Partial Security and Timeliness in Real-Time Database Systems

Sang H. Son

Department of Computer Science
University of Virginia
Charlottesville, VA 22903
son@cs.virginia.edu

A real-time system is one whose basic specification and design correctness arguments must include its ability to meet its timing constraints. This implies that its correctness depends not only on the logical correctness, but also on the timeliness of its actions. Typically, real-time systems are associated with critical applications, in which human lives or expensive machinery may be at stake. Their missions are often long-lived and non-interruptible, making maintenance or reconfiguration difficult. Although it is commonly believed that meeting the timing requirements is a matter of increasing system throughput sufficiently, research in real-time systems has discredited this notion.

As real-time systems continue to evolve, their applications become more complex, and often require timely access to—and predictable processing of—massive amounts of data. This need for advanced data management functionalities in real-time systems poses formidable intellectual and engineering challenges that must be tackled to allow for practical solutions to the problems faced in design and development of complex *real-time database systems*.

The importance of real-time database systems in an increasing number of applications, such as aerospace and defense systems, industrial automation, business information systems, traffic control, and telecommunication has resulted in an increased research effort in this area. In many of these applications, security is another important requirement, since the system maintains sensitive information to be shared by multiple users with different levels of security clearance. As more and more of such systems are in use, one cannot avoid the need for integrating them. That is, real-time systems have to be made acceptably secure and the secure systems need to support the timeliness requirements of real-time applications.

For example, electronic commerce becomes an interesting application where both security and real-time requirements should be considered together. Security is critical in electronic commerce applications where the system often manages sensitive information (such as credit card numbers). A real-time database is a critical infrastructure that is essential to support complex and flexible services to manage requests in the context of highly dynamic workload with diverged requirements. Service providers are usually very sensitive to the needs of the clients, including the implicit and explicit timing constraints. If providing full security to each activity causes the system to become less timely (e.g., missing deadlines), the service providers would be reluctant in providing necessary

security guarantees.

Traditionally, the notion of security has been considered binary [4]. A system is either secure or not. A security hole either exists or not. The problem with such binary notion of security is that in many cases, it is critical to develop a system that provides an acceptable level of security and risks, based on the notion of partial security rather than unconditional absolute security, to satisfy other conflicting requirements such as real-time. In that regard, it is important to consider multiple security service levels for real-time database applications. To achieve that, we need to define the meaning of partial security, for security violations of sensitive data must be strictly controlled, while the cost of providing that level of service should not reduce the timeliness of the system.

To improve the timeliness and security in such applications, several issues need to be carefully considered. The research in this area has been focused in identifying architectural and transaction processing issues. It is clear that a more rigorous model to capture the characteristics and semantics of transactions and data in secure real-time databases is necessary for efficient processing to improve the timeliness of the system. In addition, new approaches to supporting both requirements in transaction scheduling and concurrency control that can make trade-offs if necessary, need to be developed and analyzed.

Because sensitive information must be safeguarded, the scheduling algorithms should consider timeliness as well as security requirements in their scheduling decisions. Security is concerned with the ability of a system to enforce a certain policy governing the use, modification, and destruction of information. There are two different policies that have been studied: discretionary security policy and mandatory (or multilevel) security policy. Discretionary security policies define access restrictions based on the identity of users, the type of access, and objects being accessed. While discretionary access control has been used in several systems, it may not be adequate for preventing unauthorized disclosure of the information. Recently, many systems are developed based on multilevel security policy such as the Bell-LaPadula model [1]. In the Bell-LaPadula model, security policies are stated in terms of subjects and objects. A subject is an active entity that can access objects. Every object is assigned a classification, and every subject a clearance. Classifications and clearances are collectively referred to as security classes (or levels) and they are partially ordered. Database systems that support the Bell-LaPadula properties are called multilevel secure database systems.

The Bell-LaPadula model prevents direct flow of information from a higher access class to a lower access class, but the conditions are not sufficient to ensure that security is not violated indirectly through what are known as covert channels [2]. A covert channel allows indirect transfer of information from a subject at a higher access class to a subject at a lower access class. An important class of covert channels that are usually associated with concurrency control mechanisms are timing channels. A timing channel arises when a resource or object in the database is shared between subjects with different access classes. The two subjects can cooperate with each other to transfer information.

From our earlier study, it became clear that security requirements are not compatible with real-time requirements [3]. Frequently, priority inversion is necessary to avoid covert channels. Consider a transaction with a high security level and a high priority entering the database, and it finds that a transaction with a lower security level and a lower priority holds a write lock on a data item that it needs to access. If the system preempts the lower priority transaction to allow the higher priority transaction to execute, the principle of non-interference is violated, for the presence of a high security transaction affects the execution of a lower security transaction. On the other hand, if the system delays the high priority transaction, a priority inversion occurs. The system has encountered an unresolvable conflict. In general, these unresolvable conflicts occur when two transactions contend for the same resource, with one transaction having both a higher security level and a higher priority level than the other. Therefore, creating a database that is completely secure and strictly meets real-time requirements is not feasible. A system that wishes to accomplish the fusion of multi-level security and real-time requirements must make some concessions at times. In some situations, priority inversions might be allowed to protect the security of the system. In other situations, the system might allow potential covert channels so that transactions can meet their deadlines.

An important but challenging problem to be addressed in supporting security and real-time is identifying the correct metrics to evaluate the level of security obtained in an adaptable system. When the system has to trade-off security, it is important to define the exact meaning of partial security. One approach is to define security in terms of a percentage of security violations allowed in the system. However, system designers could argue about the usefulness of this metric. Even though a system may allow a very low percentage of security violations, this fact alone reveals nothing about the security of individual data. For example, a system might achieve 99% security level, but that 1% of insecurity might allow the most sensitive piece of data to leak out. For serious security applications, a more precise metric would be necessary.

A better approach involves adapting the Bell-LaPadula security model and blurring boundaries between security levels in order to allow partial security. In this scheme, violations only between certain security levels would be allowed. As the real-time performance of the system degrades, more and more boundaries can be blurred, allowing more security violations and reducing the number of security conflicts. Since there are less conflicts, this can improve the real-time performance of the system. Additionally, with this scheme, we can still make guarantees about the security of the sensitive data.

For example, consider a system with four security levels: top secret, secret, confidential, and unclassified. Partial security policies can be specified by the level of security guarantees provided, from fully secure to completely insecure. For instance, level-4 security guarantee indicates full security. Potential security violations are allowed between certain levels as guarantee levels are lowered. For level-3 security guarantee, transactions that are at the unclassified level may have conflicts with transactions at the confidential level in accessing to unclassified

data, resulting in a potential covert channel. In a sense, this represents a system with only three security levels: top secret, secret, and unclassified. However, it does not mean that there is no distinction between confidential and unclassified levels. Transactions that are at the unclassified level cannot directly access confidential data.

It is possible to combine this approach with the use of percentages to define partial security. Then, the amount of security violations between two levels for which the boundary had been blurred would be required to fall below this percentage. The above example is in fact a special case of this scheme, where the percentage can either be 0% or 100%. Note that no guarantees can be made between levels that have been assigned a non-zero percentage. Guarantees can still be made between levels designated as allowing 0% security violations; for the other levels, database designers can use different percentages to denote their preferences on where they would rather have the potential security violations occur.

Application designers should be able to specify semantic information using a specification language to express the relative importance of keeping desired level of security and meeting the timeliness requirement. A question to be addressed in that approach is the verification of the given specification. Specifications should be compiled and verified to check any inconsistency in the requirements and to clearly determine the necessary actions to be taken. We need a specification language that could be used to allow the designer to specify the database semantics and real-time/security requirements. The language should be able to support the designer in specifying rules at varying levels of detail to be referred at run-time to resolve the conflicts. Once the specification is given, it needs to be analyzed to identify any inconsistency and to generate rules based on the semantic information on data and transactions to be used at run-time. We are currently looking into the issues involved in specification languages and tools to check the completeness and consistency among the rules at different levels.

References

1. Bell, D. E. and L. J. LaPadula, Secure Computer Systems: Unified Exposition and Multics Interpretation, *Tech. Report*, The Mitre Corp., March 1976.
2. Lampson, B. W., A Note on the Confinement Problem, *Communications of the ACM*, 16(10), pp 613-615, October 1973.
3. Son, S. H. and R. David, Synthesis of Multilevel Security and Timing Requirements in Complex Real-Time Database Systems, *Complex Systems Engineering Synthesis and Assessment Technology Workshop*, Washington, DC, July 1994.
4. Ting, T. C., How Secure is Secure? Opening Remarks, *IFIP WG 11.3 Working conference on Database Security*, Rensselaerville, New York, Aug. 1995.

This article was processed using the LaTeX macro package with LLNCS style