공동저작 시스템에서의 동시성 제어와 쓰기 권한 제어

유 재 홍[†] · 성 미 영[††]

요 약

이 논문에서는 공동저작 시스템에서의 동시성 제어와 쓰기 권한 제어를 위한 효율적인 방안을 제시한다. 본 연구에서 다루는 공동저작 시스템의 문서는 논리 객체들로 구성된 트리 구조와 트리 구조의 단말에 연결된 내용 조각들로 표현된다. 이러한 문서의 논리구조를 다루는 공동저작 시스템의 동시성 제어 기법으로 우리는 계층적으로 구성된 객체들에 대해 각 계층별로 잠금을 요청할 수 있는 다중 잠금(multiple granularity locking) 기법을 확장하는 접근을 채택하였다. 계층적으로 구성된 객체들에 대한 연산을 세분화하여 잠금 호환 테이블(locking compatibility table)을 정의하고, 이 테이블을 기반으로 잠금을 허락하는 확장된 다중 잠금(extended multiple granularity locking) 기법을 제안한다. 이 기법은 여러 사용자들의 공유 객체에 대한 동시 접근을 극대화하는 장점이 있다. 또한 이 논문에서는 그룹/비그룹(Group/Non-Group) 개념을 적용하여 비그룹 사용자들의 쓰기를 방지함으로써 매우 합리적으로 저작권을 보호할 수 있는 쓰기 권한 기법을 제안한다.

A Framework for Concurrency Control and Writing Authority Control in Collaborative Writing Systems

Jae-Hong Yoo[†] · Mee-Young Sung[††]

ABSTRACT

This paper presents the efficient mechanisms for concurrency control and writing authority control in collaborative writing systems. The documents in our collaborative writing system are represented by the tree structures which consist of the logical objects and the content objects connected to the terminal objects of trees. For concurrency control, we adopted the approach to extend the multiple-granularity-locking-scheme. This scheme allows us to lock any objects at each level of the hierarchy. We also defined the locking compatibility table by analysing the operations applicable to any objects at each level of the hierarchy. We finally suggest the extended-multiple-granularity-locking mechanism which uses the locking compatibility table for deciding to lock an object. This scheme gives the benefit to maximize the possibility of concurrent accessing to the shared objects. In addition, we suggest a mechanism for writing authority control which prohibits the Non-Group users from modifying the shared objects based on the concept of Group/Non-Group. The proposed mechanism allows us to protect copyright very reasonably

1. 서 론

그 동안의 컴퓨터 기술과 네트워크 기술의 발달로 멀리 떨어져 있는 여러 사용자들이 화이트 보드, 문서, 데이터베이스 등의 공유 객체를 대상으로 하여 동기적 공동작업을 할 수 있게 되었다. 실시간으로 공동 작업하는 시스템에서는 여러 사용자로부터 발생한 이벤트들이 서로 순서가 바뀌어 도착할 수 있고, 이것은 공유 객체의 일관성을 유지하거나 사용자들이 실제 작업

※ 본 논문은 인천대학교 '97 교내 연구비 지원으로 수행되었음.
† 준 회 원 : 인천대학교 대학원 컴퓨터공학과
†† 종신회원 : 인천대학교 컴퓨터공학과 교수
논문접수 : 1999년 3월 4일, 심사완료 : 1999년 12월 19일

상황에 대한 기대를 만족시키는데 많은 어려움을 가지고 있다[1]. 그러므로 실시간 공동작업 시스템에서는 사용자들의 연산을 동기화하는 동시성 제어(concurrency control) 메커니즘이 필수적으로 요구된다[2].

실시간 공동작업 시스템에서 이용되는 공유 객체들은 여러 저자에 의해 만들어져 여러 시스템에 분산되어 있을 수 있다. 이들 공유 객체의 저자들은 자신이 창조한 객체들이 그 객체를 다룰 권한이 없는 사용자에 의해 그 내용이 보여지거나 고의 또는 실수로 훼손되지 않도록 보호하는 쓰기 권한 제어(writing authority control)의 필요를 느끼게 된다.

여러 공동 작업 중 가장 활용도가 높고 중요한 것이 공동 저작이다. 이 연구에서는 공동 작업 중 공동 저작에 초점을 맞추어 공동저작 시스템에서의 동시성 제어와 쓰기 권한 제어를 위한 효율적인 방안을 제시하고자 한다.

본 논문의 2장에서 공동 작업 시스템에서 사용되는 동시성 제어방법과 문제점을 논한다. 3장에서는 우리가 만든 공동 저작 도구인 MissCW에서 제어와 관련된 시스템 구조에 대해서 논한다. 4장에서는 사용자들에게 보다 많은 동시성을 부여하기 위한 방법을 논한다. 5장에서는 각 사용자의 객체 권한에 따른 접근 제어방법을 논한다. 6장에서는 결론 및 앞으로의 연구 방향을 논한다.

2. 공동작업 시스템의 동시성 제어 기법들

동시성 제어 기법은 데이터의 일관성을 유지하면서 가능한 한 여러 사용자가 공동 문서에 병행적으로 접근하여 연산을 수행하도록 하는 공동 작업 시스템의 구현상의 한 기법이다. 기존의 공동저작 시스템이나 데이터베이스에서 쓰였던 동시성 제어 방법에는 직렬화를 이용한 방법, 잠금을 이용한 방법, 다중 단위 잠금을 이용한 방법, 사용자 인식을 이용한 방법 등이 있다.

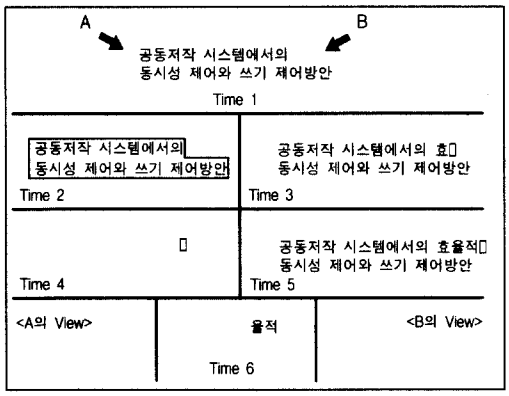
2.1 직렬화(serialization)를 이용한 동시성 제어

이 방법은 병행적으로 발생하는 여러 프로세스의 이벤트들을 직렬화하여 순차적으로 처리하는 기법이다. 이러한 방법은 공동작업 시스템에서는 사용자가 혼동을 일으킬 수 있는 동작을 유발시킬 수 있다. 예를 들어 동시에 둘 이상의 사용자가 같은 객체에 접근하여 수정을 한다면 각 사용자가 보게 되는 해당 객체는 불일치가 일어날 수 있다.

위의 단점을 보완하는 낙관적 직렬화(optimistic serialization) 기법은 이벤트들이 섞이더라도 불일치를 찾아내고 수정할 수 있도록 지원하는 기법이다. 그러나 이 방법 또한 (그림 1)에서 보여지는 다음과 같은 문제점이 발생한다[1].

Time 1 : 사용자A와 B가 동시에 공유 문서 객체에 접근한다.
Time 2 : 사용자A가 그림과 같이 한 문장을 블록으로 설정한다.
Time 3 : 사용자B는 A에 의해 블록화된 문장 중간에 '효'를 삽입한다.
Time 4 : 사용자A는 블록화된 영역을 삭제한다.
Time 5 : 사용자B는 '효' 다음에 '율적'이라는 단어를 삽입한다.

결국 사용자A와 B가 보게 되는 문장은 B가 나중에 삽입한 문장 '율적'뿐이다. 이를 해결하기 위한 연속적인 undo처리는 복잡하고 많은 비용이 든다[2, 3].



(그림 1) 낙관적 직렬화의 문제점

2.2 잠금(locking)을 이용한 동시성 제어

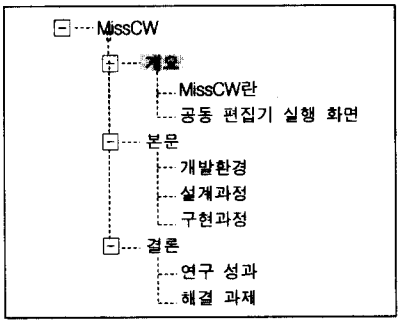
이 방법은 임계 영역 안에 잠금 키를 가지고 들어가 작업을 하는 것이다. 즉 먼저 객체의 이용 권한을 얻은 사용자는 다른 사용자로 하여금 다른 사용자의 간섭 없이 객체를 수정, 편집 할 수 있게 하는 기법이다. 이 방법은 구현이 간단하여 공동 작업 시스템에서 가장 많이 사용하는 방법이기는 하나 잠금 상태에 있는 객체를 접근하기 위해 장시간 기다려야 하는 단점이

있다[4].

위의 단점을 개선한 낙관적 잠금(optimistic locking) 기법은 사용 권한을 얻기를 기다리는 동안 미리 그 객체에 대해 조작할 수 있도록 지원하는 기법이다. 잠금을 요청한 후에 잠정적인 승인을 얻어 임시적인 연산을 수행하다가 요청한 잠금 권한을 얻으면 실제로 그 연산을 실시한다. 만약, 요청한 잠금 권한을 얻지 못하면 미리 수행한 내용은 undo 처리한다.

2.3 다중 잠금(multiple granularity locking)을 이용한 동시성 제어

공동 작업 시스템에서 사용되는 공동 문서들은 많은 경우 계층적 구조를 이루고 있다. 예를 들어 하나의 문서는 (그림 2)와 같은 계층 구조를 가질 수 있다. 이러한 계층 구조의 성격을 지닌 공동 문서에서는 계층 구조상의 어느 한 계층에 대한 접근을 제한함으로써 그 하위 계층을 모두 이에 따르게 할 수 있다. (그림 2)와 같은 구조에서 회의에 참여한 사용자 A가 「개요」라는 계층 전체에 사용을 제한한다면 「개요」 이하 객체들의 접근이 불가능하다. 그러나 사용자 A가 「개요」라는 계층에 사용 제한을 하려는 시점에 다른 사용자에 의해 그 하위 계층에 있는 객체가 사용 중이라면 이러한 동작은 허락되지 않아야 한다.



(그림 2) 계층구조를 가지는 문서의 예

이렇게 계층적으로 구성된 객체들에 대해서 각 계층별로 잠금을 요청할 수 있는 방법이 다중 잠금 기법(multiple granularity locking) 이다[5]. 이 방법에서는 최적의 동시성 제어 크기(granularity of concurrency control)가 중요한 문제로 대두된다. 잠그는 단위의 크기가 작을 수록 여러 사용자가 동시에 작업하는 기회가 늘어나는 장점이 있는 반면에 잠금 처리의 비용이 많이 든다.

이 방법은 객체를 어느 한 사용자가 관리하게 할 경우 객체의 일관성을 쉽게 유지 할 수 있는 반면, 세분화된 객체들의 동시성을 제한하는 단점이 있다.

2.4 객체에 접근 제한을 부여하는 동시성 제어

이 방법은 잠금과 동시에 적용 될 수 있는 방법으로 객체를 소유하고 있는 사용자나 객체 관리를 하는 관리자가 다른 사용자들이 객체에 접근할 수 있는 권한을 부여할 수 있게 하는 방법이다. 이를 위해서 사용자가 권한을 부여할 수 있는 수단과 다른 사용자가 접근이 가능한지를 인지할 수 있는 식별자(indicator)가 필요하다.

2.5 사용자 인식(user awareness)을 이용한 동시성 제어

사용자의 인식을 이용한 동시성 제어 방법은 사용자가 어떤 객체를 어떤 용도로 사용하고 있다는 것을 다른 사용자에게 알리는 방법으로 사용자가 같은 객체의 동일한 작업에 대한 접근을 하지 않을 것으로 기대하는 것이다.

이 방법에서도 현재 객체가 사용 가능한지 알아 볼 수 있는 식별자가 필요하다. 즉 어느 한 사용자가 임의의 객체에 대하여 수정, 편집 등의 작업을 한다면 그 객체가 해당하는 영역의 배경색을 변화시키는 등의 방법으로 다른 사용자에게 자발적으로 해당 객체에 대한 접근을 배제시키는 효과를 도모하는 것이다[6].

3. 공동 저작 모델

이 장에서는 공동 저작 시스템에서의 동시성 제어 문제를 살펴보기 전에 본 연구에서 구현한 다중 사용자 동기적 공동 저작 시스템인 MissCW(multiuser interactive system for synchronous collaborative writing)의 전반적인 시스템 구조에 대해 간단히 설명한다[7, 8].

3.1 공동 편집기

MissCW에서 다루는 모든 문서들은 논리 객체 단위로 분해(decomposition)된다. 한 논리 객체는 논리 객체 단위로 다시 분해될 수 있다. 이렇게 구성된 트리 구조의 단말에 그 문서의 내용조각들이 연결되게 된다[7]. 그러므로 MissCW의 공동 편집기는 문서의 논리적 구조를 나타내는 트리를 편집할 수 있게 해주는 구

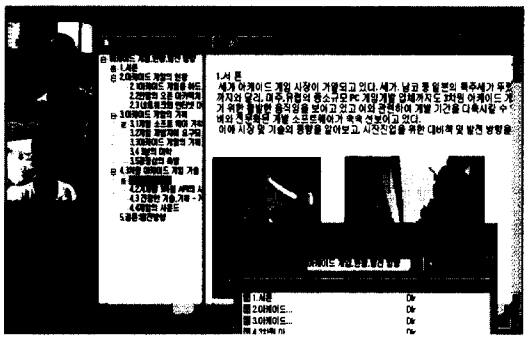
조 편집기와 단말에 연결된 내용조각을 편집할 수 있는 내용 편집기를 제공한다. 또한 새로운 공유 객체를 가상 디렉토리(3.3절 참조)에 등록 시켜주는 객체 등록기, 이미 만들어진 객체들의 목록과 속성 및 내용을 열람 할 수 있게 해주는 객체 열람기, 그리고 문서 구조에 정의된 대로 내용 조각들을 찾아와 페이지로 자동 변환 시켜주는 페이지 출력기로 구성된다.

3.2 공유 객체 관리자

공유 객체 관리자는 미들웨어로서 객체들을 일관성 있게 유지하며 응용 프로그램이 객체들을 효율적으로 이용할 수 있도록 도와주는 일을 한다. 공유 객체 관리자는 객체 제어기와 객체 전송기로 구성된다. 객체 제어기는 가상 노드를 형성하여 총체적인 제어를 담당한다. 가상 노드는 물리적으로는 한 사이트의 공유 객체 관리자 내부에 존재한다. 그리고 각 사이트에는 객체 전송기가 있어 객체의 내용을 빠르게 전송하는 일을 한다.

3.3 가상 디렉토리

MissCW에서는 등록된 공유 객체의 용이한 검색을 위해 (그림 3)에서 보는 바와 같이 공유자 디렉토리와 공동문서 디렉토리로 구성된 가상 디렉토리를 지원한다. 공유자 디렉토리는 공유자 이름으로 분류된 객체의 속성을 검색할 수 있는 구조를 말한다. 공유문서 디렉토리는 구조 편집기로 구성한 문서의 논리 구조대로 분류된 객체의 속성을 검색할 수 있는 구조를 말한다. 가상 디렉토리는 공유 객체 정보에 대한 시각화의 의미와 가상 디렉토리의 상속성을 이용하여 효율적으로 잠금 처리를 할 수 있게 하는 효과가 있다[8].



(그림 3) 객체 열람기의 가상 디렉토리

4. 동시성 제어 방안

동시성 제어 방안에는 극단적인 두 가지 방법이 있다. 데이터 베이스에서 사용되는 엄격한 방법과 동시 접근에 대해 아무런 제한을 하지 않는 방법이 있다. 우리는 중간정도의 제어성을 가지는 동시성 제어방법에 대해 논의한다.

동시 접근을 극대화하기 위해 문서를 계층적 구조로 분해하였으며, 문서를 작은 단위로 분해하는 것은 동시성 제어의 크기(granularity of concurrency control)를 줄이는 효과가 있으며, 이 크기가 작을수록 여러 사용자가 동시에 작업하는 기회가 늘어난다.

이 논문에서는 가능한 한 많은 사용자에게 동시 작업의 기회를 더욱 늘려 주기 위해 논리적으로 분해된 객체에 수행되는 작업들을 세분화 시켜 다중 잠금 기법(2장 참조)의 확장을 시도하였다. 다시 말하면, 텍스트의 경우 객체의 변경(Modify), 읽기(Read), 변경 상황 보기(Modify-Read), 이동(Move) 등의 객체에 대한 연산을 응용프로그램 상에서 각각 독립된 작업으로 분리함으로써, 동시성의 효율을 극대화시키고자 하였다.

<표 1>은 이와 같은 연산 세분화의 결과로 정의된 잠금 호환 테이블이다. <표 1>에 나타나있는 Exclusive와 Share 잠금은 해당 객체의 저작자가 부여할 수 있는 권한으로, Exclusive로 지정되면 다른 사용자는 그 객체에 대해 어떠한 영향도 주지 못한다. Share로 지정된 객체는 여러 사용자가 Modify, Read, Move, Modify-Read 등의 잠금을 사용해 작업을 할 수 있다.

〈표 1〉 잠금 호환 테이블

Lock Request	Lock held						
	Ex-clusive	Share				Delete	Insert
		Modify	Read	Move	Modify-Read		
Exclusive	No	No	No	No	-	No	No
Modify	No	No	Yes	Yes	-	No	Yes
Read	No	Yes	Yes	Yes	-	No	Yes
Move	No	Yes	Yes	No	-	No	Yes
Modify-Read	No	Yes	No	No	-	No	Yes
Delete	No	No	No	No	-	No	No
Insert	No	Yes	Yes	Yes	-	No	No

예를 들어, Share인 어떤 객체에 대해 한 사용자가 그 객체에 Modify를 하고 있던 도중, 다른 사용자가

그 객체에 대해 Read나 Move등의 변화를 준다고 가정해보자. 이 경우 응용프로그램은 Modify를 하고 있는 사용자 이외의 다른 사용자가 객체에 대해 수행한 Read나 Move등의 작업을 현재 Modify를 하고 있는 사용자의 화면상에는 적용하지 않으면서 Read나 Move 등의 작업을 수행할 수 있는 환경을 제공해 준다. 이 때, Modify 중인 사용자 이외의 다른 사용자가 수행한 객체에 대한 변화는 지역 프로그램(local program)내부에서 처리되어 저장되며, Modify 작업이 끝나게 되면 저장되어 있는 변화 내용을 참조하여 객체를 변경하고 이를 화면에 표시하게 된다. 여기서 사용자 인식을 이용한 동시성 제어 기법을 부가적으로 사용해서 다른 사람들로 하여금 객체가 변환 중이라는 것을 알게 함으로써 Modify-Read를 하거나 접근을 포기하게 한다.

5. 쓰기 권한 제어 방안

사용자들은 공유 객체들에 대해 빠른 접근과 조작을 원하는 반면, 자신이 작성한 문서에 대해 보호 받기를 원한다. 이 절에서는 <표 1>의 잠금 호환 테이블에서 정의한 여러 권한 중 Modify권한에 한해서 동시성을 제한함으로써 저작권을 보호하는 쓰기 권한 제어 방법을 논의한다.

우선 각 객체의 저작자에게 저작권을 부여하여 자신의 공유 객체에 대해서만 Modify 권한이 주어지게 한다. 그리고, 리프(leaf)를 제외한 복합 객체를 구성하는 노드들에 대해서 그룹 모드(Group mode)와 비그룹 모드(Non-Group mode)를 정의 할 수 있게 하였다. 이것은 사용자들이 속한 그룹 별로 객체의 Modify권한의 범위를 부여하는 것이다. 예를 들어 초기 문서 상태에서 한 사용자가 어떤 노드에 그룹화를 시도하면, 그 사용자는 그룹화된 노드를 따라 하위에 연결된 모든 문서 객체에 Modify권한을 가지며 나머지 사용자는 자신의 객체에만 Modify권한을 가질 것이다. 이것은 그룹화를 시도한 사람이 해당 복합객체 대한 수정에 있어 리더가 되며 나머지 사용자들은 자신의 객체에 대해 리더가 된다는 의미로 볼 수 있다. 그룹화 하는 순간 그룹에 속하는 사람들의 그룹 테이블이 생성된다. <표 2>는 그룹/비그룹 모드에 따른 접근 권한을 보여준다.

만약, 객체 권한에 대해 각 사용자마다 우선순위(the order of priority)가 부여되어 있다고 가정하면, 위의 <표 2>에서 한 사용자가 한 객체를 G(Group)로 설정했을 때, 우선순위가 높은 다른 사용자가 G 잠금을 요청한다면 Yes도 가능할 것이다.

〈표 2〉 그룹/비그룹 잠금 테이블

Lock Request	Lock held	
	G(Group)	NG(Non-Group)
G	No	Yes
NG	No	Yes

다음의 (그림 4)와 (그림 5)는 그룹화 과정 및 사용자가 공유 객체에 접근하는 과정을 처리하는 알고리즘을 나타낸다. 사용자가 공유 객체에 접근하는 과정은 우선, 사용자가 접근한 객체가 그룹화 되어 있는지 조사하고 자신이 그 그룹에 속해 있고 리더인지 확인한다. 자신이 리더임이 확인되면 잠금 호환 테이블을 참조하여 Modify를 수행한다. 자신이 리더가 아닐 경우에는 사용자가 접근한 객체가 자신의 객체일 경우 잠금 호환 테이블을 참조하여 Modify를 수행한다.

```
DECLARATION PART
class TREE          // 트리 클래스
pos                 // 사용자가 그룹화 하고자 하는 트리의 위치
class GT            // 그룹 테이블
PROCEDURE PART
if IsGroup(TREE.pos) then return
        else GT = MakeGroupTable(User)
// 그룹테이블이 아니라면 그룹을 형성하고 그룹테이블을 만든다.
FUNCTION DECLARATION PART
function IsGroup(TREE.pos)
 if pos is in GruopArea then return TRUE
                    else return FALSE
function MakeGroupTable(User)
    Create GT
    GT.Register(User) // 그룹의 리더로 설정..
    return GT
```

(그림 4) 객체의 그룹화 알고리즘

```
DECLARATION PART
User                // 객체에 접근하고자하는 사용자
PATH                // 단일 경로명
class OBJECT        // 객체 클래스
class GT            // Group Table 클래스
class CLN           // 전체 경로명 클래스
class SOT           // 공유객체 테이블 클래스

PROCEDURE PART
```

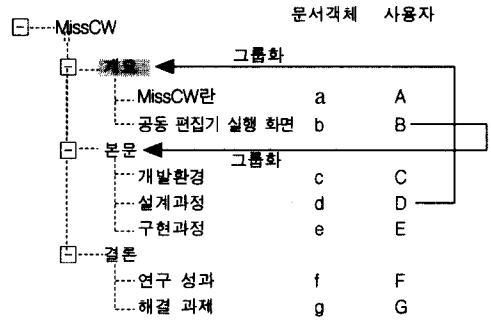
```
CLN = MatchSharedObjTable(SelectTreeObject(User))
// 사용자가 선택한 객체를 가지고 공유객체 테이블 클래스로부터
// 완전한 경로명을 얻음.
while((PATH = CLN.GetNext()) != NULL)
{// PATH가 NULL일 때까지 바로 다음의 경로를 얻음.
    TREE = SOT.MoveTree(PATH)
    // 공유객체테이블의 현재 위치를 PATH객체가 포함하
       는 경로로 이동.
    GT.GetGroup(TREE)          //그룹테이블을 받음.
    if IsGroup(TREE.pos)       // 트리가 그룹화 되었는지
                                  확인.
      then if GT.IsMember(User) & GT.IsLeader(User)
             then ReferenceLockTable()
             else retrun
      else if IsLeaf(TREE.pos) & GT.IsLeader(User)
   // 트리가 단말노드(객체) 이면서 자신의 객체인지를 확인.
             then ReferenceLockTable()
}

FUNCTION DECLARATION PART
function SelectTreeObject()
      return OBJECT
function MatchSharedObjTable(OBJECT)
      return CLN
function GT.IsMember(User)
  if User is Member then return TRUE
                 else return FALSE
function GT.IsLeader(User)
  if User is Leader then return TRUE
                 else return FALSE
function CLN.GetNext()
      return PATH   // 전체경로 내에 존재하는 현재
                       위치에서
                    // 다음의 단일 경로를 리턴
function SOT.MoveTree(PATH)
      return TREE   // PATH가 포함하는 단일 경로로
                       부터 시작하여
                    //하위 경로를 모두 포함하는 TREE
                       객체를 리턴
function ReferenceLockTable()
                    // 잠금 호환 테이블을 참조하여 접
                       근권한 여부를 판별
```

(그림 5) 객체의 접근 알고리즘

다음의 (그림 6)은 모든 사용자가 자신의 단말 객체를 각각 하나씩 가지고 있으며, 사용자 B가 본문 부분에 그룹화를 하고 사용자 D가 개요부분에 그룹화를 하는 모습을 보여주는 그림이다.

(그림 6)에서 사용자 B만 그룹화를 시도했을 경우, (그림 4)의 알고리즘에 따라 <표 3>과 같이 그룹 테이블이 생성된다. 이 그룹 테이블은 그룹의 멤버들의 문서에 대한 접근 권한 정보를 가지고 있다.



(그림 6) 계층적 문서 구조에서의 그룹화

〈표 3〉 사용자 B의 그룹화 결과로 생성된 그룹 테이블

사용자	접근 가능 문서 객체
B	b, c, d, e
C	c
D	d
E	e

또, (그림 6)에서 사용자 D만 그룹화를 시도했을 경우, (그림 4)의 알고리즘에 따라 <표 4>과 같이 그룹 테이블이 생성된다.

〈표 4〉 사용자 D의 그룹화 결과로 생성된 그룹 테이블

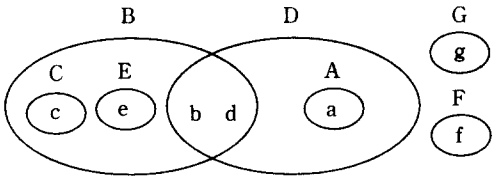
사용자	접근 가능 문서 객체
D	a, b, d
A	a
B	b

〈표 5〉 사용자 B와 사용자 D의 그룹화 결과로 생성된 그룹 테이블

사용자	접근 가능 문서 객체
A	a
B	b, c, d, e
C	c
D	a, b, d,
E	e
F	f
G	g

사용자 B와 사용자 D가 모두 (그림 6)에서와 같이 그룹화를 시도했을 경우, 모든 사용자들의 쓰기 권한

은 <표 5>와 같으며, 이것의 포함 관계를 나타내면 (그림 7)과 같다. (그림 7)에서 보는 바와 같이 객체 a에 대해서는 사용자 A와 D에게 쓰기 권한이 주어지며 객체 b에 대해서는 사용자 B와 D에게 쓰기 권한이 주어진다. 만일 객체 b에 대하여 쓰기 권한을 가진 사용자 B, D가 동시에 작업을 요청하게 되면 <표 1>의 잠금 호환 테이블에 준하여 동시성 제어를 하면 된다.



(그림 7) 쓰기권한 포함 관계

6. 결 론

이 논문에서는 공동저작 시스템에서의 동시성 제어와 쓰기 권한 제어를 위한 효율적인 방안을 제시하였다. 동시성 제어 기법으로 이 연구에서 제안하는 방법은 확장된 다중 잠금 기법이다. 확장된 다중 잠금 기법은 계층적으로 구성된 객체들에 대한 연산을 세분화하여 잠금 호환 테이블을 정의하고, 이 테이블을 기반으로 잠금을 허락하는 방법으로 공유 객체에 대한 동시 접근의 가능성을 극대화할 수 있다. 한편, 쓰기 권한 기법으로는 그룹/비그룹 개념을 적용하여 비그룹 사용자들의 쓰기를 방지하여 합리적으로 저작권을 보호하는 매커니즘을 제안하였다. 이 방법은 객체를 어느 한 사용자가 관리하게 하여 객체의 일관성을 쉽게 유지할 수 있게 한다.

현재의 공동 작업 시스템들에는 여러 가지 동시성 제어 방법이 이용되고 있지만 이들은 고유한 장단점들이 있기 때문에 어떤 방법이 가장 효율적이라고는 말하기는 힘들다. 따라서 사용자의 시스템 사용 목적이나 사용자의 수 등의 여러 가지 사항을 고려하여 시스템에 알맞는 동시성 제어 방법을 선택하는 것이 가장 효율적이라고 말할 수 있다. 이 연구로 우리가 얻은 결론은, 동시성 제어 기법으로 다중 잠금 기법, 사용자 인식 기법, 접근 제한 기법을 혼합 적용하고 undo기능을 포함하여 낙관성을 높였을 때 가장 이상적으로 동시성 제어가 가능하다는 것이다. 그러나 이 방법은 성능(performance)면에서 효율적이지 못할 가능성이 있다. 향후 연구는 이 논문에서 제안한 동시성 제어 및 쓰기 권한 제어 기법에 대한 체계적이고 이론적인 성능을 평가하는 과제와 성능을 향상시키는 과제에 집중될 것이다.

참 고 문 헌

[1] Saul Greenberg and David Marwood, "Real Time Groupware as a Distributed System : Concurrency Control and its Effect on the Interface," Proceedings on CSCW '94, ACM Press, pp.212-213, October 22-26, 1994.

[2] Matthias Ressel, Doris Nitsche-Ruhland, and Rul Gunzenhauser, "An Integrating, Transformation-Oriented Approach to Concurrency Control and Undo in Group Editors," Proceedings on CSCW '96, ACM Press, pp.295-296, November 16-20, 1996.

[3] ATUL PRAKASH and MICHAEL J. KNISTER, "A Framework for Undoing Action in Collaborative Systems," Proceedings on ACM Transactions on Computer-Human Interaction, pp.300, Vol.1, No.4, December 1994.

[4] Saul Greenberg and David Marwood, "Real Time Groupware as a Distributed System : Concurrency Control and its Effect on the Interface," Proceedings on CSCW '94, ACM Press, pp.211-212, October 22-26, 1994.

[5] Jonathan Munson and Prasun Dewan, "A Concurrency Control Framework for Collaborative Systems," Proceedings on CSCW '96, ACM Press, pp.280, November 16-20, 1996.

[6] 정용득, 최종명, 송후용, "웹미팅 : 인트라넷 그룹웨어 시스템", 1997년도 정보과학회 봄 학술 pp.439-440, 1997.

[7] 성미영, "MissCW : 다중 사용자와 동기적 공동저작 시스템", 한국정보처리학회 논문지 제3권 제7호, pp.1701-1704, 1997.12.

[8] 성미영, 김엄준, 유재홍, 홍지철, 송은주, "MissCW

에서의 공유 객체 검색을 위한 가상 디렉토리의 설계와 구현", 1997년도 정보과학회 봄 학술 논문집 pp.580-582, 1997.

유 재 흥

e-mail : jhyoo@isis.inchon.ac.kr

1998년 인천대학교 전자계산학과 (공학사)

1998년~현재 인천대학교 컴퓨터공학과 석사과정

관심분야 : 멀티미디어 협동 컴퓨팅, 에이전트, 가상현실

성 미 영

e-mail : mysung@lion.inchon.ac.kr

1982년 서울대학교 식품영양학과 (학사), 계산통계학과(계산학 전공) 부전공

1987년 프랑스 INSA de Lyon 컴퓨터공학과(공학석사)

1990년 프랑스 INSA de Lyon 전산학과(공학박사)

1990년~1993년 한국전자통신연구소 컴퓨터연구단 선임연구원

1993년~현재 인천대학교 컴퓨터공학과 부교수

관심분야 : 멀티미디어 협동 컴퓨팅, 멀티미디어 저작, 에이전트, 음성 인터페이스