Database Normalization

Module Name: **Relational Database Management System**

Module Code: **CS 1211**

Instructors: **Jamila L.**



Anomalies in DBMS

Anomalies refer to *problems or inconsistency* which happened during the operations performed on the table in DBMS.

- There can be many reasons that anomaly occur for example, It occurs when data is stored multiple times unnecessarily in the database i.e. redundant data is present or it occur when all the data is stored in a single table.

There are three types of anomalies that occur when the database is not normalized.

i. Insertion Anomaly

ii. Update Anomaly

iii. Deletion Anomaly

Anomalies in DBMS

i. **Insertion Anomaly:** An insertion anomaly occurs when it is not possible to insert data into a database because some required information is missing or incomplete.

   Example: If a database requires every record to have a primary key, and no value is provided for that key, the record cannot be inserted into the table.

ii. **Deletion Anomaly:** A deletion anomaly occurs when deleting a record unintentionally results in the loss of other important data.

   Example: If a database contains information about customers and their orders, deleting a customer record may also delete all the orders associated with that customer, leading to loss of valuable information.

iii. **Update Anomaly:** An update anomaly occurs when modifying data in a database leads to inconsistencies because the same data is stored in multiple places.

   Example: If a database contains employee salary information in multiple records, and the salary is updated in one record but not in others, it may result in incorrect calculations and reporting.

*** These anomalies can be removed with the process of Normalization and bring the database to a consistent state.

Functional Dependency

Functional dependency (FD) is a set of constraints between two attributes in a relation.

- Functional dependency says that if two tuples have same values for attributes A1, A2,..., An, then those two tuples must have to have same values for attributes B1, B2, ..., Bn.
- Functional dependency is represented by an arrow sign (→) that is, X→Y, where X functionally determines Y.
- The left-hand side attributes determine the values of attributes on the right-hand side.

***In short, functional dependency is all about relationships between attributes (columns) in a table. We can say that the value of one attribute (or a group of attributes) can be used to determine the value of another attribute in the same table.

Functional Dependency

Key Components of Functional Dependency

In functional dependencies we must keep in mind several terms or components:

Determinant:

The attribute(s) used to determine the other attribute (Left hand attribute)

Dependent:

The attribute whose value is determined (Right hand attribute)

Introduction to Normalization

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.

Need of Normalization

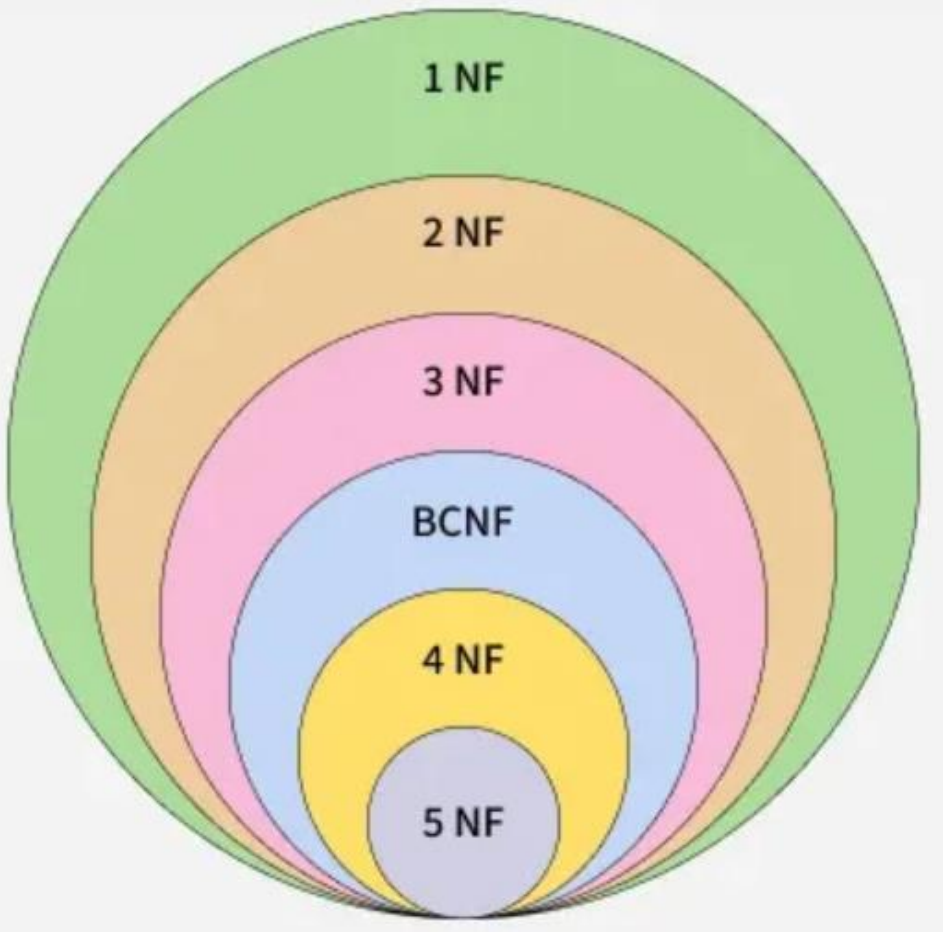
The primary objective for normalizing the relations is to eliminate the anomalies by ensuring the following feature applied:

- **Elimination of Data Redundancy:** Data redundancy refers to the repetition of data in different parts of the database. Normalization helps in reducing or eliminating this redundancy, which can improve the efficiency and consistency of the database.
- **Ensuring Data Consistency:** By eliminating redundancy, normalization helps in preventing inconsistencies and contradictions that can arise due to different versions of the same data.
- **Simplification of Data Management:** By breaking down a complex data structure into simpler tables, normalization makes it easier to manage the data, update it, and retrieve it.
- **Improved Database Design:** By organizing the data in a structured and systematic way, normalization makes it easier to design and maintain the database. It also makes the database more flexible and adaptable to changing business needs.
- **Standardization:** By organizing the data into tables and defining relationships between them, normalization helps in ensuring that the data is stored in a consistent and standard uniform manner.

Normal Forms in DBMS

Normal forms are a set of progressive rules (or design checkpoints) for relational schemas that reduce redundancy and prevent data anomalies.

- Each normal form (1NF, 2NF, 3NF, BCNF, 4NF, 5NF) is stricter than the previous one; meeting a higher normal form implies the lower ones are satisfied.
- Think of them as layers of cleanliness for your tables: the deeper you go, the fewer redundancy and integrity problems you'll have.



Normal Forms in DBMS

1. First Normal Form (1NF): **Eliminating Duplicate Records**

A table is in 1NF must satisfies the following conditions:

i. All columns contain atomic values (i.e., indivisible values).
ii. Each row is unique (i.e., no duplicate rows).
iii. Each column has a unique name.
iv. The order in which data is stored does not matter.

Example of 1NF Violation:

If a table has a column "Phone Numbers" that stores multiple phone numbers in a single cell, it violates 1NF. To bring it into 1NF, you need to separate phone numbers into individual rows.

Normal Forms in DBMS

2. Second Normal Form (2NF): **Eliminating Partial Dependency**

A relation is in 2NF must satisfies the following conditions:

i. Must satisfy all conditions of 1NF

ii. Contains no partial dependency exists; meaning every non-prime attribute (non-key attribute) must depend on the entire primary key, not just a part of it.

Example: For a composite key (StudentID, CourseID), if the "StudentName" depends only on "StudentID" and not on the entire key, it violates 2NF. To normalize, move StudentName into a separate table where it depends only on "StudentID".

Normal Forms in DBMS

3. Third Normal Form (3NF): **Eliminating Transitive Dependency**

A relation is in 3NF if it *satisfies 2NF* and additionally, there are no transitive dependencies.

- In simpler terms, non-prime attributes should not depend on other non-prime attributes.

Example: Consider a table with (StudentID, CourseID, Instructor). If Instructor depends on "CourseID", and "CourseID" depends on "StudentID", then Instructor indirectly depends on "StudentID", which violates 3NF. To resolve this, place Instructor in a separate table linked by "CourseID".

Normal Forms in DBMS

4. Boyce-Codd Normal Form (BCNF): The Strongest Form of 3NF

BCNF is a stricter version of 3NF where for every non-trivial functional dependency (X → Y), X must be a superkey (a unique identifier for a record in the table).

Example: If a table has a dependency (StudentID, CourseID) → Instructor, but neither "StudentID" nor "CourseID" is a superkey, then it violates BCNF. To bring it into BCNF, decompose the table so that each determinant is a candidate key.

Normal Forms in DBMS

5. Fourth Normal Form (4NF): Removing Multi-Valued Dependencies

A table is in 4NF if it is in BCNF and has no multi-valued dependencies. A multi-valued dependency occurs when one attribute determines another, and both attributes are independent of all other attributes in the table.

Example: Consider a table where (StudentID, Language, Hobby) are attributes. If a student can have multiple hobbies and languages, a multi-valued dependency exists. To resolve this, split the table into separate tables for Languages and Hobbies.

Normal Forms in DBMS

6. Fifth Normal Form (5NF): Eliminating Join Dependency

5NF is achieved when a table is in 4NF and all join dependencies are removed. This form ensures that every table is fully decomposed into smaller tables that are logically connected without losing information.

Example: If a table contains (StudentID, Course, Instructor) and there is a dependency where all combinations of these columns are needed for a specific relationship, you would split them into smaller tables to remove redundancy.

Thank you!!