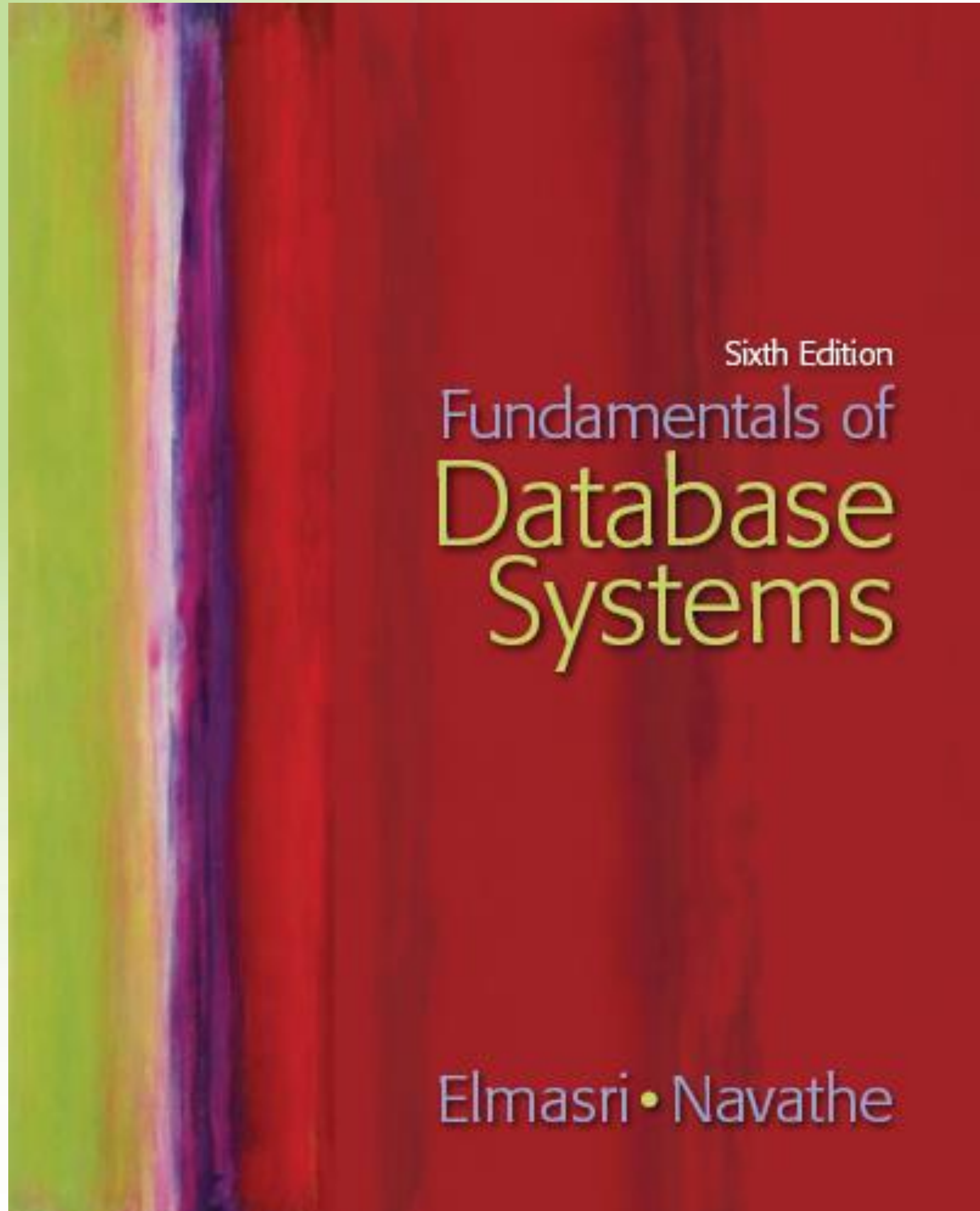
Chapter 25

Distributed Databases and Client-Server Architectures



Addison-Wesley
is an imprint of
PEARSON

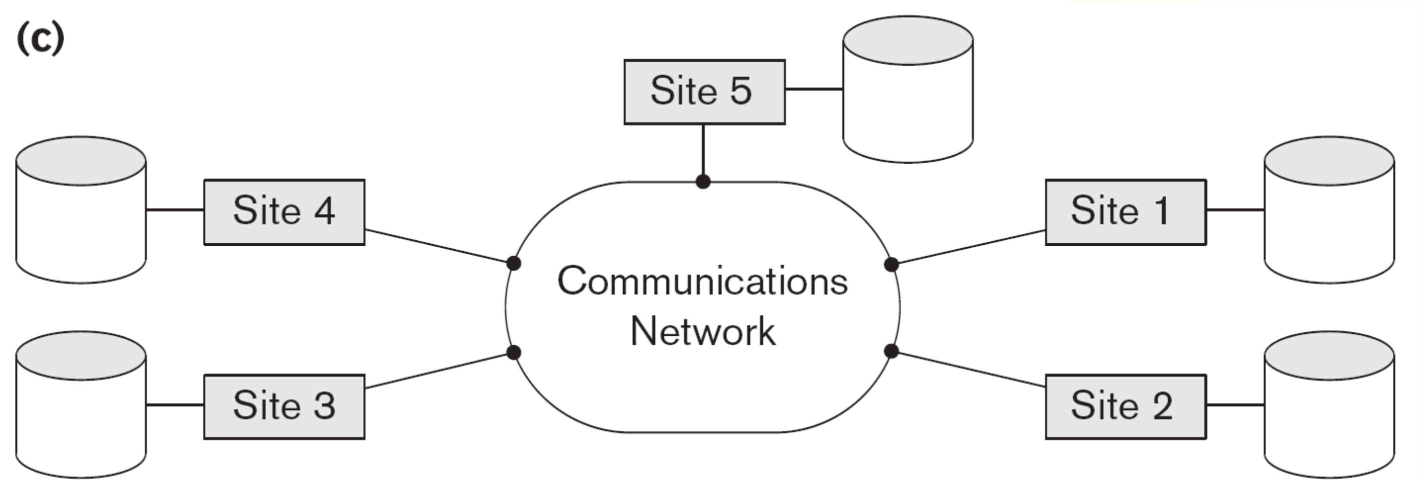
Distributed Database Concepts

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

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).

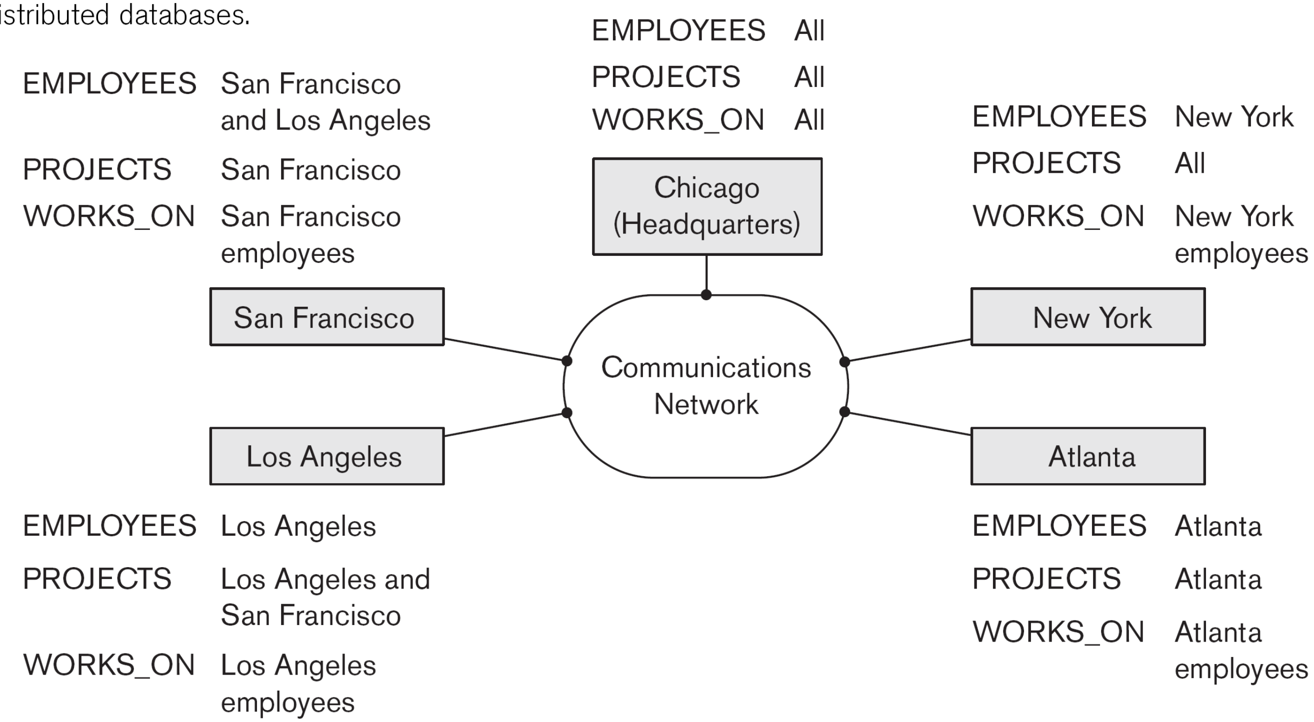
Figure 25.3
Some different database system architectures. (a) Shared nothing architecture. (b) A networked architecture with a centralized database at one of the sites. (c) A truly distributed database architecture.



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.
      - 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 as shown in the above diagram.
    - 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**

Data Fragmentation, Replication and Allocation

- **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.

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 of 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

- **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 ≠ 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 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.

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, 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.

Data Fragmentation, Replication and Allocation

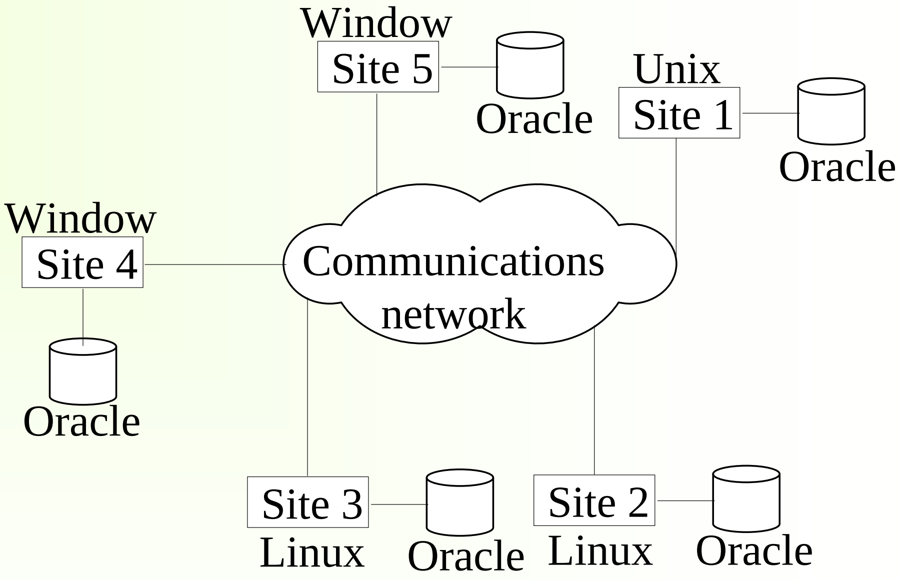
- **Fragmentation schema**
  - A definition of a set of fragments (horizontal or vertical or horizontal and vertical) that includes all attributes and tuples in the database that satisfies the condition that the whole database can be reconstructed from the fragments by applying some sequence of UNION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

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.
  - 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.

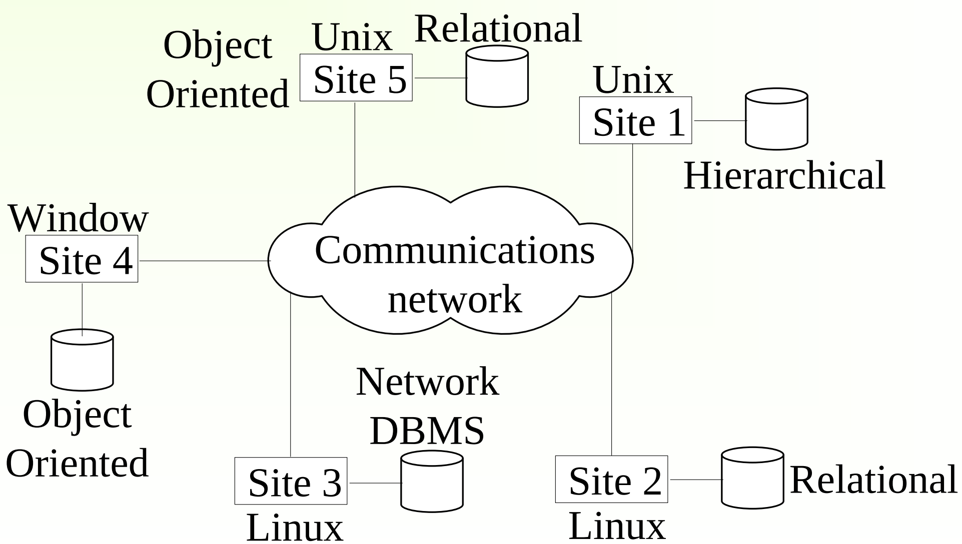
Types of Distributed Database Systems

- 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.

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.

Addison-Wesley is an imprint of PEARSON

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.

Addison-Wesley
is an imprint of
PEARSON

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

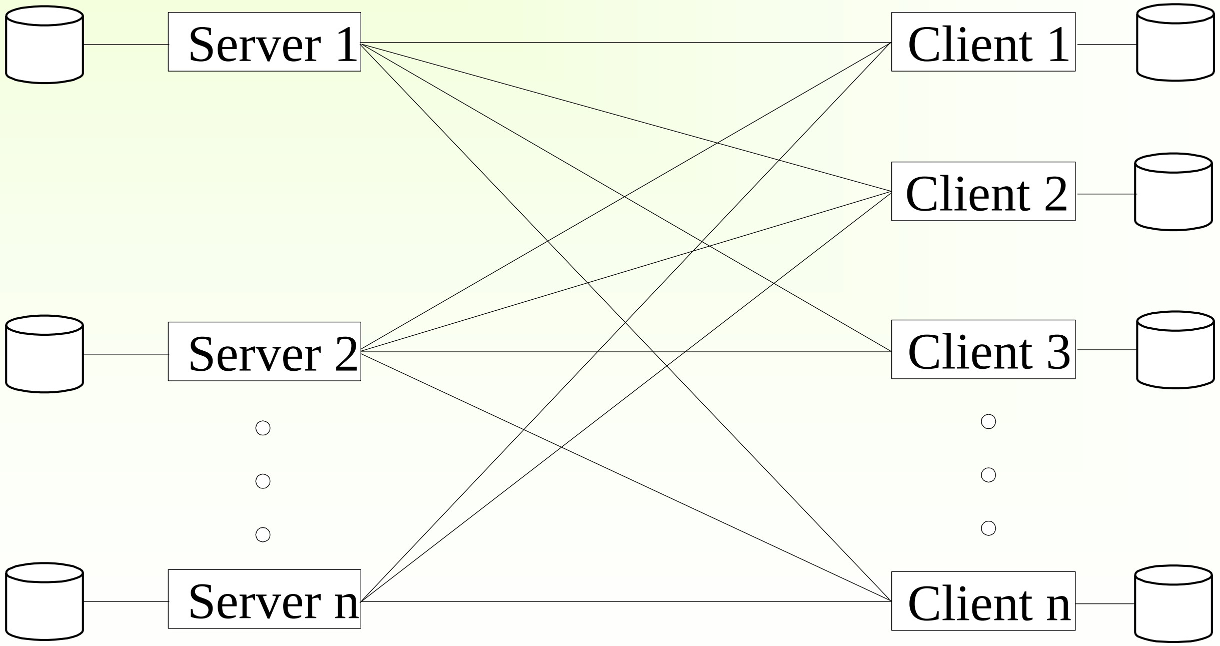
- Now suppose the result site is 2. Possible strategies :
  1. Transfer Employee relation to site 2, execute the query and present the result to the user at site 2.
     - Total transfer size = 1,000,000 bytes for both queries Q and Q'.
  2. Transfer Department relation to site 1, execute join at site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

Query Processing in Distributed Databases

- Semijoin:
  - Objective is to reduce the number of tuples 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.

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

- Clients reach server for desired service, but server does reach clients.
- The server software is responsible for local data management at a site, much like centralized DBMS software.
- The client software is responsible for most of the distribution function.
- The communication software manages communication among clients and servers.

Client-Server Database Architecture

- 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.

Recap

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