

Unit 6

Distributed Databases



5th Edition

Elmasri / Navathe

Distributed Database Concepts

- A transaction can be executed by multiple networked computers in a unified manner.
- A **distributed database (DDB)** processes Unit of execution (a transaction) in a distributed manner.
A distributed database (DDB) can be defined as
 - A distributed database (DDB) is a collection of multiple logically related database distributed over a computer network, and a distributed database management system as a software system that manages a distributed database while making the distribution transparent to the user.

Differences between DDB and Multiprocessor Systems

- shared storage (primary memory or disk).

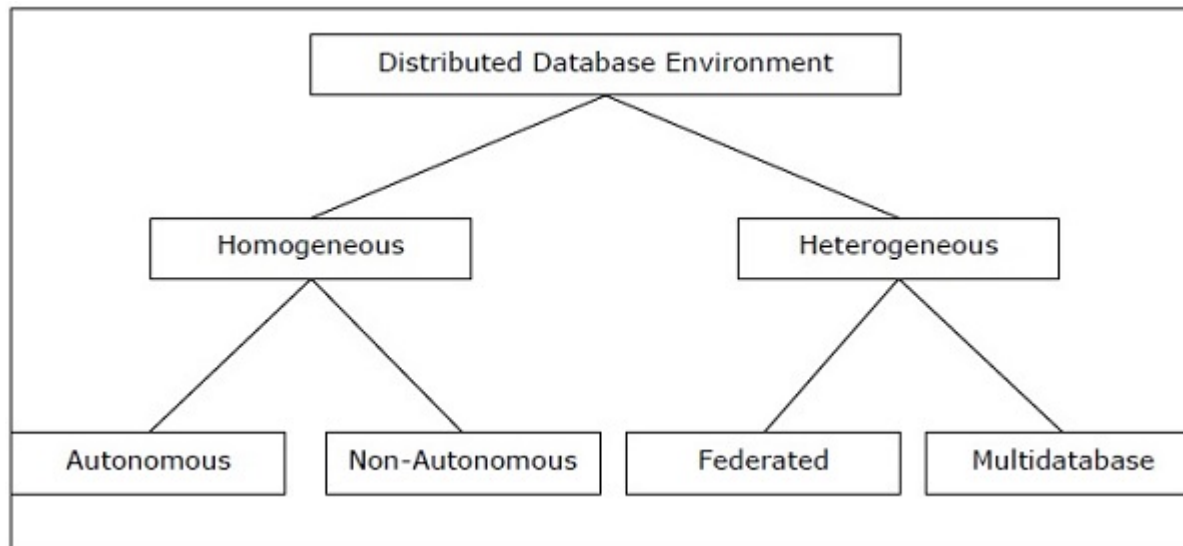
For a database to be called distributed, the following minimum conditions should be satisfied:

- **Connection of database nodes over a computer network.** There are multiple computers, called **sites** or **nodes**. These sites must be connected by an underlying **communication network** to transmit data and commands among sites, as shown later in Figure (c).
- **Logical interrelation of the connected databases.** It is essential that the information in the databases be logically related.
- **Absence of homogeneity constraint among connected nodes.** It is not necessary that all nodes be identical in terms of data, hardware, and software.

Reasons for Distributed Database

- Business unit autonomy and distribution
- Data sharing
- Data communication costs
- Data communication reliability and costs
- Multiple application vendors
- Database recovery
- Transaction and analytic processing

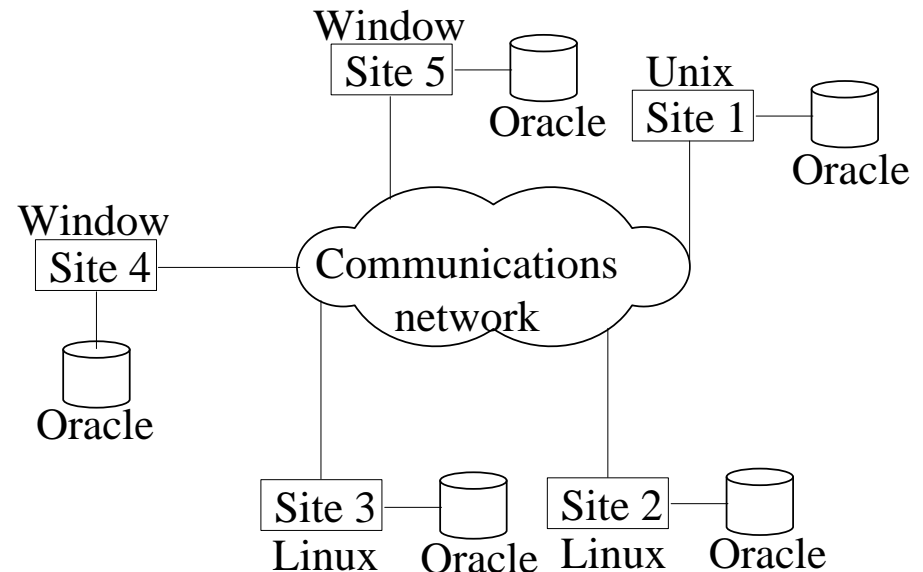
Types of DDBS



Types of Distributed Database Systems

■ Homogeneous

- All sites of the database system have identical setup, i.e., same database system software.
- The underlying operating system may be different.
 - For example, all sites run Oracle or DB2, or Sybase or some other database system.
- The underlying operating systems can be a mixture of Linux, Window, Unix, etc.



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.

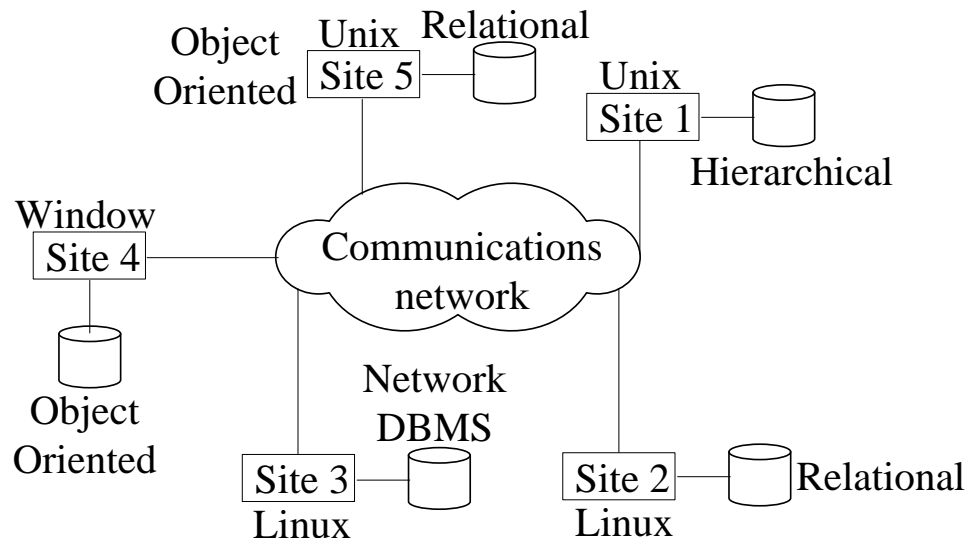
Types of Homogeneous Distributed Database

- 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.

Types of Distributed Database Systems

■ Heterogeneous

- Federated: Each site may run different database system but the data access is managed through a single conceptual schema.
 - There is some global view or schema of the federation of databases that is shared by the applications.
- Multidatabase: There is no one conceptual global schema. For data access a schema is constructed dynamically as needed by the application software.



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.

Types of Distributed Database Systems

Semantic Heterogeneity. Semantic heterogeneity occurs when there are differences in the meaning, interpretation, Transaction and policy constraints, Derivation of summaries and intended use of the same or related data. Semantic heterogeneity among component database systems (DBSs) creates the biggest hurdle in designing global schemas of heterogeneous databases.

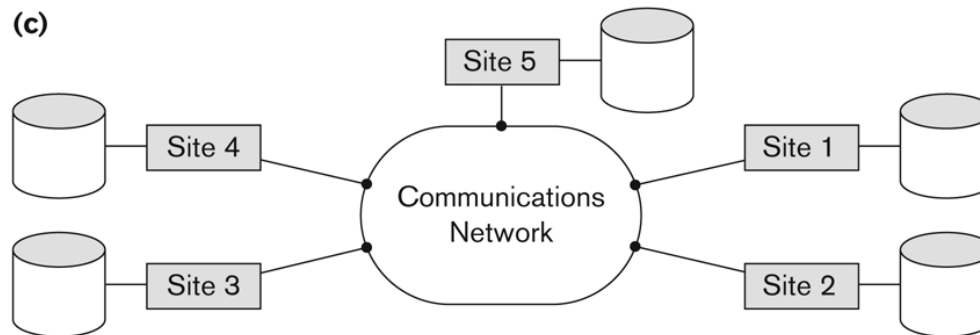
Distributed Database System

- Advantages

- Management of distributed data with different **levels of transparency:**
 - This refers to the physical placement of data (files, relations, etc.) which is not known to the user (distribution transparency).

Figure 25.1

Some different database system architectures. (a) Shared nothing architecture. (b) A networked architecture with a centralized database at one of the sites. (c) A truly distributed database architecture.

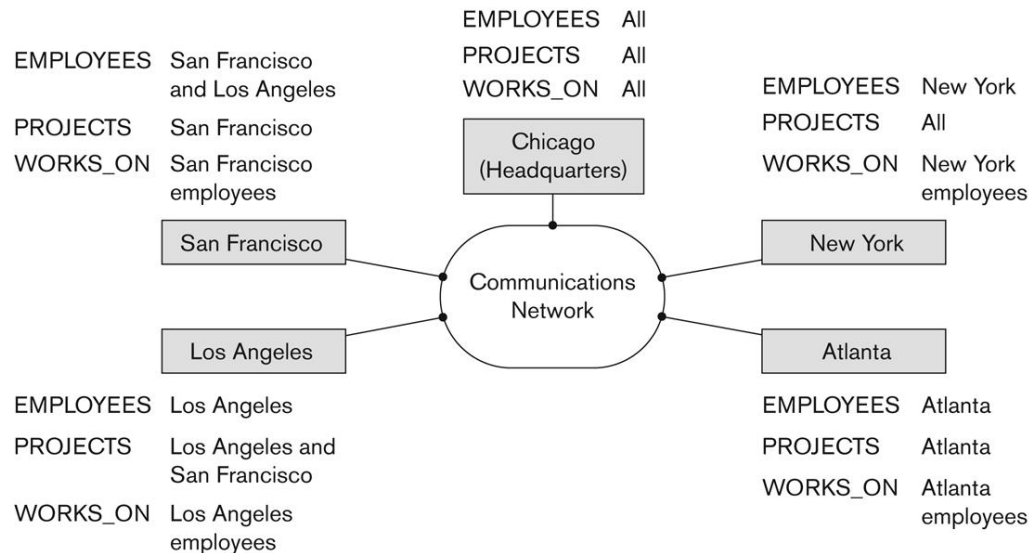


Distributed Database System

- Advantages (transparency, contd.)
 - The EMPLOYEE, PROJECT, and WORKS_ON tables may be fragmented horizontally and stored with possible replication as shown below.

Figure 25.2

Data distribution and replication among distributed databases.



Distributed Database System

- Advantages (transparency, contd.)
 - **Distribution and Network transparency:**
 - Users do not have to worry about operational details of the network.
 - There is Location transparency, which refers to freedom of issuing command from any location without affecting its working.
 - Then there is Naming transparency, which allows access to any names object (files, relations, etc.) from any location.

Distributed Database System

- Advantages (transparency, contd.)
 - **Replication transparency:**
 - It allows to store copies of a data at multiple sites.
 - This is done to minimize access time to the required data.
 - **Fragmentation transparency:**
 - Allows to fragment a relation horizontally (create a subset of tuples of a relation) or vertically (create a subset of columns of a relation).

Distributed Database System

- Other Advantages

- **Increased reliability and availability:**

- Reliability refers to system live time, that is, system is running efficiently most of the time. Availability is the probability that the system is continuously available (usable or accessible) during a time interval.
 - A distributed database system has multiple nodes (computers) and if one fails then others are available to do the job.

Distributed Database System

- Other Advantages (contd.)
 - **Improved performance:**
 - A distributed DBMS fragments the database to keep data closer to where it is needed most.
 - This reduces data management (access and modification) time significantly.
 - **Easier expansion (scalability):**
 - Allows new nodes (computers) to be added anytime without chaining the entire configuration.

Data Fragmentation, Replication and Allocation

- **Data Fragmentation**

- Split a relation into logically related and correct parts. A relation can be fragmented in two ways:
 - **Horizontal Fragmentation**
 - **Vertical Fragmentation**

Select Operation – Example

■ Relation r

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

■ $\sigma_{A=B \wedge D > 5}(r)$

A	B	C	D
α	α	1	7
β	β	23	10

Project Operation – Example

Relation r :

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

$\Pi_{A,C}(r)$

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

fragmentation

Example

Data Fragmentation, Replication and Allocation

- **Fragmentation schema**

- A definition of a set of fragments (horizontal or vertical or horizontal and vertical) that includes all attributes and tuples in the database that satisfies the condition that the whole database can be reconstructed from the fragments by applying some sequence of UNION (or OUTER JOIN) and UNION operations.

- **Allocation schema**

- It describes the distribution of fragments to sites of distributed databases. It can be fully or partially replicated or can be partitioned.

- **Global Schema : Specification of global relations**

Data Fragmentation, Replication and Allocation

■ Data Replication

- Database is replicated to all sites.
- **Full replication** the entire database is replicated
- **No replication**
- **partial replication** some selected part is replicated to some of the sites.
- Data replication is achieved through a replication schema.

■ Several factors influence the decision to use data replication:

- Database size.: band width, transmission cost etc.
- Usage frequency: to decide the frequency of data updation
- Costs, including those for performance, software overhead, and management associated with synchronizing transactions and their components vs. fault- tolerance benefits that are associated with replicated data.
- Data replication information is stored in the distributed data catalog (DDC), Whose contents are used by the TP to decide which copy of a database fragment to access

■ **Data Distribution (Data Allocation)**

The selected portion of the database is distributed to the database sites.

- Data allocation describes the process of deciding where to locate data. Data allocation strategies are as follows:
 - With centralized data allocation, the entire database is stored at one site.
 - With partitioned data allocation , the database is divided into two or more disjointed parts (fragments) and stored at two or more sites.
 - With replicated data allocation , copies of one or more database fragments are
 - stored at several sites.

Architecture of Distributed Databases

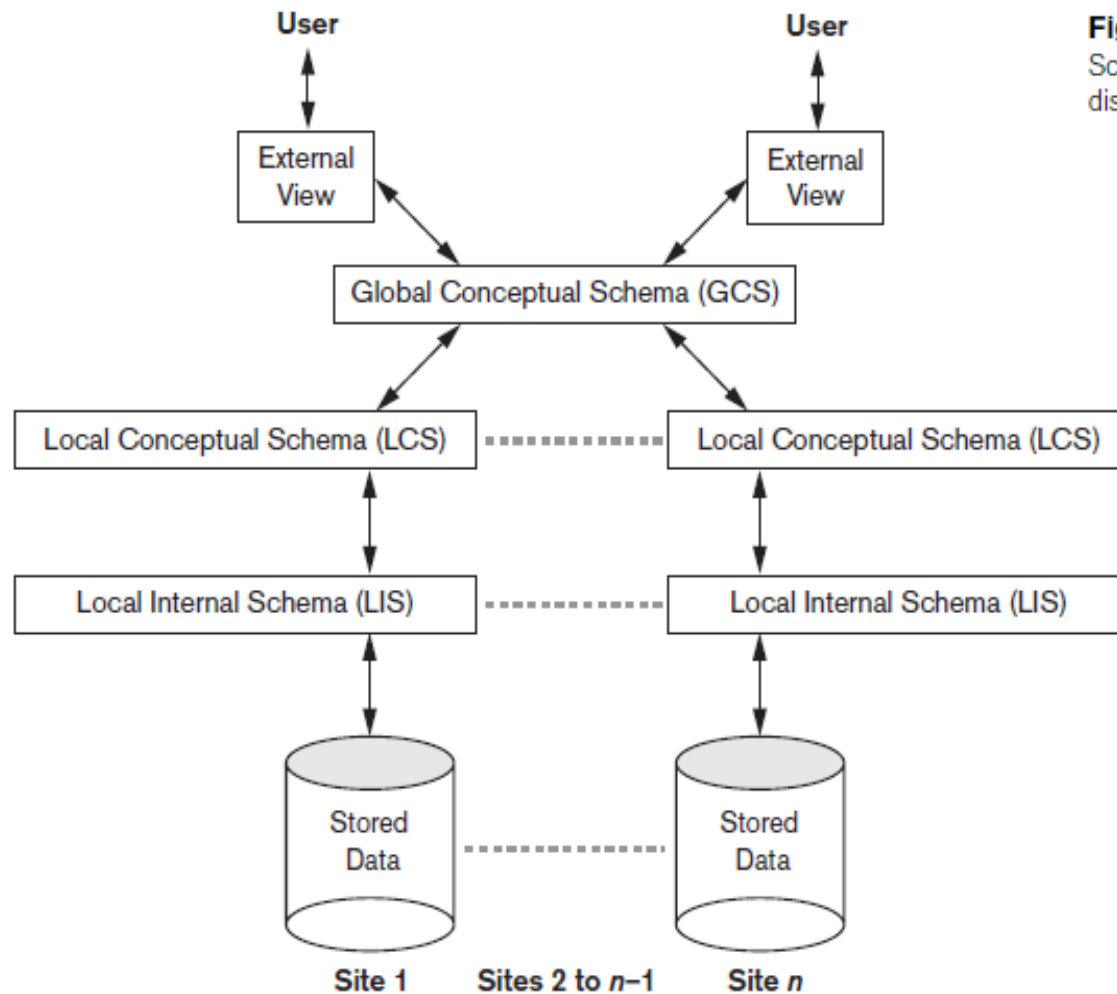


Figure 25.4
Schema architecture of
distributed databases.

Architecture of Distributed Databases

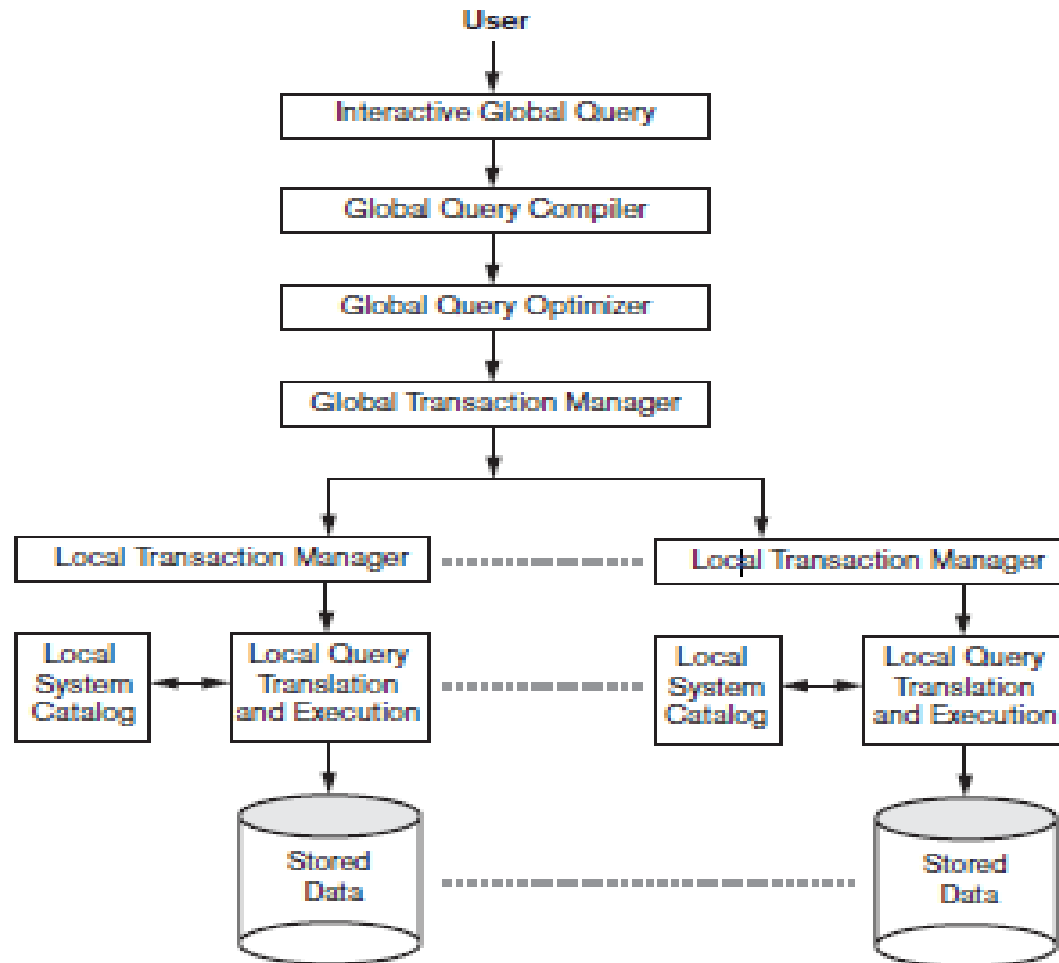


Figure 25.5
Component architecture
of distributed databases.

Concurrency Control and Recovery

- Distributed Databases encounter a number of concurrency control and recovery problems which are not present in centralized databases. Some of them are listed below.
 - Dealing with multiple copies of data items
 - Failure of individual sites
 - Communication link failure
 - Distributed commit
 - Distributed deadlock

Concurrency Control and Recovery

■ Details

- Dealing with multiple copies of data items:
 - The concurrency control must maintain global consistency. Likewise the recovery mechanism must recover all copies and maintain consistency after recovery.
- Failure of individual sites:
 - Database availability must not be affected due to the failure of one or two sites and the recovery scheme must recover them before they are available for use.

Concurrency Control and Recovery

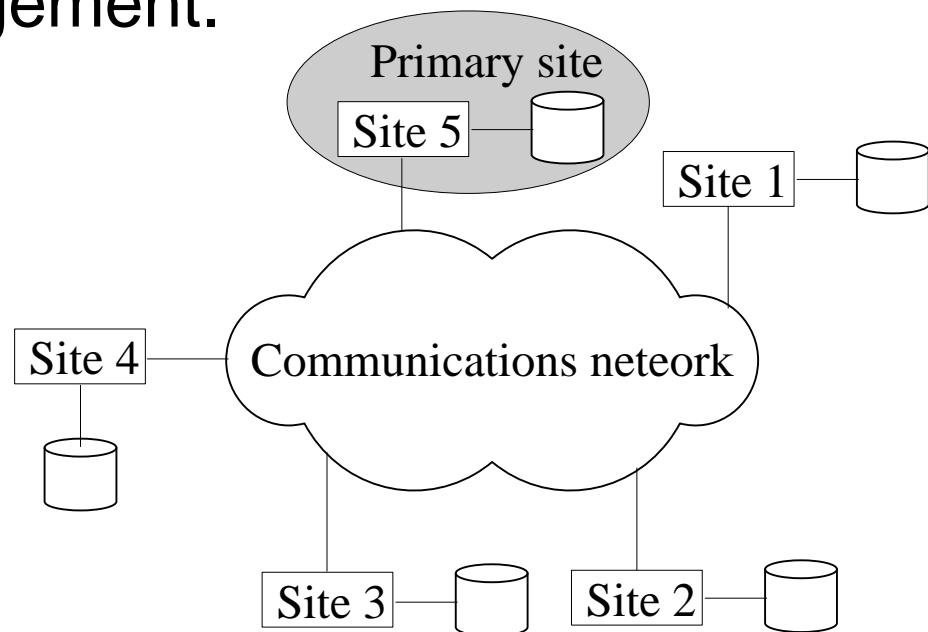
- Details (contd.)
 - Communication link failure:
 - This failure may create network partition which would affect database availability even though all database sites may be running.
 - Distributed commit:
 - A transaction may be fragmented and they may be executed by a number of sites. This require a two or three-phase commit approach for transaction commit.
 - Distributed deadlock:
 - Since transactions are processed at multiple sites, two or more sites may get involved in deadlock. This must be resolved in a distributed manner.

Concurrency control

- Based on distinguished copy of data item.
- Distinguished copy holds lock and unlock on that data item.

Concurrency Control and Recovery

- Distributed Concurrency control based on a distributed copy of a data item
 1. **Primary site technique:** A single site is designated as a primary site which serves as a coordinator for transaction management.



Concurrency Control and Recovery

- Transaction management:
 - Concurrency control and commit are managed by this site.
 - In two phase locking, this site manages locking and releasing data items. If all transactions follow two-phase policy at all sites, then serializability is guaranteed.

Concurrency Control and Recovery

■ Transaction Management

■ Advantages:

- An extension to the centralized two phase locking so implementation and management is simple.
- Data items are locked only at one site but they can be accessed at any site.

- once a transaction obtains a Read_lock on a data item from the primary site, it can access any copy of that data item. However, once a transaction obtains a Write_lock and updates a data item, the DDBMS is responsible for updating *all copies of the data item before releasing the lock*.

■ Disadvantages:

- All transaction management activities go to primary site which is likely to overload the site.
- If the primary site fails, the entire system is inaccessible.
- To aid recovery a backup site is designated which behaves as a shadow of primary site. In case of primary site failure, backup site can act as primary site.

Concurrency Control and Recovery

2. Primary Copy Technique:

- This method attempts to distribute the load of lock coordination among various sites by having the distinguished copies of different data items *stored at different sites*.
- *Failure of one site affects any transactions that are* accessing locks on items whose primary copies reside at that site, but other transactions are not affected.
- Advantages:
 - Since primary copies are distributed at various sites, a single site is not overloaded with locking and unlocking requests.
- Disadvantages:
 - Identification of a primary copy is complex. A distributed directory must be maintained, possibly at all sites.

Concurrency Control and Recovery

- Recovery from a coordinator failure
 - In both approaches a coordinator site or copy may become unavailable. This will require the selection of a new coordinator.
- Primary site approach with no backup site:
 - Aborts and restarts all active transactions at all sites. Elects a new coordinator and initiates transaction processing.
- Primary site approach with backup site:
 - All locking information is maintained at both the primary and the backup sites. In case of primary site failure, the backup site takes over as the primary site, and a new backup site is chosen.
- Primary and backup sites fail or no backup site:
 - Use election process to select a new coordinator site.

Concurrency Control and Recovery

3. Concurrency control based on voting:

- *There is no primary copy of coordinator.*
- No distinguished copy; rather, a lock request is sent to all sites that includes a copy of the data item.
- Each copy maintains its own lock and can grant or deny the request for it.
- If a transaction that requests a lock is granted that lock by a *majority of the copies, it holds the lock and informs all copies that it has been granted the lock.*
- If a transaction does not receive a majority of votes granting it a lock within a certain *time-out period, it cancels its request and informs all sites of the cancellation.*
- The voting method is considered a truly distributed concurrency control method

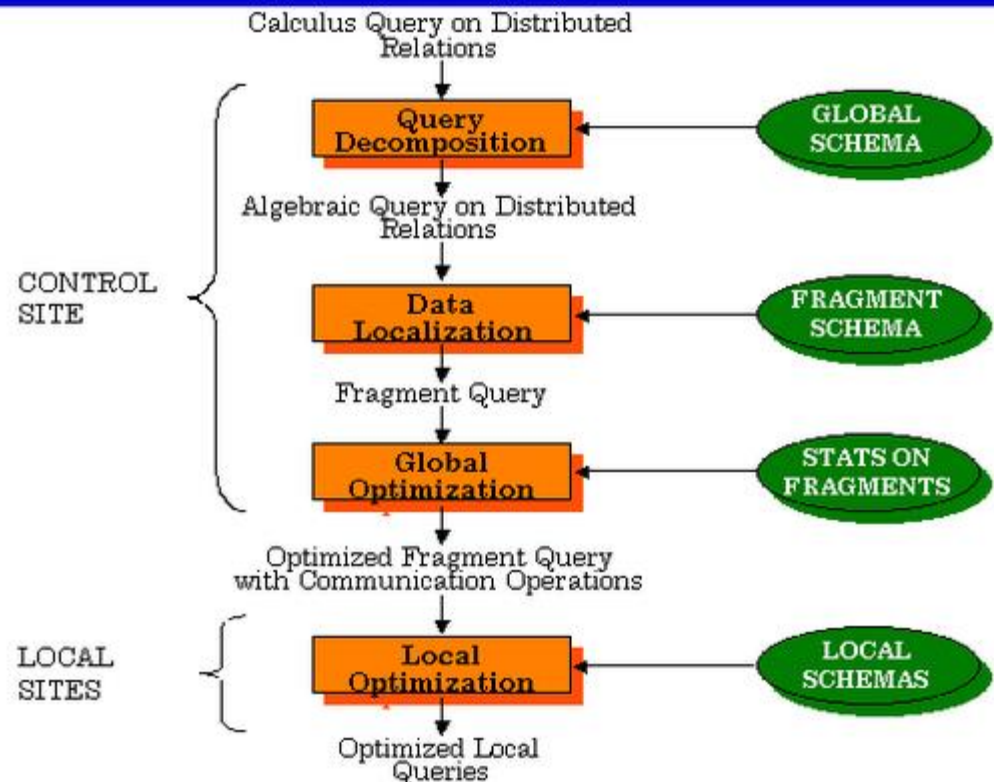
Query Processing in Distributed Databases

1. Query Mapping. The input query on distributed data is specified formally using a query language. It is then translated into an algebraic query on global relations. It is first normalized, analyzed for semantic errors, simplified, and finally restructured into an algebraic query.

2. Localization.

3. Global Query Optimization

4. Local Query Optimization



Distributed query processing:example

Site 1:

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

10,000 records

each record is 100 bytes long

Ssn field is 9 bytes long

Fname field is 15 bytes long

Dno field is 4 bytes long

Lname field is 15 bytes long

Site 2:

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------

100 records

each record is 35 bytes long

Dnumber field is 4 bytes long

Dname field is 10 bytes long

Mgr_ssn field is 9 bytes long

Data Transfer Costs of Distributed Query Processing:

- Cost of transferring data (files and results) over the network.
 - This cost is usually high so some optimization is necessary.
 - Example relations: Employee at site 1 and Department at Site 2
 - Employee at site 1. 10,000 rows. Row size = 100 bytes. Table size = 10^6 bytes.
 - Department at Site 2. 100 rows. Row size = 35 bytes. Table size = 3,500 bytes.

Fname	Minit	Lname	<u>SSN</u>	Bdate	Address	Sex	Salary	Superssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	----------	-----

Dname	Dnumber	Mgrssn	Mgrstartdate
-------	---------	--------	--------------

- Q: For each employee, retrieve employee name and department name Where the employee works.
- Q: $\Pi_{Fname, Lname, Dname} (Employee \bowtie_{Dno = Dnumber} Department)$

Query Processing in Distributed Databases

■ Result

- The result of this query will have 10,000 tuples, assuming that every employee is related to a department.
- Suppose each result tuple is 40 bytes long.
- The query is submitted at site 3 and the result is sent to this site.
- Problem: Employee and Department relations are not present at site 3.

Query Processing in Distributed Databases

- Strategies:
 1. Transfer Employee and Department to site 3.
 - Total transfer bytes = $1,000,000 + 3500 = 1,003,500$ bytes.
 2. Transfer Employee to site 2, execute join at site 2 and send the result to site 3.
 - Total transfer size = $400,000 + 1,000,000 = 1,400,000$ bytes.
 - Query result size = $40 * 10,000 = 400,000$ bytes.
 3. Transfer Department relation to site 1, execute the join at site 1, and send the result to site 3.
 - Total bytes transferred = $400,000 + 3500 = 403,500$ bytes.
- Optimization criteria: minimizing data transfer.

Query Processing in Distributed Databases

- Strategies:

1. Transfer Employee and Department to site 3.

- Total transfer bytes = $1,000,000 + 3500 = 1,003,500$ bytes.

2. Transfer Employee to site 2, execute join at site 2 and send the result to site 3.

- Query result size = $40 * 10,000 = 400,000$ bytes. Total transfer size = $400,000 + 1,000,000 = 1,400,000$ bytes.

3. Transfer Department relation to site 1, execute the join at site 1, and send the result to site 3.

- Total bytes transferred = $400,000 + 3500 = 403,500$ bytes.

- Optimization criteria: minimizing data transfer.

- Preferred approach: strategy 3.

Query Processing in Distributed Databases

- Consider the query
 - Q': For each department, retrieve the department name and the name of the department manager
- Relational Algebra expression:
 - $\Pi_{Fname, Lname, Dname} (Employee \bowtie_{Mgrssn = SSN} Department)$

Query Processing in Distributed Databases

- The result of this query will have 100 tuples, assuming that every department has a manager, the execution strategies are:
 1. Transfer Employee and Department to the result site and perform the join at site 3.
 - Total bytes transferred = $1,000,000 + 3500 = 1,003,500$ bytes.
 2. Transfer Employee to site 2, execute join at site 2 and send the result to site 3. Query result size = $40 * 100 = 4000$ bytes.
 - Total transfer size = $4000 + 1,000,000 = 1,004,000$ bytes.
 3. Transfer Department relation to site 1, execute join at site 1 and send the result to site 3.
 - Total transfer size = $4000 + 3500 = 7500$ bytes.

Query Processing in Distributed Databases

- The result of this query will have 100 tuples, assuming that every department has a manager, the execution strategies are:
 1. Transfer Employee and Department to the result site and perform the join at site 3.
 - Total bytes transferred = $1,000,000 + 3500 = 1,003,500$ bytes.
 2. Transfer Employee to site 2, execute join at site 2 and send the result to site 3. Query result size = $40 * 100 = 4000$ bytes.
 - Total transfer size = $4000 + 1,000,000 = 1,004,000$ bytes.
 3. Transfer Department relation to site 1, execute join at site 1 and send the result to site 3.
 - Total transfer size = $4000 + 3500 = 7500$ bytes.
- Preferred strategy: Choose strategy 3.

Query Processing in Distributed Databases

- Now suppose the result site is 2. Possible strategies :
 1. Transfer Employee relation to site 2, execute the query and present the result to the user at site 2.
 - Total transfer size = 1,000,000 bytes for both queries Q and Q'.
 2. Transfer Department relation to site 1, execute join at site 1 and send the result back to site 2.
 - Total transfer size for Q = $400,000 + 3500 = 403,500$ bytesand for
Q' = $4000 + 3500 = 7500$ bytes.

Query Processing in Distributed Databases

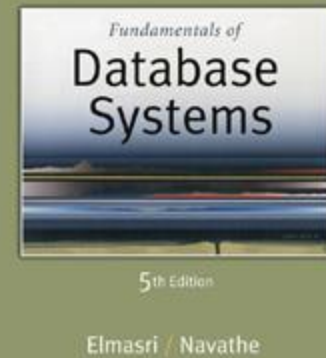
- **Semijoin:**
 - Objective is to reduce the number of tuples and attributes in a relation before transferring it to another site.
- **Example execution of Q or Q':**
 1. Project the join attributes of Department at site 2, and transfer them to site 1. For Q, $4 * 100 = 400$ bytes are transferred and for Q', $9 * 100 = 900$ bytes are transferred.
 2. Join the transferred file with the Employee relation at site 1, and transfer the required attributes from the resulting file to site 2. For Q, $34 * 10,000 = 340,000$ bytes are transferred and for Q', $39 * 100 = 3900$ bytes are transferred.
 3. Execute the query by joining the transferred file with Department and present the result to the user at site 2.

Query Processing in Distributed Databases

A semijoin operation

$R \text{ semijoin}_{A=B} S$, where A and B are domain-compatible attributes of R and S , respectively, produces the same result as the relational algebra expression $\pi_R(R \bowtie_{A=B} S)$. In a distributed environment where R and S reside at different sites, the semijoin is typically implemented by first transferring $F = \pi_B(S)$ to the site where R resides and then joining F with R , thus leading to the strategy discussed here. Notice that the semijoin operation is not commutative;

Query and Update Decomposition



Site 1

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------

DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

PROJECT

Pname	<u>Pnumber</u>	Plocation	Dnum
-------	----------------	-----------	------

WORKS_ON

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
-------------	-----------------------	-----	-------	--------------

Figure 3.5

Schema diagram for the COMPANY relational database schema.

Site 2

(a) EMPD_5

Fname	Minit	Lname	<u>Ssn</u>	Salary	Super_ssn	Dno
John	B	Smith	123456789	30000	333445555	5
Franklin	T	Wong	333445555	40000	888665555	5
Ramesh	K	Narayan	666884444	38000	333445555	5
Joyce	A	English	453453453	25000	333445555	5

DEP_5

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22

DEP_5_LOCS

<u>Dnumber</u>	<u>Location</u>
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON_5

<u>Essn</u>	<u>Pno</u>	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0

PROJS_5

Pname	<u>Pnumber</u>	Plocation	Dnum
Product X	1	Bellaire	5
Product Y	2	Sugarland	5
Product Z	3	Houston	5

Data at site 2

Figure 25.8

Allocation of fragments to sites. (a) Relation fragments at site 2 corresponding to department 5. (b) Relation fragments at site 3 corresponding to department 4.

Site 3

(b) EMPD_4

Fname	Minit	Lname	<u>Ssn</u>	Salary	Super_ssn	Dno
Alicia	J	Zelaya	999887777	25000	987654321	4
Jennifer	S	Wallace	987654321	43000	888665555	4
Ahmad	V	Jabbar	987987987	25000	987654321	4

DEP_4

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
Administration	4	987654321	1995-01-01

DEP_4_LOCS

<u>Dnumber</u>	<u>Location</u>
4	Stafford

WORKS_ON_4

<u>Essn</u>	<u>Pno</u>	Hours
333445555	10	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0

PROJS_4

Pname	<u>Pnumber</u>	Plocation	Dnum
Computerization	10	Stafford	4
New_benefits	30	Stafford	4

Data at site 3

Query and Update Decomposition

EMPD5

attribute list: Fname, Minit, Lname, Ssn, Salary, Super_ssn, Dno

guard condition: Dno=5

DEP5

attribute list: * (all attributes Dname, Dnumber, Mgr_ssn, Mgr_start_date)

guard condition: Dnumber=5

DEP5_LOCS

attribute list: * (all attributes Dnumber, Location)

guard condition: Dnumber=5

PROJS5

attribute list: * (all attributes Pname, Pnumber, Plocation, Dnum)

guard condition: Dnum=5

WORKS_ON5

attribute list: * (all attributes Essn, Pno, Hours)

guard condition: Essn IN (π_{Ssn} (EMPD5)) OR Pno IN ($\pi_{Pnumber}$ (PROJS5))

Query decomposition

Q: SELECT Fname, Lname, Hours
FROM EMPLOYEE, PROJECT, WORKS_ON
WHERE Dnum=5 **AND** Pnumber=Pno **AND** Essn=Ssn;

Suppose that the query is submitted at site 2, which is where the query result will be needed. The DDBMS can determine from the guard condition on PROJS5 and WORKS_ON5 that all tuples satisfying the conditions (Dnum = 5 AND Pnumber = Pno) reside at site 2. Hence, it may decompose the query into the following relational algebra subqueries:

$$T_1 \leftarrow \pi_{\text{Essn}}(\text{PROJS5} \bowtie_{\text{Pnumber}=\text{Pno}} \text{WORKS_ON5})$$

$$T_2 \leftarrow \pi_{\text{Essn}, \text{Fname}, \text{Lname}}(T_1 \bowtie_{\text{Essn}=\text{Ssn}} \text{EMPLOYEE})$$

$$\text{RESULT} \leftarrow \pi_{\text{Fname}, \text{Lname}, \text{Hours}}(T_2 * \text{WORKS_ON5})$$

This decomposition can be used to execute the query by using a semijoin strategy.

The DDBMS knows from the guard conditions that PROJ5 contains exactly those tuples satisfying ($Dnum = 5$) and that WORKS_ON5 contains all tuples to be joined with PROJ5; hence, subquery T_1 can be executed at site 2, and the projected column Essn can be sent to site 1. Subquery T_2 can then be executed at site 1, and the result can be sent back to site 2, where the final query result is calculated and displayed to the user. An alternative strategy would be to send the query Q itself to site 1, which includes all the database tuples, where it would be executed locally and from which the result would be sent back to site 2. The query optimizer would estimate the costs of both strategies and would choose the one with the lower cost estimate.

Decompose update query

(b) EMPD4

attribute list: Fname, Minit, Lname, Ssn, Salary, Super_ssn, Dno
guard condition: Dno=4

DEP4

attribute list: * (all attributes Dname, Dnumber, Mgr_ssn, Mgr_start_date)
guard condition: Dnumber=4

DEP4_LOCS

attribute list: * (all attributes Dnumber, Location)
guard condition: Dnumber=4

PROJS4

attribute list: * (all attributes Pname, Pnumber, Plocation, Dnum)
guard condition: Dnum=4

WORKS_ON4

attribute list: * (all attributes Essn, Pno, Hours)
guard condition: Essn IN (π_{Ssn} (EMPD4))
OR Pno IN ($\pi_{Pnumber}$ (PROJS4))

Figure 25.11

Guard conditions and attributes lists for fragments.

(a) Site 2 fragments. (b) Site 3 fragments.

Decompose update query

Figure 25.11. When the DDBMS decomposes an update request, it can determine which fragments must be updated by examining their guard conditions. For example, a user request to insert a new EMPLOYEE tuple $\langle \text{'Alex'}, \text{'B'}, \text{'Coleman'}, \text{'345671239'}, \text{'22-APR-64'}, \text{'3306 Sandstone, Houston, TX'}, \text{M}, 33000, \text{'987654321'}, 4 \rangle$ would be decomposed by the DDBMS into two insert requests: the first inserts the preceding tuple in the EMPLOYEE fragment at site 1, and the second inserts the projected tuple $\langle \text{'Alex'}, \text{'B'}, \text{'Coleman'}, \text{'345671239'}, 33000, \text{'987654321'}, 4 \rangle$ in the EMPD4 fragment at site 3.