Database Management Systems

Module 25: Relational Database Design/5

Partha Pratim Das

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

ppd@cse.iitkgp.ac.in

Module Recap

- Studied Algorithms for Properties of Functional Dependencies

Module Objectives

- To Understand the Characterizations for Lossless Join Decomposition
- To Understand the Characterizations for Dependency Preservation

Module Outline

- Lossless Join Decomposition
- Dependency Preservation

Lossless Join Decomposition

Lossless Join Decomposition

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

$$r = \pi_{R_1}(r) \bowtie \pi_{R_2}(r)$$

- A decomposition of R into R_1 and R_2 is lossless join if at least one of the following dependencies is in F^+:
  - $R_1 \cap R_2 \to R_1$
  - $R_1 \cap R_2 \to R_2$
- 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

To Identify whether a decomposition is lossy or lossless, it must satisfy the following conditions:

- $R_1 \cup R_2 = R$
- $R_1 \cap R_2 \neq \phi$ and
- $R_1 \cap R_2 \to R_1$ or $R_1 \cap R_2 \to R_2$

Lossless Join Decomposition (2): Example

- Consider **Supplier_Parts** schema: **Supplier_Parts(S#, Sname, City, P#, Qty)**
- Having dependencies: **S#** $\rightarrow$ **Sname, S#** $\rightarrow$ **City, (S#, P#)** $\rightarrow$ **Qty**
- Decompose as: **Supplier(<u>S#</u>, Sname, City, Qty): Parts(<u>P#, Qty</u>)**
- Take Natural Join to reconstruct: **Supplier** $\bowtie$ **Parts**

S#	Sname	City	P#	Qty
3	Smith	London	301	20
5	Nick	NY	500	50
2	Steve	Boston	20	10
5	Nick	NY	400	40
5	Nick	NY	301	10

S#	Sname	City	Qty
3	Smith	London	20
5	Nick	NY	50
2	Steve	Boston	10
5	Nick	NY	40
5	Nick	NY	10

P#	Qty
301	20
500	50
20	10
400	40
301	10

S#	Sname	City	P#	Qty
3	Smith	London	301	20
5	Nick	NY	500	50
5	**Nick**	**NY**	**20**	**10**
2	Steve	Boston	20	10
5	Nick	NY	400	40
5	Nick	NY	301	10
2	**Steve**	**Boston**	**301**	**10**

- We get extra tuples! **Join is Lossy!**
- Common attribute **Qty** is not a superkey in **Supplier** or in **Parts**
- Does not preserve **(S#, P#)** $\rightarrow$ **Qty**

Lossless Join Decomposition (3): Example

- Consider **Supplier_Parts** schema: **Supplier_Parts(S#, Sname, City, P#, Qty)**
- Having dependencies: **S#** $\rightarrow$ **Sname, S#** $\rightarrow$ **City, (S#, P#)** $\rightarrow$ **Qty**
- Decompose as: **Supplier(S#, Sname, City)**: **Parts(S#, P#, Qty)**
- Take Natural Join to reconstruct: **Supplier** $\bowtie$ **Parts**

S#	Sname	City	P#	Qty
3	Smith	London	301	20
5	Nick	NY	500	50
2	Steve	Boston	20	10
5	Nick	NY	400	40
5	Nick	NY	301	10

S#	Sname	City
3	Smith	London
5	Nick	NY
2	Steve	Boston
5	Nick	NY
5	Nick	NY

S#	P#	Qty
3	301	20
5	500	50
2	20	10
5	400	40
5	301	10

S#	Sname	City	P#	Qty
3	Smith	London	301	20
5	Nick	NY	500	50
2	Steve	Boston	20	10
5	Nick	NY	400	40
5	Nick	NY	301	10

- We get back the original relation. **Join is Lossless**.
- Common attribute **S#** is a superkey in **Supplier**
- Preserves all dependencies

Lossless Join Decomposition (4): 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 $R_1 \bowtie R_2$)

Practice Problems on Lossless Join

- **Check if the decomposition of R into D is lossless:**
  a) $R(ABC) : F = \{A \rightarrow B, A \rightarrow C\}$. $D = R_1(AB), R_2(BC)$
  b) $R(ABCDEF) : F = \{A \rightarrow B, B \rightarrow C, C \rightarrow D, E \rightarrow F\}$.
  $D = R_1(AB), R_2(BCD), R_3(DEF)$
  c) $R(ABCDEF) : F = \{A \rightarrow B, C \rightarrow DE, AC \rightarrow F\}$. $D = R_1(BE), R_2(ACDEF)$
  d) $R(ABCDEG) : F = \{AB \rightarrow C, AC \rightarrow B, AD \rightarrow E, B \rightarrow D, BC \rightarrow A, E \rightarrow G\}$
     i) $D1 = R_1(AB), R_2(BC), R_3(ABDE), R_4(EG)$
     ii) $D2 = R_1(ABC), R_2(ACDE), R_3(ADG)$
  e) $R(ABCDEFGHIJ) : F = \{AB \rightarrow C, B \rightarrow F, D \rightarrow IJ, A \rightarrow DE, F \rightarrow GH\}$
     i) $D1 = R_1(ABC), R_2(ADE), R_3(BF), R_4(FGH), R_5(DIJ)$
     ii) $D2 = R_1(ABCDE), R_2(BFGH), R_3(DIJ)$
     iii) $D3 = R_1(ABCD), R_2(DE), R_3(BF), R_4(FGH), R_5(DIJ)$

Dependency Preservation

Dependency Preservation

Dependency Preservation

- Let F_i be the set of dependencies F^+ that include only attributes in R_i
  - A decomposition is **dependency preserving**, if

$$(F_1 \cup F_2 \cup \cdots \cup F_n)^+ = F^+$$

  - If it is not, then checking updates for violation of functional dependencies may require computing joins, which is expensive

Let R be the original relational schema having FD set F. Let R_1 and R_2 having FD set F_1 and F_2 respectively, are the decomposed sub-relations of R. The decomposition of R is said to be preserving if

- $F_1 \cup F_2 \equiv F$ {Decomposition Preserving Dependency}
- If $F_1 \cup F_2 \subset F$ {Decomposition NOT Preserving Dependency} and
- $F_1 \cup F_2 \supset F$ {this is not possible}

Dependency Preservation (2): Testing

- To check if a dependency $\alpha \rightarrow \beta$ is preserved in a decomposition of R into $D = \{R_1, R_2, \ldots, R_n\}$ we apply the following test (with attribute closure done with respect to F)
- The **restriction** of F^+ to R_i is the set of all functional dependencies in F^+ that include only attributes of R_i.
  - compute F^+;
for each schema R_i in D **do**
    begin
        F_i = the restriction of F^+ to R_i;
    end
$F' = \phi$
for each restriction F_i **do**
    begin
        $F' = F' \cup F_i$
    end
compute F'^+;
if $(F'^+ = F^+)$ **then** return (true)
                        else return (false);
- The procedure for checking dependency preservation takes exponential time to compute F^+ and $(F_1 \cup F_2 \cup \cdots \cup F_n)^+$

Dependency Preservation (3): Example

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

R1	R2	R3
$F_1 = \{\mathbf{A} \rightarrow ABCD$, $\mathbf{B} \rightarrow B$, $\mathbf{C} \rightarrow C$, $\mathbf{D} \rightarrow D$, $\mathbf{AB} \rightarrow ABCD$, $\mathbf{BC} \rightarrow ABCD$, $\mathbf{CD} \rightarrow CD$, $\mathbf{AD} \rightarrow ABCD$, $\mathbf{ABC} \rightarrow ABCD$, $\mathbf{ABD} \rightarrow ABCD$, $\mathbf{ACD} \rightarrow ABCD$, $\mathbf{BCD} \rightarrow ABCD\}$	$F_2 = \{\mathbf{B} \rightarrow BF$, $\mathbf{F} \rightarrow F\}$	$F_3 = \{\mathbf{D} \rightarrow DE$, $\mathbf{E} \rightarrow E\}$

- $F' = F_1 \cup F_2 \cup F_3$.
- Checking for: $\mathbf{A} \rightarrow E$, $\mathbf{A} \rightarrow F$ in F'^+
  - ▷ $A \rightarrow D$ (from R1), $D \rightarrow E$ (from R3) : **A→ E** (By Transitivity)
  - ▷ $A \rightarrow B$ (from R1), $B \rightarrow F$ (from R2) : **A→ F** (By Transitivity)
- Checking for: $\mathbf{BC} \rightarrow E$, $\mathbf{BC} \rightarrow F$ in F'^+
  - ▷ $BC \rightarrow D$ (from R1), $D \rightarrow E$ (from R3) : **BC→ E** (By Transitivity)
  - ▷ $B \rightarrow F$ (from R2) : **BC→ F** (By Augmentation)

Hence all dependencies are preserved.

Dependency Preservation (4): Example

- **R** (A, B, C, D)
  $\mathbf{F} = \{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A\}$
- 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.

Dependency Preservation (5): Testing

- To check if a dependency $\alpha \rightarrow \beta$ is preserved in a decomposition of R into $R_1, R_2, \cdots, R_n$ we apply the following test (with attribute closure done with respect to F)
  - *result* $= \alpha$
    while (changes to result) do
        for each R_i in the decomposition
            $t = (result \cap R_i)^+ \cap Ri$
            $result = result \cup t$
  - 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 $(F_1 \cup F_2 \cup \cdots \cup F_n)^+$

Dependency Preservation (6): Example

- $R(ABCDEF) :. \; F = \{A \rightarrow BCD, A \rightarrow EF, BC \rightarrow AD, BC \rightarrow E, BC \rightarrow F, B \rightarrow F, D \rightarrow E\}$
- $Decomp = \{ABCD, BF, DE\}$
- On projections:

ABCD (R1)	BF (R2)	DE (R3)
A → BCD BC → AD	B → F	D → E

- Need to check for: $\cancel{A \rightarrow BCD}$, $\boldsymbol{A \rightarrow EF}$, $\cancel{BC \rightarrow AD}$, $\boldsymbol{BC \rightarrow E}$, $\boldsymbol{BC \rightarrow F}$, $\cancel{B \rightarrow F}$, $\cancel{D \rightarrow E}$
- $(BC)+/F1 = ABCD$. $(ABCD)+/F2 = ABCDF$. $\boldsymbol{(ABCDF)}+/\boldsymbol{F3} = \boldsymbol{ABCDEF}$. Preserves $\boldsymbol{BC \rightarrow E, BC \rightarrow F}$
  $BC \rightarrow AD \; (R1)$, $AD \rightarrow E \; (R3)$ implies $BC \rightarrow E$
  $B \rightarrow F \; (R2)$ implies $BC \rightarrow F$
- $(A)+/F1 = ABCD$. $(ABCD)+/F2 = ABCDF$. $\boldsymbol{(ABCDF)}+/\boldsymbol{F3} = \boldsymbol{ABCDEF}$. Preserves $\boldsymbol{A \rightarrow EF}$
  $A \rightarrow B \; (R1)$, $B \rightarrow F \; (R2)$ implies $A \rightarrow F$
  $A \rightarrow D \; (R1)$, $D \rightarrow E \; (R3)$ implies $A \rightarrow E$

Dependency Preservation (7): Example

- $R(ABCDEF) : F = \{A \rightarrow BCD, A \rightarrow EF, BC \rightarrow AD, BC \rightarrow E, BC \rightarrow F, B \rightarrow F, D \rightarrow E\}$. $Decomp = \{ABCD, BF, DE\}$
- On projections:

ABCD (R1)	BF (R2)	DE (R3)
A → B, A → C, A → D, BC → A, BC → D	B → F	D → E

- Infer reverse FD's:
  - $B+/F = BF : B \rightarrow A$ cannot be inferred
  - $C+/F = C : C \rightarrow A$ cannot be inferred
  - $D+/F = DE : D \rightarrow A$ and $D \rightarrow BC$ cannot be inferred
  - $A+/F = ABCDEF : A \rightarrow BC$ can be inferred, but it is equal to $A \rightarrow B$ and $A \rightarrow C$
  - $F+/F = F : F \rightarrow B$ cannot be inferred
  - $E+/F = E : E \rightarrow D$ cannot be inferred
- Need to check for: $\cancel{A \rightarrow BCD}, \boldsymbol{A \rightarrow EF}, \cancel{BC \rightarrow AD}, \boldsymbol{BC \rightarrow E}, \boldsymbol{BC \rightarrow F}, \cancel{B \rightarrow F}, \cancel{D \rightarrow E}$
  - $(BC)+/F = ABCDEF$. Preserves $\boldsymbol{BC \rightarrow E, BC \rightarrow F}$
  - $(A)+/F = ABCDEF$. Preserves $\boldsymbol{A \rightarrow EF}$

Practice Problems on Dependency Preservation

- **Check whether the decomposition of R into D is preserving dependency:**

  a) $R(ABCD) : F = \{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A\}.\ D = \{AB, BC, CD\}$
  b) $R(ABCDEF) : F = \{AB \rightarrow CD, C \rightarrow D, D \rightarrow E, E \rightarrow F\}.\ D = \{AB, CDE, EF\}$
  c) $R(ABCDEG) : F = \{AB \rightarrow C, AC \rightarrow B, BC \rightarrow A, AD \rightarrow E, B \rightarrow D, E \rightarrow G\}.\ D = \{ABC, ACDE, ADG\}$
  d) $R(ABCD) : F = \{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow B\}.\ D = \{AB, BC, BD\}$
  e) $R(ABCDE) : F = \{A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A\}.\ D = \{ABCE, BD\}$

Module Summary

- Understood the Characterization for and Determination of Lossless Join
- Understood the Characterization for and Determination of Dependency Preservation

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".