## Database Systems II

**Lecture 11**

**Distributed Database Systems**

## Lecture Outline

* What is a Distributed Database?
  + Centralized Vs DDBs
  + Parallel Vs DDBs
* Distributed Data Design
* Types of DDBs
* Reference Architecture for DDBMS
* Advantages of DDBs

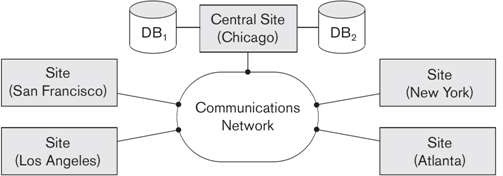
What is a Distributed Database?

* A collection of multiple logically related database distributed over a computer network.
* Database whose relations *reside* on different sites
* Database whose some of its relations are *replicated* at different

sites

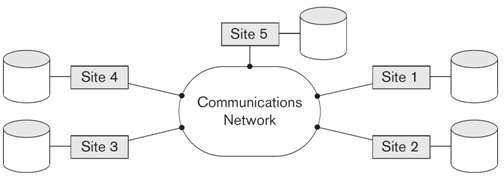
* Database whose relations are *split* between different sites
* A *Distributed Database Management System (DDBMS)* is the software system that manages a distributed database and makes the distribution *transparent* to the user.

**Centralized Vs. Distributed Databases**



**In centralized database**

* Data is located in one place (one server)
* All DBMS functionalities are done by that server



**In Distributed Databases**

* Data is stored in multiple places (each is running a DBMS)
* DBMS functionalities are distributed over many machines

# Why Might Data be Distributed

* To minimize communication costs or response time
* Maintain control and security
* To increase its availability in the event of failure.
* Data is too large

## Parallel VS. Distributed Databases

**In Parallel Database System (**To improve performance through parallelization**)**

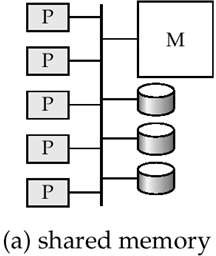
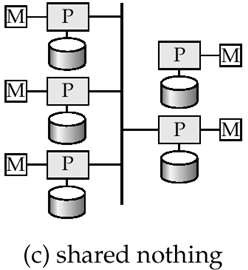
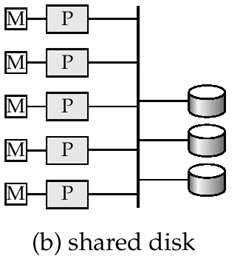
* DBMS running across multiple processors and disks that is designed to execute operations in parallel, whenever possible, in order to improve performance.
* Distributed processing usually imply parallel processing (not distribution of data)
* Can have parallel processing on a single machine

**In Distributed Database System (**To increased availability**)**

* Data is physically stored across several sites, and each site is managed by a DBMS capable of running independent of the other sites.
* In contrast to parallel databases, sharing data is the key of a DDBs

Three possible architectures for passing and processing data:

* 1. **Shared memory** -- processors share a common memory
  2. **Shared disk** -- processors share a common disk
  3. **Shared nothing** -- processors share neither a common memory nor common disk

###### In Parallel Databases

* Machines are physically close to each other, e.g., same server room
* Machines connects with dedicated high-speed LANs and switches
* Communication cost is assumed to be small
* Can *shared-memory*, *shared-disk*, or *shared-nothing* architecture

###### In Distributed Databases

* Machines can be far from each other, e.g., in different continent
* Can be connected using public-purpose network, e.g., Internet
* Communication cost and problems cannot be ignored
* *Usually shared-nothing architecture.*
* **Three key issues:**

1. **Fragmentation**

Relation may be divided into a number of sub- relations, which are then distributed.

#### Allocation

Each fragment is stored at site with “optimal” distribution**.**

#### Replication

Copy of fragment may be maintained at several sites.

**Distributed Data Design: Fragmentation**

###### Data Fragmentation

* + Split a relation into logically related parts. A relation can be fragmented in three ways:
    - Horizontal Fragmentation
    - Vertical Fragmentation
    - Mixed (Hybrid) fragmentation

**Horizontal Fragmentation**

* + It is a horizontal subset of a relation which contain tuples that

satisfy a 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.

* + To reconstruct R from horizontal fragments a UNION is applied.

**Horizontal fragmentation**

|  |  |  |  |
| --- | --- | --- | --- |
| 6/1/07 | 424252 | Couch | $570 |
| 6/1/07 | 256623 | Car | $1123 |
| 6/2/07 | 636353 | Bike | $86 |
| 6/5/07 | 662113 | Chair | $10 |
| 6/7/07 | 121113 | Lamp | $19 |
| 6/9/07 | 887734 | Bike | $56 |
| 6/11/07 | 252111 | Scooter | $18 |
| 6/11/07 | 116458 | Hammer | $8000 |

*Server 1 Server 2 Server 3 Server 4*

* + Horizontal Fragmentation Example:

***P1* =**  **Dno=‘5’(*Employee*)**  **Site 1**

***P2* =**  **Dno=‘7’(*Employee*)**  **Site 2**

To reconstruct Employee relation:

***P*1 U *P2***

**Vertical Fragmentation**

* + - It is a subset of a relation which is created by a subset of columns. There is no selection condition used in vertical fragmentation.
    - Consider the Employee relation. A vertical fragment can be created

by projecting (π) the values of Name, DoB, 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.
    - To reconstruct R from complete vertical fragments a OUTER JOIN is applied.

**Distributed Data Design: Fragmentation**

* **Vertical Fragmentation**

636353

662113

424252

256623

121113

887734

252111

116458

636353

662113

424252

256623

121113

887734

252111

116458

116458

252111

887734

121113

662113

636353

256623

424252

|  |  |  |
| --- | --- | --- |
| 6/1/07 | Couch | $570 |
| 6/1/07 | Car | $1123 |
| 6/2/07 | Bike | $86 |
| 6/5/07 | Chair | $10 |
| 6/7/07 | Lamp | $19 |
| 6/9/07 | Bike | $56 |
| 6/11/07 | Scooter | $18 |
| 6/11/07 | Hammer | $8000 |

*Server 2 Server 3 Server 4*

**Distributed Data Design: Fragmentation**

##### Vertical Fragmentation Example:

**S1 =**  **staffNo, position, DOB, salary(Staff)**

**S2 =**  **staffNo, fName, lName, branchNo(Staff)**

To reconstruct Staff relation:

**S1 S2**

Slide 25- 17

**Distributed Data Design: Fragmentation**

**Correctness of Fragmentation *(Three Rules)***

* **Completeness:**

If relation *R* is decomposed into fragments *R1*, *R2*, ... *Rn*, each data item that can be found in *R* must appear in at least one fragment.

* **Reconstruction:**
  + Must be possible to define a relational operation that will reconstruct R from the

fragments.

* + Reconstruction for horizontal fragmentation is Union operation and Outer Join for vertical .
* **Disjointness:**

If data item *di* appears in fragment *Ri*, then it should not appear in any other fragment. Exception: vertical fragmentation, where primary key attributes must be repeated to allow reconstruction.

**Distributed Data Design: Replication**

#### Data Replication

* + In *full replication* the entire database is replicated and in *partial (selective) replication* some selected part is replicated to some of the sites.
  + *Synchronous* and *asynchronous* replication.

#### Advantages of Replication

* **Availability**: failure of site containing relation *r* does not result in unavailability of *R* is replicas exist.
* **Parallelism**: queries on *R* may be processed by several nodes in parallel.
* **Reduced data transfer**: relation *R* is available locally at each site containing a replica of *R*.

#### Disadvantages of Replication

* Increased cost of updates: each replica of relation *r*

must be updated.

* Increased complexity of concurrency control: concurrent updates to distinct replicas may lead to inconsistent data unless special concurrency control mechanisms are implemented.

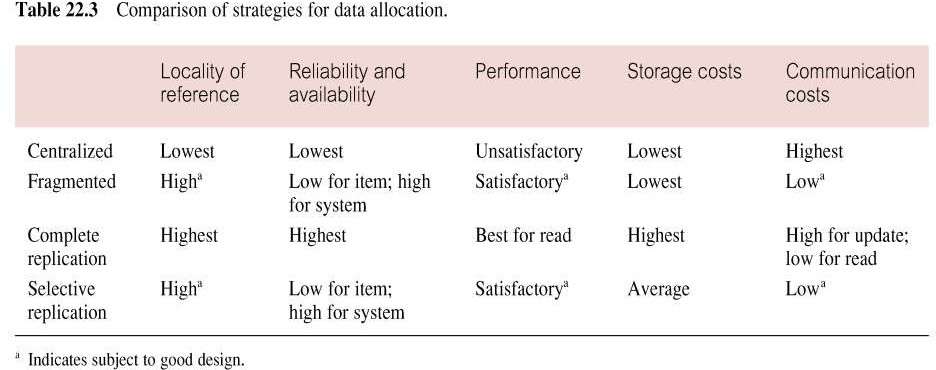
###### Fragmentation schema

* It describes all set of fragments (horizontal, vertical, or mixed) that includes all attributes and tuples in the database that satisfies the condition so that the whole database can be reconstructed when needed.

###### Allocation schema

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

**Comparison of Strategies for Data Distribution**



#### Types of Distributed Database Systems

* Three main factors are used to differentiate between different types of DDBMSs.

###### Autonomy

* + **Distribution**

###### Heterogeneity

* Autonomy refers to the ***distribution of control*** not of data
* It indicates the degree to which individual DBMS can operate independently (*self control*).
* Autonomy is a function of number of factors like whether the component systems exchange Info,

whether they can independently execute transactions, and whether one is allowed to modify them.

* + Design autonomy:
    - Individual DBMSs are free to use the data models and transaction management technique that they prefer.
  + Communication autonomy:
    - Each individual DBMSs is free to make its own decision on providing other DBMSs with information.
  + Execution autonomy:
    - Each DBMS can execute the transactions that are submitted to it in anyway that it wants to.
* *Autonomy* refers to distribution of **control** while *Distribution*

refers to the taxonomy of **data**.

* Here we consider physical distribution of data over multiple sites.
* The user sees data as one logical pool.
* Three levels:
  + non-distributed option (centralized).
  + client-server distribution
  + peer –to – peer distribution(Full)
* The client/ server distribution concentrates data management duties at servers while the client focuses on providing the application environment including the user interface.
* In peer to peer system, there is no distinction of client machines and server. Each machine has full DBMS functionality and can communicate with other machines to execute queries and transactions.
* Heterogeneity may occur in various forms in distributed systems from hardware heterogeneity and differences in networking protocols to variation in data managers.
* The important ones are related to data models, query languages and transaction management protocols.
* Representing data with different data modeling tools creates heterogeneity because of inherent expressive

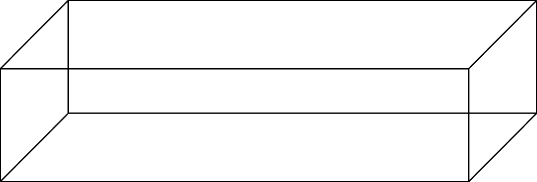
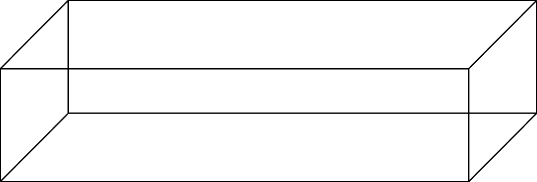
powers and limitations of individual data models.

* + Autonomy(A): Controller
    1. – Right Integration
    2. – Semi-autonomous System 2 - Isolation
  + Heterogeneity(H):

1. – Homogeneous
2. - Heterogeneous
   * Distribution(D): Data Management
3. – No Distribution
4. – Client – serve Architecture 2 – Peer-to-peer Architecture

### Classification of DDBMS

**Distribution**



Peer-to-Peer

Distributed DBS

Distributed Multi- DBS

Client\server

**Autonomy**

Multi-DBS

**Heterogeneity**

Federated DBS

* **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.

* **Multi-database:** There is no one conceptual global schema. For data access a schema is constructed dynamically as needed by the application software.

DDBMS Architecture with a GCS

GCS

GES2

GES n

GES1

LCS 2

Multi-database Layer

Local System Layer

LCS n

LIS n

LIS 2

LIS 1

LCS 2

LCS 1

ES n

ES2

ES1

LIS 2

|  |  |
| --- | --- |
| LCS 1 | |
|  |  |
| LIS 1 | |

|  |  |
| --- | --- |
| LCS n | |
|  |  |
| LIS n | |

DDBMS Architecture without a GCS

###### 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.
* **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.