Scott Fenton

Homework 6

1. Exercise 19.3

1. The functional dependencies are R: Z -> Y, X -> Y, and XZ -> Y

2. Given the changes the FD remains the same. R: Z -> Y, X -> Y, and XZ -> Y

3. Given the change the FDs are: R: X -> Y, Y -> X, XZ -> Y, and YZ -> X

2. Exercise 19.2

1. List all keys for R: The Keys are CDE, ACD, BCD.
2. Is R in 3NF? Yes, because B, E, and A are parts of each of the keys
3. Is R in BCNF? No, because A, BC, and ED, do not contain a key.

3.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Department | Room | Period |
| Java 1 | CS | 100 | 5 |
| Java 2 | CS | 110 | 4 |
| Data Structures | CS | 100 | 6 |
| Calc 1 | Math | 350 | 6 |
| Calc 2 | Math | 390 | 6 |
| Linear Algebra | Math | 350 | 4 |

**A**. Find a FD with one attribute on each side that holds for this instance

R -> D.

**B**. Disprove the proposed FD “department period -> Room” DP -> R

DP -> R cannot work because Department-room MATH-6 corresponds to two room values of 350 and 390.

**C**. Find a single-column key and a multi-column key that hold for this instance

A single column key that holds true is the “name” column.

A multi-column key that holds true is “name department”, “name room”, or “name period”.

**D**.  Propose a decomposition that removes the redundancy implied by the FD you found in a. Show the smaller table in full and state what column can be dropped in the larger table compared to the original table. Hint: this is a lot like the R -> W dependency example of Figures 19.1 and 19.2 and in the slides of Lecture 20.

NDRP R->D

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Department | Room | Period |
| Java 1 | CS | 100 | 5 |
| Java 2 | CS | 110 | 4 |
| Data Structures | CS | 100 | 6 |
| Calc 1 | Math | 350 | 6 |
| Calc 2 | Math | 390 | 6 |
| Linear Algebra | Math | 350 | 4 |

|  |  |
| --- | --- |
| Room | Department |
| 100 | CS |
| 110 | CS |
| 350 | Math |
| 390 | Math |

By creating this second table, we can preserve the relationship between Room and department. Updating room will change department, and deleting a name, does not affect the room-department relationship.

4. Exercise 19.10 (a. and b.) For part b., a good decomposition means it provides lossless join(s) and preserves FDs. For each case 1-4, first check if the join(s) are lossless and give up if not. If lossless, determine whether the FDs are preserved.

1. B 🡪 C, D 🡪 A;

A. Candidate keys? BD.

B. Good Decomposition? No, the decomposition is lossy because the join of AD and BC could be much larger than ABCD.

2. AB -> C, C -> A, C -> D

A. Candidate keys? AB,BC

B. Good Decomposition? Yes, the decomposition is lossless. We can see this because ACD union BC -> ACD. It is not FD preserving because since AB->C is not preserved.

3. A -> BC, C ->AD;

A. Candidate keys? A,C

B. Good Decomposition? Yes, because it is already in BCNF form. It is not FD preserving since

C->AD cannot be enforced.

4. A -> B, B -> C, C -> D

A. Candidate keys? A

B. Good Decomposition? Yes, the decomposition is lossless, but does not preserve FD since

B -> C is not preserved.

5. A -> B, B -> C, C -> D

A. Candidate keys? A

B. Good Decomposition? Yes, this is a lossless BCNF decomposition. It does not preserve FD for example B->C.

5. Find a BCNF decomposition of the relation of 19.10 for parts 1. and 2., using the FD sets given for each case. Analyze the BCNF decomposition for unpreserved dependencies. If you find a 3NF decomposition that preserves dependencies along the way, report it. Note that the first step is to find the keys, so you can test FDs for as described on pp. 616-618. When you have a choice as to which non-BCNF FD to work on (in part 2), choose the first such FD and use it for a decomposition, then later come back and determine what happens if you choose the second one.

1. B 🡪 C, D 🡪 A;

1. Candidate keys? BD.

2. AB -> C, C -> A, C -> D

1. Candidate keys? AB,BC

3. A -> BC, C ->AD;

1. Candidate keys? A,C

4. A -> B, B -> C, C -> D

1. Candidate keys? A

5. A -> B, B -> C, C -> D

1. Candidate keys? A

6.

CONNECT CS630/ flightmgr;

DROP USER webapp\_user CASCADE;

CREATE USER webapp\_user

IDENTIFIED BY sesame

DEFAULT TABLESPACE users

QUOTA UNLIMITED ON users;

--Grant a privilege to all users

GRANT CREATE SESSION TO PUBLIC;

-- create the roles

CREATE ROLE webappuser;

GRANT SELECT ON flights TO webappuser;

GRANT SELECT ON employees TO webappuser;

GRANT SELECT ON certified TO webappuser;

GRANT SELECT ON aircraft TO webappuser;

GRANT SELECT ON assigned\_flights TO webappuser;

GRANT SELECT ON delayed\_flights TO webappuser;

-- assign the users to their roles

GRANT webappuser TO webapp\_user;

7.

CREATE VIEW empinfo AS

SELECT e.eid, e.ename

FROM employees e

--

GRANT SELECT ON empinfo TO clerk;