Relational Database Design

Chapter 15 in 6th Edition

10 Relational Database Design

Anomalies can be removed from relation designs by decomposing them until they are in a normal form.

Several problems should be investigated regarding a decomposition.

A decomposition of a relation scheme, R, is a set of relation schemes $\{R_1, . . . ,R_n\}$ such that $R_i \subseteq R$ for each i, and $\bigcup_{i=1}^{n} R_i = R$

Note that in a decomposition $\{R_1, . . . ,R_n\}$, the intersect of each pair of R_i and R_j does not have to be empty.

Example: R = {A, B, C, D, E}, R_1 = { A, B}, R_2 ={A, C}, R_3 = {C, D, E}

A naïve decomposition: each relation has only attribute.

A good decomposition should have the following two properties.

Dependency Preserving

- Definition: Two sets F and G of FD's are equivalent if $F^+ = G^+$.
- Given a decomposition $\{R_1, . . . ,R_n\}$ of R:

$$F_i = \{X \rightarrow Y : X \rightarrow Y \in F \& X \in R_i, Y \in R_i\}.$$

- The decomposition $\{R_1, . . . ,R_n\}$ of R is dependency preserving with respect to F if

$$F^+ = \left(\bigcup_{i=1}^{i=n} F_i\right)^+.$$

Examples

F = { A → BC, D → EG, M → A }, R = (A, B, C, D, E, G, M, A)
1) Given R_1= (A, B, C, M) and R_2 = (C, D, E, G),
F_1 = { A → BC, M → A}, F_2 = {D → EG}
F = F_1 U F_2. thus, dependency preserving

2) Suppose that F' = F U {M → D}. R_1 and R_2 remain the same.
Thus, F_1 and F_2 remain the same.
We need to verify if M→D is inferred by F_1 U F_2.
Since $M^+ |_{F1\ U\ F2}$ = {M, A, B, C}, M→D is not inferred by F_1 U F_2.
Thus, R_1 and R_2 are not dependency preserving regarding F'.

3) F'' = {A → BC, D → EG, M → A , M→C, C→ D, M→ D}
F_1 = {A → BC, M→ A, M→ C}, F_2 = {D→ EG, C→ D}
It can be verified that M→ D is inferred by F_1 and F_2.
Thus, $F''^+ = (F_1 \text{ U } F_2)^+$
Hence, R_1 and R_2 are dependency preserving regarding F''.

Lossless Join Decomposition

- A second necessary property for decomposition:
- A decomposition $\{R_1, . . . ,R_n\}$ of R is a *lossless join* decomposition with respect to a set F of FD's if for every relation instance r that satisfies F:

$$r = \pi R_1(r) \bowtie \cdots \bowtie \pi R_n(r).$$

If $r \subset \pi R_1(r) \bowtie \cdots \bowtie \pi R_n(r)$, the decomposition is *lossy.*

Lossless Join Decomposition(cont)

Example 2:
Suppose that we decompose the following relation:

STUDENT_ADVISOR

Name	Department	Advisor
Jones	Comp Sci	Smith
Ng	Chemistry	Turner
Martin	Physics	Bosky
Dulles	Decision Sci	Hall
Duke	Mathematics	James
James	Comp Sci	Clark
Evan	Comp Sci	Smith
Baxter	English	Bronte

With dependencies $\{Name \rightarrow Department, Name \rightarrow Advisor, Advisor \rightarrow Department\}$, into two relations:

Lossless Join Decomposition(cont)

STUDENT_DEPARTMENT

Name	Department
Jones	Comp Sci
Ng	Chemistry
Martin	Physics
Duke	Mathematics
Dulles	Decision Sci
James	Comp Sci
Evan	Comp Sci
Baxter	English

DEPARTMENT_ADVISOR

Department	Advisor
Comp Sci	Smith
Chemistry	Turner
Physics	Bosky
Decision Sci	Hall
Mathematics	James
Comp Sci	Clark
English	Bronte

If we join these decomposed relations we get:

Lossless Join Decomposition(cont)

Name	Department	Advisor
Jones	Comp Sci	Smith
Jones	Comp Sci	Clark*
Ng	Chemistry	Turner
Martin	Physics	Bosky
Dulles	Decision Sci	Hall
Duke	Mathematics	James
James	Comp Sci	Smith*
James	Comp Sci	Clark
Evan	Comp Sci	Smith
Evan	Comp Sci	Clark*
Baxter	English	Bronte

- This is not the same as the original relation (the tuples marked with * have been added). Thus the decomposition is <u>lossy</u>.
- Useful theorem: The decomposition $\{R_1,R_2\}$ of R is lossless iff the common attributes $R_1 \cap R_2$ form a superkey for either R_1 or R_2.

Lossless Join Decomposition(cont)

- *Example 3*: Given $R(A,B,C)$ and $F = \{A \rightarrow B\}$. The decomposition into $R_1(A,B)$ and $R_2(A,C)$ is lossless because $A \rightarrow B$ is an FD over R_1, so the common attribute A is a key of R_1.

10.1 Testing for the lossless join property

- Algorithm TEST_LJ

Step 1: Create a matrix S, each element $s_{i,j} \in S$ corresponds the relation R_i and the attribute A_j, such that:

$$s_{j,i} = a \text{ if } A_i \in R_j, \text{ otherwise } s_{j,i} = b.$$

Step 2: Repeat the following process till S has no change or one row is made up entirely of "a" symbols.

Step 2.1: For each $X \rightarrow Y$, choose the rows where the elements corresponding to X take the value a.

Step 2.2: In those chosen rows (must be at least two rows), the elements corresponding to Y also take the value a if one of the chosen rows take the value a on Y .

10.1 Testing for the lossless join property(cont)

- The decomposition is *lossless* if one row is entirely made up by "a" values.
- The algorithm can be found as the Algorithm 15.2 in E/N book.
- Note: The correctness of the algorithm is based on the assumption that no null values are allowed for the join attributes.

If and only if exists an order such that $R_i \cap M_{i-1}$ forms a superkey of R_i or M_{i-1}, where M_{i-1} is the join on $R_1, R_2, \ldots R_{i-1}$

10.1 Testing for the lossless join property(cont)

Example: $R = (A,B,C,D)$, $F = \{A \rightarrow B, A \rightarrow C, C \rightarrow D\}$.

Let $R_1 = (A,B,C)$, $R_2 = (C,D)$.

Initially, *S* is

	A	B	C	D
R_1	a	a	a	b
R_2	b	b	a	a

Note: rows 1 and 2 of S agree on $\{C\}$, which is the left hand side of $C \rightarrow D$. Therefore, change the D value on rows 1 to a, matching the value from row 2.

Now row 1 is entirely *a*'s, so the decomposition is lossless.

(Check it.)

10.1 Testing for the lossless join property(cont)

Example 4: R = (*A*,*B*,*C*,*D*,*E*),

F = {$AB \rightarrow CD, A \rightarrow E, C \rightarrow D$}. Let $R_1 = (A,B,C)$,

$R_2 = (B,C,D)$ and $R_3 = (C,D,E)$.

Example 5: $R = (A,B,C,D,E, F)$,

F = {$A \rightarrow B, C \rightarrow DE, AB \rightarrow F$}. Let $R_1 = (A,B)$,

$R_2 = (C,D,E)$ and $R_3 = (A,C, F)$.

10.1 Testing for the lossless join property(cont)

Example 5: R = (*A*,*B*,*C*,*D*,*E*, *G*),

F = {$AB \rightarrow G, C \rightarrow DE, A \rightarrow B,$}.

Let $R_1 = (A,B,)$, $R_2 = (C, D, E)$ and $R_3 = (A, C,G)$.

	A	B	C	D	E	G
R_1	a	a	b	b	b	b
R_2	b	b	a	a	a	b
R_3	a	b	a	b	b	a

	A	B	C	D	E	G
R_1	a	a	b	b	b	b
R_2	b	b	a	a	a	b
R_3	a	~~b~~ a	a	a	a	a

10.2 Lossless decomposition into BCNF

Algorithm TO_BCNF

D := {$R_1, R_2, \ldots R_n$}

While ∃ a $R_i \in$ D and R_i is not in BCNF **Do**

{ find a $X \rightarrow Y$ in R_i that violates BCNF; replace R_i in *D* by ($R_i - Y$) and ($X \cup Y$); }

F = {A→ C, A → D, C → E, E→ D, C→G},

R1 = (C, D, E, G), R2 = (A, B, C, D)

R11 = (C, E, G), R12 = (E, D) due E→D

R21 = (A, B, C), R22 = (C, D) because of C → D

10.2 Lossless decomposition into BCNF

Algorithm TO_BCNF

D := {$R_1, R_2, \ldots R_n$}

While ∃ a $R_i \in$ D and R_i is not in BCNF **Do**

{ find a $X \rightarrow Y$ in R_i that violates BCNF; replace R_i in *D* by ($R_i - Y$) and ($X \cup Y$); }

Since a $X \rightarrow Y$ violating BCNF is not always in F, the main difficulty is to verify if R_i is in BCNF; see the approach below:

1. For each subset *X* of R_i, computer X^+.
2. $X \rightarrow (X^+|_{Ri} - X)$ violates BCNF, if $X^+|_{Ri} - X \neq \emptyset$ and $R_i - X^+ \neq \emptyset$.

Here, $X^+|_{Ri} - X = \emptyset$ means that each F.D with X as the left hand side is trivial;

$R_i - X^+ = \emptyset$ means X is a superkey of R_i

10.2 Lossless decomposition into BCNF(cont)

Example 6:(From Desai 6.35)

Find a BCNF decomposition of the relation scheme below:

SHIPPING(*Ship* , *Capacity* , *Date* , *Cargo* , *Value*)

F consists of:

Ship→ *Capacity*

{*Ship* , *Date*}→ *Cargo*

{*Cargo* , *Capacity*}→ *Value*

10.2 Lossless decomposition into BCNF(cont)

From *Ship*→ *Capacity*, we decompose *SHIPPING* into

Ship → Capacity
{*Ship* , *Date*}→ *Cargo*
{*Cargo* , *Capacity*}→ *Value*

R_1(*Ship* , *Date* , *Cargo* , *Value*)

Key: {*Ship*,*Date*}

A nontrivial FD in F^+ violates BCNF:

{*Ship* , *Cargo*} → Value

and

R_2(*Ship* , *Capacity*)

Key: {*Ship*}

Only one nontrivial FD in F^+: *Ship* → *Capacity*

10.2 Lossless decomposition into BCNF(cont)

R_1 is not in BCNF so we must decompose it further into

Ship → Capacity
{*Ship* , *Date*}→ *Cargo*
{*Cargo* , *Capacity*}→ *Value*

R_{11} (*Ship* , *Date* , *Cargo*)

Key: {*Ship*,*Date*}

Only one nontrivial FD in F^+ with single attribute on the right side: {*Ship* , *Date*} →Cargo

And

R_{12} (*Ship* , *Cargo* , *Value*)

Key: {*Ship*,*Cargo*}

Only one nontrivial FD in F^+ with single attribute on the right side: {*Ship*,*Cargo*} → *Value*

This is in BCNF and the decomposition is lossless but not dependency preserving (the FD {*Capacity*,*Cargo*} → *Value*) has been lost.

10.2 Lossless decomposition into BCNF(cont)

Or we could have chosen {*Cargo* , *Capacity*} →*Value*, which would give us:

Ship → Capacity
{*Ship* , *Date*}→ *Cargo*
{*Cargo* , *Capacity*}→ *Value*

R_1 (*Ship* , *Capacity* , *Date* , *Cargo*)

Key: {*Ship*,*Date*}

A nontrivial FD in F^+ violates BCNF:

Ship → *Capacity*

and

R_2 (*Cargo* , *Capacity* , *Value*)

Key: {*Cargo*,*Capacity*}

Only one nontrivial FD in F^+ with single attribute on the right side: {*Cargo* , *Capacity*} → *Value*

10.2 Lossless decomposition into BCNF (cont)

Ship → Capacity
$\{Ship , Date\} \rightarrow Cargo$
$\{Cargo , Capacity\} \rightarrow Value$

and then from $Ship \rightarrow Capacity$,

R_{11} (*Ship* , *Date* , *Cargo*)

Key: {*Ship*,*Date*}

Only one nontrivial FD in F^+ with single attribute

on the right side: $\{Ship , Date\} \rightarrow Cargo$

And

R_{12} (*Ship* , *Capacity*)

Key: {*Ship*}

Only one nontrivial FD in F^+: $Ship \rightarrow Capacity$

This is in BCNF and the decomposition is both lossless and dependency preserving.

However, there are relation schemes for which there is no lossless, dependency preserving decomposition into BCNF.