Schema Decomposition and Normal Forms

Database Design

- Some schemas are better than others
- Design by decomposition
  - Start with "mega" relations containing all information
  - Then decompose them into smaller, better relations containing the same information
  - Do the decomposition automatically
- Automatic decomposition
  1. Designer specifies mega-relations plus *properties of the data*
  2. System decomposes based on properties
  3. Final set of relations satisfies *normal form* -- guarantees no anomalies, no lost information

Database Anomalies

- Redundancy
- Update anomaly
- Insertion anomaly
- Deletion anomaly

Roll	Name	CID	CNAME	ClassRoom	Slot	Book	Dept
23CS10003	Aryan	CS30202	DBMS	Nalanda 123	Tuesday	Korth	CSE
23CS10004	Ashutosh	CS30202	DBMS	Nalanda 123	Tuesday	Korth	CSE
23CS10063	Sanjana	CS60045	AI	Nalanda 224	Wednesday	Russel	CSE
23CS10091	Sibasis	CS60070	SLT	Nalanda 411	Wednesday	Ben David	History

Schema Decomposition

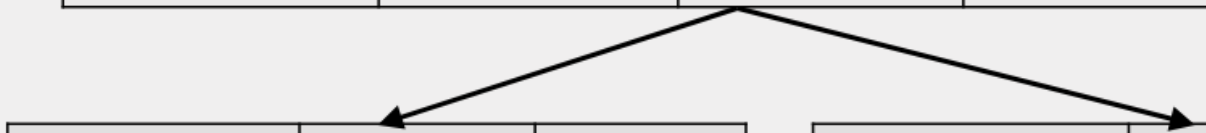
We decompose a relation $\mathbf{R}(A_1, \ldots, A_n)$ by creating

- $\mathbf{R_1}(B_1, .., B_m)$
- $\mathbf{R_2}(C_1, \ldots, C_l)$
- where $\{B_1, \ldots, B_m\} \cup \{C_1, \ldots, C_l\} = \{A_1, \ldots A_n\}$

- The instance of $\mathbf{R_1}$ is the projection of $\mathbf{R}$ onto $B_1, .., B_m$
- The instance of $\mathbf{R_2}$ is the projection of $\mathbf{R}$ onto $C_1, .., C_l$

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



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 Properties

1. No anomalies in decomposed relations
2. Can reconstruct all original information

Decomposition Goals

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

Information Loss

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

Decompose into:

$\mathbf{R_1}$(name, age)

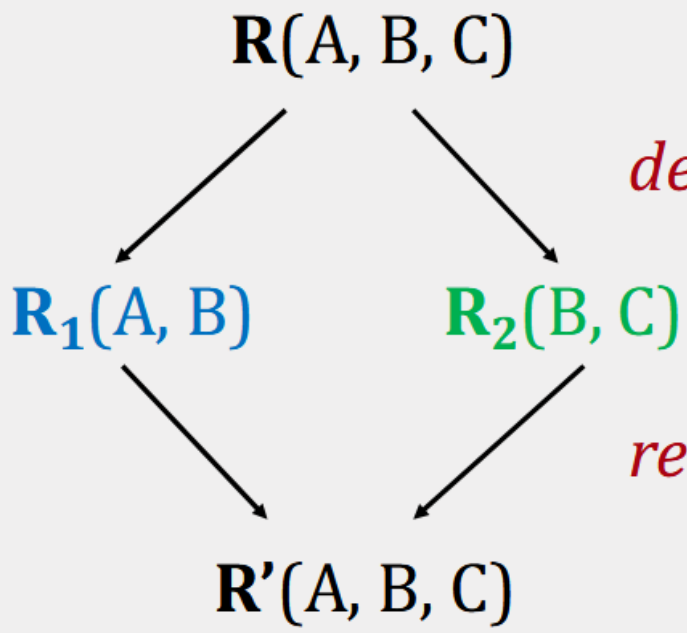
$\mathbf{R_2}$(age, phoneNumber)

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



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'**

Lossless-Join Criterion

Starting with:

- a relation $\mathbf{R(A)}$ + set F of FDs
- a decomposition of $\mathbf{R}$ into $\mathbf{R_1(A_1)}$ and $\mathbf{R_2(A_2)}$

we say that a decomposition is lossless-join **if and only if** at least one of the following FDs is in F^+ (the closure of F) :

1. $A_1 \cap A_2 \longrightarrow A_1$
2. $A_1 \cap A_2 \longrightarrow A_2$

Example

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

Lossless-join

- decomposition into $\mathbf{R_1}$(A, B, C) and $\mathbf{R_2}$(A, D)
- $\{A, B, C\} \cap \{A, D\} = \{A\}$
- For $\mathbf{R_1}$ we have indeed $A \longrightarrow B, C$

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 **dependency preserving** if by enforcing F_1 over $\mathbf{R_1}$ and F_2 over $\mathbf{R_2}$, we can enforce F over **R**

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:

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

$\mathbf{R_1}$

A	B
a_1	b
a_2	b

$\mathbf{R_2}$

A	C
a_1	c
a_2	c

recover

A	B	C
a_1	b	c
a_2	b	c

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

Database Design

- Design one single mega table to include all information
- Mega table may have certain anomalies which leads to inconsistencies in database
- Decompose table into smaller tables automatically
  - Smaller tables should have certain desired properties
  - Smaller tables should be in Normal forms

Normal Forms

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

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

more restrictive

Boyce-Codd Normal Form (BCNF)

A relation **R** is in **BCNF** 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 $X^+ = all\ attributes$

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

$$SSN \longrightarrow name, age$$

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

Example

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

$$SSN \longrightarrow name, age$$

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

Example

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

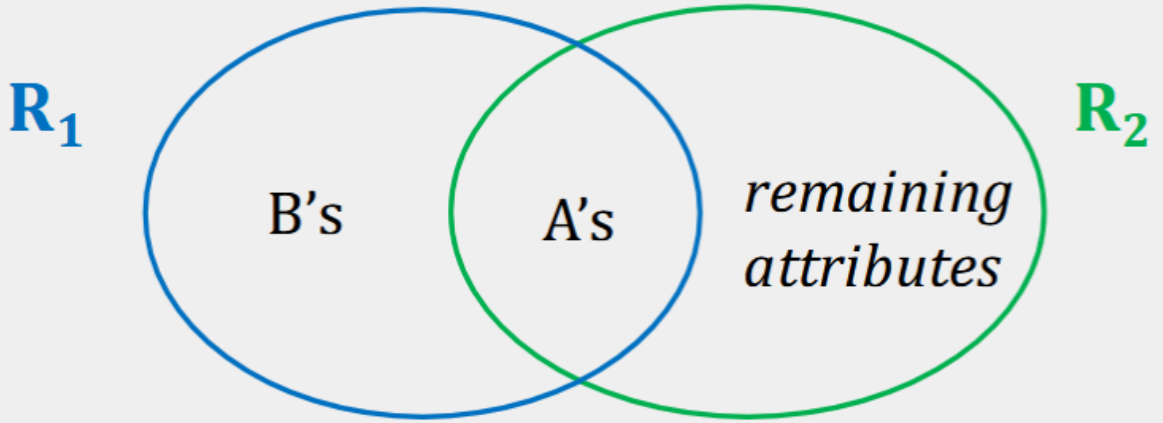
- **key** = $\{SSN, phoneNumber\}$
- The above relation is in BCNF!
- Is it possible that a binary relation is not in BCNF?

BCNF Decompisition

- Find an FD that violates the BCNF condition

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

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

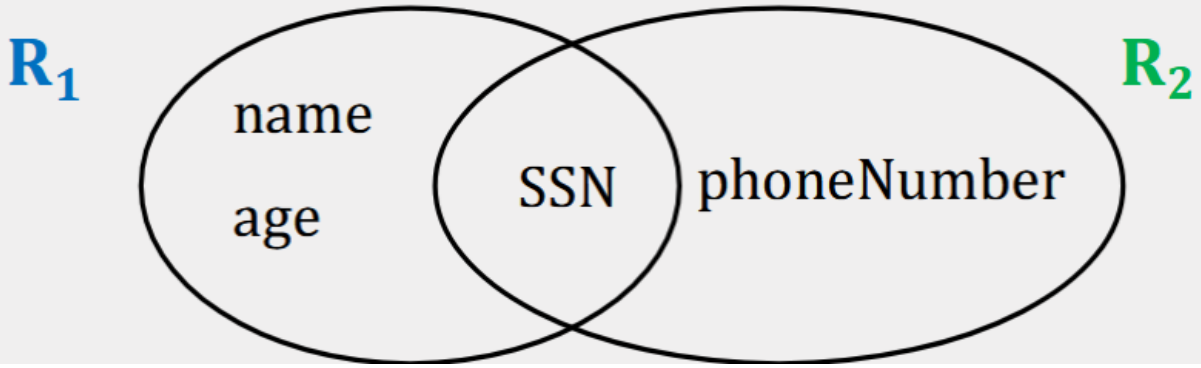


- Continue until no BCNF violations are left

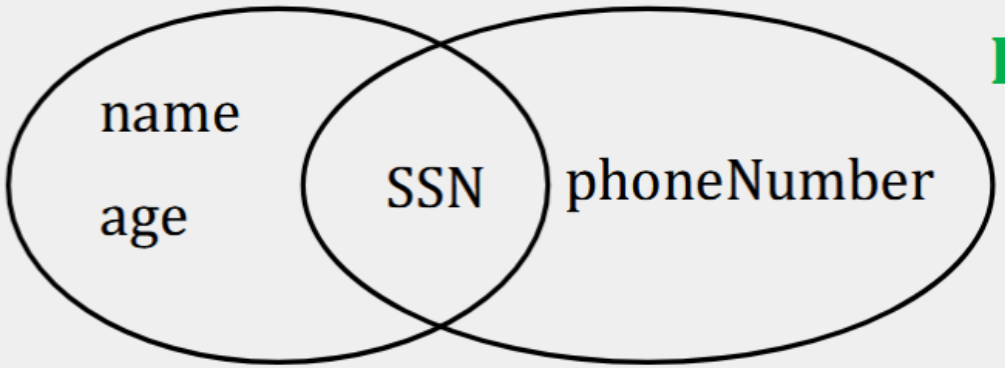
Example

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



$SSN \longrightarrow name, age$

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

Properties of BCNF Decomposition

The BCNF decomposition:

- removes certain types of redundancy
- is lossless-join
- is not always dependency preserving

BCNF is Lossless-Join

Example:

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

Example

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

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

Example

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!

Example

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!

Third Normal Form (3NF)

A relation **R** is in **3NF** if whenever $X \longrightarrow A$, one of the following is true:

- $A \in X$ (trivial FD)
- X is a superkey
- A is part of some key of **R** (prime attribute)

BCNF implies 3NF !!

3NF

- Example: **R**(A, B, C) with $A, B \longrightarrow C$ and $C \longrightarrow A$
  - is in 3NF. Why?
  - is not in BCNF. Why?

- Compromise used when BCNF not achievable: *aim for BCNF and settle for 3NF*
- Lossless-join and dependency preserving decomposition into a collection of 3NF relations is always possible!

3NF Decomposition Algorithm

1. Apply the algorithm for BCNF decomposition until all relations are in 3NF (we can stop earlier than BCNF)
2. Compute a minimal basis F' of F
3. For each non-preserved FD $X \longrightarrow A$ in F', add a new relation R(X, A)

Example

Start with relation **R** (A, B, C, D) with FDs:

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

Step 1: find a BCNF decomposition

- **R1** (B, C)
- **R2** (A, B, D)

Example

Start with relation **R** (A, B, C, D) with FDs:

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

Step 2: compute a minimal basis of the original set of FDs:

- $A \longrightarrow D$
- $B \longrightarrow C$
- $D \longrightarrow A$
- $D \longrightarrow B$

Example

Start with relation **R** (A, B, C, D) with FDs:

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

Step 3: add a new relation for any FD in the basis that is not satisfied:

- all the dependencies in *F'* are satisfied!
- the resulting decomposition **R1**, **R2** is also BCNF!

Fourth Normal Form

- A relation schema R is in **4NF** with respect to a set D of functional and multivalued dependencies if for all multivalued dependencies in D^+ of the form $\alpha \rightarrow\rightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of the following hold:
  - $\alpha \rightarrow\rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$ or $\alpha \cup \beta = R$)
  - α is a superkey for schema R
- If a relation is in 4NF it is 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