Virginia Tech. **CS 4604 – Introduction to DBMS**
Computer Science **Spring 2016, Prakash**

Homework 6: FDs, NFs and XML

(due April 13th, 2016, 4:00pm, hard-copy in-class please)

Reminders:

a. Out of 100 points. Contains 5 pages.
b. Rough time-estimates: ~5-7 hours.
c. Please type your answers. Illegible handwriting may get no points, at the discretion of the grader. Only drawings may be hand-drawn, as long as they are neat and legible.
d. There could be more than one correct answer. We shall accept them all.
e. Whenever you are making an assumption, please state is clearly.
f. **Important:** Unless otherwise mentioned, assume that you need to show your work e.g. if the question asks 'what are R's keys?' we do not just want a list; we want a step-by-step explanation as well.
g. Lead TA for this homework: Sorour Amiri.

Q1. XML [7 points]

(Courtesy Widom and Ullman) This question is based on the XML "Vehicles" document below.

```
<Vehicles>
    <Car manf="Hyundai">
        <Model>Azera</Model>
        <HorsePower>240</HorsePower>
    </Car>

    <Car manf="Toyota">
        <Model>Camry</Model>
        <HorsePower>240</HorsePower>
    </Car>

    <Truck manf="Toyota">
        <Model>Tundra</Model>
        <HorsePower>240</HorsePower>
    </Truck>

    <Car manf="Hyundai">
        <Model>Elantra</Model>
        <HorsePower>120</HorsePower>
    </Car>

    <Car manf="Toyota">
```

```
            <Model>Prius</Model>
            <HorsePower>120</HorsePower>
        </Car>
    </Vehicles>
```

Q1.1 (3 points) Which of the following XPath expressions, when applied to the "Vehicles" document, does NOT produce a sequence of exactly three items?

a) /Vehicles/Car[@manf="Toyota"]/HorsePower
b) /Vehicles/*[HorsePower>200]/Model
c) //*[@manf="Toyota"]/@manf

Justify your answer briefly (i.e. give the output of the query (queries) which do not produce a sequence of three items).

Q1.2 (4 points) In a DTD that the "Vehicles" document satisfies, which of the following element declarations would you definitely NOT find?

a) <!ELEMENT Vehicles (Car*, Truck+, Car*)>
b) <!ELEMENT Vehicles (Car+, Truck*, Car)>
c) <!ELEMENT Vehicles ((Car | Truck)*)>
d) <!ELEMENT Vehicles (Car*, Truck*, Car*, Truck*, Car*)>

Again, just justify your choice(s) briefly (i.e. explain why you would not find your selected element declaration(s) in a DTD for Vehicles).

Q2. FDs Definition [8 points]

Suppose that we have the following four tuples in an *instance* of relation schema R with five attributes ABCDE:

A	B	C	D	E
1	2	4	3	8
1	2	5	7	9
3	6	4	7	0
1	6	5	7	9

Q2.1 (4 points) Which of the following dependencies can you infer does not hold over R? Just list the ones that do not hold, giving the reason in 1 line for each.

(a) $E \rightarrow D$, (b) $D \rightarrow E$, (c) $DE \rightarrow B$, (d) $E \rightarrow AB$, (e) $BC \rightarrow ED$

Q2.2 (4 points) If ABE → ACE and ABE → ADE hold in R, which of the following dependencies can be deduced? (No explanation is needed just mark the correct FDs)

(a) ABE → ACDE (b) B → C (c) ABCE → ADE (d) ABE → AE

Q3. Inferring FDs [27 points]

Consider the relations below and sets of functional dependencies that hold in those relations and answer the following questions. For each sub-part, it is enough for you to list only completely non-trivial FDs with a single attribute on the right hand side.

Note that 'candidate key' means just 'key' (i.e. both words are interchangeable). A candidate key (or simply key) should imply the entire relation and should be minimal. On the other hand, a 'superkey' is any super-set of a candidate key.

Q3.1 R1(A, B, C, D) with FDs AB → C, C → D, and D → A.

- Q3.1.1 (3 points) What are all the non-trivial FDs that follow from the given FDs?
- Q3.1.2 (3 points) What are all the keys (i.e. candidate keys) of R1?
- Q3.1.3 (3 points) What are all the superkeys for R1 that are not keys?

Q3.2 R2(A, B, C, D) with FDs A→ B, B → C, and B → D.

- Q3.2.1 (3 points) What are all the non-trivial FDs that follow from the given FDs?
- Q3.2.2 (3 points) What are all the keys (i.e candidate keys) of R2?
- Q3.2.3 (3 points) What are all the superkeys for R2 that are not keys?

Q3.3 R3(A, B, C, D) with FDs ABD → C, BC → D, CD → A, and AD → B.

- Q3.3.1 (3 points) What are all the non-trivial FDs that follow from the given FDs?
- Q3.3.2 (3 points) What are all the keys (i.e. candidate keys) of R3?
- Q3.3.3 (3 points) What are all the superkeys for R3 that are not keys?

Q4. Projection of FDs [26 points]

Consider the relation R = {A, B, C, D, E, F} and the following set of FDs on R:

$$F = \{AB \rightarrow C, AC \rightarrow B, AD \rightarrow E, B \rightarrow D, BC \rightarrow A, E \rightarrow F\}$$

Consider also the relations: R1 (A, B, C), R2 (A, B, C, E, F), and R3 (A, B, C, D).

Q4.1 (2x3=6 points) Compute the projection of the FD-set *F* on each of the three relations R1, R2 and R3. Only write down a minimal cover.

Q4.2 (2x3=6 points) What are *all* the candidate keys (i.e. the keys) for each of the three relations R1, R2 and R3?

Q4.3 (2x2=4 points) For each of the three relations, indicate if it is (A) BCNF and (B) 3NF. Explain your answer.

Q4.4 (6 points) Is it possible to decompose each R2 and R3 into new schemas such that they are in BCNF and the decomposition is lossless and dependency preserving? Show your work.

Q4.5 (4 points) Decompose R into multiple relations so that they are in 3NF and the decomposition is lossless and dependency preserving. Use the standard 3NF synthesis algorithm. Compare your result with Q4.4.

Q5. Lossy decompositions [8 points]

Consider a relation *R(A, B, C, D, E).* Assume *R* satisfies the following functional dependencies: $C \rightarrow E$ and $A \rightarrow C$.

Suppose we decompose *R* into these two relations: *R1(A, B, C)* and *R2(C, D, E)*. So you can think of $R1 = \Pi_{\{A,B,C\}}(r)$ and $R2 = \Pi_{\{C,D,E\}}(r)$.

Q5.1 (5 points) Show that this decomposition is not a lossless-join by giving an example. That is, give an instance *r* of *R* (i.e., give a set of tuples for *R*) such that both dependencies hold on *r* but $r \neq R1 \bowtie R2$. Give the simplest instance you can find. Also show your work (i.e. compute *R1* and *R2* and the join).

Q5.2 (3 points) Can you show that the decomposition is lossy by using the theorem discussed in class? Again, show your work.

Q6. FDs in English language [24 points]

Consider the following relation, which stores information of Facebook dataset.

FacebookDB (u_id, u_name, u_location, u_friends_count, u_posts_count, u_post_importance, u_age, u_influence, p_id, p_text, p_location, p_like_count, p_reshare_count, p_parent_id, c_id, c_text, c_reply_count, c_like_count)

Now consider the following constraints in English:

Statement 1 Every user has a unique user id (u_id). Other information for users such as name (u_name), location (u_location), number of friends (u_friends_count), and posts (u_post_count), importance of posts (u_post_importance), age, and the influence value are also kept.

Statement 2	Each user can have multiple friends, and posts.
Statement 3	For every post we have p_id (which is unique), the text, the location, the number of likes, and re-shares. If the post itself is a re-share of another post, we have the parent post id.
Statement 4	There might be several comments for each post. The comments have a unique id (c_id), text, number of replies to it (c_reply_count), and number of likes (c_like_count).
Statement 5	The importance of a user's posts (u_post_importance) is computed based on number of friends and posts (u_friends_count, u_posts_count) of the user.
Statement 6	The user's influence (u_influence) is defined using the age and the importance of the user's posts (u_age, u_post_importance).

Clearly this is not a good relational design, as it contains redundancies and update, insertion and deletion anomalies. Hence we want to decompose it into a good schema.

Q6.1 (6 points) List the functional dependencies for this relation that you can construct using the English description and also specifically mention the corresponding statement number(s) you used to formulate each FD.
Note: some statement(s) might not result in a FD.

Q6.2 (3 points) Rigorously prove (using Armstrong's axioms) that you can derive the FD: u_friends_count, u_posts_count, u_age → u_influence
from the set of the FDs you found in Q5.1. Show your steps.

Q6.3 (10 points) Is the above relation in BCNF using the FDs you found in Q5.1? If not, decompose it into a set of BCNF relations (using our standard algorithm). Is your resulting decomposition dependency-preserving too?

Q6.4 (5 points) Decompose the original relation into a set of 3NF relations instead (using the standard synthesis algorithm). Show your steps. Is your resulting decomposition in BCNF as well? Explain your answer.