UNIT-3

Database Design -

① Normalization - It is the splitting of data into several tables that will be connected to each other based on the data within them.

It provide flexibility, minimum redundancy and ensure data integrity.

Problems using a single table for all information -

Insert anomaly, Update anomaly, Delete anomaly.

Normalization is a process to eliminate these problems.

② Normal Forms -

First Normal Form (1NF) - A relation in which intersection of each row and column contains one and only one value. All attributes are atomic.

Second Normal Form (2NF) - A relation is in 2NF if it is in 1NF and if all non-prime attribute are fully functionally dependent on the primary key.

Third Normal Form (3NF) - A relation is in 3NF if it is in 2NF and every non-key attribute is non transitively dependent on the primary key.

Boyce Code Normal Form (BCNF) - A relation is in BCNF if it is in 3NF and all the determinants are candidate keys.

Fourth Normal Form (4NF) - A relation is in 4NF if it is in BCNF and it has no multivalue dependency.

Fifth Normal Form (5NF) - Also known as project-join normal form (PJ/NF)

A relation R is in 5NF if and only if every non-trivial join dependency that holds for R is implied by the candidate key of R.

③ Function Dependency -

The association or dependence between attributes is called functional dependency. It is denoted as $X \rightarrow Y$ implies Y is said to be functionally dependent on X.

Prime attribute - If an attribute is a part of candidate key of the ~~relation~~ relation (primary key). Otherwise non-prime attribute.

④ Types of functional dependencies –

Full functional dependency – Given a relational schema R and functional dependency $X \to Y$. Y is full functional dependent on X if there is no Z where Z is a proper set of X such that $Z \to Y$.

Partial functional dependency – It is a functional dependency in which one or more non-key attributes functionally dependent on part of the primary key. $AB \to C$

Trivial and Non-Trivial functional dependency – $AB \to A$

Functional dependency are trivial if and only if the right hand side is a subset of the left hand side otherwise non-trivial.

Transitive functional dependency –

A transitive dependency in a relation is a functional dependency between two or more non-key attributes. $A \to B$, $B \to C$, $A \to C$

Multivalue dependency – $A \to\to B$

Let R be a relation and let A, B and C be subset of the attributes of R. Then, we say that B is multi dependent on A if and only if in every legal value of R, the set of B values matching a given AC value pairs depends on the A value and is independent of the C value.

⑤ Decomposition –

To decompose a relation schema that has many attributes into several schemas with fewer attributes.

lossless Join decomposition – If we decompose a relation R into several relations, and if we join those relations (R') and if we get exact or original values of relation R that means $R' = R$

lossy-Join decomposition – If $R' \neq R$ that means it is not a lossless-Join decomposition.

Dependency-preservation decomposition – When an update is made to the database, the system should be able to check that the update

will not create an illegal relation - that is, one that does not satisfy all the given functional dependencies.

In other terms, an update should be made when it satisfy all the given functional dependencies on a relation is known as Dependency-Preservation decomposition.

⑥ Problem with dangling tuples and null valued -

Dangling tuples - Let $r_1(R_1), r_2(R_2), \ldots, r_n(R_n)$ be a set of relations. A tuple t of relation r_i is a dangling tuple if t is not in the relation.

$$\Pi_{R_i}(r_1 \bowtie r_2 \bowtie \ldots \bowtie r_n)$$

They represent incomplete information.

Null valued -

In general, whenever a relational database schema is designed where two or more relations are interrelated via foreign keys, particular care must be devoted to watching for potential null values in foreign keys. This can cause unexpected loss of information in queries that involves join on that foreign key.

Query Optimization -

⑦ It is the process of selecting the most efficient query-evaluation plan from among the many strategies usually possible for processing a given query, especially if the query is complex.

① Steps of optimization -

(1) Parsing and translation → convert the query in usable form for query processing machine

(2) Optimization

(3) Evaluation

② Algorithms to implement various operations of relational algebra -

→ Select operation -

(A1) Linear search (=)

(A2) Binary search (>, ≥, ≤, <)

(A3) Complex selections -
- Conjunction ($\wedge$)
- Disjunction ($\vee$)

(A4) Negation ($<>$)

→ Join operations -
(A1) Nested loop join.
(A2) Index Nested-loop join.
(A3) Sort-Merge Join
(A4) Hash Join

→ Project operation -
(A1) Sorting
(A2) Hashing.

Optimization Methods -

① Heuristic based optimization -
→ The main heuristic is to apply first the operations that reduce the size of intermediate results.
→ Perform SELECT and PROJECT operations as early as possible to reduce number of tuples and number of attributes respectively.
→ The SELECT and JOIN operations that are most restrictive should be executed before other similar operations.

② Cost Estimation based optimization -
Estimate and compare the costs of executing a query using different execution strategies and choose the strategy with the lowest cost estimate.
Issues - Cost function, Number of execution strategies to be considered.