* Homogeneous Distributed database - all sites have identical database management system software, are aware of one another. Local sites surrender a portion to change schemas.
- This software must exchange info. with other sites
Ex Banking Database

* Heterogeneous Distributed database - different sites may use diff^n schemas, not be aware of one another, provide only limited facilities for cooperation in transaction processing.

* Distributed Data storage :-

① Replication - store several identical replicas of the relation on multiple sites. Alternative to Replication is to store only one copy of relation 'r'.

② Fragmentation - partitions the relation into several fragments and store each fragment at a diff^n site.

→ ① & ② can be combine

1) Data Replication

- Full replication - copy is stored in every site in the system
Pros & cons ① Availability - if one site is failed then other will fulfill requirement
② Increase parallelism - One site is useful for read while other site is useful for process. Data replication minimizes movement of data between sites.
③ Increased overhead on update - 'r' is updated on each and every site which result more overhead.
- Controlling concurrent updates by several transactions to replicated data is more complex than centralized systems.

2) Data fragmentation - Horizontal, Vertical

- Horizontal fragmentation split the relation by assigning each tuple of r to one or more fragments
- Vertical fragmentation splits the relation by decomposing the scheme R of relation r.

- In horizontal fragmentation, a relation r is partitioned into a number of subsets, $r_1, r_2, \ldots, r_n$. Each tuple must belong to at least one of the fragments so that original relation can be construct

Ex account 1 = $\sigma_{branch\_name = \text{"Hillside"}}(account)$ — database has only two branches.

account 2 = $\sigma_{branch\_name = \text{"valleyview"}}(account)$

→ It usually used to keep tuples at the sites where they are used the most, to minimize data transfer

$$\boxed{\begin{array}{l} r = r_1 \cup r_2 \cup \ldots r_n \\ r_i = \sigma_{p_i}(r) \end{array}}$$

- In vertical fragmentation decomposition used. $r(R)$ involves the several subsets of attributes $R_1, R_2, \ldots, R_n$ of schema R

$R = R_1 \cup R_2 \cup \ldots \cup R_n$

Each fragment r_i of r is defined by

$$r_i = \pi_{R_i}(r)$$

$$r = r_1 \bowtie r_2 \bowtie r_3 \bowtie \ldots \bowtie r_n$$ natural join

Ex University database with relation employee-info that stores employee-id, name, designation and salary.

→ vertical fragmentation: employee-private-info : employee-id, salary
employee-public-info : employee-id, name, designation

→ Every fragment should have primary-key so we can able to reconstruct 'r'.

* Transparency :-

- user of distributed database system should not be required to know where the data is located nor how the data can be accessed at the specific site. This characteristic called Data transparency

① Fragmentation transparency – How relation has been fragmented

② Replication – User view each data object as unique. User should not concerned with what data-objects have been replicated or where it is.

③ Location – system should be able to find any data as long as the data identifier is supplied by the user transaction

- Data items - such as relations, fragments and replicas - must have unique name. Two sites do not use the same name for distinct data items.

Transparency: Solution of - two site do not use same name

(a) Name server - all names to be registered in a central name server, which also help to find items.

→ Problem - (1) bottleneck - every time Name server is used
(2) if name server crashes, it may not be possible for any site in the system to continue to run.

(b) 2nd solution is use site prefix in name. This approach ensures that no two sites generate the same name
→ It, fails to achieve location transparency, since site identifiers are attached to names. e.g. site17.account or account@site17
→ Many database system use IP Address of a site to identify it.

→ Solution - create a set of alternative names or aliases for data items. The mapping of aliases to the real names can be stored at each site. User will be unaware of location and unaffected if the database administrator decides to move data item to another site.
- User should not refer replica of data item instead, system should determine which replica to reference on a "READ" and should update all replica on a "WRITE" request.

*Distributed Transactions :-

→ Global Transaction associated with multiple sites if any one fail then erroneous result generated. Every Transaction must follow ACID system structure.
- each sites have its own local transaction manager and transaction coordinators in distributed system

* Transaction manager - manages execution of transaction (local, Global)
- maintain a log for recovery purposes
- participating in an appropriate concurrency-control scheme to coordinate the concurrent execution of the transactions executing at that site

* Transaction Coordinator not part of Centralized environment. but in distributed system it is responsible for
- starting the execution of the transaction
- Breaking the transaction and distribute to the other sites for execution
- coordinating the termination of transaction which may committed or aborted at all sites.

System Failure Mode -

- Distributed may suffer from the same types of failure that a centralized does e.g. software error, hardware error, disk error.
- The basic failure types are

① Failure of site ② loss of messages ③ Failure of a communication link
④ Network partition

→ Loss or corruption of message is common so TCP/IP Protocols used
→ If site A and B want to communicate then route is establish on network if intermidiate link is fails then message rerouted
→ Both system is partitioned (split into two subsystems) then can not communicate

* Commit Protocols :-

Two phase Commit

- Protocol - Transaction T initiated at site S_i, where the transaction coordinator is C_i
- when T completes it execution - that is, when all sites at which T has executed inform C_i that T has completed - C_i starts the 2PC protocol.

Phase 1 :- C_i adds the record <prepare T> to the log, and forces the log onto stable storage. It then sends a prepare T message to all sites at which T executed. If answer is no, it adds a record <no T> to log and send message to C_i. If answer is yes, it record <ready T> to the log and send message to C_i

phase 2 :- when C_i receives responses to <prepare T> message from all sites, C_i can determine whether the transaction T must be committed or aborted. Transaction T can be committed if C_i receved a <ready T> message from all participating sites. Depending on the verdict either record <commit T> or <abort T> and send to all participants.

→ A site at which T executed can unconditionally abort T at any time before it send message <ready T> to the coordinator.

- In some 2PC, site sends <acknowledge T> message to the coordinator at the end of the second phase of the protocol. when the coordinator receives the <acknowledge T> message from all the sites, it adds the record <complete T> to the log.

* Handling of Failures

ⓐ Failure of a participating site :- If the coordinator C_i detects that a site has failed, it takes these actions : If the site fails before responding with a ready T message then C_i wait for abort T message. If site fail after coordinator has received the ready message from site, coordinator ignore failure of that site

→ when participating site S_k recover from failure, it must examine its log to determine the fate of those transactions that were in the midst of execution

Possible cases

① log contains a <commit T> record. In this case site → redo(T)

② log contains a <abort T> record - Undo(T)

③ log contains <ready T> records. In this case, the site must consult C_i. If C_i is up, it notifies S_k regarding T commit or abort. If C_i is down S_k send <querystatus T> message to all the sites in the system. If no site has the appropriate information then S_k can neither abort nor commit T. Thus S_k must periodically resend the <querystatus T>.

④ failure before <prepare T> means empty log. S_k must undo(T)

ⓑ Failure of the coordinator

- If coordinator fails then active sites must decide the fate of T.
- In certain cases, the participating sites cannot decide whethere to commit or abort T and these sites must wait for recovery

cases:

① active sites contain <commit T> record in it's log, then T must be committed

② active site contain <abort T> - T must be aborted

③ site not contain <ready T>, then C_i cannot decide commit or abort so it will abort T

④ <ready T> in log, but no additional control record <abort T> or <commit T>. Since coordinator has failed to determine what to do so it will wait until C_i recover. T may hold all resources and locked it hense other can not use it. Data items may be not available due to failed site but on active site as well. This is called blocking Problem.

Network Partition

① Coordinator & participants in one group. In this case, the failure has no effect on the commit protocol.

② Coordinator & participants belong to several partitions. If one partition failed then coordinator ignore and follow usual commit protocol in same partation.

→ Major disadvantage of 2PC is coordinator failure may block system

ⓒ Recovery and Concurrency Control :

- The recovery procedure must treat in-doubt transactions.
- Indoubt transactions which a <ready T> log record but not found <commit T> or <abort T>. Recovering site must determine the commit-abort status from other sites.
- Normal transaction processing at the site cannot begin until all in-doubt transactions have been committed or roll back
- Finding the status of in-doubt transaction can be slow.
- main problem in 2PC is blocked, if coordinator failed.
- Recovery algorithem - Instead a writing a <ready T> log, the algo. write <ready T, L> log record, where L is a list of all write lock held by the transaction T, when log record is written.
- At recovery time, after performing indoubt transaction T write locks reacquired. Transaction processing start at the site, even before the commit-abort status of the indoubt transactions is determined. Thus site never gets blocked.