Module 27

Partha Pratim Das

Objectives & Outline

Decomposition to 3NF
Test
Algorithm
Practice Problem

Decomposition to BCNF
Test
Algorithm
Practice Problem
Comparison

Module Summary

Database Management Systems

Module 27: Relational Database Design/7: Normal Forms

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ac.in

Module Recap

Module 27

Partha Pratim Das

Objectives & Outline

Decomposition to 3NF
Test
Algorithm
Practice Problem

Decomposition to BCNF
Test
Algorithm
Practice Problem
Comparison

Module Summary

- Studied the Normal Forms and their Importance in Relational Design – how progressive increase of constraints can minimize redundancy in a schema

Module Objectives

Module 27

Partha Pratim Das

Objectives & Outline

Decomposition to 3NF
Test
Algorithm
Practice Problem

Decomposition to BCNF
Test
Algorithm
Practice Problem
Comparison

Module Summary

- To Learn the Decomposition Algorithm for a Relation to 3NF
- To Learn the Decomposition Algorithm for a Relation to BCNF

Module Outline

Module 27

Partha Pratim Das

Objectives & Outline

Decomposition to 3NF
Test
Algorithm
Practice Problem

Decomposition to BCNF
Test
Algorithm
Practice Problem
Comparison

Module Summary

- Decomposition to 3NF
- Decomposition to BCNF

Module 27

Partha Pratim Das

Objectives & Outline

Decomposition to 3NF
- Test
- Algorithm
- Practice Problem

Decomposition to BCNF
- Test
- Algorithm
- Practice Problem
- Comparison

Module Summary

Decomposition to 3NF

3NF Decomposition: Motivation

Module 27

Partha Pratim Das

Objectives & Outline

Decomposition to 3NF
Test
Algorithm
Practice Problem

Decomposition to BCNF
Test
Algorithm
Practice Problem
Comparison

Module Summary

- There are some situations where
  - BCNF is not dependency preserving, and
  - Efficient checking for FD violation on updates is important
- Solution: define a weaker normal form, called Third Normal Form (3NF)
  - Allows some redundancy (with resultant problems; as seen above)
  - But functional dependencies can be checked on individual relations without computing a join
  - **There is always a lossless-join, dependency-preserving decomposition into 3NF**

3NF Decomposition (2): 3NF Definition

Module 27
Partha Pratim Das
Objectives & Outline
Decomposition to 3NF
Test
Algorithm
Practice Problem
Decomposition to BCNF
Test
Algorithm
Practice Problem
Comparison
Module Summary

- A relational schema $\boldsymbol{R}$ is in 3NF if for every FD $\boldsymbol{X} \rightarrow \boldsymbol{A}$ associated with R either
  - $\boldsymbol{A} \subseteq \boldsymbol{X}$ (that is, the FD is trivial) or
  - $\boldsymbol{X}$ is a superkey of $\boldsymbol{R}$ or
  - $\boldsymbol{A}$ is part of some candidate key (not just superkey!)
- **A relation in 3NF is naturally in 2NF**

3NF Decomposition (3): Testing for 3NF

Module 27

Partha Pratim Das

Objectives & Outline

Decomposition to 3NF
Test
Algorithm
Practice Problem

Decomposition to BCNF
Test
Algorithm
Practice Problem
Comparison

Module Summary

- Optimization: Need to check only FDs in F, need not check all FDs in F^+.
- Use attribute closure to check for each dependency $\alpha \to \beta$, if α is a superkey.
- If α is not a superkey, we have to verify if each attribute in β is contained in a candidate key of R
  - This test is rather more expensive, since it involve finding candidate keys
  - **Testing for 3NF has been shown to be NP-hard**
  - **Decomposition into 3NF can be done in polynomial time**

3NF Decomposition (4): Algorithm

Module 27

Partha Pratim Das

Objectives & Outline

Decomposition to 3NF
Test
Algorithm
Practice Problem

Decomposition to BCNF
Test
Algorithm
Practice Problem
Comparison

Module Summary

- Given: relation R, set F of functional dependencies
- Find: decomposition of R into a set of 3NF relation R_i
- Algorithm:
  a) Eliminate redundant FDs, resulting in a canonical cover F_c of F
  b) Create a relation $R_i = XY$ for each FD $X \rightarrow Y$ in F_c
  c) If the key K of R does not occur in any relation R_i, create one more relation $R_i = K$

3NF Decomposition (5): Algorithm

Let F_c be a canonical cover for F;
$i := 0$;
for each functional dependency $\alpha \to \beta$ in F_c **do**
  if none of the schemas $R_j, 1 \leq j \leq i$ contains $\alpha\beta$
    then begin
      $i := i + 1$;
      $R_i := \alpha\beta$
    end
if none of the schemas $R_j, 1 \leq j \leq i$ contains a candidate key for R
  then begin
      $i := i + 1$;
      $R_i :=$ any candidate key for R;
  end
/* Optionally, remove redundant relations */
repeat
if any schema R_j is contained in another schema R_k
  then /* delete R_j */
    $R_j = R$;
    $i = i - 1$;
return $(R_1, R_2, \cdots, R_i)$

3NF Decomposition (6): Algorithm

- Upon decomposition:
  - Each relation schema R_i is in 3NF
  - Decomposition is
    - Dependency Preserving
    - Lossless Join
- Prove these properties

3NF Decomposition (7): Example

- Relation schema:
  $cust\_banker\_branch = (\underline{customer\_id}, \underline{employee\_id}, branch\_name, type)$
- The functional dependencies for this relation schema are:
  a) $customer\_id, employee\_id \rightarrow branch\_name, type$
  b) $employee\_id \rightarrow branch\_name$
  c) $customer\_id, branch\_name \rightarrow employee\_id$
- We first compute a canonical cover
  - $branch\_name$ is extraneous in the RHS of the 1st dependency
  - No other attribute is extraneous, so we get $F_c =$

    $customer\_id, employee\_id \rightarrow type$
    $employee\_id \rightarrow branch\_name$
    $customer\_id, branch\_name \rightarrow employee\_id$

3NF Decomposition (8): Example

- The **for** loop generates following 3NF schema:

  $(\underline{customer\_id}, \underline{employee\_id}, type)$
  $(\underline{employee\_id}, branch\_name)$
  $(\underline{customer\_id}, \underline{branch\_name}, employee\_id)$

  - Observe that $(customer\_id, employee\_id, type)$ contains a candidate key of the original schema, so no further relation schema needs be added
- At end of for loop, detect and delete schemas, such as $(\underline{employee\_id}, branch\_name)$, which are subsets of other schemas
  - result will not depend on the order in which FDs are considered
- The resultant simplified 3NF schema is:

  $(customer\_id, employee\_id, type)$
  $(customer\_id, branch\_name, employee\_id)$

Practice Problem for 3NF Decomposition (1)

Module 27

Partha Pratim Das

Objectives & Outline

Decomposition to 3NF

Test

Algorithm

Practice Problem

Decomposition to BCNF

Test

Algorithm

Practice Problem

Comparison

Module Summary

- $R = ABCDEFGH$
- FDs $= \{A \rightarrow B, ABCD \rightarrow E, EF \rightarrow GH, ACDF \rightarrow EG\}$

Solution is given in the next slide (hidden from presentation – check after you have solved

Practice Problem for 3NF Decomposition (2)

Module 27

Partha Pratim Das

Objectives & Outline

Decomposition to 3NF
Test
Algorithm
Practice Problem

Decomposition to BCNF
Test
Algorithm
Practice Problem
Comparison

Module Summary

- $R = CSJDPQV$
- FDs $= \{C \rightarrow CSJDPQV, SD \rightarrow P, JP \rightarrow C, J \rightarrow S\}$

Solution is given in the next slide (hidden from presentation – check after you have solved)

Practice Problem for 3NF Decomposition (3)

Decompose the following schema to 3NF in the following steps

- Compute all keys for R
- Compute a Canonical Cover F_c for F Put the FDs into alphabetical order.
- Using F_c, employ the 3NFdecom algorithm to obtain a lossless and dependency preserving decomposition of relation R into a collection of relations that are in 3NF
- Does your schema allow redundancy?

- $R(ABCDEFGH)$:
  $F = \{A \rightarrow CD, ACF \rightarrow G, AD \rightarrow BEF, BCG \rightarrow D, CF \rightarrow AH, CH \rightarrow G, D \rightarrow B, H \rightarrow DEG\}$
- $R(ABCDE)$:
  $F = \{A \rightarrow B, A \rightarrow C, C \rightarrow D, A \rightarrow E\}$
- $R(ABCDE)$:
  $F = \{A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A\}$
- $R(ABCD)$:
  $F = \{A \rightarrow D, AB \rightarrow C, AD \rightarrow C, B \rightarrow C, D \rightarrow AB\}$

Module 27

Partha Pratim Das

Objectives & Outline

Decomposition to 3NF
Test
Algorithm
Practice Problem

Decomposition to BCNF
Test
Algorithm
Practice Problem
Comparison

Module Summary

Decomposition to BCNF

BCNF Decomposition: BCNF Definition

Module 27

Partha Pratim Das

Objectives & Outline

Decomposition to 3NF
Test
Algorithm
Practice Problem

Decomposition to BCNF
Test
Algorithm
Practice Problem
Comparison

Module Summary

- A relation schema R is in BCNF with respect to a set F of FDs if for all FDs 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 (that is, $\beta \subseteq \alpha$)
  - α is a superkey for R

BCNF Decomposition (2): Testing for BCNF

Module 27

Partha Pratim Das

Objectives & Outline

Decomposition to 3NF
Test
Algorithm
Practice Problem

Decomposition to BCNF
Test
Algorithm
Practice Problem
Comparison

Module Summary

- To check if a non-trivial dependency $\alpha \rightarrow \beta$ causes a violation of BCNF
  a) Compute α^+ (the attribute closure of α), and
  b) 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.

BCNF Decomposition (3): Testing for BCNF Decomposition

- 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 Ri

BCNF Decomposition (4): Testing Dependency Preservation: Using Closure Set of FD (Exp. Algo.): Module 25

Consider the example given below, we will apply both the algorithms to check dependency preservation and will discuss the results.

- **R** (A, B, C, D)
  F $= \{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A\}$
- Decomposition: **R1**(A, B) **R2**(B, C) **R3**(C, D)
  - $A \rightarrow B$ is preserved on table R1
  - $B \rightarrow C$ is preserved on table R2
  - $C \rightarrow D$ is preserved on table R3
  - We have to check whether the one remaining FD: **D→A** is preserved or not.

R1	R2	R3
$F_1=\{\mathbf{A} \rightarrow AB, \mathbf{B} \rightarrow BA\}$	$F_2=\{\mathbf{B} \rightarrow BC, \mathbf{C} \rightarrow CB\}$	$F_3=\{\mathbf{C} \rightarrow CD, \mathbf{D} \rightarrow DC\}$

- $F' = F_1 \cup F_2 \cup F_3$.
- Checking for: $\mathbf{D} \rightarrow A$ in F'^{+}
  - $D \rightarrow C$ (from R3), $C \rightarrow B$ (from R2), $B \rightarrow A$ (from R1) : **D→ A** (By Transitivity)
    Hence all dependencies are preserved.

BCNF Decomposition (4): Testing Dependency Preservation: Using Closure of Attributes (Poly. Algo.): Module 25

Module 27

Partha Pratim Das

Objectives & Outline

Decomposition to 3NF
Test
Algorithm
Practice Problem

Decomposition to BCNF
Test
Algorithm
Practice Problem
Comparison

Module Summary

- $R(ABCD) :. \ F = \{A \to B, B \to C, C \to D, D \to A\}$
- $Decomp = \{AB, BC, CD\}$
- On projections:

R1	R2	R3
F1 A → B	F2 B → C	F3 C → D

In this algo F1, F2, F3 are not the closure sets, rather the set of dependencies directly applicable on R1, R2, R3 respectively.

- Need to check for: $A \to B, B \to C, C \to D, \boldsymbol{D \to A}$
- $(D)+/F1 = D$. $(D)+/F2 = D$. $(D)+/F3 = D$. So, $\boldsymbol{D \to A}$ could not be preserved.
- In the previous method we saw the dependency was preserved. In reality also it is preserved. Therefore the polynomial time algorithm may not work in case of all examples. To prove preservation Algo 2 is sufficient but not necessary whereas Algo 1 is both sufficient as well as necessary.

Note: This difference in result can occur in any example where a functional dependency of one decomposed table uses another functional dependency in its closure which is not applicable on any of the decomposed table because of absence of all attributes in the table.

BCNF Decomposition (4): Algorithm

Module 27

Partha Pratim Das

Objectives & Outline

Decomposition to 3NF
Test
Algorithm
Practice Problem

Decomposition to BCNF
Test
Algorithm
Practice Problem
Comparison

Module Summary

a) For all dependencies $A \rightarrow B$ in F^+, check if A is a superkey
   - By using attribute closure

b) If not, then
   - Choose a dependency in F^+ that breaks the BCNF rules, say $A \rightarrow B$
   - Create $R1 = AB$
   - Create $R2 = (R - (B - A))$
   - Note that: $R1 \cap R2 = A$ and $A \rightarrow AB$ $(= R1)$, so this is lossless decomposition

c) Repeat for $R1$, and $R2$
   - By defining $F1^+$ to be all dependencies in F that contain only attributes in $R1$
   - Similarly $F2^+$

BCNF Decomposition (5): Algorithm

Module 27

Partha Pratim Das

Objectives & Outline

Decomposition to 3NF
Test
Algorithm
Practice Problem

Decomposition to BCNF
Test
Algorithm
Practice Problem
Comparison

Module Summary

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 $\alpha \rightarrow \beta$ is not in F^+,
        and $\alpha \cap \beta = \phi$;
      result $:= (result - R_i) \cup (R_i - \beta) \cup (\alpha, \beta)$;
    end
  else *done* $:=$ *true*;

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

BCNF Decomposition (6): Example

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

BCNF Decomposition (7): Example

Module 27

Partha Pratim Das

Objectives & Outline

Decomposition to 3NF

Test

Algorithm

Practice Problem

Decomposition to BCNF

Test

Algorithm

Practice Problem

Comparison

Module Summary

- *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 (8): Example

- *course* is in BCNF
  - How do we know this?
- *building, room_number* → *capacity* holds on
  class-1(course_id, sec_id, semester, year, building, room_number, capacity, time_slot_id)
  - 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.

BCNF Decomposition (8): Dependency Preservation

Module 27

Partha Pratim Das

Objectives & Outline

Decomposition to 3NF
- Test
- Algorithm
- Practice Problem

Decomposition to BCNF
- Test
- Algorithm
- Practice Problem
- Comparison

Module Summary

- 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

Practice Problem for BCNF Decomposition

Module 27
Partha Pratim Das
Objectives & Outline
Decomposition to 3NF
Test
Algorithm
Practice Problem
Decomposition to BCNF
Test
Algorithm
Practice Problem
Comparison
Module Summary

Decompose the following schema to BCNF

- $R = ABCDE$. $F = \{A \rightarrow B, BC \rightarrow D\}$
- $R = ABCDEH$. $F = \{A \rightarrow BC, E \rightarrow HA\}$
- $R = CSJDPQV$. $F = \{C \rightarrow CSJDPQV, SD \rightarrow P, JP \rightarrow C, J \rightarrow S\}$
- $R = ABCD$. $F = \{C \rightarrow D, C \rightarrow A, B \rightarrow C\}$

Comparison of BCNF and 3NF

Module 27

Partha Pratim Das

Objectives & Outline

Decomposition to 3NF
Test
Algorithm
Practice Problem

Decomposition to BCNF
Test
Algorithm
Practice Problem
Comparison

Module Summary

- It is always possible to decompose a relation into a set of relations that are in 3NF such that:
  - the decomposition is lossless
  - the dependencies are preserved
- It is always possible to decompose a relation into a set of relations that are in BCNF such that:
  - the decomposition is lossless
  - it may not be possible to preserve dependencies.

S#	3NF	BCNF
1.	It concentrates on Primary Key	It concentrates on Candidate Key
2.	Redundancy is high as compared to BCNF	0% redundancy
3.	It preserves all the dependencies	It may not preserve the dependencies
4.	A dependency $X \rightarrow Y$ is allowed in 3NF if X is a super key or Y is a part of some key	A dependency $X \rightarrow Y$ is allowed if X is a super key

Module Summary

Module 27

Partha Pratim Das

Objectives & Outline

Decomposition to 3NF
Test
Algorithm
Practice Problem

Decomposition to BCNF
Test
Algorithm
Practice Problem
Comparison

Module Summary

- Learnt how to decompose a schema into 3NF while preserving dependency and lossless join
- Learnt how to decompose a schema into BCNF with lossless join

Slides used in this presentation are borrowed from http://db-book.com/ with kind permission of the authors.
Edited and new slides are marked with "PPD".