미디에이터 시스템의 적응적 구체화 질의 선택방법

주 길 홍† · 이 원 석††

요 약

다양한 분산정보의 통합을 위하여 전역 사용자들이 요구하는 데이터들을 지역서버의 자치성을 유지하면서 효과적으로 제공하기 위한 미디에이터 시스템의 연구가 활발하게 진행되고 있다. 그러나, 미디에이터에서의 전역질의를 효율적으로 처리하기 위한 방법의 연구는 상대적으로 매우 미흡한 실정이다. 미디에이터에서 전역질의는 원격지서버의 질의처리 단위인 부분질의의 집합으로 표현되어진다. 따라서, 미디에이터가 부분질의의 결과를 구체화방법으로 관리한다면 관련된 질의의 결과를 사용자에게 신속하게 제공할 수 있다. 그러나, 미디에이터에서는 통합 스키마의 변경이 자유롭고, 사용자의 질의가 자주 변할 수 있기 때문에 최적의 구체화부분질의를 결정하기 위해서 전역질의의 빈발정도를 지속적으로 파악해야 한다. 따라서, 부분질의의 수가 증가할수록 최적화의 수행시간이 매우 길어지기 때문에 부분질의 빈도의 최근의 변화를 적절하게 반영하지 못한다. 따라서 본 논문에서는 미디에이터 시스템의 저장공간의 활용율을 극대화하는 최적의 구체화부분질의 군을 선택하는 방법을 제안한다. 또한 과거와 최근의 부분질의의 활용형태가 다양하게 변할 수 있으므로 시간에 따른 감쇄율을 활용빈도에 적용하여 최근의 활용빈도의 변화에 민감하게 반응하고 활용형태의 변화에 따라 적응적으로 새로운 구체화부분질의 군을 선택할 수 있는 방법을 제안한다.

An Adaptive Materialized Query Selection Method in a Mediator System

Kil Hong Joo† · Won Suk Lee††

ABSTRACT

Recent researches which purport to integrate distributed information have been concentrated on developing efficient mediator systems that not only provide a high degree of autonomy for local users but also support the flexible integration of required functions for global users. However, there has been little attention on how to evaluate a global query in a mediator. A global query is transformed into a set of its sub-queries and each sub-query is the unit of evaluation in a remote server. Therefore, it is possible to speed up the execution of a global query if the previous results of frequently evaluated sub-queries are materialized in a mediator. Since the integration schema of a mediator can be incrementally modified and the evaluation frequency of a global query can also be continuously varied, query usage should be carefully monitored to determine the optimized set of materialized sub-queries. Furthermore, as the number of sub-queries increases, the optimization process itself may take too long, so that the optimized set identified by a long optimization process may become obsolete due to the recent change of query usage. This paper proposes the adaptive selection of materialized sub-queries such that available storage in a mediator can be highly utilized at any time. In order to differentiate the recent usage of a query from the past, the accumulated usage frequency of a query decays as time goes by.

키워드 : 미디에이터(Mediator), 이질적 정보시스템(Heterogeneous Information Systems), 데이터 구체화(Data Mterialization), 분산질의처리(Distributed Query Processing), 감쇄율(Decay Rate)

1. 서 론

컴퓨터와 통신 기술이 발전함에 따라 네트워크를 통한 일반 사용자들의 컴퓨터 활용 빈도와 요구하는 데이터의 양이 급격히 증가되었다. 이에 따라 최근의 정보 시스템은 정보의 활용성을 향상시키기 위하여 이질적인 시스템들을 의미상으로 연결하고 있다. 이러한 정보 시스템들의 연관된 정보들이 서로 다른 시스템에 분산되어져 있고, 각각 자치적으로 구축되고 운영되기 때문에 각각 독립적인 특성들을 가지고 있어서 상호작용에 의한 효율적인 정보공유에 많은 문제점을 갖는다. 따라서, 분산환경에서 관련된 정보를 얻기 위하여 불필요하게 많은 노력이 필요하게 되었고, 이로 인해 서로 이질적인 환경을 가지고 있는 두 개 이상의 정보 시스템들을 통합하는 경우 중점적으로 고려해야 할 사항은 사용자에게 이질적인 환경을 은폐하면서 원격지에 존재하는 시스템들의 자치성을 최대한 유지하는 것이다. 이러한 이질적인 특성을 극복하기 위한 방법으로 미디에이터(Mediator)와 래퍼(Wrapper)의 필요성이 대두되었고, 이에

† 준 회 원 : 연세대학교 대학원 컴퓨터과학과
†† 종신회원 : 연세대학교 컴퓨터과학과 교수
논문접수 : 2003년 4월 14일, 심사완료 : 2003년 11월 13일

대한 연구가 활발히 진행되고 있다[1].

미디에이터란 서로 이질적인 하드웨어 환경 및 이질적인 데이터베이스와 소프트웨어 등에 대해서 사용자가 이질적인 환경을 느끼지 못하게 은폐하면서 하나의 시스템에 접근하는 것과 같은 효과를 제공하는 미들웨어(middleware)[1]를 의미하며, 래퍼는 외부에서 활용되는 시스템의 내부기능을 포장하여 외부로 보여주어 미디에이터가 원격지서버의 구동환경에 관계없이 원격지서버의 기능들을 동일한 방법으로 활용할 수 있도록 지원하는 기술[2]을 의미한다. 일반적으로 분산데이터베이스시스템은 각 지역스키마에 대한 스키마 통합과 이들의 접근 및 갱신 형태를 설계시 분석하여 이에 적합한 전역스키마를 구성하고, 그에 따라서 지역사용자가 필요로 하는 내용을 기반으로 전역 뷰를 형성한다. 그러나, 이들 각 지역스키마들은 그 지역의 새로운 요구사항에 따라 자주 변형될 수 있으며, 변형된 지역 스키마에 따라 전역스키마가 전체적으로 수정되어야 하는 단점이 있다. 그러나, 미디에이터 시스템은 의미적으로 통합된 전역스키마를 사용하지 않고 사용자가 필요로 하는 기능에 대한 정보단위로 필요시 연관된 지역스키마들에 대한 뷰를 정의하므로 변경된 지역스키마에 영향을 받는 해당 미디에이터의 매핑 정보만 수정하면 된다[1]. 따라서, 미디에이터에서는 뷰 정의가 변경되거나 새로운 뷰가 자유롭게 추가될 수 있으며 이러한 뷰에 대한 접근형태 또한 지속적으로 변경될 수 있다.

여러 분산데이터베이스를 통합하는 뷰에 대한 전역질의가 발생했을 때 전역질의 조건을 연관된 원격지서버별로 분할된 부분질의들로 변환하여 각 부분질의를 처리하게 된다. 이때 부분질의는 원격지서버의 질의처리 단위 및 미디에이터와 원격지서버의 결과 데이터 전송단위가 된다. 미디에이터에서는 부분질의를 처리하기 위해 수정방법(query modification)과 구체화방법(view materialization)을 사용한다[3]. 수정방법은 각 원격지서버에서 모든 해당 부분질의를 처리하며, 구체화방법은 자주 사용되어지는 부분질의의 결과를 미디에이터에 저장하여 반복된 질의 처리가 요구될 때 저장된 결과를 재사용한다. 따라서, 서로 다른 전역질의에 같은 부분질의가 구체화로 구현되어 있다면 부분질의의 결과를 공유할 수 있으므로 매우 효율적이다. 또한 수정방법은 질의를 수행하기 위하여 네트워크 환경을 통하여 각 원격지서버의 테이블에 접근하여 데이터를 처리하기 때문에 질의를 수행할 때마다 네트워크를 통하여 원격지서버에 접근해야 하므로 많은 처리비용이 필요하며, 많은 네트워크의 트래픽이 발생된다. 따라서, 질의를 구성하는 기본테이블의 양이 방대해 질 경우 질의 처리비용이 많이 요구된다. 그러나, 구체화방법은 처음 질의가 요청되었을 때 저장공간에 질의의 결과를 생성하고 유지하기 때문에 항상 최신정보가 유지되어야 하며 원격지서버의 기본 테이블에 변경이 일어난 경우 즉시 갱신이 이루어져야 한다. 따라서 자주 접근되는 부분질의를 구체화부분질의로 구현할 경우 수정부분질의로 구현할 경우보다 원격지서버를 거치지 않으므로 네트워크의 트래픽을 감소시키고 사용자에게 빠르게 결과를 보여줄 수 있다[4]. 본 논문에서는 수정방법으로 구현되는 부분질의를 수정부분질의(modified sub-query)라고 정의하고, 구체화방법으로 구현되는 부분질의를 구체화부분질의(materialized sub-query)라고 정의한다.

미디에이터 시스템에서는 자주 활용되는 질의를 구체화방법으로 구현하면 높은 효율성을 가질 수 있다. 그러나, 정의된 질의 중에서 구체화방법으로 구현할 질의를 결정하는 문제는 해당 시점에서의 각 질의의 활용형태에 따라 모든 질의들의 구체화관리비용과 수정관리비용을 모두 고려하여 신속하게 최적의 관리계획을 수립해야 하는 어려움이 있다. 이러한 어려운 문제 때문에 기존의 미디에이터 시스템들[5-8]은 수정방법으로 모든 질의를 구현하여 네트워크 트래픽과 처리속도가 증가하는 단점이 있다. 그러나, 미디에이터 시스템에서도 사용자의 질의에 대한 접근형태나 데이터베이스의 갱신형태에 대해 과거의 시점과 최근의 시점의 변화를 모델링하여 각 뷰의 관리방법을 결정한다면 미디에이터에서의 성능을 향상시킬 수 있다. 따라서, 각 질의에 대해 효과적인 관리방법을 결정하기 위해서는 질의에 대한 사용자의 접근 및 갱신 형태를 정확하게 파악해야 한다. 또한 미디에이터 시스템에서 정의된 전역질의는 관련된 원격지서버에 따라 여러 개의 부분질의들로 구현되며 분해되어진 부분질의들은 미디에이터 시스템과 원격지서버의 데이터 전송단위가 된다. 따라서 본 논문에서는 미디에이터에 요구된 모든 부분질의 집합에 대해 각 부분질의의 처리방법에 따라 미디에이터에 정의된 모든 부분질의들을 특정시점에서 효율적으로 구현하는 방법을 *부분질의 구현계획*(implementation plan)이라고 정의한다. 부분질의 구현계획은 부분질의 s_i와 이 부분질의의 구현방법 $I_i \in$ {구체화방법 *or* 수정방법}으로 구성된 쌍 (s_i, I_i)로 정의한다. 따라서 본 논문에서 제안하는 알고리즘은 다음과 같은 3가지의 특징을 가지고 있다.

- 미디에이터 시스템에 정의된 부분질의들에 대한 사용자의 접근형태나 데이터베이스의 갱신형태의 변화 등과 같은 최근의 활용형태에 자체 적응적으로 모델링하는 방법.
- 미디에이터 시스템의 저장공간을 최대로 활용하면서, 미디에이터에 정의된 모든 부분질의들의 현재 활용형태에 따라 부분질의들의 전체 처리비용이 최소가 되도록 최적의 부분질의 구현계획을 생성하는 방법
- 새로운 부분질의의 추가나 부분질의들의 최근 활용형태의 변화에 따라 현재의 부분질의 구현계획에 대한

최적상태의 주기적으로 검사하여 새로운 부분질의 구현계획의 생성 필요시점을 탐지하는 방법

본 논문의 구성은 다음과 같다. 2장에서는 관련연구에 대해서 기술하고, 3장에서는 부분질의의 활용형태를 자체 적응적으로 모델링하는 방법을 기술한다. 4장에서는 부분질의들의 활용형태에 따라 최적의 부분질의 구현계획을 생성하는 방법을 기술한다. 또한 부분질의들의 최근의 변화에 현재 부분질의 구현계획의 유효성을 탐지하여 새로운 부분질의 구현계획의 정확한 생성 시점을 찾는 방법에 대해 설명한다. 5장에서는 3장과 4장에서 제안된 내용에 대한 다양한 실험을 수행하고 이에 대한 결과를 분석한다. 마지막으로 6장에서는 최종적인 결론을 맺는다.

2. 관련 연구

서로 이질적인 환경을 가지고 있는 두 개 이상의 정보 시스템들을 통합하는 경우 중점적으로 고려해야 할 사항은 사용자에게 이질적인 환경을 은폐하면서 원격지에 존재하는 시스템들의 자치성을 최대한 유지하는 것이다. TSIMMIS [6]는 미디에이터 시스템을 ORB 환경에서 구축하여 미디에이터를 통하여 이질적인 분산 환경의 데이터베이스에도 접근이 가능하다. 이를 위해 미디에이터에서는 전체 스키마와 뷰를 MSL(Mediator Specification Language)로 정의를 하고, 래퍼에서는 원격지서버의 관련정보를 WSL(Wrapper Specification Language)로 정의하기 때문에 MSL과 WSL을 매핑시키는 작업이 필요하고 자체적으로 정의한 특정한 툴킷과 언어를 통하여 데이터베이스에 접근이 가능하다. 또한 HERMES[7]는 초창기의 미디에이터 시스템으로 이미지와 텍스트, 그리고 데이터베이스를 통합하는 미디에이터 시스템이다. 이 시스템은 단일 기종의 시스템에서만 사용이 가능하고, 데이터베이스와 웹과의 연동이 이루어지지 않으나 데이터베이스, 데이터 구조, 그리고 소프트웨어 패키지 등을 통합하는 방법을 제시하였다. HERMES는 자체적인 논리적 언어(logic-based declarative language)를 통해 통합하고자 하는 서버와 데이터 교환을 수행하여 각 서버와의 통합을 가능하게 한다. 따라서, 미디에이터 관리자가 미디에이터의 언어에 기반하여 연결된 서버와 데이터베이스에 필요한 입력형식과 출력형식을 일일이 열거해야한다. HERMES는 미디에이터의 스키마와 관련된 원격지서버의 데이터와의 연동을 지원하기 위해 "yellow page"기능을 제공한다. TSIMMIS 시스템과 HERMES 시스템은 미디에이터 시스템의 전역 질의를 수정방법으로 구현하고 있다.

구체화뷰의 구현방법은 먼저 뷰를 수행하여 그 결과를 물리적으로 데이터베이스에 저장하고, 이후의 뷰에 대한 질의를 이미 저장된 결과에 대하여 수행하는 방법이므로 수정방법에 비하여 부가적인 저장공간이 필요하나, 뷰에 대한 질의 처리속도가 빠르다. 따라서, 구체화뷰를 효율적으로 사용하기 위해서는 주어진 뷰와 이들에 대한 활용형태를 고려하여 한정된 저장공간을 최적으로 활용하여 전체 질의 처리비용을 최소화해야 하며, 이러한 뷰들을 구체화뷰로 구현하는 것은 매우 중요한 문제이다[9, 10]. 이에 [11]에서는 분산 데이터베이스 환경에서 사용자의 활용빈도와 원격지 서버에 있는 기본테이블의 갱신회수를 정적으로 모델링하여 고정적으로 모델링 된 뷰 활용빈도에 대해 각 뷰를 수정방법으로 구현할때의 처리비용이 가장 큰 순서대로 구체화뷰로 선택한다. 따라서, 이 방법은 사용자의 뷰에 대한 활용빈도의 변화를 고려하지 않기 때문에 자주 사용되지 않는 뷰가 수정방법으로 질의를 처리할때의 비용이 매우 커서 구체화뷰로 구현될 가능성이 있기 때문에 저장공간의 활용측면에서 낭비를 초래할 수 있다. 또한 [22]에서는 미디에이터 시스템에서 분산데이터베이스의 통합을 위해 이질적인 환경의 시스템과 데이터베이스들이 가지는 특성들을 최대한 활용하여 유동적으로 변할 수 있는 사용자의 활용형태와 부분질의에 연관된 원격지서버의 기본테이블의 갱신형태를 적절하게 반영하는 방법을 제시하였다. 이를 위해 부분질의 처리비용을 기반으로 한 사용자의 활용패턴과 부분질의의 크기에 대한 순위를 결정하여 순위를 기반으로 빠른 시간에 구체화부분질의를 선택한다.

구체화뷰는 데이터 웨어하우스 분야에서 많이 사용되고 있다[12, 13]. 데이터 웨어하우스는 그 자체가 기존의 파일이나 데이터베이스에 대한 구체화 뷰임으로 많은 연구들이 구체화뷰의 선택 문제에 집중해서 연구하고 있다[8, 14, 15]. 구체화 뷰 선택 문제는 대상 데이터들에 대한 사용자의 뷰 활용빈도와 기본 테이블의 갱신회수를 미리 파악하여 모든 뷰에 대해서 가장 최소의 질의 처리비용으로 처리하는 구체화 뷰 집합을 찾는 방법이다. 또한 데이터 웨어하우스 시스템에서는 기본테이블에 대해 여러 질의에서 공통적으로 사용되는 부분질의를 나타내기 위해 서포팅 뷰(supporting view)를 사용한다. 서포팅 뷰는 데이터 웨어하우스 스키마의 기본테이블과 연관성이 있는 구체화 뷰로 간주할 수 있기 때문에 자주 사용되어지는 질의의 처리비용을 줄일 수 있다. [14]에서는 격자(lattice) 구조의 데이터 큐브에서의 구체화 뷰 선택 알고리즘을 제안하였다. 이 연구에서는 데이터 큐브를 격자 구조로 변환한 후 각 뷰를 구체화방법과 수정방법으로 구현할 때 발생되는 처리비용의 이익을 각각 계산하여 주어진 뷰들을 구현할 수 있는 방법을 모두 조합하여 뷰들에 대한 서로 다른 구현계획을 생성한 후 각 방법에 대한 이익을 계산하여 총 이익이 극대화하는 뷰를 구체화뷰로 선택한다. 또한 [10]에서는 향상된 질의 효율과 적은 뷰 유지비용을 갖는 조합을 찾기 위한 구체화 뷰 선택 알고리즘을 제안한다. 이 알고리즘은 0-1 정수 프로그래밍 기법을 사용하여 최적의 전역 계획을 찾아 최적의 구체

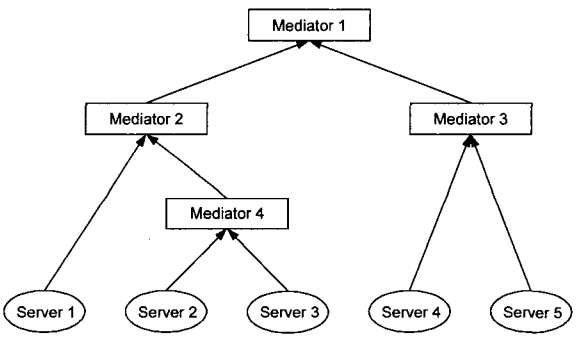
화뷰 집합을 선택한다.

데이터 웨어하우스에서 사용되어지는 다양한 구체화 뷰 선택 알고리즘들은 다음의 세 가지 이유 때문에 미디에이터 시스템에서 적용하기 어려운 점이 있다. 첫째로, 데이터 웨어 하우스 시스템에서는 구체화뷰를 선택할 때 뷰의 활용형태를 고정적으로 모델링하여 최소의 처리비용을 갖도록 하는 구체화뷰 집합을 선택[19, 20]하기 때문에 미디에이터 시스템에서와 같이 사용자의 요구에 따라 유동적으로 새로운 뷰가 추가되고, 뷰에 대한 접근형태 또한 지속적으로 변형될 수 있는 환경에는 적합하지 않다. 둘째로, 데이터 웨어하우스 시스템에서는 구체화뷰를 선택하여 저장하기 위한 물리적인 저장공간의 한계성을 고려하지 않는다. 즉, 구체화뷰의 갱신으로 구체화뷰의 용량이 감소하여 저장공간에 여유가 생겼을 때와 구체화로 구현된 뷰의 용량이 증가하여 저장공간이 초과한 경우에 대한 처리 방법을 고려하지 않는다. 마지막으로, 구체화뷰를 선택하기 위해서 모든 뷰의 두 가지 구현방법을 조합하여 모든 조합에 대한 전체 처리이익을 계산한다. 따라서, 구체화뷰를 선택하는데 많은 시간이 요구되며, 데이터 웨어하우스와 같이 스키마 설계시에만 뷰의 구현방법을 결정할 때에는 문제가 되지 않으나, 미디에이터와 같이 뷰의 활용형태의 변화에 따라 구체화뷰 선택작업이 수시로 수행되어야 하는 환경에서는 적합하지가 않다[21]. 또한 조합해야 하는 전체 뷰의 개수가 증가할수록 가능한 모든 뷰에 대한 구현방법의 조합을 고려하여 구체화뷰 집합을 찾는 문제가 NP-complete임으로[9, 17, 18] 뷰의 개수가 증가될수록 최적화된 뷰의 구현계획을 생성하는데 많은 시간이 필요하다. 따라서, 긴 최적화 과정을 통해 찾은 새로운 최적화 구현계획이 적용되는 시점의 사용자의 뷰 활용패턴이 이전과 크게 다를 수 있으므로 뷰 구현계획의 최적성이 보장되지 않는다. 따라서, 본 논문에서는 미디에이터 시스템에서 유동적으로 변화되는 뷰의 활용형태 및 기본테이블의 갱신형태에 대해 신속하고 자체 적응적인 최적의 뷰 선택 알고리즘을 제안한다.

일반적인 최적화 알고리즘[22, 23]에는 탐욕적인 방법(greedy method)과 동적프로그래밍 방법(dynamic programming) 및 분기한정 방법(branch-and-bound)이 있다. 탐욕적인 방법은 일련의 선택을 과정을 통하여 최종 해답에 도달하게 되며, 선택할 때마다 그 시점에서 가장 좋은 해답을 선택하므로 국부적(locally)으로는 최적이나 전반적으로는 최적의 해답을 항상 찾지 못하는 단점을 갖는다. 동적 프로그래밍 방법은 문제를 반복적으로 더 작은 문제로 분할한다는 점에서 분할 정복법(devide-and-conquer)과 비슷하다. 그러나 이 방법은 작은 문제를 먼저 해결하고, 그 결과를 저장한 후에 그 결과가 필요할 때마다 다시 계산하는 대신에 그 저장한 값을 이용한다. 분기한정 방법은 상태공간 트리를 사용한다는 점에서 역추적(back tracking) 방법과 매우 유사하며, 해답을 찾는 시간이 오래 걸릴 수 있으나 항상 최적의 해답을 찾는다. 모든 최적화 문제에서 최적의 해결책을 찾는 것은 매우 중요한 문제이나 미디에이터 시스템에서 최적의 구체화뷰 집합의 선택문제는 해답을 찾는 시간이 중요한 고려대상이기 때문에 본 논문에서는 제안하는 알고리즘과 일반적인 최적화 알고리즘과의 효율성을 해답을 찾는 시간을 통해 비교한다.

3. 감쇄율 기반의 사용자 활용패턴

일반적으로 미디에이터 시스템은 (그림 1)과 같이 계층적인 구조로 구현될 수 있으며 사용자가 필요로 하는 요구정보 단위에 따라 원격지서버나 하위 미디에이터 시스템의 스키마로부터 각 미디에이터의 뷰를 정의한다. 따라서, 부분질의의 구체화 관리는 단일 미디에이터보다 계층적인 미디에이터 시스템에서 보다 효율적일 수 있다. 예를 들어, (그림 1)에서 미디에이터 1에 원격지서버 1, 2, 4의 부분질의결과가 요청되었고, 이 질의결과가 미디에이터 2와 3에서 구체화방법으로 구현되어져 있다면 원격지서버에 질의를 요청하지 않고 미디에이터 2와 3으로부터 질의결과를 빠르게 받을 수 있다. 계층적인 구조를 가지는 미디에이터 시스템에서는 미디에이터 시스템간의 다양한 프로토콜을 고려해야 하나 본 논문에서는 미디에이터 시스템에서 구체화뷰 구현기법을 제시하는 것이 목적이므로 모든 미디에이터 시스템들이 원격지서버에 직접 연결되어 있다고 가정한다. 그러나, 본 논문에서 제안하는 구체화 방법은 미디에이터 시스템간의 프로토콜이 정의되면 계층적인 구조를 갖는 미디에이터 시스템에 확장되어 적용될 수 있다. 미디에이터 시스템의 계층적인 구조는 미디에이터 시스템간의 추가적인 프로토콜로 인해 쉽고 간편하게 확장할 수 있다.



(그림 1) 미디에이터 시스템의 계층적인 구조

각 부분질의에 대해 효과적인 구현방법을 결정하기 위해서는 부분질의에 대한 사용자의 접근형태 및 관련된 기본테이블의 갱신형태를 정확하게 파악해야 한다. 각 부분질의는 부분질의의 *접근회수*와 *갱신회수*로 모델링 한다. 부분질의의 접근회수는 부분질의 내용에 접근하는 회수로써 해당

부분질의를 수행하는 회수이고, 부분질의의 갱신회수는 원격지서버의 기본테이블이 갱신된 회수이다. 부분질의의 구현계획은 부분질의의 활용패턴에 따라 결정되어지기 때문에 각 부분질의의 최근 활용패턴을 감시하는 것은 매우 중요한 일이다. 따라서, 본 논문에서는 과거에 일어난 사용자의 접근이나 갱신보다 최근에 일어난 사용자의 접근이나 갱신에 더 높은 비중을 두기 위해 감쇄율을 적용한다. 이것은 부분질의의 활용이나 부분질의의 갱신에 대해 반감기(half_life)[21]를 두어 이전회수와 최근회수에 차이를 두는 방법이다. 반감기란 부분질의의 활용이나 부분질의의 기본테이블에 대한 갱신이 영향을 50%만큼 주는 시점을 말한다. 따라서, 반감기를 h라고 할 때 감쇄율 d는 식 (1)과 같이 정의한다.

$$\text{감쇄율 } d = 2^{-\frac{1}{h}} \tag{1}$$

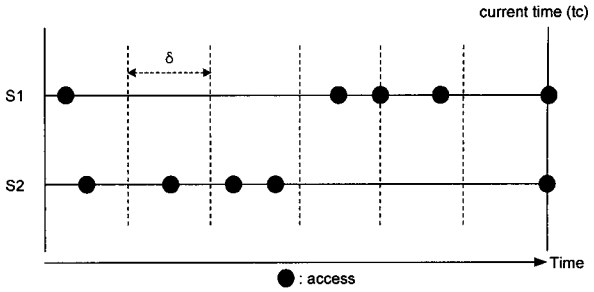
사용자가 정의한 시간간격을 δ라고 할 때 감쇄율이 적용된 이전의 접근회수와 갱신회수의 값을 다음과 같이 표현한다.

$$f^{a}_{i.new} = f^{a}_{i.old} \cdot d^{t} + 1 \quad (t = \lfloor \frac{R}{\delta} \rfloor) \tag{2}$$

$$f^{u}_{i.new} = f^{u}_{i.old} \cdot d^{t} + 1 \quad (t = \lfloor \frac{R}{\delta} \rfloor) \tag{3}$$

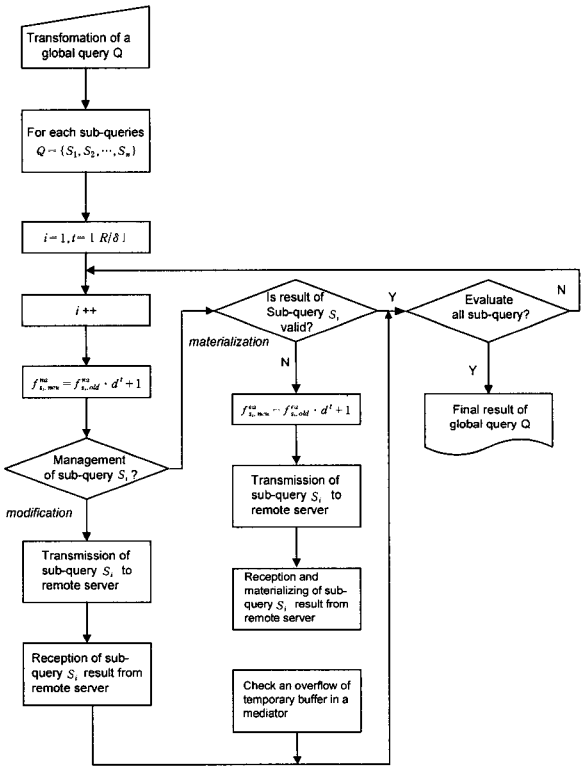
이때 $f^{a}_{i.old}$는 부분질의 s_i에 대한 가장 최근의 활용이 R시간 이전에 발생했을 때에 계산된 접근회수이고, $f^{a}_{i.new}$는 $f^{a}_{i.old}$를 기반으로 가장 최근에 새롭게 계산된 부분질의의 s_i의 접근회수이다. 이와 마찬가지로 $f^{u}_{i.old}$는 부분질의 s_i에 대한 가장 최근의 접근이 R시간 이전에 발생했을 때에 계산된 갱신회수이고, $f^{u}_{i.new}$는 $f^{u}_{i.old}$를 기반으로 가장 최근에 새롭게 계산된 부분질의 s_i의 갱신회수이다. R이 δ보다 크다면 커진 범위만큼의 감쇄율이 적용되고, 이와는 반대로 δ보다 작다면 동일한 범위 내에서 수행이 된 것으로 간주한다. 예를 들어 (그림 2)에서 부분질의 s_1과 s_2의 접근회수는 감쇄율이 적용되지 않는다면 동일하게 5로 결정되어진다. 그러나 부분질의 s_1은 부분질의 s_2보다 최근에 활용되어졌기 때문에 똑같은 회수를 반영하는 것은 정확한 결과를 사용자에게 제공하지 못한다. 따라서, 감쇄율을 적용하여 부분질의의 접근회수를 새롭게 설정한다. s_i

(그림 2)에서 반감기가 1이라고 하면 감쇄율을 적용한 부분질의 s_1의 접근회수는 $f^{a}_{s_1.tc} = 1 + 2 \cdot (2^{-1}) + (2^{-2}) + (2^{-5}) = 2.28$이고, 동일하게 부분질의 s_2의 접근회수는 $f^{a}_{s_2.tc} = 1 + 2 \cdot (2^{-3}) + (2^{-4}) + (2^{-5}) = 1.34$이다. 즉, 시간간격 δ를 어떠한 값으로 정의하는가에 따라서 부분질의의 활용패턴의 모델링 단위를 조절할 수 있다. 따라서, 시간간격 δ를 작게 설정할수록 부분질의의 활용패턴은 더욱 정확하게 모델링 된다.



(그림 2) 감쇄율이 적용된 사용자 패턴

(그림 3)은 주어진 부분질의 구현계획에 의해 미디에이터에서 관리되는 각 부분질의의 구현방법이 결정되었을 때 감쇄율을 적용한 부분질의에 대한 처리 흐름도를 나타낸다.



(그림 3) 미디에이터 시스템의 질의처리 흐름도

미디에이터에서 사용자의 질의가 발생하면 식 (2)에 의해서 해당 전역질의를 정의하는 부분질의 별로 감쇄율을 적용하여 접근회수를 증가시킨다. 이때 수정부분질의는 미디에이터에 결과가 저장되어 있지 않기 때문에 사용자의 질의가 발생할 때마다 원격지서버로부터 부분질의의 결과를 전송 받아 사용자에게 제공한다. 또한 구체화부분질의는 미디에이터에 부분질의 결과가 저장되어 있기 때문에 원격지서버에 부분질의의 실행을 요청하지 않고 신속하게 질의결과를 제공할 수 있다. 그러나 부분질의의 대상이 되는 기본테이블이 갱신 된 경우에는 새롭게 갱신된 내용이 반영된 결과를 사용자에게 제공하여야 하기 때문에 미디에이터에 구

체화되어 저장된 갱신 이전의 부분질의 결과를 무효화시키고, 식 (3)에 의해서 갱신 후에 해당 부분질의의 기본테이블에 대해 첫번 사용자의 접근이 발생할 때 해당 부분질의에 대한 감쇄율을 적용하여 갱신회수를 증가시키고, 원격지서버로부터 부분질의의 처리결과를 제공받아 사용자에게 전달하며, 동시에 새롭게 구체화시킨다.

구체화부분질의로 관리할 때 만약 구체화부분질의에 대해서 원격지서버의 기본테이블에 갱신이 일어나는 순간 미디에이터로부터 사용자가 질의결과를 얻었다면 사용자는 부정확한 결과를 얻게 된다. 따라서, 미디에이터와 원격지서버의 기본테이블간의 일관성을 유지하기 위한 방법으로 세 가지를 고려할 수 있다. 첫 번째 방법은 미디에이터에 연결되어 있는 원격지서버들의 기본테이블에 대한 모든 갱신작업을 미디에이터를 통해서만 수행하도록 하는 방법으로 사용자에게 정확한 결과를 제공하고, 일관성을 보장할 수 있으나 원격지서버의 자치성이 매우 저하되는 단점이 있다. 두 번째 방법은 원격지서버의 래퍼가 부분질의에 연관된 기본테이블에 갱신이 발생했을 경우 갱신 트랜잭션이 완료되기 전에 관련된 모든 미디에이터에게 갱신된 사실을 전달하여 관련된 모든 구체화부분질의 결과를 무효화시키는 방법이다. 이 방법은 정확한 결과를 제공하고 원격지서버의 자치성도 보장할 수 있다. 마지막으로 질의가 처리되어질 때 부분질의에 연관된 원격지서버의 기본테이블들의 결과가 유효한지를 미디에이터가 검증하는 방법으로 미디에이터에서 래퍼를 통하여 원격지서버의 기본테이블들에 대한 갱신여부의 정보를 확인함으로써 쉽게 구현되어질 수 있다.

4. 최적의 부분질의 구현계획

미디에이터 시스템의 저장공간은 제한적이며, 부분질의의 활용형태의 변화가 유동적으로 변경될 수 있으므로 한정된 저장공간에서 현재 부분질의의 활용패턴을 고려한 부분질의 구현계획의 효율성을 보장하는 것은 매우 중요한 문제이다. 따라서, 부분질의의 활용형태의 최근 변화로 인해 현재의 부분질의 구현계획이 유효하지 않다면 빠르게 새로운 최적의 부분질의 구현계획을 생성해야 한다. 이번 장에서는 부분질의의 처리비용을 기반으로 최적의 부분질의 구현계획을 생성하는 방법을 기술한다. 그리고, 현재의 부분질의 구현계획을 주기적으로 관찰하여 새로운 최적화 시점을 찾는 방법을 기술한다.

4.1 최적화를 위한 비용계산 방법

미디에이터 시스템에서 전역질의의 부분질의에 대한 일반적인 구현계획은 [정의 1]과 같이 정의 할 수 있다.

[정의 1] 부분질의 구현계획
미디에이터에 요청된 전역질의들로부터 변환된 n개의 부분질의 집합 $S=\{s_1, s_2, \cdots, s_n\}$가 주어졌을 때 미디에이터의 저장공간의 크기를 T라고 하고, 부분질의 집합 S의 전체 크기를 $Size(S)$라고 하자. 이때 구체화부분질의 집합 L_1과 수정부분질의 집합 $L_2(L_1 \cup L_2 = S,\ L_1 \cap L_2 = \emptyset,\ Size(L_1) \le T)$로 구성된 부분질의 구현계획을 $IP(L_1, L_2)$로 정의한다. ■

본 논문에서 각 부분질의의 처리비용은 감쇄율에 근거한 활용빈도를 기반으로 현재시점에서 계산되어진다. 수정부분질의는 각 부분질의 별로 활용빈도 만큼 원격지서버에서 주어진 부분질의를 처리하여 처리결과를 미디에이터에게 전달하기 때문에 수정부분질의 s_i의 비용은 식 (4)와 같이 표현된다. 이때 $size(s_i)$는 네트워크를 통해 전달되는 가장 최근의 부분질의 s_i의 결과 크기를 나타낸다. 따라서, 식 (4)에 의해서 수정부분질의의 처리비용은 감쇄율이 적용된 접근회수의 가중치에 가장 최근의 부분질의 크기의 결과 값으로 표현되어진다. 이때 원격지서버의 부분질의 처리비용이 고려되어야 하지만 네트워크를 통한 전송비용이 상대적으로 매우 크므로 고려하지 않는다. 동일하게 구체화부분질의 비용은 갱신회수만큼 결과를 전송 받기 때문에 식 (5)와 같이 계산되어진다. 부분질의 구현계획 $IP(L_1, L_2)$의 구체화비용 $C(L_1, mat)$는 식 (6)과 같이 모든 구체화부분질의의 처리비용의 합으로 나타낸다. 또한 수정비용 $C(L_2, mod)$는 식 (7)과 같이 모든 수정부분질의의 처리비용의 합으로 나타낸다. 따라서, 부분질의 구현계획 $IP(L_1, L_2)$의 전체처리비용은 식 (8)과 같이 두 비용의 합으로 표현한다

$$c(s_i, mod) = size(s_i) \cdot f_i^a \tag{4}$$

$$c(s_j, mat) = size(s_j) \cdot f_j^u \tag{5}$$

$$C(L_1, mat) = \sum_{i=1}^{k} c(s_i, mat)\ (L_1 = \{s_1, s_2, \cdots, s_k\}) \tag{6}$$

$$C(L_2, \mathrm{mod}) = \sum_{i=k+1}^{n} c(s_i, mod)\ (L_2 = \{s_{k+1}, s_{k+2}, \cdots, s_n\}) \tag{7}$$

$$Cost(IP(L_1, L_2)) = C(L_1, mat) + C(L_2, mod) \tag{8}$$

[정리 1] 임의의 부분질의집합 W_1과 W_2가 있을 때 $W_1 \cup W_2$의 비용은 W_1과 W_2의 비용의 합으로 표현한다. 따라서, $C(W_1 \cup W_2, I_i) = C(W_1, I_i) + C(W_2, I_i)\ (I_i \in \{mat, mod\})$이다.

증 명) $W_1 = (a_1, a_2, \cdots, a_k)$, $W_2 = (a_{k+1}, a_{k+2}, \cdots, a_n)$라고 하자.

$$C(W_1 \cup W_2, I_i) = \sum_{h=1}^{n} c(a_h, I_i)$$

$$= \sum_{h=1}^{k} c(a_h, I_i) + \sum_{j=k+1}^{n} c(a_j, I_i) = C(W_1, I_i) + C(W_2, I_i) \blacksquare$$

모든 부분질의 구현계획에 대해서 최적화의 목적은 미디에이터의 주어진 저장공간을 최대한 활용하면서 최소의 질의 처리비용을 갖는 부분질의 구현계획을 찾는 것이다. 미디에이터에서 새로운 부분질의 구현계획을 생성해야 하는 시점이 되면 미디에이터 시스템의 연결형태와 부분질의의 관리방법에 따라 식 (4)와 식 (5)에 의해서 각 부분질의의 두 가지 구현방법 및 그에 따른 처리비용을 계산한다. 이를 기반으로 각 부분질의의 구현방법을 결정하는 중요한 요소인 *이득*을 계산한다. 각 부분질의는 수정부분질의의 질의처리비용과 구체화부분질의의 질의처리비용의 차가 클수록 구체화부분질의로 관리하는 것이 처리비용이 적고 효율적이기 때문에 부분질의 s_i에 대한 이득은 식 (9)와 같이 수정부분질의 처리비용과 구체화부분질의 처리비용의 차로 정의한다.

$$g(s_i) = c(s_i, mod) - c(s_i, mat) \tag{9}$$

또한, 모든 부분질의 집합 S의 이득 $G(S)$는 식 (10)과 같이 정의되며, 구현계획 $IP(L_1, L_2)$의 전체 이득은 구체화부분질의 집합 L_1의 이득과 같다.

$$G(S) = \sum_{i=1}^{n} g(s_i) \tag{10}$$

$$Gain(IP(L_1, L_2)) = G(L_1) \tag{11}$$

그러므로 구현계획 $IP(L_1, L_2)$의 전체 이득은 식 (12)와 같이 표현할 수 있다. 따라서 전체 이득이 최대가 되면 이때 전체 처리비용은 최소가 된다.

$$Gain(IP(L_1, L_2)) = C(L_1 \cup L_2, mod) - G(L_1) \tag{12}$$

[정리 2] 임의의 부분질의집합 W_3과 W_4에 대해 각 부분질의 집합의 이득은 $G(W_3)$과 $G(W_4)$이라고 할 때 두 부분질의 집합의 이득의 합 $G(W_3 \cup W_4)$은 각 부분질의 집합의 이득 $G(W_3)$과 $G(W_4)$의 합으로 표현한다. 따라서, $G(W_3 \cup W_4) = G(W_3) + G(W_4)$이다.

증 명) $W_3 = (a_1, a_2, \cdots, a_k)$, $W_4 = (a_{k+1}, a_{k+2}, \cdots, a_n)$라고 하자.

$$G(W_3 \cup W_4) = \sum_{i=1}^{n} g(a_i) = \sum_{i=1}^{k} g(a_i) + \sum_{j=k+1}^{n} g(a_j)$$
$$= G(W_3) + G(W_4) \quad \blacksquare$$

그러나, 이득은 각 부분질의의 결과를 구체화하는데 필요한 저장공간의 활용도를 고려하지 않는다. 따라서, 이득으로만 부분질의의 구현방법을 결정하게 되면 용량이 큰 부분질의가 구체화부분질의가 될 확률이 높기 때문에 미디에이터의 한정된 저장공간의 활용도를 고려하여 이득을 부분질의의 크기로 정규화한다. 이를 정규이득(ng)이라고 정의하고, 부분질의 s_i의 정규이득은 식 (13)과 같이 계산한다.

$$ng(s_i) = g(s_i)/size(s_i) \tag{13}$$

4.2 부분질의 실제구현계획

미디에이터에서는 부분질의의 활용패턴과 기본테이블의 갱신패턴에 대해 빠르게 적응하여 새로운 부분질의 구현계획이 생성되어야 한다. 그러나, 기존에 구체화선택방법[9-11]들은 각 부분질의를 구현할 수 있는 두 가지 방법을 모두 고려하여 생성할 수 있는 모든 부분질의 구현계획의 비용을 비교하기 때문에 부분질의의 수가 증가될수록 탐색공간이 기하급수적으로 증가되어 최적화과정을 수행하는데 많은 시간이 필요하며, 미디에이터 시스템의 컴퓨팅 능력을 많이 사용한다. 부분질의 구현계획을 통해 부분질의 구현방법이 결정되었을 때의 부분질의 활용패턴이 부분질의 구현계획을 시작할 당시의 부분질의 활용패턴과 같지 않을 수 있다. 이는 최적화 수행시간이 길어지면 자주 변경되는 부분질의 활용형태에 빠르게 대처하지 않으므로 최근의 부분질의 활용형태를 반영하지 못하여 최적의 부분질의 구현계획인지를 확신할 수 없다. 따라서, 본 논문에서는 한번의 조합으로 최적의 부분질의 구현계획을 생성하기 위하여 부분질의 실제구현계획을 [정의 2]에서 정의한다.

[정의 2] 실제구현계획(Practical Implementation Plan) 부분질의집합 S와 미디에이터의 저장공간 T가 있을 때, S의 모든 부분질의에 대해 정규이득의 크기순으로 정렬된 순서 리스트를 $ngl(S) = \{s_1, s_2, \cdots, s_i, \cdots, s_n\}$이라고 하자. 따라서, 순서리스트 $ngl(S)$는 $ng(s_i) \geq ng(s_{i+1})$, $s_i, s_{i+1} \in S$의 관계가 성립하고, 만약 두 개 이상의 부분질의의 정규이득 크기가 같다면 결과 크기가 큰 부분질의가 더 큰 순위를 갖는다. 실제구현계획 $PIP(L_1^*, L_2^*)$은 다음의 순서에 따라 결정된다. 이때 순서리스트 $ngl(S)$는 배열 $NGL[1..n]$에 의해서 정의되고, $NGL[i]$는 i번째로 큰 정규이득을 갖는 부분질의 s_i를 의미한다.

[단계 1] $L_1^* = \emptyset$

[단계 2] For $i = 1$ to n do

if $size(NGL[i]) \leq T - Size(L_1^*)$ then

$L_1^* \leftarrow L_1^* \cup \{s_i\}$

[단계 3] $L_2^* \leftarrow S - L_1^*$ ■

실제구현계획의 구체화부분질의들 중에는 [정의 3]과 같이 항상 구체화부분질의들로 구현되어야 할 부분질의들을 *확정 구체화부분질의*로 정의한다.

[정의 3] 확정 구체화부분질의(Definite Materialized Sub-queries)

부분질의집합 S의 실제구현계획 $PIP(L_1^*, L_2^*)$에 대해 확정

구체화부분질의 DS는 구체화부분질의 집합 L_1^*의 부분집합으로 [정의 2]에서 정의된 두 개의 순서리스트 $ngl(L_1^*)$와 $ngl(S)$의 공통적인 전위(prefix) 부분질의들의 집합이다. ■

따라서, 구체화부분질의 집합 L_1^*의 확정 구체화부분질의들은 수정부분질의 집합 L_2^*의 가장 큰 정규이득보다 더 큰 정규이득을 갖는다.

[정리 3] 실제구현계획 $PIP(L_1^*, L_2^*)$의 부분질의 집합 S에 대해서 구체화부분질의집합 L_1^*이 모두 확정 구체화부분질의이고, L_1^*의 크기가 주어진 미디에이터의 저장공간 크기와 같다면 실제구현계획 $PIP(L_1^*, L_2^*)$는 최적이다. 즉, $L_1^* = DS$이고, $Size(L_1^*) = T$이면 $PIP(L_1^*, L_2^*)$는 최적이다.

증명) 부분질의집합 S의 실제구현계획 $PIP(L_1^*, L_2^*)$가 위의 조건들을 만족한다고 하자. 그리고, $max_ng(S)$는 부분질의 집합 S의 최대 정규이득이라고 하고, $min_ng(S)$는 최소정규이득이라고 하자. 이때 L_1^*의 멱집합 a와 L_2^*의 멱집합 b에 대해 $L_1 = L_1^* - a \cup b$와 $L_2 = L_2^* - b \cup a$를 만족하는 임의의 구현계획 $IP(L_1, L_2)$이 있다고 하자.

$L_1^* = DS$이므로

$$min\_ng(a) > man\_ng(b) \quad ①$$

$Size(L_1^*) = T$이므로 [정의 2]에 의해서,

$$Size(a) \geq Size(b) \quad ②$$

식 ①과 식 ②에 의해서,

$$G(a) > G(b) \quad ③$$

따라서, [정리 2], 식 ③과 식 (11)에 의해서

$$\begin{aligned} &Gain(PIP(L_1^*, L_2^*)) - Gain(IP(L_1, L_2)) \\ &\quad = G(L_1^*) - G(L_1^* - a \cup b) \\ &\quad = G(L_1^*) - G(L_1^*) + G(a) - G(b) > 0 \end{aligned}$$

따라서, 실제구현계획 $PIP(L_1^*, L_2^*)$는 최적이다. ■

그러므로, 실제구현계획 $PIP(L_1^*, L_2^*)$의 효율성은 구체화부분질의 집합 L_1^*에 대한 확정 구체화부분질의 DS의 크기의 비율에 의해서 결정된다. 따라서, 정리 3에 의해서 미디에이터의 가용저장공간에 대해 확정 구체화부분질의가 차지하는 비율에 따라 실제구현계획이 활용되어지므로, 확정 구체화부분질의가 차지하는 비율이 커질수록 실제구현계획 $PIP(L_1^*, L_2^*)$의 효율성이 향상된다.

4.3 최적화의 수행시점

최적화 과정을 수행하는데 많은 시간이 필요하므로 현재 부분질의의 활용패턴에 대한 부분질의 구현계획의 처리비용의 변화를 잘 관찰하면서 주기적으로 적절한 최적화 수행시점을 찾는 것은 매우 중요한 과제이다. 따라서, 각 부분질의의 활용패턴이 자주 변하게 되면 현재 부분질의 구현계획의 유효성이 저하되기 때문에 부분질의의 활용패턴에 따라서 부분질의 구현방법을 재결정해야 한다. 부분질의 구현계획으로 구현되어 있는 각 부분질의별로 접근회수가 갱신회수보다 크다면 기본테이블의 갱신보다 부분질의의 활용빈도가 많이 발생한 것이기 때문에 구체화부분질의로 관리하는 것이 더 효율적이고, 갱신회수가 접근회수보다 크다면 부분질의의 활용빈도보다 기본테이블의 갱신이 자주 발생한 것이기 때문에 수정부분질의로 관리하는 것이 더 효율적이다. 따라서, 수정부분질의는 접근회수의 변화가 갱신회수의 변화보다 크다면 구체화부분질의로 변경하는 것이 더 효율적이고, 구체화부분질의는 갱신회수가 접근회수에 근접할수록 수정부분질의로 변경하는 것이 더 효율적이다. 만약 현재 부분질의 구현계획의 처리비용이 더 이상 유효하지 않다면 새로운 부분질의 구현계획을 빠르게 생성하여야 한다. 현재 부분질의 구현계획 $IP(L_1, L_2)$에 대해 $Cost^{latest}(IP(L_1, L_2))$는 부분질의 구현계획의 가장 최근에 계산되어진 처리비용이라고 정의하고, $Cost^{monitoring}(IP(L_1, L_2))$는 가장 최근의 감시방법에 의해서 얻어진 처리비용이라고 정의하며, 새로운 부분질의 구현계획은 식 (12)를 만족할 때 최적화 시점을 찾는다.

$$\left| \frac{Cost^{latest}(IP(L_1, L_2)) - Cost^{monitoring}(IP(L_1, L_2))}{Cost^{latest}(IP(L_1, L_2))} \right| \geq \mu \quad (12)$$

이때 μ는 비용허용율을 의미하며, 사용자가 정의하는 파라미터로써 최근의 부분질의 구현계획의 처리비용과 감시방법에 의해 얻어진 처리비용의 허용 비율을 의미한다. 따라서, 비용허용율 μ를 조절함으로써 새로운 최적화 수행시점을 조절할 수 있다. 비용허용율 μ의 값이 작을수록 새로운 최적화 과정은 자주 발생하며, 새로운 최적화는 비용허용율 μ를 초과하는 시점에서 새롭게 시작된다.

이러한 이유 이외에도 구체화부분질의의 연관된 원격지서버의 기본테이블에 갱신이 발생했을 때 얻어진 새로운 질의결과의 크기가 이전의 질의결과의 크기보다 커서 미디에이터의 가용공간을 초과한 경우 새로운 부분질의 구현계획을 생성해야 한다. 이러한 이유 때문에 미디에이터의 저장공간이 조금이라도 초과 될 때마다 불필요한 최적화를 자주 수행하게 되어 전체적인 효율성을 감소시킨다. 이를 방지하기 위해 미디에이터에 임시저장공간을 추가하는 방법이 있다. 만약 기본 테이블의 갱신으로 인해 부분질의의 결과크기가 이전 결과크기보다 증가하여 미디에이터의 저장공간에 적

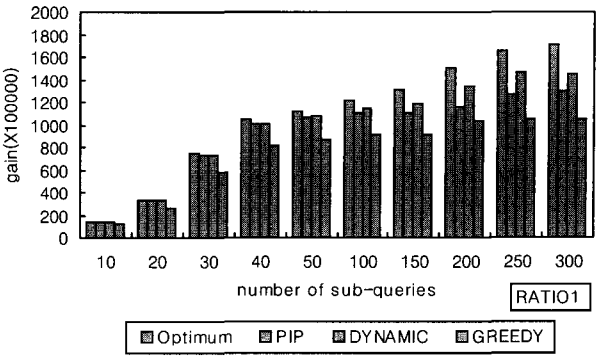
은 양의 초과가 발생한 경우 임시버퍼를 사용하여 다음 최적화가 발생할 때까지의 결과를 저장하여 두는 방법이다. 따라서, 불필요한 최적화의 수행을 방지할 수가 있다. 반면에 기본테이블의 갱신으로 인해 부분질의 결과크기가 이전 결과크기보다 감소한 경우에는 수정부분질의 중에서 정규 이득이 크고 미디에이터의 남은 저장공간에 알맞은 가능한 부분질의의 구현방법을 구체화방법으로 변경한다.

5. 실험 및 결과 분석

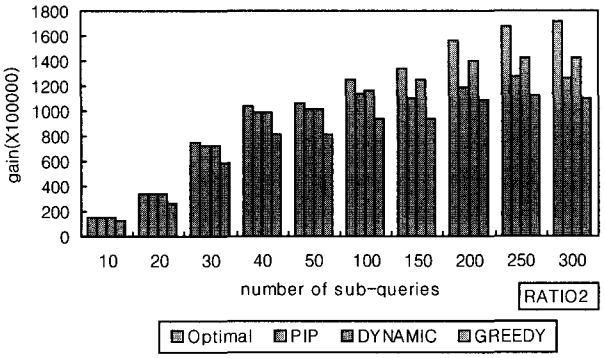
본 논문에서는 제안된 알고리즘을 다양한 관점에서 평가하기 위하여 여러 가지 실험을 수행하였다. 본 장에서는 특별한 설명을 하지 않으면 대부분의 실험에서 300개의 부분질의를 사용하였으며, 미디에이터의 전체 저장공간은 3GB로 설정하였고, 임시저장공간은 미디에이터의 전체 저장공간의 20%를 사용하였다, 또한 각 부분질의의 초기 결과 크기는 10MB에서 200MB사이에서 자유롭게 생성하였으며, 부분질의의 활용빈도와 기본테이블의 갱신회수에 대한 다양한 환경의 결과를 제공하기 위하여 서로 다른 두 종류의 로그로 실험하였다. 첫 번째 로그는 사용자의 접근회수와 기본테이블의 갱신회수의 비율이 80 : 20인 것으로써 *RATIO1*이라고 하고, 또 다른 로그는 50 : 50의 비율을 갖는 것으로써 이를 *RATIO2*라고 하였다. 각 로그의 부분질의들은 3000번의 접근 및 갱신회수를 위의 비율에 따라 자유롭게 적용하였다. 마지막으로 각 부분질의들에 갱신이 발생했을 경우 이전 부분질의 결과의 0.1배에서 10배 사이에서 자유롭게 생성되도록 하였으며, 실험의 정확성을 위해서 각 실험들은 10번씩 수행하여 평균값으로 표현하였다.

본 논문에서 제안하는 실제구현계획의 효율성을 표현하기 위하여 제안된 알고리즘은 일반적인 분기한정방법(branch-and-bound method), 탐욕적인 방법(greedy method), 동적 프로그래밍(dynamic programming) 세 가지의 최적화 방법과 결과를 비교하였다. 세 가지 방법 중에서 분기한정방법은 최적의 계획을 항상 찾는다. (그림 4)는 부분질의 개수에 따른 평가 이득의 차이를 나타낸다. 제안된 알고리즘 PIP는 탐욕적인 방법보다는 더 많은 이득을 얻지만 동적프로그래밍 방법보다는 더 적은 이득을 얻는 것을 알 수 있다.

(그림 5)는 다양한 최적화 방법들의 질의 처리비용과 최적화 계획의 질의 처리비용과의 평균 차이율을 나타낸다. 유사 데이터에 대한 연속적인 최적화 결과가 아닌 단일 최적화 과정에 대한 비용만을 비교하기 위하여 *RATIO1* 로그와 *RATIO2* 로그 각각에 대해 첫 번 최적화 과정의 처리비용만으로 비교하였다. 이때 모든 부분질의를 수정 부분방법으로만 관리하는 기존의 미디에이터 방법들을 *MOD*로 표현하였다. 평균적으로 *RATIO1* 로그에서 제안된 알고리즘 PIP는 수정방법으로만 관리할때의 비용보다 54%의 감소를 보

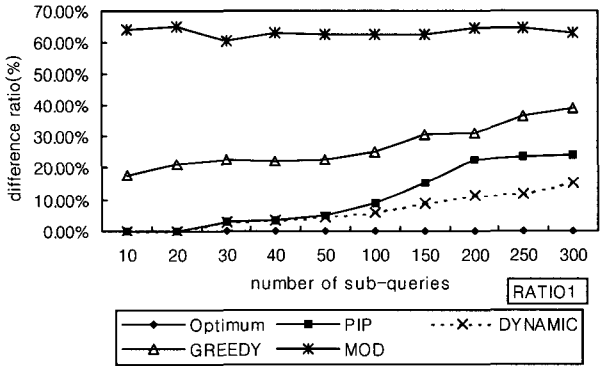


(a) RATIO1

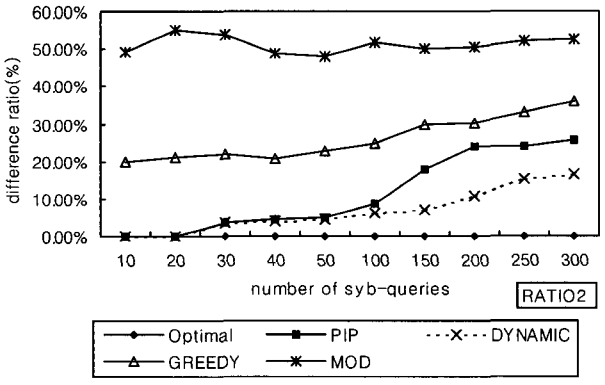


(b) RATIO2

(그림 4) 평가이득의 차이비교



(a) RATIO1

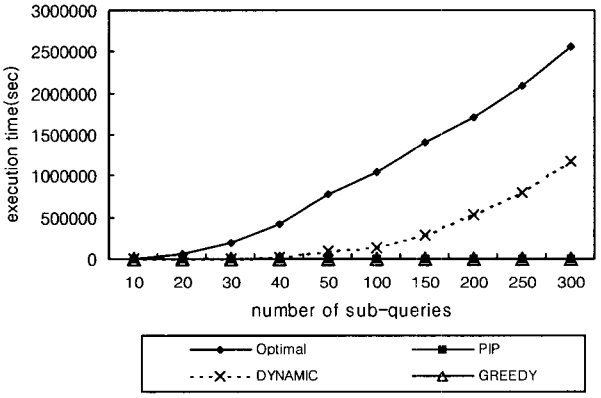


(b) RATIO2

(그림 5) 전체 평가비용의 차이율

였으며, 제안된 알고리즘과 최적화 계획과는 18%의 차이를 보였다. 또한 *RATIO2* 로그에서는 제안된 알고리즘 PIP와 수정방법으로만 관리할때의 비용보다 40%의 감소를 가져왔으며, 제안된 알고리즘과 최적화 계획과는 20%의 차이를 보였다. 따라서, 갱신비율이 증가할수록 구체화 방법의 효과는 감소한다. 제안된 알고리즘에서 모든 구체화부분질의들에 대해 평균적으로 확정 구체화부분질의의 크기는 *RATIO1* 로그에서는 90%, *RATIO2* 로그에서는 84%를 차지하였다.

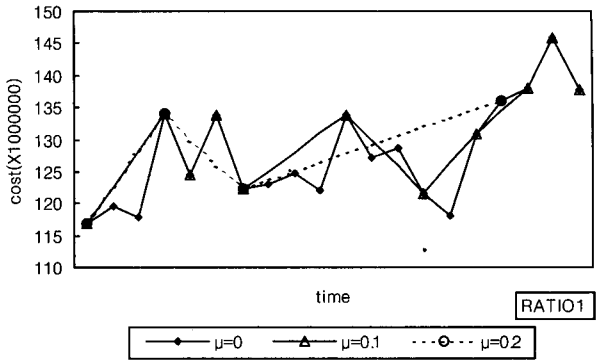
(그림 6)은 (그림 4) 및 (그림 5)와 같은 실험 환경에서의 최적화 수행시간의 비교를 나타낸다. 부분질의 개수가 증가할수록 최적화 수행시간의 변화가 크게 발생하는 것을 나타낸다. 비록 동적프로그래밍 방법이 (그림 5)에서 보듯이 최적화 계획과 더 가까이 있지만 최적화를 수행하는데 걸리는 시간은 부분질의 개수가 증가함에 따라서 기하급수적으로 늘어나는 것을 알 수 있다. 또한 탐욕적인 방법은 제안된 알고리즘 *PIP*와 수행시간은 거의 비슷하나 (그림 5)에서 보듯이 처리비용이 *PIP*보다 더 많다는 것을 알 수 있다. 평균적으로 *RATIO1* 로그와 *RATIO2* 로그의 수행시간은 거의 동일하다.



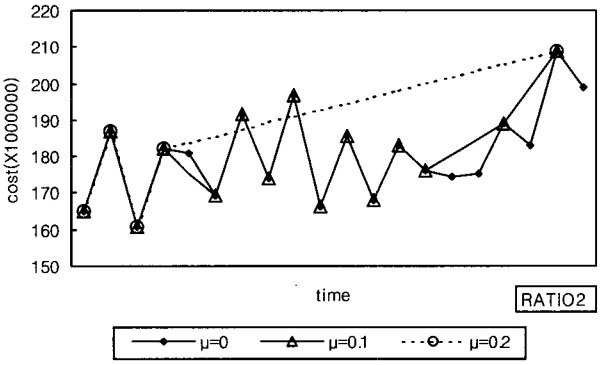
(그림 6) 최적화 수행 시간의 비교

(그림 7)은 *RATIO1* 로그와 *RATIO2* 로그에서 각 질의에 대한 연속적인 사용자 패턴의 변화에 따라 제안된 알고리즘의 처리비용의 변화를 나타낸다. 비용허용율 μ가 작을수록 새로운 최적화 과정이 자주 수행되기 때문에 뷰의 활용패턴의 최근 변화에 보다 근사하게 적응함을 보여준다.

반면에 제안된 알고리즘 PIP는 미디에이터의 임시버퍼가 초과할때마다 최적화를 수행한다. (그림 8)은 임시버퍼의 크기에 따른 최적화 수행회수의 비율을 나타낸다. 임시버퍼의 크기는 미디에이터의 저장공간 크기의 비율로 표현하였다. 따라서, 임시버퍼가 없을 때 임시버퍼의 비율은 0%이다. 초과율은 전체 최적화 과정이 수행된 회수에 대해 임시버퍼의 용량초과로 인해 수행된 회수의 비율로 나타낸다. (그림 8)에서 보듯이 임시버퍼의 용량이 증가할수록 저장공간의 부족으로 인해 발생하는 최적화 수행 회수가 많이 감

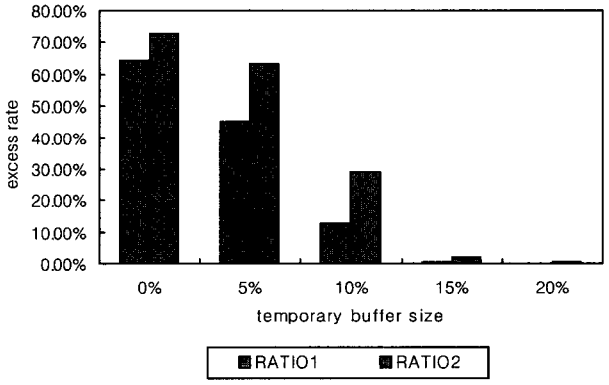


(a) RATIO1

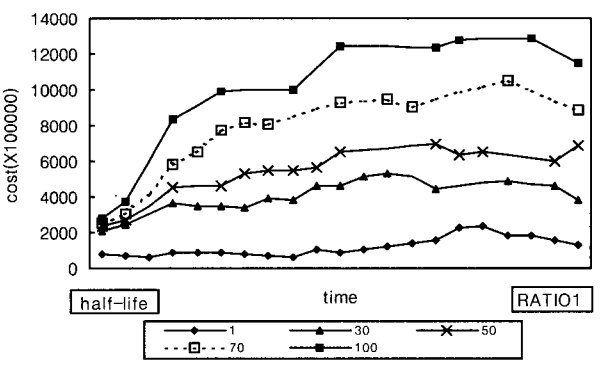


(b) RATIO2

(그림 7) 비용허용율 μ의 효과



(그림 8) 임시버퍼에 따른 비용초과 비율



(그림 9) 반감기의 효과

소된다. 만약 임시 버퍼가 전체 저장공간의 20%가 된다면 용량초과로 인한 최적화가 거의 발생하지 않는다. 또한 감쇄율의 효과를 보기 위하여 제안된 (그림 9)에서는 알고리즘에 대해 감쇄율의 효과를 나타내었다, *RATIO1* 로그에서 부분질의의 연속적인 부분질의의 활용패턴의 변화에 대해 각각 반감기를 다르게 적용했을 때의 연속적인 최적화 과정의 수행변화를 나타내며, *RATIO2* 로그에서도 거의 같은 결과를 나타낸다. (그림 9)에서 보듯이 반감기가 작을수록 최적화 과정이 자주 수행되어진다. 따라서, 반감기가 작을수록 최근의 부분질의의 활용패턴의 변화에 더 높은 가중치를 부여하게되어 최근의 부분질의의 활용패턴에 민감하게 적응할 수 있다.

6. 결 론

미디에이터 시스템에서 정의되어 있는 전역질의들을 구체화 방법으로 구현할 경우 질의 처리의 효율성을 크게 향상시킬 수 있다. 따라서 본 논문에서 제안된 알고리즘은 각 부분질의들의 활용패턴을 주기적으로 관찰함으로써 정확한 최적화 과정의 수행시점을 결정할 수 있다. 또한 부분질의의 활용패턴에 대해 감쇄율을 적용함으로써 최적화가 수행될 때 항상 최근의 활용패턴에 민감하게 반응하여 실제적으로 적응할 수 있는 부분질의 구현계획을 생성할 수 있다. 최적화 계획은 부분질의들의 모든 가능한 구현계획을 모두 검사하여 찾아야 하기때문에 최적의 부분질의 구현계획을 찾는데 많은 시간이 소비되므로 비효율적이다. 그러나 본 논문에서 제안된 알고리즘은 부분질의의 수가 크게 증가하여도 최적의 부분질의 구현계획에 근사한 부분질의 구현계획을 빠른 시간에 구현할 수 있다. 따라서, 본 논문에서 제안된 알고리즘으로 구현되어진 구체화부분질의들은 미디에이터 저장공간의 효율성을 최대로 활용하기 때문에 실제적으로 매우 효율적이나 항상 최적의 상황이 아닐 수도 있다. 또한 미디에이터에서 임시 저장공간을 사용함으로써 불필요하게 발생하는 최적화 과정의 수행회수를 줄일 수 있으며, 비용허용율 μ를 사용하여 새로운 최적화 과정의 수행시점을 결정할 수 있다. 마지막으로 본 논문에서 제안하는 방법을 다양한 실험을 통하여 증명하였으며, 본 논문에서 제안하는 최적화 알고리즘을 통해 부분질의 구현계획을 구현하는 것이 처리비용과 시간의 절감을 가져오기 때문에 매우 효율적이라는 결론을 얻을 수 있다.

참 고 문 헌

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

[2] 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.

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

[4] 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.

[5] 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.

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

[7] 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," http://www.cs.umd.edu/projects/hermes/*overview/paper*.

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

[9] H. Gupta. "Selection of view to materialized in a data warehouses," *ICDT*, 1997.

[10] J. Yang, K. Karlapalem, Q. Li, "Algorithms for materialized view design in data warehousing environment," *VLDB '97*, pp.136-145.

[11] A. Y. Levy, A. Rajaraman and J. J. Ordille, "Querying Heterogeneous Information Source Using Source Description," *VLDB*, pp.251-262, 1996.

[12] S. Chaudhuri, Krishnamurthy, S. Potamianos and K. shim, "Optimizing Queries with Materialized Views," *ICDE*, pp.190-200, 1995.

[13] Ashish Gupta, Inderpal Singh Mumick, "Maintenance of Materialized View : Problems, Techniques, and Applications," *Proc. od Intl Conf, on Data Engineering*, pp. 86-93, 1990.

[14] V. Harinarayan, A. Rajaraman and J. Ulman. "Implementing data cubes efficiently," *ACM SIGMOD International Conference of Management of Data*, Canada, June, 1996.

[15] S. Agrawal, R. Agrawal, P. M. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan, S. Sarawagi, "On the Computation of Multidimensional Aggregates," *VLDB*, pp.506-512, 1996.

[16] Chuan Zhang, Xin Yao, Jian Yang, "An Evolution Ap-

proach the materialized view selection in a data waregouse environment," *IEEE Trans. On systems, man and cybernetics*, part C, Vol.31, No.3, Setp., 2001.

[17] A. Y. Levy, A. O. Mendelzon, Y. Sagiv and D. Srivastava. "Answering Queries Using Views," *ACM SIGMOD*. pp. 95-104, 1995.

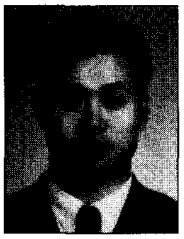
[18] H.Gupta and I. S. Mumick, "Selection of views to materialize under a maintenance cost constraint," *International Conference on Database Theory (ICDT)*, pp.453-470, 1999.

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

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

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

[22] 주길홍, 이원석, "미디에이터 시스템에서의 이질 분산데이터베이스의 통합을 위한 효율적인 뷰 관리 방법", 정보과학회논문지, Vol.28, No.4, December, 2001.



주 길 홍

e-mail : faholo@amadeus.yonsei.ac.kr

1998년 인천대학교 전자계산학과(공학사)

2000년 연세대학교 컴퓨터과학과(공학석사)

2000년~현재 연세대학교 컴퓨터과학과 재학중(박사과정)

관심분야 : 분산데이터베이스, 미디에이터 시스템, 분산질의처리, 질의최적화, 웹 데이타베이스마이닝

이 원 석

e-mail : leewo@amadeus.yonsei.ac.kr

1985년 미국 보스턴대학교 컴퓨터공학과 (공학사)

1987년 미국 퍼듀대학교 컴퓨터공학과 (공학석사)

1990년 미국 퍼듀대학교 컴퓨터공학과 (공학박사)

1990년~1992년 삼성전자 선임연구원

1993년~1999년 연세대학교 컴퓨터과학과 조교수

1999년~현재 연세대학교 컴퓨터과학과 부교수

관심분야 : 분산데이터베이스, 미디에이터시스템, 데이터마이닝, 침입탐지, 멀티미디어데이터베이스