DECOMPOSITION & SCHEMA NORMALIZATION

CS 564 - Spring 2025

ACKs: *Dan Suciu, Jignesh Patel, AnHai Doan*

WHAT IS THIS LECTURE ABOUT?

- Bad schemas lead to redundancy
- To "correct" bad schemas: decompose relations
  - lossless-join
  - dependency preserving
- BCNF : a desired normal forms

DB DESIGN THEORY

- Helps us identify the "bad" schemas and improve them
  1. express constraints on the data: functional dependencies (**FDs**)
  2. use the FDs to decompose the relations

- The process, called normalization, obtains a schema in a "normal form" that guarantees certain properties
  - examples of normal forms: **BCNF**, **3NF**, …

Schema Decomposition

WHAT IS A DECOMPOSITION?

We decompose a relation **R**(A_1, …, A_n) by creating

- **R_1**(B_1, .., B_m)
- **R_2**(C_1,…, C_k)

where $\{B_1, \ldots, B_m\} \cup \{C_1, \ldots, C_k\} = \{A_1, \ldots A_n\}$

- The instance of **R_1** is the projection of **R** onto B_1, .., B_m
- The instance of **R_2** is the projection of **R** onto C_1, .., C_l

In general we can decompose a relation into multiple relations.

EXAMPLE: DECOMPOSITION

SSN	name	age	phoneNumber
934729837	Paris	24	608-374-8422
934729837	Paris	24	603-534-8399
123123645	John	30	608-321-1163
384475687	Arun	20	206-473-8221

SSN	name	age
934729837	Paris	24
123123645	John	30
384475687	Arun	20

SSN	phoneNumber
934729837	608-374-8422
934729837	603-534-8399
123123645	608-321-1163
384475687	206-473-8221

DECOMPOSITION DESIDERATA

What should a good decomposition achieve?

1. minimize redundancy
2. avoid information loss (lossless-join)
3. preserve the FDs (dependency preserving)
4. ensure good query performance

EXAMPLE: INFORMATION LOSS

name	age	phoneNumber
Paris	24	608-374-8422
John	24	608-321-1163
Arun	20	206-473-8221

Decompose into:

R_1(name, age)

R_2(age, phoneNumber)

name	age
Paris	24
John	24
Arun	20

age	phoneNumber
24	608-374-8422
24	608-321-1163
20	206-473-8221

We can’t figure out which phoneNumber corresponds to which person!

LOSSLESS-JOIN DECOMPOSITION

$$\begin{tikzcd} & \mathbf{R}(A, B, C) \arrow[dl] \arrow[dr] & \\ \mathbf{R_1}(A, B) \arrow[dr] & & \mathbf{R_2}(B, C) \arrow[dl] \\ & \mathbf{R'}(A, B, C) & \end{tikzcd}$$

decompose (projection)

recover (natural join)

> A natural join is a join on the same attribute names

A schema decomposition is **lossless-join** if for any initial instance **R**, **R = R'**

THE CHASE ALGORITHM

The chase algorithm is a classic database technique that can be used to check for lossless-join decomposition

Running example

- relation **R**(A, B, C, D, E)
- FDs: $A \longrightarrow B, C \quad D \longrightarrow E$

Question: is the following decomposition lossless-join?

$\mathbf{R_1}$(A, D) $\mathbf{R_2}$(A, B, C) $\mathbf{R_3}$(D, E)

CHASE: INITIALIZATION

- We create a table with the attributes of the original relation
- We add one row for each relation we split to

	A	**B**	**C**	**D**	**E**
R_1(A, D)	a	b_1	c_1	d	e_1
R_2(A, B, C)	a	b	c	d_2	e_2
R_3(D, E)	a_3	b_3	c_3	d	e

Attribute not in the relation gets a subscript

Attribute in the relation - no subscript

CHASE: MAIN ALGORITHM

At every iteration, we check whether an FD is violated, and if so, we "force" it to hold

- If one has a subscript and the other not, we remove the subscript
- If both have a subscript, we make one subscript equal to the other

	A	B	C	D	E
R_1(A, D)	a	b_1	c_1	d	$e_1 \rightarrow e$
R_2(A, B, C)	a	b	c	d_2	e_2
R_3(D, E)	a_3	b_3	c_3	d	e

$A \longrightarrow B, C$
$D \longrightarrow E$

The FD $D \longrightarrow E$ is violated, so we need to drop the subscript from the first row

CHASE: MAIN ALGORITHM

$A \longrightarrow B, C$
$D \longrightarrow E$

A	B	C	D	E
a	b_1	c_1	d	e_1
a	b	c	d_2	e_2
a_3	b_3	c_3	d	e

$D \longrightarrow E$

A	B	C	D	E
a	b_1	c_1	d	e
a	b	c	d_2	e_2
a_3	b_3	c_3	d	e

$A \longrightarrow B, C$

A	B	C	D	E
a	b	c	d	e
a	b	c	d_2	e_2
a_3	b_3	c_3	d	e

At the end of the chase:

- If there is a row without subscripts, we can say that the decomposition is lossless-join
- otherwise, it is not

MORE EXAMPLES

- relation **R**(A, B, C, D)
- FD $A \longrightarrow B, C$

	A	B	C	D
$\mathbf{R_1}$				
$\mathbf{R_2}$				

Lossless-join

- decomposition into $\mathbf{R_1}$(A, B, C) and $\mathbf{R_2}$(A, D)

Not lossless-join

- decomposition into $\mathbf{R_1}$(A, B, C) and $\mathbf{R_2}$(D)

DEPENDENCY PRESERVING

Given **R** and a set of FDs F, we decompose **R** into $\mathbf{R_1}$ and $\mathbf{R_2}$. Suppose:

- $\mathbf{R_1}$ has a set of FDs F_1
- $\mathbf{R_2}$ has a set of FDs F_2
- F_1 and F_2 are computed from F

A decomposition is **<u>dependency preserving</u>** if by enforcing F_1 over $\mathbf{R_1}$ and F_2 over $\mathbf{R_2}$, we can enforce F over **R**

A NOTE ON FDS OF SPLIT RELATIONS

Given **R** and a set of FDs F, we decompose **R** into $\mathbf{R_1}$ and $\mathbf{R_2}$. How do we find the FDs F_1 that hold for $\mathbf{R_1}$?

- It is not enough to only keep the FDs from F with attributes in $\mathbf{R_1}$
- Instead, we need to find the non-trivial FDs in the fd closure of F with attributes in $\mathbf{R_1}$

Example: **R**(A, B, C) with FDs: $A \longrightarrow B \quad B \longrightarrow C$

- For $\mathbf{R_1}$(A, C) $\quad F_1 = A \longrightarrow C$

GOOD EXAMPLE

Person(SSN, name, age, canDrink)

- $SSN \longrightarrow name, age$
- $age \longrightarrow canDrink$

decomposes into

- $\mathbf{R_1}$(SSN, name, age)
  - $SSN \longrightarrow name, age$
- $\mathbf{R_2}$(age, canDrink)
  - $age \longrightarrow canDrink$

BAD EXAMPLE

R(A, B, C)

- $A \longrightarrow B$
- $B, C \longrightarrow A$

Decomposes into:

- **R_1**(A, B)
  - $A \longrightarrow B$
- **R_2**(A, C)
  - no FDs here!!

R_1

A	B
a_1	b
a_2	b

R_2

A	C
a_1	c
a_2	c

recover

A	B	C
a_1	b	c
a_2	b	c

The recovered table violates $B, C \longrightarrow A$

NORMAL FORMS

A **normal form** represents a "good" schema design:

- 1NF (flat tables/atomic values)
- 2NF
- 3NF
- **BCNF**
- 4NF
- …

more restrictive

BCNF DECOMPOSITION

BOYCE-CODD NORMAL FORM (BCNF)

A relation **R** is in **<u>BCNF</u>** if whenever $X \longrightarrow B$ is a non-trivial FD, then X is a superkey in **R**

Equivalent definition: for every attribute set X

- either $X^+ = X$
- or $X^+ = all\ attributes$

BCNF EXAMPLE 1

SSN	name	age	phoneNumber
934729837	Paris	24	608-374-8422
934729837	Paris	24	603-534-8399
123123645	John	30	608-321-1163
384475687	Arun	20	206-473-8221

$$SSN \longrightarrow name, age$$

- **key** = $\{SSN, phoneNumber\}$
- $SSN \longrightarrow name, age$ is a “bad” FD
- The above relation is **not** in BCNF!

BCNF EXAMPLE 2

SSN	name	age
934729837	Paris	24
123123645	John	30
384475687	Arun	20

$$SSN \longrightarrow name, age$$

- **key** = $\{SSN\}$
- The above relation is in BCNF!

BCNF Example 3

SSN	phoneNumber
934729837	608-374-8422
934729837	603-534-8399
123123645	608-321-1163
384475687	206-473-8221

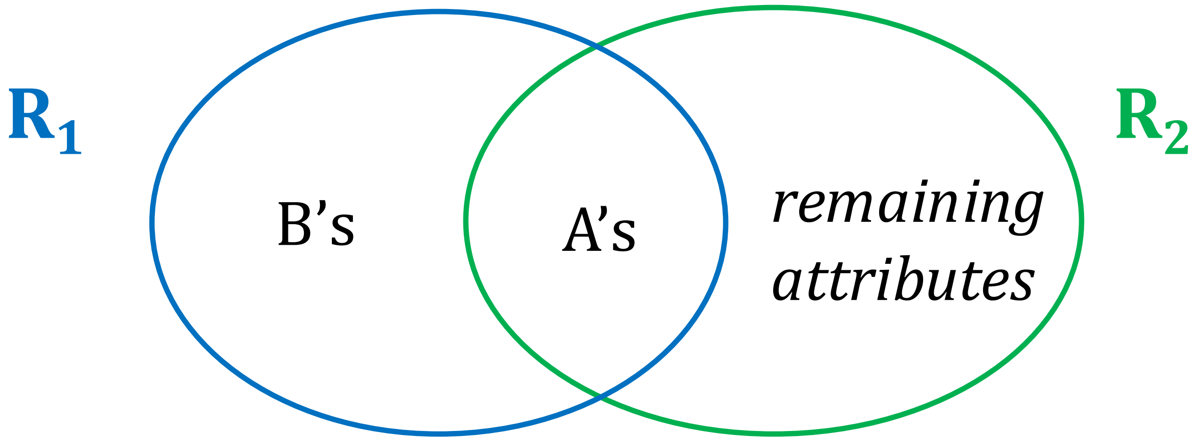
- **key** = $\{SSN, phoneNumber\}$
- The above relation is in BCNF!
- Is it possible that a binary relation is not in BCNF?

BCNF DECOMPOSITION

- Find an FD that violates the BCNF condition

$$A_1, A_2, \dots, A_n \longrightarrow B_1, B_2, \dots, B_m$$

- Decompose **R** to $\mathbf{R_1}$ and $\mathbf{R_2}$:

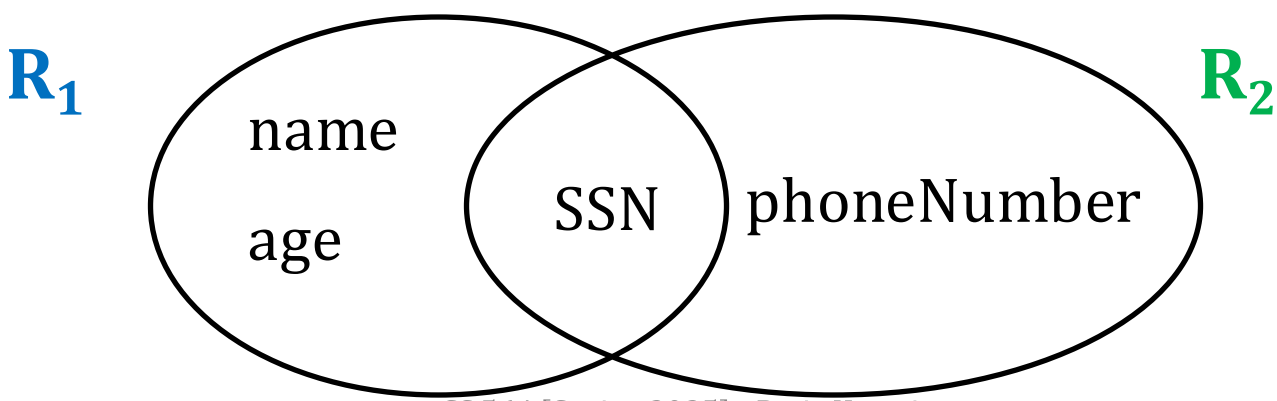


- Continue until no BCNF violations are left

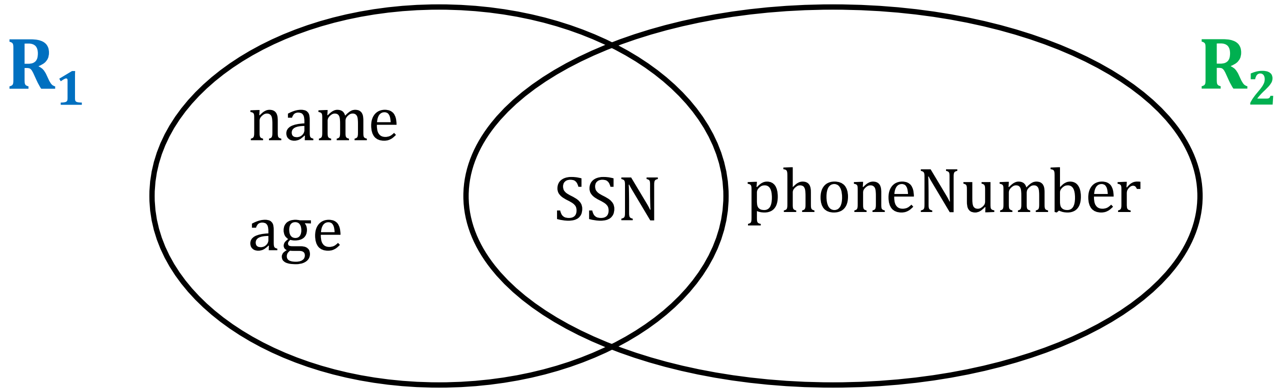
EXAMPLE

SSN	name	age	phoneNumber
934729837	Paris	24	608-374-8422
934729837	Paris	24	603-534-8399
123123645	John	30	608-321-1163
384475687	Arun	20	206-473-8221

- The FD $SSN \longrightarrow name, age$ violates BCNF
- Split into two relations $\mathbf{R_1}$, $\mathbf{R_2}$ as follows:



EXAMPLE CONT'D



$$SSN \longrightarrow name, age$$

SSN	name	age
934729837	Paris	24
123123645	John	30
384475687	Arun	20

SSN	phoneNumber
934729837	608-374-8422
934729837	603-534-8399
123123645	608-321-1163
384475687	206-473-8221

BCNF DECOMPOSITION PROPERTIES

The BCNF decomposition:

– removes certain types of redundancy

– is lossless-join

– is not always dependency preserving

BCNF IS LOSSLESS-JOIN

Example:

R(A, B, C) with $A \longrightarrow B$ decomposes into:

$\mathbf{R_1}$(A, B) and $\mathbf{R_2}$(A, C)

- The BCNF decomposition always satisfies the lossless-join criterion!

BCNF IS NOT DEPENDENCY PRESERVING

R(A, B, C)

- $A \longrightarrow B$
- $B, C \longrightarrow A$

There may not exist any BCNF decomposition that is FD preserving!

The BCNF decomposition is:

- $\mathbf{R_1}$(A, B) with FD $A \longrightarrow B$
- $\mathbf{R_2}$(A, C) with no FDs

BCNF EXAMPLE (1)

Books (author, gender, booktitle, genre, price)

- $author \longrightarrow gender$
- $booktitle \longrightarrow genre, price$

What is the candidate key?

- *(author, booktitle)* is the only one!

Is is in BCNF?

- **No**, because the left hand side of both (not trivial) FDs is not a superkey!

BCNF EXAMPLE (2)

Books (author, gender, booktitle, genre, price)

- $author \longrightarrow gender$
- $booktitle \longrightarrow genre, price$

Splitting **Books** using the FD $author \longrightarrow gender$:

- **Author** (author, gender)
  FD: $author \longrightarrow gender$ in BCNF!
- **Books2** (authos, booktitle, genre, price)
  FD: $booktitle \longrightarrow genre, price$ not in BCNF!

BCNF EXAMPLE (3)

Books (author, gender, booktitle, genre, price)

- $author \longrightarrow gender$
- $booktitle \longrightarrow genre, price$

Splitting **Books** using the FD $author \longrightarrow gender$:

- **Author** (author, gender)
  FD: $author \longrightarrow gender$ in BCNF!

- Splitting **Books2** (author, booktitle, genre, price):
  - **BookInfo** (booktitle, genre, price)
    FD: $booktitle \longrightarrow genre, price$ in BCNF!
  - **BookAuthor** (author, booktitle) in BCNF!

IS NORMALIZATION ALWAYS GOOD?

- Example: suppose A and B are always used together, but normalization says they should be in different tables
  - decomposition might produce unacceptable performance loss
- Example: data warehouses
  - huge historical DBs, rarely updated after creation
  - joins expensive or impractical