CS 145 PS2

October 25, 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 $T(A, B, C, D)$.

Solution:

```
SELECT *
FROM T AS T1, T AS T2
WHERE T1.A = T2.A AND T1.B = T2.B
    AND (T1.C <> T2.C OR T1.D <> T2.D);
```

1.2 Part (b)

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

Solution:

```
SELECT *
FROM T AS T1, T AS T2
WHERE (T1.A = T2.A OR T1.B = T2.B OR T1.C = T2.C OR T1.D = T2.D)
    AND (T1.A <> T2.A OR T1.B <> T2.B OR T1.C <> T2.C OR T1.D <> T2.D);
```

1.3 Part (c)

A **multivalued dependency (MVD)** is defined as follows: let R be a schema i.e. a set of attributes, and consider two *sets of attributes* $X \subseteq R$ and $Y \subseteq R$. We say that a multivalued dependency (MVD), written:

$$X \twoheadrightarrow Y$$

holds on R if whenever there are two tuples t_1, t_2 such that $t_1[X] = t_2[X]$, there also exists a third tuple t_3 such that:

- $t_3[X] = t_1[X] = t_2[X]$
- $t_3[Y] = t_1[Y]$
- $t_3[R \setminus Y] = t_2[R \setminus Y]$

Note that $R \setminus Y$ is all the attributes in R that are not in Y, and that t_3 need not be distinct from t_1 or t_2. Note especially that an MVD holds on an entire *relation*, meaning that any two tuples (in any order) in the relation should satisfy the above conditions if the MVD holds. **See the end of the lecture 7 slides for more on MVDs!**

5 points In this problem, write your query to check if the MVD $\{A\} \twoheadrightarrow \{B, D\}$ holds for a relation $T(A, B, C, D)$.

Solution:

```
SELECT *
FROM T AS T1, T AS T2
WHERE T1.A = T2.A
    AND NOT EXISTS (
    SELECT *
    FROM T AS T3
    WHERE T1.A = T3.A AND T1.B = T3.B AND T1.D = T3.D
        AND T2.C = T3.C
    );
```

1.4 Part (d)

5 points A *tuple-generating dependency (TGD)* between two relations A and B, having some shared attributes $X_1, ..., X_n$, holds if, for every tuple t_A in A, there is *some* tuple t_B in B such that $t_A[X_i] = t_B[X_i]$ for $i = 1, ...n$.

In other words, for every distinct tuple in A, there must exist a corresponding tuple in B, which has the same values of shared attributes.

Consider two tables Dog(dog_name, breed, owner_name) and Owner(owner_name, ssn, hometown); check for a TGD between them.

Solution:

```
SELECT * FROM Dog AS d
WHERE NOT EXISTS (
    SELECT * FROM Owner AS p
    WHERE d.owner_name = p.owner_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

*Hint: See Activity 5-3...

Solution (one possible form):

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

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

F_2 = [(rhs,lhs) for lhs,rhs in F_1]
```

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.

Solution: We need to make sure the following FDs are violated, and violated such that all of their split FDs are also each violated (in other words, for $\{A\} \rightarrow \{B, C\}$, we want both $\{A\} \rightarrow \{B\}$ and $\{A\} \rightarrow \{C\}$ to be violated):

1. $A_1 \rightarrow \{A_2, A_3, A_4\}$
2. $\{A_1, A_3\} \rightarrow \{A_2, A_4\}$
3. $\{A_1, A_4\} \rightarrow \{A_2, A_3\}$
4. $\{A_1, A_3, A_4\} \rightarrow A_2$
5. $A_2 \rightarrow \{A_1, A_3, A_4\}$
6. $\{A_2, A_3\} \rightarrow A_1$
7. $\{A_2, A_4\} \rightarrow \{A_1, A_3\}$
8. $\{A_2, A_3, A_4\} \rightarrow A_1$
9. $A_3 \rightarrow \{A_1, A_2, A_4\}$
10. $\{A_3, A_4\} \rightarrow \{A_1, A_2\}$
11. $A_4 \rightarrow \{A_1, A_2, A_3\}$

One possible solution table is thus:

```
DROP TABLE IF EXISTS R;
CREATE TABLE R (A int, B int, C int, D int);
INSERT INTO R VALUES (0, 0, 0, 0);
INSERT INTO R VALUES (0, 1, 1, 1);
INSERT INTO R VALUES (0, 2, 0, 2);
INSERT INTO R VALUES (0, 3, 3, 0);
INSERT INTO R VALUES (0, 4, 0, 0);
INSERT INTO R VALUES (5, 0, 5, 5);
INSERT INTO R VALUES (6, 0, 0, 6);
INSERT INTO R VALUES (7, 0, 7, 0);
INSERT INTO R VALUES (8, 0, 0, 0);
INSERT INTO R VALUES (9, 9, 0, 9);
INSERT INTO R VALUES (10, 10, 0, 0);
INSERT INTO R VALUES (11, 11, 11, 0);
```

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.

Solution (one of many possible):

```
DROP TABLE IF EXISTS A;
CREATE TABLE A (X INT, Y INT, Z INT);
INSERT INTO A VALUES (0, 0, 0);
INSERT INTO A VALUES (1, 1, 1);
INSERT INTO A VALUES (2, 2, 2);

DROP TABLE IF EXISTS B;
CREATE TABLE B (X INT, Y INT, Z INT);

DROP TABLE IF EXISTS C;
CREATE TABLE C AS SELECT * FROM A;
INSERT INTO C VALUES (0, 1, 2);
```

3.2 Part (b)

10 points. Create A, B and C such that the MVD $Z \twoheadrightarrow X$ 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.

Solution (one of many possible):

```
DROP TABLE IF EXISTS A;
CREATE TABLE A (X INT, Y INT, Z INT);
INSERT INTO A VALUES (1, 0, 0);
INSERT INTO A VALUES (0, 1, 0);
INSERT INTO A VALUES (1, 1, 0);
INSERT INTO A VALUES (0, 0, 0);

DROP TABLE IF EXISTS B;
CREATE TABLE B AS SELECT * FROM A;
DELETE FROM B WHERE A = 1 AND B = 0;
```

```
DROP TABLE IF EXISTS C;
CREATE TABLE C AS SELECT * FROM A;
INSERT INTO C VALUES (2, 0, 0);
```

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.

Solution:

- Suppose that we have two tuples t_1, t_2 such that $t_1[A] = t_2[A]$
- Because $A \twoheadrightarrow B$, we know by the definition of an MVD that $\exists t_3$ s.t.:
  1. $t_3[A] = t_1[A] = t_2[A]$
  2. $t_3[B] = t_1[B]$
  3. $t_3[R \setminus B] = t_2[R \setminus B]$
- Since $B \twoheadrightarrow C$, there also $\exists t_4$ s.t.
  1. $t_4[B] = t_1[B] = t_3[B]$
  2. $t_4[C] = t_1[C]$
  3. $t_4[R \setminus C] = t_3[R \setminus C]$
- To show that $A \twoheadrightarrow C \setminus B$, we would need to show that the following holds: Given $t_1[A] = t_3[A]$, there also $\exists t_4$ s.t.:
  1. $t_4[A] = t_1[A] = t_3[A]$
  2. $t_4[C \setminus B] = t_1[C \setminus B]$
  3. $t_4[R \setminus (C \setminus B)] = t_3[R \setminus (C \setminus B)]$

We now show that the 4.A-4.C hold:
4.A: We see that:

1. $t_4[R \setminus C] = t_3[R \setminus C]$ (3.C)
2. $\implies t_4[A \setminus C] = t_3[A \setminus C] = t_1[A \setminus C]$ (2.A)
3. We also know that $t_4[C] = t_1[C]$ (3.B)
4. $\implies t_4[(A \setminus C) \cup C] = t_1[(A \setminus C) \cup C]$
5. $\implies t_4[A] = t_1[A]$

4.B: $t_4[C] = t_1[C]$ (3.B) $\implies t_4[C \setminus B] = t_1[C \setminus B]$
4.C: We see that:

1. $t_4[R \setminus C] = t_3[R \setminus C]$ (3.C)

2. Also $t_4[B] = t_3[B]$ (3.A)

3. $\implies t_4[(R \setminus C) \cup B] = t_3[(R \setminus C) \cup B]$

4. $\implies t_4[R \setminus (C \setminus B)] = t_3[R \setminus (C \setminus B)]$ $\square$.