Department of Computer Information Systems

KEMU

Distributed DBMS

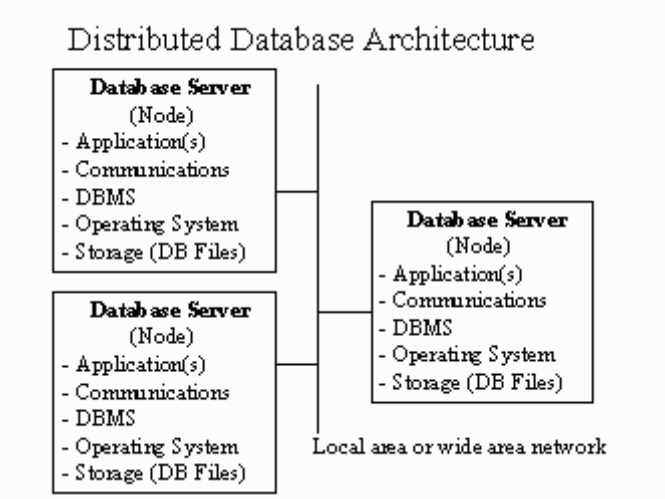
Objectives

Distributed Database Architecture

- Classifying DDBMS
  - Data fragmentation and Replication
  - Degree of Heterogeneity and Homogeneity
  - Degree of Autonomy
  - Degree of Distribution Transparency
- DDBMS Query Processing and Optimization
- DDBMS Concurrency Control
- DDBMS Transaction Processing

Distributed Database Architecture

- Database is distributed at the DBMS level.
- In a distributed database system (DDS), multiple Database Management Systems run on multiple servers (sites or nodes) connected by a network.



Classifying DDBMS

- There are four main dimensions on which DDBMS are classified:
  1. Data Distribution (Data fragmentation and/or replication)
  2. Degree of *homogeneity/heterogeneity*

3. Degree of *autonomy*
4. Degree of *distribution transparency*

Data Distribution - Data Fragmentation and Replication

- Data may be split up among the different nodes or it may be replicated.
- Tables may be located on different nodes connected by a network.
- Data may be split up (or fragmented) in several ways:
    1. **Horizontal**: Rows in a table are split up across multiple nodes.
    2. **Vertical**: Columns in a table are split across multiple nodes.
    3. Both vertical and horizontal.
- Splitting up data can improve performance by reducing contention for tables.

Customer Table

Customer ID	Name	Address	City	State	Zip
1001	Mr. Smith	123 Lexington	Smithville	KY	91232
1002	Mrs. Jones	12 Davis Ave.	Smithville	KY	91232
1003	Mr. Axe	443 Grinder Ln.	Broadville	GA	81992
1004	Mr. Builder	661 Parker Rd.	Streetville	GA	81990

Horizontal Partitioning:

Partition 1

Customer ID	Name	Address	City	State	Zip
1001	Mr. Smith	123 Lexington	Smithville	KY	91232
1002	Mrs. Jones	12 Davis Ave.	Smithville	KY	91232

Partition 2

Customer ID	Name	Address	City	State	Zip
1003	Mr. Axe	443 Grinder Ln.	Broadville	GA	81992
1004	Mr. Builder	661 Parker Rd.	Streetville	GA	81990

Vertical Partitioning:

Partition 1

CustID	Name
1001	Mr. Smith
1002	Mrs. Jones
1003	Mr. Axe
1004	Mr. Builder

Partition 2

CustID	Address	City	State	Zip
1001	123 Lexington	Smithville	KY	91232
1002	12 Davis Ave.	Smithville	KY	91232
1003	443 Grinder Ln.	Broadville	GA	81992
1004	661 Parker Rd.	Streetville	GA	81990

- Data may also be *replicated* across multiple nodes:
    1. Improve performance by moving a copy of data closer to the users.
    2. Improve reliability - if one node fails, others can continue processing the transactions.

Degree of heterogeneity/homogeneity

- Distributed databases may be made up of **homogeneous** (similar) nodes or some mix of **heterogeneous** (different) nodes.
    - **Heterogeneity** can be at many different levels:
        1. Hardware heterogeneity - networking, server/host hardware
        2. Operating system heterogeneity
        3. Component DBMS heterogeneity
        4. Query language/DML heterogeneity
        5. Programming language heterogeneity
        6. *Semantic* heterogeneity - different nodes may have different definitions or uses for the same or similar data item.
    - For example, consider a DDS with 5 nodes all running Oracle7 on Sun Sparcstations running Solaris (UNIX). All nodes have the same DB schema.

      Another DDS might have 2 nodes running Oracle7 on Suns, one node running Sybase on Windows NT and a fourth running IBM IMS on a mainframe.

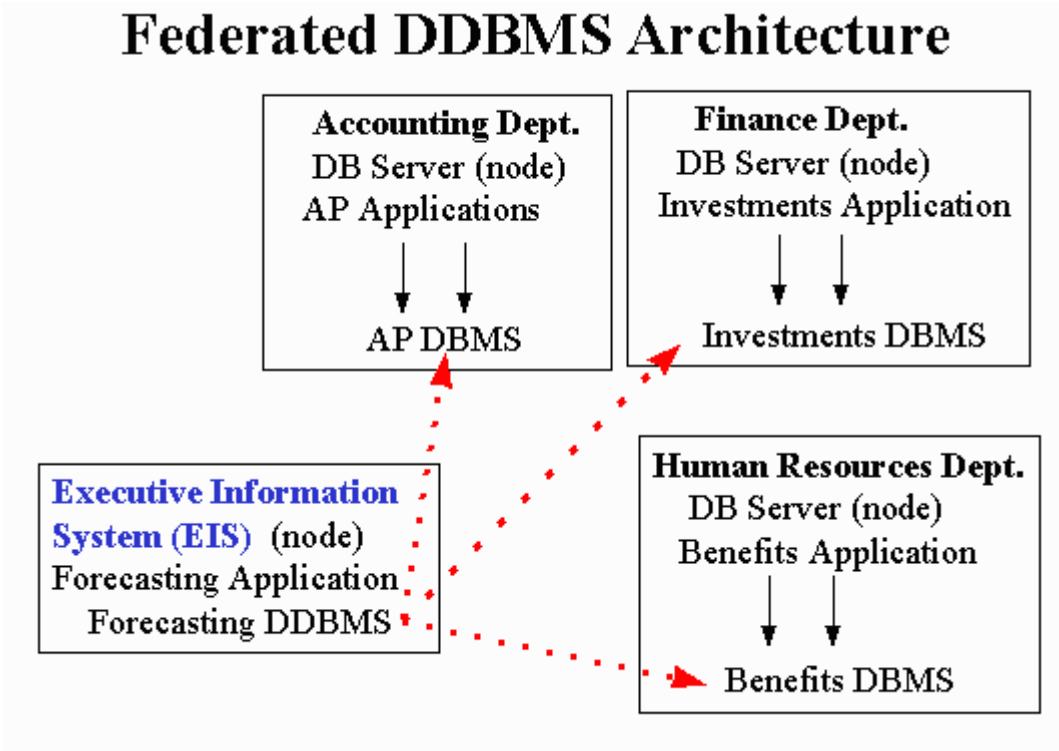
- Assumption to make is that replacing existing systems is not feasible (economically, technically, politically).

Degree of Autonomy

- Each site or node may have some degree of **autonomy** - that is, it can accept transactions locally while still participating as a node in the distributed database system.
    - In a DDS with no local autonomy, all transactions must be submitted to the distributed concurrency controller.

      Thus the entire distributed database might appear as a single "system" to the application programs (and users).

    - It is possible, however, that only certain transactions are submitted to the distributed CC while others are submitted directly to an individual node's DBMS.



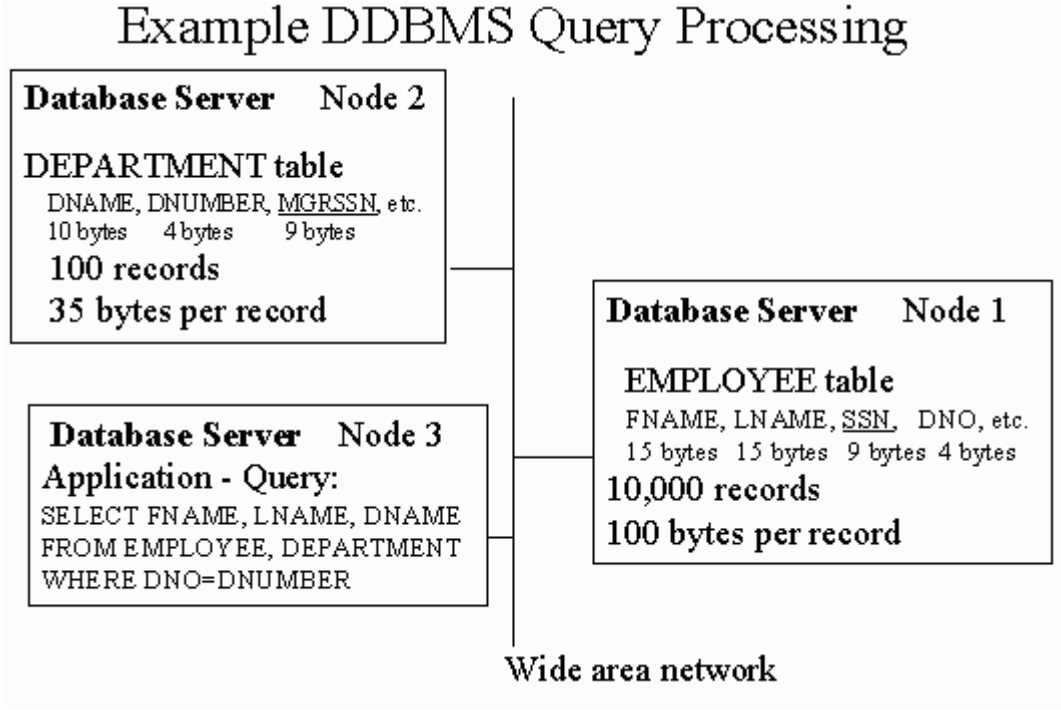
- In the above figure of a **Federated DDBMS**, each department maintains its own local applications and databases. Transactions are submitted directly from the local application to the local DBMS (the solid arrows) indicating local autonomy.
- However, other applications such as the EIS, can run federated transactions that access data from each of the component systems (the dashed lines).

Degree of Distribution Transparency

- The third dimension concerns the **degree of distribution transparency** (schema integration).
  - Through a process called *schema integration*, we can provide a unified view of the DDBMS such that it appears as a single schema to all applications (and users). This would be a high degree of schema integration.
  - If, however, applications and users must specify the specific locations of all data items, then this would be considered a low degree of schema integration.
  - There are many problems encountered with providing such an integrated schema (see notes above on heterogeneity) including the problem of **naming**.
- In general, distributed database systems can offer more flexibility, higher performance and greater levels of independence over centralized systems.
- However, distributed database systems are also much harder to design and develop, control and administrate. Security is also more difficult to enforce.
- For more on how Oracle8 manages distributed databases, look into the Oracle8i Server Distributed Database Systems Release 8i

DDBMS Query Processing and Optimization

- Recall that in a DDBMS data can be fragmented or replicated across several sites.
- Whereas in a centralized DBMS environment we are mostly concerned with Access Costs, in a DDBMS environment, we are primarily concerned with **Network costs** - the cost to ship data from one node to another to satisfy a query (specifically in joins). Cost is given in bytes to be transferred.



- This query will return 10,000 records (every employee belongs to one department). Each record will be 40 bytes long (FNAME + LNAME + DNAME = 15 + 15 + 10 = 40).
  Thus the result set will be 400,000 bytes.
- Assume cost to transfer query text between nodes can be safely ignored.
- Three alternatives:
  1. Copy all EMPLOYEE and DEPARTMENT records to node 3. Perform the join and display the results.
     Total Cost = 1,000,000 + 3,500 = 1,003,500 bytes
  2. Copy all EMPLOYEE records (1,000,000 bytes) from node 1 to node 2. Perform the join, then ship the results (400,000 bytes) to node 3.
     Total cost = 1,000,000 + 400,000 = 1,400,000 bytes
  3. Copy all DEPARTMENT records (3,500) from node 2 to node 1. Perform the join. Ship the results from node 1 to node 3 (400,000).
     Total cost = 3,500 + 400,000 = 403,500 bytes
- Another alternative is to implement a **Semijoin** - ship around only the columns required to perform the join.
- Taking the same example:
  1. Copy just the FNAME, LNAME and DNO columns from node 1 to node 3 (cost = 34 bytes times 10,000 records = 340,000 bytes)

2. Copy just the DNUMBER and DNAME columns from node 2 to node 3 (cost = 14 bytes times 100 records = 1,400 bytes)
3. Perform the join at node 3 and display the results.
   Total cost = 341,400

- Other options with semijoin are to transfer just the columns to be joined around to the sites, then go back and fetch the rest of the columns to be projected in the results.

DDBMS Concurrency Control

- Assume we have replicated our data across three nodes. How can we implement concurrency control ? Where should the locking take place ?
- One copy of each data item (table, row, etc.) is designated as the *Distinguished Copy*.
- This copy of the data item is where all locks are applied. The concurrency controller should look to this copy to determine if any locks are held.
- Several variations of distinguished copy:
  1. All distinguished copies reside on the same node: **Primary Site**
  2. Distinguished copies may reside on different sites: **Primary Copy**
- There are problems with this approach:
  - What if the primary site fails or if the node where the primary copy resides fails?
  - One solution is to designate a *backup* site that will automatically take over.
  - Another solution is to "elect" a new primary site/copy.

Another overall approach (aside from distinguished copy), is to use a *Voting protocol*: To lock a data item:

0. Send a message to all nodes that maintain a replica of this item.
1. If a node can safely lock the item, then vote "Yes", otherwise, vote "No".
2. If a majority of participating nodes vote "Yes" then the lock is granted.
3. Send the results of the vote back out to all participating sites.

With the voting approach, nodes can fail and recover while allowing transaction processing to continue.

DDBMS Transaction Processing

- We need mechanisms in place to ensure multiple copies of data are kept consistent.
- Concurrency and Commit protocols must be changed to account for replicated data.
- Recall in a centralized DB we had the notion of a commit point. In distributed DB, we need to consider committing a transaction that changes data on multiple nodes.
- Distributed Commit Protocol such as Two Phase Commit (2PC). Also called a *synchronous* replication protocol.
  1. **Phase 1**: Send a message to all nodes: "Can you commit Transaction X?"
     All nodes that can commit this transaction reply with "Yes".
  2. **Phase 2**: If all nodes reply with "Yes", then send a "Commit" message to all nodes.
     If any node replies "No", then the transaction is aborted.
- 2PC is an example of a **synchronous** replication protocol.
- In **Asynchronous replication**, we take *snapshots* of a master database and propagate the changes to other nodes on some periodic basis.
- For example, see Oracle8i Server Replication Release 8.1.7