| S. No. | Date | Title |
|---|---|---|
| | | |

* Memory model, in computing, describes the interaction of threads through memory and their shared use of the data
→ allows compiler to perform imp. optimizations

— × —— × ——→ COMPLETELY DIFFERENT —×—

(Scott Muller - D&MS).

ADA is studied on RAM (Random Access machine) because in memory, the variables are given memory in random.

——→ × —— × ——→ × ——

Main memory makes use of Data Structures

MBR – Master Boot Recorder → Keeps tracks of sectors, tracks record of change in file when the system is stopped abruptly, the MBR shrinks the data that has been changed when system is started again, MBR is activated for previous data

## ADVANCES DATABASES :

Indexing of DB is similar to MBR for hard-disk.

### * INDEXING.

To reach a particular data in large volume database we make use of indexing.

There are two types.

1. Dense 2. Sparse.

There can be single level indexing & multiple level indexing.

Clusters are measuring units. Clusters can be of variable sizes. Data in databases are stored in these clusters.

### * DENSE INDEXING.

Consider a student table consisting of student data.

| R.N | Name | % | Grade | = 35 B. |
|---|---|---|---|---|
| 10 | 15 | 5 | 5 | |

Now on increase of records size of table increases, thus no. of clusters used increase. When a data is to be found, it takes a larger amount of time.

Thus, if we ~~increase~~ make a ₹ index table with R.N and record pointer, this decreases size of table and clusters are used to store this. This it decreases time taken to search data.

This is dense index.

classmate

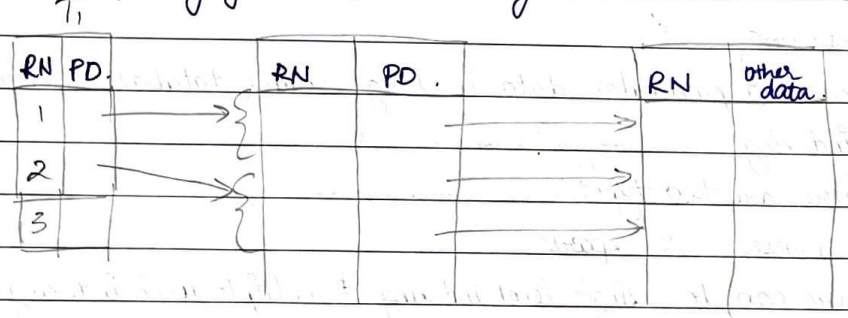
* DIVIDE & CONQUER is the approach that is mainly used.

* M way search – multilevel search tree. (M → number of branches that root node can have. (m-1) keys are present.)

* SPARSE INDEXING / MULTILEVEL INDEXING.

Here levels are added, thus called high level indexing, as it shrinks the level.

Based on complexity of data, number of levels can be increased.




To implement indexing, multilevel search tree are used.

Now, mainly B and $B^+$ tree are used.

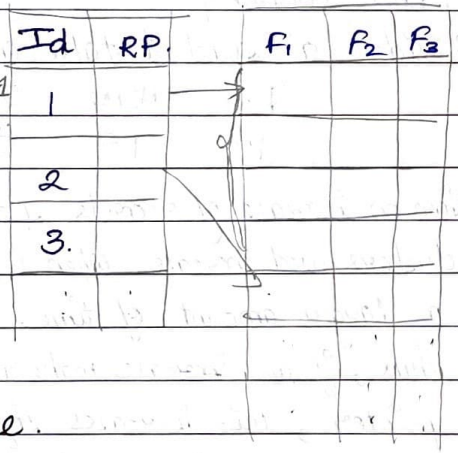
* B-TREE.

① Maximum m childs. (m-1 keys)

② Minimum m/2 children (m/2 - 1 keys)

③ All leaf nodes are at same height. ~~(To achieve this~~

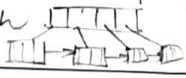
④ Insertion is done by Bottom up approach.




B-tree is a self balancing tree.

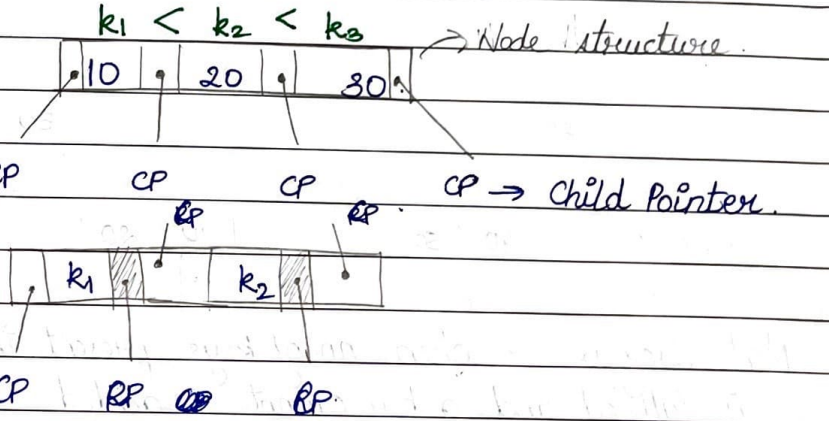


$B^+$ Tree – makes use of both tree & linked list. Here the last nodes of a branch is connected with first node of next branch (by linked list concept) $B^+$ tree is not a graph as it has backtracking approach.



eg. 10, 20, 30, 40, 50, 60.

m = 4.




Record Pointer – To point to the next record in the group of key pointed by the key.

Split Node – When the number of keys in a child is fulfiled and a new element has to be inserted, we split the node and shift one element to the top.

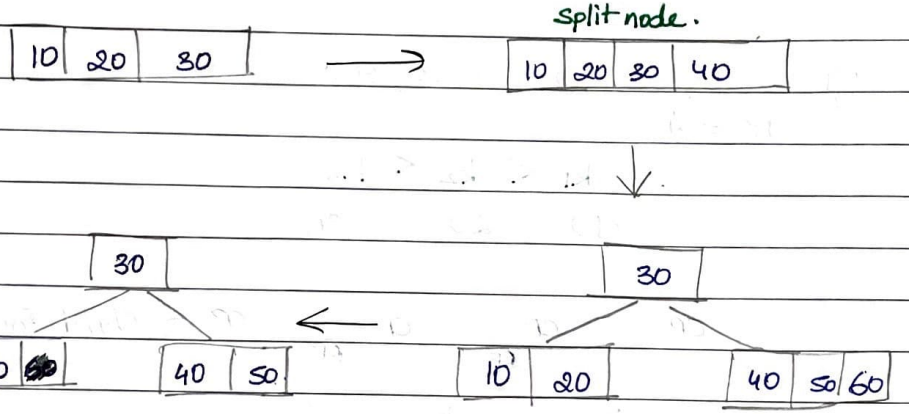
The shifting is done on the basis of ~~bias~~

Left bias tree → number of ~~children~~ keys should be more on left

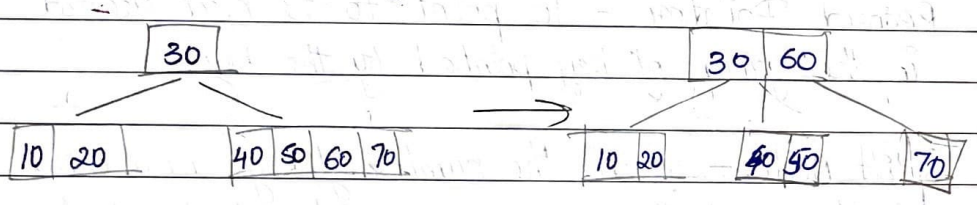
Right bias tree → number of keys should be more on right.

Having m/2 ~~trees~~ ~~is roots~~ keys in the root node is not necessary.

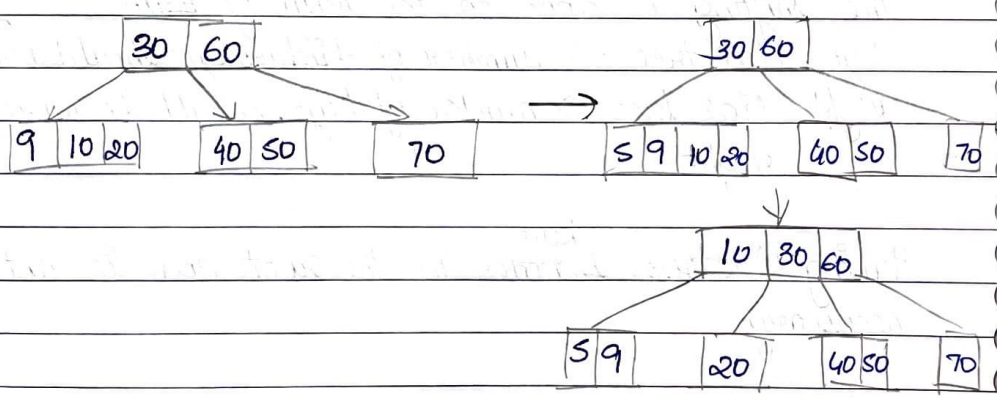
* TRY TO HAVE MAXIMUM RECORD POINTERS AT THE LOWEST LAYER. — $B^+$ TREES.




Node overflow — when no. of keys present in a node is fulfilled and a key cannot be added.




Condition → key 1 < key 2 should always be satisfied.



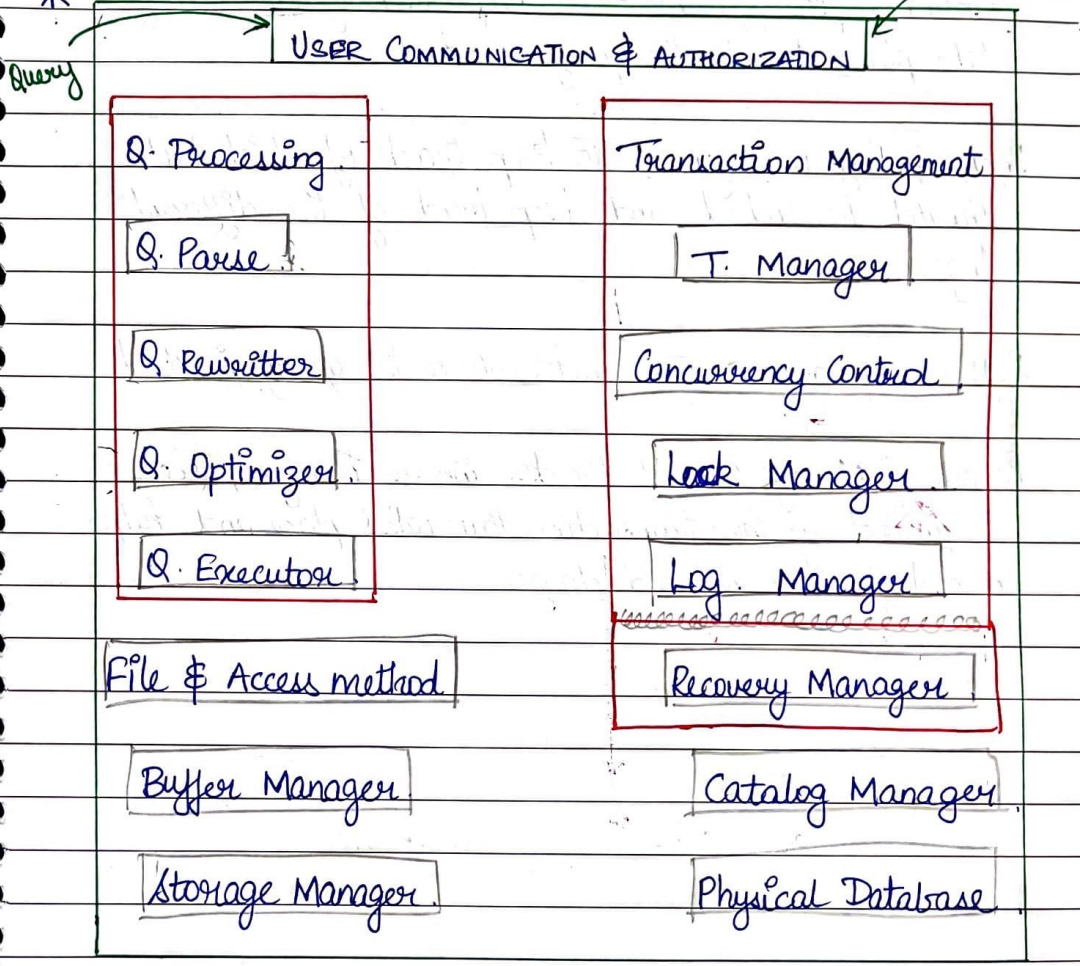

classmate

Vertical → Sparse
Horizontal → Dense.

List can be ordered or unordered.

DISADVANTAGE → to reach next node we have to ~~move~~ go to and from.

* DATABASE MANAGEMENT SYSTEM ARCHITECTURE.




Q. Rewriter — Convert it to relational algebra.

classmate

Q. Optimizer → find the best way to ~~optimize~~ execute the query.

Transaction Management → To keep track of what changes take place due to the query executed.

Lock & Log Manager → To keep track of which tables should be locked and keep track of the different logs.

Buffer Manager → It tracks where the query is stuck.

Recovery Manager → It tracks where changes are required in the transaction that takes place and helps in making the required changes.

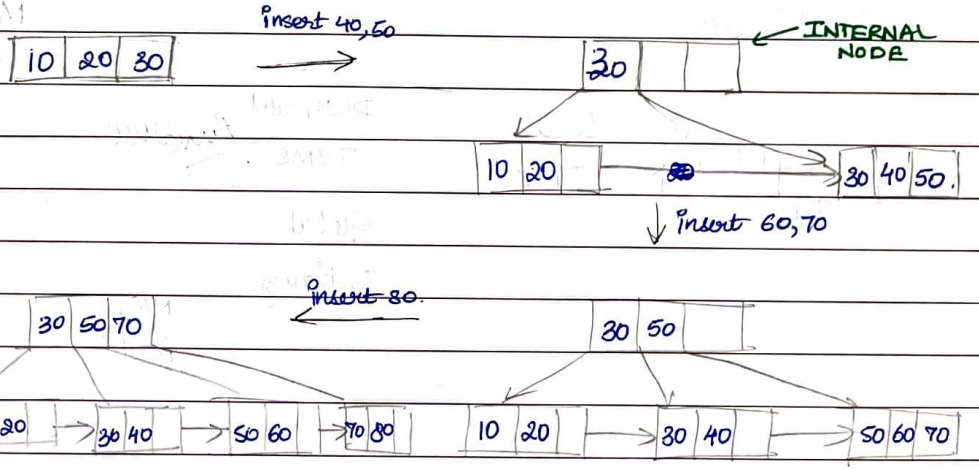
Temporal Database → Time Bounded.
Mobile Database → for devices.
NoSQL → Q. Processing is not done (except Q. optimization)
XML → meta data (Keep your own data)
Distribute Databases →



## * $B^+$ TREES.

→ # $B^+$ Trees are similar to B Trees. Difference is that the record pointers in $B^+$ Trees only come from ~~#~~ leaf nodes.
→ The key in internal nodes have a copy in the leaf nodes.
→ Each leaf node is connected to the next node in linked list format.
For creation & insertion of keys we will follow the same rules as B Trees.

eg. 10, 20, 30, 40, 50, 60, 70, 80 ; m = 4.

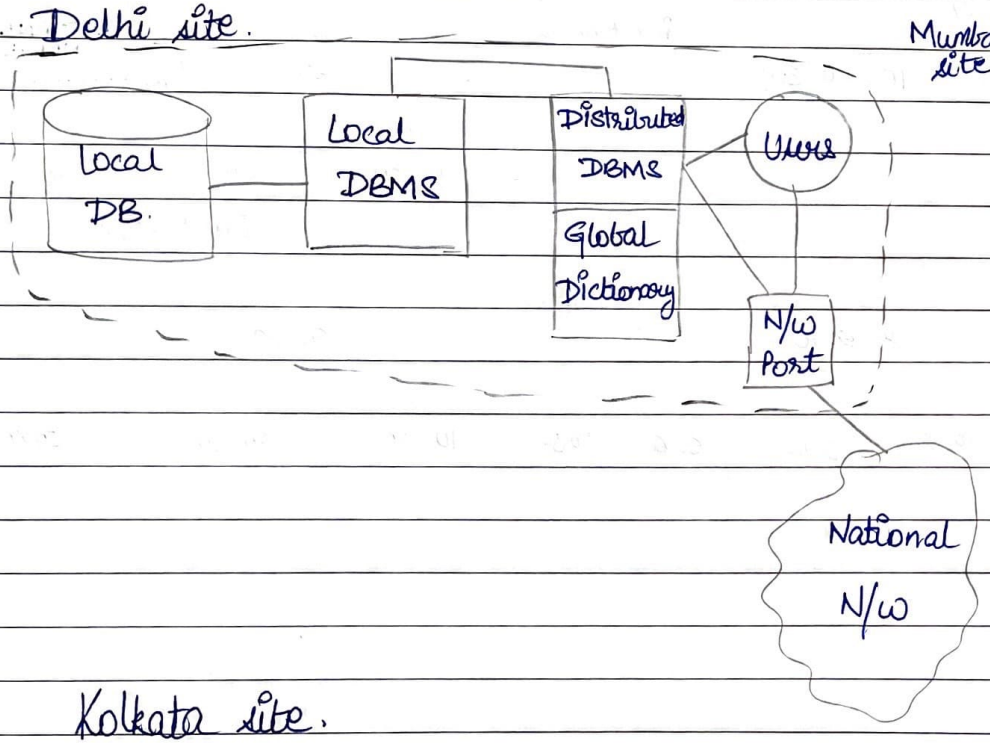


## * Components of Distributed Databases.

Global dictionary – It ~~checks~~ consists data that is required by each site. (common data for each site).
↳ It also consists of access information.

Distributed DBMS – consits of permision & access controls for some data of local DB.

Directly cannot hit local DB hit Distributed DBMS, if authorized, only then allowed to make changes to the local DB.




(based on software used)

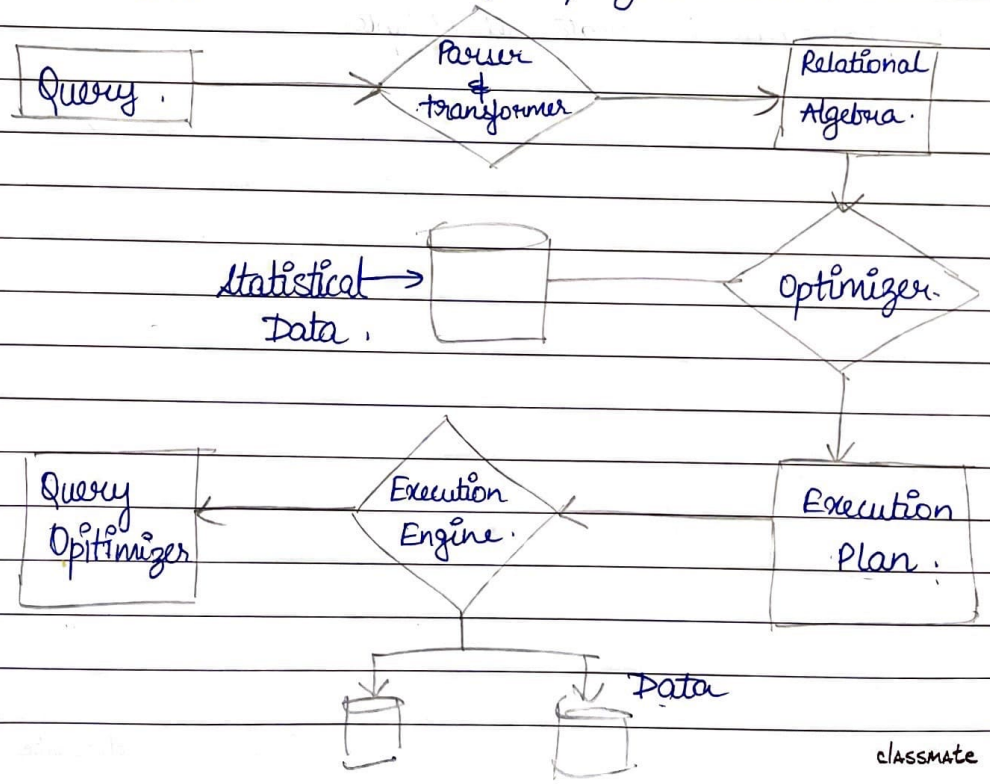
Distributed DB are categorized into two ~~parts~~ classes.

1. Homogenous → the DBMS used is same for all.
2. Hetrogeneous.
   ↳ For this to work, the DBMS should be compatible with each other to minimize the time taken for execution.

~~Statistical data~~

## * Query Optimizer.

Statistical data – ~~It~~ see's if any query is run recently which is similar to the current query being run. This helps in minimizing the time taken for query execution.




classmate

classmate

Execution Plan – To find the best method to minimize the time required to execute the query.
Evaluation Engine – Evaluates the cost function for the running query.

VERTICAL FRAGMENTATION.
→ When the entries need to be ~~changed in a~~ divided "vertically" by columns.

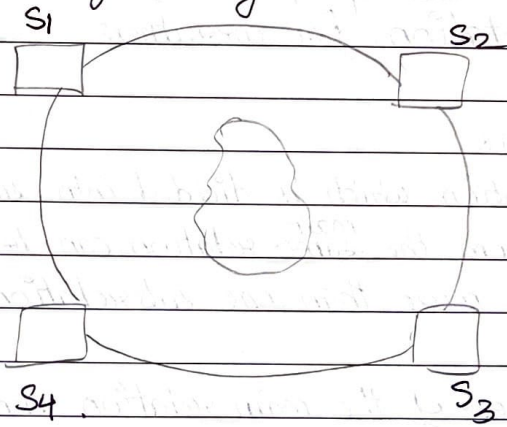
HORIZONTAL FRAGMENTATION.
→ When the entries are divided "horizontally" by grouping rows to create set of tuples.



## DISTRIBUTED DATABASES.

(last 3 pages are a part of this as well)
The sites will be connected using the different topologies present.
Types → mesh Bus
star Hybrid
ring Tree.
Techniques to apply to the different types of distributed db's
Types of Distributed DB's.
1. Fully Connected.
2. Partially Connected
3. Ring Connected.
4. Tree Connected.
5. Hybrid Connected.
6. Star Connected.

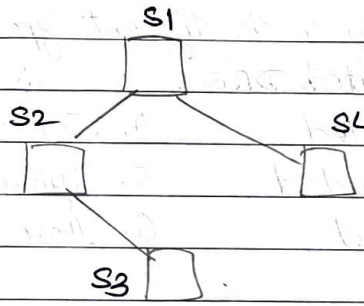
RING CONNECTED.
→ It is used for making it more secure.




→ Starvation may take place while token passing takes place.

TREE CONECTED.

→ To acheive centralization.
→ Here independency is reduced.
→ The load on the main DB is reduced as the DB of site (S3) is updated, the site (S2) first has to take care of the changes and then pass it to the root site (S1).




* To ensure ACID properties, there a 3 rules on fragmentation. for correctness.
They are.

① Completeness. (This rule is for Vertical frag.)
A main relation which is divided into sub relations, the data from the ~~same~~ main relation can be found in one or more than one subrelation, or fragments.
The attributes of the main relation should be present ~~at least~~ in one or more than one of the fragments present.

classmate

Example, R

| Emp. No. | Emp. Name | Salary |
|---|---|---|

R1 | Emp No. | Emp. Name |

R2 | Emp No. | Emp. Name | Salary |

⋮

Rn | Emp No. | Salary |

② Disjointness. (This rule is for Horizontal frag.)
The data value present in the main relation, should be present in only one of the sub relation / fragmentation.
→ This is done to avoid the repeatness of data.

③ Reconstruction.
There should be such a relation between the sub relations / fragmentation, such that, ~~or~~ we can again form the main relation.

* FRAGMENTATION.

1. HORIZONTAL → ① Primary. → Primary using minterms.
② Derived.
2. VERTICAL → ① Using grouping. (Allows overlap)
② Using spliting. (Does not allow overlap)

classmate

① Primary Horizontal Fragmentation.

Employee

| Emp Id | City | Salary | Description |
|---|---|---|---|
| 1 | Mum | 50000 | Manager |
| 2 | Mum | 50000 | Manager |
| ⋮ | ⋮ | ⋮ | ⋮ |

Applying query select * from Employee where city = Mumbai we will fetch data of employee's in city Mumbai and it will/may be stored in Distributed Databases.

② Derived Horizontal Fragmentation.

When some data is derived from one table by the primary key and the ~~so~~ remaining data is derived from another table with the help of foreign key.

Minterms

→ Used for fragmentation.

eg. ① select * from Player where city = Mumbai.
② select * from Player where city = Pune
③ select * from Player where year <= 1976.

The minterms will be.

$m_1$ ⇒ city = mumbai && city = Pune && year = 1976
$m_2$ → city = mumbai && city = Pune && year <1976.
$m_3$ ⇒ city ≠ mumbai && city = Pune && year = 1976.

The meaningless minterms are discarded & the remaining are sorted according to their priority.

DDB is very efficient as ~~central~~ dependency decreases.

VERTICAL FRAGMENTATION.

→ USING GROUPING.

- Done by Bottom up approach.
- This ~~helpin~~ helps in assurance of 'completeness' rule.

→ USING SPLITTING.

- It helps in assurance of 'disjointness' rule.
- Here, it does not allow duplication of data.

* REPLICATION.

Used for recovering data lost by local DB.

Main ~~&~~ requirement in Replication is that the data ~~in~~ in replected data should be updated if original is changed.

This is dependent on the cost required for space complex & time complex.

Two types.

① Synchronous
→ When data in original is changed then data in replicas should also be updated.
② Asynchronous.
→ After a certain commit, the data changes should be reflected in the replicated tables.

Three types.

→ ① Full Replication.
→ Here, the complete data is replicated.
② Partially Replication.
→ Here, the ~~so~~ data is stored for a certain time period ~~in~~ ~~on~~ based on requirement.
③ No Replication.
→ Data is not replicated.

classmate

Topics:
- Data allocation.
- Distributes Query Processing (Semi Join)
- Transaction Management (2PL)

---

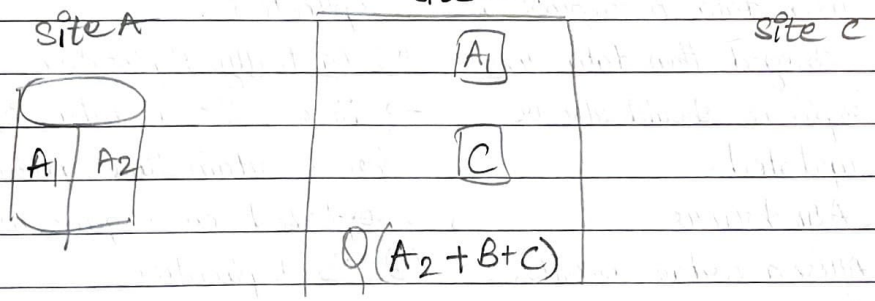
DATA ALLOCATION – When a query is recieved from a site, allocating the data (or table) for that table is called data allocation.

Data allocation is done on the basis of the partitioned data.

If we go for FULL REPLICATION, then data allocation is easy as all the data required for the query is available.

Now consider the situation where in site B, ~~or~~ a query is passed which requires data from site A $= (A_2)$., site B and site C.

Site B has ~~a~~ replicated data partially of A $(A_1)$, and C is fully replicated.






In this, it has to be managed if the data of $A_2$ is fetched to site B ~~or~~ or if the query is passed to site A for $A_2$ data.

~~No or~~ In no ~~replection~~ replication, it is difficult as we have to decide if large ~~am~~ data should be fetched or query should be passed to the sites. This increases complication.

① Find data.
② Flow of execution should be decided.
③ Data transfer.
④ Processing algorithm.
⑤ Access path – It checks the global dictonary ~~to check~~ if the data required is present or from where it can be accessed.

DISTRIBUTED QUERY PROCESSING.

When we want to club two data table present on two different databases for data allocation, we can perform semi join. (JAVA POINT).

Advantage –

~~A~~ Complete data is not passed, thus updatation of data is also easier.

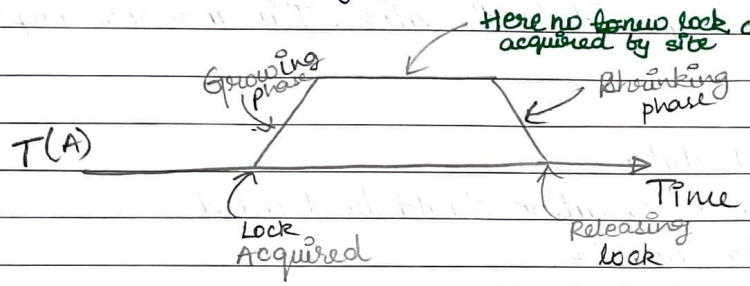
Disadvantage –

## 2 PHASE LOCK PROTOCOL :-

Until one site is ~~task~~ fetching data from the 2nd site, the 2nd site can ~~be~~ say it's is locked.

Phase 1 – Growing phase.

Phase 2 – Shrinking phase.




In phase 1, the site or table which needs to give data, starts acquiring the locks for the data.

In phase 2, once the query execution is done, the locks acquired are released.

There are 2 types of 2PL.

1. Centralized 2PL.
   the one done above.
2. Primary copy 2PL.
   It maintains the record of lock protocols for the table. Thus, if a site requires data, it will check this primary copy. This further decreases waiting time. This is ~~done~~ because, it checks with the primary copy if it is free, or if it is locked, when the lock will be released.

# DATA SECURITY.

AUTHENTICATION – To identify a user

AUTHORIZATION – To know how much data is to be shared with the user.

| ATTENDANCE | S | I | U | D | C |
|---|---|---|---|---|---|
| P | ✓ | | | | |
| VP | ✓ | | | | |
| HOD | ✓ | ✓ | ✓ | | |
| Database Admin | ✓ | ✓ | ✓ | ✓ | ✓ |
| Teachers | ✓ | | ✓ | | |
| CR's | ✓ | | | | |
| Students | ✓ | | | | |
| Class Teacher | ✓ | ✓ | | | |

GRANT COMMAND

Grant <priviledge list> On <table list> to <user list>

Grant select, insert, update on student to HOD.

## * DATABASE AUDITING

Auditing → verification of activity, inspection or examination of a process or quality system

Database auditing → monitoring and recording of selected users database actions.

## * Types of Security.

1. Legal and Ethical Issues.
   eg. access to databases of website.
2. Policy Issues.
   eg. seting time slots to access databases.
3. System Related Issues.
4. The Need To Identify Multiple Security Levels.

## * Threats To Databases.

1. Loss of integrity
→ Third party modification to database.
2. Loss of availability.
→ Not allowed to access the data.
3. Loss of confidentiality.

## * Implemented Countermeasures.

1. Access control
2. Inference control
3. Flow control. → what info. reaches which person.
4. Encryption

A DBMS typically includes a db security & authorization subsystems that is responsible for ensuring the security positions of a database against authorized access.

classmate

Covert channels → Channels that are pathways for info. to flow implicitly in ways that violate the security policy of an organization.

## * Database Security Measure.

1. Discretionary Security ~~measures~~ mechanism.
2. Mandatory Security mechanism.

## * Statistical Database.

eg. train booking system during Tatkaal,
→ Used to mainly to produce statistics on various populations.
→ May contain confedential data on individuals, which should be protected from user access.
→ SD security may not allow access to individual data.

## * Database Security and DBA.

The DBA is the central authority for managing a database system.
Priveledges.
1. Account creation → access control.
2. Priveledge granting → discretionary.
3. Priveledge revocation
4. Security level assignment → control mandotory authorization

Database Audit → consists of reviewing the log to examine all accesses and operations applied to the database during a certain time period.

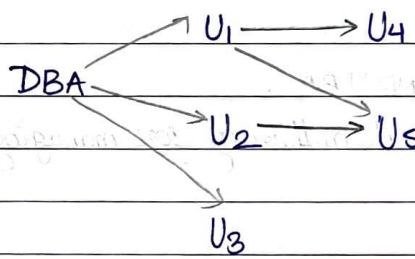
classmate

* TYPES OF DISCRETIONARY PRIVILEGES.

→ CREATE SCHEMA OR CREATE TABLE.
→ CREATE VIEW.
→ ALTER
→ DROP. → delete relation or views.
→ MODIFY
→ SELECT.

* FLOW CONTROL

The passage of authorization from one user to another may be called the passage of authorization.




* ROLES.

→ Roles permit common priveledges for a class of users can be specified just once by creating a corresponding "role".
→ Roles can be assigned to users & roles.

Create role <rolename>

1) What is meant by Granting & Revoking of Priviledges?
2) Explain Discretionary Access Control.
3) Explain role based access control for multilevel security

Trigger – EAS → Event

* TEMPORAL DATABASES.

→ Time representation, calenders & time dimensions.

⇒ Valid Time Database Schema.

Emp_VT

| Name | SSN | Salary | DNO | Sup_ssn | Vst | Vet |
|---|---|---|---|---|---|---|

Dept_VT

| DName | DNo | Total-Sal | Manager-ssn | Vst | Vet |
|---|---|---|---|---|---|

⇒ Transaction Time Database Schema.

Emp_TT.

Name SSN Salary DNo Supervisor SSN Tst Tet

Dept_TT

DName DNo Total_Sal Manger_ssn Tst Tet



⇒ Biotemporal database schema.

Emp_BT

Name SSN Salary DNo Sup_ssn Vst Vet Tst Tet

Dept_BT

DName DNo Total_Sal Man_SSn Vst Vet Tst Tet.

CREATE VIEW <view name> [column names]
AS SELECT <columns to be selected>
FROM <table names>
WHERE <conditions.

---

GRANT (S,I,U,D,C) ON <tables> TO <user>
[OPTIONAL: WITH GRANT OPTION]

↓

Grants the user to grant the feature to other users.

---

REVOKE (S,I,U,D,C) ON <tables> FROM <user>