Concurrency and Transaction Management in an Object Database

Project Proposal for SE690/SE696

Author: Jeremy T. Torres
Project Supervisor: Dr. Xiaoping Jia, DePaul University
May 18, 2003
Abstract

The main objective of an object-oriented database (OODB) is to provide persistent storage of objects and a means of querying for objects. Using the object-oriented (OO) paradigm, application developers may use programming-language-specific syntax to persist and query for objects for use in their application.

There are many varying implementations of “object-oriented” databases. On one hand, there are Object Relational databases which use a hybrid relational database implementation as the persistence engine and use the Structured Query Language (SQL) and/or an extension to SQL for access to the database. On the other hand, there are “pure” object-oriented databases which do not use existing relational database technology for their implementation and use a programming-language-specific (Java, for instance) application programming interface (API) to access the database.

Object oriented databases, in comparison with relational database implementations, share very similar implementation-specific issues. For instance, currency control and transaction management are two key issues that must be addressed. However, due to the semantic differences between the OO and relational paradigms, how the two issues are implemented vary greatly. For example, with respect to concurrency control, granularity of a data item is semantically different in a relation database versus an OODB. Consider a single Customer entity, containing first and last name attributes. In a relational database, the Customer data item would be a database record (a row or tuple) in a relational database table, with columns representing the first and last names. In an OODB, the granularity would be represented as an object (instance) of the Customer entity, whose member variables (fields) would represent the first and last name.

The overall issues of concurrency and transaction management have been previously addressed with respect to relational database implementations. As such, there is motivation for using a relational database engine to create a hybrid OODB, such as an Object Relational database. However, one is tied to the relational database engine for such system architectural conveniences. Furthermore, as most vendors of Object Relational databases have proprietary extensions to SQL, some mapping to the object-oriented model must still be performed. In contrast, OODBs using programming-language-specific objects that do not use a relational “back end” must address the issues of concurrency and transaction management if the implementation is to support more than one concurrent user.

As there are currently many vendors of OODBs, concurrency control and transaction management issues are handled differently, respective to the OODB’s implementation. For example, consider an implementation of the GemStone OODB, where each concurrent user has a “shadow” copy of a shared object table (implemented as a B-tree). Each user session modifies its own shadow copy. A merge between the shadow copy and the “master” shared object table constitutes a commit and a transaction abort would be
represented by the discarding of the user’s shadow object table. In contrast, consider the Ozone OODB, in which there is a centralized “transaction manager” associated with the database, and all remote client database command requests are associated with a transaction.

As there are multiple, varying implementations of concurrency control and transaction management in OODBs, the goal of this research is to design a flexible and extendable Java-based framework for concurrency and transaction management within the context of an object oriented database. The framework will be extendable, such that it will allow for multiple implementations of transactional policies (e.g., optimistic versus pessimistic locking). It will also be flexible enough to be a “pluggable” transactional engine providing common transactional and concurrency semantics. An existing OODB, known as the ObjectStore (a research project headed by Dr. Xiaoping Jia), will be used for this research. In the ObjectStore’s current state, it is accessible remotely over RMI; however, there is no transaction management implemented. Furthermore, Java synchronization is the only concurrency control used as a means to control updates to the database.

**Literature Review Summary**

There is a wealth of credible information available with respect to Object Oriented Databases. The Association of Computing Machinery’s (ACM) digital library was the primary fact-finding source for this research. Topics such as client-server caches, concurrency protocols, and even the design and implementation of an Object Oriented Database (e.g., Gemstone) were found. However, in the research done to date, there is no design pattern of an object-oriented database transactional “engine”.

**Study Rationale**

As mentioned in the Literature Review section, current research has not found a generic, reusable design pattern for a transactional engine, or “layer”, geared specifically toward object oriented databases. Moreover, even though there are many varying implementations of object oriented databases, there are very few research papers (as found in the initial research phase) describing an complete transactional engine implementation.

**Study Problem or Purpose**

The goals of this research are two-fold:

1. To design a flexible and extendable framework for concurrency and transaction management within the context of an object oriented database using the Java programming language. Ideally, the framework will be presented as a design pattern specifically for concurrency and transaction management of an Object Oriented database;
(2) Provide various implementations of the framework, each providing different transactional properties (e.g., optimistic versus pessimistic locking), for the ObjectStore.
   a. As the framework will be specifically geared towards the Java programming language for this research, an implementation of the Java Transaction API (JTA) within this research’s defined framework will also be provided.

Research Objectives

To provide a generic, reusable framework for concurrency and transaction management within the context of object oriented databases, coupled with various implementations via the ObjectStore.

Research Design

The generic framework proposed, designed, and implemented by this research will be compared against the strengths and weaknesses of other alternative implementations (e.g., the Gemstone and Ozone object oriented databases) found during the research process. The comparison will focus on the design and implementation of the concurrency and transactional software “layers” of an object oriented database only; the remaining “layers” are out of the scope of this research.

Work Plan

- First Presentation, May 30th, 2003: The research for this first phase will be completed for an initial UML diagram of a generic Concurrency/Transactional model will be presented.
- Summer I and II, 2003: Development and implementation of generic Concurrency/Transactional Model via the ObjectStore will be completed
- Second Presentation, September 2003. Results of developed concurrency model and implementations via the object store presented.
- Final Presentation, November 2003. Demonstration of implemented framework via ObjectStore. All results of framework and implementations, including changes since second presentation.