Educational Administration: Theory and Practice
2024, 30(5), 6348 - 6353
ISSN: 2148-2403
https://kuey.net/

Research Article

Analysis Of Fine Granularity Locking In Complex Objects

Sonal Kanungo[1*], R. D. Morena[2]

[1*] DPG Degree College, MDU University Gurugram. Email: sonalkanungo@gmail.com
[2]Department of Computer Science, Veer Narmad South Gujarat University, Surat. Email: rdmorena@vnsgu.ac.in

Citation: Sonal Kanungo, (2024) Analysis Of Fine Granularity Locking In Complex Objects, *Educational Administration: Theory and Practice, 30(5), 6348 - 6353*
Doi: 10.53555/kuey.v30i5.3943

ARTICLE INFO

ABSTRACT

Object-oriented database (OODBMS) has the capacity to support reference objects of complex structures, making them ideal for complex data demonstration. They provide a seamless integration between the application's data model and its programming language, facilitating easier and more efficient development processes. Additionally, OODBMS are employed in CAD/CAM (Computer-Aided Design/Computer-Aided Manufacturing) systems, telecommunications, scientific research and simulations. OODBMS has complex nature, therefore complex concurrency control mechanisms are needed and also that this mechanism should not degrade the performance of OODBMS. In this paper we are discussing how locks will implement on complex objects.

Keywords: Composite object, Association, Inheritance, Composition, Locking, Multiversion, Concurrency control, Serialization.

1. Introduction

Objects can represent as complex data [1]. Their complex relationships are characterized by combinations of object relationships like Inheritance, Composition, Aggregation and Association. Inheritance characterizes a 'Parent-Child' or 'IS-A' relationship [2]. Aggregation characterizes 'HAS-A' relationship. Association characterizes as 'IS-PART-OF' relationship [3]. The relationship is the connection between the entities. Relationship can be described as connectivity, cardinality, or dependency. Connectivity is the occurrence of an entity, which means the relationship with another entity is either ONE-TO-ONE, ONE-TO-MANY, MANY-TO-ONE and MANY-TO-MANY [4].

1.1.1 Association

Relationship between single objects and many other related objects is known as an association. An association is a "Use-A" relationship between two or more objects in which the objects have their own life time and there is no owner [5].

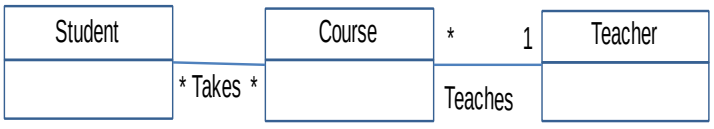

Figure-1 Association

1.1.2 Aggregation

Aggregation is a specialized form of association between two or more objects in which the objects is having its own life-cycle with ownership as well. Aggregation is a stronger form of association, representing the 'HAS-A' relationship between a component object and an aggregate object [3].

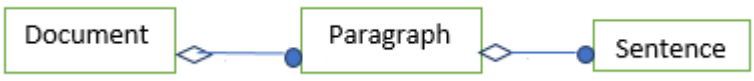

Figure 2 Aggregation

1.1.3 Composition

A Composition is strongest form of association where composite is solely responsible for managing its components including creation and destruction of these components. In this relationship child objects do not have their lifecycle without parent object [3].

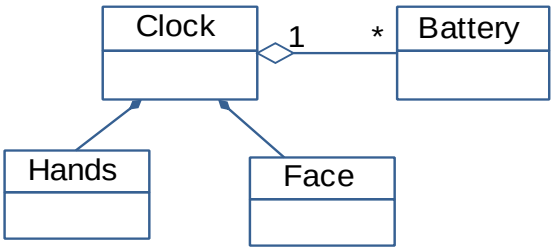

Figure-3 Composition

1.1.4 Inheritance

Inheritance is a directed relationship between two classes of the same kind. One class is called parent and other is called the child. Parent class is also known as superclass and child class is called subclass. The parent has set of instances with common properties [6]. All children have properties of super class but also have additional properties of subclasses too. The relation between a subclass and its superclasses which is known as "IS-A" [7].

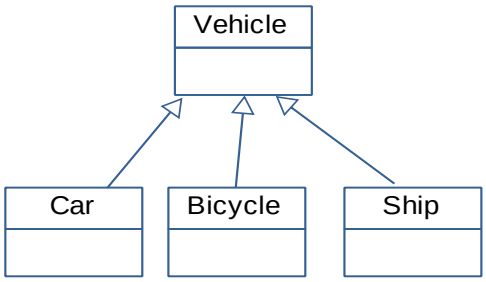

Figure 4 Inheritance

2. Concurrency Control

Several transactions can run concurrently in shared database. The system must control the interaction among the concurrent transactions; to control this variety of mechanisms can be implemented on data which are known as concurrency-control schemes.

There are different concurrency control techniques are present such as locked based, two-phase locking, Optimistic, timestamping and Multiversion currency control etc [8,9,10,11]. In Lock-Based Protocols: For concurrent access to a data item, the lock mechanism is employed. Only when a lock is in place on a data item is authorization granted to access it. Only if the request is approved can the transaction proceed. There are two ways that data objects can be locked: sharing mode (S) or exclusive mode (X)[8]. Exclusive-mode lock is provided for transactions that have the ability to read and write from the data item X. Data item S receives a shared-mode lock for transactions that can read but not write to it. [8].

2.1 Two Phase Locking

During the first stage, referred to as the growth phase, a transaction obtains all the locks that it requires. The process releases the locks during the second phase, which is referred to as the shrinking phase [8]. A process must relinquish every lock, wait, and restart if it is unable to obtain every lock during the first phase [8].

2.2 Timestamp-Based Protocols

In this algorithm, the timestamp is given to a transaction when it begins [10]. The timestamp has to be unique with respect to the timestamps of other transactions. Here, Wtimestamp is the largest time-stamp of any transaction that executed write successfully and R-timestamp is the largest time-stamp of any transaction that executed read successfully are kept. Any conflicting read and write operations are executed in timestamp order [14].

2.3 Validation-Based Protocols

This is also called as optimistic concurrency control since transaction executes fully in the hope that all will go well during validation [10]. Validation-based protocols function under the presumption that read-write conflicts between transactions happen infrequently. This permits unrestricted access to shared data items

while processing transactions. In order for a transaction to be committed, the DBMS must verify that there were no conflicts. Conflict resolution mainly leads to transaction abort [15].

2.4 The Multiversion Methods

The Multiversion mixed method uses the Timestamp ordering and Two-phase locking with multiple versions of data [11]. A timestamp is attached to each piece of data along with its version. Transactions involving multiversion updates adhere to the strict Two-phase locking (2PL) standard [8]. The read and write locks are held by the update transactions until the transaction is completed. While the transaction is holding the write lock, read-only transactions are still able to access the committed version of the data [16]. The readings never have to wait because of the prompt appropriate version return [17]. It offers more flexibility in managing the request of reads and writes since readers can read any version and writes cannot overwrite each other [18].

2.5 Multi Granularity

Everything in the database needs to be secured in the case where the transaction needs access to the entire database and the locking protocol has been used. This process takes a very long time. Because of this, the entire database is locked for improved efficiency. As such cases the lock must implemented to different granularities. The multi-granularity locking algorithm can be represented graphically as a tree [19]. Top-down order or the leaf to root structure is proposed by various granularity protocols [20]. This technique provides object-oriented concepts for fine granularity for design time requests to access data in system [21].

3. Review of Literature

WON KIM, JAY BANERJEE, HONG-TAI CHOU, JORGE F. GARZA, DARRELL WOELK The authors propose a new object-oriented database ORION, which implements composite objects. WON KIM, ELISA BERTINO, JORGE F. GARZA The authors present a new model of composite objects developed for a number of different properties. This includes independent, exclusive, dependent exclusive, independent shared, and dependent shared composite references. JAY BANERJEE, WON KIM The authors suggested the results of various issues of schema evolution such as dynamic definition and subsequent changes to a database schema in an object-oriented database environment. The framework is based on a graph-theoretic model of the class lattice with multiple inheritance. WOOCHUN JUN AND LEE GRUENWALD Authors discuss three significant issues in object-oriented database management system (OODBMS), such as semantics of methods, nested method invocation and referentially shared object. K.P. ESWARAN, J.N. GRAY, R.A. LORIE, AND I.L. TRAIGER The authors discussed the notions of transaction, consistency, and locking. It has been argued by authors that consistency is required between transactions, whether be two-phase and well-formed or conversely that if all transactions are well-formed and two-phase then any legal schedule is consistent. H.T. KUNG and JOHN T. ROBINSONS Authors propose two groups of nonlocking concurrency. The techniques used are ‘optimistic’ as they depend fundamentally on transaction backup as a control mechanism, ‘hoping’ that conflicts won’t incur between transactions. CHRISTOS H. PAPADIMITRIOU, PARIS C. KANELLAKIS discussed problem of concurrency control when the database management system supports multiple versions of the data and multiversion methodology.

4. Methodology

Transactions are arranged in queue and served as they come to system, with timestamp given when they are created. Transactions are considered to be approved if the lock mode is compatible with the existing transaction. In case of incompatibility, new transaction will go to Wait. Transaction’s locking synchronizes by mutual exclusion. A transaction must acquire a lock before accessing a data item on that object from the OODBMS. The concurrency control technique will grant a lock request only if the data item is not currently locked by another transaction. Else, it will force the requesting transaction to wait.

4.1 Locking for Objects

The transaction can request to access an object. This object can be the independent or dependent object. When we are accessing the independent object in our proposed scheme, will lock this object only. If this object is not independent then it can be either composite or associative object. It maintains reference identifications of all related objects so that it is easy to lock related objects. Whenever some client requests composite object, in order to maintain the consistency, the object lock is also applied to all component objects. Lock mode of component and container object will be in the same mode. Association can be described also as ‘HAS-A’ relationship between objects. In the case of Association, independent life cycle of reference object exists. Object lock is set on the associative object to maintain the consistency. Furthermore, even if the associated objects are having a lock, the associative object will not possess the same lock. During the locking process, associative objects are allowed to be used by various clients.

4.2 Locking for Classes

In the proposed scheme it is possible that the class transactions can request for the single class as well as related classes, component classes, associative classes or inherited classes.
In composition when a transaction is requesting the lock and if there is no lock or compatible lock exist on class; class and its component class will lock in the same mode. In Association
When a transaction is requesting the lock and if there is no lock or compatible lock exist on class; individual class will be locked as in association classes are independent. In Inheritance
with the proposed scheme, when a class transaction is made for a base class, the requested lock mode is set on the class. Then the lock is inherited to all its children including the edges. When a change is made on the definition of an attribute/ method/ instance/ relationship or the class itself, all the classes related to this class (called subclass lattice) should also be locked in the same lock mode to maintain the consistency. In a transaction, a share mode is assigned to the primary class and its subclasses, and to the domain classes and their subclasses. However, in an update transaction, an exclusive mode is assigned to the primary class and its subclasses, and to the domain classes and their subclasses.

5 Results and Analysis

This simulation examines the behavior of the algorithms under a mix of transactions for which our methodology is designed. The mix used here consists of Update transactions and Readonly transactions. Update transactions and Readonly transactions are varied from small to very large as a fraction of the overall database size. The modified algorithm that combines two different types of multiversion control concurrency and that includes the MVTO and the MVPL.
The transactions generated in serialized manner. Readonly transaction are actually assigned a proper timestamp which clearly corresponds to its actual start time and it can also always refer to a proper version. Each version is having have different maximum timestamp. Transaction requesting timestamp is less than version's timestamp. When different versions are created each of them is assigned the with write-stamp which corresponds to its commit time. Locks conflicts which are blocked, are the main cause of deadlocks. Deadlock can be resolved by aborting the victim transaction. Aborted transaction will be restarted by the system at a later time without user intervention.

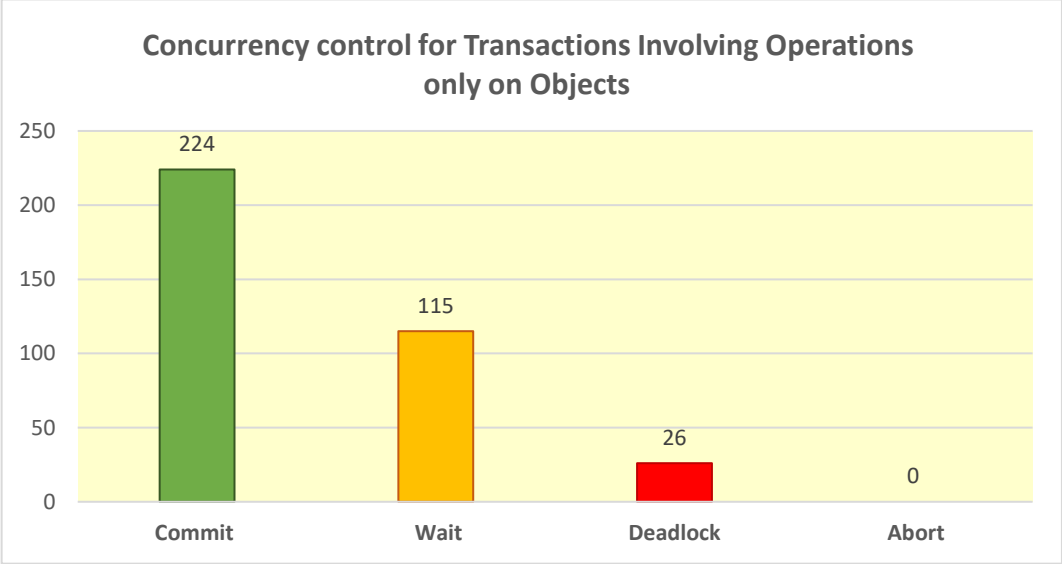

Figure-5

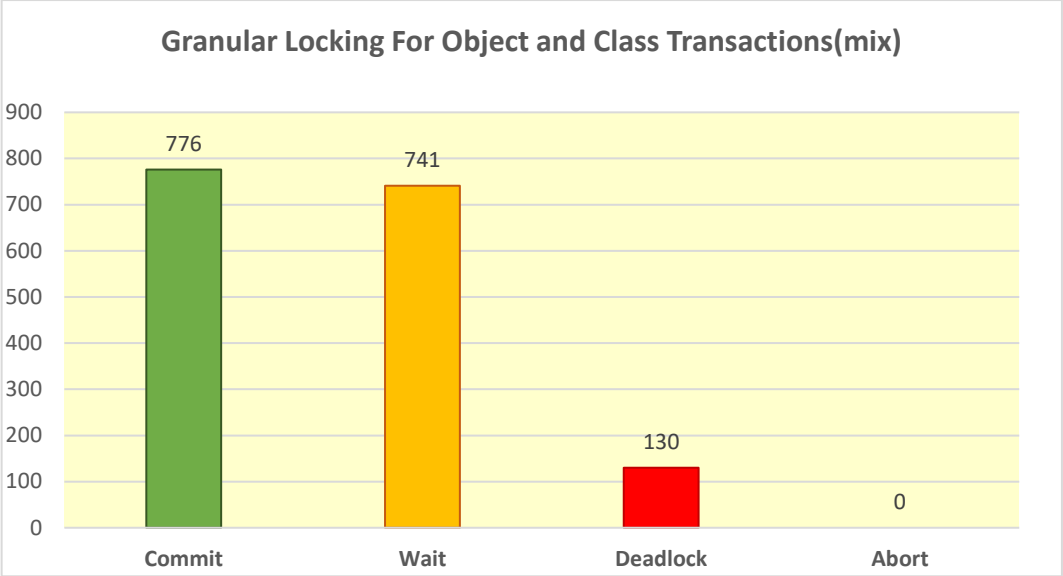

Figure-6

An object may be associated with multiple objects. A finer level of granularity of locking experiences more lock overheads but reduces the level of lock conflicts.
The finest granularity is achieved for class content level operations. The other design operations are also divided into subclass lattice and class lattice level operations. Concurrency control on the write-read conflicts and write-write conflicts between classes related by inheritance and aggregation for both classes as well as object accesses is achieved. Concurrency control technique is providing granularity lock model for fine granularity among class transactions and object transactions. The purpose for the algorithm fulfills user's expectation in terms of high number of commit and better throughput.

Conclusion

The new concurrency control mechanism provides database integrity and consistency. Our proposed scheme gives a version of classes and objects which will give better performance for Readonly transactions to enhance concurrency control.
Composite object is also used as the unit of locking so that the number of locks that must be set are minimized in retrieving a composite object from the database.
It improves the performance with a finer granularity of locking and reduces locking overhead. It provides concurrency control method, with the emphasis on high-performance; high-contention transaction processing environments, and provided a self-complete description of locking, which combine locking and multiversion.

Conflict of Interest

The authors declare no conflict of interest.

Reference

1. JAY BANERJEE, HONG-TAI CHOU, JORGE F. GARZAWON KIM, DARRELL WOELK, and NAT BALLOU(1987). Data Model Issues for Object-Oriented Applications, ACM Transactions on Office Information Systems, 5(1), 3-26.
2. BANERJEE. J., W. KIM, H.J. KIM, AND H.F. KORTH(1987). Semantics and Implementation of Schema Evolution in Object-Oriented Databases, in Proc.ACM SIGMOD Conference.
3. WON KIM, JAY BANERJEE, HONG-TAI CHOU, JORGE F. GARZA, DARRELL WOEL(1987).Composite Object Support in an Object-Oriented Database System, OOPSIA, 87.118-125.
4. RUMBAUGH J(1987). Relations as Semantic Constructs in an Object-Oriented Language, SIGPLAN Notices, 22(12).466-481
5. GARZA, J. E, AND W. KIM (1988), Transaction Management in an Object-Oriented Database System, In Proc. ACM-SIGMOD Intl. Conf. on Management of Data, Chicago.
6. WON KIM, ELISA BERTINO, JORGE F. GARZA (1989). Composite Objects Revisited, ACM, 337-347.
7. JAMES RUMBAUGH MICHAEL BLAHA WILLIAM PREMERLANI FREDERICK EDDY WILLIAM LORENSEN (1991). Object-Oriented Modeling and Design , Prentice Hall.
8. K.P. ESWARAN, J.N. GRAY, R.A. LORIE, I.L. TRAIGER (1976). The Notions of Consistency and Predicate Locks in a Database System. *Communications of the ACM,* 19(11), 624-633.

9. DAVID P. REED (1983). Implementing atomic actions on decentralized data. *ACM Transactions on Computer Systems*,1(1), 3-23.
10. H.T. KUNG, JOHN T. ROBINSON (1982). On Optimistic Methods for Concurrency Control. *ACM Transactions on Database Systems*, 6(2), 213-226.
11. CHRISTOS H. PAPADIMITRIOU, PARIS C. KANELLAKIS (1984). On Concurrency Control by Multiple Versions. *ACM Transactions on Database Systems*, 9(1),89-99.
12. PHILIP A. BERNSTEIN and NATHAN GOODMAN (1983). Multiversion Concurrency Control-Theory and Algorithms. *ACM Transactions on Database Systems*,8(4), 465-483.
13. ERIC A. BREWER (2000). Towards robust distributed systems. Conference: Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing,1,1-7.
14. SONAL KANUNGO, R.D. MORENA (2015). Analysis and Comparison of Concurrency Control Techniques. *International Journal of Advanced Research in Computer and Communication Engineering*, 4(3), 245-251.
15. SONAL KANUNGO, R.D. MORENA (2016). Comparison of Concurrency Control and Deadlock Handing in Different OODBMS. *International Journal of Engineering Research and Technology*, 5(5), 492-498.
16. SONAL KANUNGO, R.D. MORENA (2018). Evaluation of Multiversion Concurrency Control Algorithms. *International Journal of Research in Electronics and Computer Engineering,* 6 (3), 807-813.
17. SONAL KANUNGO, R.D. MORENA (2019). Effective Correctness Criteria for Serializability in Multiversion Concurrency Control Technique (2019*). International Journal of Innovative Technology and Exploring Engineering (IJITEE)*,8(12),1674-1653.
18. SONAL KANUNGO, R.D. MORENA (2017). Issues with Concurrency Control Techniques. International Journal of Electrical Electronics & Computer Science Engineering Special Issue - AET 2017,1-6.
19. SONAL KANUNGO, R.D. MORENA (2017). SEMANTIC MODEL FOR COMPLEX OBJECT IN OBJECT ORIENTED DATABASE", International Journal of Advanced Research in Computer Science, 8(15), 1651-1655.
20. WAI LAM, YALIN WANG, AND YONGBING FENG, Concurrency Control in Object-Oriented Databases. 46-65.
21. WOOCHUN JUN AND LEE GRUENWALD (2000), A Multi-Granularity Locking-Based Concurrency Control in Object Oriented Database System, Elsevier Journal of Systems and Software, 201-217.