Development of An Autonomous Heterogeneous Distributed Database System: DHIM

Haeng Rae Cho, Yoo Sung Kim, and Songchun Moon

Department of Information and Communication Engineering
Korea Advanced Institute of Science and Technology
207-43, Cheongryang, Dongdaemun, Seoul 130-012, Republic of Korea

In this paper, we design and implement a prototypical heterogeneous distributed database system, named DHIM (Distributed Heterogeneous Information Management). The objectives of DHIM are two-fold: one is to realize the global update synchronization mechanism which preserve the global serializability of concurrent executions and the local site autonomy in tightly coupled heterogeneous distributed database systems, and the other is to implement an open architectural distributed database system. The open architecture implies that DHIM is able to operate independently from the features of element local database systems. Therefore, any database system can be federated into DHIM as an element database system.

Keywords: heterogeneous distributed database system, site autonomy, update synchronization

1. INTRODUCTION

For the past two decades, a number of heterogeneous distributed database systems (HDDSs), such as MULTIBASE [9], Proteus [1], ADDS [2], and DATAFLEX [3], have been implemented. However, none of them provides the global update operations, automatic updating of data stored at a number of local database systems, in the presence of the heterogeneous local concurrency control schemes (LCCSs).

We designed and implemented an autonomous HDDBS, named distributed heterogeneous information management (DHIM) in which heterogeneity among LCCSs exists and 100 percent of site autonomy is guaranteed. The system allows both the two-phase locking scheme and the timestamp ordering scheme for candidates of LCCSs. What DHIM aims is to implement an open HDDBS that provides a global update synchronization mechanism. The open architecture means that DHIM operates independently from local database systems (LDBSs). Therefore, any database system can be federated under DHIM as an LDBS. DHIM also provides distribution transparency to assist users access data without considering the data distribution in LDBSs.

2. RELATED WORKS

The major aim of previously developed systems is to resolve the heterogeneities of local data models and local database languages. Therefore, those systems, except DATAFLEX, do not provide global update operations. That is, updating data stored at different LDBSs is either impossible or allowed to execute only in a restricted manner as off-line updates. All of the systems including DATAFLEX do not deal with the heterogeneity of LCCSs. For example, DATAFLEX restricts the local concurrency control only to the strict two-phase locking scheme.

The conventional HDDBSs can be classified into three groups on the basis of the functionality and the restriction. The first group provides only global retrieval operation but not global update operations. MULTIBASE and Proteus are included in this group. The second group allows the global update operation only in the restricted manners. This group includes IMDAS [8] and PRECI* [4]. The last group provides the global update operation without considering the heterogeneity between LCCSs. DATAFLEX is included in this group.

Our DHIM does not belong to none of the three groups. DHIM is unique in the sense that it allows
global retrieval operations and global update operations even in the case where different local concurrency control schemes are used in different LDBSs.

3. SYSTEM ARCHITECTURE OF DHIM

DHIM is implemented on top of an LDBS called information management (IM) which is the first-ever relational DBMS developed in Korea by KAIST [10]. To generate a heterogeneous environment between LCCSs, we implement two versions of IM, each of which uses two-phase locking scheme (2PL) or timestamp ordering scheme (TSO). As a query language, DHIM uses SQL for the uniform user interface since SQL is the ISO standard. To improve the extensibility of DHIM, the communication mechanism between LDBSs is implemented by using ISO standard protocol, such as the remote database access [5] and the distributed transaction processing [6]. Figure 1 depicts the system architecture of DHIM.

User application process (UAP) is an interface between users and DHIM. Users access the database by the unit of a transaction. A transaction is submitted to the UAP and it is decomposed into a set of subtransactions according to the location map information of the distributed catalog in the local site. The UAP submits the set of subtransactions to the distributed transaction process (DTP). The DTP transfers each subtransaction to the site which stores data required by the subtransaction. In that site, a new DTP is generated to execute the subtransaction. Here, the DTP that sends the subtransaction becomes the coordinator process and the recipient DTPs become participant processes. The coordinator DTP and participant DTPs terminate the execution of a global transaction in an atomic way according to the two-phase commitment (2PC) protocol. To certify whether a subtransaction does not violate the global serializability at participant sites, each participant DTP sends the certification request for the subtransaction to the global transaction scheduler (GTSCH). To resolve the heterogeneity between LCCSs while guaranteeing both global serializability and complete local site autonomy, the unified concurrency control scheme [7] is implemented. The coordination of an incoming transaction by the unified scheme is described in more details in the next section. Only the subtransaction which succeeds in its certification is transmitted to and executed at the underlying LDBS. Finally, the execution results of each subtransaction are returned to the coordinator DTP and are merged to generate an SQL result.

4. GLOBAL CONCURRENCY CONTROL SCHEME

Since the GTSCH is implemented as a preprocessor on the top of LDBSs and the communication between the DTP and the GTSCH is performed by
the unit of an SQL statement, the physical address of 
data object accessed by transactions cannot be 
known at the GTSCH. Therefore, as the granularity 
of global concurrency control, we use the predicates 
of an SQL statement rather than the physical address 
of data object, like page ID and record ID. A predi-
cate specifies the condition of data to be accessed. 
Among SQL statements, SELECT statement is 
considered as the read operation of the transaction 
model. Other statements, like INSERT, DELETE, 
and UPDATE are considered as the write operations.

The GTSCH maintains two levels of information for 
concurrency control, namely, relation level and 
operation level. As relation-level information, the 
largest timestamps of committed transactions which 
had read and written the relation are maintained. 
This information is used to check whether an 
incoming transaction obeys the timestamp ordering 
rule against committed transactions. As operation-
level information, the timestamp of the transaction 
and the operation types, like read or write, are 
maintained. These kinds of information are used to 
check whether the incoming transaction obeys the 
timestamp ordering rule or two-phase locking rule 
against active transactions. Operation-level 
information of a transaction is upgraded to the 
relation-level when the transaction is committed.

In the unified concurrency control scheme, a 
transaction is scheduled according to the type and the 
timestamp of the transaction. When the GTSCH 
receives a transaction which wants to access a 
relation, the GTSCH checks whether the transaction 
is allowed to be executed without violating global 
serializability, using relation-level information and 
operation-level information. This validation is 
performed by following steps.

Step 1) If the type of incoming transaction coincides 
with the type of transactions that the LCCS 
handles, the transaction is validated only on the 
basis of the scheduling rule of the LCCS as the 
following two substeps.

Step 1.1) If the type of LCCS is of timestamp 
ordering, the incoming transaction is validated on 
the basis of the timestamp ordering rule with 
relation-level information in order to guarantee 
that the transaction does not violate the rule 
against committed transactions. The transaction 
which succeeds in the above validation is further 
validated on the basis of the timestamp ordering 
rule with operation-level information in order to 
guarantee that the transaction does not violate the 
rule against active transactions.

Step 1.2) If the type of LCCS is of locking, the 
incoming transaction is validated on the basis of 
the two-phase locking rule with the wait-die 
protocol by using only operation-level 
information in order to guarantee that the 
transaction is not involved in any conflict or any 
deadlock with the other active transactions.

Step 2) If the type of incoming transaction is 
different from that of LCCS, the transaction is 
certified on the basis of both the timestamp 
ordering rule and the two-phase locking rule with 
the wait-die protocol. This certification is 
accomplished by performing both 1.1 and 1.2 
above.

5. EXPERIMENTAL RESULTS

In this section, we describe some of the tests that 
have been completed along with what we have 
learned from these tests. These tests include three 
types of queries: local query (LQ), global query with 
local access only (GQLA), and global query with 
remote access (GQRA). This set of tests enables us 
to investigate the interprocess communication 
overhead and the distribution overhead of DHIM. 
Table 1 shows the results of these tests with two 
sample relations, each of which has 20 tuples and 
200 tuples, respectively.

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Query Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query</td>
<td>relation1 (20 tuples)</td>
</tr>
<tr>
<td>LQ</td>
<td>0.57s</td>
</tr>
<tr>
<td>GQLA</td>
<td>2.16s</td>
</tr>
<tr>
<td>GQRA</td>
<td>2.27s</td>
</tr>
</tbody>
</table>
Since DHIM supports global query only, comparing the performance of GQLA to that of LQ executed at local DBMS (IM) enables us to investigate the interprocess communication overhead of DHIM. Table 1 shows that the execution time of GQLA is much longer than that of LQ. It results from the similarities between user application process (UAP) of DHIM and query processor (QP) of IM. This implies that the facilities for query processing are duplicated on UAP and QP, and additional messages should be transferred between them. One possible solution to this disadvantage is to eliminate UAP and to extend the role of QP such that QP can execute global queries. Since QP executes all types of queries, query processing can be performed efficiently and message overhead can be reduced. This modified system architecture, however, suffers from poor extensibility, because the system architecture is strongly dependent on the local DBMS, IM. Note that one of the objectives of DHIM is to implement an open architectural distributed database system. This requires that the system architecture of DHIM should be independent of local DBMS, even though there may be some performance degradation. As a result, we are currently developing an efficient implementation technique to reduce the performance degradation, while preserving the system architecture of DHIM.

By comparing the performance of GQLA to that of GQRA, we can investigate the distribution overhead of DHIM. This test was performed on two SUN SPARC2 workstations connected through Ethernet with 10Mbps data transfer rate. Table 1 shows that the execution time of GQRA is a little longer than that of GQLA. This implies that networking overhead is the only overhead for GQRA compared with GQLA, and the networking overhead can be reduced when the data transfer rate of the network is high.

6. CONCLUSIONS

We design and implement an autonomous HDDBS, named DHIM, in which the unified scheme is implemented to preserve the global serializability and at the same time to guarantee full local site autonomy. DHIM provides the update synchronization mechanism which can operate in the presence of heterogeneity between LCCSs. DHIM has been in operation on SUN SPARC2 workstations connected through Ethernet protocols since February 1992. To include more functionality in DHIM, we are now in the stage of including deadlock resolution scheme and recovery scheme.

REFERENCES

5. ISO/CD 9579, "Information Processing Systems - Open System Interconnection - Remote Database Access".
6. ISO/DIS 10026, "Information Processing Systems - Open System Interconnection - Distributed Transaction Processing".