

# Normalisation

# LECTURE

- ⇒ Purpose of Normalisation
- ⇒ Redundancy and Data Anomalies
- ⇒ Repeating Groups
- ⇒ Functional Dependency
- ⇒ Transitive Dependency
- ⇒ Stages of Normalisation

# *Purpose of Normalisation*

- ❖ To avoid redundancy by storing each 'fact' within the database only once.
- ❖ To put data into a form that conforms to relational principles (e.g., single valued attributes, each relation represents one entity) - no repeating groups.
- ❖ To put the data into a form that is more able to accurately accommodate change.
- ❖ To avoid certain updating 'anomalies'.
- ❖ To facilitate the enforcement of data constraints.

# Redundancy and Data Anomalies

Redundant data is where we have stored the same ‘information’ more than once. i.e., the redundant data could be removed without the loss of information.

**Example:** We have the following relation that contains staff and department details:

staffNo	job	dept	dname	city
SL10	Salesman	10	Sales	Stratford
SA51	Manager	20	Accounts	Barking
DS40	Clerk	20	Accounts	Barking
OS45	Clerk	30	Operations	Barking



*Such ‘redundancy’  
could lead to the  
following ‘anomalies’*

**Insert Anomaly:** We can’t insert a dept without inserting a member of staff that works in that department

**Update Anomaly:** We could change the name of the dept that SA51 works in without simultaneously changing the dept that DS40 works in.

**Deletion Anomaly:** By removing employee SL10 we have removed all information pertaining to the Sales dept.

# Repeating Groups

A repeating group is an attribute (or set of attributes) that can have **more than one** value for a primary key value.

**Example:** We have the following relation that contains staff and department details and a list of telephone contact numbers for each member of staff.

<b>staffNo</b>	<b>job</b>	<b>dept</b>	<b>dname</b>	<b>city</b>	<b>contact number</b>
SL10	Salesman	10	Sales	Stratford	018111777, 018111888, 079311122
SA51	Manager	20	Accounts	Barking	017111777
DS40	Clerk	20	Accounts	Barking	
OS45	Clerk	30	Operations	Barking	079311555

Repeating Groups are not allowed in a relational design, since all attributes have to be 'atomic' - i.e., there can only be one value per cell in a table!

# Functional Dependency

**Formal Definition:** Attribute **B** is functionally dependant upon attribute **A** (or a collection of attributes) if a value of **A** determines a single value of attribute **B** at any one time.

**Formal Notation:**  $A \rightarrow B$  This should be read as ‘**A determines B**’ or ‘**B is functionally dependant on A**’. **A** is called the *determinant* and **B** is called the *object of the determinant*.

Example:

staffNo	job	dept	dname
SL10	Salesman	10	Sales
SA51	Manager	20	Accounts
DS40	Clerk	20	Accounts
OS45	Clerk	30	Operations

## Functional Dependencies

staffNo  $\rightarrow$  job

staffNo  $\rightarrow$  dept

staffNo  $\rightarrow$  dname

dept  $\rightarrow$  dname

# Functional Dependency

**Compound Determinants:** If more than one attribute is necessary to determine another attribute in an entity, then such a determinant is termed a **composite determinant**.

**Full Functional Dependency:** Only of relevance with composite determinants. This is the situation when it is necessary to use all the attributes of the composite determinant to identify its object uniquely.

Example:

order#	line#	qty	price
A001	001	10	200
A002	001	20	400
A002	002	20	800
A004	001	15	300

## Full Functional Dependencies

(Order#, line#)  $\rightarrow$  qty

(Order#, line#)  $\rightarrow$  price

# Functional Dependency

**Partial Functional Dependency:** This is the situation that exists if it is necessary to only use a subset of the attributes of the composite determinant to identify its object uniquely.

Example:

student#	unit#	room	grade
9900100	A01	TH224	2
9900010	A01	TH224	14
9901011	A02	JS075	3
9900001	A01	TH224	16

Full Functional Dependencies

(student#, unit#)  $\rightarrow$  grade

Partial Functional Dependencies

unit#  $\rightarrow$  room

Repetition of data!





# Transitive Dependency

**Definition:** A transitive dependency exists when there is an intermediate functional dependency.

**Formal Notation:** If  $A \rightarrow B$  and  $B \rightarrow C$ , then it can be stated that the following transitive dependency exists:  $A \rightarrow B \rightarrow C$

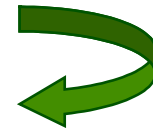
Example:

staffNo	job	dept	dname
SL10	Salesman	10	Sales
SA51	Manager	20	Accounts
DS40	Clerk	20	Accounts
OS45	Clerk	30	Operations

## Transitive Dependencies

staffNo  $\rightarrow$  dept

dept  $\rightarrow$  dname



staffNo  $\rightarrow$  dept  $\rightarrow$  dname

**Repetition of data!**

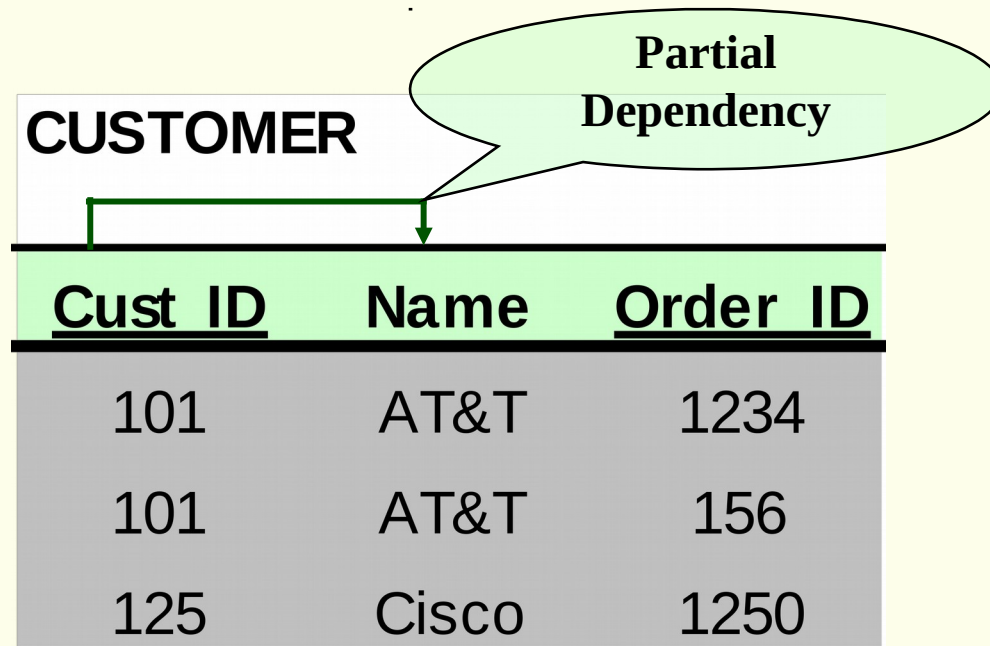
# Dependencies: Definitions

- ❖ **Multivalued Attributes** (or **repeating groups**): non-key attributes or groups of non-key attributes the values of which are not uniquely identified by (directly or indirectly) (not functionally dependent on) the value of the Primary Key (or its part).

STUDENT			
<u>Stud ID</u>	Name	Course_ID	Units
101	Lennon	MSI 250	3.00
101	Lennon	MSI 415	3.00
125	Johnson	MSI 331	3.00

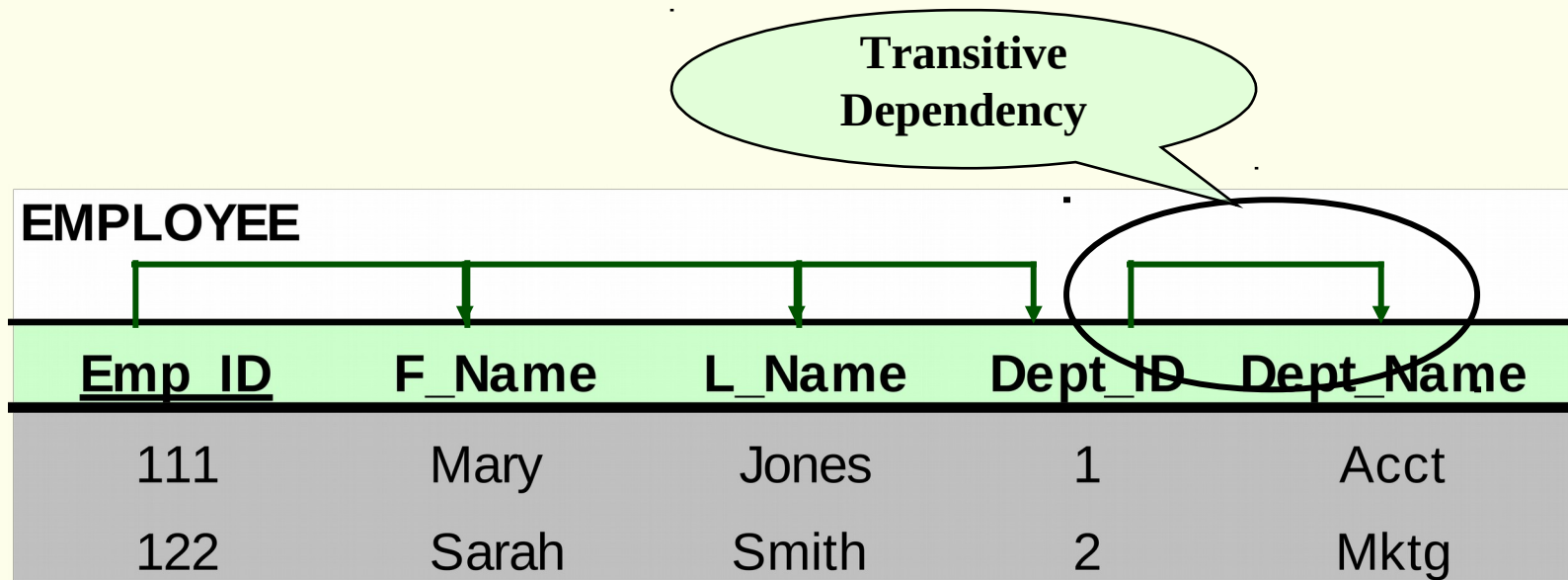
# Dependencies: Definitions

- ❖ **Partial Dependency** – when a non-key attribute is determined by a part, but not the whole, of a **COMPOSITE** primary key.



# Dependencies: Definitions

- ❖ **Transitive Dependency** – when a non-key attribute determines another non-key attribute.



# *Normalisation - Relational Model*

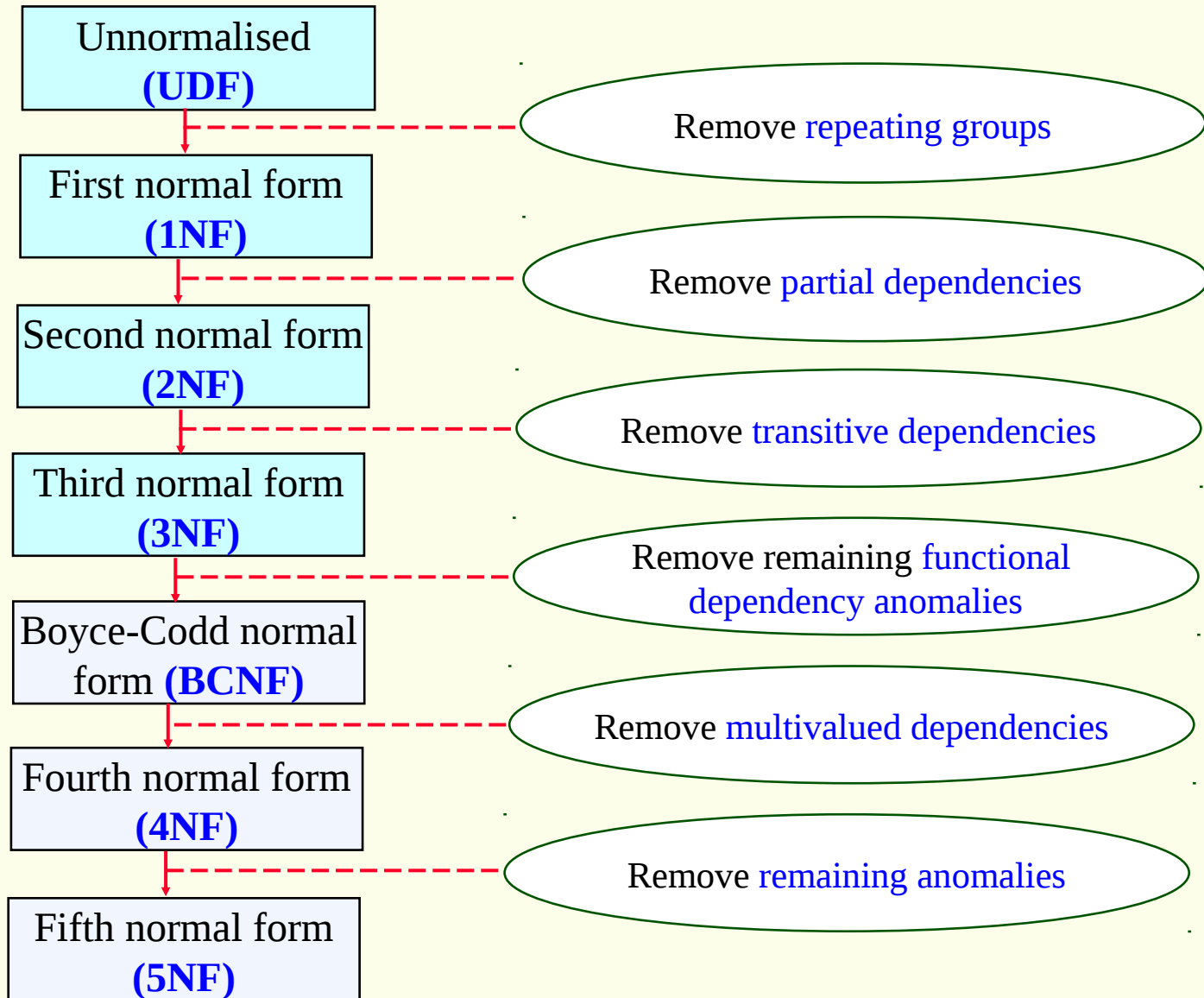
In order to comply with the relational model it is necessary to 1) remove repeating groups and 2) avoid redundancy and data anomalies by removing partial and transitive functional dependencies.

**Relational Database Design:** All attributes in a table must be atomic, and solely dependant upon the fully primary key of that table.

**NORMALISATION  
ACHIEVES THIS!**

**THE KEY, THE WHOLE KEY, AND NOTHING BUT THE KEY!**

# *Stages of Normalisation*



# *Unnormalised Normal Form (UNF)*

**Definition:** A relation is unnormalised when it has not had any normalisation rules applied to it, and it suffers from various anomalies.

This only tends to occur where the relation has been designed using a **‘bottom-up approach’**. *i.e., the capturing of attributes to a ‘**Universal Relation**’ from a screen layout, manual report, manual document, etc...*

# Unnormalised Normal Form (UNF)

## ORDER

Customer No: 001964  
Name: Mark Campbell  
Address: 1 The House  
Leytonstone  
E11 9ZZ

Order Number: 00012345  
Order Date: 14-Feb-2002

Product Number	Product Description	Unit Price	Order Quantity	Line Total
T5060	Hook	5.00	5	25.00
PT42	Bolt	2.50	10	20.50
QZE48	Spanner	20.00	1	20.00

Order Total: 65.50

**ORDER** (order-no, order-date, cust-no, cust-name, cust-add,  
(*prod-no, prod-desc, unit-price, ord-qty, line-total*)\*, order-total



# *First Normal Form (1NF)*

**Definition:** A relation is in 1NF if, and only if, all its underlying attributes contain atomic values only.

**Remove repeating groups into a new relation**

A repeating group is shown by a pair of brackets within the relational schema.

**ORDER** (order-no, order-date, cust-no, cust-name, cust-add,  
*(prod-no, prod-desc, unit-price, ord-qty, line-total)\**, order-total

## **Steps from UNF to 1NF:**

- ❖ Remove the outermost repeating group (and any nested repeated groups it may contain) and create a new relation to contain it.
- ❖ Add to this relation a copy of the PK of the relation immediately enclosing it.
- ❖ Name the new entity (*appending the number 1 to indicate 1NF*)
- ❖ Determine the PK of the new entity
- ❖ Repeat steps until no more repeating groups.

# Example - UNF to 1NF


**ORDER** (order-no, order-date, cust-no, cust-name, cust-add,  
(*prod-no, prod-desc, unit-price, ord-qty, line-total*)\*, order-total

1. Remove the outermost repeating group (and any nested repeated groups it may contain) and create a new relation to contain it. (*rename original to indicate 1NF*)

**ORDER-1** (order-no, order-date, cust-no, cust-name, cust-add, order-total  
(prod-no, prod-desc, unit-price, ord-qty, line-total)

2. Add to this relation a copy of the PK of the relation immediately enclosing it.

**ORDER-1** (order-no, order-date, cust-no, cust-name, cust-add, order-total  
(order-no, prod-no, prod-desc, unit-price, ord-qty, line-total)



3. Name the new entity (*appending the number 1 to indicate 1NF*)

**ORDER-LINE-1** (order-no, prod-no, prod-desc, unit-price, ord-qty, line-total)

4. Determine the PK of the new entity

**ORDER-LINE-1** (order-no, prod-no, prod-desc, unit-price, ord-qty, line-total)

# Second Normal Form (2NF)

**Definition:** A relation is in 2NF if, and only if, it is in 1NF and every non-key attribute is fully dependent on the primary key.

**Remove partial functional dependencies into a new relation**

## Steps from 1NF to 2NF:

- ❖ Remove the offending attributes that are only partially functionally dependent on the composite key, and place them in a new relation.
- ❖ Add to this relation a copy of the attribute(s) which are the determinants of these offending attributes. These will automatically become the primary key of this new relation.
- ❖ Name the new entity (*appending the number 2 to indicate 2NF*)
- ❖ Rename the original entity (*ending with a 2 to indicate 2NF*)

# Example - 1NF to 2NF

ORDER-LINE-1 (order-no, prod-no, prod-desc, unit-price, ord-qty, line-total)

1. Remove the offending attributes that are only partially functionally dependent on the composite key, and place them in a new relation.

ORDER-LINE-1 (order-no, prod-no, ord-qty, line-total)

(prod-desc, unit-price)

2. Add to this relation a copy of the attribute(s) which determines these offending attributes. These will automatically become the primary key of this new relation..

ORDER-LINE-1 (order-no, prod-no, ord-qty, line-total)

(prod-no, prod-desc, unit-price)



3. Name the new entity (*appending the number 2 to indicate 2NF*)

PRODUCT-2 (prod-no, prod-desc, unit-price)

4. Rename the original entity (*ending with a 2 to indicate 2NF*)

ORDER-LINE-2 (order-no, prod-no, ord-qty, line-total)

# Third Normal Form (3NF)

**Definition:** A relation is in 3NF if, and only if, it is in 2NF and every non-key attribute is non-transitively dependent on the primary key.

**Remove transitive dependencies into a new relation**

## Steps from 2NF to 3NF:

- ❖ Remove the offending attributes that are transitively dependent on non-key attribute(s), and place them in a new relation.
- ❖ Add to this relation a copy of the attribute(s) which are the determinants of these offending attributes. These will automatically become the primary key of this new relation.
- ❖ Name the new entity (*appending the number 3 to indicate 3NF*)
- ❖ Rename the original entity (*ending with a 3 to indicate 3NF*)

# Example - 2NF to 3NF

ORDER-2 (order-no, order-date, cust-no, cust-name, cust-add, order-total)

1. Remove the offending attributes that are transitively dependent on non-key attributes, and place them in a new relation.

ORDER-2 (order-no, order-date, cust-no, order-total)

(cust-name, cust-add )

2. Add to this relation a copy of the attribute(s) which determines these offending attributes. These will automatically become the primary key of this new relation..

ORDER-2 (order-no, order-date, cust-no, order-total)

(cust-no, cust-name, cust-add )



3. Name the new entity (*appending the number 3 to indicate 3NF*)

CUSTOMER-3 (cust-no, cust-name, cust-add )

4. Rename the original entity (*ending with a 3 to indicate 3NF*)

ORDER-3 (order-no, order-date, cust-no, order-total)

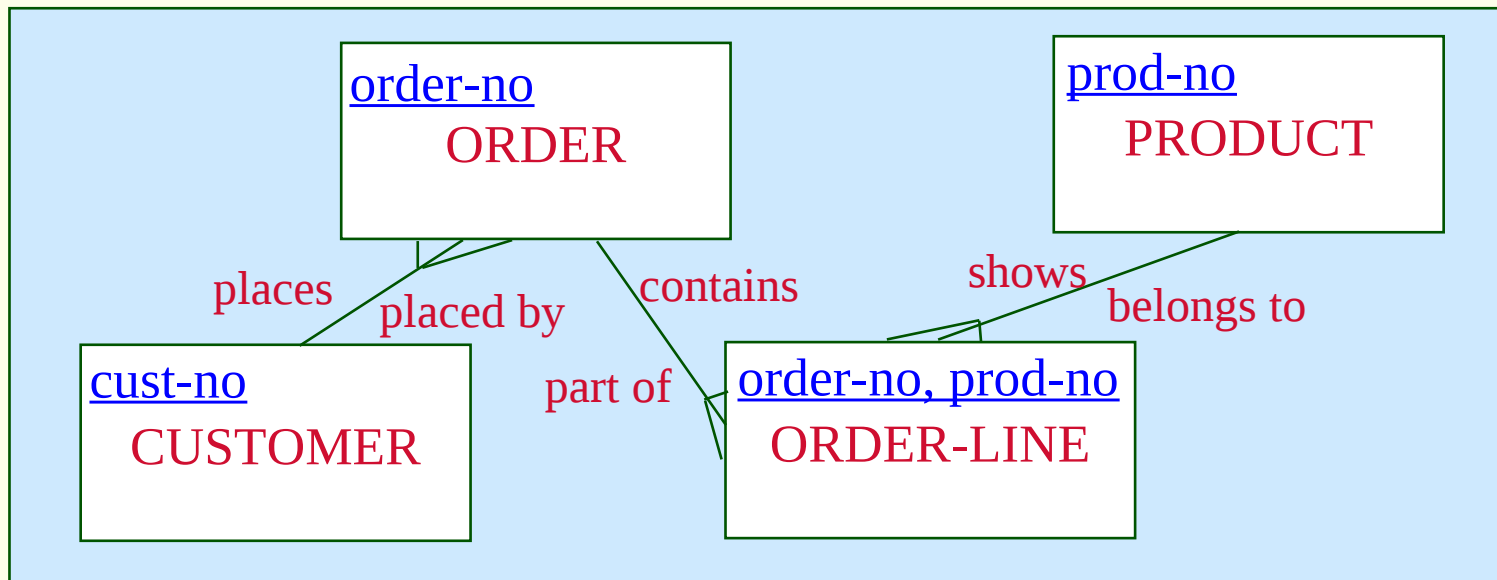
# Example - Relations in 3NF

ORDER-3 (order-no, order-date, **cust-no**, order-total)

CUSTOMER-3 (cust-no, cust-name, cust-add )

PRODUCT-2 (prod-no, prod-desc, unit-price)

ORDER-LINE-2 (order-no, prod-no, ord-qty, line-total)



# Boyce-Codd Normal Form (BCNF)

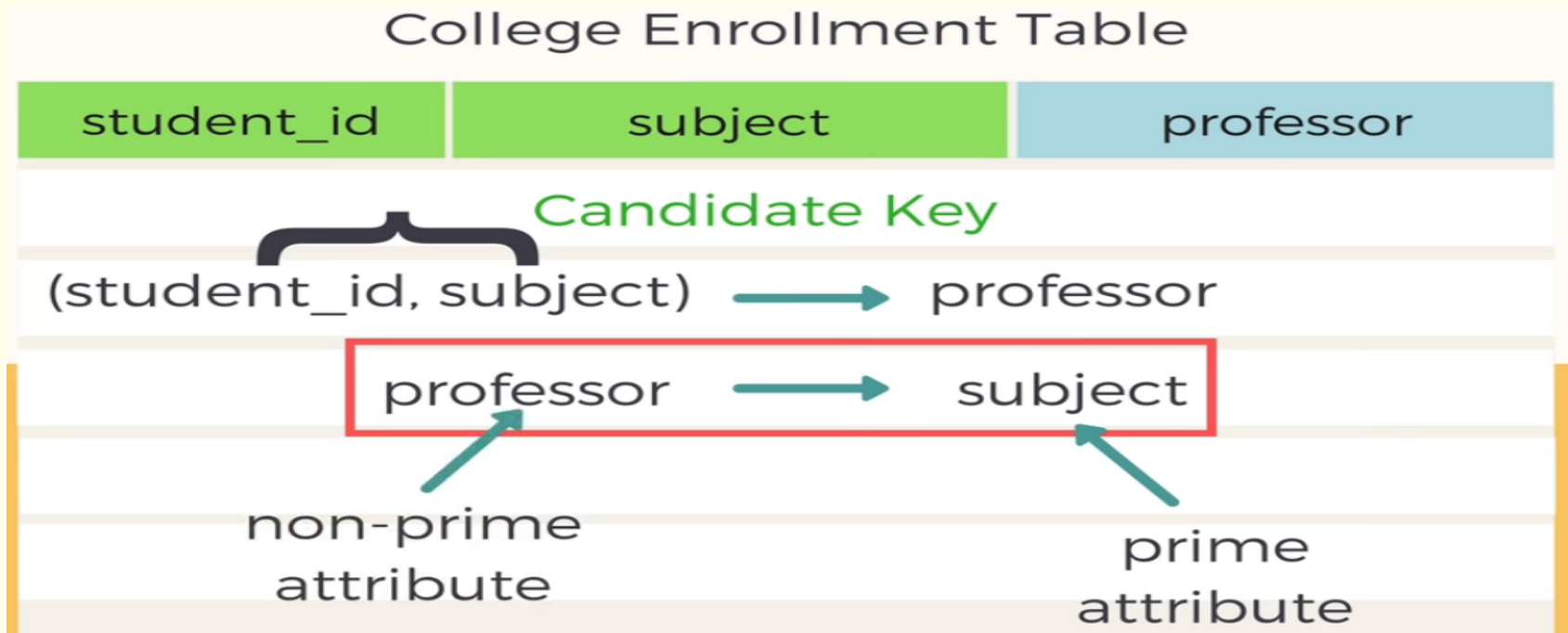
**Definition:** A relation is in BCNF if, and only if, it is in 3NF and every determinant is a candidate key.

**Remove remaining functional dependency anomalies**

student_id	subject	professor
101	Java	P. Java
101	C++	P. Cpp
102	Java	P. Java2
103	C#	P. Chash
104	Java	P. Java



# Boyce-Codd Normal Form (BCNF)



## Why this table is not in BCNF?

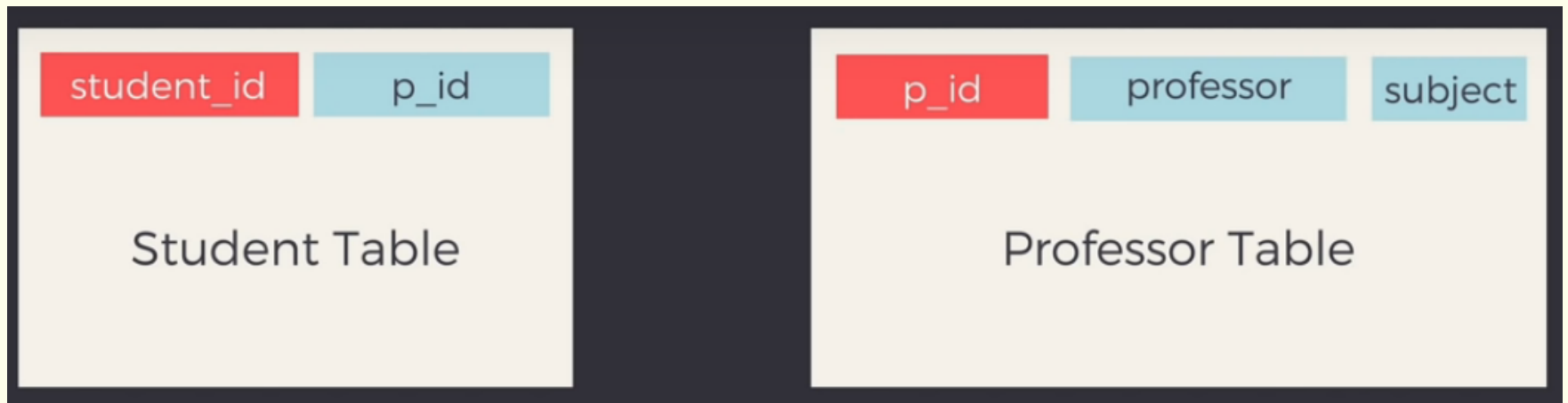
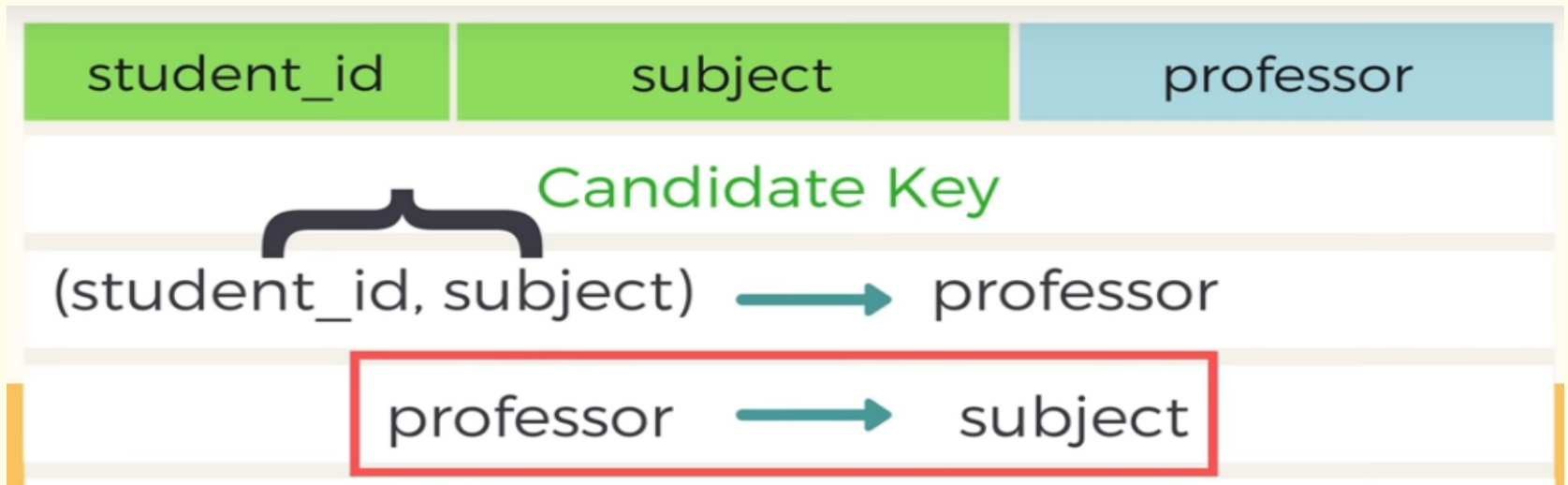
- ❖ In the above table, `student_id`, `subject` form primary key, which means `subject` column is a prime attribute.
- ❖ But, there is one more dependency, `professor → subject`.
- ❖ And while `subject` is a prime attribute, `professor` is a non-prime attribute, which is not allowed by BCNF.

# How to make the table satisfy BCNF?

College Enrollment Table

student_id	subject	professor
101	Java	P. Java
101	C++	P. Cpp
102	Java	P. Java2
103	C#	P. Chash
104	Java	P. Java

## Example - 3NF to BCNF



# Multi-valued Dependency

**Definition:** A Multi-valued dependency exists when there is an Multi-valued dependency between  $A \twoheadrightarrow B$ , and  $A \twoheadrightarrow C$ .

**A table is said to have multi-valued dependency, if the following conditions are true,**

- ❖ For a dependency  $A \twoheadrightarrow B$ , if for a single value of A, multiple value of B exists.
- ❖ Table should have at-least 3 columns for it to have a multi-valued dependency.
- ❖ for a relation  $R(A,B,C)$ , if there is a multi-valued dependency between, A and B, then B and C should be independent of each other.

A	B	C
A1	B1	C1
	B2	C2

# *Fourth Normal Form (4NF)*

**Definition:** A relation is in 4NF if, and only if, it is in Boyce-Codd Normal Form(BCNF). and the table should not have any Multi-valued Dependency.

**Remove multivalued dependencies into a new relation**

ENROLMENT TABLE		
s_id	course	hobby
1	Science	Cricket
1	Maths	Hockey
2	C#	Cricket
2	Php	Hockey

# Fourth Normal Form (4NF)

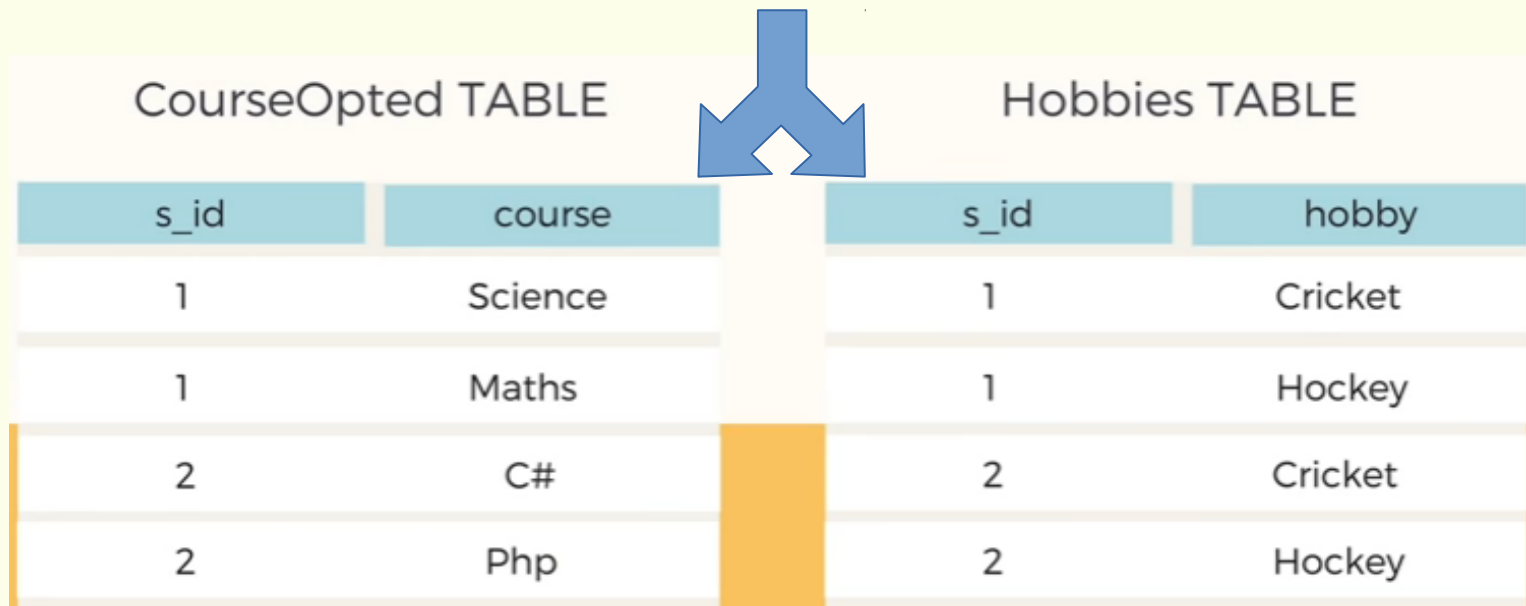
s_id	course	hobby
1	Science	Cricket
1	Maths	Hockey
1	Science	Hockey
1	Maths	Cricket

No relationship

- ❖ As you can see in the table above, student with s\_id 1 has opted for two courses, Science and Maths, and has two hobbies, Cricket and Hockey.
- ❖ And, there is no relationship between the columns course and hobby. They are independent of each other.
- ❖ So there is multi-value dependency, which leads to unnecessary repetition of data and other anomalies as well.

## Example - BCNF to 4NF

ENROLMENT TABLE		
s_id	course	hobby
1	Science	Cricket
1	Maths	Hockey
2	C#	Cricket
2	Php	Hockey



# *Fifth Normal Form (5NF)*

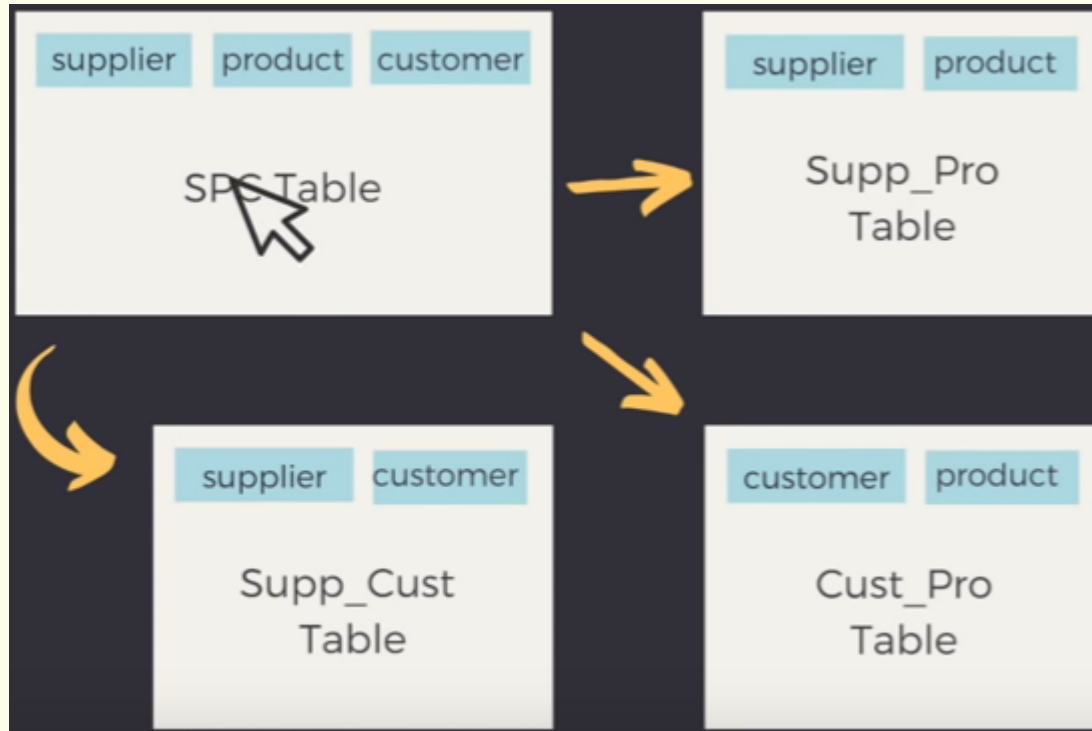
**Definition:** A relation is in 5NF if, and only if, it is in fourth Normal Form(4NF) and not contains any join dependency and joining should be lossless.

**Remove join dependencies into a new relation**

SPC Table		
supplier	product	customer
ACME	72X SW	FORD
ACME	GEAR L	GM
ROBUSTO	E SWITCH	FORD
ROBUSTO	OBD II	MERCEDES
ALWAT	72X SW	GM
ALWAT	OBD II	MERCEDES
ALWAT	GEAR L	MERCEDES



## *Example - 4NF to 5NF*



# *Decomposition of a Relation*

**Definition:** The process of breaking up or dividing a single relation into two or more sub relations is called as decomposition of a relation.

## **Properties of Decomposition :-**

The following two properties must be followed when decomposing a relation

### **1) Lossless decomposition**

- ❖ When the sub relations are joined back, the same original relation is obtained that was decomposed.
- ❖ No information is lost from the original relation during decomposition.

### **2) Dependency Preservation**

- ❖ None of the functional dependencies that holds on the original relation are lost.
- ❖ The sub relations still hold or satisfy the functional dependencies of the original relation.

# *Types of Decomposition*

## **Types of Decomposition**

```
graph TD; A[Types of Decomposition] --> B[Lossless Join Decomposition]; A --> C[Lossy Join Decomposition];
```

**Lossless Join  
Decomposition**

**Lossy Join  
Decomposition**

# Decomposition of a Relation

## Lossless decomposition :-

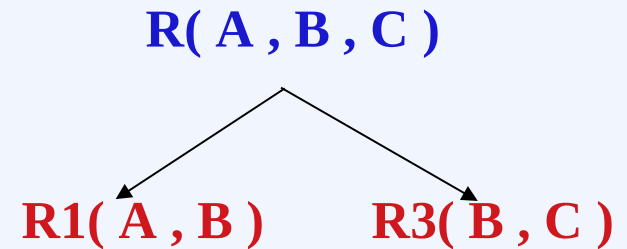
- ❖ Consider there is a relation R which is decomposed into sub relations R1 , R2 , ..... , Rn.
- ❖ This decomposition is called lossless join decomposition when the join of the sub relations results in the same relation R that was decomposed.
- ❖ For lossless join decomposition, we always have

$$R1 \bowtie R2 \bowtie R3 \dots\dots \bowtie Rn = R$$

where  $\bowtie$  is a natural join operator

Example, Consider the following relation R( A , B , C ), this relation is decomposed into two sub relations R1( A , B ) and R3( B , C ).

$$R1 \bowtie R3 = R$$



if we perform the natural join (  $\bowtie$  ) of the sub relations R1 and R3 , we get relation is same as the original relation R.

# Decomposition of a Relation

**Example:** Consider the following relation  $R(A, B, C)$

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

$R(A, B, C)$

R relation is decomposed into two sub relations as  $R_1(A, B)$  and  $R_3(A, C)$

A	B
1	2
2	2
3	3

$R_1(A, B)$

A	C
1	1
2	2
3	2

$R_3(A, C)$

For lossless decomposition, we must have  **$R_1 \bowtie R_3 = R$**

# Decomposition of a Relation

## Lossy Join Decomposition :-

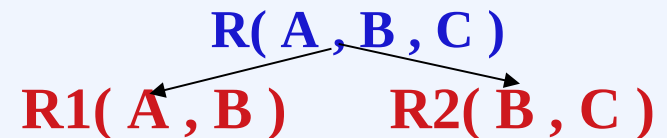
- ❖ Consider there is a relation R which is decomposed into sub relations R1 , R2 , ..... , Rn.
- ❖ This decomposition is called lossy join decomposition when the join of the sub relations does not result in the same relation R that was decomposed.
- ❖ The natural join of the sub relations is always found to have some extraneous tuples.
- ❖ For lossless join decomposition, we always have

$$R1 \bowtie R2 \bowtie R3 \dots\dots \bowtie Rn \supset R$$

where  $\bowtie$  is a natural join operator

Example, Consider the following relation R( A , B , C ), this relation is decomposed into two sub relations R1( A , B ) and R2( B , C ).

$$R1 \bowtie R2 \supset R$$



if we perform the natural join ( $\bowtie$ ) of the sub relations R1 and R2, we get relation is not same as the relation R & contains some extraneous tuples.

# Decomposition of a Relation

**Example:** Consider the following relation  $R(A, B, C)$

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

$R(A, B, C)$

R relation is decomposed into two sub relations as  $R_1(A, B)$  and  $R_2(B, C)$

A	B
1	2
2	2
3	3

$R_1(A, B)$

B	C
2	1
2	2
3	2

$R_2(B, C)$

For lossy join decomposition, we must have  $R_1 \bowtie R_2 \supset R$

# *Decomposition of a Relation*

## **Determining whether decomposition is Lossless Or Lossy :-**

Consider a relation R is decomposed into two sub relations R1 and R2. Then,

- ❖ If all the following conditions satisfy, then the **decomposition is lossless**.
- ❖ If any of these conditions fail, then the **decomposition is lossy**.

1) Union of both the sub relations must contain all the attributes that are present in the original relation R.

$$\mathbf{R1 \cup R2 = R}$$

2) Intersection of both the sub relations must not be null.

In other words, there must be some common attribute which is present in both the sub relations.

$$\mathbf{R1 \cap R2 \neq \emptyset}$$

3) Intersection of both the sub relations must be a super key of either R1 or R2 or both.

$$\mathbf{R1 \cap R2 = \text{Super key of R1 or R2 or both R1 \& R2}}$$



# DETERMINE DECOMPOSITION IS LOSSLESS OR LOSSY

**Problem-01:** Consider a relation schema  $R ( A , B , C , D )$  with the functional dependencies  $A \rightarrow B$  and  $C \rightarrow D$ . Determine whether the decomposition of  $R$  into  $R1 ( A , B )$  and  $R2 ( C , D )$  is lossless or lossy.

**Solution :-** We will check all the conditions one by one. If any of the conditions fail, then the decomposition is lossy otherwise lossless.

**Step 1:**  $R1 ( A , B ) \cup R2 ( C , D ) = R ( A , B , C , D )$

1) union of the sub relations contain all the attributes of relation  $R$ .

Thus, condition-01 satisfies.

**Step 2:**  $R1 ( A , B ) \cap R2 ( C , D ) = \Phi$

2) intersection of the sub relations is null.

So, condition-02 fails.

*Thus, we conclude that the decomposition is lossy.*

# Decomposition of a Relation

**Problem-02:** Consider A relation  $R(A,B,C,D)$  with FD set  $\{A \rightarrow BC\}$  is decomposed into  $R_1(A,B,C)$  and  $R_2(A,D)$ . Determine whether the decomposition of  $R$  into  $R_1(A,B,C)$  and  $R_2(A,D)$  is lossless or lossy.

**Solution :-** We will check all the conditions one by one. If any of the conditions fail, then the decomposition is lossy otherwise lossless.

**Step 1:**  $R_1(A,B,C) \cup R_2(A,D) = R(A,B,C,D)$

1) union of the sub relations contain all the attributes of relation  $R$ .

Thus, condition-01 satisfies.

**Step 2:**  $R_1(A,B,C) \cap R_2(A,D) = \Phi$   
 $A = \Phi$

2) intersection of the sub relations is not null. So, condition-02 satisfies.

**Step 3:**  $R_1(A,B,C) \cap R_2(A,D) = A$  is a key of  $R_1(ABC)$  because  $A \rightarrow BC$  is given

1) Attribute 'A' can determine all the attributes of sub relation  $R_1$ . it is a super key of the sub relation  $R_1$ . Thus, condition-03 satisfies.

*Thus, we conclude that the decomposition is lossless.*

# Dependency Preserving Decomposition

**Definition:** If we decompose a relation  $R$  into relations  $R_1$  and  $R_2$ , all dependencies of  $R$  must be part of either  $R_1$  or  $R_2$  or must be derivable from combination of functional dependencies(FD) of  $R_1$  and  $R_2$

## Example 1

Suppose a relation  $R(A,B,C,D)$  with **FD set**  $\{A \rightarrow BC\}$  is decomposed into  $R_1(A,B,C)$  and  $R_2(A,D)$ .

which is dependency preserving because *FD  $A \rightarrow B,C$  is a part of  $R_1(A,B,C)$*

## Example 2

❖ Consider a schema  $R(A,B,C,D)$  and functional dependencies  $A \rightarrow B$  and  $C \rightarrow D$  which is decomposed into  $R_1(A,B)$  and  $R_2(C,D)$

This decomposition is *dependency preserving decomposition* because

$A \rightarrow B$  can be ensured in  $R_1(A,B)$

$C \rightarrow D$  can be ensured in  $R_2(C,D)$