Chapter 19

Distributed DBMSs - Concepts and Design

Transparencies

Chapter - Objectives

- **Concepts**
- **Advantages and disadvantages of distributed databases.**
- **Functions and architecture for a DDBMS.**
- **Distributed database design.**
- **Levels of transparency.**
- **Comparison criteria for DDBMSs.**

Concepts

Distributed Database

A logically interrelated collection of shared data (and a description of this data), physically distributed over a computer network.

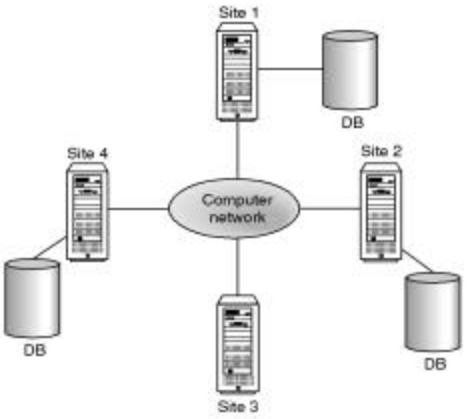
Distributed DBMS

Software system that permits the management of the distributed database and makes the distribution transparent to users.

Concepts

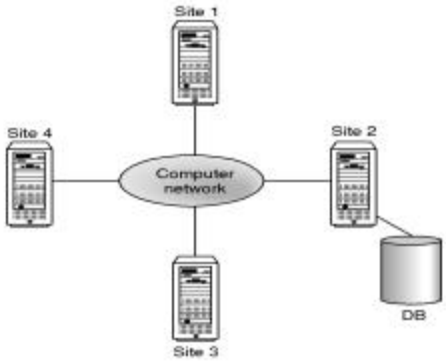
- **Collection of logically-related shared data.**
- **Data split into fragments.**
- **Fragments may be replicated.**
- **Fragments/replicas allocated to sites.**
- **Sites linked by a communications network.**
- **Data at each site is under control of a DBMS.**
- **DBMSs handle local applications autonomously.**
- **Each DBMS participates in at least one global application.**

Distributed DBMS



Distributed Processing

A centralized database that can be accessed over a computer network.



Parallel DBMS

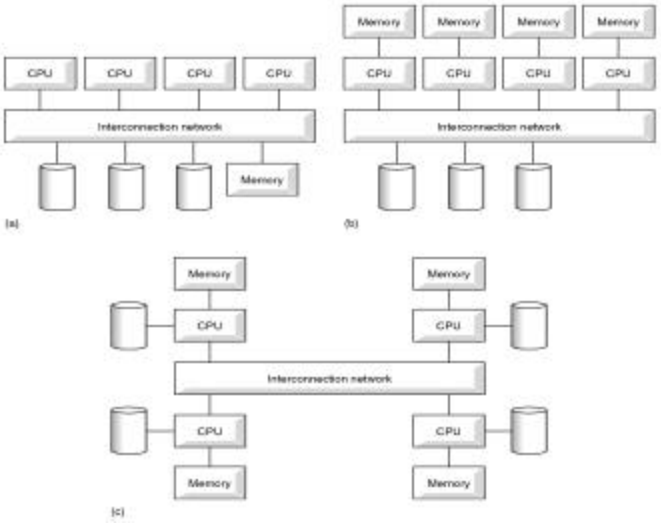
A DBMS running across multiple processors and disks designed to execute operations in parallel, whenever possible, to improve performance.

- **Based on premise that single processor systems can no longer meet requirements for cost-effective scalability, reliability, and performance.**
- **Parallel DBMSs link multiple, smaller machines to achieve same throughput as single, larger machine, with greater scalability and reliability.**

Parallel DBMS

- **Main architectures for parallel DBMSs are:**
  - **Shared memory**
  - **Shared disk**
  - **Shared nothing.**

Parallel DBMS



Advantages of DDBMSs

- **Organizational Structure**
- **Shareability and Local Autonomy**
- **Improved Availability**
- **Improved Reliability**
- **Improved Performance**
- **Economics**
- **Modular Growth**

Disadvantages of DDBMSs

- **Complexity**
- **Cost**
- **Security**
- **Integrity Control More Difficult**
- **Lack of Standards**
- **Lack of Experience**
- **Database Design More Complex**

Types of DDBMS

- **Homogeneous DDBMS**
- **Heterogeneous DDBMS**

Homogeneous DDBMS

- **All sites use same DBMS product.**
- **Much easier to design and manage.**
- **Approach provides incremental growth and allows increased performance.**

Heterogeneous DDBMS

- **Sites may run different DBMS products, with possibly different underlying data models.**
- **Occurs when sites have implemented their own databases and integration is considered later.**
- **Translations required to allow for:**
  - **Different hardware.**
  - **Different DBMS products.**
  - **Different hardware and different DBMS products.**
- **Typical solution is to use *gateways*.**

Open Database Access and Interoperability

- **Open Group has formed a Working Group to provide specifications that will create database infrastructure environment where there is:**
- **Common SQL API that allows client applications to be written that do not need to know vendor of DBMS they are accessing.**
  - **Common database protocol that enables DBMS from one vendor to communicate directly with DBMS from another vendor without the need for a gateway.**

Open Database Access and Interoperability

  - **A common network protocol that allows communications between different DBMSs.**

- **Most ambitious goal is to find a way to enable transaction to span DBMSs from different vendors without use of a gateway.**

Multidatabase System (MDBS)

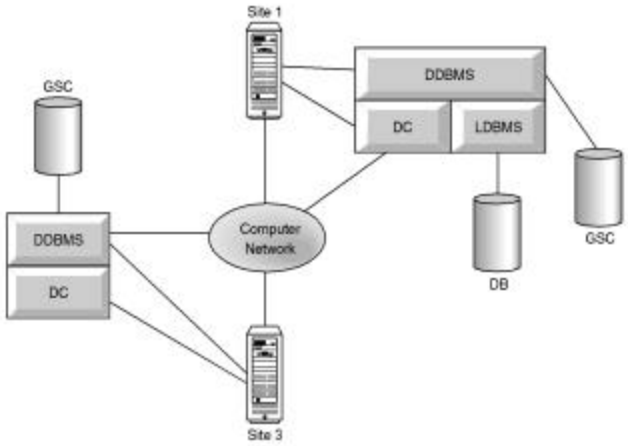
DDBMS in which each site maintains complete autonomy.

- **DBMS that resides transparently on top of existing database and file systems and presents a single database to its users.**
- **Allows users to access and share data without requiring physical database integration.**
- ***Unfederated MDBS* (no local users) and *federated MDBS*.**

Functions of a DDBMS

- **Expect DDBMS to have at least the functionality of a DBMS.**
- **Also to have following functionality:**
  - **Extended communication services.**
  - **Extended Data Dictionary.**
  - **Distributed query processing.**
  - **Extended concurrency control.**
  - **Extended recovery services.**

Components of a DDBMS



Distributed Database Design

- **Three key issues:**
  - **Fragmentation.**
  - **Allocation**
  - **Replication**

Distributed Database Design

Fragmentation

Relation may be divided into a number of sub-relations, which are then distributed.

Allocation

Each fragment is stored at site with "optimal" distribution.

Replication

Copy of fragment may be maintained at several sites.

Fragmentation

- **Definition and allocation of fragments carried out strategically to achieve:**
  - **Locality of Reference**
  - **Improved Reliability and Availability**
  - **Improved Performance**
  - **Balanced Storage Capacities and Costs**
  - **Minimal Communication Costs.**
- **Involves analyzing most important applications, based on quantitative/qualitative information.**

Fragmentation

- **Quantitative information may include:**
  - **frequency with which an application is run;**
  - **site from which an application is run;**
  - **performance criteria for transactions and applications.**
- **Qualitative information may include transactions that are executed by application, type of access (read or write), and predicates of read operations.**

Data Allocation

- **Four alternative strategies regarding placement of data:**
  - **Centralized**
  - **Partitioned (or Fragmented)**
  - **Complete Replication**
  - **Selective Replication**

Data Allocation

Centralized

Consists of single database and DBMS stored at one site with users distributed across the network.

Partitioned

Database partitioned into disjoint fragments, each fragment assigned to one site.

Data Allocation

Complete Replication

Consists of maintaining complete copy of database at each site.

Selective Replication

Combination of partitioning, replication, and centralization.

Comparison of Strategies for Data Distribution

Table 19.3 Comparison of strategies for data allocation.

	Locality of reference	*Reliability and availability*	*Performance*	*Storage costs*	*Communication costs*
Centralized	lowest	lowest	unsatisfactory	lowest	highest
Partitioned	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.

Why Fragment?

- **Usage**
  - **Applications work with views rather than entire relations.**
- **Efficiency**
  - **Data is stored close to where it is most frequently used.**
  - **Data that is not needed by local applications is not stored.**

Why Fragment?

- **Parallelism**
  - **With fragments as unit of distribution, transaction can be divided into several subqueries that operate on fragments.**
- **Security**
  - **Data not required by local applications is not stored and so not available to unauthorized users.**

Why Fragment?

- **Disadvantages**
  - **Performance**
  - **Integrity.**

Correctness of Fragmentation

- **Three correctness rules:**
  - **Completeness**
  - **Reconstruction**
  - **Disjointness.**

Correctness of Fragmentation

Completeness

If relation *R* is decomposed into fragments R_1, R_2, ... R_n, each data item that can be found in *R* must appear in at least one fragment.

Reconstruction

- **Must be possible to define a relational operation that will reconstruct *R* from the fragments.**
- **Reconstruction for horizontal fragmentation is Union operation and Join for vertical .**

Correctness of Fragmentation

Disjointness

- **If data item d_i appears in fragment R_i, then it should not appear in any other fragment.**
- **Exception: vertical fragmentation, where primary key attributes must be repeated to allow reconstruction.**
- **For horizontal fragmentation, data item is a tuple**
- **For vertical fragmentation, data item is an attribute.**

Types of Fragmentation

- **Four types of fragmentation:**
  - **Horizontal**
  - **Vertical**
  - **Mixed**
  - **Derived.**
- **Other possibility is no fragmentation:**
  - **If relation is small and not updated frequently, may be better not to fragment relation.**

Transparencies in a DDBMS

- **Distribution Transparency**
  - **Fragmentation Transparency**
  - **Location Transparency**
  - **Replication Transparency**
  - **Local Mapping Transparency**
  - **Naming Transparency**

Transparencies in a DDBMS

- **Transaction Transparency**
  - **Concurrency Transparency**
  - **Failure Transparency**
- **Performance Transparency**
  - **DBMS Transparency**
- **DBMS Transparency**

Distribution Transparency

- **Distribution transparency allows user to perceive database as single, logical entity.**
- **If DDBMS exhibits distribution transparency, user does not need to know:**
  - **data is fragmented (fragmentation transparency),**
  - **location of data items (location transparency),**
  - **otherwise call this local mapping transparency.**
- **With replication transparency, user is unaware of replication of fragments .**

Naming Transparency

- **Each item in a DDB must have a unique name.**
- **DDBMS must ensure that no two sites create a database object with same name.**
- **One solution is to create central name server. However, this results in:**
  - **loss of some local autonomy;**
  - **central site may become a bottleneck;**
  - **low availability; if the central site fails, remaining sites cannot create any new objects.**

Naming Transparency

- **Alternative solution - prefix object with identifier of site that created it.**
- **For example, Branch created at site S_1 might be named S1.BRANCH.**
- **Also need to identify each fragment and its copies.**
- **Thus, copy 2 of fragment 3 of Branch created at site S_1 might be referred to as S1.BRANCH.F3.C2.**
- **However, this results in loss of distribution transparency.**

Naming Transparency

- **An approach that resolves these problems uses aliases for each database object.**
- **Thus, S1.BRANCH.F3.C2 might be known as local_branch by user at site S_1.**
- **DDBMS has task of mapping an alias to appropriate database object.**

Transaction Transparency

- **Ensures that all distributed transactions maintain distributed database's integrity and consistency.**
- **Distributed transaction accesses data stored at more than one location.**
- **Each transaction is divided into number of subtransactions, one for each site that has to be accessed.**
- **DDBMS must ensure the indivisibility of both the global transaction and each subtransactions.**

Example - Distributed Transaction

- **T prints out names of all staff, using schema defined above as S_1, S_2, S_{21}, S_{22}, and S_{23}. Define three subtransactions T_{S3}, T_{S5}, and T_{S7} to represent agents at sites 3, 5, and 7.**

Time	T_{s_3}	T_{s_5}	T_{s_7}
t_1	begin_transaction	begin_transaction	begin_transaction
t_2	read(fname, lname)	read(fname, lname)	read(fname, lname)
t_3	print(fname, lname)	print(fname, lname)	print(fname, lname)
t_4	end_transaction	end_transaction	end_transaction

Concurrency Transparency

- **All transactions must execute independently and be logically consistent with results obtained if transactions executed one at a time, in some arbitrary serial order.**
- **Same fundamental principles as for centralized DBMS.**
- **DDBMS must ensure both global and local transactions do not interfere with each other.**
- **Similarly, DDBMS must ensure consistency of all subtransactions of global transaction.**

Concurrency Transparency

- **Replication makes concurrency more complex.**
- **If a copy of a replicated data item is updated, update must be propagated to all copies.**
- **Could propagate changes as part of original transaction, making it an atomic operation.**
- **However, if one site holding copy is not reachable, then transaction is delayed until site is reachable.**

Concurrency Transparency

- **Could limit update propagation to only those sites currently available. Remaining sites updated when they become available again.**
- **Could allow updates to copies to happen asynchronously, sometime after the original update. Delay in regaining consistency may range from a few seconds to several hours.**

Failure Transparency

- **DDBMS must ensure atomicity and durability of global transaction.**
- **Means ensuring that subtransactions of global transaction either all commit or all abort.**
- **Thus, DDBMS must synchronize global transaction to ensure that all subtransactions have completed successfully before recording a final COMMIT for global transaction.**
- **Must do this in presence of site and network failures.**

Performance Transparency

- **DDBMS must perform as if it were a centralized DBMS.**
  - **DDBMS should not suffer any performance degradation due to distributed architecture.**
  - **DDBMS should determine most cost-effective strategy to execute a request.**

Performance Transparency - Example

Table 19.4 Comparison of distributed query processing strategies.

	Strategy	*Time*
(1)	Move Renter relation to London and process query there.	16.7 minutes
(2)	Move Property and Viewing relations to Glasgow and process query there.	28 hours
(3)	Join Property and Viewing relations at London, select tuples for Aberdeen properties and for each of these in turn, check at Glasgow to determine if associated Max_Price > £200,000.	2.3 days
(4)	Select renters with Max_Price > £200,000 at Glasgow and for each one found, check at London for a viewing involving that renter and an Aberdeen property.	20 seconds
(5)	Join Property and Viewing relations at London, select Aberdeen properties and project result over Pno and Rno and move this result to Glasgow for matching with Max_Price > £200,000.	16.7 minutes
(6)	Select renters with Max_Price > £200,000 at Glasgow and move the result to London for matching with Aberdeen properties.	1 second

Date's 12 Rules for a DDBMS

0. Fundamental Principle

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

1. Local Autonomy
2. No Reliance on a Central Site
3. Continuous Operation
4. Location Independence
5. Fragmentation Independence
6. Replication Independence

Date's 12 Rules for a DDBMS

7. Distributed Query Processing
8. Distributed Transaction Processing
9. Hardware Independence
10. Operating System Independence
11. Network Independence
12. Database Independence

- **Last four rules are ideals.**