Research Journal of Engineering Sciences ISSN 2278 – 9472
Vol. **1(2),** 70-73, August **(2012)** *Res. J. Engineering Sci.*

Short Communication

Concurrency Control and Security issues of Distributed Databases Transaction

Gupta V.K., Sheetlani Jitendra, Gupta Dhiraj and Shukla Brahma Datta
NIMS University, Jaipur, Rajasthan, INDIA

Available online at: www.isca.in
Received 13th August 2012, revised 18th August 2012, accepted 22nd August 2012

Abstract

This paper reviews the coverage of concurrency control and security in distributed Network. As distributed networks become more popular, the need for improvement in distributed database management systems becomes even more important. The most important of these factors of distributed database are single level and multilevel access controls, protection against inference, and maintenance of integrity. The review shows that many concurrency issues and many security threats. In this paper we survey, consolidate, and present the state of the art in distributed database concurrency control. The heart of our analysts is a different factor of concurrency control and security. This paper will examine the underlying features of the distributed database Management system. Learning the task of distributed database management system will lead us to a successful design. The design will improve scalability, accessibility and flexibility while accessing various types of data. We further propose solutions for some of the security concerns that pertain to a distributed database system.

Keywords: Concurrency control, distributed database management systems, transaction, query optimization, locking, security, scalability, accessibility, architecture, component.

Introduction

Today's business environment has an increasing need for distributed database and client/server applications as the desire for reliable, scalable and accessible information is steadily rising. Distributed database systems provide an improvement on communication and data processing due to its data distribution throughout different network sites. Not only is data access faster, but a single-point of failure is less likely to occur, and it provides local control of data for users. However, there is some complexity when attempting to manage and control distributed database systems.

In this paper we describe distributed database system and their issues. A major issue of distributed database is concurrency control problem and security. So we are also describing concurrency control problem and different concurrency control technique.

Overview on Distributed database Systems: In a distributed database system[1] the database is stored/spread physically across computers or sites in different locations that are connected together by some form of data communication network. They may be spread over WAN or LAN. The computers may be of different types such as IBM Mainframes, VAXs, SUN work station, PCs etc managed by different operating systems and each fragment of the data base may be managed by a different DBMS such as Oracle, Ingress, and Microsoft SOL server.

Distributed database management system (DDBMS) In a DDS, database applications running at any of the system's sites should be able to operate on any of the database fragments transparently i.e., as if the data come from a single database managed by one DBMS. The software that manages a distributed database in such a way is called DDBMS.

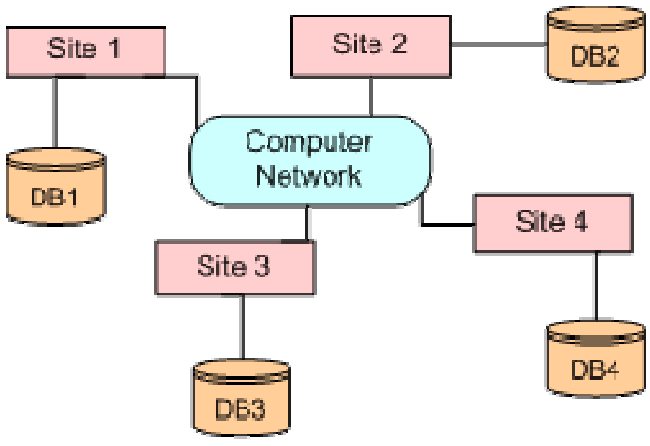

Figure-1
Distributed Database

Distributed database design: The methodology used for the logical design of a centralized database applies to the design of the distributed one as well. However, for a distributed database three additional factors have to be considered.

Data Fragmentation[2]: Before we decide how to distribute the data we must determine the logical units of distribution. The database may be broken up into logical units called fragments which will be stored at different sites. The simplest logical units are the tables themselves.

Horizontal fragmentation: A horizontal fragment of a table is a subset of rows in it. So horizontal fragmentation divides a table 'horizontally' by selecting the relevant rows and these fragments can be assigned to different sides in the distributed system (for ex. Euston Road branch gets the fragment where myTable.branch ='Euston Road').

Vertical fragmentation: a vertical fragment of a table keeps only certain attributes of it. It divides a table vertically by columns. It is necessary to include the primary key of the table in each vertical fragment so that the full table can be reconstructed if needed.

Mixed fragmentation: in a mixed fragmentation each fragment can be specified by a SELECT-PROJECT combination of operations. In this case the original table can be reconstructed be applying union and natural join operations in the appropriate order.

Data Replication: A copy of each fragment can be maintained at several sites. Data replication is the design process of deciding which fragments will be replicated.

Data Allocation: Each fragment has to be allocated to one or more sites, where it'll be stored. There are three strategies regarding the allocation of data:

Fragmented (or Partitioned): The database is partitioned into disjoint fragments, with each fragment assigned to one site (no replication). This is also called 'non-redundant allocation'.

Complete replication: A complete copy of the database is maintained at each site (no fragmentation). Here, storage costs and communication costs for updates are most expensive. To overcome some of these problems, snapshots are sometimes used. A snapshot is a copy of the data at a given time. Copies are updated periodically.

Selective replication: A combination of fragmentation and replication.

Material and Methods

Concurrency Control Technique for distributed database: Concurrency control (CC)[3] is an integral part of a database system, and is the activity of coordinating the actions of transactions that operate in parallel, access shared data, and potentially interfere with one another. Concurrency control has been actively investigated for the past several years, and the problem for nondistributed DBMSs is well understood. We now review the concurrency control technique with respect to distributed database.

Concurrency control is the database management activity of coordinating the actions of database manipulating process that separate concurrently that access shared data and can potentially interfere with one another. The main issue of concurrency control is to ensure the serialisability of concurrently executed transactions. Whenever transactions are concurrently executed, read and write-operations to the same data item may cause conflicts among these transactions. To deal with those conflicts, a concurrency protocol has two basic options: it can try to avoid them or to detect and eliminate them once they have occurred.

Distributed Concurrency control: Concurrency control with respect to distributed DBMS, there is often an additional layer of information processing involved besides the local structures of the singular DB systems (local level), concurrency must also be ensured among those systems (global level). This extra layer causes an increase in complexity as well as communication costs. Therefore, concurrency algorithms used in singular centralised DBMS can not necessarily be transferred to DDBMS without further modification.

Distributed Concurrency control Techniques Locking: Locking[4] is a widely used method to allow concurrent transactions. A transaction may only access a piece of data if the appropriate lock can be obtained. The locking mechanism itself can be easily applied to DDBMSs. Locking can occur at various levels; this is referred to as multi-granularity locking. The two extremes are to either lock an entire database for each data access, or to lock each data record or even data field that is to be read or written. It is obvious that both levels will be inappropriate for almost all applications. Distributed DBMSs introduce an additional level compared to singular DBMSs — above the single DBMSs exists a global level which could theoretically be used for locking. However, this would prove even more useless than locking at database level.

Timestamp Ordering: Timestamp ordering is based on the principle that a transaction's operation on a data item is only executed if its timestamp is newer than the timestamps of all transactions[5] that have previously accessed the data item. The schedules produced by timestamp methods are serialisable. Timestamps must be drawn from a totally ordered domain — this is usually achieved by using tuples composed of a time value as primary ordering field and the transaction manager's unique number as secondary field. Whenever a transaction is aborted and restarted, it is assigned a new (and hence larger) timestamp. Due to its nature, the timestamp ordering technique can be easily applied to distributed DBMS.

Each local scheduler maintains the additional counters required to store the last accesses for the data items it is responsible for. No further communications among the schedulers or at a global level is needed. However, to guarantee the atomicity of

transactions, transactions must be prevented from seeing partial updates caused by other transactions. This is usually accomplished by deferred updates or pre-writes, where all updates made by a transaction are written into buffers which are flushed upon commit of the transaction.

Optimistic Protocols: When conflicts occur rather rarely, it might prove efficient to let transactions proceed as far as possible and to check for conflicts upon commit time — if a conflict is detected, the transaction has to be aborted. To ensure atomicity, all updates must be made to local copies of the data, which are transferred into the DBMS upon commit. A transaction is first performed on a local level at all participating single DBMSs, using a timestamping based algorithm that ensures local serialisability. If all local validations succeed, a globally acting algorithm verifies that the same serialisation occurs at all sites. If the global validation phase completes successfully, the transaction's changes are written to the database.

Hybrid Protocols: The methods mentioned above can be combined in various ways in order to make use of either advantage. For example, a distributed optimistic 2PL[6] scheduler might attempt an optimistic execution first, and only if this one fails, it undergoes the second attempt by means of 2PL.

Security in distributed database: Security means protection of information and information system from unauthorized access, modification and misuse of information or destruction. The basic of distributed database security[7] is to deal with protecting data from people or software having malicious intentions.

Distributed systems pose four main security components: security authentication, authorization, access control and encryption.

Authentication: Usually authentication is realized by a "smart token" which is a hardware device in the size of a pocket computer or credit card that creates a password and transfers it to the authentication server that is linked up to the network.

Authorization: The aim here is to supply one secured access point enabling the users to link up to the network once and allow them access to authorized resources. The authorization is examined via software servers enabling the client, acting in the name of the user, to prove his identity to the authentication server, without sending information over the network that would reveal the client or the party rendering the service.

Encryption: Implemented using intricate algorithms[8] such as RSA, PGP, DES based on the use of public and private key systems.

Access control: Implemented via access matrices, access lists, capabilities list. These lists define access authorization to the computer resources for the user.

The DDBMS manages all of the distributed data. Multidatabase architectures are Architectures where each local database is managed by a local DBMS and the various DBMSs are connected through a DDBMS. These multidatabase architectures[9] have been studied extensively in the literature. They can be grouped according to whether they are based on tightly coupled or loosely coupled approaches. An orthogonal method is to group the multidatabase systems depending on whether they are based on homogeneous or heterogeneous DBMSs. The components of a DDBMS includes i. ***Distributed query processor (DQP)*** handles distributed queries. ii. ***Distributed transaction manager (DTM)***[10] processing distributed transactions. iii. ***Distributed metadata manager (DMM)*** for managing distributed metadata. iv. ***Distributed integrity manager (DIM)*** for enforcing integrity constraints. v. ***Distributed security manager (DSM)*** for enforcing security constraints.

Results and Discussion

Distributed database systems are a reality. Many organizations are now deploying distributed database systems. Therefore, we have no choice but to ensure that these systems operate in a secure environment. We believe that as more and more technologies emerge, the impact of secure distributed database systems on these technologies will be significant.

We have presented a new family of concurrency control techniques that use the interconnection network in a distributed database system as an aid to concurrency control. The results of my research are:

We have presented concurrency control techniques for distributed database. Distributed concurrency control techniques can be grouped into two general classes as pessimistic and optimistic.

We tried to enhance the security features available in distributed database management system. We are trying to use object oriented distributed database in place of relational distributed database.

We have discussed distributed database concurrency control and security issues in general. Currently, the RDBMS is the better choice for a distributed application.

Conclusion

Distributed database systems are a reality. Many organizations are now deploying distributed database systems. Therefore, we have no choice but to ensure that these systems operate in a secure environment and integrity. Security is concerned with the assurance of confidentiality, integrity, and availability of information in all forms. There are many tools and techniques that can support the management of distributed database security. We discuss the basic concept of concurrency control in

distributed database systems and also discussed the various techniques for concurrency control in distributed environments. It is really important for database to have the ACID properties to perform.

We are in the process of investigating schemes by which the performance of high security level transactions can be improved without compromising with the security. Further we are looking to secure real time distributed systems by which the performance of high security level transactions can be improved without compromising the security.

References

1. Philip A. Bernstein, Vassos Hadzilacos and Nathan Goodman, Concurrency Control and Recovery in Database Systems, *Addison-Wesley* **(1987)**

2. M. Tamer Ozsu and Patrick Valduriez, Principles of Distributed Database Systems, *Prentice-Hall*, II edition, **(1999)**

3. Bernstein P. and Goodman N., Concurrency Control in Distributed Database Systems, *ACM Computing Surveys*, 13/2, **(1981)**

4. Srinivasa Rashmi and Williams Craig, Distributed Transaction Processing on an Ordering Network **(2002)**

5. Kaur Manpreet, Transaction Processing in Distributed Databases, *Amritsar College of Engg. and Tech*, Amritsar **(2006)**

6. Sheetlani Jitendra and Gupta V.K., Concurrency Issues of Distributed Advance Transaction Process, *Res. J. Recent Sci.,* **1(ISC-2011)**, 426-429 **(2012)**

7. Gupta Dhiraj and Gupta V.K., Approaches for Deadlock Detection and Deadlock Prevention for Distributed, *Res. J. Recent Sci.,* **1(ISC-2011)**, 422-425 **(2012)**

8. Shukla Brahma Datta and Gupta V.K., Performance Interoperability between RDBs and OODBs, *Res. J. Recent Sci.,* **1(ISC-2011)**, 419-421 **(2012)**

9. Tiwari Nitin, Solanki Rajdeepsingh and Pandya Gajrajsingh, Intrusion Detection and Prevention System(IDPS) Technoloy- Network Behavior Analysis System (NBAS), *ISCA J. Engineering Sci.,* **1(1),** 51-56 **(2012)**

10. Navathe Elmasri, Database Concepts, *Pearson Education*, IV edition **(2003)**