임베디드 DBMS 환경을 위한 DBMS 독립적인 데이터 동기화 기법

장우석 강영호 노형준 정병대 손성용
포디홈네트(주) 기업부설연구소, {linus, tristan, urmine, nicolas, xtra}@4dhome.net

김상욱
강원대학교 컴퓨터정보통신공학부, wook@kangwon.ac.kr

A DBMS-Independent Approach for Data Synchronization in Embedded DBMS Environment

Woo Seog Jang, Yeong Ho Kang, Hyong Joon Noh, Byong Dae Jung, Sung-Yong Son
R&D Center, 4DHomeNet, Inc.

Sang-Wook Kim
Division of Computer, Information and Communication Engineering, Kangwon National University

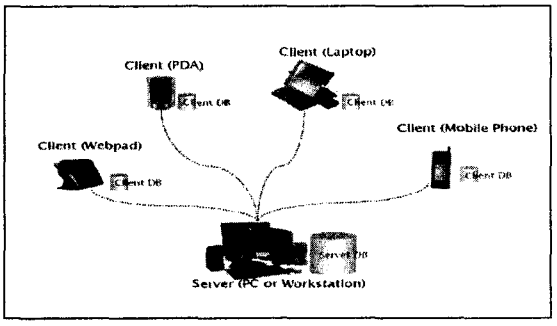
요 약

임베디드 DBMS는 일반적으로 고성능, 대용량의 서버 DBMS로부터 다운로드 받은 데이터를 관리한다. 이와 같은 임베디드 환경에서는 대부분의 경우 클라이언트와 서버가 접속되지 않은 상태에서 데이터 변경이 이루어진다. 데이터 동기화란 서버와 클라이언트에서 비접속 기간동안 각각 발생한 변경 내용들을 서로 비교, 교환하여 데이터 불일치성을 해결하는 작업을 말한다. 다양한 이기종 DBMS가 사용되는 임베디드 DBMS 환경에서는 현재 DBMS 벤더들마다 내부적으로 자체 개발한 동기화 방식을 이용하기 때문에 이기종 DBMS간의 동기화에는 추가적인 부담이 필요하다. 본 논문에서는 사용되는 DBMS에 독립적인 데이터 동기화 방법을 제안한다. 이 방법을 이용하면 동기화가 DBMS의 응용 프로그램처럼 동작하도록 할 수 있기 때문에 DBMS의 내부 구조에 변화를 주지 않고 동기화 서버를 구현할 수 있다. 이러한 DBMS에 독립적인 특징은 임베디드 DBMS와 동기화 서버를 이용한 응용 프로그램이 유연성과 상호 운용성을 가질 수 있도록 해 준다.

1. 서론[1]

하드웨어 기술과 무선 인터넷 기술의 발전에 따라 PDA(personal digital assistant), 스마트 폰(smart phone)과 같은 유·무선 소형 이동 단말기(small mobile device)가 널리 퍼지는 포스트 PC 시대가 도래하고 있다. 사용자들은 장소에 관계없이 이동 단말기를 사용하여 다양한 데이터를 보다 편리하고 효율적으로 공유하기 원한다. 이와 같은 데이터 서비스 요구들을 만족시키기 위하여 소형 이동 단말기에 적합한 임베디드 DBMS(embedded DBMS)[1][2]가 필요하게 되었다. 사용자들은 이동 단말기에 탑재된 임베디드 DBMS를 이용하여 필요한 정보를 효과적으로 관리할 수 있다. 현재 포디홈네트(주)에서는 강원대학교 데이타 및 지식공학 연구실과 더불어 정보통신부 선도기술개발 사업의 일환으로 인터넷 정보가전용 내장형 DBMS를 개발 중에 있다. 본 논문의 목적은 개발 중 획득한 일부 기술을 동일한 분야의 학자 및 개발자들과 공유하기 위한 것이다.

임베디드 DBMS 환경에서는 이동 단말기들에 저장된 정보를 공유하기 위하여 일반적으로 클라이언트-서버 구조를 이용한다[3][4][5][6]. 클라이언트들은 대부분의 시간 동안 비접속 상태로 있다가 사용자가 원하거나 정해진 시점이 되면 유·무선 통신을 통해 서버와 접속된다. 따라서 클라이언트들은 일반적으로 서버의 데이터베이스에 있는 데이터 중 필요한 일부를 자신들의 로컬 데이터베이스에 다운로드 받아 사용한다. 그림 1은 클라이언트-서버 구조를 기반으로 하는 전형적인 임베디드 DBMS 환경을 나타낸 것이다.



[그림 1] 클라이언트-서버 구조를 기반으로 하는 임베디드 DBMS 환경

클라이언트-서버 구조에서 양측의 데이터베이스는 접속되지 않은 상태에서 각각 변경될 수 있으므로 일시적인 불일치성(temporary inconsistency)이 필연적으로 발생하게 된다. 데이터 동기화(data synchronization)란 중복된 데이터 사이에서 발생한 불일치성을 해결하는 수단이며[7], 이는 이동형 임베디드 DBMS를 위한 핵심적인 기능이다. 데이터 동기화의 기본적인 전략은 서버

[1] 본 연구는 대한민국 정보통신부 정보통신연구진흥원의 2000~2002년 선도기술개발 제 4차 사업의 일환인 인터넷 정보가전용 내장형 DBMS 개발과제(과제번호 : 2002-S-12)의 지원으로 수행되었습니다.

DBMS와 클라이언트 DBMS에서 각각 수행된 변경을 탐지하여 변경된 내용들을 서로 상대측 데이터베이스에 적용하는 것이다. 임베디드 DBMS 환경에서는 같은 데이터를 서버와 클라이언트가 서로 다르게 변경하는 경우가 발생하는데 이것을 충돌(conflict)이라 한다[2]. 충돌을 탐지하고 해결하는 기능은 데이터 동기화의 필수적인 기능이다.

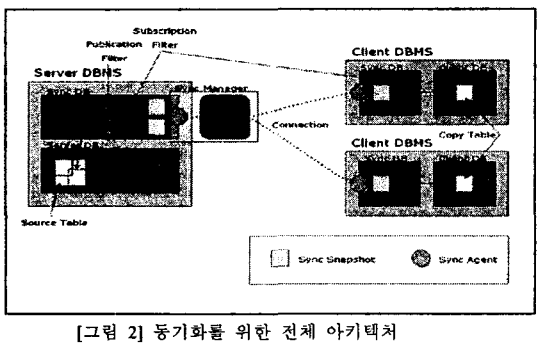
동기화 관리자를 개발하는 데 있어 가장 중요한 요구 사항의 하나는 임베디드 DBMS 환경에서 사용하는 다양한 이기종 DBMS와 유연하게 결합하여 사용될 수 있어야 한다는 점이다. 임베디드 DBMS 환경에서는 탑재되는 플랫폼, 사용 가능한 리소스, 요구되는 성능, 서비스 용도, 가격 등의 기준에 따라 각각 선정된 다수의 이기종 임베디드 DBMS가 클라이언트 내에 탑재되어 사용된다[1]. 마찬가지로 서버 내에서 사용되는 고성능 서버 DBMS도 사용자의 여건에 따라서 여러 종류가 사용될 수 있다. 다수의 이기종 DBMS가 혼재하는 환경에서 추가적인 부담 없이 동기화를 수행하기 위해서는 DBMS에 독립적으로 수행되는 동기화 관리자의 개발이 필수적이다.

여러 상용 임베디드 DBMS에서 지원하고 있는 동기화 기법들은 참고 문헌 [2], [7] 등에서 개략적으로 소개되고 있다. 이러한 동기화 기법들에서는 일반적으로 데이터 동기화에 필요한 레코드의 변경 정보를 DBMS 내부에서 자체적으로 저장한다. 이를 위하여 변경 이전 값(before-image)[3], 타임스탬프(timestamp)[5], 트랜잭션 로그(transaction log)[6] 등이 이용된다. 이러한 기존의 방법에서는 레코드 변경시 타임스탬프를 변경하거나, 로그를 기록하거나, 이전 값을 저장하는 등의 기능이 DBMS에 내부적으로 구현되어 있다. 즉, 개발하는 동기화 기능이 특정 DBMS에 종속되며 이 결과, 응용 환경에서 사용하는 DBMS가 바뀌는 경우 기존에 개발된 동기화 기능을 그대로 사용할 수 없다는 문제점이 있다.

본 논문에서는 다양한 이기종 임베디드 DBMS들이 혼재하는 환경에서 발생하는 이러한 심각한 문제를 해결하기 위하여 DBMS에 독립적인 동기화 방법을 제안한다. 제안된 방법에서는 최근 동기화 시의 동기화 대상 테이블의 복사본을 별도의 데이터베이스에 저장함으로써 이후에 변경되는 레코드에 대한 정보를 파악한다. 제안된 동기화 방법의 가장 큰 특성은 어떠한 DBMS 혹은 임베디드 DBMS가 사용되는 경우에도 제안된 동기화 기법을 그대로 적용할 수 있다는 것이다.

2. 동기화를 위한 전체 아키텍처

본 장에서는 본 논문에서 제안하는 임베디드 DBMS 환경에서의 DBMS 독립적인 동기화 기법을 위한 전체 아키텍처를 그림 2와 같이 제시한다.



[그림 2] 동기화를 위한 전체 아키텍처

동기화는 하나의 서버 DBMS와 다수의 클라이언트 DBMS들을 대상으로 한다. 동기화를 위해 동기화 서버와 동기화 에이전트가 존재한다. 동기화 서버는 서버 DBMS와 클라이언트 DBMS를 연결하고, 변경된 데이터를 받아서 충돌을 탐지하고 해결한 후, 상대편 DBMS의 데이터베이스에 그 변경을 반영하는 역할을 한다. 동기화 에이전트는 서버 DBMS와 클라이언트 DBMS에 각각 존재하며, 동기화 대상이 되는 테이블을 검색하여 변경을 탐지하고 동기화에 필요한 정보인 연결 정보, 인증 정보, 레코드 필터링 조건 등을 관리한다. 동기화 서버와 에이전트는 자신들의 역할을 수행하기 위해 필요한 데이터를 저장하기 위해 별도의 데이터베이스를 이용할 수 있는데 이를 동기화 데이터베이스(sync DB)라고 정의한다.

원본 테이블(source table)은 서버 측에 존재하는 동기화의 대상 테이블을 의미하고, 사본 테이블(copy table)은 클라이언트 측에 존재하는 동기화의 대상 테이블을 의미한다. 동기화가 수행되기 위해서는 우선 서버 DBMS의 원본 테이블이 지정되어야 한다. 일반적으로 동기화는 발행-인용(publish-subscribe) 방식으로 수행되는데, 서버의 원본 테이블에서 발행(publish)할 영역에 대한 조건과 함께 클라이언트의 사본 테이블에서 인용(subscribe)할 영역에 대한 조건을 설정한다. 이 두 조건을 동시에 만족하는 내용이 최종 동기화 대상이 된다. 이와 같이 조건을 설정하여 동기화에 필요한 영역을 지정하는 것을 필터링(filtering)이라고 한다[4][5].

서버와 클라이언트 각각의 동기화 에이전트는 원본 테이블과 사본 테이블의 최근 동기화 시의 내용을 별도로 저장해 둔다. 별도로 저장된 최근 동기화 시의 테이블 내용을 본 논문에서는 스냅샷(snapshot)이라고 정의한다. 동기화 명령이 내려지면 우선 변경된 레코드를 탐지해야 하는데, 이 때 각 DBMS에서 따로 레코드의 변경을 기록해 두지 않고도 스냅샷과의 비교를 통해 변경된 레코드를 찾아낼 수 있다.[2] 스냅샷을 이용한 변경의 탐지가 바로 데이터 동기화가 DBMS에 독립적으로 동작할 수 있도록 하는 핵심적인 개념이다.

3. 동기화를 위한 전략

본 장에서는 본 논문에서 제안하는 임베디드 DBMS 환경에서의 DBMS 독립적인 동기화 기법을 위한 전략을 제시한다.

클라이언트 측의 동기화 에이전트는 필터링 조건과 함께 서버 데이터베이스로의 연결 정보인 네트워크 주소, 데이터베이스 이름 및 인증 정보 등을 함께 관리한다. 이들 정보는 크지 않으므로 동기화가 실행될 때 클라이언트 자신의 연결 정보와 함께 동기화 서버로 전송된다. 물론 동기화 서버가 클라이언트마다 필터링 조건 정보와 연결 정보를 직접 관리할 수도 있지만, 동기화 서버의 부담을 줄이기 위해 클라이언트가 늘어남에 따라 가변적으로 변하는 이러한 정보들을 클라이언트 각자가 저장하는 것이 더 합리적이다.

필터링을 통해 원하는 데이터가 서버에서 클라이언트로 복제(replication)되면, 복제된 당시의 스냅샷이 동기화 에이전트가 관리하는 데이터베이스에 저장된다. 스냅샷은 서버와 클라이언트에서 동기화의 대상이 되는 원본/사본 테이블과 대응되어 최근 동기화

[2] 참고 문헌 [7]에서 Oracle의 동기화 모듈인 iConnect가 사전 이미지(before-image) 테이블을 이용하여 변경을 탐지한다는 것에 대해 언급하고 있다. iConnect의 변경 탐지 방식은 이전 동기화 시의 스냅샷 테이블을 이용하는 본 논문에서 제시한 변경 탐지 방법과 비슷하다. 그러나 iConnect의 방식은 DBMS(Oracle Lite) 내부에서 동기화를 위해 지원하는 기능으로, 본 논문에서 주장하는 바와 같이 DBMS와 동기화 모듈의 독립성을 위한 동기화 모듈의 기능이 아니라는 점에서 다소 차이가 있다.

이후에 발생한 변경을 탐지하는 데 이용된다. 동기화 서버를 통해 동기화 명령이 서버 데이터베이스와 클라이언트 데이터베이스를 관리하는 동기화 에이전트에 내려지면, 각각의 에이전트는 최근 동기화 된 시점부터 당시까지 원본/사본 테이블에 어떤 레코드가 삽입, 갱신, 삭제되었는지를 스냅샷과의 비교를 통해 탐색한다. 스냅샷에는 존재하지 않고 원본/사본 테이블에만 존재하는 레코드는 새로 삽입된 레코드이고, 스냅샷에만 존재하고 원본/사본 테이블에는 존재하지 않는 레코드는 삭제된 레코드이며, 스냅샷과 원본/사본 테이블 양측에 존재하면서 그 내용이 달라진 레코드는 갱신된 레코드이다. 물론 양측에 존재하고 내용이 달라지지 않은 레코드는 변경이 일어나지 않은 레코드이다.

삽입된 레코드, 갱신된 레코드, 삭제된 레코드는 동기화 에이전트에서 검색되어 각각 임시 테이블에 저장된다. 즉, 삽입 테이블, 갱신 테이블, 삭제 테이블이 만들어진다. 이들을 통칭하여 변경 테이블이라 하는데, 변경 테이블은 동기화 에이전트가 관리하는 동기화 데이터베이스에 저장된다. 참고로 삭제 테이블의 경우에는 기본 키를 제외한 필드들은 의미가 없으므로 기본 키만으로 테이블을 구성할 수 있다.

서버와 클라이언트 양측에서 변경 탐지가 끝나면 변경된 레코드들로 이루어진 변경 테이블들이 동기화 서버로 전송된다. 동기화 서버에서는 충돌을 탐지하는데, 충돌이란 양쪽에서 동일한 레코드 ID(또는 기본 키)를 가진 레코드에 대해 서로 다른 변경을 했을 경우에 발생한다. 충돌은 네 가지 종류로 나눌 수 있다. 양쪽에서 서로 다른 값으로 갱신한 경우를 갱신-갱신 충돌(update-update conflict)이라 하고, 서버에서 갱신된 레코드를 클라이언트에서는 삭제한 경우를 갱신-삭제 충돌(update-delete conflict)이라 하고, 서버에서 삭제된 레코드를 클라이언트에서는 갱신한 경우를 삭제-갱신 충돌(delete-update conflict)이라 하고, 양쪽에서 기본 키가 같은 서로 다른 레코드를 삽입한 경우를 삽입-삽입 충돌(insert-insert conflict)이라 한다[2][7].

동기화 서버에서 충돌을 해결하기 위해 여러 가지 규칙을 설정해야 한다. 충돌이 일어난 레코드에 대하여 양측의 변경을 동시에 반영하기가 불가능하므로 클라이언트 측의 변경을 무시하고 서버 측의 변경을 따르거나 서버 측의 변경을 무시하고 클라이언트 측의 변경을 따르는 것 중에 하나를 보통 선택한다. 그러나 특별한 경우 양쪽에서 일어난 변경이 동시에 반영되는 경우도 발생한다. 숫자 데이터에서 양쪽의 증감치를 합하여 갱신하기도 하고, 문자 데이터의 경우 두 데이터를 결합하여 갱신하기도 하는 특별한 옵션을 선택할 수도 있다[4][8]. 충돌이 발생한 모든 레코드들에 대하여 동기화를 요청한 사용자에게 어떤 변경을 적용할 것인지 선택하도록 하는 방법도 가능하다. 이와 같이 충돌이 해결된 변경 레코드는 각각 상대측 데이터베이스에 적용된다. 이상 없이 적용이 완료되면 테이블의 내용과 동일하게 스냅샷을 갱신하고 동기화 작업을 마친다.

동기화 작업에 의해 삽입, 갱신, 삭제되는 동작들은 하나의 트랜잭션으로 간주하여 실행되어야 한다. 따라서 동기화 작업 진행 중에 시스템 장애나 오류가 생기는 경우에는 서버나 클라이언트에 동기화의 대상이 되는 데이터 중 일부만 반영되고 일부는 반영되지 않기 때문에 데이터의 일관성에 문제가 발생하게 된다. 이러한 문제가 발생하는 경우에는 트랜잭션 롤백을 통해 동기화 이전 상태로 복구시킴으로써 데이터베이스를 안전하게 보호할 수 있다. 시스템 장애나 오류가 해결되면 사용자는 다시 동기화 명령을 내려 데이터를 동기화 시킬 수 있다.

본 논문에서 제안한 동기화 기법의 가장 큰 장점은 동기화 기능이 DBMS의 어플리케이션으로 동작하므로 DBMS의 수행 방식에 영향을 주지 않고 독립적으로 수행될 수 있다는 것이다. 모든 DBMS가 공통적으로 지원하고 있는 API인 SQL, JDBC, ODBC등을 이용함으로써 사용자가 지정한 데이터베이스와 테이블 구조에 수정을 가하지 않고 이기종 DBMS와 연동이 가능하도록 하였다. 따라서 본 논문에서 제시된 DBMS 독립적인 기법을 이용하면 유연성(flexibility)과 상호 운용성(interoperability)을 가진 동기화 시스템을 제작할 수 있다.

4. 결론

임베디드 DBMS 환경에서는 다양한 이기종 DBMS가 사용된다. 그러나 현재 DBMS 벤더들마다 별도의 동기화 방식을 이용하기 때문에 이기종 DBMS간의 동기화에는 추가적인 부담이 필요하다. 본 논문에서는 이러한 추가적인 부담 없이 동기화를 수행하도록 하기 위하여 DBMS 독립적으로 수행되는 동기화 방법을 제안하였다. 본 논문에서 제안한 방법은 동기화 당시의 테이블의 복사본, 즉 스냅샷을 동기화를 위해 별도로 저장하는 것이다. 스냅샷을 이용하면 동기화가 DBMS의 응용 프로그램처럼 동작하도록 할 수 있기 때문에 DBMS의 내부 구조에 변화를 주지 않고 동기화 서버를 구현할 수 있다. 스냅샷을 이용하는 방법의 단점인 클라이언트 측의 저장 공간 부하는 장점인 DBMS에 독립성에 비하여 감수할 수 있는 수준으로 판단된다. DBMS에 독립적인 특징은 임베디드 DBMS와 동기화 서버를 이용한 응용 프로그램이 유연성과 상호 운용성을 가질 수 있도록 해 준다.

제안된 방법은 클라이언트가 저장공간에 대해 심각한 제약이 없는 경우 DBMS에 대한 독립성을 잘 보장하고 있다. 향후 연구 방향으로는 DBMS 독립성을 보장하면서 저장 공간 부하를 줄여 나가는 방법을 고안하는 것을 고려하고 있다.

5. 참고 문헌

[1] Michael A. Olson, "Selecting and Implementing an Embedded Database System," IEEE Computer Magazine, pp 27-34, September 2000.

[2] 김상욱, 오세봉, 김만순, 박정일, 상용 내장형 DBMS 환경을 위한 Synchronization 기법에 관한 연구, 정보통신 선도기반 기술개발사업 인터넷 정보가전용 임베디드 DBMS (과제번호 : 2000-S-169) 기술보고서, 2001년 10월.

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

[4] Pointbase, Pointbase UniSync Developer's Guide, Version 4.2, Whitepaper

[5] IBM, DB2 Sync Server Administration Guide 7.2.1, Whitepaper

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

[7] 이상윤, 박순영, 이미영, 김명준, "이동 DBMS의 데이터 동기화 기술 분석," 데이터베이스 연구회지, 17권 3호, pp. 29-41, 2001년 9월.

[8] Jim Gray, Pat Helland, Patrik O'Neil, Dennis Shasha, "The Dangers of Replication and a Solution," ACM SIGMOD 1996.

[9] Sixto Ortiz, Jr., "Embedded Databases Come out of Hiding," IEEE Computer Magazine, pp 16-19, March 2000.

[10] 정병대, 장우석, 강성일, 이진호, "댁내 데이터의 상호운용성에 관한 연구," 정보과학회 춘계 학술대회 논문집, pp 88-90, 2001년 4월.

[11] 노형준, 장우석, 정병대, 이홍규, 손성용, 이진호, "정보가전용 임베디드 DBMS에서의 다중 버전을 이용한 동시성 제어," 정보과학회 추계 학술대회 논문집, pp 121-123, 2001년 10월.

[12] 이상윤, "동기화 표준 SyncML의 표준화 동향," ETRI IT정보센터 주간기술동향, pp 14-28, 2002년 3월.