The DBMS can be classified according to the number of users and the database site locations. These are:

On the basis of the number of users:

- Single-user DBMS
- Multi-user DBMS

On the basis of the site location

- Centralized DBMS
- Parallel DBMS
- Distributed DBMS
- Client/server DBMS

will discuss about some of the important types of DBMS system, which are presently being used.

The database system may be multi-user or single-user. The configuration of the hardware and the size of the organization will determine whether it is a multi-user system or a single user system.

In single user system the database resides on one computer and is only accessed by one user at a time. This one user may design, maintain, and write database programs.

Due to large amount of data management most systems are multi-user. In this situation the data are both integrated and shared. A database is integrated when the same information is not recorded in two places. For example, both the Library department and the Account department of the college database may need student addresses. Even though both departments may access different portions of the database, the students' addresses should only reside in one place. It is the job of the DBA to make sure that the DBMS makes the correct addresses available from one central storage area.

Centralized Database System

The centralized database system consists of a single processor together with its associated data storage devices and other peripherals. It is physically confined to a single location. Data can be accessed from the multiple sites with the use of a computer network while the database is maintained at the central site.

Disadvantages of Centralized Database System

When the central site computer or database system goes down, then every on (users) is blocked from using the system until the system comes back.

• Communication costs from the terminals to the central site can be expensive.

Parallel Processing Defined

Parallel processing divides a large task into many smaller tasks, and executes the smaller tasks concurrently on several nodes. As a result, the larger task completes more quickly.

For example, in a bank with only one teller, all customers must form a single queue to b served. With two tellers, the task can be effectively split so that customers form two queue and are served twice as fast-or they can form a single queue to provide fairness. This is a instance in which parallel processing is an effective solution

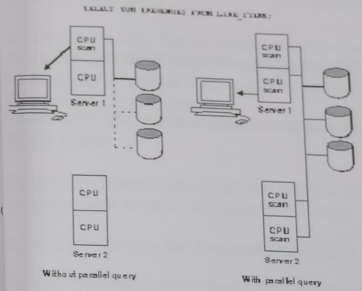
"Parallel Databases improve system performance by using multiple resources and operations parallel"

1] Introduction to Parallel Databases

Companies need to handle huge amount of data with high data transfer rat The client server and centralized system is not much efficient. The need to improve th efficiency gave birth to the concept of Parallel Databases. Parallel database syste architecture consists of a multiple Central Processing Units (CPUs) and data storage disk i parallel. Hence, they improve processing and Input/Output (I/O) speeds. Parallel databas ystems are used in the application that have to query extremely large databases or that hav process an extremely large number of transactions per second.

Parallel database system improves performance of data processing using ltiple resources in parallel, like multiple CPU and disks are used parallely. lso performs many parallelization operations like, data loading and query processing.

Parallel Databases improve system performance by using multiple resources and operations parallel



Advantages of a Parallel Database System

• Parallel database systems are very useful for the applications that have to query extremely large databases (of the order of terabytes, for example, 1012 bytes) or that have to process an extremely large number of transactions per second (of the order of thousands of transactions per second).

• In a parallel database system, the throughput (that is, the number of tasks that can b completed in a given time interval) and the response time (that is, the amount of time it take to complete a single task from the time it is· submitted) are very high.

Disadvantages of a Parallel Database System

• In a parallel database system, there· is a startup cost associated with initiating a sin process and the startup-time may overshadow the actual processing time, affecting speed adversely.

• Since process executing in a parallel system often access shared resources, a slowdown may result from interference of each new process as it completes with existing processes commonly held resources, such as shared data storage disks, system bus and so on.

1.1] Goals of Parallel Databases

The concept of Parallel Database was built with a goal to:

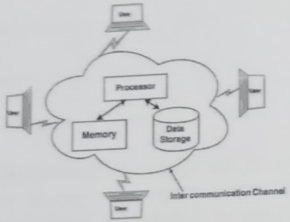
i) Improve performance:

The performance of the system can be improved by connecting multiple CPU and disk parallel. Many small processors can also be connected in parallel.

ii) Improve availability of data:
Data can be copied to multiple locations to improve the availability of data.
For example: if a module contains a relation (table in database) which is unavailable then i
is important to make it available from another module.

iii) Improve reliability:
Reliability of system is improved with completeness, accuracy and availability of data.

iv) Provide distributed access of data:
Companies having many branches in multiple cities can access data with the help of paralle
database system.



Parallel database system

What Are the Benefits of Parallel Database?

Parallel database technology can benefit certain kinds of applications by enabling:

- Higher Performance
- Higher Availability
- Greater Flexibility
- More Users

Higher Performance

With more CPUs available to an application, higher speedup and scaleup can be attained.
The improvement in performance depends on the degree of inter-node locking and
synchronization activities. Each lock operation is processor and message intensive; there ca
be a lot of latency. The volume of lock operations and database contention, as well as the
throughput and performance of the IDLM, ultimately determine the scalability of the systen

Higher Availability

Nodes are isolated from each other, so a failure at one node does not bring the whole system
down. The remaining nodes can recover the failed node and continue to provide data access

to users. This means that data is much more available than it would be with a single node upon node failure, and amounts to significantly higher availability of the database.

Greater Flexibility

An Oracle Parallel Server environment is extremely flexible. Instances can be allocated or deallocated as necessary. When there is high demand for the database, more instances can be temporarily allocated. The instances can be deallocated and used for other purposes once they are no longer necessary.

More Users

Parallel database technology can make it possible to overcome memory limits, enabling a single system to serve thousands of users.

1.2] Parameters for Parallel Databases

You can measure the performance goals of parallel processing in terms of two important properties:

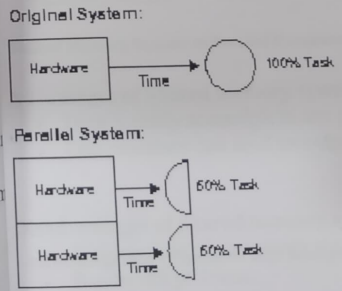
- **Speedup**
- **Scaleup**

Some parameters to judge the performance of Parallel Databases are:
1. Response time: It is the time taken to complete a single task for given time.

2. Speed up in Parallel database:

Speedup is the extent to which more hardware can perform the same task in less time than the original system. With added hardware, speedup holds the task constant and measures time savings. Figure 1-5 shows how each parallel hardware system performs half of the original task in half the time required to perform it on a single system.

Figure 1-5 Speedup



With good speedup, additional processors reduce system response time. You can measu
speedup using this

formula: $\text{Speedup} = \frac{\text{time\_original}}{\text{Time\_Parallel}}$

where

Time_Parallel is the elapsed time spent by a larger, parallel system on the given task

For example, if the original system took 60 seconds to perform a task, and two parallel systems took 30 seconds, then the value of speedup would equal 2. $2 = \frac{60}{30}$

A value of *n*, where *n* times more hardware is used indicates the ideal of linear speedup: when twice as much hardware can perform the same task in half the time (or when three times as much hardware performs the same task in a third of the time, and so on).

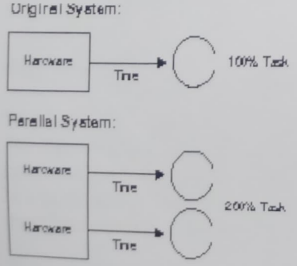
Attention: For most OLTP applications, no speedup can be expected: only scaleup. The overhead due to synchronization may, in fact, cause speed-down.

3. Scale up in Parallel database:

Scale-up is the ability to keep performance constant, when number of process and resources increases proportionally.

Scaleup is the factor *m* that expresses how much more work can be done in the same time period by a system *n* times larger. With added hardware, a formula for scaleup holds the time constant, and measures the increased size of the job which can be done.

Figure 1-6 Scaleup



With good scaleup, if transaction volumes grow, you can keep response time constant by adding hardware resources such as CPUs.

You can measure scaleup using this

formula: $\text{Scaleup} = \frac{\text{volume\_Parallel}}{\text{volume\_Original}}$

where

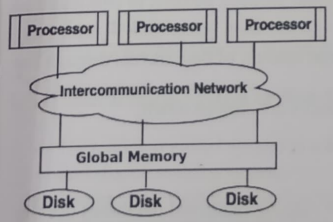
Volume_Parallel is the transaction volume processed in a given amount of time on a parallel system

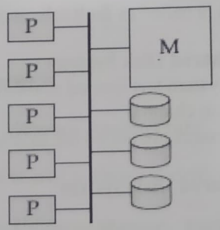
For example, if the original system can process 100 transactions in a given amount of time, and the parallel system can process 200 transactions in this amount of time, then the value of scaleup would be equal to 2. That is, 200/100 = 2. A value of 2 indicates the ideal of linear scaleup: when twice as much hardware can process twice the data volume in the same amount of time.

1.3] Types of Parallel Database Architecture

1.3.1] Shared memory system

- Shared memory system uses multiple processors which is attached to a global shared memory via intercommunication channel or communication bus.
- Shared memory system have large amount of cache memory at each processors, so referencing of the shared memory is avoided.
- If a processor performs a write operation to memory location, the data should be updated or removed from that location.





(a) shared memory

Shared Memory System in Parallel Databases

Advantages of Shared memory system

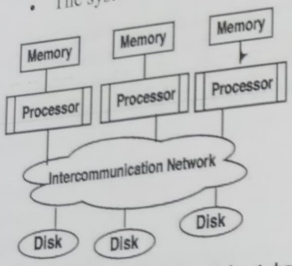
- Data is easily accessible to any processor.
- One processor can send message to other efficiently.

Disadvantages of Shared memory system

- Waiting time of processors is increased due to more number of processors.
- Bandwidth problem.

1.3.2] Shared Disk System

- Shared disk system uses multiple processors which are accessible to multiple disks v
  intercommunication channel and every processor has local memory.
- Each processor has its own memory so the data sharing is efficient.
- The system built around this system are called as clusters.



Shared disk system in Parallel Databases

Advantages of Shared Disk System

- Fault tolerance is achieved using shared disk system.
- **Fault tolerance:** If a processor or its memory fails, the other processor can complete the task. This is called as fault tolerance.

Disadvantage of Shared Disk System

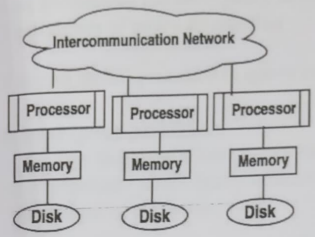
- Shared disk system has limited scalability as large amount of data travels through the interconnection channel.
- If more processors are added the existing processors are slowed down.

Applications of Shared Disk System

Digital Equipment Corporation(DEC): DEC cluster running relational databases use the shared disk system and now owned by Oracle.

1.3.3] Shared nothing disk system

- Each processor in the shared nothing system has its own local memory and local disk
- Processors can communicate with each other through intercommunication channel.
- Any processor can act as a server to serve the data which is stored on local disk.



Shared nothing disk system in Parallel Databases

Advantages of Shared nothing disk system

- Number of processors and disk can be connected as per the requirement in share nothing disk system.
- Shared nothing disk system can support for many processor, which makes the system more scalable.

Disadvantages of Shared nothing disk system

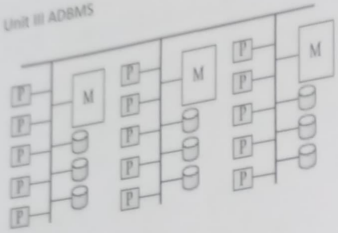
- Data partitioning is required in shared nothing disk system.
- Cost of communication for accessing local disk is much higher.

Applications of Shared nothing disk system

- Tera data database machine.
- The Grace and Gamma research prototypes.

1.3.4] Hierarchical System or Non-Uniform Memory Architecture

- Hierarchical model system is a **hybrid of** shared memory system, shared disk system and shared nothing system.
- Hierarchical model is also known as **Non-Uniform Memory Architecture (NUMA).**
- In this system each group of processor has a local memory. But processors from other groups can access memory which is associated with the other group in coherent.
- **NUMA** uses local and remote memory(Memory from other group), hence it will take longer time to communicate with each other.



(d) hierarchical

Advantages of NUMA

- Improves the scalability of the system.
- Memory bottleneck(shortage of memory) problem is minimized in this architecture.

Disadvantages of NUMA

The cost of the architecture is higher compared to other architectures.

Introduction to Query Optimization

- Query optimization is a difficult part of the query processing.
- It determines the efficient way to execute a query with different possible query plan
- It cannot be accessed directly by users once the queries are submitted to the database server or parsed by the parser.
- A query is passed to the query optimizer where optimization occurs.
- Main aim of Query Optimization is to minimize the cost function,
  I/O Cost + CPU Cost + Communication Cost
- It defines how an RDBMS can improve the performance of the query by re-ordering the operations.
- It is the process of selecting the most efficient query evaluation plan from among various strategies if the query is complex.
- It computes the same result as per the given expression, but it is a least costly way of generating result.

Importance of Query Optimization

- Query optimization provides faster query processing.
- It requires less cost per query.
- It gives less stress to the database.
- It provides high performance of the system.
- It consumes less memory.

1.4] Evaluating Parallel Query in Parallel Databases

Techniques of query Evaluation

The two techniques used in query evaluation are as follows:

1. Inter query parallelism

- This technique allows to run multiple queries on different processors simultaneously.
- Pipelined parallelism is achieved by using inter query parallelism, which improves the output of the system.

For example: If there are 6 queries, each query will take 3 seconds for evaluation. Thus, the total time taken to complete evaluation process is 18 seconds. Inter query parallelism achieves this task only in 3 seconds.

- However, Inter query parallelism is difficult to achieve every time.

2. Intra Query Parallelism

- In this technique query is divided in sub queries which can run simultaneously on different processors, this will minimize the query evaluation time.
- Intra query parallelism improves the response time of the system.

For Example: If we have 6 queries, which can take 3 seconds to complete the evaluation process, the total time to complete the evaluation process is 18 seconds. But We can achieve this task in only 3 seconds by using intra query evaluation as each query is divided in sub-queries.

1.4] Optimization of Parallel Query

- Parallel Query optimization is nothing but selecting the efficient query evaluation plan.
- Parallel Query optimization plays an important role in developing system to minimize the cost of query evaluation.

Two factors play a very important in parallel query optimization.

a) total time spent to find the best plan.
b) amount of time required to execute the plan.

1.4.1] Goals of Query optimization.

Query Optimization is done with an aim to:

- Speed up the queries by finding the queries which can give the fastest result on execution.
- Increase the performance of the system.

- Select the best query evaluation plan.
- Avoid the unwanted plan.

1.4.2] Approaches of Query Optimization.

Following are the three approaches to Query Optimization:

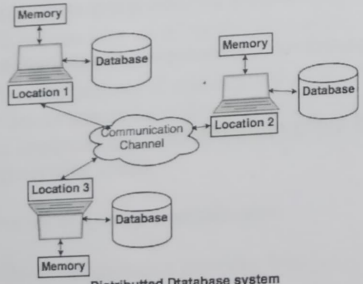
1. Horizontal partitioning: Tables are created vertically using columns.
2. Vertical partitioning: Tables are created with fewer columns and partition the table row wise.
3. De-normalization: In this approach multiple tables are combined into one table.

2] Distributed Databases

Distributed Databases system was developed to improve reliability, availability and performance of database.

2.1) What are distributed databases?

- Distributed database is a system in which storage devices are not connected to a common processing unit.
- Database is controlled by Distributed Database Management System and data may b stored at the same location or spread over the interconnected network. It is a loosely coupled system.
- Shared nothing architecture is used in distributed databases.



Distributted Dtatabase system

- The above diagram is a typical example of distributed database system, in which communication channel is used to communicate with the different locations and eve system has its own memory and database.

2.2) Goals of Distributed Database system.

The concept of distributed database was built with a goal to improve:

Reliability: In distributed database system, if one system fails down or stops working for some time another system can complete the task.
Availability: In distributed database system reliability can be achieved even if sever fails down. Another system is available to serve the client request.
Performance: Performance can be achieved by distributing database over different locations. So the databases are available to every location which is easy to maintain.

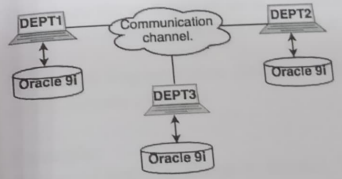
2.3) Types of distributed databases.

The two types of distributed systems are as follows:

2.3.1. Homogeneous distributed databases system:

- Homogeneous distributed database system is a network of two or more databases (With same type of DBMS software) which can be stored on one or more machines.
- So, in this system data can be accessed and modified simultaneously on severa databases in the network. Homogeneous distributed system are easy to handle.

Example: Consider that we have three departments using Oracle-9i for DBMS. If some changes are made in one department then, it would update the other department also.

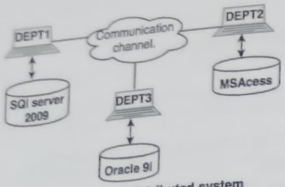


Homogeneous distributed system

2.3.2. Heterogeneous distributed database system.

- Heterogeneous distributed database system is a network of two or more databas different types of DBMS software, which can be stored on one or more machin
- In this system data can be accessible to several databases in the network with t of generic connectivity (ODBC and JDBC).

Example: In the following diagram, different DBMS software are accessible to eac other using ODBC and JDBC.



Heterogeneous distributed system

2.4 Architectures of Distributed DBMS

The basic types of distributed DBMS are as follows:

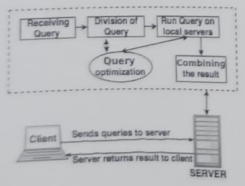
2.4.1 Client-server architecture of Distributed system.

- A client server architecture has a number of clients and a few servers connected i network.
- A client sends a query to one of the servers. The earliest available server solves it a replies.
- A Client-server architecture is simple to implement and execute due to centraliz server system.

Client-server architecture

2.4.2 Collaborating server architecture.

- Collaborating server architecture is designed to run a single query on multiple serve
- Servers break single query into multiple small queries and the result is sent to t client.
- Collaborating server architecture has a collection of database servers. Each serve capable for executing the current transactions across the databases.



Collabrorating server architecture

2.4.3. Middleware architecture.

- Middleware architectures are designed in such a way that single query is executed on multiple servers.
- This system needs only one server which is capable of managing queries and transactions from multiple servers.
- Middleware architecture uses local servers to handle local queries and transactions.
- The softwares are used for execution of queries and transactions across one or more independent database servers, this type of software is called as middleware.

Fragmentation in Distributed System

2.5) What is fragmentation?

- The process of dividing the database into a smaller multiple parts is called as **fragmentation.**
- These fragments may be stored at different locations.
- The data fragmentation process should be carrried out in such a way that the reconstruction of original database from the fragments is possible.

Types of data Fragmentation

There are three types of data fragmentation:

2.5.1. Horizontal data fragmentation

Horizontal fragmentation divides a relation(table) horizontally into the group of rows to create subsets of tables.

Example:
Account (Acc_No, Balance, Branch_Name, Type).
In this example if values are inserted in table Branch_Name as Pune, Baroda, Delhi.

The query can be written as:
SELECT*FROM ACCOUNT WHERE Branch_Name= "Baroda"

Types of horizontal data fragmentation are as follows:

1) Primary horizontal fragmentation
Primary horizontal fragmentation is the process of fragmenting a single table, row wise a set of conditions.

Example:

Acc_No	Balance	Branch_Name
A_101	5000	Pune

For the above table we can define any simple condition like, Branch_Name= 'Pune', Branch_Name= 'Delhi', Balance < 50,000

Fragmentation1:
SELECT * FROM Account WHERE Branch_Name= 'Pune' AND Balance < 50,000

Fragmentation2:
SELECT * FROM Account WHERE Branch_Name= 'Delhi' AND Balance < 50,000

2) Derived horizontal fragmentation
Fragmentation derived from the primary relation is called as derived horizontal fragmentation.

Example: Refer the example of primary fragmentation given above.

The following fragmentation are derived from primary fragmentation.

Fragmentation1:
SELECT * FROM Account WHERE Branch_Name= 'Baroda' AND Balance < 50,000

Fragmentation2:
SELECT * FROM Account WHERE Branch_Name= 'Delhi' AND Balance < 50,000

3) Complete horizontal fragmentation
- The complete horizontal fragmentation generates a set of horizontal fragmentation, which includes every table of original relation.
- Completeness is required for reconstruction of relation so that every table belongs to least one of the partitions.

4) Disjoint horizontal fragmentation
The disjoint horizontal fragmentation generates a set of horizontal fragmentation in which two fragments have common tables. That means every table of relation belongs to only one fragment.

5) Reconstruction of horizontal fragmentation
Reconstruction of horizontal fragmentation can be performed using UNION operation on fragments.

2.5.2 Vertical Fragmentation

Vertical fragmentation divides a relation(table) vertically into groups of columns to create subsets of tables.

Example:

Acc_No	Balance	Branch_Name
A_101	5000	Pune
A_102	10,000	Baroda
A_103	25,000	Delhi

Fragmentation1:
SELECT * FROM Acc_NO

Fragmentation2:
SELECT * FROM Balance

Complete vertical fragmentation
- The complete vertical fragmentation generates a set of vertical fragments, which can include all the attributes of original relation.
- Reconstruction of vertical fragmentation is performed by using **Full Outer Join** operation on fragments.

2.5.3) Hybrid Fragmentation

- Hybrid fragmentation can be achieved by performing horizontal and vertical partition together.
- Mixed fragmentation is group of rows and columns in relation.

Example: Consider the following table which consists of employee information.

Emp_ID	Emp_Name	Emp_Address	Emp_Age	Emp_Salary
101	Surendra	Baroda	25	15000
102	Jaya	Pune	37	12000
103	Jayesh	Pune	47	10000

Fragmentation1:
SELECT * FROM Emp_Name WHERE Emp_Age < 40

Fragmentation2:
SELECT * FROM Emp_Id WHERE Emp_Address= 'Pune' AND Salary < 14000

Reconstruction of Hybrid Fragmentation
The original relation in hybrid fragmentation is reconstructed by performing **UNIO and FULL OUTER JOIN.**

2.6)Data Replication in Distributed System

What is data replication?

Data replication is the process in which the data is copied at multiple locations (Different computers or servers) to improve the availability of data.

2.6.1 Goals of data replication

Data replication is done with an aim to:

- Increase the availability of data.
- Speed up the query evaluation.

2.6.2 Types of data replication

There are two types of data replication:

2.6.2.1. Synchronous Replication:
In synchronous replication, the replica will be modified immediately after some changes are made in the relation table. So there is no difference between original data and replica.

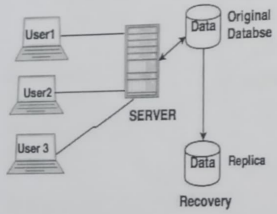
2.6.2.2 Asynchronous replication:
In asynchronous replication, the replica will be modified after commit is fired on to the database.

2.6.3 Replication Schemes

The three replication schemes are as follows:

1. Full Replication

In full replication scheme, the database is available to almost every location or user in communication network.



Full Replication Process In Distributed System

Advantages of full replication

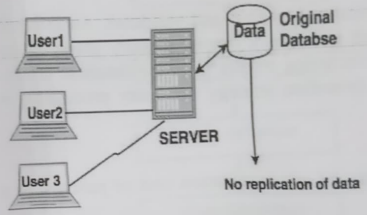
- High availability of data, as database is available to almost every location.
- Faster execution of queries.

Disadvantages of full replication

- Concurrency control is difficult to achieve in full replication.
- Update operation is slower.

2. No Replication

No replication means, each fragment is stored exactly at one location.



No Replication Process in Distributed Databases

Advantages of no replication

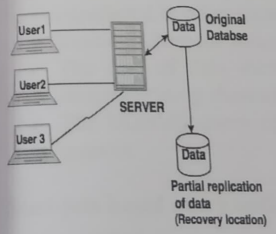
- Concurrency can be minimized.
- Easy recovery of data.

Disadvantages of no replication

- Poor availability of data.
- Slows down the query execution process, as multiple clients are accessing the same server.

3. Partial replication

Partial replication means only some fragments are replicated from the database.



Partial Replication Process In Distributed System

Advantages of partial replication

The number of replicas created for fragments depend upon the importance of data in that fragment.

Distributed databases - Query processing and Optimization

When a query is placed, it is at first scanned, parsed and validated. An interna Representation of the query is then created such as a query tree or a query graph. The alternative execution strategies are devised for retrieving results from the database table The process of choosing the most appropriate execution strategy for query processing is called query optimization.

DDBMS processes and optimizes a query in terms of communication cost of processing a distributed query and other parameters.

Query Optimization Issues in DDBMS

In DDBMS, query optimization is a crucial task. The complexity is high since number of alternative strategies may increase exponentially due to the following factors:

- The presence of a number of fragments.
- Distribution of the fragments or tables across various sites.
- The speed of communication links
- Disparity in local processing capabilities.

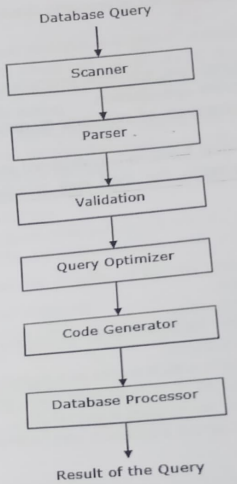
Hence, in a distributed system, the target is often to find a good execution strategy for query processing rather than the best one.

The time to execute a query is the sum of the following:

- Time to communicate queries to databases.
- Time to execute local query fragments.
- Time to assemble data from different sites.
- Time to display results to the application.

Query Processing

Query processing is a set of all activities starting from query placement to displaying the results of the query. The steps are as shown in the following diagram:



Various factors which are considered while processing a query are as follows:

Costs of Data transfer

- This is a very important factor while processing queries. The intermediate data is transferred to other location for data processing and the final result will be sent to the location where the actual query is processing.
- The cost of data increases if the locations are connected via high performance communicating channel.
- The DDBMS query optimization algorithms are used to minimize the cost of data transfer.

Semi-join based query optimization

- Semi-join is used to reduce the number of relations in a table before transferring it to another location.
- Only joining columns are transferred in this method.
- This method reduces the cost of data transfer.

Cost based query optimization

- Query optimization involves many operations like, selection, projection, aggregation.
- Cost of communication is considered in query optimization.
- In centralized database system, the information of relations at remote location is obtained from the server system catalogs.
- The data (query) which is manipulated at local location is considered as a sub query to other global locations. This process estimates the total cost which is needed to compute the intermediate relations.

Distributed Transactions

- A Distributed Databases Management System should be able to survive in a system failure without losing any data in the database.
- This property is provided in transaction processing.
- The local transaction works only on own location(Local Location) where it is considered as a global transaction for other locations.
- Transactions are assigned to transaction monitor which works as a supervisor.
- A distributed transaction process is designed to distribute data over many locations and transactions are carried out successfully or terminated successfully.
- Transaction Processing is very useful for concurrent execution and recovery of data.

Recovery in Distributed Databases
What is recovery in distributed databases?

Recovery is the most complicated process in distributed databases. Recovery of a failed system in the communication network is very difficult.

For example:
Consider that, location A sends message to location B and expects response from B but B is unable to receive it. There are several problems for this situation which are as follows.

- Message was failed due to failure in the network.
- Location B sent message but not delivered to location A.
- Location B crashed down.
- So it is actually very difficult to find the cause of failure in a large communication network.
- Distributed commit in the network is also a serious problem which can affect the recovery in a distributed databases.

two-phase protocol

algorithm which can coordinate
decide to commit or terminate the transactions. The protocol
terminate action.
The two-phase protocol ensures that all participant which are accessing the database
server can receive and implement the same action (Commit or terminate), in case of
local network failure.
Two-phase commit protocol provides automatic recovery mechanism in case of a
system failure.
The location at which original transaction takes place is called as coordinator and
where the sub process takes place is called as **Cohort.**

Commit request:
In commit phase the coordinator attempts to prepare all cohorts and take necessary steps to commit or terminate the transactions.

Commit phase:
The commit phase is based on voting of cohorts and the coordinator decides to commit or terminate the transaction.

ncurrency problems in distributed databases.

me problems which occur while accessing the database are as follows:

Failure at local locations
hen system recovers from failure the database is out dated compared to other locations. So
is necessary to update the database.

2. Failure at communication location
System should have a ability to manage temporary failure in a communicating network in distributed databases. In this case, partition occurs which can limit the communication between two locations.

3. Dealing with multiple copies of data
It is very important to maintain multiple copies of distributed data at different locations.

4. Distributed commit
While committing a transaction which is accessing databases stored on multiple locations, failure occurs on some location during the commit process then this problem is called as distributed commit.

5. Distributed deadlock

Deadlock can occur at several locations due to recovery problem and concurrency problem (multiple locations are accessing same system in the communication network).

Concurrency Controls in distributed databases

There are three different ways of making distinguish copy of data by applying:

1) Lock based protocol

A lock is applied to avoid concurrency problem between two transaction in such a way that the lock is applied on one transaction and other transaction can access it only when the lock is released. The lock is applied on write or read operations. It is an important method to avoid deadlock.

2) Shared lock system (Read lock)

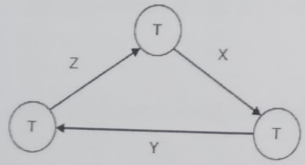
The transaction can activate shared lock on data to read its content. The lock is shared in such a way that any other transaction can activate the shared lock on the same data for reading purpose.

3) Exclusive lock

The transaction can activate exclusive lock on a data to read and write operation. In this system, no other transaction can activate any kind of lock on that same data

DDBMS Deadlock Handling

Deadlock is a state of a database system having two or more transactions, when each transaction is waiting for a data item that is being locked by some other transaction. A deadlock can be indicated by a cycle in the wait-for-graph. This is a directed graph in which the vertices denote transactions and the edges denote waits for data items .For example, in the following wait-for-graph, transaction T1 is waiting for data item X which is locked by T3. T3 is waiting for Y which is locked by T2 and T2 is waiting for Z which is locked by T1. Hence, a waiting cycle is formed, and none of the transaction scan proceeds executing.



Deadlock Prevention

The deadlock prevention approach does not allow any transaction to acquire locks that will lead to deadlocks. The convention is that when more than one transactions request for locking the same data item, only one of them is granted the lock. One of the most popular deadlock prevention methods is pre-acquisition of all the locks. In this method, a transaction acquires all the locks before starting to execute and retains the locks for the entire duration of transaction. If another transaction needs any of the already acquired locks, it has to wait until all the locks it needs are available. Using this approach, the system is prevented from being deadlocked since none of the waiting transactions are holding any lock.

Deadlock Avoidance

The deadlock avoidance approach handles deadlocks before they occur. It analyzes the transactions and the locks to determine whether or not waiting leads to a deadlock.

The method can be briefly stated as follows. Transactions start executing and request data items that they need to lock. The lock manager checks whether the lock is available. If it is available, the lock manager allocates the data item and the transaction acquires the lock. However, if the item is locked by some other transaction in incompatible mode, the lock manager runs an algorithm to test whether keeping the transaction in waiting state will cause a deadlock or not. Accordingly, the algorithm decides whether the transaction can wait or one of the transactions should be aborted. There are two algorithms for this purpose, namely wait-dieand wound-wait. Let us assume that there are two transactions, T1 and T2, where T1 tries to lock a data item which is already locked by T2. The algorithms are as follows:

- Wait-Die: If T1 is older than T2, T1 is allowed to wait. Otherwise, if T1 is younger than T2, T1 is aborted and later restarted.
- Wound-Wait: If T1 is older than T2, T2 is aborted and later restarted. Otherwise, if T1 is younger than T2, T1 is allowed to wait.

Deadlock Detection and Removal

The deadlock detection and removal approach runs a deadlock detection algorithm periodically and removes deadlock in case there is one. It does not check for deadlock when a transaction places a request for a lock. When a transaction requests a lock, the lock manager checks whether it is available. If it is available, the transaction is allowed to lock the data item; otherwise the transaction is allowed to wait. Since there are no precautions while granting lock requests, some of the transactions may be deadlocked. To detect deadlocks, the lock manager periodically checks if the wait-for-graph has cycles. If the system is deadlocked, the lock manager chooses a victim transaction from each cycle. The victim is aborted and rolled back; and then restarted later. Some of the methods used for victim selection are:

- Choose the youngest transaction.
- Choose the transaction with fewest data items.
- Choose the transaction that has performed least number of updates.