NORMALIZATION ASSESSMENT

Name: Gaurav Nandgopal Deshmukh

USN: 72232433C

Q.1. What's normalization, and how does it differ from denormalization?

Ans: Normalization is the process of organizing data into smaller, related tables to eliminate redundancy and ensure data consistency. It helps in efficient data storage and avoids anomalies.

Denormalization does the opposite—it combines tables to speed up data retrieval, often at the cost of some redundancy.

Q.2. What are the various types of normalization levels or normalization forms?

Ans: Normalization Levels:

• 1NF: Each row is a unique value, ensures atomicity and helps provide a primary key.

• 2NF: 1NF + removes partial dependency, non-prime attributes depend entirely on the primary key.

• 3NF: 2NF + removes transitive dependency, non-prime attributes depend directly on the primary key.

• BCNF: 3NF + removes anomalies caused by composite keys or overlapping candidate keys, every determinant is a candidate key

Q.3. What are keys in normalization?

Ans: Keys are attributes used to uniquely identify rows in a table.

Types include: -

1) Primary Key: Unique and non-null
2) Candidate Key: Any field that can be a primary key
3) Composite Key: A key made from multiple columns

4) Foreign Key: Links to a key in another table
5) Super Key: A set of attributes that can uniquely identify a record

Q.4. What is denormalization, and when might it be used in database design?

Ans: Denormalization is a technique where normalized tables are combined to reduce the complexity of joins and improve performance.

It's typically used in reporting systems or scenarios where fast data access is more important than eliminating redundancy.

Q.5. Find the highest normal form of a relation R(A,B,C,D,E) with FD set as {BC→D, AC→BE, B→E}.

Ans: To find the highest normal form, we identify AC as a candidate key since it can derive all attributes.

But the dependency B→E is a partial dependency, as B is only part of the candidate key AC. This violates 2NF, so the relation is in 1NF but not in 2NF.

Hence, the highest normal form is 1NF.

Q.6. A shaving set company sells 4 different types of razors, Elegance, Smooth, Soft and Executive. Elegance sells at Rs. 48, Smooth at Rs. 63, Soft at Rs. 78 and Executive at Rs. 173 per piece. The table below shows the numbers of each razor sold in each quarter of a year.

Ans: To find the product with the highest revenue contribution, we multiply the total units sold in the year by their respective prices.

Elegance: (27300 + 25322 + 24872 + 23167) × 48 = 100661 × 48 = Rs. 4831728

Smooth: (20069 + 19922 + 22429 + 18729) × 63 = 81149 × 63 = Rs. 5112387

Soft: (17612 + 18454 + 19545 + 16956) × 78 = 72567 × 78 = Rs. 5661126

Executive: (9909 + 9942 + 10243 + 10019) × 173 = 40113 × 173 = Rs. 6940449

So, the Executive razor contributes the greatest revenue to the company during that year.

Q.7. Suppose you are given a relation R = (A, B, C, D, E) with the following functional dependencies: {CE → D, D → B, B → A}

a. Find all candidate keys.

b. Identify the best normal form that R satisfies (1NF, 2NF, 3NF, or BCNF).

c. If the relation is not in BCNF, decompose it until it becomes BCNF. At each step, identify a new relation, decompose and re-compute the keys and the normal forms they satisfy.

Ans: a. Find all candidate keys.

To find the candidate key, compute closure of CE. CE+ = {C, E} → D → B → with B → A, and we already have B and C, so A is added. So CE+ = {A, B, C, D, E}, which means CE is a candidate Key.

b. Identify the best normal form that R satisfies (1NF, 2NF, 3NF, or BCNF).

- CE is the only candidate key.
- R is in 1NF (assumed).
- No partial dependencies → R is in 2NF.
- Has transitive dependencies (D → B, B → A) → violates 3NF.
- So, **R is in 2NF only.**

c. If the relation is not in BCNF, decompose it until it becomes BCNF. At each step, identify a new relation, decompose and re-compute the keys and the normal forms they satisfy.

For R = (A,B,C,D,E) with FDs: {CE→D, D→B, C→A}

1. Violation: C→A (C is not a superkey)

    a. Split into: R1(C,A) and R2(B,C,D,E)
2. Violation in R2: D→B (D is not a superkey)
    a. Split R2 into: R21(D,B) and R22(C,D,E)
3. R22 has CE→D which is fine (CE is a key)

Final result:

- R1(C,A) with key C
- R21(D,B) with key D
- R22(C,D,E) with key CE

All relations are now in BCNF.

Q.8. Suppose you are given a relation R = (A, B, C, D, E) with the following functional dependencies: BD → E , A → C

a. Show that the decomposition into R1 = (A, B, C) and R2 = (D, E) is lossy. You can show using any method. The suggestion is to show how spurious tuples result from this decomposition with respect to the table below:

A	**B**	**C**	**D**	**E**
1	**2**	**3**	**5**	**5**
1	**8**	**3**	**4**	**4**

Ans: Step 1: Decompose the original relation R into R1 and R2:

R1 (A,B,C):

A	B	C
1	2	3
1	8	3

R2 (D,E):

D	E
5	5
4	4

Step 2: Perform a natural join of R1 and R2: Since R1 and R2 don't share any attributes, the join becomes a Cartesian product:

R1 ⋈ R2:

A	B	C	D	E
1	2	3	5	5
1	2	3	4	4
1	8	3	5	5
1	8	3	4	4

Step 3: Compare with original relation: The result has 4 tuples, but the original relation only had 2 tuples. The following are spurious tuples not in the original relation:

- (1, 2, 3, 4, 4)
- (1, 8, 3, 5, 5)

This proves the decomposition is lossy because the join operation produced additional tuples that weren't in the original relation.

Q.9. Explain a process of transforming a table into BCNF.

Ans:

1. Identify all functional dependencies in the relation
2. Find the candidate key(s)
3. Check each FD for BCNF violations (a violation occurs when a determinant is not a superkey)
4. For each violation, decompose the relation:
5. Create a relation with the determinant and its dependent attributes
6. Create another relation with the determinant and remaining attributes
7. Check each new relation for BCNF violations
8. Repeat the decomposition process until all relations are in BCNF

Q.10. What is a lossless-join decomposition, and why is it important in normalization?

Ans: A decomposition is lossless-join when the original relation can be reconstructed from the decomposed relations without losing or gaining any tuples.

Why important:

- Preserves all information in the original relation
- Ensures data integrity during decomposition
- Allows data to be retrieved accurately when relations are joined
- Essential property for normalization to be useful in practice.