Journal of Korean Institute of Intelligent Systems
Vol. 25, No. 6, December 2015, pp. 551-557
http://dx.doi.org/10.5391/JKIIS.2015.25.6.551

ISSN(Print) 1976-9172
ISSN(Online) 2288-2324

JKIIS

불안정 모바일 네트워크 환경에서 공유 데이터 의미 동기화 기법

Semantic Synchronization of Shared Data for Unstable Mobile Environment

홍동권*†
Dong-Kweon Hong†

*계명대학교 컴퓨터공학과
*Department of Computer Engineering, Keimyung University

요 약

공유 데이터의 동기화 방법은 데이터와 응용의 특성에 따라 적절한 방법이 필요하다. 본 논문에서는 비트랜잭션 데이터(nontransactional data)에 대한 비연결 모드 동기화 기법으로 의미 동기화 기법(semantic Approach 줄여서 semanticAppr)을 제안하고, 의미 동기화 기법이 다중 사용자간의 상호 협업 과정에서 비트랜잭션 데이터의 데이터 무결성을 위한 직렬화를 완화하여 각 사용자의 작업 손실이 줄어들게 함을 보인다. 또 클라이언트에서 서버로 문서 전체를 전송하는 것이 아니라 문서에 대한 연산을 로그 생성하여 전송함으로써 데이터 전송 양을 급격히 줄인다.

키워드 : 공유 데이터, 도큐멘트 동기화, 협업, 비트랜잭션 데이터

Abstract

Synchronization methods for shared data need to be selected properly based on characteristics of data and applications. In this paper we suggest a new semantic synchronization method, semanticAppr, for non_transactional data in disconnected mode. Our approach reduces loss of works in cooperative environments by weakening constraint of serializability. In addition it reduces data transfer by sending operation log instead document itself.

Key Words : Shared Data, Document Synchronization, Cooperative Work, Nontransactional Data.

Received: Aug. 10, 2015
Revised : Sep. 3, 2015
Accepted: Sep. 14, 2015
†Corresponding author
dkhong@kmu.ac.kr

1. 서 론

비정형의 문서(Document) 또는 정형 데이터의 공유는 공유 수준과 환경에 따라 서로 다른 동기화 기법이 필요하다. 공유 환경을 살펴보면 사용자들이 네트워크에 계속 연결되어 실시간으로 그 내용을 공유하는 경우도 있으며, 반면에 네트워크 연결이 되지 않아 USB 또는 전용 커넥터를 사용하여 데이터를 동기화하는 것까지 다양한 수준의 연결 정도가 존재한다. 공유 환경을 살펴보면 사용자들이 네트워크에 계속 연결되어 실시간으로 그 내용을 공유하는 경우도 있으며, 반면에 네트워크 연결이 되지 않아 USB 또는 전용 커넥터를 사용하여 데이터를 동기화하는 것까지 다양한 수준의 연결 정도가 존재한다.

공유하는 데이터의 종류도 그림, 비정형의 텍스트 문서, 구조화된 문서, 정형의 데이터베이스 데이터 등으로 다양한 형식을 가진다. 데이터베이스의 데이터 공유는 트랜잭션과 동시성 제어 기술 등으로 관계형 데이터베이스에서 충분히 연구되어 왔으며, 데스크탑 환경에서 문서의 공유는 XML 데이터베이스의 기술과 함께 활발히 연구되고 있다[1][2]. 특히 스프레드시트와 같은 구조화된 문서는 항상 온라인 되는 경우에는 비정형 문서와는 달리 쉽게 관계형 테이블로 매칭될 수 있으므로 관계형 기술을 적용한다면 쉽게 공유할 수 있다[2].

본 논문은 2014년도 계명대학교 비사 일반 연구 기금으로 이루어졌음.

모바일 환경에서의 데이터 및 문서 공유는 기존의 데스크탑 환경과는 차이가 있다. 모바일 환경에서의 불안정한 네트워크 연결로 인하여 모바일에서는 오프라인으로 사용하다가 다시 온라인 되면 동기화해야 하므로 계속 온라인 되어 있는 데스크탑 환경에서의 동기화 결과와는 다른 결과를 보일 수 있다. 또 오프라인 상태에서의 변경 내용을 온라인 되었을 때 일괄 동기화하는 방법이나 심지어는 전용 케이블을 사용하여 동기화하는 것까지 고려하여 동기화 방법을 구성해야 하는 어려움이 있다.

동기화 연산의 충돌 해결 방법(Conflict resolution)으로 대표적인 방식은 충돌이 되었을 때 단순히 연산의 요청 순서에 의해 시간적으로 늦은 충돌 연산을 중단하는 방식을 사용하고 있다. 하지만 이 충돌 해결 방법을 모바일 환경에서의 동기화 방법으로 그대로 적용할 경우 모바일 단말기에서 오프라인으로 작업한 많은 변경 내용들이 동기화 요청 순서에 의해 무효화되는 비효율성을 보이게 된다. 이런 문제점을 해결하기 위해 본 논문에서는 네트워크 연결이 불안정한 모바일 환경에서 공유 데이터의 동기화 과정에서 새로운 비용 기반 효율성 (Cost Based Effectiveness) 개념을 정의하고, 그 의미를 기반으로 새로운 의미 중심의 효율적인 동기화 방법을 제안하고 그 성능을 분석한다.

2. 관련 연구

다큐멘트의 협업 과정에서 각 사용자의 다중 버전을 누적한 후 사용자가 나중에 적절히 통합하는 방법은 사용자에게 많은 부담을 주는 방법이다. 관계형 데이터베이스 분야에서는 데이터의 무결성을 위하여 공유 데이터의 동기화에 대하여 오랜 기간 연구가 진행 되어 왔으며 지금까지 서버쪽 온라인 데이터의 동기화에 가장 널리 사용되는 방식은 잠금 방식(Locking)이다[1]. 데이터를 사용한 클라이언트와 서버의 데이터 동기화에서 그 결과는 요청 순서에 의해 좌우되며 DBMS는 잠금 방식을 사용하여 변경 요청의 결과가 직렬화(Serialization)되게 수행한다. 이 방법은 ACID 특성의 트랜잭션(Transaction) 연산 환경에 적합한 방식이며 입력(Insert), 삭제(Delete), 변경(Update)의 3가지 연산을 사용한다. 하지만 잠금 방식은 클라이언트로 복사해온 데이터를 변경한 후 서버로 저장할 때는 비효율성을 보이게 된다. 특히 클라이언트에 복사되어온 데이터가 클라이언트에 오래 머물게 되면 다른 트랜잭션들을 계속 기다리게 만든다. 따라서 대부분의 경우 Dirty Read를 허용하게 하고 데이터의 변경 시점에 변경의 유효성을 검사하는 OCC(Optimistic Concurrency Control) 방식이 더 널리 사용되고 있다[1]. 이 경우 클라이언트에서 데이터베이스에 다른 클라이언트에서 동시에 특정 레코드에 변경 연산을 요청할 경우 충돌 데이터에 나중에 요청된 변경 연산은 거절(reject)되거나 또는 덮어쓰기(overwrite)를 사용하게 된다[1]. Optimistic 동시성 제어를 사용하는 경우 다중 클라이언트가 같은 테이블의 데이터를 읽어간 후 다시 데이터를 업데이트할 경우 격리 수준 (level of isolation)에 따라 다음의 2가지 경우를 고려한다.

1) 레코드 전체가 유효해야 하는 경우
2) 업데이트되는 컬럼만 유효해야 하는 경우

만약 유효하지 않으면 이미 다른 클라이언트에 의해 데이터가 변경되었다는 것을 의미하며, 업데이트를 시도하는 클라이언트 데이터의 의미가 없다고 판단하며 업데이트 시도는 실패하게 된다.

온라인 환경의 동시성 제어에서 온라인 사용자에게 이미 응답을 보낸 경우에는 그 결과를 되돌릴 수는 없다(durability). 하지만 모바일 사용자가 늘어나면서 온라인 연산 요청들만 있는 것이 아니라 배치 단위의 다중 요청이 혼합되어 있는 경우도 발생한다. 이 경우 전역 최적화 (global optimization)는 단순 잠금 방식보다 더 효율적일 수 있다[3]. 구글 Docs에서는 온라인 되어 있을 경우 실시간 문서 공유 기능도 가능하다. 하지만 오프라인 상태에서의 일반적인 문서 공유는 의미가 불분명하다.

XML 문서를 공유하는 환경에서 동시성 제어에 대한 연구들은 그 문서들이 항상 온라인 되어 있는 환경을 가정하고 있다. XML 문서를 계층적 트리로 표현하고 트리의 각 노드들을 동시에 억세스할 때 발생하는 동시성을 해결하기 위한 계층적 잠금 방식들이 많이 제안되었다[4][5]. 하지만 이들 잠금 방식들은 온라인으로 연결되어 있는 상태에서 가능하며 오프라인 작업 후 동기화는 고려하지 않는다.

XML 문서의 변경 내용을 검사하는 방법은 다양한 연구 결과가 발표되었다[6][7][8][9][10]. 이들 기법들은 XML 문서 내부에 어떤 변경이 생겼는지, 심지어는 엘리먼트 내부의 텍스트가 어떻게 변경 되었는지를 파악할 수 있다. 지금까지의 연구들은 충돌을 발생시킨 연산들이 우선순위가 없다는 가정으로 진행되었다. 예를 들어 컬럼의 내용이 문자열인 경우 문자의 길이가 더 긴 것이 우선순위가 더 높아지는 경우 많은 작업들이 취소되는 것을 막을 수 있다. 동시성 제어가 비용 기반 효율성(Cost Based Effectiveness)을 전혀 고려하지 않고 진행되고 있다.

3. 모바일 동기화

본 논문은 다수의 모바일 사용자에서 서로 독립적으로 작업을 수행한 후 한꺼번에 통합하는 과정에서 발생하는 데이터 충돌 시 독립적인 작업의 손해를 최소화하기 위한 동시성 문제에 의미를 부여한 해결 방법인 SemanticAppr 기법을 제안한다. 사용 문서는 XML 모델로 표현되지만 내부 구현에 영향을 받지 않으므로 XML 문서가 다시 관계형 테이블로 맵핑 표현하여 구현하는 방식을 사용할 수도 있다. 운영 환경은 항상 온라인 되어 있는 최상의 네트워크 환경은 아니므로 오프라인에서 일정 시간 작업 후 네트워크의 상황이 좋아지거나 또는 물리적으로 연결하여 다시 온라인이 되면 그때 한꺼번에 동기화하는 환경을 가정하며, 본 논문에서 사용하는 데이터베이스

의 테이블 스키마는 다음과 같은 구조를 가진다.

```
〈my_doc〉
〈A〉my part here 〈/A〉
〈B〉 part for some other person 〈/B〉
〈C〉some other text here. 〈/C〉
〈D〉1250〈/D〉
〈/my_doc〉
```

그림 1. 샘플 XML 데이터
Fig. 1. A Sample of XML data

그림 1의 샘플은 다음의 그림 2와 같은 관계형 스키마로 표현 가능하며 본 논문의 내용을 설명하는데 차이점은 없다. 하지만 그림 2의 스키마는 비정형 형태의 문서들을 저장하는데 공간적인 비효율성을 보인다. 그림 2에서 rec_id는 엘리먼트의 순서 정보를 유지하기 위해 듀이번호로 사용된다[11].

rec_id	key	value

그림 2. 비정형 문서를 위한 관계형 스키마
Fig. 2. Relational schema for a unstructured document

관계형 테이블에 입력된 데이터는 나중에 rec_id를 사용한 통합 과정을 거쳐 그림 2의 데이터에서 다시 그림 1의 XML 문서를 구성한다. 하지만 새로 입력된 데이터와 결합된 데이터와의 재결합과정에서 충돌이 발생할 수 있다. 숫자 데이터의 경우 가장 최신 데이터로 업데이트하면 되지만 문자열 데이터의 경우 의미에 따라 다음의 2가지가 가능하다.

1) 문자열의 결합이 필요한 경우
2) 문자열을 교체하는 경우

본 논문에서는 같은 엘리먼트에 대한 데이터는 그림 2의 형식에 계속 데이터를 누적한 후 나중에 통합하는 방식이 아니라 그림 1의 기본 템플릿에 Update 방식으로 (의미상 더적합) 진행한다. 따라서 단순 엎어쓰기 (OverWrite)나 또는 요청의 거절하기(Reject)가 연산 요청과 동시에 결정되는데 이때 비용 기반 효율성을 적용한다.

정의 1: 비용 (Cost)
Cost(w)는 w를 생성하기 위한 비용이다. 비용은 작업의 특성에 따라 다양하게 정의될 수 있다 (시간, 노력, 크기, ...). 엘리먼트의 내용을 구성하는 문자열이 w일 때 Cost(w)는 사용자가 문자열 w를 구성하기 위하여 사용한 비용을 의미한다.

정의 2: 비용 기반 효율성 (Cost Based Effectiveness)
Cost(w)〉 Cost(y)일 때 w를 유지하는 것이 효율성이 높다고 정의한다.

3.1 연산 로그의 구성

네트워크가 원활해지면 로컬 클라이언트에서 변경된 내용은 서버에 전달되고 동기화가 이루어진다. 이때 변경 전 내용의 전송도 필요하다. 만약 서버로의 변경 요청이 성공하면 변경된 내용이 새로운 로컬 데이터가 되며, 요청이 실패할 경우 서버의 내용을 다시 읽어온다. SemanticAppt 기법은 클라이언트에서의 문서 변경에 대한 로그를 생성하고, 그 로그를 서버로 보내는 방식을 사용한다.

표 1. SemanticAppr 연산 리스트 및 의미
Table 1. Meaning and list of SemanticAppr operations

operation	meaning
insert(Element, ordno, XMLcontent)	insert XMLcontent as *ordno*th *child of* Element
delete(Element)	delete *Element* and all its descendants
replace (old_element, new_element)	replace *old_element* with *new_element*

로컬 문서의 변경에 대한 변경 로그의 생성 과정은 다음과 같다.

1) 로컬 문서에 작업을 적용
2) 작업이 종료되면 DeltaXML[10] 등의 XML 문서 비교 프로그램을 이용하여 로컬 원본과 변경된 내용을 비교하여 표 1에 해당하는 로그를 생성한다.
3) 로그를 서버로 전송할 때는 XML 또는 JSON 형식으로 데이터를 전송하며 본 연구에서는 JSON 형식으로 표현하였다.

3.2 로그의 실행

클라이언트는 기본적으로 서버에 있는 파일의 리스트를 확인하는 기능, 서버의 파일을 클라이언트로 읽어오는 기능이 있다. 클라이언트로 데이터를 읽어온 후 표 1의 의미를 가진 연산을 수행한다. 여러 개의 업데이트 요청에 대해서 단순 순차적 업데이트는 데이터의 의미를 잃어버리게 할 수도 있다. 즉 이미 변경된 데이터의 내용이 무엇인지 알지 못하는 상태에서 새로운 데이터를 쓰기 하는 것은 다른 사용자의 노력을 없애버리는 결과를 가져오게 된다. 즉 전체 비용이 많아지는 결과를 초래한다. 따라서 로그의 실행은 다음의 동시성 제어를 이용하여 전체 비용이 줄어들게 실행된다.

3.3 동시성 제어

새로운 데이터 (엘리먼트)의 입력은 항상 새로운 오브젝트 ID를 부여하므로 다른 사용자들의 연산에 영향을 주지 않는다. 각 연산에서 충돌이 발생하는 경우를 살펴보자.

Case 1: 입력〈-〉 다른 연산 (입력, 삭제, 변경)
입력 연산은 다른 연산에 영향을 주지 않는다.

Case 2: 삭제 〈-〉 삭제 연산의 충돌
이미 삭제된 것을 다시 삭제 요청하는 것은 아무런 영향도 주지 않는다.

Case 3: 삭제 -〉 변경
삭제와 변경 연산의 충돌에서 이미 삭제된 데이터의 변경 요청이 무시되면 변경 노력이 허비되는 비효율성을 보이게 된다. 따라서 Cost를 낭비하지 않기 위하여 삭제된 데이터를 복구하고, 변경을 수행한 후 그 다음 처리를 수행한다.

Case 4: 변경 -〉삭제
반대로 삭제 요청한 레코드가 이미 변경이 되어 있으면 Cost를 낭비하지 않기 위하여 삭제하지 못하게 한다 .

Case 5: 변경 -〉변경
여러 의미를 부여한 충돌 해결 방법이 필요하다. 원본 데이터와 새로운 데이터의 차이점을 알아낸 후 결합(merge) 방법은 다음의 경우에 따라 적용한다. (Cost의 비교가 필요)
(1) 단순 추가, (2) 부분 수정, 3) 부분 삭제

변경-〉변경의 경우 그 의미를 서버 쪽에서 판단하기 위해서는 클라이언트에서 서버로 전달해야 하는 데이터의 양이 많아지므로 클라이언트에서 판단하는 것이 유리하다. 따라서 클라이언트에서는 입력된 내용과 복사해온 내용의 차이를 이용하여 전송할 로그를 결정하고, 서버에서는 Optimistic 방법을 적용하면서 그 동안 데이터가 바뀌었는지 확인한다. 만약 변경이 없었다면 연산이 문제없이 수행되지만 만약 바뀌었으면 연산을 거절하는 것이 아니라 어떤 변경이(변경의 의미) 이루어졌는지를 다시 판단하고 그에 적합한 액션을 취한다. 특히 엘리먼트 단위의 비교가 이루어지므로 같은 엘리먼트의 변경을 요청했을 때 나중 변경 요청을 거절되는 것이 아니라 Cost 비교에 의해 다음과 같이 수행된다.

1) 엘리먼트 내에서의 위치가 서로 다르면 변경 수행 가능
2) 만약 변경 위치가 중첩이 되면 변경 된 문서의 양을 비교
   2-1) old 변경 양이 많으면 새로운 변경을 무시
   2-2) new 변경 양이 많으면 old를 new로 변경 과정을 요약하면 다음의 표 2와 같다.

표 2. 의미를 부여한 연산 처리
Table 2. Semantic execution of operation

change in old	true meaning of element update		
	delete	insert	update
delete	if different id then delete, otherwise already deleted	insert new because id is different	recover old and then execute update using Cost()
insert	element id is different	element id is different	element id is different
update	reject delete op according to Cost()	element id is different	when element id is the same, compare Cost()

4. 분석 및 비교

본 논문에서 제안한 의미 방식이 (SemanticAppr) 얼마나 작업의 효율성을 높일 수 있는지를 분석하기 위하여 다음의 2가지 방식과 비교한다.

1) 문서 전체를 클라이언트에서 서버로 다시 전송하며 전통적인 Reject, Overwrite를 사용하는 기법 (BasicAppr)

2) 문서 전체를 클라이언트에서 서버로 다시 전송, 서버의 현재 문서 cur_doc는 클라이언트에서 복사해간 old_doc와는 다를 수 있으므로 클라이언트에서 수정하여 서버로 보낸 new_doc와 cur_doc를 문서 비교하여 어떻게 합칠 것인가를 결정하는 방법 – 이 방법은 엘리먼트 단위가 아니라 문서 전체를 비교하므로 다음의 표 3과 같은 어려움이 있다(TotalAppr).

표 3. TotalAppr 방식에서 문서 비교
Table 3. Doc comparison in TotalAppr approach

changes of cur_doc	content of new_doc		
	delete	insert	update
text delete	Delete do not appear	Only inserted one appear	Only updated one apper
text insert	Delete do not appear	Concatenation of different texts	Concatenation of different texts
text update	Delete do not appear	Concatenation of different texts	Concatenation of different texts

4.1 모의실험 시나리오

1) 각각의 모바일 어플리케이션에서 서버의 데이터를 복사해서 가져감

2) 독립적으로 연산을 수행 (문서에 작업)

3) 서버로 동기화를 요청하면 BasicAppr과 TotalAppr 방식은 변경된 문서를 서버로 전송, 반면에 SemanticAppr은 문서에 어떤 연산을 수행했는지 (Change detection program을 적용) 분석한 후 작업 (연산 리스트) 로그를 구성

4) SematicAppr 방식은 위 형식의 작업 로그를 서버로 전송하며 서버에서는
   4-1) 원본 데이터의 변경이 전혀 없었으면 롤 포워드 (Roll forward) 수행
   4-2) 원본 데이터의 변경이 발생 했으면 표 1에 따라 의미에 맞게 수행

4.2 실험 결과

다음 그림 3의 원본 데이터를 사용하여 SemanticAppr 방식이 어떤 장점을 보이는지 알아본다.

```
〈my_doc〉
〈A〉my part here 〈/A〉
〈B〉 part for some other person 〈/B〉
〈C〉some other text here. 이 부분은 상당히 긴 텍스트가 들어갈 수 있다. 〈/C〉
〈D〉1250〈/D〉
〈/my_doc〉
```

그림 3. 모의 실험 원본 데이터
Fig. 3. Original data for experimentation

먼저 3개의 클라이언트가 원본 데이터를 복사한 후 각각 그림 4의 변경을 하고 다시 동기화를 시도한다.

그림 4. 각 사용자의 변경 내용
Fig. 4. Operation of users

4.2.1 A -〉 B -〉 C 순서의 동기화

그림 5는 semanticAppr이 가장 장점을 보이는 경우이다. 각 방식의 실행 결과는 다음과 같다.

(1) basicAppr: A에 의해서 먼저 삭제가 이루어지고, 그 다음 B와 C가 동기화를 시도할 때 복사해온 원본과 서버의 최신 내용이 다르므로 B, C의 동기화는 실패로 끝난다. 동기화를 위해서 각 사용자의 원본과 변경 문서 전체가 같이 전송되는 문제점과 사용자의 많은 노력들이 낭비되는 결과를 보인다.

(2) totalAppr: A에 의해서 먼저 삭제가 이루어지고, 그 다음 B가 동기화를 시도하면 서버의 최신 내용과 B의 차이점인 엘리먼트 〈C〉를 찾아내고, B가 변경한 엘리먼트 〈C〉가 추가된 내용이 최신 내용이 된다. 그 다음 사용자 C가 동기화를 시도하면 엘리먼트 〈C〉가 다르다는 것을 판별하고 사용자B와 C의 엘리먼트〈C〉가 혼합된 내용이 엘리먼트 〈C〉의 최종 값이 된다. 동기화를 위해서 각 사용자의 원본과 변경 문서 전체가 같이 전송되는 문제점을 보인다.

(3) semanticAppr: delete(/my_doc/C), replace(/my_doc/C, 'some other text here. 이 부분의 변경이 발생'), replace(/my_doc/C, 'some other text here. 이 부분에는 상당히 긴 텍스트가 들어갈 수 있다. 또 다른 내용의 추가가 이루어졌다.')의 로그가 각 사용자에 의해서 전송된다. 각 연산은 표 1에 의해서 수행된다. 가장 먼저 delete가 수행되고, 그 다음 replace는 삭제된 내용의 복원 후 변경을 수행한다. 마지막 replace는 이전 replace와 충돌되며, 변경 양이 더 많으므로 앞의 replace를 무시하고 마지막 replace를 적용시킨다. 그리고 문서 전체를 보내는 방식에 비해서 로그만 보내기 때문에 문서의 크기가 클 경우 이 방법은 큰 장점을 보인다.

그림 5. A-〉B-〉C 순서의 동기화 실행 결과
Fig. 5. Results of A-〉B-〉C order

4.2.2 C -〉B -〉A 순서의 동기화

(1) basicAppr: C에 의해서 먼저 변경이 이루어지고, 그 다음 B와 A가 동기화를 시도할 때 복사해온 원본과 서버의 최신 내용이 다르므로 B, A의 동기화는 실패로 끝난다.

(2) totalAppr: C에 의해서 먼저 변경이 이루어지고, 그 다음 B와 A가 동기화를 시도할 때 복사해온 원본과 서버의 최신 내용이 다르므로 B, A의 직접 동기화는 실패로 끝난다.

하지만 문서 내용의 차이를 이용하여 사용자 B와 사용자C의 변경 내용이 혼합되어 추가된다.

(3) semanticAppr: delete(/my_doc/C), replace(/my_doc/C, 'some other text here. 이 부분의 변경이 발생'), replace(/my_doc/C, 'some other text here. 이 부분에는 상당히 긴 텍스트가 들어갈 수 있다. 또 다른 내용의 추가가 이루어졌다.')의 로그가 각 사용자에 의해서 전송된다. 각 연산은 표 1에 의해서 수행되는데 가장 먼저 사용자 C의 replace 수행되고, 그 다음 사용자 B의 replace는 변경 양이 적어서 거절된다. 마지막 delete는 replace와 충돌되며 변경 내용을 무효화 시키는 연산이어서 거절된다.

그림 6. C->B->A 순서의 동기화
Fig. 6. Synchronization of C->B->A order

그림 6은 C->B->A 순서로 동기화가 진행되었을 때 결과를 보이고 있다.

5. 결론 및 향후 연구 방향

공유 데이터의 동기화 방법은 데이터와 응용의 특성에 맞추어 적절한 방법이 필요하다. 간단한 숫자, 문자 등으로 구성된 트랜잭션 데이터의 동기화 방법으로는 서버쪽 기법으로는 잠금 기법이 가장 널리 사용되며, 비연결 모드(Disconnected mode)에서 클라이언트 작업 후 서버와의 동기화 방법으로는 OCC가 가장 널리 사용되고 있다. OCC 기법은 유효성 검사 과정이 필수적이며 유효 기준으로는 이전 버전과 새로운 버전의 일치성이 트랜잭션 데이터에 사용되고 있다.

본 논문은 비트랜잭션 데이터의 비연결 모드 동기화 기법으로 의미 동기화 기법 (semanticAppr)을 제안하고 의미 동기화 기법이 어떤 협업 상황에서 기존의 방법보다 더 좋은 성능을 나타내는지를 보였다. semanticAppr 기법은 비트랜잭션 데이터의 데이터 무결성을 위한 직렬화를 완화하고 다중 사용자간의 상호 협업 과정에서 비용 (Cost 함수 사용) 개념을 도입하여 각 사용자의 작업 손실이 단순 최소화가 아니라 의미 있게 손실이 줄어들게 한다. 또 클라이언트에서 서버로 문서 전체를 전송하는 것이 아니라 문서에 대한 연산을 로그 생성하여 로그를 전송함으로써 데이터 전송이 문서의 양이 아니라 로그에 비례하게 된다.

로그의 생성은 표 1의 내용에 따라 생성되며 각 연산의 충돌 해결은 표 2에 따라 해결하게 된다. semanticAppr은 숫자와 간단한 문자열이 컬럼의 형식으로 사용되는 OLTP에는 적합하지 않지만 다큐멘트 데이터의 협업 과정에서 발생하는 동기화 처리에 적합한 방법으로 평가된다. 본 논문에서 제안한 semanticAppr는 XML로 표현된 그림의 공동 작업에 쉽게 적용될 수 있다. 예를 들어 XAML로 표현된 마이크로소프트의 WPF에 적용하면 XAML로 서로 독립적인 작업 후 그 결과를 통합하여 WPF를 구성할 수 있다.

References

[1] Hector Garcia-Molina, Jeffery D. Ullman, Jennifer Widom, *Database Systems, The complete book*, Prentice Hall, 2002.

[2] A. Chaudhri, Roberto Zicari, Awais Rashid, *XML Data Management: Native XML and XML Enabled Database Systems*, Addison-Wesley, 2003.

[3] Timos K.Sellis, "Globol Query optimization", *ACM SIGMOD Record*, vol. 15, Issue2, pp. 191-205, 1986.

[4] M.Haustein, T.Harder, "Optimizing lock protocols for native XML processing," *Data and Knowledge Engineering*, vol.65(1), pp.147-173, 2008.

[5] Eunjung Lee, "Multi-Granularity User Friendly List Locking Protocol for XML Repetitive Data", *International Journal Data Engineering (IJDE)*, Volume (2), Issue (3), 2011.

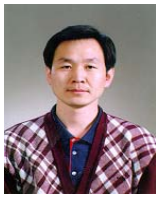
[6] Luuk Peters, "Change Detection in XML Trees: a Survey", 3rd twente *Student Conference on IT*, Enschede June, 2005.

[7] Tancred Lindholm, "A three-way merge for XML documents", *Proceedings of the 2004 ACM symposium on Document engineering*, pp.*1-10*, Oct. 2004.

[8] Sudarshan S. Chawathe, Anand Rajaraman, Hector Garcia-Molina, Jennifer Widom, "Change Detection in Hierarchically Structured Information", *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data,* pp. 493-504. June, 1996.

[9] Sudarshan S. Chawathe and Hector Garcia-Molina, "Meaningful change detection in structured data", *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pp.26-37. May. 1997.
[10] Robin La Fontaine, "DeltaXML, Change Control for XML: Do It Right", *XML Europe 2003*, May 2003.
[11] P. O'Neil, E. O'Neil, S. Pal, I. Cseri, G. Schaller, "ORDPATHs: Insert-friendly XML node labels", *Proceeding of ACM SIGMOD,* pp. 903-908 June Paris, France 2004.

저 자 소 개



홍동권(Dong-Kweon Hong)
1985년 : 경북대학교 전자과 공학사.
1992년 : U. of Florida 컴퓨터공학 석사
1995년 : U. of Florida 컴퓨터공학 박사
1997년~현재 : 계명대학교 컴퓨터공학과 교수

관심분야 : 데이터베이스, 데이터 마이닝, 머신 러닝
Phone : +82-53-580-5281
Fax : +82-53-580-5165
E-mail : dkhong@kmu.ac.kr