미디에이터를 이용한 분산 데이터베이스의 통합 뷰관리 시스템 *

주길홍, 이원석
연세대학교 컴퓨터과학과 데이터베이스 연구실

Integrated View Management System of Distributed Database Using Mediator

Kilhong Joo, Wonsuk Lee
Department of Computer Science, Yonsei University

요 약

오늘날 컴퓨터의 기술이 발전함에 따라서 하드웨어와 소프트웨어가 같수록 다양화되고, 네트워크의 컴퓨팅 환경도 다양해져 가고 있다 또한 사용자가 요구하는 데이터 양도 급속히 증가하고 있고, 네트워크 환경에서 많은 정보들을 얻으려고 한다 그러나 기존의 데이터베이스들은 서로 다른 모델로 표현이 되어져있고 각기 서로 다른 운영체제에서 사용되고 있다 이러한 문제점을 극복하기 위해서 미디에이터 시스템이 연구되고 있고 현재의 미디에이터 시스템은 관리자가 각 데이터베이스의 구조와 내용을 알고 직접 통합시킬 뷰를 생성하고 있다 또한 데이터를 통합시키기 위하여 생성한 뷰 관리에도 수정(modfication)만을 지원하고 있다 따라서 본 논문에서는 기존의 다양한 환경의 데이터베이스를 CORBA를 사용하여 구축하고 웹과 연결함으로써 사용자가 다양하게 원하는 정보를 얻을 수 있고 통합된 뷰 관리를 수정(modification)과 구체화(materialization)의 두 가지 방법으로 관리하여 효율성을 높이도록 한다

1. 서론

오늘날의 컴퓨팅 환경은 기존의 하드웨어 및 소프트웨어들이 각기 제한된 범위내에서 서로 유용한 기능들을 제공하고 있으나 네트워크 상에서는 서로 독립적으로 운용이 되고 있기 때문에 효율적으로 상호운용이 되지 않고 있다.[1] 따라서 분산 환경에서 시스템의 성능 개선이나 유지보수에 많은 비용이 요구되고 있고 이질적인 데이터 모델들이 서로 다른 운영환경에서 데이터를 통합하려고 함으로 사용자의 요구에 적절히 대응하지 못하고 있는 실정이다. 그러나 점차 네트워크 환경의 시스템으로 발전을 하고 사용자가 원하는 데이터 양이 증가하게 됨에 따라서 네트워크를 통한 데이터의 통합 및 공유의 필요성이 대두되었고, 분산 환경에서 데이터베이스의 통합이 필요하게 되었다. 이러한 이유 때문에 미디에이터 시스템의 연구가 진행되어오고 있는데 기존의 미디에이터 시스템은 데이터의 표현과 질의 처리에 서로 다른 데이터 타입을 사용함으로써 처리가 복잡하고 결과적으로 관리자가 모든 정보를 파악하고 데이터베이스를 통합해야하기 때문에[2,4,5,6] 관리자의 비중이 증가하며 통합된 뷰 관리[2,3,4,5,6]도 수정만 가능하기 때문에 네트워크의 트래픽이 증가되고 처리속도가 증가하게 된다

따라서 본 논문에서는 서로 다른 운영체제에 있는 이질적인 데이터베이스들을 카탈로그의 정보를 통해 관계있는 정보를 관리자에게 제공하여 통합된 뷰의 생성을 쉽게 하며 이를 바탕으로 사용자에게 편리한 정보 검색기능을 제공한다. 또한 관리자의 입력 내용을 통합된 질의문으로 생성하기 때문에 이질적인 데이터베이스에서 사용이 가능하며 수정(modification)으로만 관리가 되었던 기존의 미디에이터 시스템들에 비해 구체화(materialization)로도 뷰를 관리함으로써 네트워크의 트래픽을 줄이고 효율성을 증대화 한다.

* 본 논문은 정보통신부의 '99대학기초연구지원사업의 대학기초연구과제(접수번호C1-1999-1266-00)로 수행되었음

2. 시스템 내용

2.1 각 미디에이터(Mediator)의 비교내용

현재 널리 알려진 미디에이터들의 특징을 살펴보면 다음과 같다. HERMES[4]는 단일 시스템에서만 사용이 가능하며 데이터베이스와 웹과의 연동이 이루어지지 않고, 특정한 툴킷(Toolkit)을 사용하여 데이터베이스에 접근을 한다. DISCO[2]는 이질적인 데이터베이스는 접근이 가능하나 단일 시스템에 존재해야 하는 단점이 있고, 새로운 데이터가 추가될 때마다 관리자가 전체 스키마(global schema)를 재정의 해야한다. TSIMMIS[5,6]는 이질적인 시스템은 지원하나 데이터베이스에 접근하기 위해서 역시 특정한 툴킷을 사용하므로 웹에서의 접근이 까다롭다.

2.2 미디에이터(Mediator)의 구현사항

본 시스템에서 구현하는 미디에이터는 다음과 같은 기능을 지원한다. 첫째는 CORBA를 사용하여 ORB 환경에서 구현했기 때문에 다수의 이질적인 데이터베이스로부터 데이터 검색이 가능하다. 둘째는 데이터의 통합이 동적으로 이루어진다. 이것은 시스템이 운용되고 있는 중에도 글로벌 스키

마에 대한 정보를 자유롭게 변경할 수 있다. 셋째는 사용자의 질의를 분석하여 이질적인 데이터베이스가 알 수 있도록 시스템에서 직접 공통적인 질의문을 생성한다 넷째는 통합 데이터 처리 프로세스를 통해 정보화하여 사용자에게 전달한다. 따라서 사용자는 원하는 결과를 하나의 로컬 데이터베이스에 접근한 것처럼 느끼게 된다. 다섯째는 각기 이질적인(heterogeneous) 데이터 모델에서 래퍼(wrapper)를 교체함으로써 쉽게 통합되고 관리된다. 여섯째는 뷰의 관리를 수정과 구체화의 두가지 방법을 모두 사용한다. 따라서 네트워크의 트래픽을 줄이고 관리의 효율성을 증가시킨다.

2.3 래퍼(Wrapper)의 구현사항

래퍼란 기존의 시스템을 새로운 소프트웨어 아키텍처에서 하나의 객체로 보여지게 하는 미들웨어를 의미한다. 즉, 여기서는 각각의 정보 저장소가 제공하는 기능은 CORBA를 통하여 각 구성요소들에게 접근 가능하도록 해야 하는데 이를 가능하게 해주는 것이 래퍼다[7]

본 시스템에서 구현하는 래퍼는 다음과 같은 네가지의 기능을 한다. 첫째는 데이터 모델링(Data Modeling)으로 이것은 데이터베이스의 정보들을 사용하기 쉽게 정리한다. 둘째는 메소드 호출(Method Invocation)로 Garlic system에서 제안한 기능으로 질의에 관련된 메소드들을 호출한다. 셋째는 질의 계획(Query Planning)으로 미디에이터에게 항상 최상의 정보를 제공하기 위해서 자신의 데이터베이스 내용을 계획한다. 즉, 카탈로그 정보나 조인에 관한 정보등을 관리한다. 넷째는 질의 수행(Query Execution)으로 미디에이터로부터 받은 질의 내용을 데이터베이스에 연결하여 수행한 후 결과를 되돌려 준다.

2.4 뷰 관리 방법

본 시스템에서는 수정과 구체화의 두 가지 뷰 관리 방법을 모두 제공한다. 수정은 사용자의 요구가 들어올 때마다 네트워크를 통해서 각각의 지역 데이터베이스에 접근하여 내용을 가져오는 것이고, 구체화는 미디에이터의 저장소에 뷰의 내용이 저장되어 있어서 바로 결과를 보여 줄 수 있는 방법이다. 알고리즘을 실행하기 전에 다음의 뷰 카탈로그 정보를 갱신하게 된다. 뷰 카탈로그는 총 9개의 필드로 구성되어 있다.

필드 이름	설명	필드 이름	설명
table_name	테이블이름	all_update	전체 갱신 횟수
site	사이트	all_call	전체 호출 횟수
size	용량	type	mat/mo
num_update	갱신 횟수	ratio	비율
num_call	호출횟수		

[표 1] 뷰 카탈로그 정보

그러나 미디에이터의 저장소에는 용량의 한계가 있기 때문에 [알고리즘1]을 통해서 관리 방법을 결정한다

```
현재의 구체화로 되어 있는 가중치(weight)를 구한다
구체화의 가중치 중 제일 작은 것과 현재 테이블의 가중치를 비교
  if (제일 작은 가중치 > 현재 테이블의 가중치)
    현재의 테이블은 수정(modification)
  else
    제일 작은 가중치는 수정(modification)
    관련된 내용 삭제(delete)
    현재의 테이블은 구체화(materialization)
    현재 테이블의 내용 삽입(insert)
  end if
```

[알고리즘 1]

여기서 가중치의 값에 따라 관리 방법이 결정이 되는데 다음과 같은 방법으로 가중치를 구한다. 가중치를 구할 때는 감쇠율(decay rate)[8]를 사용하여 구하며 방법은 다음과 같다

$$P_{t,k} = \sum_{j=1}^{k} (W_{t,j}\ 2^{-b(k-j)}) \quad \text{----- 식(1)}$$

$P_{t,1} = W_{t,1}$ ----- 식(2)

$P_{t,k} = 2^{-b}P_{t,k-1} + W_{t,k}$ ----- 식(3)

$W_{t,k}$ = (호출 횟수/전체 호출 횟수) - (갱신 수/전체 갱신 수) : 테이블의 가중치 ----- 식(4)

(∵ 구체화가 되기 위해서는 사용자의 호출이 많고 테이블의 갱신(update)이 적어야 하기 때문에)

b : 감쇠율(decay rate) (half-life를 30일로 한 경우 ➔ $-\log_2(0.5)/30 = 0.0333$)

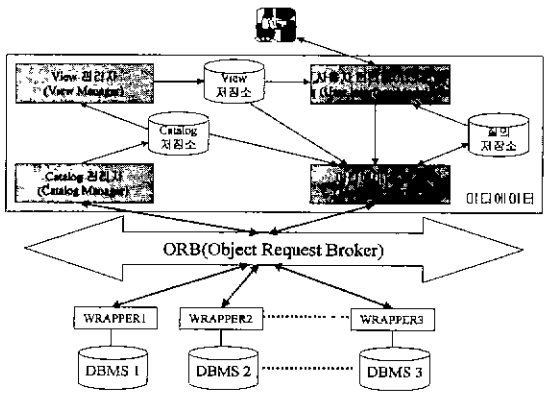
t : table

여기서 감쇠율을 사용한 것은 기간이 지남에 따라 가중치를 다르게 하기 위함이다 즉 half-life가 30이면 30일 이전에 갱신이 되었거나 호출된 것은 생각하지 않는다. 따라서 최근에 사용자의 호출이나 테이블의 갱신이 일어난 것에 가중치를 높게두고 시간이 오래될수록 낮은 가중치를 부여하게 된다.

3. 구현모델

3.1 본 시스템의 구성

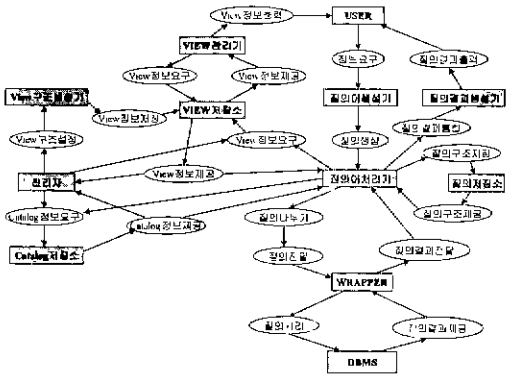
본 시스템의 미디에이터는 카탈로그 관리자, 뷰 관리자, 사용자 인터페이스 모듈, 미디에이터 엔진의 4가지의 모듈로 구분되이 있다. [그림 1]은 본 시스템의 전체 구성도이다.



[그림 1] 시스템 구성도

각 모듈의 기능을 살펴보면 다음과 같다. 첫째 카탈로그 관리자는 연결되어 있는 사이트의 데이터베이스를 검색하여 얻은 카탈로그 정보, 스키마 정보를 받아서 카탈로그 저장소에 저장하고 관리한다. 이 경우 조인이나 합집합, 교집합 등의 연결관계도 같이 조사하여 저장한다. 둘째로 뷰관리자는 사용자가 웹을 통하여 질의를 할 수 있도록 필요한 스키마와 테이블 정보와 설명을 보여 줄 수 있게 사용자 인터페이스 모듈에게 정보를 제공하며, 카탈로그의 정보를 받아서 조인이 가능한 테이블을 중심으로 뷰 구조를 생성한다. 또한 생성된 뷰 구조의 관리를 하기 위한 정보를 생성한다. 셋째로 사용자 인터페이스 모듈은 사용자가 웹 브라우저에 연결이 되면 뷰 관리자로부터 정보를 받아 화면에서 질의를

할 수 있도록 화면을 구성하고 정보를 제공한다. 또한 웹 브라우저를 통한 사용자의 질의를 바탕으로 목표에 맞는 질의어를 생성하고 생성된 질의문을 미디에이터 엔진에 제공하며, 미디에이터 엔진으로부터 최종 질의결과를 받아서 사용자에게 웹브라우저를 통하여 보이준다. 넷째로 미디에이터 엔진은 물리적으로 떨어져있는 데이터베이스에 질의문을 전송하고 래퍼로부터 처리 결과를 받고 넘겨진 결과를 뷰 저장소에 저장하고 질의 결과를 사용자 인터페이스 모듈에 제공한다. 또한, 사용자의 호출이 있거나 데이블의 내용이나 구조가 바뀌었을 때 뷰의 관리를 구체화로 할 것인지 수정으로 할 것인지를 결정한다. 이러한 기능들을 데이터 흐름도(Data Flow Diagram)로 나타내면 [그림 2]와 같다.



[그림 2] 데이터 흐름도

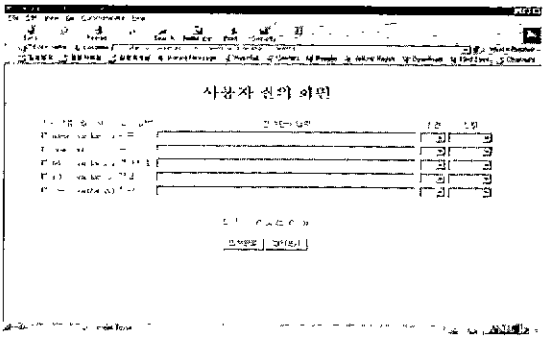
3.2 뷰 정보 생성 및 사용자 인터페이스

뷰관리자는 카탈로그 저장소로부터 테이블의 관련 정보를 얻는다. 조인(join)이나 합집합(union), 교집합(intersection) 등의 관련정보를 토대로 뷰를 생성하게 된다. 모든 뷰에 대해서 [표 2]와 같은 형태의 데이블이 생성되고, 생성된 뷰의 내용은 뷰저장소에 저장이 된다. 각 필드는 조인 가능한 필드 수만큼 생성되고 마지막에 타입(type)이 있는데 이것은 뷰 관리가 수정인지 구체화인지를 나타낸다. 또한 구분(flag)을 두어서 1부터 8까지 각 뷰의 정보가 지정된다. 첫 번째부터 차례대로 필드이름, 필드타입, 필드크기, 필드설명, 테이블이름, 사이트이름, 연관된 테이블정보, 연관된 필드의 정보가 저장이 된다 [표 2]는 뷰 저장소의 생성된 스키마 예제이고 [그림 3]은 [표 2]를 바탕으로 구성된 사용자의 질의 화면이다.

조인 가능한 테이블 수

flag	val	va2	va3	va4	va5	type
1	name	year	tel	school	dept	mo
2	char	int	char	char	char	mo
3	10		20	20	20	mo
4	이름	나이	전화번호	학교	학과	mo
5	person	person	person	private	private	mo
6	165.132.12.145	165.132.12.145	165.132.12.145	165.132.12.139	165.132.12.139	mo
7	private					mo
8	name					mo

[표 2] 뷰 저장소의 뷰 스키마 구조 예제



[그림 3] 사용자 인터페이스

4. 결론

본 논문에서는 분산 데이터베이스를 통합, 관리하기 위한 뷰관리 미디에이터 시스템을 구축하였다. 기존에 관리자가 모든 데이터베이스의 정보를 조사하여 사용자에 알맞은 뷰를 생성했던것에 반해 본 논문에서는 관계성있는 테이블을 중심으로 정보를 생성하여 관리자가 그 정보만으로 알맞은 뷰를 생성할 수 있다. 이를 위해 CORBA 환경을 바탕으로 구현을 했으며 이것은 서로 다른 운영환경과 서로 다른 구현언어를 자유롭게 사용할 수 있는 장점이 있다. 또한 생성된 뷰를 이용하여 사용자의 인터페이스를 구축하고 사용자가 질의한 내용을 바탕으로 통합된 질의문을 자동으로 생성하며 관리자의 기능을 대폭 줄였으며 사용자는 자신의 로컬 데이터베이스에 접근하는 것처럼 결과를 가져 올 수 있다. 기존의 미디에이터 시스템은 통합된 뷰 관리를 수정으로만 했었는데 본 미디에이터 시스템은 구체화까지 가능하게 했다. 이를 구분하기 위하여 감쇠율(decay rate)을 이용하여 가중치를 계산했으며, 구체화를 사용함으로써 네트워크의 트래픽과 오버헤드를 감소시키고 사용자의 질의에 대한 처리 속도가 증가되었으며 저장 공간을 효율적으로 사용할 수 있다. 또한 본 미디에이터 시스템은 각 미디에이터간의 확장이 가능하다

5. 참고 문헌

[1] "Network Management" A. Leinwand and K. F Conroy, Addison-Wesley Publishing Company, Inc pp 17-36, 1996

[2] "Scaling Access to Heterogeneous Data Source with DISCO" Anthony Tomasic, Louiqa Raschid, Patric Valduriez, IEEE Transactions on Knowledge and Data Engineering, Vol 10, No 5, September/October 1998

[3] "A query language for a web-site management system" M F Fernandez, D.Florescu, A Y.Levy, D Suci, SIGMOD Record, vol 26, no 3, Sept 1997

[4] "Hybrid Knowledge Bases", J Lu, A.Nerode, V.S.Subrahmanian, IEEE Transactions on Knowledge and Data Engineering,1994

[5] "Object Fusion in Mediator System" Proceedings of the 22nd VLDB Conference Mumbai(Bombay), India, 1996

[6] "MedMaker A Mediation System Based on Declarative Specifications" 1996 IEEE

[7] Don't Scrap It, Wrap It! A Wrapper Architecture for Legacy Data Sources Mary Tork Roth, Peter Schwarz Proceedings of the 23rd VLDB Conference Athens Greece 1997.

[8] "The NIDES Statistical Component Description and Justification" Harold S Javitz and Alfonso Valdes SRI International Menlo Park, California 94025 March, 1993 pp.1-14