A Non-Restrictive Concurrency Control for Object Oriented Databases*

D. Agrawal and A. El Abbadi

Department of Computer Science, University of California, Santa Barbara, CA 93106, USA

Abstract

We propose an algorithm for executing transactions in object oriented databases. The object oriented database model generalizes the classical model of database concurrency control by permitting accesses to class and instance objects, by permitting arbitrary operations on objects as opposed to traditional read and write operations, and by allowing for the nested execution of transactions on objects. In this paper, we first develop a uniform methodology for treating both classes and instances. We then develop a two phase locking protocol with a new relationship between locks called ordered sharing for an object oriented database. Ordered sharing does not restrict the execution of conflicting operations. Finally, we extend the protocol to allow for nesting. The resulting protocol permits more concurrency that other known locking-based protocols.

1 Introduction

In this paper, we present a concurrency control protocol to deal with the problem of shared access to objects in object oriented databases. Traditionally, database systems are modeled as a collection of objects that can only be read or written by transactions [BHGS?]. More recently, many researchers have shown that greater concurrency can be achieved in object oriented databases by using the type specific properties of objects [Kor83, SS84, BR87, Wei89a, Wei89b]. An object oriented database differs from classical 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 database concurrency control by permitting nested transactions as opposed to flat transactions and by permitting arbitrary operations on objects as opposed to traditional read and write operations. Nested transactions increase performance and reliability, because synchronization of transactions becomes more flexible, and also because they provide

* This research is supported by the NSF under grant numbers IRI-9004998 and IRI-9117904.
finer control over failures than the flat transactions. Another aspect of object oriented databases is related to the notion of extensibility or schema evolution. That is, object oriented databases allow the classes to be modified concurrently with the accesses to the objects defined by these classes.

In this paper, we present protocols for executing atomic transactions on objects. We begin by introducing the notion of atomic operations to modify the methods and attributes of a class. This approach is novel since it provides a methodology for uniform treatment of both classes and objects from a concurrency control point-of-view. We then propose a new relationship between locks called ordered sharing [AE90] for synchronizing the execution of concurrent transactions in object oriented databases. We first describe an extension of two phase locking with ordered sharing for executing flat transactions, which are sequences of arbitrary atomic operations on classes and instances. The new lock relationship does not block the execution of conflicting operations, and hence increases concurrency and improves performance [AEL91]. The protocol permits more concurrency than the original two phase locking protocol for abstract data types [Wei89b]. Also, unlike previous protocols for object oriented databases [GK88, CF90], our protocol performs schema evolution at a fine granularity and uses the semantics of these update operations on classes.

We then extend the model of the database to include nested executions of transactions on objects. We present a two phase locking protocol with ordered sharing for synchronizing the execution of nested transactions. We show that the proposed protocol permits more concurrency than the nested two phase locking protocol (N2PL) [Moe85]. Also, given an execution with an acyclic serialization graph [HH91], the proposed protocol can accept this execution without reordering the order of atomic operations. The commit order, however, may be altered. To the best of our knowledge, this is the first attempt to integrate two distinguishing aspects of object oriented databases, i.e., schema evolution and nested execution of transactions.

The paper is organized as follows. In the next section, we develop a model for operations on classes and objects and present a synchronization mechanism for method executions on objects. In Section 3, we extend this protocol to nested execution of atomic transactions on objects. The paper concludes with a summary of our results.

2 Atomic Transactions in Object Oriented Databases

In this section, we present the model of an object oriented database [BKKK87, BBB+88, LRV88, RGN90] and present the correctness criterion used in such databases. Next, we present a concurrency control protocol to synchronize the execution of concurrent transactions in object oriented databases. The section concludes with a brief discussion about the implementation of the protocol.

2.1 Model

An object oriented database is a collection of classes and instances of these classes. A class defines a set of attributes for its instances and atomic methods or atomic procedures that are used to manipulate these instances. We allow inheritance of properties (attributes