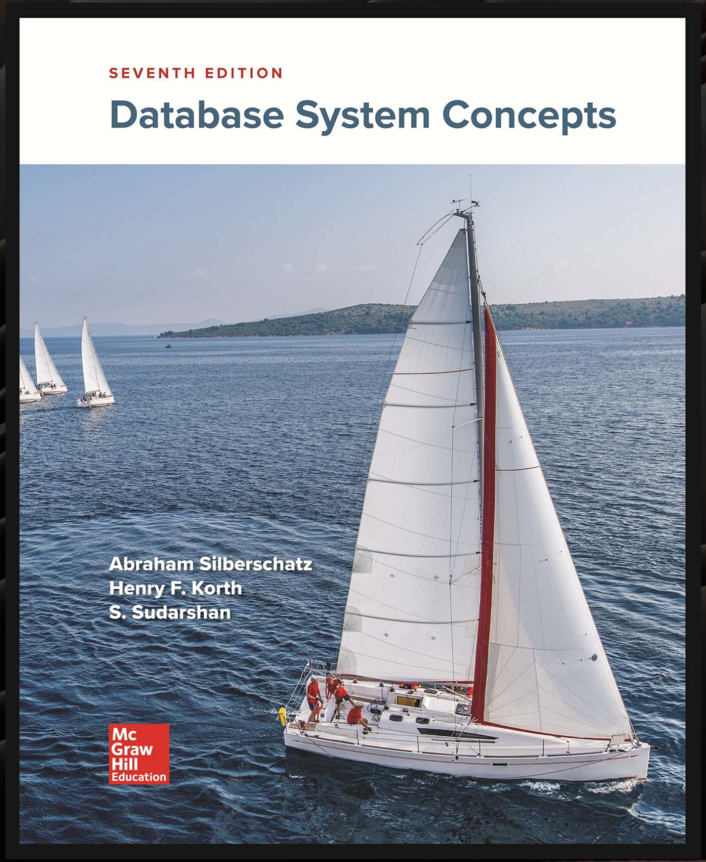
IF2240 – Basis Data

Relational Database Design (part 2b)

KNOWLEDGE & SOFTWARE ENGINEERING



Sumber

- Silberschatz, Korth, Sudarshan: “Database System Concepts”, 7th Edition
  - Chapter 7: Relational Database Design

Capaian

- Mahasiswa dapat menghasilkan desain basis data relasional yang baik berdasarkan prinsip-prinsip yang diberikan

Outline

- Features of Good Relational Design
- Functional Dependencies
- Decomposition Using Functional Dependencies
- Normal Forms
- Functional Dependency Theory
- Algorithms for Decomposition using Functional Dependencies
- Database-Design Process

Canonical Cover

The effort spent in checking for violations can be reduced by testing a simplified set of functional dependencies that has the same closure as the given set.

This simplified set is termed the **canonical cover**

To define canonical cover we must first define **extraneous attributes**.

An attribute of a functional dependency in F is **extraneous** if we can remove it without changing F^+

Extraneous Attributes Left-hand side

Removing an attribute from the lhs of a FD could make it a stronger constraint.

For example, removing B from $AB \rightarrow C$, gives a possibly stronger $A \rightarrow C$

It may be stronger because $A \rightarrow C$ logically implies $AB \rightarrow C$

But, $AB \rightarrow C$ does not, on its own, logically imply $A \rightarrow C$

But, we may be able to remove B from $AB \rightarrow C$ safely.

For example, suppose that $F = \{AB \rightarrow C, A \rightarrow D, D \rightarrow C\}$

Then we can show that F logically implies $A \rightarrow C$,

making B extraneous in $AB \rightarrow C$.

Extraneous Attributes Left-hand side

Consider a set F of functional dependencies and the functional dependency $\alpha \rightarrow \beta$ in F.

Attribute A is **extraneous** in α if

$A \in \alpha$ and

F logically implies $(F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha - A) \rightarrow \beta\}$.

To test if attribute $A \in \alpha$ is extraneous in α

Let $\gamma = \alpha - \{A\}$.

Check if $\gamma \rightarrow \beta$ can be inferred from F.

Compute γ^+ using the dependencies in F

If γ^+ includes all attributes in β then A is extraneous in α

Extraneous Attributes Left-hand side

Consider a set F of functional dependencies and the functional dependency $\alpha \to \beta$ in F.

Attribute A is **extraneous** in α if

$A \in \alpha$ and	F logically implies $(F - \{\alpha \to \beta\}) \cup \{(\alpha - A) \to \beta\}$.

To test if attribute $A \in \alpha$ is extraneous in α

Let $\gamma = \alpha - \{A\}$.	Check if $\gamma \to \beta$ can be inferred from F.

- Compute γ^+ using the dependencies in F
- If γ^+ includes all attributes in β then A is extraneous in α

- Let $F = \{AB \to C, A \to B, B \to C\}$
- To check if B is extraneous in $AB \to C$ we check if F implies $F' = \{A \to C, A \to B, B \to C\}$
- Check if $A \to C$ can be inferred from F
  - Compute A^+ under F
  - A^+ is ABC, which includes C
  - This implies that B is extraneous

Extraneous Attributes Right-hand side

Removing an attribute from the rhs of a FD could make it a weaker constraint.

For example, if we have $AB \to CD$ and remove C,

we get the possibly weaker result $AB \to D$.

It may be weaker because using just $AB \to D$, we can no longer infer $AB \to C$.

But, we may be able to remove C from $AB \to CD$ safely.

For example, suppose that $F = \{ AB \to CD, A \to C \}$

Even after replacing $AB \to CD$ by $AB \to D$, we can still infer $AB \to C$

and thus $AB \to CD$.

Extraneous Attributes Right-hand side

Consider a set F of functional dependencies and the functional dependency $\alpha \rightarrow \beta$ in F.

Attribute A is **extraneous** in β if

$A \in \beta$ and	$(F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$ logically implies F

To test if attribute $A \in \beta$ is extraneous in β

Let $F' = (F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$	check that α^+ using F' contains A; if it does, A is extraneous in β

KNOWLEDGE & SOFTWARE ENGINEERING

Extraneous Attributes Right-hand side

Consider a set F of functional dependencies and the functional dependency $\alpha \rightarrow \beta$ in F.

Attribute A is extraneous in β if

$A \in \beta$ and	$(F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$ logically implies F

To test if attribute $A \in \beta$ is extraneous in β

Let $F' = (F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$	check that α^+ using F' contains A; if it does, A is extraneous in β

- Let $F = \{AB \rightarrow CD, A \rightarrow E, E \rightarrow C\}$
- To check if C is extraneous in $AB \rightarrow CD$, we check if $F' = \{AB \rightarrow D, A \rightarrow E, E \rightarrow C\}$ *implies F*
- Check if $AB \rightarrow CD$ can be inferred from F"
  - Compute AB^+ under F'
  - AB^+ is $ABCDE$, which includes CD
  - This implies that C is extraneous

Canonical Cover

A canonical cover for F is a set of dependencies F_c such that

- F logically implies all dependencies in F_c , and
- F_c logically implies all dependencies in F, and
- No functional dependency in F_c contains an extraneous attribute, and
- Each left side of functional dependency in F_c is unique.
  - That is, there are no two dependencies in F_c
    - $\alpha_1 \rightarrow \beta_1$ and $\alpha_2 \rightarrow \beta_2$ such that
    - $\alpha_1 = \alpha_2$

Canonical Cover

Algorithm to compute a canonical cover for F

$F_c := F$

repeat

Use the union rule to replace any dependencies in F_c of the form $\alpha_1 \rightarrow \beta_1$ and $\alpha_1 \rightarrow \beta_2$ with $\alpha_1 \rightarrow \beta_1 \beta_2$

Find a functional dependency $\alpha \rightarrow \beta$ in F_c with an extraneous attribute either in α or in β

/* Note: test for extraneous attributes done using F_c, not F*/

If an extraneous attribute is found, delete it from $\alpha \rightarrow \beta$

until (F_c not change)

Note: Union rule may become applicable after some extraneous attributes have been deleted, so it has to be re-applied

Canonical Cover

F_c := F

repeat

Use the union rule to replace any dependencies in F_c of the form $\alpha_1 \rightarrow \beta_1$ and $\alpha_1 \rightarrow \beta_2$ with $\alpha_1 \rightarrow \beta_1 \beta_2$

Find a functional dependency $\alpha \rightarrow \beta$ in F_c with an extraneous attribute either in α or in β

/* Note: test for extraneous attributes done using F_c, not F*/

If an extraneous attribute is found, delete it from $\alpha \rightarrow \beta$

until (F_c not change)

Note: Union rule may become applicable after some extraneous attributes have been deleted, so it has to be re-applied

$R = (A, B, C)$

$F = \{A \rightarrow BC, \ B \rightarrow C, \ A \rightarrow B, \ AB \rightarrow C\}$

Combine $A \rightarrow BC$ and $A \rightarrow B$ into $A \rightarrow BC$

- Set is now $\{A \rightarrow BC, B \rightarrow C, AB \rightarrow C\}$

A is extraneous in $AB \rightarrow C$

- Set is now $\{A \rightarrow BC, B \rightarrow C\}$

C is extraneous in $A \rightarrow BC$

- Set is now $\{A \rightarrow B, B \rightarrow C\}$

The canonical cover is: $\{ A \rightarrow B, \ B \rightarrow C \}$

KNOWLEDGE & SOFTWARE ENGINEERING

Dependency Preservation

- Let F_i be the set of dependencies F^+ that include only attributes in R_i.
- A decomposition is **dependency preserving**, if $(F_1 \cup F_2 \cup \ldots \cup F_n)^+ = F^+$
- Using the above definition, testing for dependency preservation take exponential time.

Dependency Preservation

Let F_i be the set of dependencies F^+ that include only attributes in R_i.

A decomposition is **dependency preserving**, if $(F_1 \cup F_2 \cup \ldots \cup F_n)^+ = F^+$

Using the above definition, testing for dependency preservation take exponential time.

Let F be the set of dependencies on schema R and let $R_1, R_2, \ldots, R_n$ be a decomposition of R.

The restriction of F to R_i is the set F_i of all functional dependencies in F^+ that include **only** attributes of R_i.

Note that the definition of restriction uses all dependencies in F^+, not just those in F.

The set of restrictions $F_1, F_2, \ldots, F_n$ is the set of functional dependencies that can be checked efficiently.

Testing for Dependency Preservation

- To check if a dependency $\alpha \rightarrow \beta$ is preserved in a decomposition of R into R_1, R_2, …, R_n , apply the following test (with attribute closure done with respect to F)
- Apply the test on all dependencies in F to check if a decomposition is dependency preserving
- This procedure takes polynomial time

```
result = α
repeat
    for each Ri in the decomposition
        t = (result ∩ Ri)+ ∩ Ri
        result = result ∪ t
until (result does not change)
```

If *result* contains all attributes in β, then the functional dependency $\alpha \rightarrow \beta$ is preserved.

Testing for Dependency Preservation

```
result = α
repeat
    for each Ri in the decomposition
        t = (result ∩ Ri)+ ∩ Ri
        result = result ∪ t
until (result does not change)
```

If *result* contains all attributes in β, then the functional dependency $\alpha \rightarrow \beta$ is preserved.

$R = (A, B, C)$

$F = \{A \rightarrow B, \ B \rightarrow C\}$

Key $= \{A\}$

R is not in BCNF

Decomposition $R_1 = (A, B), \ F_1 = \{A \rightarrow B\}$

$R_2 = (B, C), \ F_2 = \{B \rightarrow C\}$

- R_1 and R_2 in BCNF
- Lossless-join decomposition
- Dependency preserving

KNOWLEDGE & SOFTWARE ENGINEERING

Testing for Dependency Preservation

```
result = α
repeat
    for each Ri in the decomposition
        t = (result ∩ Ri)+ ∩ Ri
        result = result ∪ t
until (result does not change)
```

If *result* contains all attributes in β, then the functional dependency $\alpha \rightarrow \beta$ is preserved.

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

R is not in BCNF

result = B
R_1:
$(\text{result} \cap R_1)^+ = B^+ = BC$
$\text{result} = B \cup (BC \cap R_1) = B$
R_2:
$(\text{result} \cap R_2) = \emptyset$
result = B

Decomposition $R_1 = (A, B),\ \ F_1 = \{A \rightarrow B\}$
$R_2 = (A, C),\ \ F_2 = \{A \rightarrow C\}$

- R_1 and R_2 in BCNF
- Lossless-join decomposition
- Not Dependency preserving
  - *$A \rightarrow B$ is preserved*
  - *$B \rightarrow C$ is not preseved*

KNOWLEDGE & SOFTWARE ENGINEERING

Algorithm for Decomposition Using Functional Dependencies

NEXT MEETING…