for this course. These datasets are not large, but they are sufficiently diverse in content, scope and structure.

All datasets are available from the course web page:

http://www.csc.calpoly.edu/∼dekhtyar/365-Fall2007

Each dataset comes with a README file, which contains the exact specifications of all data files included in the dataset, and briefly explains the meaning of the dataset. Before starting your work on a dataset, **please study carefully the README file and make sure you understand the structure of the dataset!** All data files in each dataset are stored in the CSV (comma-separated values) format. All text values are enclosed in single quotes. All dates are given in default Oracle's format. A .zip file for each dataset is avilable.

Brief d escriptions of each of the course datasets is given below.

CSU dataset. Type: multidimensional statistical data. This dataset contains various statistics about the California State University system. Historic information, such as annual enrollments and graduations is included, as well as information about enrollments by discipline at all campuses in a single year, and information about faculty lines and campus fees.

CARS dataset. Type: normalized[1] dataset. This database stores information about the properties (such as number of cylinders, milage per gallon, engine displacement, etc) for over 400 domestic and import cars produced between 1970 and 1982. In formation is split into several files: starting with lists of continents and countries, going onto the lists of automakers and the models and makes they were producing.

BAKERY dataset. Type: OLTP (on-line transaction processing) dataset. This dataset records information about one month of sales from a small bakery to a list of its dedicated customers. The datset captures the notions of on-line transaction (a single purchase) and market baskets (each purchase may contain more than one item).

MARATHON dataset. Type: universal table. This dataset consists of a single CSV file documenting the performance of participants of a half-marathon race. The performance is tracked for the entire race, as well as within each gender/age category.

STUDENTS dataset. Type: simple normalized. This is a variation of the you have encountered (different names, no bus routes). The dataset consists of a list of students assigned to grades and classrooms, and a separate list of teachers assigned to classroms. This is the simplest dataset and can be used as "staging grounds" for most of the activities in this and future labs.

AIRLINES dataset. Type: graph. This dataset will be of great use when we study PL/SQL in the second half of the course. The dataset stores information about a number of airlines, and the flights these airlines

[1] In CS 366 you will get an opportunity to study the formal meaning of the term "normalized" when applied to databases. An informal explanation: a "normalized" database is a database, where information is split into a large number of tables, reducing redundancy in data.

have between 100 different airports. It can be viewed as a multicolored graph with 100 nodes representing airports, edges representing direct flights and edge colors representing the airlines running the flights.

Your Lab 2 assignment should be performed for each of the datasets in the list above.

The Task

Your assignment consists of three parts. In the first part of the assignment, you will design a relational database for each of the datasets, and instantiate it with the data provided to you.

In the second part of the assignment, you will perform some schema changes in some of the datasets, as well as some content alterations.

In the third part of the assignment, you will demonstrate your proficiency with the use of SQL*plus commands for formatting of the results of queries.

Part 1: Database design and creation.

This part of the assignemnt refers to all datasets. That is, all tasks outlined in this section must be performed on all datasets.

You need to do the following:

1. For each dataset, create a relational database to store its data. The following rules must be followed:
   (a) The tables of the database must match the files of the dataset one for one.
   (b) You are allowed to choose any (hopefully meaningful and non-offensive) names for all relational tables and columns in them.
   (c) You must properly detect and declare all constraints, including primary key, candidate key (SQL's UNIQUE), and referential integrity/foreign key constraints.
2. Write and test an SQL script for creation of each database (one script per database).
3. Write and test an SQL script for deleting all tables from each database (one script per database).
4. Prepare and test SQL scripts for populating each database with the data avilable from the .csv files. (Hint: use any available programming/scripting language, to convert the .csv file into a list of SQL's statements for adding records to the database. There is a relatively simple way to solve this problem forever with a single Perl script, for example). You should have one SQL script per database table.

5. **Note:** While your table/column names can be any of your choice, the names of all scripts you must prepare and submit are defined in this assignment. Please refer to the **Submission Instructions** section for the specs for proper script naming.

6. Write and test the scripts for checking the contents of each database. A simple statement to check the contents of a database table is

```
SELECT * FROM <Table>;
```

   This statement will result in all records in a table printed in their entirety.

Part 2: Database schema/data modifications

The assignments in this part are specific to individual databases you create in Part 1 of the lab. Please execute them only on the specified datasets.

1. [**STUDENTS dataset.**] Extend the database structure to include the information about the bus route for each student. Update the database as follows:

   - All 4th graders are assigned bus route 51.
   - All students in classroom 109 are assigned bus route 52.
   - All 5th graders who are not in classroom 109 are walkers (bust route 0).
   - All other students are assigned bus route 53.

   You can use any way of setting the new values (and any number of commands) which results in correct assignments being made. Write an SQL script documenting the entire procedure, finish it with a `SELECT * FROM <TeachersTable>;` SQL statement.

2. [**BAKERY dataset**]. Suppose the bakery is located in Hayti, MO in the quad-sate area (MO-TN-KY-AR). The customers come from locales in the 55-65 mile radius from these four states (study an online map of the area). Assign to each customer a location (town, state) of your choice. Ensure that at least one customer exists from each of the four states. Write an SQL script changing the database schema and updating the database appropriately.

3. [**CARS dataset**]. Write SQL command(s) converting the weight of cars from lbs into kilograms. Write SQL commands that clean the cars dataset, leaving in the `cars-data` tables only information about cars produced before 1980. Prepare an SQL script with appropriate statements, end it with `SELECT * FROM <Cars-dataTable>;`.

4. [**CSU dataset**]. Univeristies comprising what is now known as the CSU system have been brought together in 1960. Since then, CSU

system has been extended to include new campuses. The Campuses table contains information about the dates different universities were founded. Based on this data and information supplied below, you need to write SQL statements to (a) add a new column to the Campuses table the year the campus joined the CSU system and (b) populate the new field. For all campuses except for the Maritime Academy, if the campus was founded before 1960, it joined the CSU system in 1960. The Maritime Academy joined the CSU system in 1995. All campuses founded after 1960 joined the CSU system on the year they were found. Create an SQL script containing approrpiate statements, and end it with `SELECT * FROM <CampusesTable>;`.

Note: in the SQL queries above (as well as below) a table name in angle brackets (e.g., `<Cars-dataTable>`) refers to the name you assign to the relational table storing the data of the specified `.csv` file (in our example - `cars-data.csv`). You will substute the actual table name for this placeholder. In the queries below, the attribute names in angle brackets are understood in a similar manner - they need to be replaced by the actual column names from your relational schema.

Part 3: SQL*plus mastery

Each task in this part relates to a specific dataset, just as in Part 2 of this assignment, and should be performed only for that dataset.

This assignment is independent from the assignments in Part 2 of the lab. That is, I will be testing your scripts for Part 3 on the databases constructed by the scripts from Part 1.

1. **[STUDENTS dataset].** Consider the following SQL query, which outputs the names of each student and his/her teacher:

   ```
   SELECT S.<FirstName>, S.<LastName>, T.<FirstName>, T.<LastName>
   FROM <listTable> S, <teachersTable> T
   WHERE S.<Classroom> = T.<Classroom>
   ORDER BY T.<LastName>;
   ```

   Substitute appropriate column names for their placeholders in this query. Before proceeding with the tasks below, make sure this query runs and returns the correct result.

   Create an SQL script, containing SQL*plus commands which format the result of this query as follows:

   - There are no pagebreaks in the provided output.
   - The name of the teacher appears only once, and there is an empty line between the records for students with different teachers.

- Student name column headers are "Student: First name" and "Student: Last name". Teachers' first name column header is "Teacher", teachers' last name column header is empty.
- The columns are separated with double colon: ("::").
- Each retrieved record fits in a single line.

2. **[BAKERY dataset].** Consider the following SQL statement:

```
SELECT R.<Date>, R.<RecieptNumber>, C.<FirstName>, C.<LastName>
FROM <RecieptsTable> R, <CustomersTable> C
WHERE R.<CustomerId> = C.<Id>
ORDER BY C.<LastName>, R.<Date>;
```

This statement outputs the list of reciepts (sales) with the names of the customers (rather than their Ids) sorted by customer's last name, and with a secondary sort by the purchase date.

Substitute appropriate column names for their placeholders in this query. Before proceeding with the tasks below, make sure this query runs and returns the correct result.

Create an SQL script containing SQL*plus commands which format the result of this query as follows.

- The result is broken into 2 pages, each page contains 100 records (there are 200 reciepts in the database). (make sure you set the appropriate parameters correctly).
- Each customer name appears exactly once (duplicate occurrences of the name are left blank).
- The column separator for the output is "**"; the column headers are separated from the data by a line of "-" characters.
- Each record in the output fits into a single line of the report;
- The column headers are as follows: "Sale Date", "Reciept#", "Customer", "".

Extra Credit

For extra credit do the following:

- Find a data collection on the web. The data you will be using must come from a source that either explicitly allows its use for non-commercial purposes, or the nature of the data is such that such use is permissible. Examples of such sources include (but are not limited to) government statistical data (you, the taxtpayer, have paid for this data to be collected), any public domain data collections, any scientific data that was published, or any data you assemble yourself from various sources.

- Create a database schema, extract the data that matches the schema, and insert the data into the database.
- Submit SQL scripts for database creation, removal of all tables of the database and database population (one per table). Additionally, submit a `README` file describing the nature of the database and the meanings (if necessary) of the table columns.

Submission Instructions

General Instructions

Please, follow these instructions exactly. Up to 10% of the Lab 2 grade will be assigned for conformance to the assignment specifications, **including the subimssion instructions**.

Please, **name your files exactly as requested** (including capitalization), and submit all files **in a single archive**. Correct submission simplifies grading, and ensures its correctness.

From now on, please include your name and Cal Poly email address in all files you are submitting. If you are submitting code/scripts, include, at the beginning of the file a few comment lines with this information. Files that cannot be authentificated by observing their content will result in penalties assessed for your work.

Specific Instructions

You must submit all your files in a single archive. Accepted formats are `gzipped tar` (`.tar.gz`) or `zip` (`.zip`). The file you are submitting must be named `lab2-ilastname.ext`, where *i* stands for the initial of your first name, and *lastname* is your last name. E.g., if I were submitting this file, the name would be `lab2-adekhtyar.zip` or `lab2-adekhtyar.tar.gz`.

Inside it, the archive shall contain six directories named `AIRLINES, CSU, CARS, BAKERY, MARATHON` and `STUDENTS` after the datset names. In addition, the root of the directory must contain a `README` file, which should, at a minimum, contain your name, Cal Poly email, any specific comments concerning your submission.

Each directory shall contain all SQL scripts built by you for the specific dataset in response to all parts of the lab. The scripts shall be named as follows:

- database creation scripts. The filename shall be `<dataset>-setup.sql`. Here, `<dataset>` is the name of the directory in ALL CAPS. E.g., for the CARS dataset, the filename will be `CARS-setup.sql`.
- table deletion scripts. The filename shall be `<dataset>-cleanup.sql`.

- table population scripts. The filename shall be `<dataset>-build-<table>.sql`. Here `<table>` is the name of the `.csv` file (not the name of the table that you are building). E.g., for the table populating the list of car makers in the CARS database, the filename will be `CARS-build-car-makers.sql`.
- testing scripts. These are scripts described in item 6 of Part 1 assignment. The filename shall be `<dataset>-test.sql`.
- modification scripts. Each modification task in Part 2 of the lab asks you to create a single script. The filename shall be `<dataset>-modify.sql`.
- SQL*plus formatting scripts. Use the filenames `STUDENTS-formatted.sql` and `BAKERY-formatted.sql` respectively.

Extra Credit submission. If you have completed the new dataset extra credit assignment, include a fifth directory with your submission. The name of the directory should be the (preferably relatively short) name of your dataset in ALL CAPS. The naming of the scripts for your extra credit dataset should follow the same rules as the naming of the scripts for the regular assignment.

Testing

Your submission will be tested by running all scripts you supply and checking the produced output for correctness. I may also use some extra scripts to verify the correctness of the databases you have constructed.

I will use one of my Oracle accounts to test your scripts - not your personal Oracle account.

If you are aware of any bugs, or incorrect behavior of your SQL scripts, I strongly suggest that you mention it in the README file.