Topics

- Why DBMS?
- Why Distributed DBMS?
- Promises of Distributed Database Systems
- Complicating Factors & Problem Areas

What Is a DBMS?

- Database: A very large, integrated collection of data.
- Models real-world enterprise.
- Entities (e.g., students, courses)
- Relationships (e.g., Mike is taking CS4620)

A Database Management System (DBMS) is a software package designed to store and manage databases.
Why Use a DBMS?
- Data independence and efficient access.
- Reduced application development time.
- Data integrity and security.
- Uniform and centralized data administration.
- Concurrent access, recovery from crashes.

(Centralized) Database Processing

Topics
- Why DBMS?
- Why Distributed DBMS?
- Promises of Distributed Database Systems
- Complicating Factors & Problem Areas
It’s a New (Distributed) World!

- Network is ubiquitous
  - Information can be shipped to anywhere (Good!)
  - Demand may come from anywhere
    - Can you handle it???
    - Network delay (latency) matters!!!
- Commodity computing is cheap
- Idea: decouple integration & centralization

Distributed Computing

- No universal definition
- Working definition:
  - A number of autonomous processing elements that are interconnected by a computer network and cooperate in performing their assigned tasks.

Why Distributed Computing?

- The promise:
  - Cost-efficiency
  - Performance (low latency)
  - Scalability
  - Fault-tolerance:
    - Reliability
    - Availability
Idea: Distributed Database Processing

Integration ≠ Centralization

Distributed Database System:
- Distributed database (DDB)
  - a collection of multiple, logically interrelated databases distributed over a computer network.
- Distributed database management system (D–DBMS)
  - software that manages the DDB
  - provides an access mechanism that makes this distribution transparent to the users.
- Distributed database system (DDBS) = DDB + D–DBMS

These are not 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
- Processors at different sites are interconnected by a computer network:
  - no multiprocessors (parallel database systems)
- Distributed database is a database
  - not a collection of files with data logically related as exhibited in the users’ access patterns
- D-DBMS is a full-fledged DBMS
  - not a mere remote file system
Topics
- Why DBMS?
- Why Distributed DBMS?
- Promises of Distributed Database Systems
- Complicating Factors & Problem Areas

Promises of DDBS
- **Transparent** management of distributed, fragmented, and replicated data
- Improved reliability/availability (**fault-tolerance**) 
- Improved **performance**
- **Scalability**
  - Easier and more cost-efficient system expansion

Transparency
- **Definition:**
  - Separation of the higher level semantics of a system from the lower level implementation issues.
- **Types**
  - Data independence
  - Network (distribution) transparency
  - Replication transparency
  - Fragmentation transparency
Data Independence

- Common with centralized DBMS
- Logical data independence
  - Changes to the logical data structure are transparent to applications
  - E.g., a new field should be irrelevant to old applications
- Physical data independence
  - Changes to the physical data layout are transparent
  - E.g., new DBMS version, new disk, new OS, etc
- Note: Performance consideration may require breaking the transparency

Network Transparency

- Application is shielded from the network details
- Location transparency
  - System operation is the same regardless of data location and host system
  - Note: Users may have to specify location (as part of the name)
- Naming transparency
  - Unique identifiers for all objects regardless of location

Replication Transparency

- In a DDBS the same piece of data may exist at multiple locations (why?)
- Hide this from the user
- Issues?
Fragmentation Transparency

- Each DB relation may be fragmented among a number of locations (why?)
- Hide this from the user!
- Type of fragmentation
  - Horizontal (selection)
  - Vertical (projection)
  - Hybrid

Example

<table>
<thead>
<tr>
<th>FNAME</th>
<th>LNAME</th>
<th>TITLE</th>
<th>ENO</th>
<th>ENO</th>
<th>HID</th>
<th>TUR</th>
</tr>
</thead>
<tbody>
<tr>
<td>J. Doe</td>
<td>M. Smith</td>
<td>Elect. Eng.</td>
<td>E3</td>
<td>E3</td>
<td>E3</td>
<td>E3</td>
</tr>
<tr>
<td>L. Clark</td>
<td>R. Owen</td>
<td>Programmer</td>
<td>E5</td>
<td>E5</td>
<td>E5</td>
<td>E5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FNAME</th>
<th>LNAME</th>
<th>TITLE</th>
<th>ENO</th>
<th>ENO</th>
<th>HID</th>
<th>TUR</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Instrumentation</td>
<td>Elect. Eng.</td>
<td>P1</td>
<td>P1</td>
<td>P1</td>
<td>P1</td>
</tr>
<tr>
<td>P3</td>
<td>CAD/CAM</td>
<td>Programmer</td>
<td>P3</td>
<td>P3</td>
<td>P3</td>
<td>P3</td>
</tr>
<tr>
<td>P4</td>
<td>Maintenance</td>
<td>Engineer</td>
<td>P4</td>
<td>P4</td>
<td>P4</td>
<td>P4</td>
</tr>
</tbody>
</table>

Transparent Access

```sql
SELECT ENO, ENO
FROM EMP, SAL
WHERE DUR > 12
AND ENO = E"NO.ENG
AND SAL.TITLE = EMP.TITLE
```
Improving Performance

- Reduce latency (network delay)
  - Move the data close to its use
    - Implies fragmentation and/or replication
  - Issues?
- Speedup query execution
  - Inter-query parallelism
  - Intra-query parallelism
Parallelism Dilemma

- Query processing
  - High replication \(\Rightarrow\) high query parallelism
  - Full replication allows max parallelism
- Update processing
  - High replication \(\Rightarrow\) complicated updates
  - E.g., distributed concurrency control and commit protocols
  - No replication incurs min overhead
- Reducing transparency can help, or even be necessary

Scalability

- Ideal solution:
  - Just add capacity (CPU, storage, bandwidth)
- Problems?
  - Complexity grows very fast
  - Performance may not scale up
  - Even well-designed systems can do this only within a certain range

Topics

- Why DBMS?
- Why Distributed DBMS?
- Promises of Distributed DBMS
- Complicating Factors & Problem Areas
Complicating Factors

- DDBS Complexity
- Cost
  - Hardware vs. software
  - Maintenance
- Distribution of Control
- Security

DDBS Problem Areas

- Database design:
  - How to distribute the data?
  - Replication/fragmentation
  - Optimal solution is NP-hard!
- Query processing
  - How to convert user transactions to data manipulation instructions?
  - Costs: processing, communication
  - Optimal solution is NP-hard!

DDBS Problem Areas

- Directory management
  - How to manage meta information about the data?
    - E.g., description, location, etc.
    - (Almost) the same issues as with the data itself!
- Concurrency control
  - How to synchronize concurrent accesses?
  - Goal: mutual (eventual) consistency
    - Multiple copies must eventually converge to the same value
  - Common approaches:
    - locking
    - timestamping
DDBS Problem Areas

- Reliability
  - Fault detection
  - Fault recovery
- OS support
  - OS design goal may be in conflict with DDBS ones
- Heterogeneous DB (*multidatabase systems*)
  - Realistic scenario
  - Interoperability
    - Applies as a separate concern to most of the issues so far

DDBS Problems: The Big Picture