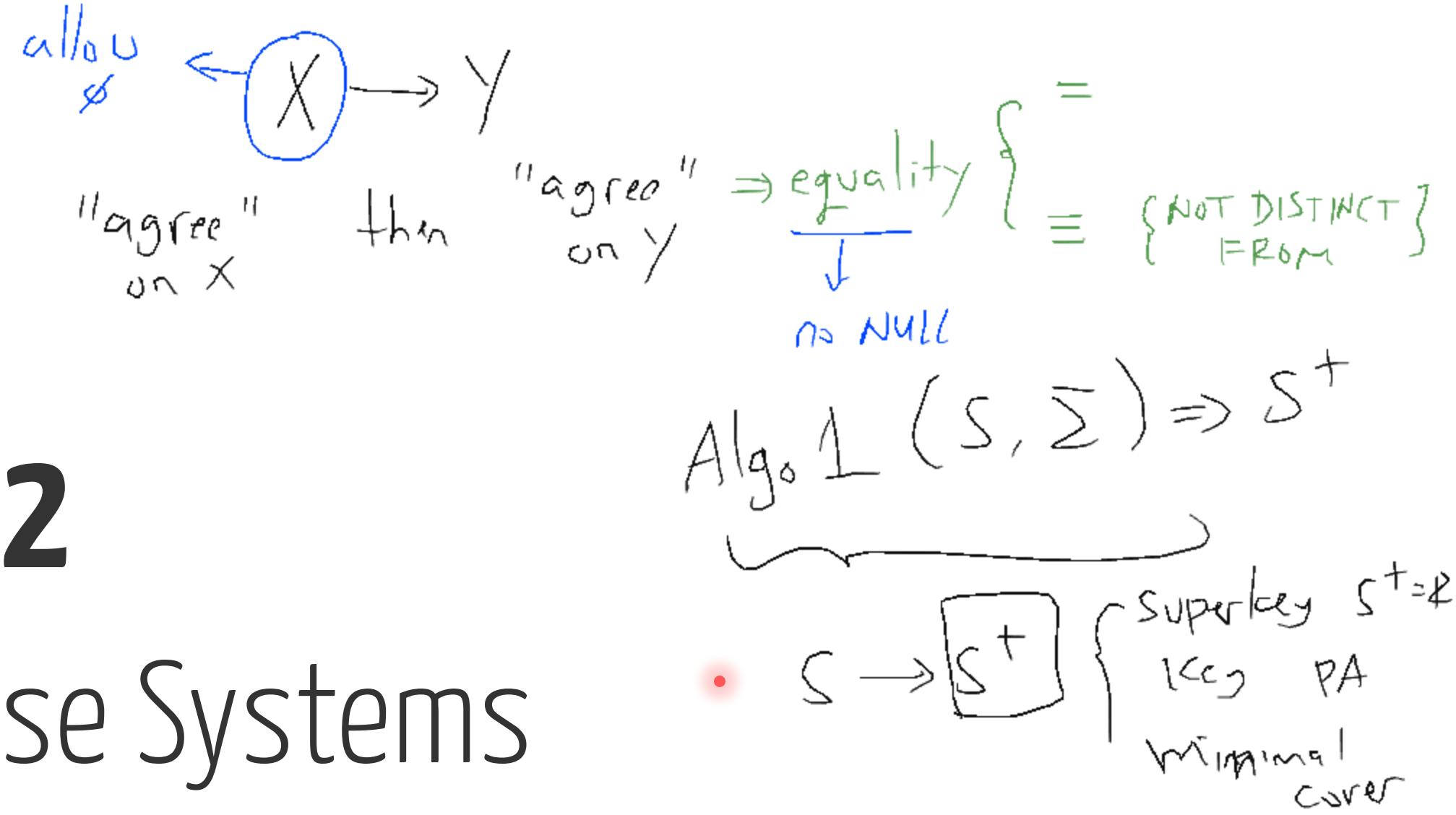


CS2102

Database Systems

L10: Boyce-Codd Normal Form

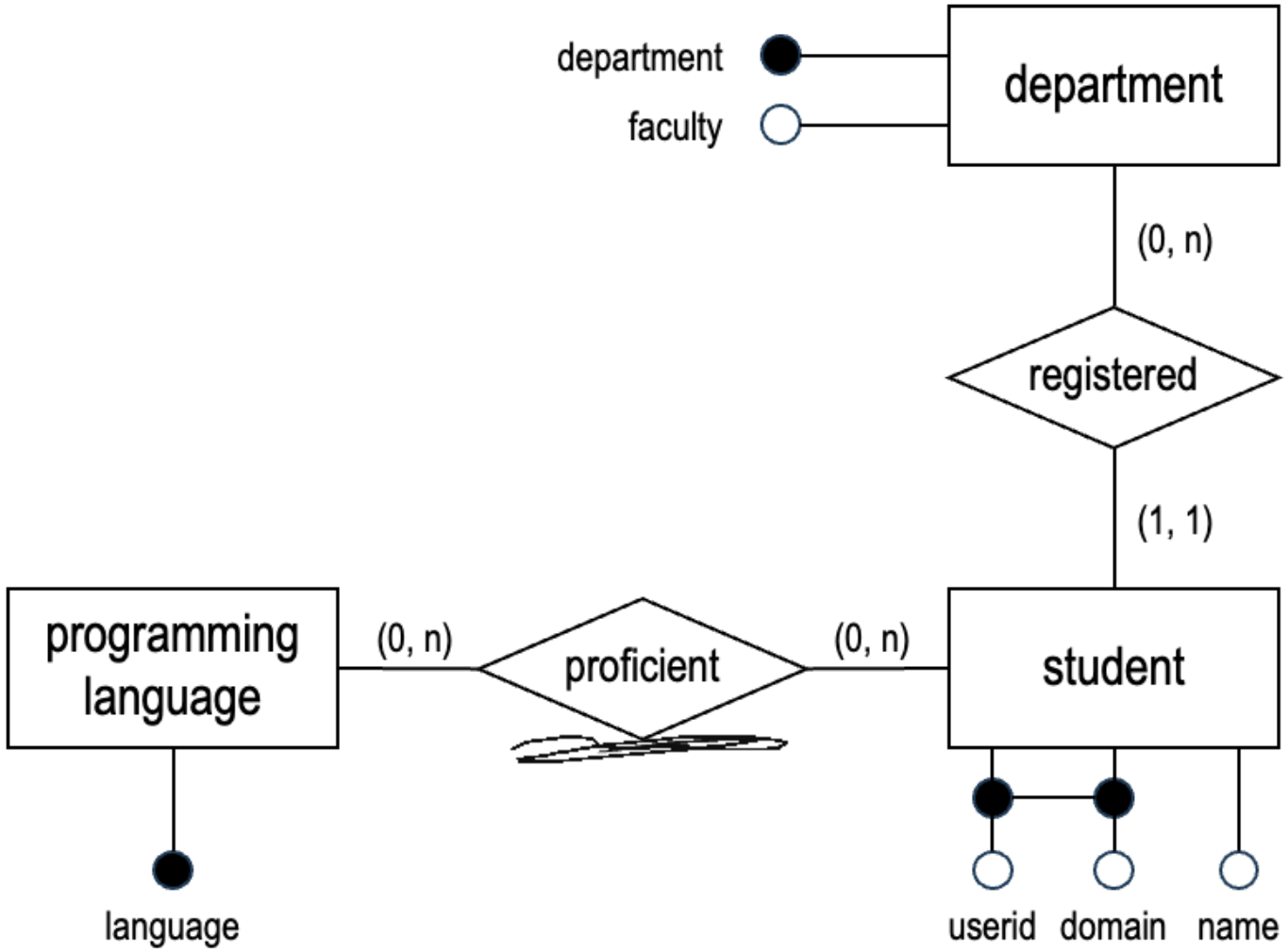
Requirement

Proficiency

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

Question

How many tables do we need to enforce all constraints?

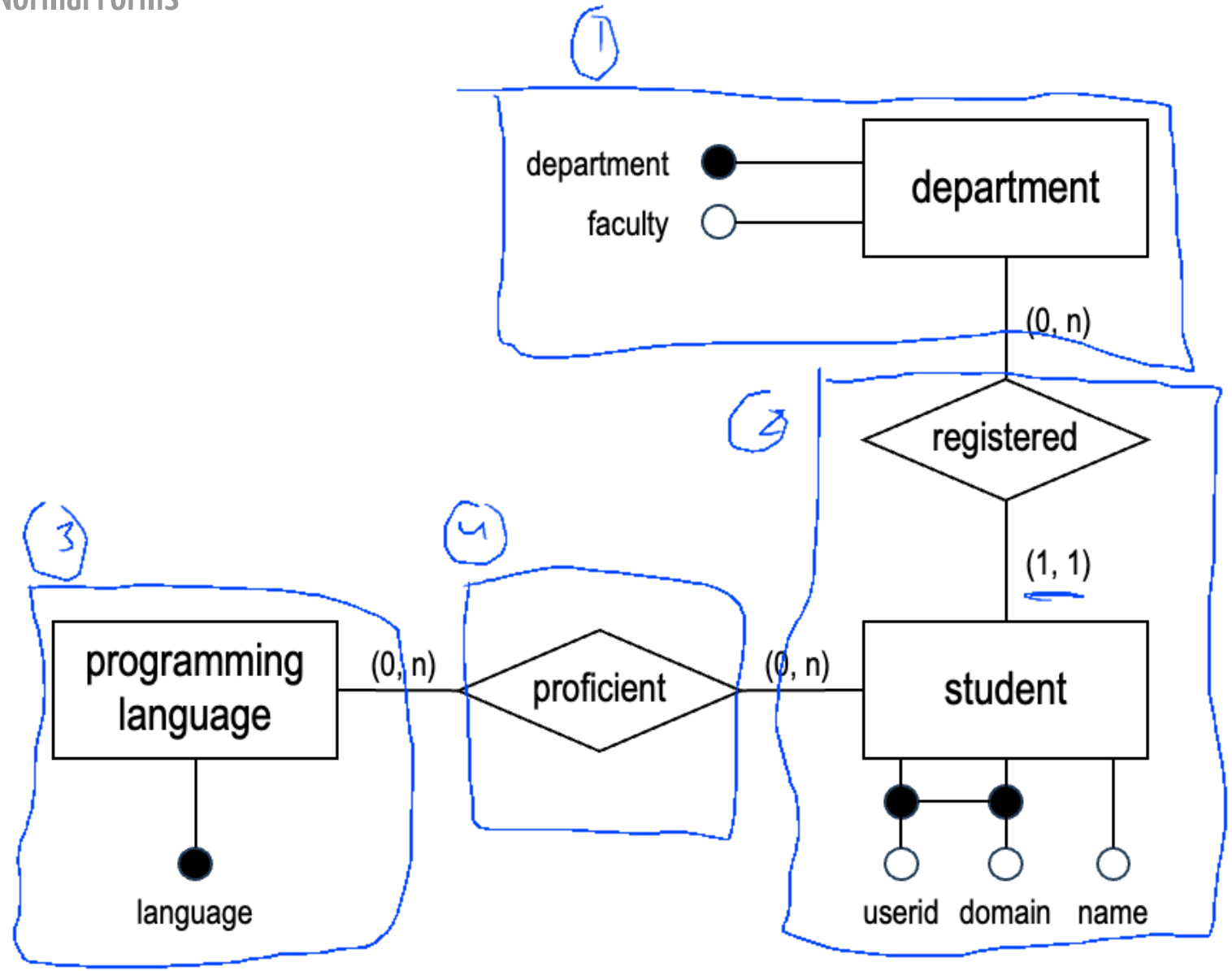


Schema

Proficiency

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

1. *department(<u>department</u>, faculty)*
2. *student(<u>userid, domain</u>, name, department)*
3. *language(<u>language</u>)*
4. *proficient(<u>language</u>, userid, domain)*



Schema

One Table

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

(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

Mapping

A: name
B: userid
C: domain
D: department
E: faculty
F: language

Schema

One Table

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

(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

FD

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

Question

Can we enforce this?

Schema

One Table

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

(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

FD

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

Question

Can we enforce this?

YES!

$\{B,C,F\}$ as *PK*.

Schema

One Table

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

(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

FD

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

Question

Can we enforce this?

Schema

One Table

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

(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

FD

$$\{D\} \rightarrow \{E\}$$

Question

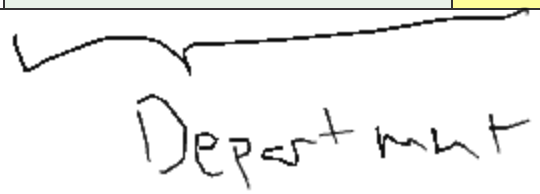
Can we enforce this?

Schema

One Table

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

(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



FD

$$\{D\} \rightarrow \{E\}$$

Question

Can we enforce this?

Split!

Create **Department**.

Redundant

Proficiency

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

Department

(D) department	(E) faculty
computer science	computing
computer engineering	engineering
pharmacy	medicine

$\Pi_{[dept, fac]}(Proficiency)$

Redundant Storage

Caused by {D} → {E}. ***Solution:*** (i) Split the table, (ii) Record {D,E} separately *(e.g., `Department` table)*, but (iii) Still need {D} to be remembered *(to get back the `faculty`)*.

Update

Proficiency

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

Department

(D) department	(E) faculty
computer science	**informatics**
computer engineering	engineering
pharmacy	medicine

Update Anomaly

Same issue as before, caused by {D} → {E}. Splitting table allows us to update only on the `Department` table.

Delete

Proficiency

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

Department

(D) department	(E) faculty
computer science	informatics
computer engineering	engineering
pharmacy	**medicine**

Deletion Anomaly

Same issue as before, caused by {D} → {E}. With split table, even if Ami Mokhtar is deleted, the department and faculty is still recorded in the `Department` table.

Insert

Proficiency

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

Department

(D) department	(E) faculty
computer science	informatics
computer engineering	engineering
pharmacy	medicine
data analytics	**business**

Insertion Anomaly

Same issue as before, caused by $\{D\} \rightarrow \{E\}$. With split table, we can still record new department/faculty in the `Department` table.

Recombination

Proficiency

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

Department

(D) department	(E) faculty
computer science	computing
computer engineering	engineering
pharmacy	medicine
data analytics	business

Recombination

We can use ***outer join**** *(is it left, right, or full outer join?)* to combine the two tables and recreate the original table with NULL values that we intended.

Recombination

Proficiency

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

Department

(D) department	(E) faculty
computer science	computing
computer engineering	engineering
pharmacy	medicine
data analytics	business

```sql
SELECT p.name, p.userid, p.domain, p.department, d.faculty, p.language
FROM proficiency p FULL OUTER JOIN department d
  ON p.department = d.department;
```

Recombination

Combined

(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
null	null	null	pharmacy	medicine	null
null	null	null	data analytics	business	null

```
SELECT p.name, p.userid, p.domain, p.department, d.faculty, p.language
FROM proficiency p FULL OUTER JOIN department d
  ON p.department = d.department;
```

Recombination

Combined

(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

```sql
SELECT *
FROM proficiency p NATURAL JOIN department d;
-- automatic condition/attributes
```

Functional Dependency

For FD, we are only concerned with `NATURAL JOIN`.

To Do

Proficiency

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

Department

(D) department	(E) faculty
computer science	computing
computer engineering	engineering
pharmacy	medicine
data analytics	business

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

To Do

Check that we ***still*** have similar anomalies with the other non-trivial functional dependencies, even after we split the table as above *(e.g., {B,C} → {A,D})*.

Purpose

Purpose of Normal Forms

The purpose of the ***normal forms*** is to recognize designs that **enforce functional dependencies** by means of the main SQL constraints (*i.e.*, `PRIMARY KEY`, `UNIQUE`, `NOT NULL`, `FOREIGN KEY`, and `CHECK`) and thus protects against data anomalies.

Purpose of Normalization

The purpose of the ***normalization*** is to **transform** *(decompose or split tables)* a poor design into a design that enforces functional dependencies by means of the main SQL constraints.

* NF does not guarantee "nice properties"
  ↳ Algorithm guarantees the properties

Keys

(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

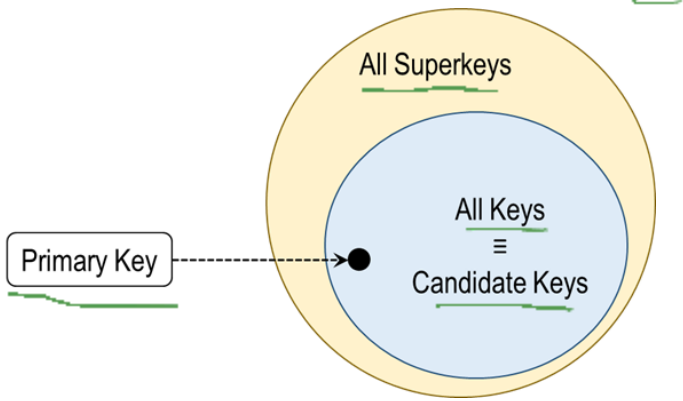
Functional Dependencies

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

Candidate Keys

Verify that one candidate key is {userid, domain, language}.
Show that this is the *only candidate key*.

Using the mapping, the candidate keys are { {B,C,F} }.



Basic

The candidate key is {userid, domain, language},

yet some attributes such as name and department depend only on a *proper subset of a candidate key* (*i.e., they are not **fully dependent** on the primary key*).

$$\{\texttt{userid}, \texttt{domain}\} \rightarrow \{\texttt{name}, \texttt{department}\}$$

yet some attributes such as faculty also depend on *other attributes* (*i.e., they are **transitively dependent** on the primary key*).

$$\{\texttt{department}\} \rightarrow \{\texttt{faculty}\}$$

Issues

These attributes describe the **student** and the **department**. They do not depend on or inform us about the relationship with the programming language. We are *mixing several entities and relationships* in the same table.

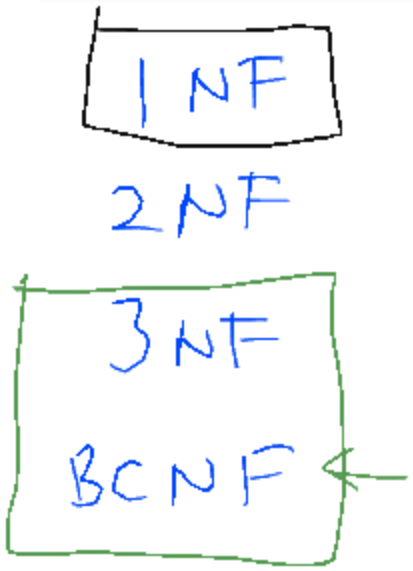
Definition

3.5 NF

Boyce-Codd Normal Form

A non-key field must provide a fact about the key[s], the whole key[s], and nothing but the key[s], *[so help me Codd.]*

W. Kent in **"A Simple Guide to Five Normal Forms in Relational Database Theory"**
Communication of the ACM, Volume 26, Number 2 (1983)



4NF
5NF
6NF

✓ "Adi"

✗ "99999999, 88888888"

Theorem

Boyce-Codd Normal Form

A relation R with a set of functional dependencies Σ is in *BCNF* if and only if **for every** functional dependency $X \rightarrow \{A\} \in \Sigma^+$: ← set of fd closure

- $X \rightarrow \{A\}$ is *trivial*, **or**
- X is a *superkey*

Lemma 2. *A relation R is BCNF iff for every elementary FD of R, say, $X \rightarrow A$, X is a key of R.*

Proof. Easy.

Note

For relation R **before decomposition**, it is sufficient only to look at Σ.

*A New Normal Form for the Design of Relational Database Schemata

Theorem

Boyce-Codd Normal Form

A relation R with a set of functional dependencies Σ is in *BCNF* if and only if **for every** functional dependency $X \rightarrow \{A\} \in \Sigma^+$:

- $X \rightarrow \{A\}$ is *trivial*, **or**
- X is a *superkey*

negation

More Formally

$$\forall\, (X \rightarrow \{A\}) \in \Sigma^+ \colon \underbrace{(A \in X)}_{\text{trivial}} \vee \underbrace{(X^+ = R)}_{\text{superkey}}$$

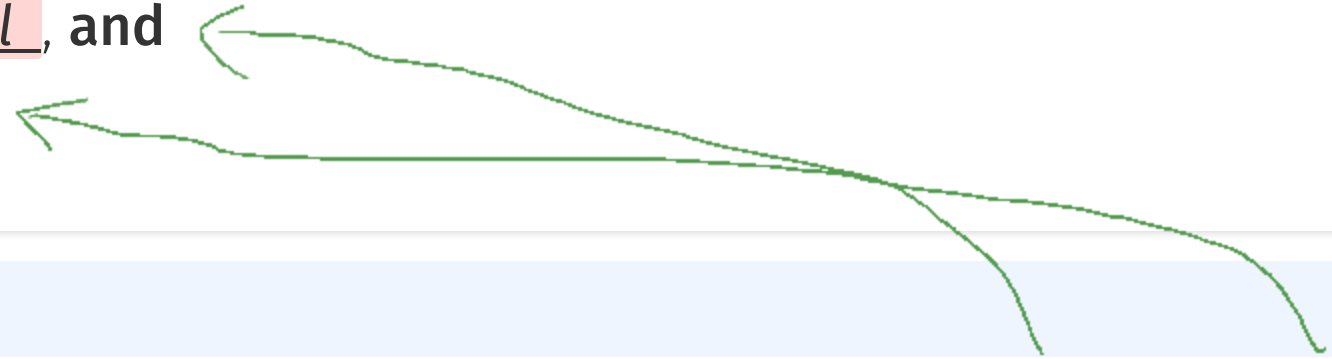
*A New Normal Form for the Design of Relational Database Schemata

Theorem

Boyce-Codd Normal Form Violation

A relation R with a set of functional dependencies Σ is NOT in *BCNF* if and only if **there exists** a functional dependency $X \rightarrow \{A\} \in \Sigma^+$:

- $X \rightarrow \{A\}$ is NON-*trivial*, **and**
- X is NOT a *superkey*



More Formally

$$\neg\ \forall\ (X \rightarrow \{A\}) \in \Sigma^+ :\ (A \in X)\ \vee\ (X^+ = R) \quad \equiv \quad \exists\ (X \rightarrow \{A\}) \in \Sigma^+ :\ (A \notin X)\ \vee\ (X^+ \neq R)$$

*A New Normal Form for the Design of Relational Database Schemata

Theorem

Boyce-Codd Normal Form Violation

A relation R with a set of functional dependencies Σ is NOT in *BCNF* if and only if **there exists** a functional dependency $X \rightarrow \{A\} \in \Sigma^+$:

- $X \rightarrow \{A\}$ is NON-*trivial*, **and**
- X is NOT a *superkey*

More Formally

$\neg\ \forall\ (X \rightarrow \{A\}) \in \Sigma^+$: $(A \in X)$ $\vee$ $(X^+ = R)$ $\equiv$ $\exists\ (X \rightarrow \{A\}) \in \Sigma^+$: $(A \notin X)$ $\vee$ $(X^+ \neq R)$

More but not All

$X \subset X^+ \neq R$

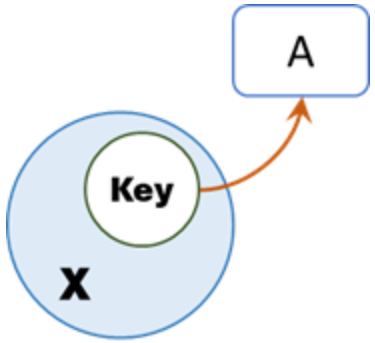
$X \subset X^+$ ⇒ more ⟶ $X \rightarrow X^+$ contains non trivial fd

$X^+ \neq R$ ⇒ not all ⟶ X is not a superkey

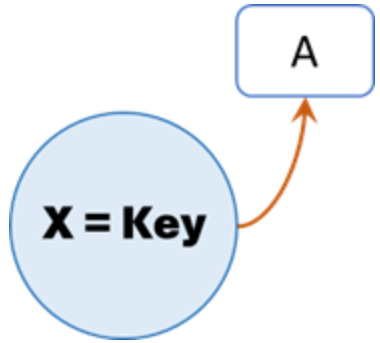
*A New Normal Form for the Design of Relational Database Schemata

Intuition

For some candidate key and a functional dependency $X \rightarrow \{A\}$, we must have one of the following:



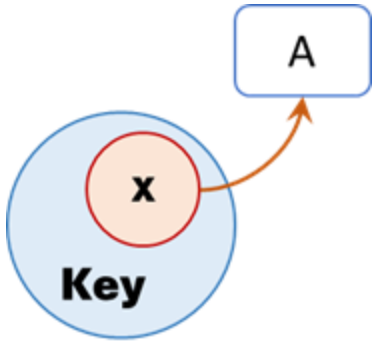
X is a **superset** of the candidate key.
*(X is a **superkey**)*



X is **the** candidate key.
*(X is a **key**)*

Intuition

For some candidate key and a functional dependency $X \rightarrow \{A\}$, we **cannot** have the following:



X is a **subset** of the candidate key.

(How can this occur?)

Note

Usually this indicates that **X** and **A** are part of an **entity set** unrelated to the rest of the attributes.

Example

Proficiency

(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

BC → AD
BCP → ADE
D → E

{E} ⊆ {D}
{D}⁺ = {D,E}

Candidate Keys

userid
domain
language

BC⁺ = ABCD ≠ R

Violation

Consider {department} → {faculty}.

non-trivial: {faculty} ⊄ {department}

not superkey: {department}⁺ = {department, faculty}

More but not All

{D}⁺ = {D,E}

{D} ⊂ {D,E} ≠ {A,B,C,D,E,F}

Decomposition

Proficiency

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

Department

(D) department	(E) faculty
computer science	informatics
computer engineering	engineering
pharmacy	medicine

Note

Department := $\pi_{[\text{department, faculty}]}(\text{Proficiency})$

Solution

The solution is *(again)* to decompose the table into two ***fragments***. We now need to study the two new tables and their *(projected)* functional dependencies.

Decomposition ℹ | Lossless-Join | Projection | Preservation | Algorithm

Definition

Binary Decomposition

A ***binary decomposition*** of a table R is a **pair** of tables $\delta = \{R_1, R_2\}$ such that

$R = R_1 \cup R_2$

General Decomposition

A ***decomposition*** of a table R is a set of tables $\delta = \{R_1, ..., R_n\}$ such that

$R = R_1 \cup ... \cup R_n$

Note

In other words, we do not lose any attributes from the decomposition.

Definition

Lossless-Join Decomposition

A binary decomposition is ***lossless-join*** if and only if the **natural join** of its two **fragments** *(i.e., the two tables resulting from the decomposition)* ***equals the initial table***. Otherwise, the decomposition is **lossy**.

```
SELECT *
FROM proficiency p NATURAL JOIN department d;
```

Note

Lossless-join property may also be called **non-additive join**. Lossy join *adds rows*.

In other words, given a decomposition of R into $\delta = \{R_1, R_2\}$, we have a lossless-join decomposition if $R_1 \bowtie R_2 = R$.

Examples

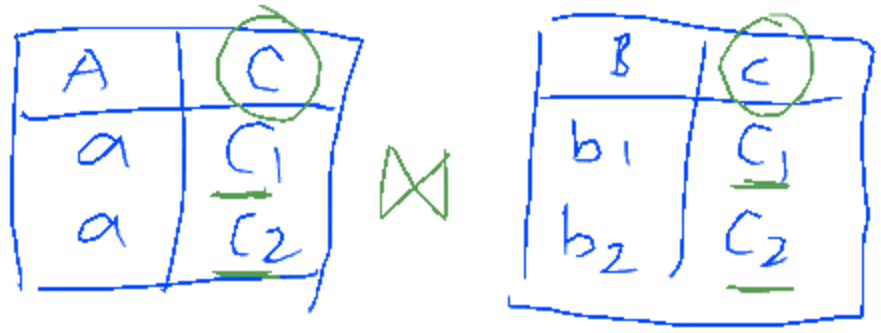
Lossless-Join

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

Original

A	B	C
a	b_1	c_1
a	b_2	c_2

Decomposition into {A,C} and {B,C}



Recombined

A	B	C
a	b_1	c_1
a	b_2	c_2

Examples

Lossless-Join

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

Original

A	B	C
a	b_1	c_1
a	b_2	c_2

Decomposition into {A,C} and {B,C}

$R_1 := \pi_{[A,C]}(R)$

A	C
a	c_1
a	c_2

$R_2 := \pi_{[B,C]}(R)$

B	C
b_1	c_1
b_2	c_2

Recombined

$R_1 \bowtie R_2$

A	B	C
a	b_1	c_1
a	b_2	c_2

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

Examples

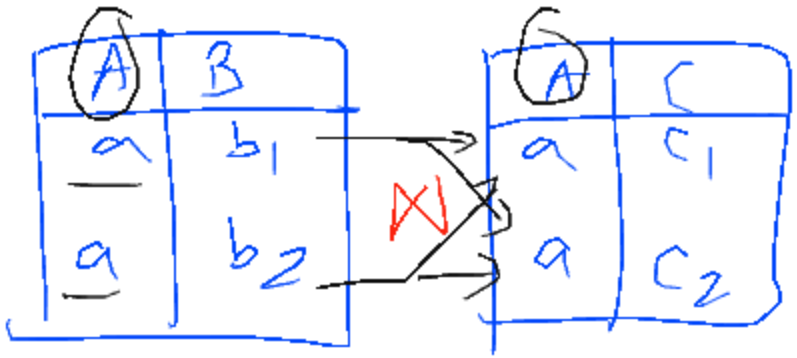
Lossy-Join

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

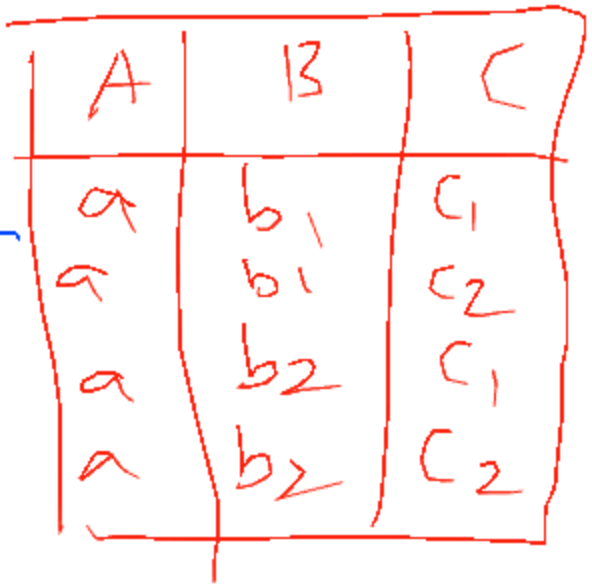
Original

A	B	C
a	b_1	c_1
a	b_2	c_2

Decomposition into {A,B} and {A,C}



Recombined



Examples

Lossy-Join

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

Original

A	B	C
a	b_1	c_1
a	b_2	c_2

Decomposition into {A,B} and {A,C}

$R_1 := \pi_{[A,B]}(R)$

A	B
a	b_1
a	b_2

$R_2 := \pi_{[A,C]}(R)$

A	C
a	c_1
a	c_2

Recombined

$R_1 \bowtie R_2$

A	B	C
a	b_1	c_1
a	b_1	c_2
a	b_2	c_1
a	b_2	c_2

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

Lemma

Lemma #1: Lossless-Join Binary Decomposition

A **binary decomposition** of R into R_1 and R_2 is ***lossless-join*** if $R = R_1 \cup R_2$ and

$(R_1 \cap R_2) \rightarrow R_1$ or $(R_1 \cap R_2) \rightarrow R_2$

Note

If $(R_1 \cap R_2)$ is the primary key of one of the two tables, then it can be a foreign key in the other table referencing the primary key.

Lemma

Lemma #1: Lossless-Join Binary Decomposition

A **binary decomposition** of R into R_1 and R_2 is ***lossless-join*** if $R = R_1 \cup R_2$ and

$(R_1 \cap R_2) \rightarrow R_1$ or $(R_1 \cap R_2) \rightarrow R_2$

Steps

1. Find the **intersection** $(R_1 \cap R_2)$. Let this be called $R_\cap$.
2. Compute the **attribute closure** $R_\cap^+$ with Σ.
3. Check if $R_1 \subseteq R_\cap^+$ or $R_2 \subseteq R_\cap^+$ *(or both)*.

Lemma

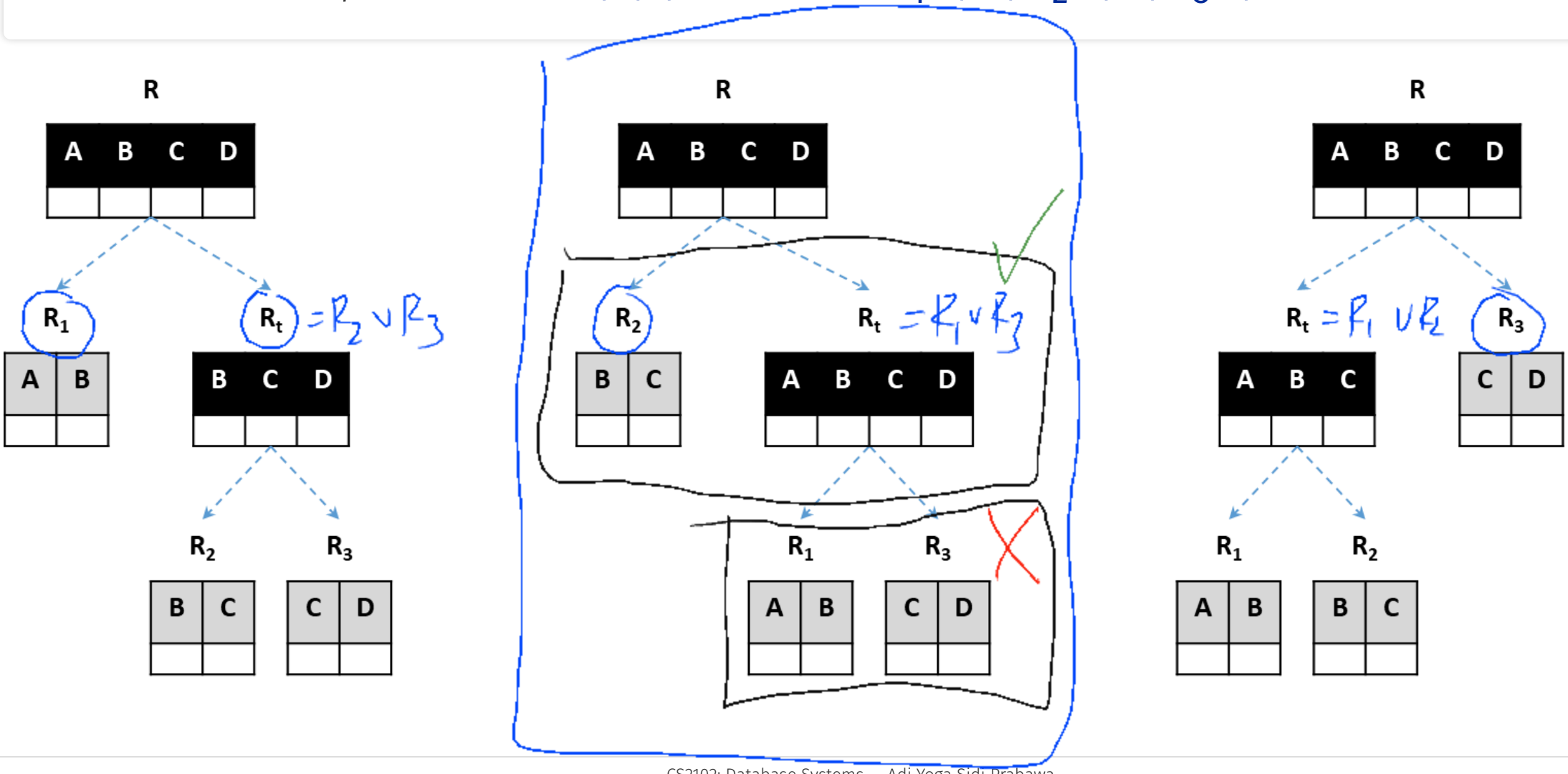
Lemma #2: Lossless-Join Decomposition

A **decomposition** of R into R_1, R_2, ..., R_n is ***lossless-join*** if there exists **at least one sequence** of binary lossless-join decomposition that generates that decomposition.

*There is an algorithm called **Chase Algorithm** that can simplify the check.

Visualization

Consider a decomposition of $R(A,B,C,D)$ into $\delta = \{R_1(A,B), R_2(B,C), R_3(C,D)\}$.



Example

Question

Consider $R(A,B,C,D,E,F)$ with $\Sigma = \{ \{A,B\} \rightarrow \{C\}, \{B,C\} \rightarrow \{A\}, \{D\} \rightarrow \{B\}, \{D\} \rightarrow \{E\} \}$. Consider the decomposition into $\delta = \{R_1(A,B,C), R_2(B,D,F), R_3(A,D,E)\}$. Is the decomposition a lossless-join decomposition?

Example

Question

Consider $R(A,B,C,D,E,F)$ with $\Sigma = \{ \{A,B\} \rightarrow \{C\}, \{B,C\} \rightarrow \{A\}, \{D\} \rightarrow \{B\}, \{D\} \rightarrow \{E\} \}$. Consider the decomposition into $\delta = \{R_1(A,B,C), R_2(B,D,F), R_3(A,D,E)\}$. Is the decomposition a lossless-join decomposition?

Consider $Rt = R_2 \cup R_3$

Lossy

- From R to $\{R_1, \{A,B,D,E,F\}\}$
  - $\{A,B\}^+ = \{A,B,C\} \supseteq R_1$
- From Rt to $\{R_2, R_3\}$
  - $\{D\}^+ = \{B,D,E\} \not\supseteq R_2$
  - $\{D\}^+ = \{B,D,E\} \not\supseteq R_3$

Consider $Rt = R_1 \cup R_3$

Lossy

- From R to $\{R_2, \{A,B,C,D,E\}\}$
  - $\{B,D\}^+ = \{B,D,E\} \not\supseteq R_2$
  - $\{B,D\}^+ = \{B,D,E\} \not\supseteq R_t$

Consider $Rt = R_1 \cup R_2$

Lossy

- From R to $\{R_3, \{A,B,C,D,F\}\}$
  - $\{A,D\}^+ = \{A,B,C,D,E\} \supseteq R_2$
- From Rt to $\{R_1, R_2\}$
  - $\{B\}^+ = \{B\} \not\supseteq R_1$
  - $\{B\}^+ = \{B\} \not\supseteq R_2$

Example

Question

Consider $R(A,B,C,D,E,F)$ with Σ = { {A,B} → {C}, {B,C} → {A}, {D} → {B}, {D} → {E} }. Consider the decomposition into $\delta = \{R_1(A,B,C), R_2(B,D,E), R_3(A,D,F)\}$. Is the decomposition a lossless-join decomposition?

Example

Question

Consider $R(A,B,C,D,E,F)$ with $\Sigma = \{ \{A,B\} \rightarrow \{C\}, \{B,C\} \rightarrow \{A\}, \{D\} \rightarrow \{B\}, \{D\} \rightarrow \{E\} \}$. Consider the decomposition into $\delta = \{R_1(A,B,C), R_2(B,D,E), R_3(A,D,F)\}$. Is the decomposition a lossless-join decomposition?

Consider $Rt = R_2 \cup R_3$

Lossless

- From R to $\{R_1, \{A,B,D,E,F\}\}$
  - $\{A,B\}^+ = \{A,B,C\} \supseteq R_1$
- From Rt to $\{R_2, R_3\}$
  - $\{D\}^+ = \{B,D,E\} \supseteq R_2$

Consider $Rt = R_1 \cup R_3$

Lossy

- From R to $\{R_2, \{A,B,C,D,F\}\}$
  - $\{B,D\}^+ = \{B,D,E\} \supseteq R_2$
- From Rt to $\{R_1, R_3\}$
  - $\{A\}^+ = \{A\} \not\supseteq R_1$
  - $\{A\}^+ = \{A\} \not\supseteq R_3$

Consider $Rt = R_1 \cup R_2$

Lossy

- From R to $\{R_3, \{A,B,C,D,E\}\}$
  - $\{A,D\}^+ = \{A,B,C,D,E\} \supseteq R_t$
- From Rt to $\{R_1, R_2\}$
  - $\{B\}^+ = \{B\} \not\supseteq R_1$
  - $\{B\}^+ = \{B\} \not\supseteq R_2$

Definition

Projected Functional Dependencies

Consider a relation R with a set of functional dependencies Σ. A set Σ' of ***projected functional dependencies*** on R' from R with Σ, where $R' \subseteq R$, is the **set of functional dependencies** such that

for all $X \rightarrow Y \in \Sigma^+$, we have $X \subseteq R'$ and $Y \subseteq R'$

Given R', we may also denote the projection of Σ on R' by $\Sigma|_{R'}$.

Definition

Projected Functional Dependencies

Consider a relation R with a set of functional dependencies Σ. A set Σ' of ***projected functional dependencies*** on R' from R with Σ, where $R' \subseteq R$, is the **set of functional dependencies** such that

for all $X \rightarrow Y \in \Sigma^+$, we have $X \subseteq R'$ and $Y \subseteq R'$

Given R', we may also denote the projection of Σ on R' by $\Sigma|_{R'}$.

Note

In other words,

$X \rightarrow Y$ is ***logically entailed*** by Σ X and Y contains ***only attributes*** in R'

Although it should contain **all** logically entailed functional dependencies, we often want to find any **equivalent** set.

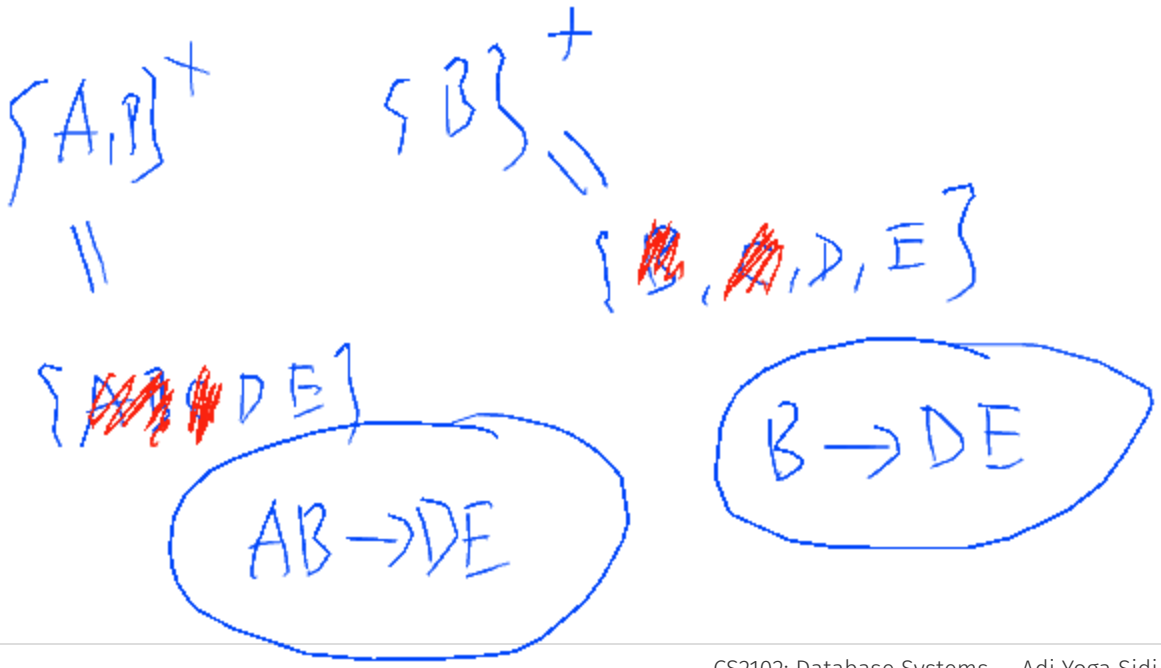
Example

Question

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

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

What is **a** set of projected functional dependencies Σ' on $R' = \{A, B, D, E\}$ from R with Σ?
This is denoted as $\Sigma|_{R'}$.



Example

Question

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

What is **a** set of projected functional dependencies Σ' on $R' = \{A, B, D, E\}$ from R with Σ?
This is denoted as $\Sigma|_{R'}$.

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

Example

Question

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

What is **a** set of projected functional dependencies Σ' on $R' = \{A, B, D, E\}$ from R with Σ?
This is denoted as $\Sigma|_{R'}$.

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

Example

Question

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

What is **a** set of projected functional dependencies Σ' on $R' = \{A, B, D, E\}$ from R with Σ?
This is denoted as $\Sigma|_{R'}$.

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

Example

Question

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

What is **a** set of projected functional dependencies Σ' on $R' = \{A, B, D, E\}$ from R with Σ?
This is denoted as $\Sigma|_{R'}$.

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

Issue

Do NOT forget that we need to look at all functional dependencies in Σ^+.

Steps

Step 1: List All Subset of Attributes

{A}	*{B}*	*{D}*	*{E}*
{A,B}	*{A,D}*	*{A,E}*	
{B,D}	*{B,E}*	*{D,E}*	
{A,B,D}	*{A,B,E}*	*{A,D,E}*	*{B,D,E}*
{A,B,D,E}			

Steps

Step 2: Compute the Attribute Closures

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

Steps

Step 3: Intersect Right Hand Side with R'

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

Steps

Step 4: Subtract Left Hand Side from Right Hand Side

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

Steps

Step 5: Construct Functional Dependencies

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

Note

We can ignore the trivial functional dependencies as we want to find a simpler **equivalent** set.

Projection

Explicit Step

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

=

Previous Answer

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

=

Minimal Cover of Projection

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

Note

Any one of these is a good solution for the $\Sigma|_{\{A,B,D,E\}}$.

Definition

Dependency Preserving Decomposition

A decomposition of R with Σ into $\delta = \{R_1, \ldots, R_n\}$ with the respective sets of functional dependencies $\Sigma|_{R_1}, \ldots, \Sigma|_{R_n}$ is ***dependency preserving*** if and only if

$$\Sigma^+ = (\Sigma|_{R_1} \cup \ldots \cup \Sigma|_{R_n})^+$$

Note

We simply need to check that all functional dependencies $\sigma \in \Sigma$ can be ***logically entailed*** by

$$(\Sigma|_{R_1} \cup \ldots \cup \Sigma|_{R_n})$$

By construction, all functional dependencies $\sigma' \in (\Sigma|_{R_1} \cup \ldots \cup \Sigma|_{R_n})$ can be logically entailed by Σ.

Examples

Example #1

Let $R = \{A, B, C\}$ with $\Sigma = \{ \{C\} \rightarrow \{B\} \}$ *(the candidate key is {A, C})*. Is the decomposition a ***dependency preserving*** lossless-join decomposition?

$R_1 = \{A, C\}$ $\quad \Sigma|_{R_1} = \emptyset$

$R_2 = \{B, C\}$ $\quad \Sigma|_{R_2} = \{ \{C\} \rightarrow \{B\} \}$

$$\Sigma_U = (\Sigma|_{R_1} \cup \Sigma|_{R_2}) = \{ \{C\} \rightarrow \{B\} \}$$

Examples

Example #1

Let $R = \{A, B, C\}$ with $\Sigma = \{ \{C\} \rightarrow \{B\} \}$ *(the candidate key is {A, C})*. Is the decomposition a ***dependency preserving*** lossless-join decomposition?

$R_1 = \{A, C\}$ projected set of functional dependencies $\Sigma|_{R_1} = \varnothing$

$R_2 = \{B, C\}$ projected set of functional dependencies $\Sigma|_{R_2} = \{ \{C\} \rightarrow \{B\} \}$

Note

$\Sigma|_{R_1} = \varnothing$ does not mean that there is no functional dependencies in the projection. Instead, it means that all functional dependencies are all trivial.

Examples

Example #1

Let $R = \{A, B, C\}$ with $\Sigma = \{ \{C\} \rightarrow \{B\} \}$ *(the candidate key is {A, C})*. Is the decomposition a ***dependency preserving*** lossless-join decomposition?

$R_1 = \{A, C\}$ projected set of functional dependencies $\Sigma|_{R_1} = \varnothing$

$R_2 = \{B, C\}$ projected set of functional dependencies $\Sigma|_{R_2} = \{ \{C\} \rightarrow \{B\} \}$

Check

1. $(\Sigma|_{R_1} \cup \Sigma|_{R_2}) = \{ \{C\} \rightarrow \{B\} \}$
2. Check each fd in Σ
   - $\{C\} \rightarrow \{B\}$: trivial

3. Since $(\Sigma|_{R_1} \cup \Sigma|_{R_2}) \equiv \Sigma$, this is a dependency preserving decomposition.

Examples

Example #2

Let $R = \{A, B, C\}$ with $\Sigma = \{ \{C\} \rightarrow \{B\}, \{A,B\} \rightarrow \{C\} \}$ *(the candidate key are {A, B} and {A, C})*. Is the decomposition a ***dependency preserving*** lossless-join decomposition?

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

$R_1 = \{A, C\}$ $\quad \Sigma|_{R_1} = \emptyset$

$R_2 = \{B, C\}$ $\quad \Sigma|_{R_2} = \{ \{C\} \rightarrow \{B\} \}$

$\Sigma' = \{ \{C\} \rightarrow \{B\} \} \neq \Sigma$

Examples

Example #2

Let $R = \{A, B, C\}$ with $\Sigma = \{ \{C\} \rightarrow \{B\}, \{A,B\} \rightarrow \{C\} \}$ *(the candidate key are {A, B} and {A, C})*. Is the decomposition a ***dependency preserving*** lossless-join decomposition?

$R_1 = \{A, C\}$ projected set of functional dependencies $\Sigma|_{R_1} = \varnothing$

$R_2 = \{B, C\}$ projected set of functional dependencies $\Sigma|_{R_2} = \{ \{C\} \rightarrow \{B\} \}$

Examples

Example #2

Let $R = \{A, B, C\}$ with $\Sigma = \{ \{C\} \rightarrow \{B\}, \{A,B\} \rightarrow \{C\} \}$ *(the candidate key are {A, B} and {A, C})*. Is the decomposition a ***dependency preserving*** lossless-join decomposition?

$R_1 = \{A, C\}$	projected set of functional dependencies $\Sigma\vert_{R_1} = \varnothing$
$R_2 = \{B, C\}$	projected set of functional dependencies $\Sigma\vert_{R_2} = \{ \{C\} \rightarrow \{B\} \}$

Check

1. $(\Sigma\vert_{R_1} \cup \Sigma\vert_{R_2}) = \{ \{C\} \rightarrow \{B\} \}$
2. Check each fd in Σ
   - $\{C\} \rightarrow \{B\}$: trivial
   - $\{A,B\} \rightarrow \{C\}$: since $\{A,B\}^+$ w.r.t. $(\Sigma\vert_{R_1} \cup \Sigma\vert_{R_2})$ is {A,B}, it is **not entailed**.
3. Hence $(\Sigma\vert_{R_1} \cup \Sigma\vert_{R_2}) \not\equiv \Sigma$, and $\{A,B\} \rightarrow \{C\}$ is not preserved.

Examples

Example #3

Let $R = \{A, B, C\}$ with $\Sigma = \{ \{A\} \rightarrow \{B\}, \{B\} \rightarrow \{C\}, \{A\} \rightarrow \{C\} \}$ *(the candidate key is {A})*. Is the decomposition a ***dependency preserving*** lossless-join decomposition?

$R_1 = \{A, B\}$

$R_2 = \{B, C\}$

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

$\Sigma|_{R_1} = \{\{A\} \rightarrow \{B\}\}$

$\Sigma|_{R_2} = \{\{B\} \rightarrow \{C\}\}$

$\Sigma_U = \{\{A\} \rightarrow \{B\}, \{B\} \rightarrow \{C\}\} \equiv \Sigma$

Examples

Example #3

Let $R = \{A, B, C\}$ with $\Sigma = \{ \{A\} \rightarrow \{B\}, \{B\} \rightarrow \{C\}, \{A\} \rightarrow \{C\} \}$ *(the candidate key is {A})*. Is the decomposition a ***dependency preserving*** lossless-join decomposition?

$R_1 = \{A, B\}$ projected set of functional dependencies $\Sigma|_{R_1} = \{ \{A\} \rightarrow \{B\} \}$

$R_2 = \{B, C\}$ projected set of functional dependencies $\Sigma|_{R_2} = \{ \{B\} \rightarrow \{C\} \}$

Check

1. $(\Sigma|_{R_1} \cup \Sigma|_{R_2}) = \{ \{A\} \rightarrow \{B\}, \{B\} \rightarrow \{C\} \}$
2. Check each fd in Σ
   - $\{A\} \rightarrow \{B\}$: trivial $\{B\} \rightarrow \{C\}$: trivial
   - $\{A\} \rightarrow \{C\}$: since $\{A\}^+$ w.r.t. $(\Sigma|_{R_1} \cup \Sigma|_{R_2})$ is {A,B,C}, it is **entailed**.
3. Hence $(\Sigma|_{R_1} \cup \Sigma|_{R_2}) \equiv \Sigma$, and $\{A\} \rightarrow \{C\}$ is preserved.

Examples

Example #3

Let $R = \{A, B, C\}$ with $\Sigma = \{ \{A\} \rightarrow \{B\}, \{B\} \rightarrow \{C\}, \{A\} \rightarrow \{C\} \}$ *(the candidate key is {A})*. Is the decomposition a ***dependency preserving*** lossless-join decomposition?

$R_1 = \{A, B\}$ projected set of functional dependencies $\Sigma|_{R_1} = \{ \{A\} \rightarrow \{B\} \}$

$R_2 = \{B, C\}$ projected set of functional dependencies $\Sigma|_{R_2} = \{ \{B\} \rightarrow \{C\} \}$

Try It Out

$A \rightarrow B$ $B \rightarrow C$ $A \not\rightarrow C$

A	B

B	C

A	B	C

Basic

Preliminary

When a relation is not in BCNF*, we can pick one of the functional dependencies violating the BCNF definition and use it to **decompose the relation** into two relations. We **continue decomposing** until every fragment is in BCNF.

*The same algorithm work for other normal forms.

Basic

Preliminary

When a relation is not in BCNF*, we can pick one of the functional dependencies violating the BCNF definition and use it to **decompose the relation** into two relations. We **continue decomposing** until every fragment is in BCNF.

What We Want

We want the decomposition algorithm to at least guarantee a **lossless-join** decomposition.

*The same algorithm work for other normal forms.

Basic

Preliminary

When a relation is not in BCNF*, we can pick one of the functional dependencies violating the BCNF definition and use it to **decompose the relation** into two relations. We **continue decomposing** until every fragment is in BCNF.

What We Want

We want the decomposition algorithm to at least guarantee a **lossless-join** decomposition.

Issues

However, such approach may not be **dependency preserving**.

*The same algorithm work for other normal forms.

BCNF Decomposition

Algorithm #4: BCNF Decomposition

Let $X \rightarrow Y$ be a functional dependency in Σ that violates the BCNF definition *(i.e., non trivial and X is not a superkey)*. We use $X \rightarrow Y$ to ***decompose*** R into two relations R_1 and R_2 in the following way:

$$R_1 = X^+$$
$$R_2 = (R - X^+) \cup X$$

We must now check whether R_1 and R_2 with the ***respective sets of projected functional dependencies*** $\Sigma|_{R_1}$ and $\Sigma|_{R_2}$ are in BCNF.

If they are not, we ***recursively continue the decomposition***.

BCNF Decomposition

Algorithm #4: BCNF Decomposition

input R, Σ
output δ

begin
  find one $X \rightarrow Y$ such that $X \subset X^+ \neq R$.
    if none, then return $\delta = \{R\}$ *(i.e., no decomposition)*
  $R_1 := X^+$
  $R_2 := (R - X^+) \cup X$
  return $\delta =$ Algo4$(R_1, \Sigma|_{R_1}) \cup$ Algo4$(R_2, \Sigma|_{R_2})$

Example

$\{C\} \not\subseteq \{B\}$ $\{B\}^+ = \{B, C, D, E\}$

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

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

Is R with Σ in **BCNF**?

Example

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

Is R with Σ in **BCNF**?

NO

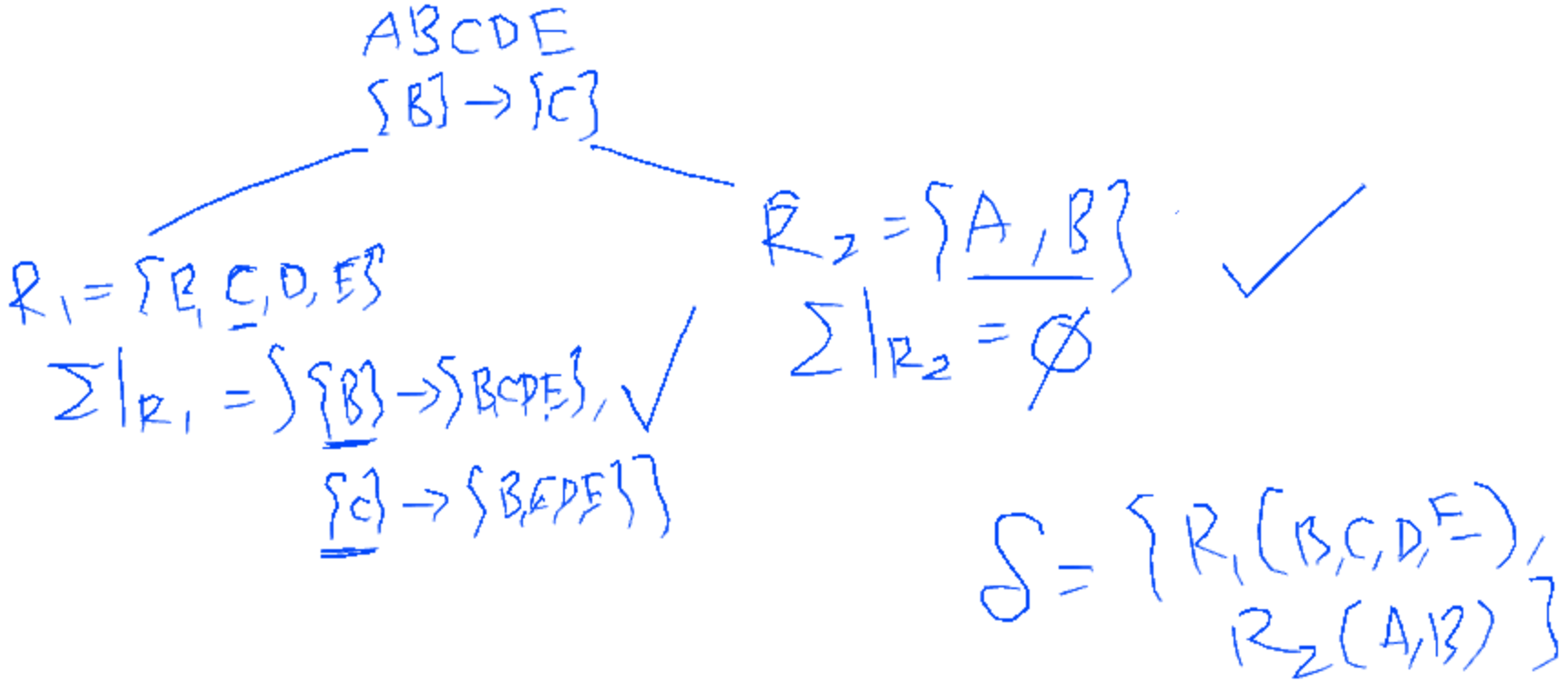
There are two **candidate keys** $\{A, B\}$ and $\{A, C\}$.
Consider $\{C\} \rightarrow \{D\}$. Since $\{D\} \not\subseteq \{C\}$, it is **non-trivial**. Additionally, $\{C\}$ is **not a superkey**.

Example

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

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

Decompose R with Σ into a lossless decomposition in **BCNF**.



Example

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

Decompose R with Σ into a lossless decomposition in **BCNF**.

Steps

Using $\{C\} \rightarrow \{D\}$ from before *(non-trivial and {C} is not a superkey)*. Compute $\{C\}^+ = \{B, C, D, E\}$.
Decompose R into two fragments and the projected functional dependencies:

$R_1 = \{C\}^+ = \{B, C, D, E\}$
$\Sigma|_{R_1} = \{ \{C\} \rightarrow \{B, D, E\}, \{B\} \rightarrow \{C\} \}$

Is R_1 with $\Sigma|_{R_1}$ in BCNF?

$R_2 = (R - \{C\}^+) \cup \{C\} = \{A, C\}$
$\Sigma|_{R_2} = \varnothing$

Is R_2 with $\Sigma|_{R_2}$ in BCNF?

Example

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

Decompose R with Σ into a lossless decomposition in **BCNF**.

Lossless?

Is the decomposition lossless?

Example

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

Decompose R with Σ into a lossless decomposition in **BCNF**.

Lossless?

Is the decomposition lossless?

Yes, the decomposition is guaranteed to be *lossless* *(by Theorem 7 later)*

Example

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

Decompose R with Σ into a lossless decomposition in **BCNF**.

Dependency Preserving?

Have we lost any functional dependency?

$(\Sigma|_{R_1} \cup \Sigma|_{R_2}) = \{ \{C\} \rightarrow \{B, D, E\}, \{B\} \rightarrow \{C\} \}$
$\Sigma = \{ \{A,B\} \rightarrow \{C,D,E\}, \{A,C\} \rightarrow \{B,D,E\}, \{B\} \rightarrow \{C\}, \{C\} \rightarrow \{B\}, \{C\} \rightarrow \{D\}, \{B\} \rightarrow \{E\}, \{C\} \rightarrow \{E\} \}$

Example

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

Decompose R with Σ into a lossless decomposition in **BCNF**.

Dependency Preserving?

Have we lost any functional dependency?

$(\Sigma|_{R_1} \cup \Sigma|_{R_2}) = \{ \{C\} \rightarrow \{B, D, E\}, \{B\} \rightarrow \{C\} \}$
$\Sigma = \{ \{A,B\} \rightarrow \{C,D,E\}, \{A,C\} \rightarrow \{B,D,E\}, \{B\} \rightarrow \{C\}, \{C\} \rightarrow \{B\}, \{C\} \rightarrow \{D\}, \{B\} \rightarrow \{E\}, \{C\} \rightarrow \{E\} \}$

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

Example

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

Decompose R with Σ into a lossless decomposition in **BCNF**.

Dependency Preserving?

Have we lost any functional dependency?

$(\Sigma|_{R_1} \cup \Sigma|_{R_2}) = \{ \{C\} \rightarrow \{B, D, E\}, \{B\} \rightarrow \{C\} \}$
$\Sigma = \{ \{A,B\} \rightarrow \{C,D,E\}, \{A,C\} \rightarrow \{B,D,E\}, \{B\} \rightarrow \{C\}, \{C\} \rightarrow \{B\}, \{C\} \rightarrow \{D\}, \{B\} \rightarrow \{E\}, \{C\} \rightarrow \{E\} \}$

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

Example

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

Decompose R with Σ into a lossless decomposition in **BCNF**.

Dependency Preserving?

Have we lost any functional dependency?

$(\Sigma|_{R_1} \cup \Sigma|_{R_2}) = \{ \{C\} \rightarrow \{B, D, E\}, \{B\} \rightarrow \{C\} \}$
$\Sigma = \{ \{A,B\} \rightarrow \{C,D,E\}, \{A,C\} \rightarrow \{B,D,E\}, \{B\} \rightarrow \{C\}, \{C\} \rightarrow \{B\}, \{C\} \rightarrow \{D\}, \{B\} \rightarrow \{E\}, \{C\} \rightarrow \{E\} \}$

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

Example

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

Decompose R with Σ into a lossless decomposition in **BCNF**.

Dependency Preserving?

Have we lost any functional dependency?

$(\Sigma|_{R_1} \cup \Sigma|_{R_2}) = \{ \{C\} \rightarrow \{B, D, E\}, \{B\} \rightarrow \{C\} \}$
$\Sigma = \{ \{A,B\} \rightarrow \{C,D,E\}, \{A,C\} \rightarrow \{B,D,E\}, \{B\} \rightarrow \{C\}, \{C\} \rightarrow \{B\}, \{C\} \rightarrow \{D\}, \{B\} \rightarrow \{E\}, \{C\} \rightarrow \{E\} \}$

No, the decomposition is ***dependency preserving***.

Example

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

Can we choose another dependency to decompose and reach a different result?

Example

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

Can we choose another dependency to decompose and reach a different result?

YES!

Try it out.

Theorems

Theorem #7: Lossless Join Decomposition ➥

The BCNF decomposition algorithm is a lossless-join decomposition.

Theorem #8: Termination ➥

The BCNF decomposition algorithm will terminate.

Our Case ❶❷❸❹

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

R(A, B, C, D, E, F)

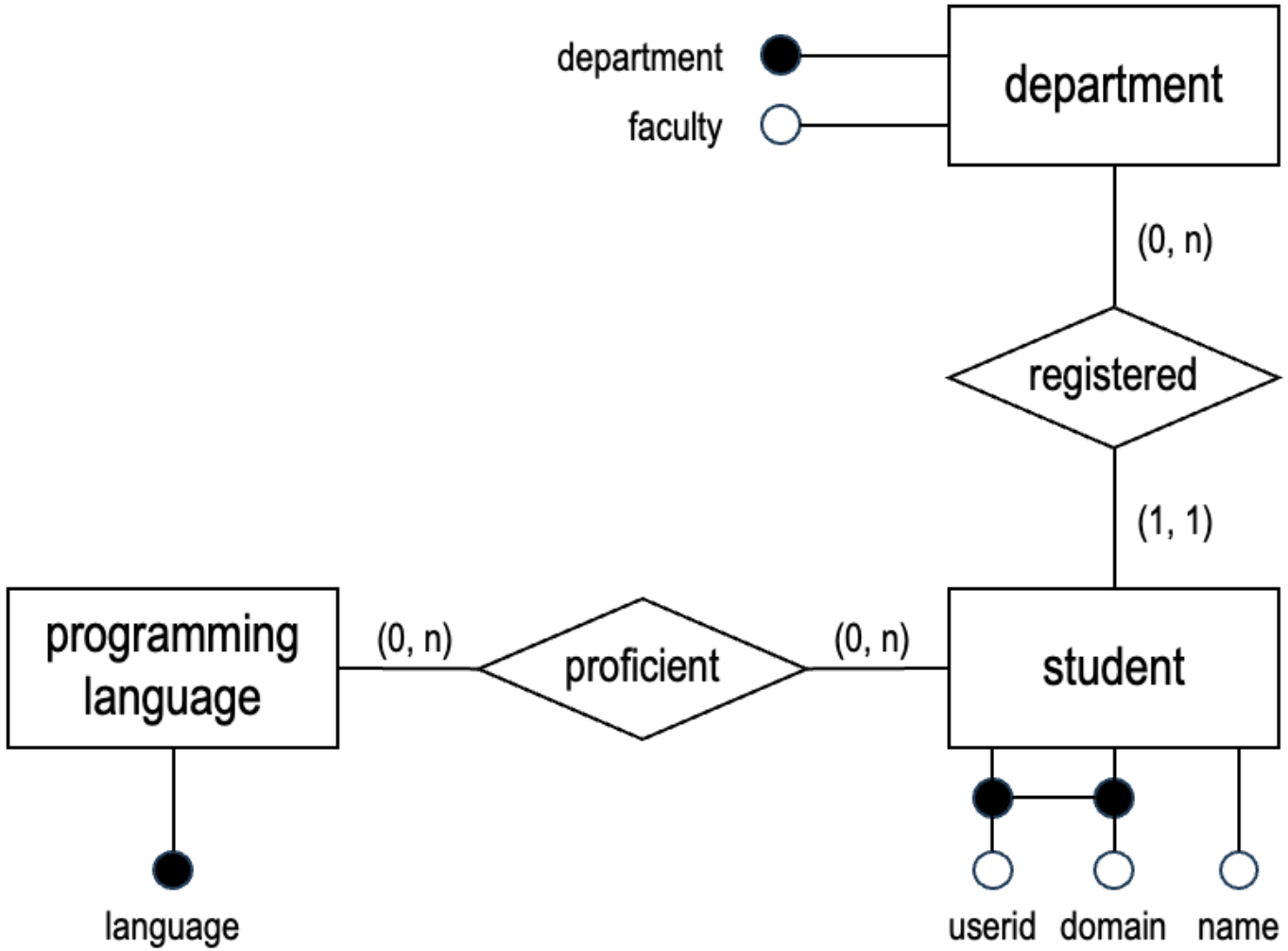
$$\{D\} \rightarrow \{E\}$$
$$\{B,C\} \rightarrow \{A,D\}$$
$$\{B,C,F\} \rightarrow \{A,D,E\}$$

Mapping

A: name
B: userid
C: domain
D: department
E: faculty
F: language

Our Case ❶❷❸❹

ERD



R(A, B, C, D, E, F)

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

Mapping

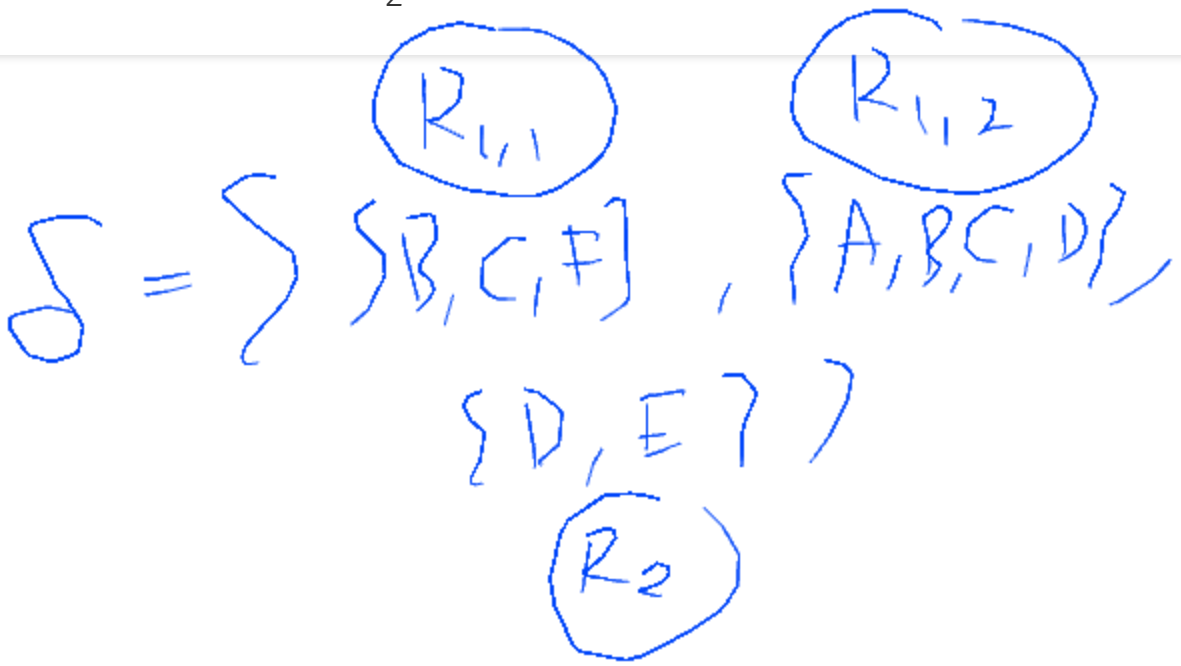
A: name
B: userid
C: domain
D: department
E: faculty
F: language

Our Case ❶❷❸❹

Decomposition

We may get the following lossless-join dependency preserving BCNF decomposition.

- $R_{1,1}$ = {<u>B</u>, <u>C</u>, <u>F</u>} $\quad \Sigma|_{R_{1,1}} = \varnothing$
- $R_{1,2}$ = {A, <u>B</u>, <u>C</u>, D} $\quad \Sigma|_{R_{1,2}} = \{ \{D\} \rightarrow \{E\} \} = \{ \{B,C\} \rightarrow \{A,D\} \}$
- R_2 = {<u>D</u>, E} $\quad \Sigma|_{R_2} = \{ \{D\} \rightarrow \{E\} \}$



R(A, B, C, D, E, F)

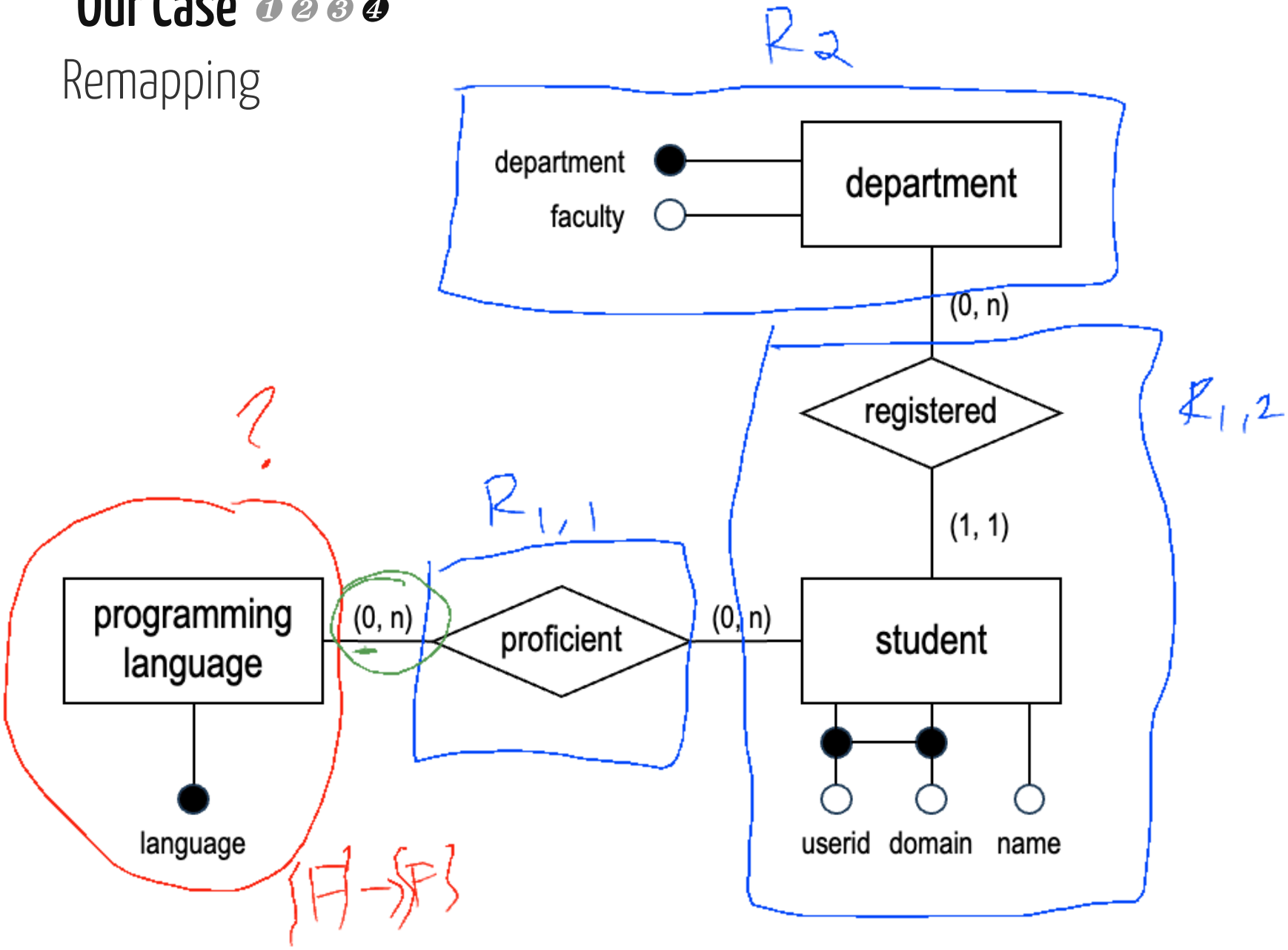
{D} → {E}
{B,C} → {A,D}
{B,C,F} → {A,D,E}

Mapping

A: name
B: userid
C: domain
D: department
E: faculty
F: language

Our Case ❶❷❸❹

Remapping



Fragments

- $R_{1,1} = \{\underline{B}, \underline{C}, \underline{F}\}$
- $R_{1,2} = \{A, \underline{B}, \underline{C}, D\}$
- $R_2 = \{\underline{D}, E\}$

Mapping

A: name
B: userid
C: domain
D: department
E: faculty
F: language

```
postgres=# exit
```

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

Proof Sketch

Only for Reading ; Not Tested

Thm #7 | Thm #8

Proof ↩

The decomposition produces two fragments: $R_1 = X^+$ and $R_2 = (R - X^+) \cup X$.

- By definition, $R_1 \cap R_2 = X$

  (we remove X^+ from R_2 and we add back X)
- By definition, $X \rightarrow X^+$

  (this is the definition of X^+)
- Hence, this is a lossless-join binary decomposition

  (using Lemma #1)
- Further decomposition will yield lossless-join decomposition

  (using Lemma #2)

Proof ↩

The decomposition produces two fragments: $R_1 = X^+$ and $R_2 = (R - X^+) \cup X$.

Recap that we decompose based on a violation $X \rightarrow X^+$.

- In R_1, the violation $X \rightarrow X^+$ is no longer a violation as X is the **superkey**.
  (by construction)
- In R_2, the violation $X \rightarrow X^+$ is no longer a violation as X it becomes $X \rightarrow X$, hence trivial.
  (by construction)

For completeness, we should show that we do not add a new violation and there is only a finite number of possible violations. Note that other violations may be duplicated in both R_1 and R_2, so a total *(global)* violation may increase. But locally, R_1 has fewer violations than R and R_2 has fewer violations than R.

```
postgres=# exit
```

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