In-class activities: Nov 6, 2017

Today’s activities focus on an aspect of database design: understanding constraints that hold over your data (functional dependencies) and using them to refine your schema. You will work on today’s exercises on paper or your whiteboard. Work through these problems with your 3-person team and check your solutions with the other teams at your table.

Pacing:

As always, you should be mindful of your pace through the activities. Take time with problems that you feel you need to understand better, and feel free to go faster on problems where you feel you already have a good grasp of things. However, make sure that you are making steady progress. If you need assistance, let us know. We will make several interruptions to discuss individual steps in class, and help you maintain your pace.

In this activity sheet, you will find problems marked as “optional exercise”. Skip those if you are falling behind, and try them on your own at home.

Quiz:

As usual, there will be a short end-of-class quiz. As before, you should indicate your 3-person team on the quiz. We keep track of that based on table and laptop numbers. Note your table number, indicated on the whiteboard and monitor assigned to your table. There are two laptop numbers, one on yellow sticker and one on white sticker. Indicate both if possible, but at least one is necessary. If you cannot find these numbers, please talk to the instructors or TAs before you turn in your quiz.
**Step 1: Checking whether FDs hold**

Functional dependencies define constraints over data. A functional dependency may or may not hold over a data instance. A functional dependency \( A \rightarrow B \) holds over a data instance of a relation \( R \), if for any two tuples in \( R \) that share the same value for attribute \( A \), the tuples also share the same value for attribute \( B \).

Consider the following instance of a relation \( R(A,B,C,D) \):

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

Consider each of the following functional dependencies and determine whether they hold on this data instance.

1. \( A \rightarrow B \)
2. \( B \rightarrow A \)
3. \( B \rightarrow C \)
4. \( B \rightarrow D \)
5. \( D \rightarrow B \)
6. \( D \rightarrow A \)
7. \( AB \rightarrow C \)
8. \( AB \rightarrow D \)
9. \( BD \rightarrow C \)
Step 2: Computing closure

Consider the following schema and set of functional dependencies:

R(A, B, C, D, E, F, G) with functional dependencies:

A → D
D → C
F → EG
DC → BF

Compute the appropriate closures to determine whether each of the following dependencies hold:

A → E
AC → E
C → F
D → G
FC → B

Step 3: Candidate keys

1. Look at relation R from step 2. Can you identify the key of relation R?
2. Look at relation R from step 1. Which are the possible keys to relation R?
Step 4: BCNF decomposition (normalization)

1. Is relation R from step 2 in BCNF? Why? If it is not in BCNF, decompose it.

2. Suppose we are told that R(A, B, C, D) is in BCNF, and that exactly 3 out of the 4 FDs listed below hold for R. Choose the FD that R does not satisfy, and explain your reasoning.

1: A → BCD
2: BC → A
3: CD → B
4: D → C

*** Optional exercise ***

3. BCNF decomposition does not always yield the same result. The outcome depends on the order you choose to decompose on. Can you come up with an example (a relation R and a set of FDs) that demonstrates this?
Step 5: Lossless vs lossy decomposition

1. Consider relation R from step 2. Is the decomposition of R into R1(A,D) and R2(B,C,D,E,F,G) in BCNF? Is it a lossless decomposition?

2. Consider relation R from step 2. Is the decomposition of R into R1(D,E,G) and R2(A,B,C,E,F) in BCNF? Is it a lossless decomposition?

*** Optional exercise ***

3. Consider a relation R(A,B,C,D) and the functional dependencies that hold over R:

   \[ ABC \rightarrow D \]
   \[ D \rightarrow A \]

   Is the decomposition of R into R1(A,B,C) and R2(B,C,D) in BCNF? Is it a lossless decomposition?
Step 6: Schema refinement in an application

You are designing an application that maintains and analyzes information on product orders. The data is currently stored in the following relation:

Orders (OrderDate, CustomerID, CustomerName, Zipcode
  ProductID, ProductName, ItemPrice, Amount, Total)

Here is a description of the attributes, and some assumptions:

- OrderDate: The date of the order.
- CustomerID: A number that uniquely identifies a customer.
- CustomerName: The name of a customer. Multiple customers may have the same name.
- ZipCode: The customer's zipcode.
- ProductID: A number that uniquely identifies a product.
- ProductName: The name of a product. Multiple products may have the same name.
- ItemPrice: The price of the product.
- Amount: How many items of the particular product are in this order.
  - We assume that customer orders for a particular product on the same day are combined, and the number or products in the order is represented by this amount. So, the date, customer, and product determine a particular amount.
- Total: The total price. This is equal to ItemPrice*Amount.

1. Can you identify the functional dependencies that hold on relation Orders?

2. Can you refine this initial (poor) design?

3. Can you see any reason why you may prefer to not force your schema to conform to BCNF in this application? Consider what useful queries may become slower due to the decomposition.