The Journal of Korean
Institute of CALS/EC
Vol. 4, No. 3, Dec. 1999

EasyConnection : 이질 데이터베이스들의 효율적 연동 방식

변광준*

EasyConnection : An Efficient Approach to Integrating Heterogeneous Databases

Kwang June Byeon

Abstract

There have been several approaches to integrating heterogeneous database systems in enterprises. Due to their inherent limitations, however, it has been difficult to apply them to actual situations in enterprises. In this paper, we propose EasyConnection, a system that is based on the flexible and extensible architecture and overcomes the limitations of the previous approaches. EasyConnection consists of Wrappers, Sharing Manager, and Clients, and the connections among these components are based on CORBA. It can also be used as the information management mechanism for such application systems as ERP, PDM, and EAI.

Keyword: Database Integration

* 아주대학교 정보 및 컴퓨터공학부 부교수

1. 서론

기업에서 정보는 일반적으로 네트워크에 연결된 이질적인 데이터베이스 시스템들에 의해 저장 관리되고 있다. 이질적인 데이터베이스 시스템들 중 일부는 부분적으로 통합되어 서로간에 정보 공유가 이루어지기도 하지만 대체적으로 각 데이터베이스 시스템은 독립적이고 자율적인 방식으로 정보를 관리하고 있다. 따라서, 데이터베이스 시스템의 자율성을 최대한 보장하면서 효율적인 정보 공유를 가능하게 해 주는 시스템은 정보의 통합 관리를 통해 기업의 경쟁력을 높이려는 국내외 많은 기업들이 지속적으로 관심을 가지고 투자해 온 부분이다. 이와 같은 시스템은 데이터베이스 분야에서는 연방(federated) 데이터베이스 시스템, 멀티데이터베이스 시스템 등으로 불린다[Bukhres, Elmagarmid, 1996; Sheth, Larson, 1990].

현재까지 제안된 대부분의 연방 데이터베이스 시스템들은 공통적으로 데이터베이스 스키마(schema)들을 통합하고 통합된 스키마에 대해 전역(global) 질의를 처리하는 강결합(tightly-coupled) 방식을 채택하였고[Katz, Goodman, 1981; Landers, Rosenberg, 1982], 일부 시스템들의 경우만이 스키마 통합 없이 전역 질의를 처리하는 약결합(loosely-coupled) 방식을 채택하였다[Litwin, Abdellatif, 1986]. 하지만 이들이 채택한 스키마 통합 및 전역 질의는 데이터베이스 시스템의 수가 몇 개인 경우에만 적용 가능하고, 약결합 방식에서는 사용자가 데이터베이스 스키마들에 대한 정보를 미리 스스로 알고 있어야 하는 문제점을 가지고 있다. 따라서 많은 기업들은 연방 데이터베이스 시스템을 구축하기보다는 ODBC, JDBC 등을 기반으로 스키마 통합이나 전역 질의 없이 각 데이터베이스 시스템에 대한 개별적인 연결과 질의를 지원하는 해결 방식을 채택하게 되었다. 하지만 이 방식도 사용자가 연결하는 데이터베이스의 스키마에 대한 정보를 미리 스스로 알고 있어야 하는 문제점을 가지고 있다.

본 연구에서는 이에 착안해 기존의 데이터베이스 연동 방식들이 가지는 문제점들을 극복하고 시스템의 확장성 및 유연성에 중점을 두어 기업의 실제 업무 환경에 적용할 수 있는 시스템인 EasyConnection을 설계하고 현재 그 프로토타입을 구현하고 있다. EasyConnection의 구조는 각 데이터베이스 시스템별로 존재하는 Wrapper, 다수의 클라이언트, 하나의 공유 매니저로 구성된다. Wrapper는 공유 매니저와의 상호작용을 통해 해당 데이터베이스 시스템을 연방 데이터베이스 시스템에 능동적으로 참여/탈퇴시키고, 클라이언트의 질의를 데이터베이스 시스템에 맞게 전달하는 역할을 담당한다. 공유 매니저는 Wrapper들이 제공하는 메타데이터를 저장 관리하고 클라이언트에게 필요한 메타데이터를 제공해 질의할 수 있도록 해 주며, 클라이언트는 공유 매니저로부터 얻은 메타데이터를 이용해 관련 Wrapper에게 질의를 전달해 처리하는 역할을 담당한다. EasyConnection은 기업내의 통합 데이터베이스 시스템, ERP, PDM, EAI(Enterprise Application Integration) 등의 구현을 위한 정보 관리 체제에 적용될 수 있을 것이다.

본 논문의 구성은 2장에서 기존의 관련 연구들과 문제점들을 설명한다. 3장에서는 EasyConnection의 시스템 구조 및 구성 요소들의 기능에 대해 설명한다. 4장에서는 구현 현황 및 향후 연구 계획을 설명하고 결론을 맺는다.

2. 관련 연구

연방 데이터베이스 문제는 데이터베이스 분야에서 지난 20여년 동안 연구되어 왔으며, 그 동안 제안된 시스템들은 크게 강결합 방식과 약결합 방식으로 구분된다[Sheth, Larson, 1990]. 강결합 방식은 약결합 방식에 비해 월등히 많은 시스템들에서 채택된 방식으로서, 각 데이터베이스 스키마를 객체지향 모델에 기반해 통합하고 통합된 스키마를 이용하여 복수의 데이터베이스에 대한 객체지향 전역 질의를 지원함으로써 일종의 분산 데이터베이스 시스템을 구축하는 데에 초점을 맞추었다[Katz, Goodman, 1981; Landers, Rosenberg, 1982]. 하지만 이 방식은 데이터베이스의 크기가 어느 정도 되는 실제 상황에서는 단지 2-3개의 데이터베이스 스키마를 통합하는 것도 쉽지 않고 그 이상일 경우에는 거의 불가능한 문제점을 가지고 있다.

이에 반해 약결합 방식은 스키마 통합 없이 객체지향 전역 질의를 가능하게 하는 방식으로 강결합 방식의 제약성을 일부 해결했다[Litwin, Abdellatif, 1986]. 하지만 이 방식은 질의를 하기 위해 필수적으로 알고 있어야 하는 데이터베이스 스키마에 대한 정보가 사용자에게 제공되지 않고 사용자가 스스로 미리 알고 있어야 하는 문제점을 가지고 있다. 이용해야 하는 데이터베이스의 수가 많은 경우에는 각 데이터베이스의 스키마를 모두 기억하는 불가능하기 때문에 이 방식 역시 데이터베이스 시스템의 수가 몇 개인 경우에만 적용 가능하다.

위의 두 방식이 공통적으로 지원하는 전역 질의는 개념적으로는 반드시 필요하지만 시스템의 구현이라는 측면에서는 상당히 어려운 작업을 필요로 한다. 전역 질의의 처리 및 최적화, 이와 관련된 트랜잭션 관리 등은 그 동안 많은 관련 연구가 진행되어 왔고 현재도 진행되고 있는 상황이다. 따라서, 기업의 실제 업무 환경에서는 대부분의 경우 스키마 통합 및 전역 질의를 기반으로 하는 연방 데이터베이스 시스템을 구축하기보다는 ODBC, JDBC 등을 이용해 각 데이터베이스 시스템에 대한 개별적인 접근과 관계형 질의를 가능하게 하는 방식(이하 무결합 방식)을 채택하게 되었다. 하지만 이 방식도 약결합 방식과 같이 사용자가 스스로 연결된 데이터베이스의 스키마를 미리 알고 있어야 하는 문제점을 가지고 있다.

연방 데이터베이스 시스템 문제를 CORBA [Orfali, et. al., 1996]를 이용한 분산 객체 시스템을 기반으로 해결하려는 연구들도 시도되었다[Benatallah, Bouguettaya, 1997; Dogac, et. al., 1995; Kilic, et. al., 1995]. 하지만 이들은 새로운 방식을 제시하기보다는 스키마 통합 및 전역 질의를 기반으로 하는 기존의 강결합 방식을 이용하고, 단지 이질적인 시스템들의 연결을 용이하게 해 주는 CORBA의 장점을 이질적인 데이터베이스 시스템의 연결에 적용해 연방 데이터베이스 시스템을 구현한 수준에 머물렀다.

3. 시스템 구조 및 기능

기존의 연방 데이터베이스 시스템 구축 방식인 강결합 및 약결합 방식은 2장에서 살펴본 바와 같이 기업의 실제 업무 환경에 적용하기에는 각기 제약성을 가지고 있다. 본 연구에서는 이에 착안해 연방 데이터베이스 시스템을 구축하기보다는 가장 현실성이 있는 무결합 방식의 장점을 최대한 살리고 동시에 단점을 해결하며 시스템의 확장성 및 유연성에 중점을 둠으로써, 실제 업무 환경에 적용가능한 시스템인 EasyConnection을 CORBA를 기반으로 설계하고 현재 그 프로토타입을 구현하고 있다.

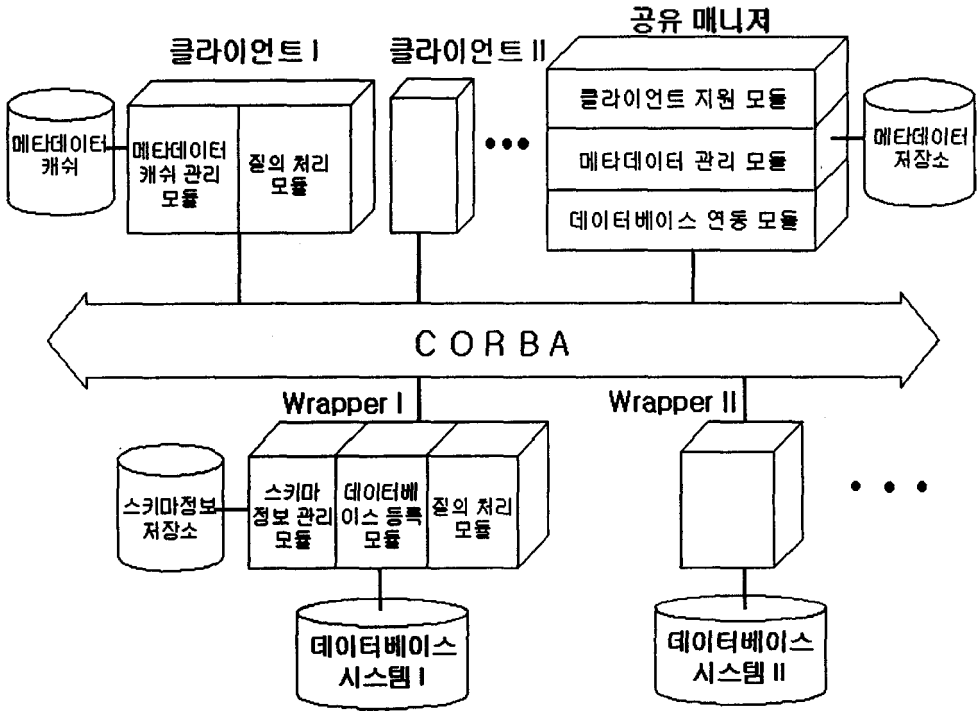
EasyConnection은 무결합 방식과 마찬가지로 스키마 통합 및 전역 질의를 배제하고 각 데이터베이스에 대한 개별적인 연결과 관계형 질의를 지원함으로써, 기존의 강결합 및 약결합 방식이 가지는 문제점을 해결해 준다. 또한, 질의를 하기 위해 필요한 데이터베이스 스키마에 대한 정보를 사용자에게 제공하는 별도의 모듈(즉, 클라이언트)을 가지고 있으며, 이 모듈은 사용자가 질의를 하기 전에 데이터베이스들의 스키마를 알고 있어야 하는 무결합 방식의 문제점을 해결해 준다. 따라서, 사용자는 클라이언트로부터 제공되는 스키마 정보를 이용해 찾고자 하는 데이터를 보유하는 데이터베이스 시스템을 쉽게 알아내 질의를 요청할 수 있게 된다. 각 클라이언트는 여러 데이터베이스들에 존재하는 서로 다른 스키마 정보를 보유할 수 있기 때문에, 데이터베이스들위에 정의된 각기 다른 뷰(view)의 역할을 수행한다.

EasyConnection은 각 데이터베이스 시스템이 다른 데이터베이스 시스템들과 능동적으로 연동하고 분리할 수 있게 해 주는 기능도 제공하며, 이로 인해 시스템 구조의 확장성 및 유연성이 상대적으로 뛰어난 특징을 가진다. 또한, 시스템 구조가 CORBA를 기반으로 함으로써 이질 데이터베이스 시스템들의 연결을 구현하는 작업이 상대적으로 용이한 특징도 가진다. EasyConnection에서 지원되는 관계형 질의를 객체지향 질의에 비해 제한적이라고 볼 수 있으나, 아직까지 객체지향 데이터 모델 및 질의어의 진정한 표준이 존재하지 않은 실정을 감안한다면 관계형 질의를 채택하는 것이 보다 현실적이라 할 수 있다[Cattell, Barry, 1997].

시스템의 구조는 <그림 1>에서 보는 바와 같이 데이터베이스 연동에 참여하는 각 데이터베이스 시스템에 연결되어 별도로 존재하는 Wrapper, 하나 이상의 클라이언트, 그리고 하나의 공유 매니저로 구성된다. Wrapper, 클라이언트, 공유 매니저는 CORBA로 연결되어 있으며, Wrapper와 각 데이터베이스 시스템은 JDBC로 연결되어 있다.

3.1 Wrapper

Wrapper는 <그림 1>에서 보는 바와 같이 스키마 정보 저장소, 스키마 정보 관리 모듈, 데이터베이스 등록 모듈, 질의 처리 모듈로 구성된다. 스키마 정보 저장소 및 스키마 정보 관리 모듈은 Wrapper에 연결된 데이터베이스의 스키마중 외부와 공유하는 스키마와 이에 대한 부연 설명을 저장 관리한다. 저장된 내용은 <그림 2>에서 보는 바와 같이 Database Info, Table, Info, Attribute Info의 3개 테이블로 구분되고, 밑줄쳐진 애트리뷰트들은 주요 키(primary key)를 나타낸다.



<그림 1> 시스템 구조

Database Info

데이터베이스 이름	IP 주소	사용자 ID	패스워드	부연 설명

Table Info

테이블 이름	부연 설명

Attribute Info

테이블 이름	애트리뷰트 이름	도메인	Access Control Flag	부연 설명

<그림 2> 공유되는 스키마 정보

Database Info는 해당 데이터베이스의 이름, 데이터베이스가 위치하고 있는 컴퓨터의 IP 주소, 외부 사용자가 접근할 때 필요한 사용자 ID와 패스워드, 그리고 이해를 돕기 위해 데이터베이스 자체에 대한 부연 설명을 포함하며, 따라서 테이블의 내용은 1개의 튜플(tuple)로만 구성된다. Table Info는 외부와 공유되는 각각의 테이블 또는 뷰(view)의 이름과 부연 설명을 포함한다. Attribute Info는 Table Info에 포함된 테이블 또는 뷰를 구성하는 각 애트리뷰트에 대해서 애트리뷰트가 속한 테이블 이름, 애트리뷰트 이름, 애트리뷰트의 도메인 정의, 접근 제어 현황(Access Control Flag), 그리고 부연 설명을 포함한다.

외부와 공유할 테이블 및 뷰들은 해당 데이터베이스 시스템이 다른 데이터베이스 시스템들과 연동하려는 시점에서 정하게 되며, 이 때 만일 데이터베이스내에 있는 테이블 및 뷰들의 형태와 다른 형태로 공유하려면 경우에 따라 다음의 두 가지 방법을 이용하게 된다. 첫번째 방법은 새로운 뷰들을 원하는 형태로 만든 후 이들에 대한 정보를 스키마 정보 저장소에 포함시키는 것이다. 예를 들면, 데이터베이스내에 있는 Staff과 Department라는 2개의 테이블들을 공유하지 않고 기획 부서에서 일하는 스탭들에 대한 정보만을 공유하고 싶다면, Staff과 Department을 조건에 맞게 Join해서 정의한 뷰를 만든 후 이 뷰를 공유하게 된다.

두번째 방법은 새로운 뷰를 만들지 않고 접근 제어 현황(Access Control Flag)을 이용하는 것이다. 예를 들면, 데이베이스내에 있는 Staff 테이블을 전부 공유하지 않고 이를 구성하는 3개의 애트리뷰트들만을 외부에서 읽을 수만 있게 하고 싶다면, Staff 테이블에 대한 정보를 스키마 정보 저장소에 포함시키되 해당 애트리뷰트들에 대해서만 Read 권한을 허용해 외부와 공유하게 된다. 물론 이 예제는 새로운 뷰를 생성해 처리할 수도 있다.

데이터베이스 등록 모듈은 공유 매니저와의 상호작용을 통해 데이터베이스 시스템을 다른 데이터베이스 시스템들과 능동적으로 연동 또는 분리해준다. 이 모듈은 연동할 때 CORBA로 연결된 시스템들에게 연동에 참여하겠다는 메시지를 보내고 이 메시지는 공유 매니저만이 감지해 등록을 시작하라는 메시지를 답변하게 된다. 이와 같이 공유 매니저가 발견되면 데이터베이스 등록 모듈은 스키마 정보 관리 모듈로부터 얻은 스키마 정보를 공유 매니저에 보내서 공유 매니저의 메타데이터 저장소에 저장시킨다.

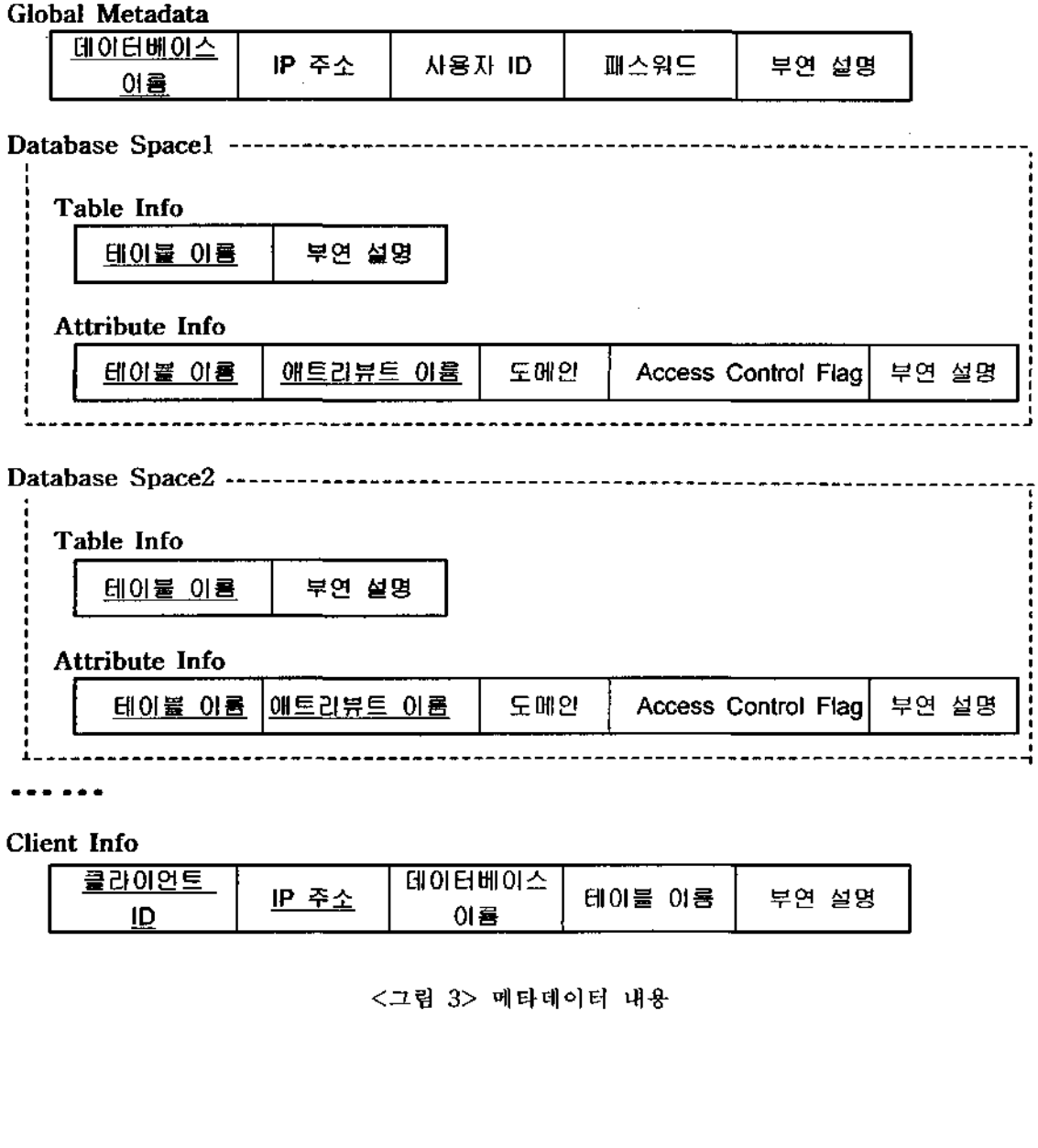
분리 역시 연동과 유사한 방법으로 진행된다. 데이터베이스 등록 모듈은 공유 매니저에게 분리하겠다는 메시지를 보내면, 공유 매니저는 분리를 시작하겠다는 메시지를 답변을 보낸 후 자신의 메타데이터 저장소로부터 해당 스키마 정보를 삭제시킨다. 데이터베이스 등록 모듈이 제공하는 이러한 능동적인 연동 및 분리는 EasyConnection을 기존의 데이터베이스 연동 방식들과 차별화시키는 기능이라고 하겠다.

질의 처리 모듈이 제공하는 기능은 비교적 간단하다. 클라이언트로부터 받은 관계형 질의(즉, SQL 질의)를 JDBC로 데이터베이스 시스템을 연결해 전달하고 처리된 결과를 받아 클라이언트에게 전달한다. 질의 처리 모듈의 이러한 특성은 연동에 참여하는 데이터베이스 시스템들이 JDBC로 연결될 수 있는 관계형 데이터베이스 시스템

들(또는 파일 시스템들)인 환경을 가정하고 있다. 이와 같은 가정은 EasyConnection의 제약성이라 할 수 있겠으나 무결합 방식에서 볼 수 있듯이 기업의 실제 업무 환경에서 현재 가장 많이 사용되고 있는 데이터베이스 시스템들이 관계형인 것을 감안한다면 큰 문제가 되지 않을 것으로 예상된다.

3.2 공유 매니저

공유 매니저는 <그림 1>에서 보는 바와 같이 메타데이터 저장소, 메타데이터 관리 모듈, 데이터베이스 연동 모듈, 클라이언트 지원 모듈로 구성된다. 메타데이터 저장소 및 메타데이터 저장 관리 모듈은 연동에 참



<그림 3> 메타데이터 내용

여하는 데이터베이스 시스템들과 연결된 Wrapper들이 보내온 스키마 정보를 저장 관리한다. 저장된 내용은 <그림 3>에서 보는 바와 같이 Global Metadata라는 테이블과 Database Space1, Database Space 2, … 의 테이블 군, 그리고 Client Info 테이블로 구분되고, 밑줄쳐진 애트리뷰트들은 주요 키를 나타낸다.

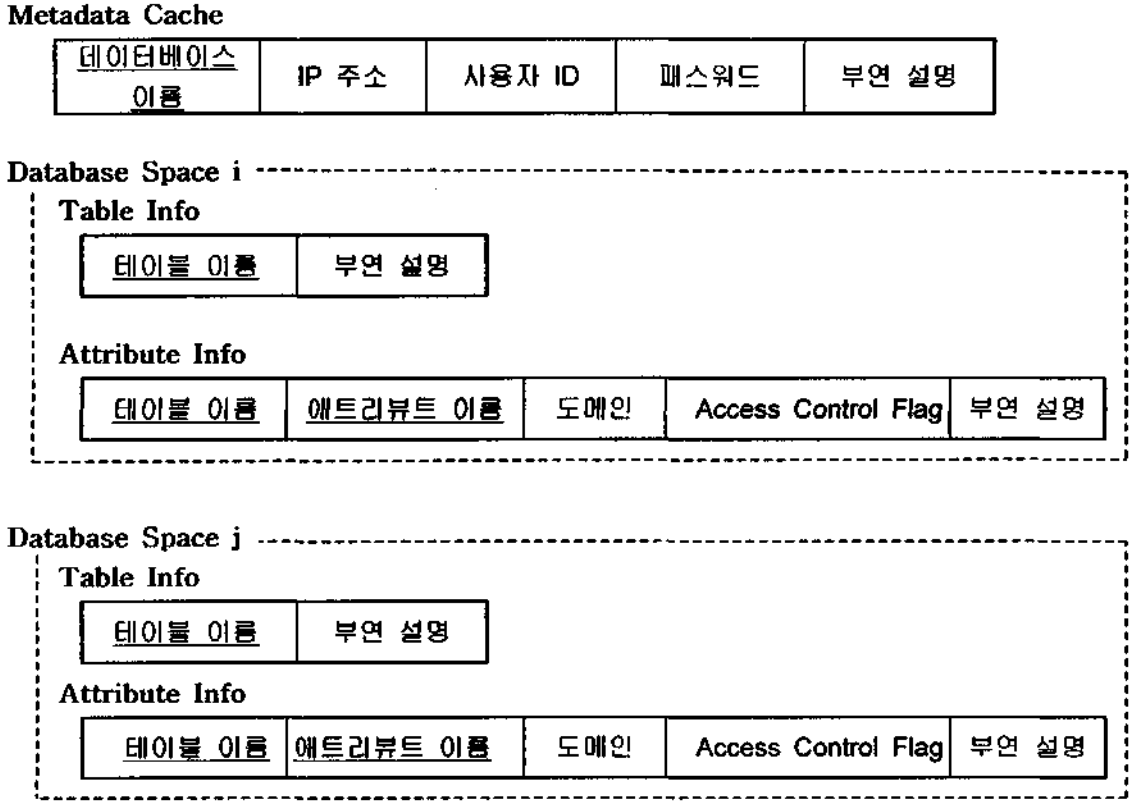
Global Metadata는 연동에 참여한 데이터베이스 시스템들에 대한 기본적인 정보로서 각 데이터베이스 시스템별로 데이터베이스의 이름, 데이터베이스가 위치하고 있는 컴퓨터의 IP 주소, 접근할 때 필요한 사용자 ID와 패스워드, 그리고 부연 설명을 포함한다. 이 테이블의 내용은 각 Wrapper의 스키마 정보 저장소에 있는 Database Info 테이블(<그림 2> 참조)을 구성하는 단일 튜플이 모여 구성된다. 따라서, Global Metadata에 저장되는 튜플의 개수는 연동에 참여한 데이터베이스 시스템들의 수 (또는 공유 매니저와 연결된 Wrapper의 수)와 동일하다. Database Space i는 연동에 참여한 각 데이터베이스 시스템이 공유를 위해 외부로 제공하는 스키마 정보를 나타내며, 내용은 데이터베이스 시스템 i에 연결된 Wrapper i의 스키마 저장소에 존재하는 Table Info 및 Attribute Info와 동일한 내용이다. Client Info는 공유 매니저가 제공하는 메타데이터를 사용하는 클라이언트의 ID, 클라이언트가 위치한 컴퓨터의 IP 주소, 제공받은 데이터베이스 및 테이블 이름으로 구성된다. 또한, Client Info의 데이터베이스 이름은 Global Metadata의 데이터베이스 이름에 대한 참조 키(foreign key)이고, 테이블 이름은 각 Database Space에 있는 Table Info의 테이블 이름을 참조한다.

데이터베이스 연동 모듈은 Wrapper의 데이터베이스 등록 모듈과의 상호작용을 통해 데이터베이스 시스템의 연동 및 분리를 지원한다. 이 모듈은 Wrapper부터의 메시지를 항상 모니터링하고 있다가 연동을 요청하는 메시지를 감지하면 등록 시작을 허락하는 메시지를 보낸다. 이 메시지를 받은 해당 Wrapper의 데이터베이스 등록 모듈이 보내온 데이터베이스 시스템에 대한 기본 정보(단일 튜플)를 메타데이터 관리 모듈에게 전달해 메타데이터 저장소의 Global Metadata에 새로운 튜플로 저장한다. 또한, 연동되는 데이터베이스 시스템을 위한 Database Space i를 메타데이터 저장소에 메타데이터 관리 모듈을 통해 만들고, 여기에 Wrapper의 데이터베이스 등록 모듈이 보내온 Database Info 및 Attribute Info의 내용을 저장한다. 메타데이터의 등록 작업이 끝나면 클라이언트 지원 모듈을 통해 클라이언트들에게 새로운 메타데이터의 확보를 알리는 메시지를 전달해 이용할 수 있도록 한다.

Wrapper로부터 분리를 요청하는 메시지를 감지하면 분리 시작을 허락하는 메시지를 답변하고, 현재 해당 Wrapper를 이용하고 있는 클라이언트들에게 데이터베이스 시스템의 분리를 알린다. 또한, 메타데이터 관리 모듈을 통해 메타데이터 저장소의 Global Metadata 테이블로부터 분리되는 데이터베이스 시스템에 해당되는 튜플을 삭제하고, 이 데이터베이스 시스템을 위한 Database Space i를 메타데이터 저장소로부터 삭제한다.

클라이언트 지원 모듈은 Wrapper들을 접근해 질의를 요청하려는 클라이언트에게 해당 Wrapper들이 외부에 공개한 스키마

정보를 제공한다. 이 모듈은 클라이언트로 하여금 공유 매니저의 메타데이터 저장소에 있는 스키마 정보를 데이터베이스 단위로 검색하고 데이터베이스내의 원하는 정보만을 선택해 자신의 메타데이터 캐쉬 저장소에 다운로드받을 수 있도록 해 준다. 다운로드가 이루어지면 클라이언트 지원 모듈은 메타데이터 관리 모듈을 통해 메타데이터 저장소의 Client Info 테이블에 다운로드된 데이터베이스 및 테이블 관련 정보를 입력시킨다. EasyConnection의 현 버전에서는 테이블 단위의 다운로드만을 허용하고 있다. 예를 들면, 공유 매니저의 메타데이터 저장소에 Database 1에서 Database 10까지 10개의 데이터베이스로부터 공개된 스키마 정보가 들어있다고 하자. 이 경우 클라이언트 지원 모듈은 클라이언트가 원하는 데이터베이스들만을 선택할 수 있고, 이 데이터베이스들의 스키마 정보 중 원하는 테이블들에 관련된 정보만을 다운로드받을 수 있게 해 준다.



<그림 4> 메타데이터 캐쉬 내용

3.3 클라이언트

클라이언트는 <그림 1>에서 보는 바와 같이 메타데이터 캐쉬(Cache), 메타데이터 캐쉬 관리 모듈, 질의 처리 모듈, 워크 스페이스(Work Space)로 구성된다. 메타데이터 캐쉬 및 메타데이터 캐쉬 관리 모듈은 공유 매니저로부터 다운로드한 스키마 정보를 저장 관리한다. 저장된 내용은 <그림 4>에서 보는 바와 같이 Metadata Cache라는 테이블과 Database Space i, Database Space j, … 의 데이블 군으로 구분되고, 밑줄쳐진 애트리뷰트들은 주요 키를 나타낸다.

Metadata Cache는 공유 매니저의 클라이언트 지원 모듈을 통해 선택한 데이터베이스 시스템들에 대한 기본 정보를 나타내며, 공유 매니저의 메타데이터 저장소에 있는 Global Metadata 테이블로부터 해당 데이터베이스 시스템들에 관련된 튜플들을 다운로드받아 저장한다. Database Space i는 선택한 각 데이터베이스별로 만들어지며, Table Info 및 Attribute Info는 데이터베이스내에서 별도로 선택한 테이블들과 이들을 구성하는 애트리뷰트들에 대한 스키마 정보를 역시 공유 매니저의 메타데이터 저장소로부터 다운로드받아 저장한다.

메타데이터 캐쉬 관리 모듈은 메타데이터 캐쉬 저장소를 관리하는 기능외에 공유 매니저의 클라이언트 지원 모듈과의 상호작용을 통해 메타데이터를 검색하고 다운로드하는 역할을 담당한다. 또한, 데이터베이스 시스템의 분리로 인해 메타데이터의 삭제를 알리는 클라이언트 지원 모듈로부터의 메시지를 받아 메타데이터 캐쉬 저장소로부터 해당 메타데이터를 삭제하기도 한다. 메타데이터 캐쉬 관리 모듈은 질의를 하려는 사용자에게 이용할 수 있는 데이터베이스 및 테이블, 나아가 애트리뷰트에 대한 스키마 정보를 제공해 정확한 질의를 할 수 있게 함으로써 사용자와 클라이언트의 질의 처리 모듈과의 상호작용을 지원한다.

질의 처리 모듈은 사용자가 SQL2로 표현해 입력한 관계형 질의를 메타데이터 캐쉬 관리 모듈과의 상호작용을 통해 검사하고 올바른 질의일 경우 해당 Wrapper에게 전달하고 Wrapper의 답변을 기다린다. Wrapper가 처리해 결과를 보내오면 질의 처리 모듈은 이를 사용자에게 전달한다. 질의 처리 모듈은 메타데이터 캐쉬 관리 모듈의 지원을 받아 사용자가 질의를 입력하기 전에 현재 사용가능한 데이터베이스 및 테이블에 관한 스키마 정보를 확인할 수 있게 해 준다.

4. 결론

EasyConnection의 구현 환경은 연동에 참여하는 데이터베이스 시스템으로 Sun Sparc Server 1000에 설치된 Oracle 7.3.3과 Windows NT에 설치된 MS SQL Server 6.5를 채택하고 있으며, 이들과 연결된 Wrapper들은 해당 데이터베이스 시스템이 위치하고 있는 플랫폼에 위치한다. 공유 매니저는 별도의 Windows NT에 위치하고 클라이언트는 Windows 98에 위치한다. Wrapper, 공유 매니저, 클라이언트간의 연결은 IIOP를 지원하는 CORBA 제품인 IONA OrbixMT2.3c를 이용하고, Wrapper와 데이터베이스 시스템의 연결은 JDBC를 이용한다. 또한, Wrapper, 공유 매니저, 클라이언트의 구현은 JDK 1.1.6을 이용하고 있다.

EasyConnection은 기존의 연방 데이터베이스 시스템 구축 방식인 강결합 및 약결합 방식이 가지는 제약성으로 인해 기업의 실제 업무 환경에 적용하기에 어려운 점을 착안해, 연방 데이터베이스 시스템을 구축하기보다는 현실성이 있는 무결합 방식의 장점을 최대한 살리고 단점을 해결한 시스템이다. 이 방식은 무결합 방식과 마찬가지로 스키마 통합 및 전역 질의를 배제하고 각 데이터베이스에 대한 개별적인 연결과 관계형 질의를 지원한다. 또한, 질의를 하기 위해 필요한 데이터베이스 스키마에 대한 정보를 사용자에게 제공하는 별도의 모듈인 클라이언트를 가지고 있기 때문에 무결합 방식이 가지는 문제점을 해결해 준다.

EasyConnection은 각 데이터베이스 시스템이 다른 데이터베이스 시스템들과 능동적으로 연동하고 분리할 수 있게 해 주는 기능도 제공하며, 이로 인해 시스템 구조의 확장성 및 유연성이 상대적으로 뛰어난 특징을 가진다. 시스템 구조는 CORBA를 기반으로 함으로써 이질 데이터베이스 시스템들의 연결을 구현하는 작업이 상대적으로 용이한 특징도 가진다. EasyConnection에서 지원되는 관계형 질의를 객체지향 질의에 비해 제한적이라고 볼 수 있으나, 아직까지 객체지향 데이터 모델 및 질의어의 진정한 표준이 존재하지 않은 실정을 감안한다면 관계형 질의를 채택하는 것이 보다 현실적이라 할 수 있다.

EasyConnection은 단순히 데이터베이스 시스템의 연동에 뿐만 아니라 기업내 통합 정보 시스템을 구축하는데 적용할 수 있으며, ERP, PDM, EAI 등에 대해서도 구현을 위한 하부 정보 관리 체제로 이용할 수 있을 것이다. 또한, 최근 웹 상에서의 정보 표준으로 자리잡아 가고 있는 XML을 이용해 EasyConnection의 스키마 정보 및 메타데이터를 표현함으로써 보다 이질적인 시스템들의 연동에 적용할 수 있는 시스템으로 개선될 수 있을 것으로 예상되며, 현재 이러한 작업을 진행하고 있다.

참고문헌

[Benatallah, Bouguettaya, 1997] Benatallah, B. and Bouguettaya, A., "Data Sharing on the Web", International Conference on Enterprise Distributed Object Computing, Oct. 1997.

[Bukhres, Elmagarmid, 1996] Bukhres, O. and Elmagarmid, A., Eds., Object-Oriented Multidatabase Systems, Prentice Hall, 1996.

[Cattell, Barry, 1997] Cattell, R. and Barry, D., Eds., The Object Database Standard: ODMG 2.0, Morgan Kaufmann, 1997.

[Dogac, et. al., 1995] Dogac, A., et. al., "METU Object-Oriented Database System", SIGMOD Record, Vol. 24, No. 3, Sept. 1995.

[Katz, Goodman, 1981] Katz, R. H., and Goodman, N., "View Processing in MULTIBASE, A Heterogeneous Database System", ER, pp.257-278, 1981.

[Kilic, et. al., 1995] Kilic E., Ozhan G., Dengi C. Kesim N. and Koksal P., "Experiences in Using CORBA for a Multidatabase Implementation", 6th Int. Conf. on Database, 1995.

[Landers, Rosenberg, 1982] Landers, T. A., and Rosenberg, R., "An Overview of MULTIBASE", DDB, pp.153-184, 1982.

[Litwin, Abdellatif, 1986] Litwin W. and Abdellatif A., Multidatabase Interoperability, 1986

[Orfali, et. al., 1996] Orfali, R., Harkey, D., and Edwards, J., The Essential CORBA: System Integration Using Distributed Objects, Wiley, 1996.

[Sheth, Larson, 1990] Sheth A. P. and Larson J. A., "Federated Database Systems for Managing Distributed Heterogeneous, and Autonomous Databases", ACM Computing Surveys, Vol. 22, No. 3, 1990.

저자 소개

변광준

1985년 서울대학교 컴퓨터공학 학사

1987년 Pennsylvania State University 전산학 석사

1993년 University of Southern California 전산학 박사

1994년 ~ 현재 아주대학교 정보 및 컴퓨터공학부 조교수, 부교수

관심분야 : 통합 데이터베이스 시스템, 전자상거래, EAI