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

# 정보가전 환경에서 중복된 데이터의 효과적인 동기화

강영호, 장우석, 노형준, 정병대, 손성용
포디홈네트 주식회사

김상욱
강원대학교 컴퓨터정보통신공학부

# Effective Synchronization of Replicated Data in Information Appliance Environment

Yeong Ho Kang, Wooseog Jang, Gary Noh, Byung Dae Jung, Sung-Yong Son
4DHomeNet, Inc.

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

## 요 약

정보가전 환경은 정보가전들 간의 홈 네트워크를 기반으로 한 상호 정보 교환을 통하여 사용자에게 보다 진보적인 홈 서비스를 제공한다. 정보가전 환경에서는 서로 다른 두 정보가전 내에 동일한 데이터가 중복되는 상황이 발생 가능하므로, 이러한 중복된 데이터의 일관성을 유지시켜 주기 위한 동기화 기능의 지원이 요구된다. 본 논문에서는 정보가전 환경에서 중복된 데이터의 일관성을 유지 시키기 위한 효과적인 동기화 기법을 제안하였다. 제안하는 동기화 기법은 네트워크 전송량과 데이터 저장공간을 최소화함으로써 통합 데이터 관리 아키텍춰에서 좋은 성능을 발휘할 수 있다는 것이 큰 장점이다. 또한, 이 동기화 기법은 중앙집중방식의 통합 데이터 관리가 불가능한 현재의 환경과 이것이 가능하게 될 미래의 환경에 모두 적용할 수 있다.

## 1. 서론[1]

정보가전 환경이란 정보가전들이 홈 네트워크를 통해 상호 연결되며, 또한 외부 인터넷과 홈 게이트웨이(home gateway)[Etr01]를 통해 연결되는 환경을 의미한다. 정보가전 환경에서는 정보가전들 간의 상호 정보 교환이 가능하므로 보다 진보적인 홈 서비스를 제공하는 것이 가능하다[Par01b]. 정보가전들간의 효과적인 정보의 교환 및 처리를 위하여 각 정보가전은 데이터 저장 모듈을 포함해야 한다.

임베디드 DBMS(Embedded DBMS)란 정보가전과 같은 기기에 탑재되어, 데이터의 저장, 검색, 변경을 수행하는 소형 DBMS 이다[Ols00][Kim01]. 홈 네트워크의 대중화가 안된 현재, 대부분의 정보가전에서는 자체적으로 개발된 데이터 관리 모듈을 탑재하고 있으나, 보다 편리한 응용의 개발을 위하여 조만간 임베디드 DBMS 를 저장 모듈로서 사용하는 것이 일반화될 전망이다.

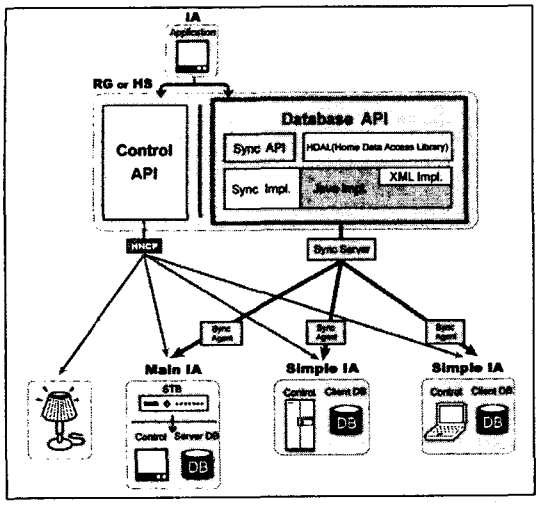
정보가전 환경에서는 서로 다른 두 정보가전내에 동일한 데이터가 중복되는 상황이 발생 가능하다. 이러한 중복된 데이터는 양측에서 모두 변경될 수 있다. 이러한 변경을 양쪽 모두에 일관되게 반영하지 않는 경우, 데이터의 비일관성(data inconsistency) 문제로 인하여 응용의 올바른 수행을 보장할 수 없다. 동기화(synchronization)란 이러한 중복된 데이터의 일관성(consistency)을 유지시켜 주기 위한 핵심 기능이다.

본 논문에서는 정보가전 환경에서 중복된 데이터를 위한 효과적인 동기화 기법을 제안한다. 제안된 동기화 기법은 참고 문헌 [Noh02] 에서 제시한 " 홈 네트워크 환경을 위한 통합 데이터 관리 아키텍춰" 를 기반으로 한다. 이 아키텍춰는 현재 과도기적인 정보가전 환경의 특성을 고려하여 중앙에 데이터 센터를 구축하는 것이 가능한 경우와 그것이 불가능한 경우를 모두 수용하는 것이 큰 특징이다. 따라서 데이터 센터의 구축이 불가능한 환경에서도 데이터를 효과적으로 관리할 수 있으며, 이 환경에 데이터 센터가 새롭게 도입되더라도 이를 쉽게 수용할 수 있다[Noh02]. 본 논문에서 제안하는 동기화 기법은 통합 데이터 관리 아키텍춰에서 성능에 큰 영향을 미치는 네트워크 전송량과 데이터 저장공간을 최소화하는데 그 설계의 초점을 맞추었다.

## 2. 정보가전 환경을 위한 통합 데이터 관리 아키텍춰

댁내 데이터를 일관성있게 유지하는 가장 쉬운 방법은 데이터 센터를 만드는 것이다[Noh02]. 여기서 데이터 센터란 댁내의 모든 데이터를 보관하기 위한 대용량의 저장소를 의미한다. 데이터 센터는 댁내 데이터 저장소를 일원화함으로써 모든 데이터를 쉽게 유지하도록 한다. 그러나 데이터 센터가 존재하지 않는 정보가전 환경이 발생할 수 있으며, 전체 정보가전 환경이 데이터 센터에 지나치게 종속된다는 문제점을 갖는다. 따라서 참고 문헌 [Noh02] 에서는 댁내에 데이터 저장소가 단일화된 경우와 다양하게 분산된 경우를 모두 수용하기 위하여 분산 DBMS 들을 바탕으로 데이터를 관리하고, 이를 통합하여 중앙 집중적인 뷰를 제공하는 데이터 관리 아키텍춰를 제안하였다([그림 1] 참고).

[1] 본 논문은 대한민국 정보통신부 정보통신연구진흥원의 2000 년 선도기술개발 제 4 차 사업의 일환인 "인터넷 정보가전용 내장형 DBMS (과제번호 2002-S-12)" 개발과제의 지원으로 작성된 것입니다.




[그림 1] 정보가전 환경을 위한 통합 데이터 관리 아키텍춰.

댁내 환경은 크게 제어성 데이터와 정보성 데이터의 흐름이 있다. 이들 중 정보성 데이터를 활용한 환경은 주로 정보가전이나 댁내 이동 매체의 지능화된 응용 프로그램들이 활용하는 환경이다. 이는 제어를 판단하는 기준을 확장하게 되므로, 홈 오토메이션 환경에도 도움이 된다. 댁내 데이터를 효과적으로 관리하기 위한 시스템은 홈 서버(HomeServer:HS)[Par01]나 레지던셜 게이트웨이 Residential Gateway(RG)[Woo01]에 탑재되어 기존의 제어 관리를 위한 시스템과 연계한 활용도 가능하다.

HDAL(Home Data Access Library)는 데이터를 활용한 서비스 개발자를 위하여 구상된 API 이다. Control API 는 제어 관련 서비스를, Database API 가 데이터 관련 서비스를 제공한다. 정보가전은 데이터의 원본을 관리하는 Main IA 와 원본의 일부를 복사한 데이터를 관리하는 Simple IA 로 분류된다. 동기화 에이전트(Sync Agent)는 동기화 서버(Sync Server)모듈과 서로 통신하며 데이터 동기화를 지원한다. 개발자에게 데이터 액세스와 동기화를 지원하기 위해 Sync API 와 HDAL API 를 제공하며, Sync Impl(ementation)과 Java Impl(ementation)은 이 API 를 실제 구현한 것을 의미한다.

이러한 시스템 아키텍춰는 각 정보가전내에 이기종 DBMS 가 공존하는 일반적인 환경을 대상으로 하며, 하나의 데이터 센터를 갖춘 환경에도 적용할 수 있다. 또한, 복수의 이기종 DBMS 를 관리하고, 사용자의 데이터 접근을 단일화하는 통합 데이터베이스 관리 모듈을 탑재함으로써, 홈 네트워크 서비스 개발자가 데이터 저장의 물리적 위치를 알지 못하더라도 쉽게 응용 프로그램 개발을 진행할 수 있도록 지원한다.

## 3. 기존 기법의 문제점

현재 홈 네트워크의 표준은 정해지지 않았으며, 이더넷(Ethernet), 홈피엔에이(HomePNA), 전력선(PowerLine) 등의 유선기술과 적외선(IrDA), 주파수무선통신(RF) 등의 무선기술이 동시에 거론되고 있다[Par01a][Par01b][Etr01]. 이러한 홈 네트워크의 특성은 외부의 인터넷이나 기업의 인트라넷과는 달리 고속의 데이터 전송을 보장할 수 없는 것이 큰 특징이다. 또한, 정보가전 환경을 위한 홈 서버와 정보가전은 각각 기업형 서버와 일반 PC 와는 달리 그 사양이 낮으므로 과다한 네트워크 처리 오버헤드로 인한 성능문제가 야기될 수 있다. 뿐만 아니라, 정보가전은 일반 PC 와는 달리 매우 작은 저장공간을 가진다. 따라서 이러한 정보가전 환경에서의 동기화 기법은 이러한 네트워크 전송량과 저장공간을 최소화할 수 있도록 설계 되어야 한다.

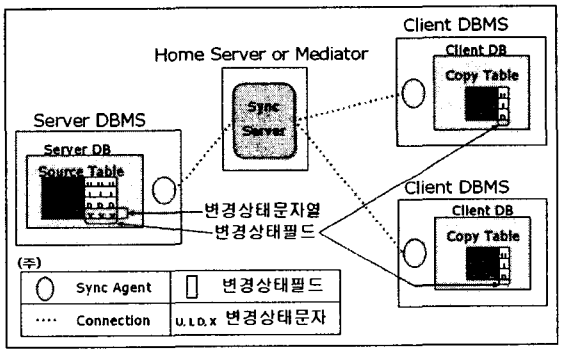
기존의 이동 환경을 대상으로 상용 임베디드 DBMS 에서 사용하는 동기화는 크게 타임스탬프(timestamp)를 이용하는 방식과 변경 전의 값(old value)을 이용하는 방식으로 분류된다[Kim01][Lee01]. 본 논문에서는 이 두 방식의 영문 이니셜을 따서 간략히 TS 방식과 OV 방식으로 칭한다.

TS 방식은 서버와 클라이언트에 존재하는 동기화 대상이 되는 각 레코드에 타임스탬프 필드를 추가하는 것이다. 타임스탬프는 서버에 존재하는 레코드의 최종 변경 시점을 표현하는 값이다. 클라이언트는 서버로부터 데이터를 다운로드 할 때, 각 레코드와 대응되는 타임스탬프 값을 함께 다운로드 받는다. 이 값은 클라이언트의 임베디드 DBMS 내에서 유지되다가 동기화 시에 충돌 감지를 위하여 사용된다. OV 방식은 클라이언트가 서버로부터 다운로드 한 변경 전의 각 레코드 값을 이용한다. 클라이언트에서 변경이 일어난 후에도 이 값은 클라이언트의 임베디드 DBMS 내에서 그대로 유지되다가, 동기화 시에 충돌 감지를 위하여 사용된다.

OV 방식은 두 가지 버전의 값을 모두 유지해야 하므로 정보가전의 저장공간에 오버헤드가 크고, TS 방식은 타임스탬프 전송에 따른 네트워크 부하가 크다[Lee01][Kim01]. 따라서 이 두 가지 방식 모두 전술한 정보가전 환경에서 동기화 기법이 갖추어야 할 두 가지 특징을 만족하지 못한다.

## 4. 제안하는 기법

본 논문에서는 네트워크 전송량과 저장공간을 최소화하는 CSIC (Change Status Information Character) 기반 동기화 기법을 제안한다.




[그림 2] CSIC 기반 동기화 기법을 위한 아키텍춰

[그림 2]는 CSIC 기반 동기화 기법을 위한 아키텍춰를 나타낸 것이다. 여기서, Sync Server 는 동기화할 데이터의 변경탐지와 충돌발생시 충돌해결을 담당하는 모듈이다. 이 모듈은 정보가전 환경 내에 홈 서버가 존재하면 홈 서버에 탑재되고, 홈 서버가 존재하지 않으면 홈 네트워크를 제어하는 컨트롤박스 같은 미디에이터에 탑재된다. Sync Server 를 지원하는 Sync Agent 는 임베디드 DBMS 가 존재하는 정보가전에 탑재되어 데이터와 변경상태문자 및 연결정보 등의 동기화 정보를 송/수신하는 것을 담당한다. 서버 DBMS 는 중복된 데이터 중 원본 데이터(Source Table)를 가지고 있는 정보가전내에 탑재된 DBMS 이고, 클라이언트 DBMS 는 원본 데이터로부터 복제한 데이터를(Copy Table) 가지는 정보가전내에 탑재된 DBMS 이다.

레코드의 변경 상태를 기록하고 있는 문자를 변경상태문자라 정의하며, 변경상태문자는 N (변경 없음), U(갱신됨), I(삽입됨), D(삭제됨), X(동기화 되지 않거나 클라이언트 DB 에서 삭제됨: 서버 DB 에만 존재) 등의 값을 갖는다. 현재 동기화 되고 있는 클라이언트 DBMS 에 해당하는 변경상태문자(지역 변경일 경우 없음)를 CSC(Currently Synchronizing Character 라고 정의하고, 현재 동기화 되고 있지 않은 클라이언트 DBMS 에 해당하는 변경상태 문자(지역 변경일 경우 모든 변경상태 문자가 해당됨)를 Non-CSC 라고 정의한다.

하나의 변경상태문자는 각 클라이언트와 일대일 대응된다. 따라서 서버 DBMS 내의 각 레코드는 이 레코드를 복제한 " 클라이언트들의 수+1" 만큼의 변경상태문자들을 가지게 된다. 각 레코드에

서 이 변경상태문자들의 열을 포함하는 필드를 변경상태필드라고 정의한다.

### 4.1. 동기화 초기 설정

서버 DBMS 와 클라이언트 DBMS 는 각각 변경 상태 필드를 가진다. 클라이언트 DBMS 의 변경상태필드는 1-Charatcer 로 나타내며, 서버 DBMS 측에서는 해당 데이터가 복제된 클라이언트 DBMS 의 개수만큼의 길이를 가지는 변경상태문자열을 가진다. 클라이언트 DBMS 측의 변경상태필드는 서버 DBMS 측의 변경상태문자열 중에 그와 대응되는 변경상태필드를 가진다. 레코드가 처음 복제되면 해당 레코드의 변경상태문자는 초기값으로' N' 을 가지게 된다. (예 1 참고)

예 1) 초기 상태

— 다음과 같은 서버 DBMS 와 2 개의 클라이언트 DBMS 가 있다고 가정하자.

서버 DB (초기 상태)

| Primary Key | Column | Change Status |
|---|---|---|
| 1 | Alpha | NNN |
| 2 | Bravo | NNN |
| 3 | Charlie | NNN |

클라이언트 DB1, DB2 (초기 상태)

| PK | Column | CS |
|---|---|---|
| 1 | Alpha | N |
| 2 | Bravo | N |
| 3 | Charlie | N |

| PK | Column | CS |
|---|---|---|
| 1 | Alpha | N |
| 2 | Bravo | N |
| 3 | Charlie | N |

### 4.2. 변경의 발생 및 탐지

클라이언트 DBMS 와 서버 DBMS 내에서 레코드의 변경이 발생하면, 해당 레코드와 대응되는 변경상태필드내의 변경상태문자는 제 4.1 절에서 설명한 바와 같이 변경 연산의 종류에 따라 N (변경 없음), U(갱신됨), I(삽입됨), D(삭제됨), X(동기화 되지 않거나 클라이언트 DB 에서 삭제됨: 서버 DB 에만 존재) 등의 값으로 갱신된다. 또한, Sync Server 는 클라이언트 DBMS 측 또는 서버 DBMS 측의 변경상태필드의 변경상태문자를 참조하여 해당 레코드가 변경되었음을 탐지한다. (예 2 참고)

예 2) 검색 및 변경 시 변경 상태 필드 설정 방법

— 우선 서버 DB 에서 자체적인 변경이 일어났다고 하자. 레코드 1 은 Update 되고, 레코드 2 는 Delete 되고, 레코드 4 가 Insert 되었다고 하자. 이 경우 다음과 같이 테이블이 변경된다.

서버 DB (*자체 변경*)

| Primary Key | Column | Change Status |
|---|---|---|
| 1 | *Foxtrot* | *UUU* |
| 2 | Bravo | *DDD* |
| 3 | Charlie | NNN |
| *4* | *Echo* | *III* |

클라이언트 DB1, DB2 (변경 없음)

| PK | Column | CS |
|---|---|---|
| 1 | Alpha | N |
| 2 | Bravo | N |
| 3 | Charlie | N |

| PK | Column | CS |
|---|---|---|
| 1 | Alpha | N |
| 2 | Bravo | N |
| 3 | Charlie | N |

### 4.3. 동기화 수행

동기화 명령이 내려지면, 서버 DBMS 와 클라이언트 DBMS 에서 탐지된 변경된 레코드들이 각각 동기화 서버로 전송된다. 동기화 서버에서는 이들 레코드들 간에 충돌(conflict)이 존재하는지의 여부를 검사한다. 충돌이란 복제된 동일한 레코드가 둘 이상의 DBMS 내에서 서로 다른 방식으로 변경된 상황을 의미한다. 충돌의 발생 여부를 판정하기 위하여 서버 DBMS 측에서 가져온 레코드의 CSC 와 클라이언트 DBMS 쪽에서 가져온 변경상태문자를 비교하여 다음 경우에 해당하는지 검사한다[Jan02].

- 갱신-갱신 충돌: 서버 DBMS 에서 갱신 ◻ 클라이언트 DBMS 에서 갱신
- 삽입-삽입 충돌: 서버 DBMS 에서 삽입 ◻ 클라이언트 DBMS 에서 삽입
- 갱신-삭제 충돌: 서버 DBMS 에서 갱신 ◻ 클라이언트 DBMS 에서 삭제
- 삭제-갱신 충돌: 서버 DBMS 에서 삭제 ◻ 클라이언트 DBMS 에서 갱신

충돌이 발견되면 충돌 해결 규칙에 따라 취소(Discard)할 레코드를 각각 정하여 취소하고, 충돌이 발생하지 않은 레코드들을 실제로 서버 DBMS 와 클라이언트 DBMS 로 전송한다. 충돌 해결 규칙에는 서버 DBMS 우선, 클라이언트 DBMS 우선이 있다[Jan02]. 서버 DBMS 우선 규칙은 동일한 키를 가진 레코드에 대해서 클라이언트 DBMS 와 서버 DBMS 에서 동시에 변경이 일어난 경우 서버 DBMS 의 변경을 우선 반영하는 규칙이다. 반면, 클라이언트 DBMS 우선 규칙은 동일한 키를 가진 레코드에 대해서 클라이언트 DBMS 와 서버 DBMS 에서 동시에 변경이 일어난 경우 클라이언트 측 DBMS 의 변경을 우선 반영하는 규칙이다.

클라이언트 DBMS 로 전송된 서버 DBMS 측 변경은 그대로 클라이언트 DBMS 측에 적용된다. 또한, 클라이언트 DBMS 에서는 변경을 적용한 후 변경상태문자가 D 로 설정된 레코드들을 삭제하고, 나머지 레코드들의 변경상태문자를 N 으로 재설정한 후, 종료한다. (예 3 참고)

예 3) 동기화 시 변경 상태 필드 설정 방법

— 이후 서버 DB 가 클라이언트 DB1 과 동기화를 시도하면 다음과 같이 변경된다.

서버 DB (*클라이언트 DB1 과 동기화*)

| Primary Key | Column | Change Status |
|---|---|---|
| 1 | Foxtrot | *M*UU |
| 2 | Bravo | *X*DD |
| 3 | Charlie | NNN |
| 4 | Echo | *M*I |

클라이언트 DB1(*서버 DB 와 동기화*), DV2 (변경 없음)

| PK | Column | CS |
|---|---|---|
| 1 | *Foxtrot* | *N* |
| *2* | *Bravo* | *N* |
| 3 | Charlie | N |
| *4* | *Echo* | *N* |

| PK | Column | CS |
|---|---|---|
| 1 | Alpha | N |
| 2 | Bravo | N |
| 3 | Charlie | N |

서버 DBMS 측으로 넘어온 클라이언트 DBMS 의 변경들 역시 서버 DBMS 측에 반영되는데, 이 경우는 위의 경우보다 좀더 복잡하다. 즉, 서버 DBMS 측 원본 레코드 내에 존재하는 변경상태필드의 변경상태문자들에 대한 설정 규칙이 필요하다. 한 클라이언트 DBMS 에서 일어난 변경이 다른 클라이언트 DBMS 에도 반영되도록 하기 위하여 Non-CSC 들도 적절히 설정되어야 한다. 각 변경상태문자의 설정을 위한 규칙은 다음과 같다.

i. Non-CSC 의 설정 규칙
(동기화가 일어날 때 또는 서버 DBMS 자체에서 변경 시)

- 동기화 경우:
  - 클라이언트 DBMS 우선일 경우에만 적용.
  - 서버 DBMS 우선일 경우에는 불변
- 서버 DBMS 에서 자체 변경하는 경우: 항상 적용

| Insert | Delete | Update |
|---|---|---|
| 없음->I | | |
| N -> U | N -> D | N -> U |
| I -> I | I -> X | I -> I |
| D -> U | D -> D | D -> U |
| U -> U | U -> D | U -> U |
| X -> I | X -> X | X -> X |

ii. CSC-변경상태문자의 설정 규칙

| 변경 종류 | 이전 문자 | 변경 후 문자 |
|---|---|---|
| Insert | 없음 | N |
| | I → 충돌 | N (Any Win) |
| Delete | N | X |
| | I | 일어날 수 없음 |
| | D | X |

| | | |
|---|---|---|
| | U → 충돌 | X (클라이언트 우선) |
| | | N (서버우선) |
| | X | X |
| Update | N | N |
| | I | 일어날 수 없음 |
| | D → 충돌 | N (클라이언트 우선) |
| | | X (서버우선) |
| | U → 충돌 | N (Any Win) |
| | X | X |

서버 DBMS 측의 변경상태문자를 설정할 때는 먼저 Non-CSC 를 설정하고, CSC 를 나중에 설정한다. 그 이유는 변경상태문자들이 모두 X 로 이루어진 레코드에 대한 두 가지 의미 때문이다. 하나는 처음부터 이 레코드는 동기화의 대상이 아니어서 어느 클라이언트 DBMS 에도 복제되어가지 않았다는 의미이고, 다른 하나는 동기화의 대상이지만 삭제된 후에 모든 클라이언트 DBMS 들에 대해 적용을 완료했다는 의미이다. 전자의 경우는 실제로 삭제하면 안되지만 후자의 경우는 실제로 삭제해야 한다.

이 경우를 위해 CSC 가 X 로 변경되기 전에 Non-CSC 들을 살펴보아야 한다. 만일 모든 Non-CSC 가 X 라면 이 레코드는 삭제된 후 동기화되는 것을 의미하므로 실제로 서버 DBMS 에서 삭제해야 한다. CSC 가 X 인 레코드는 처음부터 동기화의 대상이 되지 않기 때문에 서버 DBMS 에서 자체적으로 삭제하지 않는 한 삭제되지 않을 것이다. 실제로 서버 DBMS 에서 자체적으로 데이터를 읽을 때도 변경상태문자가 X 만으로 이루어진 레코드는 읽고, D 가 섞여 있는 레코드는 읽지 않아야 한다.

## 5. 논의 사항

본 안에서 제안하는 CSIC 기반 동기화 기법은 다음과 같은 장점을 갖는다.

첫째, 클라이언트 DBMS 와 연동되어야 하는 동기화 모듈을 간소화 하여 독립성을 높였다. 동기화를 담당하는 모듈을 동기화 서버와 동기화 에이전트로 분리시키고, 동기화의 주된 처리를 담당하는 동기화 서버를 미디에이터에 탑재하였다. 즉 어떤 클라이언트 DBMS 가 사용되는 상황이라도 간소화된 클라이언트 동기화 모듈만 연동하면 동기화가 가능하게 함으로써 클라이언트 DBMS 와의 독립성을 높였다.

둘째, [Noh01]에서 제안한 정보가전용 통합 데이터 관리 아키텍춰에서 네트워크 전송량을 극소화할 수 있다. 즉, 동기화 대상인 레코드들 중에서 변경상태문자를 이용한 변경의 검출 과정을 통하여 변경이 확인된 것들만을 전송함으로 네트워크 전송량을 크게 줄일 수 있다.

셋째, 정보가전에서 저장공간의 오버헤드를 극소화할 수 있다. 즉, 변경탐지문자를 사용함으로써 서버 DBMS 와 클라이언트 DBMS 에서 각 레코드마다 두 가지 버전의 값을 유지해야 하는 OV 방식의 저장공간 문제를 해결할 수 있다.

## 6. 결론

정보가전 환경에서는 정보가전들 간의 홈 네트워크를 통한 상호 정보 교환이 가능하므로 보다 진보적인 홈 서비스를 제공하는 것이 가능하다[Par01b]. 정보가전 환경에서는 서로 다른 두 정보가전 내에 동일한 데이터가 중복되는 상황이 발생 가능하다. 이러한 중복된 데이터는 양측에서 모두 변경될 수 있으며, 이러한 변경을 양쪽 모두에 일관되게 반영하지 않는 경우 데이터의 비일관성 문제로 인하여 응용의 올바른 수행을 보장할 수 없다. 동기화란 이러한 중복된 데이터의 일관성을 유지시켜 주기 위한 핵심 기능이다.

본 논문에서는 정보가전 환경에서 중복된 데이터를 위한 효과적인 동기화 기법을 제안하였다. 제안된 동기화 기법은 참고 문헌 [Noh02] 에서 제시한 " 홈 네트워크 환경을 위한 통합 데이터 관리 아키텍춰" 를 기반으로 한다. 이 아키텍춰는 중앙에 데이터 센터를 구축하는 것이 가능한 경우와 그것이 불가능한 경우를 모두 수용한다. 제안하는 동기화 기법은 네트워크 전송량과 데이터 저장 공간을 최소화함으로써 통합 데이터 관리 아키텍춰에서 좋은 성능을 발휘할 수 있다는 장점을 갖는다. 또한, 이 동기화 기법은 중앙집중방식의 통합 데이터 관리가 불가능한 현재의 환경과 이것이 가능하게 될 미래의 환경에 모두 적용할 수 있다.

참고문헌


[Etr01] 한국전자통신연구원, 40 대 품목 기술 / 시장 통합 요약보고서 (IT 전략품목 - 네트워크정보가전), pp. 282-290, 2001 년.

[Gra96] Jim Gray, Pat Helland, Patrik O'Neil, and Dennis Shasha, "The Dangers of Replication and a Solution," In Proc. Intl. Conf. On Management of Data, ACM SIGMOD, 1996.

[IBM01] IBM, DB2 Sync Server Administration Guide 7.2.1, A White Paper.

[Jan02] 장우석 외, "임베디드 DBMS 환경에서의 데이터 동기화: DBMS 독립적인 접근 방안," 한국멀티미디어 학회지, 제 20 권 제 7 호, 2002 년 7 월.

[Kim01] 김상욱, 오세봉, 김만순, 박정일, 상용 임베디드 DBMS 환경을 위한 *Synchronization* 기법에 관한 연구, 정보통신 선도기반 기술개발사업 인터넷 정보가전용 내장형 DBMS (과제번호: 2000-S-169) 기술보고서, 2001 년 10 월.

[Kim02] 김상욱 외, "임베디드 DBMS 환경에서의 동기화를 위한 프레임워크,"한국정보과학회지, 제 20 권, 제 7 호, 2002 년 7 월.

[Lee01] 이상윤, 박순영, 이미영, 김명준, "이동 DBMS 의 데이터 동기화 기술 분석," 데이터베이스 연구회지, 제 17 권, 제 3 호, pp. 29-41, 2001 년 9 월.

[Lee02] J. M. Lee et al., "HNCP (Home Network Control Protocol) for *Home Appliances*", *IEEE International Conference on Consumer* Electronics, pp. 312-313, June 2002.

[Noh02] 노형준 외, "홈 네트워크 환경에서 효과적인 데이터 관리를 위한 시스템 구조,"정보통신설비 논문지, 제 1 권, 제 1 호, 2002 년 8 월.

[Ols00] Michael A. Olson, "Selecting and Implementing an Embedded Database System," IEEE Computer, pp 27-34, Sept. 2000.

[Ora01] Oracle, Oracle 9i Lite Administration Guide 5.0.1, A White Paper.

[Par00] 박광로, 김재명, 김종원, 양재우, "홈 게이트웨이 기술", 정보통신학회지, 제 17 권, 제 11 호, pp 1593-1602, 2000 년 11 월.

[Par01a] 박용우, Home Networking 관련 유선 인터페이스 표준화 논의, KISDI IT FOCUS, 2001 년 2 월.

[Par01b] 박용우, 정보통신기기편: 홈네트워킹, 정보통신산업동향, pp. 293-318, 2001 년 10 월.

[Pos02] 포항공과대학 지능정보시스템연구실, 정보가전용 내장형 DBMS 응용 프로그램 라이브러리(HDAL) 표준안(제출중), 2002 년 8 월.

[Syb99] Sybase, Synchronization Technologies for Mobile and Embedded Computing, A White Paper.

[Uni01] Pointbase, Pointbase UniSync Developer's Guide, Version 4.2, A White Paper.

[Woo01] 우승택, 최대석, 김연숙, 이진호, 이정태, Residential Gateway 하드웨어 요구사항에 대한 분석, 정보과학회 2001 봄 학술발표 논문집(B), 제 28 권, 제 1 호. pp 151~152, 2001 년 3 월.