DISTRIBUTED DATABASE DESIGN

The design of a distributed database introduces three new issues:

- How to partition the database into fragments.
- Which fragments to replicate.
- Where to locate those fragments and replicas.

DATA FRAGMENTATION

Data fragmentation allows you to break a single object into two or more segments or fragments. The object might be a user’s database, a system database, or a table. 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.

There are three types of data fragmentation strategies:

- **Horizontal fragmentation** refers to the division of a relation into subsets (fragments) of tuples (rows). Each fragment is stored at a different node, and each fragment has unique rows. However, the unique rows all have the same attributes (columns). In short, each fragment represents the equivalent of a SELECT statement, with the WHERE clause on a single attribute.

- **Vertical fragmentation** refers to the division of a relation into attribute (column) subsets. Each subset (fragment) is stored at a different node, and each fragment has unique columns—with the exception of the key column, which is common to all fragments.

- **Mixed fragmentation** refers to a combination of horizontal and vertical strategies. In other words, a table may be divided into several horizontal subsets (rows), each one having a subset of the attributes (columns).

To illustrate the fragmentation strategies, let’s use the CUSTOMER table for the XYZ Company, depicted in Figure bellow.

![CUSTOMER Table](image)
**Horizontal Fragmentation**
Suppose XYZ Company’s corporate management requires information about its customers in all three states, but company locations in each state (TN, FL, and GA) require data regarding local customers only. Based on such requirements, you decide to distribute the data 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>CUST_STATE = ‘TN’</td>
<td>NAS</td>
<td>10, 12</td>
<td>2</td>
</tr>
<tr>
<td>CUST_H2</td>
<td>Georgia</td>
<td>CUST_STATE = ‘GA’</td>
<td>ATL</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>CUST_H3</td>
<td>Florida</td>
<td>CUST_STATE = ‘FL’</td>
<td>TAM</td>
<td>11, 13, 14</td>
<td>3</td>
</tr>
</tbody>
</table>

**Table name: CUST_H1**
Location: Tennessee  
Node: NAS

<table>
<thead>
<tr>
<th>CUS_NUM</th>
<th>CUS_NAME</th>
<th>CUS_ADDRESS</th>
<th>CUST_STATE</th>
<th>CUST_LIMIT</th>
<th>CUST_BAL</th>
<th>CUST_RATING</th>
<th>CUST_DUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Sinex, Inc.</td>
<td>12 Main St.</td>
<td>TN</td>
<td>3500.00</td>
<td>2700.00</td>
<td>3</td>
<td>1245.00</td>
</tr>
<tr>
<td>12</td>
<td>Mynux Corp.</td>
<td>910 Eagle St.</td>
<td>TN</td>
<td>4000.00</td>
<td>3500.00</td>
<td>3</td>
<td>3400.00</td>
</tr>
</tbody>
</table>

**Table name: CUST_H2**
Location: Georgia  
Node: ATL

<table>
<thead>
<tr>
<th>CUS_NUM</th>
<th>CUS_NAME</th>
<th>CUS_ADDRESS</th>
<th>CUST_STATE</th>
<th>CUST_LIMIT</th>
<th>CUST_BAL</th>
<th>CUST_RATING</th>
<th>CUST_DUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>NBCC Corp.</td>
<td>909 High Ave.</td>
<td>GA</td>
<td>2000.00</td>
<td>350.00</td>
<td>2</td>
<td>50.00</td>
</tr>
</tbody>
</table>

**Table name: CUST_H3**
Location: Florida  
Node: TAM

<table>
<thead>
<tr>
<th>CUS_NUM</th>
<th>CUS_NAME</th>
<th>CUS_ADDRESS</th>
<th>CUST_STATE</th>
<th>CUST_LIMIT</th>
<th>CUST_BAL</th>
<th>CUST_RATING</th>
<th>CUST_DUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Martin Corp.</td>
<td>321 Sunset Blvd.</td>
<td>FL</td>
<td>6000.00</td>
<td>1200.00</td>
<td>1</td>
<td>0.00</td>
</tr>
<tr>
<td>13</td>
<td>BTBC, Inc.</td>
<td>Rue du Monde</td>
<td>FL</td>
<td>6000.00</td>
<td>5890.00</td>
<td>3</td>
<td>1090.00</td>
</tr>
<tr>
<td>14</td>
<td>Victory, Inc.</td>
<td>123 Maple St.</td>
<td>FL</td>
<td>1200.00</td>
<td>550.00</td>
<td>1</td>
<td>0.00</td>
</tr>
</tbody>
</table>

**Vertical Fragmentation**
Suppose the company is divided into two departments: the service department and the collections department. Each department is located in a separate building, and each has an interest in only a few of the CUSTOMER table’s attributes.

<table>
<thead>
<tr>
<th>FRAGMENT NAME</th>
<th>LOCATION</th>
<th>NODE NAME</th>
<th>ATTRIBUTE NAMES</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUST_V1</td>
<td>Service Bldg.</td>
<td>SVC</td>
<td>CUS_NUM, CUS_NAME, CUS_ADDRESS, CUST_STATE</td>
</tr>
<tr>
<td>CUST_V2</td>
<td>Collection Bldg.</td>
<td>ARC</td>
<td>CUS_NUM, CUS_LIMIT, CUS_BAL, CUST_RATING, CUST_DUE</td>
</tr>
</tbody>
</table>
Mixed Fragmentation

Mixed fragmentation requires a two-step procedure. First, horizontal fragmentation is introduced for each site based on the location within a state (CUS_STATE). The horizontal fragmentation yields the subsets of customer tuples (horizontal fragments) that are located at each site. Because the departments are located in different buildings, vertical fragmentation is used within each horizontal fragment to divide the attributes, thus meeting each department’s information needs at each sub site. Mixed fragmentation yields the results displayed in Table bellow.
DATA REPLICATION

Data replication refers to the 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.
Replicated data are subject to the mutual consistency rule. The **mutual consistency rule** requires that all copies of data fragments be identical. Therefore, to maintain data consistency among the replicas, the DDBMS must ensure that a database update is performed at all sites where replicas exist.

- replication has some benefits (such as improved data availability, better load distribution, improved data failure-tolerance, and reduced query costs).
- it also imposes additional DDBMS processing overhead—because each data copy must be maintained by the system. Furthermore, because the data are replicated at another site, there are associated storage costs and increased transaction times (as data must be updated at several sites concurrently to comply with the mutual consistency rule).
- To illustrate the replica overhead imposed on a DDBMS. Consider the processes that DDBMS

  - If the database is fragmented, the DDBMS must decompose a query into sub queries to access the appropriate fragments.
  - If the database is replicated, the DDBMS must decide which copy to access. A READ operation selects the nearest copy to satisfy the transaction. A WRITE operation requires that all copies be selected and to satisfy the mutual consistency rule.
  - The TP sends a data request to each selected DP for execution.
  - The DP receives and executes each request and sends the data back to the TP.
  - The TP assembles the DP responses

Three replication scenarios exist:

- A **fully replicated** database stores multiple copies of each database fragment at multiple sites. In this case, all database fragments are replicated. A fully replicated database can be impractical due to the overhead it imposes on the system.
- A **partially replicated** database stores multiple copies of some database fragments at multiple sites. Most DDBMSs are able to handle the partially replicated database well.
- An **unreplicated** database stores each database fragment at a single site. Therefore,
there are no database fragments.

Several factors influence the decision to use data replication:

- Database size. The amount of data replicated will have an impact on the storage requirements and data transmission costs. Replicating large amounts of data requires a window of time and high I bandwidth that could affect other applications.
- Usage frequency. The frequency of data usage determines how frequently the data needs to be updated. Frequently used data needs to be updated more often, for example, than large data sets that are used only every quarter.
- Costs, including those for performance, software overhead, and management associated with synchronizing transactions and their components vs. fault-tolerance benefits that are associated with replicated data.

- Data replication information is stored in the distributed data catalog (DDC), whose contents are used by the TP to decide which copy of a database fragment to access.

DATA ALLOCATION
Data allocation describes the process of deciding where to locate data. Data allocation strategies are as follows:

- With centralized data allocation, the entire database is stored at one site.
- With partitioned data allocation, the database is divided into two or more disjointed parts (fragments) and stored at two or more sites.
- With replicated data allocation, copies of one or more database fragments are stored at several sites.