100 Points

Due October 8th, 5pm.
Submit electronically. Turn-in written work for Problem 1.
Turn-in a single python source file for Problem 2.

1. (55 pts) Schema Design

Imagine that you are establishing a database company, ArtBase, that builds a product for art galleries. The core of this product is a database with a schema that captures all the information that galleries need to maintain. Galleries keep information about artists, their names (which are unique), birthplaces, age, and style of art. For each piece of artwork, the artist, the year it was made, its unique title, its type of art (e.g., painting, lithograph, sculpture, photograph), and its price must be stored. Pieces of artwork are also classified into groups of various kinds, for example, portraits, still lifes, works by Picasso, or works of the 19th century; a given piece may belong to more than one group. Each group is identified by a name (like those just given) that describes the group. Finally, galleries keep information about customers. For each customer, galleries keep that persons unique name, address, total amount of dollars spent in the gallery, and the artists and groups of art that the customer tends to like.

(a) (25 pts) ER Diagram

Draw an ER diagram for the database. Indicate one-to-many or one-to-one relationships with arrows where appropriate.

(b) (30 pts) ER Diagram to Relations

Write SQL statements to create the corresponding relations to the ER diagram you designed above. You should write SQL CREATE TABLE commands and specify both primary key, unique, and foreign key constraints.

2. (45 pts) Implementing SQL operations in Python

As an introduction to Python, you will implement simplified versions of the SELECT, FROM, and WHERE operations in Python. Tuples will be represented as dictionaries, which contain key-value pairs. The keys can be considered the schema of the tuple. Tables will be represented as a list of dictionaries. For example, this declares a table R with three tuples, each containing three integers.

R1 = [ {'a':1,'b':2,'c':3}, {'a':1,'b':2,'c':2}, {'a':2,'b':1,'c':3} ]

You will implement the following functions:

- sql_from( table_list )
This function takes a list of tables (as above, each table is a list of dictionaries). You can assume the input list consists of exactly two elements. The function returns a table representing the cross-product of the inputs, as the FROM clause would in SQL.

• sql_where( table, filter_list )

This function takes as input a table and a list of filter conditions. A filter condition is a dictionary, containing keys which should be present in the keys of the tuples in the table. The value associated with a key in the filter list is a constant that column will be compared with. Filter conditions are assumed to be conjunctively combined. So if filter_list was {'a':1, 'd':2} this would return only tuples whose column 'a' has value 1 and whose column is 'd' has value 2. The function returns a list representing a table, in which tuples have been filtered out according to the conditions.

• sql_select( table, position_list)

This function takes a table and a position list. The position list contains the keys of the columns that will be retained in the output. The function should return a table in which the columns corresponding to positions not mentioned are removed.

Starter code and sample output is available on the edlab machines at:
/courses/cs400/cs445/cs445.f2010/homework4/