

CS2102

Database Systems

L09: Anomalies and Functional Dependencies

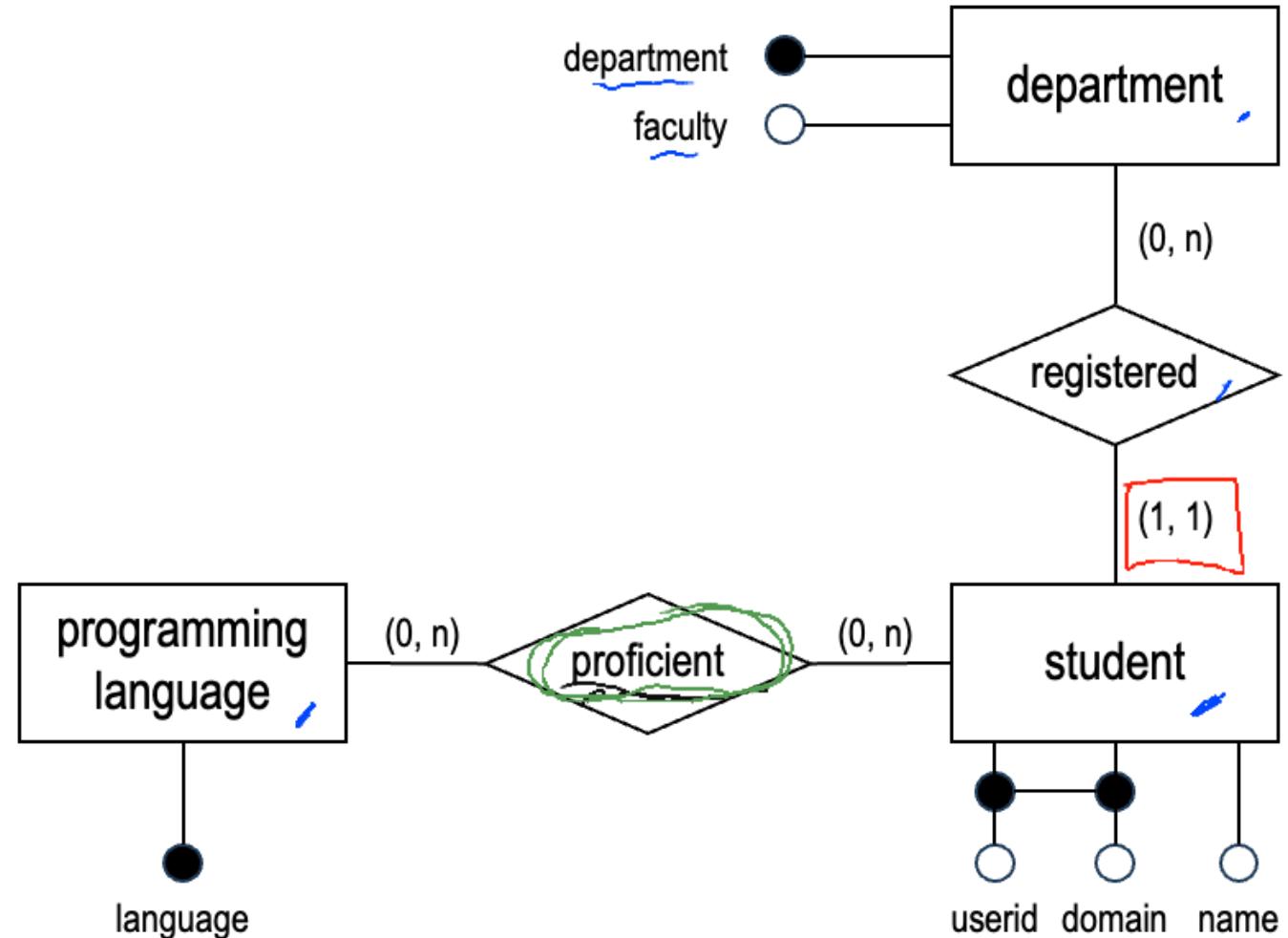
Requirement

Proficiency

Sentosa University of Technology (SUT) records the **programming language skills** of the **students** of its different **faculties** and **departments**.

Question

How many tables do we need to enforce all constraints?



Schema

One Table

We store everything in **one table (proficiency)**. What could go wrong?

| name | userid | domain | department | faculty | language |
|-------------|--------|--------------|----------------------|-------------|------------|
| Tan Hee Wee | tanh | comp.sut.edu | computer science | computing | JavaScript |
| Tan Hee Wee | tanh | comp.sut.edu | computer science | computing | Python |
| Tan Hee Wee | tanh | comp.sut.edu | computer science | computing | C++ |
| Tan Hee Wee | tanhw | eng.sut.edu | computer engineering | engineering | C++ |
| Tan Hee Wee | tanhw | eng.sut.edu | computer engineering | engineering | Fortran |
| Bjorn Sale | bjorn | eng.sut.edu | computer engineering | engineering | C++ |
| Bjorn Sale | bjorn | eng.sut.edu | computer engineering | engineering | Java |
| Ami Mokhtar | ami | med.sut.edu | pharmacy | medicine | Rust |

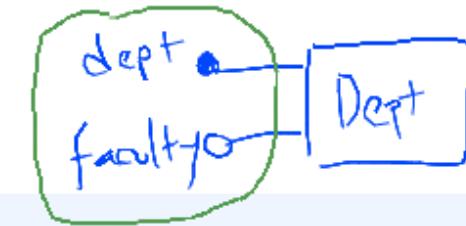
Schema

One Table

department uniquely identify faculty

We store everything in **one table** (proficiency). What could go wrong?

| name | userid | domain | department | faculty | language |
|-------------|--------|--------------|----------------------|-------------|------------|
| Tan Hee Wee | tanh | comp.sut.edu | computer science | computing | JavaScript |
| Tan Hee Wee | tanh | comp.sut.edu | computer science | computing | Python |
| Tan Hee Wee | tanh | comp.sut.edu | computer science | computing | C++ |
| Tan Hee Wee | tanhw | eng.sut.edu | computer engineering | engineering | C++ |
| Tan Hee Wee | tanhw | eng.sut.edu | computer engineering | engineering | Fortran |
| Bjorn Sale | bjorn | eng.sut.edu | computer engineering | engineering | C++ |
| Bjorn Sale | bjorn | eng.sut.edu | computer engineering | engineering | Java |
| Ami Mokhtar | ami | med.sut.edu | pharmacy | medicine | Rust |



Requirement

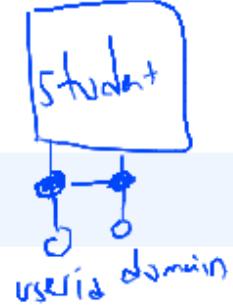
Each faculty is organized into several departments.

A department belongs to one faculty only and there is no two departments with the same name in SUT.

Schema

One Table

We store everything in **one table (proficiency)**. What could go wrong?



| name | userid | domain | department | faculty | language |
|-------------|--------|--------------|----------------------|-------------|------------|
| Tan Hee Wee | tanh | comp.sut.edu | computer science | computing | JavaScript |
| Tan Hee Wee | tanh | comp.sut.edu | computer science | computing | Python |
| Tan Hee Wee | tanh | comp.sut.edu | computer science | computing | C++ |
| Tan Hee Wee | tanhw | eng.sut.edu | computer engineering | engineering | C++ |
| Tan Hee Wee | tanhw | eng.sut.edu | computer engineering | engineering | Fortran |
| Bjorn Sale | bjorn | eng.sut.edu | computer engineering | engineering | C++ |
| Bjorn Sale | bjorn | eng.sut.edu | computer engineering | engineering | Java |
| Ami Mokhtar | ami | med.sut.edu | pharmacy | medicine | Rust |

Requirement

Students are identified by their email. The email of a student is composed of her **[userid and domain.]**

We record the primary department of the student.

Schema

One Table

We store everything in **one table (proficiency)**. What could go wrong?

| name | userid | domain | department | faculty | language |
|-------------|---------------|---------------|----------------------|----------------|-----------------|
| Tan Hee Wee | tanh | comp.sut.edu | computer science | computing | JavaScript |
| Tan Hee Wee | tanh | comp.sut.edu | computer science | computing | Python |
| Tan Hee Wee | tanh | comp.sut.edu | computer science | computing | C++ |
| Tan Hee Wee | tanhw | eng.sut.edu | computer engineering | engineering | C++ |
| Tan Hee Wee | tanhw | eng.sut.edu | computer engineering | engineering | Fortran |
| Bjorn Sale | bjorn | eng.sut.edu | computer engineering | engineering | C++ |
| Bjorn Sale | bjorn | eng.sut.edu | computer engineering | engineering | Java |
| Ami Mokhtar | ami | med.sut.edu | pharmacy | medicine | Rust |

Requirement

For each student, the database records **(only once)** **each programming language** in which the student is proficient.

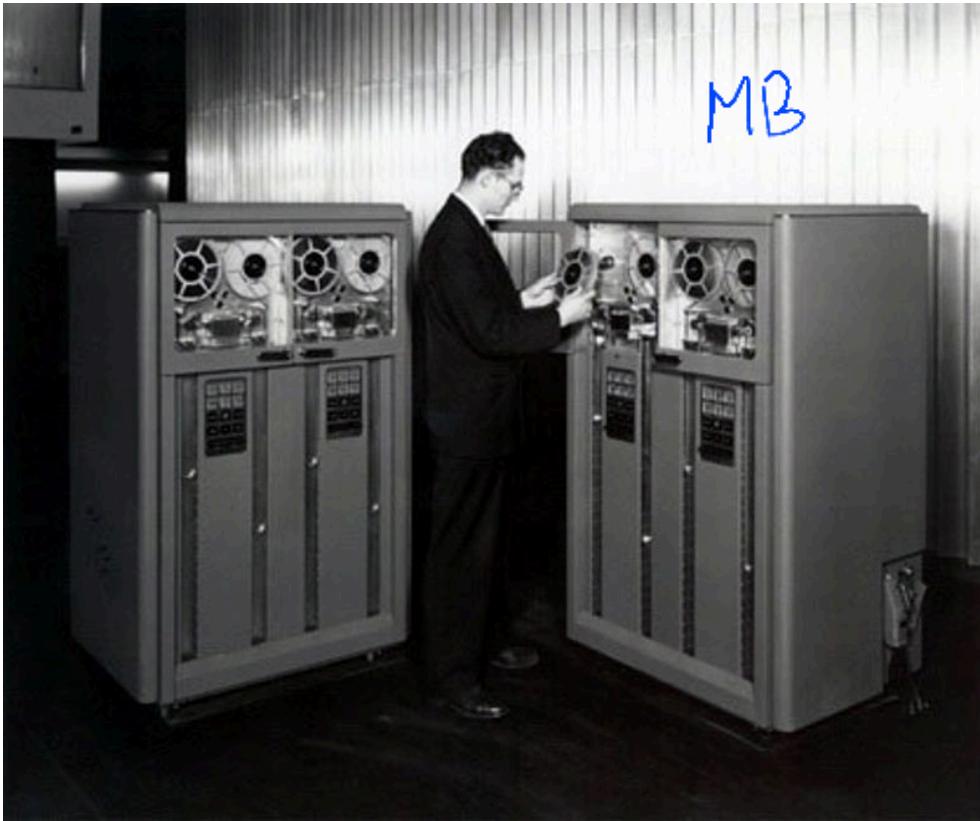
Redundancy

| name | userid | domain | department | faculty | language |
|-------------|--------|--------------|----------------------|-------------|------------|
| Tan Hee Wee | tanh | comp.sut.edu | computer science | computing | JavaScript |
| Tan Hee Wee | tanh | comp.sut.edu | computer science | computing | Python |
| Tan Hee Wee | tanh | comp.sut.edu | computer science | computing | C++ |
| Tan Hee Wee | tanhw | eng.sut.edu | computer engineering | engineering | C++ |
| Tan Hee Wee | tanhw | eng.sut.edu | computer engineering | engineering | Fortran |
| Bjorn Sale | bjorn | eng.sut.edu | computer engineering | engineering | C++ |
| Bjorn Sale | bjorn | eng.sut.edu | computer engineering | engineering | Java |
| Ami Mokhtar | ami | med.sut.edu | pharmacy | medicine | Rust |

Redundant Storage

The faculty of a department is repeated for every student of the department and every time the student is proficient in a programming language.

Redundant Storage



MB

Seagate
IronWolf Pro 24TB Enterprise NAS
Internal HDD Hard Drive – CMR 3.5 Inch
SATA 6Gb/s 7200 RPM 512MB Cache...

24 TB

4.6 ★★★★★ (24,044)
300+ bought in past month

\$479⁹⁹ ~~\$649.99~~
\$9.17 delivery

Ships to Singapore
Sold by QuickDealStore

Add to cart

IBM 726 (*rental USD 850 per month in 1952*)

Update

| name | userid | domain | department | faculty | language |
|-------------|--------|--------------|----------------------|---------------|------------|
| Tan Hee Wee | tanh | comp.sut.edu | computer science | computing X | JavaScript |
| Tan Hee Wee | tanh | comp.sut.edu | computer science | computing X | Python |
| Tan Hee Wee | tanh | comp.sut.edu | computer science | informatics ✓ | C++ |
| Tan Hee Wee | tanhw | eng.sut.edu | computer engineering | engineering | C++ |
| Tan Hee Wee | tanhw | eng.sut.edu | computer engineering | engineering | Fortran |
| Bjorn Sale | bjorn | eng.sut.edu | computer engineering | engineering | C++ |
| Bjorn Sale | bjorn | eng.sut.edu | computer engineering | engineering | Java |
| Ami Mokhtar | ami | med.sut.edu | pharmacy | medicine | Rust |

Update Anomaly

```
UPDATE proficiency SET faculty='informatics' WHERE language='C++' AND domain='comp.sut.edu' ←
```

We may **forget** to update all faculties for the same department.

order_items

Delete

| name | userid | domain | department | faculty | language |
|------------------------|----------------|------------------------|----------------------|-------------|------------|
| Tan Hee Wee | tanh | comp.sut.edu | computer science | computing | JavaScript |
| Tan Hee Wee | tanh | comp.sut.edu | computer science | computing | Python |
| Tan Hee Wee | tanh | comp.sut.edu | computer science | informatics | C++ |
| Tan Hee Wee | tanhw | eng.sut.edu | computer engineering | engineering | C++ |
| Tan Hee Wee | tanhw | eng.sut.edu | computer engineering | engineering | Fortran |
| Bjorn Sale | bjorn | eng.sut.edu | computer engineering | engineering | C++ |
| Bjorn Sale | bjorn | eng.sut.edu | computer engineering | engineering | Java |
| Ami Mokhtar | ami | med.sut.edu | pharmacy | medicine | Rust |

Deletion Anomaly

```
DELETE FROM proficiency WHERE userid='ami' AND domain='med.sut.edu'
```

We may **forget** we even have faculty of medicine.

Insert

| name | userid | domain | department | faculty | language |
|-------------|---------------|---------------|----------------------|----------------|-----------------|
| Tan Hee Wee | tanh | comp.sut.edu | computer science | computing | JavaScript |
| Tan Hee Wee | tanh | comp.sut.edu | computer science | computing | Python |
| Tan Hee Wee | tanh | comp.sut.edu | computer science | informatics | C++ |
| Tan Hee Wee | tanhw | eng.sut.edu | computer engineering | engineering | C++ |
| Tan Hee Wee | tanhw | eng.sut.edu | computer engineering | engineering | Fortran |
| Bjorn Sale | bjorn | eng.sut.edu | computer engineering | engineering | C++ |
| Bjorn Sale | bjorn | eng.sut.edu | computer engineering | engineering | Java |
| NULL // | NULL // | NULL // | pharmacy | medicine | NULL // |

Insertion Anomaly

We cannot record medicine unless there is a student because we cannot insert **NULL** due to some columns being **PRIMARY KEY***.

*Can you guess what should be the primary key of the table?



Source

| name | userid | domain | department | faculty | language |
|--------------------|--------------|--------------------|-----------------------------|------------------|------------|
| Tan Hee Wee | tanh | comp.sut.edu | computer science | <u>computing</u> | JavaScript |
| Tan Hee Wee | tanh | comp.sut.edu | computer science | <u>computing</u> | Python |
| Tan Hee Wee | tanh | comp.sut.edu | computer science | <u>computing</u> | C++ |
| <u>Tan Hee Wee</u> | <u>tanhw</u> | <u>eng.sut.edu</u> | <u>computer engineering</u> | engineering | C++ |
| <u>Tan Hee Wee</u> | <u>tanhw</u> | <u>eng.sut.edu</u> | <u>computer engineering</u> | engineering | Fortran |
| Bjorn Sale | bjorn | eng.sut.edu | computer engineering | engineering | C++ |
| Bjorn Sale | bjorn | eng.sut.edu | computer engineering | engineering | Java |
| <u>Ami Mokhtar</u> | <u>ami</u> | <u>med.sut.edu</u> | <u>pharmacy</u> | <u>medicine</u> | Rust |

Source of Anomalies

The source of these anomalies is because some columns are uniquely determined by other columns. This can be translated into an integrity constraint called functional dependencies.

identified ↗

Functional Dependency

Definition

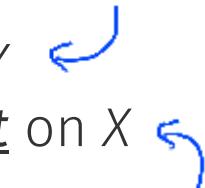
An instance r of a relation schema R satisfies the functional dependencies σ of the form $X \rightarrow Y$ with $X \subseteq R$ and $Y \subseteq R^*$, if and only if two tuples of r agree on their X -values, then they agree on their Y -values.

Terminology

$X \rightarrow Y$ reads:

| X | Y | $X \rightarrow Y$ |
|---|---|-------------------|
| T | T | T |
| T | F | F |
| F | T | F |
| F | F | T |

X functionally determines Y
 Y is functionally dependent on X



X determines Y
 $[X$ implies Y (casually)]

$$\begin{cases} r_1.A = r_2.A \text{ AND} \\ r_1.B = r_2.B \end{cases}$$

IS NOT DISTINCT FROM

$X \rightarrow Y$

*We abuse the notation a little, by $X \subseteq R$ we meant the set of columns from schema R . So we can write it as either $R(A, B, C, D)$ or $R = \{A, B, C, D\}$.

Set Functional Dependency

Definition

An instance r of a relation schema R satisfies a **set of functional dependencies** Σ if and only if it satisfies all the functional dependencies σ in the set Σ .
 $r \rightarrow \Sigma$ one table

Valid Instance

An instance r of a relation schema R is a valid instance of R with Σ iff it satisfies Σ .

Violation

In other words, there are no functional dependencies $\sigma \in \Sigma$ that are violated (*i.e., not satisfied*).

$X \rightarrow Y$ is violated if there are two tuples of r agree on their X-values, **BUT do not agree on their Y-values**.

$$\Sigma = \{ \underbrace{\{A\} \rightarrow \{B\}}, \underbrace{\{B\} \rightarrow \{C\}}, \dots \}$$

Holds

Definition

A relation R with a **set of functional dependencies Σ** --denoted as R with Σ -- refers to the set of valid instances of R with respect to the functional dependencies in Σ .

schema

Holds

We say that a set of functional dependencies Σ holds on a relation R

Note

Note that in this case, we only consider the valid instances of R with Σ . If Σ holds on R , then there cannot be violation and consequently, every instance r must be a valid instance of R .

Example

Let $R = \{A, B, C, D\}$ and $\sigma = \{A, \underline{B}\} \rightarrow \{\underline{D}\}$

Valid

| A | B | C | D |
|---|---|---|---|
| 1 | 2 | a | 4 |
| 1 | 2 | b | 4 |
| 1 | 3 | c | 4 |

Violate

| A | B | C | D |
|---|---|---|---|
| 1 | 2 | a | 4 |
| 1 | 2 | b | 3 |
| 1 | 3 | c | 4 |

Example

Let $R = \{A, B, C, D\}$ and $\sigma = \{A, B\} \rightarrow \{D\}$

Valid

| A | B | C | D |
|---|---|---|---|
| | | | |

PK UP NN FK CHK

Violate

| A | B | C | D |
|---|---|---|---|
| 1 | 2 | a | 4 |
| 1 | 2 | b | 3 |

Note

Empty instance is always a (vacuously) valid instance.

Note

Both examples are the smallest (*valid* and *invalid*) instances.

Trivial

Definition

A functional dependency $\sigma : \underline{X} \rightarrow Y$ is trivial if and only if $\underline{Y} \subseteq X$. In other words, a set X always uniquely determine its subset X .

Let $R = \{A, B, C\}$

$$\begin{array}{l} \{A\} \rightarrow \{A\} \\ \{A, B\} \rightarrow \{A\} \\ \{A, B\} \rightarrow \{\} \end{array}$$

is trivial ✓
 is trivial ✓
is trivial (also denoted as $\{A, B\} \rightarrow \emptyset$)

→ always satisfied

$$\begin{aligned} X &= \{A, B\} \\ Y &= \{A\} \end{aligned}$$

if they agree on $A \cup B$, then they agree on A

Non-Trivial

Definition

A functional dependency $\sigma: X \rightarrow Y$ is non-trivial if and only if $Y \not\subseteq X$. Literally, the negation of the definition of trivial.

Let $R = \{A, B, C\}$

- $\{A\} \rightarrow \{B\}$ is non-trivial
- $\{A,C\} \rightarrow \{B,C\}$ is non-trivial
- $\{\} \rightarrow \{A,B\}$ is non-trivial

$\{B\} \not\subseteq \{A\}$ $\{B,C\} \not\subseteq \{A,C\}$ $\{A,B\} \not\subseteq \emptyset$

overlap

Constant

Note

$\emptyset \rightarrow \{A,B\}$ means that the pair of attributes $\{A,B\}$ are **constants**.

Completely

Definition

A functional dependency $\sigma : X \rightarrow Y$ is completely non-trivial if and only if $Y \neq \emptyset$ and $Y \cap X = \emptyset$. In other words, they have no common attributes.

Let $R = \{A, B, C\}$

$$\begin{array}{ll} \{A\} \rightarrow \{B\} & \\ \{A,C\} \rightarrow \{B,C\} & \\ \underline{\{A,B\} \rightarrow \{\}} & \end{array}$$

- is completely non-trivial
- is **not** completely non-trivial but still non-trivial
- is **not** completely non-trivial as it is trivial

$$\begin{array}{ccc} \{A,C\} \rightarrow \{B,C\} & & \\ \nearrow & & \searrow \\ \{A,C\} \rightarrow \{C\} & & \{A,C\} \rightarrow \{B\} \end{array}$$

Theorem #1 ➔

A non-trivial (*but not completely non-trivial*) functional dependency can be split into a trivial functional dependency and a completely non-trivial functional dependency.

Superkey

$$R = \{A, B, C, D\}$$

$$R \subseteq R$$

Definition

Let R be a relation. Let $S \subseteq R$ be a **set of attributes** of R . S is a **superkey** of R if and only if $S \rightarrow R$. A superkey is a **superset of a key** (*a key or candidate key is defined later*).

In Other Words

A **superkey** is a set of attributes of a relation whose knowledge determines the value of the entire tuple.

Theorem #2 →

A relation R have at least 1 superkey.

$$\begin{aligned} S &= \{A, B, C\} \\ \underline{\{A, B\}} \rightarrow \underline{\{A, B, C, D\}} \\ \underline{\{A, B\}} \rightarrow \underline{\{A, B, C, D\}} \end{aligned}$$

Candidate Key

Definition

Let R be a relation. Let $S \subseteq R$ be a set of attributes of R . S is a candidate key of R if and only if $S \rightarrow R$ and for all $T \subset S$, T is NOT a superkey of R .

In Other Words

A **candidate key** is a minimal* superkey. (Note: minimal is not the same as minimum.)

↑ vs minimum

Primary Key

The **primary key** is the candidate key that the designer prefers.



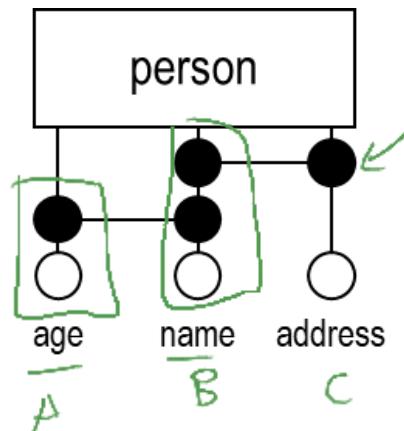
*By **minimal** superkey, we meant that if any attribute of a candidate key is removed, the set is no longer a superkey.

$$\{D, E\} \supseteq \{D\}$$

Candidate Key

Definition

Let R be a relation. Let $S \subseteq R$ be a set of attributes of R . S is a candidate key of R if and only if $S \rightarrow R$ and for all $T \subset S$, T is NOT a superkey of R .



Candidate Keys

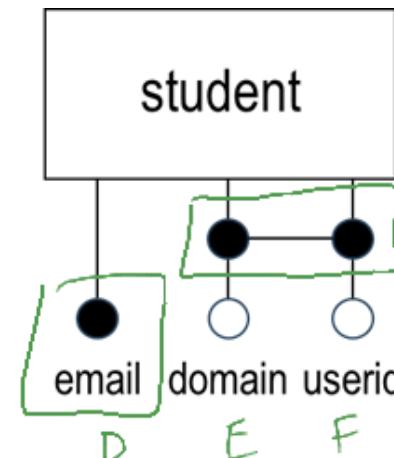
$\{ \text{age, name} \},$
 $\{ \text{name, address} \}$

$$\{A\} \not\rightarrow \{A,B,C\}$$

$$\{B\} \not\rightarrow \{A,B,C\}$$

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

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



Candidate Keys

$\{ \text{email} \},$
 $\{ \text{domain, userid} \}$

$$\{D, \cancel{E}\} \rightarrow \{D,E,F\}$$

$$\{D\} \rightarrow \{D,E,F\}$$

$$\{E,F\} \rightarrow \{D,E,F\}$$

$$\{F\} \not\rightarrow \{D,E,F\}$$

$$\{E\} \not\rightarrow \{D,E,F\}$$

Prime Attribute

Definition

Let Σ be a **set of functional dependencies** on a relation schema R . A prime attribute is an attribute that appears in some candidate key of R with Σ .

Note

Only need to appear in at least one candidate key. In other words, if there are multiple candidate keys K_1, K_2, \dots, K_n , the prime attribute is $[K_1 \cup K_2 \cup \dots \cup K_n]$.

Otherwise

Attributes that are NOT prime attributes are called non-prime attributes.

Example

Let $R = \{A, B, C, D\}$ and $\Sigma = \{\underline{\{A, B\}} \rightarrow \underline{\{C, D\}}, \underline{\{C\}} \rightarrow \underline{\{A, B\}}\}$

Superkeys

- ✓ $\{A, B, C, D\}$
- ✓ $\{A, B\}$
- ✓ $\{C, D\}$

etc...

Candidate Keys

- ✓ $\{A, B\}$
- ✓ $\{C\}$

Prime Attributes

- | | |
|---|------------------------|
| A | $\{A, B\} \cup \{C\}$ |
| B | |
| C | (i.e., $\{A, B, C\}$) |

Question

How do we know which are superkeys and which are not?

Instance

| (A) name | (B) userid | (C) domain | (D) department | (E) faculty | (F) language |
|-------------|------------|--------------|----------------------|-------------|--------------|
| Tan Hee Wee | tanh | comp.sut.edu | computer science | computing | JavaScript |
| Tan Hee Wee | tanh | comp.sut.edu | computer science | computing | Python |
| Tan Hee Wee | tanh | comp.sut.edu | computer science | computing | C++ |
| Tan Hee Wee | tanhw | eng.sut.edu | computer engineering | engineering | C++ |
| Tan Hee Wee | tanhw | eng.sut.edu | computer engineering | engineering | Fortran |
| Bjorn Sale | bjorn | eng.sut.edu | computer engineering | engineering | C++ |
| Bjorn Sale | bjorn | eng.sut.edu | computer engineering | engineering | Java |
| Ami Mokhtar | ami | med.sut.edu | pharmacy | medicine | Rust |

Hold

$$\{D\} \rightarrow \{E\} \quad \checkmark$$

$$\{B,C\} \rightarrow \{E\} \quad \checkmark$$

$$\{B,C\} \rightarrow \{D\} \quad \checkmark$$

$$\{A,B,C,D,F\} \rightarrow \{E\} \quad \checkmark$$

$$\{F\} \rightarrow \{F\} \quad \checkmark$$

$$\{B,C\} \rightarrow \{B\} \quad \checkmark$$

Do Not Hold

$$\{B\} \rightarrow \{A\} \quad \text{(how do we know?)}$$

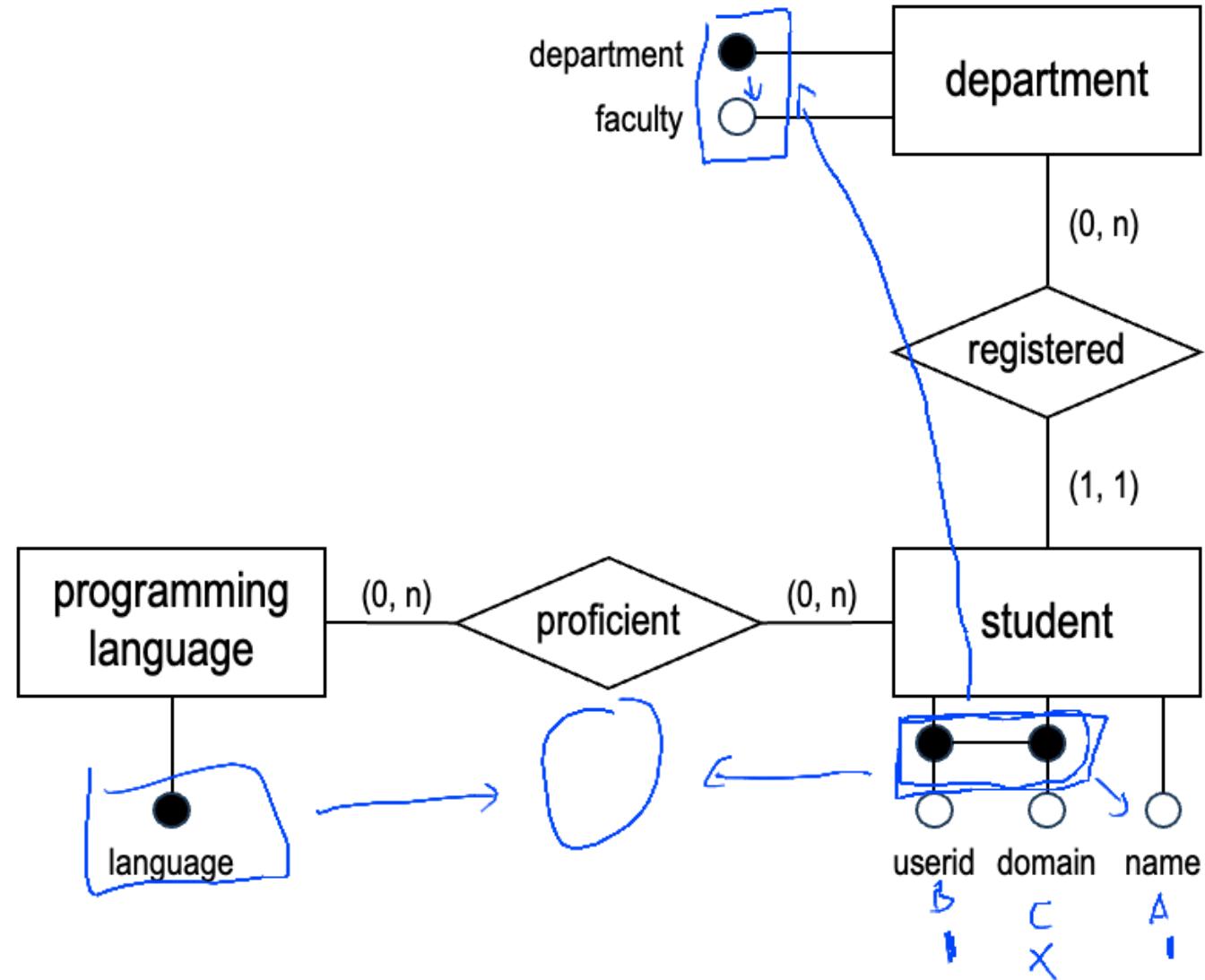
$$\{F\} \rightarrow \{E\} \quad \times$$

$$\{C,E\} \rightarrow \{B\} \quad \times$$

The Case

Sentosa University of Technology (SUT) records the **programming language skills** of the students of its **different faculties** and **departments**.

$\{B\} \rightarrow \{A\}$



Closure of Σ

$$\textcircled{N}, + \\ (\{\}, +)$$

$$1+2=3 \\ x+y=2$$

Definition

Let Σ be a set of functional dependencies of a relation schema R . The **closure of Σ** -- denoted by Σ^+ -- is the **set of all functional dependencies logically entailed** by the functional dependencies in Σ .

Example

Let $R = \{A, B, C, D\}$ and $\Sigma = \{ \{A\} \rightarrow \{B\}, \{C\} \rightarrow \{A\} \}$

$$\Sigma^+ = \{ \{A\} \rightarrow \{B\}, \{C\} \rightarrow \{A\}, \{A\} \rightarrow \{A\}, \{D\} \rightarrow \{D\}, \{A, B\} \rightarrow \{A\}, \{A, C\} \rightarrow \{B, C\}, \{A, D\} \rightarrow \{B\}, \{C\} \rightarrow \{B\}, \dots \}$$

(how do we know this without entity-relationship diagram?)

Note

This is the **closure** of the **set of functional dependencies**. We have a simpler closure later.

Proof?

Example

Let $R = \{A, B, C, D\}$ and $\Sigma = \{\{A\} \rightarrow \{B\}, \{C\} \rightarrow \{A\}\}$



$\{C\} \rightarrow \{B\} \in \Sigma^+?$

1. Assume $\{C\} \rightarrow \{B\}$ does not hold.

Then there are two rows (A_1, B_1, C, D_1) and (A_2, B_2, C, D_2) such that $B_1 \neq B_2$.

2. Since $\underline{\{C\}} \rightarrow \{A\}$ holds, then $A_1 = A_2$ because $C = C$.

We rewrite both as \underline{A} to get $(\underline{A}, B_1, C, D_1)$ and $(\underline{A}, B_2, C, D_2)$.

3. Since $\underline{\{A\}} \rightarrow \{B\}$ holds, then $B_1 = B_2$ because $A = A$.

We rewrite both as \underline{B} to get $(\underline{A}, \underline{B}, C, D_1)$ and $(\underline{A}, \underline{B}, C, D_2)$.

4. But this contradicts (1) that states $B_1 \neq B_2$.

5. So the assumption (1) must be wrong and we conclude $\{C\} \rightarrow \{B\}$ holds.

Closure of S

Definition

Let Σ be a set of functional dependencies of a relation schema R . The closure of $S \subseteq R$ (i.e., a set of attributes) --denoted by S^+ -- is the **set of all attributes functionally dependent** on S .

Example

Let $R = \{A, B, C, D\}$ and $\Sigma = \{\{A\} \rightarrow \{B\}, \{C\} \rightarrow \{A\}\}$

$$\{A\}^+ = \{A, B\}$$

$$\{B\}^+ = \{B\}$$

$$\{C\}^+ = \{A, B, C\}$$

$$\{C, D\}^+ = \{A, B, C, D\}$$

$$\begin{array}{c} \{C\} \rightarrow \{A, B, C\} \\ \Downarrow \\ \{C\} \rightarrow \{A\} \\ \{C\} \rightarrow \{B\} \\ \{C\} \rightarrow \{A, B\} \end{array}$$

Note

Because S^+ is **functionally dependent** on S , we have

$$S \rightarrow S^+$$

In fact, for all $Y \subseteq S^+$, we have

$$S \rightarrow Y$$

Rules

Armstrong Axioms

Let R be a set of attributes. The following **inference rules** are the **Armstrong Axiom**.

• **Reflexivity**

$$\forall X \subseteq R \forall Y \subseteq R ((Y \subseteq X) \Rightarrow (X \rightarrow Y)) \quad \left\{ \Rightarrow \text{trivial} \right.$$

Trivial functional dependencies are always valid

• **Augmentation**

$$\forall X \subseteq R \forall Y \subseteq R \forall Z \subseteq R ((X \rightarrow Y) \Rightarrow ((X \cup Z) \rightarrow (Y \cup Z))) \quad \left\{ \begin{array}{l} \\ \\ \end{array} \right.$$

*You can add the same attribute to both left hand and right hand side **at the same time***

• **Transitivity**

$$\forall X \subseteq R \forall Y \subseteq R \forall Z \subseteq R ((X \rightarrow Y \wedge Y \rightarrow Z) \Rightarrow (X \rightarrow Z)) \quad \left\{ \begin{array}{l} \\ \\ \\ \\ \end{array} \right.$$

You can short-circuit the arrows

***Out of scope.** We put this for completeness of explanation only.

Example

Question

Let $R = \{A, B, C, D\}$ and $\Sigma = \{\{A\} \rightarrow \{B\}, \{C\} \rightarrow \{A\}\}$. Can we prove that $\underline{\{C,D\} \rightarrow \{B,A\}} \in \Sigma^+$?

Proof

1. We know that $\{A\} \rightarrow \{B\}$.
 2. We know that $\{C\} \rightarrow \{A\}$.
 3. Therefore $\{A\} \rightarrow \{A,B\}$.
 4. Therefore $\{C\} \rightarrow \{A,B\}$.
 5. Therefore $\{C,D\} \rightarrow \{A,B,D\}$.
 6. Therefore $\{A,B,D\} \rightarrow \{A,B\}$.
 7. Therefore $\{C,D\} \rightarrow \{A,B\}$.
- 
- (given in Σ)
(given in Σ)
(Augmentation of (1) with {A})
(Transitivity of (2) and (3))
(Augmentation of (4) with {D})
(Reflexivity since $\{A,B\} \subseteq \{A,B,D\}$)
(Transitivity of (5) and (6))

□

*Out of scope. We put this for completeness of explanation only.

Theorems

Theorem #3: Soundness ➔

The **Reflexivity** inference rule is sound (*i.e., correct, valid*).

The **Augmentation** inference rule is sound (*i.e., correct, valid*).

The **Transitivity** inference rule is sound (*i.e., correct, valid*).

All functional dependencies obtained using Armstrong axioms are **guaranteed to hold**.

Theorem #4: Completeness ➔

The **Armstrong axiom** inference rule is *complete*.

All functional dependencies that holds **can be obtained** using Armstrong axioms.

Note

What we are interested in is the **consequence** of the theorem. We can formalize this is as algorithm.



Attribute Closure

Algorithm #1: Attribute Closure**input** S, Σ **output** S^+ **begin**

```
//  
|  $\Omega := \Sigma;$  //  $\Omega$  stands for "unused"  
|  $\Gamma := S;$  //  $\Gamma$  stands for "closure"  
while ( $X \rightarrow Y \in \Omega$ ) and ( $X \subseteq \Gamma$ ) do  
|  $\Omega := \Omega - \{ X \rightarrow Y \};$   
|  $\Gamma := \Gamma \cup Y;$   
return  $\Gamma$ 
```

Attribute Closure

Algorithm #1: Attribute Closure

input S, Σ **output** S^+ **begin** $\Omega := \Sigma$; // Ω stands for "unused" $\Gamma := \{S\}$; // Γ stands for "closure" **while** $(X \rightarrow Y \in \Omega)$ **and** $(X \subseteq \Gamma)$ **do** $\Omega := \Omega - \{X \rightarrow Y\}$; $\Gamma := \Gamma \cup Y$;**return** Γ

$$\Omega = \{ \dots \}$$

$$\Gamma = \boxed{\{A, B, C\}}$$

Question

Let $R = \{A, B, C, D\}$
 $\Sigma = \{ \{A\} \rightarrow \{B\}, \{C\} \rightarrow \{A\} \}$.

Compute $\{C\}^+$ using **Algorithm #1**.

Attribute Closure

Algorithm #1: Attribute Closure**input** S, Σ **output** S^+ **begin** $\Omega := \Sigma; \quad // \Omega \text{ stands for "unused"}$ $\Gamma := S; \quad // \Gamma \text{ stands for "closure"}$ **while** ($X \rightarrow Y \in \Omega$) **and** ($X \subseteq \Gamma$) **do** $\Omega := \Omega - \{ X \rightarrow Y \};$ $\Gamma := \Gamma \cup Y;$ **return** Γ

$$\begin{aligned}1. \quad & \Omega = \{ \{A\} \rightarrow \{B\}, \{C\} \rightarrow \{A\} \} \\& \Gamma = \{C\}\end{aligned}$$

QuestionLet $R = \{A, B, C, D\}$ $\Sigma = \{ \{A\} \rightarrow \{B\}, \{C\} \rightarrow \{A\} \}.$ Compute $\{C\}^+$ using **Algorithm #1**.

Attribute Closure

Algorithm #1: Attribute Closure**input** S, Σ **output** S^+ **begin** $\Omega := \Sigma; \quad // \Omega \text{ stands for "unused"}$ $\Gamma := S; \quad // \Gamma \text{ stands for "closure"}$ **while** ($X \rightarrow Y \in \Omega$) **and** ($X \subseteq \Gamma$) **do** $\Omega := \Omega - \{ X \rightarrow Y \};$ $\Gamma := \Gamma \cup Y;$ **return** Γ

1. $\Omega = \{ \{A\} \rightarrow \{B\}, \{C\} \rightarrow \{A\} \}$

$\Gamma = \{C\}$

2. use $\{C\} \rightarrow \{A\}$ ($\{C\} \subseteq \Gamma$)

QuestionLet $R = \{A, B, C, D\}$ $\Sigma = \{ \{A\} \rightarrow \{B\}, \{C\} \rightarrow \{A\} \}.$ Compute $\{C\}^+$ using **Algorithm #1**.

Attribute Closure

Algorithm #1: Attribute Closure**input** S, Σ **output** S^+ **begin** $\Omega := \Sigma; \quad // \Omega \text{ stands for "unused"}$ $\Gamma := S; \quad // \Gamma \text{ stands for "closure"}$ **while** ($X \rightarrow Y \in \Omega$) **and** ($X \subseteq \Gamma$) **do** $\Omega := \Omega - \{ X \rightarrow Y \};$ $\Gamma := \Gamma \cup Y;$ **return** Γ

$$1. \Omega = \{ \{A\} \rightarrow \{B\}, \{C\} \rightarrow \{A\} \}$$

$$\Gamma = \{C\}$$

$$2. \text{use } \{C\} \rightarrow \{A\} \quad (\{C\} \subseteq \Gamma)$$

$$\Omega = \{ \{A\} \rightarrow \{B\} \}$$

Question

Let $R = \{A, B, C, D\}$

$\Sigma = \{ \{A\} \rightarrow \{B\}, \{C\} \rightarrow \{A\} \}$.

Compute $\{C\}^+$ using **Algorithm #1**.

Attribute Closure

Algorithm #1: Attribute Closure**input** S, Σ **output** S^+ **begin** $\Omega := \Sigma; \quad // \Omega \text{ stands for "unused"}$ $\Gamma := S; \quad // \Gamma \text{ stands for "closure"}$ **while** ($X \rightarrow Y \in \Omega$) **and** ($X \subseteq \Gamma$) **do** $\Omega := \Omega - \{X \rightarrow Y\};$ $\Gamma := \Gamma \cup Y;$ **return** Γ

1. $\Omega = \{ \{A\} \rightarrow \{B\}, \{C\} \rightarrow \{A\} \}$

$\Gamma = \{C\}$

2. use $\{C\} \rightarrow \{A\}$ ($\{C\} \subseteq \Gamma$)

$\Omega = \{ \{A\} \rightarrow \{B\} \}$

$\Gamma = \{C\} \cup \{A\} = \{A, C\}$

QuestionLet $R = \{A, B, C, D\}$ $\Sigma = \{ \{A\} \rightarrow \{B\}, \{C\} \rightarrow \{A\} \}.$ Compute $\{C\}^+$ using **Algorithm #1**.

Attribute Closure

Algorithm #1: Attribute Closure**input** S, Σ **output** S^+ **begin** $\Omega := \Sigma;$ // Ω stands for "unused" $\Gamma := S;$ // Γ stands for "closure"**while** ($X \rightarrow Y \in \Omega$) **and** ($X \subseteq \Gamma$) **do** $\Omega := \Omega - \{ X \rightarrow Y \};$ $\Gamma := \Gamma \cup Y;$ **return** Γ

1. $\Omega = \{ \{A\} \rightarrow \{B\}, \{C\} \rightarrow \{A\} \}$

$\Gamma = \{C\}$

2. use $\{C\} \rightarrow \{A\}$ ($\{C\} \subseteq \Gamma$)

$\Omega = \{ \{A\} \rightarrow \{B\} \}$

$\Gamma = \{C\} \cup \{A\} = \{A, C\}$

3. use $\{A\} \rightarrow \{B\}$ ($\{A\} \subseteq \Gamma$)

$\Omega = \emptyset$

$\Gamma = \{A, C\} \cup \{B\} = \{A, B, C\}$

QuestionLet $R = \{A, B, C, D\}$ $\Sigma = \{ \{A\} \rightarrow \{B\}, \{C\} \rightarrow \{A\} \}.$ Compute $\{C\}^+$ using **Algorithm #1**.

Attribute Closure

Algorithm #1: Attribute Closure**input** S, Σ **output** S^+ **begin** $\Omega := \Sigma; \quad // \Omega \text{ stands for "unused"}$ $\Gamma := S; \quad // \Gamma \text{ stands for "closure"}$ **while** ($X \rightarrow Y \in \Omega$) **and** ($X \subseteq \Gamma$) **do** $\Omega := \Omega - \{X \rightarrow Y\};$ $\Gamma := \Gamma \cup Y;$ **return** Γ

1. $\Omega = \{ \{A\} \rightarrow \{B\}, \{C\} \rightarrow \{A\} \}$

$\Gamma = \{C\}$

2. use $\{C\} \rightarrow \{A\}$ ($\{C\} \subseteq \Gamma$)

$\Omega = \{ \{A\} \rightarrow \{B\} \}$

$\Gamma = \{C\} \cup \{A\} = \{A, C\}$

3. use $\{A\} \rightarrow \{B\}$ ($\{A\} \subseteq \Gamma$)

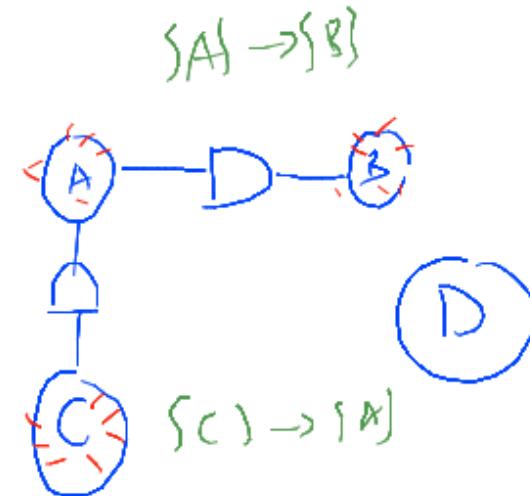
$\Omega = \emptyset$

$\Gamma = \{A, C\} \cup \{B\} = \{A, B, C\}$

4. return $\Gamma = \{A, B, C\}$

QuestionLet $R = \{A, B, C, D\}$ $\Sigma = \{ \{A\} \rightarrow \{B\}, \{C\} \rightarrow \{A\} \}.$ Compute $\{C\}^+$ using **Algorithm #1**.

Attribute Closure

Algorithm #1: Attribute Closure**input** S, Σ **output** S^+ **begin** $\Omega := \Sigma$; // Ω stands for "unused" $\Gamma := S$; // Γ stands for "closure"**while** ($X \rightarrow Y \in \Omega$) **and** ($X \subseteq \Gamma$) **do** $\Omega := \Omega - \{X \rightarrow Y\}$; $\Gamma := \Gamma \cup Y$;**return** Γ 

$$\{C\}^+ = \{A, B, C\}$$

Algo 1 :

- Superkey
- key
- Prime Attribute
- closure of Σ

Question

Let $R = \{A, B, C, D\}$ $\Sigma = \{ \{A\} \rightarrow \{B\}, \{C\} \rightarrow \{A\} \}$.Compute $\{C\}^+$ using **Algorithm #1**.



Equivalent Σ

Definition

Two sets of functional dependencies Σ_1 and Σ_2 are equivalent if and only if they have the same closure. More formally,

$$\Sigma_1 \equiv \Sigma_2 \Leftrightarrow \underline{\Sigma}_1^+ = \underline{\Sigma}_2^+$$

Cover

Σ_1 is a cover of Σ_2 (*and Σ_2 is a cover of Σ_1*) if and only if $\Sigma_1 \equiv \Sigma_2$.



Example

$$\bar{\Sigma}_1 \cap \bar{\Sigma}_2 = \emptyset$$

Question

Let $R = \{A, B, C\}$. Are the following set of functional dependencies equivalent?

$$\Sigma_1 = \{ \{A\} \rightarrow \{B\}, \{B\} \rightarrow \{C\}, \{C\} \rightarrow \{A\} \}$$

$$\Sigma_2 = \{ \{A\} \rightarrow \{C\}, \{C\} \rightarrow \{B\}, \{B\} \rightarrow \{A\} \}$$

Example

Question

Let $R = \{A, B, C\}$. Are the following set of functional dependencies equivalent?

$$\Sigma_1 = \{ \{A\} \rightarrow \{B\}, \{B\} \rightarrow \{C\}, \{C\} \rightarrow \{A\} \}$$

$$\Sigma_2 = \{ \{A\} \rightarrow \{C\}, \{C\} \rightarrow \{B\}, \{B\} \rightarrow \{A\} \}$$

- $\Sigma_1^+ = \{ \underbrace{\{A\} \rightarrow \{B\}}, \underbrace{\{B\} \rightarrow \{C\}}, \underbrace{\{C\} \rightarrow \{A\}}, \{A\} \rightarrow \{B,C\}, \{B\} \rightarrow \{A,C\}, \{C\} \rightarrow \{A,B\}, \{A\} \rightarrow \{C\}, \underbrace{\{C\} \rightarrow \{B\}}, \underbrace{\{B\} \rightarrow \{A\}} , \dots \}$
- $\Sigma_2^+ = \{ \underbrace{\{A\} \rightarrow \{C\}}, \underbrace{\{C\} \rightarrow \{B\}}, \underbrace{\{B\} \rightarrow \{A\}}, \{A\} \rightarrow \{B,C\}, \{B\} \rightarrow \{A,C\}, \{C\} \rightarrow \{A,B\}, \{A\} \rightarrow \{B\}, \underbrace{\{B\} \rightarrow \{C\}}, \underbrace{\{C\} \rightarrow \{A\}} , \dots \}$

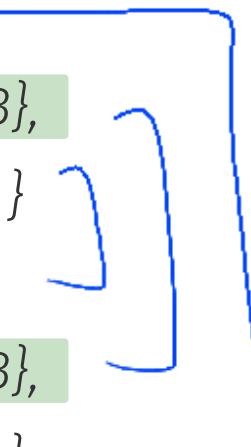
Example

Question

Let $R = \{A, B, C\}$. Are the following set of functional dependencies equivalent?

$$\Sigma_1 = \{ \{A\} \rightarrow \{B\}, \{B\} \rightarrow \{C\}, \{C\} \rightarrow \{A\} \}$$

$$\Sigma_2 = \{ \{A\} \rightarrow \{C\}, \{C\} \rightarrow \{B\}, \{B\} \rightarrow \{A\} \}$$

- $\Sigma_1^+ = \{ \{A\} \rightarrow \{B\}, \{B\} \rightarrow \{C\}, \{C\} \rightarrow \{A\},$
 $\{A\} \rightarrow \{B,C\}, \{B\} \rightarrow \{A,C\}, \{C\} \rightarrow \{A,B\},$
 $\{A\} \rightarrow \{C\}, \{C\} \rightarrow \{B\}, \{B\} \rightarrow \{A\} , \dots \}$ 
- $\Sigma_2^+ = \{ \{A\} \rightarrow \{C\}, \{C\} \rightarrow \{B\}, \{B\} \rightarrow \{A\},$
 $\{A\} \rightarrow \{B,C\}, \{B\} \rightarrow \{A,C\}, \{C\} \rightarrow \{A,B\},$
 $\{A\} \rightarrow \{B\}, \{B\} \rightarrow \{C\}, \{C\} \rightarrow \{A\} , \dots \}$ 

Theorem

Theorem #5 ➔

Two sets of functional dependencies Σ_1 and Σ_2 are equivalent if and only if all of the following condition is satisfied.

- All functional dependency σ in Σ_1 is a logical consequence of the functional dependencies in Σ_2 .
- All functional dependency σ in Σ_2 is a logical consequence of the functional dependencies in Σ_1 .

Minimal

Definition

A set Σ of functional dependencies is minimal if and only if

- The **right hand side** of every functional dependency Σ is minimal.
i.e., every functional dependency is of the form $X \rightarrow \{A\}$.
- The **left hand side** of every functional dependency Σ is minimal.
*i.e., for every functional dependency in Σ of the form $X \rightarrow \{A\}$
there is no functional dependency $Y \rightarrow \{A\}$ in Σ^+ such that $Y \subset X$.*
- The **set itself** is minimal.
i.e., none of the functional dependency in Σ can be derived from other functional dependencies in Σ .

Σ is a minimal cover of Σ' if and only if Σ is both minimal and a cover for Σ' .

Minimal

In Other Words

A set Σ of functional dependencies is *minimal* if and only if

- The **right hand side** of every functional dependency Σ only has one attribute.
- Cannot remove any attribute from the **left hand side**.
- Cannot remove any functional dependencies from Σ .

Canonical

Definition



A set Σ is a canonical cover of Σ' if there is a Σ_1 that is a [minimal cover] of Σ' and we produce Σ by **regrouping** all the functional dependencies with the same left hand side in Σ_1 .

Note

In other words, we find **minimal cover** and then we merge the right hand side if the left hand side are equal.

- $\Sigma = \{ \{A\} \rightarrow \{B\}, \{A\} \rightarrow \{C\} \}$
- $\Sigma = \{ \{A\} \rightarrow \underline{\{B,C\}} \}$

is **minimal** but **not** canonical.
is canonical (*but is it minimal?*).

Theorem

Theorem #6 ➔

Every set of functional dependencies has at least one minimal cover (also called as *minimal basis*).
The minimal cover **may not be unique**.

Example

$$\Sigma_1 = \{ \{A\} \rightarrow \{B\}, \{B\} \rightarrow \{C\}, \{C\} \rightarrow \{A\} \}$$

$$\Sigma_2 = \{ \{A\} \rightarrow \{C\}, \{C\} \rightarrow \{B\}, \{B\} \rightarrow \{A\} \}$$

$\Sigma_1 \equiv \Sigma_2$ and they are minimal.



Algorithm

Algorithm #2: Minimal Cover**input** Σ **output** Σ_C

1. Simplify (*minimize*) the **right hand side** of every functional dependency in Σ
produce Σ_A
2. Simplify (*minimize*) the **left hand side** of every functional dependency in Σ_A
produce Σ_B
3. Simplify (*minimize*) the **set** Σ_B to produce Σ_C
4. Return Σ_C



Algorithm

Algorithm #3: Canonical Cover**input** Σ **output** Σ_D

1. Simplify (*minimize*) the **right hand side** of every functional dependency in Σ
produce Σ_A
2. Simplify (*minimize*) the **left hand side** of every functional dependency in Σ_A
produce Σ_B
3. Simplify (*minimize*) the **set** Σ_B to produce Σ_C
4. **[Regroup]** all functional dependencies *with the same left hand side* in Σ_C
produce Σ_D
5. Return Σ_D



Simplification

Step #2: Simplifying Left Hand Side ➔

Let $X \rightarrow \{A\}$ be a functional dependency in Σ' . Attribute $B \in X$ can be removed from X if

$(X - \{B\}) \rightarrow \{A\}$ is logically entailed by Σ'

Then we can replace $X \rightarrow \{A\}$ by $(X - \{B\}) \rightarrow \{A\}$ in Σ'

$$\{C\}_{\Sigma'}^+ = \{C, \cancel{A}, \cancel{D}, \cancel{B}\}$$

$$\{C \rightarrow \{A\}\}$$

$$\{ \cancel{B}, \cancel{C}, \cancel{D} \rightarrow \cancel{\{A\}} \}$$

Step #3: Simplifying the Set ➔

Let $X \rightarrow Y$ be a functional dependency in Σ' . It can be removed from Σ' if

$X \rightarrow Y$ is logically entailed by $(\Sigma' - \{X \rightarrow Y\})$

Then we can replace Σ' by $(\Sigma' - \{X \rightarrow Y\})$.

$$X_{\Sigma'}^+ = \{B, C, \cancel{A}, \cancel{D}\}$$

Simplification

Step #2: Simplifying Left Hand Side

Consider $X \rightarrow \{A\}$ in Σ' . Consider an attribute $B \in X$ and construct $(X - \{B\}) \rightarrow \{A\}$.

Compute $(X - \{B\})^+ \text{ with respect to } \Sigma'$ Algo |

If $A \in (X - \{B\})^+$, then we can replace $X \rightarrow \{A\}$ by $(X - \{B\}) \rightarrow \{A\}$ in Σ'

Step #3: Simplifying the Set

Consider $X \rightarrow Y$ in Σ' . Consider $\Sigma'' = \Sigma' - \{X \rightarrow Y\}$.

Compute $X^+ \text{ with respect to } \Sigma''$ Algo |

Then we can replace Σ' by $\Sigma'' = (\Sigma' - \{X \rightarrow Y\})$.

Question

$$2^5 - 1 = 31$$

$$R = \{A, B, C, D, E\}$$

$$\Sigma = \{\underline{\{A, B\}} \rightarrow \underline{\{C, D, E\}}, \underline{\{A, C\}} \rightarrow \underline{\{B, D, E\}}, \underline{\{B\}} \rightarrow \underline{\{C\}}, \underline{\{C\}} \rightarrow \underline{\{B\}}, \underline{\{C\}} \rightarrow \underline{\{D\}}, \underline{\{B\}} \rightarrow \underline{\{E\}}, \underline{\{C\}} \rightarrow \underline{\{E\}}\}$$

(A) B, C, D, E

1. Find all the candidate keys.
2. Find a minimal cover.
3. Find a canonical cover.



Tips 1 : start from small to big

Tips 2 : attr not in any RHS is in all keys.

Compute Closures of Singletons

$$\begin{cases} \{A\}^+ = \{A\} \\ \{B\}^+ = \{B, \cancel{C}, \cancel{D}, \cancel{E}\} \\ \{C\}^+ = \{B, \cancel{C}, \cancel{D}, E\} \\ \{D\}^+ = \{D\} \\ \{E\}^+ = \{E\} \end{cases}$$

Superset of A

Tips #1

Always start from the smallest cardinality
(i.e., size).

Tips #2

If any attribute does not appear on the right hand side, it must be part of all candidate keys.

Compute Closures of Pairs

$$\{A,B\}^+ = \{A, B, C, D, E\}$$

(a candidate key) ✓

$$\{A,C\}^+ = \{A, B, C, D, E\}$$

(a candidate key) ✓

$$\{A,D\}^+ = \{A, D\}$$

$$\{A,E\}^+ = \{A, E\}$$

$$\cancel{\{B,C\}^+ = \{B, C, D, E\}}$$

$$\cancel{\{B,D\}^+ = \{B, C, D, E\}}$$

$$\cancel{\{B,E\}^+ = \{B, C, D, E\}}$$

$$\cancel{\{C,D\}^+ = \{B, C, D, E\}}$$

$$\cancel{\{C,E\}^+ = \{B, C, D, E\}}$$

$$\cancel{\{D,E\}^+ = \{D, E\}}$$

Note

Any set of attributes containing {A,B} or {A,C} is a superkey but not a key.

Super set of {A}

not super set of
 $\{A, B\}$ or $\{A, C\}$

$\{A, D, E\}$

Compute Closures of Triples

$$\{A, D, E\}^+ = \underline{\{A, D, E\}}$$

$$\cancel{\{B, C, D\}}^+ = \{B, C, D, E\}$$

$$\cancel{\{B, C, E\}}^+ = \{B, C, D, E\}$$

$$\cancel{\{B, D, E\}}^+ = \{B, C, D, E\}$$

$$\cancel{\{C, D, E\}}^+ = \{B, C, D, E\}$$

Answer

There are no more set of attributes containing A but not superset of {A,B} or {A,C}.

Answer: $\{A, B\}$ and $\{A, C\}$ are the only candidate keys

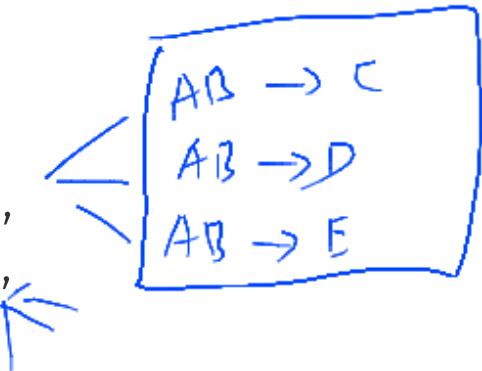
Candidate Keys = { $\{A, B\}$, $\{A, C\}$ }

$\{A, B, D\}$ ✓

~~$\{B, C, E\}$~~

Original

$\Sigma = \{$
 $\{A,B\} \rightarrow \{C,D,E\},$
 $\{A,C\} \rightarrow \{B,D,E\},$
 $\{B\} \rightarrow \{C\},$
 $\{C\} \rightarrow \{B\},$
 $\{C\} \rightarrow \{D\},$
 $\{B\} \rightarrow \{E\},$
 $\{C\} \rightarrow \{E\}$
 $\}$



Step #1: Simplify Right Hand Side (easy)

 $\Sigma_1 = \{$

- $\{A,B\} \rightarrow \{C\},$ *from $\{A,B\} \rightarrow \{C,D,E\}$*
- $\{A,B\} \rightarrow \{D\},$ *from $\{A,B\} \rightarrow \{C,D,E\}$*
- $\{A,B\} \rightarrow \{E\},$ *from $\{A,B\} \rightarrow \{C,D,E\}$*
- $\{A,C\} \rightarrow \{B\},$ *from $\{A,C\} \rightarrow \{B,D,E\}$*
- $\{A,C\} \rightarrow \{D\},$ *from $\{A,C\} \rightarrow \{B,D,E\}$*
- $\{A,C\} \rightarrow \{E\},$ *from $\{A,C\} \rightarrow \{B,D,E\}$*
- $\{B\} \rightarrow \{C\},$
- $\{C\} \rightarrow \{B\},$
- $\{C\} \rightarrow \{D\},$
- $\{B\} \rightarrow \{E\},$
- $\{C\} \rightarrow \{E\}$

 $\}$

Step #2: Simplify Left Hand Side (hard) ↗ $\Sigma_2 = \{$ $\{\textcolor{red}{A}, \textcolor{blue}{B}\} \rightarrow \{C\},$ $\{\textcolor{red}{A}, \textcolor{blue}{B}\} \rightarrow \{D\},$ $\{\textcolor{red}{A}, \textcolor{blue}{B}\} \rightarrow \{E\},$ $\{\textcolor{red}{A}, C\} \rightarrow \{B\},$ $\{\textcolor{red}{A}, C\} \rightarrow \{D\},$ $\{\textcolor{red}{A}, C\} \rightarrow \{E\},$ $\{B\} \rightarrow \{C\},$ $\{C\} \rightarrow \{B\},$ $\{C\} \rightarrow \{D\},$ $\{B\} \rightarrow \{E\},$ $\{C\} \rightarrow \{E\}$ $\}$

Step #2: Simplify Left Hand Side (hard)

 $\Sigma_2 = \{$ ~~{A,B} → {C},~~*replaced with {B} → {C}*~~{A,B} → {D},~~*replaced with {B} → {D}*~~{A,B} → {E},~~*replaced with {B} → {E}*~~{A,C} → {B},~~*replaced with {C} → {B}*~~{A,C} → {D},~~*replaced with {C} → {D}*~~{A,C} → {E},~~*replaced with {C} → {E}*

{B} → {C},

{C} → {B},

{C} → {D},

{B} → {E},

{C} → {E}

}

Step #3: Simplify Set (medium)

$$\Sigma_3 = \{$$

$\{B\} \rightarrow \{C\},$
 $\{B\} \rightarrow \{D\},$
 $\{B\} \rightarrow \{E\},$
 $\{C\} \rightarrow \{B\},$
 $\{C\} \rightarrow \{D\},$
 $\{C\} \rightarrow \{E\},$
 $\{B\} \rightarrow \{C\},$
 $\{C\} \rightarrow \{B\},$
 $\{C\} \rightarrow \{D\},$
 $\{B\} \rightarrow \{E\},$
 $\{C\} \rightarrow \{E\}$

$$\}$$

Step #3: Simplify Set (medium)

$\Sigma_3 = \{$

~~{B} → {C},~~ *duplicate*

~~{B} → {D},~~

~~{B} → {E},~~ *duplicate*

~~{C} → {B},~~ *duplicate*

~~{C} → {D},~~ *duplicate*

~~{C} → {E},~~ *duplicate*

{B} → {C},

{C} → {B},

{C} → {D},

{B} → {E},

{C} → {E}

}

Step #3: Simplify Set (medium) ➔
$$\Sigma_3 = \{$$

$\{B\} \rightarrow \{D\},$
 $\{B\} \rightarrow \{C\},$
 $\{C\} \rightarrow \{B\},$
 ~~$\{C\} \rightarrow \{D\},$~~
 $\{B\} \rightarrow \{E\},$
 ~~$\{C\} \rightarrow \{E\}$~~

$$\}$$

Step #3: Simplify Set (medium)

 $\Sigma_3 = \{$ ~~{B} → {D},~~ $\{B\} \rightarrow \{C\}$ and $\{C\} \rightarrow \{D\}$ $\{B\} \rightarrow \{C\},$ $\{C\} \rightarrow \{B\},$ $\{C\} \rightarrow \{D\},$ ~~{B} → {E},~~ $\{B\} \rightarrow \{C\}$ and $\{C\} \rightarrow \{E\}$ $\{C\} \rightarrow \{E\}$ $\}$

One Possible Answer

$$\Sigma_3 = \{$$

$\{B\} \rightarrow \{C\},$
 $\{C\} \rightarrow \{B\},$
 $\{C\} \rightarrow \{D\},$
 $\{C\} \rightarrow \{E\}$

$$\}$$

One Possible Answer

$$\Sigma_3 = \{$$

$\{B\} \rightarrow \{C\},$ ✓ } $B \rightarrow C$
 $\{C\} \rightarrow \{B\},$
 $\{C\} \rightarrow \{D\},$
 $\{C\} \rightarrow \{E\}$
 $\}$ } $C \rightarrow BDE$

Note

We could reach different minimal covers by considering the constraints in a different order.

Other Possible Answers

$$\Sigma_3 = \{$$

$\{C\} \rightarrow \{B\},$ ✓ } $C \rightarrow B$
 $\{B\} \rightarrow \{C\},$
 $\{B\} \rightarrow \{D\},$
 $\{B\} \rightarrow \{E\}$
 $\}$

$$\Sigma_3 = \{$$

$\{C\} \rightarrow \{B\},$
 $\{B\} \rightarrow \{C\},$
 $\{B\} \rightarrow \{D\},$
 $\{C\} \rightarrow \{E\}$
 $\}$

$B \rightarrow CD$
 $C \rightarrow BE$

One Possible Answer

$$\Sigma_4 = \{ \{B\} \rightarrow \{C\}, \{C\} \rightarrow \{B,D,E\} \}$$



Other Possible Answers

$$\Sigma_4 = \{ \{C\} \rightarrow \{B\}, \{B\} \rightarrow \{C,D,E\} \}$$



$$\Sigma_4 = \{ \{B\} \rightarrow \{C,D\}, \{C\} \rightarrow \{B,E\} \}$$



$$\Sigma_4 = \{ \{B\} \rightarrow \{C,E\}, \{C\} \rightarrow \{B,D\} \}$$



```
postgres=# exit
```

```
Press any key to continue . . .
```

Proof Sketch

Only for Reading ; Not Tested

Proof ↵

Let $X \rightarrow Y$ be a non-trivial but not completely non-trivial functional dependency. Then

- $X \rightarrow (Y - X)$ is completely non-trivial
 - $\Rightarrow (Y - X)$ is non-empty because $X \rightarrow Y$ is not completely non-trivial
 - $\Rightarrow X \cap (Y - X) = \emptyset$ by definition
- $X \rightarrow (Y \cap X)$ is trivial
 - $\Rightarrow (Y \cap X) \subseteq X$ by definition

So we can split $X \rightarrow Y$ into $X \rightarrow (Y - X)$ and $X \rightarrow (Y \cap X)$.

Addendum: Why Can We Split?

The proof utilizes Armstrong axioms. Re-read the proof once you have covered Armstrong axiom. We want to show that if $X \rightarrow Y$ holds, then for all $Z \subseteq Y$, $X \rightarrow Z$ holds.

1. We know that $X \rightarrow Y$. *(given in Σ)*
2. Therefore $Y \rightarrow Z$. *(Reflexivity since $Z \subseteq Y$)*
3. Therefore $X \rightarrow Z$. *(Transitivity of (1) and (2))*

□

Proof ↵

It is sufficient to show that at least one set $X \subseteq R$ exists such that $X \rightarrow R$. We omit the full proof but it is quite easy to see that $R \rightarrow R$ holds.

Note that relation is defined using set. So there are no duplicate rows.

Proof ↵

We will sketch a proof for the transitivity property.

1. Let Σ be a set of functional dependencies on a relation schema R .

Let $X \rightarrow Y$ and $Y \rightarrow Z$ be in Σ .

2. We know that for all valid instance r of R with Σ

$$\forall t_1 \in r \forall t_2 \in r ((t_1[X] = t_2[X]) \Rightarrow (t_1[Y] = t_2[Y]))$$

by definition of functional dependency.

3. We know that for all valid instance r of R with Σ

$$\forall t_1 \in r \forall t_2 \in r ((t_1[Y] = t_2[Y]) \Rightarrow (t_1[Z] = t_2[Z]))$$

by definition of functional dependency.

4. Therefore, for all valid instance r of R with Σ

$$\forall t_1 \in r \forall t_2 \in r ((t_1[X] = t_2[X]) \Rightarrow (t_1[Z] = t_2[Z]))$$

by definition of functional dependency.

5. Therefore, $X \rightarrow Z \in \Sigma^+$.

Proof Sketch #1 ↵

We will prove that for any set of attributes $S \in R$, then $S \rightarrow S^+$ can be derived by Armstrong axioms.

1. This is *recursively* true because every step of the attribute closure algorithm is of the form $S \rightarrow S^i$ and $X \rightarrow Y$ with $X \subseteq S^i$.
2. Therefore $S^i \rightarrow X$ by Reflexivity.
3. Therefore $S^i \rightarrow (X \cup S^i)$ by Augmentation of (2) with S^i .
4. Therefore $S \rightarrow (X \cup S^i)$ by Transitivity of (1) and (3).
5. Therefore $(X \cup S^i) \rightarrow (Y \cup S^i)$ by Augmentation of $X \rightarrow Y$ with S^i .
6. Therefore $S \rightarrow S^{i+1}$ where $S^{i+1} = (Y \cup S^i)$ by transitivity of (4) and (5).

Proof Sketch #2 ↵

We have shown that $S \rightarrow S^+$ can be derived. Now we need to show that if $(X \rightarrow Y) \in \Sigma^+$ then it can be derived from $X \rightarrow X^+$.

1. We know that $Y \subseteq X^+$ by property of attribute closure.
2. Therefore $X^+ \rightarrow Y$ by Reflexivity.
3. Therefore $X \rightarrow Y$ by transitivity of $X \rightarrow X^+$ and $X^+ \rightarrow Y$.

Proof ↵

This is practically from the definition of Σ^+ and equivalence.

Proof ↵

If Σ is minimal, then Σ is a minimal cover for Σ .

Proof ↵

- Let $\Sigma'' = (\Sigma' \cup (X - \{B\}) \rightarrow \{A\})$ and $\Sigma''' = \Sigma'' - \{X \rightarrow \{A\}\}$.
- Since $(X - \{B\}) \rightarrow \{A\}$ is logically entailed by Σ' , Σ' logically entails Σ'' .
As required by the step.
- Since Σ'' has all functional dependencies Σ' has, Σ'' logically entails Σ' .
Since $\Sigma'' \supseteq \Sigma'$. Therefore $\Sigma' \equiv \Sigma''$.
- Since Σ'' has all functional dependencies Σ''' has, Σ'' logically entails Σ''' .
Since $\Sigma'' \supseteq \Sigma'''$.
- Show Σ''' can logically entail Σ'' by showing that Σ''' can logically entail $X \rightarrow \{A\}$.
 - 1) We know $(X - \{B\}) \rightarrow \{A\}$ is in Σ''' .
 - 2) By augmentation of (1) with $\{B\}$, we obtain $(X - \{B\}) \cup \{B\} \rightarrow \{A\} \cup \{B\} = X \rightarrow \{A, B\}$.
 - 3) By reflexivity, since $\{A\} \subseteq \{A, B\}$, we have $\{A, B\} \rightarrow \{A\}$.
 - 4) By transitivity of (2) and (3), we have $X \rightarrow \{A\}$.

Hence Σ' and Σ''' are equivalent by transitivity of equivalence.

Proof ↵

- Let $\Sigma'' = (\Sigma' - \{X \rightarrow Y\})$.
- Since $X \rightarrow Y$ is logically entailed by Σ'' , Σ'' logically entails $\Sigma'' \cup \{X \rightarrow Y\}$.
- But $\Sigma'' \cup \{X \rightarrow Y\} = \Sigma'$, so Σ'' logically entails Σ' .
- Since Σ' has all functional dependencies Σ'' has, Σ' logically entails Σ'' .

Since $\Sigma' \supseteq \Sigma''$.

Hence Σ' and Σ'' are equivalent.

Proof ↵

Although not mentioned as a theorem, we can show that given $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow (Y \cup Z)$ holds.

1. We know that $X \rightarrow Y$. *(given in Σ)*
2. We know that $X \rightarrow Z$. *(given in Σ)*
3. Therefore $X \rightarrow (X \cup Y)$. *(Augmentation of (1) with X)*
4. Therefore $(X \cup Y) \rightarrow (Y \cup Z)$. *(Augmentation of (2) with Y)*
5. Therefore $X \rightarrow (Y \cup Z)$. *(Transitivity of (3) and (4))*

□

This together with addendum on the proof of Theorem #1 implies that we can merge and split the right hand side of any functional dependencies.

Steps

Consider $\{A, B\} \rightarrow \{C\}$.

1. Check if we can remove **B** from $\{A, B\} \rightarrow \{C\}$
 - A. Compute $\{A\}^+$ with respect to Σ_2 . $\{A\}^+ = \{A\}$
 - B. Since $\{C\} \not\subseteq \{A\}^+$, we cannot remove **B** from $\{A, B\} \rightarrow \{C\}$
2. Check if we can remove **A** from $\{A, B\} \rightarrow \{C\}$
 - A. Compute $\{B\}^+$ with respect to Σ_2 . $\{B\}^+ = \{B, C, D, E\}$
 - B. Since $\{C\} \not\subseteq \{B\}^+$, we can remove **A** from $\{A, B\} \rightarrow \{C\}$
 - C. Replace $\{A, B\} \rightarrow \{C\}$ with $\{B\} \rightarrow \{C\}$

Current Σ

$$\Sigma_2 = \{$$

~~$\{A, B\} \rightarrow \{C\}$~~ $\Rightarrow \{B\} \rightarrow \{C\},$

$\{A, B\} \rightarrow \{D\},$

$\{A, B\} \rightarrow \{E\},$

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

$\{A, C\} \rightarrow \{D\},$

$\{A, C\} \rightarrow \{E\},$

$\{B\} \rightarrow \{C\},$

$\{C\} \rightarrow \{B\},$

$\{C\} \rightarrow \{D\},$

$\{B\} \rightarrow \{E\},$

$\{C\} \rightarrow \{E\}$

$$\}$$

Steps

Consider $\{A,B\} \rightarrow \{D\}$.

1. Check if we can remove **B** from $\{A,B\} \rightarrow \{D\}$
 - A. Compute $\{A\}^+$ with respect to Σ_2 . $\{A\}^+ = \{A\}$
 - B. Since $\{D\} \not\subseteq \{A\}^+$, we cannot remove **B** from $\{A,B\} \rightarrow \{D\}$

2. Check if we can remove **A** from $\{A,B\} \rightarrow \{D\}$
 - A. Compute $\{B\}^+$ with respect to Σ_2 . $\{B\}^+ = \{B, C, D, E\}$
 - B. Since $\{D\} \not\subseteq \{B\}^+$, we can remove **A** from $\{A,B\} \rightarrow \{D\}$
 - C. Replace $\{A,B\} \rightarrow \{D\}$ with $\{B\} \rightarrow \{D\}$

Current Σ

$$\begin{aligned}\Sigma_2 = \{ & \\ & \{B\} \rightarrow \{C\}, \\ & \cancel{\{A,B\} \rightarrow \{D\}} \Rightarrow \{B\} \rightarrow \{D\}, \\ & \{A,B\} \rightarrow \{E\}, \\ & \{A,C\} \rightarrow \{B\}, \\ & \{A,C\} \rightarrow \{D\}, \\ & \{A,C\} \rightarrow \{E\}, \\ & \{B\} \rightarrow \{C\}, \\ & \{C\} \rightarrow \{B\}, \\ & \{C\} \rightarrow \{D\}, \\ & \{B\} \rightarrow \{E\}, \\ & \{C\} \rightarrow \{E\}\end{aligned}\}$$

Steps

Consider $\{A, B\} \rightarrow \{E\}$.

1. Check if we can remove **B** from $\{A, B\} \rightarrow \{E\}$
 - A. Compute $\{A\}^+$ with respect to Σ_2 . $\{A\}^+ = \{A\}$
 - B. Since $\{E\} \not\subseteq \{A\}^+$, we cannot remove **B** from $\{A, B\} \rightarrow \{E\}$

2. Check if we can remove **A** from $\{A, B\} \rightarrow \{E\}$
 - A. Compute $\{B\}^+$ with respect to Σ_2 . $\{B\}^+ = \{B, C, D, E\}$
 - B. Since $\{E\} \not\subseteq \{B\}^+$, we can remove **A** from $\{A, B\} \rightarrow \{E\}$
 - C. Replace $\{A, B\} \rightarrow \{E\}$ with $\{B\} \rightarrow \{E\}$

Current Σ

$$\begin{aligned}\Sigma_2 = \{ & \\ & \{B\} \rightarrow \{C\}, \\ & \{B\} \rightarrow \{D\}, \\ & \cancel{\{A, B\} \rightarrow \{E\}} \Rightarrow \{B\} \rightarrow \{E\}, \\ & \{A, C\} \rightarrow \{B\}, \\ & \{A, C\} \rightarrow \{D\}, \\ & \{A, C\} \rightarrow \{E\}, \\ & \{B\} \rightarrow \{C\}, \\ & \{C\} \rightarrow \{B\}, \\ & \{C\} \rightarrow \{D\}, \\ & \{B\} \rightarrow \{E\}, \\ & \{C\} \rightarrow \{E\} \\\}\end{aligned}$$

Steps

Consider $\{A,C\} \rightarrow \{B\}$.

1. Check if we can remove **C** from $\{A,C\} \rightarrow \{B\}$
 - A. Compute $\{A\}^+$ with respect to Σ_2 . $\{A\}^+ = \{A\}$
 - B. Since $\{B\} \not\subseteq \{A\}^+$, we cannot remove **C** from $\{A,C\} \rightarrow \{B\}$
2. Check if we can remove **A** from $\{A,C\} \rightarrow \{B\}$
 - A. Compute $\{C\}^+$ with respect to Σ_2 . $\{C\}^+ = \{B, C, D, E\}$
 - B. Since $\{B\} \not\subseteq \{C\}^+$, we can remove **A** from $\{A,C\} \rightarrow \{B\}$
 - C. Replace $\{A,C\} \rightarrow \{B\}$ with $\{C\} \rightarrow \{B\}$

Current Σ

$$\begin{aligned}\Sigma_2 = \{ & \\ & \{B\} \rightarrow \{C\}, \\ & \{B\} \rightarrow \{D\}, \\ & \{B\} \rightarrow \{E\}, \\ & \cancel{\{A,C\} \rightarrow \{B\}} \Rightarrow \{C\} \rightarrow \{B\}, \\ & \{A,C\} \rightarrow \{D\}, \\ & \{A,C\} \rightarrow \{E\}, \\ & \{B\} \rightarrow \{C\}, \\ & \{C\} \rightarrow \{B\}, \\ & \{C\} \rightarrow \{D\}, \\ & \{B\} \rightarrow \{E\}, \\ & \{C\} \rightarrow \{E\} \end{aligned}\}$$

Steps

Consider $\{A,C\} \rightarrow \{D\}$.

1. Check if we can remove **C** from $\{A,C\} \rightarrow \{D\}$
 - A. Compute $\{A\}^+$ with respect to Σ_2 . $\{A\}^+ = \{A\}$
 - B. Since $\{D\} \not\subseteq \{A\}^+$, we cannot remove **C** from $\{A,C\} \rightarrow \{D\}$
2. Check if we can remove **A** from $\{A,C\} \rightarrow \{D\}$
 - A. Compute $\{C\}^+$ with respect to Σ_2 . $\{C\}^+ = \{B, C, D, E\}$
 - B. Since $\{D\} \not\subseteq \{C\}^+$, we can remove **A** from $\{A,C\} \rightarrow \{D\}$
 - C. Replace $\{A,C\} \rightarrow \{D\}$ with $\{C\} \rightarrow \{D\}$

Current Σ

$$\begin{aligned}\Sigma_2 = \{ & \{B\} \rightarrow \{C\}, \\ & \{B\} \rightarrow \{D\}, \\ & \{B\} \rightarrow \{E\}, \\ & \{C\} \rightarrow \{B\}, \\ & \{A, C\} \rightarrow \{D\} \Rightarrow \{C\} \rightarrow \{D\}, \\ & \{A, C\} \rightarrow \{E\}, \\ & \{B\} \rightarrow \{C\}, \\ & \{C\} \rightarrow \{B\}, \\ & \{C\} \rightarrow \{D\}, \\ & \{B\} \rightarrow \{E\}, \\ & \{C\} \rightarrow \{E\}\end{aligned}\}$$

Steps

Consider $\{A,C\} \rightarrow \{E\}$.

1. Check if we can remove **C** from $\{A,C\} \rightarrow \{E\}$
 - A. Compute $\{A\}^+$ with respect to Σ_2 . $\{A\}^+ = \{A\}$
 - B. Since $\{E\} \not\subseteq \{A\}^+$, we cannot remove **C** from $\{A,C\} \rightarrow \{E\}$
2. Check if we can remove **A** from $\{A,C\} \rightarrow \{E\}$
 - A. Compute $\{C\}^+$ with respect to Σ_2 . $\{C\}^+ = \{B, C, D, E\}$
 - B. Since $\{E\} \not\subseteq \{C\}^+$, we can remove **A** from $\{A,C\} \rightarrow \{E\}$
 - C. Replace $\{A,C\} \rightarrow \{E\}$ with $\{C\} \rightarrow \{E\}$

Current Σ

$$\begin{aligned}\Sigma_2 = \{ & \{B\} \rightarrow \{C\}, \\ & \{B\} \rightarrow \{D\}, \\ & \{B\} \rightarrow \{E\}, \\ & \{C\} \rightarrow \{B\}, \\ & \{C\} \rightarrow \{D\}, \\ & \{A, C\} \rightarrow \{E\} \Rightarrow \{C\} \rightarrow \{E\}, \\ & \{B\} \rightarrow \{C\}, \\ & \{C\} \rightarrow \{B\}, \\ & \{C\} \rightarrow \{D\}, \\ & \{B\} \rightarrow \{E\}, \\ & \{C\} \rightarrow \{E\}\}\end{aligned}$$

Steps

At this point, it is safe to stop because the rest of the functional dependencies only have a single attribute on the left-hand side.

While it is possible in general to have an empty set as the left-hand side (*e.g.*, $\{\} \rightarrow \{A\}$), our initial Σ must already have at least one functional dependency with an empty set as the left-hand side.

Since there is no such functional dependencies, it will not be suddenly introduced.

Current Σ

$$\begin{aligned}\Sigma_2 = \{ & \\ & \{B\} \rightarrow \{C\}, \\ & \{B\} \rightarrow \{D\}, \\ & \{B\} \rightarrow \{E\}, \\ & \{C\} \rightarrow \{B\}, \\ & \{C\} \rightarrow \{D\}, \\ & \{C\} \rightarrow \{E\}, \\ & \textcolor{blue}{\{B\} \rightarrow \{C\}}, \\ & \textcolor{blue}{\{C\} \rightarrow \{B\}}, \\ & \textcolor{blue}{\{C\} \rightarrow \{D\}}, \\ & \textcolor{blue}{\{B\} \rightarrow \{E\}}, \\ & \textcolor{blue}{\{C\} \rightarrow \{E\}}\end{aligned}\}$$

Working Σ
$$\begin{aligned}\Sigma'_2 = \{ \\ \{B\} \rightarrow \{D\}, \\ \{B\} \rightarrow \{C\}, \\ \{C\} \rightarrow \{B\}, \\ \{C\} \rightarrow \{D\}, \\ \{B\} \rightarrow \{E\}, \\ \{C\} \rightarrow \{E\}\end{aligned}\}$$
Steps

Consider $\{B\} \rightarrow \{D\}$, check if we can remove this functional dependency.

1. Compute Σ'_2 (see left) by removing $\{B\} \rightarrow \{D\}$ from Σ_2 .
2. Compute $\{B\}^+$ with respect to Σ'_2
$$\{B\}^+ = \{B, C, D, E\}$$
3. Since $\{D\} \subseteq \{B\}^+$, we can remove $\{B\} \rightarrow \{D\}$

Current Σ
$$\begin{aligned}\Sigma_2 = \{ \\ \cancel{\{B\}} \rightarrow \cancel{\{D\}}, \\ \{B\} \rightarrow \{C\}, \\ \{C\} \rightarrow \{B\}, \\ \{C\} \rightarrow \{D\}, \\ \{B\} \rightarrow \{E\}, \\ \{C\} \rightarrow \{E\}\end{aligned}\}$$

Working Σ

$$\Sigma'_2 = \{$$

$$\begin{aligned} & \{B\} \rightarrow \{C\}, \\ & \{C\} \rightarrow \{B\}, \\ & \{C\} \rightarrow \{D\}, \\ & \{B\} \rightarrow \{E\}, \\ & \{C\} \rightarrow \{E\} \end{aligned}$$

}

Steps

Consider $\{B\} \rightarrow \{C\}$, check if we can remove this functional dependency.

1. Compute Σ'_2 (*see left*) by removing $\{B\} \rightarrow \{C\}$ from Σ_2 .
2. Compute $\{B\}^+$ with respect to Σ'_2
$$\{B\}^+ = \{B, E\}$$
3. Since $\{C\} \not\subseteq \{B\}^+$, we cannot remove $\{B\} \rightarrow \{C\}$

Current Σ

$$\Sigma_2 = \{$$

$$\begin{aligned} & \{B\} \rightarrow \{C\}, \\ & \{C\} \rightarrow \{B\}, \\ & \{C\} \rightarrow \{D\}, \\ & \{B\} \rightarrow \{E\}, \\ & \{C\} \rightarrow \{E\} \end{aligned}$$

}

Working Σ

$$\Sigma'_2 = \{$$

$$\begin{aligned} \{B\} &\rightarrow \{C\}, \\ \{C\} &\rightarrow \{B\}, \\ \{C\} &\rightarrow \{D\}, \\ \{B\} &\rightarrow \{E\}, \\ \{C\} &\rightarrow \{E\} \end{aligned}$$

}

Steps

Consider $\{C\} \rightarrow \{B\}$, check if we can remove this functional dependency.

1. Compute Σ'_2 (*see left*) by removing $\{C\} \rightarrow \{B\}$ from Σ_2 .
2. Compute $\{C\}^+$ with respect to Σ'_2
 $\{C\}^+ = \{C, D, E\}$
3. Since $\{B\} \not\subseteq \{C\}^+$, we cannot remove $\{C\} \rightarrow \{B\}$

Current Σ

$$\Sigma_2 = \{$$

$$\begin{aligned} \{B\} &\rightarrow \{C\}, \\ \{C\} &\rightarrow \{B\}, \\ \{C\} &\rightarrow \{D\}, \\ \{B\} &\rightarrow \{E\}, \\ \{C\} &\rightarrow \{E\} \end{aligned}$$

}

Working Σ

$$\Sigma'_2 = \{$$

$$\begin{aligned} & \{B\} \rightarrow \{C\}, \\ & \{C\} \rightarrow \{B\}, \\ & \cancel{\{C\} \rightarrow \{D\}}, \\ & \{B\} \rightarrow \{E\}, \\ & \{C\} \rightarrow \{E\} \end{aligned}$$

}

Steps

Consider $\{C\} \rightarrow \{D\}$, check if we can remove this functional dependency.

1. Compute Σ'_2 (see left) by removing $\{C\} \rightarrow \{D\}$ from Σ_2 .
2. Compute $\{C\}^+$ with respect to Σ'_2
$$\{C\}^+ = \{B, C, E\}$$
3. Since $\{D\} \not\subseteq \{C\}^+$, we cannot remove $\{C\} \rightarrow \{D\}$

Current Σ

$$\Sigma_2 = \{$$

$$\begin{aligned} & \{B\} \rightarrow \{C\}, \\ & \{C\} \rightarrow \{B\}, \\ & \{C\} \rightarrow \{D\}, \\ & \{B\} \rightarrow \{E\}, \\ & \{C\} \rightarrow \{E\} \end{aligned}$$

}

Working Σ

$$\Sigma'_2 = \{$$

$$\begin{aligned} \{B\} &\rightarrow \{C\}, \\ \{C\} &\rightarrow \{B\}, \\ \{C\} &\rightarrow \{D\}, \\ \{B\} &\rightarrow \{E\}, \\ \{C\} &\rightarrow \{E\} \end{aligned}$$

}

Steps

Consider $\{B\} \rightarrow \{E\}$, check if we can remove this functional dependency.

1. Compute Σ'_2 (see left) by removing $\{B\} \rightarrow \{E\}$ from Σ_2 .
2. Compute $\{B\}^+$ with respect to Σ'_2
$$\{B\}^+ = \{B, C, D, E\}$$
3. Since $\{E\} \subseteq \{B\}^+$, we can remove $\{B\} \rightarrow \{E\}$

Current Σ

$$\Sigma_2 = \{$$

$$\begin{aligned} \{B\} &\rightarrow \{C\}, \\ \{C\} &\rightarrow \{B\}, \\ \{C\} &\rightarrow \{D\}, \\ \cancel{\{B\}} &\rightarrow \cancel{\{E\}}, \\ \{C\} &\rightarrow \{E\} \end{aligned}$$

}

Working Σ

$$\Sigma'_2 = \{$$

$$\begin{aligned} \{B\} &\rightarrow \{C\}, \\ \{C\} &\rightarrow \{B\}, \\ \{C\} &\rightarrow \{D\}, \end{aligned}$$

$$\begin{aligned} \{C\} &\rightarrow \{E\} \\ \} \end{aligned}$$

Steps

Consider $\{C\} \rightarrow \{E\}$, check if we can remove this functional dependency.

1. Compute Σ'_2 (*see left*) by removing $\{C\} \rightarrow \{E\}$ from Σ_2 .
2. Compute $\{C\}^+$ with respect to Σ'_2
$$\{C\}^+ = \{B, C, D\}$$
3. Since $\{E\} \not\subseteq \{C\}^+$, we can remove $\{C\} \rightarrow \{E\}$

Current Σ

$$\Sigma_2 = \{$$

$$\begin{aligned} \{B\} &\rightarrow \{C\}, \\ \{C\} &\rightarrow \{B\}, \\ \{C\} &\rightarrow \{D\}, \end{aligned}$$

$$\begin{aligned} \{C\} &\rightarrow \{E\} \\ \} \end{aligned}$$

```
postgres=# exit
```

Press any key to continue . . .

