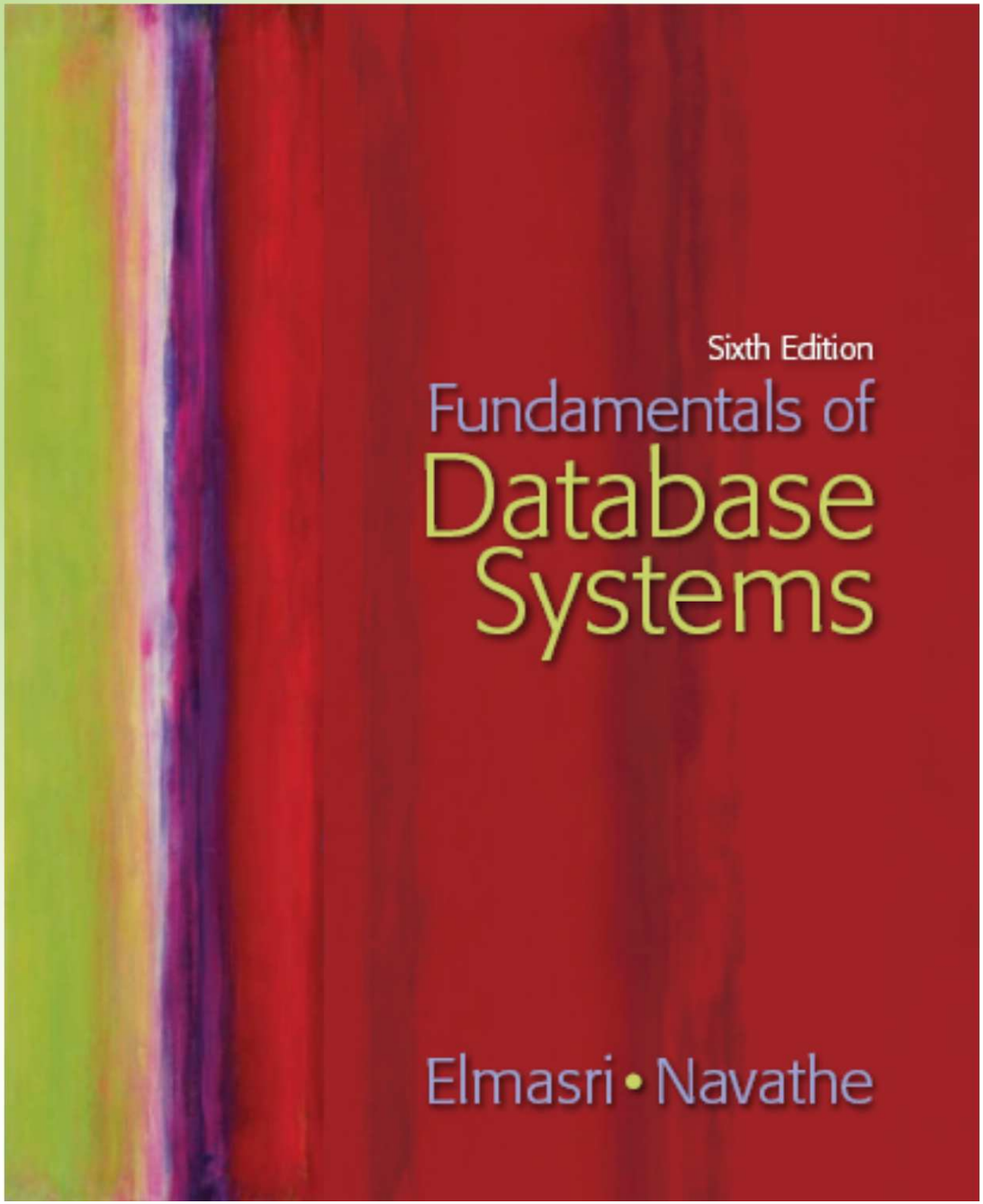
Chapter 25

Distributed Databases and Client-Server Architectures



Addison-Wesley
is an imprint of
PEARSON

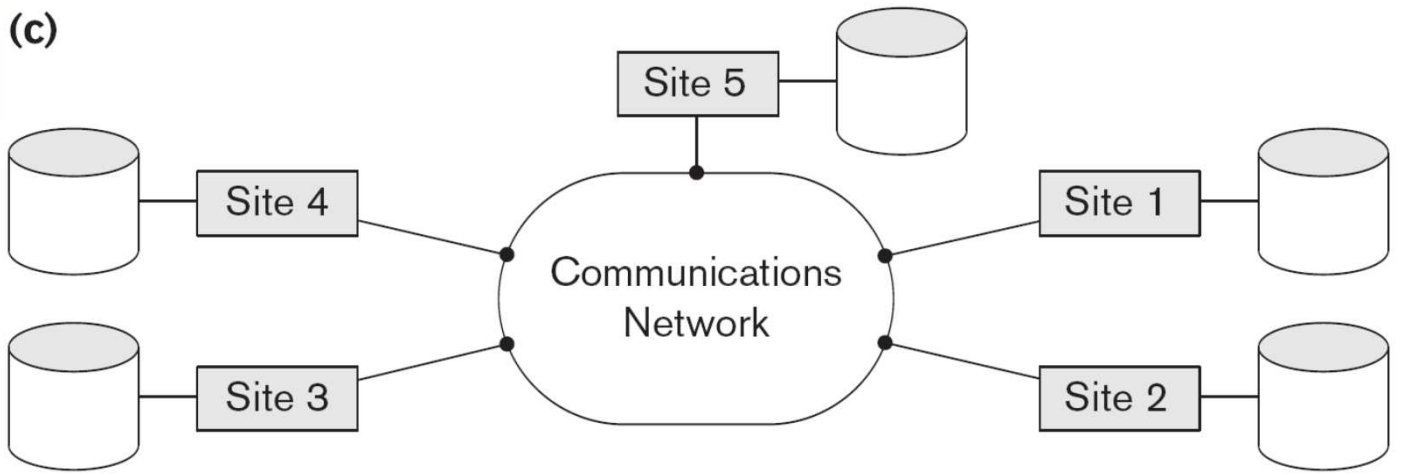
Distributed Database Concepts

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

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).

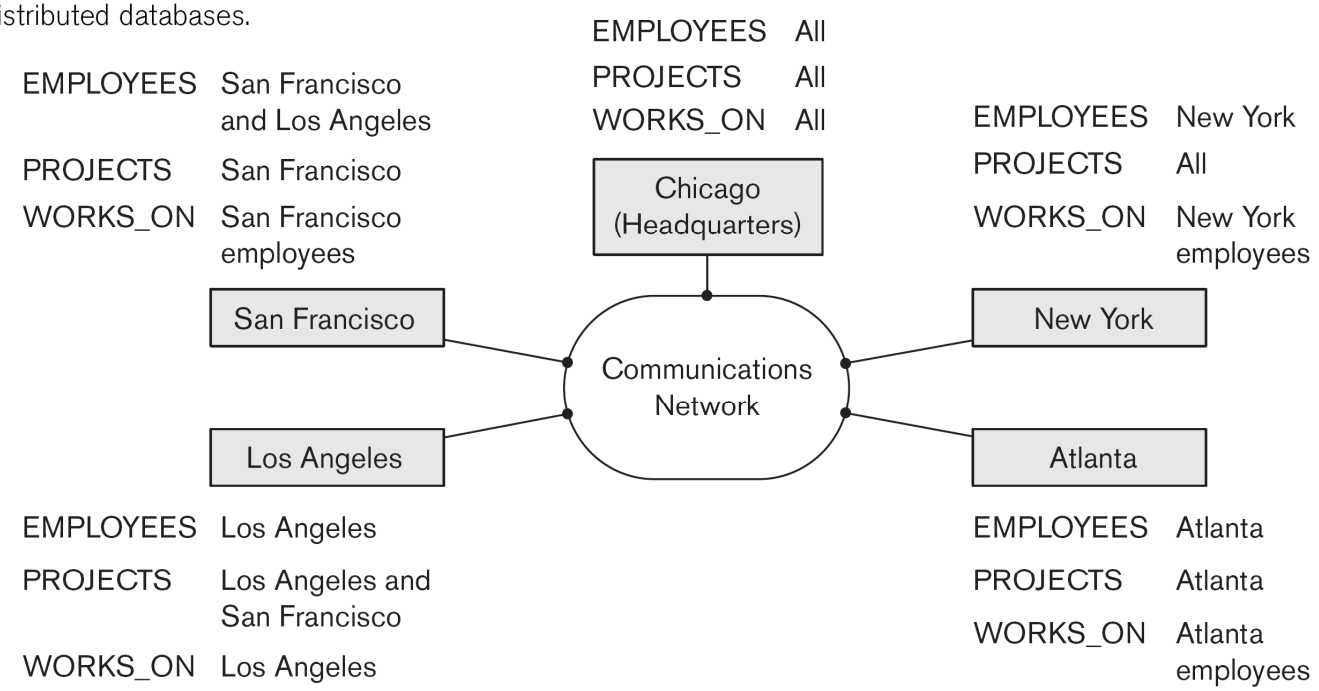
Figure 25.3
Some different database system architectures. (a) Shared nothing architecture. (b) A networked architecture with a centralized database at one of the sites. (c) A truly distributed database architecture.



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 25.1
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.
      - Also, 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 as shown in the previous diagram (also serves as backup of data).
    - 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 availability and reliability**:
    - Availability is the probability that a system, at a point in time, will be operational and able to deliver the requested services.
    - Reliability is the probability of failure-free operation over a specified time, in a given environment, for a specific purpose.
    - 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 changing 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**

Data Fragmentation, Replication and Allocation

- **Horizontal fragmentation**
  - It is a horizontal subset of a relation which contain those tuples which satisfy selection conditions.
  - Consider the Employee relation with selection condition (DNO = 5). All tuples that satisfy this condition will create a subset which will be a horizontal fragment of Employee relation.
  - A selection condition may be composed of several conditions connected by AND or OR.
  - Derived horizontal fragmentation: It is the partitioning of a primary relation to other secondary relations which are related with foreign keys.

Data Fragmentation, Replication and Allocation

- **Horizontal fragmentation: an example**

(a)

EMPD_5

Fname	Minit	Lname	Ssn	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	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22

DEP_5_LOCS

Dnumber	Location
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON_5

Essn	Pno	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	Pnumber	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.

(b)

EMPD_4

Fname	Minit	Lname	Ssn	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	Dnumber	Mgr_ssn	Mgr_start_date
Administration	4	987654321	1995-01-01

DEP_4_LOCS

Dnumber	Location
4	Stafford

WORKS_ON_4

Essn	Pno	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	Pnumber	Plocation	Dnum
Computerization	10	Stafford	4
New_benefits	30	Stafford	4

Data at site 3

Data Fragmentation, Replication and Allocation

- **Vertical fragmentation**
  - It is a subset of a relation which is created by a subset of columns. Thus a vertical fragment of a relation will contain values of selected columns. There is no selection condition used in vertical fragmentation.
  - Consider the Employee relation. A vertical fragment can be created by keeping the values of Name, Bdate, Sex, and Address.
  - Because there is no condition for creating a vertical fragment, each fragment must include the primary key attribute of the parent relation Employee. In this way all vertical fragments of a relation are connected.

Data Fragmentation, Replication and Allocation

- **Vertical fragmentation: an example**
  - **Emp_Generals**

Fname	Minit	Lname	Ssn	Bdate	Address	Sex
John	B	Smith	123456789	May-11-1987	Add1	M
Joyce	A	English	453453453	Apr-30-1982	Add2	F
-	-	-	-	-	-	-

  - **Emp_Business**

Ssn	Salary	Super_ssn	Dno
123456789	3000	987654321	5
453453453	3500	987654321	5
-	-	-	-

Data Fragmentation, Replication and Allocation

- **Representation**
  - **Horizontal fragmentation**
    - Each horizontal fragment on a relation can be specified by a σ_{Ci} (R) operation in the relational algebra.
    - Complete horizontal fragmentation
      - A set of horizontal fragments whose conditions C1, C2, …, Cn include all the tuples in R- that is, every tuple in R satisfies (C1 OR C2 OR … OR Cn).
      - Disjoint complete horizontal fragmentation: No tuple in R satisfies (Ci AND Cj) where $i \neq j$.
    - To reconstruct R from horizontal fragments a UNION is applied.

Data Fragmentation, Replication and Allocation

- **Representation**
  - **Vertical fragmentation**
    - A vertical fragment on a relation can be specified by a $\Pi_{Li}(R)$ operation in the relational algebra.
    - Complete vertical fragmentation
      - A set of vertical fragments whose projection lists L1, L2, …, Ln include all the attributes in R but share only the primary key of R. In this case the projection lists satisfy the following two conditions:
      - $L1 \cup L2 \cup ... \cup Ln = ATTRS(R)$
      - $Li \cap Lj = PK(R)$ for any $i \neq j$, where ATTRS (R) is the set of attributes of R and PK(R) is the primary key of R.
    - To reconstruct R from complete vertical fragments an OUTER UNION is applied.

Data Fragmentation, Replication and Allocation

- **Representation**
  - **Mixed (Hybrid) fragmentation**
    - A combination of Vertical fragmentation and Horizontal fragmentation.
    - This is achieved by SELECT-PROJECT operations which is represented by $\Pi_{Li}(\sigma_{Ci}(R))$.
    - If C = True (Select all tuples) and L ≠ ATTRS(R), we get a vertical fragment, If C ≠ True and L = ATTRS(R), we get an horizontal fragment and if C ≠ True and L ≠ ATTRS(R), we get a mixed fragment.
    - If C = True and L = ATTRS(R), then R can itself be considered a fragment.

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 OUTER 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.

Data Fragmentation, Replication and Allocation

- **Data Replication**
  - Database is replicated to all sites.
  - In full replication the entire database is replicated and in partial replication some selected part is replicated to some of the sites.

Query Processing in Distributed Databases

- Issues
  - 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.

Fname	Minit	Lname	SSN	Bdate	Address	Sex	Salary	Superssn	Dno

      - Department at Site 2. 100 rows. Row size = 35 bytes. Table size = 3,500 bytes.

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.
     - Query result size = 40 * 10,000 = 400,000 bytes. Total transfer size = 1,000,000 + 400,000 = 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 = 3500 + 400,000= 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 $\bowtie_{SSN = Mgrssn}$ 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 = 1,000,000 + 4000 = 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 = 3500 + 4000 = 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 = 3500 + 400,000 = 403,500 bytes and for Q' = 3500 + 4000 = 7500 bytes.

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

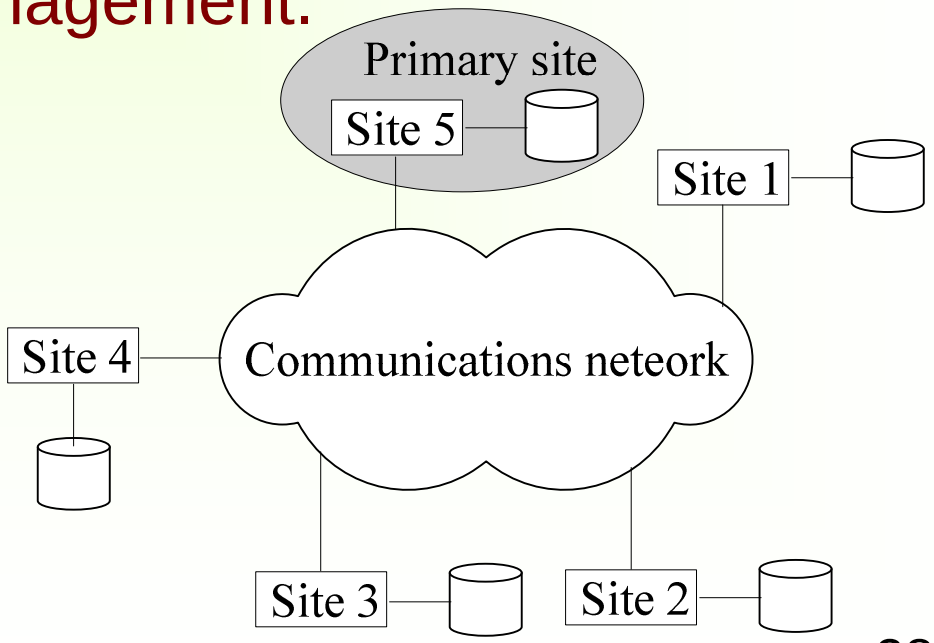
- Details
  - Dealing with multiple copies of data items
    - The concurrency control must maintain global consistency. Likewise the recovery mechanism must recover all copies and maintain consistency after recovery.
  - Failure of individual sites
    - Database availability must not be affected due 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 and Recovery

- Distributed Concurrency control based on a distinguished distributed copy of a data item
  - 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:
    - It is 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 (the copies).
  - 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

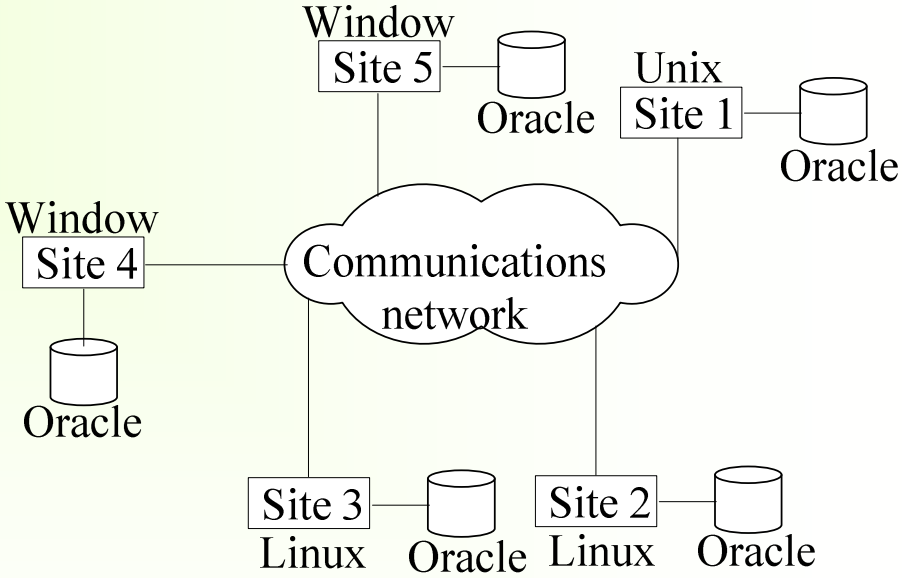
- Primary Copy Technique:
  - In this approach, instead of a site, a data item is designated as primary copy. To lock a data item just the primary copy of the data item is locked. The primary copies are stored in different sites.
- Advantages:
  - Since primary copies are distributed at various sites, 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:
  - Suspends all active transactions, designates the backup site as the primary site and identifies a new back up site. Primary site receives all transaction management information to resume processing.
- Primary and backup sites fail or no backup site:
  - Use election process to select a new coordinator site.

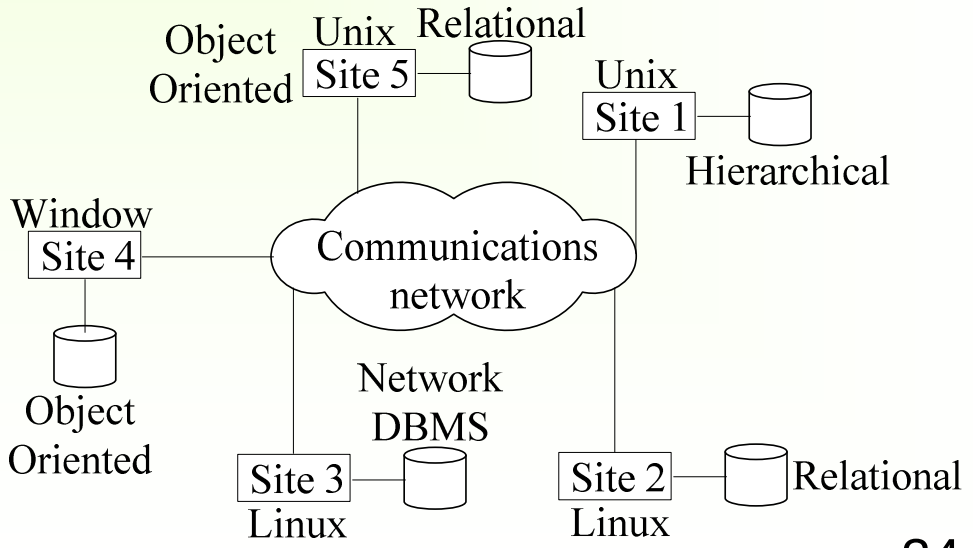
Types of Distributed Database Systems

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



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.
    - This implies that the degree of local autonomy is minimum. Each site must adhere to a centralized access policy. There may be a global schema.
  - Multidatabase: There is no one conceptual global schema. For data access a schema is constructed dynamically as needed by the application software.



Types of Distributed Database Systems

- Federated Database Management Systems Issues
  - Differences in data models:
    - Relational, Objected oriented, hierarchical, network, etc.
  - Differences in constraints:
    - Each site may have their own data accessing and processing constraints.
  - Differences in query language:
    - Some site may use SQL, some may use SQL-89, some may use SQL-92, and so on.