

## **Functional Dependency (FD) :-**

The functional dependency is a relationship that exists between two attributes. It typically exists between the primary key and non-key attribute within a table.

Functional dependency helps you to maintain the quality of data in the database. A functional dependency is denoted by an arrow  $\rightarrow$ . The functional dependency of X on Y is represented by  $X \rightarrow Y$ . Functional Dependency plays a vital role to find the difference between good and bad database design.

The left side of FD is known as a determinant, the right side of the production is known as a dependent.

**Example:** Employee table with attributes: Emp\_Id, Emp\_Name, Emp\_Address. Here Emp\_Id attribute can uniquely identify the Emp\_Name attribute of employee table because if we know the Emp\_Id, we can tell that employee name associated with it.

Functional dependency can be written as:

$\text{Emp\_Id} \rightarrow \text{Emp\_Name}$

(Emp\_Name is functionally dependent on Emp\_Id)

## **Types of Functional Dependencies**

- Trivial functional dependency
- non-trivial functional dependency
- Multivalued dependency
- Transitive dependency

## **Trivial functional dependency :**

- $A \rightarrow B$  has trivial functional dependency if B is a subset of A.
- The following dependencies are also trivial like:  $A \rightarrow A$ ,  $B \rightarrow B$

## **Example:**

Consider a table with two columns Employee\_Id and Employee\_Name.

$\{\text{Employee\_id}, \text{Employee\_Name}\} \rightarrow \text{Employee\_Id}$  is a trivial functional dependency as

Employee\_Id is a subset of  $\{\text{Employee\_Id}, \text{Employee\_Name}\}$ .

Also,  $\text{Employee\_Id} \rightarrow \text{Employee\_Id}$  and  $\text{Employee\_Name} \rightarrow \text{Employee\_Name}$  are trivial dependencies too.

### **Non-trivial functional dependency :**

- $A \rightarrow B$  has a non-trivial functional dependency if B is not a subset of A.
- When A intersection B is NULL, then  $A \rightarrow B$  is called as complete non-trivial.

#### **Example:**

ID  $\rightarrow$  Name,  
Name  $\rightarrow$  DOB

### **Multivalued dependency :**

Multivalued dependency occurs in the situation where there are multiple independent multivalued attributes in a single table. A multivalued dependency is a complete constraint between two sets of attributes in a relation. It requires that certain tuples be present in a relation.

#### **Example:**

Car_model	Maf_year	Color
H001	2017	Metallic
H001	2017	Green
H005	2018	Metallic
H005	2018	Blue
H010	2015	Metallic
H033	2012	Gray

In this example, maf\_year and color are independent of each other but dependent on car\_model. In this example, these two columns are said to be multivalue dependent on car\_model.

This dependence can be represented like this:

car\_model -> maf\_year

car\_model-> colour

## **Transitive dependency:**

A transitive is a type of functional dependency which happens when it is indirectly formed by two functional dependencies.

### **Example:**

Company	CEO	Age
Microsoft	Satya Nadella	51
Google	Sundar Pichai	46
Alibaba	Jack Ma	54

{Company} -> {CEO} (if we know the company, we know its CEO's name)

{CEO} -> {Age} If we know the CEO, we know the Age

Therefore according to the rule of transitive dependency:

{Company} -> {Age} should hold, that makes sense because if we know the company name, we can know his age.

Note: You need to remember that transitive dependency can only occur in a relation of three or more attributes.

## **Normalization :-**

Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy (repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables. Normalization divides the larger table into the smaller table and links them using relationship.

Normalization is used for mainly two purposes,

- Eliminating redundant (useless) data.
- Ensuring data dependencies make sense i.e data is logically stored.

**Types of Normal Forms** - There are the four types of normal forms:

- 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)

Normal Form	Description
1NF	A relation is in 1NF if it contains an atomic value.
2NF	A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
3NF	A relation will be in 3NF if it is in 2NF and no transitive dependency exists.
4NF	A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
5NF	A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.

### **First Normal Form (1NF)**

- A relation will be 1NF if it contains an atomic value.
- It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.

**Example:** Relation EMPLOYEE is not in 1NF because of multi-valued attribute EMP\_PHONE.

#### **EMPLOYEE table:**

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385, 9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389, 8589830302	Punjab

The decomposition of the EMPLOYEE table into 1NF has been shown below:

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385	UP
14	John	9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389	Punjab
12	Sam	8589830302	Punjab

## **Second Normal Form (2NF)**

- In the 2NF, relational must be in 1NF.
- In the second normal form, all non-key attributes are fully functional dependent on the primary key

**Example:** Let's assume, a school can store the data of teachers and the subjects they teach. In a school, a teacher can teach more than one subject.

### **TEACHER table**

TEACHER_ID	SUBJECT	TEACHER_AGE
25	Chemistry	30
25	Biology	30
47	English	35
83	Math	38
83	Computer	38

In the given table, non-prime attribute TEACHER\_AGE is dependent on TEACHER\_ID which is a proper subset of a candidate key. That's why it violates the rule for 2NF.

To convert the given table into 2NF, we decompose it into two tables:

### **TEACHER\_DETAIL table:**

TEACHER_ID	TEACHER_AGE
25	30
47	35
83	38

### **TEACHER\_SUBJECT table:**

TEACHER_ID	SUBJECT
25	Chemistry
25	Biology
47	English
83	Math
83	Computer

### **Third Normal Form (3NF)**

- A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency.
- 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
- If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.

A relation is in third normal form if it holds at least one of the following conditions for every non-trivial function dependency  $X \rightarrow Y$ .

- X is a super key.
- Y is a prime attribute, i.e., each element of Y is part of some candidate key.

**Example:**      **EMPLOYEE\_DETAIL table:**

EMP_ID	EMP_NAME	EMP_ZIP	EMP_STATE	EMP_CITY
222	Harry	201010	UP	Noida
333	Stephan	02228	US	Boston
444	Lan	60007	US	Chicago
555	Katharine	06389	UK	Norwich
666	John	462007	MP	Bhopal

**Super key in the table above:**

1. {EMP\_ID}, {EMP\_ID, EMP\_NAME}, {EMP\_ID, EMP\_NAME, EMP\_ZIP}....so on

**Candidate key:** {EMP\_ID}

**Non-prime attributes:** In the given table, all attributes except EMP\_ID are non-prime.

Here, EMP\_STATE & EMP\_CITY dependent on EMP\_ZIP and EMP\_ZIP dependent on EMP\_ID. The non-prime attributes (EMP\_STATE, EMP\_CITY) transitively dependent on super key(EMP\_ID). It violates the rule of third normal form.

That's why we need to move the EMP\_CITY and EMP\_STATE to the new <EMPLOYEE\_ZIP> table, with EMP\_ZIP as a Primary key.

**EMPLOYEE table:**

EMP_ID	EMP_NAME	EMP_ZIP
222	Harry	201010
333	Stephan	02228
444	Lan	60007
555	Katharine	06389
666	John	462007

**EMPLOYEE\_ZIP table:**

EMP_ZIP	EMP_STATE	EMP_CITY
201010	UP	Noida
02228	US	Boston
60007	US	Chicago
06389	UK	Norwich
462007	MP	Bhopal

**Boyce Codd normal form (BCNF)**

- BCNF is the advance version of 3NF. It is stricter than 3NF.
- A table is in BCNF if every functional dependency  $X \rightarrow Y$ , X is the super key of the table.
- For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

**Example:** Let's assume there is a company where employees work in more than one department.

**EMPLOYEE table:**

EMP_ID	EMP_COUNTRY	EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
264	India	Designing	D394	283
264	India	Testing	D394	300
364	UK	Stores	D283	232
364	UK	Developing	D283	549

**In the above table Functional dependencies are as follows:**

$EMP\_ID \rightarrow EMP\_COUNTRY$

$EMP\_DEPT \rightarrow \{DEPT\_TYPE, EMP\_DEPT\_NO\}$

**Candidate key: {EMP-ID, EMP-DEPT}**

The table is not in BCNF because neither EMP\_DEPT nor EMP\_ID alone are keys. To convert the given table into BCNF, we decompose it into three tables:

**EMP\_COUNTRY table:**

EMP_ID	EMP_COUNTRY
264	India
264	India

**EMP\_DEPT table:**

EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
Designing	D394	283
Testing	D394	300
Stores	D283	232
Developing	D283	549

**EMP\_DEPT\_MAPPING table:**

EMP_ID	EMP_DEPT
D394	283
D394	300
D283	232
D283	549

**Functional dependencies:**

1. EMP\_ID → EMP\_COUNTRY
2. EMP\_DEPT → {DEPT\_TYPE, EMP\_DEPT\_NO}

**Candidate keys:**

**For the first table:** EMP\_ID

**For the second table:** EMP\_DEPT

**For the third table:** {EMP\_ID, EMP\_DEPT}

Now, this is in BCNF because left side part of both the functional dependencies is a key.



### Fourth normal form (4NF)

- A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
- For a dependency  $A \twoheadrightarrow B$ , if for a single value of A, multiple values of B exists, then the relation will be a multi-valued dependency.

#### Example:

**STUDENT Table :**

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

The given STUDENT table is in 3NF, but the COURSE and HOBBY are two independent entity. Hence, there is no relationship between COURSE and HOBBY.

In the STUDENT relation, a 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, which leads to unnecessary repetition of data.

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

### Fifth normal form (5NF)

- A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.
- 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy.
- 5NF is also known as Project-join normal form (PJ/NF).

### Example:

SUBJECT	LECTURER	SEMESTER
Computer	Anshika	Semester 1
Computer	John	Semester 1
Math	John	Semester 1
Math	Akash	Semester 2
Chemistry	Praveen	Semester 1

In the above table, John takes both Computer and Math class for Semester 1 but he doesn't take Math class for Semester 2. In this case, combination of all these fields required to identify a valid data.

Suppose we add a new Semester as Semester 3 but do not know about the subject and who will be taking that subject so we leave Lecturer and Subject as NULL. But all three columns together acts as a primary key, so we can't leave other two columns blank.

So to make the above table into 5NF, we can decompose it into three relations P1, P2 & P3:

**P1**

SEMESTER	SUBJECT
Semester 1	Computer
Semester 1	Math
Semester 1	Chemistry
Semester 2	Math

**P2**

SUBJECT	LECTURER
Computer	Anshika
Computer	John
Math	John
Math	Akash
Chemistry	Praveen

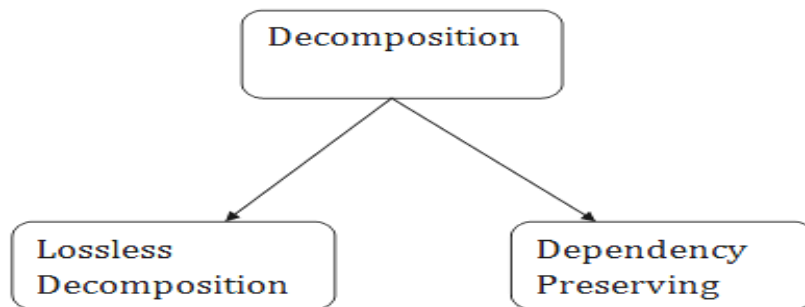
**P3**

SEMSTER	LECTURER
Semester 1	Anshika
Semester 1	John
Semester 1	John
Semester 2	Akash
Semester 1	Praveen

## **Relational Decomposition:-**

- When a relation in the relational model is not in appropriate normal form then the decomposition of a relation is required.
- In a database, it breaks the table into multiple tables.
- If the relation has no proper decomposition, then it may lead to problems like loss of information.
- Decomposition is used to eliminate some of the problems of bad design like anomalies, inconsistencies, and redundancy.

## **Types of Decomposition**



## **Lossless Decomposition:**

- If the information is not lost from the relation that is decomposed, then the decomposition will be lossless.
- The lossless decomposition guarantees that the join of relations will result in the same relation as it was decomposed.
- The relation is said to be lossless decomposition if natural joins of all the decomposition give the original relation.

## **Example:**

### **EMPLOYEE\_DEPARTMENT table:**

EMP_ID	EMP_NAME	EMP_AGE	EMP_CITY	DEPT_ID	DEPT_NAME
22	Denim	28	Mumbai	827	Sales
33	Alina	25	Delhi	438	Marketing
46	Stephan	30	Bangalore	869	Finance
52	Katherine	36	Mumbai	575	Production
60	Jack	40	Noida	678	Testing

The above relation is decomposed into two relations EMPLOYEE and DEPARTMENT

**EMPLOYEE table:**

EMP_ID	EMP_NAME	EMP_AGE	EMP_CITY
22	Denim	28	Mumbai
33	Alina	25	Delhi
46	Stephan	30	Bangalore
52	Katherine	36	Mumbai
60	Jack	40	Noida

**DEPARTMENT table**

DEPT_ID	EMP_ID	DEPT_NAME
827	22	Sales
438	33	Marketing
869	46	Finance
575	52	Production
678	60	Testing

Now, when these two relations are joined on the common column "EMP\_ID", then the resultant relation will look like:

**Employee ⋈ Department**

EMP_ID	EMP_NAME	EMP_AGE	EMP_CITY	DEPT_ID	DEPT_NAME
22	Denim	28	Mumbai	827	Sales
33	Alina	25	Delhi	438	Marketing
46	Stephan	30	Bangalore	869	Finance
52	Katherine	36	Mumbai	575	Production
60	Jack	40	Noida	678	Testing

Hence, the decomposition is Lossless join decomposition.

### **Dependency Preserving:**

- It is an important constraint of the database.
- In the dependency preservation, at least one decomposed table must satisfy every dependency.
- If a relation R is decomposed into relation R1 and R2, then the dependencies of R either must be a part of R1 or R2 or must be derivable from the combination of functional dependencies of R1 and R2.
- For example, suppose there is a relation R (A, B, C, D) with functional dependency set (A->BC). The relational R is decomposed into R1(ABC) and R2(AD) which is dependency preserving because FD A->BC is a part of relation R1(ABC).

### **Multivalued Dependency:-**

- Multivalued dependency occurs when two attributes in a table are independent of each other but, both depend on a third attribute.
- A multivalued dependency consists of at least two attributes that are dependent on a third attribute that's why it always requires at least three attributes.

**Example:** Suppose there is a bike manufacturer company which produces two colors(white and black) of each model every year.

BIKE_MODEL	MANUF_YEAR	COLOR
M2011	2008	White
M2001	2008	Black
M3001	2013	White
M3001	2013	Black
M4006	2017	White
M4006	2017	Black

Here columns COLOR and MANUF\_YEAR are dependent on BIKE\_MODEL and independent of each other.

In this case, these two columns can be called as multivalued dependent on BIKE\_MODEL. The representation of these dependencies is shown below:

BIKE\_MODEL → → MANUF\_YEAR  
BIKE\_MODEL → → COLOR

This can be read as "BIKE\_MODEL multidetermined MANUF\_YEAR" and "BIKE\_MODEL multidetermined COLOR".

## **Join Dependency:-**

Join decomposition is a further generalization of Multivalued dependencies.

If the join of  $R_1$  and  $R_2$  over  $C$  is equal to relation  $R$ , then we can say that a join dependency (JD) exists.

Where  $R_1$  and  $R_2$  are the decompositions  $R_1(A, B, C)$  and  $R_2(C, D)$  of a given relations  $R(A, B, C, D)$ .

Alternatively,  $R_1$  and  $R_2$  are a lossless decomposition of  $R$ .

- A JD  $\bowtie \{R_1, R_2, \dots, R_n\}$  is said to hold over a relation  $R$  if  $R_1, R_2, \dots, R_n$  is a lossless-join decomposition.
- The  $\bowtie(A, B, C, D), (C, D)$  will be a JD of  $R$  if the join of join's attribute is equal to the relation  $R$ .
- Here,  $\bowtie(R_1, R_2, R_3)$  is used to indicate that relation  $R_1, R_2, R_3$  and so on are a JD of  $R$ .