Distributed Database Management System (DDBS)

**Motivation:** Data is used at multiple distributed sites (e.g. Branch offices).

Communication between sites is

----- **costly**

----- **potentially unreliable**

**Solution:**

----- Allow sites to store/maintain the data they use most often/specialize in

----- Sharing with other sites/HQs if combinations of data necessary
Network Topology

- Fully connected network
- Partially connected network
- Tree structured network
- Star network
- Ring network

Cost/reliability
#of hops
Tradeoffs between

1. keeping data in centralized headquarters:
   • simpler maintenance
   • simpler consistency enforcement
   • possibly more efficient if many updates, aggregate computations

2. or distributed across branch offices:
   • lower communication cost
   • reliability
   • parallelism can be implemented locally
Advantages of DDBS (heterogeneous)

- **Interconnectivity** of pre-existing DBs
- **Expandability** (don’t need to replace whole system to grow)
- **Cost** (many small engines on PC’s cheaper than mainframes) → *issue: communication costs vs. hardware computation costs.*
- **Performance** (place data near where used)
- **Availability** and reliability
Complicating factors

• Maintaining data **consistency** *(in face of replication and sharing)*

• **Distributed directory management** *(who controls mapping of data to sites)*

• Security

• **Heterogeneous Databases** *(different database architectures)*
Distributed Database Design Issues

Options for storing a relation $R$ across multiple sites:

- **Replication** *(maintain copies/replicas of $R$ on multiple sites)*
- **Fragmentation** *(Relation store in fragments/pieces on multiple sites)*
- combination of both
REPLICATION

(Vertical)

FRAGMENTATION

(Horizontal)
Replication

• **Issues:** (whole database replication v.s. no replication)
  - what to replicate? (all relations or only frequently user shared data)
  - where to replicate? (function of communication costs, usage needs, resources)
  - which relations to replicate?
  -” primary copy” of relation (simplifies consistency enforcement, but where located?)
Replication (cont)

• Advantages:
  
  – **Improved availability** *(multiple sources for a relation if a site is down)*

        ↑

  – **Increased parallelism** *(sites can process (primarily) read-only operations in parallel, minimizing data transfer)*

    *(well suited for read-only, majority read-only data access)*
Replication (cont)

• **Disadvantages:**
  - problems/overhead for writes/updates
  - costs of consistency enforcement
    - updates propagated to all sites (communication costs)
    - costs of synchronization/locking for consistency enforcement on update greater than in single source models.

➢ Complicates concurrency and recover

➢ Replication inefficient in databases with frequent updates
FRAGMENTATION

• Vertical
• Horizontal
• mixel

Issues:
- **completeness**: Every tuple/attribute in some fragment
- **reconstruction**: easy way of reconstructing full relation
- **transparency**
Fragments contain subsets of complete tuples (all attributes at all sites)

**How to reconstruct**

\[ R = R_{s1} \cup R_{s2} \cup \ldots \cup R_{sn} \]
Horizontal Fragmentation

• Example Usefulness:
  
  - Each branch office maintains complete attribute set of its employees (salary, benefits, address/phone, departments, projects, etc.)

  - Site of Fragment easily determined by a key attribute value - e.g. Branch_office*
VERTICAL FRAGMENTATION

Original Relation

(R) t1
   t2
   tn

How to Reconstruct:
R=Rs1 Rs2 Rsn

TID – Tuple ID
Hidden Attribute to ensure account and simple join reconstruction

RS1

RS2

TID

1
2
n

SITE1

SITE2

RS1.TID=RS2.TID

Join condition
Example usefulness:

Salary  Office
Benefits  Office
Directory (Name|address|phone|fax)
Dependents Management Office
each control their own appropriate
attribute for all corporate branch offices

VERTICAL — Attribute-centered management
(keep all instances of an attribute in one place)

HORIZONTAL — tuple/individual-centered management
(keep all values of a tuple in one place)
## MIXED FRAGMENTATION

![Diagram showing mixed fragmentation with attributes and values](image)

- **Rs1**: A1, A2, A3
- **Rs2**: A1, A2, A3
- **Rs3**: A4, A5
- **Rs4**: A4, A5

### Attributes
- **Salary Attributes**: A1, A2, A3
- **Benefit Attributes**: A4, A5

### Regions
- **USA**
- **Europe**
Partition of Attributes/tuples need not be disjoint
TRANSPARENCY

Fragmentation Transparency
- User doesn’t need to know mapping between relations and fragmented subrelations

Replication Transparency
- User doesn’t need to know about existence or location of other copies (treat as if single copy of DB)

Location and Naming Transparency
- Use shouldn’t need to know about location and full names of data on the server

$$\Pi_{\text{Salary} (\sigma_{\text{ssn}=\text{so(Employee)}})} \text{ Site27, Employee. Fragment3. Replica7}$$
**Issues 1:**

Parallel Processing across Fragments

\[ \Pi_{\text{LName}}(\sigma \text{salary}>40,000(\text{Employee})) \]

\[ \Rightarrow \Pi_{\text{LName}}(\sigma \text{salary}>40,000(\text{Emp1})) \cup \Pi_{\text{LName}}(\sigma \text{salary}>40,000(\text{Emp2})) \]

= \text{Emp1 U Emp2}

Horizontal fragmentations

2 Fragments

Execution in Parallel on fragments

and union results together
Site1  Site2  Site3

50K  1K  3K

(A \times B) \times C

1K  \times  3K

\underline{0.5K}

A \times (B \times C)

50K  \times  0.5K

\underline{0.5K}

\underline{0.5K}

Joins- symmetric and associative

Parallel Processing

(\sigma_{xx}(A)) \times (B \times C)
JOIN STRATEGIES

\[ R = \Pi \text{Fnames, Cnames, Dnames} \ (\text{Employee} \bowtie \text{Department}) \]

<table>
<thead>
<tr>
<th>Site 3</th>
<th>Site 1</th>
<th>Site 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 records, 2000 bytes</td>
<td>10,000 records, 1,000,000 bytes</td>
<td>100 records, 3000 bytes</td>
</tr>
</tbody>
</table>

Strategies:

1) Ship both relations to the result site and join there

2) Ship employee to 2, join at 2, results to 3

3) Ship Department to 1, join at 1, results to 3

\[ \Rightarrow \text{minimize total communication cost of data transfer} \]


**RECOVERY IN DDBS**

- transaction managers / coordinators
- log managers

**Problems:**
- failure of site
- failure of link
- loss of messages

\[
\begin{align*}
\text{Difficult to know which had occurred} \\
\text{if server is down, elect new server} \Rightarrow \text{what about network partitioning?}
\end{align*}
\]

\[\text{Diagram of network partitioning with original server and newly elected server.}\]