Types of Functional dependencies in DBMS

Difficulty Level : Easy • Last Updated : 21 Aug, 2021

Prerequisite: Functional dependency and attribute closure

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**. Functional dependencies are used to mathematically express relations among database entities and are very important to understand advanced concepts in Relational Database System and understanding problems in competitive exams like Gate.

Example:

roll_no	name	dept_name	dept_building
42	abc	CO	A4
43	pqr	IT	A3
44	xyz	CO	A4
45	xyz	IT	A3

47	jkl	ME	B2

From the above table we can conclude some valid functional dependencies:

- roll_no → { name, dept_name, dept_building }, → Here, roll_no can determine values of fields name, dept_name and dept_building, hence a valid Functional dependency
- roll_no → dept_name , Since, roll_no can determine whole set of {name, dept_name, dept_building}, it can determine its subset dept_name also.
- dept_name → dept_building , Dept_name can identify the dept_building accurately, since departments with different dept_name will also have a different dept_building
- More valid functional dependencies: roll_no → name, {roll_no, name} ⇢ {dept_name, dept_building}, etc.

Here are some invalid functional dependencies:

- name → dept_name Students with the same name can have different dept_name, hence this is not a valid functional dependency.
- dept_building → dept_name There can be multiple departments in the same building, For example, in the above table departments ME and EC are in the same building B2, hence dept_building → dept_name is an invalid functional dependency.
- More invalid functional dependencies: name → roll_no, {name, dept_name} → roll_no, dept_building → roll_no, etc.

Armstrong's axioms/properties of functional dependencies:

1. **Reflexivity:** If Y is a subset of X, then X→Y holds by reflexivity rule
   For example, {roll_no, name} → name is valid.
2. **Augmentation:** If X → Y is a valid dependency, then XZ → YZ is also valid by the augmentation rule.
   For example, If {roll_no, name} → dept_building is valid, hence {roll_no, name, dept_name} → {dept_building, dept_name} is also valid. →

Types of Functional dependencies in DBMS:

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

1. Trivial Functional Dependency

In **Trivial Functional Dependency**, a dependent is always a subset of the determinant.
i.e. If **X → Y** and **Y is the subset of X**, then it is called trivial functional dependency

For example,

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

Here, **{roll_no, name} → name** is a trivial functional dependency, since the dependent **name** is a subset of determinant set **{roll_no, name}**
Similarly, **roll_no → roll_no** is also an example of trivial functional dependency.

2. Non-trivial Functional Dependency

In **Non-trivial functional dependency**, the dependent is strictly not a subset of the determinant.
i.e. If **X → Y** and **Y is not a subset of X**, then it is called Non-trivial functional dependency.

For example,

43	pqr	18
44	xyz	18

Here, **roll_no → name** is a non-trivial functional dependency, since the dependent **name** is **not a subset of** determinant **roll_no**
Similarly, **{roll_no, name} → age** is also a non-trivial functional dependency, since **age** is **not a subset of {roll_no, name}**

3. Multivalued Functional Dependency

In **Multivalued functional dependency**, entities of the dependent set are **not dependent on each other.**
i.e. If **a → {b, c}** and there exists **no functional dependency** between **b and c**, then it is called a **multivalued functional dependency.**

For example,

roll_no	name	age
42	abc	17
43	pqr	18
44	xyz	18
45	abc	19

Here, **roll_no → {name, age}** is a multivalued functional dependency, since the dependents **name** & **age** are **not dependent** on each other(i.e. **name → age** or **age → me doesn't exist !**)

4. Transitive Functional Dependency

For example,

enrol_no	name	dept	building_no
42	abc	CO	4
43	pqr	EC	2
44	xyz	IT	1
45	abc	EC	2

Here, **enrol_no → dept** and **dept → building_no**,
Hence, according to the axiom of transitivity, **enrol_no → building_no** is a valid functional dependency. This is an indirect functional dependency, hence called Transitive functional dependency.

RECOMMENDED ARTICLES

Page : 1 2 3

01 Canonical Cover of Functional Dependencies in DBMS
22, May 17

02 Finding Attribute Closure and Candidate Keys using Functional Dependencies
18, Nov 15

03 Equivalence of Functional Dependencies
14, Dec 15

04 Finding Additional functional dependencies in a relation
01, May 20

05 Finding the candidate keys for Sub relations using Functional Dependencies
01, May 20

06 Allowed Functional Dependencies (FD) in Various Normal Forms (NF)
06, May 20

07 Armstrong's Axioms in Functional Dependency in DBMS
11, Oct 17

08 Functional Completeness in Digital Logic
31, Oct 17

Guru99

Functional Dependency in DBMS: What is, Types and Examples

By Richard Peterson Updated February 19, 2022

What is Functional Dependency?

Functional Dependency (FD) is a constraint that determines the relation of one attribute to another attribute in a Database Management System (DBMS). Functional Dependency helps to maintain the quality of data in the database. It plays a vital role to find the difference between good and bad database design.

A functional dependency is denoted by an arrow " → ". The functional dependency of X on Y is represented by X → Y. Let's understand Functional Dependency in DBMS with example.

Example:

Employee number	Employee Name	Salary	City
1	Dana	50000	San Francisco
2	Francis	38000	London
3	Andrew	25000	Tokyo

In this example, if we know the value of Employee number, we can obtain Employee Name, city, salary, etc. By this, we can say that the city, Employee Name, and salary are functionally depended on Employee number.

In this tutorial, you will learn:

- Transitive dependency in DBMS
- What is Normalization?
- Advantages of Functional Dependency

Key terms

Here, are some key terms for Functional Dependency in Database:

Key Terms	Description
Axiom	Axioms is a set of inference rules used to infer all the functional dependencies on a relational database.
Decomposition	It is a rule that suggests if you have a table that appears to contain two entities which are determined by the same primary key then you should consider breaking them up into two different tables.
Dependent	It is displayed on the right side of the functional dependency diagram.
Determinant	It is displayed on the left side of the functional dependency Diagram.
Union	It suggests that if two tables are separate, and the PK is the same, you should consider putting them. together

Rules of Functional Dependencies

Below are the Three most important rules for Functional Dependency in Database:

- Reflexive rule –. If X is a set of attributes and Y is_subset_of X, then X holds a value of Y.
- Augmentation rule: When x -> y holds, and c is attribute set, then ac -> bc also holds. That is adding attributes which do not change the basic dependencies.
- Transitivity rule: This rule is very much similar to the transitive rule in algebra if x -> y holds and y -> z holds, then x -> z also holds. X -> y is called as functionally that determines y.

Types of Functional Dependencies in DBMS

- **Multivalued Dependency**
- **Trivial Functional Dependency**
- **Non-Trivial Functional Dependency**
- **Transitive Dependency**

Multivalued Dependency in DBMS

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. Consider the following Multivalued Dependency Example to understand.

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

Trivial Functional Dependency in DBMS

The Trivial dependency is a set of attributes which are called a trivial if the set of attributes are included in that attribute.

So, X -> Y is a trivial functional dependency if Y is a subset of X. Let's understand with a Trivial Functional Dependency Example.

For example:

Emp_id	Emp_name
AS555	Harry
AS811	George
AS999	Kevin

Consider this table of with two columns Emp_id and Emp_name.

{Emp_id, Emp_name} -> Emp_id is a trivial functional dependency as Emp_id is a subset of {Emp_id,Emp_name}.

Non Trivial Functional Dependency in DBMS

Functional dependency which also known as a nontrivial dependency occurs when A->B holds true where B is not a subset of A. In a relationship, if attribute B is not a subset of attribute A, then it is considered as a non-trivial dependency.

Company	CEO	Age
Microsoft	Satya Nadella	51
Google	Sundar Pichai	46
Apple	Tim Cook	57

Example:

(Company} -> {CEO} (if we know the Company, we knows the CEO name)

Transitive Dependency in DBMS

A Transitive Dependency is a type of functional dependency which happens when “t” is indirectly formed by two functional dependencies. Let’s understand with the following Transitive Dependency Example.

Example:

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

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

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

Therefore according to the rule of 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.

What is Normalization?

Normalization is a method of organizing the data in the database which helps you to avoid data redundancy, insertion, update & deletion anomaly. It is a process of analyzing the relation schemas based on their different functional dependencies and primary key.

Normalization is inherent to relational database theory. It may have the effect of duplicating the same data within the database which may result in the creation of additional tables.

- Functional Dependency avoids data redundancy. Therefore same data do not repeat at multiple locations in that database
- It helps you to maintain the quality of data in the database
- It helps you to defined meanings and constraints of databases
- It helps you to identify bad designs
- It helps you to find the facts regarding the database design

Summary

- Functional Dependency is when one attribute determines another attribute in a DBMS system.
- Axiom, Decomposition, Dependent, Determinant, Union are key terms for functional dependency
- Four types of functional dependency are 1) Multivalued 2) Trivial 3) Non-trivial 4) Transitive
- Multivalued dependency occurs in the situation where there are multiple independent multivalued attributes in a single table
- The Trivial dependency occurs when a set of attributes which are called a trivial if the set of attributes are included in that attribute
- Nontrivial dependency occurs when A->B holds true where B is not a subset of A
- A transitive is a type of functional dependency which happens when it is indirectly formed by two functional dependencies
- Normalization is a method of organizing the data in the database which helps you to avoid data redundancy

You Might Like:

- DBMS Tutorial: Database Management System Notes
- Top 50 Database (DBMS) Interview Questions & Answers (2022)
- Indexing in DBMS: What is, Types of Indexes with EXAMPLES
- Difference Between DDL and DML Command in DBMS: What is?
- File System vs DBMS: Key Differences

Functional dependency in DBMS

What is Functional Dependency

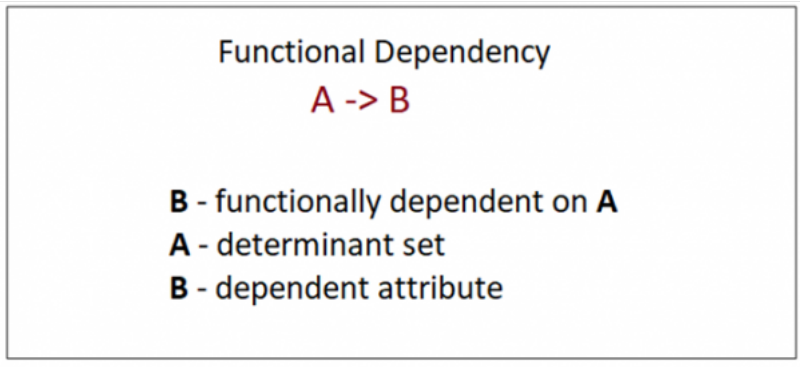
Functional dependency in DBMS, as the name suggests is a relationship between attributes of a table dependent on each other. Introduced by E. F. Codd, it helps in preventing data redundancy and gets to know about bad designs.

To understand the concept thoroughly, let us consider P is a relation with attributes A and B. Functional Dependency is represented by -> (arrow sign)

Then the following will represent the functional dependency between attributes with an arrow sign −

```
A -> B
```

Above suggests the following:



Example

The following is an example that would make it easier to understand functional dependency −

We have a **<Department>** table with two attributes − **DeptId** and **DeptName**.

```
DeptId = Department ID
DeptName = Department Name
```

The **DeptId** is our primary key. Here, **DeptId** uniquely identifies the **DeptName** attribute. This is because if you want to know the department name, then at first you need to have the **DeptId**.

DeptId	DeptName
001	Finance
002	Marketing
003	HR

Therefore, the above functional dependency between **DeptId** and **DeptName** can be determined as **DeptId** is functionally dependent on **DeptName** −

```
DeptId -> DeptName
```

Types of Functional Dependency

Functional Dependency has three forms −

- Trivial Functional Dependency
- Non-Trivial Functional Dependency
- Completely Non-Trivial Functional Dependency

Let us begin with Trivial Functional Dependency −

Trivial Functional Dependency

It occurs when B is a subset of A in −

```
A ->B
```

Example

We are considering the same **<Department>** table with two attributes to understand the concept of trivial dependency.

The following is a trivial functional dependency since **DeptId** is a subset of **DeptId** and **DeptName**

```
{ DeptId,  DeptName } -> Dept Id
```

Non –Trivial Functional Dependency

It occurs when B is not a subset of A in −

```
A ->B
```

Example

```
DeptId ->  DeptName
```

The above is a non-trivial functional dependency since DeptName is a not a subset of DeptId.

Completely Non - Trivial Functional Dependency

It occurs when A intersection B is null in −

```
A ->B
```

Armstrong’s Axioms Property of Functional Dependency

Armstrong’s Axioms property was developed by William Armstrong in 1974 to reason about functional dependencies.

The property suggests rules that hold true if the following are satisfied:

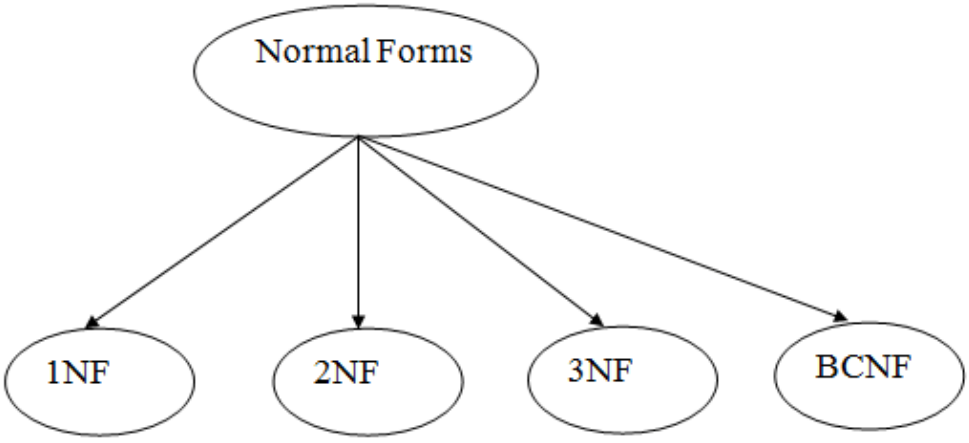
- **Transitivity**
  If A->B and B->C, then A->C i.e. a transitive relation.
- **Reflexivity**
  A-> B, if B is a subset of A.
- **Augmentation**
  The last rule suggests: AC->BC, if A->B

Normalization

- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies.
- Normalization divides the larger table into the smaller table and links them using relationship.
- The normal form is used to reduce redundancy from the database table.

Types of Normal Forms

There are the four types of normal forms:



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 transition 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.

Inference Rule (IR):

- The Armstrong's axioms are the basic inference rule.
- Armstrong's axioms are used to conclude functional dependencies on a relational database.
- The inference rule is a type of assertion. It can apply to a set of FD(functional dependency) to derive other FD.
- Using the inference rule, we can derive additional functional dependency from the initial set.

The Functional dependency has 6 types of inference rule:

1. Reflexive Rule (IR_1)

In the reflexive rule, if Y is a subset of X, then X determines Y.

> If $X \supseteq Y$ then $X \rightarrow Y$

Example:

> X = {a, b, c, d, e}
> Y = {a, b, c}

2. Augmentation Rule (IR_2)

The augmentation is also called as a partial dependency. In augmentation, if X determines Y, then XZ determines YZ for any Z.

> If $X \rightarrow Y$ then $XZ \rightarrow YZ$

Example:

> For R(ABCD), if $A \rightarrow B$ then $AC \rightarrow BC$

3. Transitive Rule (IR_3)

In the transitive rule, if X determines Y and Y determine Z, then X must also determine Z.

> If $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$

4. Union Rule (IR_4)

Union rule says, if X determines Y and X determines Z, then X must also determine Y and Z.

> If X → Y and X → Z then X → YZ

Proof:

> 1. X → Y (given)
> 2. X → Z (given)
> 3. X → XY (using IR_2 on 1 by augmentation with X. Where XX = X)
> 4. XY → YZ (using IR_2 on 2 by augmentation with Y)
> 5. X → YZ (using IR_3 on 3 and 4)

5. Decomposition Rule (IR_5)

Decomposition rule is also known as project rule. It is the reverse of union rule.

This Rule says, if X determines Y and Z, then X determines Y and X determines Z separately.

> If X → YZ then X → Y and X → Z

Proof:

> 1. X → YZ (given)
> 2. YZ → Y (using IR_1 Rule)
> 3. X → Y (using IR_3 on 1 and 2)

6. Pseudo transitive Rule (IR_6)

In Pseudo transitive Rule, if X determines Y and YZ determines W, then XZ determines W.

> If X → Y and YZ → W then XZ → W

Proof:

> 1. X → Y (given)
> 2. WY → Z (given)
> 3. WX → WY (using IR_2 on 1 by augmenting with W)
> 4. WX → Z (using IR_3 on 3 and 2)

Functional Dependency

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.

> X → Y

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

For example:

Assume we have an employee table with attributes: Emp_Id, Emp_Name, Emp_Address.

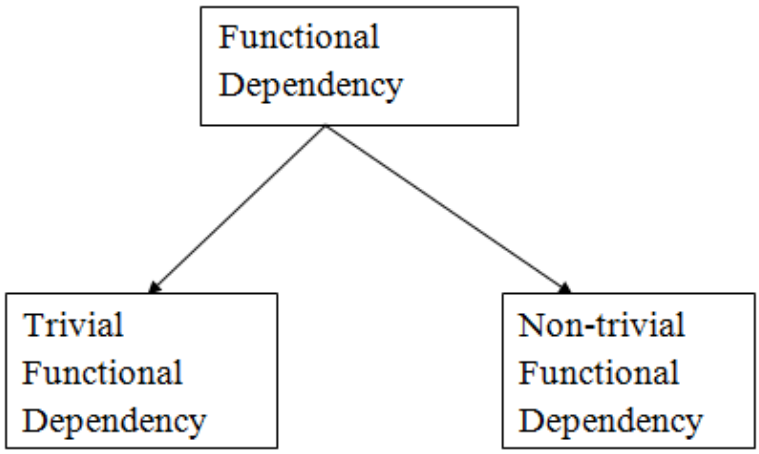
Here Emp_Id attribute can uniquely identify the Emp_Name attribute of employee table because if we know the Emp_Id, we can tell that employee name associated with it.

Functional dependency can be written as:

> Emp_Id → Emp_Name

We can say that Emp_Name is functionally dependent on Emp_Id.

Types of Functional dependency



1. Trivial functional dependency

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

Example:

> Consider a table with two columns Employee_Id and Employee_Name.
> {Employee_id, Employee_Name} → Employee_Id is a trivial functional dependency as Employee_Id is a subset of {Employee_Id, Employee_Name}.
> Also, Employee_Id → Employee_Id and Employee_Name → Employee_Name are trivial depenc

2. Non-trivial functional dependency

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

Example:

> ID → Name,
> Name → DOB

← Prev Next →

Feedback

- Send your Feedback to feedback@javatpoint.com