Lab 2: Database Creation

Due date: Tuesday, October 2, **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 information about your accounts was given to you during the Thursday, September 27, class.

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-Fall2012

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 descriptions of each of the course datasets are 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. Information 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 dataset you have encountered (different names, no GPA) in Lab 1. 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.

WINE dataset. Type: normalized (somewhat). The dataset lists ratings of a variety of single-grape California wines of different vintages as given by the Wine Spectator magazine. The dataset consists of a list of wines complete with their ratings on a 100 point scale, their reported sales prices and the production volumes. Two additional lists: appelation/American Viticultural Areas (AVAs) and grape varieties are available as well.

INN dataset. Type: simple OLTP. The dataset contains information about one year's (2010) worth of completed hotel reservations for a small Bed & Breakfast inn. The inn contains 10 uniquely named rooms - each with its own set of features. The dataset lists all reservations that *commenced* 2010[2] and specifies for each room reservation the checkin and checkout dates, the name of the person making the reservation and the number of adults and kids staying in the room.

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

The Task

For this lab you will design a relational database for each of the datasets, and instantiate it with the data provided to you.

Database design and creation.

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).

[2] Some reservations made at the end of the year end in 2011.

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 statements for adding records to the database. There is a relatively simple way to solve this problem (almost) 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. **Be aware that incorrect names for submitted files, EVEN incorrect capitalization will lead to significant deductions in your lab score!**

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. The query

   ```
   SELECT COUNT(*) FROM <Table>;
   ```

   will output the number of tuples in the table.

WINE dataset: Special Notes

To make your life a bit more interesting, when creating the database for the `WINE` dataset, please, take into account the following information:

1. The list of appelations/AVAs contained n the `appelations.csv` file, lists a state for each appelation/AVA entry[3]. Similarly, there is a "State" column in the file `wine.csv`, describing the state of the origin of the wine. We note, that `wine.csv` does record the appelation/AVA for each wine (the "Appelation" column), and that the state of origin of the wine is essentially the state of the wine's appelation. Therefore, there is a duplication of this information in the CSV files in the dataset.

2. The last column of the dataset, "Drink" is designed to contain the advice of the wine raters on when the wine is best consumed. Most of the entries contain a text value `'now'`. Some entires contain a number representing the year when we wine will start being at its best.

For your `WINE` database you should do the following:

[3] In the current dataset all appelations/AVAs are from California, but state information still shall be recorded for each appelation/AVA.

- Include state information as part of the appelation/AVA information.
- DO NOT include state information with the wine score (including appelation information will suffice).
- DO NOT include the drink advice column with the wine score information.

Note 1: There is a number of other interesting features in this dataset regarding the primary/foreign key identification. Be careful!

Note 2: There is some missing data from this dataset - some wines do not have production volume information. You have to deal with this properly.

Note 3: In winemaking, an "appelation" is essentially a region from which the grapes used in making the wine originate. The most broad appelation in our dataset is the entire state of California. According to US law, county names are legal/valid descriptions of grape origins. US recognizes other regions as legal/valid specifications of grape origins. Such regions are called AVAs, a.k.a., American Viticultural Areas. In California, AVAs can cross county borders and also be embedded into one another. For example, "Russian River Valley" AVA is located inside "Sonoma Valley" AVA, while "Lodi" AVA spans multiple counties. For simplicity, in the latter cases, `appelations.csv` file lists the main county of an AVA. Some AVAs are larger than a county. E.g., "Central Coast" and "North Coast" AVAs span multiple counties. For such AVAs, `appelations.csv` lists the value `'N/A'` in the "County" column.

File `appelations.csv` lists all grape origin specifications that can appear on wine labels. The last column specifies whether the origin specification is an AVA or not.

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 25% 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.

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.ext`, i.e., either `lab2.tar.gz` or `lab2.zip`

Inside it, the archive shall contain eight directories named `AIRLINES, CSU, CARS, BAKERY, MARATHON, STUDENTS, WINE` and `INN` (same as the dataset names). In addition, the root of the directory must contain a `README` file, which should, at a minimum, contain your name, Cal Poly email, and any specific comments concerning your submission.

If you are submitting extra credit assignments, put the directory/directories for your dataset(s) in ALLCAPS here as well. Follow the same naming conventions for your extra credit SQL scripts as proscribed below.

Each directory shall contain all SQL scripts built by you for the specific dataset. 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`. E.g., for the CARS dataset, the filename will be `CARS-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`. Use **the same capitalization** as the filename in the dataset: e.g., `CSU-build-Campuses.sql` and `INN-build-Rooms.sql`, but `CARS-build-cars-data.csv` and `BAKERY-build-items.sql`.

- testing scripts. These are scripts described in item 6 of Part 1 assignment. The filename shall be `<dataset>-test.sql`. E.g., `CARS-test.sql`

Where to submit

Once you created your submission archive, submit it using the following `handin` command:

```
handin dekhtyar lab02 <file>
```

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.