Orange Coast College  
Business Division  
CS/CIS Department  
Fall 2004  
CIS 182  
Introduction to Database Concepts

Instructor  
Dr. Martha Malaty  
Text & Original Presentations  
Chapter 10

Distributed Database Management Systems

In this chapter, you will learn:

• What a distributed database management system (DDBMS) is and what its components are
• How database implementation is affected by different levels of data and process distribution
• How transactions are managed in a distributed database environment
• How database design is affected by the distributed database environment
Distributed Database

• **Webopedia**
  – A database that consists of two or more data files located at different sites on a computer network. Because the database is distributed, different users can access it without interfering with one another. However, the DBMS must periodically synchronize the scattered databases to make sure that they all have consistent data.

• **FOLDOC**:  
  – A collection of several different databases that looks like a single database to the user. An example is the Internet Domain Name System (DNS).
Distributed Database Management System (DDBMS)

- Governs the storage & processing of logically related data over interconnected computer systems
- Data & processing functions are distributed among several sites
- Whatis.com
  - A centralized application that manages a distributed database as if it were all stored on the same computer. The DDBMS synchronizes all the data periodically, and in cases where multiple users must access the same data, ensures that updates and deletes performed on the data at one location will be automatically reflected in the data stored elsewhere.
DDBMS Evolution

• 1970’s:
  – Structured information.
  – Formal reports in standard format
  – 3GL programming languages
  – Centralized DBMS
DDBMS Evolution

• 1980’s:
  – Social and Technical Changes
  – Increased global competition
  – Customer demands in favor of decentralization
  – Many corporations used LAN’s
  – More dynamic business environment
  – Two database requirements:
    • Ad hoc capability required
    • Decentralized management structure
DDBMS Evolution

• 1990’s:
  – New forces
    • Dynamic business environment and centralized database’s shortcomings created a demand for applications based on data access from different sources at multiple locations
  – Internet and the World Wide Web used for data access and distribution
  – Data analysis through data mining and data warehousing
Centralized DBMS

- Corporate data in a centralized site
  - The DBMS and the data reside in one location (Single tier)
- Dumb terminals were used to access the DBMS through teleprocessing
- Disadvantages:
  - Performance degradation as the number of remote locations over long distance increases
  - As data increased, information retrieval became slower
  - High maintenance & operating cost for central mainframes
  - Reliability problems due to dependency on a central site
  - Difficult to get ad-hoc information
Centralized DBMS

• Actions done
  1. Receive application request from end user
  2. Validate, analyze, & decompose request
  3. Map request’s logical-to-physical component
  4. Decompose request into several disk I/O operations
  5. Search for, locate, read, & validate data
  6. Ensure DB consistency, security, & integrity
  7. Validate data for conditions specified by the request, if any
  8. Present request data in required format back to the user

FIGURE 10.1 CENTRALIZED DATABASE MANAGEMENT SYSTEM
Decentralized/Distributed DBMS (DDBMS)

- Can be implemented in many different ways
- Several arrangements (topology)
  - Star
  - Ring
  - Network
- Logically related data
- Interconnected computer systems
- Data & processing functions reside on multiple sites
DDBMS Advantages

• Faster data access
  – Data located near the site that has the greatest demand to match business requirements
  – Data subsets needed are usually locally stored & accessed
• Faster data processing
  – System’s workload spread out over several sites
• Growth facilitation
  – New sites can be added without affecting operation of other sites
• Improved communications
  – Local sites foster better communication among departments
• Reduced operating costs
  – Compared to mainframe costs
• User-friendly interface
  – GUI & simplified user training
• Less danger of single-point failure
  – In case of failure, workload picked up by other workstations
• Processor independence
  – Request independent on specific processor
DDBMS Disadvantages

- Complexity of management and control
  - Application must know data location & combine data from different sites
  - DBA must coordinate DB activities to prevent data anomalies
  - Many problems must be addressed;
    - e.g. transaction management, concurrency control, security, backup, recovery, query optimization, path selection,…

- Security
  - Data management responsibility shared among different people among different sites

- Lack of standards
  - Different vendors employ different techniques to manage data distribution

- Increased storage requirements
  - Multiple copies of data at different sites

- Greater difficulty in managing data environment
  - Disc access & storage is more complex

- Increased training costs
  - More than that of centralized since more people are involved
Types of Distribution

• Distributed Processing
  – Processing is performed over several computers connected via the network
  – Data is centralized

• Distributed Database
  – Data is stored over several computers connected via the network
  – Processing is centralized

• Fully Distributed
  – Both data & processing are distributed
Distributed Processing Environment

- Shares DB’s logical processing among physically, networked independent sites
Distributed Database Environment

- Stores logically related database over physically independent sites
- DB composed of several “DB fragments”

**FIGURE 10.3 DISTRIBUTED DATABASE ENVIRONMENT**

[Diagram showing distributed database environment with three sites: Site 1 in Miami with user Alan, Site 2 in New York with user Betty, Site 3 in Atlanta with user Hernando. Each site has a computer and a database management system (DBMS) with data (E1, E2, E3). There is a communication network connecting the sites.]
Characteristics of Distributed Management Systems

- Application/end user interface
- Validation to analyze data requests
- Transformation to determine request components
- Query optimization to find the best access strategy
- Mapping to determine the data location
- I/O interface to read or write data
- Formatting to prepare the data for presentation
- Security to provide data privacy
- Backup and recovery
- DB Administration
- Concurrency Control
- Transaction Management
Characteristics of Distributed Management Systems (continued)

- Must perform all the functions of a centralized DBMS
- Must handle all necessary functions imposed by the distribution of data and processing
- Must perform these additional functions *transparently* to the end user
A Fully Distributed DBMS

- Perform all centralized DBMS functions + data distribution & processing functions
- Both users see only one logical DB
- Users don’t need to know the names or locations of the fragments

**Figure 10.4** A Fully Distributed Database Management System
DDBMS Components

• Must include (at least) the following components:
  – Computer workstations
  – Network hardware and software
  – Communications media
  – Transaction processor (TP) (or, application processor (AP), or transaction manager (TM))
    • Software component found in each computer that requests data
  – Data processor (DP) or data manager (DM)
    • Software component residing on each computer that stores and retrieves data located at the site
    • May be a centralized DBMS
Distributed Database System Components

Figure 10.5 Distributed Database System Components

Note: Each TP can access data on any DP, and each DP handles all requests for local data from any TP.
DDBMS Protocols

- **FOLDOC & Wikipedia**
- Interface with network to transport data and commands between DPs and TPs
- Synchronize data received from DPs and route to appropriate TPs
- Ensure common database functions
  - Security
  - Concurrency control
  - Backup and recovery
- DP’s & TP’s
  - Can be added to the system without affecting other components
  - Can reside on the same computer
Database Systems Classification

- Database systems can be classified based on process distribution and data distribution
  - Single-site processing, single-site data (SPSD)
  - Multiple-site processing, single-site data (MPSD)
  - Multiple-site processing, Multiple-site data (MPMD)

### TABLE 10.1 Database Systems: Levels of Data and Process Distribution

<table>
<thead>
<tr>
<th>SINGLE-SITE DATA</th>
<th>MULTIPLE-SITE DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-site process</td>
<td>Host DBMS (Mainframes)</td>
</tr>
<tr>
<td></td>
<td>Not applicable</td>
</tr>
<tr>
<td></td>
<td>(Requires multiple processes)</td>
</tr>
<tr>
<td>Multiple-site process</td>
<td>File server</td>
</tr>
<tr>
<td></td>
<td>Fully distributed</td>
</tr>
<tr>
<td></td>
<td>Client/server DBMS (LAN DBMS)</td>
</tr>
<tr>
<td></td>
<td>Client/server DDBMS</td>
</tr>
</tbody>
</table>
SPSD

- Everything is on a single CPU or host computer (mainframe, midrange, or PC)
  - All processing
  - All data
  - DBMS
- Processing cannot be done on end user’s side of the system
- DBMS accessed by dumb terminals and runs under time-sharing multitasking OS
- TP & DP functions embedded within the DBMS & handled by a single CPU
- Typical of mainframe and minicomputer DBMSs
- Typical of 1st generation of single-user microcomputer database
FIGURE 10.6 SINGLE-SITE PROCESSING, SINGLE-SITE DATA (CENTRALIZED)
MPSD

- Multiple processes run on different computers sharing a single data repository
- MPSD scenario requires a network file server running conventional applications that are accessed through a LAN
- Many multi-user accounting applications, running under a personal computer network, fit such a description
- TP on each workstation acts as a redirector routing all data requests to the file server
- Variation known as client/server architecture
FIGURE 10.7  MULTIPLE-SITE PROCESSING, SINGLE-SITE DATA
MPSD

- Example:
  - File server has CUSTOMER table with 10,000 rows
  - 50 rows have balance > $1,000
  - Site A issues query
    ```sql
    SELECT * FROM CUSTOMERS
    WHERE CUST_BALANCE > 1000;
    ```
  - All 10,000 rows must travel through the network to be evaluated at site A

- Disadvantages:
  - Very limited distribution capabilities
  - End user must make direct reference to the file server for accessing data
  - Entire files travel through the network
  - All data selection, search, . . . take place at the end-user workstation
  - Slow response time & high communication cost
MPMD

- Fully distributed database management system with support for multiple data processors and transaction processors at multiple sites
- Classified as either homogeneous or heterogeneous
- Homogeneous DDBMSs
  - Integrate only one type of centralized DBMS over a network
- Heterogeneous DDBMSs
  - Integrate different types of centralized DBMSs over a network
- Fully heterogeneous DDBMS
  - Support different DBMSs that may even support different data models (relational, hierarchical, or network) running under different computer systems, such as mainframes and microcomputers
Heterogeneous Distributed Database Scenario

**FIGURE 10.8 HETEROGENEOUS DISTRIBUTED DATABASE SCENARIO**

<table>
<thead>
<tr>
<th>Platform</th>
<th>DBMS</th>
<th>Operating System</th>
<th>Network Communications Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM 3090</td>
<td>DB2</td>
<td>MVS</td>
<td>APPC LU 6.2</td>
</tr>
<tr>
<td>DEC/VAX</td>
<td>VAX rdb</td>
<td>MVS</td>
<td>DECnet</td>
</tr>
<tr>
<td>IBM AS/400</td>
<td>SQL/400</td>
<td>OS/400</td>
<td>3270</td>
</tr>
<tr>
<td>RISC computer</td>
<td>Informix</td>
<td>UNIX</td>
<td>TCP/IP</td>
</tr>
<tr>
<td>Pentium CPU</td>
<td>Oracle</td>
<td>Windows XP</td>
<td>NetBIOS</td>
</tr>
</tbody>
</table>
Heterogeneous Distributed Database

- Restrictions
  - Remote access is provided on read-only basis
  - No write privilege
  - Restricted number of remote tables that can be accessed in a single transaction
  - Restricted number of databases that may be accessed
  - Restricted database models that may be accessed
    - E.g. can access relational but not network or hierarchical
Distributed Database Transparency Features

- Allow end user to feel like database’s only user
- Features include:
  - Distribution transparency
    - Distributed DB treated as a single logical DB
  - Transaction transparency
    - Update data on several sites & either entirely completed or aborted
  - Failure transparency
    - Continues operation even if a node fails
  - Performance transparency
    - Performs as if it were centralized
  - Heterogeneity transparency
    - Allows integration of several different local database systems under a common global schema
Distribution Transparency

• Allows management of a physically dispersed database as though it were a centralized database

• Three levels of distribution transparency are recognized:
  – Fragmentation transparency
    • Neither fragment names nor location are needed
    • End user doesn’t need to know that data is fragmented
  – Location transparency
    • Only fragment name needed
    • End user must specify Fragment names but not the location of fragments
  – Local mapping transparency
    • End user must specify both fragment name & location
A Summary of Transparency Features

**TABLE 10.2 A SUMMARY OF TRANSPARENCY FEATURES**

<table>
<thead>
<tr>
<th>IF THE SQL STATEMENT REQUIRES:</th>
<th>THEN THE DBMS SUPPORTS</th>
<th>LEVEL OF DISTRIBUTION TRANSPARENCY</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRAGMENT NAME?</td>
<td>LOCATION NAME?</td>
<td>--------------------------</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Local mapping</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>Location transparency</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>Fragmentation transparency</td>
</tr>
</tbody>
</table>
Distribution Transparency

• Example: EMPLOYEE table
  • (EMP_NAME, EMP_DOB, EMP_ADDRESS, EMP_DEPARTMENT, EMP_SALARY)
  – 3 fragments
    • (E1, E2, E3)
    • Distributed over different locations
      – E1 in New York, E2 in Atlanta, E3 in Miami

FIGURE 10.9 FRAGMENT LOCATIONS
Distribution Transparency Examples

• Fragmentation Transparency
SELECT *
FROM EMPLOYEE
WHERE EMP_DOB < '01-JAN-1940';

• Location Transparency: Name the fragment
SELECT *
FROM E1
WHERE EMP_DOB < '01-JAN-1940';
UNION
SELECT *
FROM E2
WHERE EMP_DOB < '01-JAN-1940';
UNION
SELECT *
FROM E3
WHERE EMP_DOB < '01-JAN-1940';

• Local mapping transparency: Name fragment & location
SELECT *
FROM E1 NODE NY
WHERE EMP_DOB < '01-JAN-1940';
UNION
SELECT *
FROM E2 NODE ATL
WHERE EMP_DOB < '01-JAN-1940';
UNION
SELECT *
FROM E3 NODE MIA
WHERE EMP_DOB < '01-JAN-1940';
Transaction Transparency

• Ensures database transactions will maintain distributed database’s integrity and consistency
• Completed only if all involved database sites complete their part of the transaction
• Management mechanisms
  – Remote request
  – Remote transaction
  – Distributed request
  – Distributed transaction
A Remote Request

- Lets a single SQL statement access data to be processed by a single remote database processor.
- SQL request can only reference data on one remote site.

**FIGURE 10.10 A REMOTE REQUEST**

```sql
SELECT * 
FROM CUSTOMER 
WHERE CUS_STATE = 'AL';
```

Comment: The request is directed to the CUSTOMER table at Site B.
A Remote Transaction

- Accesses data at a single remote site
- Transaction = several SQL statements or requests
- Several SQL requests can only reference data on one remote site

**FIGURE 10.11 A REMOTE TRANSACTION**

```
BEGIN WORK;
UPDTE PRODUCT
SET PROD_QTY = PROD_QTY - 1
WHERE PROD_NUM = '231785';
INSERT INTO INVOICE (CUS_NUM, INV_DATE, INV_TOTAL)
VALUES '100', '15-FEB-2004', 120.00;
COMMIT WORK;
```
Distributed Requests

• Several SQL requests
• Each request accesses data from more than one DP site
• Reference one or more fragments with only one request
• Able to partition a DB table into several fragments
• Have fragmentation transparency
• Location & partition of data should be transparent to the user
A Distributed Request

- The same select statement is referencing tables in different locations

**FIGURE 10.13 A DISTRIBUTED REQUEST**

```
BEGIN WORK;
SELECT CUS_NUM, INV_TOTAL
FROM CUSTOMER, INVOICE
WHERE CUS_NUM = ’100’ AND
INVOICE.CUS_NUM = CUSTOMER.CUS_NUM;
COMMIT WORK;
```
A Distributed Transaction

- Reference data on different remote DP sites on a network
- Several SQL requests reference data on several remote sites

**FIGURE 10.12 A DISTRIBUTED TRANSACTION**

```
BEGIN WORK;
UPDATE PRODUCT
    SET PROD_QTY = PROD_QTY - 1
    WHERE PROD_NUM = '231785';
INSERT INTO INVOICE (CUS_NUM, INV_DATE, INV_TOTAL)
    VALUES ('100', '15-FEB-2004', 120.00);
UPDATE CUSTOMER
    SET CUS_BALANCE = CUS_BALANCE + 120
    WHERE CUS_NUM = 'T00';
COMMIT WORK;
```
Distributed Concurrency Control

- Multi-site, multiple-process operations more likely to create data inconsistencies & deadlocked transactions
- Problems: Premature commit
  - If part of the transaction is committed by some of the DP units while other DP units could not commit the transaction’s result. This would yields inconsistent database
- TP component of DDBMS must ensure that all parts of the transaction, at all sites, are completed before a final COMMIT is issued to record the transaction
Premature COMMIT Example

**Figure 10.15 The Effect of a Premature COMMIT**

- **Site A**
  - Data are committed
  - LOCK (X)
  - WRITE (X)
  - COMMIT

- **Site B**
  - DP
  - LOCK (Y)
  - WRITE (Y)
  - COMMIT

- **Site C**
  - DP
  - LOCK (Z)
  - Rollback at site C
  - ... ROLLBACK

Can't roll back sites A and B
Two-Phase Commit Protocol

• Distributed databases make it possible for a transaction to access data at several sites
• Guarantees that, if a portion of a transaction can’t be committed, all changes made at other sites participating in the transaction will be undone to maintain consistency
• Final COMMIT must not be issued until all sites have committed their parts of the transaction
• Two-phase commit protocol requires each individual DP’s transaction log entry be written before the database fragment is actually updated
• See chapter 9 for more details
Two-Phase Commit Protocols

• DO-UNDO-REDO protocol
  – Roll back / or forward transactions using “Transaction log” entries
  – Three types of operations
    • DO
      – Perform the operation & record the “before” & “after” values in transaction log
    • UNDO
      – Reverses an operation using log entries
    • REDO
      – Redoes an operation using log entries
• Write-ahead protocol
  – Forces the log entry to be written to permanent storage before the actual operation takes place
DO-UNDO-REDO protocol

- Two kinds of nodes
  - Coordinator
  - Subordinates
- Two phases
  - Preparation
    1. Coordinator sends message to all subordinates
    2. Subordinates receive the message, write transaction log using write-ahead protocol, & send “Acknowledge” to coordinator
    3. Coordinator confirms all are ready to commit or abort the action
  - Final Commit
    1. Reached if all subordinates commit
    2. Ensures all subordinates have committed or aborted
    3. Coordinator broadcasts COMMIT message to all subordinates & waits for reply
    4. Subordinates receive the message & update DB using the DO protocol
    5. Subordinates reply with COMMITED or NOT COMMITED to coordinator
Performance Transparency and Query Optimization

- Objective of query optimization routine is to minimize total cost associated with the execution of a request.
- Costs associated with a request are a function of the:
  - Access time (I/O) cost
  - Communication cost
  - CPU time cost
- Basis for query optimization algorithms
  - Optimum execution order
  - Minimize communication costs by choosing sites accessed
- Must provide distribution transparency (hide the fact that the data is distributed), as well as replica transparency (DDBMS’s ability to hide the existence of multiple copies of data from the user)
Query optimization techniques
Classifications

- Operation mode:
  - Manual or automatic
- Timing classification:
  - Static or dynamic
- Classification according to information type
  - Statistically based or rule-based algorithms
Query Optimization

• Operation modes:
  – Automatic
    • DDBMS finds the most cost-effective access path without user intervention
  – Manual
    • Optimization selected & scheduled by the user/programmer
Query Optimization

• Timing classification
  – Dynamic optimization
    • At query execution time
    • Access strategy dynamically determined using up-to-date DB information
    • Determined every time the query is executed
  – Static optimization
    • At query compile time
    • Common when SQL statements are embedded in procedural programming languages (e.g. COBOL, Pascal, …)
Query Optimization

• Classification according to information type
  – Statistically based query optimization
    • Provide information about DB Characteristics
      – Size
      – Number of records
      – Average access time
      – Number of requests serviced
      – Number of users with access rights, . . .
  • Can be manual or dynamic
    – Rule-based query optimization
      • Set of user-defined rules to determine the best access strategy
      • Entered by end user or DBA
      • Typically very general in nature
Distributed Database Design

• In addition to the design principles used in centralized DBMS, 3 new issues
  – Partition database into fragments
    • Horizontal
    • Vertical
    • Mixed
  – Fragments to replicate: Storage of data copies at multiple sites
    • Fully
    • Partially
    • Un-replicated
    • Factors: DB size, usage frequency, cost, & performance
  – Data allocation: Where to locate data
    • Centralized
    • Partitioned
    • Replicated
Data Fragmentation

- Breaks single object into two or more segments or fragments
- Each fragment can be stored at any site over a computer network
- Information about data fragmentation is stored in the distributed data catalog (DDC), from which it is accessed by the TP to process user requests
Data Fragmentation Strategies

- **Horizontal fragmentation:**
  - Division of a relation into subsets (fragments) of tuples (rows)
  - Fragments include unique tuples (rows) and have the same columns
  - Equivalent to SELECT-WHERE statement

- **Vertical fragmentation:**
  - Division of a relation into attribute (column) subsets
  - Fragments include unique columns subset, except for the key column, which should be common in all fragments
  - Equivalent to PROJECT operation

- **Mixed fragmentation:**
  - Combination of horizontal and vertical strategies
Fragmentation Example

- **Example: CUSTOMER Table**

  **FIGURE 10.16 A SAMPLE CUSTOMER TABLE**

<table>
<thead>
<tr>
<th>CUS_NUM</th>
<th>CUS_NAME</th>
<th>CUS_ADDRESS</th>
<th>CUS_STATE</th>
<th>CUS_LIMIT</th>
<th>CUS_BAL</th>
<th>CUS_RATING</th>
<th>CUS_DUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Sinex, Inc.</td>
<td>12 Main St.</td>
<td>TN</td>
<td>$3,500.00</td>
<td>$2,700.00</td>
<td>3</td>
<td>$1,245.00</td>
</tr>
<tr>
<td>11</td>
<td>Martin Corp.</td>
<td>321 Sunset Blvd.</td>
<td>FL</td>
<td>$6,000.00</td>
<td>$1,200.00</td>
<td>1</td>
<td>$0.00</td>
</tr>
<tr>
<td>12</td>
<td>Mynux Corp.</td>
<td>910 Eagle St.</td>
<td>TN</td>
<td>$4,000.00</td>
<td>$3,500.00</td>
<td>3</td>
<td>$3,400.00</td>
</tr>
<tr>
<td>13</td>
<td>BTBC, Inc.</td>
<td>Rue du Monde</td>
<td>FL</td>
<td>$6,000.00</td>
<td>$5,890.00</td>
<td>3</td>
<td>$1,090.00</td>
</tr>
<tr>
<td>14</td>
<td>Victory, Inc.</td>
<td>123 Maple St.</td>
<td>FL</td>
<td>$1,200.00</td>
<td>$550.00</td>
<td>1</td>
<td>$0.00</td>
</tr>
<tr>
<td>15</td>
<td>NBCC Corp.</td>
<td>909 High Ave.</td>
<td>GA</td>
<td>$2,000.00</td>
<td>$350.00</td>
<td>2</td>
<td>$50.00</td>
</tr>
</tbody>
</table>

- **Horizontal Fragmentation**

  **TABLE 10.3 HORIZONTAL FRAGMENTATION OF THE CUSTOMER TABLE BY STATE**

<table>
<thead>
<tr>
<th>FRAGMENT NAME</th>
<th>LOCATION</th>
<th>CONDITION</th>
<th>NODE NAME</th>
<th>CUSTOMER NUMBERS</th>
<th>NUMBER OF ROWS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUST_H1</td>
<td>Tennessee</td>
<td>CUS_STATE = ‘TN’</td>
<td>NAS</td>
<td>10, 12</td>
<td>2</td>
</tr>
<tr>
<td>CUST_H2</td>
<td>Georgia</td>
<td>CUS_STATE = ‘GA’</td>
<td>ATL</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>CUST_H3</td>
<td>Florida</td>
<td>CUS_STATE = ‘FL’</td>
<td>TAM</td>
<td>11, 13, 14</td>
<td>3</td>
</tr>
</tbody>
</table>
### Fragmentation Example (Continued)

- Horizontally fragmented table

**Figure 10.17  Table Fragments in Three Locations**

<table>
<thead>
<tr>
<th>Table name: CUST_H1</th>
<th>Location: Tennessee</th>
<th>Node: NAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUS_NUM</td>
<td>CUS_NAME</td>
<td>CUS_ADDRESS</td>
</tr>
<tr>
<td>10</td>
<td>Sinex, Inc.</td>
<td>12 Main St.</td>
</tr>
<tr>
<td>12</td>
<td>Mynux Corp.</td>
<td>910 Eagle St.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table name: CUST_H2</th>
<th>Location: Georgia</th>
<th>Node: ALT</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUS_NUM</td>
<td>CUS_NAME</td>
<td>CUS_ADDRESS</td>
</tr>
<tr>
<td>15</td>
<td>NBCC Corp.</td>
<td>909 High Ave.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table name: CUST_H3</th>
<th>Location: Florida</th>
<th>Node: TAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUS_NUM</td>
<td>CUS_NAME</td>
<td>CUS_ADDRESS</td>
</tr>
<tr>
<td>11</td>
<td>Martin Corp.</td>
<td>321 Sunset Blvd.</td>
</tr>
<tr>
<td>13</td>
<td>BTBC, Inc.</td>
<td>Rue du Monde</td>
</tr>
<tr>
<td>14</td>
<td>Victory, Inc.</td>
<td>123 Maple St.</td>
</tr>
</tbody>
</table>
Fragmentation Example (Continued)

- Vertically fragmented table

**Figure 10.18 Vertically Fragmented Table Contents**

<table>
<thead>
<tr>
<th>CUS_NUM</th>
<th>CUS_NAME</th>
<th>CUS_ADDRESS</th>
<th>CUS_STATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Sinex, Inc.</td>
<td>12 Main St.</td>
<td>TN</td>
</tr>
<tr>
<td>11</td>
<td>Martin Corp.</td>
<td>321 Sunset Blvd.</td>
<td>FL</td>
</tr>
<tr>
<td>12</td>
<td>Mynux Corp.</td>
<td>910 Eagle St.</td>
<td>TN</td>
</tr>
<tr>
<td>13</td>
<td>BTBC, Inc.</td>
<td>Rue du Monde</td>
<td>FL</td>
</tr>
<tr>
<td>14</td>
<td>Victory, Inc.</td>
<td>123 Maple St.</td>
<td>FL</td>
</tr>
<tr>
<td>15</td>
<td>NBCC Corp.</td>
<td>909 High Ave.</td>
<td>GA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CUS_NUM</th>
<th>CUS_LIMIT</th>
<th>CUS_BAL</th>
<th>CUS_RATING</th>
<th>CUS_DUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>$3,500.00</td>
<td>$2,700.00</td>
<td>3</td>
<td>$1,245.00</td>
</tr>
<tr>
<td>11</td>
<td>$6,000.00</td>
<td>$1,200.00</td>
<td>1</td>
<td>$0.00</td>
</tr>
<tr>
<td>12</td>
<td>$4,000.00</td>
<td>$3,500.00</td>
<td>3</td>
<td>$3,400.00</td>
</tr>
<tr>
<td>13</td>
<td>$6,000.00</td>
<td>$5,890.00</td>
<td>3</td>
<td>$1,090.00</td>
</tr>
<tr>
<td>14</td>
<td>$1,200.00</td>
<td>$550.00</td>
<td>1</td>
<td>$0.00</td>
</tr>
<tr>
<td>15</td>
<td>$2,000.00</td>
<td>$350.00</td>
<td>2</td>
<td>$50.00</td>
</tr>
</tbody>
</table>
Fragmentation Example (Continued)

- **Mixed fragmentation**

**TABLE 10.5 Mixed Fragmentation of the CUSTOMER Table**

<table>
<thead>
<tr>
<th>FRAGMENT NAME</th>
<th>LOCATION</th>
<th>HORIZONTAL CRITERIA</th>
<th>NODE NAME</th>
<th>RESULTING ROWS AT SITE</th>
<th>VERTICAL CRITERIA AT EACH FRAGMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUST_M1</td>
<td>TN-Service</td>
<td>CUS_STATE = ‘TN’</td>
<td>NAS-S</td>
<td>10, 12</td>
<td>CUS_NUM, CUS_NAME, CUS_ADDRESS, CUS_STATE</td>
</tr>
<tr>
<td>CUST_M2</td>
<td>TN-Collection</td>
<td>CUS_STATE = ‘TN’</td>
<td>NAS-C</td>
<td>10, 12</td>
<td>CUS_NUM, CUS_LIMIT, CUS_BAL, CUS_RATING, CUS_DUE</td>
</tr>
<tr>
<td>CUST_M3</td>
<td>GA-Service</td>
<td>CUS_STATE = ‘GA’</td>
<td>ATL-S</td>
<td>15</td>
<td>CUS_NUM, CUS_NAME, CUS_ADDRESS, CUS_STATE</td>
</tr>
<tr>
<td>CUST_M4</td>
<td>GA-Collection</td>
<td>CUS_STATE = ‘GA’</td>
<td>ATL-C</td>
<td>15</td>
<td>CUS_NUM, CUS_LIMIT, CUS_BAL, CUS_RATING, CUS_DUE</td>
</tr>
<tr>
<td>CUST_M5</td>
<td>FL-Service</td>
<td>CUS_STATE = ‘FL’</td>
<td>TAM-S</td>
<td>11, 13, 14</td>
<td>CUS_NUM, CUS_NAME, CUS_ADDRESS, CUS_STATE</td>
</tr>
<tr>
<td>CUST_M6</td>
<td>FL-Collection</td>
<td>CUS_STATE = ‘FL’</td>
<td>TAM-C</td>
<td>11, 13, 14</td>
<td>CUS_NUM, CUS_LIMIT, CUS_BAL, CUS_RATING, CUS_DUE</td>
</tr>
</tbody>
</table>
### Fragmentation Example (Continued)

- **Mixed fragmentation**

**Figure 10.19 Table Contents After the Mixed Fragmentation Process**

<table>
<thead>
<tr>
<th>Table name: CUST_M1 Location: TN-Service</th>
<th>Node: NAS-S</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUS.NUM</td>
<td>CUS_NAME</td>
</tr>
<tr>
<td>10</td>
<td>Sinex, Inc.</td>
</tr>
<tr>
<td>12</td>
<td>Mynux Corp.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table name: CUST_M2 Location: TN-Collection</th>
<th>Node: NAS-C</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUS_NUM</td>
<td>CUS_LIMIT</td>
</tr>
<tr>
<td>10</td>
<td>$3,500.00</td>
</tr>
<tr>
<td>12</td>
<td>$4,000.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table name: CUST_M3 Location: GA-Service</th>
<th>Node: ATL-S</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUS_NUM</td>
<td>CUS_NAME</td>
</tr>
<tr>
<td>15</td>
<td>NBOC Corp.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table name: CUST_M4 Location: GA-Collection</th>
<th>Node: ATL-C</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUS_NUM</td>
<td>CUS_LIMIT</td>
</tr>
<tr>
<td>15</td>
<td>$2,000.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table name: CUST_M5 Location: FL-Service</th>
<th>Node: TAM-S</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUS_NUM</td>
<td>CUS_NAME</td>
</tr>
<tr>
<td>10</td>
<td>Martin Corp.</td>
</tr>
<tr>
<td>12</td>
<td>BTBC, Inc.</td>
</tr>
<tr>
<td>14</td>
<td>Victory, Inc.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table name: CUST_M6 Location: FL-Collection</th>
<th>Node: TAM-C</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUS_NUM</td>
<td>CUS_LIMIT</td>
</tr>
<tr>
<td>11</td>
<td>$6,000.00</td>
</tr>
<tr>
<td>13</td>
<td>$6,000.00</td>
</tr>
<tr>
<td>14</td>
<td>$1,200.00</td>
</tr>
</tbody>
</table>
Data Replication

• Storage of data copies at multiple sites served by a computer network
• Fragment copies can be stored at several sites to serve specific information requirements
  – Can enhance data availability and response time
  – Can help to reduce communication and total query costs
Data Replication

FIGURE 10.20 DATA REPLICATION

Site S1

Site S2

Site S3

DP

A 1

A 1

A 2

A 2
Replication Scenarios

- Fully replicated database:
  - Stores multiple copies of each database fragment at multiple sites
  - Can be impractical due to amount of overhead
- Partially replicated database:
  - Stores multiple copies of some database fragments at multiple sites
  - Most DDBMSs are able to handle the partially replicated database well
- Unreplicated database:
  - Stores each database fragment at a single site
  - No duplicate database fragments
  - Dangerous!!
Data Allocation

• Deciding where to locate data
• Allocation strategies:
  – Centralized data allocation
    • Entire database is stored at one site
  – Partitioned data allocation
    • Database is divided into several disjointed parts (fragments) and stored at several sites
  – Replicated data allocation
    • Copies of one or more database fragments are stored at several sites
Client/Server vs. DDBMS

- FOLDOC
- Whatis.com

Way in which computers interact to form a system
- Features a user of resources, or a client, and a provider of resources, or a server
- Can be used to implement a DBMS in which the client is the TP and the server is the DP
Client/Server Advantages

• Less expensive than alternate minicomputer or mainframe solutions
• Allow end user to use microcomputer’s GUI, thereby improving functionality and simplicity
• More people with PC skills than with mainframe skills in the job market
• PC is well established in the workplace
• Numerous data analysis and query tools exist to facilitate interaction with DBMSs available in the PC market
• Considerable cost advantage to offloading applications development from the mainframe to powerful PCs
Client/Server Disadvantages

• Creates a more complex environment, in which different platforms (LANs, operating systems, and so on) are often difficult to manage.
• An increase in the number of users and processing sites often paves the way for security problems.
• Possible to spread data access to a much wider circle of users increases demand for people with broad knowledge of computers and software increases burden of training and cost of maintaining the environment.
C. J. Date’s Twelve Commandments for Distributed Databases

1. Local site independence
2. Central site independence
3. Failure independence
4. Location transparency
5. Fragmentation transparency
6. Replication transparency
7. Distributed query processing
8. Distributed transaction processing
9. Hardware independence
10. Operating system independence
11. Network independence
12. Database independence
Summary

- Distributed database stores logically related data in two or more physically independent sites connected via a computer network
- Database is divided into fragments
- Distributed databases require distributed processing
- Main components of a DDBMS are the transaction processor and the data processor
- Current database systems can be classified by extent to which they support processing and data distribution
- DDBMS characteristics are best described as a set of transparencies
- A transaction is formed by one or more database requests
- A database can be replicated over several different sites on a computer network
- Client/server architecture refers to the way in which two computers interact over a computer network to form a system