SYLLABUS

Database design: Dependencies and Normal forms - Functional Dependencies, Armstrong's axioms for FD's, closure of a set of FD's, minimal covers-Non- loss decomposition-First, Second, Third Normal Forms, Dependency Preservation-Boyce/Codd Normal Form-Multivalued Dependencies and Fourth Normal Form- Join Dependencies and Fifth Normal

---

Normal forms (upto BCNF).
Introduction to Normalization
- **Normalization** is a process of organizing the data in the database.
- It is a systematic approach of decomposing tables to eliminate data redundancy.
- It was developed by **E. F. Codd.**
- Normalization is a multi-step process that puts the data into a tabular form by removing the duplicate data from the relation tables.
- It is a step by step decomposition of complex records into simple records.
- It is also called as Canonical Synthesis.
- It is the technique of building database structures to store data.

Definition of Normalization
"Normalization is a process of designing a consistent database by minimizing redundancy and ensuring data integrity through decomposition which is lossless."

Features of Normalization
- Normalization avoids the data redundancy.
- It is a formal process of developing data structures.
- It promotes the data integrity.
- It ensures data dependencies make sense that means data is logically stored.
- It eliminates the undesirable characteristics like Insertion, Updation and Deletion Anomalies.

Types of Normalization
Following are the types of Normalization:

1. First Normal Form
2. Second Normal Form
3. Third Normal Form
4. Fourth Normal Form
5. Fifth Normal Form
6. BCNF (Boyce – Codd Normal Form)
7. DKNF (Domain Key Normal Form)

1. First Normal Form (1NF)

- First Normal Form (1NF) is a simple form of Normalization.
- It simplifies each attribute in a relation.
- In 1NF, there should not be any repeating group of data.
- Each set of column must have a unique value.
- It contains atomic values because the table cannot hold multiple values.

Example: Employee Table

ECode	Employee_Name	Department_Name
1	ABC	Sales, Production
2	PQR	Human Resource
3	XYZ	Quality Assurance, Marketing

Employee Table using 1NF

ECode	Employee_Name	Department_Name
1	ABC	Sales
1	ABC	Production
2	PQR	Human Resource
3	XYZ	Quality Assurance
3	XYZ	Marketing

2. Second Normal Form (2NF)

- In 2NF, the table is required in 1NF.
- The main rule of 2NF is, *'No non-prime attribute is dependent on the proper subset of any candidate key of the table.'*
- An attribute which is not part of candidate key is known as non-prime attribute.

Example : Employee Table using 1NF

ECode	Employee_Name	Employee_Age
1	ABC	38
1	ABC	38

2	PQR	38
3	XYZ	40
3	XYZ	40

Candidate Key: ECode, Employee_Name
Non prime attribute: Employee_Age
  •   The above table is in 1NF. Each attribute has atomic values. However, it is not in 2NF because non prime attribute Employee_Age is dependent on ECode alone, which is a proper subset of candidate key. This violates the rule for 2NF as the rule says 'No non-prime attribute is dependent on the proper subset of any candidate key of the table'.

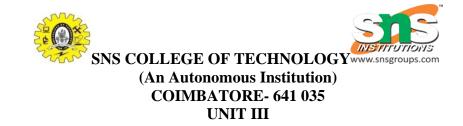**2NF (Second Normal Form) : Employee1 Table**

ECode	Employee_Age
1	38
2	38
3	40

Employee2 Table

ECode	Employee_Name
1	ABC
1	ABC
2	PQR
3	XYZ
3	XYZ

  •   Now, the above tables comply with the Second Normal Form (2NF).

3. Third Normal Form (3NF)
  •   Third Normal Form (3NF) is used to minimize the transitive redundancy.
  •   In 3NF, the table is required in 2NF.
  •   While using the 2NF table, there should not be any transitive partial dependency.

- 3NF reduces the duplication of data and also achieves the data integrity.

Example : <Employee> Table

EId	Ename	DOB	City	State	Zip
001	ABC	10/05/1990	Pune	Maharashtra	411038
002	XYZ	11/05/1988	Mumbai	Maharashtra	400007

- In the above <Employee> table, EId is a primary key but City, State depends upon Zip code.
- The dependency between Zip and other fields is called Transitive Dependency.
- Therefore we apply 3NF. So, we need to move the city and state to the new <Employee_Table2> table, with Zip as a Primary key.

<Employee_Table1> Table

EId	Ename	DOB	Zip
001	ABC	10/05/1990	411038
002	XYZ	11/05/1988	400007

<Employee_Table2> Table

City	State	Zip
Pune	Maharashtra	411038
Mumbai	Maharashtra	400007

- The advantage of removing transitive dependency is, it reduces the amount of data dependencies and achieves the data integrity.
- In the above example, using with the 3NF, there is no redundancy of data while inserting the new records.
- The City, State and Zip code will be stored in the separate table. And therefore the updation becomes more easier because of no data redundancy.

4. BCNF (Boyce – Code Normal Form)

- BCNF which stands for Boyce – Code Normal From is developed by Raymond F. Boyce and E. F. Codd in 1974.
- BCNF is a higher version of 3NF.

- It deals with the certain type of anomaly which is not handled by 3NF.
- A table complies with BCNF if it is in 3NF and any attribute is fully functionally dependent that is A → B. (Attribute 'A' is determinant).
- If every determinant is a candidate key, then it is said to be BCNF.
- Candidate key has the ability to become a primary key. It is a column in a table.

Example : <EmployeeMain> Table

Empid	Ename	DeptName	DepType
E001	ABC	Production	D001
E002	XYZ	Sales	D002

The functional dependencies are:
Empid → EmpName
DeptName → DeptType

Candidate Key:
Empid
DeptName

- The above table is not in BCNF as neither Empid nor DeptName alone are keys.
- We can break the table in three tables to make it comply with BCNF.

<Employee> Table

Empid	EmpName
E001	ABC
E002	XYZ

<Department> Table

DeptName	DeptType
Production	D001
Sales	D002

<Emp_Dept> Table

Empid	DeptName

E001	Production
E002	Sales

Now, the functional dependencies are:
Empid → EmpName
DeptName → DeptType

Candidate Key:
<Employee> Table : Empid
<Department> Table : DeptType
<Emp_Dept> Table : Empid, DeptType
- So, now both the functional dependencies left side part is a key, so it is in the BCNF.

5. Fourth Normal Form (4NF)
- Fourth Normal Form (4NF) does not have non-trivial multivalued dependencies other than a candidate key.
- 4NF builds on the first three normal forms (1NF, 2NF and 3NF) and the BCNF.
- It does not contain more than one multivalued dependency.
- This normal form is rarely used outside of academic circles.

For example : A table contains a list of three things that is 'Student', 'Teacher', 'Book'. Teacher is in charge of Student and recommended book for each student. These three elements (Student, Teacher and Book) are independent of one another. Changing the student's recommended book, for instance, has no effect on the student itself. This is an example of multivalued dependency, where an item depends on more than one value. In this example, the student depends on both teacher and book.
- Therefore, 4NF states that a table should not have more than one dependencies.

6. Fifth Normal Form (5NF)
- 5NF is also knows as Project-Join Normal Form (PJ/NF).
- It is designed for reducing the redundancy in relational databases.
- 5NF requires semantically related multiple relationships, which are rare.
- In 5NF, if an attribute is multivalued attribute, then it must be taken out as a separate entity.
- While performing 5NF, the table must be in 4NF.

7. DKNF (Domain Key Normal Form)
- DKNF stands for Domain Key Normal Form requires the database that contains no constraints other than domain constraints and key constraints.
- In DKNF, it is easy to build a database.

- It avoids general constraints in the database which are not clear domain or key constraints.
- The 3NF, 4NF, 5NF and BCNF are special cases of the DKNF.
- It is achieved when every constraint on the relation is a logical consequence of the definition.