**Functional Dependency**

The attributes of a table is said to be dependent on each other when an attribute of a table uniquely identifies another attribute of the same table.

For example: Suppose we have a student table with attributes: Stu\_Id, Stu\_Name, Stu\_Age. Here Stu\_Id attribute uniquely identifies the Stu\_Name attribute of student table because if we know the student id we can tell the student name associated with it. This is known as functional dependency and can be written as Stu\_Id->Stu\_Name or in words we can say Stu\_Name is functionally dependent on Stu\_Id.

**Formally**:  
If column A of a table uniquely identifies the column B of same table then it can represented as A->B (Attribute B is functionally dependent on attribute A)

**Types of Functional Dependencies**

* [**Trivial functional dependency**](http://beginnersbook.com/2015/04/trivial-functional-dependency-in-dbms/)
* [**non-trivial functional dependency**](http://beginnersbook.com/2015/04/non-trivial-functional-dependency-in-dbms/)
* [**Multivalued dependency**](http://beginnersbook.com/2015/04/multivalued-dependency-in-dbms/)
* [**Transitive dependency**](http://beginnersbook.com/2015/04/transitive-dependency-in-dbms/)

# Trivial functional dependency in DBMS with example

The dependency of an attribute on a set of attributes is known as trivial functional dependency if the set of attributes includes that attribute.

**Symbolically**: A ->B is trivial functional dependency if B is a subset of A.

The following dependencies are also trivial: A->A & B->B

**For example**: Consider a table with two columns Student\_id and Student\_Name.

{Student\_Id, Student\_Name} -> Student\_Id is a trivial functional dependency as Student\_Id is a subset of {Student\_Id, Student\_Name}.  That makes sense because if we know the values of Student\_Id and Student\_Name then the value of Student\_Id can be uniquely determined.

Also, Student\_Id -> Student\_Id & Student\_Name -> Student\_Name are trivial dependencies too.

# Non trivial functional dependency in DBMS

If a functional dependency X->Y holds true where Y is not a subset of X then this dependency is called non trivial Functional dependency.

**For example**:  
An employee table with three attributes: emp\_id, emp\_name, emp\_address.  
The following functional dependencies are non-trivial:  
emp\_id -> emp\_name (emp\_name is not a subset of emp\_id)  
emp\_id -> emp\_address (emp\_address is not a subset of emp\_id)

On the other hand, the following dependencies are trivial:  
{emp\_id, emp\_name} -> emp\_name [emp\_name is a subset of {emp\_id, emp\_name}]

**Completely non trivial FD**:  
If a FD X->Y holds true where X intersection Y is null then this dependency is said to be completely non trivial function dependency.

# Multivalued dependency in DBMS

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

**For example**: Consider a bike manufacture company, which produces two colors (Black and red) 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 these two columns are said to be multivalued dependent on bike\_model. These dependencies can be represented like this:

bike\_model ->> manuf\_year

bike\_model ->> color

# Transitive dependency in DBMS

A functional dependency is said to be transitive if it is indirectly formed by two functional dependencies. For e.g.

X -> Z is a transitive dependency if the following three functional dependencies hold true:

* X->Y
* Y does not ->X
* Y->Z

**Note:** A transitive dependency can only occur in a relation of three of more attributes. This dependency helps us normalizing the database in 3NF (3rd Normal Form).

**Example**: Let’s take an example to understand it better:

|  |  |  |
| --- | --- | --- |
| **Book x** | **Author y** | **Author\_age z** |
| Game of Thrones | George R. R. Martin | 66 |
| Harry Potter | J. K. Rowling | 49 |
| Dying of the Light | George R. R. Martin | 66 |

* X->Y
* Y does not ->X
* Y->Z  
  {Book} ->{Author} (if we know the book, we knows the author name)

{Author} does not ->{Book}

{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 we can know the author’s age.