Unit 4: Normalization

4.1. Need of Normalization (Consequences of Bad Design-Insert, Update & Delete Anomalies)
4.2. Normalization
    4.2.1. First Normal Form
    4.2.2. Second Normal Form
    4.2.3. Third Normal Form
    4.2.4. BCNF

Database normalization is a database schema design technique, by which an existing schema is modified to minimize redundancy and dependency of data.

Normalization split a large table into smaller tables and define relationships between them to increases the clarity in organizing data.

Some facts about database normalization

- The words normalization and normal form refer to the structure of a database.
- Normalization was developed by IBM researcher E.F. Codd In the 1970s.
- Normalization increases the clarity in organizing data in Database.

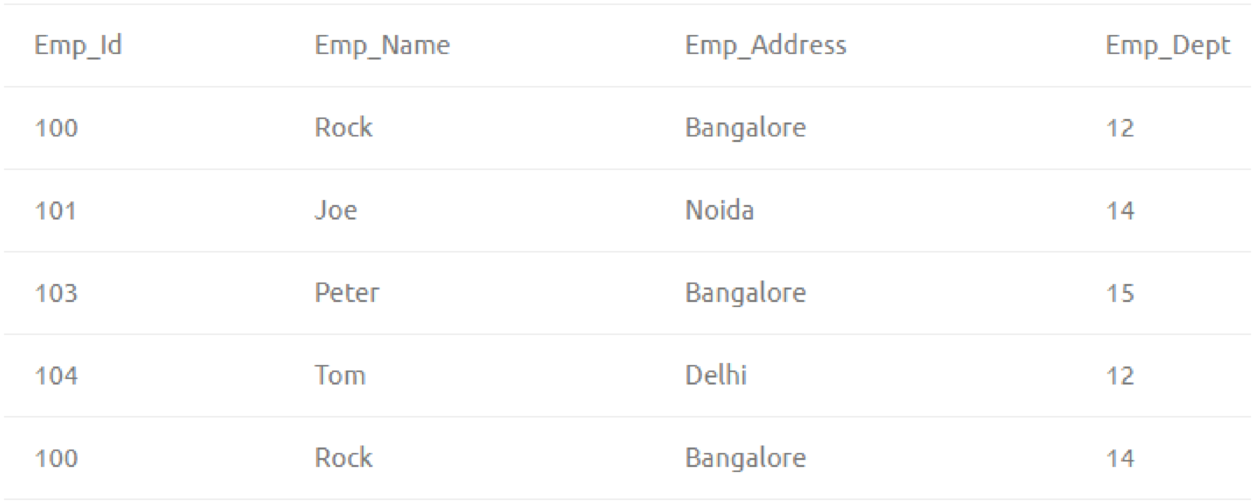
Normalization of a Database is achieved by following a set of rules called 'forms' in creating the database.

Normalization in DBMS

Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly and deletion anomaly. Normalization is a database design technique which organizes tables in a manner that reduces redundancy and dependency of data. It divides larger tables to smaller tables and links them using relationships.

Normalization is also the process of simplifying the design of a database so that it achieves the optimal structure.

Anomalies in DBMS
There are three types of anomalies that occur when the database is not normalized.
1. Insertion Anomaly
2. Update Anomaly
3. Deletion Anomaly
Let us assume we have Employee table as given below.



Emp_Id	Emp_Name	Emp_Address	Emp_Dept
100	Rock	Bangalore	12
101	Joe	Noida	14
103	Peter	Bangalore	15
104	Tom	Delhi	12
100	Rock	Bangalore	14

Update anomaly: Update anomaly is something when we are trying to update some records in table, and that update is causing data inconsistency.
For example, in the above table we have two records for EmpId 100 as he belongs to two departments of the company. If we want to update the address of Rock then we have to update the same in two rows or the data will become inconsistent. If somehow, the correct address gets updated in one department but not in other then as per the database, Rock would be having two different addresses, which is not correct and would lead to inconsistent data.

Insert anomaly: Insert anomaly is something when we are not able to insert data into tables due to some constraints. Suppose a new employee joins the company, who is under training and currently not assigned to any department then we would not be able to insert the data into the table if Emp_Dept field doesn’t allow nulls.

Delete anomaly: Delete anomaly is something when we delete some data from the table, and due to that delete operation we loss some other useful data.

For example, if at a point of time the company closes the department 103 then deleting the rows that are having Emp_Dept as 103 would also delete the information of employee Peter since she is assigned only to this department.

Normalization is a method to remove all these anomalies and bring the database to a consistent state.

We have below normal forms which are used to eliminate or reduce redundancy in database tables.

1. First normal form(1NF)
2. Second normal form(2NF)
3. Third normal form(3NF)
4. Boyce-Codd normal form (BCNF)

ADVANTAGES OF NORMALIZATION

Here we can see why normalization is an attractive prospect in RDBMS concepts.

1) A smaller database can be maintained as normalization eliminates the duplicate data. Overall size of the database is reduced as a result.

2) Better performance is ensured which can be linked to the above point. As databases become lesser in size, the passes through the data becomes faster and shorter thereby improving response time and speed.

3) Narrower tables are possible as normalized tables will be fine-tuned and will have lesser columns which allows for more data records per page.

4) Fewer indexes per table ensures faster maintenance tasks (index rebuilds).

5) Also realizes the option of joining only the tables that are needed.

DISADVANTAGES OF NORMALIZATION

1) More tables to join as by spreading out data into more tables, the need to join table's increases and the task becomes more tedious. The database becomes harder to realize as well.

2) Tables will contain codes rather than real data as the repeated data will be stored as lines of codes rather than the true data. Therefore, there is always a need to go to the lookup table.

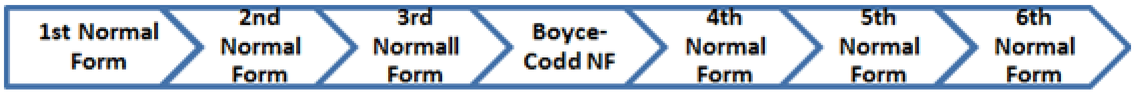
3) Data model becomes extremely difficult to query against as the data model is optimized for applications, not for ad hoc querying. (Ad hoc query is a query that cannot be determined before the issuance of the query. It consists of an SQL that is constructed dynamically and is usually constructed by desktop friendly query tools.). Hence it is hard to model the database without knowing what the customer desires.

4) As the normal form type progresses, the performance becomes slower and slower.

5) Proper knowledge is required on the various normal forms to execute the normalization process efficiently. Careless use may lead to terrible design filled with major anomalies and data inconsistency.

Database normalization rules

Database normalization process is divided into following the normal form:



First Normal Form (1NF)

1NF (First Normal Form) Rules

- Each table cell should contain a single value.
- Each record needs to be unique.

Example:

Sample Employee table, it displays employees are working with multiple departments.

Employee	Age	Department
Melvin	32	Marketing, Sales
Edward	45	Quality Assurance
Alex	36	Human Resource

Employee table following 1NF:

Employee	Age	Department
Melvin	32	Marketing
Melvin	32	Sales
Edward	45	Quality Assurance
Alex	36	Human Resource

Second normal form(2NF)

A table is said to be in 2NF if:

1. Table is in 1NF
2. It has no Partial Dependency, i.e., no non-prime attribute is dependent on any proper subset of any candidate key of the table.

First we will understand what are Prime and Non-prime attributes.

Prime attribute – An attribute, which is a part of the candidate key, is known as a prime attribute.
Non-prime attribute – An attribute, which is not a part of the candidate key, is said to be a non-prime attribute.
For example, we have following table which is having employee data.

Emp_Id	Dept_Id	Emp_Name	Dept_Name
100	12	Rock	IT
101	14	Joe	Finance
100	14	Rock	IT
104	12	Tom	Admin

Above table is in 1NF as all columns are having atomic values. Here **Emp_Id** and **Dept_Id** are the prime attributes. As per 2NF rule Emp_Name and Dept_Name must be dependent upon both prime attributes, but here Emp_name can be identified by Emp_Id and Dept_Name can be identified by Dept_Id alone. So here partial dependency exists. To make this relation in 2NF we have to break above table as:

Emp_Id	Emp_Name	Dept_Id
100	Rock	12
101	Joe	14
100	Rock	14
104	Tom	12

Dept_Id	Dept_Name
12	IT
14	Finance
14	IT
12	Admin

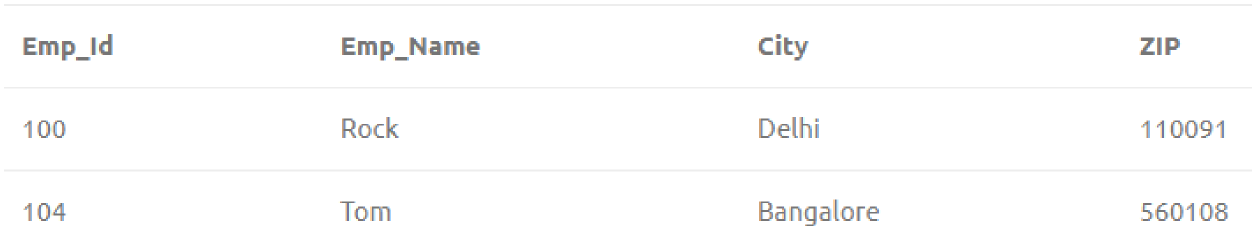
ow there is no partial dependency exist.

Third normal form(3NF)

For a relation to be in Third Normal Form it must satisfy the following –

1. It must be in Second Normal form
2. No non-prime attribute is transitively dependent on prime key attribute.

For example, we have below table for storing employee data.



Emp_Id	Emp_Name	City	ZIP
100	Rock	Delhi	110091
104	Tom	Bangalore	560108

In above relation Emp_Id is the only prime key attribute.
Now If we see **City** can be identified by **Emp_Id** as well as **ZIP**. ZIP is not a prime attribute, and also it is not a super key. So we hold below 2 relationships here.

Emp_Id -> ZIP (ZIP can be identified by Emp_Id)
ZIP -> City (City can be identified by ZIP)

Therefore, below transitive dependency is true for above relation.

Emp_Id -> ZIP -> City

To convert this relation into 3NF we wil break this into 2 relations as:

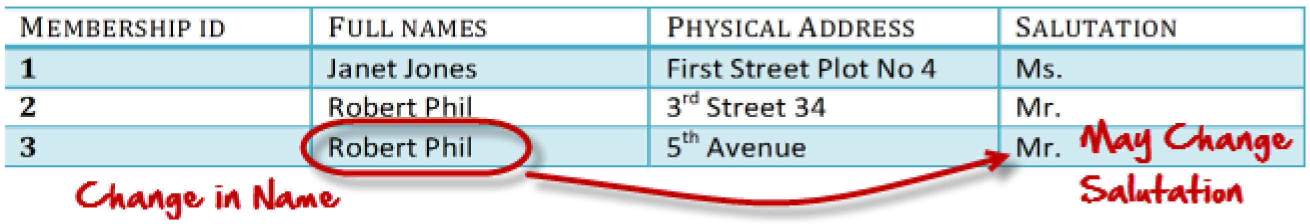
Emp_Id	Emp_Name	ZIP
100	Rock	110091
104	Tom	560108

ZIP	City
110091	Delhi
560108	Bangalore

What are transitive functional dependencies?

A transitive functional dependency is when changing a non-key column, might cause any of the other non-key columns to change

Consider the table Changing the non-key column Full Name may change Salutation.



MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3rd Street 34	Mr.
3	Robert Phil	5th Avenue	Mr.

Change in Name
May Change Salutation

The entity should be considered already in 2NF and no column entry should be dependent on any other entry (value) other than the key for the table.

If such an entity exists, move it outside into a new table.

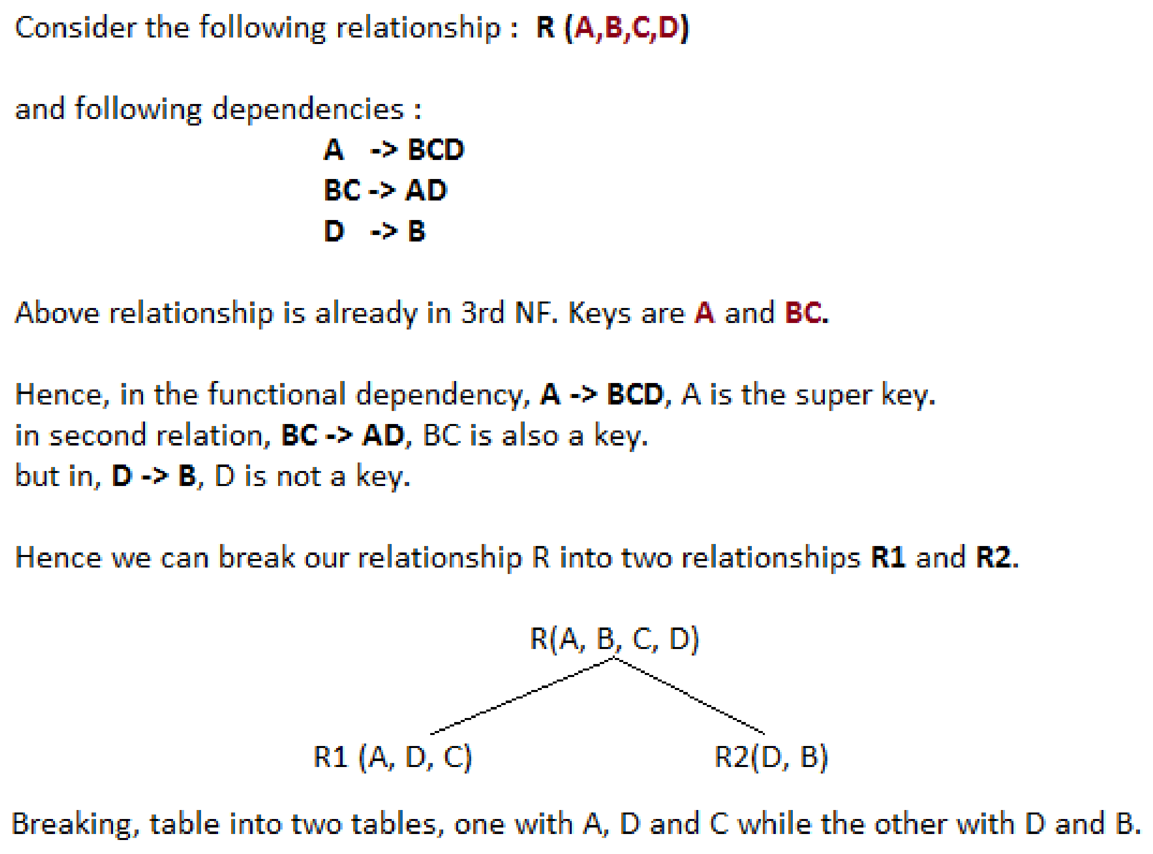
3NF is achieved are considered as the database is normalized.

Boyce and Codd Normal Form (BCNF)

Boyce and Codd Normal Form is a higher version of the Third Normal form. This form deals with certain type of anomaly that is not handled by 3NF. A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF. For a table to be in BCNF, following conditions must be satisfied:

- R must be in 3rd Normal Form
- and, for each functional dependency (X → Y), X should be a super

  Key.



Consider the following relationship : **R (A,B,C,D)**

and following dependencies :

A -> BCD
BC -> AD
D -> B

Above relationship is already in 3rd NF. Keys are **A** and **BC.**

Hence, in the functional dependency, **A -> BCD**, A is the super key.
in second relation, **BC -> AD**, BC is also a key.
but in, **D -> B**, D is not a key.

Hence we can break our relationship R into two relationships **R1** and **R2.**

$$\begin{tikzcd}
 & R(A, B, C, D) \arrow[dl, dash] \arrow[dr, dash] & \\
R1 (A, D, C) & & R2(D, B)
\end{tikzcd}$$

Breaking, table into two tables, one with A, D and C while the other with D and B.