Handout 3: Practice Questions on Dependencies and Normalization

CS 4604 (Spring 2013)

April 8, 2013

Problem 1 (From the 2005 midterm examination) Consider the relation

`Inventory(Manufacturer, Brandname, Type, Weight, Store)`.

This relation stores the items that a grocery store stocks. Each tuple in relation `Inventory` represents the fact that a store sells an item of a particular type and brand name manufactured by a particular company. The relation also stores the weight of the item. Two tuples that such a relation may contain are:

`(Kellogg's Company, Frosted Flakes, Cereal, 14oz., Hokies Holesome Foods)` and
`(Kraft Foods, Philadelphia, Cream Cheese, 8oz., Healthy Hokies Store)`.

Convert each of the next three sentences in English about `Inventory` into a functional or a multi-valued dependency. *Consider each of these three sentences independently.*

1. A manufacturer holds the trademark for a brand name of an item of a particular type, i.e., no two manufacturers can use the same brand name for items of the same type. For example, two different manufacturers cannot use the brand name `Philadelphia` for the food type `Cream Cheese`.

2. For each type, each store sells only one brand name made by each manufacturer. For example, `Hokies Holesome Foods` does not sell any `Cereal` other than `Frosted Flakes` that is manufactured by `Kellogg's Company`.

3. If a particular item (specified by its manufacturer, brand name, and type) is available in a particular weight at a store, then that weight is available at all stores carrying that food item.

For the rest of this question, assume that all the functional and/or multi-valued dependencies you specified in the previous three parts hold in `Inventory`, as do any dependencies that follow from them. However, no other dependencies hold in `Inventory`.

4. What are the keys for `Inventory`?

5. What normal forms does `Inventory` satisfy?

6. Consider the decomposition of `Inventory` into
   `Inventory1(Manufacturer, Brandname, Type, Store)` and
   `Inventory2(Manufacturer, Brandname, Type, Weight)`.
   Use the chase to show that this decomposition of `Inventory` is not lossless-join.

7. Modify one of the attributes in either `Inventory1` or in `Inventory2` to obtain a lossless-join decomposition. Use the chase to prove that this new decomposition is lossless-join.

8. State all the normal forms that the decomposition in part 6 satisfies.

9. Decompose `Inventory` into a set of relations that are in BCNF such that the decomposition is lossless join. Is this decomposition dependency-preserving?

10. Use the 3NF synthesis algorithm to compute a lossless-join dependency-preserving decomposition of `Inventory` into relations that are in 3NF. Are all the relations in BCNF?

Problem 2 (From the 2006 midterm examination) Excited by what you are learning in CS 4604, you decide to create a database to track the songs your favourite 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 schema will serve:

`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 realise 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 or a multi-valued dependency. You can use the first letter of each attribute as an abbreviation for the attribute. *Consider each of these four sentences independently.* If you cannot write down a functional or a multi-valued 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.

For the next two parts of this question, assume that all the functional and/or multi-valued dependencies you specified in the previous parts hold in `Concerts`, as do any dependencies that follow from them. However, no other dependencies hold in `Concerts`.

5. Use (a) Armstrong's axioms and (b) the chase to derive the FD `City Year Album` $\rightarrow$ `Song`.

6. What are the keys for `Concerts`?

7. What normal forms does `Concerts` satisfy?

8. You realise 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) Use the chase to determine whether the decomposition of `Concerts` into `Concerts1` and `Concerts2` is lossless-join.

(c) For each of the four FDs from the previous page, specify which relation (`Concerts1`, `Concerts2`, both, or neither) you can use to verify the FD.

FD 1

FD 2

FD 3

FD 4

9. After a few years of using relations `Concerts1` and `Concerts2` and faithfully recording the band's performances in it, your realise that your beloved band plays a mean trick on its audience. In each city, all its concerts (spread over different years, of course) feature exactly the same songs! For example, the songs the band played in Blacksburg in 1996 are identical to the songs the band played in Blacksburg in 2000 and in 2004. Write down the dependencies that model this discovery and the names of the relations (`Concerts1` and/or `Concerts2`) in which these dependencies hold.

10. Decompose the relations affected in the previous part into 4NF. Just apply one step of the 4NF decomposition algorithm.

11. Use the 3NF synthesis algorithm to decompose `Concerts` into a set of dependency-preserving relations each of which is in 3NF. Is each relation also in BCNF? If not, is there any way of decomposing the BCNF-violating relations so that the new relations are in BCNF, yet the overall decomposition is dependency-preserving?