Towards a Multilevel Secure Database Management System for Real-Time Applications

Sang H. Son†
Department of Computer Science
University of Virginia
Charlottesville, VA 22903

Bhavani Thuraisingham
The MITRE Corporation
Bedford, MA 01730

ABSTRACT

Database systems for real-time applications must satisfy timing constraints associated with transactions, in addition to maintaining data consistency. In addition to real-time requirements, security is usually required in many applications, because sensitive information must be safeguarded. Multilevel security requirements introduce a new dimension to transaction processing in real-time database systems. This paper addresses issues that must be investigated in order to design and develop a multilevel secure database management system for real-time applications.

1. INTRODUCTION

A real-time database management system (RTDBMS) is a transaction processing system where transactions have explicit timing constraints. Typically, a timing constraint is expressed in the form of a deadline, a certain time in the future by which a transaction needs to be completed. A deadline is said to be hard if it cannot be missed or else the result is useless. If a deadline can be missed, it is a soft deadline. With soft deadlines, the usefulness of a result may decrease after the deadline is missed. In RTDBMS, the correctness of transaction processing depends not only on maintaining consistency constraints and producing correct results, but also on the time at which a transaction is completed. Transactions must be scheduled and processed in such a way that they can be completed before their corresponding deadlines expire. Real-time database systems are being used for a variety of applications such as process control, mission critical applications in command and control systems and radar systems, computer integrated manufacturing systems, and air traffic control systems, among others.

Conventional data models and databases are not adequate for time-critical applications, since they are not designed to provide features required to support real-time transactions. They are designed to provide good average performance, while possibly yielding unacceptable worst-case response times. Very few of them allow users to specify or ensure timing constraints. During the last few years, several research and development efforts have been reported on RTDBMSs [SIGM88, HUAN91, KIM93, SHA91, SON90, SON91, SON92, SON92b, SON93, SON93b].

While the advances were being made in the area of RTDBMS, during the past decade, much work has been carried out on the design and development of a multilevel secure database management system (MLS/DBMS), mainly based on the relational data model (see, for example, [AFSB83, DENN87, STAC90]). As a result, at present, some of these MLS/DBMSs are commercially available. In these database systems, users cleared at different levels are expected to access and share a database consisting of data at different sensitivity levels. The MLS/DBMS must ensure...
that users obtain data classified at or below their security levels.

Most of these MLS/DBMSs are expected to be used for C3I applications. In principle, however, any database system that maintains sensitive information to be shared by multiple users with different levels of security clearance requires multilevel security. Many of these applications also require RTDBMSs. As more and more of such systems are in use, one cannot avoid the need for integrating them. That is, the RTDBMSs have to be made multilevel secure and the MLS/DBMSs need to support real-time applications. To our knowledge no work has been reported on developing DBMSs which are multilevel secure and which support real-time applications. Recently, at the fifth Rome Laboratory Workshop in Database Security, we presented a note on the security impact on real-time database management systems [THUR92]. Our focus was on the security impact on transaction processing. In this paper, we review the security impact on real-time transaction processing as well as discuss architectural issues and data modeling issues to support real-time applications which are also multilevel.

In section 2 we discuss the security impact on real-time transaction processing. In section 3 we discuss the impact of timing requirements on the architectures proposed for MLS/DBMSs. Data model issues are discussed in section 4. The paper is concluded in section 5.

2. TRANSACTION PROCESSING

Timing constraints have been found to have conflicts with consistency requirements. As a result, various transaction processing algorithms have been adapted to support real-time applications [HUA91, SHA91, SON91, SON92]. Similarly, since security constraints have also been found to have conflicts with consistency requirements, various transaction processing algorithms have been adapted for a multilevel environment [THUR90]. When transactions have to satisfy timing, consistency, and security requirements, then there are many conflicts that have to be resolved. The impact of these requirements on transaction characterization, transaction scheduling, concurrency control, and buffer management, as discussed in [THUR92], are given below.

Transaction Characterization: In addition to properties such as atomicity, consistency, isolation, durability, deadlines, and priority, transactions in a multilevel environment have additional properties such as security levels and levels of execution. That is, each transaction is assigned a security level and it usually executes at a level which dominates its security level. For some applications, a transaction may have to change its level during execution. Many issues need to be resolved. For example, should a higher level transaction have higher priority over a lower level transaction or is the transaction priority independent of the security level?

Transaction Scheduling: In a non-real-time MLS/DBMS, scheduling algorithms are being designed in such a way that whenever there is contention for a resource between transactions at different security levels, then the one at the lower level has the priority. However, in a real-time environment, transactions have deadlines and priorities assigned to them. These priorities may interfere with the security levels. The question is, what is the best way to schedule transactions which not only ensures that the deadlines are met, but also guarantees that higher level transactions do not interfere with lower level ones? Is it at all possible to design such scheduling algorithms or should some compromise be made between security and timeliness?

Concurrency Control: This is closely related to transaction scheduling. In a non-real-time MLS/DBMS, algorithms such as locking, timestamping, and validation are being adapted to function in such a way that higher level transactions do not interfere with lower level ones. The question is, what is the impact of deadlines on these algorithms? We illustrate the essential points with an example.

One of the solutions that is being proposed for concurrency control in an MLS/DBMS is for transactions at different levels to work with multiple copies of a data item. Suppose 0 is an Unclassified data item. Two copies of 0 are maintained at the Unclassified level, 01 for Unclassified transactions and 02 for higher level (i.e., Confidential, Secret, and Topsecret) transactions. Suppose a Secret transaction requests a read operation on 02 and an Unclassified transaction is already writing into 01. Then the Secret
transaction must wait until the Unclassified transaction completes the write operation and subsequently creates a new version of the modified object for the high level transactions to read. In a real-time environment, the Secret transaction could be a critical one and waiting for the Unclassified transaction to finish could result in a catastrophic situation. If on the other hand the Unclassified transaction’s write request is aborted, then the higher level transaction has interfered and a security violation occurs. If the Secret transaction is allowed to read O2 immediately while the Unclassified transaction is writing into O1, then there is a consistency problem as the Secret transaction is reading an older version.

Buffer Management: Different buffer management algorithms are being proposed for MLS/DBMSs. That is, buffers must be allocated in such a way that higher level transactions do not signal information to lower level ones by resource exhaustion. Usually lower level transactions are allocated buffers in such a way that the presence of higher level transactions are not of any concern. However, in a real-time environment, if a higher level transaction is more critical, then it must be given higher priority. As a result, a lower level transaction may have to wait for buffers. This could result in a security violation.

Recovery: Research on recovery techniques for MLS/DBMSs is only just beginning. Work needs to be directed towards incorporating timing constraints and deadlines of transactions into such recovery algorithms. In a non-real-time environment, transactions are recovered in such a way that the presence of higher level transactions do not interfere with lower level ones. In a real-time environment, the higher level transaction may have higher priority and may have to be recovered first. This could result in a security violation.

3. ARCHITECTURAL ISSUES

Various architectures have been proposed for an MLS/DBMS. These include operating system providing mandatory security and trusted subject-based architecture. In the first architecture, all of the mandatory access control is provided by the operating system. Therefore, the multilevel database data must be stored in single-level operating system files. Access to these files is controlled by the operating system. When a user at level L queries for the data, the MLS/DBMS must combine the data in the single-level files which are at or below the user’s level and give the response to the user. If a query, which could be issued as part of a transaction, has a timing deadline, then the recombination algorithm has to be adapted accordingly. It has been noted that the recombination algorithms have an impact on performance. Therefore, the suitability of such an architecture for real-time applications needs to be investigated.

In the trusted subject-based architecture, the database data is stored in operating system files. Access to these files is controlled by the operating system. However, the DBMS controls access to its own objects. For example, Unclassified, Confidential, and Secret tuples could be stored in a Secret file. The operating system controls access to the Secret file. A trusted module of the DBMS examines the tuples in the file and releases the tuples depending on the user’s level. In this architecture it is not necessary to recombine single-level tuples into a multilevel relation. As a result it is expected to give better performance. Therefore, it may be more appropriate for real-time applications. However, since the DBMS is trusted, it may be difficult to evaluate such a system.

One of the approaches to meeting the timing constraints in both architectures is to assign priority to the data. If the query is time critical, then only the high priority data could be considered in the response.

4. DATA MODEL ISSUES

Most of the MLS/DBMSs that are being developed are based on the relational data model. Recently, some work is being carried out on designing MLS/DBMSs which are based on an object-oriented data model (see for example [THUR91]). From a preliminary analysis, it appears that the issues for supporting real-time applications are similar for both types of MLS/DBMSs. Recently, object-oriented database systems are becoming popular for real-time applications [OOPS92]. It is claimed that the object-oriented data model has several advantages over the relational model in terms of better modeling
capability for complex real-world data and the encapsulation mechanisms that allow easy integration of constraints [WOLF93]. If such applications require multilevel security, then an object-oriented DBMS which supports such applications needs to be made multilevel secure.

Regardless of whether the data model is relational or object-oriented, the question is, should one consider incorporating multilevel security into an RTDBMS or adapt an MLS/DBMS to support real-time applications?

5. CONCLUSIONS

In summary, timing constraints, consistency constraints, and security conflict with one another. It is difficult to reconcile these conflicting requirements. Even in non-real-time systems, it is difficult to achieve a balance between security and consistency. For example, in one of the solutions to concurrency control that we described earlier, multiple copies of a data item need to be maintained. This means that storage management becomes an issue. In non-multilevel real-time systems, one needs to develop techniques that ensure timeliness as well and consistency. In a multilevel real-time environment, techniques have to be developed which ensure security, timeliness, and consistency. Due to the complexities involved, we envisage that trade-offs need to be made on a case-by-case basis. That is, various types of Multilevel Secure Real-time DBMSs (MLS/RTDBMS) need to be identified and architectures and algorithms need to be developed for each type of system.

REFERENCES


[SON92b] Son, S. H., J. Lee, and Y. Lin, "Hybrid Protocols using Dynamic


