제 18 회 한국정보처리학회 추계학술발표대회 논문집 제 9 권 제 2 호 (2002.11)

MoIM-SyncML 에서의 데이터베이스 변경 추적 기법

이강우, 박남식, 함호상
한국전자통신연구원 이동분산처리연구팀
e-mail : {kwlee,nspark,hsham}@etri.re.kr

Tracking on Database Updates in MoIM-SyncML

Kang-Woo Lee, Nam-Sik Park, Ho-Sang Ham
Mobile Distributed Computing Research Team,
Electronics and Telecommunications Research Institute

요 약

SyncML 은 데이터 동기화에 관한 개방형 업체 표준으로, 통신 프로토콜, 구동 장치 및 사용 응용과 무관한 동기 프로토콜을 제공하고, 동기 될 데이터의 형식에도 무관하게 동작하도록 정의되었다. 최근 들어 이러한 특징은 기업 데이터의 동기화 서비스를 제공하는 도구로 관심 받기 시작하였다. 대부분의 기업 응용은 데이터베이스를 기반으로 수행되기 때문에 데이터베이스 정보의 동기 기능이 중요한 요소이다. 이런 이유로 최근에 개발되는 대부분의 SyncML 엔진은 데이터베이스 동기 기능을 갖추고 있다.

SyncML 을 이용한 데이터베이스 동기 기능을 제공하기 위해서는 데이터베이스 내에 발생되는 갱신 정보를 SyncML 동기 장치에 전달할 필요가 있다. 이를 위해, 데이터베이스를 갱신하는 응용을 수정할 필요성이 발생하였다. 그러나 데이터베이스의 경우 다수의 응용이 동일한 데이터베이스를 접근하여 갱신하는 경우가 많아, 이런 모든 응용을 수정하는 방법은 비현실적이다. 현재 개발된 대부분의 제품은 이러한 문제점을 갖고 있다.

MoIM-SyncML 은 SyncML 표준 규격을 기반으로 개발된 동기 엔진으로, 트리거와 뷰를 이용하여 기존 데이터베이스 응용의 수정 없이 데이터베이스 갱신 정보를 수집하는 것이 가능하도록 설계되었다. 본 논문에서는 트리거와 뷰를 이용하여 데이터베이스에 발생하는 갱신 정보를 수집하는 기법에 대해 설명한다.

1. 연구 동기 및 필요성

PDA, 휴대용 PC, 호출기, 그리고 휴대용 전화기와 같은 (무선) 휴대 기기가 점차 보편화됨에 따라, 보다 많은 사람들이 여러 휴대 기기를 보유하여 유용한 정보를 보관, 관리하게 되었다. 그러나 휴대하는 기기의 증가로 동일한 개인 정보가 여러 휴대 기기에 중첩되어 기록되는 경우가 많았으며, 이 정보들의 갱신이 각각 독립적으로 수행되는 경우가 많아. 정보 내용이 불일치되는 경우 점차 증가하였다. 그러므로 사용자들은 독립된 여러 기기에 기록된 정보들의 내용을 주기적으로 서로 일치시킬 필요성을 느끼게 되었다. 이런 필요성을 만족시키기 위해 등장한 기술이 데이터 동기화 (*data synchronization*) 기술이다. 데이터 동기화 기술은 서로 독립적으로 갱신되어 상이하나 개념적으로 동일한 데이터 값을 다시 일치하도록 하는 기술이다.

SyncML[6,7]은 데이터 동기화를 위한 개방형 업체 표준으로 장치의 하드웨어, 통신 프로토콜, 수행 응용, 그리고 데이터의 종류에 무관한 데이터 동기화 프로토콜을 지원한다. SyncML 은 XML[1]을 기반으로 하여 동기 대상의 데이터 형식이 서로 다르거나 소프트웨어 시스템이 서로 상이하여도 올바르게 작동하게 된다.

최근까지 출시된 대부분의 SyncML 제품들은 휴대 장치들 사이의 개인 정보의 데이터 동기화를 주 응용으로 개발되었다. 그러나 휴대 장치의 기능이 점차 발전됨에 따라, 점차 이를 기업 업무 처리에 사용하는

추세로 발전되고 있다. 이로 인해 휴대 장치 내의 기업 정보와 기업 서버의 정보 사이의 데이터 동기화의 필요성이 대두되었다. 특히 대부분의 기업 응용은 데이터베이스를 통해 데이터를 관리하기 때문에, 휴대 장치에 저장된 데이터와 서버 데이터베이스에 저장된 데이터 사이의 동기화 기능을 제공하는 SyncML 시스템의 개발에 관심을 기울이기 시작하였다. 최근 들어 출시되는 SyncML 제품들의 경우 데이터베이스 동기화 기능을 지원하는 제품들이 점차 증가하고 있다.

데이터베이스 테이블에 데이터 동기화 기능을 추가하기 위해서는, 해당 테이블에 발생되는 모든 갱신 작업의 정보가 SyncML 엔진에 전달되어야 한다. 그러나 데이터베이스의 경우 다수의 응용이 동일한 테이블에 접속하여 갱신하는 경우가 많아, 이런 모든 응용을 수정하는 것은 비현실적이다. 현재 개발된 대부분의 제품은 이런 문제점을 갖고 있다.

MoIM-SyncML 은 SyncML 표준 규격을 기반으로 개발된 동기 엔진으로, 다양한 저장소에 대한 동기 기능을 제공한다. 특히 많은 기업 응용의 동기 서비스 제공하기 위해 데이터베이스 동기 기능도 포함되어 있다. MoIM-SyncML 은 트리거와 뷰를 이용하여 기존 응용의 수정 없이 데이터베이스 갱신 정보의 수집이 가능하도록 설계되었다. 본 논문에서는 트리거와 뷰를 이용하여 데이터베이스에 발생하는 갱신 정보를 수집하는 기법에 대해 설명한다.

본 논문의 구성은 다음과 같다. 먼저 2 절에서는 MoIM-SyncML 시스템에 관한 간략한 개요를 설명하며 특히 데이터베이스 동기 기능을 지원하는 경우 갱신 추적의 어려움을 설명한다. 3 절에서는 트리거와 뷰를 이용한 데이터베이스 갱신 추적 방식을 설명하고, 이 방식을 사용하는 동기 과정을 보여준다. 4 절에서는 본 논문의 연구 내용과 관련된 타 기관의 연구 내용을 나열하고, 마지막으로 5 절에서 논문의 결과를 정리한다.

2. MoIM-SyncML 개요

MoIM-SyncML 은 Java 언어를 사용하여 SyncML 표준 규격을 구현한 동기 엔진으로 기업 수준의 데이터 동기 플레임워크(framework)를 제공하는 MoIM-Sync 시스템의 하부 엔진으로 사용되고 있다. 본 엔진은 처음부터 기업 응용들을 위한 미들웨어를 목표로 설계되어 기존에 개발된 많은 기업 응용들과의 연동성에 많은 초점을 두어 설계 및 개발되었다.

2.1. 어댑터 기반 동기 엔진

동기 엔진에 있어서 다양한 기업 응용들과의 연동성의 중심은 기존 기업 응용이 사용하던 여러 종류의 저장소에 대한 동기 지원이다. 즉, 기업 응용이 사용하는 다양한 종류의 저장소에 저장된 데이터를 다른 장치의 저장소에 저장된 데이터와의 동기를 쉽게 개발할 수 있는 도구를 제공하는 것이었다.

MoIM-SyncML 은 이런 기능을 제공하기 위해 어댑터(adapter)[3] 개념을 도입하였다. MoIM-SyncML 에서는 저장소를 접근하는 통일된 인터페이스를 정의하고, 엔진은 이 인터페이스를 기반으로 개발되었다. 특정 저장소를 MoIM-SyncML 과 연동시키기 위해서는 해당 저장소가 정의된 인터페이스를 지원하는 추가 모듈을 작성할 필요가 있는데, 이것이 어댑터이다.

어댑터 기능을 통해 응용 개발자는 연동시키고자 하는 정의된 인터페이스를 지원하는 모듈을 작성하는 것만으로 MoIM-SyncML 과 연동시킬 수 있다.

2.2. MoIM-SyncML 에서의 갱신 추적

SyncML 프로토콜 문서에 의하면, 프로토콜을 사용하기 위해서는 동기 가담하는 클라이언트와 서버가 모두 각자가 사용하는 저장소에서 발생된 변경 정보를 추적하는 것을 가정하였다. 만일 이미 개발된 응용이 사용하는 저장소에 동기 기능을 제공하는 경우는 저장소의 데이터가 변경되면 해당 정보를 SyncML 시스템에 직간접적으로 전달할 필요가 있다. MoIM-SyncML 에서는 변경된 데이터의 식별자, 변경 종류, 그리고 변경을 요청한 주체의 식별자를 필요로 한다. MoIM-SyncML 에서는 전달된 정보를 바탕으로 다음번 동기 과정에 사용한다.

SyncML 과 연동된 저장소의 경우, 갱신은 두 가지로 나눌 수 있다. 첫째는 지역 응용에 의한 갱신이고, 둘째는 동기 과정 중 원격 저장소에서 발생된 변경을 지역 저장소에 반영하여 생기는 갱신이다. 이중 동기 과정 중 발생되는 변경의 경우는 SyncML 엔진에 의해 수행되기 때문에, 해당 변경 정보를 손쉽게 추적할 수 있는 반면, 지역 응용에 의해 발생되는 변경은 SyncML 엔진과 독립적으로 발생되는 것이므로, 이 경우는 해당 변경 정보를 SyncML 엔진에 전달할 필요가 있다. 그러므로 다양한 저장소의 데이터를 SyncML 프로토콜을 이용하여 동기하려는 경우, 해당 저장소의 데이터를 갱신하는 지역 응용과의 연동이 중요하다.

2.3. 데이터베이스 연동시 변경 추적의 어려움

데이터베이스를 사용하는 응용의 경우, MoIM-SyncML 엔진과 연동하기 위해서는 해당 응용이 사용하는 데이터베이스에 발생되는 모든 갱신 정보를 MoIM-SyncML 에 전달하여야 한다. 이를 위해 MoIM-SyncML 과 연동되기 위해서 대부분의 경우 해당 응용을 수정하여 변경 발생시 해당 정보를 MoIM-SyncML 엔진에 전달하는 부분을 추가할 필요가 있게 된다. 이것은 데이터베이스를 사용하는 응용을 MoIM-SyncML 과 연동시키는 경우에도 동일하다. 데이터베이스 응용의 경우도 데이터베이스 갱신 시 해당 정보를 MoIM-SyncML 에게 전달하도록 수정하여야 한다. 그러나 데이터베이스 응용의 경우는 여러 응용이 하나의 데이터베이스 테이블 공유하는 경우가 많기 때문에, 단순히 해당 응용만을 수정하는 것만으로 충분하지 않고, 동일 테이블을 접근하여 갱신할 수 있는 모든 응용을 수정해야 하는 경우가 발생한다. 더구나 그런 응용 중에 일부는 응용을 수정할 수 없는 경우가 빈번히 발생한다. 그러므로 데이터베이스 동기 기능을 추가하기

위해서는 응용의 수정 없이 데이터 항목 갱신 정보를 동기 엔진에게 전달할 수 있는 방법이 강구되어야 한다.

3. MoIM-SyncML 에서의 해결 방법

본 절에서는 데이터베이스를 사용하는 응용을 위해 MoIM-SyncML 에서 사용하는 방법을 설명한다. MoIM-SyncML 은 앞 절에서 언급한 문제를 현재 대부분의 DBMS 에서 제공하는 트리거 (trigger)와 뷰 (view) 기능을 이용하여 해결한다.

MoIM-SyncML 은 트리거를 이용하여 데이터베이스의 갱신을 감지하고, 해당 변경 연산 (즉, 삽입/삭제/갱신)을 알 수 있다. 또한 뷰 기능을 이용하여 데이터베이스 갱신을 유발한 주체를 알 수 있다.

본 절에서는 또한 이 두 가지 기능을 사용하여 연락처 정보 테이블에 대한 동기 과정을 예를 들어 보여준다.

3.1. 트리거 기반 변경 추적

트리거는 SQL 또는 Java 그리고 C 언어와 같은 절차 언어로 작성된 프로시저로, 데이터베이스 테이블이나 뷰가 갱신될 때, 자동적으로 수행된다. 이를 이용하여 특정 테이블 대상으로 트리거를 작성하면 해당 테이블에 레코드의 변경을 감지할 수 있다.

MoIM-SyncML 은 동기 대상 테이블마다 세가지 변경 연산 (삽입, 삭제, 변경) 마다 하나의 트리거를 생성한다. 테이블 갱신으로 호출되는 트리거는 해당 변경 정보를 아래의 스키마를 갖는 미리 지정된 테이블 'change_log'에 삽입한다.

```
CREATE TABLE change_log (
    seqno NUMERIC not null,
    peer_id VARCHAR(128) not null,
    data_id VARCHAR(128) not null,
    utype CHAR(1) not null, -- 'A', 'D', 'R' 중 하나
)
```

'change_log' 테이블은 'seqno', 'peer_id', 'data_id', 그리고 'utype' 이라는 컬럼으로 구성되어 있고, 각각의 용도는 다음과 같다.

- seqno: 'change_log' 테이블에 삽입되는 모든 레코드마다 부여되는 번호로, 각 레코드는 모두 유일 값을 갖게 되고, 테이블에 삽입되는 순서에 따라 순차 증가하는 값을 갖는다. MoIM-SyncML 엔진은 이 컬럼을 통해 테이블을 레코드 삽입 순서대로 검색할 수 있게 된다.
- peer_id: 레코드 갱신을 유발한 주체의 식별자를 기록한다. 지역 데이터베이스 응용에 의해 갱신된 경우는 미리 지정된 식별자 값 (구현에서는 길이 0 문자열)가 기록되고, 동기 과정 중 원격 갱신을 반영하기 위해 갱신된 경우는 해당 동기 상대의 식별자가 기록된다.
- data_id: 갱신된 레코드의 식별자가 기록된다.
- utype: 갱신 종류가 기록된다. 삽입인 경우에는 'A'가 삭제인 경우는 'D'가, 그리고 변경인 경우에는 'R'이 기록된다.

그림 1은 'contacts' 라는 테이블에 삽입 감지용 트리거를 ORACLE 에서 구현한 예를 보여준다. 본 예에서는 'seqno' 컬럼 값을 위해 ORACLE 에서 제공하는 'SEQUENCE'를 이용하였다. 예를 보면 'contacts' 테이블에 레코드 삽입이 발생하는 경우, data_id 값으로 삽입된 레코드의 'id' 컬럼 값이 기록되는 것을 알 수 있고, peer_id 값으로 길이 0 의 문자열이 기록되어 지역 응용에 의한 갱신이라는 것을 기록하게 된다.

```
CREATE SEQUENCE seqno_gen
  INCREMENT BY 1
  START WITH 1
  NOCACHE
  NOCYCLE;

CREATE TRIGGER insert_on_contacts
AFTER INSERT ON contacts
FOR EACH ROW
BEGIN
  INSERT INTO change_log
    VALUES (seqno_gen.NEXTVAL, '', :new.id, 'A');
END;
```

A: ADD
D: DELETE
R: REPLACED

그림 1 'contacts' 테이블에 대한 삽입 트리거의 예

트리거를 이용하면 데이터베이스 테이블의 레코드 갱신이 발생하는 경우, 해당 갱신 연산, 대상 테이블 등에 관한 정보를 얻을 수 있다. 그러나 트리거 만으로는 갱신 요청한 주체를 알 수 없다는 한계를 갖는다. 그림 1의 예에서는 지역 응용에 의한 갱신만을 추적할 수 있고, 동기 과정 중 발생되는 갱신의 경우 갱신을 유발한 동기 상대의 식별자를 알 수 없다는 문제를 계속 갖게 된다. 다음 절에서는 뷰를 통하여 이 문제를 해결하는 방안을 설명한다.

3.2. 뷰기반 변경 요청자 확인

앞서 언급한 바와 같이 지역 응용에 의한 갱신 정보는 대상 테이블의 트리거를 작성하여, 트리거 내에서 갱신 정보를 미리 지정된 change_log 테이블에 저장함으로써 해결할 수 있었다. 그러나 동기 과정 중 발생되는 갱신의 경우에는 갱신을 요청한 동기 상대의 식별자를 기록할 필요가 있기 때문에 트리거만으로는 해결할 수 없다.

MoIM-SyncML 에서는 이런 문제를 해결하기 위해, 동기 과정 중에 동적으로 동기 대상 테이블에 대한 뷰를 생성하고, 이 뷰에 대한 트리거를 작성하는 방식을 사용한다. 즉, 특정 동기 상대 'A'와의 동기 과정을 시작할 때, 과정 시작 전에 해당 동기 과정 중 갱신할 테이블에 대해 동일한 테이블 스키마와 같은 레코드 집합을 갖는 뷰를 생성한다. 그리고 생성된 뷰에 대해 앞서 3..1 절에서 언급한 바와 같이 세가지 트리거를 작성한다. 이때 작성된 트리거 코드 내에서 'change_log' 테이블에 갱신 정보 레코드를 저장하는 부분 중 'peer_id' 컬럼 부분은 동기 상대의 식별자를 넣도록 한다. 다음의 SQL 은 'kwlee'라는 동기 상대를

위해 'contacts_kwlee' 라는 뷰와 삽입 감지 트리거의 예를 보여준다.

```
CREATE VIEW contacts_kwlee
AS SELECT * FROM contacts;

CREATE TRIGGER contacts_kwlee_insert
AFTER INSERT ON contacts_kwlee
FOR EACH ROW
BEGIN
INSERT INTO change_log
VALUES (seqno_gen.NEXTVAL, 'kwlee', :new.id, 'A');
END;
```

이런 방법으로 뷰와 그와 연관된 트리거 생성 후, 동기 과정을 수행하되, 갱신 대상 테이블을 원본 테이블을 사용하지 않고, 생성된 뷰를 사용하여 새로 생성된 트리거가 불리도록 한다. 뷰를 사용하였기 때문에 비록 갱신이 뷰에 반영되어도 실제 원본 테이블에 갱신 반영되기 때문에 정확성에 문제를 발생되지 않고, 뷰 정의 시 두개 이상의 테이블에 대한 조인으로 구성된 뷰가 아니고, 특별한 제한 조건이 없는 한 개의 테이블에 대한 뷰이기에 뷰 갱신으로 야기되는 문제가 발생되지 않는다.

동기 과정이 끝나면, 동적으로 생성된 뷰와 그와 연관된 트리거는 모두 제거되어 동기 과정 중에만 유지되기 때문에 데이터베이스에 유발되는 부하는 제한적이라 할 수 있다. 또한 자주 동기 과정을 수행하는 상대의 경우는 해당 뷰와 트리거를 계속 유지하여 다음 번 동일 상대와의 동기 과정 중에 재사용하면 뷰와 트리거의 생성/제거에 따른 부하를 줄일 수 있다.

4. 관련 연구

SyncML 표준이 발표된 이래 현재 이 규격을 따른 많은 SyncML 시스템이 구현되었다. SyncML 단체의 웹 사이트[6]에 의하면 현재 본 논문이 쓰여지던 시점에서 81 개의 SyncML 적합 제품들이 있다. 그 제품들의 많은 수는 일반 개인 정보의 동기 기능 만을 제공하고, 일부만이 데이터베이스 동기화 기능을 제공하고 있다. 더구나 데이터베이스 동기화 기능을 제공하는 제품 역시 대부분 데이터 갱신 정보를 응용 수준에서 동기 엔진에게 알리는 방식을 사용하는 단점을 갖는다. 앞서 언급한 바와 같이 이 방식은 해당 응용 외의 다른 응용이 해당 데이터베이스 테이블을 갱신하는 경우, 올바른 동기 기능을 제공할 수 없는 문제점을 갖는다.

데이터 동기화는 SyncML 에 의해 처음 언급된 것은 아니다. 이전부터 연구되어 오던 약-일관성 복사본 관리 시스템 (weakly-consistent replica management system)[2,4,5] 에서도 데이터 동기화 기능에 대해 언급해 왔다. 실제 이들 시스템에서 다루는 것은 SyncML 에서 언급한 그것보다 훨씬 복잡하고 다양한 환경에서 사용할 수 있는 동기 과정을 다루고 있다. SyncML 에서 언급된 방식은 이러한 방식의 제한형으로 볼 수 있다. 그러나 기존 약-일관성 복사본 관리 시스템에서 사용하는 동기 방식 알고리즘은 너무 복잡하여 실제 시스템에 구현하기 어려운 난점을 갖고 있다. 또한 이러한 시스템들 역시 데이터베이스 동기시 갱신 정보를 동기 장치에게 직접적으로 알리는 방식을 사용하는 문제점을 갖는다.

5. 결론

SyncML 은 장치의 하드웨어/소프트웨어 플랫폼에 무관하여 공통의 데이터 동기화 기능을 정의한 표준이다. 현재 많은 관련 업체들이 SyncML 규격에 따르는 제품을 출시하고 있다. 기존의 데이터 동기화는 PDA, 휴대전화에서의 개인 정보의 동기화 서비스에 집중되었으나, 최근 들어 기업 응용에 데이터 동기화 기능을 추가하는 것에 관심을 갖기 시작하였다. 특히 데이터베이스를 내의 데이터에 대한 동기 지원이 많은 관심을 받기 시작하였다. 이런 결과로 최근 들어 출시되는 SyncML 제품들의 경우 데이터베이스 동기화 기능을 지원하는 제품들이 점차 증가하고 있다.

데이터베이스 테이블에 데이터 동기화 기능을 추가하기 위해서는, 해당 테이블에 발생되는 모든 갱신 작업의 정보가 SyncML 시스템에 전달될 필요가 있다. 그러나 데이터베이스의 경우 여러 응용이 동일한 테이블에 접속하여 갱신하는 경우가 많아, 이런 모든 응용을 수정하는 것은 비현실적이다. 현재 출시되는 대부분의 제품은 이러한 문제점을 갖고 있다.

MoIM-SyncML 은 SyncML 표준 규격을 기반으로 개발된 동기 엔진으로, 다양한 저장소에 대한 동기 기능을 제공한다. 특히 많은 기업 응용의 동기 서비스 제공하기 위해 데이터베이스 동기 기능도 포함되어 있다. MoIM-SyncML 은 트리거와 뷰를 이용하여 기존 응용의 수정 없이 데이터베이스 갱신 정보의 획득이 가능하도록 설계되었다. 본 논문에서는 트리거와 뷰를 이용하여 데이터베이스에 발생하는 갱신 정보를 획득하는 기법에 대해 설명하였다. 설명된 기능은 현재 구현 중에 있다.

참고문헌

[1] Neil Bradley, XML Companion, Addison-Wesley, Third Edition, 2002.

[2] Norman H. Cohen, "Design and Implementation of the MNCRS Java Framework for Mobile Data Synchronization", *Research Report RC 21774, IBM T. J. Watson Research Center*, Nov. 2000.

[3] Eric Gamma, Richard Helm, Ralph Johnson, and John Vlissides, *Design Patterns: Elements of Resuable Object-Oriented Sofware*, Addison-Wesley, 1994.

[4] James J. Kistler and M. Satyanarayanan, "Disconnected Operation in the Coda File System", ACM TOCS 10(1):3-25, Feb. 1992.

[5] D. B. Terry, K. Petersen, M. J. Spreitzer, and M. M. Theimer, "The Case for Non-transparent Replication: Examples from Bayou", *IEEE Data Engineering*, Dec. 1998.

[6] SyncML Homepage, *www.syncml.org*.

[7] SyncML, "SyncML Sync Protocol, version 1.1, http://www.syncml.org/docs/syncml_sync_protocol_v11_20020215.pdf, 2002.