**Types of Functional dependencies in DBMS**

 [**Functional dependency and attribute closure**](https://www.geeksforgeeks.org/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 .If the information stored in a table can uniquely determine another information in the same table, then it is called Functional Dependency. Consider it as an association between two attributes of the same relation.

If P functionally determines Q, then

|  |
| --- |
| **P -> Q** |

Let us see an example −

**<Employee>**

|  |  |  |
| --- | --- | --- |
| **EmpID** | **EmpName** | **EmpAge** |
| E01 | Amit | 28 |
| E02 | Rohit | 31 |

In the above table, **EmpName**is functionally dependent on **EmpID**because **EmpName**can take only one value for the given value of **EmpID:**

|  |
| --- |
| **EmpID -> EmpName** |

The same is displayed below −



**Example:**

| 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 |

**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:**

1. name → dept\_name   Students with the same name can have different dept\_name, hence this is not a valid functional dependency.

2 .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.

3.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.→
3. **Transitivity**: If X → Y and Y → Z are both valid dependencies, then X→Z is also valid by the Transitivity rule.
For example, roll\_no → dept\_name & dept\_name → dept\_building, then roll\_no → dept\_building 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,**

| **roll\_no** | name | age |
| --- | --- | --- |
| 42 | abc | 17 |
| 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 example1**

| 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 → name doesn’t exist !**)

 Actually,when existence of one or more rows in a table implies one or more other rows in the same table, then the **Multi-valued dependencies occur.**

If a table has attributes P, Q and R, then Q and R are multi-valued facts of P.

It is represented by double arrow −

|  |
| --- |
| **->->** |

For our example:

|  |
| --- |
| **P->->QQ->->R** |

In the above case, Multivalued Dependency exists only if Q and R are independent attribute.

 Multivalued dependency occurs when there are more than one independent multivalued attributes in a table

**Example2:**

Consider a bike manufacture company which produces two colours(black and white)in each model every year:

|  |  |  |
| --- | --- | --- |
| **Bike\_model** | **Manuf\_year** | **color** |
| M1001 | 2007 | black |
| M1001 | 2007 | red |
| M2012 | 2008 | black |
| M2012 | 2008 | red |
| M2222 | 2009 | black |
| M2222 | 2009 | red |

Here,columns **manuf\_year** and **color** are independent of each other and dependent on **bike\_model**.In this case this two columns are said to be multivalued dependent on **bike\_model**

 **This dependencies can be represented like this.**

**Bike\_model->->manuf\_year**

**Bike\_model->->color**

**4. Transitive Functional Dependency**

In transitive functional dependency, dependent is indirectly dependent on determinant.

i.e. If **a → b** & **b → c**, then according to axiom of transitivity, **a → c**. This is a **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.

 A transitive dependency can only occur in a relation of three or more attributes.This dependency helps us normalizing the database in 3NF(third normal form)

Let us take another example:

Example:

|  |  |  |
| --- | --- | --- |
| **book** | **author** | **author\_age** |
| Game of thrones | George R.R.Martin | 66 |
| Harry potter | J.K Rowling | 49 |
| Dying of the light | George R.R.Martin | 66 |

**Book🡪author** ( if we know the book,we know the author name)

**Author🡪author\_age**

 Therefore as per the rule of transitive dependency

**Book🡪author\_age** should hold,that makes sense because if we know the book name then we can the author’s age

## Fully-functionally Dependency

An attribute is fully functional dependent on another attribute, if it is Functionally Dependent on that attribute and not on any of its proper subset.

For example, an attribute Q is fully functional dependent on another attribute P, if it is Functionally Dependent on P and not on any of the proper subset of P.

Let us see an example −

**<ProjectCost>**

|  |  |
| --- | --- |
| **ProjectID** | **ProjectCost** |
| 001 | 1000 |
| 002 | 5000 |

**<EmployeeProject>**

|  |  |  |
| --- | --- | --- |
| **EmpID** | **ProjectID** | **Days**(spent on the project) |
| E099 | 001 | 320 |
| E056 | 002 | 190 |

The above relations states:

|  |
| --- |
| **EmpID, ProjectID, ProjectCost -> Days** |

However, it is not fully functional dependent.

Whereas the subset **{EmpID, ProjectID}** can easily determine the**{Days}** spent on the project by the employee.

This summarizes and gives our fully functional dependency −

|  |
| --- |
| **{EmpID, ProjectID}  -> (Days)** |

## Partial Dependency

Partial Dependency occurs when a **nonprime attribute** is functionally dependent on part of a candidate key.

**The** **2nd Normal Form (2NF) eliminates the Partial Dependency**. Let us see an example −

**<StudentProject>**

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

In the above table, we have partial dependency; let us see how −

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

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.

The **StudentName**can be determined by **StudentID**that makes the relation Partial Dependent. **StudentID🡪studentname**

The **ProjectName**can be determined by **ProjectID**, which that the relation Partial Dependent. **ProjectNo🡪projectname**