DBMS

It is a collection of interrelated data & a set of programs to access those data.

Goal :- to store & retrieve DB information that is convenient & efficient.

* DB Applications
* Problems with traditional File Processing System :-
  - → Data Redundancy & inconsistency
  - → Difficult to access
  - → Data isolation
  - → Integrity problems
  - → Security problems

* Three-Tier Architecture
* Data Independence & its types
* Schema & Instances
* Data Model :- a collection of conceptual tools for describing data, data relationships, semantics & constraints.
  - → Relational Model ⇐ Network data model, Hierarchical data model
  - → ER Model
  - → Object-based data Model
  - → Semistructured Data Model (XML)

Introduction to DB system architecture :-

The architecture of a DB system is influenced by the computer system, by such aspects of architecture as :-

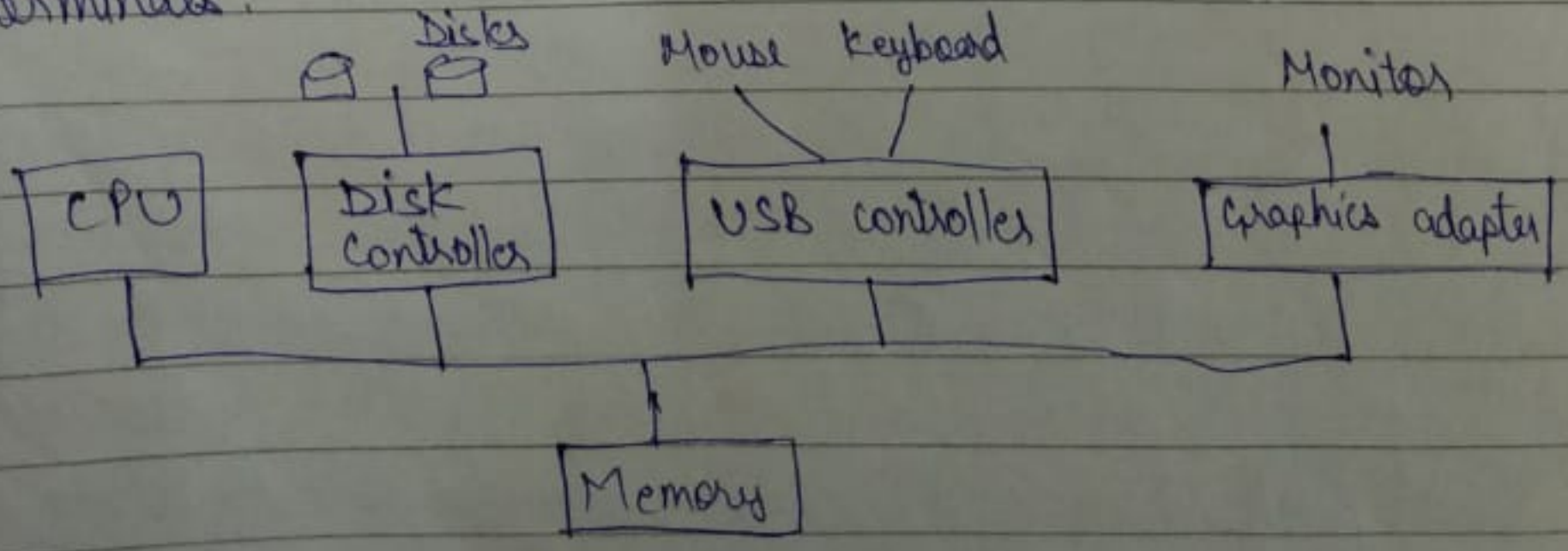
a). Networking :- some tasks to be executed on server system & some on client system.

b). Parallel processing :- to speed up the activities (faster response).

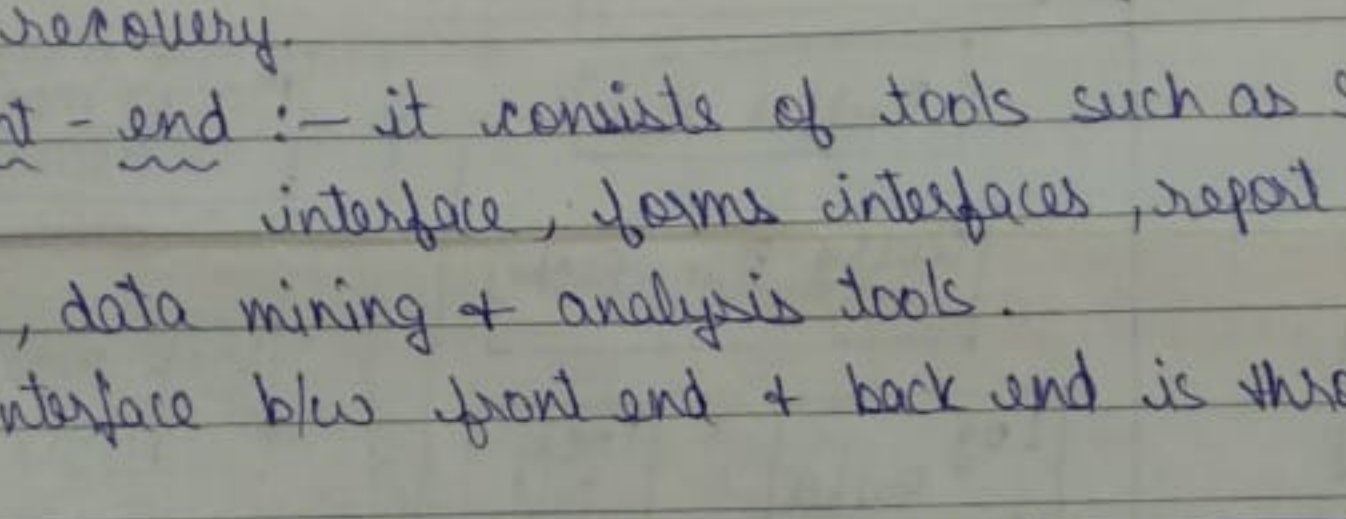
c). Distribution :- to reside the data on the sites where they are generated or most needed, still accessible from other sites. It keeps multiple copies of DB across different sites.

I). CENTRALIZED SYSTEMS ARCHITECTURE :-

* They run on a single computer system and don't interact with other computer systems.
* One / more CPUs and a no. of device controllers connected through a common bus providing access to shared m/m.
* Single-user system :- PC (desktop unit) having one CPU and one / two hard disks, one user.
* Multi-user system :- more disks, more memory, multiple CPUs & a multi-user OS. It serves a large no. of users who are connected to the system via terminals.



II). CLIENT - SERVER SYSTEMS :- A centralized system act as server system that satisfies requests generated by client systems.

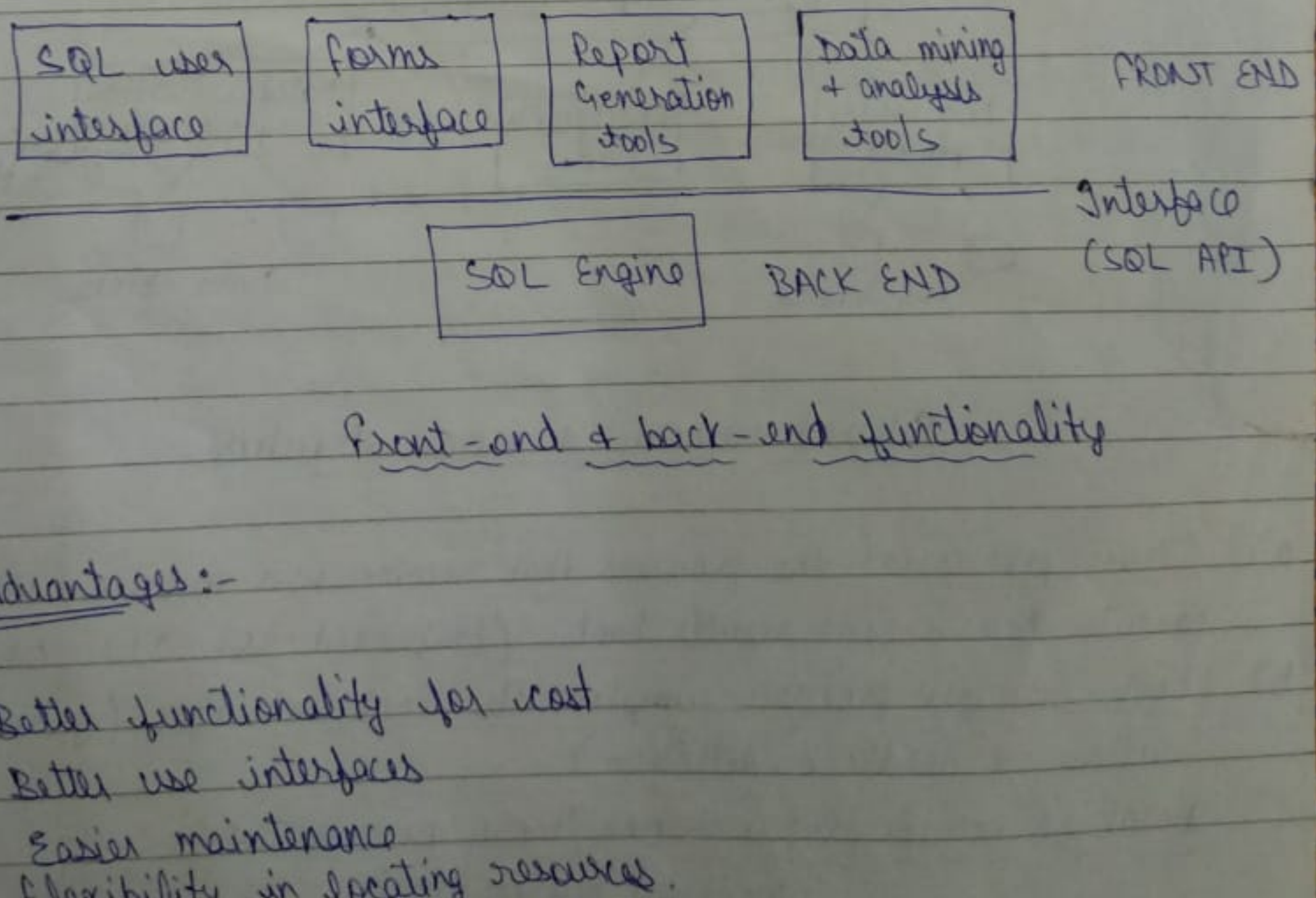


Database functionality can be divided into :-

1). Back - end :- it manages access structures, query evaluation + optimization, concurrency control and recovery.

2). Front - end :- it consists of tools such as SQL user interface, forms interfaces, report generation tools, data mining + analysis tools.

⇒ Interface b/w front end + back end is through SQL.

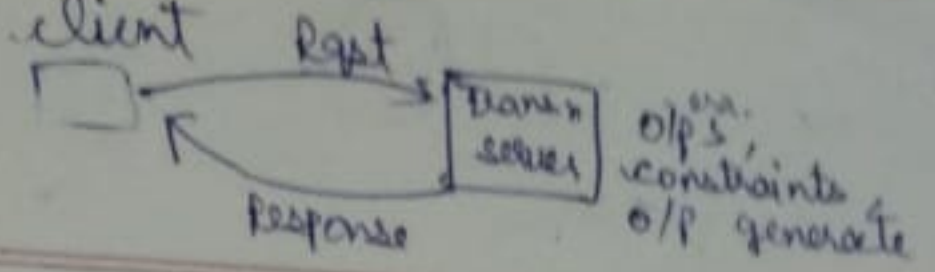


Front - end + back - end functionality

Advantages :-

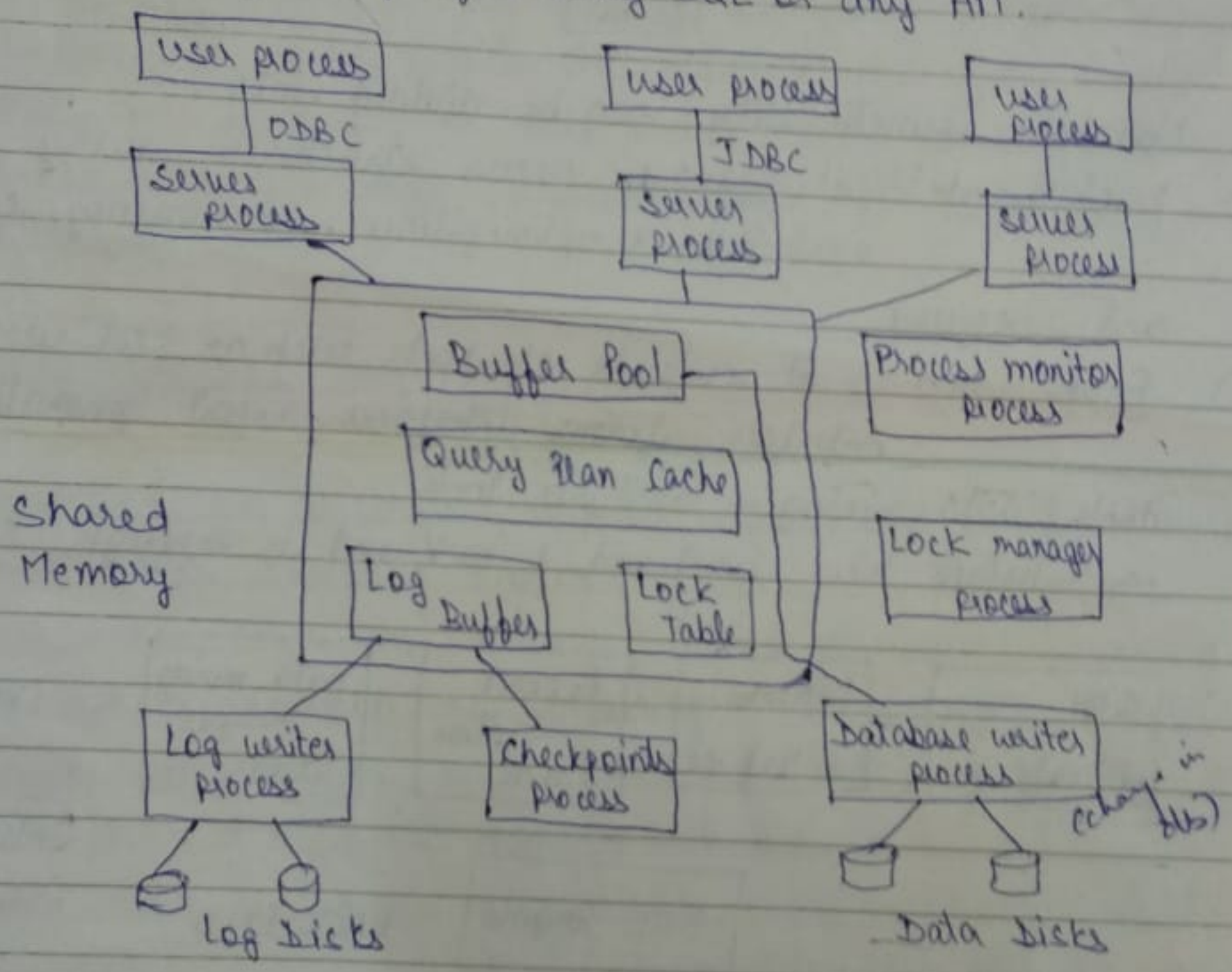
1. Better functionality for cost
2. Better use interfaces
3. Easier maintenance
4. Flexibility in locating resources.

(7)



III). SERVER SYSTEM ARCHITECTURE :- Broadly classified as

1). Transaction Servers :- aka query ~~process~~ servers systems, provide an interface to which clients can send requests to perform an action. Servers then execute the Transactions & send the results back to the clients. Requests may be specified using SQL or any API.



Shared memory & process structure

a). Servers processes :- the processes that receive user queries (Trans.s), execute them & send results back. (Embedded SQL, JDBC, ODBC)

b). Lock managers process :- implements lock manager (grant, release & deadlock detection).

c). Database writer process :- one/more processes that o/p modified buffer blocks back to disk.

d). Log writer process :- it outputs log records from the log record buffer to stable storage.
→ Server processes simply add log records to the log record buffer in shared m/m.

e). Checkpoint process :- performs periodic checkpoints.

f). Process monitor process :- monitors other processes, if any process fails, it takes recovery actions (abort, restart).

Shared m/m contains all shared data :-
→ Buffer pool → Lock table
→ Log buffer, containing log records waiting to be o/p.
→ Query plan cache, which can be reused if same query is submitted.

* For managing shared m/m, semaphores (mutual exclusion) and concept of locks can be used.

2). Data Servers :- allow clients to interact with servers by making requests to read or update data. To read or update pages, tuples or objects (smaller than a file). It also provides indexing & transaction facilities for data.
→ Used in LAN, high-speed connection b/w clients & servers.
→ High-speed make possible the processing at client machine and send data back to servers.
→ It requires full back-end functionality at clients.
→ Used in object-oriented DB systems.

* Issues of this architecture in comparison to local m/m reference :-

a). Page shipping v/s item shipping :-

3). Cloud - Based Servers :- In it, the service provider runs its own software, but runs it on computers provided by another company. These computers are not real but are simulated machines known as virtual machines.

IV). PARALLEL SYSTEMS :- It improves processing & I/O speeds by using multiple processors & disks in parallel. They can query DB of terabytes (10^{12} bytes)

a). Coarse-grain parallel systems :- consists of a small no. of powerful processors. DBs on these machines don't partition a single query among the processors, instead they run each query on a single CPU, allowing multiple queries to run concurrently. More transactions run per second.

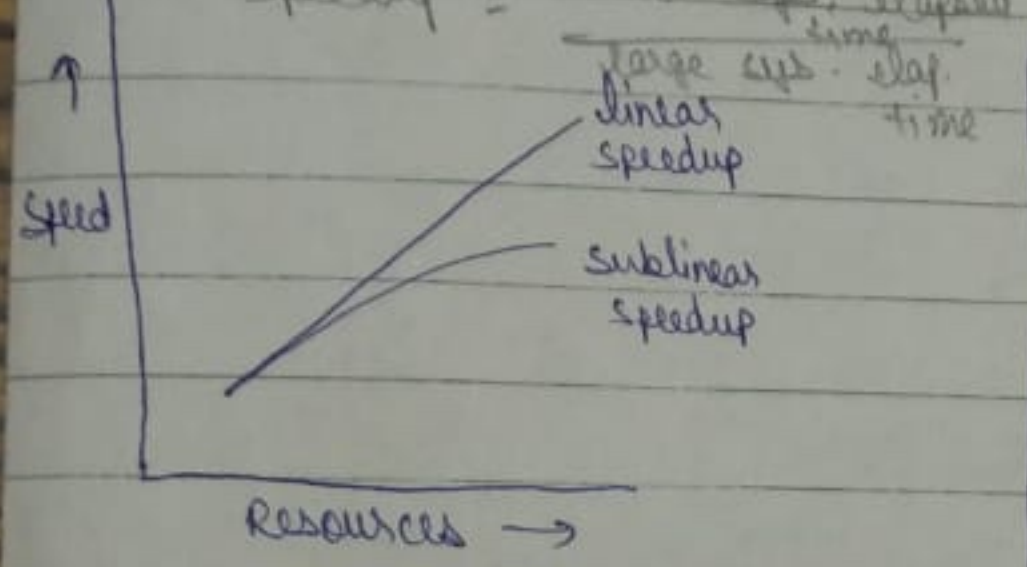
b). Fine-grain parallel (massively ||el) systems :- use thousands of smaller computers

1). Speedup and scaleup :-

Speedup :- Running a task in less time by increasing the degree of parallelism.

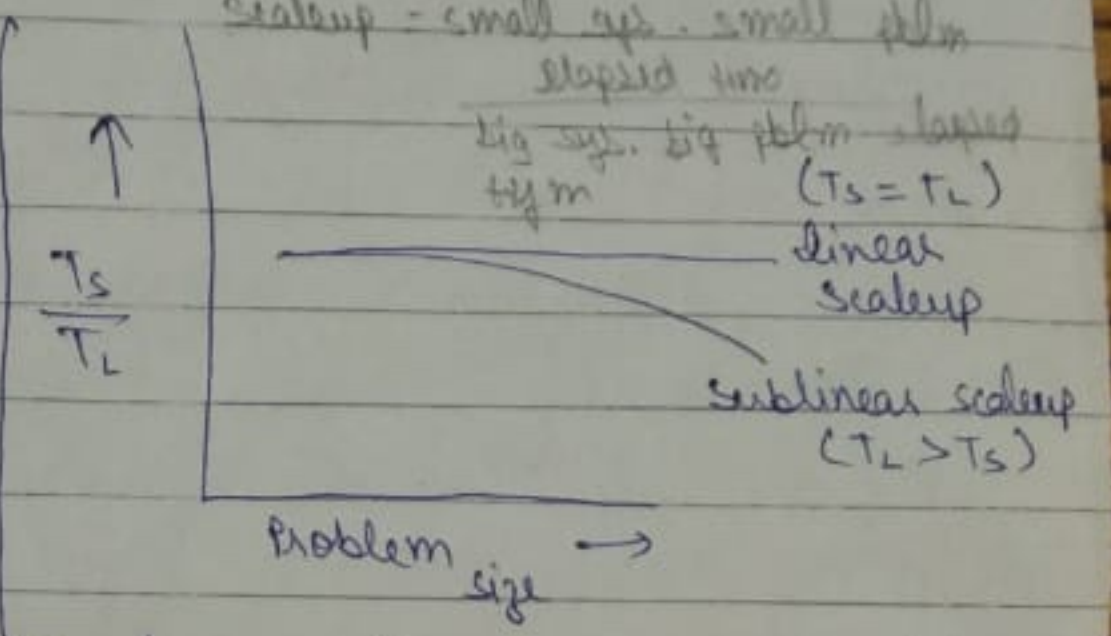
Scaleup :- Handling larger tasks by increasing degree of parallelism.

speedup = $\frac{\text{small sys. elapsed time}}{\text{large sys. elap. time}}$



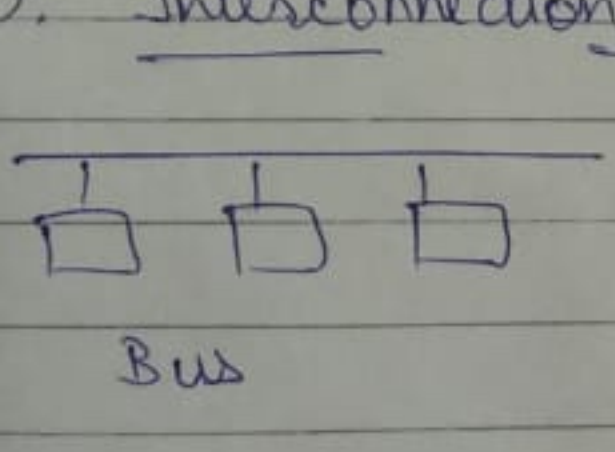
Speedup with increasing resources

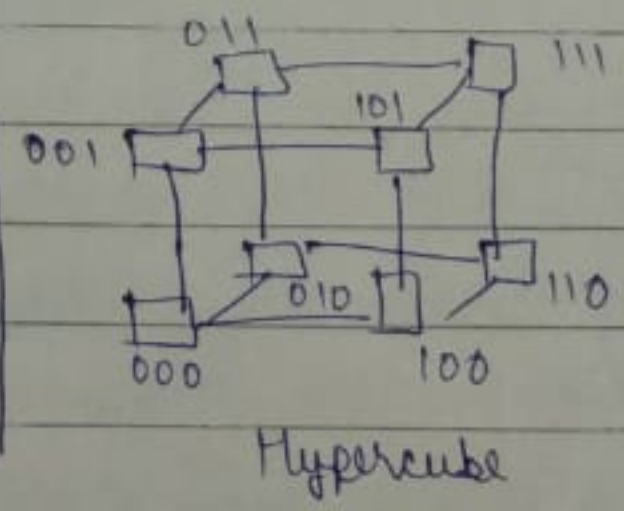
scaleup = $\frac{\text{small sys. small prbm elapsed time}}{\text{big sys. big prbm elapsed time}}$ ($T_s = T_L$)



Scaleup with increasing problem size & resources

2). Interconnection Networks:-





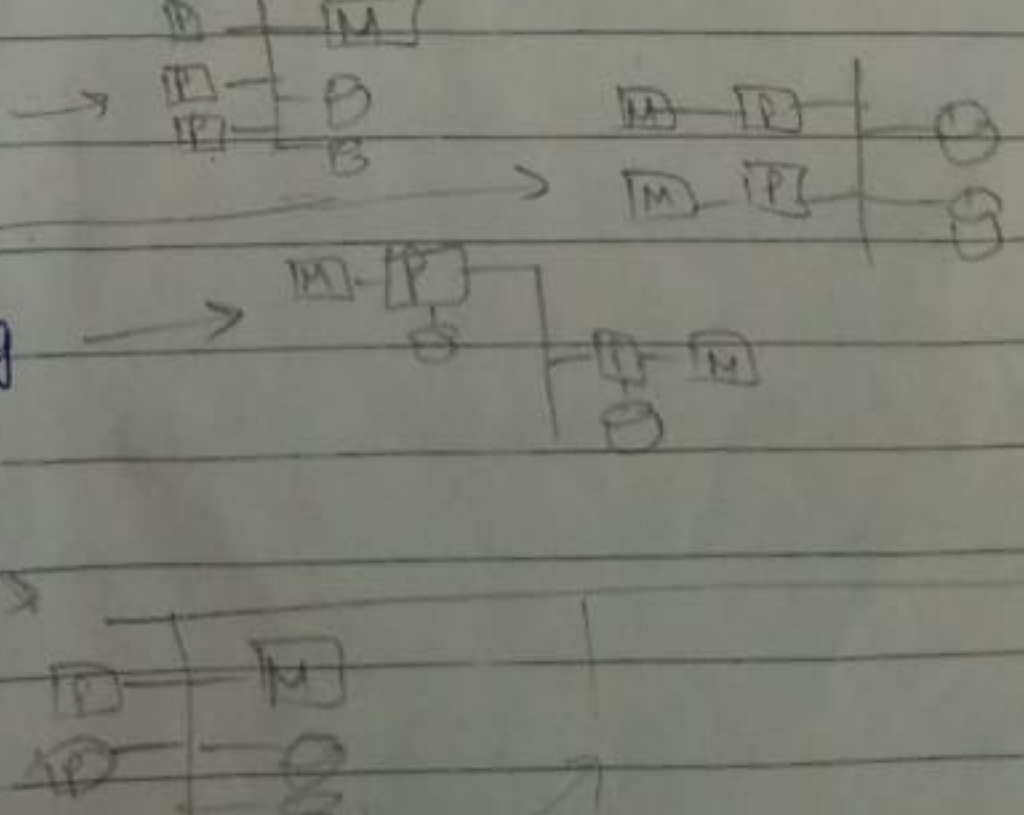
3). Parallel Database Architectures :-

a). Shared memory

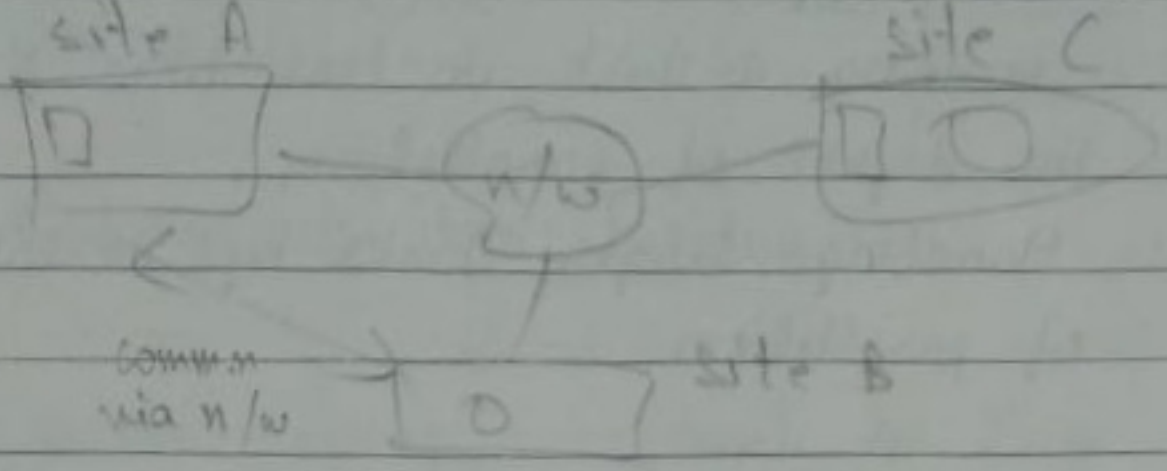
b). Shared disk

c). Shared nothing

d). Hierarchical



Distributed



① → Homogeneous : same s/w / schema on all sites
Goal : Hiding distr n

② Heterogeneous : Diff s/w / schema on diff. sites
Goal : more functionality

Local transaction | Global txn.
accessed by site which
initiated it.

Trade-off : 1) chasing data.
2) Autonomy → control on data stored locally.
3) Higher sys. availability

Disad's : 1) complexity
2) S/w cost 3) More bugs 4) Added processing overhead

Impl'n issues
→ Atomicity needed
→ 2PC (2 phase commit protocol)
→ Deadlock detection