On Dynamic Fragmentation of Distributed Databases Using Partial Replication

DAVID PINTO, GUADALUPE TORRES
Facultad de Ciencias de la Computación
Benemérita Universidad Autónoma de Puebla
Blvd. Valsequillo y 14 Sur, Ciudad Universitaria
MEXICO

Abstract: - This paper addresses the problem of dynamically reallocating data in a partionable distributed database with changing access patterns. Traditionally, fragmentation in distributed databases has been determined by off-line analysis and optimization, however, there are some enterprises having users accessing their databases under changing access patterns. Comisión Federal de Electricidad (CFE) is one of them that has required an approach to dynamic fragmentation, i.e., an algorithm that can reallocate data while database is on-line. We describe the algorithm used in this case and finally, we indicate the current status of implementation.

Key-Words: - Distributed Database Systems, Horizontal Fragmentation, Partial Replication.

1 Introduction

We can think Database Distributed Systems (DDS) as a union of two efforts: databases and computer networks. A DDS is a collection of processes that manage with transparency multiples distributed databases logically related on a computer network.

Data allocation is a critical aspect of distributed database systems: a poorly-designed data allocation can lead to inefficient computation, high access costs, and high network loads [1, 2] whereas a well-designed data allocation can enhance data availability, diminish access time, and minimize overall usage of resources [1, 3]. It is thus very important to provide distributed database systems with an efficient means of achieving effective data allocation.

In order to distribute data in a distributed database, we need to use fragmentation of information. As we explain after, there are three kinds of fragmentation: vertical, horizontal and mixed. In this paper we describe a project that use horizontal fragmentation with partial replication. To determine the place of replicated data, we use user's frequency access.

We also proposed a search technique called *slave-master search* (*SMS*) to reduce the search space to two computers only.

2 Problem Formulation

CFE is an enterprise that provides automatic cashiers so that its users can may their payment. This cashiers are distributed around the city, but at this moment, when a user would check their payment, the system verify into centralized database (regional) to know the status of this client. However, it is very usual that the connection between the cashier and the centralized database is broken, and therefore the client cannot do the payment, specially at weekends.

CFE has distributed their database in regions, assuming that each user should pay near by their address, but, often each user used to pay near by their job or maybe their parent's job. So, it seems that we cannot determine off-line the distribution of database fragments.

Based on the above requirements, it is necessary to design an application that can improve performance of data access.

As we know, every database is build to provide users high availability of data, however, a major cost in executing queries in a distributed database is the data transfer cost incurred in transferring multiple database objects (fragments).

2.1 Fragmentation

The objective of fragmentation is to determine fragments to locate at different sites so as to minimize the total data transfer cost incurred in executing a set of queries.

Three kinds of fragmentation can be applied to a relation in a database: Vertical Fragmentation, Horizontal Fragmentation and Mixed Fragmentation.

2.1.1 Vertical Fragmentation

Vertical fragments are created by dividing a global relation **R** on its attributes by applying the project operator:

$$\mathbf{R}^j = \Pi_{\{\mathbf{Aj}\},\mathbf{key}}\,\mathbf{R},\ where\ 1 \le j \le m \qquad (1)$$

where $\{\mathbf{A_j}\}$ is a set of attributes not in the primary key, upon which the vertical fragment is defined and *m* is the maximum number of fragments.

A vertical fragmentation schema is complete when every attribute in the original global relation can be found in some vertical fragment defined on that relation. Then the reconstruction rule is satisfied by a join on the primary key(s):

$$\forall \mathbf{R}^j \in \{\mathbf{R}^1, \mathbf{R}^2, \ldots, \mathbf{R}^m\} : \mathbf{R} = \blacktriangleright\blacktriangleleft_{\mathbf{key}} \mathbf{R}^j \qquad (2)$$

The disjoint ness rule does not apply in a strict sense to vertical fragmentation as the reconstruction rule can only be satisfied when the primary key is included in each fragment. So excluding the primary key, no data item should occur in more than one vertical fragment [1, 9].

2.1.1 Horizontal Fragmentation

Horizontal fragmentation divides a global relation **R** on its tuples by use of the selection operator:

$$\mathbf{R}^j = \sigma_{\mathbf{Pj}}(\mathbf{R}),\ where\ 1 \le j \le m \qquad (3)$$

where $\mathbf{P_j}$ is the selection condition as a simple predicate and *m* is the maximum number of fragments.

The horizontal fragmentation schema satisfies the completeness rule if the selection predicates are complete. Furthermore, if a horizontal fragmentation schema is complete, the reconstruction rule is satisfied by a union operation over all the fragments:

$$\forall \mathbf{R}^j \in \{\mathbf{R}^1, \mathbf{R}^2, \ldots, \mathbf{R}^m\} : \mathbf{R} = \cup\, \mathbf{R}^j \qquad (4)$$

Finally, disjointness is ensured when the selection predicates defining the fragments are mutually exclusive [1, 9].

Derived horizontal fragmentation occurs when a member relation inherits the horizontal fragmentation of its owner. If the completeness and disjointness rules are satisfied for the owner fragments, they are intrinsically satisfied for the child fragments. The global relation can be reconstructed by the application of the union operator, as for primary horizontal fragmentation [10].

2.1.1 Mixed Fragmentation

A hybrid fragmentation schema is a combination of horizontal and vertical fragments. If the correctness and disjointness rules are satisfied for the comprising fragments, they are implicitly satisfied for the entire hybrid schema. Reconstruction is achieved by applying the reconstruction operators in reverse order of fragment definition. That is, if a global relation underwent horizontal followed by vertical fragmentation, the reconstruction would consist of a join followed by a union [10, 9].

2.2 Replication

Reliability and performance are the two major purposes of data replication.

Data replication allows availability, especially for distributed databases, but it also implies a big problem to keep consistency.

Replication of data can be complete or partial. Complete replication means that the information need to be replicated at any computer into the system, other side, partial replication only use a few computers to replicate the information. In this paper we use partial replication into the application developed.

3 Problem Solution

In order to give a solution to CFE, we developed an application that use horizontal fragmentation with partial replication. The figure 1 shows the structure used for replication in the system developed. Obviously, this solution is punctual, that means that is built for a special database schema. However, the algorithm used for distribution and also to keep consistency can be applied to others applications.

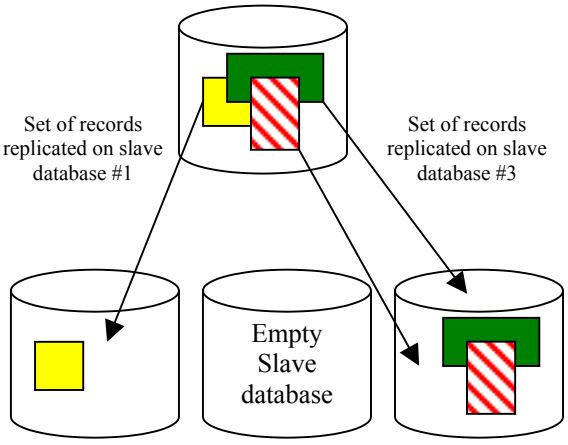

Figure 1. Structure of replication schema.

The algorithm verifies each query and keep a counter to know the place and frequency of user access, if the system detects an access pattern, then it copy a set of records into a slave database. This approach allows improve the performance of queries around database.

Although this solution seems very simple, works fine with the requirements.

It is necessary that every computer having a slave database execute the next algorithm:

RBy2(*bound*) Algorithm

1. For each query request, a slave computer increments a counter (*ctr*) for the user that have made the request.

2. If *ctr* reach ***bound*** number (parameter of this algorithm), then this computer is a candidate to have a set of records replicated and need to follow steps 3 and 4, else step 5.

    3. Request the set of records that the user is asking for and save this information into the slave database.

    4. Reset the user local counter to cero.

5. end.

In order to keep consistency into the distributed database it is necessary that both master and slave computers have the same information, but in this case, since the user will access their own information from the slave computer, it is not necessary to copy the information immediately, but later. This schema of information management allows database availability even when the connection between slave and master database is broken.

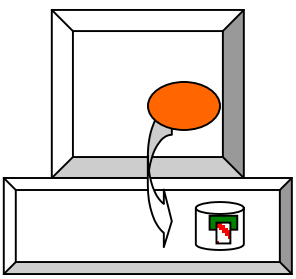

Figure 2. Sense of local database.

An algorithm *SMS* has been designed to provide access to local database before sending the query to the master computer, this algorithm test for the information in the slave database (figure 2), if the records are founded the information is given to the user, otherwise the query is sent to the master database as shown in figure 3.

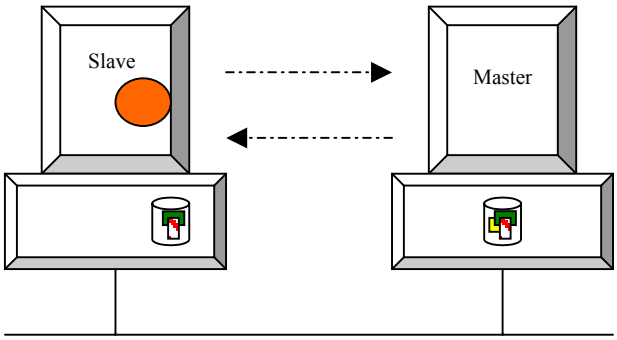

Figure 3. Sending query to master database.

In this case, all the information can be founded at master database, but the system can detect when a user has an access pattern and it copies the information needed to a slave computer (partial replication), so that the user have their information nearest, diminishing the search time.

For the application developed, we used communication with sockets using client-server model with Java language (figure 5).

We designed a basic database schema that is used to manage all the information about users of the company (CFE), this schema is shown at figure 4.

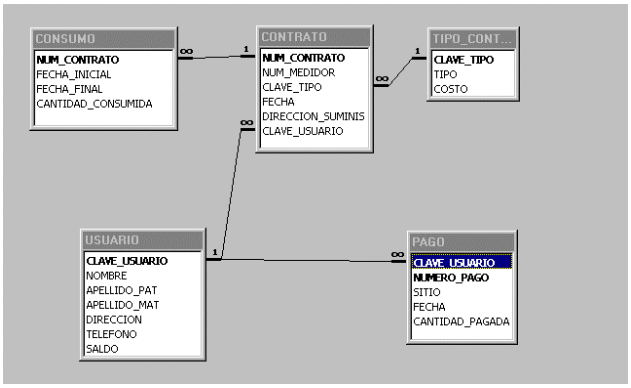

Figure 4. Database schema used for the application.

Of course, the schema shown at figure 4 is only a subschema of real, since the real one have around 50 fields only for user's table.

All probes was made under a local network; however, the real application should be executed

using dial-up connection, therefore, it is very important to have experimental results under this kind of connection. We hope to have those results by march 2002, but just know results data are only for a local computer network.

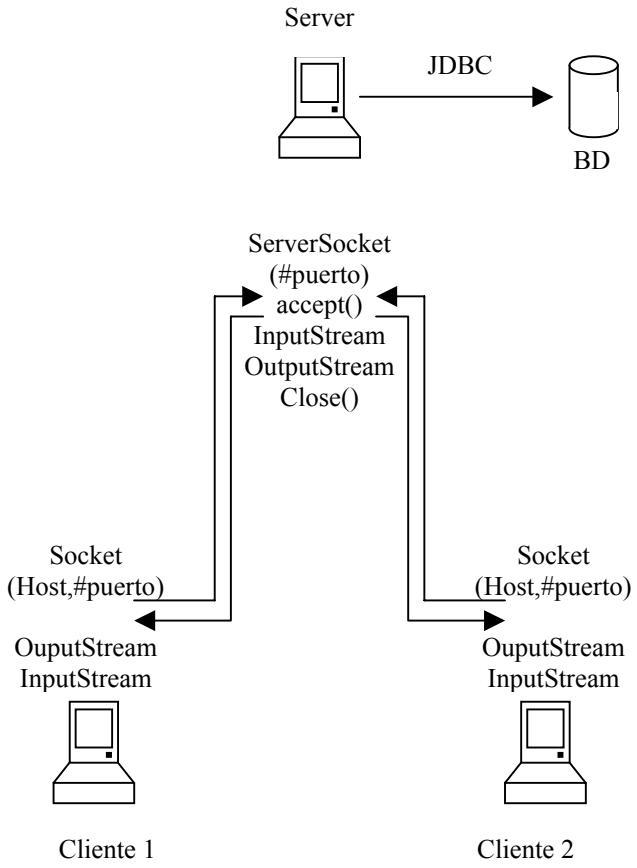

Figure 5. Communication with sockets using TCP/IP protocol.

4 Conclusion

We have presented an algorithm for a punctual solution of CFE. This algorithm can move information between databases, replicating part of data into slaves databases.

The algorithm is based on two techniques: Slave-Master Search, to provide fast access to database on user queries and replication slave-master (two computers) to get availability.

We tested the algorithm in a laboratory of databases with 16 computers, and it seems to be a solution for the problem established by the enterprise, since the results obtained in a local computer network have a good behavior.

Next step would be check how the algorithm works under dial-up connection.

References:

[1] Ozsu M.T., Valduriez P., *Principles of Distributed Database Systems*, Prentice-Hall PTR, 1999.

[2] Sacca D., Wiederhold G., Database partitioning in a cluster of processors, *ACM Transactions on Database Systems*, Vol.10, No.1, 1985, pp. 29-56.

[3] Apers P.M.G., Data Allocation in distributed database systems, *ACM Transactions on Database Systems,* Vol.13, No.3, pp. 263-304, 1988.

[4] Shepherd J.A., Harangsri B., Chen H.L, Ngu A.H.H., A Two-Phase Approach to Data Allocation in Distributed Databases, *Fourth International Conference on Database Systems for Advanced Applications, World Scientific Press*, Singapur, Singapur Abril 1995, 1996.

[5] Navathe S.B., Karlapalem K., Minyoung Ra., A Mixed Fragmentation Methodology for Initial Distributed Database Design, 1997.

[6] Muthuraj J., Chakravarthy S., Varadarajan R., Navathe S.B., A Formal Approach to the Vertical Partioning Problem in Distributed Database Design. PDIS, 1993.

[7] Karlapalem K., Ng Moon Pun, Query-Driven Data Allocation Algorithms for Distributed Database Systems, DEXA 1997.

[8] Brunstrom A., Leutenegger S.T., Simha R., Experimental Evaluation of Dynamic Data Allocation Strategies in a Distributed Database With Changing Workloads, *Fourth International Conference on Information and Knowledge Management*, November 1995.

[9] Elmasri R., Navathe S., *Fundamentals of Database Systems*, Benjamin/Cummings, 1994.

[10] Ceri S., Pelagatti G., *Distributed Databases: Principles and Systems*, McGraw-Hill, 1983.