What is a distributed DBMS?

Distributed DBMS Architecture

What is a Distributed Database System?

A distributed database (DDB) is a collection of multiple, logically interrelated databases distributed over a computer network.

A distributed database management system (D–DBMS) is the software that manages the DDB and provides an access mechanism that makes this distribution transparent to the users.

Distributed database system (DDBS) = DDB + D–DBMS

What is not a DDBS?

- A timesharing computer system
- A loosely or tightly coupled multiprocessor system
- A database system which resides at one of the nodes of a network of computers - this is a centralized database on a network node

Centralized DBMS on a Network

Distributed DBMS Environment

Implicit Assumptions

- Data stored at a number of sites → each site logically consists of a single processor
- Processors at different sites are interconnected by a computer network → not a multiprocessor system
  - Parallel database systems
- Distributed database is a database, not a collection of files → data logically related as exhibited in the users’ access patterns
  - Relational data model
- D–DBMS is a full-fledged DBMS
  - Not remote file system, not a TP system
Data Delivery Alternatives

- Delivery modes
  - Pull-only
  - Push-only
  - Hybrid
- Frequency
  - Periodic
  - Conditional
  - Ad-hoc or irregular
- Communication Methods
  - Unicast
  - One-to-many
- Note: not all combinations make sense

Distributed DBMS Promises

- Transparent management of distributed, fragmented, and replicated data
- Improved reliability/availability through distributed transactions
- Improved performance
- Easier and more economical system expansion

Transparency

- Transparency is the separation of the higher level semantics of a system from the lower level implementation issues.
- Fundamental issue is to provide data independence in the distributed environment
  - Network (distribution) transparency
  - Replication transparency
  - Fragmentation transparency
    - horizontal fragmentation: selection
    - vertical fragmentation: projection
    - hybrid

Example

<table>
<thead>
<tr>
<th>EMP</th>
<th>ENAME</th>
<th>TITLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Doe</td>
<td>Manager</td>
</tr>
<tr>
<td>P2</td>
<td>Smith</td>
<td>Analyst</td>
</tr>
<tr>
<td>P3</td>
<td>Jones</td>
<td>Consultant</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ASG</th>
<th>NAME</th>
<th>RESP</th>
<th>DUR</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>Eng</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>E2</td>
<td>Eng</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>E3</td>
<td>Eng</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>E4</td>
<td>Eng</td>
<td>26</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PAY</th>
<th>TITLE</th>
<th>SAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>Eng</td>
<td>40000</td>
</tr>
<tr>
<td>E2</td>
<td>Eng</td>
<td>34000</td>
</tr>
<tr>
<td>E3</td>
<td>Eng</td>
<td>27000</td>
</tr>
<tr>
<td>E4</td>
<td>Eng</td>
<td>24000</td>
</tr>
</tbody>
</table>

Transparent Access

```sql
SELECT ENAME, SAL
FROM EMP, ASG, PAY
WHERE DUR > 12
AND EMP.ENO = ASG.ENO
AND PAY.TITLE = EMP.TITLE
```

Distributed Database - User View
Distributed DBMS - Reality

Types of Transparency

• Data independence
• Network transparency (or distribution transparency)
  ➡ Location transparency
  ➡ Fragmentation transparency
• Replication transparency
• Fragmentation transparency

Reliability Through Transactions

• Replicated components and data should make distributed DBMS more reliable.
• Distributed transactions provide
  ➡ Concurrency transparency
  ➡ Failure atomicity
• Distributed transaction support requires implementation of
  ➡ Distributed concurrency control protocols
  ➡ Commit protocols
• Data replication
  ➡ Great for read-intensive workloads, problematic for updates
  ➡ Replication protocols

Potentially Improved Performance

• Proximity of data to its points of use
  ➡ Requires some support for fragmentation and replication
• Parallelism in execution
  ➡ Inter-query parallelism
  ➡ Intra-query parallelism

Parallelism Requirements

• Have as much of the data required by each application at the site where the application executes
  ➡ Full replication
• How about updates?
  ➡ Mutual consistency
  ➡ Freshness of copies

Distributed DBMS Issues

• Distributed Database Design
  ➡ How to distribute the database
  ➡ Replicated & non-replicated database distribution
  ➡ A related problem in directory management
• Query Processing
  ➡ Convert user transactions to data manipulation instructions
  ➡ Optimization problem
    ➡ \text{min}(\text{cost}) = \text{data transmission} + \text{local processing}
  ➡ General formulation is NP-hard
Distributed DBMS Issues

- Concurrency Control
  - Synchronization of concurrent accesses
  - Consistency and isolation of transactions’ effects
  - Deadlock management
- Reliability
  - How to make the system resilient to failures
  - Atomicity and durability

Relationship Between Issues

Architecture

- Defines the structure of the system
- Components identified
- Functions of each component defined
- Interrelationships and interactions between components defined

ANSI/SPARC Architecture

DBMS Implementation Alternatives

Dimensions of the Problem

- Distribution
  - Whether the components of the system are located on the same machine or not
- Heterogeneity
  - Various levels (hardware, communications, operating system)
  - DBMS important one
    - Data model, query language, transaction management algorithms
- Autonomy
  - Not well understood and most troublesome
  - Various versions
    - Design autonomy: Ability of a component DBMS to decide on issues related to its own design
    - Communication autonomy: Ability of a component DBMS to decide whether and how to communicate with other DBMSs
    - Execution autonomy: Ability of a component DBMS to execute local operations in any manner it wants to.