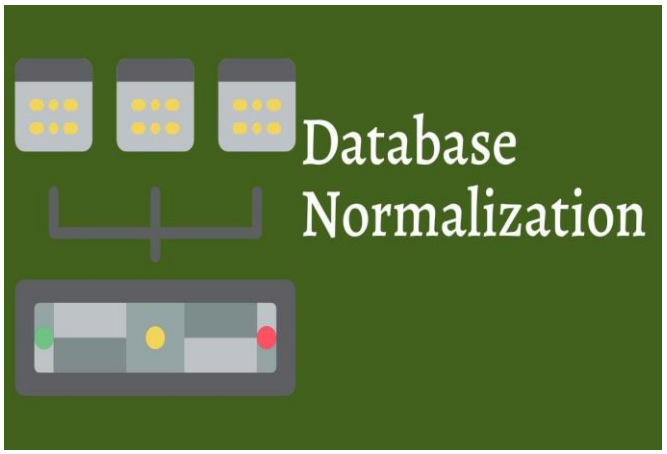
DBMS Normalization: 1NF, 2NF, 3NF Database Example

What is Database Normalization?



Normalization is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies. Normalization rules divides larger tables into smaller tables and links 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.

Types of Normal Forms in DBMS

Here is a list of Normal Forms in SQL:

- **1NF (First Normal Form):**
  column contains atomic (indivisible) values, and each record is unique. This eliminates repeating groups, thereby structuring data into tables and columns.

Ensures that the database table is organized such that each

- **2NF (Second Normal Form):** Builds on 1NF by We need to remove redundant data from a table that is being applied to multiple rows. and placing them in separate tables. It requires all non-key attributes to be fully functional on the primary key.
- **3NF (Third Normal Form):** Extends 2NF by ensuring that all non-key attributes are not only fully functional on the primary key but also independent of each other. This eliminates transitive dependency.

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	3rd Street 34	Forgetting Sarah Marshal, Daddy's Little Girls	Mr.
Robert Phil	5th Avenue	Clash of the Titans	Mr.

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

First Normal Form (1NF)

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

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	3rd Street 34	Forgetting Sarah Marshal	Mr.
Robert Phil	3rd Street 34	Daddy's Little Girls	Mr.
Robert Phil	5th 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 arecord uniquely are called non-key columns.

What is a Primary Key?

Primary Key in DBMS

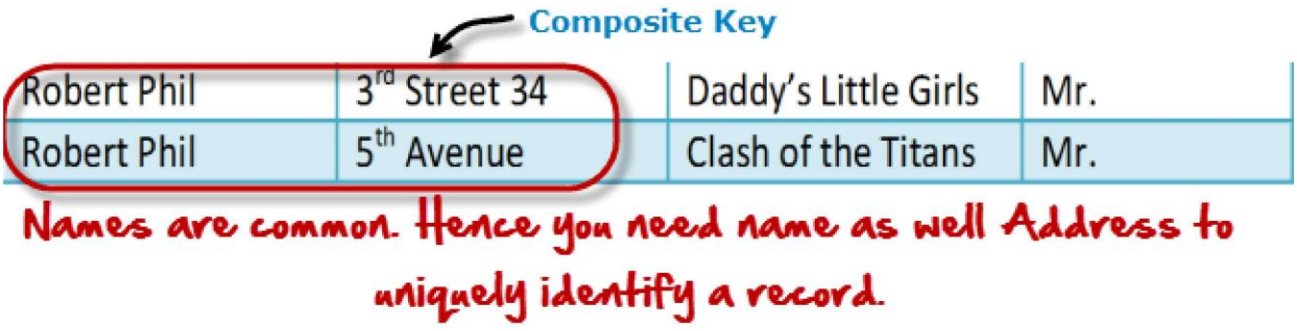
A primary is a single column value used to identify a database record uniquely.

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 anew record is inserted.

What is Composite Key?

A composite key is a primary key composed of multiple columns used to identify a record uniquely
different places.
In our database, we have two people with the same name Robert Phil, but they live in



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

Second Normal Form (2NF)

- Rule 1- Be in 1NF
- Rule 2- Single Column Primary Key that does not functionally dependent on any subset of candidate key relation

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	3rd Street 34	Mr.
3	Robert Phil	5th 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

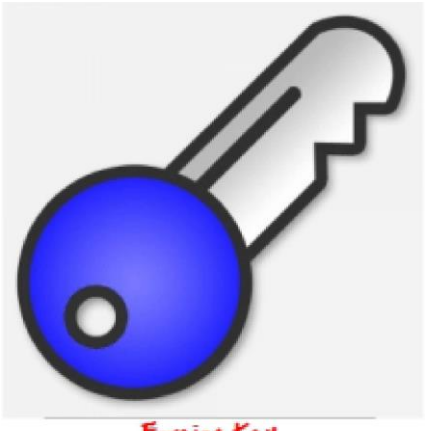
We have divided our 1NF table into two tables viz. Table 1 and Table2. Table 1 contains member information. Table 2 contains information on movies rented.

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

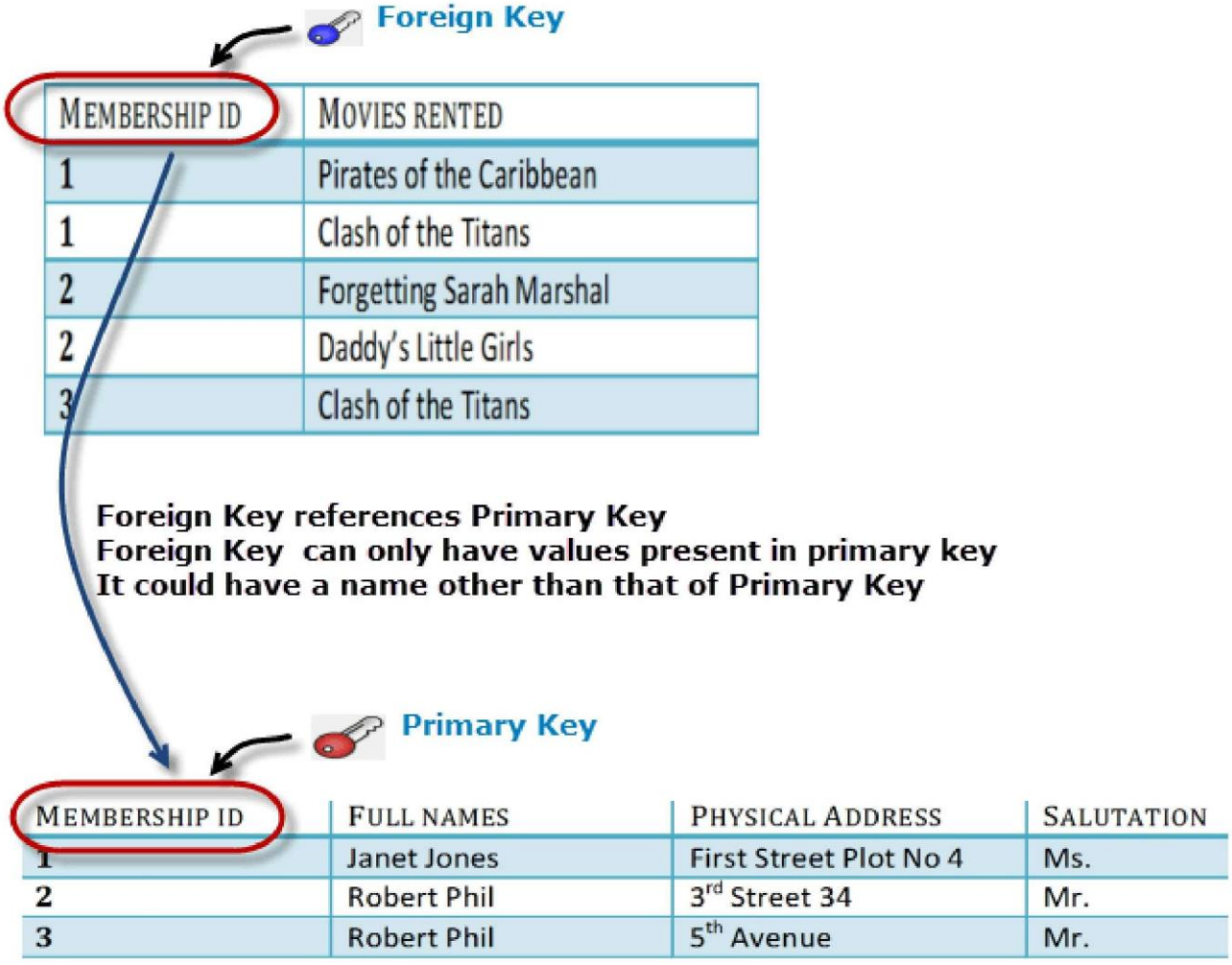
MEMBERSHIP ID	MOVIES RENTED
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans



Foreign Key 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

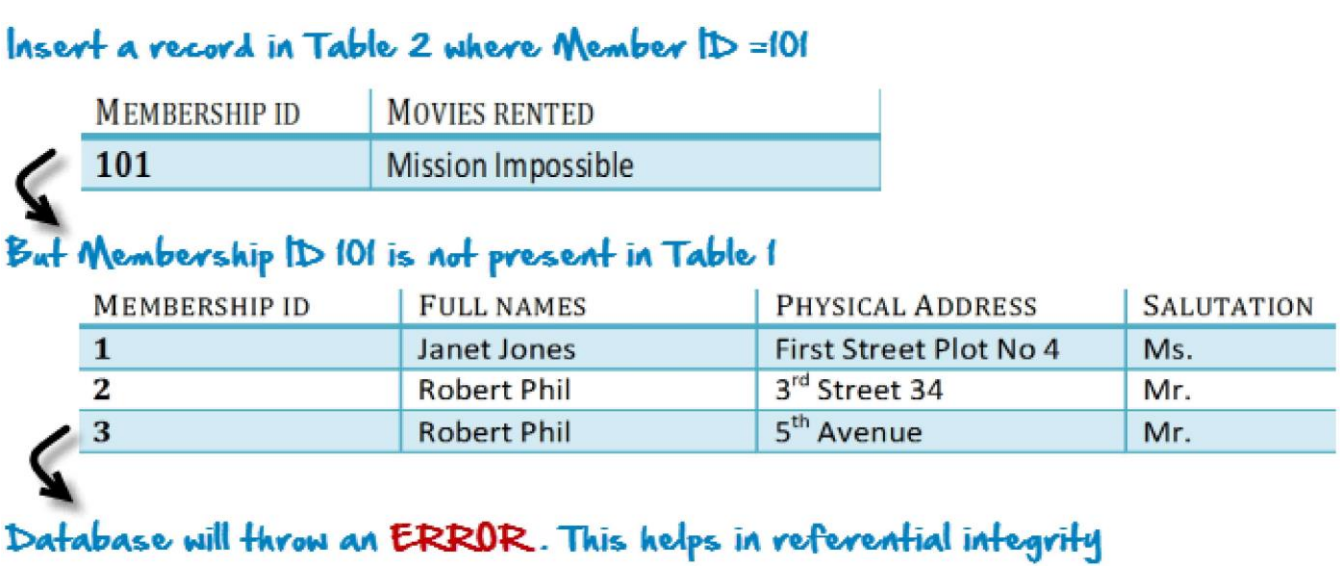


MEMBERSHIP ID	MOVIES RENTED
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

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.

Why do you need a foreign key?

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



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 Table2 as foreign key of membership id from Table1

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	3rd Street 34	Mr.
3	Robert Phil	5th Avenue	Mr.

Change in Name

May Change Salutation

Let's move into 3NF

Third Normal Form (3NF)

- Rule 1- Be in 2NF
- Rule 2- Has no transitive functional dependencies

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	3rd Street 34	1
3	Robert Phil	5th 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 normalisation in DBMS in brief in the following.

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…

Advantages of Normal Form

- **Improve Data Consistency:** Normalization ensures that each piece of data is stored in only one place, reducing the chances of inconsistent data. When datais updated, it only needs to be updated in one place, ensuring consistency.
- **Reduce Data Redundancy:** Normalization helps eliminate duplicate data by dividing it into multiple, related tables. This can save storage space and also make the database more efficient.
- **Improve Query Performance:** Normalized databases are often easier to query. Because data is organized logically, queries can be optimized to run faster.
- **Make Data More Meaningful:** Normalization involves grouping data in a way that makes sense and is intuitive. This can make the database easier to understand and use, especially for people who didn't design the database.
- **Reduce the Chances of Anomalies:** Anomalies are problems that can occur when adding, updating, or deleting data. Normalization can reduce the chances of these anomalies by ensuring that data is logically organized.

Disadvantages of Normalization

- **Increased Complexity:** Normalization can lead to complex relations. An extensive number of tables with foreign keys can be difficult to manage, leading to confusion.
- **Reduced Flexibility:** Due to the strict rules of normalization, there might be less flexibility in storing data that doesn't adhere to these rules.
- **Increased Storage Requirements:** While normalization reduces redundancy, It may be necessary to allocate more storage space to accommodate the additional tables and indices.
- **Performance Overhead:** Joining multiple tables can be costly in terms of performance. The more normalized the data, the more joins are needed, which can slow down data retrieval times.
- **Loss of Data Context:** Normalization breaks down data into separate tables, which can lead to a loss of business context. Examining related tables is necessary to

understand the context of a piece of data.

- **Need for Expert Knowledge:** Implementing a normalized database requires a deep understanding of the data, the relationships between data, and the normalization rules. This requires expert knowledge and can be time-consuming.

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 are record in a Table and cannot be null
- A foreign key helps connect table and references a primary key