Virginia Tech. **CS 4604 – Introduction to DBMS**
Computer Science **Spring 2015, Prakash**

Homework 6: FDs, NFs and XML

(due April 15th, 2015, 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: Elaheh Raisi.

Q1. FDs Definition [8 points]

(Modified version of Exercise 19.6 in your textbook) Suppose that we have the following three tuples in a legal *instance* of relation schema S with three attributes ABC:

A	B	C
1	X	a
4	X	b
5	Y	a

Q1.1 (5 points) Which of the following dependencies can you infer does not hold over schema S?

(a) $A \rightarrow B$, (b) $BC \rightarrow A$, (c) $B \rightarrow C$, (d) $B \rightarrow A$, (e) $C \rightarrow AB$

Q1.2 (3 points) Can you identify any functional dependencies that *may* hold over S?

Q2. Inferring FDs [27 points]

Consider the following relations and sets of functional dependencies that hold in those relations. 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.

Q2.1 R1(A, B, C, D) with FDs A → C, C → D, and D → B.

Q2.1.1 (3 points) What are all the non-trivial FDs that follow from the given FDs?

Q2.1.2 (3 points) What are all the keys (i.e. candidate keys) of R1?

Q2.1.3 (3 points) What are all the superkeys for R1?

Q2.2 R2(A, B, C, D) with FDs AD → C, C → A, and D → B.

Q2.2.1 (3 points) What are all the non-trivial FDs that follow from the given FDs?

Q2.2.2 (3 points) What are all the keys (i.e candidate keys) of R2?

Q2.2.3 (3 points) What are all the superkeys for R2?

Q2.3 R3(A, B, C, D, E) with FDs AB → D, B → E, C → B, and DE → C.

Q2.3.1 (3 points) What are all the non-trivial FDs that follow from the given FDs?

Q2.3.2 (3 points) What are all the keys (i.e. candidate keys) of R3?

Q2.3.3 (3 points) What are all the superkeys for R3?

Q3. Projection of FDs [26 points]

Consider the the relation schema r = {P, Q, R, S, T, U, V} and the following FDs set:

F = {PR → S, P → T, PT → R, S → U, ST → V, TV → S, QT → V, V → Q}

Consider also the relational schemas: R1 (P, Q, R, S, T), R2 (Q, T, V), and R3 (S, T, U, V).

Q3.1 (3x3=9 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.

Q3.2 (2x3=6 points) What are *all* the candidate keys (i.e. the keys) for each of the three relations R1, R2 and R3?

Q3.3 (2x3=6 points) For each of the three relations, indicate if it is (A) BCNF and (B) 3NF. Explain your answer.

Q3.4 (3 points) Decompose R3 into two relational schemas R31 and R32 so that they are in 3NF and the decomposition is lossless and dependency preserving. Show your work. Use the standard 3NF synthesis algorithm.

Q3.5 (2 points) Is it possible to decompose R3 into two schemas (say R3′ and R3′′) such that both R3′ and R3′′ are in *BCNF* and it is both lossless-join and dependency-preserving decomposition?

Q4. Creating dependencies [8 points]

A relation *R(A, B, C)* satisfies the functional dependencies $A \rightarrow B$ and $A \rightarrow C$.

Q4.1 (4 points) State another functional dependency that *R* must satisfy so these three functional dependencies (i.e., $A \rightarrow B$, $A \rightarrow C$ and the new functional dependency) allow us to infer that R also $B \rightarrow C$. Your answer cannot contain *B* or *C* on the right hand side! Show your work.

For the next two parts of this question, assume that the new functional dependency you stated in Q4.1 also holds in *R*. Again, show your work for the next two parts as well.

Q4.2 (2 points) What are *R*'s keys?

Q4.3 (2x1=2 points) Is *R* in (A) BCNF and (C) 3NF?

Q5. FDs in English language [24 points]

Consider the following table, which stores customer-order information.

CustomerOrderDB (CustomerNo, Name, Balance, CreditLimit, Ship-to-Address, OrderNo, OrderDate, DetailNo, ItemNo, QuantityOrdered, ItemExpireDate, DiscountRate, ItemPrice, ItemTax)

Now consider the following constraints in English:

Statement 1 Every customer has a unique customer number (CustomerNo). Other information for customer like Name, Balance and Credit Limit (CreditLimit) are kept.

Statement 2 Every customer can have several ship-to-addresses.

Statement 3 For every order we have HEADING information: order number (OrederNo), which is unique for every order. Order date (OrderDate) and Ship-to-address of customer.

Statement 4 Customers might request several items in one order. So, the order details that are detail number (DetailNo), item number (ItemNo) and quantity ordered (QuantityOrdered) should be recorded.

Statement 5 Every Item has Item number (ItemNo), Item Expire Date (ItemExpireDate), Discount Rate (DiscountRate), Item Price (ItemPrice) and Item Tax (ItemTax).

Statement 6 Discount Rate (DiscountRate) is computed based on the expiration date of Item (ItemExpireDate); the closer to the date of expiration; the higher discount will be applied.

Statement 7 Item tax (ItemTax) is defined using discount rate (DiscountRate) and item price (ItemPrice). Tax on discounted price is different from the tax on items without discount.

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.

Q5.1 (7 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.
Hint: You should get 6 FDs.

Q5.2 (3 points) Rigorously prove (using Armstrong's axioms) that you can derive the FD: ItemExpireDate ItemPrice → ItemTax from the set of the FDs you found in Q5.1. Show your steps.

Q5.3 (9 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.

Q5.4 (4 points) Is your resulting decomposition in Q5.3 dependency-preserving? If it is not dependency-preserving, explain your answer, and then decompose the original relation into a set of 3NF relations instead. Show your steps.

Q6. 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>
```

Q6.1 (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).

Q6.2 (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).