# For
## SEMESTER-II
### Paper II : Database Management Systems(CODE: CSC-RC-2016)
**Topics: Functional dependencies, Normalization and Normal forms up to 3NF**

---

## Functional Dependencies

A *functional dependency* (FD) is a relationship between two attributes, typically between the primary key and other non-key attributes within a table. A functional dependency denoted by

**X → Y** , is an association between two sets of attribute X and Y. Here, **X** is called the *determinant*, and Y is called the *dependent*.

For example,
SIN ————-> Name, Address, Birthdate

Here, SIN determines Name, Address and Birthdate.So, SIN is the determinant and Name, Address and Birthdate are the dependents.

### Rules of Functional Dependencies

1. Reflexive rule : If Y is a subset of X, then X determines Y .
   $$\text{If } Y \subseteq X, \text{ then } X \rightarrow Y$$
2. Augmentation rule: It is also known as a partial dependency, says if X determines Y, then XZ determines YZ for any Z
   $$\text{If } X \rightarrow Y, \text{ then } XZ \rightarrow YZ \text{ for any } Z$$
3. Transitivity rule: Transitivity says if X determines Y, and Y determines Z, then X must also determine Z
   $$\text{If } X \rightarrow Y \text{ and } Y \rightarrow Z, \text{ then } X \rightarrow Z$$

### Types of functional dependency

The following are types functional dependency in DBMS:

1. Fully-Functional Dependency
2. Partial Dependency
3. Transitive Dependency
4. Trivial Dependency
5. Multivalued Dependency

**Full functional Dependency**

A functional dependency X→Y is said to be a full functional dependency, if removal of any attribute A from X, the dependency does not hold any more. i.e. Y is ***fully functional*** dependent on X, if it is ***Functionally*** Dependent on X and not on any of the proper subset of X.
For example,

{Emp_num,Proj_num} → Hour
Is a full functional dependency. Here, *Hour* is the working time by an employee in a project.

**Partial functional Dependency**

A functional dependency X→Y is said to be a partial functional dependency, if after removal of any attribute A from X, the dependency does not holds. i.e. Y is dependent on a proper subset of X. So X is partially dependent on X.
For example,
If {Emp_num,Proj_num} → Emp_name but also Emp_num → Emp_name then Emp_name is partially functionally dependent on {Empl_num,Proj_num}.

**Transitive dependency**
A functional dependency is X → Z is said to be a transitive functional dependency if there exists the functional dependencies X → Y and Y → Z. i.e. it is an indirect relationship.
For example,

EMP_NUM → JOB_CLASS

is a transitive dependency which comes from EMP_NUM → JOB_CLASS and
JOB_CLASS → CHG_HOUR

Trivial functional dependency

A functional dependency X → Y is said to be a trivial functional dependency if Y is a subset of X.

For example,

{Emp_num,Emp_name} → Emp_num
is a trivial functional dependency since Emp_num is a subset of {Emp_num,Emp_name}.

**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: Consider the following table

| Car_model | Manufr_year | Color |
|---|---|---|
| H001 | 2017 | Metallic |
| H001 | 2017 | Green |
| H005 | 2018 | Metallic |
| H005 | 2018 | Blue |
| H010 | 2015 | Metallic |
| H033 | 2012 | Gray |

The functional dependencies
car_model -> manufr_year
car_model-> colour
are multivalued dependency since manufr_year and color both are multivalued attribute

## Normalization

Normalization is the process of decomposing the relations into well structured relations to organize the data in the database to remove redundancy of data, insertion anomaly, update anomaly and deletion anomaly.

**Insertion anomaly**

If a tuple is inserted in referencing relation and referencing attribute value is not present in referenced attribute, it will not allow inserting in referencing relation.
For Example,

**STUDENT**

| STUD_NO | STUD_NAME | STUD_PHONE | STUD_STATE | STUD_COUNTRY | STUD_AGE |
|---|---|---|---|---|---|
| 1 | RAM | 9716271721 | Haryana | India | 20 |
| 2 | RAM | 9898291281 | Punjab | India | 19 |
| 3 | SUJIT | 7898291981 | Rajsthan | India | 18 |
| 4 | SURESH | | Punjab | India | 21 |

**Table 1**

**STUDENT_COURSE**

| STUD_NO | COURSE_NO | COURSE_NAME |
|---|---|---|
| 1 | C1 | DBMS |
| 2 | C2 | Computer Networks |
| 1 | C2 | Computer Networks |

**Table 2**

Here, if we try to insert a record in STUDENT_COURSE with STUD_NO =7, it will not allow.

**Deletion and Updation anomaly**

If a tuple is deleted or updated from referenced relation and referenced attribute value is used by referencing attribute in referencing relation, it will not allow deleting the tuple from referenced relation. For Example,

**STUDENT**

| STUD_NO | STUD_NAME | STUD_PHONE | STUD_STATE | STUD_COUNTRY | STUD_AGE |
|---|---|---|---|---|---|
| 1 | RAM | 9716271721 | Haryana | India | 20 |
| 2 | RAM | 9898291281 | Punjab | India | 19 |
| 3 | SUJIT | 7898291981 | Rajsthan | India | 18 |
| 4 | SURESH | | Punjab | India | 21 |

**Table 1**

**STUDENT_COURSE**

| STUD_NO | COURSE_NO | COURSE_NAME |
|---|---|---|
| 1 | C1 | DBMS |
| 2 | C2 | Computer Networks |
| 1 | C2 | Computer Networks |

**Table 2**

Here, if we try to delete a record from STUDENT with STUD_NO =1, it will not allow.

The normal forms used for normalization are:

1. First normal form(1NF)
2. Second normal form(2NF)
3. Third normal form(3NF)

4. Boyce & Codd normal form (BCNF)
5. Fourth normal form (4NF)
6. Fifth normal form (5NF)

**First normal form (1NF)**

The first normal form is based on the simple or atomic attribute and single valued attribute. A relation is said to be in 1NF if all the attributes of the relation are atomic and single valued.

Example-

Suppose a company wants to store the names and contact details of its employees. It creates a table that looks like this:

**Employee**

| **Emp_id** | **Emp_name** | **Emp_address** | **Emp_mobile** |
|---|---|---|---|
| 101 | Herschel | New Delhi | 8912312390 |
| 102 | Jon | Kanpur | 8812121212<br>9900012222 |
| 103 | Ron | Chennai | 7778881212 |
| 104 | Lester | Bangalore | 9990000123<br>8123450987 |

Here, the Relation employee is not in 1NF, because employees with employee id 102 and 104 are having two phone numbers. i.e. the Emp_mobile attribute is a multi valued attribute.

After normalization to 1NF the relation will be like this:

| **Emp_id** | **Emp_name** | **Emp_address** | **Emp_mobile** |
|---|---|---|---|
| 101 | Herschel | New Delhi | 8912312390 |
| 102 | Jon | Kanpur | 8812121212 |
| 102 | Jon | Kanpur | 9900012222 |
| 103 | Ron | Chennai | 7778881212 |
| 104 | Lester | Bangalore | 9990000123 |
| 104 | Lester | Bangalore | 8123450987 |

**Second normal form (2NF)**

The second normal form is based on the full functional dependency. A relation is said to be in 2NF if it is first in 1NF and all non-key attributes are fully functional dependent on the primary key.. i.e. no partial dependency exists.

Example-

Consider the following example:

**TABLE_PURCHASE_DETAIL**

| Customer ID | Store ID | Purchase Location |
|---|---|---|
| 1 | 1 | Los Angeles |
| 1 | 3 | San Francisco |
| 2 | 1 | Los Angeles |
| 3 | 2 | New York |
| 4 | 3 | San Francisco |

This table has a composite primary key {Customer ID, Store ID}. The non-key attribute is Purchase Location. In this case, Purchase Location only depends on Store ID, which is a part of the primary key. Therefore, this table does not satisfy second normal form.

To bring this table to second normal form, we have to break the table into two tables as shown below.

**TABLE_PURCHASE**

| Customer ID | Store ID |
|---|---|
| 1 | 1 |
| 1 | 3 |
| 2 | 1 |
| 3 | 2 |
| 4 | 3 |

**TABLE_STORE**

| Store ID | Purchase Location |
|---|---|
| 1 | Los Angeles |
| 2 | New York |
| 3 | San Francisco |

Now, in the table TABLE_STORE, the column Purchase Location is fully dependent on the primary key of that table, which is Store ID.

**Third normal form (3NF)**

The third normal form is based on the transitive dependency. A relation is said to be in 3NF if it is first in 2NF and no transitive functional dependency exists.

Example-

Consider the following table:

**TABLE_BOOK_DETAIL**

| Book ID | Genre ID | Genre Type | Price |
|---|---|---|---|
| 1 | 1 | Gardening | 25.99 |
| 2 | 2 | Sports | 14.99 |
| 3 | 1 | Gardening | 10.00 |
| 4 | 3 | Travel | 12.99 |
| 5 | 2 | Sports | 17.99 |

In the above table, Book ID determines Genre ID, and Genre ID determines Genre Type. Therefore, Book ID determines Genre Type via Genre ID and we have transitive functional dependency, and this structure does not satisfy third normal form.

To bring this table to 3NF, we split the table into two as follows:

**TABLE_BOOK**

| Book ID | Genre ID | Price |
|---|---|---|
| 1 | 1 | 25.99 |
| 2 | 2 | 14.99 |
| 3 | 1 | 10.00 |
| 4 | 3 | 12.99 |
| 5 | 2 | 17.99 |

**TABLE_GENRE**

| Genre ID | Genre Type |
|---|---|
| 1 | Gardening |
| 2 | Sports |
| 3 | Travel |

Now all non-key attributes are fully functional dependent only on the primary key. In TABLE_BOOK, both Genre ID and Price are only dependent on Book ID. In TABLE_GENRE, Genre Type is only dependent on Genre ID.