Chapter 8: Relational Database Design

Database System Concepts, 6th Ed.

Chapter 8: Relational Database Design

- Features of Good Relational Design
- Atomic Domains and First Normal Form
- Decomposition Using Functional Dependencies
- Boyce-Codd Normal Form
- Third Normal Form
- Fourth Normal Form
- More Normal Forms
- Database-Design Process
- Functional Dependency Theory

Combine Schemas?

- Suppose we combine *instructor* and *department* into *inst_dept*
- Result is possible **repetition** of information

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

How to split a Schema?

- How to split *inst_dept (<u>ID</u>, name, salary, dept_name, building, budget)?*

How to split a Schema?

- How to split *inst_dept (ID, name, salary, dept_name, building, budget)?*
  - *dept(dept_name, building, budget*)

How to split a Schema?

- How to split *inst_dept (ID, name, salary, dept_name, building, budget)?*
  - *dept(dept_name, building, budget)*
  - *inst(ID, name, salary, dept_name)*

How to split a Schema?

- How to split *inst_dept (ID, name, salary, dept_name, building, budget)?*
  - *dept(dept_name, building, budget*)
  - *inst(ID, name, salary, dept_name)*
- Why?

How to split a Schema?

- How to split *inst_dept (ID, name, salary, dept_name, building, budget)?*
  - *dept(dept_name, building, budget)*
  - *inst(ID, name, salary, dept_name)*
- Why?
  - **functional dependency**:

How to split a Schema?

- How to split *inst_dept (ID, name, salary, dept_name, building, budget)?*
  - *dept(dept_name, building, budget)*
  - *inst(ID, name, salary, dept_name)*
- Why?
  - **functional dependency**:

    dept_name $\rightarrow$ *building, budget*

How to split a Schema?

- How to split *inst_dept (ID, name, salary, dept_name, building, budget)?*
  - *dept(dept_name, building, budget*)
  - *inst(ID, name, salary, dept_name)*
- Why?
  - **functional dependency**:

    dept_name → *building, budget*

- How about splitting *inst(ID, name, salary, dept_name)* into the following?
  - *inst1(ID, name)*
  - *inst2*(*name, salary, dept_name*)

A Lossy Decomposition

ID	name	street	city	salary
⋮				
57766	Kim	Main	Perryridge	75000
98776	Kim	North	Hampton	67000
⋮				

employee

ID	name
⋮	
57766	Kim
98776	Kim
⋮	

name	street	city	salary
⋮			
Kim	Main	Perryridge	75000
Kim	North	Hampton	67000
⋮			

natural join

ID	name	street	city	salary
⋮				
57766	Kim	Main	Perryridge	75000
57766	Kim	North	Hampton	67000
98776	Kim	Main	Perryridge	75000
98776	Kim	North	Hampton	67000
⋮				

Tuesday, March 26, 2013

Lossless-join Decomposition

- A decomposition of R into R_1 and R_2 is **lossless** if at least one of the following dependencies holds (**common attributes** form a **superkey** in R_1 or R_2) :
  - $R_1 \cap R_2 \rightarrow R_1$
  - $R_1 \cap R_2 \rightarrow R_2$

Goal — Devise a Theory for the Following

- Decide whether a particular relation R is in "good" form.
- If R is not in "good" form, decompose it into a set of relations $\{R_1, R_2, ..., R_n\}$ such that
  - each relation is in good form
  - the decomposition is **lossless**
- Our theory is based on:
  - **functional** dependencies
  - **multivalued** dependencies

- Features of Good Relational Design
- **Atomic Domains and First Normal Form**
- Decomposition Using Functional Dependencies
- Boyce-Codd Normal Form
- Third Normal Form
- Fourth Normal Form
- More Normal Forms
- Database-Design Process
- Functional Dependency Theory

First Normal Form

- Domain is **atomic** if its elements are considered to be indivisible units
  - Examples of non-atomic domains:
    - set of names, composite attributes
- *A relational schema R is in* ***first normal form*** *if the domains of all attributes of R are atomic*

Chapter 8: Relational Database Design

- Features of Good Relational Design
- Atomic Domains and First Normal Form
- **Decomposition Using Functional Dependencies**
- Boyce-Codd Normal Form
- Third Normal Form
- Fourth Normal Form
- More Normal Forms
- Database-Design Process
- Functional Dependency Theory

Functional Dependencies

- **The value for a certain set of attributes determines uniquely the value for another set of attributes.**
  - E.g: SSN → name
- Let *R* be a relation schema

$$\alpha \subseteq R \text{ and } \beta \subseteq R$$

- The **functional dependency**

$$\alpha \rightarrow \beta$$

if and only if

$$t_1[\alpha] = t_2[\alpha] \;\Rightarrow\; t_1[\beta] = t_2[\beta]$$

- Example: Consider *r*(A,*B*) with the following instance of *r.*

1	4
1	5
3	7

- $A \rightarrow B$ *or* $B \rightarrow A$*?*

Functional Dependencies

- **The value for a certain set of attributes determines uniquely the value for another set of attributes.**
  - E.g: SSN → name
- Let *R* be a relation schema

$$\alpha \subseteq R \text{ and } \beta \subseteq R$$

- The **functional dependency**

$$\alpha \rightarrow \beta$$

if and only if

$$t_1[\alpha] = t_2[\alpha] \Rightarrow t_1[\beta] = t_2[\beta]$$

- Example: Consider *r*(A,*B*) with the following instance of *r.*

1	4
1	5
3	7

- $A \rightarrow B$ or $B \rightarrow A$?

  $B \rightarrow A$

Functional Dependencies (Cont.)

- *A* functional dependency is **trivial** if it is satisfied by all instances of a relation
  - Example*:*
    - *name* $\rightarrow$ *name*
    - *ID, name* $\rightarrow$ *ID*
  - In general, $\alpha \rightarrow \beta$ is trivial if $\beta \subseteq \alpha$

Closure of a Set of Functional Dependencies

- Given a set F of functional dependencies, other functional dependencies are implied by F.
  - If $A \rightarrow B$ and $B \rightarrow C$, then $A \rightarrow C$
- The set of **all** functional dependencies implied by F is the **closure** of F.
- We denote the *closure* of F by $\mathbf{F^+}$.
- *Is* F^+ a superset of F?

Closure of a Set of Functional Dependencies

- Given a set F of functional dependencies, other functional dependencies are implied by F.
  - If $A \rightarrow B$ and $B \rightarrow C$, then $A \rightarrow C$
- The set of **all** functional dependencies implied by F is the **closure** of F.
- We denote the *closure* of F by F^+.
- *Is* F^+ a superset of F?
  - yes

Boyce-Codd Normal Form

A relation schema R is in BCNF
with respect to a set F of functional dependencies
if for **each nontrivial** FD $\alpha \rightarrow \beta$ in F^+, α is a superkey for R

Is the following in BCNF?

instr_dept (<u>*ID*</u>, *name, salary,* <u>*dept_name*</u>, *building, budget*)

Boyce-Codd Normal Form

A relation schema *R* is in BCNF
with respect to a set *F* of functional dependencies
if for **each nontrivial** FD $\alpha \rightarrow \beta$ in F^+, α is a superkey for *R*

Is the following in BCNF?

instr_dept (*ID, name, salary, dept_name, building, budget*)

No, because *dept_name→ building, budget,*
but *dept_name* is not a superkey

Decomposing a Schema into BCNF

Given:

instr_dept (*ID, name, salary, dept_name, building, budget*)

and

dept_name → *building, budget,*

How to split *instr_dept?*

Decomposing a Schema into BCNF

Given:

instr_dept (<u>*ID,*</u> *name, salary,* <u>*dept_name,*</u> *building, budget*)

and

dept_name→ *building, budget,*

How to split *instr_dept?*

dept (<u>*dept_name,*</u> *building, budget*)

Decomposing a Schema into BCNF

Given:

instr_dept (*ID, name, salary, dept_name, building, budget*)

and

dept_name→ *building, budget,*

How to split *instr_dept?*

dept (*dept_name, building, budget*)   *instr(ID, name, salary, dept_name)*

Decomposing a Schema into BCNF

Given:

instr_dept (*ID, name, salary, dept_name, building, budget*)

and

dept_name→ *building, budget,*

How to split *instr_dept?*

dept (*dept_name, building, budget*) *instr*(*ID, name, salary, dept_name)*

Is it a ***lossless decomposion****?*

Decomposing a Schema into BCNF

Given:

instr_dept (*ID, name, salary, dept_name, building, budget*)

and

dept_name→ *building, budget,*

How to split *instr_dept?*

dept (*dept_name, building, budget*) *instr*(*ID, name, salary, dept_name)*

*Is it a **lossless decomposion**?*

- Given schema *R* and a non-trivial dependency $\alpha \rightarrow \beta$ that causes a violation of BCNF.

  We decompose *R* into:

  - (α U β)
  - (*R* - (β - α))

Decomposing a Schema into BCNF

Given:

instr_dept (ID, name, salary, dept_name, building, budget)

and

dept_name $\rightarrow$ *building, budget,*

How to split *instr_dept?*

dept (dept_name, building, budget) *instr(ID, name, salary, dept_name)*

*Is it a **lossless decomposion**?*

- Given schema *R* and a non-trivial dependency $\alpha \rightarrow \beta$ that causes a violation of BCNF.

  We decompose *R* into:
  - $(\alpha \cup \beta)$
  - $(R - (\beta - \alpha))$

- In our example,
  - α = *dept_name*
  - β = *building, budget*
  - $(\alpha \cup \beta)$ = (***dept_name***, *building, budget*)
  - $(R - (\beta - \alpha))$ = (*ID, name, salary,* ***dept_name***)

BCNF and Dependency Preservation

- Functional dependencies are costly to check unless they pertain to only one relation
  - e.g., Given R1=(J, K), R2=(K, L),
    F={$JK \rightarrow L$} is expensive to check since it requires R1 $\bowtie$ R2
- If it is sufficient to test functional dependencies on each individual relation of a decomposition, then that decomposition is ***dependency preserving***.
- It is not always possible to achieve both BCNF and dependency preservation

e.g.,

$$R(J, K, L)$$
$$F = \{JK \rightarrow L, \quad L \rightarrow K\}$$

BCNF and Dependency Preservation

- Functional dependencies are costly to check unless they pertain to only one relation
  - e.g., Given R1=(J, K), R2=(K, L),
    F={$JK \rightarrow L$} is expensive to check since it requires R1 $\bowtie$ R2
- If it is sufficient to test functional dependencies on each individual relation of a decomposition, then that decomposition is ***dependency preserving***.
- It is not always possible to achieve both BCNF and dependency preservation

e.g.,

$R(J, K, L)$ $\qquad$ $R1(L, K)$

$$F = \{JK \rightarrow L,$$
$$L \rightarrow K\}$$

BCNF and Dependency Preservation

- Functional dependencies are costly to check unless they pertain to only one relation
  - e.g., Given R1=(J, K), R2=(K, L),
    F={$JK \rightarrow L$} is expensive to check since it requires R1 $\bowtie$ R2
- If it is sufficient to test functional dependencies on each individual relation of a decomposition, then that decomposition is ***dependency preserving***.
- It is not always possible to achieve both BCNF and dependency preservation

e.g.,

$R(J, K, L)$ $\quad\quad$ $R1(L, K)$

$F = \{JK \rightarrow L,$

$\quad\quad L \rightarrow K\}$ $\quad\quad$ $R2(J, L)$

BCNF and Dependency Preservation

- Functional dependencies are costly to check unless they pertain to only one relation
  - e.g., Given R1=(J, K), R2=(K, L),
    F={$JK \rightarrow L$} is expensive to check since it requires R1 $\bowtie$ R2
- If it is sufficient to test functional dependencies on each individual relation of a decomposition, then that decomposition is ***dependency preserving***.
- It is not always possible to achieve both BCNF and dependency preservation

e.g.,

$R(J, K, L)$

$F = \{JK \rightarrow L,$
$\quad L \rightarrow K\}$

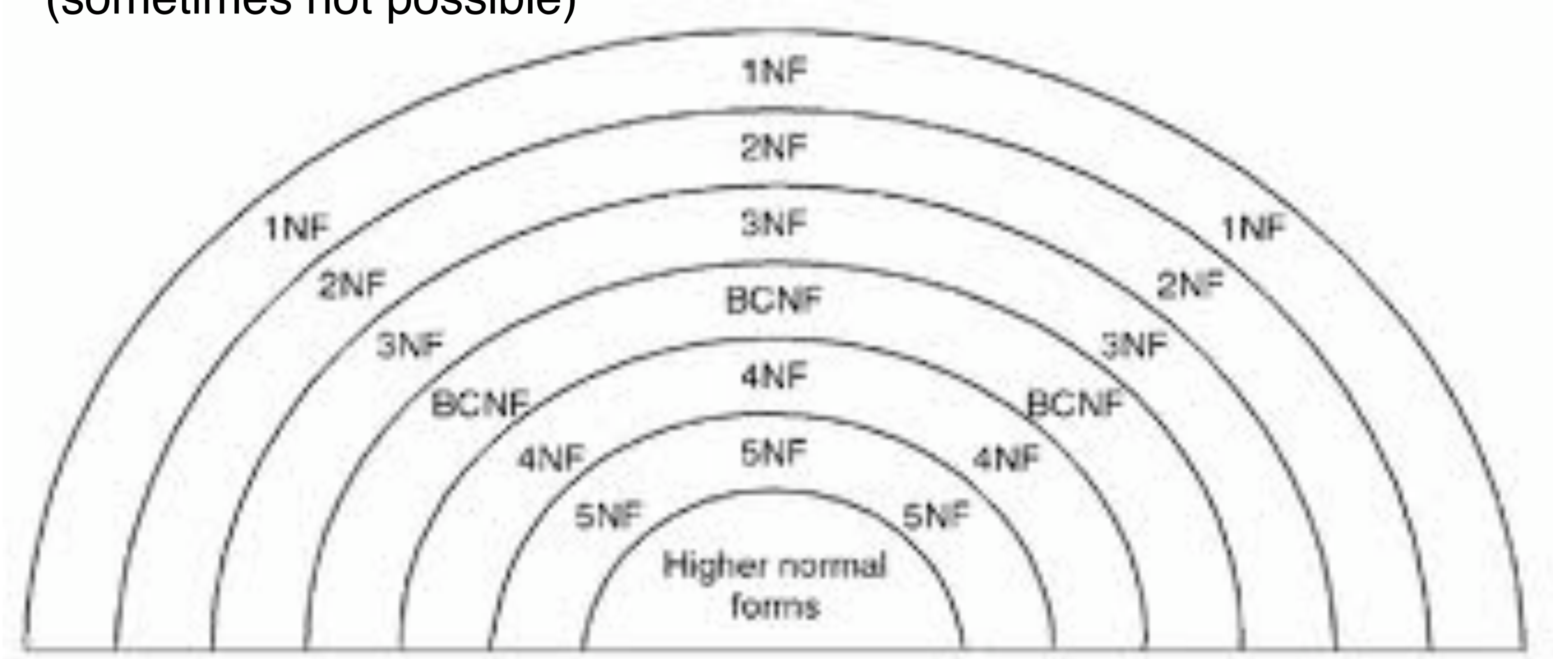
$R1(L, K)$

$R2(J, L)$

- We can consider a weaker normal form, known as *third normal form.*

Goals of Normalization

- Let R be a relation schema with a set F of functional dependencies.
- Decide whether R is in "good" form.
- If R is not in "good" form, decompose it into a set of relation schema $\{R_1, R_2, ..., R_n\}$ such that
  - each relation schema is in good form
  - the decomposition is a lossless-join decomposition
  - Preferably, the decomposition should be dependency preserving (sometimes not possible)



Chapter 8: Relational Database Design

- Features of Good Relational Design
- Atomic Domains and First Normal Form
- Decomposition Using Functional Dependencies
- Boyce-Codd Normal Form
- **Third Normal Form**
- Fourth Normal Form
- More Normal Forms
- Database-Design Process
- Functional Dependency Theory

Third Normal Form

- A relation schema *R* is in **third normal form (3NF)** if for all:

  $$\alpha \rightarrow \beta \text{ in } F^+$$

  at least one of the following holds:
  - $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \in \alpha$)
  - α is a superkey for *R*
  - **Each attribute *A* in $\beta - \alpha$ is contained in a candidate key for *R*.**
- Third condition is a relaxation of BCNF to ensure dependency preservation.
  - *R (J, L, K)*
$F = \{\boldsymbol{JK} \rightarrow L,\ \ L \rightarrow \boldsymbol{K}\}$
  - Want to say that $L \rightarrow K$ *is OK.*
- If a relation is in BCNF it is in 3NF (since in BCNF one of the first two conditions above must hold).

Chapter 8: Relational Database Design

- Features of Good Relational Design
- Atomic Domains and First Normal Form
- Decomposition Using Functional Dependencies
- Boyce-Codd Normal Form
- Third Normal Form
- **Fourth Normal Form**
- More Normal Forms
- Database-Design Process
- Functional Dependency Theory

How good is BCNF?

- Consider a relation

  inst_info (ID, child_name, phone)

  - where an instructor may have multiple phones (512-555-1234 and 512-555-4321) and multiple children (David and William)

ID	*child_name*	*phone*
99999	David	512-555-1234
99999	David	512-555-4321
99999	William	512-555-1234
99999	William	512-555-4321

- There are no non-trivial functional dependencies and therefore the relation is in BCNF
- But, there is redundancy. Solution?

How good is BCNF?

- Consider a relation

  inst_info (ID, child_name, phone)

  - where an instructor may have multiple phones (512-555-1234 and 512-555-4321) and multiple children (David and William)

ID	*child_name*	*phone*
99999	David	512-555-1234
99999	David	512-555-4321
99999	William	512-555-1234
99999	William	512-555-4321

- There are no non-trivial functional dependencies and therefore the relation is in BCNF
- But, there is redundancy. Solution?
- Decompose *inst_info* into inst_child(ID, child_name) and inst_phone(ID, phone)

Fourth Normal Form

- A relation schema *R* is in **4NF** with respect to a set *D* of functional and multivalued dependencies if for all multivalued dependencies of the form $\alpha \twoheadrightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of the following holds:
  - $\alpha \twoheadrightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$ or $\alpha \cup \beta = R$)
  - α is a superkey for schema *R*
- If a relation is in 4NF it is in BCNF

Chapter 8: Relational Database Design

- Features of Good Relational Design
- Atomic Domains and First Normal Form
- Decomposition Using Functional Dependencies
- Boyce-Codd Normal Form
- Third Normal Form
- Fourth Normal Form
- **More Normal Forms**
- Database-Design Process
- Functional Dependency Theory

Further Normal Forms

- **Join dependencies** generalize multivalued dependencies
  - lead to **project-join normal form (PJNF)** (also called **fifth normal form**)
- A class of even more general constraints, leads to a normal form called **domain-key normal form**.
- Problem with these generalized constraints: are hard to reason with, and no set of sound and complete set of inference rules exists.
- Hence rarely used

Chapter 8: Relational Database Design

- Features of Good Relational Design
- Atomic Domains and First Normal Form
- Decomposition Using Functional Dependencies
- Boyce-Codd Normal Form
- Third Normal Form
- Fourth Normal Form
- More Normal Forms
- **Database-Design Process**
- Functional Dependency Theory

ER Model and Normalization

- When an E-R diagram is carefully designed, the tables generated from the E-R diagram should not need further normalization.
- However, in a real (imperfect) design, there can be functional dependencies from non-key attributes to other attributes
  - An *employee* entity with attributes *department_name* and *building*, and with a FD *department_name* → *building*
  - Good design would have made department an entity

Denormalization for Performance

- May want to use non-normalized schema for performance (no join)
  - faster lookup
  - extra space and extra execution time for updates

Chapter 8: Relational Database Design

- Features of Good Relational Design
- Atomic Domains and First Normal Form
- Decomposition Using Functional Dependencies
- Boyce-Codd Normal Form
- Third Normal Form
- Fourth Normal Form
- More Normal Forms
- Database-Design Process
- **Functional Dependency Theory**

Closure of a Set of Functional Dependencies

- We can find F^{+}, the closure of F, by repeatedly applying **Armstrong's Axioms:**
  - if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$ (**reflexivity**)
    - e.g., {ID, name} → ID
  - if $\alpha \rightarrow \beta$, then $\gamma\, \alpha \rightarrow \gamma\, \beta$ (**augmentation**)
    - e.g., If ID → name, then {ID, address} → {name, address}
  - if $\alpha \rightarrow \beta$, and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$ (**transitivity**)
    - e.g., If ID → SSN and SSN → name, then ID → name

Closure of Functional Dependencies (Cont.)

- Additional rules:
  - If $\alpha \rightarrow \beta$ holds *and* $\alpha \rightarrow \gamma$ holds, then $\alpha \rightarrow \beta\,\gamma$ holds (**union**)
    - e.g., If ID → name and ID → address, then ID → {name, address}
  - If $\alpha \rightarrow \beta\,\gamma$ holds, then $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds (**decomposition**)
    - e.g., If ID → {name, address}, then ID → name and ID → address
  - If $\alpha \rightarrow \beta$ holds *and* $\gamma\,\beta \rightarrow \delta$ holds, then $\alpha\,\gamma \rightarrow \delta$ holds (**pseudotransitivity**)
    - e.g., If ID → name and {name, address} → phone_number, then {ID, address} → phone_number.

The above rules can be inferred from Armstrong's axioms.

Closure of Attribute Sets

- Given a set of attributes α, define the ***closure*** of α **under** *F* (denoted by α^+) as the set of attributes that are functionally determined by α under *F*

- Algorithm to compute α^+, the closure of α under *F*

> *result* := α;
> **while** (changes to *result*) **do**
>   **for each** $\beta \rightarrow \gamma$ **in** *F* **do**
>   **begin**
>   **if** $\beta \subseteq$ *result* **then** *result* := *result* $\cup\ \gamma$
>   **end**

Example of Attribute Set Closure

- $R = (A, B, C, G, H, I)$
- $F = \{A \rightarrow B$
  $A \rightarrow C$
  $CG \rightarrow H$
  $CG \rightarrow I$
  $B \rightarrow H\}$
- $(AG)^+$
  1. AG

Example of Attribute Set Closure

- $R = (A, B, C, G, H, I)$
- $F = \{A \rightarrow B$
  $A \rightarrow C$
  $CG \rightarrow H$
  $CG \rightarrow I$
  $B \rightarrow H\}$
- $(AG)^+$
  1. AG
  2. $ABCG$ $(A \rightarrow C$ and $A \rightarrow B)$

Example of Attribute Set Closure

- $R = (A, B, C, G, H, I)$
- $F = \{A \rightarrow B$
  $A \rightarrow C$
  $CG \rightarrow H$
  $CG \rightarrow I$
  $B \rightarrow H\}$
- $(AG)^+$
  1. AG
  2. $ABCG$ $\quad$ ($A \rightarrow C$ and $A \rightarrow B$)
  3. $ABCGH$ $\quad$ ($CG \rightarrow H$)

Example of Attribute Set Closure

- $R = (A, B, C, G, H, I)$
- $F = \{A \rightarrow B$
  $A \rightarrow C$
  $CG \rightarrow H$
  $CG \rightarrow I$
  $B \rightarrow H\}$
- $(AG)^+$
  1. AG
  2. $ABCG$ $(A \rightarrow C$ and $A \rightarrow B)$
  3. $ABCGH$ $(CG \rightarrow H)$
  4. $ABCGHI$ $(CG \rightarrow I)$

Example of Attribute Set Closure

- $R = (A, B, C, G, H, I)$
- $F = \{A \rightarrow B$
  $A \rightarrow C$
  $CG \rightarrow H$
  $CG \rightarrow I$
  $B \rightarrow H\}$
- $(AG)^+$
  1. AG
  2. $ABCG$ $(A \rightarrow C$ and $A \rightarrow B)$
  3. $ABCGH$ $(CG \rightarrow H)$
  4. $ABCGHI$ $(CG \rightarrow I)$
- Is AG a candidate key?

Example of Attribute Set Closure

- $R = (A, B, C, G, H, I)$
- $F = \{A \rightarrow B$
  $A \rightarrow C$
  $CG \rightarrow H$
  $CG \rightarrow I$
  $B \rightarrow H\}$
- $(AG)^+$
  1. AG
  2. $ABCG$ $(A \rightarrow C$ and $A \rightarrow B)$
  3. $ABCGH$ $(CG \rightarrow H)$
  4. $ABCGHI$ $(CG \rightarrow I)$
- Is AG a candidate key?
  1. Is AG a super key?

Example of Attribute Set Closure

- $R = (A, B, C, G, H, I)$
- $F = \{A \rightarrow B$
  $A \rightarrow C$
  $CG \rightarrow H$
  $CG \rightarrow I$
  $B \rightarrow H\}$
- $(AG)^+$
  1. AG
  2. $ABCG$ $(A \rightarrow C$ and $A \rightarrow B)$
  3. $ABCGH$ $(CG \rightarrow H)$
  4. $ABCGHI$ $(CG \rightarrow I)$
- Is AG a candidate key?
  1. Is AG a super key?
     1. Is $AG \rightarrow R$? (Is $(AG)^+ \supseteq R$?)

Example of Attribute Set Closure

- $R = (A, B, C, G, H, I)$
- $F = \{A \rightarrow B$
  $A \rightarrow C$
  $CG \rightarrow H$
  $CG \rightarrow I$
  $B \rightarrow H\}$
- $(AG)^+$
  1. AG
  2. $ABCG$ $(A \rightarrow C$ and $A \rightarrow B)$
  3. $ABCGH$ $(CG \rightarrow H)$
  4. $ABCGHI$ $(CG \rightarrow I)$
- Is AG a candidate key?
  1. Is AG a super key?
     1. Is $AG \rightarrow R$? (Is $(AG)^+ \supseteq R$?)
  2. Is any subset of AG a superkey?

Tuesday, March 26, 2013

Example of Attribute Set Closure

- $R = (A, B, C, G, H, I)$
- $F = \{A \rightarrow B$
  $A \rightarrow C$
  $CG \rightarrow H$
  $CG \rightarrow I$
  $B \rightarrow H\}$
- $(AG)^+$
  1. AG
  2. $ABCG$ $\quad (A \rightarrow C$ and $A \rightarrow B)$
  3. $ABCGH$ $\quad (CG \rightarrow H)$
  4. $ABCGHI$ $\quad (CG \rightarrow I)$
- Is AG a candidate key?
  1. Is AG a super key?
     1. Is $AG \rightarrow R$? (Is $(AG)^+ \supseteq R$?)
  2. Is any subset of AG a superkey?
     1. Is $A \rightarrow R$? (Is $(A)^+ \supseteq R$?)

Example of Attribute Set Closure

- $R = (A, B, C, G, H, I)$
- $F = \{A \rightarrow B$
  $A \rightarrow C$
  $CG \rightarrow H$
  $CG \rightarrow I$
  $B \rightarrow H\}$
- $(AG)^+$
  1. AG
  2. $ABCG$ $(A \rightarrow C$ and $A \rightarrow B)$
  3. $ABCGH$ $(CG \rightarrow H)$
  4. $ABCGHI$ $(CG \rightarrow I)$
- Is AG a candidate key?
  1. Is AG a super key?
     1. Is $AG \rightarrow R$? (Is $(AG)^+ \supseteq R$?)
  2. Is any subset of AG a superkey?
     1. Is $A \rightarrow R$? (Is $(A)^+ \supseteq R$?)
     2. Is $G \rightarrow R$? (Is $(G)^+ \supseteq R$?)

Uses of Attribute Closure

- Testing for superkey:
  - To test if α is a superkey, we compute $\alpha^{+,}$ and check if α^+ contains all attributes of *R*.
- Testing functional dependencies
  - To check if a functional dependency $\alpha \rightarrow \beta$ holds, compute α^+ by using attribute closure, and then check if it contains β.
- Computing closure of F
  - For each $\gamma \subseteq R$, we find the closure γ^+.

Canonical Cover

- A canonical cover of F is a “minimal” set of functional dependencies equivalent to F, having no redundant dependencies or redundant parts of dependencies
  - E.g.: on **RHS**: F1=$\{A \rightarrow B,\ \ B \rightarrow C,\ \ A \rightarrow \boldsymbol{C}D\}$ can be simplified to
    F2=$\{A \rightarrow B,\ \ B \rightarrow C,\ \ A \rightarrow D\}$
  - since **A→C in F1 can be inferred from F2** (and F1 trivially implies F2)

  - E.g.: on **LHS**: F1=$\{A \rightarrow B,\ \ B \rightarrow C,\ \ A\boldsymbol{C} \rightarrow D\}$ can be simplified to
    F2=$\{A \rightarrow B,\ \ B \rightarrow C,\ \ A \rightarrow D\}$
  - since $\boldsymbol{A} \rightarrow$ **D in F2 can be inferred from F1** (and F2 trivially implies F1)

Extraneous Attributes

- Example: Given $F = \{A \rightarrow C, A\boldsymbol{B} \rightarrow C\}$
  - Is B is extraneous in $AB \rightarrow C$?

Extraneous Attributes

- Example: Given $F = \{A \rightarrow C, A\mathbf{B} \rightarrow C\}$
  - Is B is extraneous in $AB \rightarrow C$?
  - Yes, because $F' = \{A \rightarrow C\}$ can be implied from F.

Extraneous Attributes

- Example: Given $F = \{A \rightarrow C, A\mathbf{B} \rightarrow C\}$
  - Is B is extraneous in $AB \rightarrow C$?
  - Yes, because $F' = \{A \rightarrow C\}$ can be implied from F.
- Example: Given $F = \{A \rightarrow C, AB \rightarrow \mathbf{C}D\}$
  - Is C is extraneous in $AB \rightarrow CD$?

Extraneous Attributes

- Example: Given $F = \{A \rightarrow C, A\mathbf{B} \rightarrow C\}$
  - Is B is extraneous in $AB \rightarrow C$?
  - Yes, because $F' = \{A \rightarrow C\}$ can be implied from F.
- Example: Given $F = \{A \rightarrow C, AB \rightarrow \mathbf{C}D\}$
  - Is C is extraneous in $AB \rightarrow CD$?
  - Yes, since F can be implied from $F' = \{A \rightarrow C, AB \rightarrow D\}$.

Extraneous Attributes

- Example: Given $F = \{A \rightarrow C, A\mathbf{B} \rightarrow C\}$
  - Is B is extraneous in $AB \rightarrow C$?
  - Yes, because $F' = \{A \rightarrow C\}$ can be implied from F.
- Example: Given $F = \{A \rightarrow C, AB \rightarrow \mathbf{C}D\}$
  - Is C is extraneous in $AB \rightarrow CD$?
  - Yes, since F can be implied from $F' = \{A \rightarrow C, AB \rightarrow D\}$.
- 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 $(F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha - A) \rightarrow \beta\}$.
  - Attribute A is **extraneous** in β if $A \in \beta$ and the set of functional dependencies $(F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$ logically implies F.
- *Note:* implication in the opposite direction is trivial in each of the cases above, since a "stronger" functional dependency always implies a weaker one

Computing a Canonical Cover

- $R = (A, B, C)$
  $F = \{A \rightarrow BC$
  $B \rightarrow C$
  $A \rightarrow B$
  $AB \rightarrow C\}$
- 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 $\boldsymbol{A}B \rightarrow C$
  - Check if $B \rightarrow C$ is implied by the other dependencies
    - Yes: in fact, $B \rightarrow C$ is already present!
  - Set is now $\{A \rightarrow BC, B \rightarrow C\}$
- C is extraneous in $A \rightarrow B\boldsymbol{C}$
  - Check if $A \rightarrow C$ is logically implied by $A \rightarrow B$ and the other dependencies
    - Yes: using transitivity on $A \rightarrow B$ *and* $B \rightarrow$ C.
      - Can use attribute closure of A in more complex cases
- The canonical cover is: $A \rightarrow B$
  $B \rightarrow C$

Chapter 8: Relational Database Design

- Features of Good Relational Design
- Atomic Domains and First Normal Form
- Decomposition Using Functional Dependencies
- **Boyce-Codd Normal Form**
- Third Normal Form
- **Fourth Normal Form**
- More Normal Forms
- Database-Design Process
- **Functional Dependency Theory**