Vineeta Verma, Gunjan Verma, Deepak Sisodia, Parul Vashist/ International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622
www.ijera.com Vol. 1, Issue 2, pp.051-054

Strict 2 Phase Locking in Organizational Data Protection

Vineeta Verma, Assistant Professor
(Sardar Valabhbhai Patel University of Agriculture & Technology, Meerut)
Gunjan Verma, Assistant Professor
(Meerut institute of Engineering & Technology, Meerut)
Deepak Sisodia, Assistant Professor
(Sardar Valabhbhai Patel University of Agriculture & Technology, Meerut)
Parul Vashist, Assistant Professor
(Greater Noida Institute of Technology, Greater Noida)

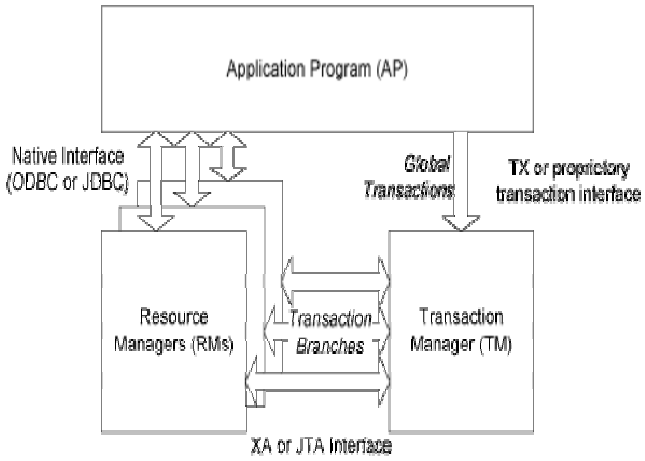
Abstract: Privacy and data security have been issues since the advent of paper and pencil. But with the emergence of networked computing and the Internet, controls that worked effectively for the last 20 or 30 years have to be reevaluated. This paper addresses first the protection of organizational data that has to be reassessed.

Keywords: Serialization, Two – Phase locking, XA interface, 2PL, PC.

I. Introduction

When the computer was not invented and people were not aware of that most the people and organizations kept their data and files in manual way to maintain its security they use lock and key. When computer came in light people started to use computer. Computer made the work easy and data can be arranged in a proper way to access by using database.

There are many applications that require to access database which is located in different databases. Local transactions are executed under the control of pre-existing and autonomous local database management systems (DBMSs) while global transactions accessing several DBMSs are under the control of a Global Transaction Manager (GTM). A global transaction is split into subtransactions, each one being considered as a local transaction by the local DBMS. Various protocols have been proposed to guarantee the ACID properties of global transactions when the accessed local DBMSs support different serialization and validation policies. Typically, most relational and object-oriented commercial DBMSs serialize local transactions by 2 Phase-Locking and support the XA interface defined in the X/Open distributed transaction processing **DTP** model to participate in a standardized 2PC protocol.



Description of Distributed Transaction Processing Model

MDBSs can be explained by the fact that all transaction schedules are globally serializable if (i) all local DBMSs are rigorous and (ii) all local DBMSs participate in a 2PC protocol. Unfortunately,

this ignores the fact that some local DBMSs exploit the strict Two Phase Locking (2PL) optimization to increase inter-transaction parallelism. DBMSs supporting both strict 2PL and Two Phase Commit (2PC) protocols are called strict-2PL, PC DBMSs.

In databases, a popular method to guarantee transaction serializability is the 2-Phase Locking (2PL) protocol. Strict 2PL protocol is an important variation of 2PL protocol. In Strict 2PL, a transaction must gain exclusive access to the shared data between its initial access to the data and the end of the transaction. Exclusive access means:
(I) No other transaction can write a data if the transaction is exclusively reading the data.
(II) No other transaction can read or write a data if the transaction is exclusively writing the data. Not violating strict 2PL is sufficient yet not necessary for serializability.

II. 2 Phase Locking

There are many ways by with we can locked the data. Here we are using two ways:

1. **Shared**. If a transaction T_i has obtained a shared-mode lock (denoted by S) on item Q, then *T_i can read, but cannot write.*

2. **Exclusive**. If a transaction T_i has obtained an exclusive-mode lock (denoted by X) on item Q, then *T_i can both read and write.*

The locks can be defined according to their blocking behaviour:

- An existing *write-lock* on a database object blocks an intended *write* upon the same object by another transaction by blocking a respective *write-lock* from being acquired by the other transaction. The second write-lock will be acquired and the requested write of the object will take place after the existing write-lock is released.
- A *write-lock* blocks an intended (already requested/issued) *read* by another transaction by blocking the respective *read-lock*.
- A *read-lock* blocks an intended *write* by another transaction by blocking the respective *write-lock*.
- A *read-lock* does not block an intended *read* by another transaction. The respective *read-lock* for the intended read is acquired (shared with the previous read) immediately after the intended read is requested, and then the intended read itself takes place.

We can show a compatibility table for these modes:

| S | X |

S | T | F |

X | F | F |

For a transaction to unlock a data item immediately after its final access of that data item is not always desirable, since serializability may not be ensured.

In databases and transaction processing **two-phase locking**, (**2PL**) is a concurrency control that guarantees serializability. The protocol utilizes locks that block other transactions from accessing the same data during a transaction's life.

One protocol that ensures **serializability** is the two-phase locking protocol. This protocol requires that each transaction issue lock and unlock requests in two phases:

1. Growing phase. A transaction may obtain locks, but may not release any lock.
2. Shrinking phase. A transaction may release locks, but may not obtain any new locks.

Initially, a transaction is in the growing phase. The transaction acquires locks as needed. Once the transaction releases a lock, it enters the shrinking phase, and it can issue no more lock requests. Two-phase locking does not ensure freedom from deadlock. Cascading rollback may also occur.

III. Strict Two-Phase Locking

Cascading rollbacks can be avoided by a modification of two-phase locking called the ***strict two-phase locking*** protocol. The strict two-phase locking protocol requires that in addition to locking being two-phase, all exclusive-mode locks taken by a transaction must be held until that transaction commits.

The **strict two-phase locking** (S2PL) class of schedules is the intersection of the 2PL class with the class of schedules possessing the ***Strictness*** property.

To comply with the S2PL protocol a transaction needs to comply with 2PL, and release its *write (exclusive)* locks only after it has ended, i.e., being either *committed* or *aborted*. On the other hand, *read (shared)* locks are released regularly during phase 2. Implementing general S2PL requires explicit support of phase-1 end, separate from transaction end, and no such widely utilized product implementation is known.

S2PL is a special case of 2PL, i.e., the S2PL class is a proper subclass of 2PL.

Security Issues: Database Security and Authorization

The data stored in the database need to be protected from unauthorized access, malicious destruction or alteration, and accidental introduction of inconsistency.

To Protect a Database, Security Measures take several levels.

- **Physical**. The site or sites containing the computer systems must be physically secured against armed or surreptitious entry by intruders.

- **Human**. Users must be authorized carefully to reduce the chance of any such user giving access to an intruder in exchange for a bribe or other favors.

- **Operating System**. No matter how secure the database system is, weakness in operating-system security may serve as a means of unauthorized access to the database.

- **Network**. Since almost all database systems allow remote access through terminals or networks, software-level security with the network software is as important as physical security, both on the Internet and in networks private to an enterprise.

- **Database System**. Some database-system users may be authorized to access only a limited portion of the database. Other user may be allowed to issue queries, but may be forbidden to modify the data. It is the responsibility of the database system to ensure that these authorization restrictions are not violated.

IV. Authorization

A user may have several forms of authorization on parts of the database:

- **Read authorization** allows reading, but not modification, of data.
- **Insert authorization** allows insertion of new data, but not modification of existing data.
- **Update authorization** allows modification, but not deletion, of data.
- **Delete authorization** allows deletion of data.

A user may be assigned some, all, or none of these.

A user may be authorized to change the database schema:

- Index authorization
- Resource authorization (new relations)
- Alteration authorization (add or delete attributes)
- Drop authorization (delete relations)

For DBA

- Account creation
- Privilege granting
- Privilege revocation
- Security level assignment

V. Conclusion

Although 2-PC provides autonomy of a transaction, the required processing load is rather heavy. The

transaction speed is always limited by the resource manager with the slowest response, and the network traffic and latency is double that of a normal transaction because of the intermediate, commit request. Although 2-PC provides some reliability in achieving ACID compliance for distributed transactions. By using strict Two Phase locking method for data security and transaction handling we can also avoid the problem of concurrency control. Transaction that holds a shared lock can be upgraded to hold an exclusive lock. There are several lock-based concurrency control schemes (Strict 2PL, 2PL). Locks directly implement the notions of conflict. The lock manager keeps track of the locks issued.Deadlocks can either be prevented or detected by these schemes.

VI. References

[1] A. Razavi, S.Moschoyiannis, P.Krause. A Coordination Model for Distributed Transactions in Digital Business Ecosystems. In Proc. IEEE Int'l Conf on Digital Ecosystems and Technologies (IEEEDEST' 07). IEEE Computer Society, 2007.

[2] C. J. Date , An introduction to Database Systems (5th edition), Addison Wesley, USA, 1996.

[3] S. Greenberg and D. Marwood. Real time groupware as a distributed system: concurrency control and its effect on the interface. In Proc. ACM Conference on Computer Supported Cooperative Work, pages 207– 217. ACM Press, Nov. 1994.

[4] P. Bernstein, N. Goodman, and V. Hadzilacos. Concurrency Control and Recorvery in Database Systems. Addision-Welsley, 1987.

[5] J. Gray, A. Reuter. Transaction processing: Concepts and Techniques, Morgan Kaufmann Publishers, USA, 1993.

[6] A. Elmagarmid , Database Transaction Model for Advanced applications, Morgan – Kaufmann, 1994.

[7] T. Kakeshita, Xu Haiyan , "Transaction sequencing problems for maximal parallelism", Second International Workshop on Transaction and Query Processing (IEEE), 2-3 Feb. 1992, pp: 215 – 216, 1992.

[8] M.S. Haghjoo, M.P. Papazoglou, "TrActorS: a transactional actor system for distributed query processing", Proceedings of the 12th International Conference on Distributed Computing Systems (IEEE CNF), 9-12 June 1992, pp: 682 – 689, 1992.

[9] A. Razavi, P. Malone, S.Moschoyiannis, B.Jennings, P.Krause. A Distributed Transaction and Accounting Model for Digital Ecosystem Composed Services. In Proc. IEEE Int'l Conf on Digital Ecosystems and Technologies (IEEE-DEST'07). IEEE Computer Society, 2007.

[10] J. E. B. Moss , Nested transaction an approach to Reliable Distributed Computing, MIT Press, USA,1985.