Handout 3: Functional Dependencies and Normalization

Q1: Inferring FDs

Consider the following schemas and sets of functional dependencies that hold in those schemas. It is enough for you to list only completely non-trivial FDs with a single attribute on the right hand side.

1). R1(A, B, C, D) with FD's C → B, C → A and A B → D.

Q1.1.1 What are all the nontrivial FDs that follow from the given FDs?

Q1.1.2 What are all the keys of R1?

Q1.1.3 What are all the superkeys for R1 that are not keys?

2). R2(A, B, C, D) with FD's A B → C, C → D and D → B.

Q1.2.1 What are all the nontrivial FDs that follow from the given FDs?

Q1.2.2 What are all the keys of R2?

Q1.2.3 What are all the superkeys for R2 that are not keys?

3). R3(A, B, C, D, E, F) with FD's E → F, E → D, A D → B, A D → C, and F → B.

Q1.3.1 What are all the nontrivial FDs that follow from the given FDs?

Q1.3.2 What are all the keys of R3?

Q1.3.3 What are all the superkeys for R3 that are not keys?

Q2: Projection of FDs [12 points]

For all the parts of this question, again you only need to write down FDs which are completely non-trivial and which have single attributes on the right hand side. Also, in your result, a minimal basis (aka canonical cover or minimal cover) is enough.

Q2.1 Suppose we have a relation OrderPizza(customer_id, customer_name, pizza_id, pizza_name, ordertime, quantity, slices) which has the following set of FDs:

customer_id → customer_name,
pizza_id → pizza_name,
customer_id pizza_id ordertime → quantity slices

Project these FDs onto the relation X(customer_id, customer_name, quantity, ordertime).

Q2.2 Suppose we have a relation TakeCourses(student_id, student_name, professor_id, professor_name, course_number, department, capacity) which has the following set of FDs:

student_id → student_name,
professor_id → professor_name,
course_number department → professor_id capacity

Project these FDs onto the relation Y(professor_name, student_name, course_number, department).

Q2.3 Suppose we have a relation R(A, B, C, D, E, F) with the following set of FDs:

B → A, E → C, F → D, C D → B and C F → A

Project these FDs onto the relation Z(A, E, F).

Q3: 3NF Decomposition

Consider a relation **Stocks**(B, O, I, S, Q, D), whose attributes may be thought of informally as broker, officer (of the broker), investor, stock, quantity (of the stock owned by the investor), and dividend (of the stock). Let the set of FD's for **Stocks** be:

S → D, I → B, IS → Q, and B → O

Q3.1 List all the keys for **Stocks**.

Q3.2 Use the 3NF synthesis algorithm to find a lossless-join, dependency-preserving decomposition of **Stocks** into a set of 3NF relations. Please show your steps.

Q4: FDs in English Language

Consider the following relational schema (similar to the MOVIE database in the textbook):

StarringMovie(actor_name, actor_address, studio_name, studio_address, title, year, budget, role, dailywage)

Consider also the following constraints in English:

Statement 1 Every actor has a unique name and an address.

Statement 2 Every studio has a unique name and an address.

Statement 3 Every movie has a release year, budget and a unique movie name, and it is owned by only one studio.

Statement 4 An actor can play multiple **different** roles in the same movie. For example, an actor can play a "teacher" and a "policeman", but cannot play two "students" in a campus/school movie. Multiple actors can play the same role in the same movie (so it is possible that there may be many actors playing the role of "students" in a movie.).

Statement 5 If multiple actors play the same role in the same movie, they **must** get the same daily wage. For example, all "students" in the same movie should get the same daily wage. Note that if they play a "student" in different movies, they may get different daily wages. Also, if different actors play different roles in the same movie, they **may** get the different daily wages, e.g., someone playing a "teacher" may get more or less per day than someone playing a "student" in the same movie.

Q4.1 Is this a good relational design or not? Explain your answer: without going into any of the normal forms, list the different types of anomalies this relation has.

Q4.2 Based on the English description, list all the non-trivial functional dependencies for this schema and list also the corresponding statements you used to formulate each FD. (**Note**: a minimal cover is enough and some FDs may come from multiple statements).

Q4.3 Is the above schema in BCNF? If not, try to convert it into BCNF first. Is your result of decomposition dependency-preserving? If it is not dependency preserving, explain your answer, and then transform it into 3NF. Show your steps.

Q5: Normal Forms and Implied FDs

Consider the schema relation R(A, B, C, D, E) and the following set of FDs:

$$C \rightarrow A\ D,\ D\ E \rightarrow B,\ A \rightarrow E \text{ and } B \rightarrow C$$

Q5.1 List all the keys of R.

Q5.2 Indicate all the BCNF violations, if any.

Q5.3 Decompose the relations, as necessary, into collections of relations that are in BCNF. You can choose any of BCNF violations that you listed in Q5.2 above to decompose. Please show your steps.

Q5.4 For any given relation X with set of FDs F holding in it, are we required to consider FDs that are not in F but follow from it (i.e. the 'implied FDs') to figure out if the relation violates BCNF? If YES, give an example. If NO, prove why we don't need to.

Hint: If an implied FD $Y \rightarrow Z$ violates BCNF, can we deduce anything about any of the FDs in the given set F?

Q5.5 Is the result of your decomposition (in Q5.3) dependency preserving? Explain your answer. If it is not dependency preserving, then transform it into 3NF. Make sure you show your steps.

Q6: A Concert database

Excited by what you are learning in CS4604, you decide to create a database to track the songs your favorite band plays in its live concerts. Since you decide that E/R diagrams are for kids, you decide to create a relation schema directly for your database. After much consideration, you believe that a single relation schema will serve the purpose:

Concerts(City, Venue, Year, Month, Date, Song, Album)

In this relation, City (e.g., "Blacksburg") and Venue (e.g., "Cassell Colisseum") record where the concert took place and Year, Month, and Date keep track of when the concert took place. The idea is that these five attributes uniquely specify a concert. The attribute Song records the name of a song performed at a concert. You add the attribute Album to record which album the song belongs to. Perfect!

However, after using the database for a few months, you realize that your band (and the real world) have some characteristics that you should model in your database. Convert each of the next four sentences about Concerts into a functional dependency. You can use the first letter of each attribute as an abbreviation for the attribute.

Consider each of the next four (1-4) sentences independently. If you cannot write down a functional dependency, say so, and explain why you cannot, if possible. Do not assume any other constraints, even if they seem reasonable to you.

1. Each song appears in at most one album. In other words, the band does not repeat the same song in different albums.

2. A city does not have two venues with the same name. In other words, City and Venue serve to identify the location of a concert uniquely.

3. In an effort to please its fans, the band plays at most one song from any album in a given concert.

4. The manager books the band in any city at most once every year.

5. Use Armstrong's axioms derive the FD City Year Album→ Song.

6. What are the keys for Concerts?

7. What normal forms does Concerts satisfy?

8. You realize that you must decompose Concerts into multiple relations. Here is a candidate decomposition into two relations:

   Concerts1(City, Venue, Year, Month, Date)
   Concerts2(City, Year, Song, Album)

   (a) For each relation Concerts1 and Concerts2, state what normal forms it satisfies.

   (b) Is the decomposition of Concerts into Concerts1 and Concerts2 is lossless-join?

   (c) Is this decomposition dependency preserving?