

DATABASE MANAGEMENT SYSTEMS

Lecture 13 Additional Notes

Distributed Database Management Systems (DDBMS)

Advantages and Disadvantages of DDBMSs¹

Advantages

In the case of a DBMS running on a standalone PC, views are usually a convenience, defined to simplify database requests. However, in a multi-user DBMS, views play a central role in defining the structure of the database and enforcing security and integrity. The major advantages of views are described next.

Reflects organizational structure

Many organizations are naturally distributed over several locations. For example, DreamHome has many offices in different cities. It is natural for databases used in such an application to be distributed over these locations. DreamHome may keep a database at each branch office containing details of such things as the staff who work at that location, the properties that are for rent, and the clients who own or wish to rent out these properties. The staff at a branch office will make local inquiries of the database. The company headquarters may wish to make global inquiries involving the access of data at all or a number of branches.

Improved shareability and local autonomy

The geographical distribution of an organization can be reflected in the distribution of the data; users at one site can access data stored at other sites. Data can be placed at the site close to the users who normally use that data. In this way, users have local control of the data and they can consequently establish and enforce local policies regarding the use of this data. A global DBA is responsible for the entire system. Generally, part of this responsibility is devolved to the local level, so that the local DBA can manage the local DBMS.

Improved availability

In a centralized DBMS, a computer failure terminates the operations of the DBMS. However, a failure at one site of a DDBMS or a failure of a communication link making some sites inaccessible does not make the entire system inoperable. Distributed DBMSs are designed to continue to function despite such failures. If a single node fails, the system may be able to reroute the failed node's requests to another site.

¹ Source: [© Pearson - Database Systems: A Practical Approach to Design, Implementation, and Management - <http://app.kortext.com/read/80118/?page=791-794>]

Improved reliability

Because data may be replicated so that it exists at more than one site, the failure of a node or a communication link does not necessarily make the data inaccessible.

Improved performance

As the data is located near the site of “greatest demand,” and given the inherent parallelism of distributed DBMSs, speed of database access may be better than that achievable from a remote centralized database. Furthermore, since each site handles only a part of the entire database, there may not be the same contention for CPU and I/O services as characterized by a centralized DBMS.

Economics

In the 1960s, computing power was calculated according to the square of the costs of the equipment: three times the cost would provide nine times the power. This was known as Grosch’s Law. 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.

Modular growth

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. In a centralized DBMS, growth may entail changes to both

hardware (the procurement of a more powerful system) and software (the procurement of a more powerful or more configurable DBMS). The transition to the new hardware/software could give rise to many difficulties.

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.

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. For example, although more people will not necessarily rent properties just because the Internet exists, DreamHome may lose some of its market share if it does not allow clients to view properties online now.

Disadvantages

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.

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.

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 have 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.

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.

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.

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.

Database design more complex

Besides the normal difficulties of designing a 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.

Distributed Relational Database Design²

Fragmentation

The definition and allocation of fragments are carried out strategically to achieve the following objectives:

- **Locality of reference.** Where possible, data should be stored close to where it is used. If a fragment is used at several sites, it may be advantageous to store copies of the fragment at these sites.
- **Improved reliability and availability.** Reliability and availability are improved by replication: there is another copy of the fragment available at another site in the event of one site failing.
- **Acceptable performance.** Bad allocation may result in bottlenecks occurring; that is, a site may become inundated with requests from other sites, perhaps causing a significant degradation in performance. Alternatively, bad allocation may result in underutilization of resources.
- **Balanced storage capacities and costs.** Consideration should be given to the availability and cost of storage at each site, so that cheap mass storage can be used where possible. This must be balanced against locality of reference.
- **Minimal communication costs.** Consideration should be given to the cost of remote requests. Retrieval costs are minimized when locality of reference is maximized or when each site has its own copy of the data. However, when replicated data is updated, the update has to be performed at all sites holding a duplicate copy, thereby increasing communication costs.

Data Allocation

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.

² Source: [© Pearson - Database Systems: A Practical Approach to Design, Implementation, and Management - <http://app.kortext.com/read/80118/?page=807>]

Fragmented (or partitioned)

This strategy partitions the database into disjoint fragments, with each fragment 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.

	Locality of reference	Reliability & availability	Performance	Storage Costs	Communication Costs
Centralised	Lowest, except central site	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*

*Indicates: subject to good design

Date's Twelve Rules for a DDBMS³

Fundamental principle

To the user, a distributed system should look exactly like a non-distributed system.

1. Local autonomy

The sites in a distributed system should be autonomous. In this context, autonomy means that:

- local data is locally owned and managed;
- local operations remain purely local;
- all operations at a given site are controlled by that site.

2. No reliance on a central site

There should be no one site without which the system cannot operate. This implies that there should be no central servers for services such as transaction management, deadlock detection, query optimization, and management of the global system catalog.

3. Continuous operation

Ideally, there should never be a need for a planned system shutdown for operations such as:

- adding or removing a site from the system;
- the dynamic creation and deletion of fragments at one or more sites.

4. Location independence

Location independence is equivalent to location transparency. The user should be able to access the database from any site. Furthermore, the user should be able to access all data as if it were stored at the user's site, no matter where it is physically stored.

5. Fragmentation independence

The user should be able to access the data, no matter how it is fragmented.

6. Replication independence

The user should be unaware that data has been replicated. Thus, the user should not be able to access a particular copy of a data item directly and neither should the user have to specifically update all copies of a data item.

7. Distributed query processing

The system should be capable of processing queries that reference data at more than one site.

8. Distributed transaction processing

³ Source: [© Pearson - Database Systems: A Practical Approach to Design, Implementation, and Management - <http://app.kortext.com/read/80118/?page=825-826>]

The system should support the transaction as the unit of recovery. The system should ensure that both global and local transactions conform to the ACID rules for transactions: atomicity, consistency, isolation, and durability.

9. Hardware independence

It should be possible to run the DDBMS on a variety of hardware platforms.

10. Operating system independence

As a corollary to the previous rule, it should be possible to run the DDBMS on a variety of operating systems.

11. Network independence

Again, it should be possible to run the DDBMS on a variety of disparate communication networks.

12. Database independence

It should be possible to have a DDBMS made up of different local DBMSs, perhaps supporting different underlying data models. In other words, the system should support heterogeneity.

The last four rules are ideals. Because the rules are so general, and as there is a lack of standards in computer and network architectures, we can expect only partial compliance from vendors in the foreseeable future.