DISTRIBUTED DATABASES

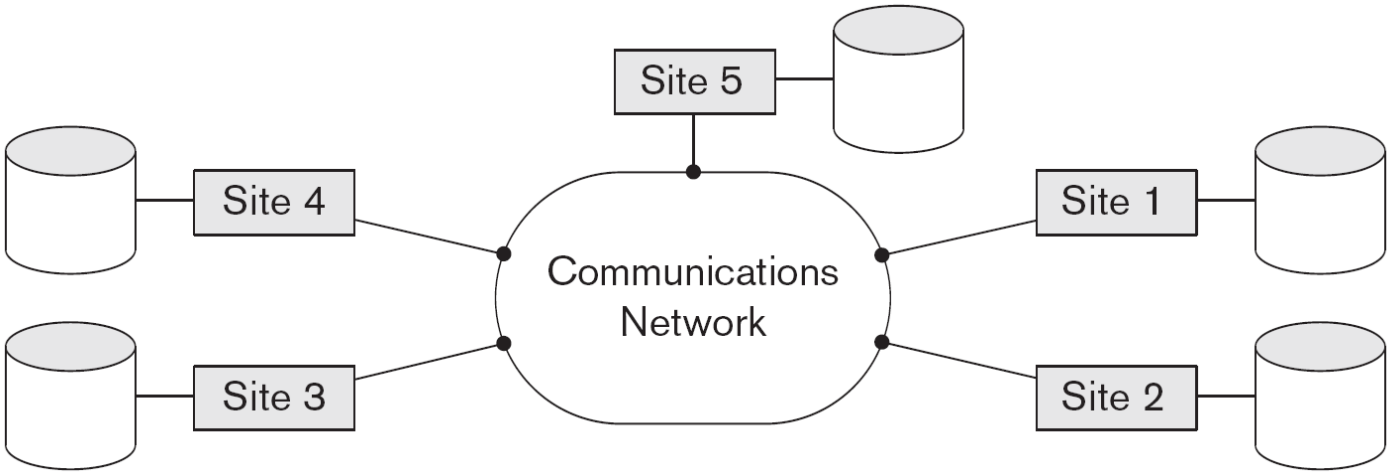
CHAPTER 25

LECTURE OVERVIEW

- What are distributed databases?
- Transparency and autonomy
- Fragmentation, allocation, replication
- Homogeneous vs. heterogeneous DDBMS
- Distributed transaction processing

WHAT ARE DISTRIBUTED DATABASES?

- Distributed Database
  - A logically interrelated collection of shared data (and a description of this data), physically distributed over a computer network
- Distributed DBMS
  - Software system that permits the management of the distributed database and makes the distribution transparent to users
  - Decentralized processing, but (logically) integrated information resources



WHY DDBMSS?

- Advantages
  - Potential for parallel execution
    - More resources available to process queries faster
  - Potential for **replicated** data
    - Processing closer to users' locations
      - Reduced communications time
    - Duplication in case of failures
- Disadvantages
  - Management more complex
  - Higher cost of installation and operation
  - Higher cost of protection (security)
  - Transaction control more difficult

PROPERTIES

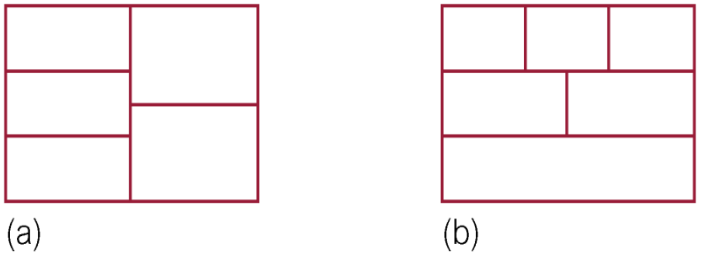
- **Transparency**
  - Hiding distribution details from users
- **Autonomy**
  - Degree to which databases in a connected distributed database can operate independently
    - High autonomy allows flexibility for a participating DBs and DBMSs
    - Can each have its own data model?
    - Can each decide how much and which data to share?
    - Can each decide which transactions to execute and in what order?

DATA ALLOCATION

- Assume users are geographically distributed.
- Alternative strategies for placing data
  - **Centralized**
    - Single database stored at one site
    - DBMS functionality might be partially distributed
  - **Partitioned**
    - Database partitioned into disjoint fragments
    - Each fragment assigned to one site
  - **(Complete) Replication**
    - Complete copy of database at each site
  - **Selective Replication**
    - Combination of partitioning, replication, and centralization.

FRAGMENTATION

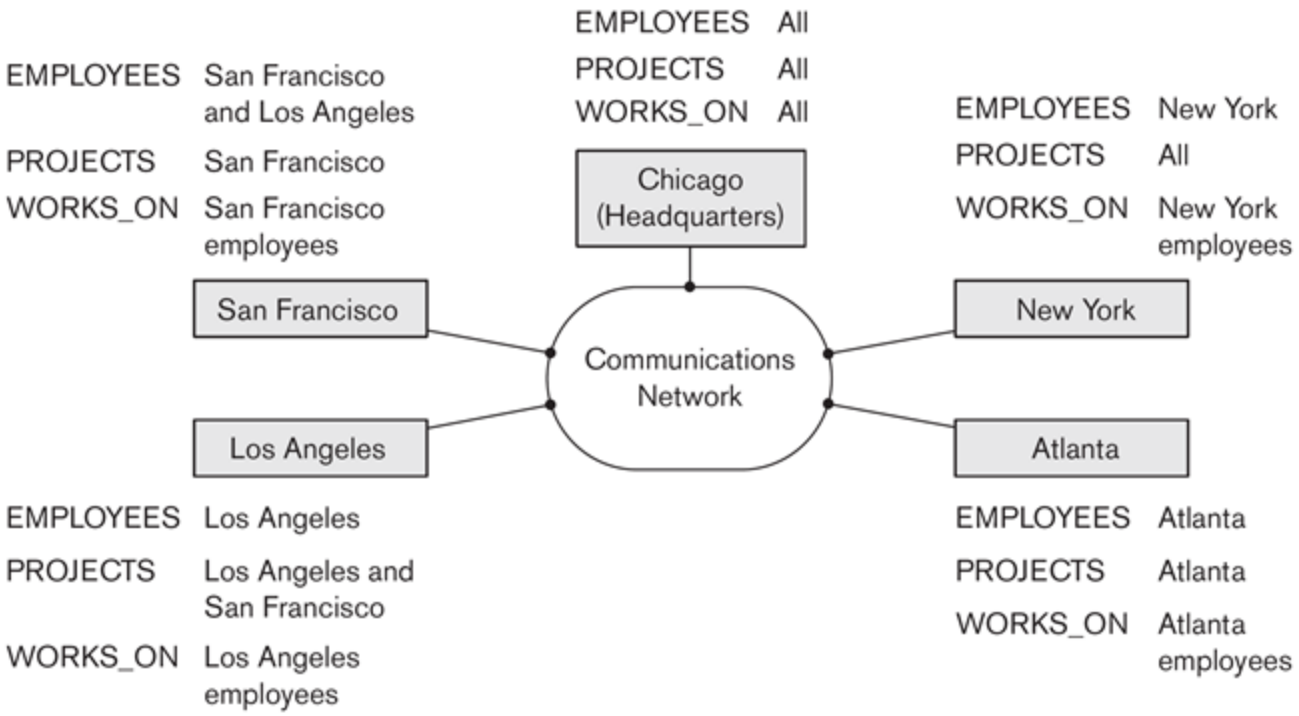
- Splitting data among sites
  - **Horizontal fragmentation**: distributing rows of tables
  - **Vertical fragmentation**: distributing (columns of) tables



Mixed fragmentation

  - With or without replication
- Usage
  - Applications work with views rather than entire relations.

FRAGMENTATION EXAMPLE

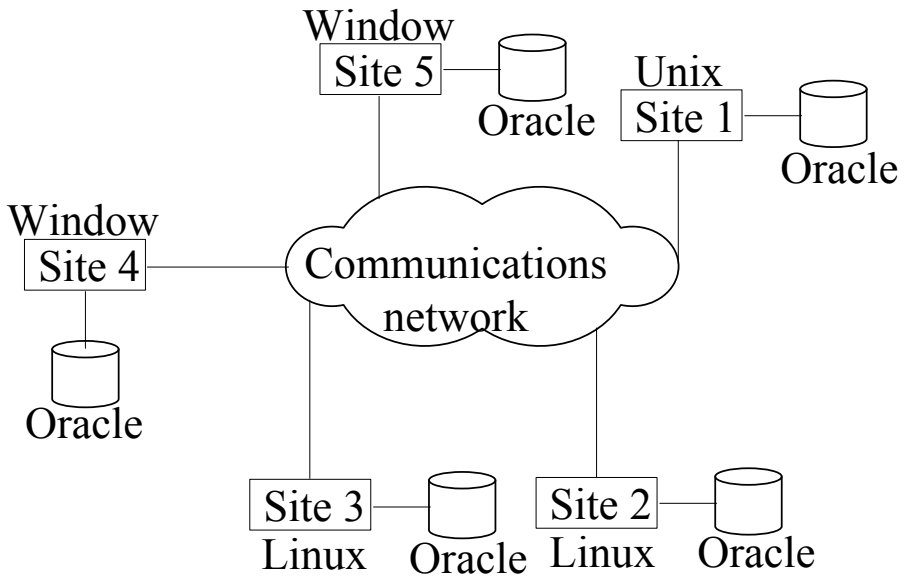


WHY FRAGMENT DATA?

- Advantages
  - Data is stored close to where it is most frequently used.
  - Data not needed by local applications is not stored.
    - Materialized view
  - Transactions can be divided into subqueries that operate on fragments and operate in parallel.
  - Data not required by local applications is not available to unauthorized users.
- Disadvantages
  - May need to access multiple sites to retrieve data
  - Need joins and unions to reconstruct complete relations
  - Difficult to maintain consistency of data across sites

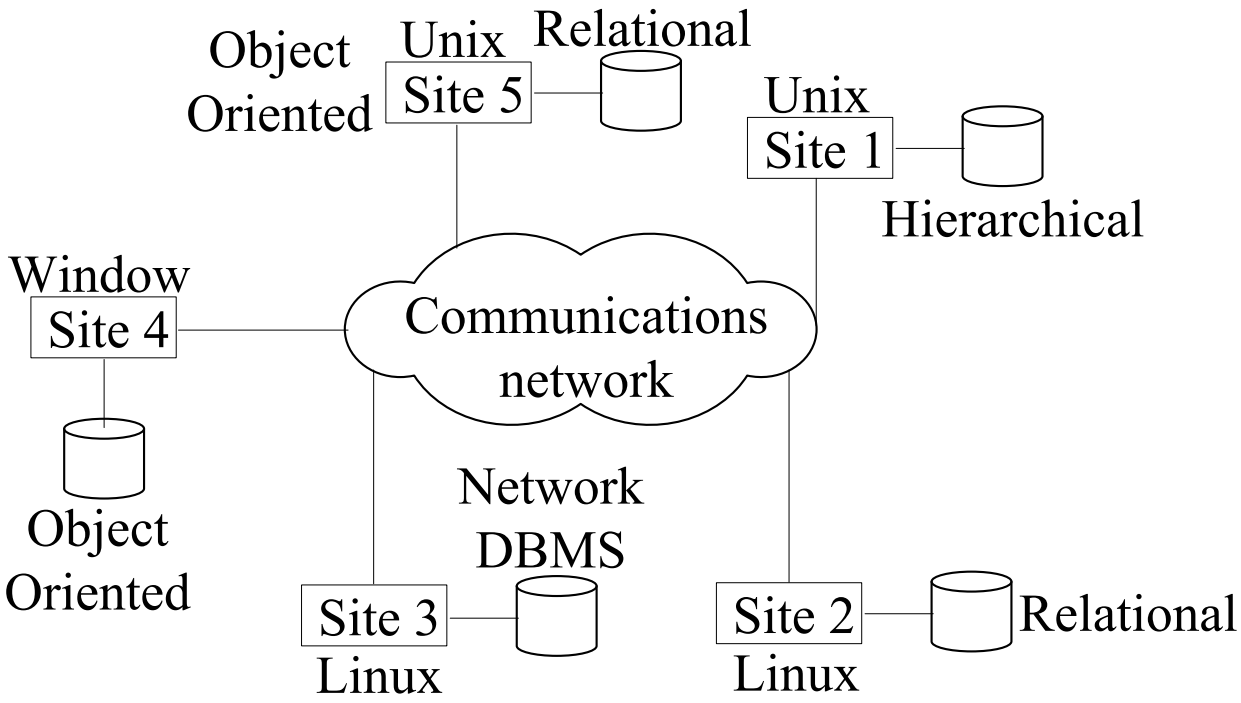
HOMOGENEOUS DDBMS

- All sites use same DBMS product.
  - Much easier to design and manage
  - Note: The deployed operating systems may differ.
- Provides for incremental growth and allows increased performance



HETEROGENEOUS DDBMS

- Sites may run different DBMS products, with possibly different underlying data models.
- **Wrappers**
  - Translations required at interfaces to convert queries and data into common models or dialects.



FEDERATED DATABASE SYSTEMS

- Integration of *autonomous* DBs and DBMSs
- Usually highly heterogeneous
  - Possibly without a common conceptual schema
  - Usually including *semantic heterogeneity*
    - *Meaning* of data varies across systems.
    - Different units of measure (e.g., dollars vs. euros; feet vs. meters)
    - Different attributes (e.g., based on local accounting practices)
    - Different degrees of accuracy (e.g., outdated and missing values)
- **Mediators**
  - Middleware, such as Enterprise Resource Planning (ERP)
    - Manage transport of transactions and queries
    - Integrate data from variety of sites

CLICKER QUESTION #40

Amazon maintains many geographically distributed databases and systems, each with different functions (security, recovery, query processing, etc.) and following the conventions of the country in which it is located (currency, shipping, accounting, taxes, etc.).

This is (probably) best described as a:

A. centralized database
B. distributed homogeneous database
C. heterogeneous, but not a federated, database
D. federated database
E. mess

DDBMS TRANSACTION CONTROL

- Dealing with replicated data
  - Requires global consistency
    - Replicated updates
    - Distributed concurrency control
  - Recovery mechanism must recover all copies
    - Consistent undo/redo across sites
- Dealing with failure of individual sites
  - Global awareness of failed sites
  - Non-disruption of applications that need no data from failed sites
- More complicated management
  - Distributed commit
  - Distributed locking and deadlock detection

LOCK MANAGEMENT

- **Primary site** technique
  - Single site designated as coordinator for transaction management
  - All lock requests go to primary site
    - Might overload site
  - If primary site fails, entire system inaccessible
    - To aid recovery, backup site designated to shadow primary site and replace primary site if needed
- **Primary copy** technique:
  - One replica designated as primary copy for each data item
    - Primary copies distributed at various sites
    - All lock requests for an item go to site holding primary copy of that item
    - Single site not overloaded with transaction management
  - Identification of primary copy (and backup) complex in light of site failures
    - Distributed directory must be maintained, possibly at all sites.
- **Majority vote** technique:
  - Try to lock data item at several sites having copies
  - More than 50% of sites that hold data must grant lock
    - Possible optimizations for improving either readers or writers
  - Complex protocol and deadlock detection in light of site failures

LECTURE SUMMARY

- Overview of distributed database concepts
  - What are the main reasons for and potential advantages of distributed databases?
- Data fragmentation, replication and allocation
  - What is a fragment of a relation? What are the main types of fragments? Where are they stored? Why is fragment a useful concept in distributed database design?
- Peek into distributed transaction management
  - How does the primary site method compare to the primary copy method for distributed concurrency control? When voting, why do we need to obtain so many locks?