Assignment - 3

1) What is the need of normalization? Explain 1NF, 2NF, 3NF with examples.

Normalization is a database design technique used to organize data in a relational database in order to reduce redundancy and dependency. The main goals of normalization are to eliminate data anomalies and ensure data integrity. It involves breaking down large tables into smaller, more manageable tables and establishing relationships between them.

1) First normal form (1NF):

- Definition: A table is in 1NF if it contains only atomic (indivisible) values, and there are no repeating groups or arrays.

Example: Suppose you have a table that stores information about students and their courses. A non-1NF table might look like this:

Student_ID	Courses
1	Math, Physics
2	Chemistry, Biology, Physics

To bring this table into 1NF, you would split the repeating groups into separate.

Student_ID	Course
1	Math
1	Physics
2	Chemistry
2	Biology
2	Physics

2. Second Normal form (2NF):

* Definition: A table in 2NF if it is 1NF and all non-prime attributes are fully functionally dependent on the primary key.

Example: Consider a table that combines information about employees and projects worked on:

Employee_ID	Project_ID	Project_Name	Employee_Name
1	101	Project-A	Alice
1	102	Project-B	Alice
2	101	Project-A	Bob

To achieve 2NF, we break it into two tables

Employee table

Employee_ID	Employee_Name
1	Alice
2	Bob

Project table

Project-ID	Project-Name
101	Project-A
102	Project-B

Employee - project table.

Employee-ID	Project-ID
1	101
1	102
2	101

3) Third Normal Form (3NF):

* Definition:- A table is in 3NF and all transitive dependence are removed.

Example:- Suppose you have a table that stores information about courses and the instructors who teach them

Course-ID	Instructor	Department
101	Smith	physics
102	Jonson	Chemistry
103	Smith	physics

To achieve 3nf. you create separate tables for courses, instructors and departments.

Course-ID	Instructor-ID
101	1
102	2
103	1

Instructor table

Instructor-ID	Instructor
1	Smith
2	Jonson

Department Table

Instructor-ID	Department
1	physics
2	Chemistry

2. Consider the Universal relation R = {A, B, C, D, E, F, G, H, I, J}
and the set of functional dependencies F = {{A, B} → {C}, {A} → {D, E}
{B} → {F}, {F} → {G, H}, {D} → {I, J}. What is the key for R?
Decompose R into 2 NF and then 3NF relations

* Finding the key for R'

* Start with each individual attribute

Closure ({A}) = {A, D, E, I, J}
Closure ({B}) = {B, F, G, H}
Closure ({C}) = {C}
Closure ({D}) = {D, I, J}
Closure ({E}) = {E}
Closure ({F}) = {F, G, H}
Closure ({G}) = {G}
Closure ({H}) = {H}
Closure ({I}) = {I}
Closure ({J}) = {J}

→ Combine attributes to form potential keys

{A, B} closure = {A, B, C, D, E, F, G, H, I, J} (Thus is a superkey)

∴ So {A, B} is a key for relation R

Decomposing R into 2NF:

The key for R is {A, B} and the given functional dependencies are

{A, B} → {C}
{A} → {D, E}
{B} → {F}
{F} → {G, H}
{D} → {I, J}

R1 (A, B, C) ⇒ {A, B} → {C}

R2 (A, D, E) ⇒ {A} → {D, E}

R3 (B, F, G, H) ⇒ {B} → {F} and {F} → {G, H}

R4 (D, I, J) ⇒ {D} → {I, J}

Decomposing R into 3NF:

A relation is in 3NF if is in 2NF and all transitive dependencies are removed

→ Looking at the 2NF relations, there are no transitive dependencies present.
Therefore the relation R1, R2, R3 and R4 are already in 3NF.
so further decomposition is needed for 3NF

3] Consider the following relations:-

CAR - SALES (Car-no, Date-Sold, Salesman-no, Commission %, Discount)

(Assume a car be sold by multiple salesmen and hence Primary key is {Car-no, Salesman-no})

Additional dependencies are: Date-Sold → Discount-amt and Salesman-no → Commission%

i] Is the relation in 1NF, 2NF, 3NF? Why or why not?

ii] How would you normalize this completely?

→ Normalization Analysis:

Let's analyze the given relation Car_Sale:

✱1. Is this relation in 1NF?

Yes, the relation is in 1NF because each attribute contains atomic values, and there are no repeating groups.

✱2. Is this relation in 2NF?

The primary key is {Car-no, Salesman-no} and there are no partial dependencies on this composite key, Therefore, the relation is in 2NF

✱3. Is this relation in 3NF?

There are two additional dependencies: Date-Sold → Discount-amt and Salesman-no → Commission-% these dependencies are not dependent on the whole primary key.

✱ Transitive Dependency 1: Date-Sold → Discount-amt (Cost directly related to the primary key)

✱ Transitive Dependency 2: Salesman-no → Commission% (Cost directly related to the primary key)

⟹ Therefore, the relation is not in 3NF

ii]. Normalization steps✱

To normalize the relation into 3NF, we need to address the transitive dependencies. We can create separate relations for each set of attributes involved in these dependencies.

✱ Normalized Relations:

1) ✱ Car_Sale (Car-no, Date-Sold, Salesman-no, Discount)✱

- This relation includes attributes related to the primary key and the transitive dependency Date-Sold → Discount.

2) ✱ Discount-Amount (Date-Sold, Discount-amt)✱

- A separate relation for the transitive dependency Date-Sold → Discount-amt.

3. Salesman- Commission (Salesman- no, Commission - %)
- A Separate relation for the transitive dependency Salesman-no Commission%.

This normalization ensures that each relation has a clear purpose and avoids transitive dependencies, making the structure more robust and adherent to the principle of 3NF.

4) Define Boyce-Codd normal form. How does it differ from 3NF? Why is it considered a stronger form of 3NF?

Boyce-Codd Normal form (BCNF)

→ Boyce-Codd normal form is a higher normal form than the third normal form (3NF) and addresses certain situations where 3NF might not be sufficient to eliminate all potential problems related to functional dependencies.

→ Differences from 3NF

while both BCNF and 3NF aim to eliminate certain of redundancy and dependency issues. BCNF is more stringent than 3NF in handling functional dependencies.

1. 3NF
- In 3NF, the requirements is that there should be no transitive dependencies and non-prime attributes should not depend on other non-prime attributes.
- 3NF allows a specific type of dependency known as a transitive dependency, which is when a non-prime attribute depends on another non-prime attribute.

2. BCNF
- BCNF, on the other hand extends the requirements of 3NF by eliminating a specific kind of dependency known as a non-trivial functional dependency.
- In BCNF, for every non-trivial functional dependency (X right arrow Y) X must be superkey, this means that every determinant (X) should be a candidate key, ensuring that non-prime attributes are fully functionally dependent on superkeys.

* why BCNF is considered stronger.

BCNF is considered stronger than 3NF because it imposes a stricter condition on functional dependencies. In BCNF the criteria ensure that there are no situations where non-prime attributes depend on other non-prime attributes, even indirectly through transitive dependencies.

The Shifting conditions of BCNF help to eliminate relational types of redundancy and dependency issues that might not be addressed by 3NF alone, while ensuring BCNF might lead to more relations in the database. It provides a higher degree of data integrity and ensures that anomalies related to functional dependencies are minimized

5) State the informal guidelines for relational schemas design. Illustrate how violation of these guidelines may be harmful.

1) Avoid Redundancy:-

* Guideline: Ensure that data is not duplicated unnecessarily

* Harm of violation: Redundant data can lead to inconsistencies and increase the chances of update anomalies. It may also consume more storage space than necessary

2) Ensure Data Integrity:

* Guideline: Define appropriate primary and foreign key to maintain referential integrity

* Harm of violation: without proper integrity constraints, it becomes possible to have orphaned records or invalid references leading to data corruption and inaccuracies

3. Normalize tables:

+ Guideline: Apply normalization techniques to minimize data redundancy and dependency.

* Harm of violation: Denormalized database might suffer from issues such as insertion, update and deletion anomalies. It could lead to inefficient queries and increased storage requirements

4. Choose Appropriate data types:

+ Guideline: Select the correct data types for each attribute to ensure efficient storage and retrieval

+ Harm of violation: Inappropriate data types can lead to wasted storage, slower query performance and potential loss of precision or accuracy in the stored data.

5. Define clear and Consistent Naming Conventions:

* Guideline: Use meaningful and consistent names for tables, columns and other database objects

** Harm of violation: Inconsistent or unclear naming conventions can make it difficult for developers and administrators to understand the structure of the database, leading to confusion and errors.

6. Consider performance implications: Design with workload and query patterns in mind for better performance

7. Document the schema: provide documentation for clarity and ease of maintenance.

8. Consider security and privacy: implement access controls and encryption to protect sensitive data