韓國IT서비스學會誌
第8卷 第2號
2009年 6月

메시지다이제스트 테이블 기반의 모바일 XML 문서 동기화 기법*

박 성 진**

A Synchronization Technique of Mobile XML Documents using the Message Digest Tables*

Seong-Jin Park**

■ Abstract ■

In mobile XML environments, the efficient synchronization technique is required to maintain the consistency of replicated XML data because the same data can be replicated between so many different XML documents. In this paper, we propose a message digest based synchronization technique to maintain the consistency of replicated data between client XML documents and a server XML database in mobile XML environments.

The proposed an XML data synchronization technique(XSA) has the advantage of model generality aspect and storage space aspect by using the tuple-based message digest table to detect the data conflicts. The proposed technique can be applicable to the application requiring the consistency of replicated XML data without any restrictions in the heterogeneous(between hierarchical XML documents and relational XML databases) mobile environments.

Keyword : XML, Message Digest, Data Synchronization, Mobile Environment

논문투고일 : 2009년 04월 17일 논문수정완료일 : 2009년 06월 13일 논문게재확정일 : 2009년 06월 15일
* 이 논문은 한신대학교 학술연구비 지원에 의하여 연구되었음
** 한신대학교 컴퓨터공학부 교수

1. 서 론

XML은 웹 상에서의 데이터 교환을 위해 제안된 표준 언어로 사용자가 문서의 구조를 자유롭게 정의할 수 있으며 문서 안에 문서의 구조에 대한 정보를 포함한다. 이러한 XML의 특성은 모든 형식의 데이터가 XML 형태로 기술될 수 있도록 해주며, 웹에서 운용되는 모든 데이터가 통합, 저장, 처리될 수 있는 기반을 제공한다.

현재 많은 정보시스템들이 유무선 인터넷 환경에서 XML을 기반으로 구축·활용되고 있어 다양한 XML 정보들이 유무선 인터넷을 통해 제공되고 있다. 특히, 무선 인터넷 기술의 급속한 발전에 따라 다양한 유·무선 소형 모바일 단말기가 널리 사용되는 포스트 PC 시대가 도래함으로써 필요한 데이터만을 중복저장하여 손쉽게 사용할 목적으로 소형 데이터저장소로서의 XML 문서의 활용이 증대되고 있다[20]. 이에 따라 모바일 단말기안의 개인 XML 문서와 서버 데이터베이스안의 통합 XML 데이터들 간의 버전 관리와 변경 데이터에 대한 효율적인 동기화 기법이 필요하다. 즉, 모바일 단말기에 저장된 클라이언트 XML 데이터와 데이터베이스 서버의 통합 XML 데이터들은 오프라인 상태에서 각각 변경될 수 있으므로 일시적인 불일치성(temporary inconsistency)이 발생하게 된다. 모바일 XML 데이터 동기화(data synchronization)란 바로 이러한 중복된 XML 데이터 사이에서 발생한 불일치성을 해결하는 수단[1, 3]이며 XML 문서기반의 다양한 모바일 어플리케이션 서비스를 위한 핵심 기능이다.

본 논문에서는 모바일 단말기안의 XML 문서들과 데이터베이스안의 통합 XML 데이터 사이의 변경을 탐지하고 동기화시키기 위한 기법을 제안하였다. 제안한 기법은 동기화 대상인 XML 문서들에 대한 메시지 다이제스트 테이블을 RDB에 생성함으로써 XML 문서간에 변경되는 데이터를 엘리먼트 튜플 단위로 탐지하고 동기화를 수행한다. 제안한 동기화 기법의 특징은 계층형 XML 문서와 관계형 XML 데이터 사이의 이질적인 모델들에 대한 새로운 동기화 기법으로 기존 방식과는 달리 엘리먼트 단위 정보를 활용하고 메시지 다이제스트를 통해 충돌 탐지를 위한 데이터의 양을 최소화한 DBMS 독립적인 데이터 동기화 기법이라는 점이다.

2. 관련 연구

웹상의 일반적인 XML 문서와 구별되는 동기화 대상인 모바일 XML 문서의 특성은 다음과 같다.

첫째, 모바일 데이터베이스를 대신하는 저장소로써 다량의 데이터 저장에 따른 XML 문서의 크기가 커서 동기화 비용이 높다. 둘째, 상이한 구조를 갖는 XML 리파지토리안의 관계형 데이터 모델과의 이질적인 모델간의 동기화가 필요하다. 셋째, 데이터 중심의 XML 문서 구조[8] 형식을 따라[그림 8]에 보는 바와 같이 엘리먼트의 애트리뷰트가 주요 저장 항목으로 사용된다. 넷째, 데이터의 존재 여부가 중요하기 때문에 문서 트리 구조에서 형제 관계에 있는 엘리먼트 노드나 애트리뷰트의 순서는 무의미하다. 다섯째, 업무 처리를 위한 데이터 참조와 데이터 변경이 주로 수행되므로 문서 일부 내용의 이동 및 복사와 같은 XML 문서의 구조적 변화는 거의 없다.

위와 같은 문서 특성에 따라 순서 같은 구조적 변경보다는 엘리먼트 단위의 변경 탐지가 필요한 대용량의 XML 문서와 RDB 기반의 XML 리파지토리 데이터간의 이종 모델 동기화 기법에 대한 연구가 필요하다.

모바일 환경에서는 중복된 데이터를 개인 XML 문서와 통합 데이터베이스 XML 데이터 간에 서로 다르게 변경하는 경우가 발생하는데 이것을 충돌(conflict)이라 한다[2]. 충돌을 탐지하고 해결하는 기능은 XML 데이터 동기화의 필수적인 기능이다.

충돌을 해결하기 위한 변경 탐지에 대한 기존 연구들로는 MH-DIFF[14], ATBE[13], X-Diff[19],

DiffXML[18] 그리고 BULD Diff[4] 등과 같이 계층적 데이터에 대한 변경 탐지 알고리즘과 관계형 데이터에 대한 변경 탐지 알고리즘[6, 9, 15, 21], 그리고 유닉스 diff 유틸리티와 같은 문자열에 대한 변화 탐지 알고리즘 등이 있다.

XML 문서안의 데이터들은 계층적으로 구조화(hierarchical structured)된 형태로 이에 대한 변화의 탐지는 NP-hard에 속하는 문제이다[14]. 따라서, 기존 연구들은 트리 구조로 표현된 XML 문서간의 변화를 효율적으로 탐지할 수 있는 다양한 휴리스틱 알고리즘들을 제안하고 있다.

대부분의 연구들은 XML 문서를 순서트리(ordered tree)로 표현한 다음 트리로 표현된 두 문서간에 서로 대응되는 노드들을 결정하고, 이에 대한 편집 연산 스크립트를 생성하여 새로운 문서 버전을 생성하는 다양한 형태의 트리구조와 휴리스틱 알고리즘을 제시하고 있다[12]. 경로매칭 알고리즘을 사용하여 서브트리의 이동, 복사와 같은 구조적 변화를 효율적으로 탐지하는 연구[7]도 이에 속한다. 최근에는 X-Diff[19]와 BULD Diff[18]처럼 대응 노드들의 검색 대상을 감소시키고 비교 후보 노드들의 정렬 문제를 개선한 연구결과도 제시되고 있다. 그러나, 기존의 연구 기법들을 적용할 경우, XML 문서가 갖는 계층적인 구조 정보에 집중함으로써 XML 문서의 엘리먼트와 애트리뷰트 등의 단위정보를 활용하지 못하게 된다. 또한, 내부 노드의 삽입, 삭제와 같은 구조적 변경에 매우 민감하게 된다[11]. 본 논문에서는 XML 문서의 엘리먼트 단위 정보를 활용하여 동일한 엘리먼트 노드들끼리만 대응 여부를 검색하도록 함으로써 대응 노드 후보 범위를 축소하면서 불필요한 정렬 문제를 해결하였으며 메시지 다이제스트 테이블을 통하여 키값이 없어도 엘리먼트 단위의 변경 여부를 탐지할 수 있도록 하였다.

한편, 엘리먼트 튜플들간의 동기화를 위해서 기존 관계형 DBMS에서 지원하고 있는 동기화 기법들을 사용하는 것을 고려할 수 있으나 변경 전의 각 레코드 값을 DBMS 내에 그대로 유지하다가, 동기화 시에 충돌 감지를 위하여 사용하는 변경 이전 값(OV; Old Value) 방식[21]의 경우, 두 가지 버전의 값을 모두 유지해야 하므로 저장 공간에 오버헤드가 크고, 존재하는 각 레코드의 최종 변경 시점을 표현하는 필드를 추가하는 타임스탬프(TS; TimeStamps) 방식[9]은 타임스탬프 전송에 따른 네트워크 부하가 크다는 단점이 있다. 그리고, 두 방식 모두 특정 DBMS에 종속적인 동기화 기법이기 때문에 DBMS가 바뀌는 경우 동기화 기능을 그대로 사용할 수 없다는 문제점이 있다. 제안한 동기화 기법은 특정 DBMS에 종속되지 않고 XML 데이터의 변경 유무, 충돌 여부를 파악하여 동기화시킬 수 있다.

특히, 서로 다른 모델구조를 갖는 계층형 XML 문서와 관계형 데이터베이스 데이터 간의 동기화를 위해 기존 XML 문서 동기화 기법과 기존 관계형 데이터 동기화 기법을 단순 조합할 경우에 순차적으로 이원화하여 동기화를 수행해야하는 하지만 제시한 XSA 알고리즘을 사용할 경우, 상이한 모델간의 동기화 과정을 일원화하여 한꺼번에 수행할 수 있다.

3. 메시지 다이제스트 기반의 XML 동기화 모델

3.1 모바일 XML 동기화 문제

논문에서 제시하는 모바일 XML 데이터 동기화의 기본적인 전략은 개인 모바일 XML 문서들에서 수행된 데이터 변경을 탐지하여 변경된 내용들을 통합 데이터베이스안의 XML 데이터에 적용하는 것이다.

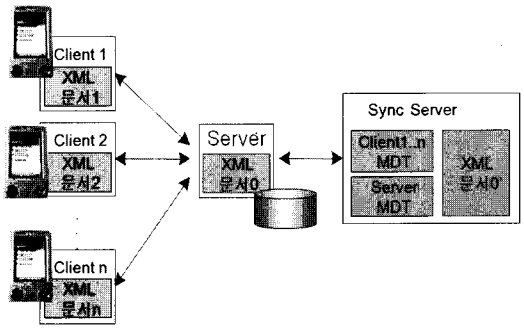
XML 동기화는 중복된 데이터들을 포함하고 있는 XML 문서들 간의 변경 여부를 탐지하여 특정 시점에서의 통합된 하나의 XML 문서 버전을 생성해야 한다[16]. 예를 들어, 통합된 D0.XML 문서의 일부 내용들이 분할되어 D1.XML, D2.XML, D3.XML에 중복저장되어 있는 경우, 이후에 변경

된 D1'.XML, D2'.XML, D3'.XML 문서와의 동기화를 통해서 최신의 갱신 데이터를 갖는 D0'.XML 문서를 생성할 수 있어야한다. 기존 연구들은 구조가 동일한 XML 문서의 변경 탐지를 통한 버전 관리에 집중되어 있으며 변경내용을 반영하기 위한 연산 스크립트생성 등 복잡하고 추가적인 절차를 따라야 한다.

모바일 XML 문서는 RDB안의 XML 저장소.역할을 모바일 장치 안에서 대신하는 관계로 일반적인 XML 문서와는 그 구조와 특성이 다르다. 즉, RDB의 테이블을 계층형 문서형식으로 표현하기 때문에 정형화되어 있고 제한적이다. 즉, 동일한 이름의 엘리먼트는 문서 안에 어느 위치에나 반복될 수는 있으나 그 하위 애트리뷰트들은 모두 동일하다. 이는 서로 다른 애트리뷰트들을 갖는 동일한 이름의 테이블 객체가 존재할 수 없는 이유에서이다. 자유롭게 변경이 일어나는 일반 XML 문서와는 달리 정형화되어 변경 발생 유형이 일정하고 반복적이다. 특히, 엘리먼트의 애트리뷰트들이 중요한 데이터 저장 항목으로 의미적으로 밀접하게 연관된 데이터 그룹을 저장한다.

따라서, 엘리먼트, 텍스트와 동일하게 애트리뷰트를 노드로 표현할 경우, 많은 노드로 인해 계층구조가 복잡해진다. 또한, 연관된 애트리뷰트 노드들을 독립적으로 변경탐지 비교를 하는 것은 엘리먼트 중심의 애트리뷰트들간의 연관 관계를 충분히 활용하지 못해 비교연산이 불필요하게 증가하며 모바일 XML 문서안의 다량의 데이터를 고려할 때 더욱 비효율적이다. 이는 RDB안의 데이터 처리를 엔터티 단위를 고려하여 튜플 단위로 처리하지 않고 단순한 테이블 행과 열의 구조만을 고려하여 개별 셀의 컬럼값 단위로 처리하는 경우와 마찬가지이다. 따라서, 제안한 동기화 기법에서는 애트리뷰트를 단독 노드로 간주하지 않고 엘리먼트 서브 노드로 간주하여 엘리먼트는 튜플로, 엘리먼트 애트리뷰트는 칼럼으로 표현하는 엘리먼트 테이블(ET)을 사용하여 엘리먼트 단위만 비교가 이루어지도록 하였다. 이는 계층형 문서 모델과 관계형 문서 모델간의 자연스러운 변환이 가능하도록 하며 불필요한 애트리뷰트 노드간의 비교연산을 줄임으로써 대량의 모바일 XML 데이터 처리에 적합하다.

본 논문에서는 [그림 1]과 같이 중복분할된 XML 서브 문서들의 다양한 변경이 발생할 경우, 이에 대한 적절한 변경 탐지를 통한 새로운 통합 XML 문서 생성이 가능한 MDT 기반 동기화 기법을 제안하였다.



[그림 1] 모바일 XML 동기화

제안한 동기화 기법의 특성은 XML 문서 구조가 동일한(DTD나 XML Schema가 동일) 문서들의 동일 레벨 즉, 일치하는 엘리먼트들 끼리만의 비교를 통해 변경여부를 확인함으로써 비교 대상을 감소시켰으며 메시지 다이제스트 알고리즘을 활용하여 동일 엘리먼트 값들의 변경 여부를 빠르고 손쉽게 검증할 수 있도록 하였다. 또한, XML 문서들을 엘리먼트 중심의 관계형 데이터베이스에 저장함으로서 모델간의 상이함에도 불구하고 비교가 쉽게 이루어질 뿐만 아니라 동기화된 결과 문서의 생성도 별도의 스크립트 연산 생성 없이도 가능하다.

3.2 XML 동기화 테이블 구조

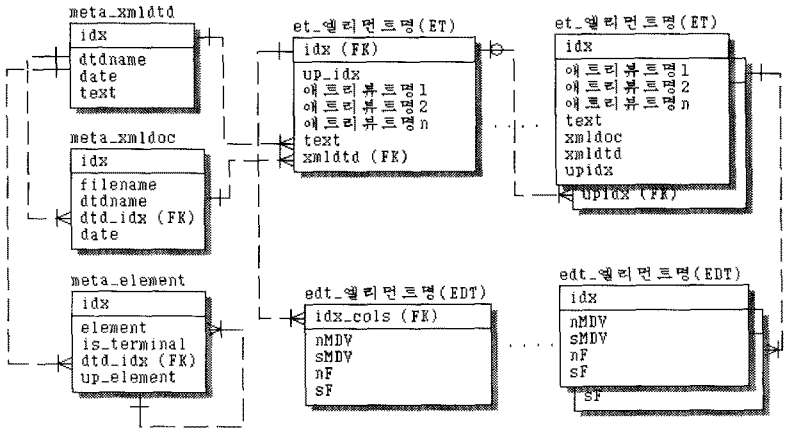
DTD 분석에 따른 중복 분할된 XML 문서들의 동기화를 위한 저장 구조를 관계형 데이터베이스 안에 생성한다. 기존 DTD 의존적 혹은 독립적인

스키마 생성 방식들[5, 17] 중에서 제시한 스키마 구조는 엘리먼트 구조 의존적 스키마 형태로 XML 문서가 갖는 계층적인 구조적 특성을 관계형 데이터 모델의 외래키(foreign key)를 사용하여 엘리먼트간의 상하(직계) 관계를 일대 다의 참조 관계로 변환하여 표현하였다. 특히, XML 데이터의 수정이 가능하도록 문서를 엘리먼트 단위로 쪼개어 저장하는 분할저장 방식을 사용하였으며 엘리먼트를 관계형 데이터베이스의 테이블에 대응시키고, 애트리뷰트는 엘리먼트 테이블의 칼럼으로 매핑시켜 다양한 XML 문서간의 동기화가 가능하도록 하였다.

XML 문서를 관계형 데이터베이스에 저장할 때 XML 문서에 대한 메타 데이터와 엘리먼트 속성들에 대한 정보도 함께 저장할 수 있도록 [그림 2]와 같은 구조를 사용하였다.

우선, XML 문서에 대해 메타정보를 입력할 수 있도록 메타 테이블을 생성한다. [그림 2]에서 meta_xmldtd 테이블은 DTD 문서별로 고유한 id값과 DTD 문서의 이름, 입력시간, DTD 문서 전체 내용을 저장한다. meta_xmldoc 테이블의 경우, 입력한 XML 문서에 대한 고유한 id값, XML 문서의 파일명, 참조하는 DTD 인덱스 값과 DTD 문서명, 입력시간 등이 저장된다. 이를 통해 어떤 문서가 언제 입력되었는지 알 수 있으며, XML 문서를 복원하는 경우 XML 문서명을 확인하는 데에도 사용된다. meta_element 테이블의 경우, 엘리먼트들간의 구조정보를 저장하기 때문에 엘리먼트에 대한 구조 검색이 용이하며 여기에는 각 엘리먼트의 고유한 id값과 엘리먼트명, 단말 노드 유무(단말이면 '1', 아니면 '0'), 참조하는 DTD 문서의 인덱스, 상위 엘리먼트에 대한 인덱스 값을 저장한다.

XML 문서의 DTD를 파싱함으로써 [그림 2]와 같은 et_로 시작하는 엘리먼트 테이블(ET; Element Table)을 생성한다. 먼저, 유일(unique)한 엘리먼트명을 갖는 테이블이 엘리먼트 수만큼 생성되며 각 엘리먼트 테이블은 고유 인덱스(idx) 컬럼과 상위 엘리먼트와의 연결을 위한 상위 인덱스(up_idx) 컬럼을 갖는다. 또한, XML 문서 정보에 대한 인덱스(xmldoc)와 DTD 정보 참조를 위한 인덱스(xmldtd)가 포함된다. 루트 엘리먼트의 경우 상위 엘리먼트가 없으므로 상위 인덱스 값은 갖지 않으며 그 외에 XML 문서의 내용을 저장하기 위해서 각 엘리먼트의 애트리뷰트는 테이블의 컬럼명으로, 엘리먼트의 텍스트는 'text'라는 컬럼명으로 각각 고유한 컬럼이 생성된다. 엘리먼트 테이블당 edt_로 시작하는 하나의 메시지다이제스트 테이블(EDT;Element Digest Table)을 생성하며 노드 다이제스트값 nMDV, 하위노드 메시지다이제스트값 nSDV, 노드와 하위노드들의 변경여부를 나타내는 nF, sF 플래그 칼럼이 해당한다.



[그림 2] 메타 테이블 및 엘리먼트 테이블 구조

3.3 엘리먼트 다이제스트 테이블

제안하는 XSA 동기화 알고리즘은 메시지 다이제스트 MD4[10]를 기반으로 한다. MD4는 보안 관련 표준알고리즘의 하나로 그 용도는 해시 함수와 같다. 입력으로 다양한 크기의 메시지를 받아서 출력으로 유일한 128비트의 지문(fingerprint) 형 메시지 다이제스트를 생성한다. 즉, 생성된 메시지 다이제스트 결과 값인 DigestedData1과 Digested Data2의 값이 동일하면 원본 입력데이터인 Data1과 Data2는 동일한 데이터로 간주할 수 있다. 이는 메시지 다이제스트 함수가 생성하는 결과 값이 입력 값에 따라 유일하기 때문이며 이러한 다이제스트 방법은 대량의 데이터가 소량의 데이터로 변경됨으로, 데이터의 내용 변형은 고려하지 않고 데이터의 변화 여부를 판단하는 방법으로 유용하다. 일반적으로는 메시지 다이제스트 방법을 보안 분야에서 전송되는 데이터간의 변조 여부를 판단하기 위해서 사용하지만 본 논문에서는 이 방법을 XML 엘리먼트 튜플 단위의 불일치 탐지를 위한 메시지 다이제스트 테이블 생성에 적용하였다.

기존 연구처럼 서브트리의 구조와 내용에 대한 해시값을 사용하여 변경 여부를 검사하는 방법의 경우, XML 문서 트리의 구조에 따라 하위 서브트리의 해시값을 포함하고 있어 변경이 없는 하위 서브트리에 대한 불필요한 탐지를 방지할 수 있지만 노드가 많은 경우에는 해시값 생성이 부담이 될 수 있다. 본 논문의 메시지 다이제스트 방법은 밀접한 연관성을 갖는 속성 노드들을 그룹핑하여 엘리먼트 단위로 다이제스트함으로써 개별 애트리뷰트 노드들의 해싱 비용을 줄이면서 하위 엘리먼트의 다이제스트 값이 상위 엘리먼트에 재귀적으로 내포하도록 함으로써 변경 없는 하위 엘리먼트의 불필요한 비교도 방지하였다.

또한, 사용한 메시지 다이제스트 방법은 메시지 변조 방지에 사용되는 해시 알고리즘으로 서로 다른 엘리먼트끼리 같은 해시값을 가질 확률이 상대적으로 극히 적다. 여기에 엘리먼트 하위의 애트리뷰트 중에는 RDB의 기본 키에 해당하는 유일(unique)한 칼럼 값이 반드시 포함되어 다이제스트 결과 값의 유일성을 더욱 강화하여 별도의 비교키 값이 없어도 대응하는 엘리먼트간의 동기화가 가능하다.

XML 문서안의 데이터 즉, 엘리먼트 값들의 변화를 알기 위해서 엘리먼트 테이블의 각 튜플과 애트리뷰트를 모두 식별하여 직접 비교하지 않고도 메시지 다이제스트 결과값 하나만으로 엘리먼트 튜플 단위의 비교가 가능하다. 이 과정에서 엘리먼트 다이제스트 테이블이 추가적으로 필요하다.

EDT 테이블은 데이터베이스안의 튜플들(XML 문서의 엘리먼트 정보들)을 하나의 메시지로 보고 메시지 다이제스트를 적용한 결과 값들을 저장하기 위한 저장소이다. 불일치 탐지는 두 XML 문서의 동일한 EDT 테이블 사이의 불일치 탐지를 의미한다. 먼저, X 엘리먼트를 위한 EDT 테이블의 스키마는 다음과 같다.

edt_X(idx_cols, nMDV, sMDV, nF, sF)

이때, EDT 테이블의 idx_cols은 대응되는 엘리먼트 테이블의 xmldtd, xmldoc, up_idx, idx 4개 컬럼 값의 조합을 줄여 표현한 것이며, nMDV은 엘리먼트 테이블의 idx_cols 해당 애트리뷰트를 제외한 나머지 애트리뷰트 값들(av_5, av_6, …, av_n)을 문자

$nMDV = hexstr(MD4(av_5 \oplus av_6 \oplus \cdots \oplus av_n))$
n : 애트리뷰트의 개수(릴레이션 ET의 degree)
avk : k번째 애트리뷰트 값
⊕ : 문자열 결합 연산자
MD4(m) : 메시지 m을 다이제스트하는 함수
hexstr(h) : 128비트 해시 값 h를 16진수 표기법에 따른 32바이트로 변환하는 문자열 함수

sMDV = hexstr(MD4(sMDV1 ⊕ … ⊕ sMDVn))
n : 자식 엘리먼트의 개수
sMDVk : k번째 자식 엘리먼트의 서브트리 다이제스트 값
⊕ : 문자열 결합 연산자
MD4(m) : 메시지 m을 다이제스트하는 함수
hexstr(h) : 128비트 해시 값 h를 16진수 표기법에 따른 32바이트로 변환하는 문자열 함수

열 결합한 뒤, 이 값에 대해 MD4 알고리즘을 적용하여 얻은 128비트 해시 값을 32바이트 문자열로 표현한 값으로 다음과 같이 정의할 수 있다. nMDV들은 이전 동기화 이후 각 EDT 테이블간의 튜플 단위의 불일치 여부를 탐지하기 위해 사용된다.

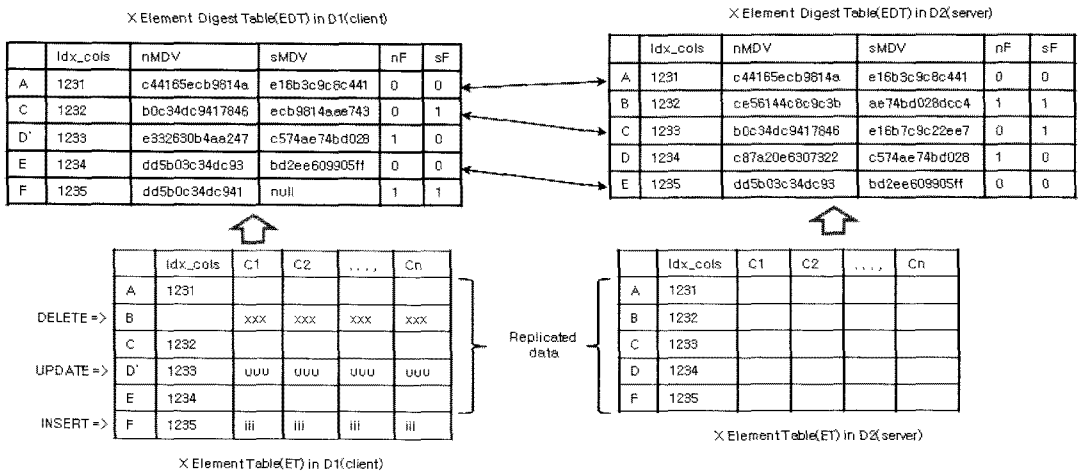
한편, EDT 테이블사이의 nMDV 비교시 불필요한 하위 노드들에 대한 비교횟수를 줄이기 위하여 변경이 발생한 하위 엘리먼트들에 대해서만 불일치 유무를 검색할 필요가 있다. 이를 위해 특정 엘리먼트 노드의 모든 하위 자손 노드 엘리먼트의 nMDV 값들을 포함하여 한꺼번에 다이제스트한 결과 값인 sMDV를 정의하면 다음과 같다.

이때, k번째 자식 엘리먼트의 서브트리 다이제스트값 $sMDV_k$는 다시 k번째 자식 엘리먼트를 부모노드로 하는 또 다른 n개의 손자노드들의 서브트리 다이제스트값 sMDV를 문자열 결합한 뒤 다시 다이제스트한 값이다. 따라서, 특정 엘리먼트들간의 sMDV 값들이 서로 같다면 하위 엘리먼트들간에는 변경이 없으므로 하위 노드들에 대해서는 더 이상 동기화 여부를 판단하지 않아도 된다.

4. 데이터 동기화 알고리즘

4.1 XSA동기화 알고리즘

XSA(XML Synchronization Algorithm) 알고리즘은 [그림 3]과 같은 ET 및 EDT 테이블들을 대상으로 적용할 수 있다. [그림 3]에서 EDT는 [그림 4]의 XSA 전처리 과정을 통해 각 엘리먼트 튜플의 nMDV와 sMDV 값이 하위 엘리먼트 테이블부터 유도되었고 상위 부모 엘리먼트의 sMDV값이 1(change)로 설정되어 변경내용이 있음을 가정하였다. EDT 테이블에는 추가로 2개의 nF, sF 비트 플래그 칼럼을 두어 XML 문서의 노드 데이터의 변경과 서브트리 노드 데이터의 변경 유무를



[그림 3] XSA 알고리즘 적용에 따른 ET와 EDT 테이블 예

표시(0 : unchanged, 1 : changed)하였다.

[그림 3]은 클라이언트 XML 문서에 3가지 변경(B노드는 삭제, D노드는 D'노드로 변경, F노드는 추가)이 발생한 경우를 반영한 예이다. 클라이언트와 서버 엘리먼트 간에 동일한 EDT 테이블의 nMDV 값이 일치하면 해당 노드에 변경이 없음을, 추가로 sMDV값까지 일치하면 모든 하위 노드들에도 변경이 없음을 의미한다. 따라서, nF가 1이면 변경이 있음을 sF가 1이면 하위 노드 중에 변경이 있음을 의미한다.

각 튜플의 메시지 다이제스트 값 nMDV 값이 동일하면 두 튜플의 엘리먼트 데이터는 동일하다. [그림 3]에서 D1은 변경 여부를 식별하기 위한 XML 문서를, D2는 D1의 변경 내역을 반영하여 동기화 시키고자 하는 원본 XML 문서로서 동기화 과정을 통해 D1의 변경 내용이 D2에 반영되어 D2' XML 문서가 생성된다. D1과 D2의 순서를 바꾸어 알고리즘을 적용해도 결과는 동일하지만 각 ET의 카디널러티가 작은 XML 문서를 D1으로 하는 것이 비교대상을 줄일 수 있어 효율적이다.

1) 단계 1_전처리(EDT 테이블 구축)

상향식으로 최하위 단말 노드 Xi, Yi의 ET 테이블로부터 EDT 테이블 생성을 시작한다. 각 EDT 튜플의 nMDV값을 구하고 자식 EDT 튜플들의 nMDV값을 합산하여 다이제스트하여 sMDV값을 생성하고 자식 EDT가 없는 단말 EDT 테이블 튜플의 경우에는 sMDV값으로 null값을 설정한다. 이 과정을 최상위 루트 노드 Xi, Yi EDT 테이블까지 진행한다.

2) 단계 2_변경탐지 비교

클라이언트 쪽의 동기화 대상 문서를 D1, 서버 쪽의 원본 문서를 D2라 하면 D1, D2의 최상위 Xi, Yi EDT테이블부터 하향식으로 비교를 시작한다. 두 문서 D1, D2에 속한 Xi, Yi EDT 테이블의 각 튜플에 대해서 만약 부모 EDT 튜플이 없다면 Xi와 Yi EDT 테이블 튜플사이에 일치하는 nMDV 값이 존재하는지 비교한다. 만약, 있다면 각 Xi와 Yi 튜플의 nF에 0으로 설정(노드 비변경)하고 Xi와 Yi EDT 테이블 튜플 사이에 일치하는 sMDV 값이 존재하는지 비교하여 있다면 각 Xi와 Yi 튜플의 sF에 0설정(비변경)하고 없다면 각 Xi와 Yi EDT 테이블 튜플의 sF에 1로 설정(변경)한다. 한편, 일치하는 nMDV값이 존재하지 않는다면 각 Xi와 Yi 튜플의 nF에 1로 설정(변경)하고 Xi와 Yi EDT 테이블 튜플 사이에 일치하는 sMDV값이 존재하는지 비교하여 있다면 각 Xi와 Yi 튜플의 sF에 0설정(비변경)하고 없다면 각 Xi와 Yi 튜플의 sF에 1설정(변경)한다. 이 경우, 클라이언트 새로운 엘리먼트 자료가 삽입되었거나 혹은 서버에 기존 엘리먼트가 삭제된 경우이다.

만약 부모 EDT 튜플이 있다면 부모 EDT 튜플의 sF가 0이면 nF와 sF를 0으로 설정(서브트리 비변경)한다. 그리고, 부모 EDT 튜플의 sF가 1이면 Xi와 Yi 튜플사이에 일치하는 nMDV값이 존재

```
1  문서 D1, D2의 최하위단말노드를 Xi, Yi로 설정
2  Xi, Yi의 ETi(X), ETi(Y) 테이블로부터 EDTi(X), EDTi(Y) 테이블 생성
3  각 EDTi(X), EDTi(Y) 튜플j의 nMDVj 값을 생성
4  EDTi(X), EDTi(Y) 튜플j의 자식 EDTk 튜플들의 nMDV값을 합산다이제스트하여 sMDV 값을 생성
5  자식 EDTk 튜플들이 없는 경우, sMDV값을 null로 설정
6  if Xi, Yi가 최상위루트노드이면
7      종료
8  else {
9      Xi, Yi의 부모노드를 새로운 Xi, Yi로 설정
10     2~10과정을 반복 }
```

[그림 4] XSA 알고리즘(전처리 단계)

```
1  문서 D1, D2의 최상위노드 Xi와 Yi의 EDTi 테이블들부터 비교 시작
2  if (각 EDTi 튜플 j의 부모 EDT 튜플이 존재하는가?) {
3    if (각 부모 EDT 튜플의 sF가 0인가?) {
4       nF와 sF를 0으로 설정//서브트리 비변경 }
5    else {
6       if (각 부모 EDT 튜플의 sF가 1인가?) {
7            if (Xi와 Yi의 각 EDTi 튜플 j사이에 일치하는 nMDV값이 존재하는가?) {
8                 일치하는 튜플 j들의 nF에 0 설정//노드 비변경
9                 if (Xi와 Yi의 각 EDTi 튜플 j사이에 일치하는 sMDV값이 존재하는가?)
10                 일치하는 튜플 j들의 sF에 0 설정//비변경
11                else
12                 일치하는 튜플 j들의 sF에 1 설정//변경 }
13           else {
14                일치하는 튜플 j들의 nF에 1 설정//노드 변경
15                if (Xi와 Yi의 각 EDTi 튜플 j사이에 일치하는 sMDV값이 존재하는가?)
16                 일치하는 튜플 j들의 sF에 0 설정//비변경
17                else
18                 일치하는 튜플 j들의 sF에 1 설정//변경 }
19 } } }
20 else {
21   if (Xi와 Yi의 각 EDTi 튜플 j사이에 일치하는 nMDV값이 존재하는가?) {
22    일치하는 튜플 j들의 nF에 0 설정//노드 비변경
23      if (Xi와 Yi의 각 EDTi 튜플 j사이에 일치하는 sMDV값이 존재하는가?)
24          일치하는 튜플 j들의 sF에 0설정//비변경
25      else
26          일치하는 튜플 j들의 sF에 1설정//변경 }
27   else {
28      일치하는 튜플 j들의 nF에 1설정//노드 변경
29      if (Xi와 Yi의 각 EDTi 튜플 j사이에 일치하는 sMDV값이 존재하는가?)
30          일치하는 튜플 j들의 sF에 0설정//비변경
31      else
32          일치하는 튜플 j들의 sF에 1설정//변경 }
33 } }
34 if Xi, Yi가 자식노드가 있다면 {
35   Xi, Yi의 자식노드를 새로운 Xi, Yi로 설정
36   2~36 과정을 반복 }
37 else
38   종료
```

[그림 5] XSA 알고리즘(변경탐지 비교 단계)

하는지 비교한다. 만약 존재한다면 각 Xi와 Yi 튜플의 nF에 0으로 설정(노드 비변경)하고 Xi와 Yi EDT 테이블 튜플 사이에 일치하는 sMDV값이 존재하는지 비교하여 있을 경우에는 각 Xi와 Yi 튜플의 sF에 0으로 설정(비변경)하고 없을 경우에는 Xi와 Yi 튜플의 sF에 1로 설정(변경)한다.

만약, 일치하는 nMDV값이 존재하지 않는다면 각 Xi와 Yi 튜플의 nF에 1을 설정(변경)하고 Xi와 Yi EDT 테이블 튜플 사이에 일치하는 sMDV값이 존재하는지 비교한다. 만약, 있다면 각 Xi와 Yi 튜플의 sF에 0으로 설정(비변경)하고 없다면 각 Xi와 Yi 튜플의 sF에 1로 설정(변경)한다.

3) 단계 3_후처리

D1, D2의 최상위 Xi, Yi EDT 테이블부터 하향식으로 통합을 시작한다. D2의 Yi EDT 테이블 중에서 nF가 1로 설정된 튜플들을 삭제하고 D1의 Xi EDT 테이블 중에서 nF가 1로 설정된 튜플들을 Yi EDT 테이블에 추가한다. 이와 같은 과정을 하위 자손 EDT 테이블에 대해서도 동일하게 튜플

```
1  문서 D1, D2의 최상위노드를 Xi와 Yi로 설정
2  D2의 EDTi(Y) 테이블 중에서 nF가 1인 튜플들을 삭제
3  D1의 EDTi(X)테이블 중에서 nF가 1인 튜플들을 EDTi(Y) 테이블에 추가
4  if Xi, Yi가 자식노드가 있다면 {
5      Xi, Yi의 자식노드를 새로운 Xi, Yi로 설정
6      2~6 과정을 반복 }
7  else
8      종료
```

[그림 6] XSA 알고리즘(후처리 단계)

삭제와 추가 과정을 반복 적용한다.

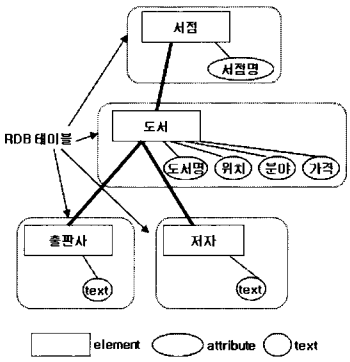
[그림 4], [그림 5], [그림 6]은 XSA 알고리즘이 [그림 3]과 같은 클라이언트와 서버쪽 EDT 간의 불일치 즉, 충돌이 발생했을 경우, 각 EDT에 튜플이 새로 추가되거나 기존 튜플이 수정, 삭제되었는지 여부에 대해 순차적으로 테스트하고 메시지 다이제스트 값을 사용하여 동기화시켜가는 과정들을 보여준다. 제안한 XSA 동기화 알고리즘은 클라이언트와 서버의 데이터가 모두 변경된 경우에는 정책 설정에 의해서 동기화 방향을 결정할 수 있다. 또한, 서버와 클라이언트에 대한 ET와 EDT 사이의 동기화는 실질적인 동기화 작업의 사전 준비 과정에 해당함으로 동기화 작업과 분리되어 실행시킬 수 있어 동기화 운영의 융통성을 제공하고 동기화 시간을 단축시킬 수 있다.

5. 적용 예

제안한 XSA 알고리즘을 방문 도서 판매를 위한 판매원들의 모바일 장치에 저장되는 XML 데이터들의 동기화에 적용하였다. 동기화 대상인 도서 정보에 관한 XML 문서들은 [그림 7]과 같은 DTD 구조를 따른다고 가정한다.

[그림 8]과 같은 도서 정보를 저장한 D2.XML 문서를 저장할 경우 XSA 전처리 과정에 따라 우선 D2.XML 문서에 대한 여러 정보가 메타 테이블에 저장된 뒤, 각 문서 내용도 서점, 도서, 출판사, 저자 4개의 ET 테이블들에 튜플들로 나누어 입력된다.

실제 엘리먼트 테이블간의 구조적인 연결은 없

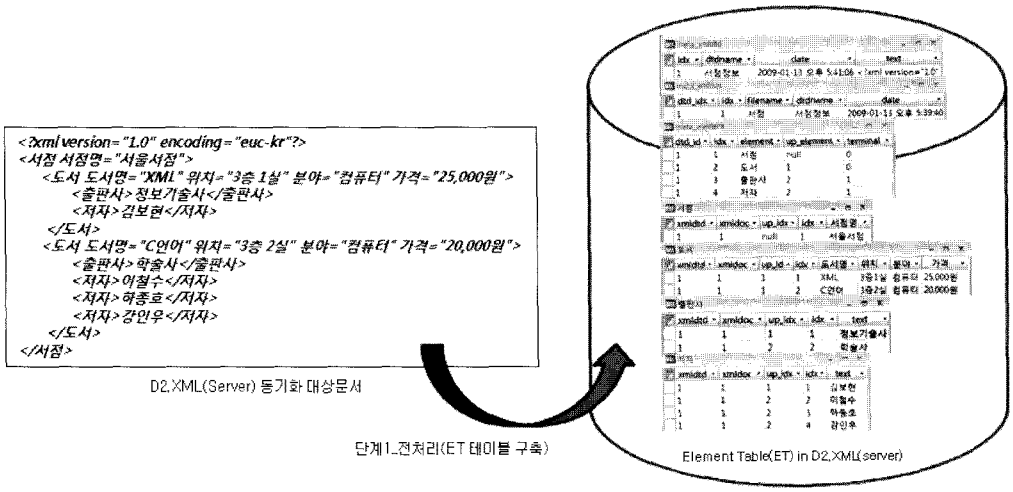


[그림 7] DTD 구조 예

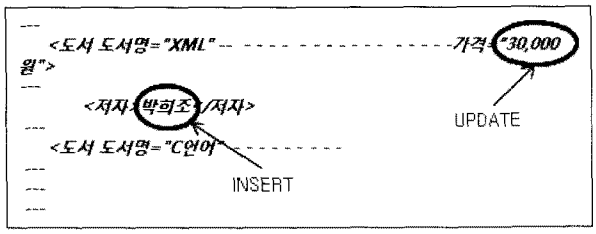
지만 구조적인 정보가 meta_element 테이블에 존재하므로 이를 이용하여 각 테이블에 매핑되어 저장되어진다. XML 문서의 내용 정보는 내부적으로는 관계형 데이터베이스 언어인 SQL의 insert 구문의 형태로 변환하여 저장하게 되는데 동일한 DTD 구조를 갖는 여러 XML 문서는 동일한 ET 테이블 안에 저장이 가능하고, 다른 DTD를 갖는 XML 문서가 있을 경우에만 새로운 구조를 추가적으로 만들어 사용한다.

[그림 9]는 서버에 저장된 D2.XML과 동일한 내용을 가진 모바일 클라이언트안의 D1.XML에 2가지 변경이 발생한 경우의 XSA 변경 탐지 과정의 일부를 보여준다.

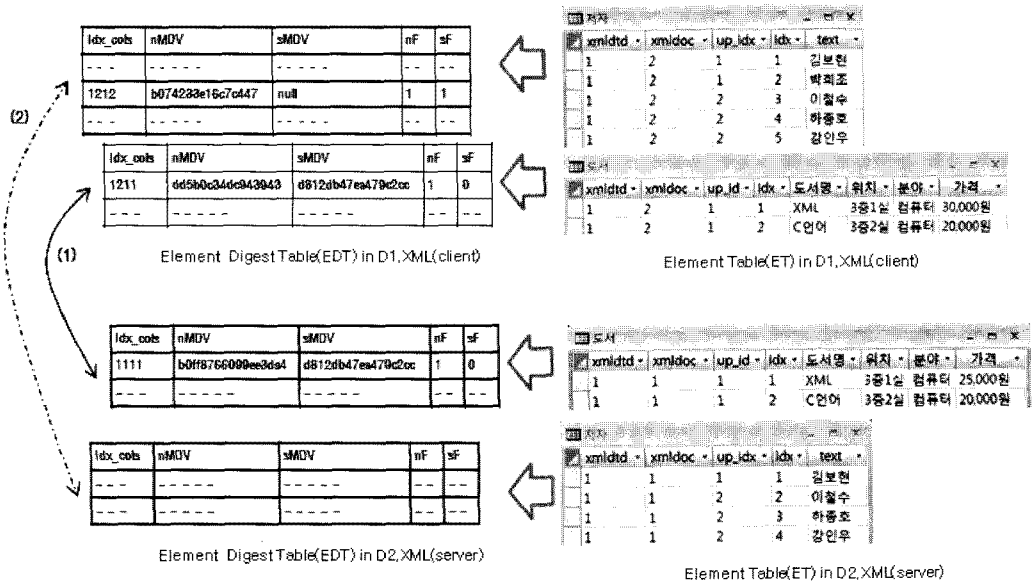
도서 엘리먼트의 가격 속성 값이 '25,000원'에서 '30,000원'으로 수정됨에 따라 (1)과 같이 두 튜플간의 nMDV값이 상이하게 되어 nF값이 1로 설정되고 하위 정보에 대한 수정 내용은 없으므로 sMDV값은 0으로 설정된다. XSA 알고리즘의 후처리 과정을 통해 nF값이 1인 튜플은 D2.XML에서 삭제



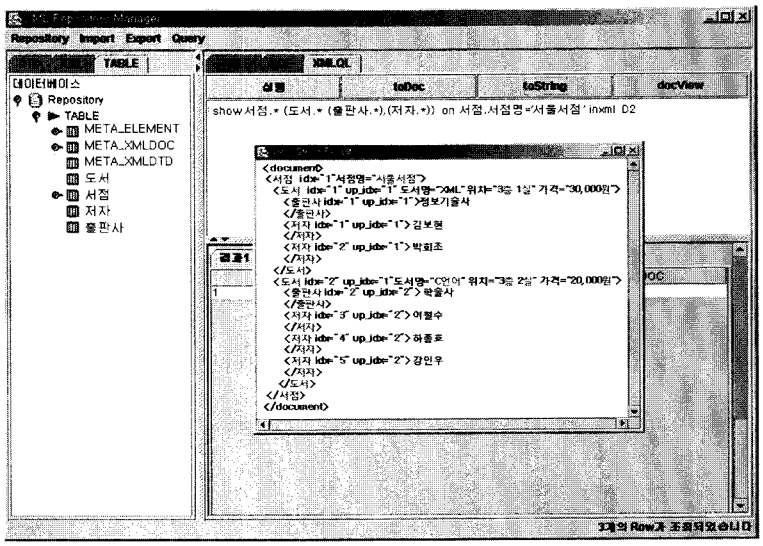
[그림 8] XML 문서 예



D1.XML(Clinet)의 변경내용



[그림 9] XSA 알고리즘 적용 예



[그림 10] XSA 동기화의 수행 결과

되고 nF값이 1인 D1.XML의 튜플은 새로운 통합 D2.XML에 추가된다. 새로이 D1.XML에 추가된 저자 엘리먼트 값 '박희조'는 새로운 튜플로 추가되어 새로운 nMDV값을 갖게 되며 하위 엘리먼트가 없으므로 sMDV값은 null값을 갖게 된다. 이후, XSA 알고리즘에 의해 (2)와 같이 일치하는 튜플이 없으므로 nF와 sF 플래그 값이 모두 1로 설정되며 새로운 통합 D2.XML에 새로운 튜플로 추가된다.

[그림 10]은 XSA 동기화 알고리즘을 수행한 후의 서버 데이터베이스에 저장된 새로운 통합 D2.XML 문서 내용을 검색한 결과를 보여준다. show절로 시작되는 질의어의 실행 결과를 통해서 D1.XML 문서에 수정되고 추가된 변경 내용들이 올바르게 동기화되어 있음을 확인할 수 있다.

XSA 알고리즘은 계층형 문서와 관계형 데이터베이스안의 데이터 간의 동기화를 지원하는 DBMS 독립적인 동기화 기법으로 메시지 다이제스트를 사용함으로써 키값이 없어도 변경 여부를 분석할 수 있으며 불필요한 하위노드들에 대한 변경 비교 연산이 필요없는 장점을 갖는다. 기존 연구와의 정성적인 특성을 비교하면 <표 1>과 같다.

〈표 1〉 동기화 기법 비교

기존연구	모델	구조	활용정보	DBMS	스크립트	변경연산
MH-DIFF[14]	계층형	트리	구조	관련없음	사용	입력/수정/삭제/이동/복사
X-Diff[19]	계층형	트리	구조	관련없음	사용	입력/수정/삭제
DiffXML[18]	계층형	트리	구조	관련없음	사용	입력/수정/삭제/이동
BULDDiff[4]	계층형	트리	구조	관련없음	사용	입력/수정/삭제/이동
OV[21]	관계형	테이블	버전	DBMS종속	없음	입력/수정/삭제
TS[9]	관계형	테이블	시간	DBMS종속	없음	입력/수정/삭제
XSA	계층/관계형	테이블	구조 (엘리먼트중심)	DBMS독립	없음	입력/수정/삭제

6. 결 론

모바일 환경에서 클라이언트 데이터 저장소로서 XML 문서의 사용이 증가하고 있다. 이에 따라, 서버 데이터베이스와의 동기화 문제가 이슈화되고 있으나 계층적인 XML 문서 구조와 관계형 데이터베이스 테이블의 구조가 상이함에 따라 기존 XML 동기화 기법이나 데이터베이스 동기화 기법을 직접 적용하기 어려운 문제점이 있다. 본 논문에서는 이질적인 두 모델간의 데이터를 동기화시킬 수 있는 엘리먼트 단위의 저장구조를 갖는 DBMS 독립적인 데이터 동기화 기법을 제안하였다.

제안한 데이터 동기화 알고리즘 XSA는 튜플 단위의 메시지 다이제스트 결과를 충돌을 탐지하기 위한 정보로 사용함으로써 이전 버전 없이도 변경여부를 탐지하고 동기화시킬 수 있다. 또한, 엘리먼트 단위정보에 기반하여 검색 대상을 축소하고 변경이 없는 하위 엘리먼트 노드들에 대한 불필요한 검색도 줄임으로써 동기화 과정을 단순화하였다. 제안한 XML 데이터 동기화 기법은 다양한 모바일 환경에서 중복된 XML 데이터간의 일치를 요구하는 많은 응용 프로그램을 위해 사용될 수 있다.

참 고 문 헌

[1] 이상윤, 박순영, 이미영, 김명준, "이동 DBMS의 데이터 동기화 기술 분석", 「데이터베이스 연구회지」, 제17권, 제3호(2001), pp.29-41.

[2] Bauer, Jonathan Andover, Bodge, Andrew, "System for synchronizing shared data between computers", United States Patent No.5 (1998), p.884.

[3] Franky Lam, Nicole Lam, and Raymond K. Wong, "Efficient synchronization for mobile XML data", *CIKM* 2002, pp.153-160.

[4] Gregory Cobena, Serge Abiteboul, and Amelie Marian, "Detecting Changes in XML Documents", *Proc. of the 18th Int'l conf. on Data Engineering* 2002, pp.41-52.

[5] Gordana Pavlovic-Lazetic, "Native XML Databases vs. Relational Databases In Dealing With XML Document", *Kragujevac J. Math.* Vol.30(2007), pp181-199.

[6] IBM, DB2 Sync Server Administration Guide 7.2.1, Whitepaper.

[7] Kyong-Ho Lee, Yoon-Chul Choy, and Sung-Bae Cho, "An Efficient Algorithm to Compute Differences between Structured Documents", *IEEE Trans. Knowl. Data Eng.* Vol.16 No.8(2004), pp.965-979.

[8] Nicolas Regal, "Review-Storing and Querying XML Data using an RDMBS", *ACM SIGMOD Digital Review,* Vol.22, No.3(1999), pp.27-34.

[9] Oracle, Oracle 9i Lite Administration Guide 5.0.1, Whitepaper.

[10] Rivest, R., "The MD4 Message Digest Algorithm", RFC 1320, *MIT and RSA Data Security,* 1992.

[11] Rodrigo Cordeirodos Santos, and Carmem S. Hara, "A Semantical Change Detection Algorithm for XML", *SEKE* 2007, pp.438-443.

[12] Shu Yao Chien, Vassilis J. Tsotras, and Carlo Zaniolo, "XML Document Versioning", *ACM SIGMOD Record*, Vol.30(2001), pp.46-53.

[13] S. J. Lim and Y. K. Ng, "An Automated Change-Detection Algorithm for HTML Documents Based on Semantic Hierarchies", *In Int'l Conf. on Data Engineering*, 2001, pp.303-312.

[14] S. S. Chawathe and H. Garcia-Molina, "Meaningful Change Detection in Structured Data", *In Proc. Of SIGMOD* 1997, pp.26-37.

[15] Sybase, Synchronization Technologies for Mobile and Embedded Computing, Whitepaper.

[16] Tancred Lindholm, "A three-way merge for

XML documents", *Proc. for the 2004 ACM symposium on Document Engineering* 2004, pp.1-10.

[17] Yasser Abdel Kader, and Barry Eaglestone, "Siobhán North : An Analysis of Relational Storage Strategies for Partially Structured XML", *WEBIST* 2008, pp.165-170.

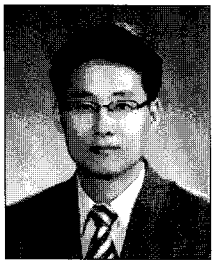
[18] Yan Chen, Sanjay Kumar Madria, and Sourav S. Bhowmick, "D iffXML : Change Detection in XML Data", *DASFAA* 2004, pp.289-301.

[19] Yuan Wang, David J. Eewitt, and Jin-yi Cai, "X-Diff : An Effective Change Detection Algorithm for XML Documents", *Proc. of the 19th Int'l conf. on Data Engineering* 2003, pp.519-530.

[20] Van Munin Chhieng, and Raymond K. Wong, "Efficient Distributed XML Data Management for Mobile Commerce", *IEEE* 2005, pp.460-465.

[21] W. Labio and H. G. Monila, "Efficient Snapshot Differential Algorithms for Data Warehousing", *In Proc. of the 20th VLDB Conf.* 1996, pp.63-74.

◈ 저 자 소 개 ◈



박 성 진 (sjpark@hs.ac.kr)

현재 한신대학교 컴퓨터공학부 교수로 재직 중이며, 고려대학교 전산학과를 졸업하고 고려대학교 일반대학원 전산학 석사 및 박사를 취득하였다. 한국전자통신연구원 소프트웨어연구소에서 선임연구원으로 ERP 패키지 시스템 개발, 시스템 통합 프로젝트 등에 참여한 적이 있다. Computers and Security 등의 국제학술지 및 한국정보과학회논문지, 한국정보처리학회논문지, 인터넷정보학회논문지, 데이터베이스저널 등의 국내학술지에 논문을 게재한 바 있다. 주요 관심분야는 데이터웨어하우징, 데이터 동기화 기술, 리파지토리 시스템 등이다.