Semantic-Based Concurrency Control for Object-Oriented Database Systems Supporting Real-Time Applications

Juhnyoung Lee and *Sang H. Son*

Department of Computer Science
University of Virginia
Charlottesville, VA 22903, USA
email{jl2q,son}@cs.virginia.edu
fax(804)982-2214

Abstract

Recently object-oriented database technology has started to gain the attention of researchers in real-time database systems. This paper investigates major issues in designing semantic-based concurrency control for object-orient database systems supporting real-time applications, and describes an approach to solving those problems in an efficient way. This approach depends on the notion of affected-set *of operations to determine operation compatibilities, and employs concurrency control algorithms augmented with priority-based conflict resolution schemes. With this method, it is impossible to maneuver the trade-off between logical and temporal consistency constraints in determining operation compatibilities, because the method deals with the two separately in different dimensions. However, this approach significantly lessens the complexity of the compatibility relation construction process, and can easily apply results from previous research on semantic-based concurrency control for object-oriented databases and on priority-based concurrency control for real-time database systems.*

1. Introduction

A real-time database system (RTDBS) is a transaction processing system where transactions have explicit timing constraints. Most work on concurrency control in real-time database systems has been done on the basis of relational data models [Son92]. Recently, however, object-oriented data models have attracted the attention of researchers in RTDBSs [Lort93, Di93]. The motivation of the researchers is to bring to bear the benefits of object-oriented database technology in designing real-time database systems. An object-oriented database differs from relational databases since information is maintained in terms of *classes* and *instances* of these classes. Both classes and instances are referred to as *objects*. Classes define both attributes and the procedures through which instances can be manipulated. The procedures associated with a class are referred to as *methods*, and a method may invoke other methods on other objects in the database. This model of execution generalizes the classical model of transaction processing by permitting *arbitrary operations* on objects as opposed to traditional read and write operations, and by permitting *nested transactions* as opposed to flat transactions.

Two major benefits of object-oriented data models are [Zdo90]: (1) better support of advanced data-intensive applications such as design databases, multimedia databases, and knowledge bases mainly by providing the capabilities for modeling, storing and manipulating *complex objects*, and (2) better software engineering in building large application systems that consist of reusable modules, mainly by providing support for *encapsulated objects*, i.e., instances of abstract data types. The need for supporting real-time database requirements with object-oriented data models may arise because real-time applications may require modeling complex encapsulated objects.

The design of an RTDBS based on an object-oriented data model presents a number of new and challenging problems. In this paper, we focus on concurrency control for real-time object-oriented database systems. In particular, we study techniques to improve concurrency of transactions executing on data objects through the use of semantic information of operations defined on objects. Such techniques are referred to as *semantic-based concurrency control*.

In general, object-oriented data models provide greater opportunities for supporting semantic-based concurrency control than traditional database models for a variety of reasons [Agra92, Di93, Muth93]. First, the capability of including user-defined operations of arbitrary complexity in data object representation provides greater semantic information about the operations that can be exploited for concurrency control. Second, because of the encapsulation mechanism of object-oriented models, operations defined on a data object provide the only means to access the object's data. Thus, data contention can occur only among operation invocations within the object. This characteristic of object-oriented data models provides greater flexibility for concurrency control in that it allows concurrency control specific to individual data objects. Finally, improved capability of handling constraints in object-oriented data models helps concurrency control. In particular, the capability of handling both logical and temporal consistency constraints in a uniform manner is beneficial for concurrency control in RTDBSs.

We should also note that in some sense, the performance improvement by using semantic information is something required to be done in object-oriented databases. Because data objects in object-oriented databases are often large and complex,

This work was supported in part by ONR, Loral and CIT.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
1994		**00-00-1994 to 00-00-1994**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Semantic-Based Concurrency Control for Object-Oriented Database Systems Supporting Real-Time Applications	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of Virginia,Department of Computer Science,151 Engineer's Way,Cahrlottesville,VA,22094-4740**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**6**	

locking an entire data object for concurrency control (or transaction validation for accesses to an entire data object in case of optimistic concurrency control) may be inefficient and unnecessary. If semantics of operations are taken into account in concurrency control, the necessity of locking the entire data object vanishes, and multiple operations that do not affect the same part of an object can potentially execute concurrently without violating the logical consistency constraints of the object.

The remainder of this paper is organized in the following fashion: Section 2 examines three major issues in designing semantic-based concurrency control for real-time database systems based on an object-oriented data model. A brief survey of related work is presented in Section 3. Section 4 describes an approach to designing semantic-based concurrency control for real-time object-oriented database systems. Finally Section 5 summarizes the main conclusions of this study and outlines future work.

2. Design Issues

Even without considering semantic information and timing constraints, concurrency control in object-oriented database systems is a more difficult problem than that in relational database systems. Object-oriented data models have inherently inefficient nature to ensure data consistency because objects of arbitrary complexity are the unit of locks and sometimes concurrency control requires to lock not only the object accessed by a transaction, but also several other objects not directly accessed by the transaction. The latter problem is mainly caused by class hierarchy in object-oriented data models. It was pointed out that (1) while a transaction accesses instances of a class, another transaction should not able to modify the definition of any of the super classes of the class, and (2) while a transaction is evaluating a query, a set of class sub-hierarchies must not be modified by a conflicting transaction [Kim90].

Use of complete semantic information helps attain higher concurrency. However, the expected performance gain does not come for free. In general, the use of semantic information in an object-oriented data model increases the complexity of concurrency control for a variety of reasons. The additional complexity may arise due to complex domain of interpretation for various operations, and possibility of using different concurrency control mechanisms specific to each object. In an RTDBS where concurrency control has to consider not only logical but also temporal consistency constraints, the concurrency control mechanism may become even more complex. In designing a semantic-based concurrency control mechanism for a real-time object-oriented database system, we need to consider the following three major issues: detection of compatibility relation of operations defined on each object, determination of an appropriate concurrency control algorithm for each object and system, and support of logical data consistency across objects in the system.

2.1. Compatibility Relation

In implementing a semantic-based concurrency control mechanism, semantics of operations in each object needs to be interpreted to determine allowable interleaving of each pair of operation invocations. For a complete use of semantic information, in general, this process has to consider various domains of semantic information, including the set of attributes affected by a particular operation invocation, and the parameters of operation invocations. In particular, in an RTDBS, temporal consistency constraints imposed on data objects and transactions may also need to be considered in the process. Furthermore, the process may become even more complex, when the system allows relaxation of serializability as the correctness criterion for logical data consistency. Due to the variety of domains of semantic information, the process of building compatibility relation tends to be application dependent. An *ad hoc* process of detecting operation compatibilities often places a heavy burden on application programmers to define the semantics of operations.

Another question related to the process of compatibility relation detection is whether a compatibility relationship of an operation to other operations in an object is determined statically when the object is created or dynamically when the operation is requested to be executed on the object. In case of the static policy, a compatibility relation is defined for an object in the form of a table when the object is created. Alternatively, the compatibility relation can be defined by a compatibility function, where compatibility is expressed as a run-time function. In general, improved concurrency may be expected when the compatibility relation is determined dynamically, because we can exploit information about the current state of the object, and actual parameters of the operations as well as other information.

2.2. Concurrency Control Algorithms

Once the compatibility relation of an object is given, an easy way to implement a concurrency control mechanism of the object is to use a simple concurrency control protocol such as two-phase locking which controls concurrent data operations by using the given compatibility relation to satisfy consistency constraints. However, because the encapsulation capability of object-oriented data models allows different concurrency control algorithms to be used for different data objects, as we explained, we may expect performance improved further by choosing for each object a customized concurrency control algorithm efficient for its operating characteristics, resource availability and workload level in the system.

Various concurrency control algorithms basically differ in two aspects: the time when they detect conflicts and the way that they resolve conflicts. Locking [Esw76] and optimistic concurrency control [Kung81] in their basic form represent the two extremes in terms of these two aspects. Locking detects conflicts as soon as they occur and resolves them using blocking. Optimistic scheme detects conflicts only at transaction commit time and resolves them using restarts.

The impact of those differences in concurrency control algorithms on performance has been the major theme in the performance study of concurrency control in both conventional and real-time database systems. The results from these studies can be summarized as follows: In conventional database systems, locking algorithms that resolve data conflicts by blocking transactions outperforms restart-oriented algorithms, especially

when physical resources are limited. If resource utilizations are low enough so that a large amount of wasted resources can be tolerated, and there are large number of transactions available to execute, then a restart-oriented algorithm such as optimistic concurrency control that allows higher concurrency is a better choice [Bern87]. In addition, the delayed conflict resolution of optimistic approach helps in making better decisions in conflict resolution, since more information about conflicting transactions is available at this later stage. On the other hand, the immediate conflict resolution policy of locking may lead to useless restarts and blocking in RTDBSs due to its lack of information on conflicting transactions at the time of conflict resolution [Hari92].

2.3. Inter-Object Data Consistency

In an object-oriented database system, concurrency control mechanisms defined in objects have to resolve inter-object inconsistency as well as satisfy local data consistency within each object. Transactions, in an object-oriented database system, consist of a set of operation invocations on one or more objects. Thus, transactions may read and write inconsistent data across objects, when the serialization order among transactions locally determined in one object does not agree with one built in another object. Without keeping information about the status of transactions executing across objects, local concurrency control mechanisms in individual objects have no way to recognize and resolve inter-object data inconsistency.

The problem of maintaining inter-object data consistency in an object-oriented database system is analogous to the problem of satisfying global data consistency in a distributed database system. Thus, concurrency control in an object-oriented database system can be made to support inter-object data consistency with similar methods used for distributed database environments. For instance, lock managers in locking protocols and validation of transactions in optimistic algorithms, need to take into account the serialization order determined across objects. There are several possible schemes for implementing such extensions to concurrency control algorithms, including the *single coordinator approach* in which the system maintains one single coordinator that resides in one single chosen object, and the *multiple coordinator approach* in which the coordinator function is distributed over several objects. Each coordinator knows the names of all the participating objects, and sends them messages to inform status of executing transactions.

As we explained, object-oriented data models provide opportunities to improve performance by using different local concurrency control mechanisms and data consistency correctness criteria specific to individual objects. We should note, however, that this benefit is mitigated by the problem of inter-object data inconsistency. Allowing object-specific concurrency control mechanisms makes the problem of inter-object inconsistency more complex, as the situation becomes analogous to one in a distributed database system consisting of heterogeneous components in different sites. In particular, when various correctness criteria are allowed for different objects, locally consistent data within an object using one correctness criterion may be inconsistent externally with data objects using different correctness criteria. In addition, if various concurrency control protocols are allowed for different objects, we need a more complex mechanism that understands and maintains the serialization order enforced by different local concurrency control algorithms to support inter-object data consistency.

3. Related Work

The problem of concurrency control for object-oriented database systems has not yet been solved effectively. There are two major approaches to the problem: (1) exploiting the structure of complex objects for enhanced concurrency or reduced overhead, and (2) exploiting the semantics of operations on encapsulated objects to enhance concurrency. Examples of approach (1) include the concurrency control mechanisms of Orion [Gar88] and O_2 systems [Cart90]. Orion uses locking on three orthogonal types of hierarchy: granularity locking (to minimize the number of locks to be set), class-lattice locking (to handle class hierarchy), and composite object locking (to handle object clusters) [Gar88]. The eight lock modes used in Orion requires a complex lock compatibility table without considering operation semantics and temporal consistency.

Approach (2) is related to work on concurrency control for abstract data types, and the use of fine and *ad hoc* commutativity relation of operations [Herl88, Schw84, Weih88]. Class hierarchy also introduces problems here, since two methods with the same name may have distinct properties of commutativity due to inheritance. Researchers in this approach attempt to solve such problems caused by class hierarchy using the notion of nested transactions. Examples of previous work using this approach include [Agra92, Muth93].

Semantic-based concurrency control for real-time object-oriented database systems has been studied only recently since object-oriented data models are adopted for supporting real-time applications. The work in [Di93] provides a comprehensive view over logical and temporal consistency constraints in designing a real-time object-oriented database system. To determine compatibility relation of operations, their approach considers not only a broad domain of semantic information affecting logical consistency, but also temporal consistency constraints. In addition, the approach allows a wide range of correctness criteria for logical consistency that relax serializability. Once a compatibility relation of operations is determined in the form of run-time compatibility functions augmented with predicates used for trade-off between temporal and logical consistency, a locking protocol is used uniformly for all objects in the system.

The approach in [Di93] is comprehensive and flexible. One significant drawback is in its complexity; in general, there is no efficient way for automating the process of detecting operation compatibilities because logical and temporal constraints are often semantically unrelated. An *ad hoc* process of compatibility relation construction with such complexity will be a serious burden on object designers. The other problem of their approach is the possibility that the variety of allowable correctness criteria for logical consistency causes inter-object inconsistency. This issue was not addressed in [Di93].

4. Our Approach

A major objective of this study is to investigate a possible approach to designing semantic-based concurrency control for a real-time object-oriented database system that alleviates the complexity associated with the design procedure, especially with the step of detecting compatibility relation. By doing so, we expect to provide a semantic-based concurrency control mechanism easy to implement, and to provide a general method for systematically building compatibility relation of operations. In addition, the other goal of this study is to apply the results from previous research on priority-driven concurrency control for real-time database systems as well as semantic-based concurrency control for object-oriented databases in designing semantic-based concurrency control for real-time object-oriented database systems.

4.1. A Simple Concurrency Control Model

To achieve the goals of this study, we made the following decisions on our object-oriented database model. First, we decided to use only serializability as the correctness criterion for logical data consistency in the system. Although there are studies on relaxation of serializability that presented correctness criteria such as epsilon-serializability [Rama91] and similarity-based correctness [Kuo92], at present there is no general purpose correctness criterion as easily understandable and implementable as serializability. Second, we decided to use only one concurrency control protocol uniformly for all of the objects in the system. Note that this decision does not require any particular concurrency control algorithm for the system. The concurrency control algorithm may be either locking or optimistic concurrency control. The choice of the concurrency control algorithm will be decided on the basis of resource availability and workload level of the system. Finally, we decided not to consider temporal consistency constraints in the process of determining compatibility relation of operations, to simplify and systematically perform the process. Timing constraints imposed on data objects and transactions are taken into account only in concurrency control protocols, which use priority and deadline information in resolving data conflicts, i.e., resolve data conflicts in favor of more urgent operation with higher priority. A number of studies on concurrency control for RTDBSs presented various algorithms with such capability and evaluated their performance [Abbo88, Hari92, Huan89, Lee93].

These decisions will considerably lessen the complexity in designing semantic-based concurrency control for real-time object-oriented database systems. In particular, the first two simplify the problem of inter-object inconsistency, and the last alleviates the complexity in compatibility relation construction. Also, with this approach, we can easily take advantage of the results from previous research on semantic-based concurrency control for object-oriented databases and priority-driven concurrency control. One disadvantage of this approach is that we cannot maneuver the trade-off between logical and temporal consistency constraints in determining operation compatibilities, because the approach deals with the two separately in different dimensions. In the following, we briefly describe a technique for determining a compatibility relation of operations in an objects, and two representative concurrency control algorithms augmented with priority-driven conflict resolution, one is based on locking and the other on optimistic concurrency control.

4.2. Compatibility Relation by Affected-Set

To determine operation compatibilities, we consider only logical consistency constraints of objects, and uses the notion of an *affected-set* originally presented in [Badr88]. When an operation is invoked, the set of attributes affected by the particular invocation is computed. The affected-set must be computed for each invocation of an operation because the affected attributes may depend on the actual arguments of the operation. In general, the affected-set of each operation is constructed from its semantic specification. We adopt *commutativity* as the basis for determining whether a particular operation invocation can be allowed to execute concurrently with those in progress. Two operations *commute* if the order in which they execute does not affect the results of the operations, i.e., results returned by the operation as well as the resulting state of the objects accessed. Two operations from different transactions commute if the affected-sets of the respective operations are non-intersecting.

We limit members of an operation's affected-set to be all attributes directly accessed by the operation and any attributes accessed by enforcement rules that may be triggered by the operation. We divide an affected-set into two subsets; one contains attributes only observed by the operation without their value being updated, and the other includes attributes modified by the operation. Those subsets are referred to as *observed-set* and *modified-set*, respectively. By checking the intersection of their observed-sets and modified-sets, the compatibility relation between a pair of operation invocations, OI_1, and OI_2 in an object is determined. In general, OI_1 and OI_2 are incompatible if at least one of observed-set(OI_1) $\cap$ modified-set(OI_2), modified-set(OI_1) $\cap$ observed-set(OI_2), and modified-set(OI_1) $\cap$ modified-set(OI_2) is not empty.

With this technique, the designer of an object type only needs to specify the semantics of operations; their compatibilities are systematically determined from these specifications. Another advantage of this approach is that compatibility of operations may be determined dynamically when the operation is requested to be executed on an object.

4.3. Real-Time Concurrency Control Algorithms

In the classical two-phase locking (2PL) protocol [Bern87], transactions set read locks on objects that they read, and these locks are later upgraded to write locks for the data objects that are updated. If a lock request is denied, the requesting transaction is blocked until the lock is released. Read locks can be shared, while write locks are exclusive. The only difference in our semantic-based concurrency control mechanism is that operations, not transactions, invoked by a transaction set locks on attributes in their affected-sets. An operation sets read locks on attributes in its observed-set, and write locks on attributes in its modified-set. It is important to note that locks set by an operation are not released until the transaction that invoked the operation is completed (or restarted), to ensure the serializable execution of

transactions within objects.

For real-time database systems, two-phase locking needs to be augmented with a priority-based conflict resolution scheme to ensure that higher priority transactions are not delayed by lower priority transactions. In the *High Priority* scheme [Abbo88], all data conflicts are resolved in favor of urgent transactions with high priority. When a transaction requests a lock on an object held by other transactions in a conflicting lock mode, if the requester's priority is higher than that of all the lock holders, the holders are restarted and the requester is granted the lock; if the requester's priority is lower, it waits for the lock holders to release the lock. In addition, a new read lock requester can join a group of read lock holders only if its priority is higher than that of all waiting write lock operations. This protocol is referred to as *2PL-HP* [Abbo88]. Although *2PL-HP* loses some of the basic 2PL algorithm's blocking factor due to the partially restart-based nature of the *High Priority* scheme, it was shown to outperform optimistic schemes under an environment where physical resources are limited [Lee94]. The locking protocol automatically supports the logical data consistency across objects, under the condition that no object is replicated in the system. The local serialization orders on individual objects can be extended to a unique global serialization order without introducing any cycles due to the *High Priority* conflict resolution policy.

In optimistic concurrency control (OCC), transactions are allowed to execute unhindered until they reach their commit point, at which time they are validated. Previous studies on concurrency control for real-time database systems have shown that OCC is well-suited to RTDBSs because of its provision of high degree of concurrency, and its policy of validation stage conflict resolution. In studies [Hari92, Lee94], it was shown that under the condition that tardy transactions are discarded from the system, OCC outperforms locking-based algorithms over a wide range of system workload and resource availability.

Among a number of OCC algorithms designed for RTDBSs, it was shown that *Precise Serialization* (PS) algorithm augmented with *Feasible Sacrifice* (FS) scheme for deadline-sensitive conflict resolution outperforms other algorithms currently known [Lee93]. The PS algorithm is a variant of the *forward validation* scheme [Haer84], so it provides flexibility required for priority-based conflict resolution unlike backward validation, and reduces the wastage of resources and time by detecting and resolving data conflicts earlier. It was shown that forward validation incurs transaction restarts not absolutely necessary to ensure data consistency. The PS algorithm solves the problem with a reasonably limited overhead for dynamic adjustment of transaction execution history. In [Lee93], it was shown that the FS scheme makes intelligent conflict resolution decisions with reasonable estimate of transaction execution time, and that in general, optimistic concurrency control provides a greater opportunities for making useful estimation of transaction execution time with reasonable amount of effort and overhead due to its phase-wise transaction execution. The FS scheme was shown to improve the real-time performance of the PS algorithm further by reducing the number of missed deadlines caused by wasted sacrifices of validating transactions, while giving precedence to more urgent transactions.

To support inter-object data consistency, we can incorporate into an OCC algorithm a *distributed validation scheme* [Thom90], with which a transaction validates at all objects that are involved in its read phase to ensure that validation requests are processed in the same order at all objects. There are a number of possible approaches to implementing such distributed validation schemes, including the use of multi-cast messages or timestamps. Although no study previously done on semantic-based concurrency control has adopted optimistic concurrency control, we think that OCC is promising for semantic-based concurrency control in real-time object-oriented database systems.

5. Conclusions and Future Work

In this paper, we have examined several issues in designing semantic-based concurrency control in an object-oriented database for supporting real-time applications, and presented a semantic-based concurrency control mechanism for such systems. The mechanism uses the notion of affected-set of operations to determine the compatibility relation of operations. Since this process can be systematically performed using the specification of operations, the designer of an object type needs not specify possibly conflicting operations. Also, the mechanism employs concurrency control algorithms that gives precedence to more urgent operations through the use of priority-driven conflict resolution to satisfy timing constraints imposed on transactions.

Our approach to designing semantic-based concurrency control for real-time object-oriented database systems is different from previous work in this area in a number of ways. First, we showed how optimistic concurrency control can be adopted in the proposed semantic-based concurrency control mechanism. We think that optimistic approach has a potential for implementing semantic-based concurrency control in real-time object-oriented database systems. Second, the proposed approach deals with logical consistency constraints and timing requirements separately in two different dimensions. As a result, the complexity associated with the process of determining operation compatibilities is significantly lessened. The price of the reduced complexity is that our approach cannot maneuver the trade-off between logical and temporal constraints in determining operation compatibilities. Finally, our approach can guarantee inter-object consistency with reasonable overhead for keeping information about the current status of transactions executing across objects.

This work can be extended in a number of directions for future research. First of all, it is generally fair to state that there is still a lack of consensus on how object-oriented data models need to be customized to efficiently support real-time applications. We would like to investigate features of object-oriented data models useful for supporting real-time applications, addressing related issues such as associating constraints and triggers with objects, specification of timing constraints as well as other constraints, and transaction management techniques. Second, in the proposed semantic-based concurrency control mechanism, we only considered attributes directly accessed by object operations to detect compatibility relation, and thus the notion of commutativity was not fully utilized. The more semantics we use in concurrency

control, the more concurrency we obtain, although the concurrency control mechanism may become more complex. We plan to examine an approach using more semantics of operations than the current scheme to achieve higher concurrency. Also, we intend to devise a mechanism to automate the process of constructing compatibility relation from object specification. It appears unthinkable to put the burden of determining commutativity for every pair of operation invocations (and providing inverse operations for recovery for every operation invocation) on application programmers or object designer. We consider a method similar to one presented in [Malt93], in which a simple form of commutativity is syntactically extracted from the source codes of operations defined on objects at compile-time, and then transitive access vectors are calculated and translated into classical access modes in order not to incur performance penalty at run-time. Finally, we are currently developing a real-time database system using a multi-user object storage system, EXODUS [Care89], which was developed at the University of Wisconsin. We plan to implement the semantic-based concurrency control mechanism presented in this paper on the system, and evaluate its performance.

REFERENCE

[Abbo88] Abbott, R., and H. Garcia-Molina, "Scheduling Real-Time Transactions: A Performance Evaluation," *Proc. of the 14th VLDB Conference*, August 1988.

[Agra92] Agrawal, D., and A. El Abbadi, "A Non-Restrictive Concurrency Control for Object Oriented Databases," *Proc. of the 3rd Int. Conf. on Extending Data Base Technology*, Vienna, Austria, March 1992.

[Badr88] Badrinath, B. R., and K. Ramamritham, "Synchronizing Transactions on Objects," *IEEE Transactions on Computers*, 37(5), May 1988.

[Bern87] Bernstein, P., V. Hadzilacos, and N. Goodman, *Concurrency Control and Recovery in Database Systems*, Addison-Wesley, Reading, MA., 1987.

[Care89] Carey, M., D. DeWitt, E. Shekita, "Storage Management for Objects in EXODUS," in *Object-Oriented Concepts, Databases, and Applications*, W. Kim and F. Lochovsky (eds.), Addison-Wesley, 1989.

[Cart90] Cart, M., and J. Ferrie, "Integrating Concurrency Control into an Object-Oriented DAtabase System," *Proc. of the Int. Conf. on Extending Data Base Technology*, 1990.

[Di93] Di Pippo, L. B. C., and V. F. Wolfe, "Object-Based Semantic Real-Time Concurrency Control," *Proc. of the 14th IEEE Real-Time System Symposium*, Dec. 1993.

[Esw76] Eswaran, K. P., et al., "The Notion of Consistency and Predicate Locks in a Database System," *Communications of ACM*, 19, November 1976.

[Gar88] Garza, J. F., and W. Kim, "Transaction Management in an Object-Oriented Database System," *ACM SIGMOD International Conference on Management of Data*, June 1988.

[Haer84] Haerder, T., "Observations on Optimistic Concurrency Control Schemes," *Information Systems*, 9(2), 1984.

[Hari92] Haritsa, J. R., M. J. Carey, and M. Livny, "Data Access Scheduling in Firm Real-Time Database Systems," *The Journal of Real-Time Systems*, Kluwer Academic Publishers, 4, 1992.

[Herl88] Herlihy, M. P., and W. E. Weihl, "Hybrid Concurrency Control for Abstract Data Types," *ACM PODS Conference*, 1988, pp. 201-210.

[Huan89] Huang, J., J. A. Stankovic, K. Ramamritham, and D. Towsley, "Experimental Evaluation of Real-Time Transaction Processing," *Proc. of the 10th IEEE Real-Time System Symposium*, Dec. 1989.

[Kim90] Kim, W., "Object-Oriented Databases: Definition and Research Directions," *IEEE Transactions on Knowledge and Data Engineering* Vol. 2 No. 3, September 1990.

[Kung81] Kung, H. T., and J. T. Robinson, "On Optimistic Methods for Concurrency Control," *ACM Transactions on Database Systems*, June 1981.

[Kuo92] Kuo, T.-W., and A. K. Mok, "Application Semantics and Concurrency Control of Real-Time Data-Intensive Applications," *Proc. of the 13th IEEE Real-Time Systems Symposium*, December 1992.

[Lee93] Lee, J., and S. H. Son, "Using Dynamic Adjustment of Serialization for Real-Time Database Systems," *Proc. of the 14th IEEE Real-Time Systems Symposium*, December 1993.

[Lee94] Lee, J., and S. H. Son, "Performance of Concurrency Control Algorithms for Real-Time Database Systems," in V. Kumar (ed.), *Performance of Concurrency Control Mechanisms in Centralized Database Systems*, Prentice Hall, 1994 (to appear).

[Lort93] Lortz, V. B., I. P. Mangiavacchi, and K. G. Shin, "An Object-Oriented Approach to Integrating Real-Time Manufacturing Systems," *Technical Report*, Dept. of Electrical Engineering and Computer Science, Univ. of Michigan, 1993.

[Malt93] Malta, C., and J. Martinez, "Automating Fine Concurrency Control in Object-Oriented Databases," *Proc. of the 9th Int. Conf. on Data Engineering*, Vienna, Austria, April 1993.

[Muth93] Muth, P., T. C. Rakow, G. Weikum, P. Brossler, C. Hasse, "Semantic Concurrency Control in Object-Oriented Database Systems," *Proc. of the 9th Int. Conf. on Data Engineering*, Vienna, Austria, April 19-23, 1993.

[Rama91] Ramamritham, K., and C. Pu, "A Formal Characterization of Epsilon Serializability," *Technical Report CUCS-044-91*, Department of Computer Science, Columbia University, 1991.

[Schw84] Schwartz, P. M., and A. Z. Spector, "Synchronizing Shared Abstract Types," *ACM Transactions on Computer Systems*, 2(3):223-250, 1984.

[Son92] Son, S. H., S. Yannopoulos, Y. K. Kim, and C. Iannacone, "Integration of a Database System with Real-Time Kernel for Time-Critical Applications," *Int. Conf. on System Integration*, June 1992.

[Thom90] Thomasian, A., "A New Distributed Optimistic Concurrency Control Method and a Comparison of its Performance with Two-Phase Locking," *Proc. of the 10th Int. Conf. on Distributed Computing Systems*, Paris, France, June 1990.

[Weih88] Weihl, W., "Commutativity-Based Concurrency Control for Abstract Data Types," *IEEE Transactions on Computers*, 37(12):1488-1505, December 1988.

[Zdo90] Zdonik, S., and D. Maier, *Readings in Object-Oriented Database Systems*, Morgan Kauffman, San Mateo, CA., 1990.