Implementing a Geographically Distributed Database System with Spatial Information

Alexey Zabrodin
Cybernetics Department
The Moscow Engineering & Physics Institute
Moscow, Russia
e-mail alex@mail.vivos.glasnet.ru

Abstract

A certain range of information systems (e.g. distributed spatial IS) have to maintain sets of views whose contents correlate with the geographic architecture of the system itself. In one of such systems, intended for communication management automation, an approach to local view integration has been used, which is based on distributed system of local servers. It has been implemented at system-specific level, but the approach can be applied to other systems as a general design paradigm. The paper discusses features of the approach, describes some implementation issues and makes approximate estimation of its performance.

1. Introduction

Today's large information systems serve hundreds and thousands of users who can access data simultaneously. In these conditions the problem of providing optimal user access is usually solved by means of data fragmentation [4]. In the relational data model (RDM) there are two general methods of inexcessive fragmentation: horizontal (applying to relations the operation of selection) and vertical (based on the operation of projection). Sometimes, however, there exists a problem of forming appropriate views, which cannot be obtained by means of inexcessive fragmentation only.

The computer-assisted communication management system (CACMS) described in this paper is a distributed client-server system. Its users are located at so-called communication centers and have to work, among other, with spatial data. The informational needs of users at each communication center are similar but require their specific local views. At the same time there must exist a single consistent and easily accessible global view. To provide such functionalities the system architecture maintains views in a distributive manner. This had required a specific replication mechanism that has been implemented at system-specific level, i.e. in the system's serverware.

The section 2 of the paper gives a general notion of the local views; section 3 describes architecture of the CACMS; section 4 contains approximate comparative estimation of system performance; it is followed by conclusions and references.

2. General approach to localized views

In many cases portions of information dealed by different users of a same system intersect with each other. This intersection may require different views of the same informational object.

Consider a system with the following features:

- Users of the system are concentrated at a certain number of mutually remote locations.
- Informational needs of the users are generally uniform throughout the system, but
- At each location users have to work with a corresponding single local view of the same informational object (i.e. their queries have to be based upon this view).

Such a situation is likely to occur in database systems containing spatial information when their distributed architecture correlates with their contents. More precise examples are geoinformation systems for transregional companies (e.g. in transport, communication, power or gas industry).

To illustrate the impact of localized views on data schema let's consider ER-schema for a simple example shown at the figure 1.

Proceedings of the Workshop on Computer Science and Information Technologies CSIT'99
Moscow, Russia, 1999

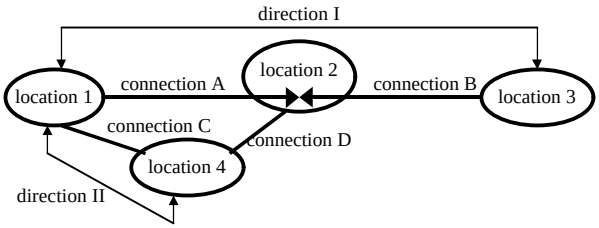

Figure 1. Example of a spatial object

The connections A and B are shown linked together. Corresponding ER-schemas for global and local views (GV and LV) can be as shown at the figures 2a and 2b (relationship cardinalities for each entity are specified in parenthesis).

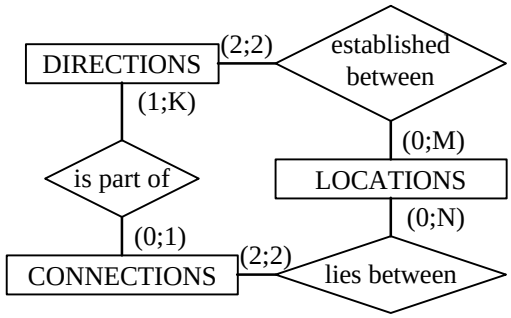

Figure 2a. Example of global ER-schema

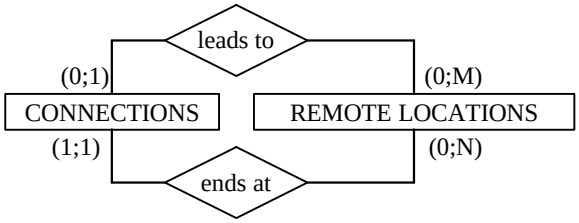

Figure 2b. Example of local ER-schema

Explanation for the relationships at fig. 2b: connection A for loc. 1 leads to loc. 3 but ends at loc. 2.

The LV naturally follows from the assumptions that:

- remote objects, except locations, aren't of interest for users,
- roles that the objects play are viewed from a local viewpoint,
- local users incline to merge the concepts of "connection" and "direction", as result of the previous features.

Developers can either decide to maintain separate local views or to leave the global one as basic schema to perform specific local tasks over it. Obviously, the first way is more preferrable from the viewpoint of portability and scalability of system components. An implementation of a system supporting GV and LV could be made in three ways:

- The GV is stored actually, its LV are stored as derivation rules (derivative approach);
- The GV and its LV are stored actually (distributive approach);
- The LV are stored actually, the GV is stored as a composition rule or not stored, if not required by users (distributive approach).

The latter seems to be the hardest to maintain if absolute data consistency is required. The distributive approach (2^{nd}) implies that users as well as end-user developers would only have to deal with local information views stored separately on their remote local sites. This approach has to be based on replication and translation mechanisms. It can be attractive for implementation for a system with the features mentioned in the beginning of the section if retrieval requests from local sites predominate over modifications (see section 4 for a comparative estimation of performance for a protocol proposed). A number of problems, however, are encountered while implementing such a system:

1. Every local client modification request must be properly translated for GV.
2. Modification mechanisms affecting shared information should support distributed locking transactions to provide consistent replication.
3. Network breaks can be "troublesome" because of the distributed locking.
4. Some nonrelational database features (e.g. unique identification of records "by instance", usually made with help of autoincrement counters) are harder to maintain.

3. Implementing the CACMS database

The database of CACMS contains information about communication system elements. Those elements represent various technical and organizational means of communication that are spatially distributed. For example, the analog to "location" entity in the sect. 2 is "communication center", "communication line" replaces "connection" and serves as resource to establish "communication channels" as "directions". Thus, the features of this database system are as follows:

- Being distributed, the system deals with spatial information, i.e. local views are desired.
- Maximum logical independence for LV applications from GV semantics is needed.
- A large part of information shared by LV is of rather static character (see sect. 2).

The corresponding database system is a distributed client-server system as shown at fig. 3. It consists of a single GV server and a number of local servers connected together with WAN whose capacity is generally less than that of LAN. The servers allow clustering. Every client can also connect to the global server directly. Thus the distinctive feature of the

architecture is its three-layer "client-server-server" structure (since local view servers can be treated as "clients" of the global view server).

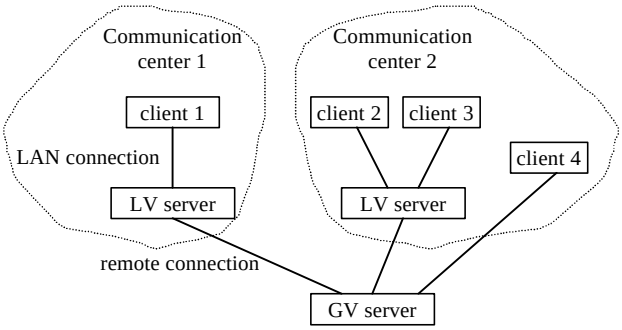

Figure 3. Simplified scheme of the CACMS database system architecture

The problem of translation is managed by means of restricted request sets. The users of the CACMS are only permitted to perform a fixed finite set of modifications upon the shared information. Specific local information, however, allows any kinds of SQL modifications, but using cursors for this purpose isn't allowed. Modification requests are of four general types:

1. Local restricted:

- Client's request to local server LSn contains a descriptor of known type with parameters.
- "Client" transaction is started by LSn.
  - LSn forms a corresponding descriptor for GV and sends it to the global server (GS).
  - Global transaction on the entire system is started by the GS.
    - The GS executes the request.
    - The GS replicates changes to local servers (which are at communication centers) except LSn.
    - The GS reports to LSn on finishing.
    - LSn executes the request locally.
    - LSn reports on finishing to the GS.
  - The GS stops the global transaction.
- LSn stops the "client" transaction and returns the result code.

2. Local common:

- Client's request to local server LSn contains an SQL statement.
- "Client" transaction is started by LSn.
  - LSn checks the attribute scope of the query for intersection with the globally shared scope, which is known to LSn.
  - If no intersection is found, the request is executed locally.
  - If there is any intersection, the request fails.
- LSn stops the "client" transaction and return the result code.

3. Global restricted:

- A request to the GS contains a descriptor of known type with parameters.
- "Client" and global transactions are started by the GS.
  - The GS executes the request and, if successful, replicates the changes to local sites.
- The GS stops the transactions and returns the result code.

4. Global common:

- A request to the GS contains an SQL statement.
- "Client" and global transactions are started by the GS.
  - The GS executes the request and, if successful, replicates the changes to local sites.
- The GS stops the transactions and returns the result code.

Due to stored local views, retrieval requests can be performed locally, not running additional queries or replication processes. Taking into account the static character of most of the information, it raises system performance significantly.

The replication mechanisms are based on subscription. A subject of subscription request is an attribute scope of interest from the global view. The local servers store the necessary global scopes to subscribe for replication and the local scopes related to shared information to allow for correct execution of local common requests. In the current version of the system only entire tables can be replicated.

The current version of the database system uses MDB files as data repositories for both global and local databases, that are accessed by means of ODBC 2.0 [1] and DAO 3.5. Local MDB files have tables attached to the global database, making it easy to download derived data, working within a single database. The database servers are developed with Microsoft Visual C++ 5.0. Servers and clients communicate by the TCP/IP and UDP protocols.

4. Performance comparison

As it was mentioned in the section 2, an alternative approach to system architecture for this case can be based on derivation of local views. A comparative estimation of performance for these two methods has been made from the point of view of network transfer and computational costs. Taking into account the system architecture shown at fig.3, consider three general operations over the DB:

1. "read" operation (data retieval)
2. "add" operation (adding data into the DB)
3. "modify" operation (updating or deleting the data).

Supposing also that the derivative approach implies that additional derivation queries must be run at the global server before execution of client requests, we can finally obtain the following formulae showing advantages of the implemented method over the alternative one:

$$Cost^{read}_{deriv} - Cost^{read}_{distrib} = C_{dq} + (F_{rem} - F_{loc})D_{reply} > 0$$ (since $F_{rem} \geq F_{loc}$).

$$Cost^{add}_{deriv} - Cost^{add}_{distrib} = (1 + W - T - N)C_{dq} -$$
$$- C_{insert} - F_{loc}D_{insert} - NF_{rem}D_{rpl} < 0 (or\ rarely \geq 0)$$

$$Cost^{modify}_{deriv} - Cost^{modify}_{distrib} = (1 + W - T - N)C_{dq} -$$
$$- C_{modify} - NF_{rem}D_{rpl} < 0 (or\ rarely \geq 0)$$

where

C - amount of computations to be executed, in cost units.

D - amount of data transferred through network.

F - network transfer cost factor, transforming D to cost (depends on network capacity).

T - average request translation cost factor. $T > 0$.

W - average "select" modification cost factor (see assumption 3 below). $W > 1$.

N - number of LV. $N \geq 2$.

dq - value belonging to derivartion query;

reply - value belonging to final reply on completion of operation;

rpl - value belonging to replication data;

rem - remote transfer factor;

loc - local transfer factor;

with the following underlying assumptions:

- the views are not distributed themselves.
- modifications upon results of a "select" query are always possible and affect the DB properly,
- the select-query-based component of cost of the modifications is proportionate to the C value (see designations) of the "select" query,
- request translation costs are proportionate to C_{dq} (see designations),
- all the local views are equivalent from the viewpoint of costs,
- transactions are always successful,
- some values like transfer costs for language constructions, protocol information, etc. are neglectible.

The immediate inference was already expected and shows that the distributive approach has performance which is always better in "read" operations and generally worse (but still possibly better) in "add" and "modify" operations. Specific estimations can be obtained by substitution of real values for all parameters.

5. Conclusion

The computer-assisted communication management system (CACMS) created to control the technical condition of communication systems is a distributed information system dealing with spatial information and needs localization of user views. An approach to maintaining local views has been developed within the system, which is based on three-layer client-server structure with one global and a number of separate local view servers. The main benefits of such architecture are essential facilitation of application design process and better performance since retrieval tasks predominate in the system lifecycle. Its shortcomings are slower performance in data modifications and considerable restrictions to possibilities of modification requests. The implementation is not yet completed in what is concerned with optimization of replication and distributed locking, but the mechanism in the whole has shown satisfactory results.

References

1. Signore R., Stegman M. "The ODBC solution. Open database connectivity in distributed environments". Binom, Moscow, 1995
2. Teorey T., Fry J. "Design of database structures". Nauka, Moscow, 1985
3. Codd E. "The relational model for database management: version 2". Addison-Wesley, New York, 1990
4. Date K. "Introduction to database systems". Dialectica, Kiev, 1998