Database Management System

Unit-4

Mrs. Kiran Bala Dubey

Assistant Professor

Department of Computer Science

Govt. N. P. G. College of Science,

Raipur

UNIT - IV: Structured Query Language (SQL)

Normalization concept in logical model; Pitfalls in database design, update anomalies: Functional dependencies, Join dependencies, Normal forms (1NF, 2NF, 3NF). Boyce Codd Normal form, Decomposition, Multi-Valued Dependencies, 4NF, 5NF.De-normalization.

Database anomalies : There are three types of anomalies that occur when the database is not normalized. These are – Insertion, update and deletion anomaly.

Example: employee table that has four attributes: emp_id, emp_name, emp_address and emp_dept .

- **Update anomaly**: In the above table we have two rows for employee Ram as he belongs to two departments of the company. If we want to update the address of Ram then we have to update the same in two rows or the data will become inconsistent.
- **Insert anomaly**: Suppose a new employee joins the company, who is under training and currently not assigned to any department then we would not be able to insert the data into the table if emp_dept field doesn't allow nulls.
- **Delete anomaly**: Suppose, if at a point of time the company closes the department D890 then deleting the rows that are having emp_dept as D890 would also delete the information of employee Mahesh since he is assigned only to this department.

Functional Dependency

- Functional Dependency (FD) is a constraint that determines the relation of one attribute to another attribute in a Database Management System.
- Functional dependency defines the dependency of attributes of a table with other attributes or Primary key attribute of the same table.
- Functional Dependency helps to maintain the quality of data in the database. It plays a vital role to find the difference between good and bad database design.
- A functional dependency is denoted by an arrow "→". The functional dependency of X on Y is represented by X → Y.

There are 5 kinds of dependencies-

1. **Fully Functional dependency –** In it all the non key attribute depends on the primary key attribute.

   Stud(Rno,snam, class, marks) Here rno is primary key.

   Rno->snam

   Rno->class

   Rno->marks

2. **Partial functional dependency –** In it the non-key attribute besides depending on primary key also depend on any other attribute of the table.

   Rno, course_name,Stud_name,address,date_of_completion

   Here rno is the primary key. The functional dependencies are-

   Rno->stud_name

   Rno->address

   Rno,course_name->date_of_completion

   So it is partial dependency.

3. Transitive dependency – In which one non-key attribute depends on the other non-key attribute.

Here if

A·>B

B->C

Then A->C

i.e. if A is determined by B and B is determined by C then A is automatically determined by C.

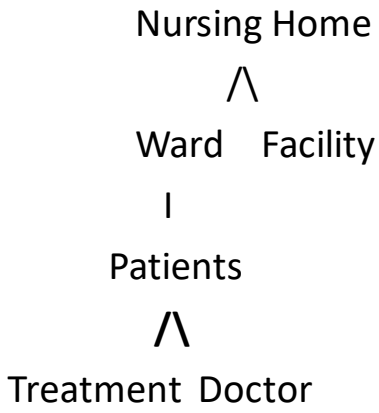
4. Multivalued dependency- Multivalued dependency occurs when two attributes in a table are independent of each other but, both depend on a third attribute.

BIKE_MODEL → → MANUF_YEAR

BIKE_MODEL → → COLOR

In this case, these two columns can be called as multivalued dependent on BIKE_MODEL.

5. Join dependency – In this hierarchical structure in which two or more attribute depends on one thing but they themselves are not related.



In this example, the ward and facility depends on Nursing Home but these two are not dependent each other. This type of dependency is known as join dependency.

Decomposition - Decomposition in DBMS removes redundancy, anomalies and inconsistencies from a database by dividing the table into multiple tables.

- **Good decomposition (Lossless Decomposition)**

  The information will not lose from the relation when decomposed. Decomposition is lossless if it is feasible to reconstruct relation R from decomposed tables using Joins. The join would result in the same original relation.

- **Bad decomposition (Lossy Decomposition)**

  As the name suggests, when a relation is decomposed into two or more relational schemas, the loss of information is unavoidable when the original relation is retrieved.

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 the undesirable characteristics like Insertion, Update and Deletion Anomalies.
- Normalization divides the larger table into the smaller table and links them using relationship.
- The normal form is used to reduce redundancy from the database table.

First Normal Form (1NF)

- 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.
- **Example:** 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

Second Normal Form (2NF)

- In the 2NF, relational must be in 1NF and all non-key attributes are fully functional dependent on the primary key
- **Example:** Let's assume, a school can store the data of teachers and the subjects they teach. In a school, a teacher can teach more than one subject.
- Teacher Table -

TEACHER_ID	SUBJECT	TEACHER_AGE
25	Chemistry	30
25	Biology	30
47	English	35
83	Math	38
83	Computer	38

In the given table, non-prime attribute TEACHER_AGE is dependent on TEACHER_ID which is a proper subset of a candidate key.

To convert the given table into 2NF, we decompose it into two tables:-

1. Teacher detail table
2. Teacher subject table

1.

TEACHER_ID	TEACHER_AGE
25	30
47	35
83	38

2.

TEACHER_ID	SUBJECT
25	Chemistry
25	Biology
47	English
83	Math
83	Computer

Third Normal Form (3NF)

- A relation will be in 3NF if it is in 2NF and not contain any transitive dependency.
- 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.

EMPLOYEE_DETAIL table:

EMP_ID	EMP_NAME	EMP_ZIP	EMP_STATE	EMP_CITY
222	Harry	201010	UP	Noida
333	Stephan	02228	US	Boston
444	Lan	60007	US	Chicago
555	Katharine	06389	UK	Norwich
666	John	462007	MP	Bhopal

Here, EMP_STATE & EMP_CITY dependent on EMP_ZIP and EMP_ZIP dependent on EMP_ID. The non-key attributes (EMP_STATE, EMP_CITY) transitively dependent on super key(EMP_ID). It violates the rule of third normal form.

That's why we need to move the EMP_CITY and EMP_STATE to the new <EMPLOYEE_ZIP> table, with EMP_ZIP as a Primary key.

EMPLOYEE table:

EMP_ID	EMP_NAME	EMP_ZIP
222	Harry	201010
333	Stephan	02228
444	Lan	60007
555	Katharine	06389
666	John	462007

EMPLOYEE_ZIP table:

EMP_ZIP	EMP_STATE	EMP_CITY
201010	UP	Noida
02228	US	Boston
60007	US	Chicago
06389	UK	Norwich
462007	MP	Bhopal

Boyce Codd normal form (BCNF)

- BCNF is the advance version of 3NF and is also known as 3.5 Normal Form.
- A table is in BCNF if every functional dependency X → Y, X is the super key of the table.
- For BCNF, the table should be in 3NF, if an attribute of a composite key is dependent on an attribute of the other composite key.
- **Example: Professor table**

Prof_code	Dept	HOD	Percent
P1	Physics	Ghosh	50
P1	Maths	Krishnan	50
P2	Chemistry	Rao	25
P2	Physics	Ghosh	75
P3	Maths	Krishnan	100

Dept

Dept	HOD
Physics	Ghosh
Maths	Krishnan
Chemistry	Rao

professor

Prof_code	Dept	Percent
P1	Physics	50
P1	Maths	50
P2	Chemistry	25
P2	Physics	75
P3	Maths	100

Fourth normal form (4NF)

- A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
- For a dependency A → B, if for a single value of A, multiple values of B exists, then the relation will be a multi-valued dependency.

Student table

STU_ID	COURSE	HOBBY
21	Computer	Dancing
21	Math	Singing
34	Chemistry	Dancing
74	Biology	Cricket
59	Physics	Hockey

In the STUDENT relation, a student with STU_ID, **21** contains two courses, **Computer** and **Math** and two hobbies, **Dancing** and **Singing**. So there is a Multi-valued dependency on STU_ID, which leads to unnecessary repetition of data. So to make the above table into 4NF, we can decompose it into two tables:

STUDENT_COURSE

STU_ID	COURSE
21	Computer
21	Math
34	Chemistry
74	Biology
59	Physics

STUDENT_HOBBY

STU_ID	HOBBY
21	Dancing
21	Singing
34	Dancing
74	Cricket
59	Hockey

Fifth normal form (5NF)

- A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.
- 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy.
- 5NF is also known as Project-join normal form (PJ/NF).

SUBJECT	LECTURER	SEMESTER
Computer	Anshika	Semester 1
Computer	John	Semester 1
Math	John	Semester 1
Math	Akash	Semester 2
Chemistry	Praveen	Semester 1

- In the given table, John takes both Computer and Math class for Semester 1 but he doesn't take Math class for Semester 2. In this case, combination of all these fields required to identify a valid data.
- Suppose we add a new Semester as Semester 3 but do not know about the subject and who will be taking that subject so we leave Lecturer and Subject as NULL. But all three columns together acts as a primary key, so we can't leave other two columns blank.
- So to make the above table into 5NF, we can decompose it into three relations P1, P2 & P3:

P1

SEMESTER	SUBJECT
Semester 1	Computer
Semester 1	Math
Semester 1	Chemistry
Semester 2	Math

P2

SUBJECT	LECTURER
Computer	Anshika
Computer	John
Math	John
Math	Akash
Chemistry	Praveen

P3

SEMSTER	LECTURER
Semester 1	Anshika
Semester 1	John
Semester 1	John
Semester 2	Akash
Semester 1	Praveen

Normal Form	Description
1NF	A relation is in 1NF if it contains an atomic value.
2NF	A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
3NF	A relation will be in 3NF if it is in 2NF and no transition dependency exists.
4NF	A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
5NF	A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.

De-normalization

- De-normalization is the process of increasing the redundancy in the database.
- It is the opposite process of normalization.
- It is mostly done for improving the performance.
- It is a strategy that database managers use to increase the performance of a database structure.
- De-normalization adds redundant data normalized database for reducing the problems with database queries which combine data from the various tables into a single table.
- The process of adding redundant data to get rid of complex join, in order to optimize database performance. This is done to speed up database access by moving from higher to lower form of normalization.
- Data is included in one table from another in order to eliminate the second table which reduces the number of JOINS in a query and thus achieves performance.

Difference Between Normalization and De-normalization

- Normalization and de-normalization are the methods used in databases. The terms are differentiable where **Normalization** is a technique of minimizing the insertion, deletion and update anomalies through eliminating the redundant data. On the other hand, **De-normalization** is the inverse process of normalization where the redundancy is added to the data to improve the performance of the specific application and data integrity.

Key Differences Between Normalization and De-normalization

- Normalization is the technique of dividing the data into multiple tables to reduce data redundancy and inconsistency and to achieve data integrity. On the other hand, De-normalization is the technique of combining the data into a single table to make data retrieval faster.
- Normalization is used in **OLTP** system, which emphasizes on making the insert, delete and update anomalies faster. As against, De-normalization is used in **OLAP** system, which emphasizes on making the search and analysis faster.
- Data integrity is maintained in normalization process while in de-normalization data integrity harder to retain.
- Redundant data is eliminated when normalization is performed whereas de-normalization increases the redundant data.
- Normalization increases the number of tables and joins. In contrast, de-normalization reduces the number of tables and join.
- Disk space is wasted in de-normalization because same data is stored in different places. On the contrary, disk space is optimized in a normalized table.

BASIS FOR COMPARISON	NORMALIZATION	DENORMALIZATION
Basic	Normalization is the process of creating a set schema to store non-redundant and consistent data.	Denormalization is the process of combining the data so that it can be queried speedily.
Purpose	To reduce the data redundancy and inconsistency.	To achieve the faster execution of the queries through introducing redundancy.
Used in	OLTP system, where the emphasize is on making the insert, delete and update anomalies faster and storing the quality data.	OLAP system, where the emphasis is on making the search and analysis faster.
Data integrity	Maintained	May not retain
Redundancy	Eliminated	Added
Number of tables	Increases	Decreases
Disk space	Optimized usage	Wastage