SQL3 확장에 의한 멀티데이터베이스 시스템의 스키마 통합

김미연[O] 송주원[+] 백두권[*]
한국통신 멀티미디어연구소[O+], 고려대학교 컴퓨터학과[*]
{miyeon, jwsong }@kt.co.kr, dkbaik@swsys2.korea.ac.kr

Schema Integration in Multidatabase System Using the Extension of SQL3

Mi-Yeon Kim[O] Ju-Won Song[+] Do-Kwon Baik[*]
Korea Telecom Multimedia Laboratory[O+], Dept. of Computer Science Korea University[*]

요 약

이기종 DB 에 분산 저장되어 있는 데이터를 통합하여 사용자가 단일한 환경 내에서 각종 데이터를 접근, 관리할 수 있는 기술로써 멀티데이터베이스 시스템(Multidatabase System: MDBS)이 연구되어 왔다. 그러나, 기존의 MDBS 는 사용하는 통합 모델에 따라 데이터 정의 언어나 데이터 처리 언어를 새로 정의하여 사용하였기 때문에, 지역 데이터베이스 시스템의 사용자는 새로운 데이터 모델과 데이터베이스 언어를 습득해야 하는 문제가 발생한다. 그러므로, 이를 해결하기 위해 현재 표준으로 진행되고 SQL3 를 기반으로 한 MDBS 의 개발이 요구된다. 본 논문에서는 SQL3 를 확장한 멀티 데이터베이스 질의 언어(Multidatabase Query Language: MQL)를 제시하고 MQL 을 이용한 3 단계 스키마 통합 방안을 제안하고자 한다.

1. 서론

지금까지 DB들은 필요에 따라 그 시대에 유행하던 기술을 이용하여 구축되어왔다. 따라서, 관계형, 객체지향형, 객체 관계형 등 다른 기술 및 이기종 시스템에 기반으로 하는 DB들이 산재하게 되었다. 이러한 환경 하에서 여러 데이터들을 동시에 이용하고자 하는 응용들이 발생되고 이에 대한 연구가 80년대 후반부터 DB 연구 분야에서 집중적으로 연구되고 있다[1][2].

이중 가장 긍정적인 방법으로 인정되고 있는 시스템 기술은 하나의 소프트웨어 통합 계층을 이용하여 하부의 지역 데이터베이스 시스템(Local Database System: LDBS)들을 통합해서 하나의 DB시스템처럼 사용할 수 있도록 하는 MDBS 기술이다[3]. 그러나, 기존의 MDBS는 사용하는 통합 모델에 따라 데이터 정의 언어나 데이터 처리 언어를 새로 정의하여 사용하였기 때문에, LDBS의 사용자는 새로운 데이터 모델과 데이터베이스 언어를 습득해야 하는 문제가 발생한다. 그러므로, 이를 해결하기 위해 표준 데이터베이스 언어를 지원하는 MDBS의 개발이 필요하다. 현재 표준으로 사용하고 있는 관계형 언어 SQL-2(혹은 SQL-92)는 객체 관계형 언어 SQL3[4][5]로 확장되고 있으며, 이에 따라 DB업체들도 SQL3를 반영하고 있는 추세이다[6][7][8]. 이러한 추세를 감안하면 SQL3를 기반으로 한 객체 관계형 MDBS의 개발이 요구된다.

본 논문에서는 SQL3기반의 객체 관계형 모델을 지원하는 MDBS에서의 스키마 통합 방안을 제시하고자 한다. 이에 따라, 객체 관계형 MDBS의 스키마 통합 기능을 지원하기 위해 SQL3를 확장한 MQL을 제시하고, MQL을 사용하여 LDB의 스키마를 삼단계로 통합하는 방안을 제시하고자 한다. 본 논문의 구성은 다음과 같다. 제 2 장에서는 기존의 MDBS를 살펴본다. 제 3 장에서는 MDBS에서의 삼단계 스키마 통합 방안과 각 단계에서 사용할 MQL을 제시한다. 제 4 장에서 결론을 맺는다.

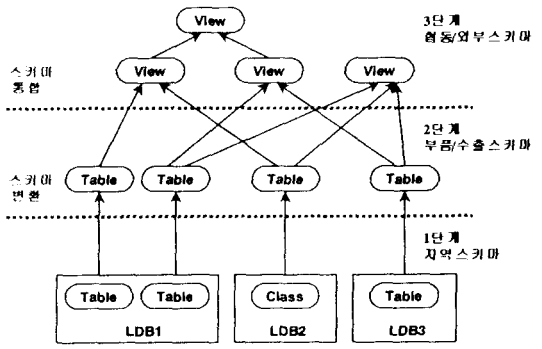
2. 관련연구

최초의 MDBS는 1980년 초 Xerox Advanced Information Tec. 사에서 시작한 MULTIBASE이다. MULTIBASE는 DAPLEX라는 함수 언어를 사용하여 함수 데이터 통합 모델을 제공한 프로토타입 시스템이다. 그 후 1980년대에 개발을 시작한 대부분의 MDBS는 관계형 통합 모델을 지원하였으며, 1980년 대 말부터 객체지향형 통합 모델을 지원하는 시스템이 등장하기 시작했다. 관계형 통합 모델을 지원한 시스템은 SQL을 기반으로 한 MDB 언어를 지원하였으며, 객체지향형 통합 모델을 지원하는 시스템은 SQL을 확장한 질의 기반 MDB 언어 또는 객체지향 프로그래밍 언어를 확장한 프로그래밍 기반 MDB 언어를 지원하였다. 관계형 통합 모델을 지원한 시스템으로 ADDS(Amoco Distributed Database System), DATAPLEX, Mermaid, Ingres/Star 등이 있다. 질의 기반 객체지향 통합 모델을 지원한 시스템으로는 Pegasus, OIS(Operational Integration System), CIS(Comandos Integration System) 등이 있다. 프로그래밍 기반 객체지향 통합 모델을 지원한 시스템으로는 ViewSystem, DOMS(Distributed Object Management System) 등이 있다. 객체 관계형 통합 모델을 지원한 시스템으로는 UNISQL/M이 있다.

3. 삼단계 스키마 통합

MDBS를 위한 스키마 구조로는 분산, 이질성, 자치성의 특성을 지원할 수 있도록 지역, 부품, 수출, 협동, 외부 스키마로 구성된 오단계 스키마 구조[9]가 제안되었다. 그러나, 오단계 스키마 구조는 MDBS의 특징을 구분하여 설명하기 위한 하나의 모델일 뿐 실제 구현 시 이들 다섯 단계를 모두 구현해야 하는 것은 아니다. 지역 스키마를 제외하고 둘 이상의 스키마들을 한 단계의 스키마로 간략히 표현하거나 특정 단계의 스키마는 생략하여 3단계 혹은 4 단계 스키마 구조를 지원할 수 있다.

본 논문에서는 부품 스키마와 수출 스키마를 한 단계로 합치고, 협동 스키마와 외부 스키마를 한 단계로 합쳐 <그림 1>와 같은 3단계 스키마 통합 방안을 제안한다.



<그림 1> 3 단계 스키마 통합 구조.

MDBS에서 3단계 스키마 통합을 하기 위해 가장 먼저 해야 할 일은 LDB를 등록하여 접근 가능한 지역 스키마를 정의하는 것이다. 다음 단계로는 지역 스키마의 테이블 또는 클래스를 전역 테이블과 대응시켜주기 위한 스키마 변환 과정이 필요하다. 마지막으로, 변환된 전역 테이블을 통합하여 사용자를 위한 전역 뷰를 생성하는 스키마 통합 과정이 필요하다. 본 장에서는 LDB 등록, 스키마 변환, 스키마 통합 과정에서 사용하기 위한 MQL을 제안한다. 단, 관계형과 객체지향 DB 모델의 용어를 통일하여 지칭하기 위해서 개체-관계형 모델의 용어를 사용하기로 한다. 그러므로, 관계형 DB의 테이블과 객체지향형 DB의 클래스를 모두 지칭하기 위해서는 '개체집합'이라는 용어를 사용한다. 테이블의 열과 클래스의 요소를 모두 지칭하기 위해서는 '속성'이라는 용어를 사용한다. 또한, 테이블의 튜플과 클래스의 객체에 대해서는 '개체'라는 용어를 사용한다. MQL문법의 표기는 확장 BNF를 이용한다. 꺾음괄호(<>)로 묶여진 비단말 기호 중, 본 논문에서 설명되지 않은 것은 SQL3 명세서[4]를 참조한다.

3.1 지역 데이터베이스 등록

LDB를 등록하여 접근 가능한 지역 스키마들을 정의한다. LDB를 등록할 때는 LDB의 고유 이름, 종류, 기반 모델링 기법, 등록 이름 및 LDB가 존재하는 호스트 이름, LDB의 버전, 사용자 이름, 암호에 대한 속성을 포함하는 다음 MQL의 LDB 정의문을 사용한다.

```
CREATE LDB <ldb name>
FROM <ldb name at host>
OF <ldb type> AS <ldb model>
WITH <ldb attr value clause> [{<ldb attr value clause> }... ]
```

위의 문장에서 <ldb name>은 MDBS에서 참조되는 LDB 이름이고 <ldb name at host>는 LDB 고유의 이름이다. <ldb type>은 LDB의 유형으로 'Informix', 'Oracle', 'Unisql' 등으로 표현한다. <ldb model>은 LDB의 모델로 관계형 DB의 경우 'RDB', 객체지향형 DB의 경우 'OODB'로 표현한다. <ldb attr value clause> 절은 등록되는 LDB의 속성을 다음과 같은 형식으로 표현한다.

```
<ldb attr value clause> ::= <ldb attr name> <ldb attr value>
<ldb attr name> ::= HOST | VERSION | USER | PASSWORD
    | MAX_AGENT | MIN_AGENT | AGENT_DECAY
```

<ldb attr name>은 등록할 LDB의 속성 이름을 지정하고, <ldb attr value>에 의해 속성 값을 표현한다. LDB가 존재하는 호스트 이름을 지정하기 위해서는 <ldb attr name>으로 HOST라는 예약어를 사용하고, LDB 버전에 대해서는 VERSION, 사용자 이름과 암호에 대해서는 각각 USER, PASSWORD라는 예약어를 사용한다. 또한, 하나의 LDB에 대하여 최대/최소로 적재되어야 할 대리 프로세스의 수에 대해서는 각각 MAX_AGENT, MIN_AGENT 예약어를 사용하고, 하나의 대리 프로세스가 시간 초과되어 자동으로 종료할 때까지 걸리는 시간은 AGENT_DECAY 예약어를 사용하여 지정한다.

3.2 스키마 변환

지역 스키마를 전역 스키마로 변환하는 과정에서는 지역 개체 집합을 전역 테이블과 대응시키고 각 지역 속성을 전역 테이블의 열로 대응시키는 작업이 필요하다. 이 때, 고려해야 할 사항은 지역 개체 집합에 대한 수정 가능성이다. 사용자가 전역 스키마을 통해 지역 데이터를 검색하도록 지원할 뿐만 아니라 LDB에 새로운 데이터를 추가하거나 기존의 지역 데이터를 수정, 삭제할 수 있는 기능이 지원되어야 한다. 이러한 요구조건을 만족시키기 위해서 전역 테이블은 지역 개체 집합과 반드시 일대일 대응되어야 한다. 이는 두 개 이상의 지역 개체 집합에 대응하는 하나의 전역 테이블을 정의할 수 없을 뿐만 아니라, 하나의 지역 개체 집합에 대응하는 두 개 이상의 전역 테이블을 정의할 수 없음을 의미한다.

3.2.1 전역 테이블 정의문

다음의 MQL의 전역 테이블 정의문을 사용하면 지역 개체 집합은 전역 테이블로 변환할 수 있다.

```
CREATE TABLE <table name>
( <table element> [ {, <table element> }... )
AS SELECT [ DISTINCT | ALL ] <ldb select list>
    FROM <ldb entity-set name>
    [ WHERE <search condition> ]
ON LDB <ldb name>
```

MQL의 전역 테이블 정의문은 SQL3의 테이블 정의문과 차이가 있다. 첫째, AS SELECT 절이 추가되어 전역 테이블에 대응하는 지역 개체 집합과 전역 테이블의 열에 대응하는 지역 속성을 정의하는 것이다. 둘째, ON LDB 절이 추가되어 해당 지역 개체 집합이 저장되어 있는 LDB를 지정한다는 것이다. <table name>으로 지정되는 전역 테이블 이름은 <ldb entity-set name>으로 지정되는 지역 개체 집합의 이름과 일대일 대응된다. 전역 테이블

의 구성 요소인 <table element>는 전역 테이블에 대한 열의 정의, 제한 조건 정의 등이 포함되며 전역 테이블의 구성 요소로 정의된 열들은 <ldb select list>에 의해 나열된 지역 개체 집합의 속성들과 순서적으로 일대일 대응된다. 전역 테이블의 열을 정의하는 <column definition>은 다음과 같은 형식으로 정의된다. 이 때, 전역 열과 대응되는 지역 속성의 값은 <data type>으로 지정된 전역 열의 데이터 타입으로 변환된다.

```
<column definition> ::= <column name> <data type>
  [ <default clause> ] [ <attribute constraint definition>... ]
```

지역 개체 집합을 전역 테이블로 변환할 때, 지역 개체 집합에 포함된 개체들 중 특정한 조건을 만족시키는 개체들만을 전역 테이블로 변환시킬 수 있다. 이는 전역 테이블을 정의할 때 AS SELECT 절에 WHERE 절을 추가하여 수평 분할한 지역 개체 집합을 전역 테이블에 대응시킴으로써 가능하다. 지역 개체 집합을 구성하는 속성 중 필요한 속성만을 추출하여 전역 테이블을 정의할 수도 있다. 이는 지역 개체 집합에 대한 AS SELECT 절에 추출하고자 하는 지역 속성 만을 지정하여 수직 분할된 지역 개체 집합과 전역 테이블을 대응시키는 것이다. 단, 하나의 지역 개체 집합에 대응한 전역 테이블은 오직 하나 만을 정의할 수 있으므로 지역 개체 집합에 대한 수평 분할로 인해 제외된 지역 개체 혹은 수직 분할로 인하여 제외된 속성이 MDBS에서 사용되지 않을 것이 확실한 경우에 한하여 수직 분할 또는 수평 분할을 적용한다.

3.2.2 타입화된 전역 테이블 정의문

다음 MQL의 타입화된 전역 테이블 정의문을 통해 지역 개체 집합을 변환할 수 있다. 타입화된 전역 테이블의 정의는 일반 전역 테이블의 정의와 달리 테이블을 구성하는 열을 정의하는 대신 OF 절에 의해 테이블에 적용되는 사용자 정의 타입(User Defined Type:UDT)을 정의한다. 이와 같이 정의된 타입화된 전역 테이블은 UDT의 각 속성에 대응한 열들로 구성되며, 테이블의 각 행은 객체 식별자를 갖게 된다. 또한, UNDER 절에 의해 슈퍼 테이블을 정의함으로써 테이블 계층을 구성할 수 있다.

```
CREATE TABLE <table name> OF <user-defined type>
[ UNDER <supertable name> ]
[ ( <table element> [ { , <table element> } ] ) ]
AS SELECT [ DISTINCT | ALL ] <ldb select list>
    FROM <ldb entity set name>
   [WHERE <search condition> ]
ON LDB <ldb name>
```

3.3 스키마 통합

전역 테이블을 정의하는 단계는 지역 스키마를 전역 스키마로 통합하기 위한 준비 단계로 볼 수 있다. 상이한 데이터 모델로 정의된 지역 스키마를 단일한 데이터 모델의 전역 스키마로 변환하는 단계가 전역 테이블을 정의하는 단계이다. 이 준비 단계가 끝나면 여러 LDB에 분산 정의된 지역 스키마를 하나의 전역 스키마로 통합할 수 있는데 이 통합 작업은 전역 뷰를 정의함으로써 이루어진다.

3.3.1 전역 뷰 정의문

다음은 MQL의 전역 뷰 정의문으로 SQL3의 뷰 정의문과 형식이 동일하다.

```
CREATE VIEW <view name>
( <view column list> )
AS <query expression> [ WITH CHECK OPTION ]
```

전역 뷰은 전역 테이블과 달리 반드시 수정 가능해야 할 필요가 없으므로, 전역 뷰와 전역 테이블은 일대일 대응될 필요가 없다. 그러므로, <query expression>은 하나 이상의 전역 테이블 또는 다른 전역 뷰로 부터 열들을 참조할 수 있다. <query expression>에 의해 검색되는 <select list>의 요소는 <view column list>의 각 요소와 일대일 대응되어야 한다. <view column list>는 뷰에 포함되는 열 이름들을 정의하는 것으로, 전역 테이블의 열과 달리 데이터 타입 또는 제한 조건 등을 정의할 수 없다. 전역 뷰의 각 열의 데이터 타입과 제한 조건은 뷰에 의해 참조되는 전역 테이블의 열에 의존한다.

3.3.2 타입화된 전역 뷰 정의문

타입화된 전역 테이블과 마찬가지로 UDT를 이용하여 다음과 같이 타입화된 전역 뷰를 정의할 수 있다. 타입화된 전역 뷰도 OF 절에 의해 UDT를 지정하며, UNDER 절에 의해 전역 뷰에 대한 계층을 구성할 수 있다.

```
CREATE VIEW <view name> OF <user-defined type>
[ UNDER <table name> ]
[ <view element list> ]
AS <query expression> [ WITH CHECK OPTION ]
```

4. 결론

LDB 사용자가 기존의 MDBS를 사용하기 위해서는 MDBS를 위해 새로 정의된 언어를 습득해야 하는 문제점이 있다. 그러므로, 이를 해결하기 위한 방안으로 현재 표준으로 진행되고 있는 SQL3를 기반으로 한 객체 관계형 모델의 MDBS를 개발할 필요가 있다.

본 논문에서는 MDBS의 전역 스키마를 구성하기 위한 3 단계 스키마 통합 방안을 제시하고, 각 단계에서 스키마를 생성하기 위한 언어로써 SQL3를 확장한 MQL을 제시하였다. 향후, 다양한 지역 스키마를 통합할 때 발생되는 의미 충돌을 분석하고 MQL을 사용하여 이러한 충돌 문제를 해결하는 방안을 연구할 예정이다.

5. 참고문헌

[1] Won Kim, "Introduction to Part2: Technology for Interoperating Legacy Databases," Modern Database Systems, pp.521-550, ACM Press and Addison Wesley, 1995.

[2] G. Tomas et al., "Heterogeneous Distributed Database Systems for Production Use," ACM Computing Surveys, Vol. 22, No. 3, pp. 237-266, Sep. 1990.

[3] E. Pitourra, O. Bukhres, and A. Elmagarmid,"Object Orientation in Multidatabase Systems," ACM Computing Surveys, Vol. 27, No. 2, pp. 141-195, June 1995.

[4] ANSI TC X3H2, ISO/IEC JTC 1/SC 21/WG, "ISOANSI Working Draft Foundation (SQL/Foundation)," March 1999.

[5] A. Eisenberg, J. Melton, "SQL:1999, Formerly Known as SQL3," ACM SIGMOD Record, Vol. 28, No. 1, March 1999.

[6] M. Carey et al, "O-O, What's Happening to DB2?," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, 1999.

[7] V. Krishnamurhty, S. Banerjee, A. Nori, "Bring Object-Relational Technology to the Main Stream," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, 1999.

[8] P. Brown, "Implementing the Spirit of SQL-99," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, 1999.

[9] A. Sheth and J. Larson, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases," ACM Computing Surveys, Vol. 22, No. 3, pp. 183-236, Sep. 1990.