# **Chapter 5: Distributed Database System**

# **Distributed Database Concepts**

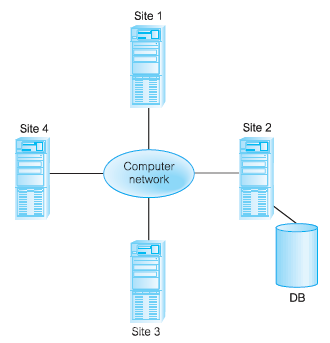
Before the development of database technology, every application used to have its own data in the application logic. Database development facilitates the integration of data available in an organization from a number of applications and enforces security on data access on a single local site. But it is not always the case that organizational data reside in one central site. This demand databases at different sites to be integrated and synchronized with all the facilities of database approach. Parallel processing is the opportunity created for computers speedy performance, besides the system should not stop processing while a site stops working. This will be made possible by computer networks and data communication optimized by internet, mobile and wireless computing and intelligent devices.

This leads to *Distributed Database Systems*. The decentralization approach to data base mirrors the natural organizational structure of companies which are logically distributed in to divisions, departments, projects and so on and physically distributed in to offices, plants, and factories each of which maintains its own operational data. *Distributed Database is not a centralized database.* Distributed DBMSs should help resolve the *islands of information* problem



***Data Distribution Strategies***

* **Distributed DB** stores logically related shared data and metadata at several physically independent sites connected via network
* **Distributed DBMS** is the software system that permits the management of a Distributed DB and makes the distribution transparent to the user.
* **Data allocation** is the process of deciding where to allocate/store particular data item.

It is important to distinguish between a distributed DBMS and distributed processing.

***There are 3 data allocation strategies:***

* Centralized,
* Partitioned,
* Replicated (Selective replication, Complete replication).

In a distributed database system, the database is logically stored as single database but physically fragmented on several computers. The computers in a distributed system communicate with each other through various communication media, such as high speed buses or telephone line.

A distributed database system consists of a collection of sites, each of which maintains a local database system (Local DBMS) but each local DBMS also participates in at least one global transaction where different databases are integrated together.

* + ***Local Transaction***: transactions that access data only in that single site
  + ***Global Transaction***: transactions that access data in several sites.

## **Data Allocation**

There are four alternative strategies regarding the placement of data:

* Centralized,
* Fragmented,
* Complete replication, and
* Selective replication.

**Centralized:** This strategy consists of a single database and DBMS stored at one site with users distributed across the network (we referred to this previously as distributed processing).Locality of reference is at its lowest as all sites, except the central site, have to use the network for all data accesses. This also means that

* **communication costs are high.**
* **Reliability and availability are low,**
* as a failure of the central site results in the loss of the entire database system.

**Fragmented (or partitioned):** This strategy partitions the database into disjoint fragments, with each fragm ent assigned to one site. If data items are located at the site where they are used most frequently, locality of reference is high.

As there is

* no replication,
* storage costs are low;
* similarly, reliability and availability are low,
* although they are higher than in the centralized case,
* as the failure of a site results in the loss of only that site’s data.
* Performance should be good and communications costs low if the distribution is designed properly.

**Complete replication:** This strategy consists of maintaining a complete copy of the database at each site.

Therefore,

* locality of reference, reliability and availability, and performance are maximized.
* However, storage costs and communication costs for updates are the most expensive.

To overcome some of these problems, **snapshots** are sometimes used. A snapshot is a copy of the data at a given time. The copies are updated periodically—for example, hourly or weekly—so they may not be always up to date. Snapshots are also sometimes used to implement views in a distributed database to improve the time it takes to perform a database operation on a view.

**Selective replication:** This strategy is a combination of fragmentation, replication, and centralization. Some data items are fragmented to achieve high locality of reference, and others that are used at many sites and are not frequently updated are replicated; otherwise, the data items are centralized. The objective of this strategy is to have all the advantages of the other approaches but none of the disadvantages. This is the most commonly used strategy, because of its flexibility.

**Comparison of strategies for data allocation**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Locality of**  **Reference** | **Reliability and**  **Availability** | **Performance** | **Storage**  **Costs** | **Communication**  **Costs** |
| **Centralized** | Lowest | Lowest | Unsatisfactory | Lowest | Highest |
| **Fragmented** | High | Low for item; high for system | Satisfactory | Lowest | Low |
| **Complete**  **Replication** | Highest | Highest | Best for read | Highest | High for update;  low for read |
| **Selective**  **replication** | High | Low for item; high for system | Satisfactory | Average | Low |

## **Fragmentation**

**Why fragment?** Before we discuss fragmentation in detail,

we list four reasons for fragmenting arelation:

* + - * **Usage***.* In general, applications work with views rather than entire relations. Therefore, for data distribution, it seems appropriate to work with subsets of relations as the unit of distribution.
      * **Efficiency***.* Data is stored close to where it is most frequently used. In addition, data that is not needed by local applications is not stored.
      * **Parallelism***.* With fragments as the unit of distribution, a transaction cans be divided into several sub queries that operate on fragments. This should increase the degree of concurrency, or parallelism, in the system, thereby allowing transactions that can do so safely to execute in parallel.
      * **Security***.* Data not required by local applications is not stored and consequently not available to unauthorized users.

attributes must be repeated to allow reconstruction. This rule ensures minimal data redundancy.

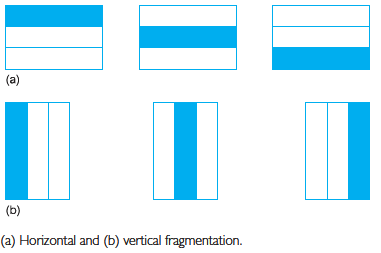
### Types of fragmentation

There are two main types of fragmentation:

* **Horizontal fragments** and
* **Vertical**. **fragments**
* Horizontal fragments are subsets of tuples and
* vertical fragments are subsets of attributes

Relation is partitioned into several fragments stored in distinct sites. The partitioning could be ***vertical***, ***horizontal*** or ***both***.

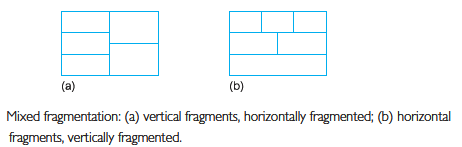
**Horizontal Fragmentation**

* Systems can share the responsibility of storing information from a single table with individual systems storing groups of rows
* Performed by the ***Selection Operation***
* The whole content of the relation is reconstructed using the ***UNION*** operation

**Vertical Fragmentation**

* Systems can share the responsibility of storing particular attributes of a table.
* ***Needs attribute with tuple number (the primary key value be repeated.)***
* Performed by the ***Projection Operation***
* The whole content of the relation is reconstructed using the ***Natural JOIN*** operation using the attribute with ***Tuple number (primary key values)***

***Both (hybrid fragmentation)***

* *A system can share the responsibility of storing* ***particular attributes*** *of a* ***subset of records*** *in a given* ***relation****.*
* *Performed by projection then selection or selection then projection relational algebra operators.*
* *Reconstruction is made by combined effect of* ***Union*** *and* ***natural join*** *operators*

## **Replication**:

* System maintains multiple copies of similar data (identical data)
* Stored in different sites, for faster retrieval and fault tolerance.
* Duplicate copies of the tables can be kept on each system (replicated). With this option, updates to the tables can become involved (of course the copies of the tables can be read-only).
* Advantage: Availability, Increased parallelism (if only reading)
* Disadvantage: increased overhead of update

# **Advantages and Disadvantages of DDBMSs**

The distribution of data and applications has potential advantages over traditional centralized database systems. Unfortunately, there are also disadvantages. In this section, we review the advantages and disadvantages of the DDBMS.

***Advantages of DDBMS***

1. **Reflects organizational structure** many organizations are naturally distributed over several locations.
2. **Many existing systems:** Maybe you have no choice. Possibly there are many different existing system, with possible different kinds of systems (Oracle, Informix, others) that need to be used together.
3. **Data sharing and distributed control: Improved sharing and local autonomy:** User at one site may be able access data that is available at another site. Each site can retain some degree of control over local data**.** We will have local as well as global database administrator
4. **Improved Reliability and availability of data:** If one site fails the rest can continue operation as long as transaction does not demand data from the failed system and the data is not replicated in other sites
5. **Improved performance - Speedup of query processing:** If a query involves data from several sites, it may be possible to split the query into sub-queries that can be executed at several sites which is parallel processing**.** Query can be sent to the least heavily loaded sites
6. **Economics:** However, it is now generally. Accepted that it costs much less to create a system of smaller computers with the equivalent power of a single large computer. This makes it more cost-effective for corporate divisions and departments to obtain separate computers. It is also much more cost-effective to add workstations to a network than to update a mainframe system. The second potential cost saving occurs where databases are geographically remote and the applications require access to distributed data. In such cases, owing to the relative expense of data being transmitted across the network as opposed to the cost of local access, it may be much more economical to partition the application and perform the processing locally at each site.
7. **Expansion** (Scalability)**:** In a distributed environment, you can easily expand by adding more machines to the network. In a distributed environment, it is much easier to handle expansion. New sites can be added to the network without affecting the operations of other sites. This flexibility allows an organization to expand relatively easily. Increasing database size can usually be handled by adding processing and storage power to the network.
8. **Integration** At the start of this section we noted that integration was a key advantage of the DBMS approach, not centralization. The integration of legacy systems is one particular example that demonstrates how some organizations are forced to rely on distributed data processing to allow their legacy systems to coexist with their more modern systems. At the same time, no one package can provide all the functionality that an organization requires nowadays. Thus, it is important for organizations to be able to integrate software components from different vendors to meet their specific requirements.
9. **Remaining competitive** There are a number of relatively recent developments that rely heavily on distributed database technology such as e-business, computer supported collaborative work, and workflow management. Many enterprises have had to reorganize their businesses and use distributed database technology to remain competitive.

***Disadvantages of DDBMS***

1. **Greater Potential for Bugs:** Parallel processing may endanger correctness of algorithms
2. **Increased Processing Overhead:** Exchange of message between sites – high communication latency**.** Due to communication jargons
3. **Complexity** A distributed DBMS that hides the distributed nature from the user and provides an acceptable level of performance, reliability, and availability is inherently more complex than a centralized DBMS. The fact that data can be replicated also adds an extra level of complexity to the distributed DBMS. If the software does not handle data replication adequately, there will be degradation in availability, reliability, and performance compared with the centralized system, and the advantages we cited earlier will become disadvantages.
4. **Cost** Increased complexity means that we can expect the procurement and maintenance costs for a DDBMS to be higher than those for a centralized DBMS. Furthermore, a distributed DBMS requires additional hardware to establish a network between sites. There are ongoing communication costs incurred with the use of this network. There are also additional labor costs to manage and maintain the local DBMSs and the underlying network.
5. **Security** In a centralized system, access to the data can be easily controlled. However, in a distributed DBMS not only does access to replicated data has to be controlled in multiple locations, but the network itself has to be made secure. In the past, networks were regarded as an insecure communication medium. Although this is still partially true, significant developments have been made to make networks more secure.
6. **Integrity control more difficult** Database integrity refers to the validity and consistency of stored data. Integrity is usually expressed in terms of constraints, which are consistency rules that the database is not permitted to violate. Enforcing integrity constraints generally requires access to a large amount of data that defines the constraint but that is not involved in the actual update operation itself. In a distributed DBMS, the communication and processing costs that are required to enforce integrity constraints may be prohibitive.
7. **Lack of standards** Although distributed DBMSs depend on effective communication, we are only now starting to see the appearance of standard communication and data access protocols. This lack of standards has significantly limited the potential of distributed DBMSs. There are also no tools or methodologies to help users convert a centralized DBMS into a distributed DBMS.
8. **Lack of experience** General-purpose distributed DBMSs have not been widely accepted, although many of the protocols and problems are well understood. Consequently, we do not yet have the same level of experience in industry as we have with centralized DBMSs. For a prospective adopter of this technology, this may be a significant deterrent.
9. **Database design more complex** Besides the normal difficulties of designing centralized database, the design of a distributed database has to take account of fragmentation of data, allocation of fragments to specific sites, and data replication.

# **Homogeneous and Heterogeneous Distributed Databases**

A DDBMS may be classified as homogeneous or heterogeneous. In a **homogeneous** system, all sites use the same DBMS product. In a **heterogeneous** system, sites may run different DBMS products, which need not be based on the same underlying data model, and so the system may be composed of relational, network, hierarchical, and object-oriented DBMSs.

* **In a homogeneous distributed database**
* All sites have identical software (DBMS)
* Are aware of each other and agree to cooperate in processing user requests.
* Each site surrenders part of its autonomy in terms of right to change schemas or software
* Appears to the user as a single system
* **In a heterogeneous distributed database**
* Different sites may use different schemas and software (DBMS)
  + - Difference in schema is a major problem for query processing
    - Difference in software is a major problem for transaction processing
* Sites may not be aware of each other and may provide only limited facilities for cooperation in transaction processing.
* May need ***gateways*** to interface one another.

# **Transparencies in a DDBMS**

Transparency hides implementation details from the user. For example, in a centralized DBMS data independence isa form of transparency—it hides changes in the definition and organization of the data from the user. A DDBMS may provide various levels of transparency. However, they all participate in the same overall objective: to make the use of the distributed database equivalent to that of a centralized database. We can identify four main types of transparency in a DDBMS:

* + - * + Distribution transparency;
        + Transaction transparency;
        + Performance transparency;
        + DBMS transparency.
      * **Data transparency:** The degree to which system user may remain unaware of the details of how and where the data items are stored in a distributed system.
      * **Distribution transparency** Even though there are many systems they appear as one- seen as a single, logical entity.
      * **Replication transparency** Copies of data floating around everywhere also seem like just one copy to the developers and users
      * **Fragmentation transparency** A table that is actually stored in parts everywhere across sites may seem like just a single table in a single
      * **Location Transparency-** the user doesn’t need to know where a data item is physically located.