Databases 2022

Darko Bozhinoski,
Ph.D. in Computer Science

AGENDA

- ❖ Distributed Database Concepts
- ❖ Data Fragmentation, Replication and Allocation
- ❖ Query Processing
- ❖ Concurrency Control and Recovery
- ❖ Types of Distributed Database Systems
- ❖ 3-Tier Client-Server Architecture
- ❖ CAP Theorem

Distributed Database Concepts

- A transaction can be executed by multiple networked computers in a unified manner.
- A **distributed database (DDB)** processes units of execution (transactions) in a distributed manner.
- 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.

What constitutes a Distributed DataBase System (DDBS)?

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

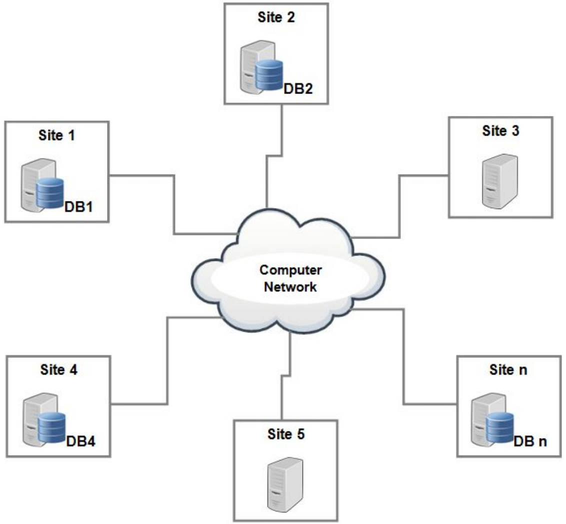
What constitutes a Distributed DataBase System (DDBS)?

- **Connection of database nodes over a computer network.** There are multiple computers, called **sites** or **nodes**. These sites must be connected by an underlying **network** to transmit data and commands a
- **L** ential th y related.
- **P** **es.** It is n dware, a

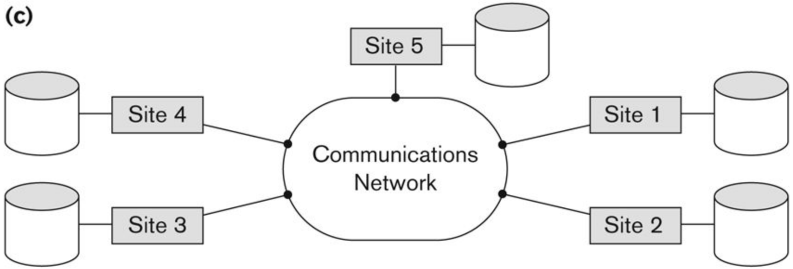
The type and topology of the network used may have a significant impact on the performance and hence on the strategies for <u>distributed query processing and distributed database design</u>.

Distributed Database System

- 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).

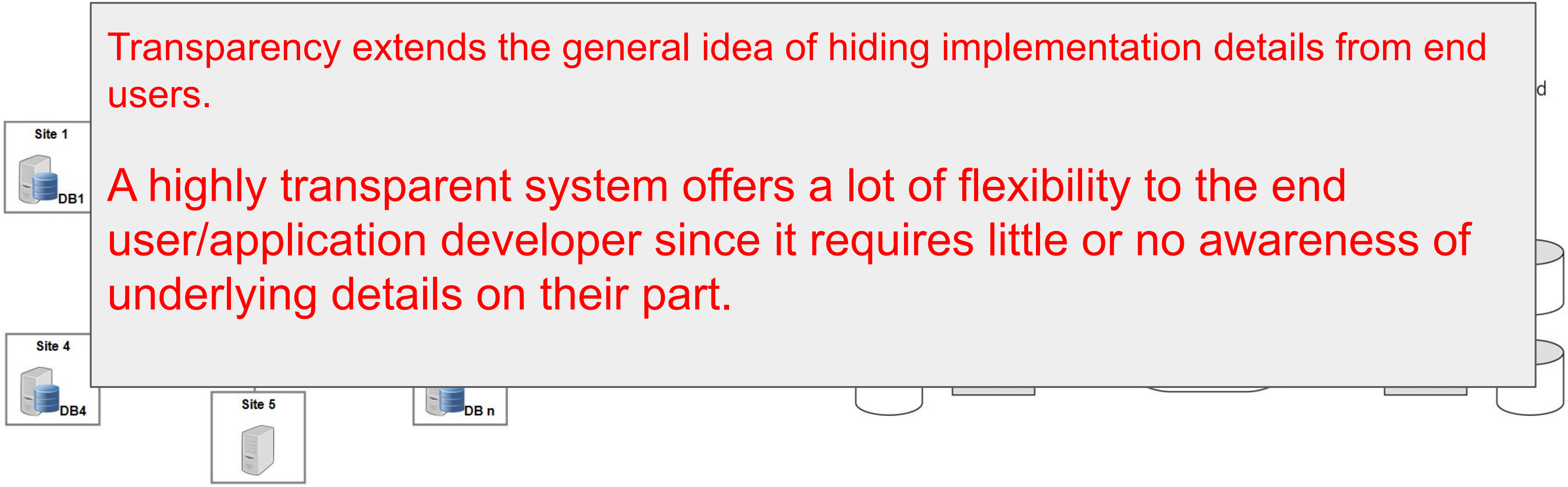


Some different database system architectures. (a) Shared nothing architecture. (b) A networked architecture with a centralized database at one of the sites. (c) A truly distributed database architecture.



Distributed Database System

- 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).



Transparency extends the general idea of hiding implementation details from end users.

A highly transparent system offers a lot of flexibility to the end user/application developer since it requires little or no awareness of underlying details on their part.

Distribution/Network transparency

- Users do not have to worry about operational details of the network.
  - Location transparency, which refers to freedom of issuing command from any location without affecting its working.
  - Naming transparency, which allows access to any names object (files, relations, etc.) from any location.

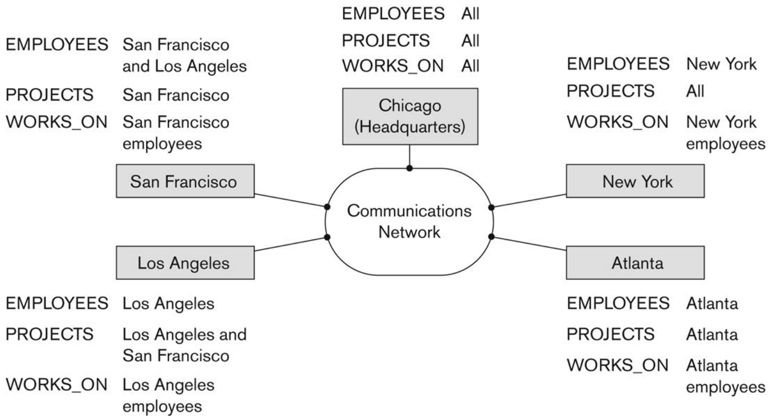
Replication and Fragmentation Transparency

- **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).

Replication and Fragmentation Example

- 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.



Chicago (Headquarters)

EMPLOYEES	All
PROJECTS	All
WORKS_ON	All

San Francisco

EMPLOYEES	San Francisco and Los Angeles
PROJECTS	San Francisco
WORKS_ON	San Francisco employees

New York

EMPLOYEES	New York
PROJECTS	All
WORKS_ON	New York employees

Los Angeles

EMPLOYEES	Los Angeles
PROJECTS	Los Angeles and San Francisco
WORKS_ON	Los Angeles employees

Atlanta

EMPLOYEES	Atlanta
PROJECTS	Atlanta
WORKS_ON	Atlanta employees

Communications Network

Reliability and Availability

- **Increased reliability and availability**:
  - Reliability refers to the probability that the system is running at a certain moment of 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.

Reliability and Availability

- **Increased reliability and availability**:
  - A **failure** is a deviation of a system’s behavior from that which is specified in order to ensure correct execution of operations.
  - **Errors** constitute that subset of system states that causes the failure.
  - **Fault** is the cause of an error.
- Different strategies to construct a system that is reliable
  - <u>Stress fault tolerance:</u> designs mechanisms that can detect and remove faults before they can result in a system failure
  - <u>Exhaustive design process</u> with <u>extensive quality control</u>

Performance and Scalability

- **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.
- **Scalability**:
  - the extent to which the system can expand its capacity while continuing to operate without interruption
  - Allows new nodes (computers) to be added anytime without chaining the entire configuration.

Data Fragmentation, Replication and Allocation

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**
- The information concerning data fragmentation, allocation, and replication is stored in a global directory that is accessed by the DDBS applications as needed.

Horizontal fragmentation

- It is a horizontal subset of a relation which contain those of tuples which satisfy selection conditions.
- Consider the Employee relation with selection condition (DNO = 5). All tuples 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.

Horizontal fragmentation (2)

- **Representation**
  - Each horizontal fragment on a relation can be specified by a $\boldsymbol{\sigma}_{Ci}$ (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 ≠ j.
  - To reconstruct R from horizontal fragments a UNION is applied.

Vertical fragmentation

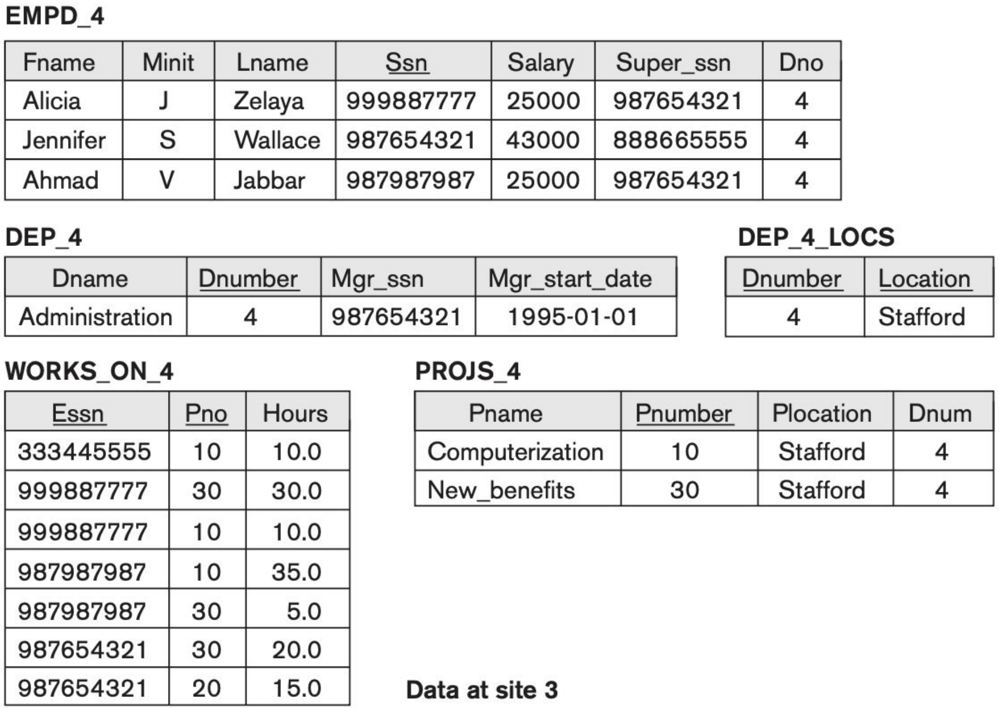
- It is a subset of a relation which is created by a subset of columns. Thus a vertical fragment of a relation will contain values of selected columns. There is no selection condition used in vertical fragmentation.
- Consider the Employee relation. A vertical fragment of this relation can be created by keeping the values of Name, Bdate, Address, and Sex in a first fragment and Ssn, Salary, Super_ssn, and Dno work-related information in a second fragment.
- 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.

Vertical fragmentation (2)

- **Representation**
  - 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 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 a OUTER UNION is applied.

Mixed (Hybrid) fragmentation

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



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

Allocation of fragments to sites.

Fragmentation and Allocation Schema

- **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.

Data 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.
  - Data replication is achieved through a replication schema.
- **Data Distribution (Data Allocation)**
  - This is relevant only in the case of partial replication or partition.
  - The selected portion of the database is distributed to the database sites.

Data Replication and Allocation

- Replication is useful in improving the availability of data: system can continue to operate as long as at least one site is up.
- **Fully replicated distributed database**: advantages and disadvantages. **What is the trade-off?**

Data Replication and Allocation

- Replication is useful in improving the availability of data: system can continue to operate as long as at least one site is up.
- **Fully replicated distributed database**: advantages and disadvantages. **What is the trade-off?**

Advantageous: Improves performance of retrieval (read performance) for global queries because the results of such queries can be obtained locally from any site;

Disadvantageous: it can slow down update operations (write performance) drastically, since a single logical update must be performed on every copy of the database to keep the copies consistent.

Data Replication and Allocation

- **Fully replicated distributed database**:
- **No replication**—that is, each fragment is stored at exactly one site. All fragments must be disjoint, except for the repetition of primary keys among vertical (or mixed) fragments. This is also called non redundant allocation.

Between these two extremes, we have a wide spectrum of partial replication of the data.

The choice of sites and the degree of replication depend on the performance and availability goals of the system and on the types and frequencies of transactions submitted at each site!

Query Processing in Distributed Databases

Query Processing in Distributed Databases

A distributed database query is processed in stages as follows:

1. **Query Mapping.** The query on distributed data is translated into an algebraic query on global relations. This translation is done by referring to the global conceptual schema and does not take into account the actual distribution and replication of data.

    - Similar to the one performed in a centralized DBMS.

2. **Localization:** maps the distributed query on the global schema to queries on individual fragments using data distribution and replication information.

Query Processing in Distributed Databases (2)

3. Global Query Optimization: selecting a strategy from a list of candidates

- Generates a distributed execution plan so that least amount of data transfer occurs across the sites. The plan states the location of the fragments, order in which query steps needs to be executed and the processes involved in transferring intermediate results.
- Time is the preferred unit for measuring cost. The total cost is a weighted combination of costs such as CPU cost, I/O costs, and communication costs.
- Communication costs over the network are the most significant.

4. Local Query Optimization: common to all sites in the DDB.

- Similar to those used in centralized systems.

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.

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 $_{Dno = Dnumber}$ Department)

Data Transfer Costs of Distributed Query Processing (2)

- **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 Strategies

- **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.

Query Processing Strategies

- **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.**)

Another Example

- 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)

Result

- The result of this query will have 100 tuples, assuming that every department has a manager.

- Execution strategies:
  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.

Result (2)

- The result of this query will have 100 tuples, assuming that every department has a manager.

- Execution strategies:
  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.**

Modifications in the Scenario.

- Now suppose the result site is 2.

What are the possible strategies?

Modifications in the Scenario.

- 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 bytes and for Q’ = 4000 + 3500 = 7500 bytes.

Concurrency Control and Recovery

Concurrency Control and Recovery (2)

- Distributed Databases encounter a number of concurrency control and recovery problems which are not present in centralized databases.
  - 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 (3)

- 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.

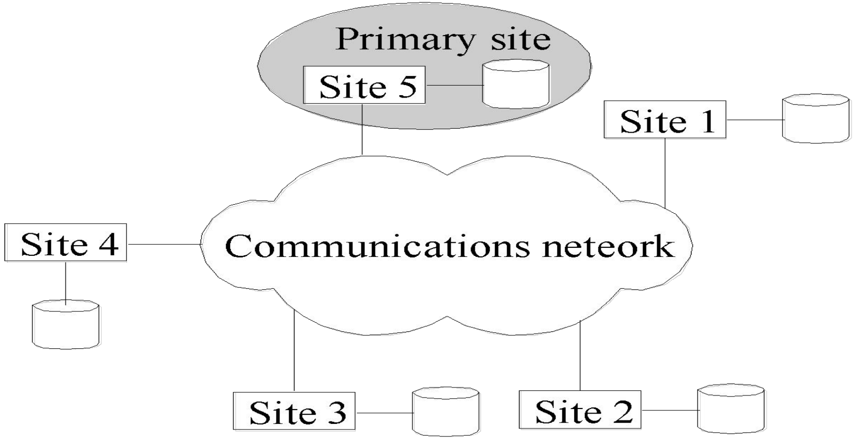
Concurrency Control and Recovery (4)

Techniques to deal with concurrency control and recovery:

- Distinguished Copy of a Data Item;
  - extending centralized locking;
  - idea: designate a particular copy of each data item as a distinguished copy.
- Concurrency control based on voting

Distinguished 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.
- ALL locks are kept at that site, and all requests for locking or unlocking are sent there.



Primary site technique

- 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.

Primary site technique

- 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.
- 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.

Primary Site with Backup Site

- 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.
- It slows down the process of acquiring locks, however, because all lock requests and granting of locks must be recorded at both the primary and the backup sites before a response is sent to the requesting transaction.
- It does not resolve the issue of having overloaded system with requests.

Primary Copy Technique

- In this approach, instead of a site, a data item partition is designated as primary copy. To lock a data item just the primary copy of the data item is locked.
- 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.

Recovery

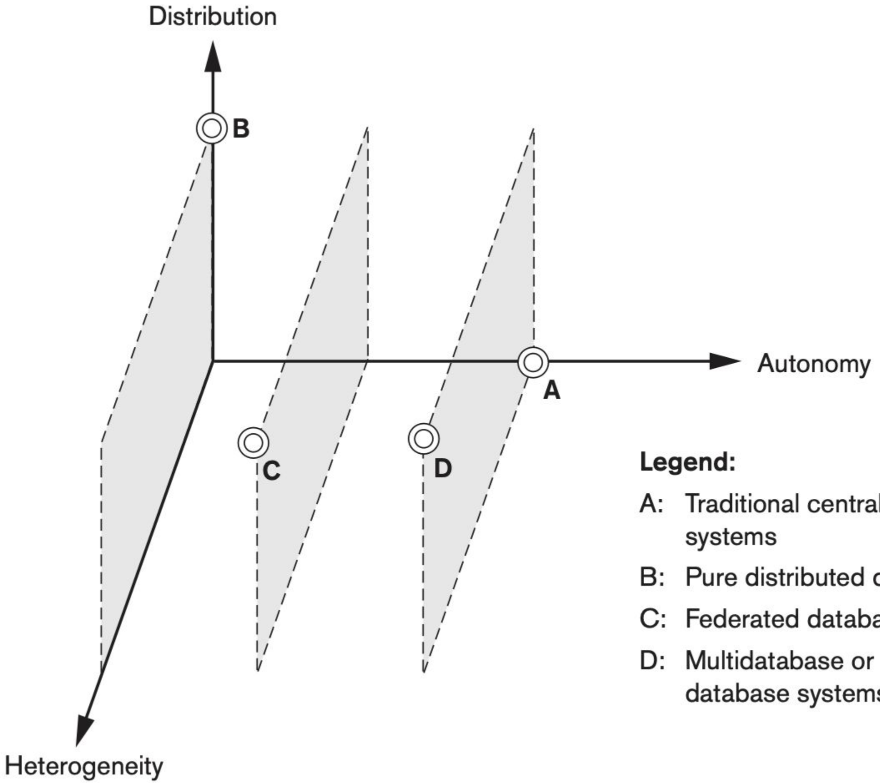
- Recovery from a coordinator failure
  - In all approaches a coordinator site or copy may become unavailable. This will require the selection of a new coordinator.
- Primary site approach with no backup site:
  - Aborts and restarts all active transactions at all sites. Elects a new coordinator and 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.

Concurrency control based on voting

- There is no primary copy of coordinator.
  - Send lock request to sites that have data item.
  - If majority of sites grant lock then the requesting transaction gets the data item.
  - Locking information (grant or denied) is sent to all these sites.
  - To avoid unacceptably long wait, a time-out period is defined. If the requesting transaction does not get any vote information then the transaction is aborted.

Types of Distributed Database Systems

Types of Distributed Database Systems



Legend:

A: Traditional centralized database systems

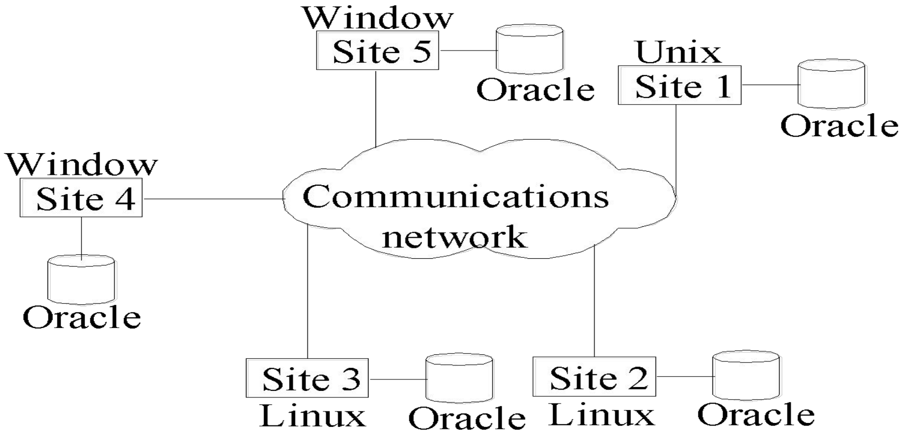
B: Pure distributed database systems

C: Federated database systems

D: Multidatabase or peer-to-peer database systems

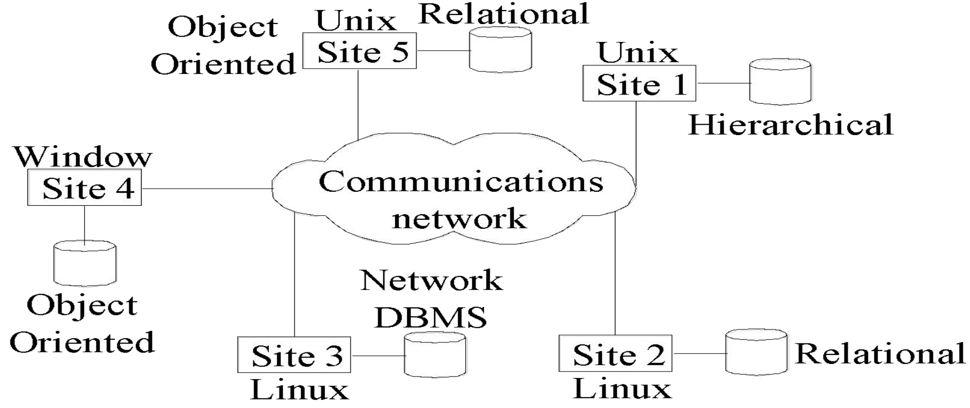
Types of Distributed Database Systems

- **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.



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.

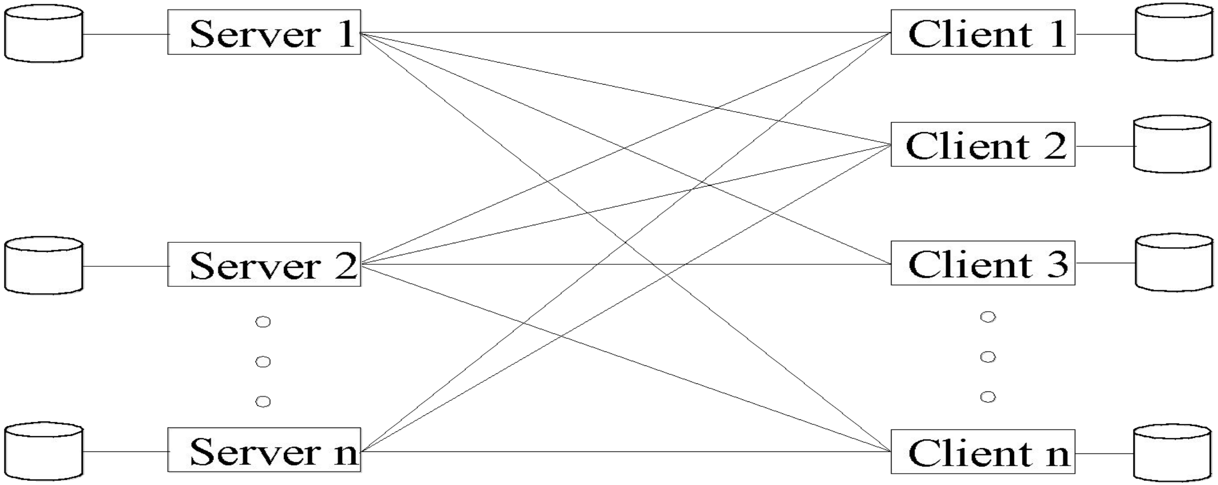


Types of Distributed Database Systems

- 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.

Client-Server Database Architecture

- It consists of clients running client software, a set of servers which provide all database functionalities and a reliable communication infrastructure.



Client-Server Database Architecture (2)

- The processing of a SQL queries goes as follows:
  - Client parses a user query and decomposes it into a number of independent sub-queries. Each subquery is sent to appropriate site for execution.
  - Each server processes its query and sends the result to the client.
  - The client combines the results of subqueries and produces the final result.

Examples

Key Value

- DynamoDB
- Azure Table
- Redis
- Riak

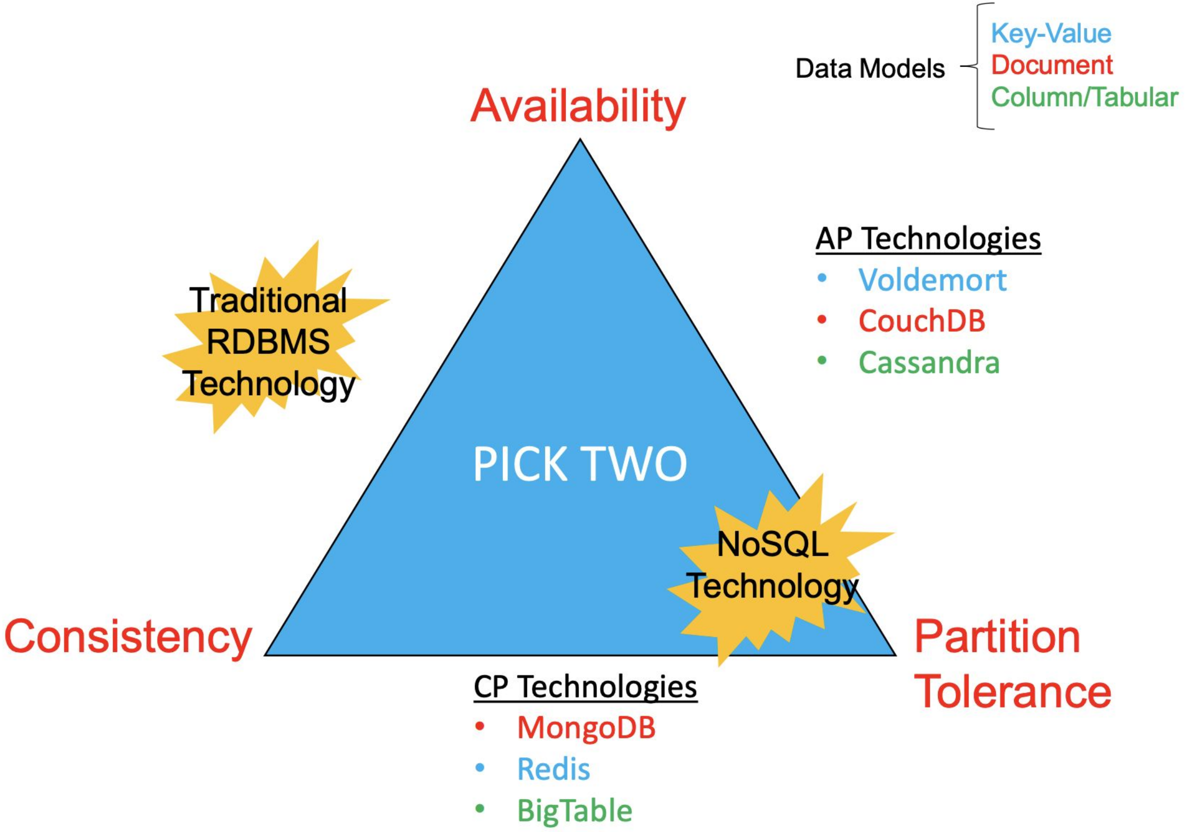
Document Centric

- MongoDB
- CouchDB
- RavenDB

Column

- Hbase
- Cassandra
- Hypertable
- SimpleDB

Consistency - CAP Theorem



Data Models

- Key-Value
- Document
- Column/Tabular

Availability

Traditional RDBMS Technology

PICK TWO

NoSQL Technology

Consistency

Partition Tolerance

AP Technologies

- Voldemort
- CouchDB
- Cassandra

CP Technologies

- MongoDB
- Redis
- BigTable

Consistency - CAP Theorem

- When data becomes distributed you need to worry about a network partition
  - Essentially this means that instances of your data store can’t communicate
- When this happens you need to choose between availability or consistency

Reading Material

- Fundamentals of Database Systems. Ramez Elmasri and Shamkant B. Navathe. Pearson. **Chapter 23.**

Q
&
A