

Lecture 9

Database Normalization

COMP3278B

Introduction to Database Management Systems

Dr. Ping Luo

Email : pluo@cs.hku.hk



Department of Computer Science, The University of Hong Kong

Acknowledgement: Dr Chui Chun Kit

Outcome based learning

Outcome 1. Information Modeling

- Able to understand the modeling of real life information in a database system.

Outcome 2. Query Languages

- Able to understand and use the languages designed for data access.

Outcome 3. System Design

- Able to understand the design of an efficient and reliable database system.

Outcome 4. Application Development

- Able to implement a practical application on a real database.

Recap Armstrong's Axioms

● We have 3 basic axioms.

- 1. **Reflexivity** - if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$.
- 2. **Transitivity** - if $\alpha \rightarrow \beta$ and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$.
- 3. **Augmentation** - if $\alpha \rightarrow \beta$, then $\gamma\alpha \rightarrow \gamma\beta$.

Recap Armstrong's Axioms

- 3 more axioms to help easier prove!
 - 4. Union - if $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$, then $\alpha \rightarrow \beta\gamma$.
 - 5. Decomposition - if $\alpha \rightarrow \beta\gamma$, then $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$.
 - 6. Pseudo-transitivity - if $\alpha \rightarrow \beta$ and $\gamma\beta \rightarrow \delta$, then $\alpha\gamma \rightarrow \delta$.

- 2 rules in tutorial!
 - 7. Extensivity - if $\alpha \rightarrow \beta$, then $\alpha \rightarrow \alpha\beta$.
 - 8. Composition - if $\alpha \rightarrow \beta$ and $\gamma \rightarrow \delta$, then $\alpha\gamma \rightarrow \beta\delta$.

Attribute set closure α^+

- Given a set F of FDs and a set of attributes α .
- The **closure of α** (denoted as α^+) is the set of attributes that can be **functionally determined by α** .

Attribute set
closure of A.

$$\begin{array}{|c|}\hline F = \{A \rightarrow B, B \rightarrow C\} \\ \hline \end{array}$$
$$\{A\}^+ = \{ A, B, C \}$$

1. $A \rightarrow A$ is always true (by **Reflexivity**).

2. $A \rightarrow B$ is given in F.

3. $A \rightarrow C$ is derived from F:

Given $A \rightarrow B$ and $B \rightarrow C$, $A \rightarrow C$ is also true (by **Transitivity**).

Attribute set closure α^+

- Given a set F of FDs and a set of attributes α .
- The **closure of α** (denoted as α^+) is the set of attributes that can be **functionally determined by α** .

Attribute set closure of A.

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

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

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

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

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

Note that we only consider **single attribute**, not attribute sets (so we do not have AB, ABC, AC...etc in $\{A, B\}^+$).

FD closure F^+

- The set of **ALL functional dependencies that can be logically implied by F** is called the closure of F (or F^+)
- To compute F^+ in a relation R :

Step 1. Treat every subset of R as α .

Step 2. For every α , compute α^+ .

Step 3. Use α as LHS, and generate an FD for every subset of α^+ on RHS.

This is the attribute set closure.

The `fd_closure()` algorithm

FD closure F^+

- Given a relation $R(N, S, P)$ and the functional dependencies $F = \{N \rightarrow S, N \rightarrow P\}$ find the FD closure F^+ .

N	S	P	NS	NP	SP	NSP

Step 1. Treat every subset of R as α .

FD closure F^+



Given a relation $R(N, S, P)$ and the functional dependencies $F = \{N \rightarrow S, N \rightarrow P\}$ find the FD closure F^+ .

Attribute set closure	N	S	P	NS	NP	SP	NSP
	{N,S,P}						

Step 2. For every α , compute α^+ .

To find the attribute set closure $\{N\}$, use the `attribute_closure()` algorithm

1. $\text{result} = \{N\}$
2. Consider the FDs with $N \rightarrow S, N \rightarrow P$, add S and P into result .
3. $\text{result} = \{N, S, P\}$

FD closure F^+



Given a relation $R(N, S, P)$ and the functional dependencies $F = \{N \rightarrow S, N \rightarrow P\}$ find the FD closure F^+ .

Attribute set closure	N	S	P	NS	NP	SP	NSP
	{N,S,P}	{S}	{P}	{N,S,P}	{N,S,P}	{S,P}	{N,S,P}

Step 2. For every α , compute α^+ .

To find the attribute set closure $\{N\}$, use the `attribute_closure()` algorithm

1. $\text{result} = \{N\}$
2. Consider the FDs with $N \rightarrow S, N \rightarrow P$, add S and P into result .
3. $\text{result} = \{N,S,P\}$

FD closure F^+



Given a relation $R(N, S, P)$ and the functional dependencies $F = \{N \rightarrow S, N \rightarrow P\}$ find the FD closure F^+ .

	N	S	P	NS	NP	SP	NSP
Attribute set closure	{N,S,P}	{S}	{P}	{N,S,P}	{N,S,P}	{S,P}	{N,S,P}
FD	$N \rightarrow N$						
	$N \rightarrow S$						
	$N \rightarrow P$						
	$N \rightarrow NS$						
	$N \rightarrow NP$						
	$N \rightarrow SP$						
	$N \rightarrow NSP$						

Step 3. Use α as LHS, and generate an FD for every subset of α^+ on RHS.

FD closure F^+



Given a relation $R(N, S, P)$ and the functional dependencies $F = \{N \rightarrow S, N \rightarrow P\}$ find the FD closure F^+ .

	N	S	P	NS	NP	SP	NSP
Attribute set closure	{N,S,P}	{S}	{P}	{N,S,P}	{N,S,P}	{S,P}	{N,S,P}
FD	$N \rightarrow N$	$S \rightarrow S$	$P \rightarrow P$	$NS \rightarrow N$	$NP \rightarrow N$	$SP \rightarrow S$	$NSP \rightarrow N$
	$N \rightarrow S$			$NS \rightarrow S$	$NP \rightarrow S$	$SP \rightarrow P$	$NSP \rightarrow S$
	$N \rightarrow P$			$NS \rightarrow P$	$NP \rightarrow P$	$SP \rightarrow SP$	$NSP \rightarrow P$
	$N \rightarrow NS$			$NS \rightarrow NS$	$NP \rightarrow NS$		$NSP \rightarrow NS$
	$N \rightarrow NP$			$NS \rightarrow NP$	$NP \rightarrow NP$		$NSP \rightarrow NP$
	$N \rightarrow SP$			$NS \rightarrow SP$	$NP \rightarrow SP$		$NSP \rightarrow SP$
	$N \rightarrow NSP$			$NS \rightarrow NSP$	$NP \rightarrow NSP$		$NSP \rightarrow NSP$

Concept



Decomposition

- Lossless-join decomposition
- Dependency preserving decomposition

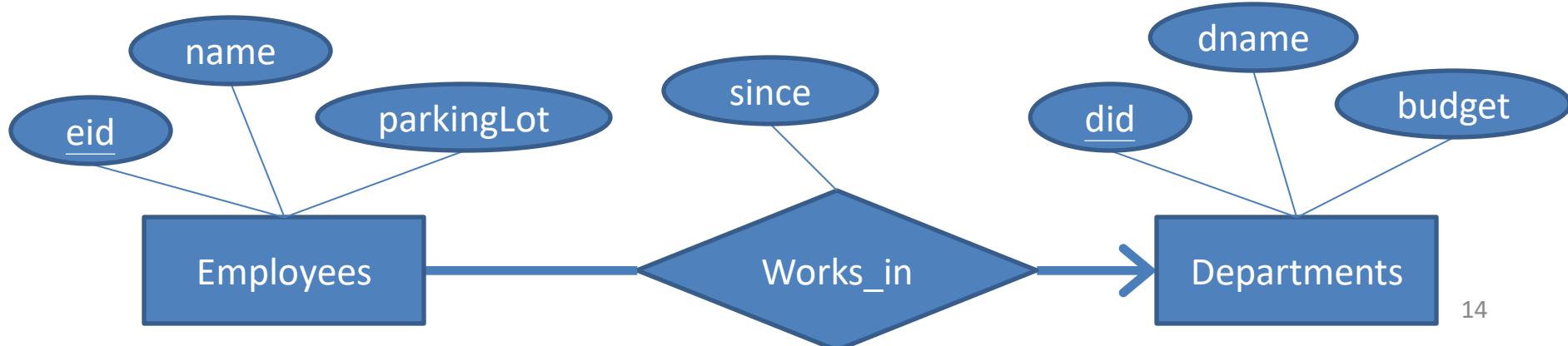


Normal form

- Boyce-Codd Normal Form (BCNF)

Motivating example

- Let's consider the following schema
 - Employees have *eid* (key), *name*, *parkingLot*.
 - Departments have *did* (key), *dname*, *budget*.
 - An employee works in exactly one department, **since** some date.
 - Employees who work in the same department must park at the same *parkingLot*.



Motivating example

Reduce to relational tables

Employees(eid, name, parkingLot, did, since)

Foreign key: *did* references Departments(did)

Departments(did, dname, budget)

Observation: In Employees table, whenever *did* is 1, *parkingLot* must be “A”!

Implication: The constraint “*Employees who work in the same department must park at the same parkingLot*” is **NOT** utilized in the design!!!

There are some **redundancy** in the Employees table.



eid	name	parkingLot	did	since
1	Kit	A	1	1/9/2014
2	Ben	B	2	2/4/2010
3	Ernest	B	2	30/5/2011
4	Betty	A	1	22/3/2013
5	David	A	1	4/11/2004
6	Joe	B	2	12/3/2008
7	Mary	B	2	14/7/2009
8	Wandy	A	1	9/8/2008

did	dname	budget
1	Human Resource	4M
2	Accounting	3.5M

Yes! As *parkingLot* is “functionally depend” on *did*, we should not put *parkingLot* in the Employee table.



We are going to learn

Database normalization

- The process of organizing the columns and tables of a relational database to minimize redundancy and dependency.

To make sure that every relation R is in a “good” form.

- If R is not “good”, **decompose** it into a set of relations $\{R_1, R_2, \dots, R_n\}$.

Question: How can we do the decomposition?

Are there any guidelines / theories developed to decompose a relation?



Yes! The theories can be explained through **functional dependencies** ☺.



Normalization goal

- We would like to meet the following goals when we decompose a relation schema R with a set of functional dependencies F into R_1, R_2, \dots, R_n
 - **1. Lossless-join** – Avoid the decomposition result in information loss.
 - **2. Reduce redundancy** – The decomposed relations R_i should be in Boyce-Codd Normal Form (BCNF).
 - **3. Dependency preserving** – Avoid the need to join the decomposed relations to check the functional dependencies when new tuples are inserted into the database.

Section 1

Lossless-join Decomposition

Example 1

R Functional dependencies

A	B	C
1	1	3
1	2	2
2	1	3
3	2	2
3	1	3
4	2	2
4	1	3

$$F = \{B \rightarrow C\}$$

The functional dependency $B \rightarrow C$ tells us that for all tuples with the same value in B , there should be **at most** one corresponding value in C (E.g., If $B=1$, $C=3$; if $B=2$, $C=2$)

Question: Will decomposing $R(A,B,C)$ into $R_1(A,B)$ and $R_2(A,C)$ cause information lost?

Decompose

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

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

A	B
1	1
1	2
2	1
3	2
3	1
4	2
4	1

A	C
1	3
1	2
2	3
3	2
3	3
4	2
4	3

Think in this way:

Is this decomposition “**lossless join decomposition**”?
I.e., Is there any information lost if we decompose R in this way?



Example 1

R

A	B	C
1	1	3
1	2	2
2	1	3
3	2	2
3	1	3
4	2	2
4	1	3

Functional dependencies $R_1 \bowtie R_2 = \pi_{A, B}(R) \bowtie \pi_{A, C}(R)$

$$F = \{B \rightarrow C\}$$



Decompose

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

A	B
1	1
1	2
2	1
3	2
3	1
4	2
4	1

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

A	C
1	3
1	2
2	3
3	2
3	3
4	2
4	3



This is a bad decomposition

A	B	C
1	1	3
1	1	2
1	2	3
1	2	2
2	1	3
3	2	2
3	2	3
3	1	2
3	1	3
4	2	2
4	2	3
4	1	2
4	1	3

To check if the decomposition will cause information lost, let's try to join R_1 and R_2 and see if we can recover R .

As we see that $R_1 \bowtie R_2 \neq R$, the decomposition has information lost.

This is NOT a lossless-join decomposition.



Example 2

R

A	B	C
1	1	3
1	2	2
2	1	3
3	2	2
3	1	3
4	2	2
4	1	3

Functional dependencies

$$F = \{B \rightarrow C\}$$



$$R_1 \bowtie R_2 = \pi_{A,B}(R) \bowtie \pi_{B,C}(R)$$

A	B	C
1	1	3
1	2	2
2	1	3
3	2	2
3	1	3
4	2	2
4	1	3

How about decomposing the relation $R(A,B,C)$ into $R_1(A,B)$ and $R_2(B,C)$?

Decompose

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

A	B
1	1
1	2
2	1
3	2
3	1
4	2
4	1

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

B	C
1	3
2	2

OK

Well done! Since $R_1 \bowtie R_2 = R$, breaking down R to R_1 and R_2 in this way has no information lost. This decomposition is a lossless-join decomposition.



Lossless-join decomposition

R

Functional dependencies

$$F = \{B \rightarrow C\}$$

A	B	C
1	1	3
1	2	2
2	1	3
3	2	2
3	1	3
4	2	2
4	1	3

What is/are the condition(s)
for a decomposition to be
lossless-join?

Example I

NOT Lossless-join decomposition

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

A	B
1	1
1	2
2	1
3	2
3	1
4	2
4	1

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

A	C
1	3
1	2
2	3
3	2
3	3
4	2
4	3



Example II

Lossless-join decomposition

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

A	B
1	1
1	2
2	1
3	2
3	1
4	2
4	1

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

B	C
1	3
2	2



Lossless-join decomposition

R

A	B	C
1	1	3
1	2	2
2	1	3
3	2	2
3	1	3
4	2	2
4	1	3

Functional dependencies

$$F = \{B \rightarrow C\}$$

1

A	B
1	1

Let's consider the first tuple (1,1,3) in R.

Note that there is only ONE tuple in R_1 with A=1, B=1.

Example I

NOT Lossless-join decomposition

$$R_1 = \pi_{A, B}(R) \quad R_2 = \pi_{A, C}(R)$$

A	B
1	1
1	2
2	1
3	2
3	1
4	2
4	1

A	C
1	3
1	2
2	3
3	2
3	3
4	2
4	3

Lossless-join decomposition

R

A	B	C
1	1	3
1	2	2
2	1	3
3	2	2
3	1	3
4	2	2
4	1	3

Functional dependencies

$$F = \{B \rightarrow C\}$$

1

A	B
1	1

Let's consider the first tuple $(1,1,3)$ in R.

2

A	C
1	3
1	2

Since $A \rightarrow AC$ is NOT a functional dependency in F^+ , there can be **more than one tuples** with $A=1$ in R_2 (e.g., $(1,3)$, $(1,2)$).

Example I

NOT Lossless-join decomposition

$$R_1 = \pi_{A, B}(R) \quad R_2 = \pi_{A, C}(R)$$

A	B
1	1
1	2
2	1
3	2
3	1
4	2
4	1

A	C
1	3
1	2
2	3
3	2
3	3
4	2
4	3

Lossless-join decomposition

R

A	B	C
1	1	3
1	2	2
2	1	3
3	2	2
3	1	3
4	2	2
4	1	3

Functional dependencies

$$F = \{B \rightarrow C\}$$

1

A	B
1	1

Let's consider the first tuple (1,1,3) in R.

2

A	C
1	3
1	2

Since $A \rightarrow AC$ is NOT a functional dependency in F^+ , there can be **more than one tuples** with A=1 in R_2 (e.g., (1,3), (1,2)).

3

A	B	C
1	1	3
1	1	2



Therefore when we join R_1 and R_2 , **more than one tuples will be generated** (i.e., (1,1) in R_1 combine with (1,3) and (1,2) in R_2)

Example I NOT Lossless-join decomposition

$$R_1 = \pi_{A,B}(R) \quad R_2 = \pi_{A,C}(R)$$

A	B
1	1
1	2
2	1
3	2
3	1
4	2
4	1

A	C
1	3
1	2
2	3
3	2
3	3
4	2
4	3

Observation:

The decomposition of $R(A,B,C)$ into $R_1(A,B)$ and $R_2(A,C)$ is NOT lossless-join because

• $A \rightarrow AC$

is NOT in F^+ , and ... (to be explained in the next slide)



Lossless-join decomposition

R

A	B	C
1	1	3
1	2	2
2	1	3
3	2	2
3	1	3
4	2	2
4	1	3

Functional dependencies

$$F = \{B \rightarrow C\}$$

1

A	C
1	3

Let's consider the first tuple (1,1,3) in R.

Note that there is only **ONE** tuple in R_2 with $A=1, C=3$.

Example I

NOT Lossless-join decomposition

$$R_1 = \pi_{A, B}(R) \quad R_2 = \pi_{A, C}(R)$$

A	B
1	1
1	2
2	1
3	2
3	1
4	2
4	1

A	C
1	3
1	2
2	3
3	2
3	3
4	2
4	3

Lossless-join decomposition

R

A	B	C
1	1	3
1	2	2
2	1	3
3	2	2
3	1	3
4	2	2
4	1	3

Functional dependencies

$$F = \{B \rightarrow C\}$$

1

A	C
1	3

Let's consider the first tuple (1,1,3) in R.

NOT Lossless-join decomposition

2

A	B
1	1
1	2

Since $A \rightarrow AB$ is NOT a functional dependency in F^+ , there can be more than one tuples with A=1 in R_1 (i.e., (1,1), (1,2)).

$$R_1 = \pi_{A, B}(R) \quad R_2 = \pi_{A, C}(R)$$

A	B
1	1
1	2
2	1
3	2
3	1
4	2
4	1

A	C
1	3
1	2
2	3
3	2
3	3
4	2
4	3

Lossless-join decomposition

R

A	B	C
1	1	3
1	2	2
2	1	3
3	2	2
3	1	3
4	2	2
4	1	3

Functional dependencies

$$F = \{B \rightarrow C\}$$

1

A	C
1	3
1	3

Let's consider the first tuple (1,1,3) in R.

NOT Lossless-join decomposition

2

A	B
1	1
1	2

Since $A \rightarrow AB$ is NOT a functional dependency in F^+ , there can be more than one tuples with $A=1$ in R_1 (i.e., (1,1), (1,2)).

3

A	B	C
1	1	3
1	2	3



Therefore when we join R_1 and R_2 , more than one tuples will be generated (i.e., (1,3) in R_2 combine with (1,1) and (1,2) in R_1)

Example I

NOT Lossless-join decomposition

$$R_1 = \pi_{A,B}(R) \quad R_2 = \pi_{A,C}(R)$$

A	B
1	1
1	2
2	1
3	2
3	1
4	2
4	1

A	C
1	3
1	2
2	3
3	2
3	3
4	2
4	3

Observation:

The decomposition of $R(A,B,C)$ into $R_1(A,B)$ and $R_2(A,C)$ is NOT lossless-join because

- $A \rightarrow AC$ (explained in previous slide), and
- $A \rightarrow AB$

are NOT in F^+ .



Lossless-join decomposition

R

A	B	C
1	1	3
1	2	2
2	1	3
3	2	2
3	1	3
4	2	2
4	1	3

Functional dependencies

$$F = \{B \rightarrow C\}$$

1

A	B
1	1

Let's consider the first tuple **(1,1,3)** in R.
Note that there is only **ONE** tuple in R_1 with A=1, B=1.

Example II Lossless-join decomposition

$$R_1 = \pi_{A, B}(R) \quad R_2 = \pi_{B, C}(R)$$

A	B
1	1
1	2
2	1
3	2
3	1
4	2
4	1

B	C
1	3
2	2

Lossless-join decomposition

R

A	B	C
1	1	3
1	2	2
2	1	3
3	2	2
3	1	3
4	2	2
4	1	3

Functional dependencies

$$F = \{B \rightarrow C\}$$

1

A	B
1	1

Let's consider the first tuple **(1,1,3)** in R. Note that there is only **ONE** tuple in R_1 with A=1, B=1.

2

B	C
1	3

Since $B \rightarrow BC$ is a functional dependency in F^+ , there is only **one tuple** with B=1 in R_2 .

Example II Lossless-join decomposition

$$R_1 = \pi_{A, B}(R) \quad R_2 = \pi_{B, C}(R)$$

A	B
1	1
1	2
2	1
3	2
3	1
4	2
4	1

B	C
1	3
2	2

Lossless-join decomposition

R

A	B	C
1	1	3
1	2	2
2	1	3
3	2	2
3	1	3
4	2	2
4	1	3

Functional dependencies

$$F = \{B \rightarrow C\}$$

1

A	B
1	1

Let's consider the first tuple **(1,1,3)** in R. Note that there is only **ONE** tuple in R_1 with A=1, B=1.

2

B	C
1	3

Since $B \rightarrow BC$ is a functional dependency in F^+ , there is only **one tuple** with B=1 in R_2 .

3

A	B	C
1	1	3



Therefore when we join R_1 and R_2 , there will be **ONLY ONE tuple generated**, and that must be the corresponding tuple **(1,1,3)** in R.

Example II Lossless-join decomposition

$$R_1 = \pi_{A, B}(R) \quad R_2 = \pi_{B, C}(R)$$

A	B
1	1
1	2
2	1
3	2
3	1
4	2
4	1

Observation:

The decomposition of R(A,B,C) into $R_1(A, B)$ and $R_2(B, C)$ is lossless-join because

$$B \rightarrow BC$$

is in F^+ .



Testing for lossless-join decomposition

- Consider a decomposition of R into R_1 and R_2 .
 - Schema of R = schema of $R_1 \cup$ schema of R_2 .
- Let $\text{schema of } R_1 \cap \text{schema of } R_2$ be R_1 and R_2 's common attributes.
 - A decomposition of R into R_1 and R_2 is lossless-join if and only if at least one of the following dependencies is in F^+ .

Schema of $R_1 \cap$ schema of $R_2 \rightarrow$ schema of R_1

OR

Schema of $R_1 \cap$ schema of $R_2 \rightarrow$ schema of R_2

Example

Question: Given $R(A,B,C)$, $F=\{B \rightarrow C\}$, is the following a lossless join decomposition of R ?

- $R_1(A, B)$, $R_2(B, C)$

Answer: To see if (R_1, R_2) is a lossless join decomposition of R , we do the following:

- Find common attributes of R_1 and R_2 : **B**
- Verify if *any* of the FD below holds in F^+ , if one of the FD holds, then the decomposition is lossless join.

$B \rightarrow R_1$ (i.e., $B \rightarrow AB?$)

$B \rightarrow R_2$ (i.e., $B \rightarrow BC?$)



- Since $B \rightarrow BC$ (by **Augmentation rule on $B \rightarrow C$**), R_1 and R_2 are lossless join decomposition of R .

Section 2

Dependency preserving

Decomposition

Dependency preserving

- When decomposing a relation, we also want to keep the functional dependencies.
 - A FD $X \rightarrow Y$ is preserved in a relation R if R contains all the attributes of X and Y.
- If a dependency is lost when R is decomposed into R_1 and R_2 :
 - When we insert a new record in R_1 and R_2 , we have to obtain $R_1 \bowtie R_2$ and check if the new record violates the lost dependency before insertion.
 - It could be very inefficient because joining is required in every insertion!

Dependency preserving

- Consider $R(A,B,C,D)$, $F = \{A \rightarrow B, B \rightarrow CD\}$

$F^+ = \{A \rightarrow B, B \rightarrow CD, A \rightarrow CD, \text{ trivial FDs}\}$

- If R is decomposed to $R_1(A,B)$, $R_2(B,C,D)$:

$F_1 = \{A \rightarrow B, \text{ trivials}\}$, the projection of F^+ on R_1
 $F_2 = \{B \rightarrow CD, \text{ trivials}\}$, the projection of F^+ on R_2

Note that $A \rightarrow CD$ is in F^+ because of the **Transitivity** axiom.

R			
A	B	C	D
1	1	3	4
2	1	3	4
3	2	2	3
4	1	3	4

Decompose

$$R_1 = \pi_{A, B}(R) \quad R_2 = \pi_{B, C, D}(R)$$

A	B
1	1
2	1
3	2
4	1

B	C	D
1	3	4
2	2	3

This is a **dependency preserving decomposition** as:

$$(F_1 \cup F_2)^+ = F^+$$

Let us illustrate the implication of dependency preserving in the next slide.



Dependency preserving

Consider $R(A,B,C,D)$, $F = \{A \rightarrow B, B \rightarrow CD\}$

$F^+ = \{A \rightarrow B, B \rightarrow CD, A \rightarrow CD, \text{ trivial FDs}\}$

Is this a lossless join decomposition?

Yes! As $B \rightarrow R_2$ (i.e., $B \rightarrow BCD$) holds in F^+ .

That mean we can recover R by $R_1 \bowtie R_2$.

Why it is dependency preserving?

Think about it...

If we insert a new record

A	B	C	D
5	1	4	4

into R_1 and R_2 :

R_1

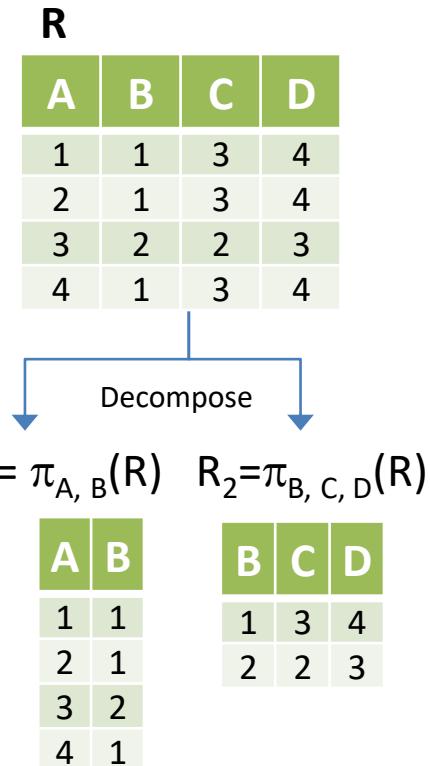
A	B
5	1

R_2

B	C	D
1	4	4

We need to check if the new record will make the database violate any FDs in F^+ .

Is such decomposition allow us to do the validation on R_1 and R_2 **ONLY?** (But no need to join R_1 and R_2 to validate it?)



Dependency preserving

- $F^+ = \{ A \rightarrow B, B \rightarrow CD, A \rightarrow CD, \text{trivials} \}$

- Inserting tuple (5,1,4,4) violates $B \rightarrow CD$.

- The decomposition is **dependency preserving** as we only need to check:

- Inserting

A	B
5	1

 violate any F_1 in R_1 ?

This involves checking $F_1 = \{A \rightarrow B\}$.

- Inserting

B	C	D
1	4	4

 violate any F_2 in R_2 ?

This involves checking $F_2 = \{B \rightarrow CD\}$.

We can check F_1 on R_1 and F_2 on R_2 only because
 $(F_1 \cup F_2)^+ = F^+$

R			
A	B	C	D
1	1	3	4
2	1	3	4
3	2	2	3
4	1	3	4
5	1	4	4

↓ Decompose ↓

$$R_1 = \pi_{A, B}(R) \quad R_2 = \pi_{B, C, D}(R)$$

A	B	B	C	D
1	1	1	3	4
2	1	2	2	3
3	2	1	4	4
5	1	1	4	4

Although among the two validations we haven't checked $A \rightarrow CD$, but since $A \rightarrow B$ is checked in F_1 , and $B \rightarrow CD$ is checked in F_2 , if we pass both F_1 and F_2 , it implies $A \rightarrow CD$.

Dependency preserving



What about decompose R to $R_1(A,B)$, $R_2(A,C,D)$?



R is decomposed to $R_1(A,B)$, $R_2(A,C,D)$

● $F^+ = \{A \rightarrow B, B \rightarrow CD, A \rightarrow CD, \text{ trivial FDs}\}$

● $F_1 = \{A \rightarrow B, \text{ trivials}\}$, the projection of F^+ on R_1

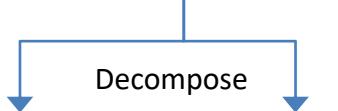
● $F_2 = \{A \rightarrow CD, \text{ trivials}\}$, the projection of F^+ on R_2

This is NOT a dependency preserving decomposition as:

$$(F_1 \cup F_2)^+ \neq F^+$$

Let us illustrate the implication of NOT dependency preserving in the next slide.

R			
A	B	C	D
1	1	3	4
2	1	3	4
3	2	2	3
4	1	3	4



$$R_1 = \pi_{A, B}(R) \quad R_2 = \pi_{A, C, D}(R)$$

A	B
1	1
2	1
3	2
4	1

A	C	D
1	3	4
2	3	4
3	2	3
4	3	4



Dependency preserving



What about decompose R to $R_1(A,B)$, $R_2(A,C,D)$?



Is this a lossless join decomposition?

Yes! As $A \rightarrow R_1$ (i.e., $A \rightarrow AB$) holds in F^+ .

That mean we can recover R by $R_1 \bowtie R_2$.



Is it dependency preserving?

Think about it...

If we insert a new record

A	B	C	D
5	1	4	4

into R_1 and R_2 :



R_1

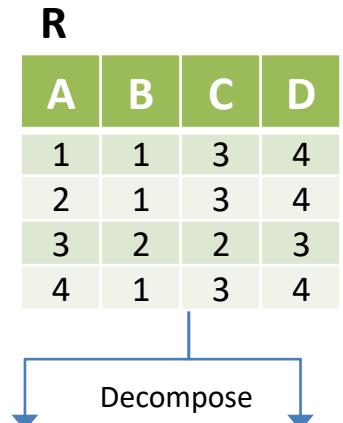
A	B
5	1



R_2

A	C	D
5	4	4

We need to check if the new record will make the database violate any FDs in F^+ . Is such decomposition allow us to do the validation on R_1 and R_2 **only** (but no need to join R_1 and R_2)?



$$R_1 = \pi_{A, B}(R) \quad R_2 = \pi_{A, C, D}(R)$$

A	B
1	1
2	1
3	2
4	1

A	C	D
1	3	4
2	3	4
3	2	3
4	3	4



Dependency preserving

- $F^+ = \{ A \rightarrow B, B \rightarrow CD, A \rightarrow CD \}$

- Inserting tuple (5,1,4,4) **violates** $B \rightarrow CD$.

- The decomposition is **NOT dependency preserving** as if we only check:

- Inserting

A	B
5	1

 violate any F_1 in R_1 ?

This involves checking $F_1 = \{A \rightarrow B\}$.

- Inserting

A	C	D
5	4	4

 violate any F_2 in R_2 ?

This involves checking $F_2 = \{A \rightarrow CD\}$.

We CANNOT check F_1 on R_1 and F_2 on R_2 only because
 $(F_1 \cup F_2)^+ \neq F^+$

Decomposition in this way requires joining tables to validate $B \rightarrow CD$ for **EVERY INSERTION!**

R

A	B	C	D
1	1	3	4
2	1	3	4
3	2	2	3
4	1	3	4
5	1	4	4

Decompose

$R_1 = \pi_{A, B}(R)$	$R_2 = \pi_{A, C, D}(R)$																														
<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>A</td><td>B</td></tr><tr><td>1</td><td>1</td></tr><tr><td>2</td><td>1</td></tr><tr><td>3</td><td>2</td></tr><tr><td>4</td><td>1</td></tr><tr><td>5</td><td>1</td></tr></table>	A	B	1	1	2	1	3	2	4	1	5	1	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>A</td><td>C</td><td>D</td></tr><tr><td>1</td><td>3</td><td>4</td></tr><tr><td>2</td><td>3</td><td>4</td></tr><tr><td>3</td><td>2</td><td>3</td></tr><tr><td>4</td><td>3</td><td>4</td></tr><tr><td>5</td><td>4</td><td>4</td></tr></table>	A	C	D	1	3	4	2	3	4	3	2	3	4	3	4	5	4	4
A	B																														
1	1																														
2	1																														
3	2																														
4	1																														
5	1																														
A	C	D																													
1	3	4																													
2	3	4																													
3	2	3																													
4	3	4																													
5	4	4																													

Although we passed F_1 and F_2 , it doesn't mean that we passed all FDs in F !

It is because **we lost the FD $B \rightarrow CD$ in the decomposition.**

Dependency preserving



What is the condition(s) for a decomposition to be **dependency preserving**?

Let F be a set of functional dependencies on R .

- R_1, R_2, \dots, R_n be a decomposition of R .
- F_i be the set of FDs in F^+ that include only attributes in R_i .

A decomposition is **dependency preserving** if and only if

$$(F_1 \cup F_2 \cup \dots \cup F_n)^+ = F^+$$

- Where F_i is the set of FDs in F^+ that include only attributes in R_i .

Example 1

- Given $R(A, B, C)$, $F = \{A \rightarrow B, B \rightarrow C\}$
 - Is $R_1(A, B)$, $R_2(B, C)$ a dependency preserving decomposition?
- First we need to find F^+ , F_1 and F_2 .
 - $F^+ = \{A \rightarrow B, B \rightarrow C, A \rightarrow C, \text{some trivial FDs}\}$
 - $F_1 = \{A \rightarrow B \text{ and trivial FDs}\}$
 - $F_2 = \{B \rightarrow C \text{ and trivial FDs}\}$

Note that $A \rightarrow C$ is in F^+ because of the **Transitivity axiom**.
- Then we check if $(F_1 \cup F_2)^+ = F^+$ is true.
 - Since $F_1 \cup F_2 = F$, this implies $(F_1 \cup F_2)^+ = F^+$.
- This decomposition is dependency preserving.

Example 2

- Given $R(A, B, C)$, $F = \{A \rightarrow B, B \rightarrow C\}$
 - Is $R_1(A, B)$, $R_2(A, C)$ a dependency preserving decomposition?
- First we need to find F^+ , F_1 and F_2 .
 - $F^+ = \{A \rightarrow B, B \rightarrow C, A \rightarrow C, \text{some trivial FDs}\}$
 - $F_1 = \{A \rightarrow B \text{ and trivial FDs}\}$
 - $F_2 = \{A \rightarrow C \text{ and trivial FDs}\}$

Note that $A \rightarrow C$ is in F^+ because of the **Transitivity axiom**.
- Then we check if $(F_1 \cup F_2)^+ = F^+$ is true.
 - Since $B \rightarrow C$ disappears in R_1 and R_2 , $(F_1 \cup F_2)^+ \neq F^+$.
- This decomposition is NOT dependency preserving.

Section 3

Boyce-Codd

Normal Form

FD and redundancy

Consider the following relation:

- Customer(*id*, *name*, *dptID*)
- $F = \{ \{id\} \rightarrow \{name, dptID\} \}$

Customer

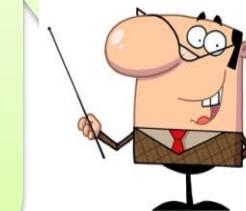
id	name	dptID
1	Kit	1
2	David	1
3	Betty	2
4	Helen	2

{*id*} is a key in Customer.

- Because the attribute closure of {*id*} (i.e., $\{id\}^+ = \{id, name, dptID\}$), which covers all attributes of Customer.

Observation: All non-trivial FDs in F form a key in the relation Customer.

- This implies that there are no other FD that is just involve a subset of columns in the relation.
- This implies that Customer has **no redundancy**.



FD and redundancy



As another example:

- Customer(*id*, *name*, *dptID*, *building*)
- $F = \{ \{id\} \rightarrow \{name, dptID, building\}$
 $\{dptID\} \rightarrow \{building\} \}$

Customer

id	name	dptID	building
1	Kit	1	CYC
2	David	1	CYC
3	Betty	2	HW
4	Helen	2	HW



$\{dptID\} \rightarrow \{building\}$ brings redundancy. Why?

- Tuples have the same *dptID* must have the same *building* (e.g., *dptID=1*, *building*=“CYC”).
- But those tuples can have different values in *id* and *name*. For each different *id* values with the same *dptID*, *building* will be repeated (**redundancy**)



For example, for tuples with (*id=1*, *dptID=1*) and (*id=2*, *dptID=1*), *building* must equal “CYC” (redundancy).

FD and redundancy



As another example:

- Customer(*id*, *name*, *dptID*, *building*)
- $F = \{ \{id\} \rightarrow \{name, dptID, building\}$
 $\{dptID\} \rightarrow \{building\} \}$



How to check?

- Check if the attribute set closure of $\{dptID\}$ covers all attributes in Customer. ($\{dptID\}^+ = \{dptID, building\} \neq \text{Customer}$)

Redundancy is related to FDs. If there is an FD $\alpha \rightarrow \beta$, where $\{\alpha\}^+$ does not cover all attributes in R, then we will have redundancy in R!

Customer

id	name	dptID	building
1	Kit	1	CYC
2	David	1	CYC
3	Betty	2	HW
4	Helen	2	HW



Boyce-Codd Normal Form

Summarizing the observations, a relation R has no redundancy, or in Boyce-Codd Normal Form (BCNF), if the following is satisfied:

- For all FDs in F^+ of the form $\alpha \rightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of the following holds:

$\alpha \rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$)

We won't bother with trivial FDs such as $A \rightarrow A$, $AB \rightarrow A$...etc

α is a key (superkey) for R

i.e., The attribute set closure of α , represented as $\{\alpha\}^+$, covers all attributes in R.

In other words, in BCNF, every non-trivial FD forms a key.



How to test for BCNF?

Formally, for verifying if R is in BCNF

- For each non-trivial dependency $\alpha \rightarrow \beta$ in F^+ (**the functional dependency closure**), check if α^+ covers the whole relation (i.e., whether α is a superkey).
- If any α^+ does not cover the whole relation, R is not in BCNF.

Simplified test:

- It suffices to check only the dependencies in the given F for violation of BCNF, rather than check all dependencies in F^+

For example, given $R(A,B,C)$; $F = \{A \rightarrow B, B \rightarrow C\}$,
we only need to check if both $\{A\}^+$ and $\{B\}^+$ cover $\{A,B,C\}$.

We do not need to derive $F^+ = \{A \rightarrow B, B \rightarrow C, A \rightarrow C, \dots\text{etc}\}$ and check each FD because $A \rightarrow C$ already considered when computing $\{A\}^+$.



How to test for BCNF?

- However, if we decompose R into R_1 and R_2 , we cannot use only F to check if the “decomposed” relations (i.e., R_1 and R_2) is BCNF, we have to use F^+ instead.

Illustration

- $R(A, B, C, D)$, $F = \{A \rightarrow B, B \rightarrow C\}$



To test if R is in BCNF, it suffices to check only the dependencies in F (but not F^+)

- $\{A\}^+$ covers all $\{A, B, C, D\}$?

Since $\{A\}^+ = \{A, B, C\} \neq \{A, B, C, D\}$, R is not in BCNF.



As illustrated through this instance, since $\{A\}^+ = \{A, B, C\} \neq \{A, B, C, D\}$, this implies that it will cause redundancy when we have tuples with the same value across $\{ABC\}$ but different values in D.

R	A	B	C	D
1	1	1	1	1
1	1	1	1	2
1	1	1	1	3
1	1	1	1	4
1	1	1	1	5

An example R that satisfies F

How to test for BCNF?



To illustrate why we cannot use only F to test decomposed relations for BCNF, let's try to **decompose R into $R_1(A, B)$ and $R_2(A, C, D)$**

Illustration

- $R(A, B, C, D), F = \{A \rightarrow B, B \rightarrow C\}$
- Is $R_2(A, C, D)$ in BCNF?



R	A	B	C	D
1	1	1	1	1
1	1	1	2	
1	1	1	3	
1	1	1	4	
1	1	1	5	

When we check R_2 , none of FDs in F is contained in R_2 . Does this mean no non-trivial FDs are in R_2 , and R_2 is in BCNF?

No! We need to use F^+ to verify if R_2 is BCNF

$R_1(A, B)$	$R_2(A, C, D)$
A B	A C D
1 1	1 1 1
	1 1 2
	1 1 3
	1 1 4
	1 1 5

How to test for BCNF?

● In $R_2(A, C, D)$, $A \rightarrow C$ is in F^+ , because:

- $A \rightarrow C$ can be obtained by **transitivity rule on $A \rightarrow B$ and $B \rightarrow C$**
- There is a non trivial FD $A \rightarrow C$ in R_2 that we have missed!

● Therefore in R_2 we check $\{A\}^+ = \{A, C\} \neq \{A, C, D\}$

- Thus, A is not a key in R_2
- R_2 is NOT in BCNF.

R	A	B	C	D
1	1	1	1	1
1	1	1	2	
1	1	1	3	
1	1	1	4	
1	1	1	5	

Conclusion: When we test whether a *decomposed relation* is in BCNF, we must project F^+ onto the relation (e.g., R_2), not F !



$R_1(A, B)$	$R_2(A, C, D)$
A B	A C D
1 1	1 1 1
	1 1 2
	1 1 3
	1 1 4
	1 1 5

Section 4

Normalization

Normalization goal

- When we decompose a relation R with a set of functional dependencies F into R_1, R_2, \dots, R_n , we try to meet the following goals:
 - 1. Lossless-join** – Avoid the decomposition result in information loss.
 - 2. No Redundancy** – The decomposed relations R_i should be in Boyce-Codd Normal Form (BCNF). (There are also other normal forms.)
 - 3. Dependency preserving** – Avoid the need to join the decomposed relations to check the functional dependencies.

Illustration

- Consider $R(A, B, C)$, $F = \{A \rightarrow B, B \rightarrow C\}$, is R in BCNF?
If not, decompose R into relations that are in BCNF.

- Is R in BCNF?

- Because $\{B\}^+ = \{B, C\} \neq \{A, B, C\}$
- Since $\{B\}^+$ does not cover all attributes in R , R is **NOT** in BCNF.

R	A	B	C
1	1	2	
2	1	2	
3	1	2	
4	1	2	

How should we decompose R such that the decomposed relations are always lossless join?

Note: A decomposition is lossless join if **at least one of the following dependencies is in F^+**

Schema of $R_1 \cap$ schema of $R_2 \rightarrow$ schema of R_1

Schema of $R_1 \cap$ schema of $R_2 \rightarrow$ schema of R_2

OR



Illustration

Idea: To make the decomposition always lossless join, we can pick the FD $A \rightarrow B$ and make the decomposed relation as:

- $R_1(A, B)$ – the attributes in the L.H.S. and R.H.S. of the FD.
 - $R_2(A, C)$ – the attribute(s) in the L.H.S. of the FD, and the remaining attributes that does not appear in R_1 .
- If we decompose the relation R in this way the following must be true:



Schema of $R_1 \cap$ schema of $R_2 \rightarrow$ schema of R_1

- Schema of $R_1 \cap$ schema of R_2 is A .
- $A \rightarrow R_1 = A \rightarrow AB$ must be true because R_1 must consists of the L.H.S. and R.H.S. of the FD $A \rightarrow B$ in F .

Illustration

$$F = \{A \rightarrow B, B \rightarrow C\} \quad F^+ = \{A \rightarrow B, B \rightarrow C, A \rightarrow C, \text{trivial FDs}\}$$

	$R_1(A, B)$	$R_2(A, C)$
F_x	$A \rightarrow B$	$A \rightarrow C$

Is $R_1(A, B)$ in BCNF?

- $F_1 = \{A \rightarrow B, \text{trivial FDs}\}$, it is a projection of F^+ on R_1 .
- Since $\{A\}^+ = \{A, B\} = R_1$, $\{A\}$ is a key in R_1 .
- Since all FDs in F_1 forms a key, R_1 is in BCNF.

R	A	B	C
1	1	2	
2	1	2	
3	1	2	
4	1	2	

Is $R_2(A, C)$ in BCNF?

- $F_2 = \{A \rightarrow C, \text{trivial FDs}\}$, it is a projection of F^+ on R_2 .
- Since $\{A\}^+ = \{A, C\} = R_2$, $\{A\}$ is a key in R_2 .
- Since all FDs in F_2 forms a key, R_2 is in BCNF.

R_1	R_2
A	A
B	C
1	1
2	2
3	2
4	2

Therefore, decomposing $R(A, B, C)$ with $F = \{A \rightarrow B, B \rightarrow C\}$ to $R_1(A, B)$ and $R_2(A, C)$ result in a lossless join decomposition (**no information lost**), and BCNF relations (**no redundancy**)

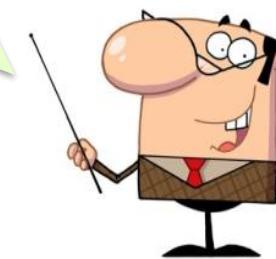


Illustration

Is the decomposition dependency preserving ?

- $F = \{A \rightarrow B, B \rightarrow C\}$
 - $(F_1 \cup F_2) = (A \rightarrow B, A \rightarrow C)$
- Since $B \rightarrow C$ disappears in R_1 and R_2 , $(F_1 \cup F_2)^+ \neq F^+$.
- The decomposition is **NOT dependency preserving.**

Note: Although the decomposition is not dependency preserving, but it is lossless join, so we can join R_1 and R_2 to test $B \rightarrow C$.



BCNF decomposition algorithm

```
result = {R};  
done = false;  
compute F+;  
while (done == false) {  
    if (there is a schema Ri in result and Ri is not in BCNF)  
        let  $\alpha \rightarrow \beta$  be a non-trivial FD that holds on Ri s.t.  $\{\alpha\}^+ \neq R_i$   
        result = (result - Ri)  $\cup$  ( $\alpha \beta$ )  $\cup$  (Ri -  $\beta$ )  
    else  
        done = true;  
}
```

α is not a key;
 $\alpha \rightarrow \beta$ causes R_i to violate BCNF

1. Delete R_i

2. Create a relation with only α and β

3. Create a relation containing R_i but with β removed.

Each R_i is in BCNF, and the decomposition must be lossless-join

Example 1

	$R_1(B, C)$	$R_2(A, B)$
F_x	$B \rightarrow C$	$A \rightarrow B$

- Consider $R(A, B, C)$, $F = \{A \rightarrow B, B \rightarrow C\}$, decompose R into relations that are in BCNF.

Alternative decomposition: To make the decomposition always lossless join, we can pick the FD $B \rightarrow C$ and make the decomposed relation as:

- $R_1(B, C)$ – the attributes in the L.H.S. and R.H.S. of the FD.
- $R_2(A, B)$ – the attribute(s) in the L.H.S. of the FD, and the remaining attributes that does not appear in R_1 .

R	A	B	C
1	1	2	
2	1	2	
3	1	2	
4	1	2	

R_1	R_2
B	A
1	1
2	1
3	1
4	1



Example 1

$$F = \{A \rightarrow B, B \rightarrow C\} \quad F^+ = \{A \rightarrow B, B \rightarrow C, A \rightarrow C, \text{trivial FDs}\}$$

	$R_1(B, C)$	$R_2(A, B)$
F_x	$B \rightarrow C$	$A \rightarrow B$

● **Decomposition: $R_1(B, C)$, $R_2(A, B)$**

● **Is $R_1(B, C)$ in BCNF?**

- $F_1 = \{B \rightarrow C, \text{trivial FDs}\}$, it is a projection of F^+ on R_1 .
- Since $\{B\}^+ = \{B, C\} = R_1$, $\{B\}$ is a key in R_1 .
- Since all FDs in F_1 forms a key, R_1 is in BCNF.

R	A	B	C
1	1	1	2
2	2	1	2
3	3	1	2
4	4	1	2

● **Is $R_2(A, B)$ in BCNF?**

- $F_2 = \{A \rightarrow B, \text{trivial FDs}\}$, it is a projection of F^+ on R_2 .
- Since $\{A\}^+ = \{A, B\} = R_2$, $\{A\}$ is a key in R_2 .
- Since all FDs in F_2 forms a key, R_2 is in BCNF.

R_1	R_2
B	A
1	1
2	1
3	1
4	1

Example 1

	$R_1(B, C)$	$R_2(A, B)$
F_x	$B \rightarrow C$	$A \rightarrow B$

Is the decomposition lossless join?

- From the illustration in example 1, the decomposition must be lossless join.

Is the decomposition dependency preserving ?

- $F = \{A \rightarrow B, B \rightarrow C\}$
- $(F_1 \cup F_2) = (B \rightarrow C, A \rightarrow B)$

Since $F = (F_1 \cup F_2)$, this implies $(F_1 \cup F_2)^+ = F^+$.

R	A	B	C
1	1	2	
2	1	2	
3	1	2	
4	1	2	

R_1	R_2
B	C
1	2
2	1
3	1
4	1

The decomposition is dependency preserving.

- That means if we insert a new tuple, if the new tuple does not violate F_1 in R_1 , and F_2 in R_2 , it won't violate F^+ in R.

Example 2



Consider a relation R in a bank:

$R(b_name, b_city, assets, c_name, l_num, amount)$

- With $F = \{ \{b_name\} \rightarrow \{assets, b_city\}, \{l_num\} \rightarrow \{amount, b_name\}, \{l_num, c_name\} \rightarrow \{\text{all other attributes}\} \}$

Each specific value in $bname$ is corresponds to at most one $\{asset, b_city\}$ value
Each l_num corresponds to at most one $\{amount, b_name\}$ value.
Each $\{l_num, c_name\}$ corresponds to at most one $\{b_name, b_city, assets, amount\}$ value.



Decomposition

- With $\{b_name\} \rightarrow \{assets, b_city\}$, $\{b_name\}^+ \neq R$,
R is not in BCNF.
- Decompose R into $R_1(b_name, assets, b_city)$ and
 $R_2(b_name, c_name, l_num, amount)$.

Example 2



Is $R_1(b_name, assets, b_city)$ in BCNF?

- $F_1 = \{ \{b_name\} \rightarrow \{assets, b_city\}, \text{trivial FDs} \}$ ← Projection of F^+ on F_1 .
- $\{b_name\}^+ = \{b_name, assets, b_city\} = R_1$,
so $\{b_name\}$ is a key in R_1 .
- Since all FD in F_1 forms a key in R_1 , R_1 is in BCNF.



Is $R_2(b_name, c_name, l_num, amount)$ in BCNF?

- $F_2 = \{ \{l_num\} \rightarrow \{amount, b_name\}, \{l_num, c_name\} \rightarrow \{\text{all attributes}\} \}$ ← Projection of F^+ on F_2 .
- $\{l_num\}^+ = \{l_num, amount, b_name\} \neq R_2$,
so $\{l_num\}$ is NOT a key in R_2 .
- Since NOT all FD in F_2 forms a key in R_2 , R_2 is NOT in BCNF.

Example 2

- Picking $\{l_num\} \rightarrow \{amount, b_name\}$, R_2 is further decomposed into:
 - $R_3(l_num, amount, b_name)$
 - $R_4(c_name, l_num)$
- Is $R_3(l_num, amount, b_name)$ in BCNF?
 - $F_3 = \{\{l_num\} \rightarrow \{amount, b_name\}\}$, trivial FDs
 - $\{l_num\}^+ = \{l_num, amount, b_name\} = R_3$, so $\{l_num\}$ is a key in R_3 .
 - Since all FD in F_3 forms a key in R_3 , R_3 is in BCNF.

Example 2

- Is $R_4(c_name, l_num)$ in BCNF?
 - $F_4 = \{\text{trivial FDs}\}$
 - Since all FD in F_4 forms a key in R_4 , R_4 is in BCNF.
- Now, R_1 , R_3 and R_4 are in BCNF;
- The decomposition is also lossless-join.

Example 2

The decomposition is also dependency preserving.

- $F_1 = \{ \{b_name\} \rightarrow \{assets, b_city\}, \text{trivial FDs}\}$
- $F_3 = \{\{I_num\} \rightarrow \{amount, b_name\}, \text{trivial FDs}\}$
 $\{I_num\} \rightarrow \{b_name\} \dots (\text{i})$
by **Decomposition of** $\{I_num\} \rightarrow \{amount, b_name\}$
 $\{I_num\} \rightarrow \{assets, b_city\} \dots (\text{ii})$
by **Transitivity of** (i) and $\{b_name\} \rightarrow \{assets, b_city\}$
 $\{I_num\} \rightarrow \{b_name, assets, b_city, amount\}$ by **Union** of F_3 and (ii)
 $\{I_num, c_name\} \rightarrow \{I_num, c_name, b_name, assets, b_city, amount\}$ by **Augmentation**
- Therefore $F_1 \cup F_3 \cup F_4 = F$, which implies $(F_1 \cup F_3 \cup F_4)^+ = F^+$.
- The decomposition is **dependency preserving**.

BCNF doesn't imply dependency preserving

- It is not always possible to get a BCNF decomposition that is dependency preserving.

- Consider $R(A, B, C)$; $F = \{ AB \rightarrow C, C \rightarrow B \}$

- There are two candidate keys:
 $\{AB\}$, and $\{AC\}$.

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

- $\{AC\}^+ = \{A, B, C\} = R$

- R is not in BCNF, since C is not a key.

- Decomposition of R must fail to preserve $AB \rightarrow C$.

R	A	B	C
1	1	2	
2	1	2	
1	2	3	

R_1	R_2	
A B	B C	
1 1	1 2	
2 1	2 3	
1 2		

Not lossless decomposition

R_1	R_2	
A B	A C	
1 1	1 2	
2 1	2 2	
1 1	1 3	

Not lossless decomposition

R_1	R_2	
A C	B C	
1 2	1 2	
2 2	2 3	
1 3		

lossless
 $F_1 = \{\emptyset\}$
 $F_2 = \{C \rightarrow B\}$

Not dependency preserving

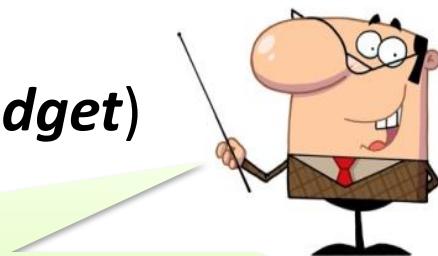
Motivating example

- Back to our motivating example, we have:
 - Employees(*eid*, *name*, *parkingLot*, *did*, *since*)
 - Departments(*did*, *dname*, *budget*)
- “Employees who work in the same department must park at the same *parkingLot*.” implies the following FD:

FD: $\text{did} \rightarrow \text{parkingLot}$
- Is Employees in BCNF?
 - $\{\text{did}\}^+ = \{\text{parkingLot}\} \neq \{\text{eid}, \text{name}, \text{parkingLot}, \text{did}, \text{since}\}$
 - Since ***did*** is not a key, Employees is NOT in BCNF.

Normalization

- Employees(*eid*, *name*, *parkingLot*, *did*, *since*) is decomposed to
 - Employees2(*eid*, *name*, *did*, *since*)
 - Dept_Lots(*did*, *parkingLot*)
- With Departments(*did*, *dname*, *budget*), the above two decomposed relations are further refined to
 - Employees2(*eid*, *name*, *did*, *since*)
 - Departments(*did*, *dname*, *parkingLot*, *budget*)



Good design: parking lots for all employees can be updated by changing their department-specific *parkingLot*.

Summary

- **Relational database design goals**
 - Lossless-join
 - No redundancy (BCNF)
 - Dependency preservation
- **It is not always possible to satisfy the three goals.**
 - A lossless join, dependency preserving decomposition into BCNF may not always be possible.
- **SQL does not provide a direct way of specifying FDs other than superkeys.**
 - Can use assertions to check FD, but it is quite expensive.

Question 1

Given a relation $R(A,B,C)$ and a functional dependency $\{A \rightarrow B\}$ that holds on R , which of the following statements is/are correct?

- a. $\{A,C\}$ is a candidate key for R .
- b. The decomposition of R into $R_1(A,B)$ and $R_2(A,C)$ is a lossless-join decomposition.
- c. The decomposition of R into $R_1(A,B)$ and $R_2(B,C)$ is a lossless-join decomposition.

Question 1

Given a relation $R(A,B,C)$ and a functional dependency $\{A \rightarrow B\}$ that holds on R , which of the following statements is/are correct?

- a. $\{A,C\}$ is a candidate key for R .
- b. The decomposition of R into $R_1(A,B)$ and $R_2(A,C)$ is a lossless-join decomposition.
- c. The decomposition of R into $R_1(A,B)$ and $R_2(B,C)$ is a lossless-join decomposition.

Question: How to test if $\{A,C\}$ is a candidate key of R or not?

Question 1

Given a relation $R(A,B,C)$ and a functional dependency $\{A \rightarrow B\}$ that holds on R , which of the following statements is/are correct?

- a. $\{A,C\}$ is a candidate key for R .
- b. The decomposition of R into $R_1(A,B)$ and $R_2(A,C)$ is a lossless-join decomposition.
- c. The decomposition of R into $R_1(A,B)$ and $R_2(B,C)$ is a lossless-join decomposition.

Question: How to test if $\{A,C\}$ is a candidate key of R or not?

Answer:

- 1) **[Superkey]** A candidate key is a superkey. If $\{A,C\}$ is a superkey, its attribute closure $\{A,C\}^+$ must contain all attributes in R !
- 2) **[Minimal]** If $\{A,C\}$ is minimal, it is a candidate key!

Is $\{A,C\}$ a superkey?

- 1) Find attribute closure of A : $\{A\}^+ = \{A,B\}$. It is not a superkey.
- 2) Find attribute closure of C : $\{C\}^+ = \{C\}$. It is not a superkey.
- 3) Find attribute closure of $\{A,C\}$: $\{A,C\}^+ = \{A,B,C\}$. Since it contains all attributes of R , **it is a superkey**.

Is $\{A,C\}$ minimal?

- 4) Since none of subset of $\{A,C\}$ is a key, $\{A,C\}$ is minimal, **it is a candidate key**.

Question 1

Given a relation $R(A,B,C)$ and a functional dependency $\{A \rightarrow B\}$ that holds on R , which of the following statements is/are correct?

- a. $\{A,C\}$ is a candidate key for R .
- b. The decomposition of R into $R_1(A,B)$ and $R_2(A,C)$ is a lossless-join decomposition.
- c. The decomposition of R into $R_1(A,B)$ and $R_2(B,C)$ is a lossless-join decomposition.

Question: What do we mean by lossless-join decomposition?

Answer: A decomposition is **lossless-join decomposition** if the original relation can be obtained after joining the decomposed relations.

Question: How to test if a decomposition is lossless-join decomposition?

Answer: A decomposition is lossless-join decomposition iff **at least one of the following dependencies is in F^+ :**
1) common attribute of R_1 and $R_2 \rightarrow$ schema of R_1
2) common attribute of R_1 and $R_2 \rightarrow$ schema of R_2

1) Common attribute of R_1 and R_2 :

Question 1

Given a relation $R(A,B,C)$ and a functional dependency $\{A \rightarrow B\}$ that holds on R , which of the following statements is/are correct?

- a. $\{A,C\}$ is a candidate key for R .
- b. The decomposition of R into $R_1(A,B)$ and $R_2(A,C)$ is a lossless-join decomposition.
- c. The decomposition of R into $R_1(A,B)$ and $R_2(B,C)$ is a lossless-join decomposition.

Question: What do we mean by lossless-join decomposition?

Answer: A decomposition is **lossless-join decomposition** if the original relation can be obtained after joining the decomposed relations.

Question: How to test if a decomposition is lossless-join decomposition?

Answer: A decomposition is lossless-join decomposition iff **at least one of the following dependencies is in F^+ :**
1) common attribute of R_1 and $R_2 \rightarrow$ schema of R_1
2) common attribute of R_1 and $R_2 \rightarrow$ schema of R_2

- 1) Common attribute of R_1 and R_2 : **A**
- 2) $A \rightarrow AB$ in F^+ ? **✓**

Since $A \rightarrow B$, $A \rightarrow AB$ is true (Augmentation), $A \rightarrow AB$ in F^+ !

- 3) Therefore, the decomposition is a lossless-join decomposition.

Question 1

Given a relation $R(A,B,C)$ and a functional dependency $\{A \rightarrow B\}$ that holds on R , which of the following statements is/are correct?

- a. $\{A,C\}$ is a candidate key for R .
- b. The decomposition of R into $R_1(A,B)$ and $R_2(A,C)$ is a lossless-join decomposition.
- c. The decomposition of R into $R_1(A,B)$ and $R_2(B,C)$ is a lossless-join decomposition.

Question: How to test if a decomposition is lossless-join decomposition?

1) Common attribute of R_1 and R_2 :

Answer: A decomposition is lossless-join decomposition iff **at least one of the following dependencies is in F^+ :**

- 1) common attribute of R_1 and $R_2 \rightarrow$ schema of R_1**
- 2) common attribute of R_1 and $R_2 \rightarrow$ schema of R_2**

Question 1

Given a relation $R(A,B,C)$ and a functional dependency $\{A \rightarrow B\}$ that holds on R , which of the following statements is/are correct?

- a. $\{A,C\}$ is a candidate key for R .
- b. The decomposition of R into $R_1(A,B)$ and $R_2(A,C)$ is a lossless-join decomposition.
- c. The decomposition of R into $R_1(A,B)$ and $R_2(B,C)$ is a lossless-join decomposition.

Question: How to test if a decomposition is lossless-join decomposition?

- 1) Common attribute of R_1 and R_2 : **B**
- 2) $B \rightarrow AB$ in F^+ ? **X**
- 3) $B \rightarrow BC$ in F^+ ? **X**
- 4) Therefore, the decomposition is **NOT** a lossless-join decomposition.

Answer: A decomposition is lossless-join decomposition iff **at least one of the following dependencies is in F^+ :**
1) common attribute of R_1 and $R_2 \rightarrow$ schema of R_1
2) common attribute of R_1 and $R_2 \rightarrow$ schema of R_2

Question 2 a

Consider the relation $R(A,B,C,D,E)$ with the following functional dependencies $F = \{A \rightarrow E, BCE \rightarrow D, CD \rightarrow BE, C \rightarrow AB, BD \rightarrow AC\}$ which hold in it.

i) Which of the following attributes is NOT in the attribute closure of C (i.e., C^+)?

- A. A.
- B. B.
- C. D.
- D. E.
- E. None of the above.

Question: What do we mean by the attribute closure of C?

Answer: The attribute closure of C is the set of attributes that can be functionally determined by C.

Question 2 a

Consider the relation $R(A,B,C,D,E)$ with the following functional dependencies $F = \{A \rightarrow E, BCE \rightarrow D, CD \rightarrow BE, C \rightarrow AB, BD \rightarrow AC\}$ which hold in it.

- i) Which of the following attributes is NOT in the attribute closure of C (i.e., C^+)?

✓ A. A.

B. B.

C. D.

D. E.

E. None of the above.

Question: What do we mean by the attribute closure of C?

Answer: The attribute closure of C is the set of attributes that can be functionally determined by C.

Question: $C \rightarrow A$ holds?

1. Since $C \rightarrow AB$,
2. $C \rightarrow A$ and $C \rightarrow B$ (decomposition)

Question 2 a

Consider the relation $R(A,B,C,D,E)$ with the following functional dependencies $F = \{A \rightarrow E, BCE \rightarrow D, CD \rightarrow BE, C \rightarrow AB, BD \rightarrow AC\}$ which hold in it.

- i) Which of the following attributes is NOT in the attribute closure of C (i.e., C^+)?

- A. A.
- B. B.
- C. D.
- D. E.
- E. None of the above.

Question: $C \rightarrow B$ holds?

Question: What do we mean by the attribute closure of C?

Answer: The attribute closure of C is the set of attributes that can be functionally determined by C.

1. Since $C \rightarrow AB$,
2. $C \rightarrow A$ and $C \rightarrow B$ (decomposition)

Question 2 a

Consider the relation $R(A,B,C,D,E)$ with the following functional dependencies $F = \{A \rightarrow E, BCE \rightarrow D, CD \rightarrow BE, C \rightarrow AB, BD \rightarrow AC\}$ which hold in it.

- i) Which of the following attributes is NOT in the attribute closure of C (i.e., C^+)?

- A. A.
- B. B.
- C. D.
- D. E.
- E. None of the above.

Question: What do we mean by the attribute closure of C?

Answer: The attribute closure of C is the set of attributes that can be functionally determined by C.

Question: $C \rightarrow D$ holds?

Think in this way :

1. Since we have $BCE \rightarrow D$, can we have $C \rightarrow BCE$?
2. We have $C \rightarrow B$, therefore $C \rightarrow BC$ (augmentation), can we show that $C \rightarrow E$?
3. Since $C \rightarrow A$ and $A \rightarrow E$, $C \rightarrow E$ (transitivity)
4. Since $C \rightarrow BC$ and $C \rightarrow E$, $C \rightarrow BCE$ (union)
5. Since $C \rightarrow BCE$ and $BCE \rightarrow D$, $C \rightarrow D$ (transitivity)

Question 2 a

Consider the relation $R(A,B,C,D,E)$ with the following functional dependencies $F = \{A \rightarrow E, BCE \rightarrow D, CD \rightarrow BE, C \rightarrow AB, BD \rightarrow AC\}$ which hold in it.

- i. Which of the following attributes is NOT in the attribute closure of C (i.e., C^+)?

- A. A.
- B. B.
- C. D.
- D. E.
- E. None of the above.

Question: What do we mean by the attribute closure of C?

Answer: The attribute closure of C is the set of attributes that can be functionally determined by C.

Question: $C \rightarrow E$ holds?

Think in this way :

1. Since we have $BCE \rightarrow D$, can we have $C \rightarrow BCE$?
2. We have $C \rightarrow B$, therefore $C \rightarrow BC$ (augmentation), can we show that $C \rightarrow E$?
3. Since $C \rightarrow A$ and $A \rightarrow E$, $C \rightarrow E$ (transitivity) (circled)
4. Since $C \rightarrow BC$ and $C \rightarrow E$, $C \rightarrow BCE$ (union)
5. Since $C \rightarrow BCE$ and $BCE \rightarrow D$, $C \rightarrow D$ (transitivity)

Question 2 a

- Since $C \rightarrow AB$,
- $C \rightarrow A$ and $C \rightarrow B$ (decomposition)
- Since $C \rightarrow A$ and $A \rightarrow E$,
- $C \rightarrow E$ (transitivity)
- Since $C \rightarrow B$ and $C \rightarrow E$,
- $C \rightarrow BE$ (union)
- Since $C \rightarrow BE$,
- $C \rightarrow BCE$ (augmentation)
- Since $BCE \rightarrow D$ and $C \rightarrow BCE$,
- $C \rightarrow D$ (transitivity)
- Hence, $C^+ = \{A, B, C, D, E\}$
- Answer: **E**

Question 2 b

Consider the relation $R(A,B,C,D,E)$ with the following functional dependencies $F = \{A \rightarrow E, BCE \rightarrow D, CD \rightarrow BE, C \rightarrow AB, BD \rightarrow AC\}$ which hold in it.

- Which of the following functional dependencies is NOT in the closure of F (i.e., F^+)?

- i. $BE \rightarrow D$
- ii. $BD \rightarrow ABCDE$
- iii. $AB \rightarrow D$
- iv. $CE \rightarrow A$

Question: What do we mean by the FD closure?

Answer: The set of all functional dependencies that can be logically implied by F is called the closure of F (or F^+).

Question 2 b

Consider the relation $R(A,B,C,D,E)$ with the following functional dependencies $F = \{A \rightarrow E, BCE \rightarrow D, CD \rightarrow BE, C \rightarrow AB, BD \rightarrow AC\}$ which hold in it.

- Which of the following functional dependencies is NOT in the closure of F (i.e., F^+)?

- ✗ i. **BE \rightarrow D**
- ii. BD \rightarrow ABCDE
- iii. AB \rightarrow D
- iv. CE \rightarrow A

Question: What do we mean by the FD closure?

Answer: The set of all functional dependencies that can be logically implied by F is called the closure of F (or F^+).

Question 2 b

Consider the relation $R(A,B,C,D,E)$ with the following functional dependencies $F = \{A \rightarrow E, BCE \rightarrow D, CD \rightarrow BE, C \rightarrow AB, BD \rightarrow AC\}$ which hold in it.

- Which of the following functional dependencies is NOT in the closure of F (i.e., F^+)?

- i. $BE \rightarrow D$
- ii. $BD \rightarrow ABCDE$
- iii. $AB \rightarrow D$
- iv. $CE \rightarrow A$

Question: What do we mean by the FD closure?

Answer: The set of all functional dependencies that can be logically implied by F is called the closure of F (or F^+).

Think in this way :

1. Start from $BD \rightarrow AC$, can we make R.H.S. more close to $ABCDE$?
2. $BD \rightarrow ABCD$ (augmentation), can we show that $ABCD \rightarrow ABCDE$?
3. Since $A \rightarrow E$, we have $ABCD \rightarrow ABCDE$ (augmentation)
4. Therefore, we can show that $BD \rightarrow ABCDE$ is true!

Question 2 b

Consider the relation $R(A,B,C,D,E)$ with the following functional dependencies $F = \{A \rightarrow E, BCE \rightarrow D, CD \rightarrow BE, C \rightarrow AB, BD \rightarrow AC\}$ which hold in it.

- Which of the following functional dependencies is NOT in the closure of F (i.e., F^+)?

- i. $BE \rightarrow D$
- ii. $BD \rightarrow ABCDE$
- iii. $AB \rightarrow D$
- iv. $CE \rightarrow A$

Question: What do we mean by the FD closure?

Answer: The set of all functional dependencies that can be logically implied by F is called the closure of F (or F^+).

Question 2 b

Consider the relation $R(A,B,C,D,E)$ with the following functional dependencies $F = \{A \rightarrow E, BCE \rightarrow D, CD \rightarrow BE, C \rightarrow AB, BD \rightarrow AC\}$ which hold in it.

- Which of the following functional dependencies is NOT in the closure of F (i.e., F^+)?

- i. $BE \rightarrow D$
- ii. $BD \rightarrow ABCDE$
- iii. $AB \rightarrow D$
- iv. $CE \rightarrow A$

Question: What do we mean by the FD closure?

Answer: The set of all functional dependencies that can be logically implied by F is called the closure of F (or F^+).

Think in this way :

1. We have $C \rightarrow AB$, and therefore we have $C \rightarrow A$ (decomposition); **can we show that $CE \rightarrow C$?**
2. **$CE \rightarrow C$ is always true due to reflexivity!**
3. Therefore, we have $CE \rightarrow C$ and $C \rightarrow A$, we can show that $CE \rightarrow A$ is true (transitivity)!

Question 2 c

Consider the relation $R(A,B,C,D,E)$ with the following functional dependencies $F = \{A \rightarrow E, BCE \rightarrow D, CD \rightarrow BE, C \rightarrow AB, BD \rightarrow AC\}$ which hold in it.

- Which of the following is/are candidate key(s) of R?
 - i. {B,E}
 - ii. {B,D}
 - iii. {C,E}

Question: How to show if a set of attribute is a **candidate key** or not?

Question 2 c

Consider the relation $R(A,B,C,D,E)$ with the following functional dependencies $F = \{A \rightarrow E, BCE \rightarrow D, CD \rightarrow BE, C \rightarrow AB, BD \rightarrow AC\}$ which hold in it.

- Which of the following is/are candidate key(s) of R ?



i. $\{B,E\}$

ii. $\{B,D\}$

iii. $\{C\}$

Question: How to show if a set of attribute is a **candidate key** or not?

Answer: 1) It has to be a super key.
2) It has to be minimal.

Question: Does $\{B,E\}^+$ covers all attributes in R ? i.e. $BE \rightarrow ABCDE$?

Question 2 c

Consider the relation $R(A,B,C,D,E)$ with the following functional dependencies $F = \{A \rightarrow E, BCE \rightarrow D, CD \rightarrow BE, C \rightarrow AB, BD \rightarrow AC\}$ which hold in it.

- Which of the following is/are candidate key(s) of R ?

 i. $\{B, E\}$

 ii. $\{B, D\}$

iii. $\{C, D\}$

Question: How to show if a set of attribute is a **candidate key** or not?

Answer: 1) It has to be a super key.
2) It has to be minimal.

Question: Does $\{B, D\}^+$ covers all attributes in R ? i.e. $BD \rightarrow ABCDE$?

Think in this way :

1. $BD \rightarrow BD$ must be true, therefore, we have to show
1) $BD \rightarrow A$, 2) $BD \rightarrow C$, and 3) $BD \rightarrow E$.
2. Since $BD \rightarrow AC$, $BD \rightarrow A$ and $BD \rightarrow C$ are true (**decomposition**)
3. How about $BD \rightarrow E$? We have $A \rightarrow E$, can we show that $BD \rightarrow A$?
4. $BD \rightarrow A$ already shown to be true, so now we have $BD \rightarrow A$ and $A \rightarrow E$, therefore **$BD \rightarrow E$ is true!**

Question 2 c

Consider the relation $R(A,B,C,D,E)$ with the following functional dependencies $F = \{A \rightarrow E, BCE \rightarrow D, CD \rightarrow BE, C \rightarrow AB, BD \rightarrow AC\}$ which hold in it.

- Which of the following is/are candidate key(s) of R?

- i. {B,E}
- ii. {B,D}
- iii. {C,E}

Question: How to show if a set of attribute is a **candidate key** or not?

Answer: 1) It has to be a super key.
2) It has to be minimal.

1. Since we have already shown in part a) that $\{C\}^+$ covers all attributes of R, therefore C is a candidate key of R.
2. Hence, {C,E} must NOT be a candidate key of R because {C,E} is not minimal.

Question 2 d

Consider the relation $R(A,B,C,D,E)$ with the following functional dependencies $F = \{A \rightarrow E, BCE \rightarrow D, CD \rightarrow BE, C \rightarrow AB, BD \rightarrow AC\}$ which hold in it.

- Suppose that relation R is decomposed into $R_1(A,B,C,D)$ and $R_2(A,C,E)$. Which of the following statements is/are correct?
 - i. This is a lossless-join decomposition.
 - ii. The decomposition is dependency preserving.
 - iii. R_1 and R_2 are in BCNF.

Question 2 d

Consider the relation $R(A,B,C,D,E)$ with the following functional dependencies $F = \{A \rightarrow E, BCE \rightarrow D, CD \rightarrow BE, C \rightarrow AB, BD \rightarrow AC\}$ which hold in it.

- Suppose that relation R is decomposed into $R_1(A,B,C,D)$ and $R_2(A,C,E)$. Which of the following statements is/are correct?

✓ i. This is a lossless-join decomposition.

ii. The decomposition is dependency preserving.

iii. R_1 and R_2 are in BCNF.

Question: How to test if a decomposition is lossless-join decomposition?

Answer: A decomposition is lossless-join decomposition iff **at least one of the following dependencies is in F^+ :**

- 1) common attribute of R_1 and $R_2 \rightarrow$ schema of R_1
- 2) common attribute of R_1 and $R_2 \rightarrow$ schema of R_2

1. Common attribute of R_1 and R_2 is $\{A,C\}$.
2. Test if any of the following FD are in F^+ :
 1. $AC \rightarrow ABCD$
 2. $AC \rightarrow ACE$
3. Since $A \rightarrow E$, $AC \rightarrow ACE$ is true (augmentation).
4. Therefore, this is a lossless-join decomposition.

Question 2 d

Consider the relation $R(A,B,C,D,E)$ with the following functional dependencies $F = \{A \rightarrow E, BCE \rightarrow D, CD \rightarrow BE, C \rightarrow AB, BD \rightarrow AC\}$ which hold in it.

- Suppose that relation R is decomposed into $R_1(A,B,C,D)$ and $R_2(A,C,E)$. Which of the following statements is/are correct?

i. This is a lossless-join decomposition.

ii. The decomposition is dependency preserving.

iii. R_1 and R_2 are in BCNF.

Question: How to test if a decomposition is dependency preserving or not?

- Determine F_1 and F_2 , the projection of F^+ on R_1 and R_2
 - $F_1 = \{CD \rightarrow B, C \rightarrow ABD, BD \rightarrow AC\}$.
 - $F_2 = \{A \rightarrow E, C \rightarrow AE\}$.
- Although $\{BCE \rightarrow D, CD \rightarrow BE\}$ are missing, but we know that $\{C\}^+ = R$, so we have $C \rightarrow ABD$ in F_1 and $C \rightarrow AE$ in F_2
 - $BCE \rightarrow D$ and $CD \rightarrow BE$ both have C on L.H.S., so these two FDs are in $(F_1 \cup F_2)^+$
- Therefore $(F_1 \cup F_2)^+ = F^+$.
- Therefore, this decomposition is dependency preserving.

Answer: A decomposition is dependency preserving iff:

$$(F_1 \cup F_2 \cup \dots \cup F_n)^+ = F^+$$

Where F_i is the set of FDs in F^+ that include only attributes in R_i .

Question 2 d

Consider the relation $R(A,B,C,D,E)$ with the following functional dependencies $F = \{A \rightarrow E, BCE \rightarrow D, CD \rightarrow BE, C \rightarrow AB, BD \rightarrow AC\}$ which hold in it.

- Suppose that relation R is decomposed into $R_1(A,B,C,D)$ and $R_2(A,C,E)$. Which of the following statements is/are correct?



i. This is a lossless-join decomposition.



ii. The decomposition is dependency preserving.

iii. R_1 and R_2 are in BCNF.

Question: How to test if the decomposed relations are in BCNF?

Test if R_1 is in BCNF:

- Determine F_1 , the projection of F^+ on R_1 .
 - $F_1 = \{CD \rightarrow B, C \rightarrow AB, BD \rightarrow AC, C \rightarrow D\}$.
- Determine if all FDs in F_1 forms a key of R_1 .
 - Is $\{CD\}$ a key in R_1 ? Yes! Because $\{CD\}^+ = \{ABCD\}$ in R_1 , contains all attributes in R_1 .
 - Is $\{C\}$ a key in R_1 ? Yes! Because $\{C\}^+ = \{ABCD\}$ in R_1 , contains all attributes in R_1 .
 - Is $\{BD\}$ a key in R_1 ? Yes! Because $\{BD\}^+ = \{ABCD\}$, contains all attributes in R_1 .
- Since all non-trivial FDs in F_1 forms a key, R_1 is in BCNF.

Answer: In BCNF, every non-trivial FD forms a key!

Question 2 d

Consider the relation $R(A,B,C,D,E)$ with the following functional dependencies $F = \{A \rightarrow E, BCE \rightarrow D, CD \rightarrow BE, C \rightarrow AB, BD \rightarrow AC\}$ which hold in it.

- Suppose that relation R is decomposed into $R_1(A,B,C,D)$ and $R_2(A,C,E)$. Which of the following statements is/are correct?

✓ i. This is a lossless-join decomposition.

✓ ii. The decomposition is dependency preserving.

✗ iii. R_1 and R_2 are in BCNF.

Question: How to test if the decomposed relations are in BCNF?

Test if R_2 is in BCNF:

- Determine F_2 , the projection of F^+ on R_2 .
 - $F_2 = \{A \rightarrow E, C \rightarrow A\}$.
- Determine if all FDs in F_2 forms a key of R_2 .
 - Is $\{A\}$ a key in R_2 ? **No!** Because $\{A\}^+ = \{AE\}$, not covering all attributes in R_2 .
- Since **not all non-trivial** FDs in F_2 forms a key, R_2 is NOT in BCNF.

Answer: In BCNF, every non-trivial FD forms a key!

Question 3

Consider the schema $R(A, B, C, D, E)$ and the set of functional dependencies $F = \{ D \rightarrow A, C \rightarrow BDE \}$ which holds in the schema.

- a) Find all candidate keys of R . Show your steps.
- b) Give a lossless join decomposition of R into relations in BCNF. Is the decomposition dependency preserving?

Question 3

Consider the schema $R(A, B, C, D, E)$ and the set of functional dependencies $F = \{ D \rightarrow A, C \rightarrow BDE \}$ which holds in the schema.

- a) Find all candidate keys of R . Show your steps.
 - Is C a candidate key?
 - Since C^+ is $\{A, B, C, D, E\}$, C is a **superkey**, and since C is **minimal**, it is a candidate key.
 - Since C is a candidate key, any supersets of C are not candidate keys.
 - How about the combinations of other attributes?
 - C does not appear on the RHS of any non-trivial FD, therefore the value of C cannot be functionally determined by other attribute(s).
 - All combinations of other attributes (i.e. A, B, D , and E) cannot form a key. (Their attribute closure must not contain C)
 - Thus, R has only one candidate key, which is C .

Question 3

Consider the schema $R(A, B, C, D, E)$ and the set of functional dependencies $F = \{ D \rightarrow A, C \rightarrow BDE \}$ which holds in the schema.

b) Give a lossless join decomposition of R into relations in BCNF. Is the decomposition dependency preserving?

■ $R_1(A, D)$, $R_2(B, C, D, E)$

■ **R1 in BCNF ?**

- F_1 is the projection of F^+ on R_1 : $F_1 = \{D \rightarrow A\}$, and the trivial FDs
- Is D a key in R_1 ?
- Since D is a key in R_1 , **all non-trivial FDs in F_1 forms a key, thus R_1 is in BCNF.**

■ **R2 in BCNF?**

- F_2 is the projection of F^+ on R_2 : $F_2 = \{C \rightarrow BDE\}$, and the trivial FDs
- Is C a key in R_2 ?
- Since C is a key in R_2 , **all non-trivial FDs in F_2 forms a key, thus R_2 is in BCNF.**

Step 1. Decompose R into two relations R_1 and R_2 , and make R_1 a BCNF.

Look at $D \rightarrow A$ in F , make R_1 as (A, D) .

Then R_2 is (B, C, D, E) .

Why include D in R_2 ?

Step 2. Check if R_1 and R_2 are in BCNF. If not, further decompose it using Step 1.

In BCNF, every non-trivial FD forms a key!

Question 3

Consider the schema $R(A, B, C, D, E)$ and the set of functional dependencies $F = \{ D \rightarrow A, C \rightarrow BDE \}$ which holds in the schema.

b) Give a lossless join decomposition of R into relations in BCNF. Is the decomposition dependency preserving?

■ $R_1(A, D), R_2(B, C, D, E)$

■ **R1 in BCNF ?**

- F_1 is the projection of F^+ on R_1 : $F_1 = \{D \rightarrow A\}$, and the trivial FDs
- Is D a key in R_1 ?
- Since D is a key in R_1 , **all non-trivial FDs in F_1 forms a key, thus R_1 is in BCNF.**

■ **R2 in BCNF?**

- F_2 is the projection of F^+ on R_2 : $F_2 = \{C \rightarrow BDE\}$, and the trivial FDs
- Is C a key in R_2 ?
- Since C is a key in R_2 , **all non-trivial FDs in F_2 forms a key, thus R_2 is in BCNF.**

■ **Is it a lossless-join decomposition?**

- Common attribute among R_1 and R_2 = D
- $D \rightarrow R_1$ (i.e. AD) holds in F^+
- Therefore it is a lossless-join decomposition.

Answer: A decomposition is lossless-join decomposition iff

at least one of the following dependencies is in F^+ :

- 1) common attribute of R_1 and $R_2 \rightarrow$ schema of R_1**
- 2) common attribute of R_1 and $R_2 \rightarrow$ schema of R_2**

Question 3

Consider the schema $R(A, B, C, D, E)$ and the set of functional dependencies $F = \{D \rightarrow A, C \rightarrow BDE\}$ which holds in the schema.

b) Give a lossless join decomposition of R into relations in BCNF. Is the decomposition dependency preserving?

- $R_1(A, D), R_2(B, C, D, E)$

■ **R1 in BCNF ?**

- F_1 is the projection of F^+ on R_1 : $F_1 = \{D \rightarrow A, \text{and the trivial FDs}\}$
- Is D a key in R_1 ?
- Since D is a key in R_1 , **all non-trivial FDs in F_1 forms a key, thus R_1 is in BCNF.**

■ **R2 in BCNF?**

- F_2 is the projection of F^+ on R_2 : $F_2 = \{C \rightarrow BDE, \text{and the trivial FDs}\}$
- Is C a key in R_2 ?
- Since C is a key in R_2 , **all non-trivial FDs in F_2 forms a key, thus R_2 is in BCNF.**

■ **Is it a lossless-join decomposition?**

- Common attribute among R_1 and R_2 = D
- $D \rightarrow R_1$ (i.e. AD) holds in F^+
- Therefore it is a lossless-join decomposition.

■ **Is it a dependency preserving decomposition?**

- Since $(F_1 \cup F_2)^+ = F^+$, it is a dependency preserving decomposition.

Answer: A decomposition is dependency preserving iff:

$$(F_1 \cup F_2 \cup \dots \cup F_n)^+ = F^+$$

Where F_i is the set of FDs in F^+ that include only attributes in R_i .

Question 4

Given $R = (A, B, C, D)$, $F = \{A \rightarrow B, BC \rightarrow D, D \rightarrow A\}$

- a) Find A^+ and C^+ .
- b) Find a candidate key of R .
- c) Is $B \rightarrow D$ in F^+ ?
- d) Is $D \rightarrow BC$ in F^+ ?
- e) Is $AC \rightarrow D$ in F^+ ?
- f) R is decomposed into $R_1(A, B, C)$ and $R_2(B, C, D)$. Is the decomposition lossless? Is the dependency preserving?
- g) Is R in BCNF? If not, decompose R into relations in BCNF.

Question 4

Given $R=(A, B, C, D)$, $F=\{A \rightarrow B, BC \rightarrow D, D \rightarrow A\}$

a) Find A^+ and C^+ .

- $A^+ = \{AB\}$
- $C^+ = \{C\}$

b) Find a candidate key of R .

- AC or BC or CD

Question 4

Given $R = (A, B, C, D)$, $F = \{A \rightarrow B, BC \rightarrow D, D \rightarrow A\}$

c) Is $B \rightarrow D$ in F^+ ?

- No. Because $B^+ = \{B\}$

d) Is $D \rightarrow BC$ in F^+ ?

- No. Because $D^+ = \{ABD\}$

e) Is $AC \rightarrow D$ in F^+ ?

- Yes. Because $A \rightarrow B$ and $BC \rightarrow D$, we know that $AC \rightarrow D$ (pseudo-transitivity)

Question 4

Given $R = (A, B, C, D)$, $F = \{A \rightarrow B, BC \rightarrow D, D \rightarrow A\}$

f) R is decomposed into $R_1(A, B, C)$ and $R_2(B, C, D)$. Is the decomposition lossless? Is the dependency preserving?

- Lossless?
 - $R_1 \cap R_2 = (B, C)$
 - $(B, C) \rightarrow R_2$
 - Hence, it is a lossless decomposition.

Question: How to test if a decomposition is lossless-join decomposition?

Answer: A decomposition is lossless-join decomposition iff **at least one of the following dependencies is in F^+ :**
1) common attribute of R_1 and $R_2 \rightarrow$ schema of R_1
2) common attribute of R_1 and $R_2 \rightarrow$ schema of R_2

Question 4

Given $R = (A, B, C, D)$, $F = \{A \rightarrow B, BC \rightarrow D, D \rightarrow A\}$

- f) R is decomposed into $R_1(A, B, C)$ and $R_2(B, C, D)$. Is the decomposition lossless? Is the dependency preserving?

Answer: A decomposition is dependency preserving iff:

$$(F_1 \cup F_2 \cup \dots \cup F_n)^+ = F^+$$

Where F_i is the set of FDs in F^+ that include only attributes in R_i .

- Lossless?
 - $R_1 \cap R_2 = (B, C)$
 - $(B, C) \rightarrow R_2$
 - Hence, it is a lossless decomposition.
- Dependency preserving?
 - $F_1 = \{A \rightarrow B\}$; $F_2 = \{BC \rightarrow D, D \rightarrow B\}$
 - Since $(F_1 \cup F_2)^+$ does not equal to F^+ ($D \rightarrow A$ is not preserved), the decomposition is not dependency preserving

Question: How to test if a decomposition is dependency preserving or not?

Question 4

Given $R=(A, B, C, D)$, $F=\{A \rightarrow B, BC \rightarrow D, D \rightarrow A\}$

g) Is R in BCNF? If not, decompose R into relations in BCNF.

- Is R in BCNF?
 - No. $A \rightarrow B$ and $D \rightarrow A$ violate BCNF as neither A nor D is a super key in R .

Question: How to test if the decomposed relations are in BCNF?

Answer: In BCNF, every non-trivial FD forms a key!

Question 4

Given $R=(A, B, C, D)$, $F=\{A \rightarrow B, BC \rightarrow D, D \rightarrow A\}$

g) Is R in BCNF? If not, decompose R into relations in BCNF.

- If not, decompose R into relations in BCNF.

- Decompose R into $R_1(A, B)$ and $R_2(A, C, D)$
- R_1 is in BCNF?
 - $F_1 = \{A \rightarrow B\}$, A is a key in R_1 , **R_1 is in BCNF!**
- R_2 is in BCNF?
 - $F_2 = \{D \rightarrow A\}$, D is not a key in R_2 , **R_2 is not in BCNF!**

Step 1. Decompose R into two relations R_1 and R_2 , and make R_1 a BCNF.

Look at $A \rightarrow B$ in F , make R_1 as (A, B) .

Then R_2 is (A, C, D) .

Step 2. Check if R_1 and R_2 are in BCNF. If not, further decompose it using Step 1.

In BCNF, every non-trivial FD forms a key!

Question 4

Given $R = (A, B, C, D)$, $F = \{A \rightarrow B, BC \rightarrow D, D \rightarrow A\}$

g) Is R in BCNF? If not, decompose R into relations in BCNF.

- If not, decompose R into relations in BCNF.
 - Decompose R into $R_1(A, B)$ and $R_2(A, C, D)$
 - R_1 is in BCNF?
 - $F_1 = \{A \rightarrow B\}$, A is a key in R_1 , **R_1 is in BCNF!**
 - R_2 is in BCNF?
 - $F_2 = \{D \rightarrow A\}$, D is not a key in R_2 , **R_2 is not in BCNF!**
 - We further decompose R_2 into $R_3(A, D)$ and $R_4(C, D)$.
 - R_3 is in BCNF?
 - $F_3 = \{D \rightarrow A\}$, D is a key in R_3 , **R_3 is in BCNF!**
 - R_4 is in BCNF?
 - $F_4 = \{\text{empty}\}$, **R_4 is in BCNF!**
 - Hence, we can decompose R into **R_1 , R_3 and R_4** in BCNF.