7.5 Algorithms for Decomposition

- BCNF decomposition
- 3NF decomposition

7.5.1 BCNF Decomposition

- A decomposition algorithm that gives *lossless* schema in BCNF is shown in Fig.7.11
  - $\boldsymbol{F^+}$ is computed at first to <u>determine whether or not α is the superkey of subschema $\boldsymbol{R_i}$</u>
  - if ***other*** methods can be used to determine whether or not α is the superkey of subschema $\boldsymbol{R_i}$, then $\boldsymbol{F^+}$ need not be computed

- An alternative easily-understood and straight-forward description of BCNF decomposition is given below

Given: ***R***, ***F*** holding on R

result := {R};
done := false;

找出non-BCNF子模式 R_i，进一步分解

R_i中分解为2部分—公共属性α，其中α → β组成单独的BCNF模式

while (not *done*) **do**

if (there is a non-BCNF subschema R_i in ***result***)

then begin

let $\alpha \to \beta$ be a nontrivial functional dependency that holds on R_i, such that

(1) α is not the superkey for R_i, i.e. $\alpha \to \beta$ is not in F^+, with respect to the restriction F_i of ***F*** to R_i *!!!*

(2) $\alpha \cap \beta = \varnothing$; /* α → β 中无冗余属性*/

result **:=(*result*** – R_i) ∪ {($R_i - \beta$)} ∪{(α, β)};

end

else *done* := **true;**

R_i is non-BCNF because of $\alpha \to \beta$

α remains in R_i

BCNF Decomposition (cont.)

- Note
  - **to determine whether or not R_i is in BCNF, the restriction of F to R_i and the candidate keys of R_i should be computed !!!**
  - R_i is not in BCNF, because with respect to $\alpha \rightarrow \beta$ that holds on R_i, α is not the super key of R_i
  - the algorithm replaces *non-BCNF* R_i *with* **$(R_i - \beta)$ and (α, β)**
    - *non-BCNF* R_i is decomposed into $(R_i - \beta)$ and BCNF **(α, β)**
  - the restriction of F to the schema (α, β) is $\alpha \rightarrow \beta$ holds, α is the superkey for (α, β), so (α, β) is in BCNF

Example 1

- $R = (A, B, C)$
  $F = \{A \rightarrow B$
  $\quad B \rightarrow C\}$
  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 2

- *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.

Example 3.
BCNF and Dependency Preservation

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

- $R = (J, K, L)$
  $F = \{JK \rightarrow L$
  $\qquad L \rightarrow K\}$
  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

Example 4
BCNF Decomposition

- Considering the schema ***R***(C, T, H, R, S, G),

and ***F***={CS →G, C →T, TH →R, HR →C, HS →R}

, give a decomposition of ***R*** into BCNF

- Step1. With respect to ***R*** and ***F***, the unique candidate key is HS,

  and ***R*** is not in BCNF

- Step2. Initially, ***result***:= {***R***}={*CTHRSG*}
  - ***R*** is not in BCNF, because there is CS →G in ***F***, and CS is not the candidate key of ***R***
  - ***R***(*CTHRSG*) is decomposed into $\boldsymbol{R_1}$(*CSG*) and $\boldsymbol{R_2}$(*CTHRS*)
  - $\boldsymbol{R_1}$(*CSG*) is in BCNF, F_1= {CS →G}

Example 4
BCNF Decomposition (cont.)

- Step3. With respect to $\boldsymbol{R_2}(CTHRS)$
  - !!!the restriction of ***F*** to $\boldsymbol{R_2}(CTHRS)$ is

    $\boldsymbol{F}_2=\{C \rightarrow T, TH \rightarrow R, HR \rightarrow C, HS \rightarrow R\}$
  - !!the candidate key is HS
  - $\boldsymbol{R_2}(CTHRS)$ is not in BCNF, because there is $C \rightarrow T$ in $\boldsymbol{F}_2$, and C is not the candidate key of $\boldsymbol{R_2}$
  - $\boldsymbol{R_2}(CTHRS)$ is decomposed into $\boldsymbol{R_{21}}(CT)$ and $\boldsymbol{R_{22}}(CHRS)$
  - $\boldsymbol{R_{21}}(CT)$ is in BCNF , $F_{21}= \{C \rightarrow T\}$

Example 4
BCNF Decomposition (cont.)

- Step4. With respect to $\boldsymbol{R_{22}}(CHRS)$
  - !!the restriction of $\boldsymbol{F}$ to $\boldsymbol{R_{22}}(CHRS)$ is

    $\boldsymbol{F}_{22}$={HR $\rightarrow$C, HS $\rightarrow$R, CH $\rightarrow$ R} /* CH $\rightarrow$TH, TH $\rightarrow$R
  - !!the candidate key is HS
  - $\boldsymbol{R_{22}}(CHRS)$ is not in BCNF, because there is HR $\rightarrow$C in $\boldsymbol{F}_{22}$, and HR is not candidate key of $\boldsymbol{R_{22}}$
  - $\boldsymbol{R_{22}}(CHRS)$ is decomposed into $\boldsymbol{R_{221}}(HRC)$ and $\boldsymbol{R_{222}}(HRS)$
  - $\boldsymbol{R_{221}}(HRC)$ is in BCNF
- Step5. With respect to $\boldsymbol{R_{222}}(HRS)$,
  - the restriction of $\boldsymbol{F}$ to $\boldsymbol{R_{222}}(HRS)$ is

    $\boldsymbol{F}_{222}$={HS $\rightarrow$R}, and the candidate key is HS
  - $\boldsymbol{R_{222}}(HRS)$ is in BCNF

Example 4
BCNF Decomposition (cont.)

- Finally, the BCNF decomposition of ***R***(C, T, H, R, S, G) is
  - $\boldsymbol{R_1}(CSG)$, $\boldsymbol{R_{21}}(CT)$, $\boldsymbol{R_{221}}(HRC)$, $\boldsymbol{R_{222}}(HRS)$

7.5.2 3NF and Its Decomposition

Third Normal Form: Motivation

- 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; we will see examples later)
  - 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 Example

- Relation *dept_advisor*:
  - *dept_advisor* (s_ID, i_ID, $dept\_name$)
    $F = \{s\_ID, dept\_name \rightarrow i\_ID,\ \ i\_ID \rightarrow dept\_name\}$
  - Two candidate keys: s_ID, $dept\_name$, and i_ID, s_ID
  - R is in 3NF
    - $s\_ID, dept\_name \rightarrow i\_ID$
      - s_ID, $dept\_name$ is a superkey
    - $i\_ID \rightarrow dept\_name$
      - $dept\_name$ is contained in a candidate key

Redundancy in 3NF

- There is some redundancy in this schema
- Example of problems due to redundancy in 3NF
  - $R = (J, K, L)$
    $F = \{JK \rightarrow L, L \rightarrow K\}$

J	L	K
j_1	l_1	k_1
j_2	l_1	k_1
j_3	l_1	k_1
null	l_2	k_2

repetition of information (e.g., the relationship l_1, k_1)

- (*i_ID, dept_name*)

Testing for 3NF

- Optimization: Need to check only FDs in F, need not check all FDs in F^+.
- Use attribute closure to check for each dependency $\alpha \rightarrow \beta$, 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
  - Interestingly, decomposition into third normal form (described shortly) can be done in polynomial time

3NF Decomposition

- A decomposition algorithm that gives *lossless* and *dependency preserving* schemas in 3NF is shown in Fig.7.12

- Note:
  - ***all*** candidate keys should be founded out
  - ***only one*** $\boldsymbol{F_c}$ should be computed *at first*

0. Find out *all candidate keys* for R

求最简FD:
去除掉传递依
赖和冗余属性

1. Find out **a** canonical cover $\boldsymbol{F_c}$ for F;

i := 0; /*R_0 is null*/

2. for each functional dependency $\alpha \rightarrow \beta$ in $\boldsymbol{F_c}$ **do**

if *none of the schemas* R_j, $1 \le j \le i$ *contains* $\alpha\beta$

then begin

$i := i + 1;$

$R_i := \alpha\beta$

end

对$\boldsymbol{F_c}$中<u>*每一个未包括在已有子模式*</u>中的函数依赖，构造一个BCNF/3NF子模式

3. if none of the schemas R_j, $1 \le j \le i$ contains a candidate key for R

then begin

$i := i + 1;$

R_i := any candidate key for R;

end

对R的候选健单独构造一个关系模式

4. return $(R_1, R_2, \ldots, R_i)$

Recap:
Computing of Candidate Keys

- 定理。如果属性A只出现在F中函数依赖的左部，则A一定是主属性，必然出现在候选键中

- 输入：关系模式R及其函数依赖集F
- 输出：R的所有候选关键字
- 方法：
  - step1.将R的所有属性分为四类：
    - L类：仅出现在F中函数依赖左部的属性
    - R类：仅出现在F中函数依赖右部的属性
    - N类：在F中函数依赖左右两边均未出现的属性
    - LR类：在F中函数依赖左右两边均出现的属性

Computing of Candidate Keys (cont.)

- 并令X_set代表L、N类，Y_set代表LR类；

- step2. 求X_set $^{+}$，若包含了R的所有属性，则X_set即为R的唯一候选关键字，转step5;

  否则转step3
- step3. 在Y_set中取一属性A，求（X _set ∪ A)$^{+}$,若它包含了R的所有属性，则转step4;

  否则，调换一属性反复进行这一过程，直到试完所有Y_set中的属性
- step4. 如果已找出所有的候选关键字，则转step5;

  否则，在Y_set中依此取两个、三个，…，求他们的属性闭包，直到其闭包包含R的所有的属性
- step5. 停止，输出结果

Computing of Candidate Keys (cont.)

- E.g.1. R(X, Y, Z, W), and ***F*** = {X→Z, Z→X, Y→XZ, W→XZ}
  - L类：Y, W
  - R、N: 空
  - LR: X, Z
  - X_set={Y, W}
  - Y_set={X, Z}
  - $(YW)^+$=R, so YW is the candidate key

Computing of Candidate Keys (cont.)

- E.g.2. R(B, D, I, O, Q, S), and

  $\boldsymbol{F} = \{B \to Q, I \to S, IS \to Q, S \to D\}$

  - L类： B, I
  - R类： D, Q
  - N类: O
  - LR类: S
  - X_set={B, I, O}
  - Y_set={S}
  - $(BIO)^+=R$, so BIO is a candidate key

Example 1
3NF Decomposition

- Consider schema R(X, Y, Z, W), and $\boldsymbol{F} = \{X \to Z, Z \to X, Y \to XZ, W \to XZ\}$ that holds on R. Give the lossless, dependency-preserving decomposition of this schema into 3NF

The solution is as follows:

- Step1. $\{YW\}^+ = R$, {YW} is the unique candidate key
- Step2. With respect to $F = \{X \to Z, Z \to X, Y \to XZ, W \to XZ\}$,
  - considering X in $Y \to XZ$ and Z in $W \to XZ$,

    $\{X \to Z, Z \to X, Y \to Z, W \to X\}$ implies $\boldsymbol{F}$, so X in $Y \to XZ$ and Z in $W \to XZ$ are all extraneous attributes, and one of ***canonical covers*** for $\boldsymbol{F}$ is

    $$F_c = \{X \to Z, Z \to X, Y \to Z, W \to X\}$$

Example 1
3NF Decomposition (cont.)

- the other ***canonical covers*** for ***F*** are

  $F_c = \{X \to Z, Z \to X, Y \to X, W \to X\}$, or

  $F_c = \{X \to Z, Z \to X, Y \to X, W \to Z\}$, or

  $F_c = \{X \to Z, Z \to X, Y \to Z, W \to Z\}$

- Step3. As far as $F_c = \{X \to Z, Z \to X, Y \to Z, W \to X\}$ is concerned,
  - on the basis of 3NF decomposition algorithm, the subschema responding to each functional dependency in F_c are

    $R_1=\{XZ\}$， $R_2=\{YZ\}$, $R_3=\{XW\}$
  - for the candidate key {YW}, $R_4=\{YW\}$is generated, so ***one of*** the 3NF decomposition is

    { XZ， YZ， XW， YW}

Example 1
3NF Decomposition (cont.)

- With respect to the other ***canonical covers***

  $F_c = \{X \rightarrow Z, Z \rightarrow X, Y \rightarrow X, W \rightarrow X\}$, or

  $F_c = \{X \rightarrow Z, Z \rightarrow X, Y \rightarrow X, W \rightarrow Z\}$, or

  $F_c = \{X \rightarrow Z, Z \rightarrow X, Y \rightarrow Z, W \rightarrow Z\}$

  and the candidate key {YW},

  the corresponding 3NF decomposition are

  { XZ, YX, WX, YW}

  { XZ, YX, WZ, YW}

  { XZ, YZ, WZ, YW}

Example 2
3NF Decomposition (cont.)

- Assume there is a schema S (Sno, Cno, Tname, Taddr, Grade).
  - each student has only one grade for each course he/she took;
  - each course is taught by only one teacher;
  - each teacher has only one address. (Suppose duplication of name is not allowed for teachers)

  1. Write down three instances of functional dependency of the schema S mentioned above.

     (Sno， Cno) →Grade

     Cno→Tname

     Tname→Taddr

Example 2
3NF Decomposition (cont.)

- 2. Write down the candidate keys of schema S.

  Sno，Cno

  由于$\{Sno, Cno\}^+ = (Sno, Cno, Tname, Taddr, Grade)$

- 3. Decompose the schema S into the third normal form.

  S1(Sno，Cno，Grade)

  S2(Cno，Tname)

  S3(Tname，Taddr)

7.5.4 Comparison of BCNF and 3NF

- 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.

Design Goals

- Goal for a relational database design is:
  - BCNF.
  - Lossless join.
  - Dependency preservation.
- If we cannot achieve this, we accept one of
  - Lack of dependency preservation
  - Redundancy due to use of 3NF

北京邮电大学

Appendix B 本章习题类型

- 给定关系表***r*(*R*)**和若干函数依赖$\alpha \rightarrow \beta$, 判断**r** 是否满足$\alpha \rightarrow \beta$
- 判断关于函数依赖的一些公式是否成立
  - 例如, 如果$A \rightarrow B$, $B \rightarrow C$, 则$A \rightarrow C$

  （根据Amstron公理系统）

- 根据文字描述，抽象出函数依赖关系
- **求候选健(算法)**
- 计算属性闭包α^+
- 计算函数依赖集F的最小正则集F_c

Appendix G 本章习题类型 (续)

- **给定关系模式R和定义在R上的函数依赖集F，判断R属于第几范式，为什么？**
  - 只考虑1NF, 2NF, 3NF, BCNF
- 判断一个模式分解是否为无损连接、函数依赖保持
- **给定非3NF的关系模式R和定义在R上的函数依赖集F，将R分解为第三范式**
- 给定非BCNF的关系模式R和定义在R上的函数依赖集F，将R分解为BCNF范式

Homework

7.6 Compute the closure of the following set F of functional dependencies for relation schema $R = (A, B, C, D, E)$.

$$A \rightarrow BC$$
$$CD \rightarrow E$$
$$B \rightarrow D$$
$$E \rightarrow A$$

List the candidate keys for R.

7.7 Using the functional dependencies of Exercise 7.6, compute the canonical cover F_c.

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

7.22 Give a lossless, dependency-preserving decomposition into 3NF of schema R of Exercise 7.1.

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

Homework

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

$$A \rightarrow BCD$$
$$BC \rightarrow DE$$
$$B \rightarrow D$$
$$D \rightarrow A$$

a. Compute B^+.

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

c. Compute a canonical cover for this set of functional dependencies F; give each step of your derivation with an explanation.

d. Give a 3NF decomposition of the given schema based on a canonical cover.

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

Homework

7.32 Consider the schema $R = (A, B, C, D, E, G)$ and the set F of functional dependencies:

$$A \rightarrow BC$$
$$BD \rightarrow E$$
$$CD \rightarrow AB$$

a. Find a nontrivial functional dependency containing no extraneous attributes that is logically implied by the above three dependencies and explain how you found it.

b. Use the BCNF decomposition algorithm to find a BCNF decomposition of R. Start with $A \rightarrow BC$. Explain your steps.

c. For your decomposition, state whether it is lossless and explain why.

d. For your decomposition, state whether it is dependency preserving and explain why.

Homework

7.33 Consider the schema $R = (A, B, C, D, E, G)$ and the set F of functional dependencies:

$$AB \rightarrow CD$$
$$ADE \rightarrow GDE$$
$$B \rightarrow GC$$
$$G \rightarrow DE$$

Use the 3NF decomposition algorithm to generate a 3NF decomposition of R, and show your work. This means:

a. A list of all candidate keys

b. A canonical cover for F, along with an explanation of the steps you took to generate it

c. The remaining steps of the algorithm, with explanation

d. The final decomposition

Have a break