

# **Database Management System**

## **Relational Database Design**

**Dr. Vishwanath**



# Outline

- Functional Dependency (FD) and its Types
- Armstrong's axioms OR Inference rules
- Closure of a set of FDs
- Closure of attribute sets
- Decomposition
- Anomaly and its types
- Normalization and normal forms

# Functional Dependency(FD)

It specifies the relationship between two sets of attributes where one attribute determines the value of another attribute.

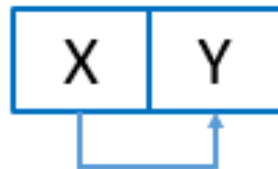
It is denoted by  $X \rightarrow Y$ , (Here attribute set X and Y are subsets of R)

Where **X** is called **Determinant**, and **Y** is called the **Dependent**.

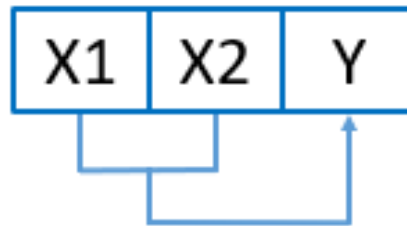
If  $X \rightarrow Y$ , we say X functionally determines Y or Y is functionally dependent on X.

# Diagrammatic representation of FD

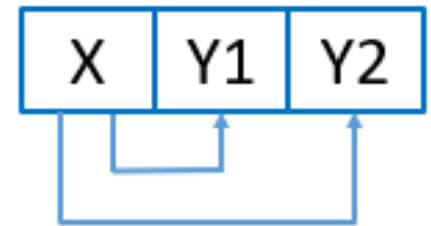
$X \rightarrow Y$



$\{X1, X2\} \rightarrow Y$

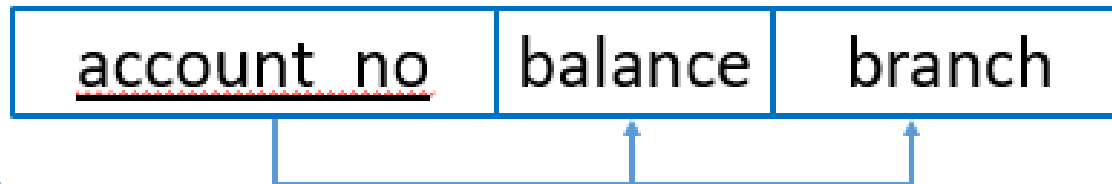


$X \rightarrow \{Y1, Y2\}$



## Example

- ▶ Consider the relation Account(account\_no, balance, branch).
- ▶ account\_no can determine balance and branch.
- ▶ So, there is a functional dependency from account\_no to balance and branch.
- ▶ This can be denoted by  $\text{account\_no} \rightarrow \{\text{balance}, \text{branch}\}$ .



# Functional Dependency (FD)

Roll_no	Name	Dept_name	Dept_building
42	Abc	CO	A4
43	Pqr	IT	A3
44	Xyz	CO	A4
45	Xyz	IT	A3
46	Mno	EC	B2
47	Jkl	ME	B2

# Functional Dependency (FD)

## Valid FD:

- Roll\_no  $\rightarrow$  {name, dept\_name, dept\_building}
- Roll\_no  $\rightarrow$  dept\_name
- Dept\_name  $\rightarrow$  dept\_building
- Roll\_no  $\rightarrow$  name
- {Roll\_no, name}  $\rightarrow$  {dept\_name, dept\_building}

## Invalid FD:

- |                   |               |           |
|-------------------|---------------|-----------|
| Name              | $\rightarrow$ | dept_name |
| dept_building     | $\rightarrow$ | dept_name |
| Name              | $\rightarrow$ | roll_no   |
| {name, dept_name} | $\rightarrow$ | roll_no   |
| dept_building     | $\rightarrow$ | roll_no   |



# Types of Functional Dependency (FD)

1. Trivial functional dependency
2. Non-Trivial functional dependency

# Trivial Functional Dependency

In **Trivial Functional Dependency**, a dependent is always a subset of the determinant.

Eg.  $\{\text{Roll\_No, Department\_Name, Semester}\} \rightarrow \text{Roll\_No}$

Here,  $\{\text{roll\_no, name}\} \rightarrow \text{name}$  is a trivial FD, since the dependent **name** is a subset of determinant set **{roll\_no, name}**.

roll_no	name	age
42	abc	17
43	pqr	18
44	xyz	18



# Non-Trivial Functional Dependency

In **Non-trivial FD**, the dependent is strictly not a subset of the determinant.

i.e. If  $X \rightarrow Y$  and **Y is not a subset of X**, then it is called Non-trivial FD

Here, **roll\_no**  $\rightarrow$  **name** is a non-trivial, the dependent **name** is **not a subset of** determinant **roll\_no**.

Similarly, **{roll\_no, name}  $\rightarrow$  age** is non-trivial FD

# Advantages of Functional Dependency

- 1. Data Normalization:** process of organizing data in a database in order to minimize redundancy and increase data integrity.
  - FD play an important part in data normalization.
- 2. Query Optimization:** FD helps to decide the connectivity between the tables and the necessary attributes.

This helps in query optimization and improves performance.

# Advantages of Functional Dependency

**3. Consistency of Data:** FD ensures the consistency of the data by removing any redundancies and also ensures that the changes made in one attribute does not affect the another set of attributes.

**4. Data Quality Improvement:** FD ensure that the data in the database to be accurate, complete and updated

FD eliminates errors and inaccuracies that might occur during data analysis and decision making

# Closure Set : Armstrong's Axioms (Inference Rules)

## 1. Reflexivity:

If **B** is a subset of **A**, then **A**→**B** holds by reflexivity rule

If **B**⊆**A** then **A**→**B**

Example:

1. {roll\_no, name} → name

2. X= {a,b,c,d,e}

Y= {a,b,c} Then **X**→**Y**

# Closure Set : Armstrong's Axioms (Inference Rules)

## 2. Augmentation:

If  $\mathbf{A} \rightarrow \mathbf{B}$  is a valid dependency, then  $\mathbf{AC} \rightarrow \mathbf{BC}$  is also valid by the augmentation rule.

Example:

$\{\text{roll\_no, name}\} \rightarrow \text{dept\_building}$

$\{\text{roll\_no, name, dept\_name}\} \rightarrow \{\text{dept\_building, dept\_name}\}$

# Closure Set : Armstrong's Axioms (Inference Rules)

## 3. Transitivity:

If  $A \rightarrow B$  and  $B \rightarrow C$  are both valid dependencies, then  $A \rightarrow C$  is valid by the Transitivity rule.

Example:  $\text{roll\_no} \rightarrow \text{dept\_name}$  &  
 $\text{dept\_name} \rightarrow \text{dept\_building}$ , then  
 $\text{roll\_no} \rightarrow \text{dept\_building}$  is also valid.

# Closure Set : Armstrong's Axioms (Inference Rules)

STU_ID	CLASS	LECTURE_HALL
1	7	L202
2	6	B101

Here, **STU\_ID**  $\rightarrow$  **CLASS**  
and **CLASS**  $\rightarrow$  **LECTURE\_HALL** number for that particular class

It means with the help of STU\_ID, we can determine LECTURE HALL

Therefore **STU\_ID** $\rightarrow$ **LECTURE\_HALL** holds true.



## Closure Set : Armstrong's Axioms (Inference Rules)

**4. Union:** If  $A \rightarrow B$  and  $A \rightarrow C$ , then  $A \rightarrow BC$ .

For example,  $STU\_ID \rightarrow STU\_NAME$ ,  
 $STU\_ID \rightarrow COURSE$

then  $STU\_ID \rightarrow \{STU\_NAME, COURSE\}$  holds true.

# Closure Set : Armstrong's Axioms (Inference Rules)

**5. Decomposition:** If  $A \rightarrow BC$ ,  
then  $A \rightarrow B$ , and  $A \rightarrow C$

For example:  $STU\_ID \rightarrow \{STU\_NAME, COURSE\}$

then  $STU\_ID \rightarrow STU\_NAME$ ,  
 $STU\_ID \rightarrow COURSE$  holds true.

# Closure Set : Armstrong's Axioms (Inference Rules)

## 7. Pseudo Transitivity

If  $A \rightarrow B$  and  $BD \rightarrow C$   
then  $AD \rightarrow C$

## 8. Composition

If  $A \rightarrow B$  and  $C \rightarrow D$   
then  $AC \rightarrow BD$

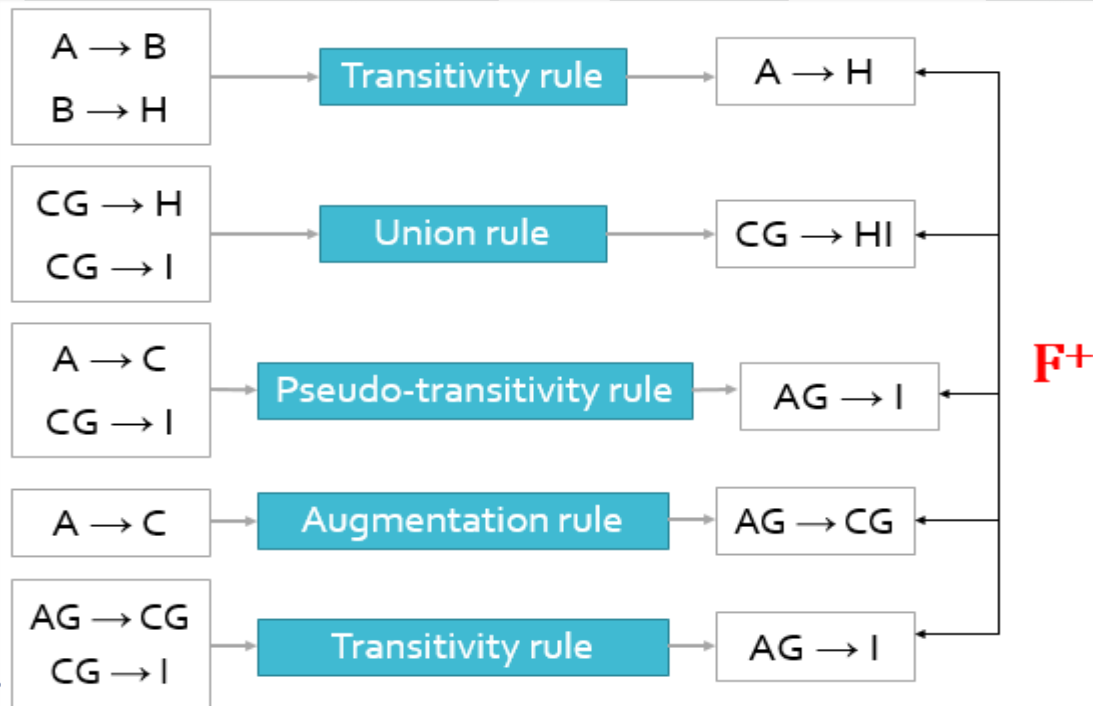
# Closure Set of Functional Dependency

- It is a set of FDs that can be created from an existing set of FD.
- For e.g.  $F = \{A \rightarrow B, B \rightarrow C\}$ , then new FD that can be created is  $A \rightarrow C$ .
- It is denoted by  $F^+$

# Closure Set of Functional Dependency

Suppose we are given a relation schema  $R(A, B, C, G, H, I)$  and the set of functional dependencies are:

**$F = (A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H)$ . Find  $F^+$**



# Closure Set of Functional Dependency

Compute the closure of the following set **F** of functional dependencies for relational schema  $R = (A, B, C, D, E, F)$

**$F = (A \rightarrow B, A \rightarrow C, CD \rightarrow E, CD \rightarrow F, B \rightarrow E)$ . Find  $F^+$**

# Closure Set of Functional Dependency

$A \rightarrow BC$	$A \rightarrow B \ \& \ A \rightarrow C$	Union Rule
$CD \rightarrow EF$	$CD \rightarrow E \ \& \ CD \rightarrow F$	Union Rule
$A \rightarrow E$	$A \rightarrow B \ \& \ B \rightarrow E$	Transitivity Rule
$AD \rightarrow E$	$A \rightarrow C \ \& \ CD \rightarrow E$	Pseudo-transitivity Rule
$AD \rightarrow F$	$A \rightarrow C \ \& \ CD \rightarrow F$	Pseudo-transitivity Rule

**$F^+$**

$$F^+ = \{A \rightarrow BC, CD \rightarrow EF, A \rightarrow E, AD \rightarrow E, AD \rightarrow F\}$$



# Closure of Attributes

It is a set of attributes that can be functionally determined from it.

Closure of attribute set  $\{X\}$  is denoted as  $\{X\}^+$ .

# Closure of Attributes

Q. Given a relation  $R(A,B,C,D)$  and FD  $\{ A \rightarrow B, B \rightarrow C, C \rightarrow D \}$ , then determine the  $A^+$ :

$A^+ = A$ , since A can determine A itself.

$A^+ = AB$ , A can also determine B.

$A^+ = ABC$ , A can also determine C with the help of B

$A^+ = ABCD$ , A can also determine D with the help of the C attribute.

Therefore,  $A^+(A \text{ closure}) = ABCD$ .

# Closure of Attributes

Q. Consider a relation R ( A , B , C , D , E , F , G ) with the functional dependencies  $A \rightarrow BC$   $BC \rightarrow DE$   $D \rightarrow F$   $CF \rightarrow G$

Find closure of attribute A and D

$$\begin{aligned} A^+ &= \{ A \} \\ &= \{ A , B , C \} \text{ ( Using } A \rightarrow BC \text{ )} \\ &= \{ A , B , C , D , E \} \text{ ( Using } BC \rightarrow DE \text{ )} \\ &= \{ A , B , C , D , E , F \} \text{ ( Using } D \rightarrow F \text{ )} \\ &= \{ A , B , C , D , E , F , G \} \text{ ( Using } CF \rightarrow G \text{ )} \\ \text{Thus, } A^+ &= \{ A , B , C , D , E , F , G \} \end{aligned}$$

$$\begin{aligned} D^+ &= \{ D \} \\ &= \{ D , F \} \text{ ( Using } D \rightarrow F \text{ )} \end{aligned}$$

# Closure of Attributes

Q. Given relational schema **R( P Q R S T U V )** having following attribute P Q R S T U and V, also there is a set of functional dependency denoted by **FD = { P → Q, QR → ST, PTV → V }**. Determine Closure of **(PR)<sup>+</sup>**.

$$PR^+ = PR$$

Now, look into a set of FD, and check that complete left side of any FD contains either P, R, or PR.

$$PR^+ = PRQ \quad \{P \rightarrow Q, P \text{ is a subset of } PR\}$$

$$PR^+ = PRQST \quad \{QR \rightarrow ST \text{ has its complete left part } QR \text{ in } PQR\}$$

$$\text{Therefore } PR^+ = PRQST$$

# Candidate Key

The minimal set of attributes that can uniquely identify a tuple is known as a candidate key.

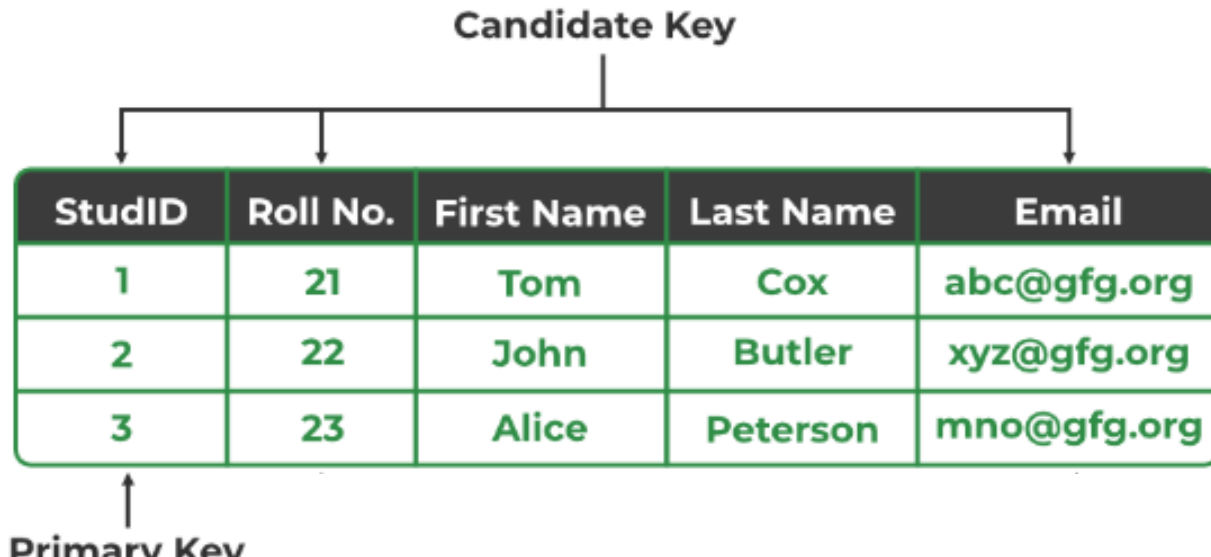
Example: Stud ID, Roll No, Email

✓ A minimal super key is called as a candidate key.

Candidate Key

StudID	Roll No.	First Name	Last Name	Email
1	21	Tom	Cox	abc@gfg.org
2	22	John	Butler	xyz@gfg.org
3	23	Alice	Peterson	mno@gfg.org

Primary Key

A diagram showing a tree structure where 'Candidate Key' is the root, branching down to 'StudID', 'Roll No.', and 'Email'. An arrow points from 'StudID' up to 'Primary Key'.

✓ Candidate Key is a Super Key whose no proper subset is a Super key

# Finding a Candidate Key

Example 1: Given  $R( X Y Z W )$  and  $FD = \{ XYZ \rightarrow W, XY \rightarrow ZW \text{ and } X \rightarrow YZW \}$

closure of  $XYZ_+ = XYZW$

closure of  $XY_+ = XYZW$

closure of  $X_+ = XYZW$

**X is the Candidate key:** As X cannot be further subdivided, or X cannot have any subset.

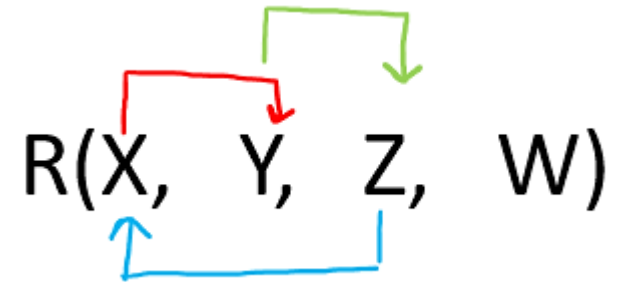
Q:  $R( X Y Z W )$  and  $FD = \{ Y \rightarrow XZW, XZW \rightarrow Y \}$ . Find no. of CK.

Ans: Y and XZW are candidate keys

# Shortcut to find a Candidate Key

Q. Give  $R(X, Y, Z, W)$  and Set of Functional Dependency  $FD = \{X \rightarrow Y, Y \rightarrow Z, Z \rightarrow X\}$ . Calculate the candidate key and no. of candidate key.

Here,  $W$  is not determined by any of the given  $FD$ , hence  $W$  will be the integral part of the Candidate key.



1. closure of  $W$  contains only  $W$ , hence it is not a candidate key.

$$W^+ = W$$

Combination of  $W$ , i.e.  $WX, WY, WZ$ .



# Shortcut to find a Candidate Key

a)  $W X^+ = W X Y Z$

Since the closure of  $WX$  contains all the attributes of  $R$ ,  
hence  $WX$  is Candidate Key

b)  $W Y^+ = W Y Z X$

$WY$  is Candidate Key

c)  $W Z^+ = W Z X Y$

$WZ$  is Candidate Key

Any further combination of  $WX$ ,  $WY$ ,  $WZ$ , i.e.  $WXY$ ,  $WXYZ$ ,  $WYZ$ ,  $WZX$  will be Super Key but not a candidate key.

**Therefore, there are 3 Candidate keys  $WX$ ,  $WY$ ,  $WZ$ .**

# Finding a Candidate Key

Q. Give  $R(X, Y, Z, W)$  and Set of Functional Dependency  $FD = \{XY \rightarrow ZW, W \rightarrow X\}$ . Calculate the candidate key and no. of candidate key.

PU

**Ans: there are TWO Candidate key Y X, Y W.**

# Finding a Candidate Key

Q. Give  $R(P, Q, R, S, T, U)$  and Set of Functional Dependency  $FD = \{ PQ \rightarrow R, R \rightarrow S, Q \rightarrow PT \}$ . Calculate the candidate key and no. of candidate key.

PU

**Ans: there is only ONE Candidate key QU.**

# Finding a Candidate Key

Q. Give  $R(A, B, C, D)$  and Set of Functional Dependency  $FD = \{ AB \rightarrow CD, C \rightarrow A, D \rightarrow B \}$ . Calculate the candidate key and no. of candidate key.

PU

**Ans: there are FOUR Candidate Keys: AB, AD, BC, and CD**

# Decomposition

Decomposition is the **process of breaking down the given relation into two or more relations.**

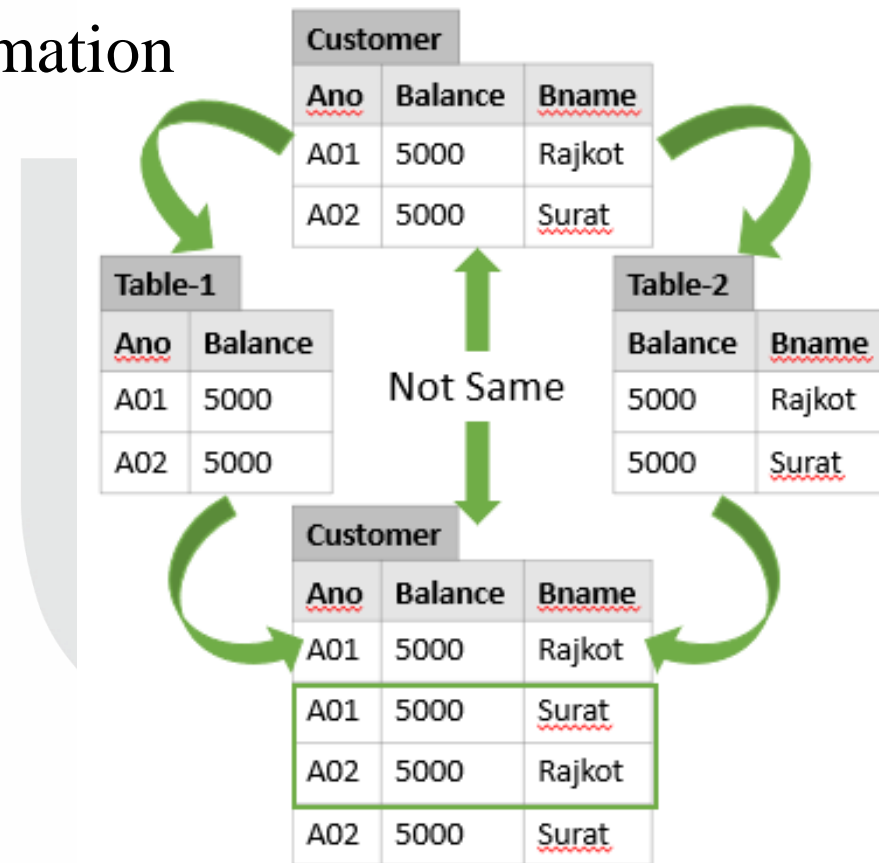
## ► Types of decomposition

- ➔ Lossy decomposition
- ➔ Lossless decomposition

# Lossy decomposition

In **lossy decomposition**, the information is lost during decomposition.

- ▶ The **disadvantage** of such kind of decomposition is that **some information is lost during retrieval of original relation.**
- ▶ From practical point of view, **decomposition should not be lossy decomposition.**

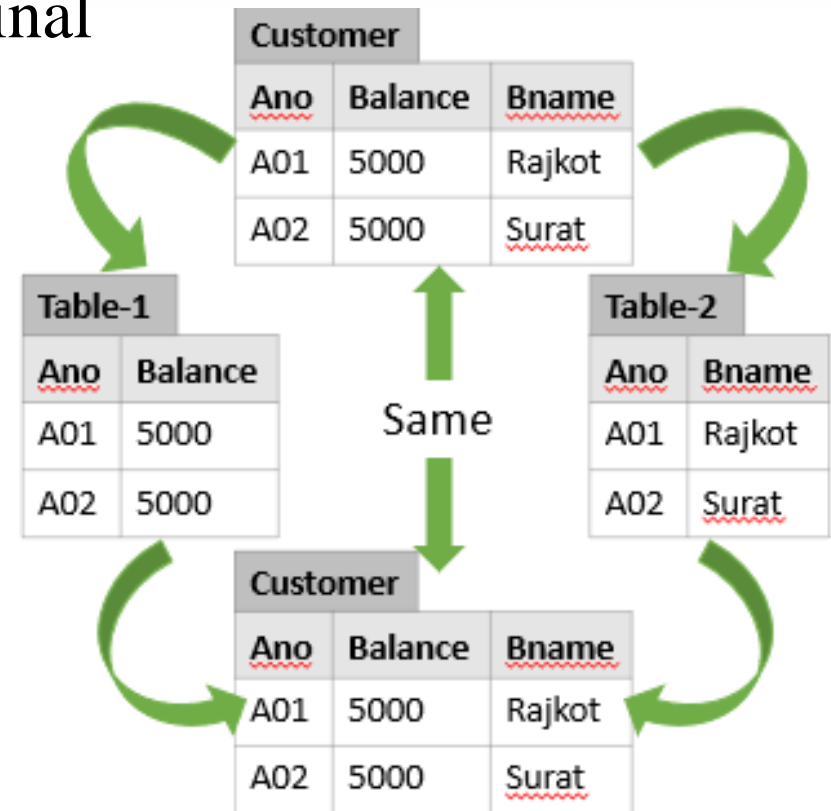


# Lossless decomposition

No information is lost from the original relation during decomposition.

When the sub relations are joined back, the same relation is obtained that was decomposed.

- ▶ This is also referred as a **non-additive (non-loss) decomposition**.
- ▶ All decompositions must be **lossless**.





# Database Anomalies

Anomalies in the relational model refer to inconsistencies or errors that can arise when working with relational databases.

Specifically in the context of data insertion, deletion, and modification.

These anomalies can be categorized into three types:

1. Insertion Anomalies
2. Deletion Anomalies
3. Update Anomalies.

# Insertion Anomalies

- Consider a relation Emp\_Dept(EID, Ename, City, DID, Dname, Manager) EID as a primary key

Emp_Dept					
<u>EID</u>	<u>Ename</u>	City	DID	<u>Dname</u>	Manager
1	Raj	Rajkot	1	CE	Shah
2	Meet	Surat	1	CE	Shah
NULL	NULL	NULL	2	IT	NULL

An insert anomaly occurs when **certain attributes cannot be inserted** into the database **without the presence of another attribute**.

Want to insert new department detail (IT)

- Suppose a **new department (IT) has been started** by the organization but **initially there is no employee appointed** for that department.
- We **want to insert that department detail** in Emp\_Dept table.
- But the **tuple for this department cannot be inserted** into this table as the **EID will have NULL value, which is not allowed because EID is primary key**.
- This kind of problem in the relation where some tuple cannot be inserted is known as insert anomaly.

# Deletion Anomalies

- Consider a relation Emp\_Dept(EID, Ename, City, DID, Dname, Manager) EID as a primary key

Emp_Dept					
<u>EID</u>	<u>Ename</u>	City	DID	<u>Dname</u>	Manager
1	Raj	Rajkot	1	CE	Shah
2	Meet	Surat	1	CE	Shah
3	Jay	Baroda	2	IT	Dave

A delete anomaly exists when **certain attributes are lost because of the deletion of another attribute.**

Want to delete (Jay) employee's detail

- Now consider **there is only one employee in some department (IT)** and that **employee leaves the organization.**
- So we **need to delete tuple of that employee (Jay).**
- But in addition to that **information about the department also deleted.**
- This kind of problem in the relation where deletion of some tuples can lead to loss of some other data not intended to be removed is known as delete anomaly.

# Update Anomalies

- Consider a relation Emp\_Dept(EID, Ename, City, Dname, Manager) EID as a primary key

Emp_Dept				
<u>EID</u>	<u>Ename</u>	<u>City</u>	<u>Dname</u>	<u>Manager</u>
1	Raj	Rajkot	CE	Sah
2	Meet	Surat	C.E	Shah
3	Jay	Baroda	Computer	Shaah
4	Hari	Rajkot	IT	Dave

An update anomaly exists **when one or more records (instance) of duplicated data is updated, but not all.**

Want to update manager of CE department

- Suppose the **manager of a (CE) department has changed**, this requires that the **Manager in all the tuples corresponding to that department must be changed** to reflect the new status.
- If we **fail to update all the tuples of given department**, then **two different records of employee working in the same department might show different Manager lead to inconsistency** in the database.

# How to deal with Insert, Delete and Update Anomalies?

<u>Emp Dept</u>					
<u>EID</u>	<u>Ename</u>	City	DID	<u>Dname</u>	Manager
1	Raj	Rajkot	1	CE	Shah
2	Meet	Surat	1	C.E	Shah
3	Jay	Baroda	2	IT	Dave
4	NULL	NULL	3	EC	NULL

<u>Emp</u>			
<u>EID</u>	<u>Ename</u>	City	DID
1	Raj	Rajkot	1
2	Meet	Surat	1
3	Jay	Baroda	2

<u>Dept</u>		
<u>DID</u>	<u>Dname</u>	Manager
1	CE	Shah
2	IT	Dave
3	EC	NULL

Such type of anomalies in the database design can be solved by using **normalization**.

# Normalization

Normalization is the **process of removing redundant data** from tables **to improve data integrity, scalability and storage efficiency.**

► What we do in normalization?

➔ Normalization generally involves **splitting an existing table into multiple (more than one) tables**, which can be **re-joined or linked** each time a query is executed.

# How many normal forms are there?

## Levels of Normalization:

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

As we move from 1NF to 5NF **number of tables** and **complexity increases** but **redundancy decreases**.

# First Normal Form (1NF)

## ► Conditions for 1NF

Each **cells of a table should contain a single value.**

- A relation R is in first normal form (1NF) if and only if it **does not contain any composite attribute or multi-valued attributes.**

OR

- A table is referred to as being in its First Normal Form if atomicity of the table is 1.



## 1NF (First Normal Form) [Example - Composite attribute]

Customer		
<u>CID</u>	Name	Address
C01	Raju	Jamnagar Road, Rajkot
C02	Mitesh	Nehru Road, Jamnagar
C03	Jay	C.G Road, Ahmedabad

- In customer relation **address is composite attribute** which is further divided into sub-attributes as “Road” and “City”.
- So customer relation is not in 1NF.

- ▶ **Problem:** It is **difficult to retrieve the list of customers living in 'Jamnagar' city** from customer table.
- ▶ The reason is that **address attribute is composite attribute** which **contains road name as well as city name in single cell**.
- ▶ It is possible that **city name word is also there in road name**.
- ▶ In our example, 'Jamnagar' word occurs in both records, in first record it is a part of road name and in second one it is the name of city.

# 1NF (First Normal Form) [Example - Composite attribute]

Customer		
<u>CID</u>	Name	Address
C01	Raju	Jamnagar Road, Rajkot
C02	<u>Mitesh</u>	Nehru Road, Jamnagar
C03	Jay	C.G Road, Ahmedabad



Customer			
<u>CID</u>	Name	Road	City
C01	Raju	Jamnagar Road	Rajkot
C02	<u>Mitesh</u>	Nehru Road	Jamnagar
C03	Jay	C.G Road	Ahmedabad

► **Solution:** Divide composite attributes into number of sub-attributes and insert value in proper sub-attribute.

**Exercise** Convert below relation into 1NF (First Normal Form)

Person		
<u>PID</u>	<u>Full Name</u>	City
P01	Raju Maheshbhai Patel	Rajkot

# 1NF (First Normal Form) [Example - Multivalued attribute]

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721, 9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

**Table 1**

Conversion to first normal form

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721	HARYANA	
1	RAM	9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

**Table 2**

# 1NF (First Normal Form) [Example - Multivalued attribute]

ID	Name	Courses
1	A	c1, c2
2	E	c3
3	M	c2, c3

In the above table, Course is a multi-valued attribute so it is not in 1NF.

Below Table is in 1NF as there is no multi-valued attribute:

ID	Name	Course
1	A	c1
1	A	c2
2	E	c3
3	M	c2
3	M	c3

## 2NF (Second Normal Form)

### Conditions for 2NF

- ▶ The first condition for the table to be in **Second Normal Form** is that the table has to be in **First Normal Form**.
- ▶ The table should not possess partial dependency.
  - ↪ every non-primary key attribute should be fully dependent on the primary key

## 2NF (Second Normal Form)

**Example:** Consider a ‘Location table’ :

This table have a primary key:  
**cust\_id, storeid.**

The non-key attribute is:  
**store\_location**

	cust_id	storeid	store_location
▶	1	D1	Toronto
	2	D3	Miami
	3	T1	California
	4	F2	Florida
	5	H3	Texas

In this case, store\_location only depends on storeid, which is a part of the primary key.

Hence, this table does **not** fulfill the **second normal form**.

## 2NF (Second Normal Form)

To bring the table to Second Normal Form, let's split the table into two parts.

cust_id	storeid
1	D1
2	D3
3	T1
4	F2
5	H3

storeid	store_location
D1	Toronto
D3	Miami
T1	California
F2	Florida
H3	Texas

After removing the partial functional dependency, the column `store_location` entirely depends on the primary key of that table, **storeid**. So, it's now in 2NF.



## 3NF (Third Normal Form)

### Conditions for 3NF:

1. The first condition for the table to be in Third Normal Form is that the table should be in the **Second Normal Form**.
2. The second condition is that **there should be no transitive dependency** for non-prime attributes, which indicates that non-prime attributes.



## 3NF (Third Normal Form)

**Example:** Consider a 'student table' :

stu_id	name	subid	sub	address
1	Arun	11	SQL	Delhi
2	Varun	12	Java	Bangalore
3	Harsh	13	C++	Delhi
4	Keshav	12	Java	Kochi

Here,  $\text{stu\_id} \rightarrow \text{subid}$ , and  $\text{subid} \rightarrow \text{sub}$

Therefore,  $\text{stu\_id} \rightarrow \text{sub}$  via  $\text{subid}$ .

This is transitive FD and does not fulfill the third normal form criteria.

## 3NF (Third Normal Form)

To change the table to the third normal form, let's divide the table;

stu_id	name	subid	address
1	Arun	11	Delhi
2	Varun	12	Bangalore
3	Harsh	13	Delhi
4	Keshav	12	Kochi

subid	subject
11	SQL
12	java
13	C++
12	Java

Now, all the non-key attributes are now fully functional, dependent only on the primary key. So, it's now in 3NF.

In the first table, **subid**, and **addresses** only depend on **stu\_id**.

In the second table, the **sub** only depends on **subid**.

# Boyce Codd Normal Form (BCNF)

It's a strict version of 3NF. It's often referred to as 3.5NF.

The first condition for the table to be in Boyce Codd Normal Form is that the table should be in the third normal form.

Secondly, for every functional dependency  $X \rightarrow Y$ ,  $X$  should be the primary key of the table.

# Boyce Codd Normal Form (BCNF)

The subject table follows these conditions:

- Each student can enroll in multiple subjects.
- Multiple professors can teach a particular subject.
- For each subject, it assigns a professor to the student.

stuid	subject	professor
1	SQL	Prof. Mishra
2	Java	Prof. Anand
2	C++	Prof. Kanth
3	Java	Prof. James
4	DBMS	Prof. Lokesh

- Here, **student\_id** and **subject** together form the primary key.
- Also, professor  $\rightarrow$  subject.

BCNF does not follow as a subject is a prime attribute, the professor is a non-prime attribute.

# Boyce Codd Normal Form (BCNF)

To transform the table into the BCNF, let's divide the table into two parts: *both table will hold a newly created column profid.*

stuid	profid	profid	subject	professor
1	101	101	SQL	Prof. Mishra
2	102	102	Java	Prof. Anand
2	103	103	C++	Prof. Kanth
3	102	102	Java	Prof. James
4	104	104	DBMS	Prof. Lokesh

# Introduction of 4th and 5th Normal Form

Two of the highest levels of database normalization are the fourth normal form (4NF) and the fifth normal form (5NF).

- ✓ Multivalued dependencies are handled by 4NF
- ✓ Join dependencies are handled by 5NF.



# 4NF (Fourth Normal Form)

## Conditions for 4NF:

1. It should be in the **Boyce-Codd Normal Form (BCNF)**.
2. The table should not contain more than one **Multi-valued Dependency**.

# 4NF (Fourth Normal Form)

STU_ID	COURSE	HOBBY
21	Computer	Dancing
21	Math	Singing
34	Chemistry	Dancing
74	Biology	Cricket
59	Physics	Hockey

The above is not in 4NF, student with STU\_ID, **21** contains two courses, **Computer** and **Math** and two hobbies, **Dancing** and **Singing**.

So there is a Multi-valued dependency on STU\_ID.



# 4NF (Fourth Normal Form)

So to make the above table into 4NF, we can decompose it into two tables:

**STUDENT\_COURSE**

STU_ID	COURSE
21	Computer
21	Math
34	Chemistry
74	Biology
59	Physics

**STUDENT\_HOBBY**

STU_ID	HOBBY
21	Dancing
21	Singing
34	Dancing
74	Cricket
59	Hockey

# 5NF (Fifth Normal Form)

## Conditions for 5NF:

1. Table should be already in 4NF.
2. It cannot be further non loss decomposed (join dependency).

# 5NF (Fifth Normal Form)

## Example:

The below relation violates the Fifth Normal Form (5NF) of Normalization bcos it can be decomposed further.

EmpName	EmpSkills	EmpJob (Assigned Work)
David	Java	E145
John	JavaScript	E146
Jamie	jQuery	E146
Emma	Java	E147

# 5NF (Fifth Normal Form)

The above relation can be decomposed into the following three tables:

EmpName	EmpSkills
David	Java
John	JavaScript
Jamie	jQuery
Emma	Java

EmpName	EmpJob
David	E145
John	E146
Jamie	E146
Emma	E147

EmpSkills	EmpJob
Java	E145
JavaScript	E146
jQuery	E146
Java	E147

Now, the relation is in 5NF as it does not violate the property of lossless join.

**Thank You!!!**

# x DIGITAL LEARNING CONTENT

0



## Parul<sup>®</sup> University



[www.paruluniversity](http://www.paruluniversity)

