미디에이터 시스템에서의 이질 분산데이타베이스의 통합을 위한 효율적인 뷰관리 방법

(An Effective View Management in a Mediator System for the Integration of Heterogeneous Distributed Database)

주 길 홍 † 이 원 석 ††
(Kil Hong Joo) (Won Suk Lee)

요 약 본 논문에서는 다양한 운영체제와 데이타베이스, 여러 소프트웨어를 포함하는 이질적인 컴퓨팅 환경에서 뷰관리 기능을 제공하는 미디에이터 시스템을 설계한다. 또한 기존의 미디에이터 시스템들이 수정(modification)방법으로만 뷰를 관리했던 것에 비해 구체화(materialization)방법을 병행하는 관리방법을 설계하고, 구체화시키기 위해 필요한 접근방법과 최적화 방법을 제안한다. 이를 위해서 뷰에 대한 접근내역에 감쇄율을 적용함으로써 최근의 접근내역의 변화가 뷰관리에 적절히 반영되도록 설계하였다. 또한 이질적인 환경의 데이타베이스와 운영체제를 극복하기 위해 코바(CORBA)를 웹과 연동하여 사용자가 일반적으로 사용하는 웹브라우저를 통해 뷰를 관리하는 시스템을 구현한다.

Abstract This paper proposes a mediator system which can provide a view management function in the today's heterogeneous computing environment including operating systems, database management systems and other software. While most mediators use view modification mechanism only, the use of view materialization is also considered in this research. In order to optimize the management cost for materialized views, the proposed method applies a decay factor for modeling the access behaviors of views. In other words, the most recently access pattern gets the highest attention in the optimization process. All the necessary jobs for users such as view management are performed on web environment by integrating it with the CORBA.

1. 서 론

컴퓨터의 기술이 발전함에 따라 일반 사용자들의 컴퓨터 활용 빈도와 요구하는 데이타의 양이 급격히 증가되었다. 그러나 현재 컴퓨팅 환경은 기존의 하드웨어와 소프트웨어들이 각기 제한된 범위 내에서는 서로 유용한 기능들을 제공하고 있으나 네트워크 상에서는 서로 독립적으로 운용되고 있기 때문에 효율적으로 상호운용이 이루어지지 않고 있다[1]. 따라서 분산환경에서 시스템의 성능개선이나 유지보수에 많은 비용이 요구되고 있고 서로 다른 운영환경에서 이질적인 데이타 모델들로 구축된 데이타의 통합에 대한 사용자의 요구에 적절히 대응하지 못하고 있는 실정이다. 이런 이유로 분산환경에서 이질적인 데이타베이스의 통합 방법으로 미디에이터 시스템의 연구가 활발히 진행되고 있다.

분산 데이타베이스를 통합하기 위해 뷰를 관리하는 방법으로 수정(modification)방법과 구체화(materialization) 방법을 고려 할 수 있다. 첫째로 수정 방법은 뷰에 대한 질의를 수행하기 위하여 네트워크 환경에서 각 원격지 서버의 테이블에 접근하여 뷰에 대한 데이타를 처리하는 방법이다. 이 방법의 단점은 실행하는데 많은 시간이 소모되고, 뷰에 대한 질의를 수행할 때마다 네트워크를 통하여 원격지 서버를 접근해야 하므로 많은 비용이 필요하며 네트워크의 트래픽과 오버헤드(overhead)가 많이 발생된다. 둘째로 구체화 방법은 처음 질의가 요청되었

† 학생회원 : 연세대학교 컴퓨터과학과
faholo@amadeus.yonsei.ac.kr
†† 종신회원 : 연세대학교 컴퓨터과학과 교수
leewo@amadeus.yonsei.ac.kr
논문접수 : 2000년 10월 9일
심사완료 : 2001년 8월 13일

을 때 미디에이터내의 저장공간에 뷰의 결과를 생성하고 유지하는 방법이다[2]. 따라서 항상 최신정보가 유지되어야 하며 원격지 서버의 기본 테이블에 변경이 일어난 경우 즉시 갱신이 이루어져야 한다. 이 방법은 자주 접근되는 뷰에 대해서 원격지 서버를 거치지 않으므로 네트워크의 트래픽을 감소시키고 사용자에게 빠르게 결과를 보여줄 수 있지만 기본 테이블이 자주 갱신될 때에는 효율적이지 못할 수 있다.

기존의 미디에이터 시스템은 데이타의 표현과 질의 처리에 서로 다른 데이타 타입을 사용함으로써 처리가 복잡하고, 관리자가 모든 정보를 파악하여 데이타베이스를 통합해야 하기 때문에 관리자의 비중이 증가하며 통합된 뷰관리[3, 4, 5, 6, 7]도 수정방법만 지원하여 네트워크의 트래픽과 처리속도가 증가하게 된다. 반면에 구체화방법으로 뷰를 관리하게 될 때에는 빠른 처리속도를 보이고 네트워크의 트래픽이 감소하지만 미디에이터의 저장공간을 효과적으로 활용하는 방법이 고려되어야 한다. 또한 기존의 뷰 관리에서의 구체화방법[8, 9]은 유동적으로 변화할 수 있는 사용자의 접근 형태나 데이타베이스의 갱신에 대한 패턴의 변화를 적절히 반영하지 못하는 단점을 갖는다. 이를 해결하기 위해서 본 논문에서는 감쇄율(decay rate)[10]을 이용하여 사용자의 접근이나 데이타베이스의 갱신에 대한 시점의 차이를 두어 최근에 발생한 사용자의 접근 및 갱신에 더 높은 비중을 부여하여 최적화 관리가 뷰의 활용 형태의 변화에 적절히 반영될 수 있도록 한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 이질적인 환경의 통합방법과 구체화 뷰 연구방법에 대해서 소개하고, 3장에서는 뷰관리 방법과 비용계산 및 최적화 기법에 대해서 설명한다. 4장에서는 3장에서 제안된 내용에 대한 다양한 실험을 수행하고 이에 대한 결과를 분석하며, 마지막으로 5장에서는 최종적인 결론을 맺는다.

2. 관련연구

2.1 이질적인 환경의 통합

서로 이질적인 환경을 가지고 있는 두 개 이상의 정보 시스템들을 통합하는 경우 중점적으로 고려해야 할 사항은 사용자에게 이질적인 환경을 은폐하면서 원격지에 존재하는 시스템들의 독자성을 최대한 유지하는 것이다. 이질적인 환경은 세 가지 측면으로 분류할 수 있다. 첫째는 데이타베이스 관리 시스템, 개발언어, 사용 소프트웨어 등의 소프트웨어적인 환경의 이질성이다. 둘째는 원격지 시스템의 운영체제, 기종 등의 하드웨어적인 것과 같은 시스템 환경의 이질성이다. 셋째는 각각의 정보 시스템이나 활용하고자 하는 시스템에 대한 정보 모델에 관련된 정보 연산의 의미적인 이질성이다.

이러한 이질적인 특성을 극복하기 위한 방법으로 미디에이터와 래퍼(wrapper)의 필요성이 대두되었고[11], 이에 대한 연구가 활발히 진행되고 있다. 미디에이터란 서로 이질적인 하드웨어 환경 및 이질적인 데이타베이스와 소프트웨어 등에 대해서 사용자가 이질적인 환경을 느끼지 못하게 은폐하면서 하나의 시스템에 접근하는 것과 같은 효과를 제공하는 미들웨어(middleware)를 의미하며[12], 래퍼는 기존의 외부에서 활용되는 시스템의 내부기능을 포장하여 외부로 보여주어[13] 미디에이터가 원격지에 있는 서버 기능들의 구동환경에 관계없이 동일한 방법으로 접근할 수 있도록 지원한다. 따라서 래핑 기법을 사용함으로써 기존에 있는 시스템을 변형하지 않고 기존의 시스템을 새로운 형태로 표현할 수 있다는 장점을 가짐으로 원격지 시스템에 대한 독자성이 최대한으로 보장될 수 있다. 또한 이기종 분산 환경에서 시스템간의 분산성을 투명하게 보장하고 시스템 및 응용 프로그램간의 상호 운용을 효과적으로 지원하는 객체지향기술로 CORBA(Common Object Request Broker Architecture)[14, 15, 16] 기반의 분산 컴퓨팅 기술이 있다. CORBA는 기존의 클라이언트-서버 구조가 아닌 클라이언트(Client)-구현객체(Object Implementation) 구조를 가진다.

현재 가장 널리 알려진 미디에이터 시스템인 HERMES, DISCO 및 TSIMMIS의 특징을 살펴보면 다음과 같고, 표 1은 특징을 비교한다.

- HERMES(HEterogeneous Reasoning and MEdiator System)

HERMES는 초창기의 미디에이터 시스템으로 이미지와 텍스트, 그리고 데이타베이스를 통합하는 미디에이터 시스템이며 단일 기종의 시스템에서만 사용이 가능하고, 데이타베이스와 웹과의 연동이 이루어지지 않으나 데이타베이스, 데이타 구조, 그리고 소프트웨어 패키지 등을 통합하는 방법을 제시하였다. 미디에이터는 다른 타입의 내용을 처리하기 위해서 미디에이터 자체적으로 고유의 언어를 통해 컴파일을 하고 생성되는 툴킷(toolkit)을 사용하여 각 사이트에 있는 데이타베이스에 접근한다. 또 하나의 단점은 미디에이터 관리자가 미디에이터의 언어에 기반하여 연결된 서버와 데이타베이스에 필요한 입력형식과 출력형식을 일일이 열거해야하며 뷰관리 방법은 수정방법만 사용한다[5, 17].

- DISCO(Distributed Information Search COmponent)

DISCO는 자바를 사용하여 미디에이터를 구현함으로써 기존의 CGI가 가지고 있던 문제점 중 많은 동시 요

구가 발생했을 때 나타날 수 있는 시스템 자원의 부족과 성능 저하문제를 해결하였으며, 응용프로그램은 자바 애플릿을 사용하여 구현함으로써 웹에서의 접근을 손쉽게 했다. 따라서 이질적인 데이타베이스의 접근이 가능하나 HERMES와 동일하게 단일 기종의 시스템에서만 가능하다. DISCO는 데이타베이스 관리자가 새로운 내용이 생기거나 미디에이터와 연결이 될 때마다 미디에이터에서 사용하는 미디에이터의 스키마 정보, 데이타 자원의 집합, 데이타 자원을 위한 지역 서버의 스키마 정보, 미디에이터 스키마와 지역 서버 스키마의 매핑(mapping) 정보를 새롭게 정의하며 뷰관리 방법은 수정방법만 사용한다[3, 18].

• TSIMMIS

TSIMMIS는 CORBA를 사용하여 미디에이터 시스템을 ORB환경에서 구축하였기 때문에 이질적인 시스템을 지원하며 또한 이질적인 데이타베이스에도 접근이 가능하다. 그러나 미디에이터에서 전체 스키마와 뷰를 MSL (Mediator Specification Language)로 따로 정의를 하고, 래퍼에서는 WSL(Wrapper Specification Language)로 각 서버의 정보를 따로 정의하기 때문에 MSL과 WSL을 매핑시키는 작업이 필요하고, 웹을 통해서 데이타베이스에 접근하는 것이 아니라 특정한 툴킷과 자체 언어를 통하여 데이타베이스에 접근이 가능하며 수정방법으로만 뷰관리를 한다[6, 7].

표 1 기존의 미디에이터 시스템 비교

	HERMES	DISCO	TSIMMIS
이질적인 시스템에 접근	×	×	○
이질적인 데이타베이스에 접근	△	○	○
웹과 연동	×	△	△
뷰관리	modification	modification	modification
분산환경	단일 기종 시스템	단일 기종 시스템	CORBA

2.2 구체화 뷰 연구방법

구체화 뷰에 관한 연구는 다양하게 이루어지고 있다. 구체화 뷰 방법은 먼저 뷰를 수행하여 그 결과를 물리적으로 데이타베이스에 저장하고, 뷰에 대한 질의를 이미 저장된 검색 결과에 대하여 직접 수행하는 방법이다. 따라서 수정방법에 비하여 뷰에 대한 질의 수행속도가 빠르고, 효율성이 높다.

• 구체화 뷰를 이용한 질의 처리 방법

[19, 20, 21, 22, 23]에서는 집계함수를 가정하지 않은 합접질의(conjunctive query)에 대한 질의 재구성 방법을 제안하였다. 특히 [22, 23, 24]은 구체화 뷰만으로 질의를 재구성하는 방법들을 제안하였는데, 이 기법들은 기본 테이블의 접근이 어려운 분산환경에 쉽게 적용될 수 있다. [19, 20, 22]에서는 질의 최적화 모듈에 구체화 뷰를 이용한 질의 재구성 기법을 통하여 최적으로 재구성된 질의를 찾아내는 문제를 다루고 있다. 또한 [21]에서는 포함 대응을 고려하여 가능한 모든 질의를 탐색함으로써 대상 질의를 찾는 문제가 NP-complete임을 증명하였으며, [25]에서는 일반화된 프로젝션(Generalized Prjection: GP)에 기반하여 질의 트리를 이용한 재구성 방법을 제안하였다. 그러나, GP는 SQL 모델에 비해 제한적이므로 SQL이 지원되는 시스템에서는 제한적으로 이용될 수밖에 없다.

지금까지의 언급한 모든 기법들은 다음과 같은 제약점이 존재한다. 첫째로 구체화 뷰가 각 질의에 대해서 일대일 대응(1-to-1 mapping)의 관계가 성립할 경우만을 고려하였다. 따라서, 구체화 뷰가 질의에 언급되지 않은 테이블을 참조하거나 질의 수행에 필요한 속성을 포함하지 않는다면 그 구체화 뷰는 고려대상에서 제외되었다. 둘째로 하나의 구체화 뷰에 질의 결과가 포함되어있는 다른 뷰가 생성된 경우 지역서버에 접근하여 결과를 제공하지 않아도 되는데도 불구하고 지역서버에 접근하여 결과를 가져오는 단점이 있다. 따라서 이용 가능한 구체화 뷰를 제한적으로 탐색함으로서 정확한 결과를 사용자에게 제공하기 어렵다.

• 관계형 구체화 뷰 관리와 객체지향 구체화 뷰 관리

구체화된 뷰는 뷰에 대한 모델링 방법에 따라 관계형 뷰와 객체지향 뷰로 나눈다. 관계형 뷰 구체화에 대한 연구[26]에서는 뷰 구체화를 위하여 뷰 테이블에 기본 테이블의 레코드 값을 복사하여 유지하는 방법을 제안하였다. 이 방법은 뷰 테이블에서 값들을 직접 저장하기 때문에 질의 처리속도가 빠르나 기본 테이블 튜플과 뷰 튜플간에 일관성 및 관련성을 유지하기가 어렵다. 만약 뷰 테이블간에 기본 키가 없는 경우나 기본 키의 값이 변경되는 경우에는 기본 테이블의 변경에 따른 뷰 테이블의 튜플을 변경하기가 매우 어렵다. 따라서 기본 테이블의 튜플이 변경된 경우 관련된 뷰 튜플들을 변경하기가 매우 어렵다[27, 28].

객체지향 뷰의 구체화 방법은 두 가지가 있다[29, 30]. 첫 번째 방법은 뷰 객체를 구체화시킬 때 실제 데이타를 저장하는 방식이다. 이 방법은 관계형 뷰 구체화 방법과 유사하며, 뷰 객체에 실제 데이타가 저장되어 있으

므로 뷰에 대한 질의 수행시 속도가 빠르다. 그러나 뷰 객체와 소스 객체간에 중복된 데이타에 대한 일관성유지 문제가 발생한다. 두 번째 방법은 구체화된 뷰 객체에 데이타가 아닌 소스 객체의 식별자를 저장하는 방식이다. 이 방법은 객체 식별자를 이용하여 뷰 객체와 소스 객체간의 데이타를 공유하며, 데이타의 중복이 없으므로 데이타의 일관성 유지가 쉽다. 그러나 뷰 객체를 검색하기 위하여 소스 객체를 추가로 접근해야 하는 오버헤드가 있다.

3. 미디에이터의 뷰 관리 방법

3.1 미디에이터 시스템 구조

본 논문에서 제안하는 미디에이터는 그림 1과 같이 카탈로그 관리자와 뷰관리자, 사용자 인터페이스 모듈, 그리고 미디에이터 엔진의 4가지 모듈과 카탈로그 저장소, 뷰 저장소 및 질의저장소의 3가지 저장소로 구성된다. 카탈로그 관리자와 뷰 관리자는 미디에이터를 위한 관리자 모듈이고 사용자 인터페이스 모듈과 미디에이터 엔진은 사용자 질의를 처리하는 모듈이다[31].

카탈로그 관리자(catalog manager)는 해당 사이트의 정보를 카탈로그 저장소에 저장하고 관리한다. ORB 환경에 연결되어 있는 각 사이트의 분산 데이타베이스를 검색한 후에 각 데이타베이스의 카탈로그 정보와 테이블 스키마 정보를 조사하여 미디에이터의 카탈로그 저장소에 저장한다. 뷰 관리자(view manager)는 사용자가 웹을 통하여 분산데이타베이스에 대한 새로운 뷰를 손쉽게 정의할 수 있도록 각 스키마에 대한 정보를 검색할 수 있게 지원하며, 기존에 정의된 뷰에 대한 내용을 효과적으로 검색하고 갱신할 수 있도록 지원한다. 또한 미디에이터에서 뷰를 관리하는데 필요한 다양한 정보들을 뷰 카탈로그 테이블과 뷰 구조 테이블로 관리한다. 사용자 인터페이스 모듈(user interface module)은 사용자가 웹 브라우저를 통해 뷰 관리자로부터 정보를 받아 원하는 뷰를 선택하여 질의를 표현하는 화면을 구성하고, 사용자의 질의에 대해 질의문을 생성하여 미디에이터 엔진에 전달하며, 원하는 질의결과를 사용자가 볼 수 있도록 웹 브라우저로 제공한다. 미디에이터 엔진(mediator engine)은 사용자 인터페이스로부터 받은 질의문을 분석하여 각각의 지역 서버에 적합하게 질의문을 분해한 후 물리적으로 떨어져있는 원격지 래퍼에게 질의문을 전송한다. 그후 각 래퍼로부터 수신된 처리결과를 결합하여 질의결과를 생성하고 사용자 인터페이스 모듈에 제공한다. 이때 구체화 방법으로 관리하는 질의 결과를 뷰 저장소에 저장하여 동일한 질의를 처리할 때 활용될 수 있도록 한다.

본 논문의 미디에이터 시스템의 특징은 크게 여섯 가지로 나누어 볼 수 있다. 첫째는 구체화방법으로 뷰를 관리할 수 있도록 하여 각 뷰의 활용형태에 따라 적합한 관리 방법으로 뷰에 대한 질의를 효과적으로 지원할 수 있다. 둘째로 미디에이터의 버퍼 관리를 위해서 접근내역에 대해 감쇄율을 적용한 최적화 방법을 사용하여 사용자의 접근 빈도의 변화에 유연하게 적응할 수 있도록 관리한다. 셋째는 최적화방법을 위해서 미디에이터에 임시버퍼를 두어 활용함으로써 빈번하게 발생하는 불필요한 최적화 과정의 수행을 방지한다. 넷째는 통합 데이타 처리 프로세서를 사용하여 이질적인 운영체제에서 서로 다른 데이타베이스들을 인식할 수 있는 환경을 제공한다. 다섯째는 데이타의 통합이 동적으로 이루어지기 때문에 시스템이 운용되고 있는 중에도 전체 스키마에 대한 정보를 자유롭게 변경할 수 있다. 여섯째는 CORBA를 사용하여 ORB환경에서 구축하고 웹으로 데이타베이스에 접근이 가능하도록 함으로써 특별한 툴킷이 없이 일반 브라우저를 통해 이질적인 환경에서 원하는 질의를 표현하고 결과를 받아 볼 수 있다. 본 논문에서는 뷰관리자와 미디에이터 엔진에서 효과적으로 뷰관리를 하는 최적화기법을 제안하고 이에 대한 실험 및 분석결과를 제시한다.

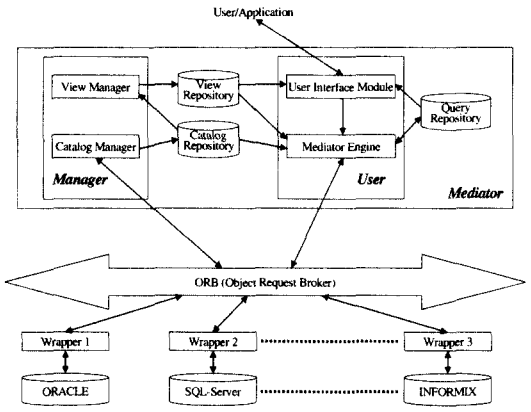

그림 1 미디에이터 시스템 구조

3.2 뷰 관리 방법

각 뷰에 대한 효과적인 관리방법을 결정하기 위해서는 뷰에 대한 접근 및 갱신 형태를 정확하게 파악해야 한다. 이때 미디에이터에 정의된 뷰는 해당 뷰를 구성하는 원격지 서버단위로 구분된 여러개의 부분질의들로 분해할 수

있으며, 뷰에 대한 접근형태는 각 뷰를 구성하는 부분질의에 대한 접근 형태로 표현되고, 명목접근회수(nominal access frequency)와 유효접근회수(effective access frequency)로 모델링된다. 명목접근회수는 부분질의 내용을 접근하는 회수로써 부분질의를 수행하는 회수이고, 유효접근회수는 미디에이터가 부분질의의 결과를 원격지 서버로부터 실제로 전송받는 회수이다. 따라서 수정방법으로 관리되는 부분질의에서는 명목접근회수와 유효접근회수는 동일하나 구체화방법으로 관리되는 부분질의에서는 유효접근회수는 부분질의를 구성하는 원격지 서버의 기본테이블에 갱신이 일어난 회수로 표현될 수 있고, 명목접근회수와 유효접근회수의 차가 미디에이터내에 저장되어있는 부분질의로부터 결과를 얻는 회수가 된다.

사용자의 질의가 발생하면 부분질의로 분해하고, 분해되어진 부분질의의 명목접근회수를 증가시킨다. 그리고 미디에이터가 부분질의를 수정방법으로 관리하는 경우는 사용자의 질의가 발생할 때마다 미디에이터에 내용이 저장되어 있지 않기 때문에 해당 부분질의의 유효접근회수를 증가시키고, 원격지 서버로부터 부분질의의 내용을 전송 받아 질의결과를 사용자에게 제공한다. 또한 미디에이터가 부분질의를 구체화방법으로 관리하는 경우는 미디에이터에 부분질의 결과가 있기 때문에 원격지 서버를 거치지 않고 신속하게 질의결과를 제공할 수 있으며, 이 경우는 해당 부분질의의 유효접근회수의 값은 변화가 없다. 그러나 부분질의의 대상이 되는 기본테이블이 갱신 된 경우에는 새롭게 갱신된 내용을 사용자에게 제공하여야 하기 때문에 갱신이 될 경우는 미디에이터에서 갱신 이전의 부분질의 결과를 삭제하고, 갱신 후에 해당 부분질의의 기본테이블에 대한 첫번 사용자의 접근이 발생할 때 해당 부분질의에 대한 유효접근회수를 증가시키고, 원격지 서버로부터 부분질의의 결과를 제공받아 사용자에게 전달하고, 미디에이터에 저장한다. 그림 2는 미디에이터에서 관리되는 부분질의에 대한 처리 흐름도를 보여주고 있다.

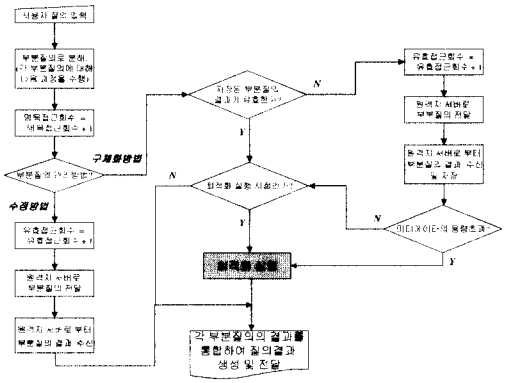

그림 2 부분질의 접근처리 흐름도

미디에이터는 구체화방법으로 관리하는 부분질의의 결과를 저장하는 저장공간이 초과할 경우마다 최적화를 수행하기 때문에 빈번히 발생하는 불필요한 최적화를 방지하기 위해서 임시버퍼를 두어 저장공간의 부족으로 인한 최적화 실행회수를 최소화한다. 따라서 미디에이터가 구체화방법으로 관리하는 부분질의의 새로운 결과를 원격지 서버로부터 전송받았을 때 미디에이터의 저장공간을 초과하였다면 미디에이터는 임시버퍼를 사용하여 질의 결과를 처리하고, 임시버퍼의 용량을 초과 할 경우 최적화를 수행하여 각 부분질의의 관리방법을 재결정한다. 그러나 부분질의의 새로운 결과가 기존의 결과보다 작아서 저장공간에 여유가 생기면 수정방법으로 관리되는 부분질의 중 순위가 높고 저장공간에 알맞은 부분질의의 관리방법을 구체화방법으로 변경한다.

만약 구체화로 관리되고 있는 부분질의에 대해서 원격지 서버에서 갱신이 일어나는 순간 미디에이터로부터 사용자가 질의 결과를 얻었다면 사용자는 부정확한 결과를 얻게 된다. 이러한 문제의 해결방법은 두 가지가 있다. 첫째는 미디에이터에 연결되어 있는 원격지 서버들의 기본테이블에 대한 모든 갱신을 미디에이터를 통해서만 수행하도록 하는 방법이다. 이 방법은 사용자에게 정확한 결과를 제공할 수 있으나 지역서버의 자치성이 저하된다. 둘째로 서버에서 부분질의의 갱신이 일어났을 경우 갱신하는 트랜잭션이 완료되기 전에 미디에이터에게 갱신된 사실을 알려주는 방법이다. 이 방법은 정확한 결과를 제공하고 지역서버의 자치성도 보장한다.

3.3 뷰 관리 최적화 기법

구체화방법으로 관리되는 부분질의는 유효접근회수 만큼 원격지 서버에서 부분질의의 처리내용을 전송받으며, 미디에이터 버퍼에 내용이 있는 경우에는 네트워크 전송과 원격지 서버의 부분질의 처리과정이 필요없이 미디에이터내에 저장된 결과를 활용한다. 따라서 유효접근회수 만큼 원격지 서버에서 부분질의 결과를 전달받아 미디에이터 버퍼에 저장하고 명목접근회수 만큼 사용자에게 결과를 전달한다. 반면에 수정방법으로 관리되는 부분질의는 매번 사용자의 접근이 일어날 때마다 원격지 서버에 접근하여 부분질의 결과를 조합한 후 미디에이터가 내용을 전송받아 사용자에게 결과를 전달한다.

부분질의를 수정방법으로 관리할 경우의 비용은 각 부분질의 별로 명목접근회수만큼 원격지 서버에서 주어진

부분질의를 처리하여 결과를 산출하고, 네트워크를 통해서 미디에이터에게 전달하며, 이를 미디에이터에서 읽어서 사용자에게 결과를 제공하는 비용이므로 식(1)과 같이 표현할 수 있다. 이때 C_i는 i번째 부분질의에 대한 원격지 서버의 처리비용을 나타내고, N_i는 i번째 부분질의의 네트워크 전송량을 부분질의의 용량으로 나타내며, b_i는 i번째 부분질의의 결과를 읽는 비용을 나타낸다. 또한 f_i^{ma}는 i번째 부분질의의 명목접근회수이다.

부분질의를 구체화방법으로 관리할 경우의 비용은 유효접근회수만큼 원격지 서버에서의 처리비용이 필요하며 명목접근회수와 유효접근회수의 차이만큼 미디에이터내에 저장된 부분질의 결과를 활용하므로 식(2)와 같이 표현될 수 있다. 이때 f_i^{ea}는 i번째 부분질의의 유효접근회수이다.

$$\text{수정관리비용 } S_i^{\mathrm{mod}} = (C_i + N_i + b_i) f_i^{ma} \quad (1)$$

$$\text{구체화관리비용 } S_i^{mat} = (C_i + N_i + b_i) f_i^{ea} + b_i (f_i^{ma} - f_i^{ea}) \quad (2)$$

부분질의의 관리방법은 최적화가 이루어진 후에 결정이 되기 때문에 부분질의 비용과 뷰 비용, 그리고 전체 비용도 최적화가 이루어진 후에 계산된다. 뷰 비용은 식(3)과 같이 뷰를 구성하는 부분질의의 처리비용과 각 부분질의의 결과를 조합하는 비용 M_c을 합한 것이다.

$$V_j = \sum_{i=1}^{l} S_i + M_c \quad (l : \text{뷰 } V_j\text{의 부분질의의 개수}) \quad (3)$$

이때 미디에이터내에서 각 부분질의를 조합하는 비용인 M_c는 각 부분질의의 관리방법에 관계없이 동일한 비용이 소모되므로 최적화할 때는 고려하지 않는다. 미디에이터가 모든 뷰를 관리하는 전체 비용은 식(4)에 나타난 대로 각 뷰 비용의 합으로 정의되며, 최적화는 모든 뷰를 관리하는 비용인 전체 비용 T가 최소화되도록 하는 각 부분질의의 관리방법을 설정하는 것이다.

$$T = \sum_{j=1}^{m} V_j \quad (m : \text{뷰의 개수}) \quad (4)$$

기존의 미디에이터 시스템들[3, 5, 6, 7, 8, 9, 17, 18]은 수정방법으로만 뷰를 관리하고, 뷰에 대한 비용을 계산할 때 갱신회수와 접근회수만을 사용하여 가중치(weight)를 계산하며 과거회수와 현재회수의 비중을 동일하게 간주함으로써 뷰에 대한 최근의 활용형태를 적절하게 반영하지 못하였다[8, 9]. 이러한 문제를 해결하기 위해 본 논문에서는 과거에 일어난 사용자의 접근이나 갱신보다 최근에 일어난 사용자의 접근이나 갱신에 보다 높은 비중을 두기 위해 감쇄율을 적용한다. 이것은 사용자의 접근이나 부분질의의 갱신에 대해 half_life[10]를 두어 이전회수와 최근회수에 차이를 두는 방법으로 half_life란 사용자의 접근이나 부분질의의 갱신이 영향을 1/2만큼 주는 시점을 말한다. 예를 들어 사용자의 접근이나 부분질의의 갱신에 대한 영향률이 50%가 되는 위치(half_life)를 1일 전이라고 가정을 하면 2일전에 발생한 접근회수가 현재시점에 주는 영향률이 25%가 되고, 3일전에 발생한 접근회수가 현재시점에 주는 영향률은 12.5%가 된다. 따라서 k일 이전에 발생한 접근회수는 현재시점에 2^{-k}*100%의 영향률을 주게 된다. 이를 수식으로 정리해서 나타내면 식(5)과 같다

$$\text{감쇄율 } d = -\frac{\log_2(0.5)}{half\_life} \quad (5)$$

각 부분질의의 가중치는 부분질의의 접근회수를 갱신된 회수로 나누어서 계산을 한다. 따라서 식(6)에서와 같이 이전 최적화가 발생했을 때의 명목접근회수와 최근 명목접근회수와의 차이를 이전 최적화가 발생했을 때의 유효접근회수와 최근 유효접근회수와의 차이로 나눈 값이 감쇄율을 적용하기 전의 i번째 부분질의 가중치인 W_i가 되며, 이 가중치에 식(7)과 같이 감쇄율을 적용시켜 갱신이나 사용자의 접근이 일어난 시점의 차이를 계산한다. 따라서 i번째 부분질의에 대해 감쇄율이 적용된 가중치 P_i를 구할 수 있다. 여기서 $f_i^{(new)}$은 i번째 부분질의의 현재 새로 발생한 최적화 시점이고, $f_i^{(old)}$는 i번째 부분질의의 가장 최근에 발생한 최적화 시점을 나타낸다. 따라서 이를 사용하여 i번째 부분질의의 가중치인 W_i를 구할 수 있다. 또한 i번째 부분질의의 현재 새로 발생한 최적화 시점의 가중치는 P_i^{new}로 나타내고, i번째 부분질의의 이전에 발생한 최적화 시점은 P_i^{old}로 나타낸다.

$$W_i = \frac{\Delta f_i^{ma}}{\Delta f_i^{ea}} = \frac{f_i^{ma(new)} - f_i^{ma(old)}}{f_i^{ea(new)} - f_i^{ea(old)}} \quad (6)$$

$$P_i^{new} = 2^{-d} P_i^{old} + W_i \quad (7)$$

모든 부분질의를 구체화방법으로 관리할 경우 미디에이터 저장 공간의 한계성 때문에 모두 미디에이터에 저장할 수 없으므로 저장공간을 가장 효율적으로 활용할 수 있는 부분질의를 구체화 시켜야 한다. 따라서 갱신회수가 상대적으로 적으면서 사용자의 접근이 빈번하게 발생하는 부분질의일수록 구체화방법으로 관리하는 것이 더욱 효율적이다.

미디에이터는 저장 공간을 효율적으로 활용하기 위해서 전체 뷰 관리비용에 대한 최적화 시기 및 이에 대한 각 부분질의의 관리방법을 설정하는 것이 중요한 요소이다. 따라서 주기적으로 전체 뷰 관리비용을 최적화하는 과정을 수행하여 최적화 결과에 따라 각 부분질의의 관리 방법을 설정한다. 최적화를 시킬 시점이 되면 이전

최적화과정 수행 이후의 명목접근회수와 유효접근회수를 이용하여 각 부분질의의 관리비용을 계산한다.

본 논문에서는 제시하는 알고리즘을 통해 구해진 최적비용을 *최적화비용*이라고 정의하고, 관리되고 있는 각 부분질의를 모두 조합하여 실제로 구한 최적비용을 *실제최적비용*이라고 정의한다. 따라서 실제최적비용을 구하여 적용하는 것이 가장 좋은 방법이지만 이는 많은 시간이 소모된다. 만약 부분질의의 개수가 n개라면 총 조합의 수가 $n!$만큼 걸리기 때문에 시간복잡도가 $O(n!)$이 된다. 따라서 부분질의의 개수가 많아질수록 실제최적비용을 계산하는데 많은 시간이 소모되므로 최적화시점에 현실적으로 적용할 수 없기 때문에 빠른 시간에 실제최적비용에 근사한 값을 찾는 것이 중요하다.

본 논문에서 최적화비용은 각각의 부분질의가 수정방법으로 관리될 때의 비용과 구체화방법으로 관리될 때의 비용을 모두 계산하여 두 비용의 차에 감쇄율을 이용한 가중치를 적용하여 계산한다. 따라서 이 값이 클수록 부분질의를 구체화방법으로 관리하는 것이 전체비용을 최소화할 수 있으며, 이 경우 시간복잡도는 $O(2n)$이다. 최적화비용은 $O_c = \sum_{i=1}^{n} (S_i^{\mathrm{mod}} - S_i^{mat}) P_i^{new}$으로, 식(1),(2),(6),(7)에 의해 식(8)과 같이 계산된다.

$$\begin{aligned} O_c &= \sum_{i=1}^{n} (S_i^{\mathrm{mod}} - S_i^{mat}) P_i^{new} \\ &= \sum_{i=1}^{n} (C_i + N_i)(f_i^{ma} - f_i^{ea})[(2^{-d} P_i^{old}) + \frac{\Delta f_i^{ma}}{\Delta f_i^{ea}}] \\ &= 2^{-d} \sum_{i=1}^{n} (C_i + N_i)(f_i^{ma} - f_i^{ea}) P_i^{old} \\ &\quad + \sum_{i=1}^{n} (C_i + N_i)(f_i^{ma} - f_i^{ea}) \frac{\Delta f_i^{ma}}{\Delta f_i^{ea}} \end{aligned} \quad (8)$$

최적화 과정은 많은 시간이 필요할 수 있으므로 가장 적절한 최적화 시점을 찾는 것이 중요하다. 수정 방법으로 관리되는 부분질의는 명목접근회수가 유효접근회수보다 클수록 관리방법을 구체화 방법으로 변경하는 것이 더 효율적이고, 구체화 방법으로 관리되는 부분질의는 유효접근회수가 명목접근회수에 근접할수록 관리방법을 수정 방법으로 변경하는 것이 더 효율적이다. 따라서 전체 부분질의 중 mo개의 부분질의가 수정방법으로 관리되고 있고 ma개의 부분질의가 구체화 방법으로 관리되고 있다면 최적화 시점은 식(9)과 같이 표현된다. 이때 k값은 전체부분질의를 처리하는 비용의 배수로 설정할 수 있으며, 관리자가 k의 값을 변화시켜 가면서 최적화 주기를 조절할 수 있다. 일반적으로 k값을 작게 설정할수록 최적화를 자주 실행하여 실제최적비용에 근접하관리하며, k값을 크게 설정할수록 최적화과정의 수행주기가 길어지게 되고, 실제최적비용과의 비용차가 커질 수 있다.

$$\sum_{i=1}^{mo} (\Delta f_i^{ma} - \Delta f_i^{ea}) S_i^{\mathrm{mod}} + \sum_{j=1}^{ma} (\Delta f_i^{ea} - \Delta f_i^{ma}) S_i^{mat} \geq k \quad (t = ma + mo) \quad (9)$$

t : 전체 부분질의 수, ma : 구체화방법 부분질의 수, mo : 수정방법 부분질의 수

또 한가지 고려해야 할 점은 부분질의의 포함관계가 있는 경우이다. 기존의 미디에이터 시스템들은 각 뷰를 독립적으로 관리하는 반면에 본 논문에서는 뷰의 부분질의 단위로 관리함으로써 동일한 부분질의가 여러 개의 뷰에 중복되어 나타날 경우 해당 부분질의를 효과적으로 처리 할 수 있다. 예를 들어 부분질의 *S1*의 결과가 부분질의 *S2*의 결과에 포함되어 있고, 부분질의 *S2*의 결과가 *S3*에 포함이 되어 있으면 $S1 \subset S2 \subset S3$의 관계가 생긴다. 이 경우 *S3*가 구체화로 관리되고 있다면 *S1*과 *S2*는 *S3*에 종속되어 *S1*과 *S2*의 결과를 원격지 서버에서 가지고 오는 것이 아니라 *S3*에서 바로 질의를 할 수 있기 때문에 비용의 감소와 시간의 절약을 가져올 수 있다.

만약 미디에이터가 구체화 방법으로 관리하는 부분질의의 기본 테이블이 갱신되어 갱신된 내용에 대한 부분질의의 결과가 기존의 결과보다 많은 용량이 필요하게 되면 최적화 과정을 실행하게 된다. 그러나 적은 양의 용량초과는 부분질의의 관리 방법이 변경되지 않을 확률이 높으므로 불필요한 최적화 과정을 수행할 수 있다. 따라서 이러한 불필요한 최적화 과정을 방지하기 위해서 미디에이터에 임시 버퍼를 두어 적은 양의 용량초과로 인해 발생하는 최적화 과정의 수행을 방지한다. 이와반대로 구체화 방법으로 관리하는 부분질의의 기본 테이블이 갱신되어 기존의 결과보다 갱신된 내용에 대한 부분질의의 결과가 적은 양을 갖게되면, 수정 방법으로 관리되고 있는 부분질의 중 구체화 방법으로 관리되어져야 하는 순위가 높은 순서로 부분질의 결과 용량이 미디에이터의 빈 용량에 알맞은 부분질의의 관리방법을 구체화 방법으로 변경시킨다.

예를 들어 7개의 부분 질의가 3개의 뷰에 연결되어 있고, 미디에이터 버퍼의 용량이 100MB이며 그중 20%인 20MB가 임시 버퍼 용량이라고 할 때 표 2는 최적화를 실행시켜야 할 시점에서 최적화가 되는 과정을 보인다. 여기서 비용의 단위는 블록이다.

표 2는 부분질의 별로 유효접근회수와 명목접근회수를 나타내고, 이를 바탕으로 구체화방법 비용과 수정방법 비용이 계산되어짐을 보여준다. 디스크 입출력 속도에 비해 네트워크 전송속도의 차이가 100배라고 가정하고, 아직 관리방법이 결정되지 않았기 때문에 구체화방법으로 관리할 때의 비용과 수정방법으로 관리할 때의 비용

표 2 최적화 실행과정 예제

뷰	부분질의	용량	부분질의 처리비용	명목 접근회수	유효 접근회수	구체화 관리비용	수정 관리비용	수정비용 - 구체화비용	관리방법
V1	S1	22	30	10	3	6910	22520	15610	구체화
V1	S2	27	33	9	5	13908	24840	10932	수정
V2	S3	36	45	12	3	11367	44172	32805	구체화
V2	S4	15	26	5	2	3127	7705	4578	수정
V2	S5	32	40	9	4	13248	29448	16200	구체화
V3	S6	20	32	6	5	10292	12312	2020	수정
V3	S7	18	28	8	2	3800	14768	10968	수정

을 모두 계산하여 계산된 관리방법의 비용을 통해 수정 방법의 비용과 구체화방법의 비용의 차가 큰 순서로 구체화방법으로 관리한다. 이때 미디에이터 버퍼의 용량을 계산해야 하며, 여기서는 80MB로 가정을 했으므로 순위가 높은 S3, S5, S1은 구체화 방법으로 관리가 되고, S7, S2, S4, S6은 수정 방법으로 관리가 되어진다.

4. 실험 및 결과 분석

본 논문의 미디에이터 시스템에 대한 실험은 이질적인 하드웨어를 실험하기 위해 UNIX 환경인 Solaris 2.6과 NT환경인 Windows NT Server 4.0을 사용하여 구축하였고, 이질적인 데이타베이스를 위하여 Solaris에서는 DBMS를 Oracle 8.1을 사용하고 Windows NT 에서는 SQL-Server 6.5를 사용하였다. 또한 CORBA가 제공하는 ORB환경에서 구축하기 위해 OrbixWeb 3.1을 사용하였고, 자바언어를 사용하여 구현하였다. 실험 데이타는 사용자가 접근할 수 있는 환경을 구성하여 부분질의와 뷰를 생성하고, 시스템에서 100일 동안 사용자의 접근과 부분질의의 갱신이 임의로 발생되도록 하여 평균 500KB의 부분질의 100개에 대해 약 7GB의 로그를 생성하였다. 또한 부분질의의 개수를 임의로 변화하게 하였으며, 부분질의의 용량변화는 각 부분질의마다 최소 1/2배에서 최대 2배 사이에서 자유롭게 변화하게 하였다. 또한 최적화실행을 위한 식(9)의 k값은 이전 최적화를 실행하였을 때의 전체부분질의 비용으로 설정하였다.

다음의 그림 3은 동일한 조건에서 half_life에 따른 비용의 변화를 보여준다. 본 실험은 half_life를 0, 7, 15, 30, 60, 100으로 나누어서 실험을 하였고, 부분질의 관리 실험일수를 100일로 하였다. Half_life가 0인 것은 감쇄율를 적용하지 않은 것을 나타내며 감쇄율을 적용하지 않으면 기존의 미디에이터들과 같이 모든 부분질의가 사용된 시기에 관계없이 동일한 비율로 적용이 되기 때문에 그림 3에서 보듯이 최적화비용이 가장 크고 실제최적비용과 많은 차이가 있음을 알 수 있다. Half_life는 관리자가 원하는 값으로 정의 할 수 있으며 시스템의 성능과 사용자의 부분질의에 대한 접근형태의 변화정도에 따라 설정할 수 있다. 본 실험에서는 모든 데이타를 임의로 생성하였기 때문에 평균적으로 half_life를 60으로 하였을 때 가장 효율적인 최적화비용을 얻을 수 있고, 실제 최적비용에 가장 가까이 접근하였지만 주기적으로 접근형태가 변화될 경우 해당 주기를 반영하여 더욱 효과적으로 최적화 시점을 찾을 수 있다.

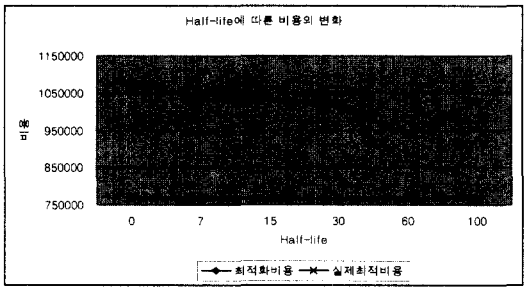

그림 3 Half-life에 따른 비용의 변화

미디에이터는 버퍼의 용량이 초과되면 최적화 과정을 수행하게 된다. 따라서 그림 4에서는 미디에이터가 가지고 있는 버퍼 중에서 임시버퍼를 달리하면서 동일한 부분질의의 접근 및 갱신에 대해 최적화 알고리즘 실행횟수를 비교하였다. 임시버퍼의 비율이 0%인 것은 임시버퍼가 없는 것을 나타내며, 임시버퍼의 비율을 5%, 10%, 20%로 증가시키면서 실험을 하였다. 따라서 그림 4는 미디에이터 저장공간의 용량초과로 인한 최적화 알고리즘 수행횟수를 나타낸다. 전체 부분질의를 25, 50, 75, 100개로 실험을 하였으며 각 부분질의의 결과 크기는 갱신될 때마다 100KB에서 1000KB 사이에서 임의로 변화시켰다. 총 100일 동안의 로그를 가지고 작업을 하였으며 전체 로그의 크기는 약 7GB였고, 총 5번 실험을 한 결과의 평균값으로 나타내었다. 임시버퍼가 0%일 때는

많은 수의 용량초과가 생겨서 최적화 알고리즘을 자주 실행하여야 하나 임시버퍼의 용량을 증가시킬수록 용량 초과회수가 줄어들며 임시버퍼가 20%일 경우는 용량 초과가 거의 일어나지 않음을 알 수 있다.

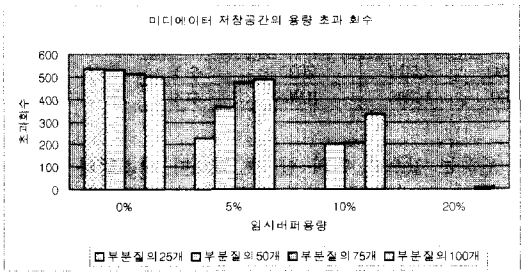

그림 4 미디에이터 저장공간의 용량초과 회수

또한 임시버퍼의 용량에 차이가 있을 때 최적화비용과 실제최적비용과의 차이를 알아보고, 동일한 조건에서 임시버퍼에 따른 최적화비용의 변화를 살펴보기 위하여 그림 5와 같은 실험을 하였다. 이 경우 그림 4의 실험과 동일한 환경에서 최적화비용과 실제최적비용의 차이가 어느 정도 일어나는지를 살펴보았다. 각 부분질의에 대해 half_life를 0, 30, 60으로 나누어 실험을 하였으며, 총 5번을 하여 평균값으로 나타내었다.

따라서 그림 5는 임시버퍼가 0%, 5%, 10%, 20%일 때 최적화비용과 실제최적비용이 어느 정도 차이가 있는지를 나타내며, 임시 버퍼의 용량에 관계없이 최적화비용이 실제최적비용에 근사한 값을 갖는다는 것을 알 수 있다. 이때 실제최적비용을 0%로 하고 half-life에 따른 최적화비용의 차이를 비교해 보았다. 본 실험에서는 그림 3의 실험결과에서와 마찬가지로 본 실험에 사용된 데이타에 대해서 평균적으로 half_life를 60으로 하였을 때 실제최적비용에 가장 가까이 접근한다는 것을 알 수 있으며, 감쇄율을 적용하지 않은 경우(*half-life = 0*)에 실제최적비용과의 차이가 가장 크게 나타나서 사용자의 접근 패턴의 변화를 적용한 감쇄율의 효과를 확인할 수 있다. 이때 그림 4와 그림 5에서 보면 실제최적비용과 최적화비용이 12%정도 차이가 나는 것을 알 수 있다. 그러나 실제최적비용을 계산하기 위해서는 모든 부분질의를 다 조합해야 하므로 n개의 부분질의가 있을 경우 시간복잡도가 $O(n!)$이나 본 논문의 최적화 알고리즘을 사용할 경우에는 $O(2n)$이다. 따라서 비용을 구할 때 많은 시간이 절약된다. 또한 임시버퍼의 용량이 작을수록 최적화를 실행하는 회수가 증가되어 불필요하게 최적화를 수행하는 비용이 추가되나 버퍼의 용량 초과가 자주 일어나지 않은 20%의 임시버퍼를 가질때의 비용과 거의 차이가 없기 때문에 임시버퍼를 20%로 잡는 것이 가장 효율적이라고 할 수 있다.

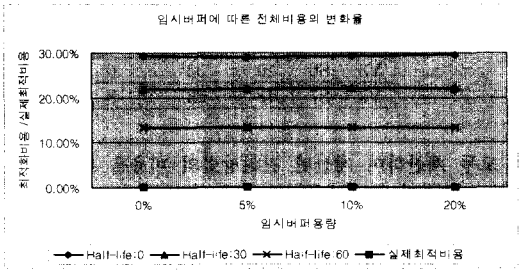

그림 5 임시버퍼에 따른 전체비용의 변화율

그림 6은 실제최적비용과 최적화비용을 구할 때 필요한 시간을 나타내고 있다. 그림 6에서 보듯이 부분질의 개수가 증가함에 따라 최적화비용을 구하는 시간에 많은 차이가 있음을 알 수 있다. 본 실험은 전체 부분질의가 100개이고 각 부분질의의 용량은 평균 500KB이다. 총 100일 동안의 자료를 가지고 실험을 했으며 생성된 로그는 약 7GB이고, 5번의 실험에 대한 평균값으로 나타내었다. 이 경우 실제최적비용을 구할 경우 1380분, 즉 23시간이 걸리는 것을 알 수 있었다. 그러나 최적화비용을 구하는데는 12분이 소요되었다. 따라서 본 논문에서 제안하는 알고리즘으로 최적화비용을 구하는 것이 많은 시간의 절약을 가져온다는 것을 알 수 있다.

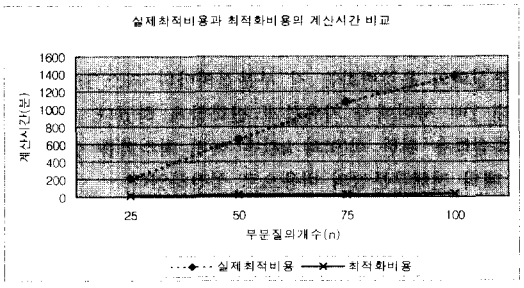

그림 6 실제최적비용과 최적화비용의 계산시간 비교

미디에이터가 구체화방법으로 관리되는 부분질의의 결과를 저장하기 위해 필요한 저장공간의 크기가 주어진 전체 부분질의의 결과크기에 대해 어느 정도일 때 가장 좋은 효과를 나타낼 수 있는지 실험을 통해 살펴보았다.

일반적으로 검색 시스템에서는 80/20 규칙을 많이 적용하며, 80/20 규칙이란 전체 부분질의 중 20%에 해당하는 부분질의에 80%의 사용자의 접근이 일어나고 나머지 80%의 부분질의에 20%의 사용자의 접근이 일어남을 의미한다. 따라서 본 논문에서는 80/20 규칙을 적용하여 사용자의 접근회수와 갱신회수를 임의로 생성하여 이에 따른 전체 비용의 변화를 살펴보았다.

그림 7은 최적의 저장공간을 나타내며, 기울기가 급하게 감소하는 부분이 최적의 저장공간을 의미한다. 본 실험에서는 80/20 규칙을 적용하여 사용자의 접근회수와 갱신회수를 임의로 생성하였기 때문에 사용자의 접근회수가 갱신회수보다 많아질수록 80/20 규칙에 따라서 많은 부분질의들에 대해 수정방법보다 구체화방법을 많이 사용하게되고, 전체부분질의의 결과크기에 대해 30%의 저장공간이 할당되었을 때 가장 효과적임을 알 수 있다. 그러나 갱신회수가 사용자의 접근회수보다 많아질수록 구체화방법보다는 수정방법을 더 많이 사용하는 것이 효과적이므로 전체부분질의의 결과크기에 대해 30%보다는 작은 저장공간이 요구됨을 알 수 있다. 따라서 부분질의가 수정방법으로 관리될 경우에는 사용자의 접근이 발생할 때마다 원격지 서버에서 부분질의의 결과를 전송받아 처리해야 하지만 구체화방법으로 관리될 경우에는 미디에이터에서 바로 결과를 처리하기 때문에 수정방법보다 많은 비용의 절감을 가져오고, 구체화 방법으로 관리되는 부분질의가 많은 경우가 수정 방법으로 관리되는 부분질의가 많을 때보다 많은 부분질의의 결과를 미디에이터에 저장해야하기 때문에 더 많은 저장공간이 필요하다는 것을 알 수 있다.

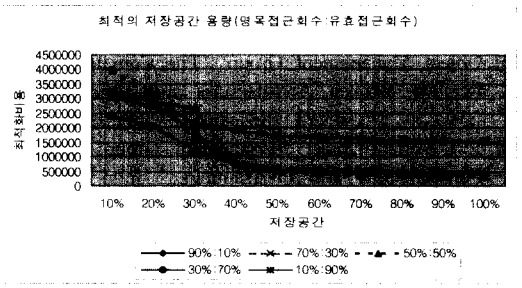

그림 7 최적의 저장공간 용량

그림 8은 수정방법으로만 뷰 관리를 했던 기존의 방법과 구체화방법을 사용한 본 알고리즘과의 최적화비용의 변화를 갱신회수와 사용자 접근회수의 변화로 나타낸다. [갱신회수 / 접근회수]를 [100% / 0%]부터 [0% / 100%]까지 변화를 시켜가면서 최적화비용의 변화를 살펴보았다. 여기서 [갱신회수 / 접근회수]가 [100% / 0%] 라는 것은 갱신회수만 발생하고 접근회수는 한번도 발생하지 않은 것을 나타내며, [60% / 40%] 는 총 생성된 회수 중 갱신회수가 60%, 접근회수가 40%임을 나타낸다.

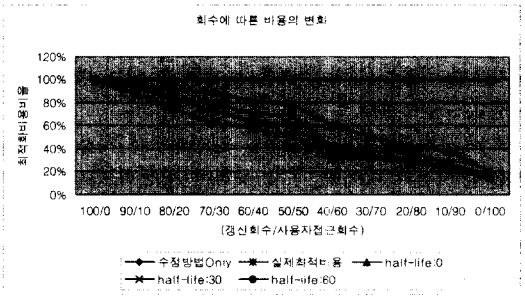

그림 8 회수에 따른 비용의 변화

갱신회수보다 사용자의 접근회수가 많을수록 기존의 수정방법으로 관리했을 때보다 최적화비용이 현저하게 줄어드는 것과 실제최적비용에 근접하는 것을 알 수 있다. 기존의 수정방법으로 뷰 관리를 하던 미디에이터들은 사용자의 접근이 발생할 때마다 원격지 서버로부터 부분질의의 결과를 받아서 사용자에게 제공하였으나 본 미디에이터 시스템은 구체화방법을 사용하기 때문에 부분질의가 구체화방법으로 관리되고 있는 경우 원격지 서버에 접근하지 않고 바로 질의결과를 보여줄 수 있어 비용이 크게 감소함을 알 수 있다. 따라서 구체화방법으로 뷰를 관리하는 것이 수정방법으로 뷰를 관리하는 것보다 많은 비용의 절감과 시간의 절약을 가져온다는 결론을 얻을 수 있고, 기존의 미디에이터들과 같이 감쇄율을 적용하지 않았을 때보다 적용시킬 때가 다양한 뷰의 활용패턴에 융이하게 대처할 수 있고, 저렴한 최적화비용을 얻을 수 있음을 알 수 있다.

5. 결 론

컴퓨터 기술과 네트워크 환경이 발전함에 따라 제한된 범위 내에서 서로 유용한 기능들을 제공하고 있는 기존의 하드웨어와 소프트웨어들을 네트워크 상에서 서로 독립적이 아닌 통합적으로 운영하는 방법의 필요성이 증가하고 있다. 이에 대해 미디에이터 시스템의 연구가 활발하게 진행되고 있으나 기존의 미디에이터 시스템들은 수정방법으로만 뷰 관리를 하고 웹에서의 접근이 각 미디에이터 시스템에 맞는 특정한 툴킷을 사용해야만 하는 단점이 있기 때문에 본 논문에서는 이질적인 환경의 시

스템과 데이타베이스들이 가지고 있는 특성들을 최대한 활용하면서 마치 하나의 시스템에 접근하는 것과 같이 일반적으로 사용하는 웹브라우저를 통하여 질의를 하고 결과를 받아 볼 수 있게 하는 미디에이터를 구현하였고, 기존의 미디에이터 시스템들이 수정방법으로만 뷰를 관리하던 것과는 달리 본 미디에이터 시스템에서는 구체화 방법으로 뷰를 관리함으로써 네트워크의 트래픽과 오버헤드를 줄이고, 전체 비용의 절감을 가져오며, 빠르게 사용자에게 원하는 질의 결과를 보여줄 수 있는 방법을 제시하고 구현하였다.

또한 구체화방법의 문제점인 갱신회수나 사용자 접근회수의 시점의 차이를 두지 못하여 유동적으로 변화할 수 있는 사용자의 접근형태나 데이타베이스의 갱신에 대한 패턴의 변화를 적절히 반영하지 못했던 점을 본 논문에서는 감쇄율을 갱신회수와 사용자의 접근회수에 적용하여 최근에 발생한 패턴에 더 큰 비중을 두어 이러한 문제점을 해결하고, 뷰 사용 패턴에 대한 모델링을 통하여 실제최적비용에 가까이 접근하는 최적화 방법을 설계하고 구현하였다. 따라서 최적화비용을 구하기 위해서 최적화 알고리즘을 사용함으로써 부분질의를 모두 조합하지 않고도 빠른 시간에 실제최적비용에 근사한 값을 구한다. 이를 증명하기 위해 다양한 실험을 통하여 본 논문에서 제시하는 최적화 방법이 최적화비용을 계산하는데 빠른 속도를 보장하고, 구체화방법으로 뷰를 관리하는 것이 수정방법으로 관리하는 것보다 많은 비용의 감소를 가져오며, 감쇄율을 사용할 때가 사용하지 않을 때보다 더 나은 성능을 보인다는 결과를 보였다.

참 고 문 헌

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

[2] Nita Goyal et al, "Preliminary Report on (Active) View Materialization in GUI Programming," proceeding of the Workshop on Materialization Views : Techniques and Applications, pp. 56-64, June 1996.

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

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

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

[6] H.Garcia-Molina, Y.Papakonstantinou, D.Quass, A. Rajaraman, Y.Sagiv, J.Ullman, V.Vassalos, J.Widom, "The TSIMMIS approach to mediation : Data models and Languages," In Journal of Intelligent Information Systems, 1997.

[7] Chen Li, Ramana Yerneni, Vasilis Vassalos, Hector Garcia-Molina, Yannis Papakonstantinou, Jeffrey Ullman, Murty Valiveti. "Capability Based Mediation in TSIMMIS," SIGMOD 98 Demo, Seattle, June 1998.

[8] Alexandros Labrinidis, Nick Roussopoulos, "On the Materialization of WebViews," ACM SIGMOD Workshop on The Web and Databases (WebDB '99) June 3-4, 1999 Philadelphia, Pennsylvania.

[9] Elena Baralis, Stefano Paraboschi, Ernest Teniente, "Materialization View Selection in a Multidimensional Database," Proceedings of the 23rd VLDB Conference Athens, Greece, 1997.

[10] Harold S.Javitz, Alfonso Valdes, "The NIDES Statistical Component: Description and Justification," SRI International Menlo Park, California 94025. March, 1994.

[11] Sophie Cluet, Claude Delobel, Jerome Simeon, Katarzyna Smaga, "Your Mediators Need Data Conversion!," SIGMOD'98 Seattle, WA, USA, 1998 ACM.

[12] http://id.inel.gov/idim/mediator_technology .html

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

[14] Steve Vinoski, "Distributed Object Computing With CORBA," C++ Reportmagazine, 1993.

[15] Object Management Group, "The Common Object Request Broker Architecture and Specification," OMG TC Document, Revision 2.0, July 1995.

[16] Jon Siegel, Object Management Group, "CORBA Fundamentals and Programming," John Wiley & Sons Inc. 1996.

[17] V.S. Subrahmanian, Sibel Adali, Anne Brink, Ross Emery, James J.Lu, Adil Rajput, Timothy J.Rogers, Robert Ross, Charles Ward, "HERMES : A Heterogeneous Reasoning and Mediator System".

[18] Anthony Tomasic, Remy Amouroux, Philippe Bonnet, Olga Kapitskaia, Hubert Naacke, Louiqa Raschid, "The Distributed Information Search Component (DISCO) and the World Wide Web," SIDMOD'97 AZ,USA.

[19] S. Chaudhuri, Krishnamurthy, S. Potamianos, and

K. shim, Optimizing Queries with Materialized Views, In proc. of ICDE, pp 190-200, 1995.
[20] C. M. Chen and N. Rossopoulos. The Implementation and Performance Evaluation of mmthe ADMS Query Optimizer: Integration Query Result Cashing and Matching. In Proc. of EDBT, pp.323-336, 1994.
[21] A. Y. Levy, A. O. Mendelzon, Y. Sagiv, and D. Srivastava. Answering Queries Using Views. In Proc. of ACM SIGMOD, pp. 95-104, 1995.
[22] O. G. Tastalos, M. H. Solomon, and Y. E. Ioannidis. The GMAP: A Versatile Tool for Physical Data Independence. In Proc. of VLDB, pp.367-378, 1994.
[23] H. Z. Yang and P. A. Larson. Query Transformation for PSJ-queries. In Proc. of VLDB, pp. 245-254, 1987.
[24] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying Heterogeneous Information Source Using Source Description. In Proc. of VLDB, pp. 251-262, 1996.
[25] A. Gupta, V. Harinarayan, and D. Quass. Aggregate-Query Processing in Data Warehousing Environments. In Proc. of VLDB, pp. 358-369, 1995.
[26] Ashish Gupta, Inderpal Singh Mumick, V. S. Subrahmanian, "Maintaining Views Incrementally," Proc. of ACM-SIGMOD Intl Conf. of Management of Data, pp. 157-166 May, 1993.
[27] Ashish Gupta, Inderpal Singh Mumick, "Maintenance of Materialized View : Problems," Techniques, and Applications, Proc. od Intl Conf, on Data Engineering, pp. 86-93, 1990.
[28] Harumi A. Kuno, Elke A. Rundensteiner, "The MuntiView OODB View System: Design and Implementation," Theory and Practice of Object System, Vol.2 No. 3, pp.203-225, 1996.
[29] Renate Motsching, "Requirements and Comparison of View Concepts for Object-Oriented Databases" INformation System, 1996.
[30] C. Souza dos Snatos, "Design and Implementation of Object-Oriented Views," Proc. of Intl Conf. and Workshop on Database and Expert Systems Applications(DEXA), pp. 91-102, 1995.
[31] 주길홍, CORBA 환경에서 분산데이타베이스의 통합 뷰관리를 위한 미디에이터 시스템 연구, 연세대학교 석사논문, 2000.

주 길 홍
1998년 인천대학교 전자계산학과 졸업(공학사). 2000년 연세대학교 컴퓨터과학과 졸업(공학석사). 2000년 ~ 현재 연세대학교 컴퓨터과학과 재학중(박사과정). 관심분야는 분산데이타베이스, 미디에이터시스템, 웹데이타마이닝

이 원 석
1985년 미국 보스턴대학교 컴퓨터공학과 졸업(공학사). 1987년 미국 퍼듀대학교 컴퓨터공학과 졸업(공학석사). 1990년 미국 퍼듀대학교 컴퓨터공학과 졸업(공학박사). 1990년 ~ 1992년 삼성전자 선임연구원. 1993년 ~ 1999년 연세대학교 컴퓨터과학과 조교수. 1999년 ~ 현재 연세대학교 컴퓨터과학과 부교수. 관심분야는 분산데이타베이스, 미디에이터시스템, 데이타마이닝, 침입탐지, 멀티미디어데이타베이스