CS2102 Database Systems

Slides adapted from Prof. Chan Chee Yong

LECTURE 09

DECOMPOSITION

PREVIOUS LECTURE

Functional dependencies

What

- Abstraction of "*uniquely identifies*" to any arbitrary attributes
- $a \rightarrow b$ means that the same set of values a will always give the same set of values b

Why

- **Superkey** "*uniquely identifies*" the entire relation R
  $a \rightarrow R$
- **Key** minimal set of superkey
- **Prime attribute** attributes in some key
  - Non-prime attribute
    $\Rightarrow$ cannot be used to "*uniquely identify*" R

PREVIOUS LECTURE

Armstrong's axiom

Basic

- Reflexivity — if $b \subseteq a$ then $a \rightarrow b$
- Augmentation — if $a \rightarrow b$ then $ac \rightarrow bc$
- Transitivity — if $a \rightarrow b$ and $b \rightarrow c$ then $a \rightarrow c$

Extended

- Union — if $a \rightarrow b$ and $a \rightarrow c$ then $a \rightarrow bc$
- Decomposition — if $a \rightarrow b$ then $a \rightarrow b'$ where $b' \subseteq b$

Properties

- Sound — any derived FD is implied by F
- Complete — all FDs in F^+ can be derived

PREVIOUS LECTURE

Closures

F^+

- Set of all FDs implied by F
- $F^+ = \{ f \mid F \vDash f \}$

How

- Start with F
- Apply Armstrong's axiom until none can produce new FDs

Properties

- Maximum size: $2^n \times 2^n$
- Most FDs are uninteresting (e.g., trivial)
  - Wasted computation

PREVIOUS LECTURE

Closures

$\boldsymbol{a^+}$

- Set of all attributes functionally depends on a
- $a^+ = \{ A \in R \mid F \vDash a \rightarrow A \}$

How

- *Invariant*: $a \rightarrow \theta$ is always implied
  - Start with $\theta = a$
  - Consider any $b \rightarrow c$ such that $b \subseteq \theta$
    - By invariant we have $a \rightarrow \theta; \theta \rightarrow b; b \rightarrow c$
    - By transitivity we have $a \rightarrow c$
    - By union we have $a \rightarrow \theta \cup c$
      - Thus, we can set $\theta = \theta \cup c$ and still maintain invariant
      - We can stop once θ cannot be expanded anymore

PREVIOUS LECTURE

Closures

$\boldsymbol{a^+}$

- Set of all attributes functionally depends on a
- $a^+ = \{ A \in R \mid F \vDash a \rightarrow A \}$

Properties

- *Invariant*: $a \rightarrow \theta$ is always implied
  - Let $a^+ = \theta$
  - By decomposition $a \rightarrow b$ where $b \subseteq a^+$
  - Hence, $F \vDash a \rightarrow b$ *if and only if* $b \subseteq a^+$
- Maximum size: n
- $a \rightarrow a^+$ can be decomposed into two components
  - Trivial $a \rightarrow a$
  - Completely non-trivial $a \rightarrow (a^+ - a)$

PREVIOUS LECTURE

Minimal covers

What

- Set G of FDs such that
  - Every FD in G is of the form $a \to A$
  - For each FD $a \to A$ in G, a has no redundant attributes
  - There are no redundant FDs in G
  - G and F are equivalent

Redundant attributes

- Let $F_1 = G \cup \{aA \to B\}$ and $F_2 = G \cup \{a \to B\}$
- If $F_1 \equiv F_2$ then A is redundant
  - In other words, having $a \to B$ instead of $aA \to B$ does not change the amount of information
- How
  - Compute a^+ w.r.t. F_1; if $B \in a^+$, then $a \to B$
  - So $F_1 \vDash F_2$ and $F_2 \vDash F_1$ which means $F_1 \equiv F_2$
    - $F_2 \vDash aA \to a$ and $F_2 \vDash a \to B$ means $F_2 \vDash aA \to B$

PREVIOUS LECTURE

Minimal covers

What

- Set G of FDs such that
  - Every FD in G is of the form $a \to A$
  - For each FD $a \to A$ in G, a has no redundant attributes
  - There are no redundant FDs in G
  - G and F are equivalent

Redundant FD

- Let $F = G \cup \{a \to B\}$
- If $F \equiv G$ then $a \to B$ is redundant
  - In other words, not having $a \to B$ does not change the amount of information
- How
  - Compute a^+ w.r.t. G; if $B \in a^+$, then $a \to B$
  - So $G \vDash F$ and $F \vDash G$ which means $F \equiv G$
    - $F \vDash G$ trivially because $G \subset F$

- Schema decomposition
  - Lossless-join decomposition
  - Dependency-preserving decomposition

Overview

- Schema decomposition
    - Lossless-join decomposition
    - Dependency-preserving decomposition

Schema decomposition

Schema decomposition

Definition

- The decomposition of schema R is a set of schemas $\{R_1, R_2, \dots, R_n\}$ (called fragments) such that
  - $R_i \subseteq R$ for each R_i
    - Each fragment is simpler than the original schema
  - $R = R_1 \cup R_2 \cup \cdots \cup R_n$
    - No attributes are missing

- Consider a relation r of R, the decomposition of r into $\{r_1, r_2, \dots, r_n\}$ is
  - $r_i = \pi_{R_i}(r)$
    - Projection operation!

Schema decomposition

Useful properties

- The decomposition must preserve information
  - Information in original relation must be equivalent to the information in decomposed relations
    - After performing joins (e.g., natural join)
  - Crucial for correctness!

- The decomposition should preserve FDs
  - FDs in original schema must be equivalent to the FDs in decomposed schemas
  - Facilitates checking of FD violations without performing joins or relying on triggers

Schema decomposition

Example

- Let $R(A, B, C)$ with FDs $F = \{A \rightarrow B\}$

R

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

- Is $R_1(A, B)$ and $R_2(B, C)$ a decomposition of R?

R1

A	***B***
a1	b1
a2	b1

R2

B	***C***
b1	c1
b1	c2

Schema decomposition

Example

- Let $R(A, B, C)$ with FDs $F = \{A \rightarrow B\}$

R

A	B	C
a1	b1	c1
a2	b1	c2

- Is $R_1(A, B)$ and $R_2(A, C)$ a decomposition of R?

R1

A	B
a1	b1
a2	b1

R2

A	C
a1	c1
a2	c2

Schema decomposition

Example

- Let $R(A, B, C)$ with FDs $F = \{A \rightarrow B\}$

R

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

- Is $R_1(A, B, C)$ and $R_2(B, C)$ a decomposition of R?

R1

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

R2

B	***C***
b1	c1
b1	c2

Schema decomposition

Example

- Let $R(A, B, C)$ with FDs $F = \{A \rightarrow B\}$

R

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

- Is $R_1(A)$, $R_2(B)$, and $R_3(C)$ a decomposition of R?

R1

A
a1
a2

R2

B
b1

R3

C
c1
c2

Schema decomposition

Example

- Let $R(A, B, C)$ with FDs $F = \{A \rightarrow B\}$

R

A	B	C
a1	b1	c1
a2	b1	c2

  - Is $R_1(A, B)$ and $R_2(B)$ a decomposition of R?

R1

A	B
a1	b1
a2	b1

R2

B
b1
b1

Lossless-join decomposition

Definition

- Consider a schema R decomposed into $\{R_1, R_2, \dots, R_n\}$ and the natural join operation $\bowtie$
  - Let the relation r of R be decomposed into $\{r_1, r_2, \dots, r_n\}$ such that
    - $r_1 = \pi_{R_1}(r), r_2 = \pi_{R_2}(r), \dots, r_n = \pi_{R_n}(r)$
  - $\{R_1, R_2, \dots, R_n\}$ is a lossless-join (or lossless) decomposition w.r.t. F *<u>if</u>*
    - $r = \pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \cdots \bowtie \pi_{R_n}(r)$
    - In other words, no information is lost by performing a join
      - But tuple (i.e., data) may increase!
    - If not a lossless-join, we call it a lossy-join (or lossy) decomposition

Lemma 1

- If $\{R_1, R_2, \dots, R_n\}$ is a decomposition of R, then for any relation r or R
  - $r \subseteq \pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \cdots \bowtie \pi_{R_n}(r)$
  - *<u>Lossy-join decomposition produces more tuple!</u>*

Lossless-join decomposition

Example

- Let $R(A, B, C)$ with FDs $F = \{A \rightarrow B\}$

R

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

- Is $R_1(A, B)$ and $R_2(B, C)$ a lossless-join decomposition of R?

R1

A	***B***
a1	b1
a2	b1

R2

B	***C***
b1	c1
b1	c2

R1 $\bowtie$ ***R2***

A	***B***	***C***

Lossless-join decomposition

Example

- Let $R(A, B, C)$ with FDs $F = \{A \rightarrow B\}$

R

A	B	C
a1	b1	c1
a2	b1	c2

- Is $R_1(A, B)$ and $R_2(A, C)$ a lossless-join decomposition of R?

R1

A	B
a1	b1
a2	b1

R2

A	C
a1	c1
a2	c2

R1 ⋈ R2

A	B	C

Lossless-join decomposition

Theorem 1

- The decomposition of R with FDs F into $\{R_1, R_2\}$ is a lossless-join decomposition w.r.t. F *if*
  - $F \vDash R_1 \cap R_2 \rightarrow R_1$

    OR
  - $F \vDash R_1 \cap R_2 \rightarrow R_2$

Intuition

- Let $\{R_1, R_2\}$ be a decomposition of R with FDs F
- Let $R_1 \cap R_2 = a$, $R_1 - R_2 = b$, and $R_2 - R_1 = c$
- Let r be an instance of R, $r_1 = \pi_{R_1}(r)$ and $r_2 = \pi_{R_2}(r)$
  - If $F \vDash a \rightarrow b$

R

A	B	C
a1	b1	c1
a2	b1	c2

R1

A	B
a1	b1
a2	b1

R2

A	C
a1	c1
a2	c2

R1 ⋈ R2

A	B	C

Lossless-join decomposition

Theorem 1

- The decomposition of R with FDs F into $\{R_1, R_2\}$ is a lossless-join decomposition w.r.t. F *if*
  - $F \vDash R_1 \cap R_2 \rightarrow R_1$

    OR
  - $F \vDash R_1 \cap R_2 \rightarrow R_2$

Intuition

- Let $\{R_1, R_2\}$ be a decomposition of R with FDs F
- Let $R_1 \cap R_2 = a$, $R_1 - R_2 = b$, and $R_2 - R_1 = c$
- Let r be an instance of R, $r_1 = \pi_{R_1}(r)$ and $r_2 = \pi_{R_2}(r)$
  - If $F \nvDash a \rightarrow b$ and $F \nvDash a \rightarrow c$

R

A	B	C
a1	b1	c1
a1	b2	c2

R1

A	B
a1	b1
a1	b2

R2

A	C
a1	c1
a1	c2

R1 ⋈ R2

A	B	C

Lossless-join decomposition

Corollary 1

- If $a \rightarrow b$ is a completely non-trivial FD that holds on R, then the decomposition of R into $\{R - b, ab\}$ is a lossless-join decomposition
  - Since $(R - b) \cap ab = a$ and $a \rightarrow b$ so $a \rightarrow ab$

Theorem 2

- If $\{R_1, R_2, \ldots, R_n\}$ is a lossless-join decomposition of R, and $\{R_{1,1}, R_{1,2}\}$ is a lossless-join decomposition of R_1m then $\{R_{1,1}, R_{1,2}, R_2, \ldots, R_n\}$ is a lossless join decomposition of R

- Combine corollary 1 and theorem 2 to decompose any R into more than 2 fragments

Lossless-join decomposition

Example

- Let $R(A, B, C, D, E)$ with FDs
$$F = \{A \to B, C \to D, CE \to A\}$$
- Is $\{R_1(A, B), R_2(C, D), R_3(A, C, E)\}$ a lossless-join decomposition?
- Steps:
  - Since this is a **decomposition** into 3 fragments, we need to find an intermediate decomposition
  - Let the intermediate be R_I, then either
    1. $\{R_I, R_3\}$ is a lossless-join decomposition of R and $\{R_1, R_2\}$ is a lossless-join decomposition of R_I where $R_I(A, B, C, D)$
    2. $\{R_I, R_2\}$ is a lossless-join decomposition of R and $\{R_1, R_3\}$ is a lossless-join decomposition of R_I where $R_I(A, B, C, E)$
    3. $\{R_I, R_1\}$ is a lossless-join decomposition of R and $\{R_2, R_3\}$ is a lossless-join decomposition of R_I where $R_I(A, C, D, E)$

Lossless-join decomposition

Example

- Let $R(A, B, C, D, E)$ with FDs
$$F = \{A \rightarrow B, C \rightarrow D, CE \rightarrow A\}$$
- Is $\{R_1(A, B), R_2(C, D), R_3(A, C, E)\}$ a lossless-join decomposition?
- Steps:
  - **Case #1**
    - Check $\{R_I(A, B, C, D), R_3(A, C, E)\}$ is a lossless-join decomposition of R
      - $R_I \cap R_1 = AC$ and $AC \rightarrow R_I$ (since $AC^+ = ABCD$)
      - ✓ $\{R_I(A, B, C, D), R_3(A, C, E)\}$ is a lossless-join decomposition of R
    - Check $\{R_1(A, B), R_2(C, D)\}$ is a lossless-join decomposition of R_I
      - $R_1 \cap R_2 = \emptyset$ but neither $\emptyset \rightarrow R_1$ nor $\emptyset \rightarrow R_2$
      - ❖ $\{R_1(A, B), R_2(C, D)\}$ is **not** a lossless-join decomposition of R_I

Lossless-join decomposition

Example

- Let $R(A, B, C, D, E)$ with FDs
$$F = \{A \rightarrow B, C \rightarrow D, CE \rightarrow A\}$$
- Is $\{R_1(A, B), R_2(C, D), R_3(A, C, E)\}$ a lossless-join decomposition?
- Steps:
  - **Case #2**
    - Check $\{R_I(A, B, C, E), R_2(C, D)\}$ is a lossless-join decomposition of R
      - $R_I \cap R_2 = C$ and $C \rightarrow R_2$
      - ✓ $\{R_I(A, B, C, E), R_2(C, D)\}$ is a lossless-join decomposition of R
    - Check $\{R_1(A, B), R_3(A, C, E)\}$ is a lossless-join decomposition of R_I
      - $R_1 \cap R_2 = A$ and $A \rightarrow R_1$
      - ✓ $\{R_1(A, B), R_3(A, C, E)\}$ is a lossless-join decomposition of R_I

Lossless-join decomposition

Example

- Let $R(A, B, C, D, E)$ with FDs
$$F = \{A \rightarrow B, C \rightarrow D, CE \rightarrow A\}$$
- Is $\{R_1(A, B), R_2(C, D), R_3(A, C, E)\}$ a lossless-join decomposition?
- Steps:
  - **Case #3**
    - Check $\{R_I(A, C, D, E), R_1(A, B)\}$ is a lossless-join decomposition of R
      - $R_I \cap R_1 = A$ and $A \rightarrow R_1$
      - ✓ $\{R_I(A, C, D, E), R_1(A, B)\}$ is a lossless-join decomposition of R
    - Check $\{R_2(C, D), R_3(A, C, E)\}$ is a lossless-join decomposition of R_I
      - $R_2 \cap R_3 = C$ and $C \rightarrow R_2$
      - ✓ $\{R_2(C, D), R_3(A, C, E)\}$ is a lossless-join decomposition of R_I

Lossless-join decomposition

Example

- Let $R(A, B, C, D, E)$ with FDs
$$F = \{A \rightarrow B, C \rightarrow D, CE \rightarrow A\}$$
- Is $\{R_1(A, B), R_2(C, D), R_3(A, C, E)\}$ a lossless-join decomposition?
- Notes:
  - We can stop after case #2 since we have found one intermediate decomposition
  - If it is a lossy-join decomposition, then we have to check until all cases have been exhausted
    - $\left(\neg\exists_d \cdot \text{lossless}(d)\right) = \left(\forall_d \cdot \neg\text{lossless}(d)\right) = \left(\forall_d \cdot \text{lossy}(d)\right)$
  - Corollary 1 can be used to guide the decomposition
    - The decomposition is based on FDs f in F

Dependency-preserving decomposition

Projection

- Consider a schema R with FDs F, and $a \subseteq R$
- The projection of F on a denoted by F_a is the set of FDs such that
  - $F_a \subseteq F^+$
    - The projection is a subset of the closure
  - $F_a = \{ b \to c \in F^+ \mid bc \subseteq a \}$
    - The projection involves only attributes in a
- Example: Let $R(A, B, C)$ with FDs $F = \{A \to B, B \to C, C \to B\}$

FD	***In*** F^+***?***	***In*** F_{AB}***?***	***In*** F_{AC}***?***
$C \to A$			
$A \to AB$			
$B \to C$			
$A \to C$			

Dependency-preserving decomposition

Projection

- Consider a schema R with FDs F, and $a \subseteq R$
- The projection of F on a denoted by F_a is the set of FDs such that
  - $F_a \subseteq F^+$
    - The projection is a subset of the closure
  - $F_a = \{ b \to c \in F^+ \mid bc \subseteq a \}$
    - The projection involves only attributes in a
- How to compute?
  - Consider any $b \subseteq a$ and compute b^+
    - Then $b \to b^+$ is in F^+ as well as $b \to c$ for any $c \subseteq b^+$
    - Let $c = b^+ \cap a$, we know that $c \subseteq b^+$
      - Then $b \to (b^+ \cap a)$ is in F^+
      - But $b \subseteq a$ and $(b^+ \cap a) \subseteq a$, so $b \to (b^+ \cap a)$ in F_a

Dependency-preserving decomposition

Projection

- Algorithm #3
- **Input** A set of attributes $a \subseteq R$ and a set of FDs F on R
- **Output** F_a

```
1. initialize θ = ∅
2. for each (b ⊆ a such that b ≠ ∅)
3.     θ = θ ∪ {b → (b⁺ ∩ a)} // w.r.t. F
4. return θ
```

- How to compute?
  - Consider any $b \subseteq a$ and compute b^+
    - Then $b \rightarrow b^+$ is in F^+ as well as $b \rightarrow c$ for any $c \subseteq b^+$
    - Let $c = b^+ \cap a$, we know that $c \subseteq b^+$
      - Then $b \rightarrow (b^+ \cap a)$ is in F^+
      - But $b \subseteq a$ and $(b^+ \cap a) \subseteq a$, so $b \rightarrow (b^+ \cap a)$ in F_a

Dependency-preserving decomposition

Projection

- Algorithm #3
- **Input** A set of attributes $a \subseteq R$ and a set of FDs F on R
- **Output** F_a

```
1. initialize θ = ∅
2. for each (b ⊆ a such that b ≠ ∅)
3.     θ = θ ∪ {b → (b⁺ ∩ a)} // w.r.t. F
4. return θ
```

- Example: Let $R(A, B, C)$ with FDs $F = \{A \rightarrow B, B \rightarrow C, C \rightarrow B\}$
  - Compute F_{AB}
    - initialize $\Rightarrow \theta = \emptyset$
    - let $b = A$, $A^+ = ABC$ $\Rightarrow \theta = \{ \quad \}$
    - let $b = B$, $B^+ = ABC$ $\Rightarrow \theta = \{ \quad \}$
    - let $b = AB$, $AB^+ = ABC$ $\Rightarrow \theta = \{ \quad \}$

Dependency-preserving decomposition

Projection

- Algorithm #3
- **Input** A set of attributes $a \subseteq R$ and a set of FDs F on R
- **Output** F_a

```
1. initialize θ = ∅
2. for each (b ⊆ a such that b ≠ ∅)
3.     θ = θ ∪ {b → (b⁺ ∩ a)} // w.r.t. F
4. return θ
```

- Example: Let $R(A, B, C)$ with FDs $F = \{A \rightarrow B, B \rightarrow C, C \rightarrow B\}$
  - Compute F_{BC}
    - initialize $\Rightarrow \theta = \emptyset$
    - let $b = B$, $B^+ = BC$ $\Rightarrow \theta = \{ \quad \}$
    - let $b = C$, $C^+ = BC$ $\Rightarrow \theta = \{ \quad \}$
    - let $b = BC$, $BC^+ = BC$ $\Rightarrow \theta = \{ \quad \}$

Dependency-preserving decomposition

Why projection?

- Consider a schema $R(A, B, C, D, E)$ and a decomposition $\{R_1(A, B, C), R_2(C, D, E)\}$
  - An FD $a \rightarrow b$ is enforced as candidate key
    - `PRIMARY KEY`
    - `NOT NULL` and UNIQUE
  - Candidate key constraints in R_1 cannot enforce an FD $a \rightarrow b$ where $b \in R_2$
  - Projection captures which FDs can be enforced by each fragments

Dependency-preserving decomposition

Definition

- Decomposition $\{R_1, R_2, \dots, R_n\}$ of R is dependency-preserving *if*
  - $\left(F_{R_1} \cup F_{R_2} \cup \cdots \cup F_{R_n}\right)$ *is equivalent to* F
    - $\left(F_{R_1} \cup F_{R_2} \cup \cdots \cup F_{R_n}\right) \equiv F$
    - $\left(F_{R_1} \cup F_{R_2} \cup \cdots \cup F_{R_n}\right)^+ = F^+$
    - $\left(F_{R_1} \cup F_{R_2} \cup \cdots \cup F_{R_n}\right) \vDash F \wedge F \vDash \left(F_{R_1} \cup F_{R_2} \cup \cdots \cup F_{R_n}\right)$

➢ Guarantees that for each update to a decomposed relation $r_i = \pi_{R_i}(r)$, FD violations for r can be detected without computing joins

Dependency-preserving decomposition

Example

- Consider $R(A, B, C)$ with FDs $F = \{A \rightarrow B, B \rightarrow C\}$
  - Is $\{R_1(A, B), R_2(B, C)\}$ dependency-preserving?
- Steps:
  - $F_{AB} = \{ \quad \}$
  - $F_{BC} = \{ \quad \}$
  - $F_{AB} \cup F_{BC} = F$
    - $(F_{AB} \cup F_{BC})^+ = F^+$
- Note:
  - $A \rightarrow C$ is neither in F_{AB} nor in F_{BC}
    - but $A \rightarrow C$ is in $(F_{AB} \cup F_{BC})^+$

Dependency-preserving decomposition

Example

- Consider $R(A, B, C)$ with FDs $F = \{A \to B, B \to C\}$
  - Is $\{R_1(A, B), R_2(B, C)\}$ dependency-preserving?
- Steps:
  - $F_{AB} = \{ \qquad \}$
  - $F_{BC} = \{ \qquad \}$
  - $F_{AB} \cup F_{BC} = F$
    - $(F_{AB} \cup F_{BC})^+ = F^+$

R

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

R1

A	B
a1	b1
a2	b2
a1	b1

R2

B	C
b1	c1
b2	c2
b1	c2

Dependency-preserving decomposition

Example

- Consider $R(A, B, C)$ with FDs $F = \{A \rightarrow B, A \rightarrow C\}$
  - Is $\{R_1(A, B), R_2(B, C)\}$ dependency-preserving?
- Steps:
  - $F_{AB} = \{ \qquad\qquad \}$
  - $F_{BC} = \{ \qquad\qquad \}$
  - $F_{AB} \cup F_{BC} = \{ \qquad\qquad \}$
    - $(F_{AB} \cup F_{BC})^+ = \{ \qquad\qquad \} \neq F^+ = \{A \rightarrow B, A \rightarrow C\}$

Dependency-preserving decomposition

Example

- Consider $R(A, B, C)$ with FDs $F = \{A \rightarrow B, A \rightarrow C\}$
  - Is $\{R_1(A, B), R_2(B, C)\}$ dependency-preserving?
- Steps:
  - $F_{AB} = \{ \qquad\qquad \}$
  - $F_{BC} = \{ \qquad\qquad \}$
  - $F_{AB} \cup F_{BC} = \{ \qquad\qquad \}$
    - $(F_{AB} \cup F_{BC})^+ = \{ \qquad\qquad\qquad \} \neq F^+ = \{A \rightarrow B, A \rightarrow C\}$

R

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

R1

A	***B***
a1	b1
a2	b2
a1	b1

R2

B	***C***
b1	c1
b2	c2
b1	c2

Dependency-preserving decomposition

Checking

- Given a decomposition $\{R_1, R_2, \dots, R_n\}$ of R, how to check if it is a dependency-preserving decomposition
  - $\left(F_{R_1} \cup F_{R_2} \cup \cdots \cup F_{R_n}\right)$ *is equivalent to* F
    - $\left(F_{R_1} \cup F_{R_2} \cup \cdots \cup F_{R_n}\right) \equiv F$
    - $\left(F_{R_1} \cup F_{R_2} \cup \cdots \cup F_{R_n}\right)^+ = F^+$
    - $\left(F_{R_1} \cup F_{R_2} \cup \cdots \cup F_{R_n}\right) \vDash F \wedge F \vDash \left(F_{R_1} \cup F_{R_2} \cup \cdots \cup F_{R_n}\right)$
  - How to compute F^+ and $\left(F_{R_1} \cup F_{R_2} \cup \cdots \cup F_{R_n}\right)^+$ efficiently?
    - Given lemma 2, we simply have to check $\left(F_{R_1} \cup F_{R_2} \cup \cdots \cup F_{R_n}\right) \vDash F$

Lemma 2

- For every decomposition $\{R_1, R_2, \dots, R_n\}$ of R
  - $F \vDash \left(F_{R_1} \cup F_{R_2} \cup \cdots \cup F_{R_n}\right)$
  - By definition, $F_{R_i} = \{\, b \to c \in F^+ \mid bc \subseteq R_i \,\}$
  - For all $b \to c$ in F_{R_i}, we also have $b \to c$ in F^+

Dependency-preserving decomposition

Checking

- Given a decomposition $\{R_1, R_2, \ldots, R_n\}$ of R, how to check if it is a dependency-preserving decomposition
  - $\left(F_{R_1} \cup F_{R_2} \cup \cdots \cup F_{R_n}\right)$ *is equivalent to* F
    - $\left(F_{R_1} \cup F_{R_2} \cup \cdots \cup F_{R_n}\right) \vDash F$
- Idea
  - Let $G = F_{R_1} \cup F_{R_2} \cup \cdots \cup F_{R_n}$
  - For each $f \in F$
    - Check $G \vDash f$
      - Let $f = a \rightarrow b$
      - Compute a^+ w.r.t. G
      - If $b \subseteq a^+$ then $G \vDash a \rightarrow b$
      - Alternatively, if $b \nsubseteq a^+$ then $G \nequiv F$
  - If pass all $b \subseteq a^+$ then $G \vDash F$

Dependency-preserving decomposition

Checking

- Algorithm #4
- **Input** A decomposition $\{R_1, \ldots, R_n\}$ of R with FDs F
- **Output** *YES* (if dependency-preserving) or *NO* (otherwise)

1. for each $(R_i \in \{R_1, \ldots, R_n\})$
2. compute F_{R_i}
3. let $G = F_{R_1} \cup \cdots \cup F_{R_n}$
4. for each (FD $a \to b \in F$)
5. compute a^+ w.r.t. G
6. if $(b \nsubseteq a^+)$ then return *NO*
7. return *YES*

Summary

- Decomposition
  - Lossless-join
    - $r = \pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \cdots \bowtie \pi_{R_n}(r)$ No information is lost
    - Theorem 1: $F \vDash R_1 \cap R_2 \to R_1 \vee F \vDash R_1 \cap R_2 \to R_2 \Rightarrow \text{lossless}$
      - Can be used to check for lossless-join
    - Corollary 1: Completely non-trivial $a \to b \Rightarrow \{R - b, ab\}$ is lossless
      - Can be used to decompose!
    - Theorem 1: $\text{lossless}(\{R_{1,1}, R_{1,2}\}, R_1) \wedge \text{lossless}(\{R_1, R_2\}, R) \Rightarrow \text{lossless}(\{R_{1,1}, R_{1,2}, R_2\}, R)$
      - Can be used to further decompose
  - Dependency-preserving
    - $(F_{R_1} \cup F_{R_2} \cup \cdots \cup F_{R_n}) \equiv F$ No FD is lost
    - FD can be checked without performing the join
    - Projection: $F_a = \{b \to c \in F^+ \mid bc \subseteq a\}$
    - Algorithm #3: computes F_a
    - Algorithm #4: check $\{R_1, R_2, \dots, R_n\}$ of R is dependency-preserving