Shane Hauck

Good, 19.5/20, see comments below.

Database Systems

HW 7 – Relational Theory

7.21 Give a lossless decomposition into BCNF of schema R of Exercise 7.1

$R = (A, B, C, D, E)$

$S = \{A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A\}$

- Compute $\{X\}^+$
  - $\{A\}^+ = (A, B, C, D, E)$ and
  - $\{CD\}^+ = (C, D, E, A, B)$
- Check BCNF
  - Does not hold because of functional dependency $B \rightarrow D$ as B is not a key.
- Compute $\{B\}^+$
  - $\{B\}^+ = (B, D)$
- Create individual relations
  - $R_1 = (B, D)$
  - $R_2 = (B, A, C, E)$
- Compute FD's for R_1 and R_2 called S_1 and S_2
  - $S_1 = \{B \rightarrow D\}$
  - $S_2 = \{A \rightarrow BC, E \rightarrow A\}$
- Check BCNF
  - R_1 holds.
  - R_2 does not hold because of functional dependency $E \rightarrow A$ as E is not a key.
- Compute $\{E\}^+$
  - $\{E\}^+ = (E, A, B, C)$
- R_2 and S_2 stays the same but BCNF now holds.
  - $R_2 = (B, A, C, E)$
  - $S_2 = \{A \rightarrow BC, E \rightarrow A\}$
- BCNF decomposition
  - $R_1 = (B, D)$
    - $S_1 = \{B \rightarrow D\}$
  - $R_2 = (A, B, C, E)$
    - $S_2 = \{A \rightarrow BC, E \rightarrow A\}$

Good

7.23 Explain what is meant by *repetition of information* and *inability to represent information*. Explain why each of these properties may indicate a bad relational database design.

Repetition of information is when within one relation, the values of one attribute are determined by the values of another attribute and both values are reused throughout the relation. This may indicate a bad relational database design because it could lead to more difficulty when updating the database. When changing the data, you may be changing the data in one place and not in the others, thus causing inconsistencies. Having repetition also cause an increase in storage space for the relation.

The inability to represent information is when relationships don't exist among all attributes in a relation resulting in unrelated attributes to be missing values. This may indicate a bad relational database design because attributes that don't have information must either be filled in as null values or the whole tuple cannot be used in the relation. This could potentially result in the loss of information.

7.24 Why are certain functional dependencies called *trivial* functional dependencies?

Certain functional dependencies are called trivial functional dependencies because they are clear to see and relatively obvious. For functional dependency to be trivial, the dependent must be included in the determinant. This allows for the functional dependency to be obvious because you already have the answer to the dependent inside the determinant.

7.26 Consider the following proposed rule for functional dependencies: If $\alpha \rightarrow \beta$ and $\gamma \rightarrow \beta$, then $\alpha \rightarrow \gamma$. Prove that this rule is *not* sound by showing a relation *r* that satisfies $\alpha \rightarrow \beta$ and $\gamma \rightarrow \beta$, but does not satisfy $\alpha \rightarrow \gamma$.

Counterexample:

- Take the schema:
  - $R = (\alpha, \beta, \gamma)$
- With functional dependencies
  - $S = \{\alpha \rightarrow \beta, \gamma \rightarrow \beta, \alpha \rightarrow \gamma\}$
- This could potentially create a table of

α	β	γ
$\alpha 1$	$\beta 1$	$\gamma 1$
$\alpha 1$	$\beta 1$	$\gamma 2$

- Checking if functional dependencies hold
  - $\alpha \rightarrow \beta$ holds
  - $\gamma \rightarrow \beta$ holds
  - $\alpha \rightarrow \gamma$ does not hold as the same value of α has two different values of γ

7.28 Using the functional dependencies of Exercise 7.6, compute B+.

$R = (A, B, C, D, E)$

Functional Dependencies: $\{A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A\}$

$\{B\}^+ = \{BD\}$ – D is functionally dependent on B, so it is added to the closure of B.

$\{BD\}^+ = \{BD\}$ – B or D are not determinants for any of the remaining functional dependencies, so no attributes are added to the closure.

Hence:

$\{B\}^+ = \{BD\}$

7.30 Consider the following set F of functional dependencies on the relation schema (A, B, C, D, E, G):

$A \rightarrow BCD$

$BC \rightarrow DE$

Good

$B \rightarrow D$

$D \rightarrow A$

a. Compute B^+.

- $\{B\}^+ = \{B, D\}$ – given by $B \rightarrow D$
- $\{BD\}^+ = \{B, D, A\}$ – given by $D \rightarrow A$
- $\{BDA\}^+ = \{B, D, A, C\}$ – given by $A \rightarrow C$
- $\{BDAC\}^+ = \{B, D, A, C, E\}$ – given by $BC \rightarrow E$
- Hence, $\{B\}^+ = \{A, B, C, D, E\}$

b. Prove (using Armstrong's axioms) that AG is a superkey .

- $X = \{AG\}$
- Via the transitivity rule we are able to figure out the AG is a super key. B, C and D are all functionally dependent of A. When using the transitivity rule (if $\alpha \rightarrow \beta$ holds and $\beta \rightarrow \gamma$ holds, then $\alpha \rightarrow \gamma$ holds), we can add E to the closure because E is functionally dependent of B and C. We now have all attributes from the relation schema in the closure of AG.
- $\{AG\}^+ = \{ABCDEG\}$

e. Give a BCNF decomposition of the given schema using the original set F of functional dependencies.

- Compute $\{X\}^+$
  - $\{AG\}^+ = (A, B, C, D, E, G)$
- Check BCNF
  - $BC \rightarrow D$ is first functional dependency to not hold.
- Compute $\{BC\}^+$
  - $\{BC\}^+ = (B, C, D, E, A)$
- Create individual relations
  - $R_1 = (B, C, D, E, A)$
  - $R_2 = (B, C, G)$
- Compute FD's for R_1 and R_2 called S_1 and S_2
  - $S_1 = \{A \rightarrow BCD, BC \rightarrow DE, B \rightarrow D, D \rightarrow A\}$
  - $S_2 = \{BCG \rightarrow BCG\}$
- Check BCNF
  - R_1 does not hold because of functional dependency $D \rightarrow A$ due to D not being in the key.
  - R_2 holds.
- Compute $\{D\}^+$
  - $\{D\}^+ = (D, A)$
- Create individual relations
  - $R_1 = (D, A)$
  - $R_3 = (D, B, C, E)$
- Compute FD's for R_1 and R_3 called S_1 and S_3
  - $S_1 = \{D \rightarrow A\}$
  - $S_3 = \{BC \rightarrow DE, B \rightarrow D\}$
- Check BCNF
  - R_1 holds.
  - R_2 holds.
- BCNF decomposition
  - $R_1 = (D, A)$
    - $S_1 = \{D \rightarrow A\}$
  - $R_2 = (B, C, G)$
    - $S_2 = \{BCG \rightarrow BCG\}$
  - $R_3 = (D, B, C, E)$
    - $S_3 = \{BC \rightarrow DE, B \rightarrow D\}$

Mistake here. If BC → D then the R1 should be R1(B,C,D) and R2(B,C,G,A,E) . Has a ripple efect through rest of answer.

7.36 Show that every schema consisting of exactly two attributes must be in BCNF regardless of the given set F of functional dependencies.

If a relation consists of exactly two attributes, the schema must be in BCNF regardless of the given set F of functional dependencies because there is no creatable set of functional dependencies that will violate BCNF. Say there are two attributes (A,B), the possible functional

dependencies are: $\{AB \rightarrow AB\}, \{A \rightarrow B\},$ and $\{B \rightarrow A\}$. The possible functional dependencies are either trivial or the candidate key is a super key, hence making the schema in BCNF.

7.37 List the three design goals for relational databases and explain why each is desirable.

1. **Lossless-join decompositions** are desirable because it allows the databases to remain accurate.
2. **Dependency preserving decompositions** are desirable because it allows for updates of the database to be correct as the functional dependencies are doing what they are supposed to do.
3. **Minimization of repetition of information** is desirable because it saves storage.

7.38 In designing a relational database, why might we choose a non-BCNF design?

When designing a relational database, we might choose a non-BCNF design because a BCNF design may not be dependency preserving. Since dependency preservation is usually considered desirable, we may consider choosing another normal form that is weaker than BCNF that allows us to preserve dependencies.

7.41 Explain why 4NF is a normal form more desirable than BCNF.

4NF is a normal form more desirable than BCNF because it reduces the repetition of information. As well as also satisfying BCNF, 4NF gets rid of multivalued dependency meaning information will not be repeated more than it has to be.

7.42 Normalize the following schema, with given constraints to 4NF.

books(*accessionno*, *isbn*, *title*, *author*, *publisher*)
users(*userid*, *name*, *deptid*, *deptname*)
accessionno → *isbn*
isbn → *title*
isbn → *publisher*
isbn →→ *author*
userid → *name*
userid → *deptid*
deptid → *deptname*

- Check to see if books is in BCNF
    - Does not hold because $accessionno \rightarrow isbn$ as accessionno is not a key.

- Compute $\{accessionno\}^+$
  - $\{accessionno\}^+ = \{accessionno, isbn\}$
- Create individual relations
  - $R_1 = \{accessionno, isbn\}$
  - $R_2 = \{isbn, title, author, publisher\}$
- Compute FD's for R_1 and R_2 called S_1 and S_2
  - $S_1 = \{accessionno \rightarrow isbn\}$
  - $S_2 = \{isbn \rightarrow title, isbn \rightarrow publisher, isbn \twoheadrightarrow author\}$
- Check BCNF
  - R_1 holds
  - R_2 holds
- Get rid of multivalued dependency in R_2. Split into 2 relations
  - $R_2 = \{isbn, title, publisher\}$
  - $R_3 = \{isbn, author\}$
- Compute FD's for R_2 and R_3 called S_2 and S_3
  - $S_2 = \{isbn \rightarrow title, isbn \rightarrow publisher\}$
  - $S_3 = \{isbn, author \rightarrow isbn, author\}$
- The previous book relation is now if 4NF with relations and functional dependencies:
  - $R_1 = \{accessionno, isbn\}$
    - $S_1 = \{accessionno \rightarrow isbn\}$
  - $R_2 = \{isbn, title, publisher\}$
    - $S_2 = \{isbn \rightarrow title, isbn \rightarrow publisher\}$
  - $R_3 = \{isbn, author\}$
    - $S_3 = \{isbn, author \rightarrow isbn, author\}$
- Check to see if users is in BCNF
  - Does not hold, userid or deptid is not a key.
- Compute $\{deptid\}^+$
  - $\{deptid\}^+ = (deptid, deptname)$
- Create individual relations
  - $R_3 = (deptid, deptname)$
  - $R_4 = (deptid, userid, name)$
- Compute FD's for R_3 and R_4 called S_3 and S_4
  - $S_3 = (deptid \rightarrow deptname)$
  - $S_4 = (userid \rightarrow name, userid \rightarrow deptid)$
- Check to see if R_3 and R_4 are in BCNF
  - R_3 holds
  - R_4 holds
- Since there is no multivalued functional dependencies, the previous users relation is now in 4NF with relations of FD's:
  - $R_3 = (deptid, deptname)$
    - $S_3 = (deptid \rightarrow deptname)$
  - $R_4 = (deptid, userid, name)$
    - $S_4 = (userid \rightarrow name, userid \rightarrow deptid)$
- The previous schema is normalized with given constraints into 4NF as follows:

- $bookAccess = \{accessionno, isbn\}$
  - $S_1 = \{accessionno \rightarrow isbn\}$
- $books = \{isbn, title, publisher\}$
  - $S_2 = \{isbn \rightarrow title, isbn \rightarrow publisher\}$
- $authors = \{isbn, author\}$
  - $S_3 = \{isbn, author \rightarrow isbn, author\}$
- $depts = (deptid, deptname)$
  - $S_3 = (deptid \rightarrow deptname)$
- $users = (deptid, userid, name)$
  - $S_4 = (userid \rightarrow name, userid \rightarrow deptid)$

Good