Dependency preservation

Getting lossless decomposition is necessary. But of course, we also want to keep dependencies, since losing a dependency means, that the corresponding constraint can be check only through natural join of the appropriate resultant relation in the decomposition. This would be very expensive, so, our aim is to get a lossless dependency preserving decomposition.

Example:

R=(A, B, C), F={A→B, B→C}

Decomposition of R: R1=(A, C) R2=(B, C)

Does this decomposition preserve the given dependencies?

Solution:

In R1 the following dependencies hold: F1'={A→A, C→C, A→C, AC→AC}

In R2 the following dependencies hold: F2'= {B→B, C→C, B→C, BC→BC}

The set of nontrivial dependencies hold on R1 and R2: F':= {B→C, A→C}

A→B can not be derived from F', so this decomposition is NOT dependency preserving.

Dependency preservation

Example:

R=(A, B, C), F={A→B, B→C}

Decomposition of R: R1=(A, B) R2=(B, C)

Does this decomposition preserve the given dependencies?

Solution:

In R1 the following dependencies hold: F1={A→B, A→A, B→B, AB→AB}

In R1 the following dependencies hold: F2= {B→B, C→C, B→C, BC→BC}

F'= F1' ∪ F2' = {A→B, B→C, trivial dependencies}

In F' all the original dependencies occur, so this decomposition preserves dependencies.

Dependency preservation

Definition:

A decomposition $D = \{R_1, \ldots, R_m\}$ of R is **dependency-preserving** wrt a set F of FDs if

$$(F_1 \cup \ldots \cup F_m)^+ = F^+,$$

Where F_i means the **projection** of the dependency set F onto R_i.
$F_i = \Pi_{Ri}(F^+)$ denotes a set of FDs $X \rightarrow Y$ in F+ such that all attributes in $X \cup Y$ are contained in Ri:

$$F_i = \Pi_{Ri}(F^+) = \{ X \rightarrow Y \mid \{X,Y\} \subseteq R_i \text{ and } X \rightarrow Y \in F+ \}$$

We do not want FDs to be lost in the decomposition.

Always possible to have a dependency-preserving decomposition D such that each R_i in D is in 3NF.

Not always possible to find a decomposition that preserves dependencies into BCNF.

Dependency preservation

Example:

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

Let S(A,C) be a decomposed relation of R. What dependencies do hold on S?

Example:

R(A, B, C, D), F={A → B, B → C}
Let S(A,C) be a decomposed relation of R. What dependencies do hold on S?

Solution: Need to compute the closure of each subset of {A,C}, wrt F^+

Compute {A}+ = {ABC}
- C is in S
- so A → C holds for S

Compute {C}+
- {C}+ = C, no new FD

Compute {AC}+
- {AC}+ = ABC, no new FD

Hence, A → C is the only non-trivial FD for S, $\Pi_S(F^+)$={A→C, + trivial FDs}

Dependency preservation

Example:

R(A, B, C, D, E), $A \rightarrow D$, $B \rightarrow E$, $DE \rightarrow C$.

Let S(A, B, C) be a decomposed relation of R. What FD-s do hold on S?

Example:

R(A, B, C, D, E), $A \rightarrow D$, $B \rightarrow E$, $DE \rightarrow C$.

Let S(A, B, C) be a decomposed relation of R. What FD-s do hold on S?

Solution: Need to compute the closure of each subset of {A, B, C}

Compute {A}+ = AD, $A \rightarrow D$, no new FD
Compute {B}+ = BE, but E is not in S, so $B \rightarrow E$ does not hold
Compute {C}+ = C, no new FD
Compute {AB}+ = ABCDE, so $AB \rightarrow C$ holds for S (since DE are not in S)
Compute {BC}+ = BCE, no new FD
Compute {AC}+ = ACD, no new FD
Compute {ABC}+ = ABCDE, no new FD

Hence, $AB \rightarrow C$ is the only nontrivial FD for S, so $\Pi_S(F^+)=\{A \rightarrow C$, + trivial FDs$\}$

Dependency preservation

The complexity of checking dependency preservation is **exponential,** since all the subsets must be calculated, and the number of subsets of an n-elements set is 2^n.

Better to decompose it directly into a dependency preserving decomposition.
The decomposition is based on the **canonical cover** (or minimal cover in other books).
The canonical cover can be interpret as the „opposit " of F+, we also might denote it by F-.
It is the minimal set of functional dependencies which is eqvivalent to F+.
The canonical cover is minimal:

- no unnecessary FD is in it
- no extraneous attributes are on the left and on the irght hand sides of the FDs

Definition:
Two sets of dependecies, G and F are eqvivalent, if G+=F+, which means, that F logically implies G and Gl ogically implies F.

Example:
$F=\{A\rightarrow C, AC\rightarrow D, E\rightarrow AD, A\rightarrow H\}$, $G=\{A\rightarrow CD, E\rightarrow AH\}$
Decide, which one of the two implies the other?

Example: F={A→C, AC→D, E→AD, E→H}, G={A→CD, E→AH}
Decide, which of the two implies the other?

Solution:

F implies G: /* using Armstrong axioms*/

Derivation of A→CD:

A→C, AC →D implies through pseudotransitivity AA→D

Then A→D, A→C implies A→DC by union rule.

Derivation of E→AH:

E→AD means E→A and E→D

E →H is given, E→A and E→H implies E→AH by union rule.

G implies F: /* using attribute closures, since X →Y holds if Yis in X+ wrt G ***/**

A_G+(A+ with respect to G)= {A, C, D}, so A→C implied by G

AC_G+={A, C, D} so AC→D implied by G

E_G+={E, A, H, C, D } so E→AD and E→H is implied by G

SO G AND F ARE EQVIVALENT!

Dependency preservation

Problem:

R(CITY, STREET, ZIP-CODE)=R(C, S, Z)
$F=\{CS \to Z, Z \to C\}$

A decomposition: R1(S, Z) and R2(C, Z) is lossless, since $Z \to C$, and Z is common in both, OR:

	C	S	Z
R1	b_{11}	a_2	a_3
R2	a_1	b_{22}	a_3

	C	S	Z
R1	a_1	a_2	a_3
R2	a_1	b_{22}	a_3

Apply $Z \to C$ b_{11} and a_1 become a_1, so the first row is full of „a"-s, which measn that it is lossless.

But the dependencies are not preserved: R1(C, Z): $F_1= \{C \to C, Z \to Z\}$
R2(C, Z): $F_2=\{ Z \to C, Z \to Z , C \to C\}$.

$CS \to Z$ can not be derived from $F_1 \cup F_2$
$(F_1 \cup F_2)^+ \neq F^+$, the decomposition is NOT dependency preserving.

Dependency preservation

Example:

R(CITY, STREET, ZIP-CODE)=R(C, S, Z)

F={CS→Z, Z→C}

Candidate keys: {C, S} és {S, Z}

What is the best normal form for R?

(S, Z), (C, Z) give BCNF but it is not dependency preserving

What about (C, S) and (S, Z)?
What about (C, Z) and (S, Z)?

Conclusion: Not always can be get a lossless dependency preserving decomposition into BCNF

BUT: There is always lossless and dependency preserving decomposition into 3NF

Dependency preservation

Definition: F minimal cover –Schilbersatz

Suppose that the minimal cover is already given as F-:

1. Search for dependencies in F- having the same attribute set on the left hand side, α:
$\alpha \rightarrow Y1$, $\alpha \rightarrow Y2$,…. $\alpha \rightarrow Yk$, and construct a relation as (α,Y1, Y2, …Yk)

2. Construcz a relation with the remainder attributes

3. In case none of the relations has a candidate key, then set one more relation with the attributes of a candidate key.

Dependency preservation

Definition: F minimal cover –Schilbersatz

Minimal cover IS NOT USED by Silberschatz, instead: canonical cover is used-already discussed in the seminar.

Definition: minimal cover:

a.) single attribute on the right hand side (decomposition rule)
b.) no extraneous attribute in the left hand side
c.) no extraneous funcional dependency

Suppose that the minimal cover is already given as F-, then the following

algorithm is a lossless dependency preserving 3NF decomposition:

Lossless Dependency Preserving 3NF Decomposition Algorithm

The minimal cover F- is given:

1. Search for dependencies in F- having the same attribute set on the left hand side, α:
$\alpha \rightarrow Y1$, $\alpha \rightarrow Y2$,…. $\alpha \rightarrow Yk$, and construct a relation as (α,Y1, Y2, …Yk)

2. Construct a relation with the remainder attributes

3. In case none of the relations has a candidate key, then set one more relation with the attributes of a candidate key.

Minimal Cover

F is a minimal set of FDs if each X→Y is: a.) |Y | = 1

b.) Left-reduced: X can't be replaced by a subset

c.) Non-redundant: X→Y can't be removed

R(ABCDIJ) and F={A→BE, AB→DE, AC→G} is given. Find minimal cover of F:

a.) A→B, A→E, AB→D, AB→E, AC→G

b.) Left-reduced : A→B, A→E,

A+={A, B, E, D}, so A→D implied by F, and also from A→D we get AB→D. That means, that instead of usind AB→D we can use A→D, since we can derive it from each direction.

For AC→G we see immediately that it is already the left reduction, since neither A→C nor A→G can not be deduced from F.

The minimal cover: F- ={A→B, A→E, A→D, AC→G}

A Dependency-Preserving Lossles-Join 3NF Decomposition Algorithm

a.) Find minimal cover
b.) Put FDs agreeing on the left-hand-side in the same schema
c.) Have extra schema for a key, if none of the above schemas contain a key

Example

R = {A,B,C,D,E,G,I,J}

a.) is given, see previous page: F- ={A→B, A→E, A→D, AC→G}

b.) R1(ABDE), R2(ACG)

c,) R3(ACIJ)

Aim of database design

BCNF:

- Lossless
- Dependency preserving

If it is impossible to get, then the optimal solution is:

3NF:

- Lossless
- Dependency preserving

Homeworks

1. R (A, B, C, D) is decomposed into R1(A, B, C), R2(C, D) and
$F=\{B \to C, AC \to D\}$.
What dependencies do hold in R1 and in R2?
Hint: Find the following closures:
$\{A\}^+=$
$\{B\}^+=$
$\{C\}^+=$
$\{A,B\}^+=$
$\{A,C\}^+=$
$\{A,D\}^+=$
$\{B,C\}^+=$
$\{B,D\}^+=$
$\{C,D\}^+=$
$\{A,B,C\}^+=$
$\{A,B,D\}^+=$
$\{B,C,D\}^+=$
$\{A,C,D\}^+=$

2. What do you thin the matter is with the algorithm below?

<u>A Dependency-Preserving 3NF Decomposition Algorithm</u>

a.) Find minimal cover
b.) Put FDs agreeing on the left-hand-side in the same schema
c.) Have extra schema for unaccounted attributes

3. A relation schema and the constraints are given:

R(MANAGER, PROJECT, DEPARTMENT)

F:={ {PROJECT, DEPT} → MANAGER, MANAGER→DEPT}

a.) What is the best normal form for R?
b.) If it is not in BCNF, try to find a lossless dependency preserving BCNF decomposition.