PURDUE UNIVERSITY

CS34800
Information Systems

Normalization

Prof. Chris Clifton

3 October 2016

Normalization

Goal = BCNF = Boyce-Codd Normal Form =
all FD's follow from the fact "key $\rightarrow$ everything."

- Formally, R is in BCNF if for every nontrivial FD for R, say $X \rightarrow A$, then X is a superkey.
  - "Nontrivial" = right-side attribute not in left side.

Why?

1. Guarantees no redundancy due to FD's.
2. Guarantees no *update anomalies* = one occurrence of a fact is updated, not all.
3. Guarantees no *deletion anomalies* = valid fact is lost when tuple is deleted.

First Normal Form

- Domain is **atomic** if its elements are considered to be indivisible units
  - Examples of non-atomic domains:
    - Set of names, composite attributes
    - Identification numbers like CS101 that can be broken up into parts
- A relational schema R is in **first normal form** if the domains of all attributes of R are atomic
- Non-atomic values complicate storage and encourage redundant (repeated) storage of data
  - Example: Set of accounts stored with each customer, and set of owners stored with each account
  - We assume all relations are in first normal form (and revisit this in Chapter 22: Object Based Databases)

First Normal Form (Cont'd)

- Atomicity is actually a property of how the elements of the domain are used.
  - Example: Strings would normally be considered indivisible
  - Suppose that students are given roll numbers which are strings of the form *CS0012* or *EE1127*
  - If the first two characters are extracted to find the department, the domain of roll numbers is not atomic.
  - Doing so is a bad idea: leads to encoding of information in application program rather than in the database.

Boyce-Codd Normal Form

A relation schema *R* is in BCNF with respect to a set *F* of functional dependencies if for all functional dependencies in F^+ of the form

$$\alpha \rightarrow \beta$$

where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of the following holds:

- $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$)
- α is a superkey for *R*

Example schema *not* in BCNF:

instr_dept (*ID, name, salary, dept_name, building, budget*)

because *dept_name* $\rightarrow$ *building, budget*
holds on *instr_dept,* but *dept_name* is not a superkey

Decomposing a Schema into BCNF

- Suppose we have a schema *R* and a non-trivial dependency $\alpha \rightarrow \beta$ causes a violation of BCNF.

  We decompose *R* into:
  - (α U β)
  - (*R* - (β - α))
- In our example,
  - α = *dept_name*
  - β = *building, budget*

  and *inst_dept* is replaced by
  - (α U β) = (*dept_name, building, budget*)
  - (*R* - (β - α)) = (*ID, name, salary, dept_name*)

BCNF and Dependency Preservation

- Constraints, including functional dependencies, are costly to check in practice unless they pertain to only one relation
- If it is sufficient to test only those dependencies on each individual relation of a decomposition in order to ensure that *all* functional dependencies hold, then that decomposition is *dependency preserving.*
- Because it is not always possible to achieve both BCNF and dependency preservation, we consider a weaker normal form, known as *third normal form.*

Third Normal Form

- A relation schema R is in **third normal form (3NF)** if for all:

  $$\alpha \rightarrow \beta \text{ in } F^+$$

  at least one of the following holds:
  - $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \in \alpha$)
  - α is a superkey for R
  - Each attribute A in $\beta - \alpha$ is contained in a candidate key for R.

    (**NOTE**: each attribute may be in a different candidate key)
- If a relation is in BCNF it is in 3NF (since in BCNF one of the first two conditions above must hold).
- Third condition is a minimal relaxation of BCNF to ensure dependency preservation (will see why later).

Goals of Normalization

- Let R be a relation scheme with a set F of functional dependencies.
- Decide whether a relation scheme R is in "good" form.
- In the case that a relation scheme R is not in "good" form, decompose it into a set of relation scheme $\{R_1, R_2, ..., R_n\}$ such that
  - each relation scheme is in good form
  - the decomposition is a lossless-join decomposition
  - Preferably, the decomposition should be dependency preserving.

Lossless-join Decomposition

- For the case of $R = (R_1, R_2)$, we require that for all possible relations r on schema R

$$r = \prod_{R1}(r) \bowtie \prod_{R2}(r)$$

- A decomposition of R into R_1 and R_2 is lossless join if at least one of the following dependencies is in F^+:
  - $R_1 \cap R_2 \rightarrow R_1$
  - $R_1 \cap R_2 \rightarrow R_2$
- The above functional dependencies are a sufficient condition for lossless join decomposition; the dependencies are a necessary condition only if all constraints are functional dependencies

Example

- $R = (A, B, C)$
  $F = \{A \rightarrow B, B \rightarrow C)$
  - Can be decomposed in two different ways
- $R_1 = (A, B)$, $R_2 = (B, C)$
  - Lossless-join decomposition:
    $R_1 \cap R_2 = \{B\}$ and $B \rightarrow BC$
  - Dependency preserving
- $R_1 = (A, B)$, $R_2 = (A, C)$
  - Lossless-join decomposition:
    $R_1 \cap R_2 = \{A\}$ and $A \rightarrow AB$
  - Not dependency preserving
    (cannot check $B \rightarrow C$ without computing $R_1 \bowtie R_2$)

Dependency Preservation

- Let F_i be the set of dependencies F^+ that include only attributes in R_i.
  - A decomposition is **dependency preserving**, if
    $(F_1 \cup F_2 \cup \ldots \cup F_n)^+ = F^+$
  - If it is not, then checking updates for violation of functional dependencies may require computing joins, which is expensive.

Decomposition to Reach BCNF

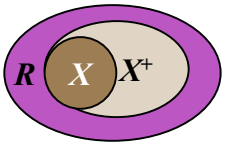
Setting: relation R, given FD's F.

Suppose relation R has BCNF violation $X \rightarrow B$.

- We need only look among FD's of F for a BCNF violation, not those that follow from F.
- Proof: If $Y \rightarrow A$ is a BCNF violation and follows from F, then the computation of Y^+ used at least one FD $X \rightarrow B$ from F.
  - X must be a subset of Y.
  - Thus, if Y is not a superkey, X cannot be a superkey either, and $X \rightarrow B$ is also a BCNF violation.

Algorithm for BCNF

1. Compute X^+.
   - Cannot be all attributes – why?
2. Decompose R into X^+ and $(R–X^+) \cup X$.



3. Find the FD's for the decomposed relations.
   - Project the FD's from F = calculate all consequents of F that involve only attributes from X^+ or only from $(R–X^+) \cup X$.

BCNF Decomposition Algorithm

result := {*R* };
done := false;
compute F^+;
while (not *done)* **do**
  if (there is a schema R_i in *result* that is not in BCNF)
    then begin
      let $\alpha \rightarrow \beta$ be a nontrivial functional dependency that
        holds on R_i such that $\alpha \rightarrow R_i$ is not in F^+,
          and $\alpha \cap \beta = \varnothing$;
        result := (*result* – R_i) $\cup$ (R_i – β) $\cup$ (α, β);
      end
    else *done* := **true;**

Note: each R_i is in BCNF, and decomposition is lossless-join.

Example of BCNF Decomposition

- $R = (A, B, C)$
  $F = \{A \rightarrow B$
  $B \rightarrow C\}$
  Key = $\{A\}$
- R is not in BCNF ($B \rightarrow C$ but B is not superkey)
- Decomposition
  - $R_1 = (B, C)$
  - $R_2 = (A,B)$

Example of BCNF Decomposition

- *class* (*course_id*, *title*, *dept_name*, *credits*, *sec_id*, *semester*, *year*, *building*, *room_number*, *capacity*, *time_slot_id*)
- Functional dependencies:
  - *course_id*→ *title*, *dept_name*, *credits*
  - *building*, *room_number*→*capacity*
  - *course_id*, *sec_id*, *semester*, *year*→*building*, *room_number*, *time_slot_id*
- A candidate key {*course_id*, *sec_id*, *semester*, *year*}.
- BCNF Decomposition:
  - *course_id*→ *title*, *dept_name*, *credits* holds
    - but *course_id* is not a superkey.
  - We replace *class* by:
    - *course*(*course_id*, *title*, *dept_name*, *credits*)
    - *class-1* (*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number, capacity*, *time_slot_id*)

BCNF Decomposition (Cont.)

- *course* is in BCNF
  - How do we know this?
- *building*, *room_number*→*capacity* holds on *class-1*
  - but {*building*, *room_number*} is not a superkey for *class-1*.
  - We replace *class-1* by:
    - *classroom* (*building*, *room_number*, *capacity*)
    - *section* (*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*)
- *classroom* and *section* are in BCNF.

Testing for Dependency Preservation

- To check if a dependency $\alpha \rightarrow \beta$ is preserved in a decomposition of R into $R_1, R_2, \ldots, R_n$ we apply the following test (with attribute closure done with respect to F)
  - *result* = α
    while (changes to *result*) do
      for each R_i in the decomposition
        $t = (result \cap R_i)^+ \cap R_i$
        $result = result \cup t$
  - If *result* contains all attributes in β, then the functional dependency
    $\alpha \rightarrow \beta$ is preserved.
- We apply the test on all dependencies in F to check if a decomposition is dependency preserving
- This procedure takes polynomial time, instead of the exponential time required to compute F^+ and $(F_1 \cup F_2 \cup \ldots \cup F_n)^+$

Example

- $R = (A, B, C)$
  $F = \{A \rightarrow B$
  $B \rightarrow C\}$
  Key = $\{A\}$
- R is not in BCNF
- Decomposition $R_1 = (A, B)$, $R_2 = (B, C)$
  - R_1 and R_2 in BCNF
  - Lossless-join decomposition
  - Dependency preserving

Testing for BCNF

- To check if a non-trivial dependency $\alpha \rightarrow \beta$ causes a violation of BCNF
  1. compute α^+ (the attribute closure of α), and
  2. verify that it includes all attributes of R, that is, it is a superkey of R.
- **Simplified test**: To check if a relation schema R is in BCNF, it suffices to check only the dependencies in the given set F for violation of BCNF, rather than checking all dependencies in F^+.
  - If none of the dependencies in F causes a violation of BCNF, then none of the dependencies in F^+ will cause a violation of BCNF either.
- However, **simplified test using only F is incorrect when testing a relation in a decomposition of R**
  - Consider $R = (A, B, C, D, E)$, with $F = \{ A \rightarrow B, BC \rightarrow D\}$
    - Decompose R into $R_1 = (A,B)$ and $R_2 = (A,C,D, E)$
    - Neither of the dependencies in F contain only attributes from (A,C,D,E) so we might be mislead into thinking R_2 satisfies BCNF.
    - In fact, dependency $AC \rightarrow D$ in F^+ shows R_2 is not in BCNF.

Testing Decomposition for BCNF

- To check if a relation R_i in a decomposition of R is in BCNF,
  - Either test R_i for BCNF with respect to the **restriction** of F to R_i (that is, all FDs in F^+ that contain only attributes from R_i)
  - or use the original set of dependencies F that hold on R, but with the following test:
    - for every set of attributes $\alpha \subseteq R_i$, check that α^+ (the attribute closure of α) either includes no attribute of $R_i - \alpha$, or includes all attributes of R_i.
    - If the condition is violated by some $\alpha \rightarrow \beta$ in F, the dependency
      $\alpha \rightarrow (\alpha^+ - \alpha) \cap R_i$
      can be shown to hold on R_i, and R_i violates BCNF.
    - We use above dependency to decompose R_i

BCNF and Dependency Preservation

It is not always possible to get a BCNF decomposition that is dependency preserving

- $R = (J, K, L)$
  $F = \{JK \rightarrow L$
  $L \rightarrow K\}$
  Two candidate keys = JK and JL
- R is not in BCNF
- Any decomposition of R will fail to preserve
  $$JK \rightarrow L$$
  This implies that testing for $JK \rightarrow L$ requires a join

PURDUE UNIVERSITY

CS34800
Information Systems

Populating a Database

Prof. Chris Clifton

5 October 2016

Indiana Center for Database Systems ™

Four Ways

- Create table from result of query
- Insert query result into a table
- Insert “a tuple at a time”
- “Bulk Loader”

Creating table from query

courses

course_title	instructor_id	room

instructors

instructor_id	name	phone

- CREATE TABLE course_instructors AS
  SELECT course_name, name, phone
  FROM courses, instructors
  WHERE courses.instructor_id = instructors.id;

Populating table from query

courses

course_title	instructor_id	room

instructors

instructor_id	name	phone

- CREATE TABLE course_instructors (
  course_name varchar(30),
  instructor_name varchar(30),
  phone number(10));
- INSERT INTO course_instructors
  SELECT course_name, name, phone
  FROM courses, instructors
  WHERE courses.instructor_id = instructors.id;

Insert tuple-by-tuple

- CREATE TABLE course_instructors (
  course_name varchar(30),
  instructor_name varchar(30),
  phone number(10));
- INSERT INTO course_instructors
  values('Information Systems',
  'Chris Clifton',
  46005) ;

Bulk Loader

- Various utilities, vendor-specific
- Oracle: SQL*Loader (sqlldr)
  - (and others)
  - We won't cover in class
- More generic – create a file of insert statements
  - sqlplus "@filename"

PURDUE
UNIVERSITY

CS34800
Information Systems

Normalization

Prof. Chris Clifton

5 October 2016

How good is BCNF?

- There are database schemas in BCNF that do not seem to be sufficiently normalized
- Consider a relation

  inst_info (ID, child_name, phone)

  - where an instructor may have more than one phone and can have multiple children

ID	*child_name*	*phone*
99999 99999 99999 99999	David David William Willian	512-555-1234 512-555-4321 512-555-1234 512-555-4321

inst_info

How good is BCNF? (Cont.)

- There are no non-trivial functional dependencies and therefore the relation is in BCNF
- Insertion anomalies – i.e., if we add a phone 981-992-3443 to 99999, we need to add two tuples

  (99999, David, 981-992-3443)
  (99999, William, 981-992-3443)

How good is BCNF? (Cont.)

- Therefore, it is better to decompose *inst_info* into:

inst_child

ID	*child_name*
99999	David
99999	David
99999	William
99999	Willian

inst_phone

ID	*phone*
99999	512-555-1234
99999	512-555-4321
99999	512-555-1234
99999	512-555-4321

This suggests the need for higher normal forms, such as Fourth Normal Form (4NF), which we shall see later.

3NF

One FD structure causes problems:

- If you decompose, you can't check all the FD's only in the decomposed relations.
- If you don't decompose, you violate BCNF.

Abstractly: $AB \rightarrow C$ and $C \rightarrow B$.

- Example 1: `title city` $\rightarrow$ `theatre` and `theatre` $\rightarrow$ `city`.
- Example 2: `street city` $\rightarrow$ `zip`, `zip` $\rightarrow$ `city`.

Keys: {*A*, *B*} and {*A*, *C*}, but $C \rightarrow B$ has a left side that is not a superkey.

- Suggests decomposition into *BC* and *AC*.
  - But you can't check the FD $AB \rightarrow C$ in only these relations.

Example

A = `street`, *B* = `city`, *C* = `zip`.

street	zip
545 Tech Sq.	02138
545 Tech Sq.	02139

city	zip
Cambridge	02138
Cambridge	02139

`zip` → `city`

`street city` → `zip`

Join:

city	street	zip
Cambridge	545 Tech Sq.	02138
Cambridge	545 Tech Sq.	02139

"Elegant" Workaround

Define the problem away.

- A relation *R* is in 3NF iff (if and only if) for every nontrivial FD $X \rightarrow A$, either:
  1. *X* is a superkey, or
  2. *A* is *prime* = member of at least one key.
- Thus, the canonical problem goes away: you don't have to decompose because all attributes are prime.

What 3NF Gives You

There are two important properties of a decomposition:

1. We should be able to recover from the decomposed relations the data of the original.
   - Recovery involves projection and join, which we shall defer until we've discussed relational algebra.
2. We should be able to check that the FD's for the original relation are satisfied by checking the projections of those FD's in the decomposed relations.

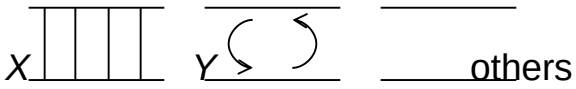
- Without proof, we assert that it is always possible to decompose into BCNF and satisfy (1).
- Also without proof, we can decompose into 3NF and satisfy both (1) and (2).
- But it is not possible to decompose into BNCF and get both (1) and (2).
  - Street-city-zip is an example of this point.

3NF Synthesis

- Given a canonical cover F_C for F
- Schema $S = \varnothing$
- $\forall\ A \rightarrow B \in Fc$
  - If there is no $R_i \in S$ such that $AB \subseteq R_i$
    - $S = S + AB$
- If there is no $R_i \in S$ containing a candidate key for R
  - $S = S$ + (any candidate key for R)

Multivalued Dependencies

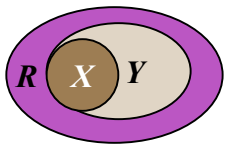
The *multivalued dependency* $X \rightarrow\rightarrow Y$ holds in a relation R if whenever we have two tuples of R that agree in all the attributes of X, then we can swap their Y components and get two new tuples that are also in R.



4NF

Eliminate redundancy due to multiplicative effect of MVD's.

- Roughly: treat MVD's as FD's for decomposition, but not for finding keys.
- Formally: R is in Fourth Normal Form if whenever MVD $X \rightarrow\rightarrow Y$ is *nontrivial* (Y is not a subset of X, and $X \cup Y$ is not all attributes), then X is a superkey.
  - Remember, $X \rightarrow Y$ implies $X \rightarrow\rightarrow Y$, so 4NF is more stringent than BCNF.
- Decompose R, using 4NF violation $X \rightarrow\rightarrow Y$, into XY and $X \cup (R—Y)$.



4NF Decomposition

- Schema $S = R$, $D+$ be the closure of the functional and multivalued dependencies
- While $\exists\ R_i \in S$ not in 4NF w.r.t. $D+$
  - Choose a nontrivial multivalued dependency $A \rightarrow\rightarrow B$ that holds on R_i, where $A \rightarrow R_i \notin D+$, and $A \cap B = \varnothing$
  - $S = (S - R_i) \cup (R_i\text{-}B) \cup (A,B)$

PURDUE UNIVERSITY

CS34800
Information Systems

Normalization

Prof. Chris Clifton

7 October 2016

Scheduling Interviews