SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT – I - DISTRIBUTED DATABASE AND INFORMATION SYSTEMS- SCSA3008

SCSA3008_DISTRIBUTED DATABASE AND INFORMATION SYSTEMS

COURSE OBJECTIVES

- To understand the role of databases and database management systems in managing organizational data and information.
- To understand the techniques used for data fragmentation, replication and allocation during the distributed database design process.
- To discuss the issues involved in resource management and process.
- To Perceive the building blocks and design of information systems.
- To acquire knowledge of information systems on Business operations.

COURSE OUTCOMES

On completion of the course, student will be able to

CO1 - Identify the introductory distributed database concepts and its structures.

CO2 - Produce the transaction management and query processing techniques in DDBMS.

CO3 - Develop in-depth understanding of relational databases and skills to optimize database performance in practice.

CO4 - Critiques on each type of databases.

CO5 - Analyse, Design and present the information systems.

C06 - Designing of decision support system and tools for Business operations.

UNIT 1 9 Hrs.

INTRODUCTORY CONCEPTS AND DESIGN OF (DDBMS)

Data Fragmentation - Replication and allocation techniques for DDBMS - Methods for designing and implementing DDBMS - designing a distributed relational database - Architectures for DDBMS - Cluster federated - parallel databases and client server architecture - Overview of query processing.

UNIT 2 9 Hrs.

DISTRIBUTED SECURITY AND DISTRIBUTED DATABASE APPLICATION TECHNOLOGIES

Overview of security techniques - Cryptographic algorithms - Digital signatures - Distributed Concurrency Control - Serializability theory - Taxonomy of concurrency control mechanisms - Distributed deadlocks – Distributed Database Recovery - Distributed Data Security - Web data management - Database Interoperability.

UNIT 3 9 Hrs

ADVANCED IN DISTRIBUTED SYSTEMS

Authentication in distributed systems - Protocols based on symmetric cryptosystems - Protocols based on asymmetric cryptosystems - Password-based authentication - Unstructured overlays - Chord distributed hash table – Content addressable networks (CAN) - Tapestry - Some other challenges in P2P system design - Tradeoffs between table storage and route lengths - Graph structures of complex networks - Internet graphs - Generalized random graph networks.

UNIT 4 9 Hrs.

FUNDAMENTALAS OF INFORMATION SYSTEMS

Defining information – Classification of information – Presentation of information systems – Basics of Information systems – Functions of information systems – Components of Information systems- Limitations of Information systems – Information System Design.

UNIT 5 9 Hrs.

ENTERPRISE COLLOBRATION SYSTEMS

Groupware – Types of groupware – Enterprise Communication tools – Enterprise Conferencing tools – Collaborative work management tools – Information System for Business operations – transaction processing systems – functional Information Systems – Decision Support systems – Executive Information systems – Online Analytical processing.

UNIT 1

INTRODUCTORY CONCEPTS AND DESIGN OF (DDBMS)

Data Fragmentation - Replication and allocation techniques for DDBMS - Methods for designing and implementing DDBMS - designing a distributed relational database - Architectures for DDBMS - Cluster federated - parallel databases and client server architecture - Overview of query processing.

INTRODUCTION

- A Distributed Database (DDB) is a collection of multiple, logically interrelated databases distributed over a computer network.
- A Distributed Database Management System (D-DBMS) is the software that manages the DDB and provides an access mechanism that makes this distribution transparent to the users.
- Distributed Database System (DDBS) = DDB + D-DBMS
- Distributed Databases
    - Reality (e.g., WWW, Grids, Cloud, Sensors, Mobiles, …)

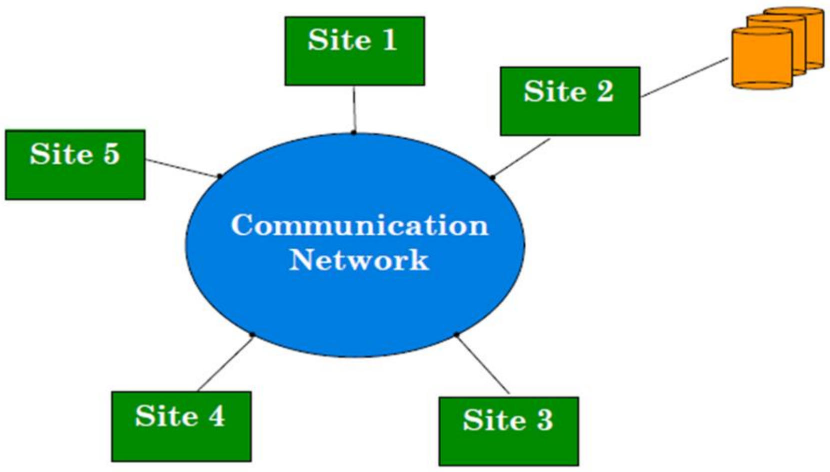


Fig 1.1 Centralized dbms on network

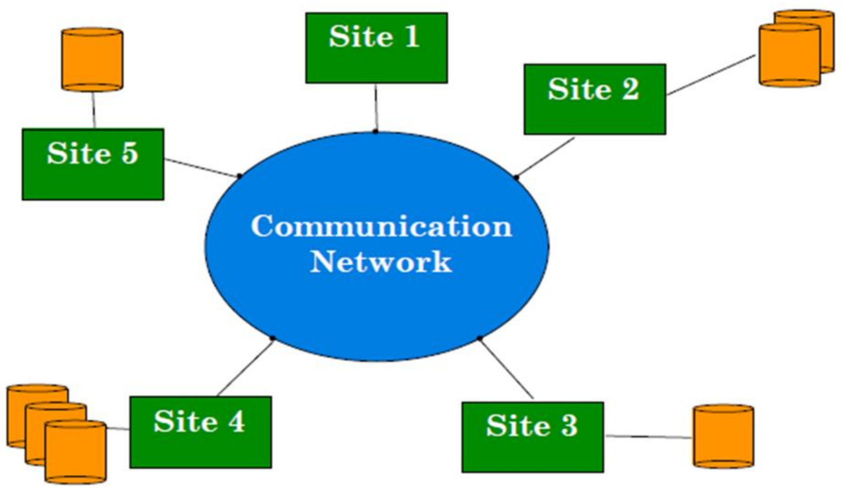


Fig 1.2 Distributed Dbms environment

Features of Distributed Databases

- Databases in the collection are logically interrelated with each other. Often they represent a single logical database.
- Data is physically stored across multiple sites.
- Data in each site can be managed by a DBMS independent of the other sites.
- The processors in the sites are connected via a network.
- They do not have any multiprocessor configuration.
- A distributed database is not a loosely connected file system.
- A distributed database incorporates transaction processing, but it is not synonymous with a transaction processing system.
- Data stored at a number of sites each site logically consists of a single processor.
- Processors at different sites are interconnected by a computer network - no multiprocessors
- Distributed database is a database, not a collection of files data logically related as exhibited in the users' access patterns
- D-DBMS is a full-fledged DBMS not remote file system, not a TP system

Advantages of Distributed Databases

- **Modular Development**
- If the system needs to be expanded to new locations or new units, in centralized database systems, the action requires substantial efforts and disruption in the existing functioning.
- However, in distributed databases, the work simply requires adding new computers and local data to the new site and finally connecting them to the distributed system, with no interruption in current functions.
- **More Reliable**
- In case of database failures, the total system of centralized databases comes to a halt.
- However, in distributed systems, when a component fails, the functioning of the system continues may be at a reduced performance. Hence DDBMS is more reliable.
- **Better Response**
- If data is distributed in an efficient manner, then user requests can be met from local data itself, thus providing faster response.
- On the other hand, in centralized systems, all queries have to pass through the central

computer for processing, which increases the response time.

- ➢ **Lower Communication Cost**
- In distributed database systems, if data is located locally where it is mostly used, then the communication costs for data manipulation can be minimized.
- This is not feasible in centralized systems.

Why Distributed Databases

- ➢ Organizational and economic reasons
- ➢ Interconnection of existing databases
- ➢ Incremental growth
- ➢ Reduced communication overhead
- ➢ Performance considerations
- ➢ Reliability and availability

Difficult of Distributed Databases

- ➢ **Need for complex and expensive software**

DDBMS demands complex and often expensive software to provide data transparency and co-ordination across the several sites.

- ➢ **Processing overhead**

Even simple operations may require a large number of communications and additional calculations to provide uniformity in data across the sites.

- ➢ **Data integrity**

The need for updating data in multiple sites pose problems of data integrity.

- ➢ **Overheads for improper data distribution**

Responsiveness of queries is largely dependent upon proper data distribution. Improper data distribution often leads to very slow response to user requests.

Data Allocation

Data Allocation is an intelligent distribution of your data pieces, (called data fragments) to improve database performance and Data Availability for end-users. It aims to reduce overall costs of transaction processing while also providing accurate data rapidly in your DDBMS systems. Data Allocation is one of the key steps in building your Distributed Database Systems. There are two common strategies used in optimal Data Allocation: Data Fragmentation and Data Replication.

Fragmentation and Replication In Distributed Database

Data Fragmentation

- ➢ Fragmentation is a process of disintegrating relations or tables into several partitions in multiple sites.
- ➢ It divides a database into various subtables and sub relations so that data can be distributed and stored efficiently.
- ➢ Database Fragmentation can be of two types: horizontal or vertical.
- ➢ In a horizontal fragmentation, each tuple of a relation r is assigned to one or more fragments.
- ➢ In vertical fragmentation, the schema for a relation r is split into numerous smaller

schemas with a common candidate key and a special attribute.

Data Replication

- Distributed Database Replication is the process of creating and maintaining multiple copies (redundancy) of data in different sites.
- The main benefit it brings to the table is that duplication of data ensures faster retrieval.
- This eliminates single points of failure and data loss issues if one site fails to deliver user requests, and hence provides you and your teams with a fault-tolerant system.
- However, Distributed Database Replication also has some disadvantages.
- To ensure accurate and correct responses to user queries, data must be constantly updated and synchronized at all times.
- Failure to do so will create inconsistencies in data, which can hamper business goals and decisions for other teams.

Methods of Data Fragmentation of a Table

Fragmentation is the task of dividing a table into a set of smaller tables. The subsets of the table are called fragments. Fragmentation can be of three types:

- Horizontal Fragmentation
- Vertical Fragmentation
- Hybrid Fragmentation(combination of horizontal and vertical).
- Horizontal fragmentation can further be classified into two techniques: primary horizontal fragmentation and derived horizontal fragmentation.

Fragmentation should be done in a way so that the original table can be reconstructed from the fragments. This is needed so that the original table can be reconstructed from the fragments whenever required. This requirement is called “reconstructiveness.”

Advantages of Fragmentation

- Since data is stored close to the site of usage, efficiency of the database system is increased.
- Local query optimization techniques are sufficient for most queries since data is locally available.
- Since irrelevant data is not available at the sites, security and privacy of the database system can be maintained.

Disadvantages of Fragmentation

- When data from different fragments are required, the access speeds may be very low.
- In case of recursive fragmentations, the job of reconstruction will need expensive techniques.
- Lack of back-up copies of data in different sites may render the database ineffective in case of failure of a site.

Vertical Fragmentation

In vertical fragmentation, the fields or columns of a table are grouped into fragments. In order to maintain reconstructiveness, each fragment should contain the primary key field(s) of the table. Vertical fragmentation can be used to enforce privacy of data.

For example, let us consider that a University database keeps records of all registered students in a Student table having the following schema.

STUDENT

Regd_No	Name	Course	Address	Semester	Fees	Marks

Now, the fees details are maintained in the accounts section. In this case, the designer will fragment the database as follows −

```
CREATE TABLE STD_FEES AS
   SELECT Regd_No, Fees
   FROM STUDENT;
```

Horizontal Fragmentation
Horizontal fragmentation groups the tuples of a table in accordance to values of one or more fields. Horizontal fragmentation should also confirm to the rule of reconstructiveness. Each horizontal fragment must have all columns of the original base table.

For example, in the student schema, if the details of all students of Computer Science Course needs to be maintained at the School of Computer Science, then the designer will horizontally fragment the database as follows −

```
CREATE COMP_STD AS
   SELECT * FROM STUDENT
   WHERE COURSE = "Computer Science";
```

Hybrid Fragmentation
In hybrid fragmentation, a combination of horizontal and vertical fragmentation techniques are used. This is the most flexible fragmentation technique since it generates fragments with minimal extraneous information. However, reconstruction of the original table is often an expensive task.

Hybrid fragmentation can be done in two alternative ways −

- At first, generate a set of horizontal fragments; then generate vertical fragments from one or more of the horizontal fragments.

- At first, generate a set of vertical fragments; then generate horizontal fragments from one or more of the vertical fragments.

Data Replication

Data replication is the process of storing separate copies of the database at two or more sites. It is a popular fault tolerance technique of distributed databases.

Advantages of Data Replication
Reliability − In case of failure of any site, the database system continues to work since a copy is available at another site(s).

Reduction in Network Load − Since local copies of data are available, query processing can be done with reduced network usage, particularly during prime hours. Data updating can be done at non-prime hours.

Quicker Response − Availability of local copies of data ensures quick query processing and consequently quick response time.

Simpler Transactions − Transactions require less number of joins of tables located at different sites and minimal coordination across the network. Thus, they become simpler in nature.

Disadvantages of Data Replication

Increased Storage Requirements − Maintaining multiple copies of data is associated with increased storage costs. The storage space required is in multiples of the storage required for a centralized system.

Increased Cost and Complexity of Data Updating − Each time a data item is updated, the update needs to be reflected in all the copies of the data at the different sites. This requires complex synchronization techniques and protocols.

Undesirable Application – Database coupling − If complex update mechanisms are not used, removing data inconsistency requires complex co-ordination at application level. This results in undesirable application – database coupling.

Some commonly used replication techniques are −

- Snapshot replication
- Near-real-time replication
- Pull replication

Types of Distributed Databases

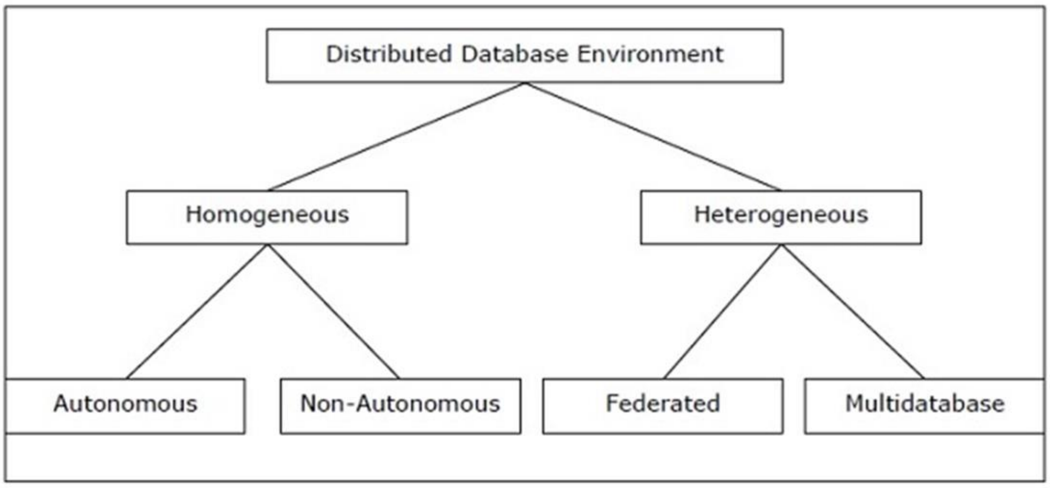


Fig 1.3 Types of Distributed Database

Homogeneous Distributed Databases

In a homogeneous distributed database, all the sites use identical DBMS and operating systems. Its properties are

- The sites use very similar software.
- The sites use identical DBMS or DBMS from the same vendor.
- Each site is aware of all other sites and cooperates with other sites to process user requests.

- The database is accessed through a single interface as if it is a single database.

Types of Homogeneous Distributed Database
There are two types of homogeneous distributed database

Autonomous − Each database is independent that functions on its own. They are integrated by a controlling application and use message passing to share data updates.

Non-autonomous − Data is distributed across the homogeneous nodes and a central or master DBMS co-ordinates data updates across the sites.

Heterogeneous Distributed Databases
In a heterogeneous distributed database, different sites have different operating systems, DBMS products and data models. Its properties are

- Different sites use dissimilar schemas and software.
- The system may be composed of a variety of DBMSs like relational, network, hierarchical or object oriented.
- Query processing is complex due to dissimilar schemas.
- Transaction processing is complex due to dissimilar software.
- A site may not be aware of other sites and so there is limited co-operation in processing user requests.

Types of Heterogeneous Distributed Databases
Federated − The heterogeneous database systems are independent in nature and integrated together so that they function as a single database system.

Un-federated − The database systems employ a central coordinating module through which the databases are accessed.

Distributed DBMS Architectures

DDBMS architectures are generally developed depending on three parameters −

Distribution − It states the physical distribution of data across the different sites.

Autonomy − It indicates the distribution of control of the database system and the degree to which each constituent DBMS can operate independently.

Heterogeneity − It refers to the uniformity or dissimilarity of the data models, system components and databases.

Architectural Models

Some of the common architectural models are −

Client - Server Architecture for DDBMS
Peer - to - Peer Architecture for DDBMS
Multi - DBMS Architecture

Client Server Architecture for DDBMS

- This is a two level architecture where the functionality is divided into servers and clients
- The server functions primarily encompass data management, query processing, optimization and transaction management
- Client functions include mainly user interface
- However, they have some functions like consistency checking and transaction management

The two different client server architecture are

- Single Server Multiple Client
- Multiple Server Multiple Client

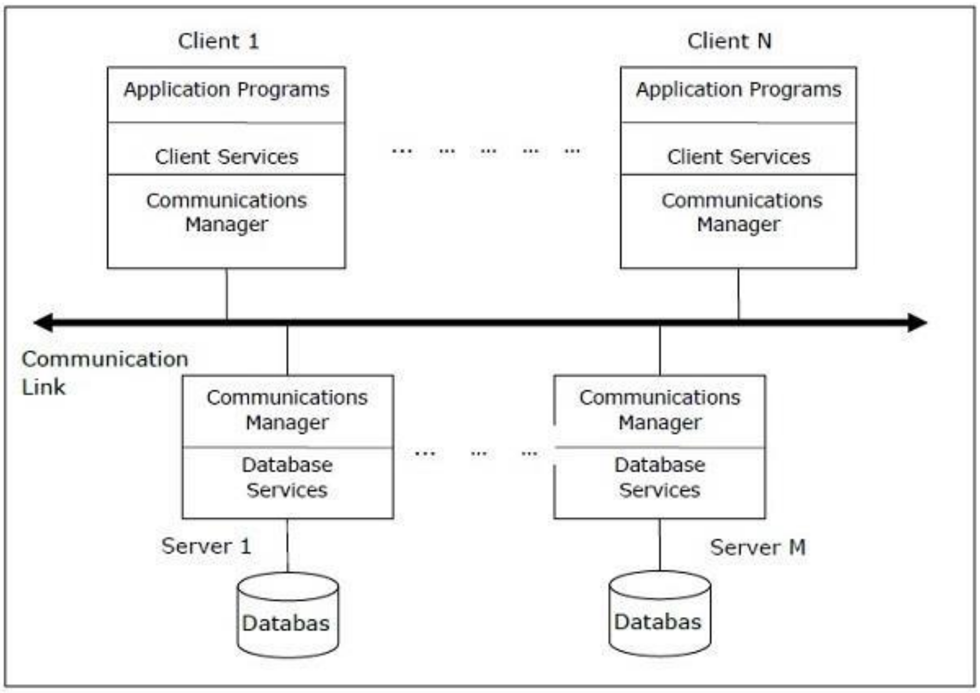
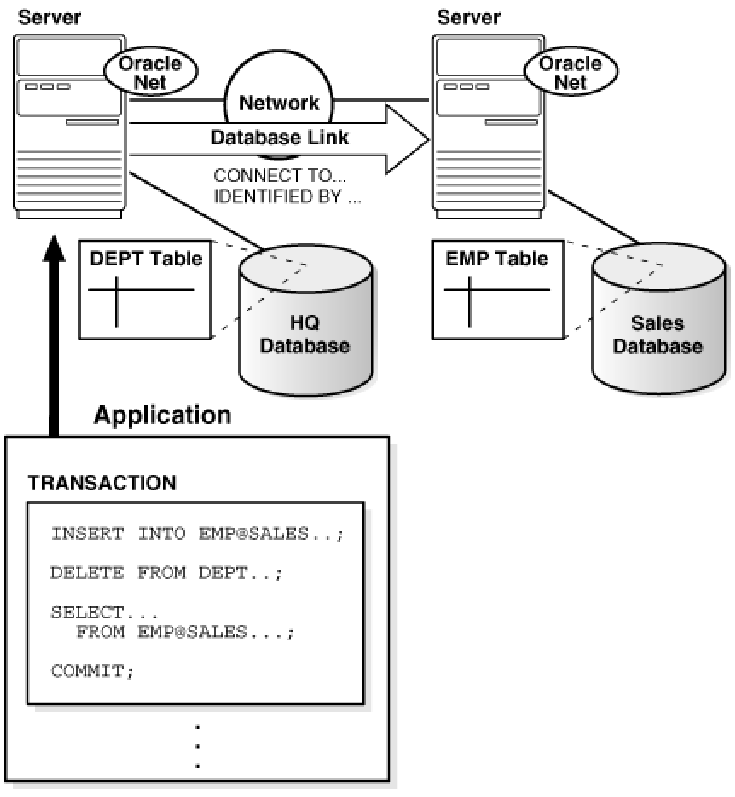


Fig 1.4 Client Server Architecture for DDBMS

Case study

- A database server is the Oracle software managing a database, and a client is an application that requests information from a server
- Each computer in a network is a node that can host one or more databases
- Each node in a distributed database system can act as a client, a server, or both, depending on the situation
- In the figure, the host for the hq database is acting as a database server when a statement is issued against its local data
- for example, the second statement in each transaction issues a statement against the local dept table, but is acting as a client when it issues a statement against remote data (for example, the first Statement in each transaction is issued against the remote table emp in the sales database)



```
INSERT INTO EMP@SALES..;

DELETE FROM DEPT..;

SELECT...
  FROM EMP@SALES...;

COMMIT;
```

Fig 1.5 Case Study

- A client can connect directly or indirectly to a database server
- A direct connection occurs when a client connects to a server and accesses information from a database contained on that server
- For example, if you connect to the hq database and access the dept table on this database as in the figure, you can issue the following
- **SELECT FROM dept**
- This query is direct because you are not accessing an object on a remote database

- In contrast, an indirect connection occurs when a client connects to a server and then accesses information contained in a database on a different server
- For example, if you connect to the hq database but access the emp table on the remote sales database as in the figure, you can issue the following
- **SELECT FROM emp@sales**
- This query is indirect because the object you are accessing is not on the database to which you are directly connected

Peer- to-Peer Architecture for DDBMS

In these systems, each peer acts both as a client and a server for imparting database services. The peers share their resource with other peers and co-ordinate their activities.
This architecture generally has four levels of schemas –
Schemas Present

- Individual internal schema definition at each site, local internal schema
- Enterprise view of data is described the global conceptual schema.

- Local organization of data at each site is describe in the local conceptual schema.
- User applications and user access to the database is supported by external schemas
- Local conceptual schemas are mappings of the global schema onto each site.
- Databases are typically designed in a top-down fashion, and, therefore all external view definitions are made globally.

Major Components of a Peer-to-Peer System
– User Processor
– Data processor

User Processor
• User-interface handler
• responsible for interpreting user commands, and formatting the result data
• Semantic data controller
• checks if the user query can be processed.
• Global Query optimizer and decomposer
• determines an execution strategy
• Translates global queries into local one.
• Distributed execution
• Coordinates the distributed execution of the user request
Data processor
• Local query optimizer
• Acts as the access path selector
• Responsible for choosing the best access path
• Local Recovery Manager
• Makes sure local database remains consistent
• Run-time support processor
• Is the interface to the operating system and contains the database buffer
• Responsible for maintaining the main memory buffers and managing the data access.

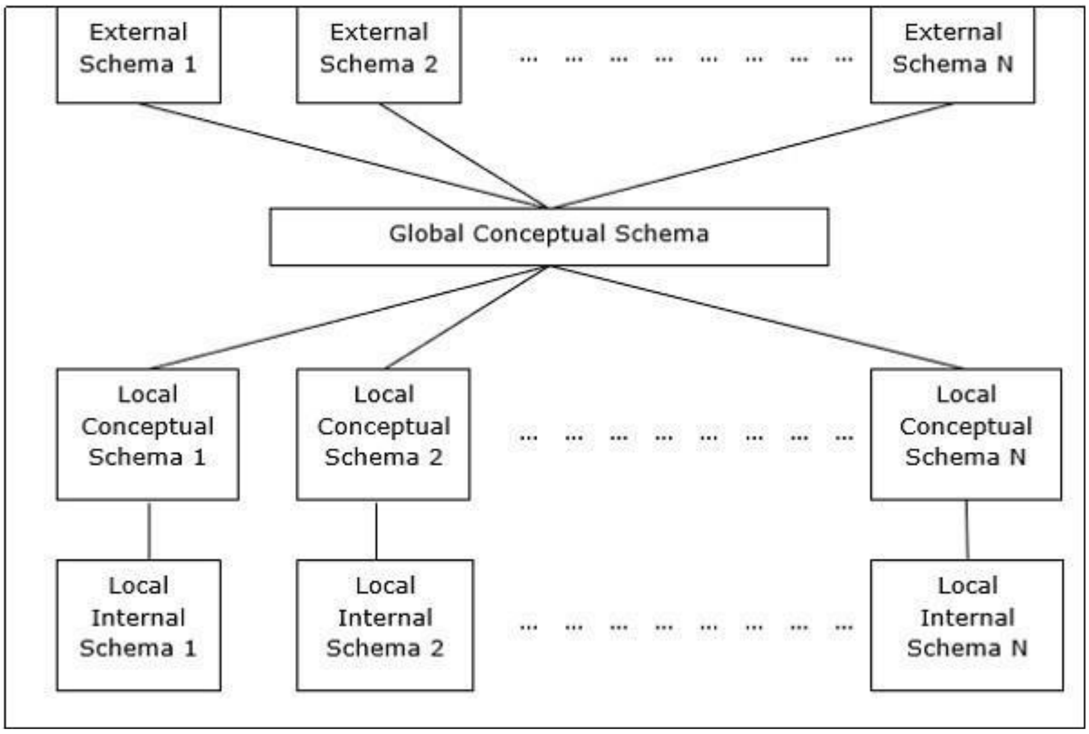


Fig 1.6 Peer to Peer Architecture

Multi - DBMS Architectures

This is an integrated database system formed by a collection of two or more autonomous database systems.

Multi-DBMS can be expressed through six levels of schemas −

• **Multi-database View Level** − Depicts multiple user views comprising of subsets of the integrated distributed database.

• **Multi-database Conceptual Level** − Depicts integrated multi-database that comprises of global logical multi-database structure definitions.

• **Multi-database Internal Level** − Depicts the data distribution across different sites and multi-database to local data mapping.

• **Local database View Level** − Depicts public view of local data.

• **Local database Conceptual Level** − Depicts local data organization at each site.

• **Local database Internal Level** − Depicts physical data organization at each site.

There are two design alternatives for multi-DBMS −

1. Model with multi-database conceptual level
2. Model without multi-database conceptual level.

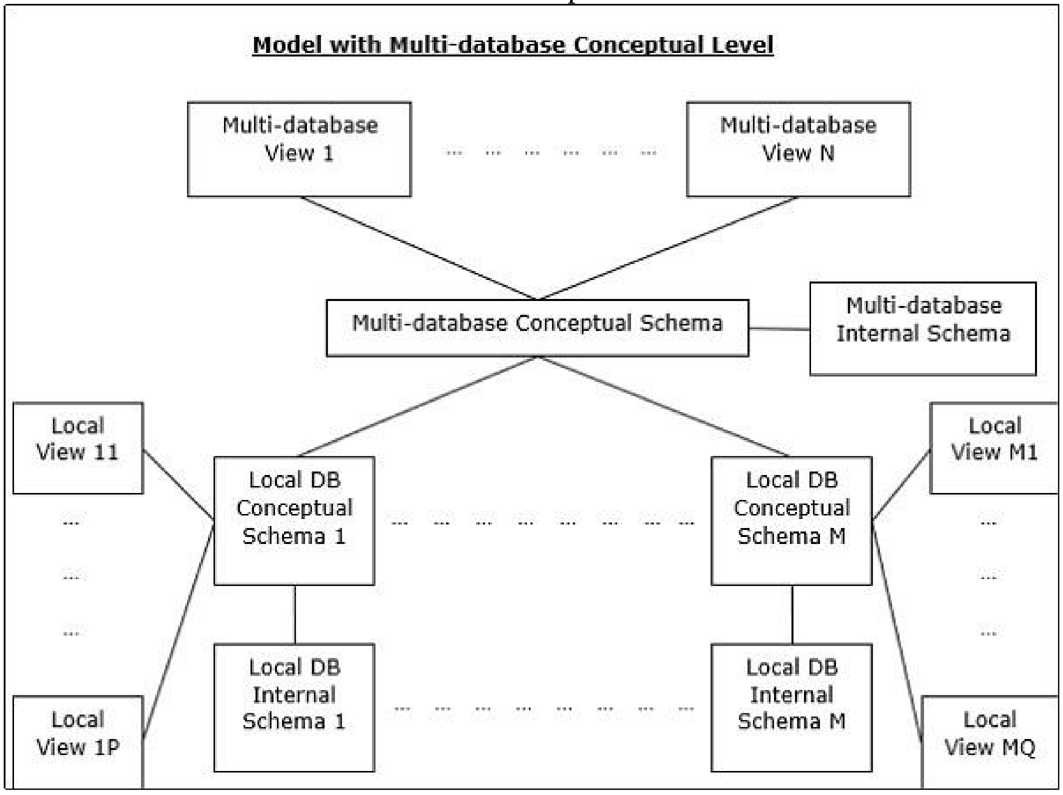

Model with Multi-database Conceptual Level
Multi-database View 1
Multi-database View N
Multi-database Conceptual Schema
Multi-database Internal Schema
Local View 11
Local View 1P
Local DB Conceptual Schema 1
Local DB Conceptual Schema M
Local View M1
Local View MQ
Local DB Internal Schema 1
Local DB Internal Schema M

Fig 1.7 Multi – DBMS Architecture

Parallel Databases

- In parallel database system data processing performance is improved by using multiple resources in parallel.
- In this CPU, the disk is used parallel to enhance the processing performance.
- Operations like data loading and query processing are performed parallel. The centralized and client server database systems are not powerful enough to handle applications which need fast processing.

- Parallel database systems have great advantages for online transaction processing and decision support applications.

- Parallel processing divides a large task into multiple tasks and each task is performed concurrently on several nodes. This gives a larger task to complete more quickly.

Architectural Models
There are several architectural models for parallel machines. The most important one are as follows −

- **Shared-memory multiple CPU** − Here, the computer has several simultaneously active CPUs that are attached to an interconnection network and share a single main memory and a common array of disk storage.

- **Shared disk architecture** − Here, each node has its own main memory but all nodes share mass storage. In practice, each node also has multiple processors.

- **Shared nothing architecture** − Here, each node has its own mass storage as well as main memory.

1. Shared Memory Architecture- In Shared Memory Architecture, there are multiple CPUs that are attached to an interconnection network. They are able to share a single or global main memory and common disk arrays. It is to be noted that, In this architecture, a single copy of a multi-threaded operating system and multithreaded DBMS can support these multiple CPUs. Also, the shared memory is a solid coupled architecture in which multiple CPUs share their memory. It is also known as Symmetric multiprocessing (SMP). This architecture has a very wide range which starts from personal workstations that support a few microprocessors in parallel via RISC.

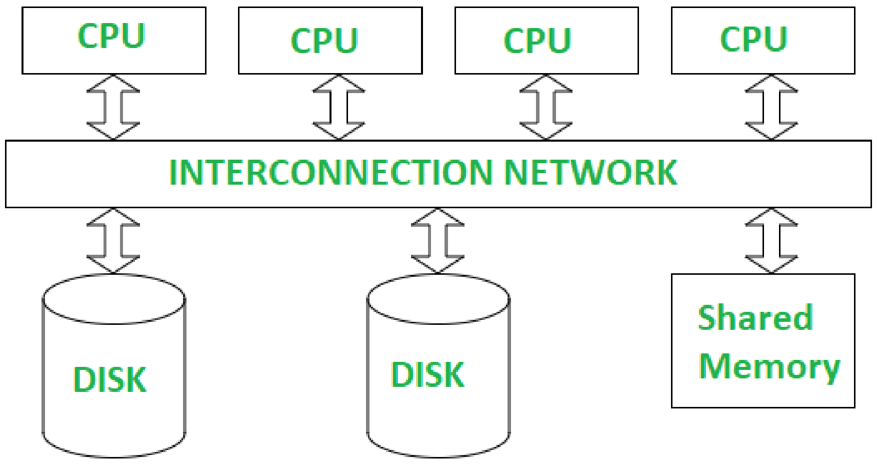


Fig 1.8 Shared Memory Architecture

Advantages :

- It has high-speed data access for a limited number of processors.

- The communication is efficient.

Disadvantages :

- It cannot use beyond 80 or 100 CPUs in parallel.
- The bus or the interconnection network gets block due to the increment of the large number of CPUs.

2. Shared Disk Architectures :

In Shared Disk Architecture, various CPUs are attached to an interconnection network. In this, each CPU has its own memory and all of them have access to the same disk. Also, note that here the memory is not shared among CPUs therefore each node has its own copy of the operating system and DBMS. Shared disk architecture is a loosely coupled architecture optimized for applications that are inherently centralized. They are also known as clusters.

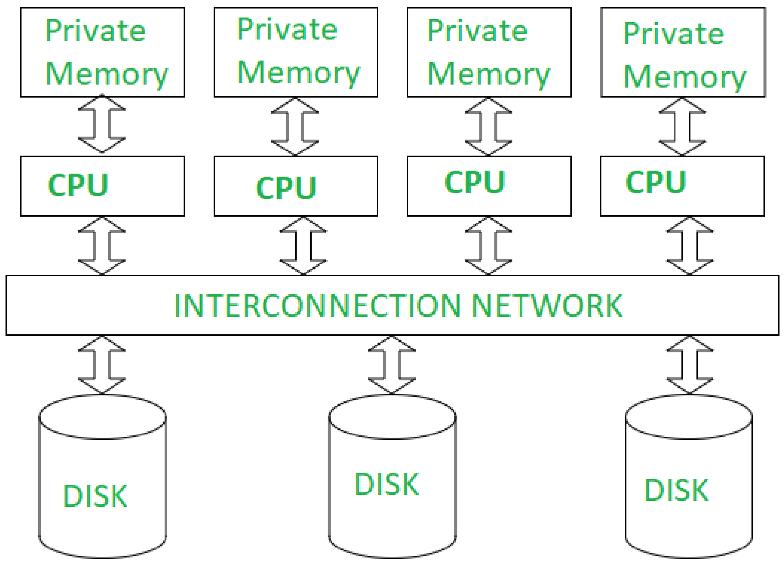


Fig 1.9 Shared Disk Architecture

Advantages :

- The interconnection network is no longer a bottleneck each CPU has its own memory.
- Load-balancing is easier in shared disk architecture.
- There is better fault tolerance.

Disadvantages :

- If the number of CPUs increases, the problems of interference and memory contentions also increase.
- There's also exists a scalability problem.

3, Shared Nothing Architecture :

Shared Nothing Architecture is multiple processor architecture in which each processor has its own memory and disk storage. In this, multiple CPUs are attached to an interconnection network through a node. Also, note that no two CPUs can access the same disk area. In this architecture, no sharing of memory or disk resources is done. It is also known as Massively parallel processing (MPP).

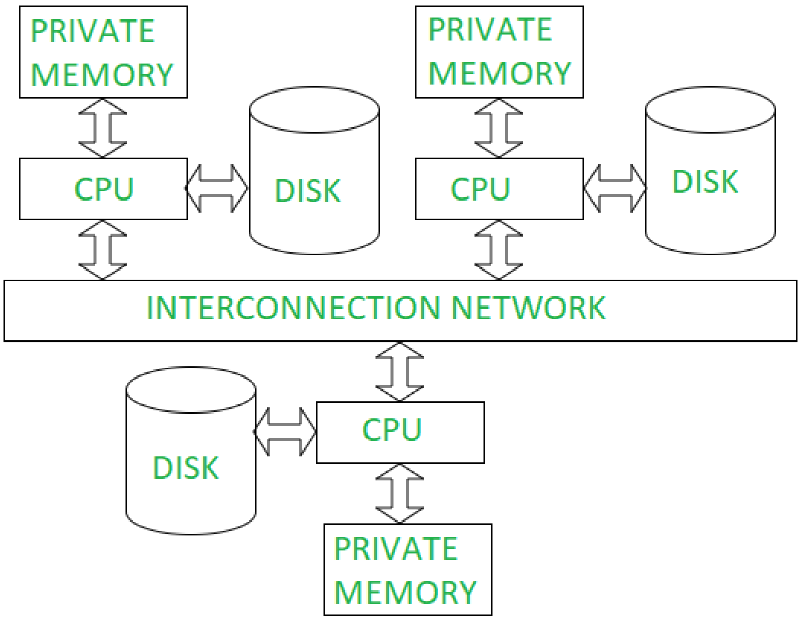


Fig 1.10 Shared Nothing Architecture

Advantages :

- It has better scalability as no sharing of resources is done
- Multiple CPUs can be added

Disadvantages:

- The cost of communications is higher as it involves sending of data and software interaction at both ends
- The cost of non-local disk access is higher than the cost of shared disk architectures.

A parallel DBMS (PDBMS) is a single DBMS running on a single cluster where the PDBMS engine automatically decides how data is distributed and queries parallelized.

Federated database system

•A federated database is a system in which several databases appear to function as a single entity. Each component database in the system is completely self-sustained and functional. When an application queries the federated database, the system figures out which of its component databases contains the data being requested and passes the request to it. Federated databases can be thought of as database virtualization in much the same way that storage virtualization makes several drives appear as one.
•A federated database may be composed of a heterogeneous collection of databases, in which

case it lets applications look at data in a more unified way without having to duplicate it across databases or make multiple queries and manually combine the results.
•In a homogeneous environment, federated databases can help distribute the load of very large databases (VLDBs). In this configuration, each component database has an identical schema but only a subset of the total rows. The federated database system distributes queries to the appropriate component database; the goal of the system is to ensure that a typical query will need to use only one component, thus drastically reducing the number of rows that need to be searched. Microsoft SQL Server has supported this type of database federation since its 2000 edition.
•When a federated database is used for load distribution, rows are distributed to its components based on a primary key. Ideally, most or all queries should end up hitting only one component database.
•For instance, a bank may use a federated database in which transactions are split by year. Users will often only look at transactions in the past year and the system will only need to touch one or two component databases. On the other hand, splitting the databases by customer ID isn't likely to work well; a given set of transactions will involve a random distribution of customer IDs, meaning that the query will be sent out to many, or potentially all, of the component databases. This eliminates the benefit of the federated database -- nearly all of the rows end up being searched -- and will only increase the query's overall latency because of the query redirects.
•Federated databases have several drawbacks, according to Hilary Cotter, a SQL Server consultant and Microsoft MVP. Each component database is a potential point of failure, and latency from any one server will delay the entire call. Your clients will have to program either the federated database or its calling applications to handle potentially incomplete query results, in case one or more of the component databases times out. They'll also have to manage each component database and keep it up to date, increasing maintenance costs.

- A federated database system (FDBS) is a type of meta-database management system (DBMS), which transparently maps multiple autonomous database systems into a single federated database.
- The constituent databases are interconnected via a computer network and may be geographically decentralized. Since the constituent database systems remain autonomous, a federated database system is a contrastable alternative to the (sometimes daunting) task of merging several disparate databases.
- A federated database, or virtual database, is a composite of all constituent databases in a federated database system.
- There is no actual data integration in the constituent disparate databases as a result of data federation.
- A DBMS can be classified as either centralized or distributed. A centralized system manages a single database while distributed manages multiple databases. A component DBS in a DBMS may be centralized or distributed.
- A multiple DBS (MDBS) can be classified into two types depending on the autonomy of the component DBS as federated and non federated.
- A nonfederated database system is an integration of component DBMS that are not autonomous.
- A federated database system consists of component DBS that are autonomous yet participate in a federation to allow partial and controlled sharing of their data.

- Federated architectures differ based on levels of integration with the component database systems and the extent of services offered by the federation. A FDBS can be categorized as loosely or tightly coupled systems.

- Loosely Coupled require component databases to construct their own federated schema. A user will typically access other component database systems by using a multidatabase language but this removes any levels of location transparency, forcing the user to have direct knowledge of the federated schema. A user imports the data they require from other component databases and integrates it with their own to form a federated schema.
- Tightly coupled system consists of component systems that use independent processes to construct and publicize an integrated federated schema.
- Multiple DBS of which FDBS are a specific type can be characterized along three dimensions: Distribution, Heterogeneity and Autonomy. Another characterization could be based on the dimension of networking, for example single databases or multiple databases in a LAN or WAN.

A federated DBMS (FDBMS) is a middleware DBMS that combines data from several different autonomous underlying DBMSs. SQL queries to the FDBMS are more or less automatically translated into queries to the underlying DBMSs.

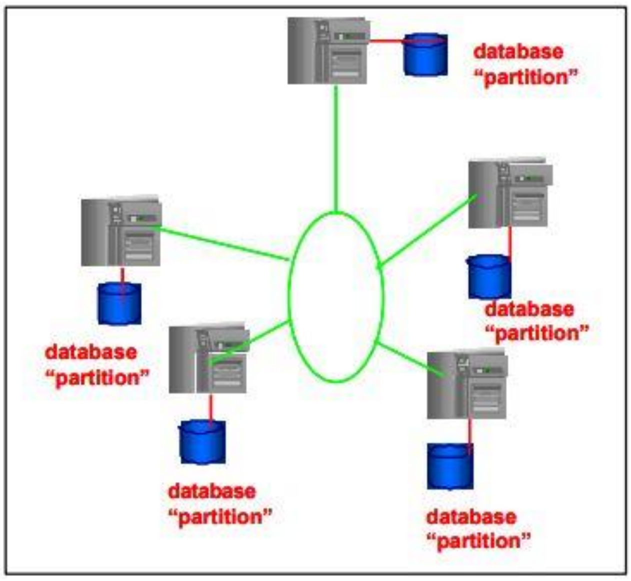


Fig 1.11 Federated DBMS

Cluster Database Architecture

A cluster consists of servers (usually SMPs), a cluster interconnect and a shared disk subsystem. Shared disk database architectures run on hardware clusters that give every participating server equal access to all disks – however, servers do not share memory. Most major hardware vendors provide shareddisk clusters today.

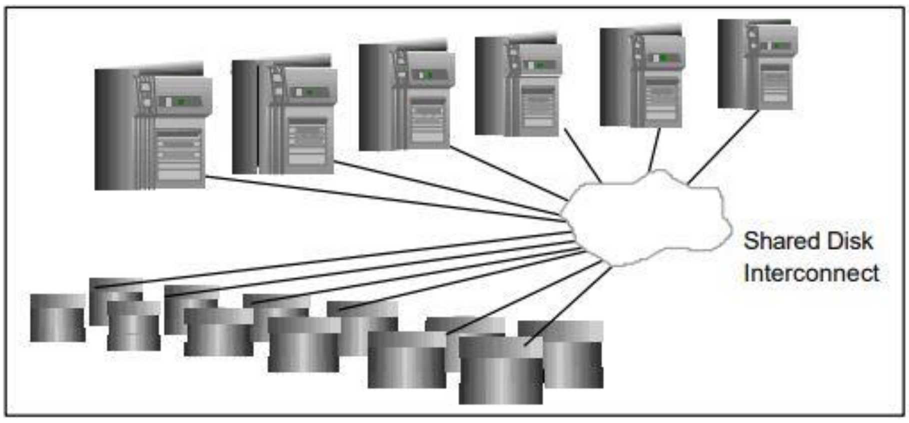


Fig 1.12 Cluster Database Architecture

A database instance runs on every node of the cluster. Transactions running on any instance can read or update any part of the database there is no notion of data ownership by a node. System performance is based on the database effectively utilizing a fast interconnect, such as the Virtual Interface Architecture (VIA), between cluster nodes. Oracle9i Real Application Clusters (RAC) is the first successful shared-disk cluster architecture and utilizes sophisticated Cache Fusion ™ shared-cache algorithms to allow high performance and scalability without data or application partitioning.

Query Processing in Distributed DBMS

Overview of query processing

- A Query processing in a distributed database management system requires the transmission of data between the computers in a network.
- A distribution strategy for a query is the ordering of data transmissions and local data processing in a database system.
- Generally, a query in Distributed DBMS requires data from multiple sites, and this need for data from different sites is called the transmission of data that causes communication costs.
- Query processing in DBMS is different from query processing in centralized DBMS due to this communication cost of data transfer over the network.
- The transmission cost is low when sites are connected through high-speed Networks and is quite significant in other networks.

1. Costs (Transfer of data) of Distributed Query processing :

- In Distributed Query processing, the data transfer cost of distributed query processing means the cost of transferring intermediate files to other sites for processing and therefore the cost of transferring the ultimate result files to the location where that result’s required.
- Let’s say that a user sends a query to site S1, which requires data from its own and also from another site S2. Now, there are three strategies to process this query which are given below:

- We can transfer the data from S2 to S1 and then process the query
- We can transfer the data from S1 to S2 and then process the query
- We can transfer the data from S1 and S2 to S3 and then process the query.

- So the choice depends on various factors like, the size of relations and the results, the communication cost between different sites, and at which the site result will be utilized.
- Commonly, the data transfer cost is calculated in terms of the size of the messages. By using the below formula, we can calculate the data transfer cost:

- Data transfer cost = C * Size

Where C refers to the cost per byte of data transferring and Size is the no. of bytes transmitted.

Example: Consider the following table EMPLOYEE and DEPARTMENT.

Site1: EMPLOYEE

EID NAME SALARY DID

EID- 10 bytes
SALARY- 20 bytes
DID- 10 bytes
Name- 20 bytes
Total records- 1000
Record Size- 60 bytes

Site2: DEPARTMENT

DID DNAME

DID- 10 bytes
DName- 20 bytes
Total records- 50
Record Size- 30 bytes

Example : Find the name of employees and their department names. Also, find the amount of data transfer to execute this query when the query is submitted to Site 3.

Answer : Considering the query is submitted at site 3 and neither of the two relations that is an EMPLOYEE and the DEPARTMENT not available at site 3. So, to execute this query, we have three strategies:

- Transfer both the tables that is EMPLOYEE and DEPARTMENT at SITE 3 then join the tables there. The total cost in this is 1000 * 60 + 50 * 30 = 60,000 + 1500 = 61500 bytes.
- Transfer the table EMPLOYEE to SITE 2, join the table at SITE 2 and then transfer the

result at SITE 3. The total cost in this is 60 * 1000 + 60 * 1000 = 120000 bytes since we have to transfer 1000 tuples having NAME and DNAME from site 1,

- Transfer the table DEPARTMENT to SITE 1, join the table at SITE 2 join the table at site1 and then transfer the result at site3. The total cost is 30 * 50 + 60 * 1000 = 61500 bytes since we have to transfer 1000 tuples having NAME and DNAME from site 1 to site 3 that is 60 bytes each.

Now, If the Optimisation criteria are to reduce the amount of data transfer, we can choose either 1 or 3 strategies from the above.

2. Using Semi join in Distributed Query processing :

The semi-join operation is used in distributed query processing to reduce the number of tuples in a table before transmitting it to another site. This reduction in the number of tuples reduces the number and the total size of the transmission that ultimately reducing the total cost of data transfer. Let's say that we have two tables R1, R2 on Site S1, and S2. Now, we will forward the joining column of one table say R1 to the site where the other table say R2 is located. This column is joined with R2 at that site. The decision whether to reduce R1 or R2 can only be made after comparing the advantages of reducing R1 with that of reducing R2. Thus, semi-join is a well-organized solution to reduce the transfer of data in distributed query processing.

Example : Find the amount of data transferred to execute the same query given in the above example using semi-join operation.

Answer : The following strategy can be used to execute the query.

Select all (or Project) the attributes of the EMPLOYEE table at site 1 and then transfer them to site 3. For this, we will transfer NAME, DID(EMPLOYEE) and the size is 25 * 1000 = 25000 bytes.
Transfer the table DEPARTMENT to site 3 and join the projected attributes of EMPLOYEE with this table. The size of the DEPARTMENT table is 25 * 50 = 1250
Applying the above scheme, the amount of data transferred to execute the query will be 25000 + 1250 = 26250 bytes.