For

Paper: E401 (INTRODUCTION TO DATABASE MANAGEMENT SYSTEM)

Topics: Normalization,1NF,2NF,3NF,BCNF

Normalization

Normalization is the process of decomposing the relations into well structured relations to organize the data in the database to remove redundancy of data, insertion anomaly, update anomaly and deletion anomaly.

Insertion anomaly

If a tuple is inserted in referencing relation and referencing attribute value is not present in referenced attribute, it will not allow inserting in referencing relation.
For Example,

STUDENT

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY	STUD_AGE
1	RAM	9716271721	Haryana	India	20
2	RAM	9898291281	Punjab	India	19
3	SUJIT	7898291981	Rajsthan	India	18
4	SURESH		Punjab	India	21

Table 1

STUDENT_COURSE

STUD_NO	COURSE_NO	COURSE_NAME
1	C1	DBMS
2	C2	Computer Networks
1	C2	Computer Networks

Table 2

Here, if we try to insert a record in STUDENT_COURSE with STUD_NO =7, it will not allow.

Deletion and Updation anomaly

If a tuple is deleted or updated from referenced relation and referenced attribute value is used by referencing attribute in referencing relation, it will not allow deleting the tuple from referenced relation. For Example,

STUDENT

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY	STUD_AGE
1	RAM	9716271721	Haryana	India	20
2	RAM	9898291281	Punjab	India	19
3	SUJIT	7898291981	Rajsthan	India	18
4	SURESH		Punjab	India	21

Table 1

STUDENT_COURSE

STUD_NO	COURSE_NO	COURSE_NAME
1	C1	DBMS
2	C2	Computer Networks
1	C2	Computer Networks

Table 2

Here, if we try to delete a record from STUDENT with STUD_NO =1, it will not allow.

The normal forms used for normalization are:

1. First normal form(1NF)
2. Second normal form(2NF)
3. Third normal form(3NF)
4. Boyce & Codd normal form (BCNF)
5. Fourth normal form (4NF)
6. Fifth normal form (5NF)

First normal form (1NF)

The first normal form is based on the simple or atomic attribute and single valued attribute. A relation is said to be in 1NF if all the attributes of the relation are atomic and single valued.

Example-

Suppose a company wants to store the names and contact details of its employees. It creates a table that looks like this:

Employee

Emp_id	Emp_name	Emp_address	Emp_mobile
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212 9900012222
103	Ron	Chennai	7778881212
104	Lester	Bangalore	9990000123 8123450987

Here, the Relation employee is not in 1NF, because employees with employee id 102 and 104 are having two phone numbers. i.e. the Emp_mobile attribute is a multi valued attribute.

After normalization to 1NF the relation will be like this:

Emp_id	**Emp_name**	**Emp_address**	**Emp_mobile**
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212
102	Jon	Kanpur	9900012222
103	Ron	Chennai	7778881212
104	Lester	Bangalore	9990000123
104	Lester	Bangalore	8123450987

Second normal form (2NF)

The second normal form is based on the full functional dependency. A relation is said to be in 2NF if it is first in 1NF and all non-key attributes are fully functional dependent on the primary key.. i.e. no partial dependency exists.

Example-

Consider the following example:

TABLE_PURCHASE_DETAIL

Customer ID	Store ID	Purchase Location
1	1	Los Angeles
1	3	San Francisco
2	1	Los Angeles
3	2	New York
4	3	San Francisco

This table has a composite primary key {Customer ID, Store ID}. The non-key attribute is Purchase Location. In this case, Purchase Location only depends on Store ID, which is a part of the primary key. Therefore, this table does not satisfy second normal form.

To bring this table to second normal form, we have to break the table into two tables as shown below.

TABLE_PURCHASE

Customer ID	Store ID
1	1
1	3
2	1
3	2
4	3

TABLE_STORE

Store ID	Purchase Location
1	Los Angeles
2	New York
3	San Francisco

Now, in the table TABLE_STORE, the column Purchase Location is fully dependent on the primary key of that table, which is Store ID.

Third normal form (3NF)

The third normal form is based on the transitive dependency. A relation is said to be in 3NF if it is first in 2NF and no transitive functional dependency exists.

Example-

Consider the following table:

TABLE_BOOK_DETAIL

Book ID	Genre ID	Genre Type	Price
1	1	Gardening	25.99
2	2	Sports	14.99
3	1	Gardening	10.00
4	3	Travel	12.99
5	2	Sports	17.99

In the above table, Book ID determines Genre ID, and Genre ID determines Genre Type. Therefore, Book ID determines Genre Type via Genre ID and we have transitive functional dependency, and this structure does not satisfy third normal form.

To bring this table to 3NF, we split the table into two as follows:

TABLE_BOOK

Book ID	Genre ID	Price
1	1	25.99
2	2	14.99
3	1	10.00
4	3	12.99
5	2	17.99

TABLE_GENRE

Genre ID	Genre Type
1	Gardening
2	Sports
3	Travel

Now all non-key attributes are fully functional dependent only on the primary key. In TABLE_BOOK, both Genre ID and Price are only dependent on Book ID. In TABLE_GENRE, Genre Type is only dependent on Genre ID.

Boyce-Codd Normal Form (BCNF)

The BCNF is based on the trivial dependency. A relation is said to be in BCNF if it is first in 3NF and no trivial functional dependency exists. That means that for every functional dependency X->Y, X should be the super key of the table.

Example-

Consider the following table, wherein employees work in more than one department.

emp_id	**emp_nationality**	**emp_dept**	**dept_type**	**dept_no_of_emp**
1001	Austrian	Production and planning	D001	200
1001	Austrian	stores	D001	250
1002	American	design and technical support	D134	100
1002	American	Purchasing department	D134	600

Functional dependencies in the table above:
emp_id -> emp_nationality
emp_dept -> {dept_type, dept_no_of_emp}
Candidate key: {emp_id, emp_dept}
The table is not in BCNF since neither emp_id nor emp_dept alone are keys.
To bring the table to BCNF the table is decomposed into three tables as follows-

emp_nationality table:

emp_id	emp_nationality
1001	Austrian
1002	American

emp_dept table:

emp_dept	dept_type	dept_no_of_emp
Production and planning	D001	200
stores	D001	250
design and technical support	D134	100
Purchasing department	D134	600

emp_dept_mapping table**:**

emp_id	emp_dept
1001	Production and planning
1001	stores
1002	design and technical support
1002	Purchasing department

Functional dependencies:
emp_id -> emp_nationality
emp_id -> emp_dept
emp_dept -> {dept_type, dept_no_of_emp}

Candidate keys:
For first table: emp_id
For second table: emp_dept
For third table: {emp_id, emp_dept}
Now, the tables emp_nationality, emp_dept, emp_dept_mapping are in BCNF.