Functional dependency in DBMS

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)

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}]

Refer: **trivial functional dependency**.

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.

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.

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 (3rdNormal Form).

Example: Let's take an example to understand it better:

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 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.

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 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 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