Normalization in DBMS - 1NF, 2NF, 3NF, BCNF, 4NF and 5NF

What is normalization in DBMS? - In the world of databases, a large amount of data can often lead to duplication and redundancy. This makes the database inefficient and difficult to manage and update. To tackle these issues, a process called normalization is employed. Normalization is the process of organizing data in a database to reduce redundancy and ensure data integrity. This article will explore the concept of normalization, define normalization in DBMS, its importance, and the different types of normal forms.

The Need of Normalization in DBMS

Purpose of data normalization in DBMS - As elucidated earlier in the definition of normalization in DBMS, normalization is pivotal in mitigating data replication within a database. It presents a structured approach to rectifying the subsequent irregularities within the database, thus enhancing its overall consistency. The following are the need for normalization in DBMS:

Anomalies within a database signify discrepancies that arise due to suboptimal planning and the presence of redundant data.

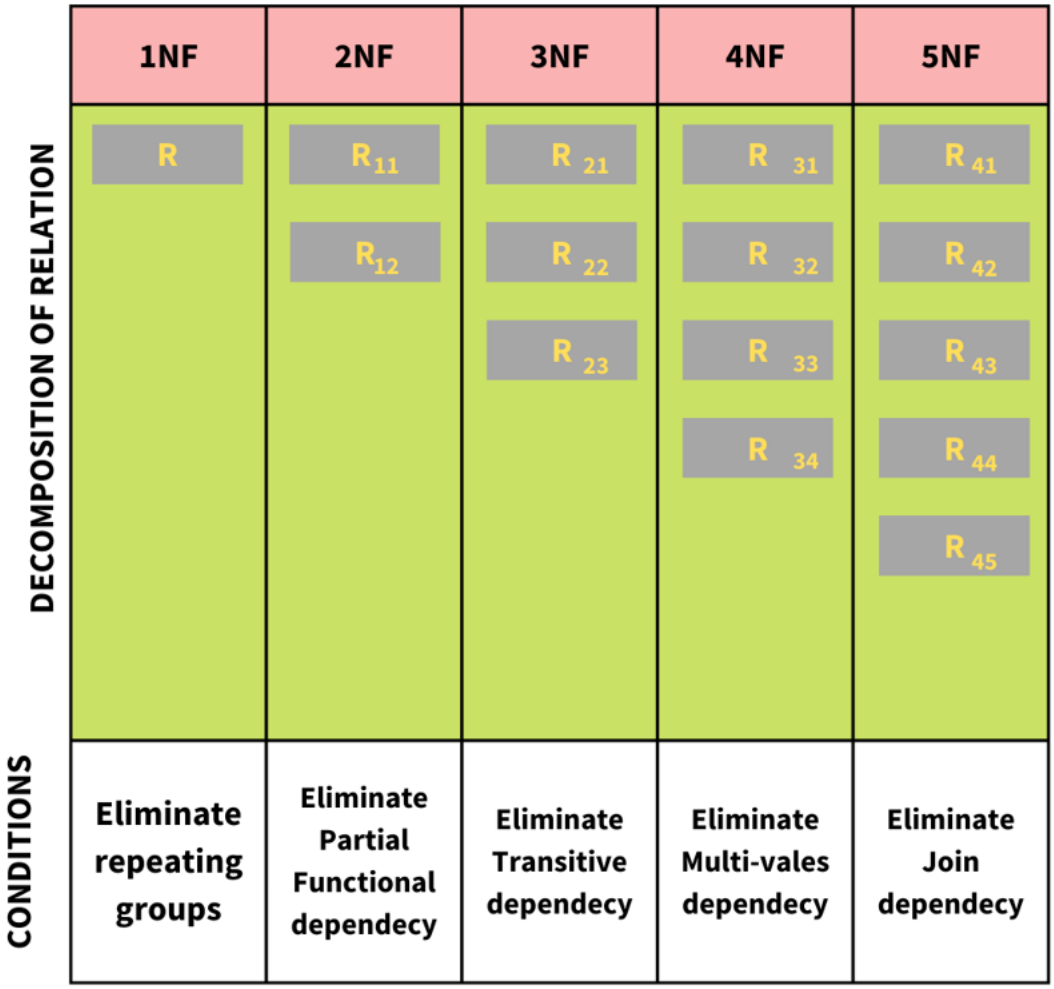
Insertion anomalies arise when the inability to incorporate data into a database stems from the absence of certain attributes during the insertion process.

Updation anomalies: These manifest when identical data elements are duplicated with indistinguishable values, lacking any interconnection between them.

Deletion anomalies come to the fore when erasing a specific portion of data inadvertently leads to removing other indispensable information from the database.

When a database is not properly normalized, it can lead to various data modification anomalies. These anomalies include insertion anomalies, deletion anomalies, and update anomalies. An insertion anomaly occurs when it is impossible to insert new data into a table due to missing information. A deletion anomaly happens when deleting data unintentionally results in the loss of other important data. An update anomaly occurs when updating a single data value requires multiple rows to be modified.

These anomalies can have severe consequences for data integrity and overall database efficiency. Therefore, normalization plays a crucial role in eliminating these anomalies and improving database performance.



Types of Normalization in DBMS

Normalization and its types in DBMS - DBMS normalization is achieved through a series of stages known as normal forms. Each normal form has a set of rules and requirements that a database table must meet to be considered normalized. Let's take a closer look at each of these normalization types in DBMS:

First Normal Form (1NF)

The first normal form (1NF) requires that a relation contains only atomic values. Atomic values are indivisible and cannot be further broken down. This means that each attribute in a table should hold a single value and cannot contain multiple values or sets.

This initial level of normalization focuses on the fundamental structure of tables. In 1NF, each individual cell within a table should hold a solitary value, and each column must possess a distinct and unique name. The primary objective of achieving the first normal form is to eradicate duplicate data entries and streamline query processes.

Second Normal Form (2NF)

To meet the requirements of the second normal form (2NF), a relation must be in 1NF and all non-key attributes must be fully functionally dependent on the primary key. This means that each non-key attribute should depend on the entire primary key and not just a part of it.

2NF addresses the issue of redundant data by establishing a requirement that each non-key attribute must rely solely on the primary key for its existence. This dictates that each column should directly correlate to the primary key and not derive dependency from other columns.

Third Normal Form (3NF)

A relation is said to be in 3NF if it is in 2NF and no transitive dependencies exist. Transitive dependencies occur when an attribute depends on another attribute through a third attribute. By eliminating transitive dependencies, data redundancy is reduced and the database is better organized.

Building upon the principles of 2NF, 3NF takes a step further by mandating that all non-key attributes remain independent of one another. This stipulation necessitates that each column maintain a direct association with the primary key and should not establish any dependencies on other columns existing within the same table.

Boyce Codd's Normal Form (BCNF)

Boyce Codd's normal form (BCNF) is a stronger version of 3NF. A relation is in BCNF if it is in 3NF, and every determinant is a candidate key. In other words, there should be no non-trivial dependencies between attributes in a table.

BCNF represents a more stringent extension of the 3NF guidelines. It ensures that each determinant within a table functions as a potential candidate key. In simpler terms, BCNF guarantees that every non-key attribute derives its significance exclusively from the candidate key, not any other attribute.

Fourth Normal Form (4NF)

A relation is in fourth normal form (4NF) if it is in BCNF and has no multi-valued dependencies. Multi-valued dependencies occur when an attribute depends on a set of values rather than a single value. By eliminating multi-valued dependencies, data redundancy is further reduced.

The concept of 4NF refines the principles of BCNF. It goes a step further by ensuring the absence of multi-valued dependencies within a table. This normalization form eliminates scenarios where a single attribute depends on a combination of other attributes, enhancing the overall integrity of the data structure.

Fifth Normal Form (5NF)

The fifth normal form (5NF) is also called the Project-Join Normal Form (PJNF). A relation is in 5NF if it is in 4NF and does not contain any join dependencies that could result in data loss during the join operation.

Representing the pinnacle of normalization, 5NF involves decomposing a table into smaller constituent tables. This decomposition serves to eradicate data

redundancy and enhance data integrity. The primary objective of achieving 5NF is to maximize the efficiency of data storage and retrieval while minimizing the chances of anomalies or inconsistencies.

Explained Normalization in DBMS with Examples

Here are examples illustrating each of the mentioned normalization forms in DBMS using a hypothetical "Students" table with various attributes:

Assuming a table structure like this:

Students (Student_ID, Course_ID, Course_Instructor, Course_Location, Instructor_Office)

First Normal Form (1NF):

Each cell in the table should contain only a single value. Columns should have unique names.

Example:

STUDENT_ID	COURSE_ID	COURSE_INSTRUCTOR	COURSE_LOCATION	INSTRUCTOR_OFFICE
101	CSCI101	Dr. Smith	Room A	Office 301
101	MATH201	Dr. Johnson	Room B	Office 202

To bring this to 1NF, you would split the table into separate tables for Students, Courses, and Instructors.

Second Normal Form (2NF):

Non-key attributes should be fully dependent on the primary key.

Example:

STUDENT_ID	COURSE_ID	COURSE_INSTRUCTOR	COURSE_LOCATION	INSTRUCTOR_OFFICE
101	CSCI101	Dr. Smith	Room A	Office 301
101	MATH201	Dr. Johnson	Room B	Office 202

Assuming (Student_ID, Course_ID) is the primary key, "Course_Instructor," "Course_Location," and "Instructor_Office" depend only on "Course_ID," violating 2NF. To achieve 2NF, you'd split the table into Students, Courses, and Instructors.

Third Normal Form (3NF):

Non-key attributes should not depend on other non-key attributes.

Example:

STUDENT_ID	COURSE_ID	COURSE_INSTRUCTOR	INSTRUCTOR_OFFICE
101	CSCI101	Dr. Smith	Office 301
101	MATH201	Dr. Johnson	Office 202

Here, "Instructor_Office" depends only on "Course_Instructor" (and not on "Course_ID"). To achieve 3NF, you'd split the table into Students, Courses, Instructors, and Locations.

Boyce-Codd Normal Form (BCNF):

Every determinant (attribute that determines another attribute) should be a candidate key.

Example:

STUDENT_ID	COURSE_ID	COURSE_INSTRUCTOR
101	CSCI101	Dr. Smith
101	MATH201	Dr. Johnson

Here, "Course_Instructor" depends only on "Course_ID," which is a candidate key. This satisfies BCNF.

Fourth Normal Form (4NF):

Tables should not have multi-valued dependencies.

Example:

STUDENT_ID	COURSE_ID	COURSE_INSTRUCTOR
101	CSCI101	Dr. Smith, Dr. Johnson

In this case, "Instructors" is multi-valued, violating 4NF. To achieve 4NF, you'd split the table into Students, Courses, and Instructors.

Fifth Normal Form (5NF):

Decompose tables to remove redundancy and improve integrity.

Example:

STUDENT_ID	COURSE_ID	COURSE_INSTRUCTOR
101	CSCI101	Dr. Smith
101	MATH201	Dr. Johnson

Decomposed into:

Students (Student_ID)

Courses (Course_ID, Course_Instructor)

This eliminates redundancy and optimizes data storage.

Advantages of Normalization in DBMS

Normalization offers several advantages when it comes to database design and management. Let's explore some of these advantages:

Minimizes Data Redundancy

One of the primary benefits of normalization is that it minimizes data redundancy. By organizing data into separate tables and eliminating duplicate information, the overall database size is reduced and storage space is optimized.

Greater Database Organization

Normalization helps in achieving greater organization within the database. By dividing data into smaller, well-structured tables, it becomes easier to manage, search, and update data. This improves overall database performance and efficiency.

Ensures Data Consistency

Normalization ensures data consistency within the database. Since data is stored in a structured and organized manner, there are fewer chances of inconsistencies or conflicting information. This leads to improved data integrity and reliability.

Flexible Database Design

With normalization, databases can be designed in a more flexible manner. As the data is divided into smaller tables, it becomes easier to add, modify, or remove data without affecting the entire database structure. This allows for greater scalability and adaptability as the database grows.

Enforces Relational Integrity

Normalization enforces the concept of relational integrity. By defining relationships between tables through primary and foreign keys, referential integrity is maintained. This ensures that data dependencies are properly managed and prevents the occurrence of orphaned or invalid data.

Disadvantages of Normalization in DBMS

While normalization DBMS offers numerous advantages, it also has some drawbacks. Let's explore the disadvantages associated with normalization:

Requires Complete Understanding of User Needs

Database normalization in DBMS requires a complete understanding of user needs before building the database. This means that careful analysis and planning are necessary to determine the appropriate level of normalization. Failing to understand user requirements can lead to a poorly designed database that is difficult to maintain and update.

Performance Degradation in Higher Normal Forms

As the level of normalization increases, i.e., moving towards higher normal forms such as 4NF and 5NF, the performance of the database can degrade. This is because more complex join operations are required to retrieve data from multiple tables, resulting in slower query execution.

Time-Consuming Process

Normalizing relations to higher degrees can be a time-consuming process. It involves analyzing data dependencies, identifying anomalies, and restructuring the database accordingly. This can be a complex task, especially for large databases with numerous tables and relationships.

Careless Decomposition can Lead to Problems

If the decomposition process is not carefully executed, it can lead to a poorly designed database. Careless decomposition can result in unnecessary joins, data redundancy, and increased complexity. This can cause serious problems in terms of data management and query performance.

Conclusion

Normalization is a crucial process in database management systems that helps organize data, reduce redundancy, and ensure data integrity.

Normal forms serve as a technique to eliminate duplication and enhance the efficiency of database storage.

In the context of the first normal form (1NF), the focus lies on examining the indivisibility of attributes within a relation.

Transitioning to the second normal form (2NF), the scrutiny extends to identifying partial dependencies within a relation.

Advancing to the third normal form (3NF), the evaluation centers around pinpointing transitive dependencies inherent within a relation.
When it comes to the Boyce-Codd normal form (BCNF), the assessment is directed toward ascertaining the presence of superkeys within the left-hand side (LHS) of all functional dependencies.
By following a series of normal forms, databases can be designed in a structured and efficient manner. While normalization offers several advantages, it is important to consider user needs and the level of normalization required carefully. By understanding the benefits, normalization definition in DBMS, and limitations of normalization, you can make informed decisions when designing and managing databases.