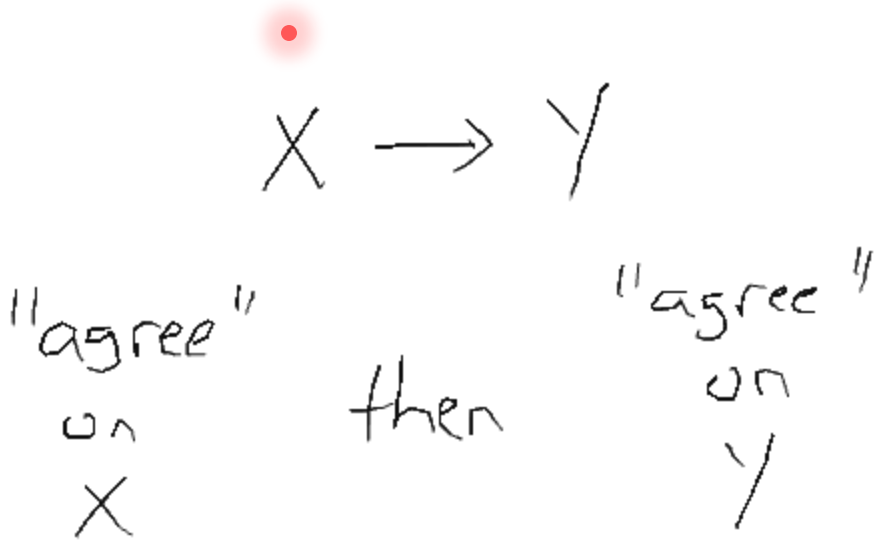


Database

Theory

Anomalies and Boyce-Codd Normal Forms

Case Study

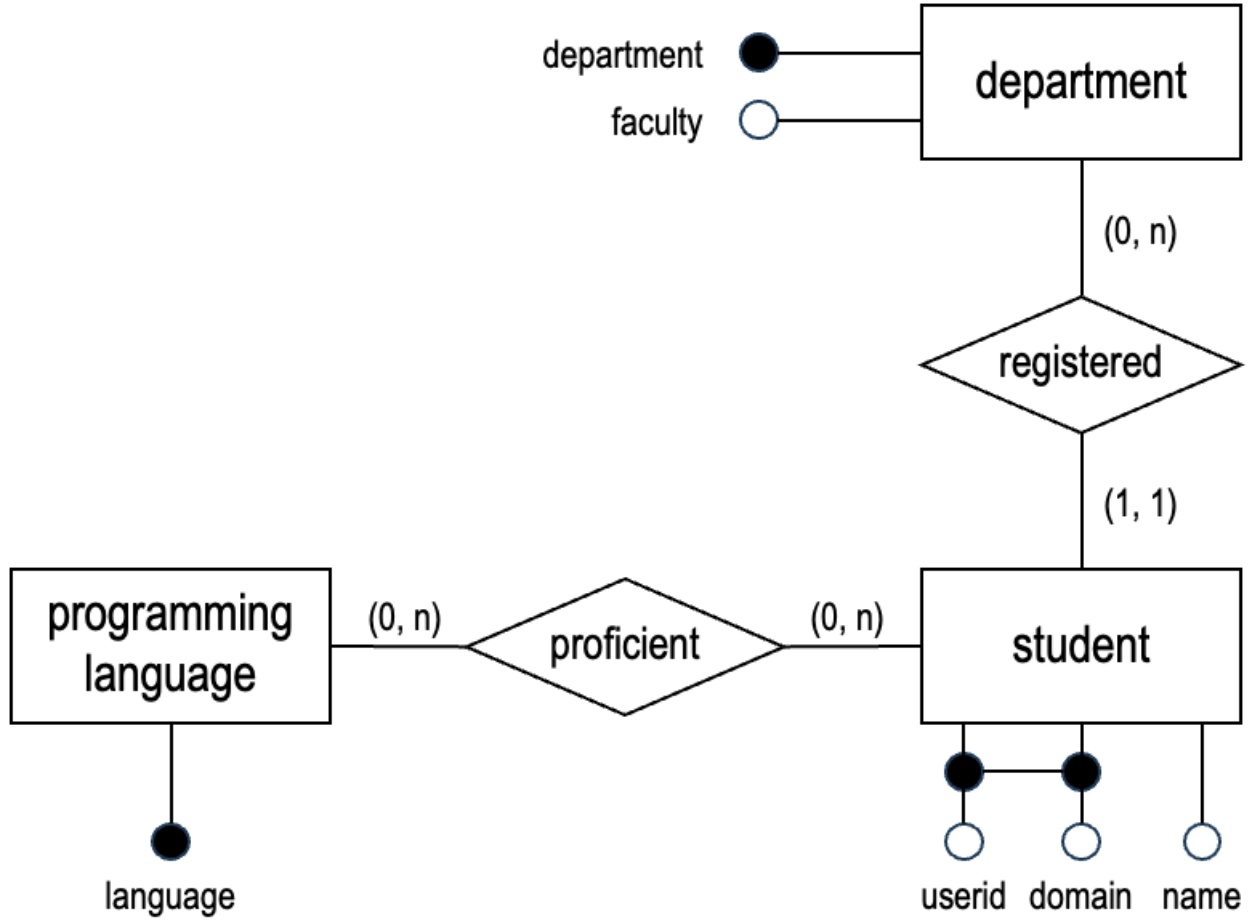
- Case
  - *SUT*
  - *Schema*
  - *Alternative*
- Anomalies
- Normal Forms

Case

SUT

The Case

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



Case Study

- Case
  - *SUT*
  - *Schema*
  - *Alternative*
- Anomalies
- Normal Forms

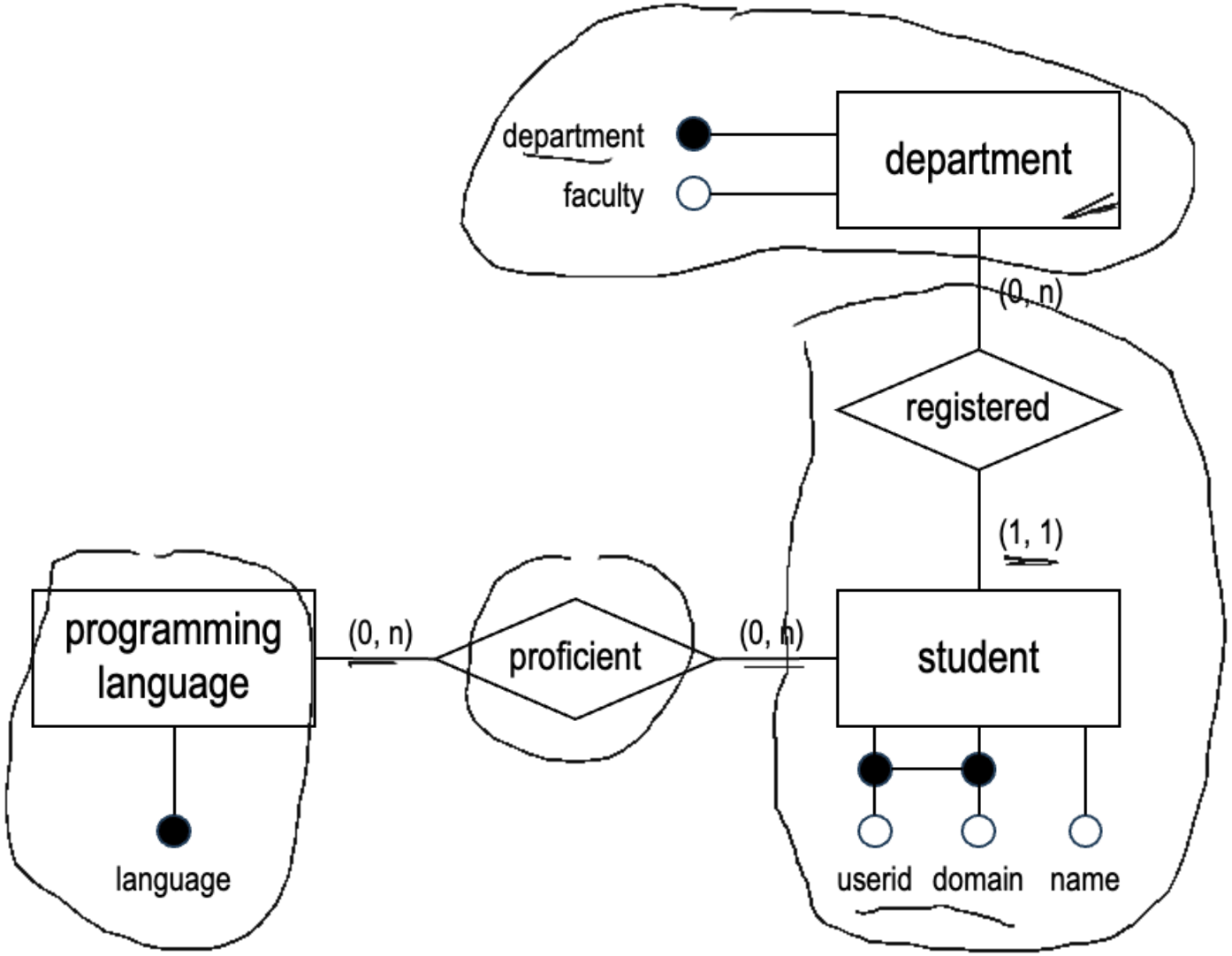
Case

Schema

The Schema

The schema should consist of 4 tables with `student` and `registered` merged.

department(<u>department</u>, *faculty*)
student(<u>userid, domain</u>,
name, department)
language(<u>language</u>)
proficient(<u>language</u>,
<u>userid, domain</u>)



Case Study

- Case
  - SUT
  - Schema
  - Alternative
- Anomalies
- Normal Forms

Case

Alternative

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

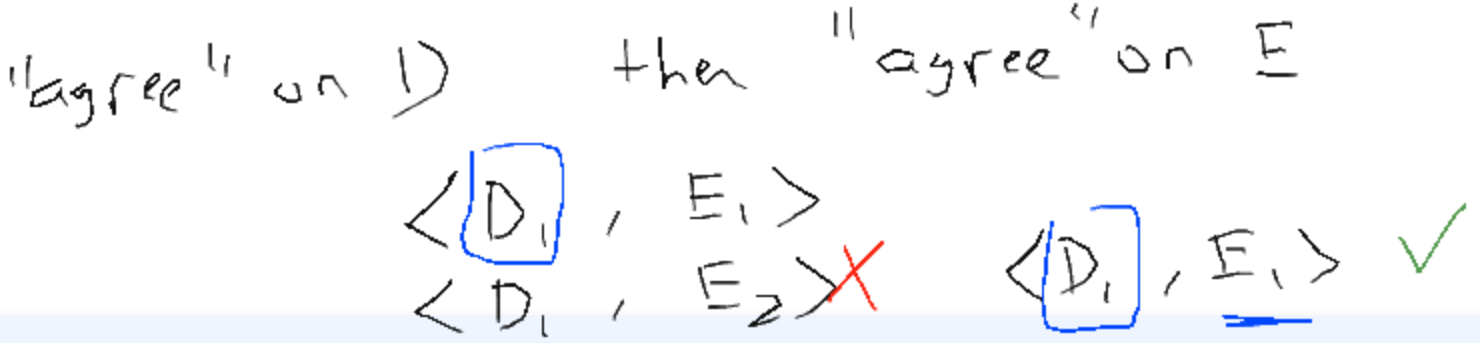
A: name
B: userid
C: domain
D: department
E: faculty
F: language

Case Study

- Case
  - SUT
  - Schema
  - Alternative
- Anomalies
- Normal Forms

Case

Alternative



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

Note

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

Question

Can we set $\{D\}$ to be the *primary key*?

Case Study

- Case
  - SUT
  - Schema
  - Alternative
- Anomalies
- Normal Forms

Case

Alternative

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

Note

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

Question

Can we set {B,C} to be the *primary key*?

Case Study

- Case
  - SUT
  - Schema
  - Alternative
- Anomalies
- Normal Forms

Case

Alternative

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

Note

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

Question

Can we set {B,C,F} to be the *primary key*?

Case Study

Case
- Anomalies
  - *Redundant*
  - *Update*
  - *Delete*
  - *Insert*
  - *Recombination*
  - *To Do*

Normal Forms

Anomalies

Redundant

$\pi_{[D,E]}(\text{Proficiency})$

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

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.

Case Study

Case
Anomalies
Redundant
Update
Delete
Insert
Recombination
To Do
Normal Forms

Anomalies

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 $\{D\} \rightarrow \{E\}$. Splitting table allows us to update only on the `Department` table.

Case Study

Case
Anomalies
Redundant
Update
Delete
Insert
Recombination
To Do
Normal Forms

Anomalies

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 $\{D\} \rightarrow \{E\}$. With split table, even if Ami Mokhtar is deleted, the department and faculty is still recorded in the `Department` table.

Case Study

Case
Anomalies
Redundant
Update
Delete
Insert
Recombination
To Do
Normal Forms

Anomalies

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.

Case Study

Case
Anomalies
Redundant
Update
Delete
Insert
Recombination
To Do
Normal Forms

Anomalies

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.

Case Study

Case
Anomalies
Redundant
Update
Delete
Insert
Recombination
To Do
Normal Forms

Anomalies

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

```
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;
```

Case Study

Case
Anomalies
Redundant
Update
Delete
Insert
Recombination
To Do
Normal Forms

Anomalies

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;
```

Case Study

Case
Anomalies
Redundant
Update
Delete
Insert
Recombination
To Do
Normal Forms

Anomalies

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

$\{B, C\} \rightarrow \{D, E\}$ ※

To Do

Verify that we have similar anomalies with the other non-trivial functional dependencies, even after we split the table as above.

Case Study

Case
Anomalies
› Normal Forms

Normal Forms

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`, and `FOREIGN KEY`) and thus protects against data anomalies.

CHECK

Purpose of Normalization

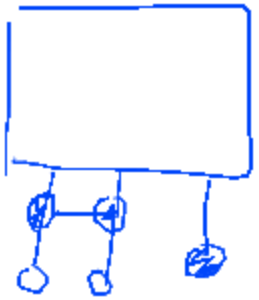
The purpose of the ***normalization*** is to **transform** *(decompose)* a poor design into a design that enforces functional dependencies by means of the main SQL constraints.

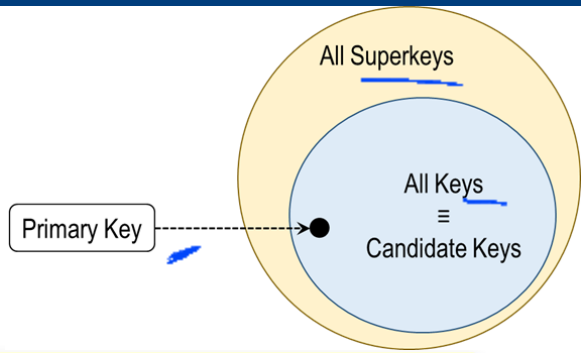
Boyce-Codd Normal Form

- Preliminary
  - *Keys*
  - *Basic*
- BCNF

Preliminary

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 the candidate key is {userid, domain, language}.

$BCF^+ = ABCDEF$

1. All keys
2. Minimal Cover

Boyce-Codd Normal Form

Preliminary
Keys
Basic
BCNF

Preliminary

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)*.

{userid, domain} → {name, department}

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

{department} → {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.

Boyce-Codd Normal Form

Preliminary
BCNF
Definition
Theorem
Intuition
Example
Decomposition

BCNF

Definition

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)

Boyce-Codd Normal Form

Preliminary
BCNF
Definition
Theorem
Intuition
Example
Decomposition

BCNF

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*

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

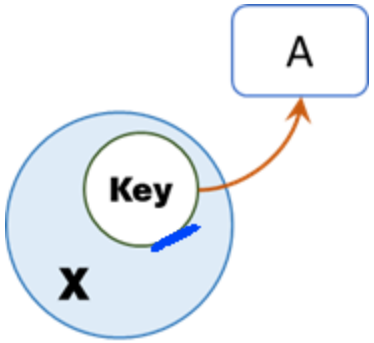
Boyce-Codd Normal Form

Preliminary
BCNF
Definition
Theorem
Intuition
Example
Decomposition

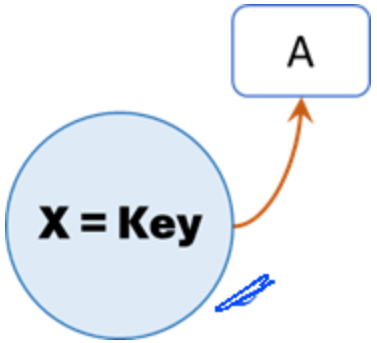
BCNF

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**)*

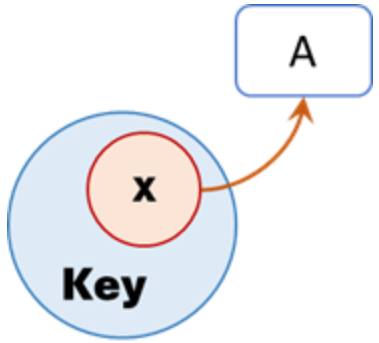
Boyce-Codd Normal Form

Preliminary
BCNF
Definition
Theorem
Intuition
Example
Decomposition

BCNF

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.

Boyce-Codd Normal Form

Preliminary
BCNF
Definition
Theorem
Intuition
Example
Decomposition

BCNF

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

Candidate Keys

{ userid, domain, language }

{B, C, F}

Violation

Consider {department} → {faculty}. It is *non-trivial* and its left hand side is *not a superkey*. It does violate the two conditions of the theorem. We have found a culprit. The table is not in BCNF.

Boyce-Codd Normal Form

Preliminary
BCNF
Definition
Theorem
Intuition
Example
Decomposition

BCNF

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.

7:23

Break

Decomposition

- Decomposition
  - *Definition*
- Lossless-Join
- Projection
- Preservation
- Algorithm

Decomposition

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.

Decomposition

Decomposition
Lossless-Join
Definition
Examples
Lemma
Visualization
Projection
Preservation
Algorithm

Lossless-Join

Definition

Lossless-Join Decomposition

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

NATURAL JOIN

```
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;
```

not lossless

Note

Lossless-join property may also be called **non-additive join**. This is because lossy join *adds rows*.

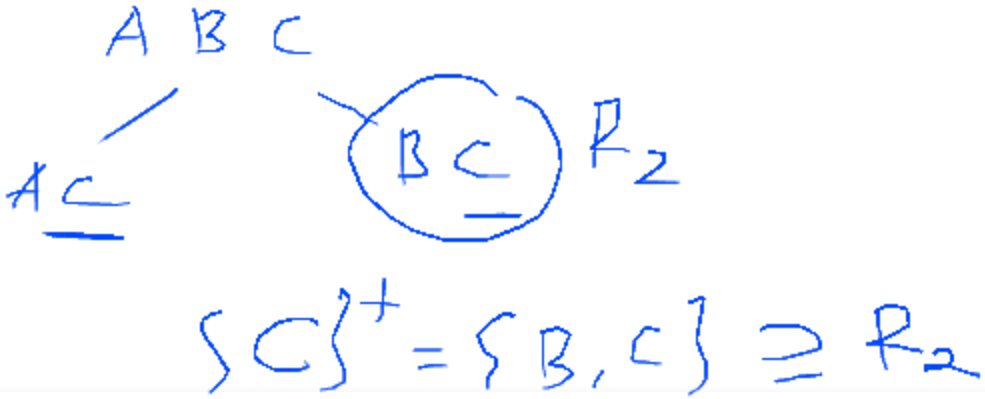
Decomposition

Decomposition
Lossless-Join
Definition
Examples
Lemma
Visualization
Projection
Preservation
Algorithm

Lossless-Join

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

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

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

Decomposition

Decomposition
Lossless-Join
Definition
Examples
Lemma
Visualization
Projection
Preservation
Algorithm

Lossless-Join

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

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

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

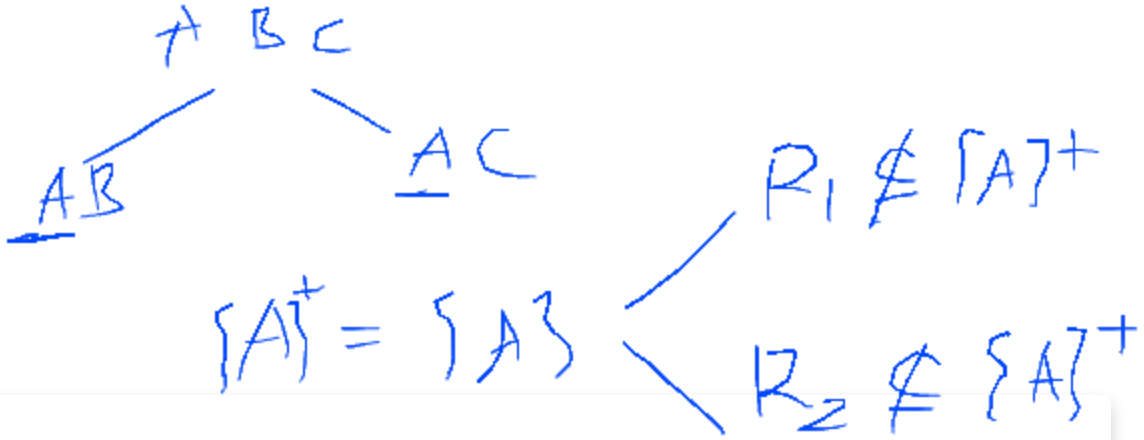
Decomposition

Decomposition
Lossless-Join
Definition
Examples
Lemma
Visualization
Projection
Preservation
Algorithm

Lossless-Join

Examples

Lossy-Join



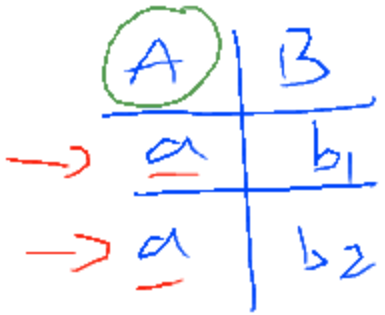
Let $R = \{A, B, C\}$ with $\Sigma = \{ \{B\} \rightarrow \{C\}, \{C\} \rightarrow \{B\} \}$

Original

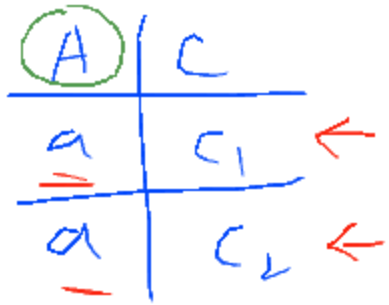
A	B	C
a	b_1	c_1
a	b_2	c_2

Decomposition

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

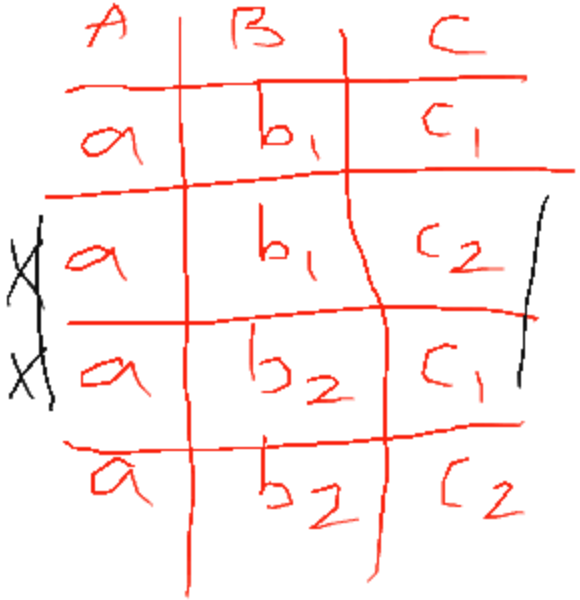


$R_2 := \pi_{[A,C]}(R)$



Recombined

$R_1 \bowtie R_2$



Decomposition

Decomposition
Lossless-Join
Definition
Examples
Lemma
Visualization
Projection
Preservation
Algorithm

Lossless-Join

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

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

A	B
a	b_1
a	b_2

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

A	C
a	c_1
a	c_2

Recombined

$R_1 \bowtie R_2$

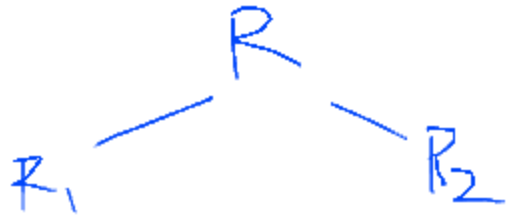
A	B	C
a	b_1	c_1
a	b_1	c_2
a	b_2	c_1
a	b_2	c_2

Decomposition

Decomposition
Lossless-Join
Definition
Examples
Lemma
Visualization
Projection
Preservation
Algorithm

Lossless-Join

Lemma



$(R_1 \cap R_2)^+ = C_n$

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 $(R_1 \cap R_2) \rightarrow R_2$

(1) $R_1 \subseteq C_n$

(2) $R_2 \subseteq C_n$

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.

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.

Decomposition

Decomposition
Lossless-Join
Definition
Examples
Lemma
Visualization
Projection
Preservation
Algorithm

Lossless-Join

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

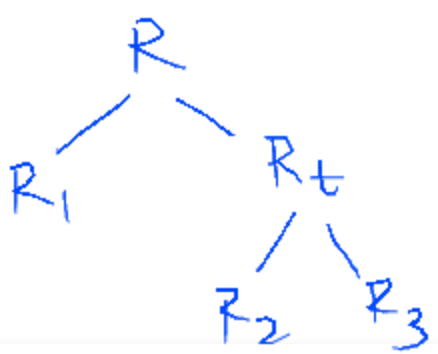
1. Find the **intersection** $(R_1 \cap R_2)$. Let this be called $R_\cap$.
2. Compute the **attribute closure** $R_\cap^+$ with Σ.
3. Check if $R_1 \subseteq R_\cap^+$ or $R_2 \subseteq R_\cap^+$ *(or both)*.

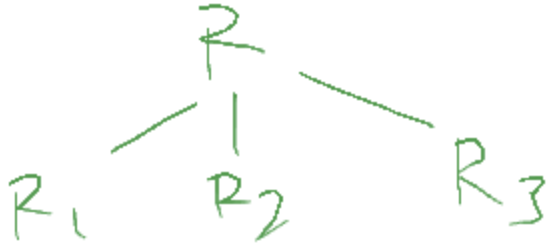
Decomposition

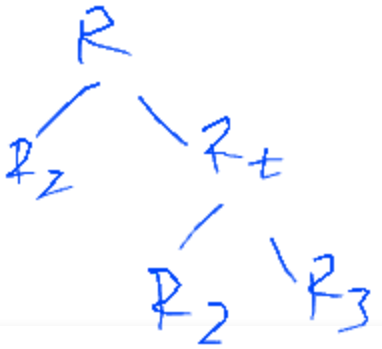
Decomposition
Lossless-Join
Definition
Examples
Lemma
Visualization
Projection
Preservation
Algorithm

Lossless-Join

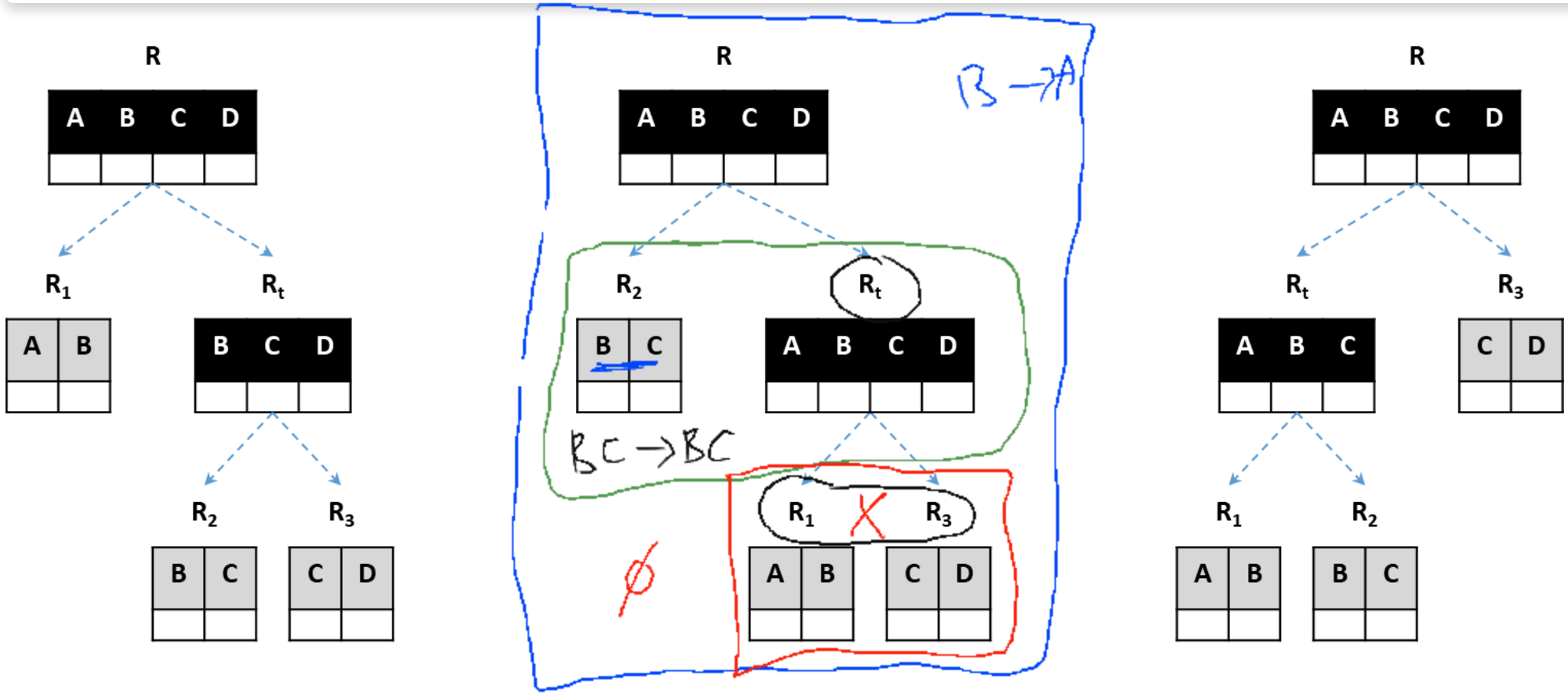
Visualization







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



Decomposition

Decomposition
Lossless-Join
Projection
Definition
Example
Steps
Preservation
Algorithm

Projection

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'

$X \rightarrow Y \in \Sigma^+$

$R' \subseteq X \wedge R' \subseteq Y$

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

Decomposition

Decomposition
Lossless-Join
› Projection
Definition
Example
Steps
Preservation
Algorithm

Projection

Example

Question

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

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

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

Decomposition

Decomposition
Lossless-Join
› Projection
Definition
Example
Steps
Preservation
Algorithm

Projection

Example

$AB^+ = ABDE$

$B^+ = BDE$

$AB \rightarrow DE$

$B \rightarrow DE$

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 (*i.e.*, $\Sigma|_{R'}$)?

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

Decomposition

Decomposition
Lossless-Join
Projection
Definition
Example
Steps
Preservation
Algorithm

Projection

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 (*i.e.*, $\Sigma|_{R'}$)?

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

Decomposition

Decomposition
Lossless-Join
Projection
Definition
Example
Steps
Preservation
Algorithm

Projection

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 (*i.e.*, $\Sigma|_{R'}$)?

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

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

Decomposition

Decomposition
Lossless-Join
› Projection
Definition
Example
Steps
Preservation
Algorithm

Projection

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 (*i.e.*, $\Sigma|_{R'}$)?

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

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

Issue

Do not forget that we need to look at **all functional dependencies** in Σ^+. Otherwise, the result may not be an **equivalent** set. Above, $\{B\} \rightarrow \{D\}$ is not logically implied by the rest.

Decomposition

Decomposition
Lossless-Join
Projection
Definition
Example
Steps
Preservation
Algorithm

Projection

Steps

Step 1: List All Subset of Attributes

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

Decomposition

Decomposition
Lossless-Join
Projection
Definition
Example
Steps
Preservation
Algorithm

Projection

Steps

Step 2: Compute the Attribute Closures

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

Decomposition

Decomposition
Lossless-Join
Projection
Definition
Example
Steps
Preservation
Algorithm

Projection

Steps

Step 3: Intersect Right Hand Side with R'

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

Decomposition

Decomposition
Lossless-Join
Projection
Definition
Example
Steps
Preservation
Algorithm

Projection

Steps

Step 4: Subtract Left Hand Side from Right Hand Side

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

Decomposition

Decomposition
Lossless-Join
Projection
Definition
Example
Steps
Preservation
Algorithm

Projection

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.

Decomposition

Decomposition
Lossless-Join
Projection
Definition
Example
Steps
Preservation
Algorithm

Projection

Steps

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

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

Decomposition

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

Decomposition
Lossless-Join
Projection
Preservation
Definition
Examples
Algorithm

Preservation

Definition

Dependency Preserving Decomposition

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

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

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

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.

Decomposition

Decomposition
Lossless-Join
Projection
Preservation
Definition
Examples
Algorithm

Preservation

Examples

Example #1

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

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

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

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

Decomposition

Decomposition
Lossless-Join
Projection
Preservation
Definition
Examples
Algorithm

Preservation

Examples

Example #1

Let $R = \{A, B, C\}$ with $\Sigma = \{ \{C\} \rightarrow \{B\} \}$ *(the candidate key is {A, C})* is the following 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 $\Sigma|_{R_2} = \{ \{C\} \rightarrow \{B\} \}$

Decomposition

Decomposition
Lossless-Join
Projection
Preservation
Definition
Examples
Algorithm

Preservation

Examples

Example #1

Let $R = \{A, B, C\}$ with $\Sigma = \{ \{C\} \rightarrow \{B\} \}$ *(the candidate key is {A, C})* is the following 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 $\Sigma|_{R_2} = \{ \{C\} \rightarrow \{B\} \}$

Steps

1. $(\Sigma|_{R_1} \cup \Sigma|_{R_2}) = \{ \{C\} \rightarrow \{B\} \}$
2. $(\Sigma|_{R_1} \cup \Sigma|_{R_2}) \equiv \Sigma$

Decomposition

Decomposition
Lossless-Join
Projection
Preservation
Definition
Examples
Algorithm

Preservation

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 following decomposition a ***dependency preserving*** lossless-join decomposition?

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

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

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

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

Decomposition

Decomposition
Lossless-Join
Projection
Preservation
Definition
Examples
Algorithm

Preservation

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

Decomposition

Decomposition
Lossless-Join
Projection
Preservation
Definition
Examples
Algorithm

Preservation

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

Steps

1. $(\Sigma|_{R_1} \cup \Sigma|_{R_2}) = \{ \{C\} \rightarrow \{B\} \}$
2. Consider $\{A,B\} \rightarrow \{C\}$.

   $\{A,B\}^+$ with respect to $(\Sigma|_{R_1} \cup \Sigma|_{R_2})$ is $\{A, B\}$.
3. $(\Sigma|_{R_1} \cup \Sigma|_{R_2}) \not\equiv \Sigma$

Decomposition

Decomposition
Lossless-Join
Projection
Preservation
Definition
Examples
Algorithm

Preservation

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

Steps

1. $(\Sigma|_{R_1} \cup \Sigma|_{R_2}) = \{ \{C\} \rightarrow \{B\} \}$
2. Consider $\{A,B\} \rightarrow \{C\}$.

   $\{A,B\}^+$ with respect to $(\Sigma|_{R_1} \cup \Sigma|_{R_2})$ is $\{A, B\}$.
3. $(\Sigma|_{R_1} \cup \Sigma|_{R_2}) \not\equiv \Sigma$

Bad News

The functional dependency $\{A, B\} \rightarrow \{C\}$ is **lost**!

Decomposition

Decomposition
Lossless-Join
Projection
Preservation
Definition
Examples
Algorithm

Preservation

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 following decomposition a ***dependency preserving*** lossless-join decomposition?

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

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

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

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

$\Sigma_J \equiv \Sigma$

Decomposition

Decomposition
Lossless-Join
Projection
Preservation
Definition
Examples
Algorithm

Preservation

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

Decomposition

Decomposition
Lossless-Join
Projection
Preservation
Definition
Examples
Algorithm

Preservation

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

Steps

1. $(\Sigma|_{R_1} \cup \Sigma|_{R_2}) = \{ \{B\} \rightarrow \{B\}, \{B\} \rightarrow \{C\} \}$
2. Consider $\{A\} \rightarrow \{C\}$ *(the rest are trivial)*.

   $\{A\}^+$ with respect to $(\Sigma|_{R_1} \cup \Sigma|_{R_2})$ is $\{A, B, C\}$.
3. $(\Sigma|_{R_1} \cup \Sigma|_{R_2}) \equiv \Sigma$

Decomposition

Decomposition
Lossless-Join
Projection
Preservation
Definition
Examples
Algorithm

Preservation

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

Steps

1. $(\Sigma|_{R_1} \cup \Sigma|_{R_2}) = \{ \{B\} \rightarrow \{B\}, \{B\} \rightarrow \{C\} \}$
2. Consider $\{A\} \rightarrow \{C\}$ *(the rest are trivial)*.

   $\{A\}^+$ with respect to $(\Sigma|_{R_1} \cup \Sigma|_{R_2})$ is $\{A, B, C\}$.
3. $(\Sigma|_{R_1} \cup \Sigma|_{R_2}) \equiv \Sigma$

Good News

The functional dependency $\{A\} \rightarrow \{C\}$ is **not lost**!

Decomposition

Decomposition
Lossless-Join
Projection
Preservation
Definition
Examples
Algorithm

Preservation

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

A → B

B → C

A	B

B	C

A	B	C

Decomposition

Decomposition
Lossless-Join
Projection
Preservation
Algorithm
Basic
BCNF Decomposition
Example
Theorems

Algorithm

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.

Decomposition

Decomposition
Lossless-Join
Projection
Preservation
Algorithm
Basic
BCNF Decomposition
Example
Theorems

Algorithm

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***.

Decomposition

Decomposition
Lossless-Join
Projection
Preservation
Algorithm
Basic
BCNF Decomposition
Example
Theorems

AB AC

Algorithm

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**?

Decomposition

Decomposition
Lossless-Join
Projection
Preservation
Algorithm
Basic
BCNF Decomposition
Example
Theorems

Algorithm

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**.

Decomposition

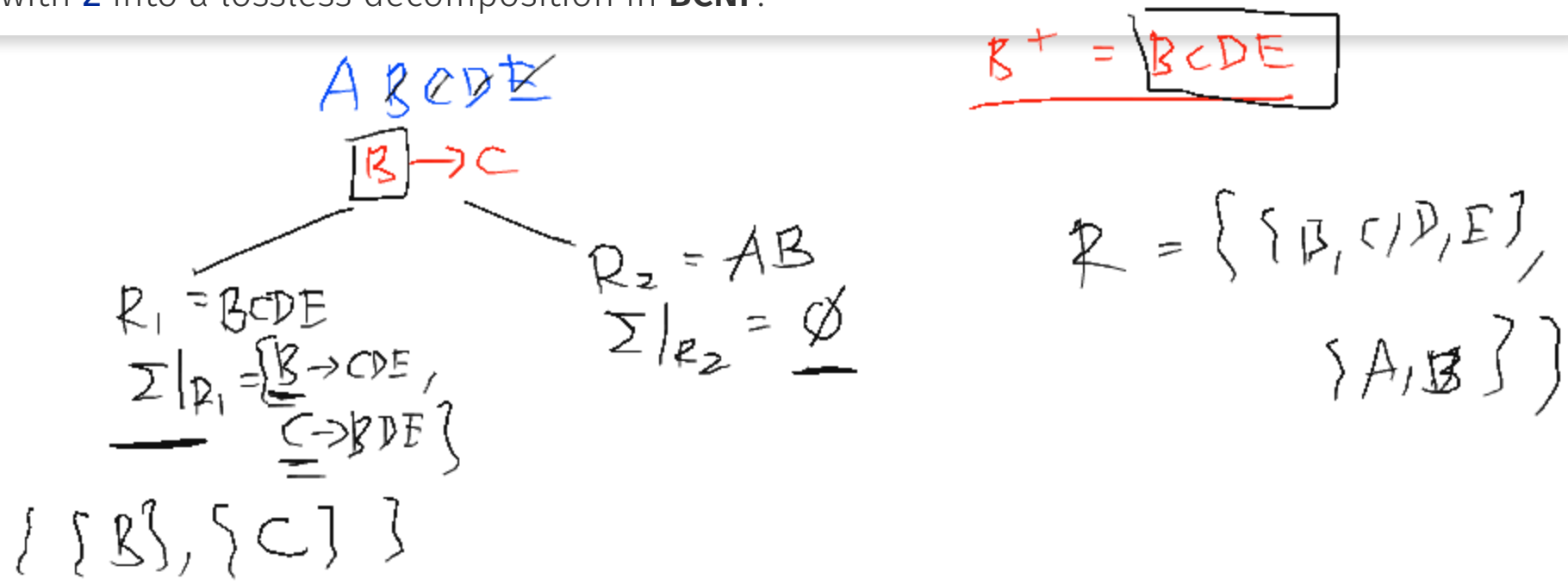
Decomposition
Lossless-Join
Projection
Preservation
Algorithm
Basic
BCNF Decomposition
Example
Theorems

Algorithm

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**.



Decomposition

Decomposition
Lossless-Join
Projection
Preservation
Algorithm
Basic
BCNF Decomposition
Example
Theorems

Algorithm

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?

Decomposition

Decomposition
Lossless-Join
Projection
Preservation
Algorithm
Basic
BCNF Decomposition
Example
Theorems

Algorithm

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?

Decomposition

Decomposition
Lossless-Join
Projection
Preservation
Algorithm
Basic
BCNF Decomposition
Example
Theorems

Algorithm

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)*

Decomposition

Decomposition
Lossless-Join
Projection
Preservation
Algorithm
Basic
BCNF Decomposition
Example
Theorems

Algorithm

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?

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

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

Decomposition

Decomposition
Lossless-Join
Projection
Preservation
Algorithm
Basic
BCNF Decomposition
Example
Theorems

Algorithm

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

Decomposition

Decomposition
Lossless-Join
Projection
Preservation
Algorithm
Basic
BCNF Decomposition
Example
Theorems

Algorithm

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

Decomposition

Decomposition
Lossless-Join
Projection
Preservation
Algorithm
Basic
BCNF Decomposition
Example
Theorems

Algorithm

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

Decomposition

Decomposition
Lossless-Join
Projection
Preservation
Algorithm
Basic
BCNF Decomposition
Example
Theorems

Algorithm

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***.

Decomposition

Decomposition
Lossless-Join
Projection
Preservation
Algorithm
Basic
BCNF Decomposition
Example
Theorems

Algorithm

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?

Decomposition

Decomposition
Lossless-Join
Projection
Preservation
Algorithm
Basic
BCNF Decomposition
Example
Theorems

Algorithm

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.

Decomposition

Decomposition
Lossless-Join
Projection
Preservation
Algorithm
Basic
BCNF Decomposition
Example
Theorems

Algorithm

Theorems

Theorem #7: Lossless Join Decomposition

The BCNF decomposition algorithm is a lossless-join decomposition.

Theorem #8: Termination

The BCNF decomposition algorithm will terminate.

Back to Our Case Study

Our Case
Instance
ERD
Decomposition

Our Case

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

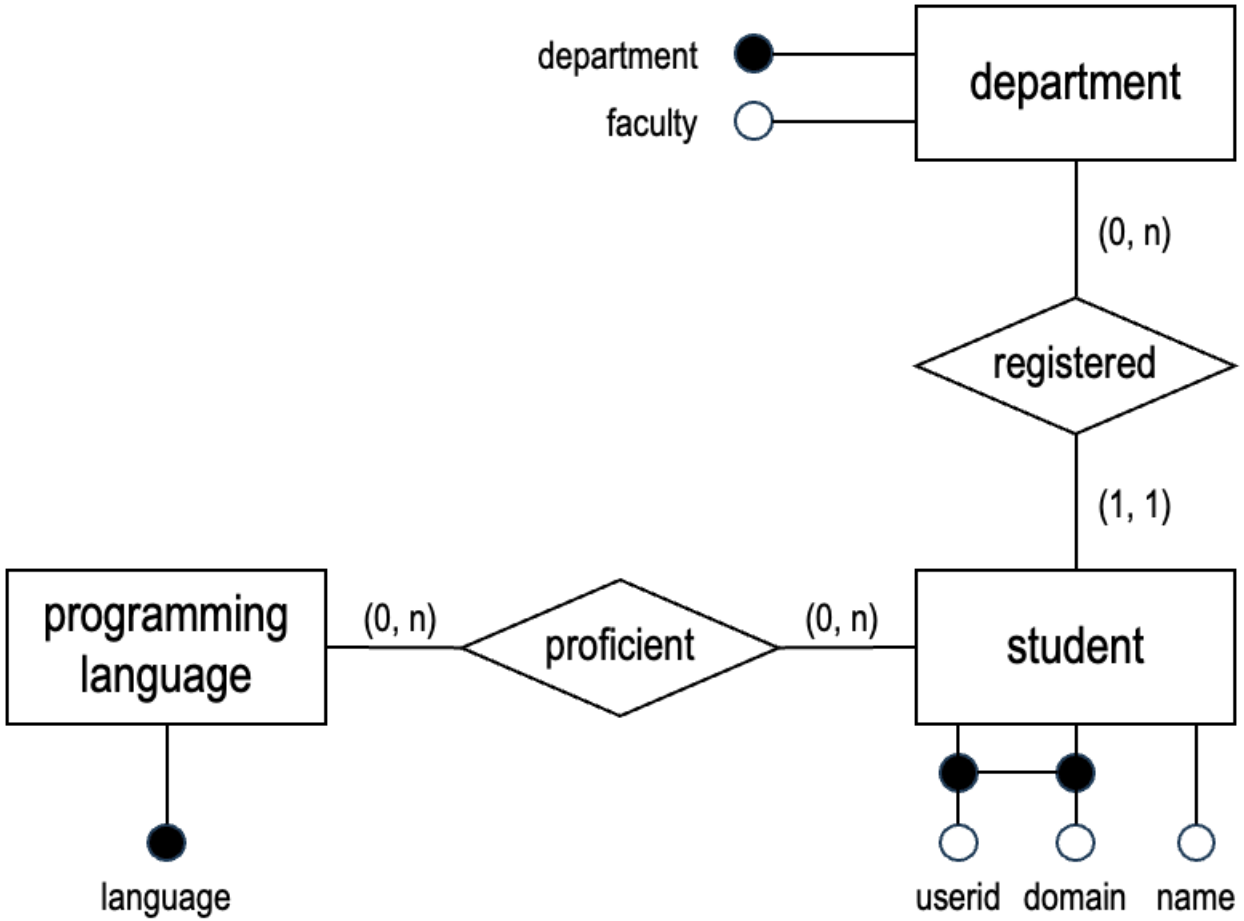
A: name
B: userid
C: domain
D: department
E: faculty
F: language

Back to Our Case Study

Our Case
Instance
ERD
Decomposition

Our Case

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

Back to Our Case Study

Our Case
Instance
ERD
Decomposition

Our Case

Decomposition

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

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

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

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

Mapping

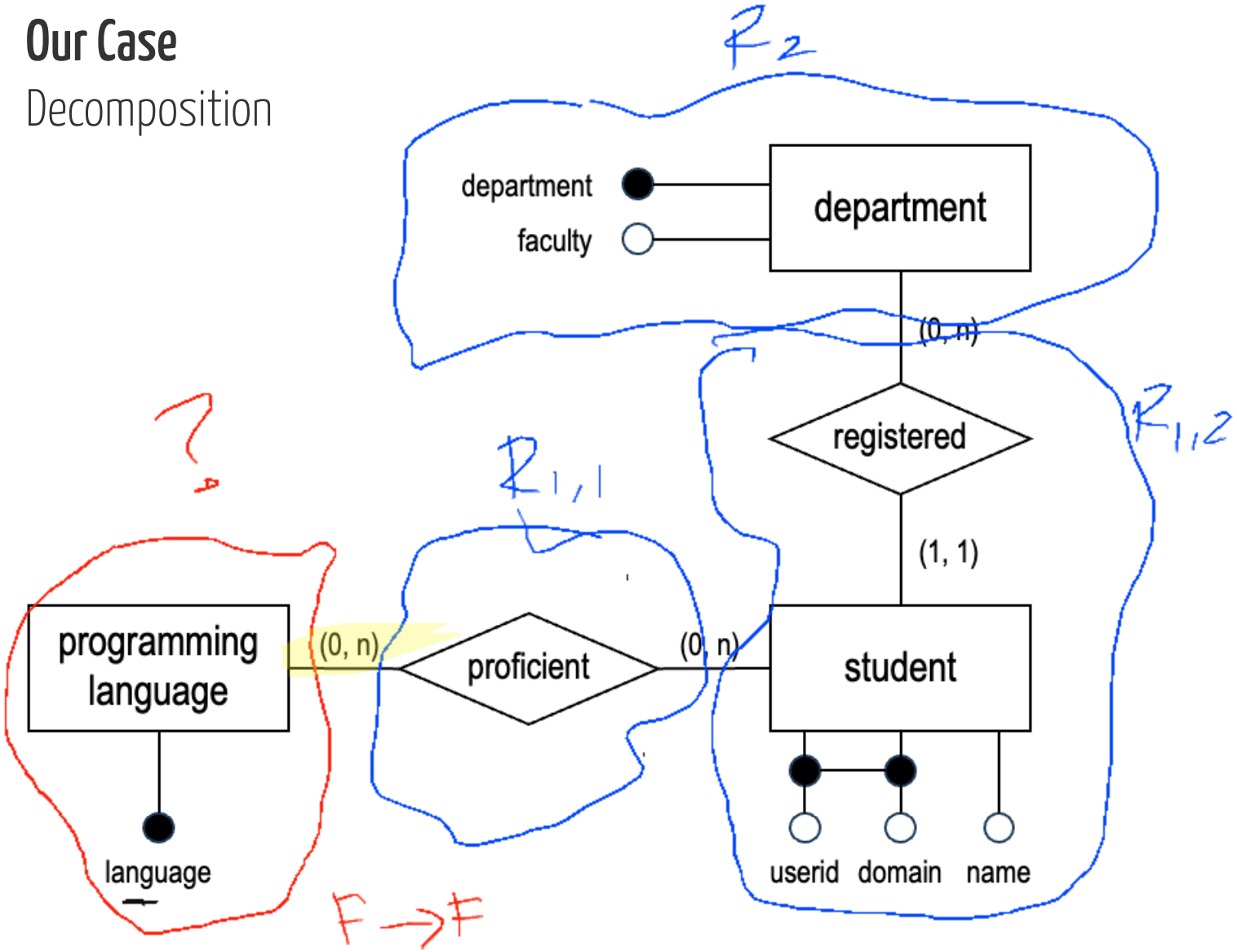
A: name
B: userid
C: domain
D: department
E: faculty
F: language

Back to Our Case Study

- Our Case
  - *Instance*
  - *ERD*
  - ***Decomposition***

Our Case

Decomposition



Fragments

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

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

Proof Sketch

Theorem #7
Theorem #8

Theorem #7

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 Sketch

Theorem #7
Theorem #8

Theorem #8

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=# \q
Press any key to continue . . .
```