Development and Performance Evaluation of a Concurrency Control Technique in Object-Oriented Database Systems

Woochun Jun[1]
[1]Department of Computer Education, Seoul National University of Education
Seoul, Korea
[e-mail: wocjun@snue.ac.kr]
Suk-Ki Hong[2]
[2]Department of Business Administration, Dankook University
Yongin, Korea
[e-mail: skhong017@dankook.ac.kr]
*Corresponding author: Suk-Ki Hong

Received October 19, 2017; revised February 19, 2017; accepted March 14, 2017; published April 30, 2018

Abstract

In this work, we propose a concurrency control scheme in object-oriented database (OODB). Since an OODB provides complex modeling power than the conventional relational databases, a concurrency control technique in OODB is also rather complicated and has influence on the overall performance. Thus, it is very important to develop a concurrency control technique with less overhead. The proposed scheme deals with class hierarchy that is a key concept in OODBs. The proposed scheme is developed on implicit locking scheme. Also, the proposed scheme is designed using data access frequency in order to reduce locking overhead than implicit locking. It means that, if access frequency information is not available, the proposed scheme works just like the existing implicit locking, In our work, the correctness of the proposed scheme is proved. The performance is analyzed depending on access types. Also, it is proved that our scheme performs works much better than the implicit locking does.

***Keywords*:** Object-oriented Database, Concurrency Control, Performance Evaluation, Multimedia, Class Hierarchy

A preliminary version of this paper was presented at APIC-IST 2017, and was selected as an outstanding paper.

http://doi.org/10.3837/tiis.2018.04.028 **ISSN : 1976-7277**

1. Introduction

OODBs have been used for many non-traditional application areas like multimedia databases and geographic information systems. These applications need more complicated modeling power for expressing complex relationships among data. In OODB, transactions can access data objects by invoking methods. Usually transaction consists of a series of method invocations on data objects [1,2]. It is essential to provide an effective transaction management scheme in which more transactions concurrently.

Concurrency control is a technique that is used to maintain database consistency [3,4]. Concurrency control in OODBs is more complicated than concurrency control schemes traditional relational databases due to the following reason[5,6]. The typical OODBs has so called class hierarchy(also called inheritance hierarchy). In class hierarchy, a class can inherit the definitions of its super class. Also, instances of a class object belong to instances of its super classes [7,8,9]. In OODBs, there are two different types of access on objects: instance object access and class definition object access [1]. Also, there are two access types accessing for more than one class objects: class definition update and IIH (Instance Access to Inheritance Hierarchy) [7,10]. A query is a typical example of IIH.

For convenience, we call MCR (Multi-Class Request) for class definition change and IIH, and OCR (One-Class Request) for accesses like class definition access and instance access to only one class. In the literature, there are two locking-based concurrency control schemes for class hierarchy: explicit locking [1,11] and implicit locking [4,5,7,12].

The purpose of this paper is to develop concurrency control scheme for class hierarchy. This paper is organized as follows. In Chapter 2, the existing approaches for dealing with inheritance hierarchy are discussed. In Chapter 3, a new scheme is proposed. In Chapter 4, we present performance analysis of the proposed scheme. In Chapter 5, the correctness of the proposed scheme is discussed. Finally, conclusions and further research issues are discussed in Chapter 6.

2. Literature Review

2.1 Background

In explicit locking, for each MCR access, a lock is needed not only a class, say A, but also on every sub class of A through inheritance hierarchy. For each OCR, a lock is only necessary for a class being accessed. Thus, for an MCR, an access to a leaf class will requires less locks than a class close to the root. In the meanwhile, the explicit locking requires more locks for an access to a class close to the root.

On the other hand, in implicit locking, a lock on a class A requires intention locks on super classes through inheritance hierarchy as well as lock on class A [4]. An intention lock on class means that possible locks will be set on a sub class. With aid of an intention lock, an MCR access to classes on inheritance hierarchy may find conflict earlier. However, due to inheritance hierarchy, implicit locking requires more locks for access to class close to leaf class.

For example, consider the inheritance hierarchy in **Fig. 2**. Orion [5] and O2 [1], are used for the explanation of two locking schemes. In **Fig. 1.a**, for an access to class E, each technique works as follows. For implicit locking, intention locks IWs for W (Write) locks need to be set for all classes from E to the root A. Thus, any MCR access on super classes of E can find conflict on root A. In the meanwhile, an explicit locking requires a Cw (Class Definition

Change) lock on subclasses of E as well as class E. Also, locking for a query on C can be done as in **Fig. 1.b**.

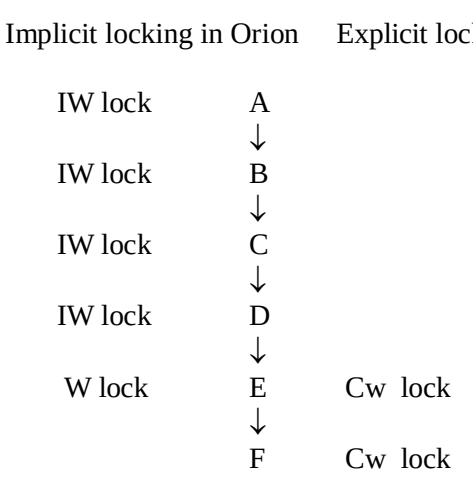

Fig. 1.a. Locking for class definition change in Orion and O_2

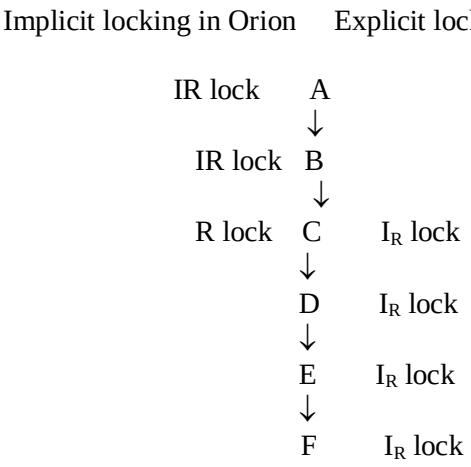

Fig. 1.b. Locking for query in Orion and O_2

2.2. Related Works

There are some research works for concurrency control techniques to reduce locking overhead for class hierarchy in OODBs as follows.

In [13], for class hierarchy with multiple inheritance , it is shown that some redundant locks can be reduced without affecting the consistency of the database. The proposed scheme, called Multiple Inheritance Implicit Locking (MIIL), is based on so-called implicit locking. The proposed scheme can eliminate redundant locks that are necessary in the existing implicit locking scheme. Intention locks are required as the existing implicit locking scheme. In this paper, it is shown that MIIL has less locking overhead than implicit locking does. In their work, only structural information such as OODB inheritance hierarchies, single inheritance and multiple inheritance are used. It means that no additional overhead is necessary for reducing

locking overhead.

Concurrency control techniques dealing with class hierarchy using access frequency information are proposed in [14,15]. In [14], the proposed technique is based on implicit locking so that some intention locking overhead can be reduced by locking on frequently access classes intead of locking on every superclass of a class to be accessed. In the meanwhile, in [15], a concurrency control scheme is proposed to reduce locking overhead. The proposed scheme is based on explicit locking. In the proposed, using access frequency information, the proposed scheme can reduce locking overhead than the existing explicit locking.

3. Development of a New Concurrency Control Scheme

3.1 The Proposed Locking Scheme

The proposed scheme is based on implicit locking. The basic idea is that, in the proposed scheme, intention locks are not set on all of the super classes of a target class that is a class to be accessed. That is, some super classes need not be locked. Note that, in implicit locking, every superclass of a target class has an intention lock through the superclass chain. We adopt a concept called FA (Frequently Accessed) class. Roughly, a FA class is a class that needs an intention lock. For an OODB, how to determine if a class is an FA class or not is discussed in next section.

Assume that a lock is requested on class A. The proposed scheme works as follows. For simplicity, strict two-phase locking [3,4] is adopted.

(Step 1) Check conflict and set lock on a target class

(Step 2) Set locks on classes until the first FA class through superclass chain

•From class A to the first FA class through the superclass chain, set an intention lock on each class.

•If the class A is a FA class, do nothing.

(Step 3) Set an intention lock on each FA class through superclass chain

(Step 4) Set lock on each subclass of A that has more than one parent superclass.

// Deal the multiple inheritance case //

For instance, consider the inheritance hierarchy as in Figure 2.a. Assume that FA classes are Cl, C4, C7, and C10, respectively. Also, assume that locks are requested by two transactions T_A and T_B as follows.

1) T_A: Access on class C7
2) T_B: Access on class C9

Assume that Li denotes locks by transaction i. As in Fig 2.b and 2.c, 16 locks and 8 locks are required for T_A and T_B by the implicit locking and the proposed scheme, respectively.

C1	C1:L_1;L_2	C1(FA):L_1;L_2
↓	↓	↓
C2	C2:L_1;L_2	C2
↓	↓	↓

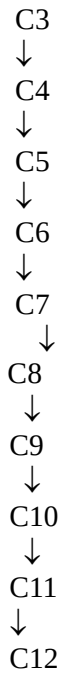

Fig 2.a.
Inheritance Hierarchy

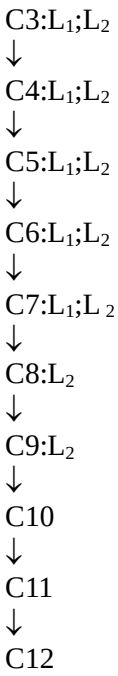

Fig. 2.b.
Implicit Locking

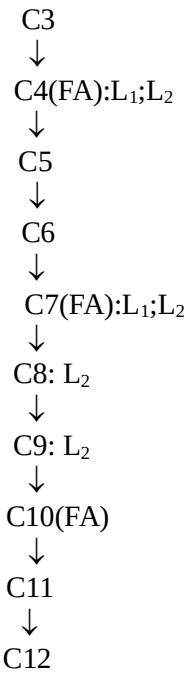

Fig. 2.c.
The Proposed Locking

In the literature, the concurrency control techniques for class hierarchy in [14,15] are also based on access frequency for class hierarchy. However, the proposed technique in this work is different as follows. At first, in [14], in order for the technique to reduce locking overhead, more specific locking modes need to be defined depending on conflict modes. However, the proposed technique in the paper does not require any special locking modes other than locking mode in Orion[5]. Also, the technique in [15] is based on explicit locking. It means that the technique is proposed to reduce locking overhead than the existing explicit locking technique.

3.2 FA Class Assignment Technique

In the proposed scheme, the following FA class assignment scheme is adopted. Assume that access frequency ratio to each class is constant, the FA class assignment scheme is constructed as follows.

Step 1)
A root class is assigned FA class.

Step 2)
// Start from each leaf class //

For each class A whose subclasses are already determined,
-calculate the number of locks (N_A) if A is assigned as FA class
-calculate the number of locks (N_B) if A is assigned not assigned as FA class

Step 3)
The class A becomes FA class only if $N_A < N_B$

For example, consider a simple single inheritance hierarchy as in Figure 3.a and assume access frequency information on each class as in Figure 3b. Note that, based on the proposed concurrency control scheme, a root class is assigned as FA class automatically so that we do not need access frequency on the root class.

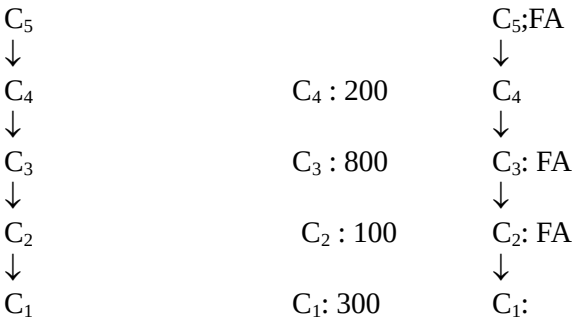

Fig. 3.a. An inheritance hierarchy

Fig. 3.b. Access frequency

Fig. 3.c. FA assignment result

For class C_1, since it is a leaf class, regardless of its access frequency, it is assigned non-FA class. For class C_2, if it is assigned FA class, the total number of locks required is 1,100 (900 locks by C_1 and 200 locks by C_2) since a transaction accessing C_1 needs 300 locks each on C_1, C_2, and C_5, respectively and a transaction accessing C_2 needs 100 locks each on C_2 and C_5, respectively. On the other hand, if it is not assigned FA class, the total number of locks is 1,900 (1,500 locks by C_1 and 400 locks by C_2) since a transaction accessing C_1 needs 300 locks each on every class and a transaction accessing C_2 needs 100 locks each on C_2, C_3, C_4, and C_5, respectively. Thus, the class C_2 becomes a FA class.

For class C_3, if it is assigned FA class, the total number of locks is 3,100 locks (1,200 locks by C_1, that is, 300 locks each on C_1, C_2, C_3, and C_5 and 300 locks by C_2, that is, 100 locks each on C_2, C_3, and, C_5, and 1,600 locks by C_3, that is, 800 locks each on C_3 and C_5).

In the meanwhile, if it is not assigned FA class, the total number of locks is 3,500 locks (900 locks by C_1, that is, 300 locks each on C_1, C_2, and C_5 and 200 locks by C_2, that is, 100 locks each on C_2 and C_5 and 2,400 locks by C_3, that is, 800 locks each on C_3, C_4, and C_5). Thus, the class C_3 becomes a FA class.

For class C_4, if it is assigned FA class, the total number of locks is 4,700 (1,500 locks by C_1, that is, 300 locks each on C_1, C_2, C_3, C_4, and C_5, and 400 locks by C_2, that is, 100 locks each on C_2, C_3, C_4, and C_5, and 2,400 locks by C_3, that is, 800 locks each on C_3, C_4, and C_5 and 400 locks by C_4, that is, 200 locks each on C_4, and C_5). If it is not assigned FA class, the total number of locks is 3,500 (1,200 locks by C_1, that is, 300 locks each on class C_1, C_2, C_3, and C_5 and 300 locks by C_2, that is, 100 locks each on C_2, C_3, and C_5, and 1,600 locks by C_3, that is, 800 locks each on class C_3 and C_5, and 400 locks by C_4, that is, 200 locks each on class C_4 and C_5). Finally, the class C_4 becomes non-FA class. The final results of FA assignment are shown in **Fig. 3**.c.

Let's consider the following example as in Figure 4. The inheritance hierarchy in Figure 4.a is same as in **Fig. 3.a**. Assuming that access frequency information for each class is given as in Figure 4.b. The class C_1 becomes non-FA class by our FA assignment scheme. For class C_2, if it is assigned FA class, the total number of locks is 500 (300 locks by C_1, that is, 100 locks each on class C_1, C_2, and C_5, and 200 locks by C_2, that is, 100 locks on C_2 and C_5). If C_2 is not assigned FA class, the total number of locks is 900 (500 locks by C_1, that is, 100 locks each on C_1, C_2, C_3, C_4, and C_5, and 400 locks by C_2, that is, locks each on C_2, C_3, C_4, and C_5). This means that C_2 becomes FA class.

For class C_3, if it is assigned FA class, the total number of locks is 900 locks (400 locks by C_1, that is, 100 locks each on C_1, C_2, C_3, and C_5 and 300 locks by C_2, that is, 100 locks each on class C_2, C_3, and C_5, and 200 locks by C_3, that is, 100 locks each on class C_3 and C_5). In the meanwhile, if it is not assigned FA class, the total number of locks is 800 locks (300 locks by C_1, that is, 100 locks each on class C_1, C_2, and C_5, and 200 locks by C_2, that is, 100 locks each on class C_2 and C_5, and 300 locks by C_3, that is, 100 locks each on class C_3, C_4, and C_5). Thus, the class C_3 becomes non-FA class.

For class C_4, if it is assigned FA class, the total number of locks is 1,200 (400 locks by C_1, that is, 100 locks each on class C_1, C_2, C_4, and C_5, and 300 locks by C_2, that is, 100 locks each on class C_2, C_4, and C_5, and 300 locks by C_3, that is, 100 locks each on class C_3, C_4, and C_5, and 200 locks by C_4, that is, 100 locks each on class C_4 and C_5). If it is not assigned FA class, the total number of locks is 1,000 (300 locks by C_1, that is, 100 locks each on class C_1, C_2, and C_5, and 200 locks by C_2, that is, 100 locks each on class C_2 and C_5, and 300 locks by C_3, that is, 100 locks each on class C_3, C_4, and C_5, and 200 locks by C_4, that is, 100 locks each on class C_4 and C_5). Finally, the class C_4 becomes non-FA class.

The final results of FA assignment are shown in **Fig. 4**.c.

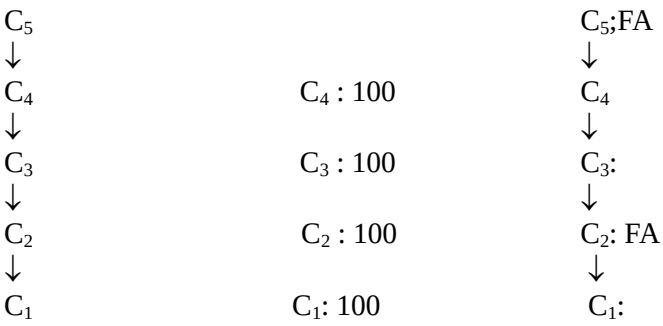

Fig. 4.a. An inheritance hierarchy

Fig.4.b. Access frequency

Fig.4.c. FA Assignment result

4. Performance Analysis of the Proposed Scheme

In this Section, we present performance analysis of the proposed scheme. We classify the performance of the proposed scheme into 3 categories, the worst case, the average case, and the best case, respectively.

4.1 The Worst Case

The worst case of the performance means that the proposed scheme works just the existing

implicit locking. Alternately, it means two following cases.

Case 1) There is no FA class except the root.

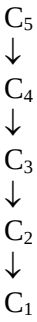

Fig. 5.a.
An inheritance hierarchy

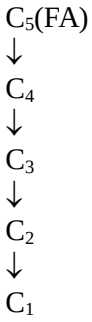

Fig. 5.b.
FA assignment result

In this case, the proposed scheme works the implicit locking no matter what kinds of accesses to the given inheritance hierarchy. This is the case that, for an inheritance hierarchy in **Fig. 5.a**, and FA assignment as shown in **Fig. 5.b**, the proposed scheme works as in the implicit locking.

Case 2) All accesses are on the classes before the second FA class.

As in case 1) above, the proposed scheme works as in the implicit locking. Consider the following **Fig. 6**.

Fig. 6.a.
An inheritance hierarchy

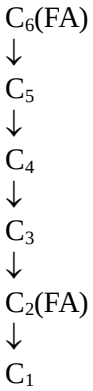

Fig. 6.b.
FA assignment result

For an inheritance hierarchy in **Fig. 6.a**, assuming FA assignment as in **Fig. 6.b,** if all accesses to the inheritance hierarchy are on class C_3, C_4, and C_5, the proposed scheme works as in the implicit locking

4.2 The Average Case

The average case means that there are some FA classes in the inheritance hierarchy and

accesses are distributed evenly all over the inheritance hierarchy. Consider the following inheritance hierarchy(**Fig. 7.a**) and FA assignment(**Fig. 7.b**).

For simplicity, assume that there is only access to each class. In this case, the total number of locks required for the implicit locking and the proposed scheme are as follows.

-Number of locks for the implicit locking: 21
-Number of locks for the proposed scheme: 15

The number of locks required for the implicit locking varies depending on the class to be accessed. That is, for an access to a class near the root, it has fewer locks. However, for an access to a class near the leaf, it induces more locking overhead.

On the other hand, the number of locks required for the proposed scheme varies depending on the number of FA classes as well as the class to be accessed. As in the implicit locking, for an access to a class near the root, it induces fewer locks. For an access to a class near the leaf, it induces more locking overhead. However, in this case, the proposed scheme needs fewer locks than the implicit locking does.

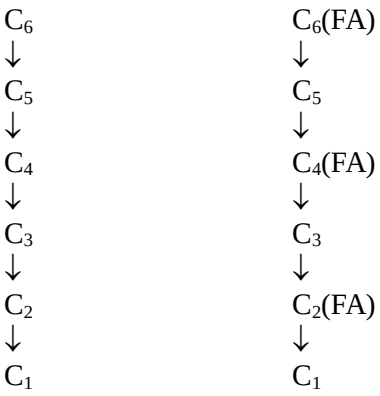

Fig. 7.a. An inheritance hierarchy

Fig. 7.b. FA assignment result

4.3 The Best Case

The best case means that every access is on the second FA class in the inheritance hierarchy. In this case, only 2 locks are required no matter what the number of classes in the inheritance hierarchy. Consider the following inheritance hierarchy(**Fig. 8.a**) and FA assignment(Figure 8.b).

For the FA assignment as in Figure 8.b, there is only one access to a class C_1 that is a leaf. Then the number of locks required for the implicit locking and the proposed locking are as follows.

-Number of locks for the implicit locking: 8

-Number of locks for the proposed scheme: 2

According to the above 3 cases(the worst case, the average case, and the best), we can conclude that the proposed scheme works better than the existing implicit locking for an access to a class near the leaf and the second FA class near the leaf.

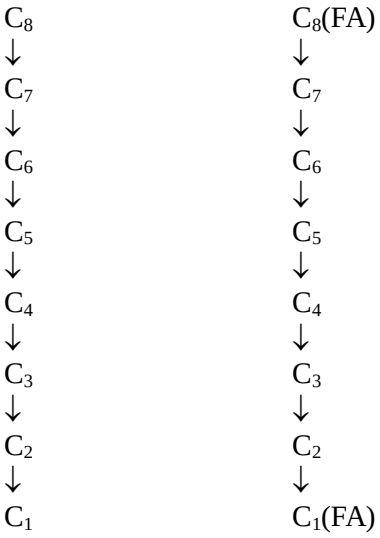

Fig. 8.a. An inheritance hierarchy

Fig. 8.b. FA assignment Result

5. Correctness of the Proposed Scheme

In this section, we show that the proposed scheme performs better than the implicit locking scheme. Since the proposed scheme requires less or equal number of locks than implicit locking for any access, we show that the proposed scheme is correct by showing that any possible conflict is detected by the proposed scheme.

Claim: The proposed scheme detects any conflicts between a lock requester (LR) and a lock holder (LH).

Proof:
There are 4 cases for access types of LR and LH,

Case 1)
-Access by LH: SCR
-Access by LR: SCR

If LH and LR are accessing different classes, there is no conflict. IF LH and LR are accessing the same target class, any conflicts can be detected on class.

Case 2)
-Access by LH: SCR

-Access by LR: MCR

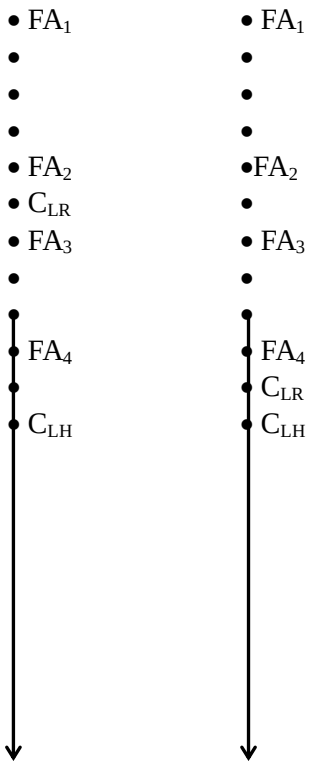

Fig. 9.a. case 2.1 **Fig. 9.b.** case 2.2

Let C_{LR} and C_{LH} be two target classes for LR and LH, respectively. If C_{LH} is a superclass of C_{LR}, there is no conflict since LR does not access C_{LH}. If C_{LH} is a subclass of C_{LR}, then there are two further cases. If there is a FA class that is a superclass of both C_{LR} and C_{LH}, then possible conflicts are detected on the first FA class through the superclass chain of LR (case 2.1). For instance, assume that there is an inheritance hierarchy with four FA classes as in **Fig. 9.a**.

In **Fig. 9.a**, the possible conflicts are detected on FA_2 since both LR and LH must have locks on FA_2. In the meanwhile, if there is no such FA class between C_{LR} and C_{LH} as in **Fig. 9.b**. (case 2.2), the possible conflict is detected on C_{LR} since LH must have an intention lock on C_{LR}.

Case 3)
-Access by LH: MCR
-Access by LR: SCR

If the C_{LH} is a subclass of the C_{LR}, there is no conflict. If C_{LH} is a superclass of C_{LR}, then there are two further cases in which conflicts will be detected. If there is at least a FA class that is the first superclass of both C_{LR} and C_{LH} as in **Fig. 10.a**, then conflict is detected on a FA_2. (case 3.1). This is because LH and LR must a lock on the FA_2 according to rules of the

proposed scheme. Otherwise (that is, there is no such FA class between C_{LH} and C_{LR} as in **Fig. 10.b**), the conflict is detected on C_{LH} since C_{LR} must set an intention lock on C_{LH}(case3.2).

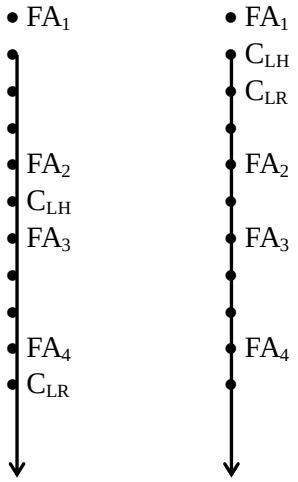

Fig. 10.a. case 3.1 **Fig. 10.b.** case 3.2

6. Conclusions and Further Research Works

OODBs have many non-traditional applications since they have higher modeling power than traditional relational databases. For many and various users, performance evaluation issue is very important. A concurrency control scheme is the one of key factors for determining overall performance evaluation in database systems.

In this paper, a class hierarchy concurrency control scheme is proposed. It is developed to reduce locking overhead for access to inheritance hierarchy. Using data access frequency, the proposed concurrency control scheme induces fewer locks than the implicit locking scheme. We also prove that the proposed concurrency control scheme needs fewer locks than the implicit locking.

Our future research goal is to propose a comprehensive scheme that combine inheritance hierarchy with composite object hierarchies. Also, we have a plan to do intensive simulation work for performance evaluation of our work.

References

[1] M. Cart and J. Ferrie, "Integrating Concurrency Control into an Object-Oriented Database System," in *Proc. of 2nd Int. Conf. on Extending Data Base Technology*, Venice, Italy, pp. 363-377, 1990. Article (CrossRef Link)

[2] V. Geetha and N. Sreenath, Semantic "Concurrency Control on Continuously Evolving OODBMS Using Access Control Lists," in *Proc. of 9th International Conference on Distributed Computing and Internet Technology*, Bhubaneswar, India,pp. 523-534, 2013. Article (CrossRef Link)

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

[4] H. Korth and A. Silberschartz, *Database System Concepts*, 2nd Edition, McGraw Hill, 1991.
[5] J. Garza and W. Kim, "Transaction Management in an Object-Oriented Database Systems," in *Proc. of ACM SIGMOD Int. Conf. on Management of Data*,pp. 37-45, 1988. Article (CrossRef Link)
[6] V. Geetha, "Semantic Based Concurrency Control in OODBMS," in *Proc. of 2011 International Conference on Recent Trends in Information Technology*, Chennai, India, pp. 1313-1318, 2011. Article (CrossRef Link)
[7] W. Kim, Introduction to Object-Oriented Databases, *The MIT Press*, Cambridge, MA, USA, 1990.
[8] R. Wazlawick, "Object-Oriented Analysis and Design for Information Systems," *Morgan Kaufman*, Burlington, MA, USA, 2014.
[9] G. Blokdyk, Object-oriented Analysis Complete Self-Assessment Guide, *Createspace Independent Pub*, North Charleston, SC, USA, 2017.
[10] J. Garza and W. Kim, "Transaction Management in an Object-Oriented Database Systems," in *Proc. of ACM SIGMOD Int. Conf. on Management of Data*,pp. 37-45,1988. Article (CrossRef Link)
[11] C. Malta and J. Martinez, "Automating Fine Concurrency Control in Object-Oriented Databases," in *Proc. of 9th IEEE Conf. on Data Engineering*, Vienna, Austria, pp. 253-260, 1993. Article (CrossRef Link)
[12] S. Lee and R. Liou, "A Multi-Granularity Locking Model for Concurrency Control in Object-Oriented Database Systems," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 8, No. 1, pp. 144-156, 1996. Article (CrossRef Link)
[13] W. Jun and S. Hong, "Development of a Concurrency Control Technique for Multiple Inheritance in Object-Oriented Databases," *Journal of Internet Computing and Services*, Vol. 15, No. 1, pp. 63-71, 2014. Article (CrossRef Link)
[14] W. Jun and L. Gruenwald, "An Effective Class Hierarchy Concurrency Control Technique in Object-Oriented Database Systems," *Journal of Information and Software Technology*, Vol. 40, No. 1, pp. 45-53, 1998. Article (CrossRef Link)
[15] W. Jun and L. Gruenwald, "An Optimal Locking Scheme in Object-Oriented Database Systems," *International Conference on Web-Age Information Management*, Shanghai, China, pp. 95-105, 2000. Article (CrossRef Link)

Woochun Jun is a professor in Dept. of Computer Education at Seoul National University of Education, Seoul, Korea. He received Ph.D. degree in Computer Science from University of Oklahoma, USA in 1997. He also received a Master's degree and BS degree in Computer Science from Sogang University, Seoul, Korea, in 1987 and 1985, respectively. His research areas include information education, information communication ethics, and gifted education in IT.

Suk-Ki Hong is a professor in the Department of Business Administration, Dankook University, Gyeonggi-do, Korea. He received Ph.D. from the University of Nebraska-Lincoln, USA, in 1996. His main research interests are e-Learning, e-Business, e-Service, Service Quality, SCM, and IT Strategies.