CS2102 Database Systems

SCHEMA REFINEMENT: DECOMPOSITIONS

Schema Decomposition

- Decomposition of a schema R is a set of schemas $\{R_1, R_2, \ldots, R_n\}$ such that $R_i \subseteq R$ and $R = R_1 \cup R_2 \cup \ldots \cup R_n$

- If $\{R_1, R_2, \ldots, R_n\}$ is a decomposition of R, then for any relation *r* of R, we have

$$r \subseteq \pi_{R1}(r) \otimes \pi_{R2}(r) \otimes \ldots \otimes \pi_{Rn}(r)$$

Schema Decomposition - Example

MovieList Database

title	director	address	phone	time
Schlinder's List	Spielberg	Holland	3355	1130
Saving Private Ryan	Spielberg	Holland	3355	1430
Noth by Northwest	Hitchcock	Orchard	1234	1400
The Godfather	Coppola	Orchard	1234	1700
Saving Private Ryan	Spielberg	Orchard	1234	2130

Movie

title	director
Schlinder's List	Spielberg
Saving Private Ryan	Spielberg
Noth by Northwest	Hitchcock
The Godfather	Coppola

Screens

address	time	title
Holland	1130	Schlinder's List
Holland	1430	Saving Private Ryan
Orchard	1400	Noth by Northwest
Orchard	1700	The Godfather
Orchard	1430	Saving Private Ryan

Cinema

address	phone
Holland	3355
Orchard	1234

Properties of Schema Decomposition

- Decomposition must preserve information
  - Data in original relation ≡ Data in decomposed relations
  - Crucial for correctness
- Decomposition should preserve FDs
  - FDs in original schema ≡ FDs in decomposed schemas
  - Facilitates checking of FD violations

Lossless-Join Decomposition

- It is important that a decomposition preserves information; we can reconstruct r from joining its projections $\{r_1, r_2, \ldots, r_n\}$

- A decomposition of R (with FDs F) into $\{R_1, R_2, \ldots, R_n\}$ is a lossless-join decomposition with respect to F if

$$\pi_{R1}(r) \otimes \pi_{R2}(r) \otimes \ldots \otimes \pi_{Rn}(r) = r$$

  for every relation r of R that satisfies F

Example

- Consider the decomposition of R(A, B, C) into { R_1(A, B), R_2(B, C) }

r

A	B	C
a1	b1	c1
a2	b1	c2

r1

A	B
a1	b1
a2	b1

r2

B	C
b1	c1
b1	c2

r1 ⊗ r2

A	B	C
a1	b1	c1
a1	b1	c2
a2	b1	c1
a2	b1	c2

- Since r $\subset$ r1 $\otimes$ r2, { R_1(A, B), R_2(B, C) } is not a lossless-join decomposition

Example

- Consider the decomposition of R(A, B, C) into { R_1(A, B), R_2(A, C) }

r

A	B	C
a1	b1	c1
a2	b1	c2

$\longrightarrow$

r1

A	B
a1	b1
a2	b1

r3

A	C
a1	c1
a2	c2

$\longrightarrow$

r1 ⊗ r3

A	B	C
a1	b1	c1
a2	b1	c2

- Since r = r1 ⊗ r3, { R_1(A, B), R_2(A, C) } is a lossless-join decomposition

Lossless-Join Decomposition

- How to determine if $\{R_1, R_2\}$ is a lossless-join decomposition of R?
- Theorem:

  The decomposition of R (with FDs F) into relations with attribute sets R_1 and R_2 is lossless with respect to F if and only if F^+ contains the FD $R_1 \cap R_2 \rightarrow R_1$ or $R_1 \cap R_2 \rightarrow R_2$
- Attributes common to R_1 and R_2 must contain a key for either R_1 and R_2

Lossless-Join Decomposition

- How to decompose R into $\{R_1, R_2\}$ such that it is a lossless-join decomposition?

- Corollary:

  If $\alpha \rightarrow \beta$ holds on R and $\alpha \cap \beta = \phi$, then the decomposition of R into $\{ R - \beta , \alpha\beta \}$ is a lossless-join decomposition

Example

- Consider R(A, B, C) with FDs F = {$A \to B$}
- Decomposition {R_1(A, B), R_2(B, C) } is not a lossless join w.r.t. F since $AB \cap BC = B$ and neither $B \to R_1$ nor $B \to R_2$ holds on R

r

A	B	C
a1	b1	c1
a2	b1	c2

$\longrightarrow$

r1

A	B
a1	b1
a2	b1

r2

B	C
b1	c1
b2	c2

$\longrightarrow$

r1 ⊗ r2

A	B	C
a1	b1	c1
a1	b1	c2
a2	b1	c1
a2	b1	c2

Example

- Consider R(A, B, C) with FDs F = {$A \rightarrow B$}
- Decomposition {R_1(A, B), R_2(A, C) } is a lossless join since $AB \cap AC = A$ and $A \rightarrow R_1$

r

A	B	C
a1	b1	c1
a2	b1	c2

$\longrightarrow$

r1

A	B
a1	b1
a2	b1

r3

A	C
a1	c1
a2	c2

$\longrightarrow$

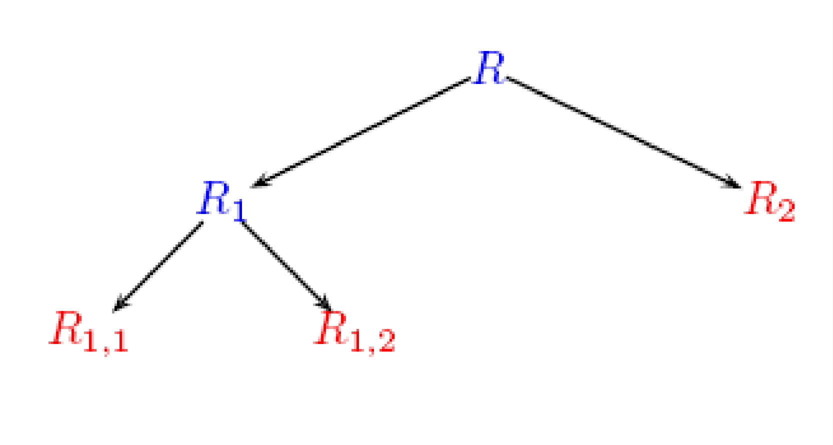
r1 ⊗ r3

A	B	C
a1	b1	c1
a2	b1	c2

Lossless-Join Decomposition

- Theorem:

  If $\{R_1, R_2\}$ is a lossless join decomposition of R, and $\{R_{11}, R_{12}\}$ is a lossless join decomposition of R_1, then $\{R_{11}, R_{12}, R_2\}$ is a lossless join decomposition of R



Example

MovieList

title	director	address	phone	time
Schlinder's List	Spielberg	Holland	3355	1130
Saving Private Ryan	Spielberg	Holland	3355	1430
Noth by Northwest	Hitchcock	Orchard	1234	1400
The Godfather	Coppola	Orchard	1234	1700
Saving Private Ryan	Spielberg	Orchard	1234	2130

Movie

title	director
Schlinder's List	Spielberg
Saving Private Ryan	Spielberg
Noth by Northwest	Hitchcock
The Godfather	Coppola

Cinema-Screens

address	phone	time	title
Holland	3355	1130	Schlinder's List
Holland	3355	1430	Saving Private Ryan
Orchard	1234	1400	Noth by Northwest
Orchard	1234	1700	The Godfather
Orchard	1234	2130	Saving Private Ryan

MovieList

title	director	address	phone	time
Schlinder's List	Spielberg	Holland	3355	1130
Saving Private Ryan	Spielberg	Holland	3355	1430
Noth by Northwest	Hitchcock	Orchard	1234	1400
The Godfather	Coppola	Orchard	1234	1700
Saving Private Ryan	Spielberg	Orchard	1234	2130

Movie

title	director
Schlinder's List	Spielberg
Saving Private Ryan	Spielberg
Noth by Northwest	Hitchcock
The Godfather	Coppola

Cinema-Screens

address	phone	time	title
Holland	3355	1130	Schlinder's List
Holland	3355	1430	Saving Private Ryan
Orchard	1234	1400	Noth by Northwest
Orchard	1234	1700	The Godfather
Orchard	1234	2130	Saving Private Ryan

Cinema

address	phone
Holland	3355
Orchard	1234

Screens

address	time	title
Holland	1130	Schlinder's List
Holland	1430	Saving Private Ryan
Orchard	1400	Noth by Northwest
Orchard	1700	The Godfather
Orchard	2130	Saving Private Ryan

Projection of FDs

- The projection of F on X (denoted by F_X) is the set of FDs in F+ that involves only attributes in X.
- Example:
  - MovieList (title, director, address, phone, time) decompose to Movie (title, director)
    Cinema (address, phone)
    Screens (address, time, title)
  - F_{Movie} = { title $\rightarrow$ director}
    F_{Cinema} = { address $\rightarrow$ phone}
    $F_{Screens}$ = { address, time $\rightarrow$ title }

Computing FD Projections

- Input: F, X
- Output: F_X
- Steps: Result = ϕ

  For each $Y \subseteq X$ do

  $T = Y^+$ (w.r.t. F)

  Result = Result $\cup \{Y \rightarrow T \cap X\}$

  Return Result

Quiz

- Consider $R(A, B, C)$ with FDs $F = \{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$. Compute F_{AB} and F_{BC}
- For F_{AB}
  - Compute $A^+ = BC$, we have $A \rightarrow BC \cap AB$
  - Compute $B^+ = AC$, we have $B \rightarrow AC \cap AB$
  - So, $F_{AB} = \{A \rightarrow B, B \rightarrow A\}$
- For F_{BC}
  - Compute $B^+ = AC$, we have $B \rightarrow AC \cap BC$
  - Compute $C^+ = AB$, we have $C \rightarrow AB \cap BC$
  - So, $F_{BC} = \{B \rightarrow C, C \rightarrow B\}$

Dependency Preserving Decomposition

- A decomposition $\{R_1, R_2, \ldots, R_n\}$ of R is dependency preserving if

$$F^+ = (F_{R1} \cup F_{R2} \cup \ldots \cup F_{Rn})^+$$

- Dependency preserving decomposition is important because any update to a relation R_i only requires us to enforce F_{Ri} in relation R_i

Example

- Consider R(A, B, C) with FDs F = { B → C, AC → B }
- Decomposition { R_1(A, B), R_2(B,C) } is not dependency preserving
  - Non-trivial FDs in $F_{R1} = \phi$
  - Non-trivial FDs in $F_{R2} = \{B \rightarrow C\}$
  - Therefore, AC → B is not in $(F_{R1} \cup F_{R2})^+$
  - That is, AC → B is not preserved

r

A	B	C
a1	b1	c1
a2	b1	c2

⟶

r1

A	B
a1	b1
a2	b1

r2

B	C
b1	c1
b2	c2

  - Inserting a new tuple (a1, b2, c1) into r will violate AC → B
  - But inserting (a1, b2) into r1 and (b2, c1) into r2 does not violate any FDs in F_{R1} and F_{R2} respectively
  - Need to compute r1 ⊗ r2 to detect violate of AC → B

Checking for Preservation of Dependencies

- Is $\{R_1, R_2, \ldots, R_n\}$ a dependency-preserving decomposition of R (with FDs F) ?
- If there exists some FD $f \in F$ such that $(F_{R1} \cup F_{R2} \cup \ldots \cup F_{Rn})^+$ does not imply *f*, then the answer is no, else the answer is yes.

Next…

Schema Refinement: Normal Forms