Unit :III
Functional Dependency and Decomposition

Que: Functional Dependency

- A functional dependency describes the relationship between two sets of attributes within a relational database table.
- It indicates that the values of one set of attributes (known as the dependent attributes) are determined by the values of another set of attributes (known as the determinant attributes).
- Functional dependencies are denoted using arrow notation (→).
- The arrow (→) indicates that the set of attributes on the left side determines the set of attributes on the right side.

Example,

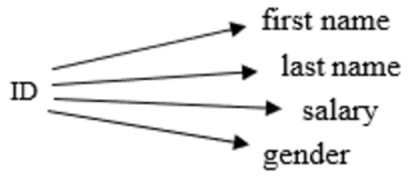
if we have attributes A and B, and A determines B, we represent it as A → B.

Que: Functional dependency diagram and examples

- Consider a company collects information about each employee such as the employee's identification number (ID), their first name, last name, salary and gender.
- As is typical, each employee is given a unique ID which serves to identify the employee.
- Hence for each value of ID there is at most one value for first name, last name, salary and gender.

Therefore, we have four functional dependencies where ID is the determinant;
We can show this as a list or graphically:

ID ➔ first name
ID ➔ last name
ID ➔ salary
ID ➔ gender



- For example, consider the gender attribute – we need to allow for more than one employee for a given gender, and so we cannot have a situation where gender functionally determines ID. So, gender ID cannot exist.
- Now consider the first name attribute. Again, we need to allow for more than one employee to have the same first name and so first name cannot determine anything. Similarly, for other attributes.

Que: Full function dependency (FFD)

- A full functional dependency is a functional dependency where the dependent attributes are determined by the determinant attributes.
- For example, in the database of employees, the employee ID number fully determines the employee's name, address, and other personal information

Example

Employee → employee ID, name, and department.

The **employee ID determines** the employee name and the department in which they work. This represents a full functional dependency.

Emp_ID	**Emp_Name**	**Dept.**
001	Amit	Computer
002	Ram	IT
003	Shayam	Mechnical

- In this table, the employee ID uniquely identifies each employee, and the employee name and department are fully dependent on the employee ID.

Employee ID (Determinant) → Employee Name, Department (Dependent attributes)

- In this example, the employee ID is a primary key of the table, and it uniquely identifies each row.
- Therefore, the functional dependency is full because all attributes in the determinant (employee ID) are necessary to determine the dependent attributes (employee name and department

Que: Partial functional dependency:

- A functional dependency is considered partial if removing one or more attributes from the determinant would still maintain the dependency.
- In this case, some attributes in the determinant are not strictly necessary to determine the dependent attributes.
- A table that represents customer purchases, including the

customer ID, customer name, product ID, product name.

- In this example, the customer ID determines the customer name, while the product ID determines the product name.
- However, the **customer name and product name are not dependent on each other**. This represents a partial functional dependency..

Customer ID	Customer Name	Product ID	Product Name
001	Amit	P001	Laptop
001	Amit	P002	Mouse
002	Ram	P003	Keyboard
002	Ram	P004	Monitor

- In this table, the **customer ID** uniquely identifies each customer, and the **customer name** is fully dependent on the customer ID.

- Similarly, the **product ID** uniquely identifies each product, and the **product name** is fully dependent on the product ID.

  Customer ID (Determinant) → Customer Name (Dependent attribute)

  Product ID (Determinant) → Product Name (Dependent attribute)

- The functional dependencies indicate that for a given customer ID, there is a unique customer name, and for a given product ID, there is a unique product name.

- In this example, both the **customer ID and product ID can be considered as candidate keys or primary keys of the table.**

Que:Types of Functional Dependencies

1. Trivial functional dependency
2. Non-Trivial functional dependency
3. Multivalued functional dependency
4. Transitive functional dependency

Trivial Functional Dependency
- In Trivial functional dependency, a dependent is always a subset of the determinant.
- In other words, a functional dependency is called trivial if the attributes on the right side are the subset of the attributes on the left side of the functional dependency.
- X → Y is called a trivial functional dependency if Y is the subset of X.

Employee table

Employee_Id	Name	Age
1	Ram	24
2	Amit	34
3	Hardik	26
4	Mohan	29

Here,
{ Employee_Id, Name } → { Name }
is a Trivial functional dependency, since the dependent Name is the subset of determinant { Employee_Id, Name }.

{ Employee_Id } → { Employee_Id }
{ Name } → { Name }
{ Age } → { Age } are also Trivial.

Non-Trivial Functional Dependency

- It is the opposite of Trivial functional dependency. In Non-Trivial functional dependency, dependent if not a subset of the determinant.
- X → Y is called a Non-trivial functional dependency if Y is not a subset of X.
- So, a functional dependency X → Y where X is a set of attributes and Y is also a set of the attribute but not a subset of X, then it is called Non-trivial functional dependency.

Employee_Id	Name	Age
1	Ram	24
2	Amit	34
3	Hardik	26
4	Mohan	29

Here,
{ Employee_Id } → { Name }
is a non-trivial functional dependency because Name(dependent) is not a subset of Employee_Id(determinant).

Similarly,
{ Employee_Id, Name } → { Age } is also a non-trivial functional dependency**.**

Multi valued Functional Dependency

In Multivalued functional dependency, attributes in the dependent set are not dependent on each other.
For example,
X → { Y, Z }, if there exists is no functional dependency between Y and Z, then it is called as Multivalued functional dependency.

Employee_Id	Name	Age
1	Ram	24
2	Amit	34
3	Hardik	26
4	Mohan	29
5	Manoj	24

Here,
{ Employee_Id } → { Name, Age } is a Multivalued functional dependency,
Since the dependent attributes Name, Age are not functionally dependent

Transitive Functional Dependency

- Consider two functional dependencies A → B and B → C then according to the transitivity axiom A → C must also exist. This is called a transitive functional dependency.
- In other words, dependent is indirectly dependent on determinant in Transitive functional dependency.

Employee_Id	Name	Department	Street Number
1	Ram	Computer	11
2	Amit	IT	24
3	Hardik	Computer	11
4	Mohan	Mechanical	71
5	Manoj	Civil	21

Here,
{ Employee_Id → Department } and { Department → Street Number } holds true.
Hence, according to the axiom of transitivity, { Employee_Id → Street Number } is a valid functional dependency.

Que: Armstrong's Axioms(Rules) For Functional Dependencies

- William Armstrong in 1974 suggested a few rules related to functional dependency.
- They are called RAT rules.

1. Reflexivity:

If A is a set of attributes and B is a subset of A, then the functional dependency A → B holds true.

For example

1. { Employee_Id, Name } → Name is valid.
2. If B is subset of A then A→B

2. Augmentation:

If a functional dependency A → B holds true, then appending any number of the attribute to both sides of dependency doesn't affect the dependency. It remains true.

For example

1. if { Employee_Id, Name } → { Name } holds true then,
   { Employee_Id, Name, Age } → { Name, Age }

2. if A→B then AC → BC

3. Transitivity:

If two functional dependencies X → Y and Y → Z hold true, then X → Z also holds true by the rule of Transitivity.

For example

1. if { Employee_Id } → { Name } holds true and { Name } → { Department } holds true, then { Employee_Id } → { Department } also holds true.

2. IF A→B and B→C then A→C

4. Self determination

A → A

5. Decomposition

If A→BC then A→B and A→C

6. Union

If A→ B and A→C then A→BC

Advantages of Functional Dependency in DBMS

1. It is used to maintain the quality of data in the database.
2. It expresses the facts about the database design.
3. It helps in clearly defining the meanings and constraints of databases.
4. It helps to identify bad designs.
5. Functional Dependency removes data redundancy where the same values should not be repeated at multiple locations in the same database table.
6. The process of Normalization starts with identifying the candidate keys in the relation. Without functional dependency, it's impossible to find candidate keys and normalize the database.

Que: What is Decomposition

- Decomposition refers to breaking down a complex problem or program into smaller, more manageable parts, which can be solved or implemented separately.

Or

Decomposition means dividing a large and complex table into multiple small and easy tables.

They are two types

1. Lossless decomposition
2. Lossy decomposition

1. Lossless decomposition

- Loss means data loss while decomposing a relational table.
- A lossless decomposition is somewhat in which data is not lost because JOIN is used.
- First, we decompose a large table into small appropriate tables, then apply natural join to reconstruct the original table.

Student Details

Sid	Name	Subject	Mobile	Address
1	Raj	English	65468154	51, Vaishalinagar
2	Jyoti	Home Science	87668545	4a, Sukhsagar
3	Vikash	Maths	26865948	H7, Civil Lines
1	Harsh	Maths	Null	R32, Gokul Villa
3	Ajay	Science	86516529	26, Karoli

We can decompose it into **two sim**ple tables as given below:

Student Subject Details:

Sid	Name	Subject
1	Raj	English
2	Jyoti	Home Science
3	Vikash	Maths
1	Harsh	Maths
3	Ajay	Science

Student Personal Details:

Sid	Mobile	Address
1	65468154	51, Vaishalinagar
2	87668545	4a, Sukhsagar
3	26865948	H7, Civil Lines
1	Null	R32, Gokul Villa
3	86516529	26, Karoli

If we want to see a common table then we can apply **Natural JOIN** between both tables like this:

Student Subject Details $\bowtie$ **Student Personal Details**

Sid	Name	Subject	Mobile	Address
1	Raj	English	65468154	51, Vaishalinagar
2	Jyoti	Home Science	87668545	4a, Sukhsagar
3	Vikash	Maths	26865948	H7, Civil Lines
1	Harsh	Maths	Null	R32, Gokul Villa
3	Ajay	Science	86516529	26, Karoli

2. Lossy decomposition

➢ lossy decomposition is when a relation gets decomposed into multiple relational schemas, in such a way that retrieving the original relation leads to a loss of information.

Sid	Name	Subject	Mobile	Address
1	Raj	English	65468154	51, Vaishalinagar
2	Jyoti	Home Science	87668545	4a, Sukhsagar
3	Vikash	Maths	26865948	H7, Civil Lines
1	Harsh	Maths	Null	R32, Gokul Villa
3	Ajay	Science	86516529	26, Karoli

If we divide this student details table into two sections as given below:

Student Subject Details:

Sid	Name	Subject
1	Raj	English
2	Jyoti	Home Science
3	Vikash	Maths
1	Harsh	Maths
3	Ajay	Science

Student Personal Details:

Mobile	Address
65468154	51, Vaishalinagar
87668545	4a, Sukhsagar
26865948	H7, Civil Lines
Null	R32, Gokul Villa
86516529	26, Karoli

- In this Student Personal Details table, the SID column is not included, so now we don't know that these mobiles numbers and address belongs to whom.
- So always decompose a table in such a manner that the data may be easily reconstructed and retrieved.