



UNIVERSITY OF  
SINDH

**Assignment Of**

**Database Systems**

**Topic : Normalization**

**Submitted By:**

**Mubashir Ali S/O**

**Mansoor Ali**

**(2k20/ITE/73) Group(A)**

**Submitted To:**

**Sir.Z.A Bhutto**

**Professor Dept:**

**Information Technology**

# Normalization

## What is Normalization?

**Normalization** is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies. Normalization rules divide larger tables into smaller tables and link them using relationships. The purpose of Normalization in SQL is to eliminate redundant (repetitive) data and ensure data is stored logically.

The inventor of the relational model Edgar Codd proposed the theory of normalization of data with the introduction of the First Normal Form, and he continued to extend theory with Second and Third Normal Form. Later he joined Raymond F. Boyce to develop the theory of Boyce-Codd Normal Form.

## Description of Normalization:

Normalization is the process of organizing data in a database. This includes creating tables and establishing relationships between those tables according to rules designed both to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency.

Redundant data wastes disk space and creates maintenance problems. If data that exists in more than one place must be changed, the data must be changed in exactly the same way in all locations. A customer address change is much easier to implement if that data is stored only in the Customers table and nowhere else in the database.

What is an "inconsistent dependency"? While it is intuitive for a user to look in the Customers table for the address of a particular customer, it may not make sense to look there for the salary of the employee who calls on that customer. The employee's salary is related to, or dependent on, the employee and thus should be moved to the Employees table. Inconsistent dependencies can make data difficult to access because the path to find the data may be missing or broken.

There are a few rules for database normalization. Each rule is called a "normal form." If the first rule is observed, the database is said to be in "first normal

form." If the first three rules are observed, the database is considered to be in "third normal form." Although other levels of normalization are possible, third normal form is considered the highest level necessary for most applications.

As with many formal rules and specifications, real world scenarios do not always allow for perfect compliance. In general, normalization requires additional tables and some customers find this cumbersome. If you decide to violate one of the first three rules of normalization, make sure that your application anticipates any problems that could occur, such as redundant data and inconsistent dependencies.

## **Database Normal Forms:**

Here is a list of Normal Forms in SQL:

- 1NF (First Normal Form)
- 2NF (Second Normal Form)
- 3NF (Third Normal Form)
- BCNF (Boyce-Codd Normal Form)
- 4NF (Fourth Normal Form)
- 5NF (Fifth Normal Form)
- 6NF (Sixth Normal Form)

The Theory of Data Normalization in MySQL server is still being developed further. For example, there are discussions even on 6th Normal Form. **However, in most practical applications, normalization achieves its best in 3rd Normal Form.** The evolution of Normalization in SQL theories is illustrated below-



**Database Normal Forms**

## **Database Normalization With Examples:**

Database **Normalization Example** can be easily understood with the help of a case study. Assume, a video library maintains a database of movies rented out. Without any normalization in database, all information is stored in one table as shown below. Let's understand Normalization database with normalization example with solution:

FULL NAMES	PHYSICAL ADDRESS	MOVIES RENTED	SALUTATION
Janet Jones	First Street Plot No 4	Pirates of the Caribbean, Clash of the Titans	Ms.
Robert Phil	3 <sup>rd</sup> Street 34	Forgetting Sarah Marshal, Daddy's Little Girls	Mr.
Robert Phil	5 <sup>th</sup> Avenue	Clash of the Titans	Mr.

Here you see **Movies Rented column has multiple values**. Now let's move into 1st Normal Forms:

### **1NF (First Normal Form) Rules:**

- Each table cell should contain a single value
- Each record needs to be unique.
- Eliminate repeating groups in individual tables.
- Create a separate table for each set of related data.
- Identify each set of related data with a primary key.

Do not use multiple fields in a single table to store similar data. For example, to track an inventory item that may come from two possible sources, an inventory record may contain fields for Vendor Code 1 and Vendor Code 2.

What happens when you add a third vendor? Adding a field is not the answer; it requires program and table modifications and does not smoothly accommodate a dynamic number of vendors. Instead, place all vendor information in a separate table called Vendors, then link inventory to vendors with an item number key, or vendors to inventory with a vendor code key.

The above table in 1NF-

### **1NF Example:**

FULL NAMES	PHYSICAL ADDRESS	MOVIES RENTED	SALUTATION
Janet Jones	First Street Plot No 4	Pirates of the Caribbean	Ms.
Janet Jones	First Street Plot No 4	Clash of the Titans	Ms.
Robert Phil	3 <sup>rd</sup> Street 34	Forgetting Sarah Marshal	Mr.
Robert Phil	3 <sup>rd</sup> Street 34	Daddy's Little Girls	Mr.
Robert Phil	5 <sup>th</sup> Avenue	Clash of the Titans	Mr.

### **Example of 1NF in DBMS**

Before we proceed let's understand a few things —

## **What is a KEY in SQL?**

A **KEY in SQL** is a value used to identify records in a table uniquely. An SQL KEY is a single column or combination of multiple columns used to uniquely

identify rows or tuples in the table. SQL Key is used to identify duplicate information, and it also helps establish a relationship between multiple tables in the database.

Note: Columns in a table that are NOT used to identify a record uniquely are called non-key columns.

## **What is a Primary Key?**



Primary Key

### **Primary Key in DBMS**

A primary is a single column value used to identify a database record uniquely.

It has following attributes

- A primary key cannot be NULL
- A primary key value must be unique
- The primary key values should rarely be changed
- The primary key must be given a value when a new record is inserted.

## **What is a Composite Key?**

A composite key is a primary key composed of multiple columns used to identify a record uniquely

In our database, we have two people with the same name Robert Phil, but they live in different places.



Robert Phil	3 <sup>rd</sup> Street 34	Daddy's Little Girls	Mr.
Robert Phil	5 <sup>th</sup> Avenue	Clash of the Titans	Mr.

*Names are common. Hence you need name as well Address to uniquely identify a record.*

### Composite key in Database

Hence, we require both Full Name and Address to identify a record uniquely. That is a composite key.

Let's move into second normal form 2NF

## 2NF (Second Normal Form) Rules:

- Create separate tables for sets of values that apply to multiple records.
- Relate these tables with a foreign key.
- Rule 1- Be in 1NF
- Rule 2- Single Column Primary Key that does not functionally dependant on any subset of candidate key relation

Records should not depend on anything other than a table's primary key (a compound key, if necessary). For example, consider a customer's address in an accounting system. The address is needed by the Customers table, but also by the Orders, Shipping, Invoices, Accounts Receivable, and Collections tables. Instead of storing the customer's address as a separate entry in each of these tables, store it in one place, either in the Customers table or in a separate Addresses table.

It is clear that we can't move forward to make our simple database in 2nd Normalization form unless we partition the table above.



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

MEMBERSHIP ID	MOVIES RENTED
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

We have divided our 1NF table into two tables viz. Table 1 and Table2. Table 1 contains member information. Table 2 contains information on movies rented.

We have introduced a new column called Membership\_id which is the primary key for table 1. Records can be uniquely identified in Table 1 using membership id.

## **Database – Foreign Key:**

In Table 2, Membership\_ID is the Foreign Key

MEMBERSHIP ID	MOVIES RENTED
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans






Foreign Key

### **Foreign Key in DBMS**


Foreign Key references the primary key of another Table! It helps connect your Tables

- A foreign key can have a different name from its primary key
- It ensures rows in one table have corresponding rows in another
- Unlike the Primary key, they do not have to be unique. Most often they aren't
- Foreign keys can be null even though primary keys can not.

 **Foreign Key**

MEMBERSHIP ID	MOVIES RENTED
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

**Foreign Key references Primary Key**  
**Foreign Key can only have values present in primary key**  
**It could have a name other than that of Primary Key**

 **Primary Key**

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

## Why do you need a foreign key?

Suppose, a novice inserts a record in Table B such as

Insert a record in Table 2 where Member ID = 101

MEMBERSHIP ID	MOVIES RENTED
101	Mission Impossible

But Membership ID 101 is not present in Table 1

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

Database will throw an **ERROR**. This helps in referential integrity

You will only be able to insert values into your foreign key that exist in the unique key in the parent table. This helps in referential integrity.

The above problem can be overcome by declaring membership id from Table 2 as foreign key of membership id from Table 1

Now, if somebody tries to insert a value in the membership id field that does not exist in the parent table, an error will be shown!

## 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 1. 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	3 <sup>rd</sup> Street 34	Mr.
3	Robert Phil	5 <sup>th</sup> Avenue	Mr.

Change in Name

May Change  
Salutation

Let's move into 3NF

## **3NF (Third Normal Form) Rules:**

- Rule 1- Be in 2NF
- Rule 2- Has no transitive functional dependencies
- Eliminate fields that do not depend on the key.

Values in a record that are not part of that record's key do not belong in the table. In general, anytime the contents of a group of fields may apply to more than a single record in the table, consider placing those fields in a separate table.

**For example,** in an Employee Recruitment table, a candidate's university name and address may be included. But you need a complete list of universities for group mailings. If university information is stored in the Candidates table, there is no way to list universities with no current candidates. Create a separate Universities table and link it to the Candidates table with a university code key.

## **Exception:**

Adhering to the third normal form, while theoretically desirable, is not always practical. If you have a Customers table and you want to eliminate all possible inter-field dependencies, you must create separate tables for cities, ZIP codes, sales representatives, customer classes, and any other factor that may be duplicated in multiple records. In theory, normalization is worth pursuing. However, many small tables may degrade performance or exceed open file and memory capacities.

It may be more feasible to apply third normal form only to data that changes frequently. If some dependent fields remain, design your application to require the user to verify all related fields when any one is changed.

To move our 2NF table into 3NF, we again need to again divide our table.

## **3NF Example**

Below is a 3NF example in SQL database:

MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION ID
1	Janet Jones	First Street Plot No 4	2
2	Robert Phil	3 <sup>rd</sup> Street 34	1
3	Robert Phil	5 <sup>th</sup> Avenue	1

MEMBERSHIP ID	MOVIES RENTED
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

SALUTATION ID	SALUTATION
1	Mr.
2	Ms.
3	Mrs.
4	Dr.

We have again divided our tables and created a new table which stores Salutations.

There are no transitive functional dependencies, and hence our table is in 3NF

In Table 3 Salutation ID is primary key, and in Table 1 Salutation ID is foreign to primary key in Table 3

Now our little example is at a level that cannot further be decomposed to attain higher normal form types of normalization in DBMS. In fact, it is already in higher normalization forms. Separate efforts for moving into next levels of normalizing data are normally needed in complex databases. However, we will be discussing next levels of Normalization in DBMS in brief in the following.

## **BCNF (Boyce-Codd Normal Form):**

Even when a database is in 3rd Normal Form, still there would be anomalies resulted if it has more than one **Candidate** Key.

Sometimes BCNF is also referred as **3.5 Normal Form**.

### **4NF (Fourth Normal Form) Rules:**

If no database table instance contains two or more, independent and multi-valued data describing the relevant entity, then it is in 4th Normal Form.

### **5NF (Fifth Normal Form) Rules:**

A table is in 5th Normal Form only if it is in 4NF and it cannot be decomposed into any number of smaller tables without loss of data.

### **6NF (Sixth Normal Form) Proposed:**

6th Normal Form is not standardized, yet however, it is being discussed by database experts for some time. Hopefully, we would have a clear & standardized definition for 6th Normal Form in the near future...

That's all to SQL Normalization!!!

### **Conclusion:**

- Database designing is critical to the successful implementation of a database management system that meets the data requirements of an enterprise system.
- Normalization in DBMS is a process which helps produce database systems that are cost-effective and have better security models.
- Functional dependencies are a very important component of the normalize data process
- Most database systems are normalized database up to the third normal forms in DBMS.
- A primary key uniquely identifies a record in a Table and cannot be null
- A foreign key helps connect table and references a primary key

**THE END**