# 디지털 도서관에서 전자상거래 트랜잭션을 위한 메타데이타 관리 기법

## (Metadata Management for E-Commerce Transactions in Digital Library)

최 일 환 † 박 석 ††
(Ilhwan Choi) (Seog Park)

**요 약** Dublin Core같은 기존의 정적인 메타데이타 집합은 서지 정보 중심의 정적인 데이터 요소를 가지므로 새로운 환경에 적용하기 위해서는 다양한 메타데이타를 위한 메타데이타 간의 통합, 웹 환경에서의 메타데이타의 표준화 문제 및 확장성 문제들이 고려되어야 한다. 특히, 디지털 라이브러리에서 전자상거래의 개념을 포함하며 서로간의 상호운영성을 위해 이벤트중심의 메타데이타 기록 방식이 등장함에 따라 기존 관리기법으로는 갱신 연산간의 차별화가 이루어지지 않아 부적절한 갱신 연산 지연이 발생하므로 이에 대한 고려 역시 필요하다.

본 논문에서는 우선 디지털 라이브러리 환경에서 완화된 트랜잭션 일관성의 적용여부를 보이며, 새로운 방식의 메타데이타 요소를 사용자 질의 트랜잭션의 판독연산에 관련있는 정적 메타데이타 요소와 전자상거래 트랜잭션의 갱신연산과 관련있는 동적 메타데이타 요소로 구분한다. 구분된 메타데이타 요소에 따라, 관련 트랜잭션들을 재분류함으로써 전자상거래 요소와 동적 갱신(전자상거래) 트랜잭션을 고려한 새로운 메타데이타 관리기법을 제안한다. 최소의 유지비용으로 갱신을 고려하는2버전과 동적 갱신 연산 충돌의 최소화를 위한 ARU(Appended Refresh Unit)를 사용함으로써 기록연산 간의 충돌을 최소화해 빠른 응답시간과 높은 최근성 비율을 보이게 된다. 성능분석을 통해, 새로운 메타데이타 환경하에서 제안한 알고리즘이 기존의 알고리즘에 비해 좋은 성능을 가짐을 보인다.

**키워드** : 전자도서관, 병행수행제어, 메타데이타

***Abstract*** Since traditional static metadata set like Dublin Core has static metadata attributes about bibliography information, integration of metadata for various metadata, problems about standard and extension of metadata must be considered for applying it to new environment. Specially, as event-driven metadata write method included the notion of e-commerce come out for interoperability in digital libraries, traditional metadata management which cannot distinguish between different kinds of update operations to new extension of metadata set occurs unsuitable waiting of update operation. So, improvement is needed about it.

In this paper, we show whether alleviative transaction consistency can be applied to digital library or not. Also it would divide newer metadata into static metadata attribute connected in read operation within user read-only transactions and dynamic metadata attribute in update operation within dynamic(e-commerce) update transactions. We propose newer metadata management algorithm considered in classification of metadata attributes and dynamic update transaction. Using two version for minimal maintenance cost and ARU(Appended Refresh Unit) for dynamic update transaction, to minimize conflict between read and write operations shows fast response time and high recency ratio. As a result of the performance evaluation, we show our algorithm is proved to be better than other algorithms in newer metadata environments.

**Key words** : Digital Library, Concurrency Control, Metadata

· 본 연구는 한국과학재단 목적기초연구(과제번호 : R01-2000-00272) 지원으로 수행되었음.
† 비 회 원 : 삼성전자
yanni@dblab.sogang.ac.kr
†† 종신회원 : 서강대학교 컴퓨터학과 교수
spark@dblab.sogang.ac.kr
논문접수 : 2000년 12월 29일
심사완료 : 2001년 11월 23일

## 1. 서 론

WWW의 폭발적인 증가와 함께, 웹에서 표현될 수 있는 다양한 매체의 자원들이 접근 가능해짐에 따라 다양한 형태의 정보 자원들에 대한 메타데이타의 표준화 문제가 등장하고 있다. 기존의 제한된 형태의 정보를 나타내는 다양한 메타데이타 포멧들 간의 상호운영성을 증대시키고, 동적인 정보를 포함하는 전자상거래의 개념을 포함하는 새로운 메타데이타에 관한 연구가 INDECS[1] 프로젝트에서 이루어지고 있다.

기존의 메타데이타는 서지 정보 중심의 정적인 정보만으로 이루어져 변형되지 않으며 갱신이 거의 일어나지 않는 상황을 고려하고 작성되었다. 따라서 새로운 매체의 자원을 표기하기 위해서는 별도의 메타데이타 구조가 필요하게 되었다. 새로운 개념의 메타데이타는 이벤트 중심적인 구조로 이루어지며, 메타데이타는 전자상거래를 위해 필요한 정보를 인코딩할 수 있게 된다. 예를 들면, 정보자원의 메타데이타는 판매자가 누구이고, 구매자는 누구인지, 그리고 어디서 다운로드 했는지의 동적인 정보들과 함께 관리적(지적 재산권 내용 등) 정보를 포함할 수 있게 된다[2].

메타데이타 관리 기법에서는 일관성을 유지하는 범위 내에서 트랜잭션 처리 시간과 최근성 여부가 치명적인 성능 평가 대상으로 간주되고 있는데, 동적인 정보를 포함하는 메타데이타의 등장으로 기존의 정적인 정보를 위한 메타데이타 관리 기법에 대해 개선사항이 요구되고 있다. 기존 논문[3]의 초점은 디지털 라이브러리의 트랜잭션 특징으로 판독 전용 트랜잭션이 대다수를 차지한다고 간주하고 빠른 질의 응답시간을 내며 메타데이타 일관성과 트랜잭션 일관성을 유지하는 관리 기법을 제안하였다. 판독 전용 트랜잭션이 대다수를 차지하는 이유는 기존의 메타데이타가 정적인 정보(주제, 저자, 출판사등)가 다수를 차지하기 때문에 판독 전용 트랜잭션보다 갱신 트랜잭션이 차지하는 비중이 상당히 낮다는 것이다. 하지만 갱신 연산이 거의 없는 환경을 고려하였기 때문에 동적인 갱신 연산이 빈번히 발생될 수 있는 환경에서는 단순히 삽입과 삭제만으로 트랜잭션 처리 시간을 최소화하기 어렵다. 그리고 또 다른 논문[4]은 갱신 연산에 대한 고려를 위해 래치를 이용한 2버전을 유지하는 2VL을 제안했다. 2버전을 유지함으로써 빈번한 갱신연산이 발생할 때 판독과 기록연산 간의 충돌로 인한 지연을 최소화하며, 래치를 사용하여 불필요한 로크의 소유를 제거하여 리프레쉬의 지연을 최소화하였다. 하지만, 전자상거래 상황을 고려한 메타데이타 요소의 특성인 정적인 요소와 동적인 요소간의 구별을 하지않았다. 그리고, 기록 래치를 소유할 때 관련된 모든 메타데이타 요소에 대해서 래치를 취득하므로 정적 메타데이타 요소에 대한 불필요한 래치 획득으로 정적 메타데이타 요소에 대한 갱신은 지연되게 된다. 다운로드, 지불 정보 등의 동적인 요소에 대한 갱신이 빈번히 발생할 때 연관성이 없는 정적인 요소의 갱신에까지 영향을 미치므로 이에 대한 판독연산의 최근성 비율이 심각하게 저하된다.

본 논문은 새로운 메타데이타 모델의 관리에 있어 요구사항을 정의하며 이에 따른 완화된 갱신일관성 기준을 적용하고, 2버전과 ARU(Appended Refresh Unit)를 통한 메타데이타베이스의 효율적인 관리방안을 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 제시하는 메타데이타 관리 기법의 연구 배경이 되는 INDECS 메타데이타의 구조와 관련 병행수행 제어 방법을 소개한다. 3장에서는 디지털 라이브러리의 메타데이타 관리를 위해 메타데이타의 속성을 통한 재분류와 그에 따른 트랜잭션의 재분류, 그리고 트랜잭션의 요구사항들을 정의한다. 그리고 메타데이타 연산을 위한 트랜잭션의 완화된 일관성 기준을 정의한다. 4장에서는 본 논문에서 제안하는 메타데이타 관리 기법을 제시하고 알고리즘의 정확성을 증명한다. 5장에서는 제시한 기법의 성능 평가를 하며 6장에서 결론을 맺는다.

## 2. 연구 배경

이 장에서는 논문의 기반 메타데이타 모델인 INDECS 메타데이타 모델과 관련 병행수행 제어기법인 래치 기반 스케쥴링과 2VL을 설명한다.

### 2.1 INDECS 메타데이타

웹을 통한 전자상거래 기반 환경 구축에 따라 기존의 메타데이타 모델에서 이벤트 기반의 전자상거래 개념을 포함한 새로운 메타데이타 모델을 필요로 하고 있다. 이런 메타데이타 표준에 대한 연구는 INDECS(INtero-perability of Data in E-Commerce Systems) 메타데이타 모델로 현실화되고 있다. 현재 새로운 차세대 URL이라 할 수 있는DOI(Digital Object Identifier)와 연계된 형태로 개발되었으며, 다양한 웹 환경에서의 다양한 자원들에 대한 지원과 표현을 하도록 하고 있다[2].

실세계의 객체를 기술하는 방안에는 다양한 관점이 존재하게 된다. 인덱스 모델은 이러한 많은 다양성이 존재한다는 관점을 인지하는 것에서 출발한다. 인덱스 메타데이타 모델에서는 유용한 메타데이타 모델을 제공하

기 위해 객체를 보는 관점에 있어 크게 네 가지의 관점을 제시한다. 인덱스는 네 가지의 관점을 정의하고 표현하는 방안을 개발하고 있다. 네 가지 관점은 추상적 시각(abstract view), 창조적 시각(creative view), 상거래 시각(commerce view), 법률적 시각(legal view) 이다.

본 논문의 메타데이타 관리 기법은 인덱스 메타데이타 모델을 그 기반으로 삼는다[5]. INDECS 메타데이타 집합은 기존의 메타데이타에 비해 동적인 정보를 유지한다는 속성을 갖는다. 객체들 사이의 관계 및 이벤트를 묘사하고, 상거래 시각을 통해 거래내역등의 다양한 이벤트를 표현하며, 법률적 시각을 통해서는 저작권자의 위임등을 표현한다.

### 2.2 병행수행 제어 기법

2.2.1 래치 기반 스케쥴링

Dublin Core 같은 정적인 메타데이타 모델 환경을 기반으로 하여 갱신이 거의 고려되지 않은 1버전의 기존의 디지털 라이브러리 메타데이타 관리 기법[3]에서는 삽입과 삭제 연산만을 사용한 갱신 연산을 수행하였다. 메타데이타에 수정이 발생할 때마다 삽입 연산과 삭제 연산을 번갈아 수행하게 된다. 이는 메타데이타에 대한 갱신이 거의 고려되지 않은 가정 하에 제안된 관리 기법이다. 따라서 판독 또는 갱신이 데이터의 1버전에서 수행되므로, 판독이나 갱신 연산이 빈번하게 충돌하게 된다. 이는 많은 수의 판독전용 트랜잭션이 상대적으로 적은 수의 갱신 트랜잭션에 의한 지연을 갖게 한다. 즉, 판독전용 트랜잭션의 중요한 성능 평가 요소인 응답 시간에 있어 지연을 초래한다.

2.2.2 2VL

동적 메타데이타 모델인 인덱스 메타데이타에 대한 관리기법인 2VL(2 Version using Latch)[4]는 래치와 2버전을 사용하여 갱신을 고려한 알고리즘으로, 판독 트랜잭션은 기본 버전의 데이터를 읽도록 하고, 갱신 트랜잭션은 신생 버전을 쓰도록 하여 트랜잭션 간의 충돌을 제거하여 응답시간을 줄이도록 한다[6]. 2버전을 사용하는 이유는 가장 최소의 추가비용으로 판독 전용 트랜잭션의 판독 연산과 갱신 트랜잭션의 갱신 연산의 충돌을 막는 방법으로서, 2VL에서는 2버전의 세 가지 상태를 허용한다. 상태 1은 기본 버전만이 존재하는 경우로, 판독 트랜잭션은 기본 버전을 읽게 된다. 상태 2는 갱신 트랜잭션이 신생 버전을 기록하고 있는 상태로, 판독 트랜잭션은 여전히 기본 버전을 읽게 된다. 신생 버전은 현재 기록 중에 있다. 상태 3은 갱신 트랜잭션의 기록이 끝난 상태로, 기본 버전과 신생 버전의 2 버전이 있다. 신생 버전이 생성된 이후의 판독 트랜잭션들은 모두 신생 버전을 읽게 되어서 최근 데이터를 읽을 수 있는 가능성이 높아진다. 기본 버전을 읽는 판독 트랜잭션들이 모두 완료되면 신생 버전을 기본 버전으로 만드는 리프레쉬 작업이 수행되고, 상태 1로 복귀한다[6].

하지만, 2VL은 전자상거래 상황을 고려한 메타데이타 요소의 특성인 정적인 요소와 동적인 요소간의 구별을 하지않았다. 따라서 이에 따른 갱신 트랜잭션의 제분류가 이루어지지 않아 갱신 트랜잭션 상호간의 불필요한 기록 래치 획득으로 트랜잭션 수행 지연을 가져온다. 즉, 동적 갱신 트랜잭션이 동적 메타데이타 요소에 대한 래치를 소유할 때 관련된 모든 메타데이타 요소에 대해서 래치를 취득하게 되어 정적 메타데이타 요소에 대한 불필요한 래치 획득을 가져옴으로 정적 메타데이타 요소에 대한 갱신을 지연되게 된다.

한편, 다운로드, 지불 정보 등의 동적인 요소에 대한 갱신이 빈번히 발생할 때, 신생 버전이 기록되어 있을 때 다른 동적 생신 트랜잭션의 갱신 연산은 단순지연되게 된다.

## 3. 디지털 라이브러리 메타데이타 관리의 요구사항

### 3.1 메타데이타 요소의 분류와 속성 정의

메타데이타 관리에 있어 요구되는 메타데이타들을 크게 두 가지 형태인 정적 메타데이타 요소와 동적 메타데이타 요소로 분류한다.

**[정적 메타데이타 요소 정의]**

기존의 메타데이타 요소로서, 실제 질의 트랜잭션이 요구하는 데이터이다. 갱신등의 연산이 발생할 확률이 적으므로 정적 데이터 요소라 한다. 예를 들면, 주제, 저자, 키워드, 출판사 등이다.

**[동적 메타데이타 요소 정의]**

INDECS메타데이타 모델에서 추가된 요소로서, 네가지 시각들 중 자주 변하는 성격을 지니고, 추가 연산에 적용되는 요소이다. 전자상거래적 트랜잭션 요소의 성격이 강하며, 다운로드, 지불, 권한의 위임등이 예이다. 추가 연산이 빈번히 발생하므로 동적 메타데이타 요소라 한다.

트랜잭션 일관성 기준 판명을 위해 사용될 메타데이타 속성을 정의한다.

**[메타데이타 속성 1]**

메타데이타는 그 구조와 상관없이 상호간의 종속성이 존재하지 않는다. 즉, 메타데이타는 다른 메타데이타로부터 유도된 데이터가 아니다.

**[메타데이타 속성 2]**

메타데이타의 갱신 트랜잭션의 연산 결과가 다른 메

타데이타에 영향을 미치지 않는다.

**[메타데이타 속성 3]**

판독전용 트랜잭션은 메타데이타베이스의 하나의 데이터에 대해 단 한 번 판독한다.

**[메타데이타 속성 4]**

메타데이타 인스턴스 안에서 정적 메타데이타 요소와 동적 메타데이타 요소로 나눈다.

**[메타데이타 속성 5]**

질의 트랜잭션과 연관되는 요소는 정적 메타데이타 요소이다. 동적 메타데이타 요소에 대한 갱신 트랜잭션의 연산은 추가연산이다.

### 3.2 메타데이타 관리에 요구되는 트랜잭션의 분류

3.1절에서 분류한 메타데이타와의 연관성에 따라 메타데이타 관리에 있어 요구되는 트랜잭션들을 세 가지 형태인 판독 전용 트랜잭션과 동적 정보 갱신 트랜잭션, 정적 정보 갱신 트랜잭션으로 분류한다. 다음 [표 1]은 트랜잭션 분류에 따른 세부 내용이다.

표 1 트랜잭션의 분류

| 분류 / 성격 | 정적 정보 갱신 트랜잭션 | 동적 정보 갱신 트랜잭션 | 판독전용 트랜잭션 |
|---|---|---|---|
| 상대적 수행시간 | 단기간 | 단기간 | 장기간 |
| 접근하는 데이터량 | 소량 | 소량 | 다양한 범위 |
| 주요 연산의 종류 | 갱신 | 추가 | 판독 |
| 판독/갱신 데이터의 최근성 요구 정도 | 절대적 | 절대적 | 절대적이지는 않음 |
| 동시 수행되는 트랜잭션 수 | 적음 | 많음 | 많음 |
| 트랜잭션 내용 | 정적 메타데이타의 갱신 | 동적 메타데이타의 추가 | 질의 처리 |
| 연관성 | 질의 트랜잭션과 연관 | 질의 트랜잭션과 연관 고려않음 | 정적 정보와 연관 |

메타데이타의 재분류에 따른 [표 1]의 갱신 트랜잭션들을 소개한다. 갱신 트랜잭션은 접근하는 메타데이타의 범위에 따라 정적 정보 갱신 트랜잭션, 동적 정보 갱신 트랜잭션으로 나눈다.

**[정적 정보 갱신 트랜잭션 정의]**

정적 메타데이타 요소를 갱신하는 트랜잭션으로, 사용자 질의 트랜잭션의 정적 메타데이타 요소에 대한 판독연산과 충돌관계에 있기 때문에 갱신 연산간의 직렬성을 만족해야 하고, 최근성을 고려한다.

**[동적 정보 갱신 트랜잭션 정의]**

동적 메타데이타 요소를 갱신하는 트랜잭션으로, 사용자 질의 트랜잭션의 판독연산과는 충돌관계에 있지 않다. 동적 정보 갱신 트랜잭션의 갱신 연산은 추가 연산의 성격을 지닌다.

### 3.3 트랜잭션 요구사항과 판독전용 트랜잭션의 완화된 일관성 기준

메타데이타베이스 역시 기존의 데이터베이스 시스템에서 고려되어 왔던 일관성 기준을 지켜야 한다. 이를 트랜잭션 일관성 기준이라 한다. 메타데이타베이스는 서로간의 종속성이 없는 데이터를 가지기 때문에, 트랜잭션 일관성 기준중 최소한의 일관성 기준을 요구한다.

판독전용 트랜잭션에 대해 적용될 수 있는 일관성 기준은 다음과 같다.

표 2 판독전용 트랜잭션에 대한 일관성 기준의 형태

| 형태 | 설명 | 다른 관점[7] |
|---|---|---|
| 엄격한 일관성 (strict consistency) | 완료 순서와 일치된 갱신 트랜잭션들의 직렬 순서를 본다 | 다른 트랜잭션들은 트랜잭션 T가 기록한 미완료 데이터를 재기록하지 않는다 |
| 강성 일관성 (strong consistency) | 갱신 트랜잭션들의 직렬 순서를 본다. | 트랜잭션 T는 다른 트랜잭션의 미완료 데이터를 읽지 않는다 |
| 약성 일관성 (weak consistency) | 독립적으로 갱신 트랜잭션들의 직렬 순서를 본다. | 트랜잭션 T는 트랜잭션이 끝나기전에 기록한 데이터를 커밋하지 않는다. |
| 갱신 일관성 (update consistency) | 판독전용 트랜잭션이 보는 값을 생성하는 갱신 트랜잭션들의 집합과 관련하여 판독전용 트랜잭션이 직렬성을 만족한다. | 트랜잭션 T는 다른 트랜잭션의 미완료 데이터를 재기록하지 않는다. |

## 4. 전자상거래 트랜잭션을 위한 메타데이타 관리기법

### 4.1 연구 동기

[그림 1]은 기존 2VL에서 질의 트랜잭션과 갱신 트랜잭션을 병행적으로 수행한 스케줄을 보여준다(X, Y, Z는 메타데이타 인스턴스이고, a,b,c는 정적 메타데이타 요소들이고, d,e는 동적 메타데이타 요소이다). [그림 1]에서 'X' 라는 메타데이타 인스턴스에 대한 갱신 연산을 수행하는 T2, T3와 T4는 서로 다른 메타데이타 요소에 대한 갱신이지만, T2,T3간의 빈번한 갱신 연산 충돌로 인해 관련없는 T4의 갱신 연산까지 지연되게 된다.

[그림 1]에서 질의 트랜잭션 Q1의 시점 2의 R(Xb)에 의한 판독연산에 대한 래치 적용으로 T3의 W(Xd) 연산은 지연되고, 시점 3에서 수행된다. 시점 3에서 수행되었던 W(Xd) 연산의 배타적 기록 래치의 적용 때문에 시점

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Q1 | R(Xa) | R(Xb) | | | R(Za) | R(Zb) | C1 | | | |
| T2 | | | W(Xd) | W(Xe) | | | C2 | | | |
| T3 | | W(Xd) | W(Xe) | | | C3 | | | | |
| T4 | | | | | W(Xc) | | | C4 | | |
| Q5 | | | | | | | R(Xc) | R(Yc) | R(Zc) | C5 |

그림 1 기존 2VL의 스케줄

3의 원래 두 기록 연산들은 시점 4로 지연되고 역시 수행순서상 T2의 R(Xd)가 먼저 수행되게 다음 시점에서 T3의 W(Xe)가 수행된다. 이러한 w-w 연산의 지연현상 때문에 트랜잭션 T4의 시점 4에서의 기록 연산 W(Xc)는 실제 시점 7에서야 수행되게 된다. 따라서 질의 트랜잭션 Q5는 T4의 갱신의 결과를 반영할 수 없게 된다.

### 4.2 데이터 관리 테이블

4.2.1 로크의 할당 단위

이벤트 기반의 메타데이타 모델에 대한 연산을 수행할 때, 하나의 로크로 전체 메타데이타를 할당하는 것은 다양한 이벤트에 대한 요소를 고려할 때 효율적이지 못하다. 만약 일관성을 해치지 않는 범위라면 메타데이타를 재분류함으로써 로크의 할당 범위를 조절하여 트랜잭션 연산간의 충돌을 최소화하여 수행시간을 단축하고, 최근성을 반영할 수 있는 알고리즘의 생성이 가능하다.

표 3 Lock의 단위

| 로크의 할당단위 | 수행 트랜잭션 |
|---|---|
| M(entire Metadata) | 관리적 트랜잭션 |
| S(Static) | 정적 정보에 대한 판독 트랜잭션, 갱신 트랜잭션 |
| D(Dynamic) | 동적 정보에 대한 갱신 트랜잭션 |

[표 3]는 할당 가능한 로크의 단위와 수행 트랜잭션이다.

할당 단위 M은 메타데이타 항목 전체에 대한 할당으로 관리적 트랜잭션( 해당 메타데이타 인스턴스에 대한 종합 및 통계 정보 등에 대한 연산을 필요로 하는 트랜잭션으로, 시스템의 부하가 적을 때 일괄처리 방식으로 진행됨)의 수행에 필요한 단위이다. S는 정적 메타데이타 요소에 대한 할당 단위로, 정적 정보를 읽는 일반적인 판독 연산이나 정적 갱신 트랜잭션의 갱신 연산의 수행에 필요하다. D는 동적 메타데이타 집합에 속하는 데이터 아이템의 갱신에 필요한 단위로, 갱신 트랜잭션에서 추가의 성격을 지니는 갱신 연산의 수행에 필요하다.

4.2.2 로크 호환 테이블

제안한 알고리즘에서는 완화된 일관성 기준인 갱신 일관성 기준을 사용할 수 있음을 이미 증명했다. 이에 따라 제안한 알고리즘에서는 래치(Latch)를 사용한다.

제안한 알고리즘에서는 판독 래치인 RL(Read Latch), 기록 래치인 WL(Write Latch), 추가 래치인 AL (Append Latch)과 리프레쉬 단계임을 알리는 CL (Certificate Latch)을 사용한다. 그리고, 삽입 로크 INS와 삭제 로크 DEL을 사용한다. RL, CL에 대해서는 단기적 공유 래치를 적용하고, WL과 AL, DEL은 배타적 래치를 적용한다. 그리고, OV(One Version)는 데이터의 상태가 1 버전임을 나타내는 마크로써 OV가 SET일 경우에만 기록 래치가 가능하다. FM(Freeze Mark)은 신버전에 대한 AL이 획득가능한지를 나타내는 마크로 FM이 SET되면 이미 추가 연산이 수행중이므로 AL이 획득불가능하고, 지연된다.

| | 판독 전용 트랜잭션 | 정적 갱신 트랜잭션 | 동적 갱신 트랜잭션 |
|---|---|---|---|
| 상태 1 | 구 버전 | 구 버전 | 구 버전 |
| 상태 2 | 구 버전 | 구 버전 | 구 버전 |
| 상태 3 | 신 버전 | 신 버전 | 구 버전 |
| 상태 4 | - | | 신 버전 |

그림 2 제안 알고리즘의 판독 데이터 테이블

제안한 알고리즘에서는 2 버전 데이터의 세 가지 상태를 허용하고, AL 연산에 의한 신버전의 상태를 추가하여 네 가지 상태를 가지게 된다. 상태 1은 버전이 하나만 존재하는 상태이고, 상태 2는 현재 갱신연산에 의해 신버전이 기록중인 상태를 말한다. 상태 3은 신버전의 생성이 끝나 리프레쉬를 기다리는 상태이다. 한편, 상태 4는 동적 갱신 트랜잭션의 AL 연산을 수행한 후를 나타낸다. 본 논문에서는 먼저 생성되어 사용되고 있는 버전을 구 버전(Old Version)이라 하고 기록 연산, 추가 연산에 의해 새로 생성된 버전을 신 버전(New Version)이라 한다. [그림 2]는 각 상태에서 판독전용 트랜잭션과 각 갱신 트랜잭션이 접근하게 되는 데이터 버전이다. 동적 갱신 트랜잭션의 상태 3은 AL에 의한 추가가 발생중인 상황으로 구버전의 판독이 필요하다. 대기중인 갱신 연산이 계속 수행되면 상태 3이 반복된다. ARU의 형성이 끝난 상태 4가 되면 신버전의 판독이 가능해진다. 이로 인한 제안 알고리즘의 로크 호환 테이블은 [그림 3]과 같다.

| Lock requester \ Lock holder | | INS | DEL | RL | WL | AL | CL |
|---|---|---|---|---|---|---|---|
| | INS | ○ | ○ | ○ | ○ | ○ | ○ |
| | DEL | ○ | × | ○ | × | × | × |
| | RL | ○ | ○ | ○ | ○ | ○ | ○ |
| | WL | ○ | × | ○ | × | × | × |
| | AL | ○ | × | ○ | × | × | × |
| | CL | ○ | × | ○ | × | × | |

그림 3 제안 알고리즘의 로크 호환 테이블

(⊿: FM이 RESET이면 AL(Append Latch)를 획득할 수 있고, FM이 SET되면 획득 불가능하다)

- 동적 정보 갱신 트랜잭션의 경우에는, 상태 2를 통해 신버전이 작성된 후에 추가로 갱신 연산이 요청될 경우, AL이 획득가능하면 상태 3으로 접어든다. 대기리스트에 대기중인 갱신연산이 있으면 계속 상태 3이 반복되게 된다. AL 연산이 수행된 후 대기리스트에 대기중인 갱신연산이 없으면 상태 4에 이르게 된다.
- AL 연산은 신버전에 대한 추가 연산으로, AL연산의 획득은 OV가 RESET되고, FM이 RESET될 때만 가능하다. 신버전에 대한 AL 연산은 배타적 래치를 사용하기 때문에 해당 데이터에 대한 연산의 직렬성이 유지된다.

4.2.3 래치 관리 테이블

관독전용 트랜잭션에 대한 갱신 일관성 기준 적용으로 래치 사용이 가능하지만, 래치는 연산이 수행되는 동안에만 페이지에 걸리는 단기 로크이고, 로크 관리자(Lock Manager)에 의해 관리되지 않아서 별도의 관리 방법이 필요하게 된다. 따라서 이를 스케쥴링 알고리즘에 반영하고, 이를 지원할 데이터의 관리 데이터들이 필요하게 되었다. 이에 대해 [그림 4]의 래치 테이블의 정보를 유지함으로써 연산 수행 순서의 직렬성을 유지한다.

| 데이터 | OV | 구 버전 관독 리스트 | 신 버전 관독 리스트 | 기록 리스트 | AL 리스트 | FM | 대기 리스트 | 리프레쉬 리스트 |
|---|---|---|---|---|---|---|---|---|
| X | S | Q1, Q4 | | | | R | | |
| Y | R | Q2 | | T5 | | R | T8 | |
| Z | R | Q3 | | | T7 | S | | T6 |

그림 4 래치 관리 테이블

완화된 일관성 기준으로 인해 트랜잭션 전체 데이터가 아닌 각 데이터에 대해서 직렬성을 유지하도록 하기 위해 데이터에 대한 리스트의 정보를 포함한 래치 테이블의 정보를 유지한다. [그림 4]에서 OV는 해당 데이터 버전을 알려주고, 기록 로크의 획득 여부를 알려주는 마크이다. 신 버전 관독 리스트는 신버전에 관독 래치를 소유하고 있는 트랜잭션 아이디들의 리스트로, AL의 획득 여부를 검사하는데 사용된다. 기록리스트는 신버전을 기록중인 트랜잭션의 아이디를 저장하게 되고, 신버전을 기록한 후 트랜잭션의 아이디는 리프레쉬 리스트로 이전된다. AL 리스트는 동적 메타데이타 요소에 대해 현재 추가 연산을 수행중인 트랜잭션의 아이디를 나타낸다. FM(Freeze Mark)은 신버전에 대한 추가연산의 여부를 알려주는 것으로, SET은 추가 연산이 진행중임을 알리고, 추가 연산의 획득 및 리프레쉬 연산의 수행을 금지한다. 한편, 리프레쉬 리스트는 데이터에 대해 갱신 또는 추가한 순서대로 트랜잭션 아이디를 유지하는 것으로, 직렬 순서상에서 뒤의 리프레쉬 연산이 선행하는 리프레쉬 연산보다 먼저 수행되지 않도록 함으로써 직렬성을 유지하게 된다.

### 4.3 관독전용 트랜잭션

관독연산은 갱신 일관성 기준에 의하여 래치를 사용한다. 관독연산에서 수행되는 메타데이타의 범위에 맞는 관독 래치를 설정하고, 이를 클라이언트의 로컬 영역으로 복사하여 사용하고, 복사가 끝나면 관독 래치는 해제된다. 한편, 각 관독 연산은 4.2.3절의 관독 데이터 테이블을 통해 관독 가능한 버전을 선정하게 된다.

### 4.4 갱신 트랜잭션

4.4.1 정적 정보 갱신 트랜잭션의 수행

정적 정보 갱신 트랜잭션의 연산은 관독 연산과 기록 연산으로 나누어 수행하고, 기존의 2VL 알고리즘의 갱신 트랜잭션의 수행을 따른다.

[그림 5]는 정적 정보 갱신 트랜잭션에 의한 버전 생성단계를 보여 주고 있다.

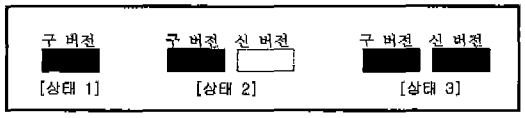


그림 5 정적 정보 갱신 트랜잭션에 의한 버전 생성

상태 1에서 관독 트랜잭션과 정적 정보 갱신 트랜잭션은 구 버전을 관독한다. 갱신 연산에 의해 WL을 획득하고 OV를 RESET 한 후 상태 2로 진행한다. 신버전이 생성되는 상태 2동안, RL에 대한 요구는 구버전을 읽게 되고, 새로운 WL에 대한 요구는 래치 호환 테이

블에 의해 획득되지 못한다. 상태 3에서는 두 개의 버전이 공존하게 되지만, 판독 데이터 테이블을 통해 해당 시점이후 RL에 대한 요구는 신버전을 읽게된다. 마지막으로 래치 관리 테이블을 참조하여 구 버전에 대한 판독 연산들이 완료되면 리프레쉬하여 상태 1로 돌아간다.

4.4.2 동적 정보 갱신 트랜잭션의 수행

동적 정보 갱신 트랜잭션 수행은 정적 정보 갱신 트랜잭션의 수행과 다르게 이루어진다. 먼저 [그림 6]을 보면, 동적 정보 갱신 트랜잭션 수행 상태 중 상태 3이 상태 3과 상태 4로 나뉜다. ARU를 위한 AL의 획득으로 신버전에 대한 추가 연산이 가능해졌기 때문이다.


그림 6 동적 정보 갱신 트랜잭션에 의한 버전 생성

[상태 2]를 통해 생성된 신버전은 대기중인 동적 갱신 연산이 있을 경우, 기록을 위해 FM을 확인하고, RESET이면 AL을 획득하고 ARU를 생성한 뒤, 상태 3으로 이동한다. 연산의 수행이 끝나면 대기리스트의 대기 연산을 확인하고, 대기중인 연산이 없으면 상태 4로 이동하고 리프레쉬를 준비하게 된다. 이때 판독 연산은 [상태 3]에서도 구버전의 데이터를 읽게 된다. 이는 동적 메타데이타 요소의 속성상 질의 트랜잭션의 판독 연산에 영향을 받지 않기 때문에 최근성 여부가 크게 중요하지는 않다.

4.4.3 ARU 생성 방법

ARU(Appended Refresh Unit)으로 갱신연산에 의해 생성된 신버전에 추가연산에 의한 데이터가 추가된 상태이다. 즉, 한 메타데이타 아이템에 대해 갱신 연산이 빈번하게 발생하고 충돌하는 상황일 때, 갱신 연산을 실행하고 리프레쉬를 완료할 때까지 갱신연산을 지연하지 않고, 신버전에 갱신 연산을 추가연산으로 적용하여 수행한 결과가 추가로 통합되어 있는 형태가 바로 ARU이다. 리프레쉬 수행동안의 시간과 대기중인 갱신연산의 단순 지연에 대한 비용손실이 줄이게 된다. 이때 갱신연산은 배타적 연산으로 반드시 직렬 순서대로 실행되므로 직렬성을 지키게 된다. 다음의 조건이 만족하면 갱신 연산이 추가 연산을 획득해 ARU를 생성할 수 있게된다.

**[ ARU생성을 위한 AL획득 조건 ]**

① 갱신 연산의 데이터 범위가 동적 메타데이타 요소이다.

② 신버전이 생성되어 있고, FM이 RESET이다.

③ 관리적 트랜잭션에 의해 로크되어 있지 않다.

**[ ARU생성을 위한 AL연산 수행 ]**

① FM을 SET한다.

② 신버전에 대한 AL 연산을 시행하여 ARU를 생성한다.

③ 래치 관리 테이블에 리프레쉬 리스트를 추가한다.

④ FM을 RESET한다.

빈번한 갱신이 수행되었을 경우, ARU는 여러 갱신 연산의 결과가 합쳐진 형태가 된다. 이때에도 AL연산은 배타적 연산으로 대기중이었던 시간의 순서대로 수행된다. 따라서 ARU에 대해 AL 연산은 직렬성이 보장된다.

4.4.4 갱신 트랜잭션의 리프레쉬

리프레쉬 작업은 제안 알고리즘에서 주의깊게 고려되어져야 할 부분중의 하나이다. 본 절에서는 리프레쉬 조건과 리프레쉬 과정으로 나누어서 알아보기로 한다. 먼저 리프레쉬 조건을 알아보기 위해 필요한 용어를 정의하기로 한다.

- OldVer_ReadNum : 래치 관리 테이블에서 충돌하는 데이터의 구 버전 판독 수
- SET_Prev_Refresh(n)=SET_Prev_Refresh(a)∪SET_Prev_Refresh(b)∪…∪SET_Prev_Refresh(m) : 충돌하는 데이터간의 직렬 순서상에서 리프레쉬 수행 순서 n이전에 위치하는 리프레쉬 집합들
- SET_Cur_Refresh(n) : 현재 리프레쉬 수행 순서 n의 리프레쉬 집합

다음은 정적 정보 갱신 트랜잭션의 RU(Refresh Unit)와 동적 정보 갱신 트랜잭션의 ARU(Appended Refresh Unit)의 리프레쉬를 수행할 수 있는 조건에 대한 것이다.

**[ 리프레쉬 수행 조건 ]**

**■ 정적 정보 갱신 트랜잭션**

① SET_Prev_Refresh(n)에 속한 모든 트랜잭션들이 이미 리프레쉬되어야 한다.

② 구버전을 판독하는 트랜잭션이 없어야 한다.
(OldVer_ReadNum = 0) ∧(SET_Prev_Refresh(n) > SET_Cur_Refresh(n))

**■ 동적 정보 갱신 트랜잭션**

① SET_Prev_Refresh(n)에 속한 모든 트랜잭션들이 이미 리프레쉬 리스트에 기록되어 있어야 한다.

② 구버전을 판독하는 트랜잭션이 없어야 한다.

③ ARU를 위한 FM이 RESET 상태에 있어야 한다.

④ 대기중인 갱신 연산이 없어야 한다.

(FM == RESET) ∧ (OldVer_ReadNum = 0) ∧ (Waiting Write operation don't exists)∧ ((SET_Prev_Refresh(n) > SET_Cur_Refresh(n))

직렬성을 위하여 리프레쉬는 선행 트랜잭션의 리프레쉬가 끝나야 수행할 수 있다.

이러한 조건이 만족되면 리프레쉬 단계로 들어간다.

[그림 7]은 제안한 알고리즘에서 가능한 스케쥴로서, 기존 2VL의 스케쥴인 [그림 1]의 상황을 제안한 알고리즘으로 해결한 상황을 나타낸다. 제안한 알고리즘에서는 이런 문제를 해결하여 T2와 T3의 ARU적용으로 갱신 연산의 충돌에 의한 단순 지연을 막아 수행시간이 크게 단축되었다.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Q1 | R(Xa) | R(Xb) | | | R(Za) | R(Zb) | C1 | | | |
| T2 | | | W(Xd) | W(Xc) | C2 | | | | | |
| T3 | | W(Xd) | W(Xe) | | C3 | | | | | |
| T4 | | | | | W(Xc) | C4 | | | | |
| Q5 | | | | | | | R(Xc) | R(Yc) | R(Zc) | C5 |

그림 7 제안한 알고리즘으로 가능한 스케쥴

다음 [표4]는 논문을 통해 살펴본 기존 알고리즘과 제안한 알고리즘과의 비교 사항이다.

## 5. 성능 평가

### 5.1 평가 모델

본 논문에서 제안하는 알고리즘의 성능을 평가하기 위해 [그림 8]의 시스템 모델[8]을 사용하였다. 그리고, 시뮬레이션 툴로는 AweSim[9]을 이용하였다.

[그림 8]의 트랜잭션 생성부에서는 랜덤하게 생성된 데이터와 판독 또는 갱신 연산을 갖는 트랜잭션을 생성한다. 생성된 트랜잭션은 일단 대기큐로 이동하여 차례로

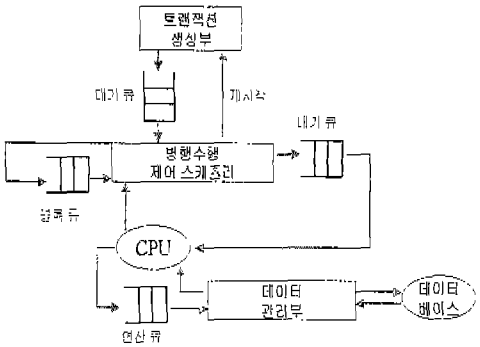


그림 8 시뮬레이션 모델

스케쥴러로 들어가게 된다. 병행수행 제어 스케쥴러는 알고리즘에 따라 하나의 연산 단위로 토크를 할당한다.

다음 [표 5]는 평가에서 사용된 각 파라미터(parameter)의 초기값들로 [10]의 수치를 참고하였다.

표 5 시뮬레이션 파라미터

| 파라미터 | 설 명 | 값 |
|---|---|---|
| 데이터베이스 크기 | 데이터베이스의 데이터 수 | 100 |
| 트랜잭션 수 | 수행되는 트랜잭션 수 | 50 |
| 갱신 트랜잭션 최대/최소 연산수 | 갱신 트랜잭션이 갖는 최대/최소 연산 수 | 20/10 |
| 판독 트랜잭션 최대/최소 연산수 | 판독전용 트랜잭션이 갖는 최대/최소 연산 수 | 40/10 |
| 디스크 접근 시간 | 데이터 접근의 I/O 시간 | 20 ms |
| CPU 프로세싱 시간 | 연산이 CPU에 머무는 시간 | 10 ms |
| 잠금 오버헤드 시간 | 연산이 잠금될 때의 오버헤드 | 10 ms |

### 5.2 평가 분석

성능평가의 대상으로는 래치 기반 스케쥴링과 2VL, 그리고 제안 알고리즘(e2VL)을 둔다. 평가기준으로는

표 4 제안한 알고리즘과 기존 알고리즘과의 비교

| | 래치기반스케쥴링 | 2VL | 제안한 알고리즘 |
|---|---|---|---|
| 메타데이타 모델 | Dublin Core | INDECS | INDECS |
| 기반 환경 예 | 서지 정보 중심의 정적 전자 도서관 | 다운로드 등의 다양한 이벤트가 발생하는 웹상의 전자 도서관 | 다운로드 등의 다양한 이벤트가 발생하는 웹상의 전자 도서관 |
| 메타데이타 분류 | 없음 | 없음 | 정적요소와 동적 요소 |
| 고려 트랜잭션 | 판독 트랜잭션 | 판독과 갱신 트랜잭션 | 판독과 정적 갱신 트랜잭션, 동적 갱신 트랜잭션 |
| 사용버전 | 1버전 | 2버전 | 2버전 |
| 수행지연원인과 해결 | 갱신 연산을 거의 고려하지 않음 | 2버전을 이용하여 갱신을 고려했지만, 빈번한 갱신 연산간의 충돌 발생 | 갱신을 고려하면서, ARU 기법을 통해 갱신 연산의 충돌 해결 |

다음의 세가지 사항을 평가하기로 한다. 첫째, 관독전용 트랜잭션 비율의 증가에 따른 갱신 트랜잭션의 수행시간을 비교한다. 둘째, 동적 갱신 트랜잭션의 비율에 따른 갱신 트랜잭션의 수행시간 변화를 비교한다. 제안한 알고리즘은 특히 동적 갱신 연산간의 충돌이 발생할 때, 연기시키는 것보다 신버전에 쓸 수 있는 조건을 가지면 신버전에 추가하여 충돌을 제거하여 수행시간을 줄이게 된다.

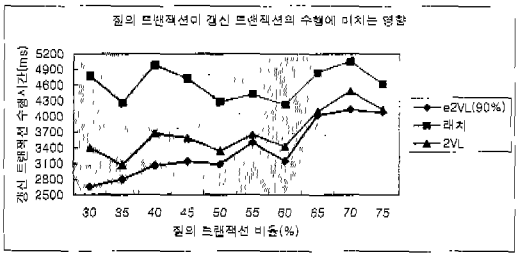


그림 9 갱신 트랜잭션 수행 시간

[그림 9]은 관독전용 트랜잭션의 비율이 증가함에 따라 갱신 트랜잭션의 수행시간이 어떻게 변화하는지를 보여 준다. 갱신 트랜잭션 수행시간 측정 식은 다음과 같다.

수행시간은 트랜잭션의 완료 시간과 트랜잭션이 대기큐에 도착한 시간의 차로 계산하며 공식은 다음과 같다.

$$\text{갱신 트랜잭션 수행시간} = \frac{\sum_{i=1}^{N} \text{종료시간}(Ti) - \text{시작시간}(Ti)}{N},$$

$N$= 수행된 전체 트랜잭션 수

관독전용 트랜잭션 비율이 적을 경우 즉, 갱신 트랜잭션 비율이 많은 경우의 기존 메타데이타 관리 기법은 갱신 트랜잭션 수행시간에 있어 2VL, 제안 알고리즘과 수행시간 간격에 많은 차이를 보인다. 갱신 트랜잭션 수행시간에서 제안 알고리즘이 2VL보다 나은 성능을 보이는 것은 빈번한 갱신 연산 간의 충돌을 ARU 기법을 통해 최소화함으로써 갱신 트랜잭션의 수행시간을 단축시켰기 때문이다. 갱신 트랜잭션의 비율이 높을수록, 그리고 그 중에 동적 갱신 트랜잭션의 비율이 높을수록 2VL과 제안 알고리즘과의 성능 차이는 커진다.

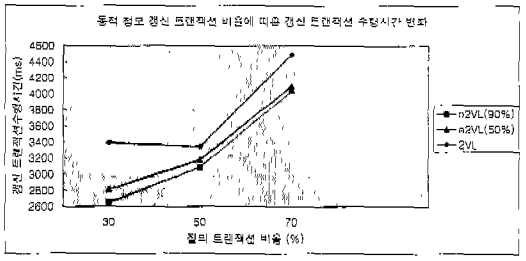


그림 10 동적 갱신 트랜잭션 비율에 따른 갱신 트랜잭션 수행시간 변화

[그림 10]은 동적 갱신 트랜잭션의 비율에 따른 갱신 트랜잭션 수행시간 변화를 나타낸다. 2VL보다는 제안 알고리즘의 수행시간이 낮고, 동적 갱신 트랜잭션의 비율이 높을수록 수행시간 역시 낮아지게 된다.

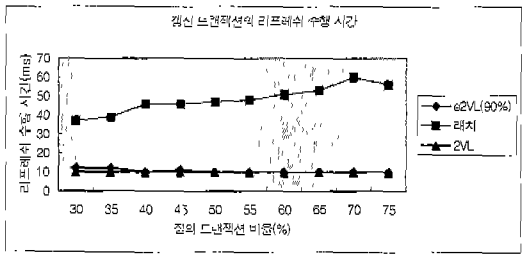


그림 11 갱신 트랜잭션의 리프레쉬 수행 시간

[그림 11]은 관독전용 트랜잭션의 비율에 따른 갱신 연산의 리프레쉬 수행시간을 나타낸다. 리프레쉬 수행시간의 측정 공식은 다음과 같다.

$$\text{리프레쉬 수행시간} = \frac{\sum_{i=1}^{N} \text{종료시간}(OPi) - \text{시작시간}(OPi)}{N},$$

$N$= 수행된 전체 리프레쉬 수,
$OPi$= $i$번째 리프레쉬 연산

제안 알고리즘과 2VL은 2버전을 사용하므로 관독전용 트랜잭션의 비율에 영향 없이 거의 일정한 수행 시간을 유지하지만 래치기반 스케줄링에서의 리프레쉬는 새로운 데이터가 삽입되고 이전 데이터가 삭제되어 사용자가 관독 가능 시점까지의 시간을 의미하기 때문에, 삭제되는 이전 버전에 대한 관독 연산이 없는 시점에서 삽입과 삭제 연산이 동시에 이루어져야 하므로 관독전용 트랜잭션의 비율에 따라 그래프가 증가하고 있다.

이상의 평가 분석에서 제안알고리즘은 갱신 트랜잭션의 비율이 증가되는 환경에서, 특히 동적 갱신 트랜잭션의 비율이 높은 환경에서 기존 메타데이타 관리 기법에 비해 트랜잭션 전체 수행시간에 있어 좋은 성능을 보였다.

## 6. 결 론

메타데이타를 관리하기 위해서는 일관성과 수행시간, 그리고 최근성을 고려해야 한다. 즉, 메타데이타의 일관성을 유지하는 범위에서 최소의 수행시간을 갖고 가장

최근의 데이터를 읽는 최근성을 확보해야 한다. 서지 정보 중심의 기존의 정적인 메타데이타에서 전자상거래의 개념을 포함하는 이벤트 중심의 표준화 메타데이타로 환경이 바뀜에 따라 동적인 정보 갱신에 대한 고려가 부족한 기존의 메타데이타 관리기법에 대한 변경이 요구되고 있다.

본 논문은 동적인 개념을 포함하는 메타데이타의 요소에 대한 성격을 정의하고 재분류하여, 관련 트랜잭션들을 판독 트랜잭션, 정적 갱신 트랜잭션, 동적 갱신 트랜잭션으로 재분류한다. 이를 통해 서지 정보 중심의 정적인 메타데이타와 관련있는 정적 갱신 트랜잭션과 전자상거래 메타데이타 요소와 관련있는 동적 갱신 트랜잭션의 락 할당 범위를 조절하고, 빈번하게 발생하게 되는 동적 갱신 트랜잭션에 대하여는 ARU라는 기법을 이용하여 갱신 연산간의 충돌로 인한 단순 지연을 방지하여 빠른 응답시간을 보장하게 하는 병행수행 제어 기법을 제시한다.

갱신이 거의 고려되지 않는 정적인 메타데이타에 대한 스케쥴링 기법인 래치 기반 스케쥴링은 동적인 메타데이타 환경에 대해서는 빈번한 갱신 때문에 응답시간과 최근성 문제에서 심각한 영향을 끼친다. 한편 동적인 메타데이타 환경에서의 2VL 기법은 갱신에 대한 고려를 하였지만, 트랜잭션 재분류가 이루어지지 않아 서로 연관없는 트랜잭션 사이의 의도하지 않는 충돌을 야기하여 역시 응답시간을 떨어뜨리게 된다. 제안 알고리즘은 갱신 연산간의 충돌을 해결하는 것에 중점을 두었기 때문에 갱신이 빈번하지 않은 환경, 예를 들면, 주식정보를 저장한 데이터베이스 같은 판독 전용 연산이 주로 수행되는 환경에는 적합하지 않다.

추후 연구로는 일관성을 해치지 않는 범위에서의 다중 그래뉼 로크를 적용하는 방법에 대한 연구가 필요하다. 한편 디지털 라이브러리의 병행수행 기법 중에서 사용자 계층에 따른 보안 사항에 대한 연구가 고려되어야 한다.

## 참 고 문 헌


[1] Rust, Godfrey & Bide, Mark, "The <indecs> metadata model", <indecs> London conference, 1999.7.5. ,URL:http://www.indecs.org

[2] Ora lassila, *"Web Metadata: A Matter of Semantics"*, IEEE Internet computing, 1998.

[3] 이해민, 박석, "디지털 도서관에서 래치에 기초한 메타데이타 관리 기법", 정보과학회 논문지 제27권1호, pp.22-32, 2000.

[4] 좌은희, 박석, "디지털 문서의 메타데이타 관리를 위한 2버전 래치 기법", 정보과학회 학술발표논문집 제27권 2호, pp.439-441, 2000.

[5] 최일환, 좌은희, 박석, 디지털 정보에 대한 식별자 부여 및 전자 상거래용 메타데이타 모델에 관한 연구, KERIS 연구보고 RR 1999-2, 1999.

[6] Hoewon Kim and Seog Park, "Two Version Concurrency control Algorithm with Query Locking for Decision Support", *Proceedings of the International Workshops on Data Warehousing & Data* Mining etc. (ER'98),pp153-164. Singapore, 1998.11.

[7] Jim Gray, Raymond A. Lorie, Gianfranco R. Putzolu, Irving L. Traiger, "Granularity of Locks in a Large Shard Database," *Very Large Data Base(VLDB)*, pp.428-451, 1975

[8] Philip A. Bernstein, V. Hadzilacos and N. Goodman. *Concurrency Control and Recovery in Database Systems*. Reading, Addison-Wesley, 1987.

[9] Alan B. Pritsker, Jean J. O'Reilly and David K. LaVal, *Simulation with Visual SLAM and AweSim* ,Systems Publishing Corporation, Indiana, 1997.

[10] Vijay. Kumar. *Performance of Concurrency Control Mechanisms in Centralized Database Systems*. Reading, MA: Prentice Hall, 1996.



최 일 환

1999년 서강대학교 컴퓨터학과 학사. 2001년 서강대학교 컴퓨터학과 공학석사. 2001년 ~ 현재 삼성전자 재직중. 관심분야는 웹과 데이터베이스, 디지털 라이브러리, 트랜잭션 관리



박 석

1978년 서울대학교 계산통계학과(이학사). 1980년 한국과학기술원 전산학과(공학석사). 1983년 한국과학기술원 전산학과(공학박사). 1983년 9월 ~ 현재 서강대학교 컴퓨터학과 교수. 1989년 ~ 1991년 University of Virginia 방문교수. 1996년 ~ 1998년 한국정보과학회 데이터베이스 연구회 운영위원장. 1997년 2월 ~ 현재 한국통신정보보호학회 이사. 1999년 1월 ~ 현재 한국정보과학회 이사 2000년 4월 ~ 현재 DASFAA Steering Committee 멤버. 관심분야는 실시간 데이터베이스, 데이터베이스 보안, 멀티미디어 데이터베이스, 트랜잭션 관리, 데이터웨어하우스, 웹과 데이터베이스