CS 145 PS2

October 13, 2015

Instructions / Notes:

- Using the IPython version of this problem set is **strongly recommended**, however you can use only this PDF to do the assignment, or replicate the functionality of the IPython version by using this PDF + your own SQLite interface
- Note that the problems reference tables in the SQLite database (in the PS1.db file) however solution queries must be for any general table of the specified format, and so use of the actual database provided is *not necessary*
- See Piazza for submission instructions
- Have fun!

1 Problem 1

20 points total. For each part of this problem you will need to construct a *single* SQL query which will check whether a certain condition holds on a specific instance of a relation, in the following way: **your query should return an empty result if and only if the condition holds on the instance.** (If the condition *doesn't hold*, your query should return something non-empty, but it doesn't matter what this is).

Note our language here: the conditions that we specify cannot be proved to hold **in general** without knowing the externally-defined functional dependencies; so what we mean is, *check whether they **could** hold in general for the relation, given a specific set of tuples.*

You may assume that there will be no NULL values in the tables, **and you may assume that the relations are *sets* rather than multisets**, but otherwise your query should work for general instances. We define the schemas of the tables used below for convenience, but in this problem you will need to construct your own test tables if you wish to use them to check your answers!

1.1 Part (a)

5 points $\{A, B\}$ is a superkey for a relation $T(A, B, C, D)$.

1.2 Part (b)

5 points The individual attributes of a relation $T(A, B, C, D)$ are each keys.

1.3 Part (c)

5 points $\{A\} \twoheadrightarrow \{B, D\}$ for a relation $T(A, B, C, D)$.

1.4 Part (d)

A *tuple-generating dependency (TGD)* is a dependency between two relations A, B of the form:

$$\forall x_1, ..., x_n,\ A(x_1, ..., x_n) \implies \exists z_1, ..., z_k, B(y_1, ..., y_n, z_1, ..., z_k) \text{ s.t. } y_1 = x_1, ..., y_n = x_n$$

In other words, for every distinct tuple in A, there must exist a corresponding tuple in B, which has the same values of certain specified attributes. Consider two tables Dog(Name, Breed, Owner) and Person(Name, SSN, Hometown); check for the TGD:

$\forall$ Name, Breed, Owner, Dog(Name, Breed, Owner) $\implies$ $\exists$ SSN, Hometown, Person(Name, SSN, Hometown) s.t. Dog.Owner = Person.Name

2 Problem 2

20 points total.

2.1 Part (a)

10 points. Consider a relation $R(A, B, C, D, E)$. Provide *two different sets* of functional dependencies, F_1 and F_2, such that each one results in R having the **largest number of distinct keys** R could possibly have. Store your lists of FDs as python lists having elements that are *pairs of sets*; for example to set F_1 as the set consisting of two FDs, $\{A, B\} \rightarrow \{C, D\}$ and $\{B\} \rightarrow \{C\}$:

```
F_1 = [(set(['A','B']), set(['C','D'])), (set(['B']), set(['C']))]
```

*Note: the above is not necessarily one of the FDs- just an example of the syntax!

*Hint: You may use any of the functions from the lecture activities (IPython not needed to access- use closure.py) if they seem helpful! However your final answer should not involve these functions directly, nor are they necessary for this problem

2.2 Part (b)

10 points. Consider a schema $R(A_1, ..., A_n)$ which has FDs $\{A_i, A_{i+1}\} \rightarrow \{A_{i+2}\}$ for $i = 1, ..., n-2$. Create an instance of R, for $n = 4$, for which these FDs hold, and no other ones do- i.e. **all other FDs are violated.** Use a series of 'INSERT' statements below to populate a table R.

3 Problem 3

20 points total. Consider a relation $R(X, Y, Z)$. In each part of this problem you will be given a condition, and you need to create the following three instances of R (as tables in SQL):

1. An instance A
2. An instance B which differs from A only in that it has one *fewer* row.
3. An instance C which differs from A only in that it has one *additional* row.

3.1 Part (a)

10 points. Create A, B and C such that each individual attribute is a key for A, but none of the individual attributes is a key for B or C. If you believe that B and/or C cannot be created, provide them as an empty table.

3.2 Part (b)

10 points. Create A, B and C such that $Z \twoheadrightarrow X$ holds in A, but not in B or C. If you believe that B and/or C cannot be created, provide them as an empty table.

4 Bonus Problem

10 points. Prove the *transitivity rule for MVDs*: If $A \twoheadrightarrow B$ and $B \twoheadrightarrow C \implies A \twoheadrightarrow C \setminus B$, using only the basic definition of an MVD; and where A, B, C are *sets of* attributes such that $A \cup B \cup C \subseteq R$, where R is the full set of attributes.