Designing Distributed Database

Distributed Database Design

Whether the database is centralized or distributed, the design principles and concepts are same. However, 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 and data replication deal with the first two issues and data allocation deals with the third issue.

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.

Data fragmentation strategies, as discussed here, are based at the table level and consist of dividing a table into logical fragments. You will explore three types of data fragmentation strategies: horizontal, vertical, and mixed.

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. This is the equivalent of the PROJECT statement in SQL.

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. The table contains the attributes CUS_NUM, CUS_NAME, CUS_ADDRESS, CUS_STATE, CUS_LIMIT, CUS_BAL, CUS_RATING, and CUS_DUE.

Table name: CUSTOMER **Database name: Ch12_Text**

CUS_NUM	CUS_NAME	CUS_ADDRESS	CUS_STATE	CUS_LIMIT	CUS_BAL	CUS_RATING	CUS_DUE
10	Sinex, Inc.	12 Main St.	TN	3500.00	2700.00	3	1245.00
11	Martin Corp.	321 Sunset Blvd.	FL	6000.00	1200.00	1	0.00
12	Mynux Corp.	910 Eagle St.	TN	4000.00	3500.00	3	3400.00
13	BTBC, Inc.	Rue du Monde	FL	6000.00	5890.00	3	1090.00
14	Victory, Inc.	123 Maple St.	FL	1200.00	550.00	1	0.00
15	NBCC Corp.	909 High Ave.	GA	2000.00	350.00	2	50.00

Horizontal Fragmentation:

Suppose that 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. Therefore, you define the horizontal fragments to conform to the structure shown in the following table.

HORIZONTAL FRAGMENTATION OF THE CUSTOMER TABLE BY STATE

FRAGMENT NAME	LOCATION	CONDITION	NODE NAME	CUSTOMER NUMBERS	NUMBER OF ROWS
CUST_H1	Tennessee	CUS_STATE = 'TN'	NAS	10, 12	2
CUST_H2	Georgia	CUS_STATE = 'GA'	ATL	15	1
CUST_H3	Florida	CUS_STATE = 'FL'	TAM	11, 13, 14	3

Fragment name: CUST_H1 Location: Tennessee Node: NAS

CUS_NUM	CUS_NAME	CUS_ADDRESS	CUS_STATE	CUS_LIMIT	CUS_BAL	CUS_RATING	CUS_DUE
10	Sinex, Inc.	12 Main St.	TN	$3,500.00	$2,700.00	3	$1,245.00
12	Mynux Corp.	910 Eagle St.	TN	$4,000.00	$3,500.00	3	$3,400.00

Fragment name: CUST_H2 Location: Georgia Node: ATL

CUS_NUM	CUS_NAME	CUS_ADDRESS	CUS_STATE	CUS_LIMIT	CUS_BAL	CUS_RATING	CUS_DUE
15	NBCC Corp.	909 High Ave.	GA	$2,000.00	$350.00	2	$50.00

Fragment name: CUST_H3 Location: Florida Node: TAM

CUS_NUM	CUS_NAME	CUS_ADDRESS	CUS_STATE	CUS_LIMIT	CUS_BAL	CUS_RATING	CUS_DUE
11	Martin Corp.	321 Sunset Blvd.	FL	$6,000.00	$1,200.00	1	$0.00
13	BTBC, Inc.	Rue du Monde	FL	$6,000.00	$5,890.00	3	$1,090.00
14	Victory, Inc.	123 Maple St.	FL	$1,200.00	$550.00	1	$0.00

TABLE FRAGMENTS IN THREE LOCATIONS

Vertical Fragmentation

You may also divide the CUSTOMER relation into vertical fragments that are composed of a collection of attributes. For example, suppose that 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. In this case, the fragments are defined as shown in the following table.

VERTICAL FRAGMENTATION OF THE CUSTOMER TABLE

FRAGMENT NAME	LOCATION	NODE NAME	ATTRIBUTE NAMES
CUST_V1	Service Bldg.	SVC	CUS_NUM, CUS_NAME, CUS_ADDRESS, CUS_STATE
CUST_V2	Collection Bldg.	ARC	CUS_NUM, CUS_LIMIT, CUS_BAL, CUS_RATING, CUS_DUE

Fragment name: CUST_V1 Location: Service Bldg. Node: SVC

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

Fragment name: CUST_V2 Location: Collection Bldg. Node: ARC

CUS_NUM	CUS_LIMIT	CUS_BAL	CUS_RATING	CUS_DUE
10	$3,500.00	$2,700.00	3	$1,245.00
11	$6,000.00	$1,200.00	1	$0.00
12	$4,000.00	$3,500.00	3	$3,400.00
13	$6,000.00	$5,890.00	3	$1,090.00
14	$1,200.00	$550.00	1	$0.00
15	$2,000.00	$350.00	2	$50.00

VERTICALLY FRAGMENTED TABLE CONTENTS

Mixed Fragmentation

The XYZ Company's structure requires that the CUSTOMER data be fragmented horizontally to accommodate the various company locations; within the locations, the data must be fragmented vertically to accommodate the two departments (service and collection). In short, the CUSTOMER table requires 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 the following Table.

MIXED FRAGMENTATION OF THE CUSTOMER TABLE

FRAGMENT NAME	LOCATION	HORIZONTAL CRITERIA	NODE NAME	RESULTING ROWS AT SITE	VERTICAL CRITERIA ATTRIBUTES AT EACH FRAGMENT
CUST_M1	TN-Service	CUS_STATE = 'TN'	NAS-S	10, 12	CUS_NUM, CUS_NAME CUS_ADDRESS, CUS_STATE
CUST_M2	TN-Collection	CUS_STATE = 'TN'	NAS-C	10, 12	CUS_NUM, CUS_LIMIT, CUS_BAL, CUS_RATING, CUS_DUE
CUST_M3	GA-Service	CUS_STATE = 'GA'	ATL-S	15	CUS_NUM, CUS_NAME CUS_ADDRESS, CUS_STATE
CUST_M4	GA-Collection	CUS_STATE = 'GA'	ATL-C	15	CUS_NUM, CUS_LIMIT, CUS_BAL, CUS_RATING, CUS_DUE
CUST_M5	FL-Service	CUS_STATE = 'FL'	TAM-S	11, 13, 14	CUS_NUM, CUS_NAME CUS_ADDRESS, CUS_STATE
CUST_M6	FL-Collection	CUS_STATE = 'FL'	TAM-C	11, 13, 14	CUS_NUM, CUS_LIMIT,

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. Because the existence of fragment copies can enhance data availability and response time, data copies can help to reduce communication and total query costs.

Suppose database A is divided into two fragments, A1 and A2. Within a replicated distributed database, the scenario depicted in the following Figure is possible: fragment A1 is stored at sites S1 and S2, while fragment A2 is stored at sites S2 and S3.

DATA REPLICATION

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.

The different replica overheads imposed on DDBMS are as follows.

- 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 updated 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.
- The problem becomes more complex when you consider additional factors such as network topology and communication throughputs.
- Three replication scenarios exist: a database can be fully replicated, partially replicated, or unreplicated.
- 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 amount of 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 duplicate 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 also on the data transmission costs. Replicating large amounts of data requires a window of time and higher network 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 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.
- Data distribution over a computer network is achieved through data partitioning, through data replication, or through a combination of both. Data allocation is closely related to the way a database is divided or fragmented. Most data allocation studies focus on one issue: which data to locate where.

Data allocation algorithms take into consideration a variety of factors, including:

- Performance and data availability goals.
- Size, the number of rows, and the number of relations that an entity maintains with other entities.
- Types of transactions to be applied to the database and the attributes accessed by each of those transactions.

Disconnected operation for mobile users. In some cases, the design might consider the use of loosely disconnected fragments for mobile users, particularly for read-only data that does not require frequent updates and for which the replica update windows (the amount of time available to perform a certain data processing task that cannot be executed concurrently with other tasks) may be longer.