# DIGITAL ASSIGNMENT

Subject Name : Database Technologies

Subject Code : ITA5008

Name: Kamran Ansari

Reg No : 22MCA0223

Q.1. Explain all First Normal Form, Second Normal Form, Third Normal Form and Boyce-Codd Normal Form (BCNF) with suitable examples.

Ans →

Database Normalization is the technique of organizing data in the database. Normalisation is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like Insertion, Update and Deletion Anamoly.

Normalisation rules are divided into four normal forms-

i) First Normal Form

ii) Second Normal Form

iii) Third Normal Form

iv) BCNF

# First Normal Form

A database is said to be in first normal form if it satisfies the following rules -

i) Single Valued Attribute (Atomic values)
Each column of table should be single valued which means they should not contain multiple values.

ii) Values in a column should be of same domain
In each column the values stored must be of the same kind or type

iii) Attributes should have Unique Names

iv) Order in which data is stored, does not matter.

Example -

| roll_no, name | subject |
|---|---|
| 101, N1 | C1, C2 |
| 103, N3 | C3 |
| 102, N2 | C4 |

Converting this relation to 1NF,

| roll_no | name | subject |
|---|---|---|
| 101 | N1 | C1 |
| 103 | N3 | C3 |
| 102 | N2 | C4 |
| 101 | N1 | C2 |

# Second Normal Form

For a table to be in Second Normal Form, it must satisfy two conditions -

i) Relation should be in First Normal Form.

ii) There should be no Partial Dependency.

Where Partial Dependency is defined as the functional dependency, where an attribute in a table depends on only a part of the primary key and not the whole key.

Example -

Score

| score_id | stud_id | sub_id | marks | teacher |
|---|---|---|---|---|
| 1 | 2201 | 5003 | 65 | T1 |
| 2 | 2202 | 5005 | 75 | T3 |
| 3 | 2203 | 5007 | 60 | T2 |

The primary key of this relation is a composite key (stud_id, sub_id) but attribute teacher only depends upon sub_id, this is partial dependency.

Converting to 2NF,

Subject

| sub_id | sub_name | teacher |
|---|---|---|

Score

| score_id | student_id | sub_id | marks |
|---|---|---|---|

# Third Normal Form

For a table to be in third normal form the following conditions should satisfy,

i. Relation should be in Second Normal Form.

ii. There should be no Transitive Dependency.

When a non-prime attribute depends on other non-prime attributes rather than depending upon the prime attributes or primary key.

Example -

| Stud-no | stud_state | stud_country | stud-name |
|---|---|---|---|

Here stud_country depends on stud-state

stud_state ⟶ stud_country

And stud_no ⟶ stud_state

This is transitive dependency.

After converting to 3NF,

Student

| Stud-no | stud-name | Stud-State |
|---|---|---|

State-country

| state | country |
|---|---|

# Boyle Codd Normal Form

For a table to be in BCNF, it must satisfy following conditions -

i) It should be in Third Normal form

ii) For any dependency $A \rightarrow B$, A should be a super key. For a dependency $A \rightarrow B$, A cannot be a non-prime attribute, if B is a prime attribute

Example,

| student_id | subject | professor |
|---|---|---|

Here professor → subject,

professor is non prime attribute while subject is prime attribute.

After converting to BCNF,

Student

| student_id | p_id |
|---|---|

Professor

| p_id | professor | subject |
|---|---|---|

Q.2. Explain about B+ tree in DBMS with example.

Ans → i) B+ trees are an enchancement to B trees that enable faster insertion, deletion, and search operations.

ii) It is a more advanced self-balancing tree. In this, all the values are present at the leaf level. The internal nodes store just the indices.

iii) Each leaf is at the same height & all leaf nodes have links to the other leaf nodes.

iv) The root node always has a minimum of two children.

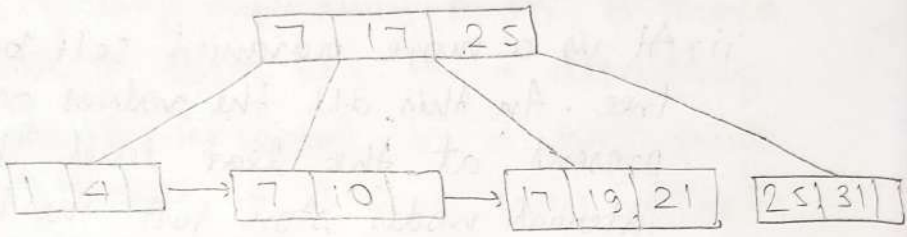
v) Compared to B trees, B+ trees are more convenient, efficient and easy to operate.

Properties of B+ Trees

1) All data is stored in leaf nodes.
2) The internal nodes store just the indices.
3) All leaf nodes have links to the other leaf nodes.
4) Root node has minimum of two children.
5) Each node except root can have maximum of m children and minimum of m/2 children.

6. Each node can contain a maximum of m-1 keys and a minimum of $\lceil m/2 \rceil - 1$ keys.

Example,

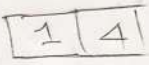



## Insertion in B+ Tree

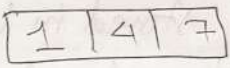
Sequence → 1, 4, 7, 10, 17

Insert 1

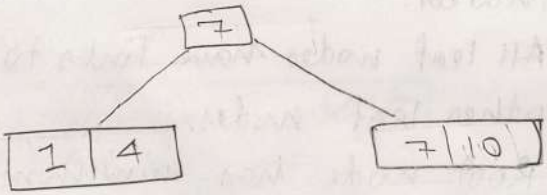

Insert 4



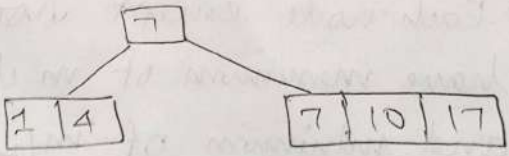

Insert 7




Insert 10

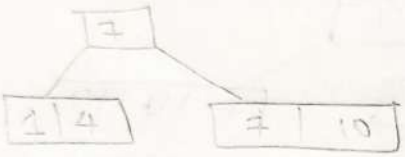



Insert 17



## Deletion in B+ Tree


Delete 17




Q.3. Explain parallel algorithms for sorting in parallel DBMS?

Ans → An algorithm is a sequence of steps followed to solve a problem. It takes input from the user and after some computation, produces an output.

In Parallel Algorithms the problem is divided into sub-problems and are executed in parallel to get individual outputs. Later on, these individual outputs are combined together to get the final desired output.

Parallel Algorithms require parallel machines or parallel computers. It can be executed simultaneously on many different processing devices and then combined together to get the correct result.

Steps for developing a parallel algorithm -

1. Decompose the problem into tasks that can be executed concurrently.
2. A problem may be decomposed in many ways.
3. Tasks maybe same, different or even indeterminate sizes.
4. A decomposition can be shown in the form of a directed graph with nodes corresponding to tasks and edges indicating the result of one task is required for processing the next. Such graph is called a task dependency graph.

Parallel Sorting Algorithms,

1. Parallel Bubble Sort
2. Parallel Merge Sort
3. Bitonic Sort

## Parallel Bubble Sort

In contrast to conventional bubble sort, which cycles through the list, compares consecutive elements and swaps them if necessary. Parrallel bubble sort works in two phases, even and odd.

Which is why it is also called odd-even bubble sort.

1) Even processes exchange values with right neighbours.

2) Odd processes exchange values with left neighbours.

## Parallel Merge Sort

Merge sort, which is a divide and conquer algorithm, sorts a vector by recursively dividing it in two parts, separately merging them and finally merging them.

In Parallel Merge Sort, collects sorted list onto one processor, merges elements as they come together. Has a simple tree structure. Parallelism is limited when near the root.

## Bitonic MergeSort

The basis for bitonic mergesort is the bitonic sequence, a list having specific properties that are exploited by the sorting algorithm.

A sequence is considered bitonic if it contains two sequences, one increasing and one decreasing

$a_1 < a_2 < \ldots < a_i > a_{i+1} > \ldots > a_n$

If we compare and exchange with elements $a_i$ and $a_{i+n/2}$ for all $i$ $(0 \leq i \leq n/2)$ in a sequence of $n$, we obtain two bitonic sequence in which all the values in one sequence are smaller than the values of the other.

Also, all values in the left sequence are less than all the values in the right sequence.

Hence, if we apply recursively these compare and exchange operations to a given bitonic sequence, we will get a sorted sequence.