Godavari Foundation's

Godavari College of Engineering, Jalgaon

Department of Computer

Lab Manual

Database System Laboratory

Practical No:- ____

Date:-_____

Name of Student:- _____

Class:-_____ **Roll No:-_____**

Title:

_____

_____

_____

_____

Aim: -

_____

_____

_____

_____

_____

_____

Software Requirement: _____

Hardware Requirement:- _____

Theory:-

Normalization

Normalization in DBMS is the process of effectively organizing data into multiple relational tables to minimize data redundancy.

A functional dependency defines the relationship between two attributes, typically between a prime attribute (primary key) and non-prime attributes. If for a table X containing attributes A and B, among which the attribute A is a primary key then the functional dependency is defined as:

Example for Functional Dependency:-

Employee Table

EMP_ID	EMP_NAME	EMP_CONTACT	MANAGER	Manager_Contact	DEPARTMENT
1	Oliver	88998899	Mr.X	33221133	Accounts
2	Barry	66776677	Mr.X	33221133	Accounts
3	Clark	44554455	Mr.Y	99887799	Sales
4	Bruce	22332233	Mr.X	33221133	Accounts
5	Kara	11001100	Mr.Y	99887799	Sales

1. Insertion Anomaly

Caused by updating the same set of repeated information again and again. This becomes a problem as the entries for a table increases with time.

2. Deletion Anomaly

It causes loss of data within the database due to its removal in some other related data set.

3. Updating Anomaly

In case of an update, it's very crucial to make sure that the given update happens for all the rows associated with the change. Even if a single row gets missed out it will lead to inconsistency of data.

Employee Table

EMP_ID	EMP_NAME	EMP_CONTACT	DEPARTMENT
1	Oliver	88998899	Accounts
2	Barry	66776677	Marketing
3	Clark	44554455	Sales
4	Bruce	22332233	Accounts
5	Kara	11001100	Sales

Manager Table

DEPARTMENT	MANAGER	Manager_Contact
Accounts	Mr.X	33221133
Marketing	Mr.X	33221133
Sales	Mr.Y	99887799
Accounts	Mr.Z	33221133
Sales	Mr.Y	99887799

For the normalization process to happen it is important to make sure that the data type of each data throughout an attribute is the same and there is no mix up within the data types.

Types of Normal Forms

1st Normal Form (1NF)

Data Atomicity is maintained. The data present in each attribute of a table cannot contain more than one value.

For each set of related data, a table is created and data set value in each is identified by a primary key applicable to the data set.

Example:-

Employee Table (Before 1NF)

EMP_ID	EMP_NAME	EMP_CONTACT
1	Oliver	88998899, 34534554
2	Barry	66776677
3	Clark	44554455
4	Bruce	22332233, 98798797
5	Kara	11001100

Employee Table (After 1NF)

EMP_ID	EMP_NAME	EMP_CONTACT
1	Oliver	88998899
1	Oliver	34534554
2	Barry	66776677
3	Clark	44554455
4	Bruce	98798797
4	Bruce	98798797
5	Kara	11001100

The multiple values from EMP_CONTACt made atomic based on the EMP_ID to satisfy the 1NF rules.

2nd Normal Form (2NF)

1. The table should be in its 1NF,
2. The table should not have any partial dependencies

Employee Table (Before 2NF)

EMP_ID	PROJECT_ID	MANAGER
1	23	Mr.X
1	67	Mr.Z
2	45	Mr.X
3	78	Mr.Y
3	23	Mr.X
4	23	Mr.X
5	78	Mr.Y
5	67	Mr.Z

- If manager details are to be fetched for an employee, multiple results are returned when searched with EMP_ID, in order to fetch one result, EMP_ID and PROJECT_ID together are considered as the Candidate Keys.
- Here the manager depends on PROJECT_ID and not on EMP_ID, this creates a partial dependency.
- There are multiple ways to eliminate this partial dependency and reduce the table to its 2nd normal form.

After 2NF

Employee Table

EMP_ID	PROJECT_ID
1	23
1	67
2	45
3	78
3	23
4	23
5	78
5	67

Project Table

PROJECT_ID	PROJECT	MANAGER
23	Extract	Mr.X
67	Load	Mr.Z
45	Extract	Mr.X
78	Transform	Mr.Y
23	Extract	Mr.X
23	Extract	Mr.X
78	Transform	Mr.Y
67	Load	Mr.Z

3rd Normal Form (3NF)

1. The table should be in its 2NF, and
2. The table should not have any transitive dependencies

Employee Table (Before 3NF)

EMP_ID	PROJECT_ID	PERFORMANCE_SCORE	HIKE($)
1	23	50	160
1	67	30	120
2	45	50	160
3	78	60	170
3	23	30	120
4	23	20	110
5	78	60	170
5	67	20	110

- The PERFORMANCE_SCORE depends on both employee and project that he is associated, but the last column HIKE depends on PERFORMANCE_SCORE.
- The hike changes with performance and here the attribute performance score is not a primary key. This forms a transitive dependency
- To eliminate the transitive dependency created, and to satisfy the 3NF, the table is broken as:

After 3NF

Employee Table

EMP_ID	PROJECT_ID	PERFORMANCE_SCORE
1	23	50
1	67	30
2	45	50
3	78	60
3	23	30
4	23	20
5	78	60
5	67	20

Performance Table

PERFORMANCE_SCORE	HIKE($)
20	110
30	120
50	160
60	170

Boyce-Codd Normal Form BCNF or 3.5 NF

1. The table should be in its 3 NF form
2. For any dependency, A à B, A should be a super key i.e. A cannot be a non-prime attribute when B is a prime attribute.

Employee Table (Before BCNF)

EMP_ID	PROJECT_ID	DEPARTMENT
1	23	Accounts
1	67	Sales
2	45	HR
3	78	Accounts
3	23	Sales
4	23	HR
5	78	Sales
5	67	HR

- EMP_ID + PROJECT_ID → DEPARTMENT
- (Candidate keys)
- DEPARTMENT which is not a super key is dependent only on PROJECT_ID but not dependent on EMP_ID
- Department → Project_ID
- (Non-prime attribute) (prime attribute)

To make this table satisfy BCNF, we need to break the table as:

After BCNF

Employee Table

EMP_ID	DEPARTMENT_ID
1	D123
1	D456
2	D789
3	D122
3	D455
4	D789
5	D454
5	D787

Department Table

DEPARTMENT_ID	DEPARTMENT	PROJECT	PROJECT_ID
D123	Accounts	Extract	23
D456	Sales	Load	67
D789	HR	Extract	45
D122	Accounts	Transform	78
D455	Sales	Extract	23
D789	HR	Extract	23
D454	Sales	Transform	78
D787	HR	Load	67

Conclusion:-

_____
_____
_____
_____