Decomposition and Normal Forms

COMPSCI 2DB3: Databases

Jelle Hellings Holly Koponen

Department of Computing and Software
McMaster University

Winter 2024

Let us revisit an example

student

<u>sid</u>	name	age	birthdate	program	department
1	Alicia	21	August 27, 2000	COMPSCI	Comp. and Soft.
2	Bo	20	December 15, 2000	SFWRENG	Comp. and Soft.
3	Celeste	22	April 24, 1999	SFWRENG	Comp. and Soft.
4	Dafni	20	February 1, 2001	COMPSCI	Comp. and Soft.
5	Eva	23	July 2, 1998	COMPSCI	Comp. and Soft.
6	Frieda	21	August 27, 2000	CLASSICS	Classics

What is wrong with this table?

Let us revisit an example

student

sid	name	age	birthdate	program	department
1	Alicia	21	August 27, 2000	COMPSCI	Comp. and Soft.
2	Bo	20	December 15, 2000	SFWRENG	Comp. and Soft.
3	Celeste	22	April 24, 1999	SFWRENG	Comp. and Soft.
4	Dafni	20	February 1, 2001	COMPSCI	Comp. and Soft.
5	Eva	23	July 2, 1998	COMPSCI	Comp. and Soft.
6	Frieda	21	August 27, 2000	CLASSICS	Classics

What is wrong with this table?

- The *age* is derivable from *birthdate*.

Let us revisit an example

student					
sid	name	age	birthdate	program	department
1	Alicia	21	August 27, 2000	COMPSCI	Comp. and Soft.
2	Bo	20	December 15, 2000	SFWRENG	Comp. and Soft.
3	Celeste	22	April 24, 1999	SFWRENG	Comp. and Soft.
4	Dafni	20	February 1, 2001	COMPSCI	Comp. and Soft.
5	Eva	23	July 2, 1998	COMPSCI	Comp. and Soft.
6	Frieda	21	August 27, 2000	CLASSICS	Classics

What is wrong with this table?

- The *age* is derivable from *birthdate*.
- The derivation of *age* from *birthdate* is stored repeatedly.

Let us revisit an example

student

sid	name	age	birthdate	program	department
1	Alicia	21	August 27, 2000	COMPSCI	Comp. and Soft.
2	Bo	20	December 15, 2000	SFWRENG	Comp. and Soft.
3	Celeste	22	April 24, 1999	SFWRENG	Comp. and Soft.
4	Dafni	20	February 1, 2001	COMPSCI	Comp. and Soft.
5	Eva	23	July 2, 1998	COMPSCI	Comp. and Soft.
6	Frieda	21	August 27, 2000	CLASSICS	Classics

What is wrong with this table?

- The *age* is derivable from *birthdate*.
- The derivation of *age* from *birthdate* is stored repeatedly.
- The *department* of a *program* is stored repeatedly.

Let us revisit an example

student					
sid	name	age	birthdate	program	department
1	Alicia	21	August 27, 2000	COMPSCI	Comp. and Soft.
2	Bo	20	December 15, 2000	SFWRENG	Comp. and Soft.
3	Celeste	22	April 24, 1999	SFWRENG	Comp. and Soft.
4	Dafni	20	February 1, 2001	COMPSCI	Comp. and Soft.
5	Eva	23	July 2, 1998	COMPSCI	Comp. and Soft.
6	Frieda	21	August 27, 2000	CLASSICS	Classics

Remember the functional dependencies

- "birthdate $\longrightarrow$ age".
- "program $\longrightarrow$ department".

Let us revisit an example

student					
sid	name	age	birthdate	program	department
1	Alicia	21	August 27, 2000	COMPSCI	Comp. and Soft.
2	Bo	20	December 15, 2000	SFWRENG	Comp. and Soft.
3	Celeste	22	April 24, 1999	SFWRENG	Comp. and Soft.
4	Dafni	20	February 1, 2001	COMPSCI	Comp. and Soft.
5	Eva	23	July 2, 1998	COMPSCI	Comp. and Soft.
6	Frieda	21	August 27, 2000	CLASSICS	Classics

Remember the functional dependencies

- "birthdate $\longrightarrow$ age".
- "program $\longrightarrow$ department".

Exactly those attributes that are involved in *redundancies*.

Let us revisit an example

Remember the functional dependencies

- "birthdate $\longrightarrow$ age".
- "program $\longrightarrow$ department".

Exactly those attributes that are involved in *redundancies*.

Improving the table structure

student

sid	name	birthdate	program
1	Alicia	August 27, 2000	COMPSCI
2	Bo	December 15, 2000	SFWRENG
3	Celeste	April 24, 1999	SFWRENG
4	Dafni	February 1, 2001	COMPSCI
5	Eva	July 2, 1998	COMPSCI
6	Frieda	August 27, 2000	CLASSICS

date_info

birthdate	age
August 27, 2000	21
December 15, 2000	20
April 24, 1999	22
February 1, 2001	20
July 2, 1998	23

prog_dept

program	department
COMPSCI	Comp. and Soft.
SFWRENG	Comp. and Soft.
CLASSICS	Classics

Let us revisit an example

Remember the functional dependencies

- "birthdate $\longrightarrow$ age".
- "program $\longrightarrow$ department".

Exactly those attributes that are involved in *redundancies*.

Improving the table structure

student

sid	name	birthdate	program
1	Alicia	August 27, 2000	COMPSCI
2	Bo	December 15, 2000	SFWRENG
3	Celeste	April 24, 1999	SFWRENG
4	Dafni	February 1, 2001	COMPSCI
5	Eva	July 2, 1998	COMPSCI
6	Frieda	August 27, 2000	CLASSICS

prog_dept

program	department
COMPSCI	Comp. and Soft.
SFWRENG	Comp. and Soft.
CLASSICS	Classics

Let us revisit a second example

course_details

course	student	TA
Programming	Celeste	Alicia
Programming	Frieda	Alicia
Programming	Celeste	Dafni
Programming	Frieda	Dafni
Databases	Bo	Eva
Databases	Dafni	Eva
Databases	Bo	Alicia
Databases	Dafni	Alicia

What is wrong with this table?

Let us revisit a second example

course_details

course	student	TA
Programming	Celeste	Alicia
Programming	Frieda	Alicia
Programming	Celeste	Dafni
Programming	Frieda	Dafni
Databases	Bo	Eva
Databases	Dafni	Eva
Databases	Bo	Alicia
Databases	Dafni	Alicia

What is wrong with this table?

- The enrolled *student*s of a *course* are stored repeatedly.

Let us revisit a second example

course_details		
course	student	TA
Programming	Celeste	Alicia
Programming	Frieda	Alicia
Programming	Celeste	Dafni
Programming	Frieda	Dafni
Databases	Bo	Eva
Databases	Dafni	Eva
Databases	Bo	Alicia
Databases	Dafni	Alicia

What is wrong with this table?

- The enrolled *student*s of a *course* are stored repeatedly.
- The *TA*s of a *course* are stored repeatedly.

Let us revisit a second example

course_details		
course	student	TA
Programming	Celeste	Alicia
Programming	Frieda	Alicia
Programming	Celeste	Dafni
Programming	Frieda	Dafni
Databases	Bo	Eva
Databases	Dafni	Eva
Databases	Bo	Alicia
Databases	Dafni	Alicia

Remember the multivalued dependencies

- "course $\twoheadrightarrow$ student" and "course $\twoheadrightarrow$ TA".

Let us revisit a second example

course_details

course	student	TA
Programming	Celeste	Alicia
Programming	Frieda	Alicia
Programming	Celeste	Dafni
Programming	Frieda	Dafni
Databases	Bo	Eva
Databases	Dafni	Eva
Databases	Bo	Alicia
Databases	Dafni	Alicia

Remember the multivalued dependencies

- “course $\twoheadrightarrow$ student” and “course $\twoheadrightarrow$ TA”.

Exactly those attributes that are involved in *redundancies*.

Let us revisit a second example

Remember the multivalued dependencies

- “course $\twoheadrightarrow$ student” and “course $\twoheadrightarrow$ TA”.

Exactly those attributes that are involved in *redundancies*.

Improving the table structure

course_students

course	TA
Programming	Celeste
Programming	Frieda
Databases	Bo
Databases	Dafni

course_TAs

course	TA
Programming	Alicia
Programming	Dafni
Databases	Eva
Databases	Alicia

The downsides of redundancies

all_course_details			
course	student	TA	Instructor
Databases	Bo	Eva	Celeste
Databases	Dafni	Eva	Celeste
Databases	Bo	Alicia	Celeste
Databases	Dafni	Alicia	Celeste

The **all_course_details** table is a big relationship table that:

- enrolls students in courses; and
- assigns TAs and instructors to courses.

We assume that each course has *exactly* one instructor.

The downsides of redundancies

all_course_details

course	student	TA	Instructor
Databases	Bo	Eva	Celeste
Databases	Dafni	Eva	Celeste
Databases	Bo	Alicia	Celeste
Databases	Dafni	Alicia	Celeste

Downside 1: Redundant storage

Information is stored repeatedly.

Celeste is the instructor of Databases is stored *four* times.

The downsides of redundancies

all_course_details			
course	student	TA	Instructor
Databases	Bo	Eva	Celeste
Databases	Dafni	Eva	Celeste
Databases	Bo	Alicia	Celeste
Databases	Dafni	Alicia	Celeste

Downside 1: Redundant storage

Information is stored repeatedly.
Celeste is the instructor of Databases is stored *four* times.

- Storage is not *free*: hardware and energy costs.
- Can negatively affect performance: less data overall can be stored in *fast* memory (e.g., CPU caches, main memory, non-volatile memory, SSDs, ...).

The downsides of redundancies

all_course_details

course	student	TA	Instructor
Databases	Bo	Eva	Celeste
Databases	Dafni	Eva	Celeste
Databases	Bo	Alicia	Celeste
Databases	Dafni	Alicia	Celeste

Downside 2: Update anomalies

Updating a value becomes harder: one has to track down and update *all copies*!

The downsides of redundancies

all_course_details

course	student	TA	Instructor
Databases	Bo	Eva	Celeste
Databases	Dafni	Eva	Celeste
Databases	Bo	Alicia	Celeste
Databases	Dafni	Alicia	Celeste

Downside 2: Update anomalies

Updating a value becomes harder: one has to track down and update *all copies*!

- Will negatively affect performance.
- Can lock the involved tables and prevent other (concurrent) usages!

The downsides of redundancies

all_course_details

course	student	TA	Instructor
Databases	Bo	Eva	Celeste
Databases	Dafni	Eva	Celeste
Databases	Bo	Alicia	Celeste
Databases	Dafni	Alicia	Celeste

Downside 3: Insert anomalies

Inserting a value becomes harder: one has to duplicate many records!
E.g., to enroll a student to a course, we need to duplicate all instructor and TA data.

We cannot create a new course without students, or TAs.

The downsides of redundancies

all_course_details

course	student	TA	Instructor
Databases	Bo	Eva	Celeste
Databases	Dafni	Eva	Celeste
Databases	Bo	Alicia	Celeste
Databases	Dafni	Alicia	Celeste

Downside 3: Insert anomalies

Inserting a value becomes harder: one has to duplicate many records!
E.g., to enroll a student to a course, we need to duplicate all instructor and TA data.

We cannot create a new course without students, or TAs.

- As inserts become more complex: will negatively affect performance.
- Workaround: create a new course by allowing `NULL` values for student and TA.
  Should an *enrollment relationship* be responsible to maintain the course *entity*?

The downsides of redundancies

all_course_details

course	student	TA	Instructor
Databases	Bo	Eva	Celeste
Databases	Dafni	Eva	Celeste
Databases	Bo	Alicia	Celeste
Databases	Dafni	Alicia	Celeste

Downside 4: Delete anomalies

Deleting a value becomes harder: one has to track down and delete *all copies*!

We cannot delete the last student or the last TA without deleting the entire course.

The downsides of redundancies

all_course_details

course	student	TA	Instructor
Databases	Bo	Eva	Celeste
Databases	Dafni	Eva	Celeste
Databases	Bo	Alicia	Celeste
Databases	Dafni	Alicia	Celeste

Downside 4: Delete anomalies

Deleting a value becomes harder: one has to track down and delete *all copies*!

We cannot delete the last student or the last TA without deleting the entire course.

- As deletes become more complex: will negatively affect performance.
- We again can work around some limitations by using `NULL` values.
  But: `NULL` values complicate queries and are to be avoided in most cases.

Observations

- Redundancies are *bad*.

Observations

- Redundancies are *bad*.
- Breaking up tables can *improve* their structure.

Observations

- Redundancies are *bad*.
- Breaking up tables can *improve* their structure.
- Breaking up tables can *eliminate* redundancies.

Observations

- Redundancies are *bad*.
- Breaking up tables can *improve* their structure.
- Breaking up tables can *eliminate* redundancies.
- Dependencies seem to indicate *where and how* to break up tables.

Observations

- Redundancies are *bad*.
- Breaking up tables can *improve* their structure.
- Breaking up tables can *eliminate* redundancies.
- Dependencies seem to indicate *where and how* to break up tables.

Definitions

A *decomposition of a relational schema* $\mathbf{R}$ consists of replacing $\mathbf{R}$ by multiple relational schemas, each over a subset of the attributes of $\mathbf{R}$.

Observations

- Redundancies are *bad*.
- Breaking up tables can *improve* their structure.
- Breaking up tables can *eliminate* redundancies.
- Dependencies seem to indicate *where and how* to break up tables.

Definitions

A *decomposition of a relational schema* **R** consists of replacing **R** by multiple relational schemas, each over a subset of the attributes of **R**.

- Decomposition is an example of *schema refinement*.
- Decomposition can reduce *redundancies*.
- Not all decompositions are *good* schema refinements.

An example of a bad decomposition

course_details

course	student	TA
Programming	Celeste	Alicia
Programming	Frieda	Alicia
Programming	Celeste	Dafni
Programming	Frieda	Dafni
Databases	Bo	Eva
Databases	Dafni	Eva
Databases	Bo	Alicia
Databases	Dafni	Alicia

An example of a bad decomposition

courses
course
Programming
Databases

students
students
Celeste
Frieda
Bo
Dafni

TAs
TA
Alicia
Dafni
Eva

An example of a bad decomposition

courses
course
Programming
Databases

students
students
Celeste
Frieda
Bo
Dafni

TAs
TA
Alicia
Dafni
Eva

This decomposition *looses* information!

- To which courses do *students* belong?
- To which courses do *TAs* belong?
- Are there TAs/students that are TA/student for *several* courses?

Criteria for good decompositions

Consider a relational schema $\mathbf{R}$ decomposed into schemas $\mathbf{R}_1, \ldots, \mathbf{R}_n$.
Let $\mathcal{I}$ be any instance of $\mathbf{R}$ that is decomposed into instances $\mathcal{I}_1$ of $\mathbf{R}_1$, …, $\mathcal{I}_n$ of $\mathbf{R}_n$.

We say that a decomposition is

Criteria for good decompositions

Consider a relational schema $\mathbf{R}$ decomposed into schemas $\mathbf{R}_1, \ldots, \mathbf{R}_n$.
Let $\mathcal{I}$ be any instance of $\mathbf{R}$ that is decomposed into instances $\mathcal{I}_1$ of $\mathbf{R}_1$, …, $\mathcal{I}_n$ of $\mathbf{R}_n$.

We say that a decomposition is

Lossless-Join if the join of the decomposed parts is always the original instance. Hence,

$$\mathbf{R} = \mathbf{R}_1 \bowtie \ldots \bowtie \mathbf{R}_n.$$

Rationale.
The decomposition *must* represent exactly the original data!

Criteria for good decompositions

Consider a relational schema $\mathbf{R}$ decomposed into schemas $\mathbf{R}_1, \ldots, \mathbf{R}_n$.
Let $\mathcal{I}$ be any instance of $\mathbf{R}$ that is decomposed into instances $\mathcal{I}_1$ of $\mathbf{R}_1$, …, $\mathcal{I}_n$ of $\mathbf{R}_n$.

We say that a decomposition is

Lossless-Join if the join of the decomposed parts is always the original instance. Hence,

$$\mathbf{R} = \mathbf{R}_1 \bowtie \ldots \bowtie \mathbf{R}_n.$$

Rationale.
The decomposition *must* represent exactly the original data!

Dependency-Preserving if all constraints on $\mathbf{R}$ can be maintained using only constraints on the individual relational schemas $\mathbf{R}_1, \ldots, \mathbf{R}_n$.

Rationale.
The decomposition *simplifies* maintaining data consistency!

An example of a bad decomposition—revisited

course_details

course	student	TA
Programming	Celeste	Alicia
Programming	Frieda	Alicia
Programming	Celeste	Dafni
Programming	Frieda	Dafni
Databases	Bo	Eva
Databases	Dafni	Eva
Databases	Bo	Alicia
Databases	Dafni	Alicia

An example of a bad decomposition—revisited

courses
course
Programming
Databases

students
students
Celeste
Frieda
Bo
Dafni

TAs
TA
Alicia
Dafni
Eva

An example of a bad decomposition—revisited

courses
course
Programming
Databases

students
students
Celeste
Frieda
Bo
Dafni

TAs
TA
Alicia
Dafni
Eva

This decomposition is *not* lossless-join.

An example of a bad decomposition—revisited

courses
course
Programming
Databases

$\bowtie$

students
students
Celeste
Frieda
Bo
Dafni

$\bowtie$

TAs
TA
Alicia
Dafni
Eva

An example of a bad decomposition—revisited

course_details

course	student	TA
Programming	Celeste	Alicia
Programming	Celeste	Dafni
Programming	Celeste	Eva
Programming	Frieda	Alicia
Programming	Frieda	Dafni
Programming	Frieda	Eva
Programming	Bo	Alicia
Programming	Bo	Dafni
Programming	Bo	Eva
⋮	⋮	⋮

An example of a bad decomposition—revisited

course_details		
course	student	TA
Programming	Celeste	Alicia
Programming	Celeste	Dafni
Programming	Celeste	Eva
Programming	Frieda	Alicia
Programming	Frieda	Dafni
Programming	Frieda	Eva
Programming	Bo	Alicia
Programming	Bo	Dafni
Programming	Bo	Eva
⋮	⋮	⋮

Eva is not a TA of Programming.
Bo is not enrolled in Programming.
⋮

An example of dependency preservation

student

sid	name	age	birthdate	program	department
1	Alicia	21	August 27, 2000	COMPSCI	Comp. and Soft.
2	Bo	20	December 15, 2000	SFWRENG	Comp. and Soft.
3	Celeste	22	April 24, 1999	SFWRENG	Comp. and Soft.
4	Dafni	20	February 1, 2001	COMPSCI	Comp. and Soft.
5	Eva	23	July 2, 1998	COMPSCI	Comp. and Soft.
6	Frieda	21	August 27, 2000	CLASSICS	Classics

Functional dependencies to preserve

- "sid $\longrightarrow$ name, age, birthdate, program, department".
- "birthdate $\longrightarrow$ age".
- "program $\longrightarrow$ department".

An example of dependency preservation

Functional dependencies to preserve

- "sid $\longrightarrow$ name, age, birthdate, program, department".
- "birthdate $\longrightarrow$ age".
- "program $\longrightarrow$ department".

student

sid	name	birthdate	program
1	Alicia	August 27, 2000	COMPSCI
2	Bo	December 15, 2000	SFWRENG
3	Celeste	April 24, 1999	SFWRENG
4	Dafni	February 1, 2001	COMPSCI
5	Eva	July 2, 1998	COMPSCI
6	Frieda	August 27, 2000	CLASSICS

date_info

birthdate	age
August 27, 2000	21
December 15, 2000	20
April 24, 1999	22
February 1, 2001	20
July 2, 1998	23

prog_dept

program	department
COMPSCI	Comp. and Soft.
SFWRENG	Comp. and Soft.
CLASSICS	Classics

An example of dependency preservation

Functional dependencies to preserve

- "sid $\longrightarrow$ name, age, birthdate, program, department".
- "birthdate $\longrightarrow$ age".
- "program $\longrightarrow$ department".

student

<u>sid</u>	name	birthdate	program
1	Alicia	August 27, 2000	COMPSCI
2	Bo	December 15, 2000	SFWRENG
3	Celeste	April 24, 1999	SFWRENG
4	Dafni	February 1, 2001	COMPSCI
5	Eva	July 2, 1998	COMPSCI
6	Frieda	August 27, 2000	CLASSICS

date_info

<u>birthdate</u>	age
August 27, 2000	21
December 15, 2000	20
April 24, 1999	22
February 1, 2001	20
July 2, 1998	23

prog_dept

<u>program</u>	department
COMPSCI	Comp. and Soft.
SFWRENG	Comp. and Soft.
CLASSICS	Classics

Improvement: every functional dependency is *now* a primary key!

Normal forms: Guidelines for improving relational schemas

A *normal form* puts requirements on the structure of relational schemas.

Normal forms: Guidelines for improving relational schemas

A *normal form* puts requirements on the structure of relational schemas.

If these requirements are met, then some kinds of problems cannot arise.

Normal forms: Guidelines for improving relational schemas

A *normal form* puts requirements on the structure of relational schemas.

If these requirements are met, then some kinds of problems cannot arise.

There are many normal forms proposed over the years.

- First Normal Form (1NF);
- Second Normal Form (2NF);
- Third Normal Form (3NF);
- Boyce-Codd Normal Form (BCNF);
- Fourth Normal Form (4NF); and
- Fifth Normal Form (5NF).

Normal forms: Guidelines for improving relational schemas

A *normal form* puts requirements on the structure of relational schemas.

If these requirements are met, then some kinds of problems cannot arise.

There are many normal forms proposed over the years.

- First Normal Form (1NF);
- Second Normal Form (2NF); $\longleftarrow$ *functional dependencies*
- Third Normal Form (3NF); $\longleftarrow$ *functional dependencies*
- Boyce-Codd Normal Form (BCNF); $\longleftarrow$ *functional dependencies*
- Fourth Normal Form (4NF); and $\longleftarrow$ *multivalued dependencies*
- Fifth Normal Form (5NF). $\longleftarrow$ *join dependencies*

Normal forms: Guidelines for improving relational schemas

A *normal form* puts requirements on the structure of relational schemas.

If these requirements are met, then some kinds of problems cannot arise.

There are many normal forms proposed over the years.

- First Normal Form (1NF);
- Second Normal Form (2NF); $\longleftarrow$ *functional dependencies*
- Third Normal Form (3NF); $\longleftarrow$ *functional dependencies*
- Boyce-Codd Normal Form (BCNF); $\longleftarrow$ *functional dependencies*
- Fourth Normal Form (4NF); and $\longleftarrow$ *multivalued dependencies*
- Fifth Normal Form (5NF). $\longleftarrow$ *join dependencies*

These normal forms are a hierarchy of successively-more restrictive requirements.

The First Normal Form (1NF)

Definition

A relational schema is in *first normal form* if the domain of every attribute is atomic.

The First Normal Form (1NF)

Definition

A relational schema is in *first normal form* if the domain of every attribute is atomic.
A domain is *atomic* if elements of the domain are “indivisible”.

The First Normal Form (1NF)

Definition

A relational schema is in *first normal form* if the domain of every attribute is atomic.
A domain is *atomic* if elements of the domain are "indivisible".

- Sets and lists *are not* atomic.
- Complex objects *are not* atomic.
- Debatable: derivable attributes are not atomic.

The First Normal Form (1NF)

Definition

A relational schema is in *first normal form* if the domain of every attribute is atomic.
A domain is *atomic* if elements of the domain are "indivisible".

- Sets and lists *are not* atomic.
- Complex objects *are not* atomic.
- Debatable: derivable attributes are not atomic.

Our relational data model enforces 1NF by definition.

The Second Normal Form (2NF)

"…mainly of historical interest …"

The Second Normal Form (2NF)

"…mainly of historical interest …"

Definition

A relational schema **R** is in *second normal form* if it is in 1NF and if each attribute A of **R**

- is part of some key of **R**; or
- there is no functional dependency of the form $X \longrightarrow A$ with X a proper subset of a key.

The Second Normal Form (2NF)

"…mainly of historical interest …"

Definition

A relational schema **R** is in *second normal form* if it is in 1NF and if each attribute A of **R**

- is part of some key of **R**; or
- there is no functional dependency of the form $X \longrightarrow A$ with X a proper subset of a key.

I did not even know that when I made these slides!

An example of 2NF in practice

degree_programs

department	program	building	type
Comp. and Soft.	Computer Science	ITB	B.A.Sc
Comp. and Soft.	Mechatronics	ITB	B.Eng.
Chemical Engineering	Chemical Engineering	JHE	B.Eng.

An example of 2NF in practice

degree_programs

department	program	building	type
Comp. and Soft.	Computer Science	ITB	B.A.Sc
Comp. and Soft.	Mechatronics	ITB	B.Eng.
Chemical Engineering	Chemical Engineering	JHE	B.Eng.

$$\text{department} \longrightarrow \text{building}$$

An example of 2NF in practice

degree_programs			
<u>department</u>	<u>program</u>	building	type
Comp. and Soft.	Computer Science	ITB	B.A.Sc
Comp. and Soft.	Mechatronics	ITB	B.Eng.
Chemical Engineering	Chemical Engineering	JHE	B.Eng.

$$\text{department} \longrightarrow \text{building}$$

The relational schema **degree_programs** is not in 2NF

Take A = "building" and X = "department".
There is a functional dependency of the form $X \longrightarrow A$ with X a proper subset of a key.

An example of 2NF in practice

degree_programs

department	program	building	type
Comp. and Soft.	Computer Science	ITB	B.A.Sc
Comp. and Soft.	Mechatronics	ITB	B.Eng.
Chemical Engineering	Chemical Engineering	JHE	B.Eng.

$$\text{department} \longrightarrow \text{building}$$

The relational schema **degree_programs** is not in 2NF

Take A = "building" and X = "department".
There is a functional dependency of the form $X \longrightarrow A$ with X a proper subset of a key.

The 2NF violation points directly at a redundancy!

An example of 2NF in practice

We can decompose along the lines of department $\longrightarrow$ building.

degree_programs

department	program	type
Comp. and Soft.	Computer Science	B.A.Sc
Comp. and Soft.	Mechatronics	B.Eng.
Chemical Engineering	Chemical Engineering	B.Eng.

department_building

department	building
Comp. and Soft.	ITB
Chemical Engineering	JHE

This is a *lossless-join* and *dependency-preserving* decomposition and the result is in 2NF.

Note

We only verify 2NF with regards to the set of given functional dependencies!

The result would *not* be in 2NF if other functional dependencies hold (e.g., program $\longrightarrow$ department).

The Third Normal Form (3NF)

Definition

A relational schema **R** is in *third normal form* with respect to functional dependencies $\mathfrak{S}$ if it is in 1NF and if, for every $(X \longrightarrow A) \in \mathfrak{S}^+$, the following holds:

- $A \subseteq X$ (the dependency is trivial);
- X is a (super)key; or
- each attribute in $A \setminus X$ is part of a key of **R**.

The Third Normal Form (3NF)

Definition

A relational schema $\mathbf{R}$ is in *third normal form* with respect to functional dependencies $\mathfrak{S}$ if it is in 1NF and if, for every $(X \longrightarrow A) \in \mathfrak{S}^+$, the following holds:

- $A \subseteq X$ (the dependency is trivial);
- X is a (super)key; or
- each attribute in $A \setminus X$ is part of a key of $\mathbf{R}$.

Key versus superkey

Superkey: Any set of attributes that can uniquely identify rows.

Key: A superkey of minimal size:
if we remove any attribute from a key, it is no longer a superkey!

Types of 3NF violations

If $X \longrightarrow A$ caused a violation of 3NF

Types of 3NF violations

If $X \longrightarrow A$ caused a violation of 3NF and *X is a proper subset of some key* then we can end up storing (X, A) pairs redundantly.

Types of 3NF violations

If $X \longrightarrow A$ caused a violation of 3NF and *X is a proper subset of some key* then we can end up storing (X, A) pairs redundantly.

degree_programs			
department	program	building	type
Computing and Software	Computer Science	ITB	B.A.Sc
Computing and Software	Mechatronics	ITB	B.Eng.
Chemical Engineering	Chemical Engineering	JHE	B.Eng.

$$\text{department} \longrightarrow \text{building}.$$

These redundancies are already recognized by 2NF.

Types of 3NF violations

If $X \longrightarrow A$ caused a violation of 3NF and *X is not a proper subset of any key* then there is a chain of dependencies *key* $\longrightarrow X$ and $X \longrightarrow A$:
we cannot relate a *key* to a X *without* already knowing the A (determined by X).

Types of 3NF violations

If $X \longrightarrow A$ caused a violation of 3NF and *X is not a proper subset of any key* then there is a chain of dependencies *key* $\longrightarrow X$ and $X \longrightarrow A$:
we cannot relate a *key* to a X *without* already knowing the A (determined by X).

student

sid	name	age	birthdate	program	department
1	Alicia	21	August 27, 2000	COMPSCI	Comp. and Soft.
2	Bo	20	December 15, 2000	SFWRENG	Comp. and Soft.
3	Celeste	22	April 24, 1999	SFWRENG	Comp. and Soft.
4	Dafni	20	February 1, 2001	COMPSCI	Comp. and Soft.
5	Eva	23	July 2, 1998	COMPSCI	Comp. and Soft.
6	Frieda	21	August 27, 2000	CLASSICS	Classics

"birthdate $\longrightarrow$ age" and "program $\longrightarrow$ department".

We cannot relate a *student* to a program *without* already knowing its department.

Decomposition into 3NF: 3NF Synthesis

DECOMPOSE-3NF($\mathbf{R}$, $\mathfrak{S}$)

Compute a decomposition of $\mathbf{R}$ that is in 3NF and that is both lossless-join and dependency-preserving.

```
result := ∅.

return result.
```

Decomposition into 3NF: 3NF Synthesis

Decompose-3NF($\mathbf{R}$, $\mathfrak{S}$)

Compute a decomposition of $\mathbf{R}$ that is in 3NF and that is both lossless-join and dependency-preserving.

$\mathit{result} := \emptyset$.
cover := a minimal cover of $\mathfrak{S}$.

return result.

Decomposition into 3NF: 3NF Synthesis

DECOMPOSE-3NF($\mathbf{R}$, $\mathfrak{S}$)

Compute a decomposition of $\mathbf{R}$ that is in 3NF and that is both lossless-join and dependency-preserving.

```
result := ∅.
cover := a minimal cover of 𝔖.
for attributes A of R such that (A ⟶ X) ∈ cover do
    Let B = {Y | (A ⟶ Y) ∈ cover}.
    Add relational schema with attributes A ∪ B to result.



return result.
```

Decomposition into 3NF: 3NF Synthesis

Decompose-3NF($\mathbf{R}$, $\mathfrak{S}$)

Compute a decomposition of **R** that is in 3NF and that is both lossless-join and dependency-preserving.

result := $\emptyset$.
cover := a minimal cover of $\mathfrak{S}$.
for attributes A of **R** such that $(A \longrightarrow X) \in \mathit{cover}$ **do**
    Let $B = \{Y \mid (A \longrightarrow Y) \in \mathit{cover}\}$.
    Add relational schema with attributes $A \cup B$ to *result*.
if none of the schemas in *result* contain a key for **R** **then**
    Let *key* be the attributes of a key of **R**.
    Add relational schema with attributes *key* to *result*.

return *result*.

Decomposition into 3NF: 3NF Synthesis

DECOMPOSE-3NF($\mathbf{R}$, $\mathfrak{S}$)

Compute a decomposition of $\mathbf{R}$ that is in 3NF and that is both lossless-join and dependency-preserving.

result := $\emptyset$.
cover := a minimal cover of $\mathfrak{S}$.
for attributes A of $\mathbf{R}$ such that $(A \longrightarrow X) \in \mathit{cover}$ **do**
    Let $B = \{Y \mid (A \longrightarrow Y) \in \mathit{cover}\}$.
    Add relational schema with attributes $A \cup B$ to *result*.
if none of the schemas in *result* contain a key for $\mathbf{R}$ **then**
    Let *key* be the attributes of a key of $\mathbf{R}$.
    Add relational schema with attributes *key* to *result*.
while the attributes of $\mathbf{R}' \in \mathit{result}$ are a subset of another schema in *result* **do**
    Remove $\mathbf{R}'$ from *result*.
return *result*.

A first example of DECOMPOSE-3NF

student

sid	name	age	birthdate	program	department
1	Alicia	21	August 27, 2000	COMPSCI	Comp. and Soft.
2	Bo	20	December 15, 2000	SFWRENG	Comp. and Soft.
3	Celeste	22	April 24, 1999	SFWRENG	Comp. and Soft.
4	Dafni	20	February 1, 2001	COMPSCI	Comp. and Soft.
5	Eva	23	July 2, 1998	COMPSCI	Comp. and Soft.
6	Frieda	21	August 27, 2000	CLASSICS	Classics

- "birthdate $\longrightarrow$ age".
- "program $\longrightarrow$ department".
- "sid $\longrightarrow$ name, age, birthdate, program, department".

A first example of Decompose-3NF

student					
<u>sid</u>	name	age	birthdate	program	department

- "birthdate $\longrightarrow$ age".
- "program $\longrightarrow$ department".
- "sid $\longrightarrow$ name, age, birthdate, program, department".

$\mathit{result} = \{\}$.

Steps of Decompose-3NF($\mathbf{R}$, $\mathfrak{S}$)

1: $\mathit{result} := \emptyset$.

A first example of DECOMPOSE-3NF

student					
<u>sid</u>	name	age	birthdate	program	department

- "birthdate $\longrightarrow$ age".
- "program $\longrightarrow$ department".
- "sid $\longrightarrow$ name, age, birthdate, program, department".

$\mathit{result} = \{\}$.

Steps of DECOMPOSE-3NF($\mathbf{R}$, $\mathfrak{S}$)

2: cover := a minimal cover of $\mathfrak{S}$.

A first example of Decompose-3NF

student					
<u>sid</u>	name	age	birthdate	program	department

- "birthdate $\longrightarrow$ age".
- "program $\longrightarrow$ department".
- "sid $\longrightarrow$ x" with $x \in$ {name, ~~age~~, birthdate, program, ~~department~~}.

$result = \{\}$.

Steps of Decompose-3NF($\mathbf{R}$, $\mathfrak{S}$)

2: *cover* := a minimal cover of $\mathfrak{S}$.

A first example of Decompose-3NF

student					
<u>sid</u>	name	age	birthdate	program	department

- “birthdate $\longrightarrow$ age”.
- “program $\longrightarrow$ department”.
- “sid $\longrightarrow$ x” with $x \in \{$name, ~~age~~, birthdate, program, ~~department~~$\}$.

$result = \{\}$.

Steps of Decompose-3NF($\mathbf{R}$, $\mathfrak{S}$)

3: **for** attributes A of $\mathbf{R}$ such that $(A \longrightarrow X) \in cover$ **do**

4: Let $B = \{Y \mid (A \longrightarrow Y) \in cover\}$.

5: Add relational schema with attributes $A \cup B$ to *result*.

A first example of Decompose-3NF

student					
<u>sid</u>	name	age	birthdate	program	department

- "birthdate $\longrightarrow$ age".
- "program $\longrightarrow$ department".
- "sid $\longrightarrow x$" with $x \in \{$name, ~~age~~, birthdate, program, ~~department~~$\}$.

$\mathit{result} = \{(\text{birthdate}, \text{age})\}$.

Steps of Decompose-3NF($\mathbf{R}$, $\mathfrak{S}$)

3: **for** attributes A of $\mathbf{R}$ such that $(A \longrightarrow X) \in \mathit{cover}$ **do**
4: Let $B = \{Y \mid (A \longrightarrow Y) \in \mathit{cover}\}$.
5: Add relational schema with attributes $A \cup B$ to *result*.

A first example of Decompose-3NF

student					
<u>sid</u>	name	age	birthdate	program	department

- "birthdate $\longrightarrow$ age".
- "program $\longrightarrow$ department".
- "sid $\longrightarrow x$" with $x \in \{\text{name}, \cancel{\text{age}}, \text{birthdate}, \text{program}, \cancel{\text{department}}\}$.

$\mathit{result} = \{(\text{birthdate}, \text{age}), (\text{program}, \text{department})\}$.

Steps of Decompose-3NF($\mathbf{R}$, $\mathfrak{S}$)

3: **for** attributes A of $\mathbf{R}$ such that $(A \longrightarrow X) \in \mathit{cover}$ **do**

4: Let $B = \{Y \mid (A \longrightarrow Y) \in \mathit{cover}\}$.

5: Add relational schema with attributes $A \cup B$ to result.

A first example of DECOMPOSE-3NF

student					
<u>sid</u>	name	age	birthdate	program	department

- "birthdate $\longrightarrow$ age".
- "program $\longrightarrow$ department".
- "sid $\longrightarrow x$" with $x \in \{$name, $\cancel{\text{age}}$, birthdate, program, $\cancel{\text{department}}\}$.

$\mathit{result} = \{$(birthdate, age), (program, department), (sid, name, birthdate, program)$\}$.

Steps of DECOMPOSE-3NF($\mathbf{R}$, $\mathfrak{S}$)

3: **for** attributes A of $\mathbf{R}$ such that $(A \longrightarrow X) \in \mathit{cover}$ **do**

4: Let $B = \{Y \mid (A \longrightarrow Y) \in \mathit{cover}\}$.

5: Add relational schema with attributes $A \cup B$ to result.

A first example of DECOMPOSE-3NF

student					
<u>sid</u>	name	age	birthdate	program	department

- "birthdate $\longrightarrow$ age".
- "program $\longrightarrow$ department".
- "sid $\longrightarrow x$" with $x \in \{\text{name}, \cancel{\text{age}}, \text{birthdate}, \text{program}, \cancel{\text{department}}\}$.

$\mathit{result} = \{(\text{birthdate}, \text{age}), (\text{program}, \text{department}), (\text{sid}, \text{name}, \text{birthdate}, \text{program})\}$.

Steps of DECOMPOSE-3NF($\mathbf{R}$, $\mathfrak{S}$)

6: **if** none of the schemas in *result* contain a key for **R then**
7: Let *key* be the attributes of a key of **R**.
8: Add relational schema with attributes *key* to *result*.

A first example of DECOMPOSE-3NF

student					
<u>sid</u>	name	age	birthdate	program	department

- "birthdate $\longrightarrow$ age".
- "program $\longrightarrow$ department".
- "sid $\longrightarrow x$" with $x \in \{\text{name}, \cancel{\text{age}}, \text{birthdate}, \text{program}, \cancel{\text{department}}\}$.

$$result = \{(\text{birthdate}, \text{age}), (\text{program}, \text{department}), (\text{sid}, \text{name}, \text{birthdate}, \text{program})\}.$$

Steps of DECOMPOSE-3NF($\mathbf{R}$, $\mathfrak{S}$)

if none of the schemas in *result* contain a key for **R** **then**
    Let *key* be the attributes of a key of **R**.
    Add relational schema with attributes *key* to *result*.

A first example of DECOMPOSE-3NF

student					
<u>sid</u>	name	age	birthdate	program	department

- "birthdate $\longrightarrow$ age".
- "program $\longrightarrow$ department".
- "sid $\longrightarrow$ x" with $x \in$ {name, ~~age~~, birthdate, program, ~~department~~}.

result = {(birthdate, age), (program, department), (sid, name, birthdate, program)}.

Steps of DECOMPOSE-3NF($\mathbf{R}$, $\mathfrak{S}$)

9: **while** the attributes of $\mathbf{R}' \in$ *result* are a subset of another schema in *result* **do**
10: Remove $\mathbf{R}'$ from *result*.
11: **return** *result*.

A first example of DECOMPOSE-3NF

student					
<u>sid</u>	name	age	birthdate	program	department

- "birthdate $\longrightarrow$ age".
- "program $\longrightarrow$ department".
- "sid $\longrightarrow x$" with $x \in \{\text{name}, \cancel{\text{age}}, \text{birthdate}, \text{program}, \cancel{\text{department}}\}$.

$$result = \{(\text{birthdate}, \text{age}), (\text{program}, \text{department}), (\text{sid}, \text{name}, \text{birthdate}, \text{program})\}.$$

date_info

<u>birthdate</u>	age
August 27, 2000	21
December 15, 2000	20
April 24, 1999	22
February 1, 2001	20
July 2, 1998	23

prog_dept

<u>program</u>	department
COMPSCI	Comp. and Soft.
SFWRENG	Comp. and Soft.
CLASSICS	Classics

student

<u>sid</u>	name	birthdate	program
1	Alicia	August 27, 2000	COMPSCI
2	Bo	December 15, 2000	SFWRENG
3	Celeste	April 24, 1999	SFWRENG
4	Dafni	February 1, 2001	COMPSCI
5	Eva	July 2, 1998	COMPSCI
6	Frieda	August 27, 2000	CLASSICS

A second example of Decompose-3NF

$$\mathbf{r}(A, B, C, D, E)$$

$$\{A \longrightarrow BCD, BC \longrightarrow DE, B \longrightarrow D, D \longrightarrow A\}$$

A second example of Decompose-3NF

$$\mathbf{r}(A, B, C, D, E)$$

$$\{A \longrightarrow BCD, BC \longrightarrow DE, B \longrightarrow D, D \longrightarrow A\}$$

$result = \{\}.$

Steps of Decompose-3NF($\mathbf{R}$, $\mathfrak{S}$)

1: $result := \emptyset.$

A second example of DECOMPOSE-3NF

$$\mathbf{r}(A, B, C, D, E)$$

$$\{A \longrightarrow BCD, BC \longrightarrow DE, B \longrightarrow D, D \longrightarrow A\}$$

$\mathit{result} = \{\}$.

Steps of DECOMPOSE-3NF($\mathbf{R}$, $\mathfrak{S}$)

2: *cover* := a minimal cover of $\mathfrak{S}$.

A second example of DECOMPOSE-3NF

$$\mathbf{r}(A, B, C, D, E)$$

$$\{A \longrightarrow BCD, BC \longrightarrow DE, B \longrightarrow D, D \longrightarrow A\}$$

$result = \{\}$.

Steps of DECOMPOSE-3NF($\mathbf{R}$, $\mathfrak{S}$)

2: *cover* := a minimal cover of $\mathfrak{S}$.

$$\{A \longrightarrow B, A \longrightarrow C, A \longrightarrow D, BC \longrightarrow D, BC \longrightarrow E, B \longrightarrow D, D \longrightarrow A\}$$

A second example of DECOMPOSE-3NF

$$\mathbf{r}(A, B, C, D, E)$$

$$\{A \longrightarrow BCD, BC \longrightarrow DE, B \longrightarrow D, D \longrightarrow A\}$$

$result = \{\}$.

Steps of DECOMPOSE-3NF($\mathbf{R}$, $\mathfrak{S}$)

2: *cover* := a minimal cover of $\mathfrak{S}$.

$$\{A \longrightarrow B, A \longrightarrow C, A \longrightarrow D, \cancel{BC \longrightarrow D}, BC \longrightarrow E, B \longrightarrow D, D \longrightarrow A\}$$

A second example of Decompose-3NF

$$\mathbf{r}(A, B, C, D, E)$$

$$\{A \longrightarrow BCD, BC \longrightarrow DE, B \longrightarrow D, D \longrightarrow A\}$$

$result = \{\}$.

Steps of Decompose-3NF($\mathbf{R}$, $\mathfrak{S}$)

2: *cover* := a minimal cover of $\mathfrak{S}$.

$$\{A \longrightarrow B, A \longrightarrow C, \cancel{A \longrightarrow D}, \cancel{BC \longrightarrow D}, BC \longrightarrow E, B \longrightarrow D, D \longrightarrow A\}$$

A second example of DECOMPOSE-3NF

$$\mathbf{r}(A, B, C, D, E)$$

$$\{A \longrightarrow B, A \longrightarrow C, BC \longrightarrow E, B \longrightarrow D, D \longrightarrow A\}$$

$result = \{\}$.

Steps of DECOMPOSE-3NF($\mathbf{R}$, $\mathfrak{S}$)

2: $cover$:= a minimal cover of $\mathfrak{S}$.

$$\{A \longrightarrow B, A \longrightarrow C, \cancel{A \longrightarrow D}, \cancel{BC \longrightarrow D}, BC \longrightarrow E, B \longrightarrow D, D \longrightarrow A\}$$

A second example of Decompose-3NF

$$\mathbf{r}(A, B, C, D, E)$$

$$\{A \longrightarrow B, A \longrightarrow C, BC \longrightarrow E, B \longrightarrow D, D \longrightarrow A\}$$

$result = \{ABC\}$.

Steps of Decompose-3NF($\mathbf{R}$, $\mathfrak{S}$)

3: **for** attributes A of $\mathbf{R}$ such that $(A \longrightarrow X) \in cover$ **do**
4: Let $B = \{Y \mid (A \longrightarrow Y) \in cover\}$.
5: Add relational schema with attributes $A \cup B$ to $result$.

A second example of DECOMPOSE-3NF

$$\mathbf{r}(A, B, C, D, E)$$

$$\{A \longrightarrow B, A \longrightarrow C, BC \longrightarrow E, B \longrightarrow D, D \longrightarrow A\}$$

$\mathit{result} = \{ABC, BCE\}$.

Steps of DECOMPOSE-3NF($\mathbf{R}$, $\mathfrak{S}$)

for attributes A of $\mathbf{R}$ such that $(A \longrightarrow X) \in \mathit{cover}$ **do**
    Let $B = \{Y \mid (A \longrightarrow Y) \in \mathit{cover}\}$.
    Add relational schema with attributes $A \cup B$ to *result*.

A second example of DECOMPOSE-3NF

$$\mathbf{r}(A, B, C, D, E)$$

$$\{A \longrightarrow B, A \longrightarrow C, BC \longrightarrow E, B \longrightarrow D, D \longrightarrow A\}$$

$result = \{ABC, BCE, BD\}$.

Steps of DECOMPOSE-3NF($\mathbf{R}$, $\mathfrak{S}$)

3: **for** attributes A of $\mathbf{R}$ such that $(A \longrightarrow X) \in cover$ **do**

4: Let $B = \{Y \mid (A \longrightarrow Y) \in cover\}$.

5: Add relational schema with attributes $A \cup B$ to *result*.

A second example of Decompose-3NF

$$\mathbf{r}(A, B, C, D, E)$$

$$\{A \longrightarrow B, A \longrightarrow C, BC \longrightarrow E, B \longrightarrow D, D \longrightarrow A\}$$

$\mathit{result} = \{ABC, BCE, BD, DA\}.$

Steps of Decompose-3NF($\mathbf{R}$, $\mathfrak{S}$)

3: **for** attributes A of $\mathbf{R}$ such that $(A \longrightarrow X) \in \mathit{cover}$ **do**
4: Let $B = \{Y \mid (A \longrightarrow Y) \in \mathit{cover}\}$.
5: Add relational schema with attributes $A \cup B$ to *result*.

A second example of DECOMPOSE-3NF

$$\mathbf{r}(A, B, C, D, E)$$

$$\{A \longrightarrow B, A \longrightarrow C, BC \longrightarrow E, B \longrightarrow D, D \longrightarrow A\}$$

$\mathit{result} = \{ABC, BCE, BD, DA\}$.

Steps of DECOMPOSE-3NF($\mathbf{R}$, $\mathfrak{S}$)

6: **if** none of the schemas in *result* contain a key for **R then**
7: Let *key* be the attributes of a key of **R**.
8: Add relational schema with attributes *key* to *result*.

A second example of DECOMPOSE-3NF

$$\mathbf{r}(A, B, C, D, E)$$

$$\{A \longrightarrow B, A \longrightarrow C, BC \longrightarrow E, B \longrightarrow D, D \longrightarrow A\}$$

$\mathit{result} = \{ABC, BCE, BD, DA\}.$

Steps of DECOMPOSE-3NF($\mathbf{R}$, $\mathfrak{S}$)

if none of the schemas in *result* contain a key for **R then**
    Let *key* be the attributes of a key of **R**.
    Add relational schema with attributes *key* to *result*.

A second example of Decompose-3NF

$$\mathbf{r}(A, B, C, D, E)$$

$$\{A \longrightarrow B, A \longrightarrow C, BC \longrightarrow E, B \longrightarrow D, D \longrightarrow A\}$$

$result = \{ABC, BCE, BD, DA\}$.

Steps of Decompose-3NF($\mathbf{R}$, $\mathfrak{S}$)

while the attributes of $\mathbf{R}' \in result$ are a subset of another schema in *result* **do**
    Remove $\mathbf{R}'$ from *result*.
return *result*.

A third example of Decompose-3NF

$$\mathbf{r}(A, B, C, D, E)$$

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

A third example of Decompose-3NF

$$\mathbf{r}(A, B, C, D, E)$$

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

$result = \{\}$.

Steps of Decompose-3NF($\mathbf{R}$, $\mathfrak{S}$)

1: $result := \emptyset$.

A third example of Decompose-3NF

$$\mathbf{r}(A, B, C, D, E)$$

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

$result = \{\}$.

Steps of Decompose-3NF($\mathbf{R}$, $\mathfrak{S}$)

2: $cover$:= a minimal cover of $\mathfrak{S}$.

A third example of DECOMPOSE-3NF

$$\mathbf{r}(A, B, C, D, E)$$

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

$result = \{\}$.

Steps of DECOMPOSE-3NF($\mathbf{R}$, $\mathfrak{S}$)

for attributes A of $\mathbf{R}$ such that $(A \longrightarrow X) \in cover$ **do**
    Let $B = \{Y \mid (A \longrightarrow Y) \in cover\}$.
    Add relational schema with attributes $A \cup B$ to *result*.

A third example of Decompose-3NF

$$\mathbf{r}(A, B, C, D, E)$$

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

$result = \{AB, DE\}$.

Steps of Decompose-3NF($\mathbf{R}$, $\mathfrak{S}$)

```
for attributes A of R such that (A ⟶ X) ∈ cover do
    Let B = {Y | (A ⟶ Y) ∈ cover}.
    Add relational schema with attributes A ∪ B to result.
```

A third example of Decompose-3NF

$$\mathbf{r}(A, B, C, D, E)$$

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

$\mathit{result} = \{AB, DE\}$.

Steps of Decompose-3NF($\mathbf{R}$, $\mathfrak{S}$)

```
if none of the schemas in result contain a key for R then
    Let key be the attributes of a key of R.
    Add relational schema with attributes key to result.
```

A third example of Decompose-3NF

$$\mathbf{r}(A, B, C, D, E)$$

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

$result = \{AB, DE\}$.

Steps of Decompose-3NF($\mathbf{R}$, $\mathfrak{S}$)

6: **if** none of the schemas in *result* contain a key for **R then**
7: Let *key* be the attributes of a key of **R**.
8: Add relational schema with attributes *key* to *result*.

We are missing C altogether!

A third example of Decompose-3NF

$$\mathbf{r}(A, B, C, D, E)$$

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

$result = \{AB, DE, ACD\}$.

Steps of Decompose-3NF($\mathbf{R}$, $\mathfrak{S}$)

```
if none of the schemas in result contain a key for R then
    Let key be the attributes of a key of R.
    Add relational schema with attributes key to result.
```

A third example of Decompose-3NF

$$\mathbf{r}(A, B, C, D, E)$$

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

$\mathit{result} = \{AB, DE, ACD\}$.

Steps of Decompose-3NF($\mathbf{R}$, $\mathfrak{S}$)

while the attributes of $\mathbf{R}' \in \mathit{result}$ are a subset of another schema in *result* **do**
    Remove $\mathbf{R}'$ from *result*.
return *result*.

A fourth example of DECOMPOSE-3NF

$$\mathbf{r}(A, B, C)$$

$$\{AC \longrightarrow B, B \longrightarrow C\}$$

A fourth example of DECOMPOSE-3NF

$$\mathbf{r}(A, B, C)$$

$$\{AC \longrightarrow B, B \longrightarrow C\}$$

$result = \{\}$.

Steps of DECOMPOSE-3NF($\mathbf{R}$, $\mathfrak{S}$)

1: $result := \emptyset$.

A fourth example of Decompose-3NF

$$\mathbf{r}(A, B, C)$$

$$\{AC \longrightarrow B, B \longrightarrow C\}$$

$\mathit{result} = \{\}$.

Steps of Decompose-3NF($\mathbf{R}$, $\mathfrak{S}$)

2: *cover* := a minimal cover of $\mathfrak{S}$.

A fourth example of Decompose-3NF

$$\mathbf{r}(A, B, C)$$

$$\{AC \longrightarrow B, B \longrightarrow C\}$$

$result = \{\}$.

Steps of Decompose-3NF($\mathbf{R}$, $\mathfrak{S}$)

for attributes A of $\mathbf{R}$ such that $(A \longrightarrow X) \in cover$ **do**
  Let $B = \{Y \mid (A \longrightarrow Y) \in cover\}$.
  Add relational schema with attributes $A \cup B$ to *result*.

A fourth example of DECOMPOSE-3NF

$$\mathbf{r}(A, B, C)$$

$$\{AC \longrightarrow B, B \longrightarrow C\}$$

$result = \{ABC, BC\}$.

Steps of DECOMPOSE-3NF($\mathbf{R}$, $\mathfrak{S}$)

for attributes A of $\mathbf{R}$ such that $(A \longrightarrow X) \in cover$ **do**
    Let $B = \{Y \mid (A \longrightarrow Y) \in cover\}$.
    Add relational schema with attributes $A \cup B$ to *result*.

A fourth example of DECOMPOSE-3NF

$$\mathbf{r}(A, B, C)$$

$$\{AC \longrightarrow B, B \longrightarrow C\}$$

$\mathit{result} = \{ABC, BC\}.$

Steps of DECOMPOSE-3NF($\mathbf{R}$, $\mathfrak{S}$)

6: **if** none of the schemas in *result* contain a key for **R** **then**
7: Let *key* be the attributes of a key of **R**.
8: Add relational schema with attributes *key* to *result*.

A fourth example of Decompose-3NF

$$\mathbf{r}(A, B, C)$$

$$\{AC \longrightarrow B, B \longrightarrow C\}$$

$result = \{ABC, BC\}$.

Steps of Decompose-3NF($\mathbf{R}$, $\mathfrak{S}$)

if none of the schemas in *result* contain a key for **R then**
    Let *key* be the attributes of a key of **R**.
    Add relational schema with attributes *key* to *result*.

A fourth example of Decompose-3NF

$$\mathbf{r}(A, B, C)$$

$$\{AC \longrightarrow B, B \longrightarrow C\}$$

result = $\{ABC, BC\}$.

Steps of Decompose-3NF($\mathbf{R}$, $\mathfrak{S}$)

while the attributes of $\mathbf{R}' \in$ *result* are a subset of another schema in *result* **do**
    Remove $\mathbf{R}'$ from *result*.

A fourth example of DECOMPOSE-3NF

$$\mathbf{r}(A, B, C)$$

$$\{AC \longrightarrow B, B \longrightarrow C\}$$

$result = \{ABC, BC\}$.

Steps of DECOMPOSE-3NF($\mathbf{R}$, $\mathfrak{S}$)

while the attributes of $\mathbf{R}' \in$ *result* are a subset of another schema in *result* **do**
    Remove $\mathbf{R}'$ from *result*.

A fourth example of DECOMPOSE-3NF

$$\mathbf{r}(A, B, C)$$

$$\{AC \longrightarrow B, B \longrightarrow C\}$$

$result = \{ABC\}$.

Steps of DECOMPOSE-3NF($\mathbf{R}$, $\mathfrak{S}$)

while the attributes of $\mathbf{R}' \in$ *result* are a subset of another schema in *result* **do**
    Remove $\mathbf{R}'$ from *result*.

A fourth example of Decompose-3NF

$$\mathbf{r}(A, B, C)$$

$$\{AC \longrightarrow B, B \longrightarrow C\}$$

$result = \{ABC\}$.

Steps of Decompose-3NF($\mathbf{R}$, $\mathfrak{S}$)

11: **return** *result*.

Redundancies in 3NF

$$\mathbf{r}(A, B, C)$$

$$\{AC \longrightarrow B, B \longrightarrow C\}$$

Redundancies in 3NF

$$\textbf{course\_info}(\text{code}, \text{instructor}, \text{department})$$

$$\{\text{“code,department} \longrightarrow \text{instructor”}, \text{“instructor} \longrightarrow \text{department”}\}$$

Redundancies in 3NF

course_info(code, instructor, department)

$$\{\text{“code,department} \longrightarrow \text{instructor”}, \text{“instructor} \longrightarrow \text{department”}\}$$

course_info

code	instructor	department
1	Alicia	Computing and Software
2	Alicia	Computing and Software
3	Bo	Chemical Engineering
4	Bo	Chemical Engineering
5	Celeste	Classics

The Boyce-Codd Normal Form (BCNF)

Definition

A relational schema $\mathbf{R}$ is in *Boyce-Codd normal form* with respect to functional dependencies $\mathfrak{S}$ if
it is in 1NF and if, for every $(X \longrightarrow A) \in \mathfrak{S}^+$, the following holds:

- $A \subseteq X$ (the dependency is trivial); or
- X is a (super)key.

The Boyce-Codd Normal Form (BCNF)

Definition

A relational schema **R** is in *Boyce-Codd normal form* with respect to functional dependencies $\mathfrak{S}$ if
it is in 1NF and if, for every $(X \longrightarrow A) \in \mathfrak{S}^+$, the following holds:

- $A \subseteq X$ (the dependency is trivial); or
- X is a (super)key.

3NF is *almost* BCNF.
BCNF misses the *exception* “each attribute in $A \setminus X$ is part of a key of **R**”.

The Boyce-Codd Normal Form (BCNF)

Definition

A relational schema $\mathbf{R}$ is in *Boyce-Codd normal form* with respect to functional dependencies $\mathfrak{S}$ if
it is in 1NF and if, for every $(X \longrightarrow A) \in \mathfrak{S}^+$, the following holds:

- $A \subseteq X$ (the dependency is trivial); or
- X is a (super)key.

3NF is *almost* BCNF.
BCNF misses the *exception* "each attribute in $A \setminus X$ is part of a key of $\mathbf{R}$".

All relational schemas in BCNF are in 3NF.

Claim: All binary relations are in BCNF

Proof

Let **R** be a binary relational scheme with two attributes, A and B.

(proof details)

Hence, we cannot have a BCNF violation in **R**.

Claim: All binary relations are in BCNF

Proof

Let **R** be a binary relational scheme with two attributes, A and B.

Consider all *possible* functional dependencies:

Hence, we cannot have a BCNF violation in **R**.

Claim: All binary relations are in BCNF

Proof

Let **R** be a binary relational scheme with two attributes, A and B.

Consider all *possible* functional dependencies:

$A \longrightarrow B$: A must be a *key*;

Hence, we cannot have a BCNF violation in **R**.

Claim: All binary relations are in BCNF

Proof

Let **R** be a binary relational scheme with two attributes, A and B.

Consider all *possible* functional dependencies:

$A \longrightarrow B$: A must be a *key*;

$B \longrightarrow A$: B must be a *key*;

Hence, we cannot have a BCNF violation in **R**.

Claim: All binary relations are in BCNF

Proof

Let **R** be a binary relational scheme with two attributes, A and B.

Consider all *possible* functional dependencies:

$A \longrightarrow B$: A must be a *key*;

$B \longrightarrow A$: B must be a *key*;

all other dependencies are *trivial* (e.g., $A \longrightarrow A$ or $AB \longrightarrow B$).

Hence, we cannot have a BCNF violation in **R**.

Decomposition into BCNF

DECOMPOSE-BCNF($\mathbf{R}$, $\mathfrak{S}$)

Compute a decomposition of $\mathbf{R}$ that is in BCNF and that is lossless-join.

1: **if** $\mathbf{R}$ is in BCNF **then**
2:     **return** $\{\mathbf{R}\}$.

Decomposition into BCNF

DECOMPOSE-BCNF($\mathbf{R}$, $\mathfrak{S}$)

Compute a decomposition of $\mathbf{R}$ that is in BCNF and that is lossless-join.

if $\mathbf{R}$ is in BCNF **then**
    return $\{\mathbf{R}\}$.
else
    Let $(X \longrightarrow A) \in \mathfrak{S}$ be a BCNF violation for $\mathbf{R}$.
    Let $\mathbf{R}_1 = X^+$ and $\mathbf{R}_2 = X \cup Z$ with Z all attributes of $\mathbf{R}$ not in X^+.
    Let $\mathfrak{S}_i$ be all functional dependencies that hold in $\mathbf{R}_i$, $i \in \{1, 2\}$.
    return DECOMPOSE-BCNF($\mathbf{R}_1$, $\mathfrak{S}_1$) $\cup$ DECOMPOSE-BCNF($\mathbf{R}_2$, $\mathfrak{S}_2$).

Decomposition into BCNF

DECOMPOSE-BCNF($\mathbf{R}$, $\mathfrak{S}$)

Compute a decomposition of $\mathbf{R}$ that is in BCNF and that is lossless-join.

if $\mathbf{R}$ is in BCNF **then**
    return $\{\mathbf{R}\}$.
else
    Let $(X \longrightarrow A) \in \mathfrak{S}$ be a BCNF violation for $\mathbf{R}$.
    Let $\mathbf{R}_1 = X^+$ and $\mathbf{R}_2 = X \cup Z$ with Z all attributes of $\mathbf{R}$ not in X^+.
    Let $\mathfrak{S}_i$ be all functional dependencies that hold in $\mathbf{R}_i$, $i \in \{1, 2\}$:

$$\mathfrak{S}_i := \{(Y \longrightarrow B) \in \mathfrak{S}^+ \mid \text{all } (Y \cup B) \text{ are attributes of } \mathbf{R}_i\}.$$

    return DECOMPOSE-BCNF($\mathbf{R}_1$, $\mathfrak{S}_1$) $\cup$ DECOMPOSE-BCNF($\mathbf{R}_2$, $\mathfrak{S}_2$).

A first example of Decompose-BCNF

$$\mathbf{r}(A, B, C)$$

$$\{AC \longrightarrow B, B \longrightarrow C\}$$

A first example of Decompose-BCNF

$$\mathbf{r}(A, B, C)$$

$$\{AC \longrightarrow B, B \longrightarrow C\}$$

Steps of Decompose-BCNF($\mathbf{R}$, $\mathfrak{S}$)

4: Let $(X \longrightarrow A) \in \mathfrak{S}$ be a BCNF violation for $\mathbf{R}$.
5: Let $\mathbf{R}_1 = X^+$ and $\mathbf{R}_2 = X \cup Z$ with Z all attributes of $\mathbf{R}$ not in X^+.
6: Let $\mathfrak{S}_i$ be all functional dependencies that hold in $\mathbf{R}_i$, $i \in \{1, 2\}$.
7: **return** Decompose-BCNF($\mathbf{R}_1$, $\mathfrak{S}_1$) $\cup$ Decompose-BCNF($\mathbf{R}_2$, $\mathfrak{S}_2$).

A first example of DECOMPOSE-BCNF

$$\mathbf{r}(A, B, C)$$

$$\{AC \longrightarrow B, B \longrightarrow C\}$$

Steps of DECOMPOSE-BCNF($\mathbf{R}$, $\mathfrak{S}$)

Let $(X \longrightarrow A) \in \mathfrak{S}$ be a BCNF violation for $\mathbf{R}$.
Let $\mathbf{R}_1 = X^+$ and $\mathbf{R}_2 = X \cup Z$ with Z all attributes of $\mathbf{R}$ not in X^+.
Let $\mathfrak{S}_i$ be all functional dependencies that hold in $\mathbf{R}_i$, $i \in \{1, 2\}$.
return DECOMPOSE-BCNF($\mathbf{R}_1$, $\mathfrak{S}_1$) $\cup$ DECOMPOSE-BCNF($\mathbf{R}_2$, $\mathfrak{S}_2$).

A first example of Decompose-BCNF

$$\mathbf{r}(A, B, C)$$

$$\{AC \longrightarrow B, B \longrightarrow C\}$$

Steps of Decompose-BCNF($\mathbf{R}$, $\mathfrak{S}$)

4: Let $(X \longrightarrow A) \in \mathfrak{S}$ be a BCNF violation for $\mathbf{R}$.
5: Let $\mathbf{R}_1 = X^+$ and $\mathbf{R}_2 = X \cup Z$ with Z all attributes of $\mathbf{R}$ not in X^+.
6: Let $\mathfrak{S}_i$ be all functional dependencies that hold in $\mathbf{R}_i$, $i \in \{1, 2\}$.
7: **return** Decompose-BCNF($\mathbf{R}_1$, $\mathfrak{S}_1$) $\cup$ Decompose-BCNF($\mathbf{R}_2$, $\mathfrak{S}_2$).

A first example of Decompose-BCNF

$$\mathbf{r}(A, B, C)$$

$$\{AC \longrightarrow B, B \longrightarrow C\}$$

Steps of Decompose-BCNF($\mathbf{R}$, $\mathfrak{S}$)

Let $(X \longrightarrow A) \in \mathfrak{S}$ be a BCNF violation for $\mathbf{R}$.
Let $\mathbf{R}_1 = X^+$ and $\mathbf{R}_2 = X \cup Z$ with Z all attributes of $\mathbf{R}$ not in X^+.
Let $\mathfrak{S}_i$ be all functional dependencies that hold in $\mathbf{R}_i$, $i \in \{1, 2\}$.
return Decompose-BCNF($\mathbf{R}_1$, $\mathfrak{S}_1$) $\cup$ Decompose-BCNF($\mathbf{R}_2$, $\mathfrak{S}_2$).

$$\mathbf{R}_1 = (BC), \mathbf{R}_2 = (BA).$$

A first example of Decompose-BCNF

$$\mathbf{r}(A, B, C)$$

$$\{AC \longrightarrow B, B \longrightarrow C\}$$

Steps of Decompose-BCNF($\mathbf{R}$, $\mathfrak{S}$)

4: Let $(X \longrightarrow A) \in \mathfrak{S}$ be a BCNF violation for $\mathbf{R}$.
5: Let $\mathbf{R}_1 = X^+$ and $\mathbf{R}_2 = X \cup Z$ with Z all attributes of $\mathbf{R}$ not in X^+.
6: Let $\mathfrak{S}_i$ be all functional dependencies that hold in $\mathbf{R}_i$, $i \in \{1, 2\}$.
7: **return** Decompose-BCNF($\mathbf{R}_1$, $\mathfrak{S}_1$) $\cup$ Decompose-BCNF($\mathbf{R}_2$, $\mathfrak{S}_2$).

$$\mathbf{R}_1 = (BC), \mathbf{R}_2 = (BA).$$

3NF versus BCNF

course_info(code, instructor, department)

{"code,department $\longrightarrow$ instructor", "instructor $\longrightarrow$ department"}

3NF

course_info

code	instructor	department
1	Alicia	Comp. and Soft.
2	Alicia	Comp. and Soft.
3	Bo	Chem. Eng.
4	Bo	Chem. Eng.
5	Celeste	Classics

versus

BCNF

course_instr

code	instructor
1	Alicia
2	Alicia
3	Bo
4	Bo
5	Celeste

instr_dep

instructor	department
Alicia	Comp. and Soft.
Bo	Chem. Eng.
Celeste	Classics

3NF versus BCNF

course_info(code, instructor, department)

{“code,department ⟶ instructor”, “instructor ⟶ department”}

3NF

course_info

code	instructor	department
1	Alicia	Comp. and Soft.
2	Alicia	Comp. and Soft.
3	Bo	Chem. Eng.
4	Bo	Chem. Eng.
5	Celeste	Classics

redundancy

versus

BCNF

course_instr

code	instructor
1	Alicia
2	Alicia
3	Bo
4	Bo
5	Celeste

instr_dep

instructor	department
Alicia	Comp. and Soft.
Bo	Chem. Eng.
Celeste	Classics

does not preserve dependencies

Dependency-preserving decompositions for functional dependencies

Consider a relational schema $\mathbf{R}$ decomposed into schemas $\mathbf{R}_1, \ldots, \mathbf{R}_n$.
Let $\mathfrak{S}$ be the functional dependencies that hold in $\mathbf{R}$.

Definition

The decomposition $\mathbf{R}_1, \ldots, \mathbf{R}_n$ of $\mathbf{R}$ is *dependency-preserving* if all constraints on $\mathbf{R}$ can be maintained using only constraints on the individual relational schemas $\mathbf{R}_1, \ldots, \mathbf{R}_n$.

Dependency-preserving decompositions for functional dependencies

Consider a relational schema $\mathbf{R}$ decomposed into schemas $\mathbf{R}_1, \ldots, \mathbf{R}_n$.
Let $\mathfrak{S}$ be the functional dependencies that hold in $\mathbf{R}$.

Definition

The decomposition $\mathbf{R}_1, \ldots, \mathbf{R}_n$ of $\mathbf{R}$ is *dependency-preserving* if all constraints on $\mathbf{R}$ can be maintained using only constraints on the individual relational schemas $\mathbf{R}_1, \ldots, \mathbf{R}_n$.

Which functional dependencies hold in $\mathbf{R}_1, \ldots, \mathbf{R}_n$?

Dependency-preserving decompositions for functional dependencies

Consider a relational schema $\mathbf{R}$ decomposed into schemas $\mathbf{R}_1, \ldots, \mathbf{R}_n$.
Let $\mathfrak{S}$ be the functional dependencies that hold in $\mathbf{R}$.

Definition

The decomposition $\mathbf{R}_1, \ldots, \mathbf{R}_n$ of $\mathbf{R}$ is *dependency-preserving* if all constraints on $\mathbf{R}$ can be maintained using only constraints on the individual relational schemas $\mathbf{R}_1, \ldots, \mathbf{R}_n$.

Which functional dependencies hold in $\mathbf{R}_1, \ldots, \mathbf{R}_n$?

The *projection* of a set of functional dependencies S of $\mathbf{R}$ onto relational schema $\mathbf{R}'$ is:

$$\{(X \longrightarrow Y) \in S \mid \text{all attributes in } X \cup Y \text{ are attributes of } \mathbf{R}'\}.$$

Dependency-preserving decompositions for functional dependencies

Consider a relational schema $\mathbf{R}$ decomposed into schemas $\mathbf{R}_1, \dots, \mathbf{R}_n$.
Let $\mathfrak{S}$ be the functional dependencies that hold in $\mathbf{R}$.

Definition

The decomposition $\mathbf{R}_1, \dots, \mathbf{R}_n$ of $\mathbf{R}$ is *dependency-preserving* if all constraints on $\mathbf{R}$ can be maintained using only constraints on the individual relational schemas $\mathbf{R}_1, \dots, \mathbf{R}_n$.

Which functional dependencies hold in $\mathbf{R}_1, \dots, \mathbf{R}_n$?

The *projection* of a set of functional dependencies S of $\mathbf{R}$ onto relational schema $\mathbf{R}'$ is:

$$\{(X \longrightarrow Y) \in S \mid \text{all attributes in } X \cup Y \text{ are attributes of } \mathbf{R}'\}.$$

In $\mathbf{R}_i$, $1 \leq i \leq n$, the projection of $\mathfrak{S}^+$ onto $\mathbf{R}_i$ hold.

Dependency-preserving decompositions for functional dependencies

Consider a relational schema $\mathbf{R}$ decomposed into schemas $\mathbf{R}_1, \dots, \mathbf{R}_n$.
Let $\mathfrak{S}$ be the functional dependencies that hold in $\mathbf{R}$.

Which functional dependencies hold in $\mathbf{R}_1, \dots, \mathbf{R}_n$?

The *projection* of a set of functional dependencies S of $\mathbf{R}$ onto relational schema $\mathbf{R}'$ is:

$$\{(X \longrightarrow Y) \in S \mid \text{all attributes in } X \cup Y \text{ are attributes of } \mathbf{R}'\}.$$

In $\mathbf{R}_i$, $1 \leq i \leq n$, the projection of $\mathfrak{S}^+$ onto $\mathbf{R}_i$ hold.

Definition (for functional dependencies)

The decomposition $\mathbf{R}_1, \dots, \mathbf{R}_n$ of $\mathbf{R}$ is *dependency-preserving* if $\mathfrak{S}^+ = (\mathfrak{S}_1 \cup \dots \cup \mathfrak{S}_n)^+$ with $\mathfrak{S}_1, \dots, \mathfrak{S}_n$ the projection of $\mathfrak{S}^+$ onto the attributes of $\mathbf{R}_1, \dots, \mathbf{R}_n$.

Dependencies in the first example of Decompose-BCNF

$$\begin{aligned}
\text{Original schema } \mathbf{R} &: (\text{code}, \text{instructor}, \text{department});\\
\text{dependencies } \mathfrak{S} \text{ in } \mathbf{R} &: \{\text{“code,department} \longrightarrow \text{instructor”},\\
&\quad\ \text{“instructor} \longrightarrow \text{department”}\},
\end{aligned}$$

$$\text{Result of } \textsc{Decompose-BCNF} : \mathbf{R}_1 = (\text{code}, \text{instructor}), \mathbf{R}_2 = (\text{instructor}, \text{department}).$$

Dependencies in the first example of Decompose-BCNF

$$\begin{aligned}
\text{Original schema } \mathbf{R} &: (\text{code}, \text{instructor}, \text{department}); \\
\text{dependencies } \mathfrak{S} \text{ in } \mathbf{R} &: \{\text{“code,department} \longrightarrow \text{instructor”}, \\
&\quad \text{“instructor} \longrightarrow \text{department”}\},
\end{aligned}$$

$$\text{Result of Decompose-BCNF} : \mathbf{R}_1 = (\text{code}, \text{instructor}), \mathbf{R}_2 = (\text{instructor}, \text{department}).$$

$$\begin{aligned}
\text{Dependencies } \mathfrak{S}_1 \text{ in } \mathbf{R}_1 &: \emptyset \text{ (minimal cover)}. \\
\text{Dependencies } \mathfrak{S}_2 \text{ in } \mathbf{R}_2 &: \text{“instructor} \longrightarrow \text{department” (minimal cover)}.
\end{aligned}$$

Dependencies in the first example of Decompose-BCNF

$$
\begin{aligned}
\text{Original schema } \mathbf{R} &: (\text{code}, \text{instructor}, \text{department}); \\
\text{dependencies } \mathfrak{S} \text{ in } \mathbf{R} &: \{\text{“code,department} \longrightarrow \text{instructor”}, \\
&\quad \text{“instructor} \longrightarrow \text{department”}\},
\end{aligned}
$$

$$
\text{Result of Decompose-BCNF} : \mathbf{R}_1 = (\text{code}, \text{instructor}), \mathbf{R}_2 = (\text{instructor}, \text{department}).
$$

$$
\begin{aligned}
\text{Dependencies } \mathfrak{S}_1 \text{ in } \mathbf{R}_1 &: \emptyset \text{ (minimal cover)}. \\
\text{Dependencies } \mathfrak{S}_2 \text{ in } \mathbf{R}_2 &: \text{“instructor} \longrightarrow \text{department” (minimal cover)}.
\end{aligned}
$$

$$
(\mathfrak{S}_1 \cup \mathfrak{S}_2)^+ : \text{“instructor} \longrightarrow \text{department” (minimal cover)}.
$$

Dependencies in the first example of Decompose-BCNF

$$\begin{aligned}
\text{Original schema } \mathbf{R} &: (\text{code}, \text{instructor}, \text{department});\\
\text{dependencies } \mathfrak{S} \text{ in } \mathbf{R} &: \{\text{“code,department} \longrightarrow \text{instructor”},\\
&\quad \text{“instructor} \longrightarrow \text{department”}\},
\end{aligned}$$

$$\text{Result of } \textsc{Decompose-BCNF} : \mathbf{R}_1 = (\text{code}, \text{instructor}), \mathbf{R}_2 = (\text{instructor}, \text{department}).$$

$$\begin{aligned}
\text{Dependencies } \mathfrak{S}_1 \text{ in } \mathbf{R}_1 &: \emptyset \text{ (minimal cover)}.\\
\text{Dependencies } \mathfrak{S}_2 \text{ in } \mathbf{R}_2 &: \text{“instructor} \longrightarrow \text{department” (minimal cover)}.
\end{aligned}$$

$$(\mathfrak{S}_1 \cup \mathfrak{S}_2)^+ : \text{“instructor} \longrightarrow \text{department” (minimal cover)}.$$

We lost “code,department $\longrightarrow$ instructor” in the decomposition!

A second example of DECOMPOSE-BCNF

student

<u>sid</u>	name	age	birthdate	program	department
1	Alicia	21	August 27, 2000	COMPSCI	Comp. and Soft.
2	Bo	20	December 15, 2000	SFWRENG	Comp. and Soft.
3	Celeste	22	April 24, 1999	SFWRENG	Comp. and Soft.
4	Dafni	20	February 1, 2001	COMPSCI	Comp. and Soft.
5	Eva	23	July 2, 1998	COMPSCI	Comp. and Soft.
6	Frieda	21	August 27, 2000	CLASSICS	Classics

- "birthdate $\longrightarrow$ age".
- "program $\longrightarrow$ department".
- "sid $\longrightarrow$ name, age, birthdate, program, department".

A second example of Decompose-BCNF

student					
<u>sid</u>	name	age	birthdate	program	department

- "birthdate $\longrightarrow$ age".
- "program $\longrightarrow$ department".
- "sid $\longrightarrow$ name, age, birthdate, program, department".

Steps of Decompose-BCNF($\mathbf{R}$, $\mathfrak{S}$)

Let $(X \longrightarrow A) \in \mathfrak{S}$ be a BCNF violation for $\mathbf{R}$.
Let $\mathbf{R}_1 = X^+$ and $\mathbf{R}_2 = X \cup Z$ with Z all attributes of $\mathbf{R}$ not in X^+.
Let $\mathfrak{S}_i$ be all functional dependencies that hold in $\mathbf{R}_i$, $i \in \{1, 2\}$.
return Decompose-BCNF($\mathbf{R}_1$, $\mathfrak{S}_1$) $\cup$ Decompose-BCNF($\mathbf{R}_2$, $\mathfrak{S}_2$).

A second example of Decompose-BCNF

student					
<u>sid</u>	name	age	birthdate	program	department

- “birthdate $\longrightarrow$ age”.
- “program $\longrightarrow$ department”.
- “sid $\longrightarrow$ name, age, birthdate, program, department”.

Steps of Decompose-BCNF($\mathbf{R}$, $\mathfrak{S}$)

“Find violating $X \longrightarrow A$ (X not a superkey), split off X^+, recurse.”

$$\mathbf{R} = (\underline{\text{sid}}, \text{name}, \text{age}, \text{birthdate}, \text{program}, \text{department})$$

A second example of DECOMPOSE-BCNF

student					
<u>sid</u>	name	age	birthdate	program	department

- "birthdate $\longrightarrow$ age".
- "program $\longrightarrow$ department".
- "sid $\longrightarrow$ name, age, birthdate, program, department".

Steps of DECOMPOSE-BCNF($\mathbf{R}$, $\mathfrak{S}$)

"Find violating $X \longrightarrow A$ (X not a superkey), split off X^+, recurse."

$$\mathbf{R} = (\underline{\text{sid}}, \text{name}, \text{age}, \text{birthdate}, \text{program}, \text{department})$$

A second example of Decompose-BCNF

student					
<u>sid</u>	name	age	birthdate	program	department

- "birthdate $\longrightarrow$ age".
- "program $\longrightarrow$ department".
- "sid $\longrightarrow$ name, age, birthdate, program, department".

Steps of Decompose-BCNF($\mathbf{R}$, $\mathfrak{S}$)

"Find violating $X \longrightarrow A$ (X not a superkey), split off X^+, recurse."

$\mathbf{R}$ = (<u>sid</u>, name, age, birthdate, program, department)

birthdate $\longrightarrow$ age

$\mathbf{R}_1$ = (birthdate, age) $\qquad$ $\mathbf{R}_2$ = (sid, name, birthdate, program, department)

A second example of Decompose-BCNF

student					
<u>sid</u>	name	age	birthdate	program	department

- "birthdate $\longrightarrow$ age".
- "program $\longrightarrow$ department".
- "sid $\longrightarrow$ name, age, birthdate, program, department".

Steps of Decompose-BCNF($\mathbf{R}$, $\mathfrak{S}$)

"Find violating $X \longrightarrow A$ (X not a superkey), split off X^+, recurse."

$\mathbf{R}$ = (<u>sid</u>, name, age, birthdate, program, department)

birthdate $\longrightarrow$ age

$\mathbf{R}_1$ = (birthdate, age) $\quad$ $\mathbf{R}_2$ = (sid, name, birthdate, program, department)

A second example of Decompose-BCNF

student

sid	name	age	birthdate	program	department

- "birthdate $\longrightarrow$ age".
- "program $\longrightarrow$ department".
- "sid $\longrightarrow$ name, age, birthdate, program, department".

Steps of Decompose-BCNF($\mathbf{R}$, $\mathfrak{S}$)

"Find violating $X \longrightarrow A$ (X not a superkey), split off X^+, recurse."

$\mathbf{R}$ = ($\underline{\text{sid}}$, name, age, birthdate, program, department)

birthdate $\longrightarrow$ age

$\mathbf{R}_1$ = (birthdate, age)

$\mathbf{R}_2$ = (sid, name, birthdate, program, department)

program $\longrightarrow$ department

$\mathbf{R}_{2,1}$ = (program, department)

$\mathbf{R}_{2,2}$ = (sid, name, birthdate, program)

A second example of Decompose-BCNF

student					
<u>sid</u>	name	age	birthdate	program	department

- "birthdate $\longrightarrow$ age".
- "program $\longrightarrow$ department".
- "sid $\longrightarrow$ name, age, birthdate, program, department".

R_1

<u>birthdate</u>	age
August 27, 2000	21
December 15, 2000	20
April 24, 1999	22
February 1, 2001	20
July 2, 1998	23

$R_{2,1}$

<u>program</u>	department
COMPSCI	Comp. and Soft.
SFWRENG	Comp. and Soft.
CLASSICS	Classics

$R_{2,2}$

<u>sid</u>	name	birthdate	program
1	Alicia	August 27, 2000	COMPSCI
2	Bo	December 15, 2000	SFWRENG
3	Celeste	April 24, 1999	SFWRENG
4	Dafni	February 1, 2001	COMPSCI
5	Eva	July 2, 1998	COMPSCI
6	Frieda	August 27, 2000	CLASSICS

A third example of Decompose-BCNF

$$\mathbf{r}(A, B, C, D, E)$$

$$\{A \longrightarrow BCD, BC \longrightarrow DE, B \longrightarrow D, D \longrightarrow A\}$$

Steps of Decompose-BCNF($\mathbf{R}$, $\mathfrak{S}$)

"Find violating $X \longrightarrow A$ (X not a superkey), split off X^+, recurse."

A third example of Decompose-BCNF

$$\mathbf{r}(A, B, C, D, E)$$

$$\{A \longrightarrow BCD, BC \longrightarrow DE, B \longrightarrow D, D \longrightarrow A\}$$

Steps of Decompose-BCNF($\mathbf{R}$, $\mathfrak{S}$)

"Find violating $X \longrightarrow A$ (X not a superkey), split off X^+, recurse."

- A, B, and D are keys!

A third example of Decompose-BCNF

$$\mathbf{r}(A, B, C, D, E)$$

$$\{A \longrightarrow BCD, BC \longrightarrow DE, B \longrightarrow D, D \longrightarrow A\}$$

Steps of Decompose-BCNF($\mathbf{R}$, $\mathfrak{S}$)

"Find violating $X \longrightarrow A$ (X not a superkey), split off X^+, recurse."

- A, B, and D are keys!
- BC is a superkey!

A third example of Decompose-BCNF

$$\mathbf{r}(A, B, C, D, E)$$

$$\{A \longrightarrow BCD, BC \longrightarrow DE, B \longrightarrow D, D \longrightarrow A\}$$

Steps of Decompose-BCNF($\mathbf{R}$, $\mathfrak{S}$)

"Find violating $X \longrightarrow A$ (X not a superkey), split off X^+, recurse."

- A, B, and D are keys!
- BC is a superkey!

This schema is already in BCNF! No steps taken.

Decompose-3NF yielded (A, B, C), (B, C, E), (B, D), (D, A)!

A fourth example of Decompose-BCNF

$$\mathbf{r}(A, B, C, D, E, F)$$

$$\{A \longrightarrow BCD, BC \longrightarrow DE, B \longrightarrow D, D \longrightarrow A\}$$

Steps of Decompose-BCNF($\mathbf{R}$, $\mathfrak{S}$)

"Find violating $X \longrightarrow A$ (X not a superkey), split off X^+, recurse."

$$\mathbf{R} = (A, B, C, D, E, F)$$

A fourth example of Decompose-BCNF

$$\mathbf{r}(A, B, C, D, E, F)$$

$$\{A \longrightarrow BCD, BC \longrightarrow DE, B \longrightarrow D, D \longrightarrow A\}$$

Steps of Decompose-BCNF($\mathbf{R}$, $\mathfrak{S}$)

"Find violating $X \longrightarrow A$ (X not a superkey), split off X^+, recurse."

$$\begin{tikzcd}
 & \mathbf{R} = (A, B, C, D, E, F) \arrow[dl, "A \longrightarrow BCD"'] \arrow[dr] & \\
\mathbf{R}_1 = (A, B, C, D, E) & & \mathbf{R}_2 = (A, F)
\end{tikzcd}$$

A fourth example of DECOMPOSE-BCNF

$$\mathbf{r}(A, B, C, D, E, F)$$

$$\{A \longrightarrow BCD, BC \longrightarrow DE, B \longrightarrow D, D \longrightarrow A\}$$

Steps of DECOMPOSE-BCNF($\mathbf{R}$, $\mathfrak{S}$)

"Find violating $X \longrightarrow A$ (X not a superkey), split off X^+, recurse."

$$\begin{array}{ccc} & \mathbf{R} = (A, B, C, D, E, F) & \\ \swarrow{\scriptstyle BC \longrightarrow DE} & & \searrow \\ \mathbf{R}_1 = (A, B, C, D, E) & & \mathbf{R}_2 = (B, C, F) \end{array}$$

A fourth example of Decompose-BCNF

$$\mathbf{r}(A, B, C, D, E, F)$$

$$\{A \longrightarrow BCD, BC \longrightarrow DE, B \longrightarrow D, D \longrightarrow A\}$$

Steps of Decompose-BCNF($\mathbf{R}$, $\mathfrak{S}$)

"Find violating $X \longrightarrow A$ (X not a superkey), split off X^+, recurse."

$$\begin{tikzcd}
& \mathbf{R} = (A, B, C, D, E, F) \arrow[dl, "B \longrightarrow D"'] \arrow[dr] & \\
\mathbf{R}_1 = (A, B, C, D, E) & & \mathbf{R}_2 = (B, F)
\end{tikzcd}$$

A fourth example of Decompose-BCNF

$$\mathbf{r}(A, B, C, D, E, F)$$

$$\{A \longrightarrow BCD, BC \longrightarrow DE, B \longrightarrow D, D \longrightarrow A\}$$

Steps of Decompose-BCNF($\mathbf{R}$, $\mathfrak{S}$)

"Find violating $X \longrightarrow A$ (X not a superkey), split off X^+, recurse."

$$\begin{tikzcd}
 & \mathbf{R} = (A, B, C, D, E, F) \arrow[dl, "D \longrightarrow E"'] \arrow[dr] & \\
\mathbf{R}_1 = (A, B, C, D, E) & & \mathbf{R}_2 = (D, F)
\end{tikzcd}$$

A fifth example of DECOMPOSE-BCNF

$$\mathbf{r}(A, B, C, D, E, F)$$

$$\{A \longrightarrow BC, BD \longrightarrow E, F \longrightarrow B, FB \longrightarrow D\}$$

Steps of DECOMPOSE-BCNF($\mathbf{R}$, $\mathfrak{S}$)

"Find violating $X \longrightarrow A$ (X not a superkey), split off X^+, recurse."

$$\mathbf{R} = (A, B, C, D, E, F)$$

A fifth example of Decompose-BCNF

$$\mathbf{r}(A, B, C, D, E, F)$$

$$\{A \longrightarrow BC, BD \longrightarrow E, F \longrightarrow B, FB \longrightarrow D\}$$

Steps of Decompose-BCNF($\mathbf{R}$, $\mathfrak{S}$)

"Find violating $X \longrightarrow A$ (X not a superkey), split off X^+, recurse."

$$\begin{tikzcd}
 & \mathbf{R} = (A, B, C, D, E, F) \arrow[dl, "A \longrightarrow BC"'] \arrow[dr] & \\
\mathbf{R}_1 = (A, B, C) & & \mathbf{R}_2 = (A, D, E, F)
\end{tikzcd}$$

A fifth example of Decompose-BCNF

$$\mathbf{r}(A, B, C, D, E, F)$$

$$\{A \longrightarrow BC, BD \longrightarrow E, F \longrightarrow B, FB \longrightarrow D\}$$

Steps of Decompose-BCNF($\mathbf{R}$, $\mathfrak{S}$)

"Find violating $X \longrightarrow A$ (X not a superkey), split off X^+, recurse."

$$\begin{tikzcd}
& \mathbf{R} = (A, B, C, D, E, F) \arrow[dl, "A \longrightarrow BC"'] \arrow[dr] & \\
\mathbf{R}_1 = (A, B, C) & & \mathbf{R}_2 = (A, D, E, F)
\end{tikzcd}$$

$(F \longrightarrow D) \in \mathfrak{S}^+!$

A fifth example of Decompose-BCNF

$$\mathbf{r}(A, B, C, D, E, F)$$

$$\{A \longrightarrow BC, BD \longrightarrow E, F \longrightarrow B, FB \longrightarrow D\}$$

Steps of Decompose-BCNF($\mathbf{R}$, $\mathfrak{S}$)

"Find violating $X \longrightarrow A$ (X not a superkey), split off X^+, recurse."

$$\begin{tikzcd}
 & \mathbf{R} = (A, B, C, D, E, F) \arrow[dl, "A \longrightarrow BC"'] \arrow[dr] & & \\
\mathbf{R}_1 = (A, B, C) & & \mathbf{R}_2 = (A, D, E, F) \arrow[dl, "F \longrightarrow D"'] \arrow[dr] & \\
 & \mathbf{R}_{2,1} = (F, D, E) & & \mathbf{R}_{2,2} = (F, A)
\end{tikzcd}$$

$(F \longrightarrow D) \in \mathfrak{S}^+!$

A fifth example of DECOMPOSE-BCNF

$$\mathbf{r}(A, B, C, D, E, F)$$

$$\{A \longrightarrow BC, BD \longrightarrow E, F \longrightarrow B, FB \longrightarrow D\}$$

Steps of DECOMPOSE-BCNF($\mathbf{R}$, $\mathfrak{S}$)

"Find violating $X \longrightarrow A$ (X not a superkey), split off X^+, recurse."

$$\begin{tikzcd}
 & \mathbf{R} = (A, B, C, D, E, F) \arrow[dl, "A \longrightarrow BC"'] \arrow[dr] & & \\
\mathbf{R}_1 = (A, B, C) & & \mathbf{R}_2 = (A, D, E, F) \arrow[dl, "F \longrightarrow D"'] \arrow[dr] & (F \longrightarrow D) \in \mathfrak{S}^+! \\
F^+ = \{F, D, E\}! & \mathbf{R}_{2,1} = (F, D, E) & & \mathbf{R}_{2,2} = (F, A)
\end{tikzcd}$$

Redundancies in BCNF

BCNF does not look at multivalued dependencies

course_details

course	student	TA
Programming	Celeste	Alicia
Programming	Frieda	Alicia
Programming	Celeste	Dafni
Programming	Frieda	Dafni
Databases	Bo	Eva
Databases	Dafni	Eva
Databases	Bo	Alicia
Databases	Dafni	Alicia

“course $\twoheadrightarrow$ student” and “course $\twoheadrightarrow$ TA”.

The Fourth Normal Form (4NF)

Definition

A relational schema $\mathbf{R}$ is in *fourth normal form* with respect to multivalued dependencies $\mathfrak{S}$ if it is in 1NF and if, for every $(X \twoheadrightarrow A) \in \mathfrak{S}^+$, the following holds:

- $A \subseteq X$, or A and X are all attributes in $\mathbf{R}$ (the dependency is trivial); or
- X is a (super)key.

The Fourth Normal Form (4NF)

Definition

A relational schema $\mathbf{R}$ is in *fourth normal form* with respect to multivalued dependencies $\mathfrak{S}$ if it is in 1NF and if, for every $(X \twoheadrightarrow A) \in \mathfrak{S}^+$, the following holds:

- $A \subseteq X$, or A and X are all attributes in $\mathbf{R}$ (the dependency is trivial); or
- X is a (super)key.

Replication: if $X \longrightarrow Y$, then $X \twoheadrightarrow Y$.

The Fourth Normal Form (4NF)

Definition

A relational schema **R** is in *fourth normal form* with respect to multivalued dependencies $\mathfrak{S}$ if it is in 1NF and if, for every $(X \longrightarrow\!\!\!\!\rightarrow A) \in \mathfrak{S}^+$, the following holds:

- $A \subseteq X$, or A and X are all attributes in **R** (the dependency is trivial); or
- X is a (super)key.

Replication: if $X \longrightarrow Y$, then $X \longrightarrow\!\!\!\!\rightarrow Y$.

BCNF is *almost* 4NF.
4NF extends the restrictions of BCNF to multivalued dependencies.

The Fourth Normal Form (4NF)

Definition

A relational schema **R** is in *fourth normal form* with respect to multivalued dependencies $\mathfrak{S}$ if it is in 1NF and if, for every $(X \twoheadrightarrow A) \in \mathfrak{S}^+$, the following holds:

- $A \subseteq X$, or A and X are all attributes in **R** (the dependency is trivial); or
- X is a (super)key.

Replication: if $X \longrightarrow Y$, then $X \twoheadrightarrow Y$.

BCNF is *almost* 4NF.
4NF extends the restrictions of BCNF to multivalued dependencies.

All relational schemas in 4NF are in BCNF.

Many relation schemas in BCNF are in 4NF

A relational schema **R** is guaranteed to be in 4NF if:

- **R** has at-most two attributes.

A relational schema **R** in 3NF is guaranteed to be in 4NF if:

- every key of **R** is a single-attribute key.

A relational schema **R** in BCNF is guaranteed to be in 4NF if:

- if **R** has a single-attribute key.

Decomposition into 4NF

DECOMPOSE-4NF($\mathbf{R}$, $\mathfrak{S}$)

Compute a decomposition of $\mathbf{R}$ that is in 4NF and that is lossless-join.

if $\mathbf{R}$ is in 4NF **then**
  return $\{\mathbf{R}\}$.

Decomposition into 4NF

DECOMPOSE-4NF($\mathbf{R}$, $\mathfrak{S}$)

Compute a decomposition of $\mathbf{R}$ that is in 4NF and that is lossless-join.

1: **if** $\mathbf{R}$ is in 4NF **then**
2: **return** $\{\mathbf{R}\}$.
3: **else**
4: Let $(X \twoheadrightarrow A) \in \mathfrak{S}^+$ be a 4NF violation for $\mathbf{R}$.
5: Let $\mathbf{R}_1 = X \cup A$ and $\mathbf{R}_2 = X \cup Z$ with Z all attributes of $\mathbf{R}$ not in A.
6: Let $\mathfrak{S}_i$ be all multivalued dependencies that hold in $\mathbf{R}_i$, $i \in \{1, 2\}$:

$$\mathfrak{S}_i := \{(Y \twoheadrightarrow B) \in \mathfrak{S}^+ \mid \text{all } (Y \cup B) \text{ are attributes of } \mathbf{R}_i\}.$$

7: **return** DECOMPOSE-4NF($\mathbf{R}_1$, $\mathfrak{S}_1$) $\cup$ DECOMPOSE-4NF($\mathbf{R}_2$, $\mathfrak{S}_2$).

Note: *every* step of DECOMPOSE-BCNF is a *valid step* in this algorithm.

A first example of Decompose-4NF

course_details		
course	student	TA
Programming	Celeste	Alicia
Programming	Frieda	Alicia
Programming	Celeste	Dafni
Programming	Frieda	Dafni
Databases	Bo	Eva
Databases	Dafni	Eva
Databases	Bo	Alicia
Databases	Dafni	Alicia

- “course $\twoheadrightarrow$ student”.
- “course $\twoheadrightarrow$ TA”.

A first example of Decompose-4NF

course_details		
course	student	TA

- "course $\twoheadrightarrow$ student".
- "course $\twoheadrightarrow$ TA".

Steps of Decompose-4NF($\mathbf{R}$, $\mathfrak{S}$)

4: Let $(X \twoheadrightarrow A) \in \mathfrak{S}$ be a 4NF violation for $\mathbf{R}$.
5: Let $\mathbf{R}_1 = X \cup A$ and $\mathbf{R}_2 = X \cup Z$ with Z all attributes of $\mathbf{R}$ not in A.
6: Let $\mathfrak{S}_i$ be all multivalued dependencies that hold in $\mathbf{R}_i$, $i \in \{1, 2\}$.
7: **return** Decompose-4NF($\mathbf{R}_1$, $\mathfrak{S}_1$) $\cup$ Decompose-4NF($\mathbf{R}_2$, $\mathfrak{S}_2$).

A first example of Decompose-4NF

course_details		
course	student	TA

- "course $\twoheadrightarrow$ student".
- "course $\twoheadrightarrow$ TA".

Steps of Decompose-4NF($\mathbf{R}$, $\mathfrak{S}$)

4: Let $(X \twoheadrightarrow A) \in \mathfrak{S}$ be a 4NF violation for $\mathbf{R}$.
5: Let $\mathbf{R}_1 = X \cup A$ and $\mathbf{R}_2 = X \cup Z$ with Z all attributes of $\mathbf{R}$ not in A.
6: Let $\mathfrak{S}_i$ be all multivalued dependencies that hold in $\mathbf{R}_i$, $i \in \{1, 2\}$.
7: **return** Decompose-4NF($\mathbf{R}_1$, $\mathfrak{S}_1$) $\cup$ Decompose-4NF($\mathbf{R}_2$, $\mathfrak{S}_2$).

A first example of DECOMPOSE-4NF

course_details		
course	student	TA

- "course $\twoheadrightarrow$ student".
- "course $\twoheadrightarrow$ TA".

Steps of DECOMPOSE-4NF($\mathbf{R}$, $\mathfrak{S}$)

4: Let $(X \twoheadrightarrow A) \in \mathfrak{S}$ be a 4NF violation for $\mathbf{R}$.
5: Let $\mathbf{R}_1 = X \cup A$ and $\mathbf{R}_2 = X \cup Z$ with Z all attributes of $\mathbf{R}$ not in A.
6: Let $\mathfrak{S}_i$ be all multivalued dependencies that hold in $\mathbf{R}_i$, $i \in \{1, 2\}$.
7: **return** DECOMPOSE-4NF($\mathbf{R}_1$, $\mathfrak{S}_1$) $\cup$ DECOMPOSE-4NF($\mathbf{R}_2$, $\mathfrak{S}_2$).

$$\mathbf{R}_1 = (\text{course}, \text{student}), \mathbf{R}_2 = (\text{course}, \text{TA}).$$

A first example of Decompose-4NF

course_details		
course	student	TA

- "course $\twoheadrightarrow$ student".
- "course $\twoheadrightarrow$ TA".

Steps of Decompose-4NF($\mathbf{R}$, $\mathfrak{S}$)

4: Let $(X \twoheadrightarrow A) \in \mathfrak{S}$ be a 4NF violation for $\mathbf{R}$.
5: Let $\mathbf{R}_1 = X \cup A$ and $\mathbf{R}_2 = X \cup Z$ with Z all attributes of $\mathbf{R}$ not in A.
6: Let $\mathfrak{S}_i$ be all multivalued dependencies that hold in $\mathbf{R}_i$, $i \in \{1, 2\}$.
7: **return** Decompose-4NF($\mathbf{R}_1$, $\mathfrak{S}_1$) $\cup$ Decompose-4NF($\mathbf{R}_2$, $\mathfrak{S}_2$).

$$\mathbf{R}_1 = (\text{course}, \text{student}), \mathbf{R}_2 = (\text{course}, \text{TA}).$$

A first example of DECOMPOSE-4NF

course_details		
course	student	TA

- "course $\twoheadrightarrow$ student".
- "course $\twoheadrightarrow$ TA".

course_students	
course	TA
Programming	Celeste
Programming	Frieda
Databases	Bo
Databases	Dafni

course_TAs	
course	TA
Programming	Alicia
Programming	Dafni
Databases	Eva
Databases	Alicia

A second example of Decompose-4NF

all_course_details			
course	student	TA	Instructor
Databases	Bo	Eva	Celeste
Databases	Dafni	Eva	Celeste
Databases	Bo	Alicia	Celeste
Databases	Dafni	Alicia	Celeste
Databases	Bo	Eva	Frieda
Databases	Dafni	Eva	Frieda
Databases	Bo	Alicia	Frieda
Databases	Dafni	Alicia	Frieda

$$\{\text{course} \twoheadrightarrow \text{student}, \text{course} \twoheadrightarrow \text{TA}, \text{course} \twoheadrightarrow \text{Instructor}\}$$

A second example of DECOMPOSE-4NF

all_course_details			
course	student	TA	Instructor

$$\{\text{course} \twoheadrightarrow \text{student}, \text{course} \twoheadrightarrow \text{TA}, \text{course} \twoheadrightarrow \text{Instructor}\}$$

Steps of DECOMPOSE-4NF($\mathbf{R}$, $\mathfrak{S}$)

"Find violating $X \twoheadrightarrow A$ (X not a superkey), split off $X \cup A$, recurse."

$$\begin{tikzcd}
 & \mathbf{R} = (C, S, T, I) \arrow[dl, "\text{course} \twoheadrightarrow \text{student}"'] \arrow[dr] & \\
\mathbf{R}_1 = (C, S) & & \mathbf{R}_2 = (C, T, I)
\end{tikzcd}$$

A second example of Decompose-4NF

all_course_details			
course	student	TA	Instructor

$$\{\text{course} \twoheadrightarrow \text{student}, \text{course} \twoheadrightarrow \text{TA}, \text{course} \twoheadrightarrow \text{Instructor}\}$$

Steps of Decompose-4NF($\mathbf{R}$, $\mathfrak{S}$)

"Find violating $X \twoheadrightarrow A$ (X not a superkey), split off $X \cup A$, recurse."

$$\begin{tikzcd}[column sep=small]
 & \mathbf{R} = (C, S, T, I) \arrow[dl, "\text{course} \twoheadrightarrow \text{student}"'] \arrow[dr] & & \\
\mathbf{R}_1 = (C, S) & & \mathbf{R}_2 = (C, T, I) \arrow[dl, "\text{course} \twoheadrightarrow \text{TA}"'] \arrow[dr] & \\
 & \mathbf{R}_{2,1} = (C, T) & & \mathbf{R}_{2,2} = (C, I)
\end{tikzcd}$$

A second example of Decompose-4NF

all_course_details			
course	student	TA	Instructor

$$\{\text{course} \twoheadrightarrow \text{student}, \text{course} \twoheadrightarrow \text{TA}, \text{course} \twoheadrightarrow \text{Instructor}\}$$

course_students

course	student
Databases	Bo
Databases	Dafni

course_TAs

course	TA
Databases	Eva
Databases	Alicia

course_instructors

course	instructor
Databases	Celeste
Databases	Frieda

Multivalued dependencies and normal forms

$$\mathbf{r}(A, B, C)$$

$$\{A \longrightarrow BC, B \longrightarrow\!\!\!\!\!\rightarrow C\}$$

Question: Is **r** in BCNF?

Vote at https://strawpoll.com/bfuszwggk.
Or: go to https://strawpoll.live and use the code **269641**.

Multivalued dependencies and normal forms

$$\mathbf{r}(A, B, C)$$

$$\{A \longrightarrow BC, B \twoheadrightarrow C\}$$

Which functional dependencies hold in $\mathbf{r}$?

We have $\{A \longrightarrow BC, B \twoheadrightarrow C\} \models B \longrightarrow C$:

Multivalued dependencies and normal forms

$$\mathbf{r}(A, B, C)$$

$$\{A \longrightarrow BC, B \twoheadrightarrow C\}$$

Which functional dependencies hold in $\mathbf{r}$?

We have $\{A \longrightarrow BC, B \twoheadrightarrow C\} \models B \longrightarrow C$:

- Apply the Decomposition rule on $A \longrightarrow BC$ to derive $A \longrightarrow C$.

Multivalued dependencies and normal forms

$$\mathbf{r}(A, B, C)$$

$$\{A \longrightarrow BC, B \twoheadrightarrow C\}$$

Which functional dependencies hold in $\mathbf{r}$?

We have $\{A \longrightarrow BC, B \twoheadrightarrow C\} \models B \longrightarrow C$:

- Apply the Decomposition rule on $A \longrightarrow BC$ to derive $A \longrightarrow C$.
- We have $\{C\} \cap \{A\} = \emptyset$ and $\{C\} \subseteq \{C\}$.

Multivalued dependencies and normal forms

$$\mathbf{r}(A, B, C)$$

$$\{A \longrightarrow BC, B \twoheadrightarrow C\}$$

Which functional dependencies hold in $\mathbf{r}$?

We have $\{A \longrightarrow BC, B \twoheadrightarrow C\} \models B \longrightarrow C$:

- Apply the Decomposition rule on $A \longrightarrow BC$ to derive $A \longrightarrow C$.
- We have $\{C\} \cap \{A\} = \emptyset$ and $\{C\} \subseteq \{C\}$.
- Hence, we can apply Coalescence rule on $B \twoheadrightarrow C$ and $A \longrightarrow C$ to derive $B \longrightarrow C$.

Multivalued dependencies and normal forms

$$\mathbf{r}(A, B, C)$$

$$\{A \longrightarrow BC, B \twoheadrightarrow C\}$$

Steps of $\textsc{Decompose-BCNF}(\mathbf{R}, \mathfrak{S})$

"Find violating $X \longrightarrow A$ (X not a superkey), split off X^+, recurse."

$$\mathbf{R} = (A, B, C)$$

Multivalued dependencies and normal forms

$$\mathbf{r}(A, B, C)$$

$$\{A \longrightarrow BC, B \twoheadrightarrow C\}$$

Steps of Decompose-BCNF($\mathbf{R}$, $\mathfrak{S}$)

"Find violating $X \longrightarrow A$ (X not a superkey), split off X^+, recurse."

$(B \longrightarrow C) \in \mathfrak{S}^+!$

$$\mathbf{R} = (A, B, C)$$

Multivalued dependencies and normal forms

$$\mathbf{r}(A, B, C)$$

$$\{A \longrightarrow BC, B \twoheadrightarrow C\}$$

Steps of DECOMPOSE-BCNF($\mathbf{R}$, $\mathfrak{S}$)

"Find violating $X \longrightarrow A$ (X not a superkey), split off X^+, recurse."

$$(B \longrightarrow C) \in \mathfrak{S}^+!$$

$$\begin{tikzcd}
 & \mathbf{R} = (A, B, C) \arrow[dl, "B \longrightarrow C"'] \arrow[dr] & \\
\mathbf{R}_{2,1} = (B, C) & & \mathbf{R}_{2,2} = (A, B)
\end{tikzcd}$$

Proof (sketch): Decompose-3NF is dependency-preserving

Decompose-3NF($\mathbf{R}$, $\mathfrak{S}$)

Compute a decomposition of **R** that is in 3NF and that is both lossless-join and dependency-preserving.

result := $\emptyset$.
cover := a minimal cover of $\mathfrak{S}$.
for attributes A of **R** such that $(A \longrightarrow X) \in$ *cover* **do**
    Let $B = \{Y \mid (A \longrightarrow Y) \in \mathit{cover}\}$.
    Add relational schema with attributes $A \cup B$ to *result*.
if none of the schemas in *result* contain a key for **R** **then**
    Let *key* be the attributes of a key of **R**.
    Add relational schema with attributes *key* to *result*.
while the attributes of $\mathbf{R}' \in$ *result* are a subset of another schema in *result* **do**
    Remove $\mathbf{R}'$ from *result*.
return *result*.

Proof (sketch): Decompose-3NF is dependency-preserving

Decompose-3NF($\mathbf{R}$, $\mathfrak{S}$)

Compute a decomposition of $\mathbf{R}$ that is in 3NF and that is both lossless-join and dependency-preserving.

result := $\emptyset$.
cover := a minimal cover of $\mathfrak{S}$.
for attributes A of $\mathbf{R}$ such that $(A \longrightarrow X) \in$ *cover* **do**
    Let $B = \{Y \mid (A \longrightarrow Y) \in \mathit{cover}\}$.
    Add relational schema with attributes $A \cup B$ to *result*.
if none of the schemas in *result* contain a key for $\mathbf{R}$ **then**
    Let *key* be the attributes of a key of $\mathbf{R}$.
    Add relational schema with attributes *key* to *result*.
while the attributes of $\mathbf{R}' \in$ *result* are a subset of another schema in *result* **do**
    Remove $\mathbf{R}'$ from *result*.
return *result*.

Create a relational schema for each $A \longrightarrow X$ in the *minimal cover*.

Proof (sketch): Decompose-3NF is dependency-preserving

Decompose-3NF($\mathbf{R}$, $\mathfrak{S}$)

Compute a decomposition of **R** that is in 3NF and that is both lossless-join and dependency-preserving.

```
result := ∅.
cover := a minimal cover of 𝔖.
for attributes A of R such that (A ⟶ X) ∈ cover do
    Let B = {Y | (A ⟶ Y) ∈ cover}.
    Add relational schema with attributes A ∪ B to result.
if none of the schemas in result contain a key for R then
    Let key be the attributes of a key of R.
    Add relational schema with attributes key to result.
while the attributes of R′ ∈ result are a subset of another schema in result do
    Remove R′ from result.
return result.
```

Create a relational schema for each $A \longrightarrow X$ in the *minimal cover*.

Remove redundant schemas.

Proof (sketch): Decompose-x, $x \in \{\text{3NF}, \text{BCNF}\}$, is lossless-join

Let $\mathbf{R}$ be a relational schema and $\mathfrak{S}$ a set of functional dependencies that hold over $\mathbf{R}$.
Let $\mathbf{R}_1$ and $\mathbf{R}_2$ be a decomposition of $\mathbf{R}$.

Theorem

The decomposition of $\mathbf{R}$ into $\mathbf{R}_1$ and $\mathbf{R}_2$ is lossless-join if there exists an $(A \longrightarrow B) \in \mathfrak{S}^+$ with:

- *A a (super)key of either $\mathbf{R}_1$ or $\mathbf{R}_2$; and*
- *the attributes in A are exactly those attributes common to $\mathbf{R}_1$ and $\mathbf{R}_2$.*

Proof (sketch): DECOMPOSE-x, $x \in \{3NF, BCNF\}$, is lossless-join

Let $\mathbf{R}$ be a relational schema and $\mathfrak{S}$ a set of functional dependencies that hold over $\mathbf{R}$.
Let $\mathbf{R}_1$ and $\mathbf{R}_2$ be a decomposition of $\mathbf{R}$.

Theorem

The decomposition of $\mathbf{R}$ into $\mathbf{R}_1$ and $\mathbf{R}_2$ is lossless-join if there exists an $(A \longrightarrow B) \in \mathfrak{S}^+$ with:

- *A a (super)key of either $\mathbf{R}_1$ or $\mathbf{R}_2$; and*
- *the attributes in A are exactly those attributes common to $\mathbf{R}_1$ and $\mathbf{R}_2$.*

Proof.

Assume $(A \longrightarrow B) \in \mathfrak{S}^+$ with A a (super)key of $\mathbf{R}_1$. We must have

$$\mathbf{R} = \quad \pi_{\mathbf{R}_1}(\mathbf{R}) \quad \bowtie \quad \pi_{\mathbf{R}_2}(\mathbf{R}) \quad .$$

Proof (sketch): DECOMPOSE-x, $x \in \{3NF, BCNF\}$, is lossless-join

Let $\mathbf{R}$ be a relational schema and $\mathfrak{S}$ a set of functional dependencies that hold over $\mathbf{R}$.
Let $\mathbf{R}_1$ and $\mathbf{R}_2$ be a decomposition of $\mathbf{R}$.

Theorem

The decomposition of $\mathbf{R}$ into $\mathbf{R}_1$ and $\mathbf{R}_2$ is lossless-join if there exists an $(A \longrightarrow B) \in \mathfrak{S}^+$ with:

- *A a (super)key of either $\mathbf{R}_1$ or $\mathbf{R}_2$; and*
- *the attributes in A are exactly those attributes common to $\mathbf{R}_1$ and $\mathbf{R}_2$.*

Proof.

Assume $(A \longrightarrow B) \in \mathfrak{S}^+$ with A a (super)key of $\mathbf{R}_1$. We must have

$$\mathbf{R} = \underbrace{\pi_{\mathbf{R}_1}(\mathbf{R})}_{\text{A \textit{single} value per } A} \bowtie \underbrace{\pi_{\mathbf{R}_2}(\mathbf{R})}_{\text{\textit{many} values per } A}.$$

Proof (sketch): DECOMPOSE-x, $x \in \{3NF, BCNF\}$, is lossless-join

Let $\mathbf{R}$ be a relational schema and $\mathfrak{S}$ a set of functional dependencies that hold over $\mathbf{R}$.
Let $\mathbf{R}_1$ and $\mathbf{R}_2$ be a decomposition of $\mathbf{R}$.

Theorem

The decomposition of $\mathbf{R}$ into $\mathbf{R}_1$ and $\mathbf{R}_2$ is lossless-join if there exists an $(A \longrightarrow B) \in \mathfrak{S}^+$ with:

- *A a (super)key of either $\mathbf{R}_1$ or $\mathbf{R}_2$; and*
- *the attributes in A are exactly those attributes common to $\mathbf{R}_1$ and $\mathbf{R}_2$.*

Proof.

Assume $(A \longrightarrow B) \in \mathfrak{S}^+$ with A a (super)key of $\mathbf{R}_1$. We must have

$$\mathbf{R} = \underbrace{\underbrace{\pi_{\mathbf{R}_1}(\mathbf{R})}_{\text{A \textit{single} value per } A} \bowtie \underbrace{\pi_{\mathbf{R}_2}(\mathbf{R})}_{\textit{many} \text{ values per } A}}_{\textit{Natural join}\text{, matching on } A}.$$