CONCURRENCY CONTROL IN TRUSTED DATABASE MANAGEMENT SYSTEMS: A SURVEY

Bhavani Thuraisingham and Hai-Ping Ko

The MITRE Corporation
Burlington Road, Bedford, MA 01730

ABSTRACT

Recently several algorithms have been proposed for concurrency control in a Trusted Database Management System (TDBMS). The various research efforts are examining the concurrency control algorithms developed for DBMSs and adapting them for a multilevel environment. This paper provides a survey of the concurrency control algorithms for a TDBMS and discusses future directions.

1. INTRODUCTION

A transaction is normally considered as a program unit that must be executed in its entirety or not executed at all. The module of a database management system (DBMS) that is responsible for transaction execution is a transaction manager. An objective in most database management systems is to execute multiple transactions concurrently. Generally, transactions could interfere with one another and, as a result, could cause the database to be inconsistent. The techniques that have been developed to ensure the consistency of a database in the midst of concurrent transaction execution are called concurrency control techniques [BERN87].

In a multilevel environment, the users are assigned different security levels, and they access a database consisting of data at different sensitivity levels. The DBMS must ensure that users only obtain information at or below their security level. Such a DBMS is called a trusted DBMS (TDBMS).[1] When transactions are executed in a multilevel environment in addition to consistency, it must be ensured that the access control policy enforced by the system is not violated and transactions executing on behalf of higher level users do not interfere with those executing at a lower level. For example, if an Unclassified transaction wants to update an Unclassified data object when a Secret transaction is already reading that same object, and if the Unclassified transaction has to wait until the Secret transaction completes the read operation, then there is undesirable interference. The objective of concurrency control algorithms will be to ensure security as well as consistency.

[1] In the literature, TDBMSs have sometimes been referred to as Multilevel Secure Database Management Systems (MLS/DBMSs).

While transaction management techniques are relatively mature for traditional database applications (such as banking and business data processing), it is only recently that concurrency control techniques are being examined for a multilevel database environment. In other words, the developments in multilevel database concurrency control are more than a decade behind the developments in database concurrency control. Furthermore, during recent years, database concurrency control has progressed beyond traditional applications and techniques are now being developed for advanced applications. Such applications involve heterogeneous environments, real-time processing, long duration transactions, and collaborative computing environments. Furthermore, theory of database concurrency control is sufficiently developed for traditional database applications. Therefore, much needs to be done on concurrency control for multilevel database applications.

Recently several algorithms have been proposed for concurrency control in a TDBMS. The various research efforts are examining the concurrency control algorithms developed for DBMSs and adapting them for a multilevel environment. Due to the influx of these algorithms, we feel that it is necessary to group these algorithms so that one gets a better perspective of the current research on concurrency control in TDBMSs. This will then enable us to determine the research directions based on the current trends on concurrency control in DBMSs. In this paper we survey the various concurrency control algorithms in TDBMSs and discuss their essential characteristics. We have also determined the areas that need further work.

The organization of this paper is as follows. In section 2 we provide a brief overview of the current trends on concurrency control in DBMSs. This is necessary if one is to determine the directions for concurrency control in TDBMSs. In section 3 we describe in detail our survey of concurrency control in TDBMSs. Future directions are given in section 4.

2. A BRIEF OVERVIEW CONCURRENCY CONTROL IN DBMSs

Transaction management and concurrency control have been a subject of much research since the mid-1970s. The early work focussed on developing algorithms for traditional database applications. These applications are mainly in business processing and the DBMS is based on the relational model. The goal is to maintain strict consistency when multiple transactions execute concurrently [BERN87].

While the algorithms were being incorporated into the numerous prototypes and products, research was also being carried out on developing (1) algorithms for handling a distributed environment and (2) a theory of database concurrency control. Distributed concurrency control algorithms focussed on ensuring the consistency of a distributed database when multiple transactions executed concurrently at different sites [BERN81, CERI84]. These algorithms also included the case where the data objects were replicated (either totally or partially) at different sites. In addition, issues on handling site and network failures were also investigated. The distributed DBMS (DDBMS) was based on the relational model, and it was assumed that the environment was homogeneous. In developing a theory of database concurrency control, the idea was to formulate conditions which will test whether a particular concurrency control algorithm will guarantee consistency of the database in the midst of multiple updates [PAPA86]. In addition to these developments, issues on data replication as well as weaker notions of consistency requirements were investigated.

Since the mid-1980s, much research is being carried out for advanced database applications. Algorithms for DBMSs based on nonrelational data models (such as object-oriented data models) [KIM88] as well as algorithms for heterogeneous and possibly autonomous environments are being investigated [SHET90]. In addition, special algorithms for handling nested transactions [MOSS85], cooperative computing environments [BARG91] as well as real-time environments [HUAN91] are being developed.

3. CONCURRENCY CONTROL ALGORITHMS IN TRUSTED DATABASE MANAGEMENT SYSTEMS

3.1 OVERVIEW

As stated in section 1, if transactions are executed serially, then there will be a performance bottleneck. Therefore, transactions usually execute concurrently as specified by a schedule. In a multilevel environment, in addition to consistency, it has to be ensured that high level transactions cannot affect the lower level ones at the local as well as the global levels. Concurrency control techniques must ensure that consistency as well as security has to be preserved. It has been shown that standard concurrency control techniques cannot be used directly for a TDBMS as they cause covert channels. For example, in the locking approach to providing concurrency control there is a potential for covert channels if subjects at different security levels access the same object concurrently. For example, suppose a Secret transaction requests read access to an Unclassified data object Q while an Unclassified transaction requests either read or write access to Q. If both transactions want only to read Q, then they can both get shared locks for Q at the same time. That is, there is no possibility for a channel. However, if the Unclassified transaction wants to write, then there is a possibility for a channel. For example, a Secret transaction can issue a sequence of requests of the form:

(get read lock Q, release read lock Q),
(get read lock Q, release read lock Q), ...

The Unclassified transaction could issue requests to write into Q. If the Secret transaction already has a read lock, then the request issued by the Unclassified transaction will be denied. The Unclassified transaction could detect the pattern of the denials and acceptances that it gets by requesting a series of write requests to the object Q. If both transactions collude, then Secret information can be covertly passed to the Unclassified transaction. A similar problem occurs when a Secret transaction requests read or write access to a Secret object Q while an Unclassified transaction requests write access to Q.

As in the case of locking technique, it has been shown that other concurrency control techniques such as timestamping also cannot be used for a TDBMS without modifications. As a result, some research efforts are under way on developing appropriate concurrency control algorithms for a TDBMS. We have classified the algorithms developed into five groups.[2] The first group consists of algorithms which are based on multiversion schedulers. The second group consists of algorithms for replicated architectures. The third group consists of algorithms for distributed database management systems. The fourth group consists of algorithms for object-oriented database systems. The remaining algorithms have been grouped into the fifth category.[3] Algorithms will be

2 Some of the algorithms assume a security policy where a transaction at level L reads an object at level L or below and writes into an object at level L or above. Some other algorithms assume a policy where a transaction at level L reads an object at level L or below and writes into an object at level L. The discussion of the algorithms given in this section is brief. For each algorithm we have given one or more references. Details of the algorithms can be obtained from the references cited.

3 Judy Froscher [FROS92], has suggested to us an alternate approach for characterizing the secure concurrency control algorithms based on the underlying architectures. The architectures include those proposed by the Air Force Summer Study such as the kernelized architecture, replicated architecture, and integrity lock architecture. It has also been pointed to us that an investigation of the concurrency control issues for kernelized and replicated architectures is being carried out by Froscher and Kang at NRL. Another approach to characterize the algorithms could depend on whether transactions are single-level or multilevel. We feel that more research needs to be done on the fundamental issues of

selected from each of these categories and discussed in the ensuing sections 3.2 to 3.6.[4]

3.2 MULTIVERSION ALGORITHMS

To our knowledge, the earliest published work on multilevel database concurrency control is that of Keefe, Tsai, and Srivastava [KEEF89]. Further work was carried out by Keefe and Tsai [KEEF90]. The algorithm developed was based on multiversion schedulers. Following their work, Maimone and Greenberg described their version of multiversion scheduling algorithm in [MAIM90]. In this section, we first describe Keefe and Tsai's multiversion scheduling algorithms as given in [KEEF90] and then discuss Maimone and Greenberg's multiversion algorithms described in [MAIM90].

Keefe and Tsai's Multiversion Scheduler

The security policy proposed is single level subjects with read at-or-below-your level and write-at-or-above your level. The scheduler creates a serial order of transactions for each data object. Initially, the serial ordering is empty. When a transaction T arrives, it is placed in the order in such a way that it precedes all transactions S such that subject-level(S) < subject-level(T) where the subject-level of a transaction is the level at which the transaction executes. T is given an order-stamp depending on the position of T in the serial order.

Each data object x has read and write order-stamps. The value of the read order-stamp of an object is the greatest order-stamp of any transaction which has read the object. The value of the write order-stamp of an object is the greatest order-stamp of any transaction that has written into the object. Whenever a transaction writes into an object, a new version of the object is created. The version of an object that is appropriate for a transaction T is the one with the greatest write order-stamp which is still less than the order-stamp of T.

Whenever a transaction T requests a read operation on object O and the operation is not the last step of the transaction, it is given the appropriate version of O. Whenever a transaction T requests a write operation on O and the operation is not the last step of the transaction, it is given the appropriate version of O if the order-stamp of T is greater than the read order-stamp of the version. The last operation of a transaction T proceeds only if for each transaction P, if P has written something that T has read, then P is already committed.

concurrency control algorithms before a successful characterization can be obtained.

[4] Since transaction recovery is not addressed in this paper, we do not discuss recovery in multilevel secure database systems. Recently, some work has been done on recovery management in such systems (see, for example, [KANGI92] for a discussion).

Keefe and Tsai prove that their algorithm is secure with respect to resource contention between transactions at different levels. That is, the algorithm does not cause covert channels due to resource contention.

Maimone and Greenberg's Multiversion Schedulers

Maimone and Greenberg [MAIM90] have designed two algorithms based on the multiversion timestamp ordering technique. One approach was implemented for Trusted Oracle, which is a hybrid approach between locking and timestamping techniques. The second algorithm is a modification of Keefe and Tsai's multiversion scheduler. We briefly discuss the two techniques.

In the Trusted Oracle scheduler, each data object has a writestamp. The writestamp is updated whenever a transaction which writes the value commits. Since it is assumed that a subject does not write up, any data object that is updated will have the security level of the transaction which updated it. Also, two-phase locking is used to ensure that at most one uncommitted version of each data object exists at any instant. For a read only transaction T, a timestamp is obtained when it begins execution. T can then read any data object provided the level of T dominates the level of the object. Also, the appropriate version of the object to read is the one with the highest write-stamp, which is lower than the timestamp of T. For read/write transactions, two-phase locking is used to prevent conflicts between read/write and write/write operations.

Maimone and Greenberg state that Keefe and Tsai's algorithm requires a trusted scheduler and then propose a modified version of the algorithm which can be implemented by single level untrusted schedulers. They describe a method of assigning timestamps across security levels, which does not cause any information leakage from a high to low level. A scheduler operating at level L assigns timestamps to transactions operating at level L. The algorithm presented is somewhat similar to Keefe and Tsai's algorithm except that it is more conservative and could use older values than it is necessary.

3.3 REPLICATED ARCHITECTURE

Shortly after Keefe and Tsai's work on multiversion concurrency control algorithm was published, Jajodia and Kogan focussed on concurrency control algorithms for replicated architecture [JAJO90]. Following Jajodia and Kogan's work, a flurry of activities was reported on concurrency control for the replicated architecture. Notable among these efforts are the concurrency control algorithms developed for the SINTRA project. In this section, we describe some of the concurrency control algorithms described for the replicated architecture.

Jajodia and Kogan's Algorithm for Replicated Databases

Jajodia and Kogan [JAJO90] proposed the first algorithm for the replicated distributed architecture. In this architecture (which was originally proposed by the Air Force Summer Study [AFSB83]), a trusted front-end is connected to multiple untrusted back-end machines. All communication between the back-end machines is via the front-end. Each back-end machine operates at a single level. A back-end machine operating at level L manages the database consisting of data up to and including level L. That is, the data at level L is replicated in the database at level L* if L* dominates L.

The algorithm proposed is as follows: A transaction at level L is sent by the front-end to the back-end machine operating at level L. The transaction executing in this back-end machine is called the parent transaction. After the parent transaction commits, the transaction is sent to the back-end machines operating at level L* where L* dominates L. These transactions are called update projections. The update projections are executed according to a specific protocol so that correctness of concurrent transactions are guaranteed. It is assumed that the scheduler is trusted.

SINTRA Project

SINTRA (Secure INformation Through Replicated Architecture) project carried out at the Naval Research Laboratory has designed several algorithms for the replicated architecture. We have selected a subset of the algorithms proposed by this project and discuss the essential points. The algorithms are: (1) Costich's algorithm [COST91], (2) the algorithm by McDermott et al [MCDE91], (4) Costich and McDermott's algorithm [COST92], and the algorithm by Kang et al [KANGM92].

In [COST91], a scheduling protocol for replicated architecture is proposed which is based on entirely untrusted processes. It is argued that while the algorithm proposed in [JAJO90] fails if the security levels are linearly ordered, the one proposed in [COST91] handles security levels which are not linearly ordered. One-copy serializability is used for correctness criteria.

In [MCDE91], an algorithm for the replicated architecture is proposed which uses replicated transactions and a set of queues organized according to security classes. It is argued that a new notion of correctness is required for such an environment. Subsequently, a new notion of correctness is proposed, and it is shown that the algorithm is correct.

In [COST92], an algorithm for the replicated architecture is given which assumes that transactions are multilevel.[5] These are transactions that operate at different security classes. A transaction model, which incorporates the notion of multilevel replicated data history, is given for the replicated architecture, and a protocol is described for executing the primary transaction and update projections in such a way that the resulting replicated data history is one copy serializable.

In [KANGM92], it is stated that the previous algorithms, such as the ones given in [JAJO90, COST91, and MCDE91], assume that the untrusted backend databases use conservative scheduling in order to preserve the scheduled ordering of conflicting updates. Since the back-end machines are off-the-shelf components, they may not use such a scheduling protocol. It is also stated in [KANG92] that the previous approaches use the conventional basic operations, read and write to describe transactions. It is argued that the traditional transaction model is not appropriate for the replicated architecture, and a new transaction model is proposed. Subsequently, different algorithms which use untrusted transaction managers and which overcome some of the problems associated with the previous approaches are described.

3.4 DISTRIBUTED DATABASE MANAGEMENT SYSTEMS

While various concurrency control algorithms were being proposed based on multiversion scheduling and for the replicated architecture, MITRE was conducting research and development activities on concurrency control for a trusted distributed database environment (TDDBMS) [THUR90]. Around the same time, SRI also conducted research on distributed database concurrency control [GREE91]. In this section, we discuss both MITRE's work and SRI's effort.

MITRE's Research

MITRE conducted some research on concurrency control algorithms for a trusted distributed database management system. The original version is published in [THUR89] and refined versions of the algorithms are given in [THUR90a]. The algorithms have since been simulated and performance results are discussed in [RUBI92a, RUBI92b]. Extensions to a limited heterogeneous environment where the nodes handle different accreditation ranges are discussed in [THUR92a]. The algorithms that have been designed are based on locking, timestamping, and validation. All of the algorithms utilize

[5] Note that multilevel transactions discussed here are transactions that execute at multiple security levels. Note that multilevel transactions have been used to denote something different in nonmultilevel DBMSs. For a discussion we refer to section 1.6.3.

two copies of a data item. Here we discuss the essential points of just the locking algorithm.[6]

For each data object classified at level L, two copies of it are maintained at level L; one copy is for all transactions operating at the level L, and the other copy is for all transactions operating at a level which strictly dominates the level L. The transactions operating at a level L may read from and write into the copy assigned to them. The transactions operating at a level which strictly dominates the level L only read from the copy assigned to them. For example, if the data object X is Unclassified, then there are two copies of X, X1, and X2 at the Unclassified level. Unclassified transactions can read from and write into the object X1. Confidential, Secret, and TopSecret transactions read from the copy X2. Since X2 is used for read-only purposes, we assume that there are no read-locks on X2. However, if an Unclassified transaction has a write-lock on X1 when a higher level transaction requests a read operation on X2, then the higher level transaction must wait until the write operation is completed. As soon as the write operation is committed by the Unclassified transaction, a copy of the data object, say X2*, is created immediately. X2*□becomes the current version to be used by the higher level transactions. That is, after an Unclassified transaction requests the write lock for X1, any read request by a higher level transaction would be directed to X2*. X2□may be deleted later by a garbage collector. To ensure consistency, X2 should be deleted only if there are no read requests queued for it. Since we assume that X2 is an Unclassified object, the process which deletes X2 must be trusted.

The algorithm is extended to a distributed environment using distributed two-phase locking and two-phase commit protocol.

SRI's Research

SRI did some early research on concurrency control for a trusted distributed database management system [DOWN89]. Further work is reported in [GREE91]. We discuss some of the essential points in SRI's approach as given in [DOWN89].

In [DOWN89], locking, timestamping, and optimistic concurrency control techniques are examined, and it is concluded that optimistic concurrency control technique could provide serializability under the assumptions of the Seaview architecture [LUNT88]. The algorithm works as follows: Suppose Ti is a higher level transaction in the validation stage. If there is a lower level transaction Tj which has a smaller transaction number than Ti and Ti starts its read phase before Tj completes its write phase and the writeset of Tj has a nonempty intersection with the readset of Ti, then Tj is rolled back and started again; otherwise Tj is committed.

As stated in [DOWN89], a problem with the proposed approach is that there is a possible starvation with higher level transactions. If transactions are continuously rolled back, then it is suggested that one possibility would be to violate serializability and let the transactions commit despite conflicts.[7]

3.5 OBJECT-ORIENTED DATABASE MANAGEMENT SYSTEMS

The only effort reported on concurrency control for secure object-oriented database systems is that of Thuraisingham [THUR90b]. In this work, the concurrency control technique proposed by Kim et al [KIM88] in the ORION model is extended to a multilevel environment. The algorithm utilizes the two-copy approach discussed in [THUR90a] but considers the full range of locks discussed in [KIM88].

Due to the complexity of the objects present in an object-oriented environment, several locking modes have been defined in [KIM88]. These locks are S, X, IS, IX, SIX, ISO, IXO, SIXO, ISOS, IXOS, and SIXOS. In [KIM88], a compatibility matrix for granularity locking and composite object locking is described. This matrix illustrates whether a lock P1 on object O can be granted to a subject T1 when a subject T2 has a lock P2 on object O. In [THUR90b], the various lock modes described in [KIM88] have been examined, and the security impact is discussed. In this approach, there are two copies of each data object (class, instance, etc.,) is maintained, one for subjects at the same level and one for subjects at higher levels. The compatibility matrix described in [KIM88] is used on a per security level basis. That is, the matrix is used to determine whether a lock P1 on object O can be granted to a subject T1 at level L when a subject T2 at level L has a lock P2 on object O. Modifications to this matrix are described in [THUR90b] when T1 and T2 are operating at different security levels.

3.6 OTHER ALGORITHMS

Notable among the other algorithms are LDV's approach to database consistency [OBRI90], the single-version timestamping algorithm proposed by Ammann and Jajodia [AMMA91], algorithm based on event count and sequences proposed by McCollum and Notargiacomo [MCCO91], two-snapshot algorithm proposed by Ammann et al [AMMA92], correctness criteria proposed by Jajodia and Atluri [JAJO92],

6 MITRE's work on concurrency control for a TDDBMS discussed in [RUBI92a, RUBI92b] was a consequence to the initial work on query processing for a TDDBMS reported in [RUBI90].

7 In [RUBI92a], an optimistic concurrency control algorithm is proposed which maintain two copies of each data item. It is informally argued that the proposed algorithm is secure, serializable, and there is no starvation.

orange locking algorithm proposed by McDermott and Jajodia [MCDE92], and an algorithm for multilevel transactions proposed by Costich and Jajodia [COST92b]. We briefly discuss these efforts.

In [OBRI90], Jajodia and Kogan's concurrency control algorithm is adapted to a kernelized DBMS architecture. It is the architecture which is used by the Lock Data Views system [STAC90]. In particular, a consistency policy for a trusted DBMS is given in [OBRI90].

Amman and Jajodia [AMMA91] have proposed a timestamp ordering algorithm which uses just a single copy. In this algorithm, if a lower level transaction requests a write on data object when a higher level transaction is reading the object, then the higher level transaction is aborted and priority is given to the lower level transaction.

McCollum and Notargiacomo [MCCO91] have proposed an algorithm for a distributed architecture based on the trusted front-end and untrusted back-end approach. The algorithm can handle no replication, partial replication, or even total replication. It extends the event count and sequencers algorithm described in [REED79] to a multilevel environment.

Ammann et al [AMMA92] have proposed an algorithm called the two-snapshot algorithm. In some ways, this algorithm is similar to the two-copy algorithm proposed in [THUR90a]. In this algorithm, two snapshots of the database are maintained at each security level. There is also a full working database at each security level. High level transaction access snapshots of low level data. As stated in [AMMA92], periodically new snapshots are taken at specified security levels and high level transactions are methodically given access to the new snapshots.

Jajodia and Atluri [JAJO92] propose three different notions of serializability which are alternatives to one-copy serializability. They argue that one-copy serializability might be too restrictive for trusted database systems and hence, the need for alternate notions of serializability.

McDermott and Jajodia have proposed concurrency control algorithms called the orange-locking algorithms [MCDE92]. It is stated that these algorithms do not use multiple versions and yet provide serializable schedules without introducing covert channels. The basic idea is to change standard locks to orange-locks when certain situations occur.

Costich and Jajodia [COST92b] have proposed an algorithm for multilevel transactions. It is stated that most of the previous approaches assume that transactions operate at a single level and that in certain cases, a transaction may have to operate at multiple security levels. It is also shown that the proposed algorithms for multilevel transactions is one-copy serializable. While the algorithms for multilevel transactions discussed in [COST92a] focussed mainly on the replicated architecture and for a limited class of transactions, [COST92b] considers a larger class of transactions as well as architectures.

4. DIRECTIONS FOR FURTHER RESEARCH

Multilevel Database Concurrency Control is following database concurrency control fairly closely. Various algorithms have been proposed, and it has been argued informally that the algorithms are serializable and secure. As pointed out to us by Froscher [FROS92], research recently began on investigating concurrency control issues which are specific to the approaches used to design a TDBMS. These approaches include those based on the replicated and kernelized architectures.

However, before one can guarantee serializability and security of the algorithms, a theory of multilevel database concurrency control needs to be developed. Such a theory will formulate conditions for checking serializability and security of the various algorithms proposed. Therefore, we believe that developing such a theory is essential if useful TDBMSs with higher assurance are to be developed.

Once it can be ensured that an algorithm is serializable and secure, then the next question is the selection of an efficient algorithm. We envision that performance will be a major consideration in such a selection. Therefore, analytical as well as simulation studies need to be carried out to determine the performance of the various algorithms proposed. The work carried out at MITRE and reported in [RUBI92a] is just the first step towards such a study.

Finally, while research should continue on designing secure concurrency algorithms for traditional multilevel database applications (such as investigating concurrency control issues for various architectures, extending nested transactions for a multilevel environment, proposing algorithms for multilevel transactions, and processing integrity and security constraints during transaction execution in a multilevel environment), research should also begin on concurrency control for advanced multilevel database applications. In particular, the security impact on heterogeneity, real-time processing, long-duration transactions, and cooperative computing environments, need to be determined. A preliminary investigation of research on transaction management for some of the new generation applications is given in [THUR92b,THUR92c, DEMU92].

ACKNOWLEDGMENT: We gratefully acknowledge Rome Laboratory for sponsoring our work on concurrency control in TDBMSs.

LIST OF REFERENCES

[AFSB83] Air Force Studies Board, 1983, Committee on Multilevel Data Management Security, *Multilevel Data Management Security*, National Academy Press.

[AMMA91] Ammann, P., and S. Jajodia, 1991, "A Timestamp Ordering Algorithm for Secure Single-Version Multilevel Databases," Proceedings of the 5th IFIP Working Conference in Database Security, Shepherdstown, West Virginia.

[AMMA92] Ammann, P. et al, 1992, A Two Snapshot Algorithm for Concurrency Control in Multilevel Secure Databases," Proceedings of the IEEE Symposium of Security and Privacy, Oakland, California.

[BARG91] Bargouti, N., and G. Kaser, 1991, "Concurrency Control in Advanced Database Applications," *ACM Computing Surveys*, Vol. 23, #3.

[BERN81] Bernstein, P., and N. Goodman, 1981, "Concurrency Control in Distributed Database Systems," *ACM Computing Surveys*, Vol. 13, #2.

[BERN87] Bernstein, P. A., V. Hadzilacos, and Goodman, N., 1987, *Concurrency Control and Recovery in Database Systems*, Addison-Wesley Publishing Company.

[CELL88] Cellary, W., E. Gelenbe, and T. Morzy, 1988, *Concurrency Control in Distributed Database Systems*, North Holland, Amsterdam.

[CERI84] Ceri, S., and G. Pelagatti, 1984, *Distributed Databases, Principles and Systems*, McGraw-Hill Book Company, NY.

[COST91] Costich, O., 1991, "Transaction Processing Using an Untrusted Scheduler in a Multilevel Database with Replicated Architecture," Proceedings of the 5th IFIP Working Conference in Database Security, Shepherdstown, West Virginia.

[COST92a] Costich, O., and J. McDermott, 1992, "A Multilevel Transaction Problem for Multilevel Secure Database Systems and Its Solution for the Replicated Architecture," Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA.

[COST92b] Costich, O., and S. Jajodia, 1992, "Maintaining Multilevel Transaction Atomicity in MLS Database Systems with Kernelized Architecture," Proceedings of the 6th IFIP Working Conference in Database Security, Vancouver, British Columbia.

[DEMU92] Demurjian, S., T.C. Ting, and B. Thuraisingham, September 1992, "Security for Collaborative Computing Environments," to be submitted for publication.

[DENN87] Denning, D. E., et al, April 1987, "A Multilevel Relational Data Model," Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA.

[DOWN89] Down, A., et al, 1989, "Issues in Distributed Database Security," Proceedings of the 5th Computer Security Applications Conference, Tucson, AZ.

[FROS92] Froscher, J., September 1992, Private Communication.

[GREE91] Greenberg, I., 1991, *Distributed Database Integrity*, Final Report A002, SRI International, Menlo Park CA.

[HUAN91] Huang, J., 1991, *Real-time Transaction Processing: Design, Implementation, and Performance Evaluation*, Ph.D. Thesis, University of Massachusetts.

[JAJO90] Jajodia, S., and B. Kogan, 1990, "Transaction Processing in Multilevel Secure Databases Using the Replicated Architecture," Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA.

[JAJO92] Jajodia, S., and V. Atluri, 1992, "Alternative Correctness Criteria for Concurrent Execution of Transactions in Multilevel Secure Databases," Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA.

[KANGI92] Kang, I. and T. Keefe, 1992, "Recovery Management for Multilevel Secure Database Systems," Proceedings of the 6th IFIP Working Conference in Database Security, Vancouver, British Columbia.

[KANGM92] Kang, M., et al, 1992, "A Practical Transaction Model and Untrusted Transaction Manager for a Multilevel Secure Database System," Proceedings of the 6th IFIP Working Conference in Database Security, Vancouver, British Columbia.

[KEEF89] Keefe, T., W.T. Tsai, and J. Srivastava, 1989, *Database Concurrency Control in Multilevel, Secure Database Management Systems*, Technical Report 89-73, University of Minnesota (a version also published in the Proceedings of the 6th IEEE Data Engineering Conference).

[KEEF90] Keefe, T., and W.T. Tsai, 1990, "Multiversion Concurrency Control for Multilevel Secure Database Systems," Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA.

[KIM88] Kim, W., et al, November 1988, *Composite Object Revisited*, MCC Technical Report, ACA-ST-387-88.

[LUNT88] Lunt, T., 1988, "Multilevel Database Systems: Meeting Class A1," Proceedings of the 2nd IFIP Working Conference in Database Security, Ontario.

[MAIM90] Maimone, W., and I. Greenberg, 1990, "Single-level Multiversion Schedulers for Multilevel Secure Database Systems, " Proceedings of the 6th Computer Security Applications Conference, Tucson, AZ.

[MCCO91] McCollum, C., and L. Notargiacomo, 1991, "Distributed Concurrency Control with Optional Data Replication," Proceedings of the 5th IFIP Working Conference in Database Security, Shepherdstown, West Virginia.

[MCDE91] McDermott, et al, 1991, "A Single-level Scheduler for the Replicated Architecture for Multilevel Secure Databases," Proceedings of the 7th Computer Security Applications Conference, St. Antonio, TX.

[MCDE92] McDermott, J., and S. Jajodia, 1992, "Orange Locking: Channel-free Database Concurrency Control Via Locking," Proceedings of the 6th IFIP Working Conference in Database Security, Vancouver, British Columbia.

[MOSS85] Moss, E., 1985, *Nested Transactions: An Approach to Reliable Distributed Computing*, The MIT Press, Cambridge, MA.

[OBRI90] R. O'Brien, et al, 1990, *Trusted Database Consistency Policy*, RADC Technical Report TR-90-387.

[PAPA86] Papadimitriou, C., 1986, *The Theory of Database Concurrency Control*, Computer Science Press.

[REED79] Reed D., et al, 1979, "Synchronization with Eventcounts and Sequencers," *Communications of the ACM*, Vol. 22, #2.

[RUBI90] Rubinovitz, H., and B. Thuraisingham, August 1990, *Secure Distributed Query Processor: An Overview*, MTR 10969, Vol. 1 (a version accepted for publication in the *Journal of Systems and Software*).

[RUBI92a] Rubinovitz, H., and B. Thuraisingham, 1992, *Design and Simulation of Query Processing and Concurrency Control Algorithms for a Trusted Distributed Database Management System*, Technical Report MTR 92B0000077, The MITRE Corporation, Bedford, MA.

[RUBI92b] Rubinovitz, H., and B. Thuraisingham, 1992, "Design and Simulation of Secure Distributed Concurrency Control Algorithms," Proceedings of the 1992 Computer Simulation Conference, Reno, NV.

[SHET90] Sheth, A., and J. Larson, September 1990, " Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases," *ACM Computing Surveys*, Vol. 22, #3.

[STAC90] Stachour, P., and B. Thuraisingham, June 1990, "Design of LDV - a Multilevel Secure Database Management System," *IEEE Transaction on Knowledge and Data Engineering*, Vol. 2, #2.

[THUR89] Thuraisingham, B., 1989, *Research Directions in Trusted Distributed Database Management Systems*, Technical Report, M89-52, Volume 2, The MITRE Corporation, Bedford, MA (not currently in public domain).

[THUR90a] Thuraisingham, B., July 1990, *Multilevel Security Issues for Distributed Database Management Systems*, Technical Report, MTP 291, The MITRE Corporation, Bedford, MA (a version also published in *Computers and Security Journal*, 1991).

[THUR90b] Thuraisingham, B., July 1990, *Issues on Developing a Multilevel Object-Oriented Data Model*, Technical Report, MTP 384, The MITRE Corporation, Bedford, MA (a version also published in the *Journal of Object-Oriented Programming*, 1991).

[THUR92a] Thuraisingham, B., and H. Rubinovitz, 1992, "Multilevel Security Issues for Distributed Database Management Systems - III," *Computers and Security Journal.*, Vol. 11.

[THUR92b] Thuraisingham, B., October 1992, "A Note on the Multilevel Security Impact on Real-time Database Management Systems," Presented at the 5th Rome Laboratory Database Security Workshop, Fredonia, NY.

[THUR92c] Thuraisingham, B. and H. Ko, September 1992, *Concurrency Control in Trusted Database Management Systems*, Technical Report, M92B0000109, The MITRE Corporation, Bedford, MA.