Unit 4

Relational Database Design

Why Relational Database Design

- Remove unnecessary data
- Remove the duplicate data
- Maintain the consistence and integrity
- Easy to maintain

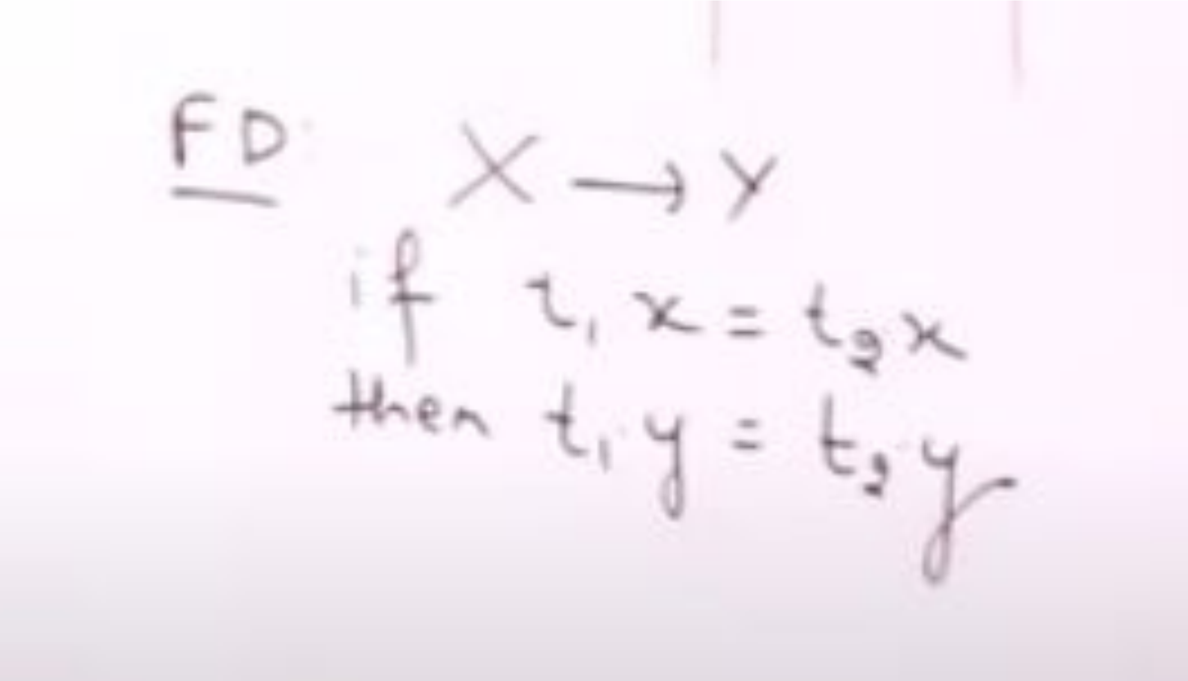
After studying this lecture, you will be able to:

- Understand the concept of Relationship Design
- Discuss about Null Values in Tuples
- Describe the Functional Dependencies and
- Understand the concept of Multivalued Dependencies

Functional Dependency

- A functional dependency is **a constraint that specifies the relationship between two sets of attributes where one set can accurately determine the value of other sets**.
- It is denoted as **X → Y**, where X is a set of attributes that is capable of determining the value of Y.
- The attribute set on the left side of the arrow, **X** is called **Determinant**, while on the right side, **Y** is called the **Dependent**.

Condition for Functional Dependency



Student

RollNo	Name	SPI	BL
101	Raju	8	0
102	Mitesh	7	1
103	Jay	7	0

X → Y

RollNo → Name, SPI, BL

Y is functionally dependent on the X or **X functionally determines Y**.

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 account_no → {balance, branch}.

R_No	Name	Marks	Dept	Course
1	A	78	CS	C1
2	B	60	EE	C1
3	A	78	CS	C3
4	B	60	EE	C3
5	C	80	IT	C2
6	D	80	EC	C2

R_NO ->Name fd y

Name->R_no n

R_no ->Marks y

Dept ->Course n

Course->Dept n

Marks-Dept n

(r_no,Name) -> Marks y

Type of Functional Dependency

- Full functional Dependency
- Partial functional Dependency
- Transitive Functional Dependency
- Trivial Functional Dependency
- Non Trivial Functional Dependency

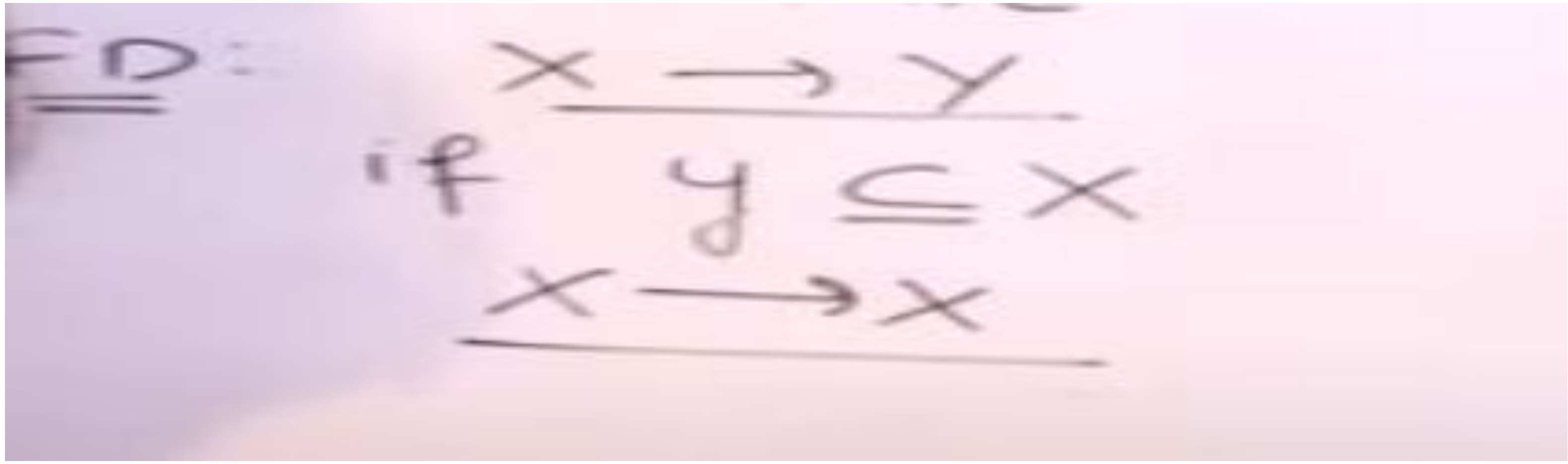
Trivial Functional Dependency

- X → Y is trivial FD if **Y is a subset of X**
- Eg. {Roll_No, Department_Name, Semester} → Roll_No

Nontrivial Functional Dependency

- X → Y is nontrivial FD if **Y is not a subset of X**
- Eg. {Roll_No, Department_Name, Semester} → Student_Name

Trivial functional dependency



Non Trivial FD

$$X \rightarrow Y$$

$$X \cap Y = \phi$$

- Armstrong's axioms are a set of rules used to infer (derive) all the functional dependencies on a relational database.

Reflexivity

- If B is a subset of A
  - then $A \rightarrow B$

Augmentation

- If $A \rightarrow B$
  - then $AC \rightarrow BC$

Self-determination

- If $A \rightarrow A$

Transitivity

- If $A \rightarrow B$ and $B \rightarrow C$
  - then $A \rightarrow C$

Pseudo Transitivity

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

Decomposition

- If $A \rightarrow BC$
  - then $A \rightarrow B$ and $A \rightarrow C$

Union

- If $A \rightarrow B$ and $A \rightarrow C$
  - then $A \rightarrow BC$

Composition

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

Full Functional Dependency

- In a relation, the attribute B is fully functional dependent on A if **B is functionally dependent on A, but not on any proper subset of A**.
- Eg. {Roll_No, Semester, Department_Name} → SPI
- We **need all three {Roll_No, Semester, Department_Name} to find SPI**.

Partial Functional Dependency

- In a relation, the attribute B is partial functional dependent on A if **B is functionally dependent on A as well as on any proper subset of A**.
- If there is some attribute that can be removed from A and the still dependency holds then it is partial functional dependancy.
- Eg. {Enrollment_No, Department_Name} → SPI
- **Enrollment_No is sufficient to find SPI**, Department_Name is not required to find SPI.

What is closure of a set of FDs?

- Given a set F set of functional dependencies, there are certain other **functional dependencies that are logically implied by F**.
- E.g.: $F = \{A \rightarrow B$ and $B \rightarrow C\}$, then we can infer that $A \rightarrow C$ (by transitivity rule)
- The set of **functional dependencies (FDs) that is logically implied by F** is called the closure of F.
- It is denoted by $\mathbf{F^+}$.

Closure of a set of FDs

- Suppose we are given a relation schema **R(A,B,C,G,H,I)** and the set of functional dependencies are:
  - **F = (A → B, A → C, CG → H, CG → I, B → H)**
- The functional dependency **A → H** is logical implied.

- Compute the closure of the following set F of functional dependencies FDs for relational schema **R = (A,B,C,D,E,F):**
  - **F = (A → B, A → C, CD → E, CD → F, B → E)**
- Find out the closure of F.

Closure of attribute sets

What is a closure of attribute sets?

- Given a set of attributes α, the closure of α under F is the **set of attributes that are functionally determined by α under F**.
- It is denoted by **α^+**.

Algorithm

- Algorithm to compute α^+, the closure of α under F
  - Steps
    1. result = α
    2. *while* (changes to result) *do*
       - for each $\beta \rightarrow \gamma$ in F do
         - begin
           - if $\beta \subseteq$ result then result = result ∪ γ
           - else result = result
         - end

- Given functional dependencies (FDs) for relational schema R = (A,B,C,D,E):
- F = {$A \rightarrow BC$, $CD \rightarrow E$, $B \rightarrow D$, $E \rightarrow A$}
  - Find Closure for A
  - Find Closure for CD
  - Find Closure for B
  - Find Closure for BC
  - Find Closure for E

Canonical cover

What is extraneous attributes?

- Let us consider a relation R with schema R = (A, B, C) and set of functional dependencies FDs **F = { AB → C, A → C }**.
- In **AB → C, B is extraneous attribute**. The reason is, there is another FD **A → C**, which means when **A alone can determine C**, the use of B is unnecessary (extra).
- An attribute of a functional dependency is said to be extraneous if we can **remove it without changing the closure of the set of functional dependencies**.

What is canonical cover?

- A canonical cover of F is a **minimal set of functional dependencies** equivalent to F, having **no redundant dependencies or redundant parts of dependencies**.
- It is denoted by $\mathbf{F_c}$
- A canonical cover for F is a set of dependencies F_c such that
  - **F logically implies** all dependencies in $\mathbf{F_c}$ and
  - $\mathbf{F_c}$ **logically implies** all dependencies in **F** and
  - **No** functional dependency in $\mathbf{F_c}$ contains an **extraneous attribute** and
  - Each **left side** of functional dependency in $\mathbf{F_c}$ is **unique**.

Decomposition Rule

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

$F_c = \{A \rightarrow BC\}$

Union Rule

Algorithm to find canonical cover

- *Repeat*
  - Use the **union rule** to replace any dependencies in F **$\alpha1 \rightarrow \beta1$** and **$\alpha1 \rightarrow \beta2$ with $\alpha1 \rightarrow \beta1\beta2$**
  - Find a functional dependency $\alpha \rightarrow \beta$ with an **extraneous attribute** either in α or in β

    /* Note: test for extraneous attributes done using F_c, not F */
    - If an **extraneous attribute is found, delete it** from $\alpha \rightarrow \beta$
- *until* F does not change

  /* Note: Union rule may become applicable after some extraneous attributes have been deleted, so it has to be re-applied */

Example

- Consider the relation schema R = (A, B, C) with FDs

  F = {A → BC, B → C, A → B, AB → C}

- Find canonical cover.

- Combine A → BC and A → B into A → BC (Union Rule)
  - Set is {A → BC, B → C, AB → C}
- A is extraneous in AB → C
  - Check if the result of deleting A from AB → C is implied by the other dependencies
    - Yes: in fact, B → C is already present
  - Set is {A → BC, B → C}
- C is extraneous in A → BC
  - Check if A → C is logically implied by A → B and the other dependencies
    - Yes: using transitivity on A → B and B → C.
  - The canonical cover is: **A → B, B → C**

- Consider the relation schema R = (A, B, C, D, E, F) with FDs

  $F = \{A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A\}$
- Find canonical cover.

- The left side of each FD in F is unique.
- Also none of the attributes in the left side or right side of any of the FDs is extraneous.
- Therefore the canonical cover F_c is equal to F.
- $F_c = \{A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A\}$

Decomposition

What is decomposition?

- Decomposition is the **process of breaking down given relation** into **two or more relations**.
- Relation R is replaced by two or more relations in such a way that:
  - Each new relation contains a **subset** of the **attributes of R**
  - Together, they all **include all tuples** and **attributes of R**
- Types of decomposition
  - Lossy decomposition
  - Lossless decomposition (non-loss decomposition)

Lossy decomposition

- The decomposition of relation R into R1 and R2 is lossy when the join of R1 and R2 does not yield the same relation as in R.
- This is also referred as lossy-join 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.

Customer

Ano	Balance	Bname
A01	5000	Rajkot
A02	5000	Surat

Table-1

Ano	Balance
A01	5000
A02	5000

Table-2

Balance	Bname
5000	Rajkot
5000	Surat

Not Same

Customer

Ano	Balance	Bname
A01	5000	Rajkot
A01	5000	Surat
A02	5000	Rajkot
A02	5000	Surat

Lossless decomposition

- The decomposition of relation R into R1 and R2 is lossless when the join of R1 and R2 produces the same relation as in R.
- This is also referred as a non-additive (non-loss) decomposition.
- All decompositions must be lossless.

Customer

Ano	Balance	Bname
A01	5000	Rajkot
A02	5000	Surat

Table-1

Ano	Balance
A01	5000
A02	5000

Table-2

Ano	Bname
A01	Rajkot
A02	Surat

Same

Customer

Ano	Balance	Bname
A01	5000	Rajkot
A02	5000	Surat

Anomaly in Database design

- Anomalies are **problems that can occur in poorly planned, un-normalized database** where all the data are stored in one table.
- There are three types of anomalies that can arise in the database because of redundancy are
  - Insert anomaly
  - Delete anomaly
  - Update / Modification anomaly

Insert anomaly

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

Emp_Dept

EID	Ename	City	DID	Dname	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.

Delete anomaly

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

Emp_Dept

EID	Ename	City	DID	Dname	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 anomaly

- Consider a relation Emp_Dept(EID, Ename, City, Dname, Manager) EID as a primary key

Emp_Dept

EID	Ename	City	Dname	Manager
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 anomaly

Emp_Dept

EID	Ename	City	DID	Dname	Manager
1	Raj	Rajkot	1	CE	Shah
2	Meet	Surat	1	C.E	Shah
3	Jay	Baroda	2	IT	Dave
NULL	NULL	NULL	3	EC	NULL

Emp

EID	Ename	City	DID
1	Raj	Rajkot	1
2	Meet	Surat	1
3	Jay	Baroda	2

Dept

DID	Dname	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**.
  - data integrity (completeness, accuracy and consistency of data)
  - scalability (ability of a system to continue to function well in a growing amount of work)
  - storage efficiency (ability to store and manage data that consumes the least amount of space)

- 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 issued (executed).

Types of Normal forms

- Normal forms:
  - 1NF (First normal form)
  - 2NF (Second normal form)
  - 3NF (Third normal form)
  - BCNF (Boyce–Codd normal form)
  - 4NF (Forth normal form)
  - 5NF (Fifth normal form)

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

First Normal Form

- It **should only have single(atomic) valued attributes/columns**.
- Values stored in a column should be of the same domain.
- All the columns in a table should have unique names. And the order in which data is stored, does not matter.

Customer

CID	Name	Address
C01	Raju	Jamnagar Road, Rajkot
C02	Mitesh	Nehru Road, Jamnagar
C03	Jay	C.G Road, Ahmedabad

Customer

CID	Name	Road	City
C01	Raju	Jamnagar Road	Rajkot
C02	Mitesh	Nehru Road	Jamnagar
C03	Jay	C.G Road	Ahmedabad

Student

<u>Rno</u>	Name	FailedinSubjects
101	Raju	DS, DBMs
102	Mitesh	DBMS, DS
103	Jay	DS, DBMS, DE
104	Jeet	DBMS, DE, DS
105	Harsh	DE, DBMS, DS
106	Neel	DE, DBMS

→

Student

<u>Rno</u>	Name
101	Raju
102	Mitesh
103	Jay
104	Jeet
105	Harsh
106	Neel

Result

<u>RID</u>	Rno	Subject
1	101	DS
2	101	DBMS
3	102	DBMS
4	102	DS
5	103	DS
…	…	…

Second Normal Form

1. It is in 1 NF and each table should contain the primary key
2. A relation R is in 2 NF
    1. If and only if it is in 1NF
    2. Every non –primary attribute is fully depend on the primary key (No partial dependency exist)

What is Partial Dependency?

- Partial Dependency occurs when a non-prime attribute is functionally dependent on part of a candidate key.

- The 2nd Normal Form (2NF) eliminates the Partial Dependency.

Example

<StudentProject>

StudentID	ProjectNo	StudentName	ProjectName
S01	199	Katie	Geo Location
S02	120	Ollie	Cluster Exploration

The prime key attributes are **StudentID** and **ProjectNo**, and

StudentID = Unique ID of the student

StudentName = Name of the student

ProjectNo = Unique ID of the project

ProjectName = Name of the project

As stated, the non-prime attributes i.e. **StudentName** and **ProjectName** should be functionally dependent on part of a candidate key, to be Partial Dependent.

To remove Partial Dependency and violation on 2NF, decompose the tables –

<StudentInfo>

StudentID	ProjectNo	StudentName
S01	199	Katie
S02	120	Ollie

<ProjectInfo>

ProjectNo	ProjectName
199	Geo Location
120	Cluster Exploration

Now the relation is in 2nd Normal form of Database Normalization.

Customer

CID	ANO	AccessDate	Balance	BranchName
C01	A01	01-01-2017	50000	Rajkot
C02	A01	01-03-2017	50000	Rajkot
C01	A02	01-05-2017	25000	Surat
C03	A02	01-07-2017	25000	Surat

Solution: Decompose relation in such a way that **resultant relations do not have any partial FD**.

- **Remove partial dependent attributes** from the relation that violets 2NF.
- **Place them in separate relation** along with the **prime attribute on which they are fully dependent**.
- The **primary key of new relation** will be the **attribute on which it is fully dependent**.
- **Keep other attributes same** as in that table with the **same primary key**.

Table-1		
ANO	Balance	BranchName
A01	50000	Rajkot
A02	25000	Surat

Table-2		
CID	ANO	AccessDate
C01	A01	01-01-2017
C02	A01	01-03-2017
C01	A02	01-05-2017
C03	A02	01-07-2017

3rd Normal Form

- It is in 2NF
- There is no Transitive dependency occur

- A relation R is in third normal form (3NF)
  - if and only if it is in **2NF** and
  - **every non-key attribute is non-transitively dependent on the primary key**

Transitive Functional Dependency

- In a relation, if attribute(s) **A → B and B → C, then A → C (means C is transitively depends on A** via B).

Sub_Fac		
Subject	**Faculty**	**Age**
DS	Shah	35
DBMS	Patel	32
DF	Shah	35

- Eg. Subject → Faculty & Faculty → Age then Subject → Age
- Therefore as per the rule of transitive dependency: Subject → Age should hold, that makes sense because if we know the subject name we can know the faculty's age.

Check 3NF or not?

Customer

ANO	Balance	BranchName	BranchAddress
A01	50000	Rajkot	Kalawad road
A02	40000	Rajkot	Kalawad Road
A03	35000	Surat	C.G Road
A04	25000	Surat	C.G Road

3NF

Customer

ANO	Balance	BranchName	BranchAddress
A01	50000	Rajkot	Kalawad road
A02	40000	Rajkot	Kalawad Road
A03	35000	Surat	C.G Road
A04	25000	Surat	C.G Road

Table-1

BranchName	BranchAddress
Rajkot	Kalawad road
Surat	C.G Road

Table-2

ANO	Balance	BranchName
A01	50000	Rajkot
A02	40000	Rajkot
A03	35000	Surat
A04	25000	Surat

BCNF (Boyce –codd Normal Form)

- It is the based on the concept of determinant
- It is in 3NF
- Every determinant should be primary key

A relation R is in Boyce-Codd normal form (BCNF)

- if and only if it is in 3NF and
- for every functional dependency $X \rightarrow Y$, X should be the primary key of the table.

BCNF or not?

Student		
RNO	**Subject**	**Faculty**
101	DS	Patel
102	DBMS	Shah
103	DS	Jadeja
104	DBMS	Dave
105	DBMS	Shah
102	DS	Patel
101	DBMS	Dave
105	DS	Jadeja

BCNF

Student

RNO	Subject	Faculty
101	DS	Patel
102	DBMS	Shah
103	DS	Jadeja
104	DBMS	Dave
105	DBMS	Shah
102	DS	Patel
101	DBMS	Dave
105	DS	Jadeja

Table-1

Faculty	Subject
Patel	DS
Shah	DBMS
Jadeja	DS
Dave	DBMS

Table-2

RNO	Faculty
101	Patel
102	Shah
103	Jadeja
104	Dave
105	Shah
102	Patel
101	Dave
105	Jadeja

Forth Normal Form

- Conditions for 4NF
- A relation R is in fourth normal form (4NF)
  - if and only if it is in **BCNF** and
  - **has no multivalued dependencies**

Student

RNO	Subject	Faculty
101	DS	Patel
101	DBMS	Patel
101	DS	Shah
101	DBMS	Shah

→

Subject

RNO	Subject
101	DS
101	DBMS

Faculty

RNO	Faculty
101	Patel
101	Shah

- Above student table **has multivalued dependency**. So student table is **not in 4NF**.

Functional dependency and multivalued attributes

- A table can have both functional dependency as well as multi-valued dependency together.
  - RNO → Address
  - RNO →→ Subject
  - RNO →→ Faculty

Student

RNO	Address	Subject	Faculty
101	C. G. Road, Rajkot	DS	Patel
101	C. G. Road, Rajkot	DBMS	Patel
101	C. G. Road, Rajkot	DS	Shah
101	C. G. Road, Rajkot	DBMS	Shah

→

Subject

RNO	Subject
101	DS
101	DBMS

Faculty

RNO	Faculty
101	Patel
101	Shah

Address

RNO	Address
101	C. G. Road, Rajkot

Fifth Normal Form

- Conditions for 5NF
- A relation R is in fifth normal form (5NF)
  - if and only if it is in **4NF** and
  - it **cannot have a lossless decomposition in to any number of smaller tables** (relations).

Student_Result

RID	RNO	Name	Subject	Result
1	101	Raj	DBMS	Pass
2	101	Raj	DS	Pass
3	101	Raj	DF	Pass
4	102	Meet	DBMS	Pass
5	102	Meet	DS	Fail
6	102	Meet	DF	Pass
7	103	Suresh	DBMS	Fail
8	103	Suresh	DS	Pass

Student_Result relation is **further decomposed** into sub-relations. So the above relation is **not in 5NF**.

Student_Result

RID	RNO	Name	Subject	Result
1	101	Raj	DBMS	Pass
2	101	Raj	DS	Pass
3	101	Raj	DF	Pass
4	102	Meet	DBMS	Pass
5	102	Meet	DS	Fail
6	102	Meet	DF	Pass
7	103	Suresh	DBMS	Fail
8	103	Suresh	DS	Pass

Student

RNO	Name
101	Raj
102	Meet
103	Suresh

Subject

SID	Name
1	DBMS
2	DS
3	DF

Result

RID	RNO	SID	Result
1	101	1	Pass
2	101	2	Pass
3	101	3	Pass
4	102	1	Pass
5	102	2	Fail
6	102	3	Pass
7	103	1	Fail
8	103	2	Pass

None of the above relations can be further decomposed into sub-relations. So the above database is in 5NF.