

4

Normalization and De-Normalization

Objectives

After completing this lesson, you should be able to:

- Database Normalization
- Types of Normalization
- 1NF, 2NF, 3NF and BCNF
- Exploration of De Normalization



Course Roadmap

RDBMS Concepts



Lesson 1: Introduction to Databases



Lesson 2: Overview of Data Model



Lesson 3: Exploration of Entity Attribute Relationships Notations



Lesson 4: Normalization and De-Normalization

You are here!



Database Normalization

Why Normalization ?

- **Normalization** is a database design technique that reduces data redundancy / Data Repetitions 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.
 - Making relations very large.
 - It isn't easy to maintain and update data as it would involve searching many records in relation.
 - Wastage and poor utilization of disk space and resources.
 - The likelihood of errors and inconsistencies increases.

- So to handle these problems, we should analyze and decompose the relations with redundant data into smaller, simpler, and well-structured relations that are satisfy desirable properties.
- **Normalization is a process of decomposing the relations into relations with fewer attributes.**
- Normalization consists of a series of guidelines that helps to guide you in creating a good database structure.

What is Normalization?

- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
- Normalization divides the larger table into smaller and links them using relationships.
- The normal form is used to reduce redundancy from the database table.

- 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.

Advantages of Normalization

- Normalization helps to minimize data redundancy.
- Greater overall database organization.
- Data consistency within the database.
- Much more flexible database design.
- Enforces the concept of relational integrity.

Disadvantages of Normalization

- You cannot start building the database before knowing what the user needs.
- The performance degrades when normalizing the relations to higher normal forms, i.e., 4NF, 5NF.
- It is very time-consuming and difficult to normalize relations of a higher degree.
- Careless decomposition may lead to a bad database design, leading to serious problems.



Types of Normalization

Types

1. 1NF (First Normal Form)
2. 2NF (Second Normal Form)
3. 3NF (Third Normal Form)
4. BCNF (Boyce-Codd Normal Form)
5. 4NF (Fourth Normal Form)
6. 5NF (Fifth Normal Form)
7. 6NF (Sixth Normal Form)

However, in most practical applications, normalization achieves its best in 3rd Normal Form.

First Normal Form (1NF)

- A relation will be 1NF if it contains an atomic value.
- It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.
- First normal form disallows the multi-valued attribute, composite attribute, and their combinations.

Without Normalization

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

Eg : 1

- Relation EMPLOYEE is not in 1NF because of multi-valued attribute EMP_PHONE.

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385, 9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389, 8589830302	Punjab

The decomposition of the EMPLOYEE table into 1NF has been shown below:

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385	UP
14	John	9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389	Punjab
12	Sam	8589830302	Punjab

Eg: After Normalization

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

Second Normal Form (2NF)

- In the 2NF, relational must be in 1NF.
- In the second normal form, all non-key attributes are fully functional dependent on the primary key

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

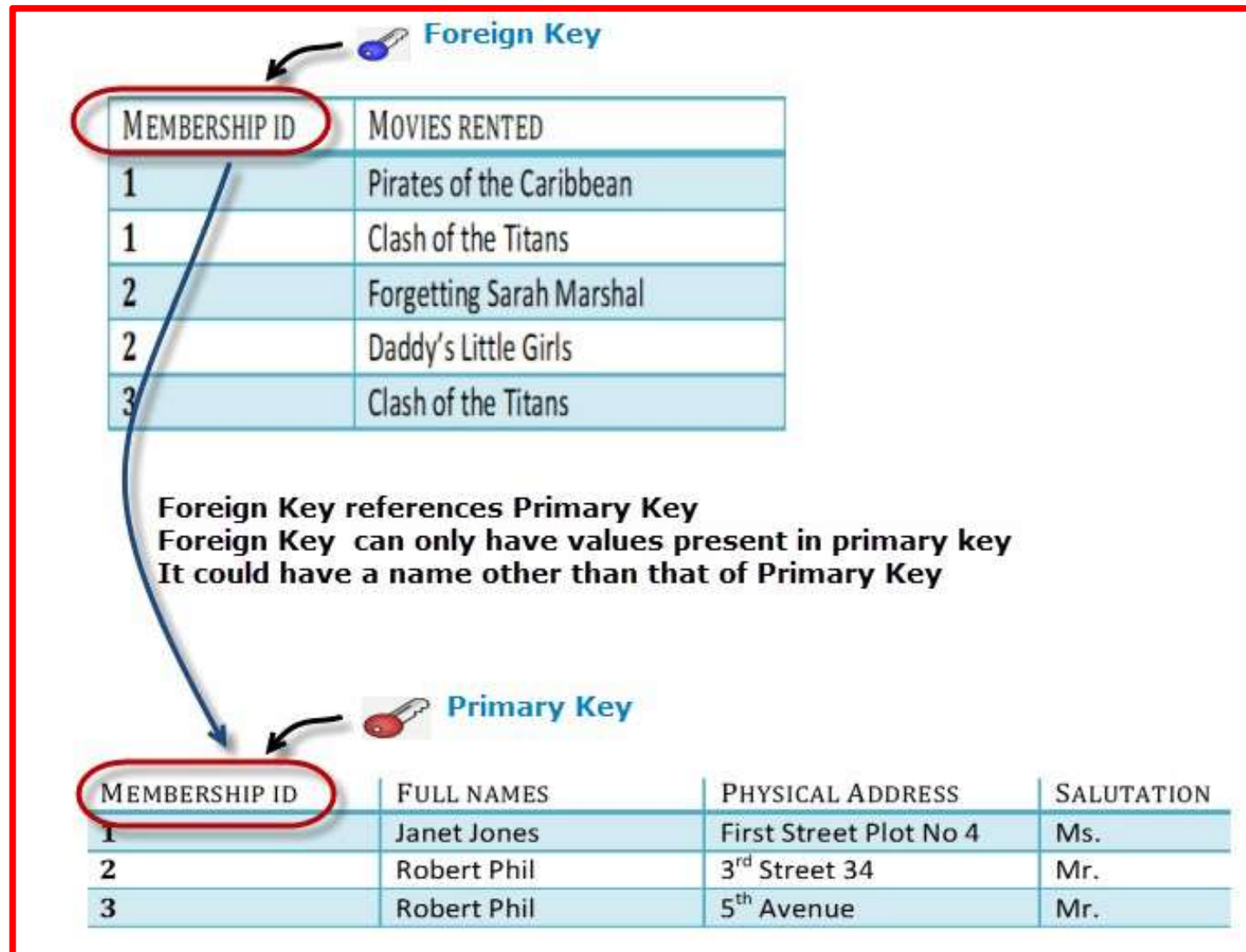
- In Table 2, Membership_ID is the Foreign Key
- Foreign Key references the primary key of another Table! It helps connect your Tables
- A foreign key can have a different name from its primary key
- It ensures rows in one table have corresponding rows in another
- Unlike the Primary key, they do not have to be unique. Most often they aren't
- Foreign keys can be null even though primary keys can not



Foreign Key

Foreign Key in DBMS

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



3NF (Third Normal Form) Rules

- 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.

- 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

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

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



De-Normalization

What Is Denormalization?

- Denormalization is a strategy used on a previously-normalized database to increase performance.
- The idea behind it is to add redundant data where we think it will help us the most.
- We can use extra attributes in an existing table, add new tables, or even create instances of existing tables.
- The usual goal is to decrease the running time of select queries by making data more accessible to the queries or by generating summarized reports in separate tables.

Note that *denormalization* does not mean ‘reversing normalization’ or ‘not to normalize’. It is an optimization technique that is applied after normalization.

- Basically, The process of taking a normalized schema and making it non-normalized is called denormalization, and designers use it to tune the performance of systems to support time-critical operations.
- In a traditional normalized database, we store data in separate logical tables and attempt to minimize redundant data. We may strive to have only one copy of each piece of data in a database.
 - **For example**, in a normalized database, we might have a Courses table and a Teachers table. Each entry in Courses would store the teacherID for a Course but not the teacherName. When we need to retrieve a list of all Courses with the Teacher's name, we would do a join between these two tables.

- In some ways, this is great; if a teacher changes his or her name, we only have to update the name in one place.
- The drawback is that if tables are large, we may spend an unnecessarily long time doing joins on tables.
- Denormalization, then, strikes a different compromise. Under denormalization, we decide that we're okay with some redundancy and some extra effort to update the database in order to get the efficiency advantages of fewer joins.

Pros and Cons of De- Normalization

Pros of Denormalization:

1. Retrieving data is faster since we do fewer joins
2. Queries to retrieve can be simpler(and therefore less likely to have bugs), since we need to look at fewer tables.

Cons of Denormalization:

1. Updates and inserts are more expensive.
2. Denormalization can make *update* and *insert* code harder to write.
3. Data may be inconsistent.
4. Data redundancy necessitates more storage.

Summary

In this lesson, you should have learned that:

- Database Normalization
- Types of Normalization
- 1NF, 2NF, 3NF and BCNF
- Exploration of De Normalization

