데이터 웨어하우스 뷰(View) 유지관리를 위한 시스템 설계

유경재*, 한영태*, 민덕기*
건국대학교 컴퓨터·정보통신공학과
e-mail : {kjyou,ythan,dkmin}@cse.konkuk.ac.kr

A System Design for Data Warehouse View Maintenance

Kyoungjae You*, Youngtae Han*, Dugki Min*
Dept. of Computer Science and Engineering, Konkuk University

요약

데이터 웨어하우스는 다양하고 서로 다른 저장소로부터 데이터를 수집하고 통합하는 역할을 담당한다. 그리고 의사결정 지원이나 OLAP 질의에 대한 효과적인 구현을 목적으로 하나 또는 그 이상의 저장소의 데이터에 대한 구체화된 뷰들을 저장 관리한다. 데이터 웨어하우스를 둘러싸고 있는 화제로는 구조, 알고리즘, 그리고 다양한 데이터베이스들 또는 다른 정보 저장소들로부터 데이터를 추출하여 가져와 하나의 저장소(Data Warehouse)에 저장하는 툴들이 있다. 최근 몇 년간 데이터 웨어하우스는 데이터베이스 업계에서 두드러진 분야로 성장하였다. 그러나 데이터베이스 연구 공동체는 제한되어 왔다. 본 논문에서는 일반적인 데이터 웨어하우스 구조의 윤곽을 잡고 데이터 웨어하우스를 위해 데이터를 수집하고 변형하고 통합하는 시스템 설계를 목적으로 한다.

1. 서론

21세기 정보기술의 화두는 시스템 통합이다. 데이터 웨어하우스는 과거 데이터의 분석에만 그치는 것이 아니라 데이터 마이닝(Data Mining), ERP(Enterprise Resource Planning), CTI(Computer Telephony Integration), EC(Electronic Commerce), KMS (Knowledge Management System) 등 다양한 기술과 접목되면서 21세기 정보기술의 중심으로 자리잡을 것이다[5].

데이터 웨어하우스를 한마디로 정의하면 기업이 경쟁력 향상을 위해서 신속하고 정확한 의사결정을 할 수 있도록 지원해주는 시스템이다. 대개의 기업에서는 거의 매일 운영계(OLTP) 시스템을 운영하면서 생긴 트랜잭션 데이터를 마그네틱 테이프(MT)에 담아두는 작업을 하고 있다. 데이터 웨어하우스는 이 데이터를 테이프 대신 디스크에 담아두자는 간단한 생각에서 출발한다. 데이터 웨어하우스는 단순한 데이터의 저장고가 아니라 관계형 데이터베이스를 근간으로 많은 데이터를 다차원적으로 신속하게 분석하여 의사결정에 도움을 주기 위한 시스템이다[5, 6].

<그림 1>에서는 기업에서 일반적으로 구성하고 있는 운영계 데이터베이스와 데이터 웨어하우스의 기본 구성도를 나타내고 있다. 데이터 웨어하우스는 <그림 1>에서 보이듯이 운영계 시스템과는 별도로 구축하는 것이 일반적이다. 데이터 웨어하우스를 구축하기 위해서는 데이터 웨어하우스를 구성하는 다음과 같은 3가지 분야 기술[5, 6]이 필요하다.

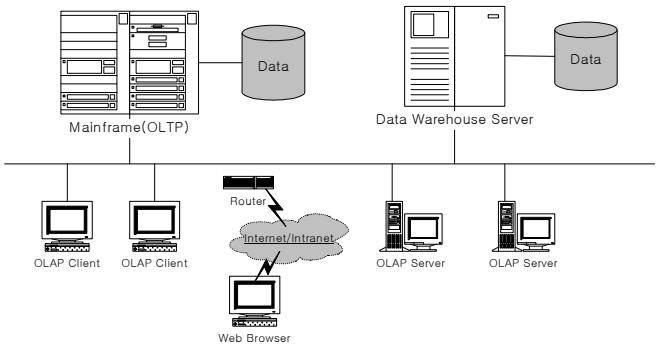

그림 1 데이터 웨어하우스 기본 구성도

- 데이터의 추출과 로드
  데이터는 운영계 시스템 에서 추출하여 데이터 웨어하우스 서버로 로드 되어야 한다.
- 다차원 모델링(Multi-Dimensional Modeling)
  이렇게 데이터 웨어하우스 서버에 올려진 데이터는 데이터 웨어하우스에서만 적용되는 다차원 모델링을 해야 한다.
- End User Access
  그리고 다차원 모델링이 되어 있는 데이터 웨어하우스 서버를 여러 가지 방식과 도구로 액세스 할 수 있어야 한다.

본 논문에서는 일반적인 데이터 웨어하우스 구조의 윤곽을 잡고 위에서 설명한 데이터 웨어하우스를 구축하기 위해 필요한 3가지 분야 기술 중 데이터의 추출과 로드에 대한 부분을 중점적으로 다룬다. 데이터의 추출과 로드는 기본적으로 서로 다른 저장소로부터 데이터를 가져 온다. 데이터를 추출할 때 바탕이 되는 것은 필요한 데이터에 대해서 정의되어진 즉, 시스템 관리자에 의해 정의되어진 뷰(View)들을 이용하는 것이다. 뷰들은 저장소(Source)의 스키마(Schema)에 밀접한 관련이 있다. 만약 저장소의 스키마의 변경사항이 발생하게 되면 해당 리소스의 스키마에 대한 뷰의 내용도 바뀌어야 한다. 이러한 데이터 추출/변환/정련을 통하여 뷰를 유지관리하는 방법론으로 스탠포드 대학에서 WHIPS 시스템 구조를 제안하였다. WHIPS 시스템의 문제점으로는 다양하고 서로 다른 저장소들에 대한 각각의 스키마 변경에 대해서는 통합 관리하지 못하고 있다. 본 논문에서는 WHIPS 시스템의 문제점을 보완하고 뷰를 유지관리하기 위한 데이터 웨어하우스 시스템 구조를 제안할 것이다.

2장에서는 데이터 웨어하우스 프레임워크와 데이터 웨어하우스 특성과 문제점에 대해서 알아보고, 3장에서는 WHIPS 시스템의 구조와 스키마 통합관리에 대한 문제점을 논한다. 4장에서는 데이터 웨어하우스 뷰(View) 유지관리를 위한 시스템 구조를 제안한다. 5장에서는 결론을 내린다.

2. 데이터 웨어하우스

데이터 웨어하우스란 한 기업이 운영(Operational)데이터베이스와는 별도로 유지되는 의사결정 지원을 위한 데이터베이스로 볼 수 있다. 본 절에서는 데이터 웨어하우스 프레임워크에 대해서 살펴보고, 데이터 웨어하우스의 특성을 파악하는 것은 중요하다. 이 내용은 4장에서 다룰 데이터 웨어하우스 뷰 유지관리를 위한 시스템 설계에서 데이터 웨어하우스 프레임워크와 데이터 웨어하우스의 특성을 고려하고 있기 때문이다.

2.1 데이터 웨어하우스 프레임워크

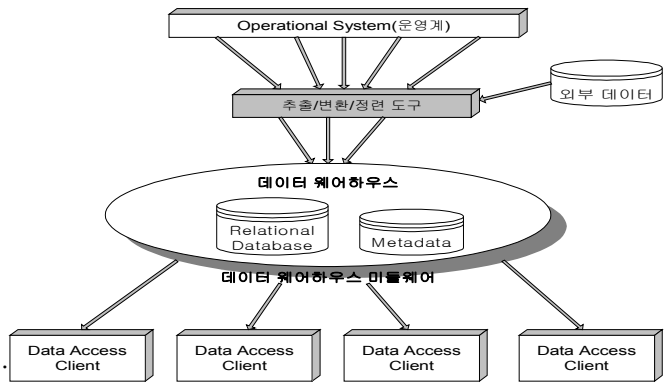

그림 2 데이터 웨어하우스 프레임워크

<그림 2>에서는 데이터 웨어하우스의 프레임워크를 보여 주고 있다. 데이터 접근 클라이언트는 일반 사용자, 지식 근로자(Knowledge Worker:임원, 관리자, 재무분석 담당자, 마케팅분석 담당자 등)이 OLAP 도구를 이용하여 업무를 수행하는 것을 나타내고 있다. 데이터 웨어하우스 미들웨어에서는 사용자의 요구에 대해 데이터 웨어하우스에 저장되어 있는 데이터를 이용하여 요구에 해당하는 정보를 만들어 사용자에게 보여 준다. <그림 2>에서 데이터 웨어하우스내의 관계형 데이터베이스(Relational Database)는 운영계로부터 추출한 데이터들을 저장한다. 그리고 메타데이터(Metadata)는 저장소에 대한 정보와 저장소에 어떻게 접촉하는가에 대한 방법에 대한 목록을 저장하고 있다. 추출/변환/정련 도구란 기존의 운영계 시스템이나 외부 데이터로부터 데이터 웨어하우스에 필요한 데이터를 추출하고 추출된 데이터를 데이터 웨어하우스에 맞게 변환하고 정련시켜 데이터 웨어하우스에 저장하는 역할을 한다. 이러한 추출/변환/정련 도구에 대해 본 논문에서 좀더 자세히 다룰 것이다.

2.2 데이터 웨어하우스의 특성

앞에서 정의한 데이터 웨어하우스 정의보다 더 엄밀하게 정의하면, 데이터 웨어하우스는 주로 조직의 의사결정을 지원하기 위해 축적되는 주제 중심적이고 통합적이며 시계열 적이고 장기 존속하는 데이터의 집합이라고 할 수 있다. 이와 같은 데이터 웨어하우스의 4가지 주요 특성[6]은 다음과 같다.

1) 주제 중심(Subject Oriented)
데이터 웨어하우스에서는 주제별로 구성됨으로써 최종 사용자와 비전산전문의 분석자 등에게 데이터를 보다 이해하기 쉬운 형태로 제공할 수 있게 된다.

2) 통합적(Integrated)
데이터 웨어하우스 내부에서 데이터의 통합은 데이터의 형식, 이름 및 기타 측면에 있어서 데이터를 일관성 있게 함으로써 달성될 수 있다. 운영계 데이터베이스에서는 그 형성 과정에서 데이터의 표현 방식에 있어 불 일치성을 많이 가지게 된다.

3) 시계열 적(Time Variant Historical) 데이터
데이터 웨어하우스는 과거의 데이터와 현재의 데이터를 동시에 유지한다는 점에서 시계열 적이라고 할 수 있다. 어떤 경우에는 데이터 웨어하우스에 예측 자료나 예산과 같은 미래의 데이터가지도 포함될 수 있다.

데이터 웨어하우스에서는 통상적으로 운영계 데이터베이스로부터 일간 및 주간 단위로 데이터가 적재되어 3년~10년간 유지된다. 이러한 점이 두 가지 형태의 데이터베이스 환경간의 중요한 차이점이다.

4) 장기 지속적(Non-Volatile)
데이터 웨어하우스의 마지막 주요 특징인 장기 지속성은 데이터 웨어하우스에 데이터가 일단 적재되고 나면 주기적인 Batch작업에 의한 갱신 이외에는 정보계 데이터베이스에 대한 Insert, Delete등의 변경이 수행되지 않는다는 것을 의미한다.

물론 데이터 웨어하우스에는 처음에 운영계 데이터베이스로부터 발생한 데이터를 변환시켜 적재된 것이다. 그 이후의 데이터 웨어하우스는 다시 적재되든지 아니면 좀 더 현실적으로 주기적인 (야간 또는 주간) Batch 작업에 의하여 운영계 데이터베이스로부터 새로이 변환된 데이터를 추가시키게 된다. 이러한 Batch 갱신 작업을 제외한다면 데이터 웨어하우스는 지속적으로 불변 상태이다.

바로 이러한 데이터 웨어하우스의 장기 지속성 때문에 복잡한 질의(Query) 처리에 대하여 데이터 웨어하우스는 고도로 최적화 될 수 있다. 또한 장기 지속성은 미리 만들어 놓은 요약 자료가 상세 데이터와 일관성을 유지하는데 있어서 야기될 수 있는 문제점들을 극복할 수 있다.

2.3 데이터 웨어하스와 운영계 데이터베이스와의 분리
다음과 같은 의문을 제기할 수도 있다. 즉, 추가적으로 정보계 업무를 위해 중복적인 데이터베이스를 도입할 것이 아니고 최종 사용자에게 운영계 데이터베이스의 자료를 선별적으로 직접 액세스할 수 있도록 허용하는 것이 더 좋은 방법이라고 주장하는 것이다. 이러한 주장은 매우 중요한 쟁점으로서 충분히 이해되지 않으면 안된다.

	운영계 데이터베이스	데이터 웨어하우스
기 능	자료처리, 운영 업무 지원	의사결정 업무 지원
데 이 터	절차 중심적, 현재 값 및 시계열 자료	요약된 내용, 일부 상세자료
이용형태	구조적 정형 적 반복적	비정형(Ad-Hoc) 일부 반복적 보고서 생성 구조적 어플리케이션
처리형태	Data entry : Batch or OLTP	최종 사용자에 의한 질의

표 2 운영계 데이터베이스와 데이터 웨어하우스

<표 1>에서는 운영계 데이터베이스와 데이터 웨어하우스의 차이점을 간략하게 보여 주고 있다. 운영계에 의한 단일 시스템 방식에는 다음과 같은 여러 가지 심각한 문제점이 존재하고 있다.

- 운영계 업무 수요와 정보계 업무 수요에 대해 두 가지 요구를 동시에 만족시키면서 성능을 최적화 시킬 수 있는 방법은 존재하지 않는다.
  운영계 수요 및 정보계 수요 모두를 위해 단일 데이터베이스를 사용한다면 그 데이터베이스는 어느 한쪽의 목적으로 최적화 될 수밖에 없다. 통상 운영계 데이터베이스는 단순 갱신작업을 위해 최적화 되고 정보계 데이터베이스는 복잡한 질의에 대처하기 위해 최적화 되기 때문에 이러한 최적화 과정은 매우 중요한 의미를 갖는다.
- 데이터 형태(Format), 의미상의 불일치
  이미 지적한 바와 같이 운영계 데이터베이스는 오랜 기간 동안 서로 다른 적용 업무 개발을 위해 일관된 데이터 모델 없이 여러 명의 개발자가 구축하여 현재의 모습으로 확대됨으로써 데이터베이스내의 데이터 형태와 의미가 종종 불일치하게 되었다. 대부분의 경우, 이러한 의미상의 불일치성은 여러 가지 주제에 걸친 분석 작업이나 회사 조직 전체에 걸친 질의 요구를 매우 숙련된 전문 분석자에게도 난해하거나 불가능한 일로 만들어 버렸다.
- 시계열 데이터의 부족
  운영계 데이터베이스에는 대개 의사결정 담당자에게 꼭 필요한 시계열 적 자료가 없다. 운영계 데이터베이스에 존재하지 않는 시계열 데이터를 데이터 웨어하우스에 상세 또는 요약 형태로 저장하는 것은 중복이 아니다.

이러한 이유로 인하여 운영계와 정보계(데이터 웨어하우스)를 분리하는 것은 매우 중요하다. 이렇게 운영계와 정보계를 분리함으로서 운영계, 정보계 시스템 모두의 성능을 향상시킬 수 있다. 이러한 운영계, 정보계의 분리 구현을 전제로 효율적인 데이터 웨어하우스 시스템 설계를 3, 4장에서 다룰 것이다.

3. WHIPS 데이터 웨어하우스 시스템

본 절에서는 4절에서 제시할 데이터 웨어하우스 뷰 유지관리를 위한 시스템 디자인의 바탕이 되는 WHIPS라는 데이터 웨어하우스 시스템 프로토타입[1, 2, 3]에 대하여 설명한다.

3.1 WHIPS 시스템 개요

데이터 웨어하우스 데이터를 표현하기 위해 관계형(Relation) 모델을 사용하고 있다. 뷰들은 스키마 모델로 정의되고 데이터 웨어하우스를 스키마(Schemas)들을 저장한다. 그리고 각 저장소의 오직 하나의 스키마를 포함하고 있음을 전제로 하고 있다. 각 모듈간의 메시지 흐름은 세 가지 구별되는 동작이 있다. 첫째로, 시스템 시작시 모듈들은 다른 모듈들이 자신들을 확인할 수 있도록 해야 한다. 새로운 저장소를 등록할 때마다 이와 비슷한 동작을 수행한다. 둘째로, 뷰가 정의 될 때마다 뷰는 초기화되고 시스템은 뷰를 유지관리하기 위해 준비한다. 셋째로, 각 정의된 뷰는 뷰에 영향을 주는 수정사항에 대해 유지관리된다.

3.2 시스템 초기화와 저장소 등록
시스템 시작시 통합기(Integrator)는 자신의 식별자(Identifier)를 생성해 내고 질의 연산자(Query Processor)를 생성한다. 웨어하우스, 메타 데이터 저장소(Meta-Data Store), 그리고 View Specifier 등은 통합기에 등록한다. 각 저장소의 모니터(Monitor)와 랩퍼(Wrapper)도 또한 해당 저장소의 메타 데이터를 등록하기 위해 통합기와 접촉한 후 그 메타 데이터는 영구적인 메타 데이터 저장소와 질의 연산자에 전달된다.

3.3 뷰 정의와 초기화
뷰들은 시스템 관리자에 의해 View Specifier에서 정의된다. View Specifier는 메타 데이터 저장소를 이용하여 각 뷰 정의(View Definition)에 대한 타입를 체크하고 나서 통합기에 뷰 정의를 전달한다. 이 정보는 뷰를 관리하는 뷰 관리자(View Manager)의 생성을 위한 것이다. 통합기는 또한 뷰와 관련된 수정사항을 전송하는 하는 것을 시작하기 위해 뷰에 포함되어 있는 저장소 모두의 모니터에 알린다. 뷰 관리자(View Manager)는 뷰를 초기화하고 유지관리할 책임이 있다. 첫째로, 뷰 관리자는 뷰 정의에 대응하는 전체(Global) 질의를 생성한다. 생성된 전체 질의는 질의 연산자에 전송한다. 질의 연산자는 뷰에 포함된 저장소의 질의 랩퍼(Wrapper)에 접촉한다. 그리고 질의 랩퍼에 의해 되돌아오는 질의 결과물들을 조인(Join)하고 나서 그 질의 결과물을 뷰 관리자에게 전송한다. 뷰 관리자는 뷰를 초기화하기 위해 질의 결과물을 웨어하우스 랩퍼에 전송한다.

3.4 뷰 유지관리(View Maintenance)
스키마 R에 대한 모니터는 모니터가 담당하고 있는 저장소에 발생하는 R에 대한 변경사항(Modifications)을 감지한다. 그리고 그러한 변경사항을 통합기(Integrator)에 전달한다. 통합기에서는 최종적으로 뷰 관리자(View Manager)에게 스키마 R에 대한 변경사항을 통보한다. 뷰 관리자에서는 데이터 일관성을 보장하기 위해 Strobe 알고리즘을 이용하여 통합기에서 알려온 뷰의 변경사항을 기존의 뷰에 적용 변경시킨다.

3.5 각 모듈간의 통신상의 메시지 전송 방법
WHIP 시스템 구조에서 뷰 유지관리 하는 동안 각 모듈간의 통신 메시지들은 비동기 적으로 보내어진다. 이러한 것은 통신에서의 지연이 어떤 모듈에서의 처리 작업을 방해, 즉 멈추게 해서는 안된다는 것이다. 메시지들이 한 저장소로부터 뷰 관리자에 도착하는 과정은 두 가지 과정이 존재한다. 첫째로, 수정에 대한 메시지는 모니터로부터 통합기를 통해 뷰 관리자에게 전달된다. 둘째로, 질의 결과물들은 랩퍼로부터 질의 처리기를 통해 뷰 관리자에게 전달된다.

4. 데이터 웨어하우스 뷰(View) 유지관리를 위한 시스템 설계
위 3절에서 설명한 WHIPS 시스템에 대한 문제점과 그 문제점을 해결하면서 데이터 웨어하우스를 위해 사용되는 뷰들에 대한 유지관리를 위한 시스템 설계에 대해 설명한다.

4.1 WHIPS 시스템 구조의 문제점
WHIP 시스템내의 모듈간의 통신에서 두 가지 메시지 전달과정이 순서적으로 도착할 것이라는 것을 보장할 수 없다. 즉 데이터의 일관성을 보장할 수 없다는 것이다. 이러한 것을 해결하기 위해 WHIP 시스템에서는 일련 번호(Sequence Number)를 사용하였다. 각 모니터에서는 스키마(Schema)마다 카운터를 갖는다. 그리고 각 모니터에서 스키마에 대한 수정사항을 통합기에 전송할 때 일련번호를 붙여서 전송한다. 게다가 각 랩퍼에서는 질의 결과물에 해당 모니터에 의해 보내진 마지막 수정사항의 일련번호를 붙인다. 즉 랩퍼와 모니터간의 통신이 있어야 한다는 것이다.

질의 처리기에서는 하나의 스키마에 대해서 랩퍼에 의해 전송되는 일련번호를 배열을 생성한다. 뷰 관리자에서도 일련번호의 배열을 사용하여 한 스키마에 대한 마지막 변경사항을 받았을 때 그에 대한 일련번호를 유지한다.

질의 결과가 도착하였을 때 뷰 관리자는 질의 결과물의 일련번호와 자신이 유지하고 있는 일련번호를 비교한다. 만약 질의 결과물의 일련번호가 뷰 관리자에서 관리하는 해당 일련번호보다 크다면 뷰 관리자는 변경사항이 도착할 때까지 기다린다. 그리고 만약 번호가 작다면 뷰 관리자는 해당 결과물을 버리고 다시 질의를 수행하게 된다. 문제점으로는 질의 결과물에 영향을 미칠 수 있는 어떠한 변경사항에 대한 최대한의 높은 일련번호를 받기 위해 각 하나의 저장소에 대한 질의를 필요로 하는 것이다. 현실적으로 하나의 저장소에 대한 질의만 수행하는 경우는 거의 없다. 또한 뷰 관리자의 개수가 해당 저장소의 개수만큼 생성이 되어야 한다는 것이다. 뷰 관리자를 관리하는 통합기의 오버헤드가 발생하게 된다. 또한 모니터와 랩퍼간의 통신이 있어야 만이 랩퍼에서 질의 결과물을 추출한 다음 변경사항이 완료된 일련번호를 붙일 수 있는 것이다.

4.2 데이터 웨어하우스 뷰 유지관리를 위한 시스템 설계

<그림 3>에서 WHIPS 시스템상의 문제점을 해결하기 위해 제시하는 데이터 웨어하우스 뷰 유지 관리를

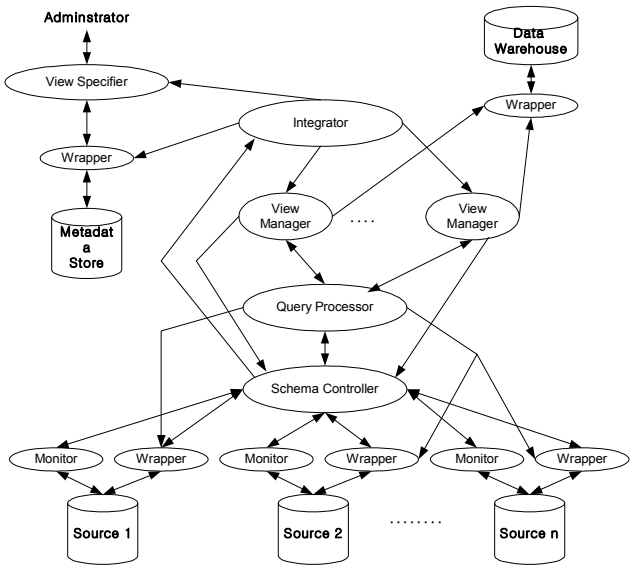

그림 3 데이터 웨어하우스 뷰 유지관리를 위한 시스템 구조

위한 시스템의 구조를 보여주고 있다. WHIPS 시스템 상의 모듈간의 통신 문제점을 해결하기 위해서는 WHIPS 시스템의 뷰 관리자와 질의 연산자에서 배열을 이용하여 각각 유지하고 있는 모든 저장소에 대한

스키마 일련번호를 통합관리할 필요가 있다. 이러한 일련번호를 통합관리하는 기능을 모듈을 스키마 제어기(Schema Contoller)에서 담당하도록 하였다. 뷰 관리자나 질의 처리기에서 배열을 유지하여 각 수정사항이나 질의 결과물에 대한 일련번호를 확인하지 않고 스키마 제어기(Schema Controller)에서 모든 저장소의 스키마에 대한 일련번호의 확인작업을 통해 질의 결과물에 대한 적합성을 결정하여 질의 결과물을 질의 연산자에 전송하게 된다.

4.2.1 시스템 초기화와 저장소 등록

시스템 시작 시 통합기(Integrator)는 자신의 식별자(Identifier)를 생성해 내고 질의 연산자(Query Processor)와 스키마 제어기(Schema Controller)를 생성한다. 웨어하우스, 메타 데이터 저장소(Meta-Data Store), 그리고 View Specifier 등은 통합기에 등록한다. 각 저장소의 모니터(Monitor)와 랩퍼(Wrapper)는 또한 해당 저장소의 메타 데이터를 등록하기 위해 스키마 통합기에 접촉 한다. 스키마 통합기는 접촉한 해당 저장소의 모니터와 랩퍼의 정보를 통합기에 전송한다. 통합기 에서는 스키마 통합기에 의해 전송된 해당 저장소의 메타 데이터를 영구적인 메타 데이터 저장소와 질의 연산자에 전달된다.

4.2.2 뷰 유지관리(View Maintenance)

스키마 R에 대한 모니터는 모니터가 담당하고 있는 저장소에 발생하는 R에 대한 변경사항을 감지한다. 그리고 그러한 변경사항을 스키마 제어기(Schema Controller)에 전달한다. 스키마 제어기는 모니터로부터 전달된 스키마 R에 대한 내용을 통합기에 전달한다. 통합기에서는 뷰 관리자에게 스키마 R에 대한 변경사항을 통보한다. 뷰 관리자에서는 통합기에서 알려온 뷰의 변경사항을 기존의 뷰에 적용 변경시킨다. 그리고 최종적으로 변경사항을 해당 뷰에 적용시켰음을 알리는 메시지를 스키마 제어기에 전송한다. 스키마 제어기에서는 관리하고 있는 저장소의 일련번호를 수정 완료한다. 그리고 스키마 R을 담당하는 랩퍼에 해당 일련번호를 전송하여 랩퍼에서 일련번호를 변경할 수 있게 한다.

4.2.3 질의 결과물에 대한 처리

질의 연산자는 뷰 관리자로부터 전송된 전체(Global) 질의에 대해 뷰에 해당하는 저장소를 담당하는 랩퍼에 질의를 전송한다. 랩퍼는 해당 질의에 대해 질의 결과물을 추출하고 해당 결과물에 스키마 R에 대한 일련번호를 추가하여 스키마 제어기에 전송한다. 만약 질의 결과물의 일련번호가 스키마 제어기의 일련번호와 일치하면 그대로 질의 연산자에 전송한다. 그러나 질의 결과물의 일련번호가 스키마 제어기의 일련번호와 일치하지 않는다면 즉, 질의 결과물의 일련번호가 낮다면 스키마 제어기에서는 질의 결과물의 내용을 무시하고 해당 랩퍼에 스키마 제어기의 일련번호를 전송하여 랩퍼의 일련번호가 수정될 때 해당 질의를 다시 수행할 수 있도록 지시한다. 이러한 과정이 가능한 것은 뷰 유지관리(4.3.2절)에서 설명한 과정이 아직 랩퍼에 도착하지 않았기 때문에 발생한 것이다. 즉, 랩퍼는 스키마 제어기로부터 변경사항이 적용된 일련번호를 받은 다음 해당 질의를 수행하게 되는 것이다. 그리고 스키마 R을 담당하고 있는 모니터와 랩퍼간의 통신이 이루어지지 않기 때문에 랩퍼에서 유지하고 있는 일련번호가 스키마 제어기에서 유지하고 있는 일련번호보다 더 높은 일련번호를 유지할 수 없다. 이러한 스키마 제어기의 구현으로 WHIPS 시스템 구조에서 문제되었던 부분을 해결할 수 있다.

5. 결론

본 논문에서는 데이터 웨어하우스를 구축하기 위해 필요한 요소 기술 중 데이터의 추출과 로드에 관련된 부분을 지원하기 위해 다양하고 서로 다른 저장소들에 대한 스키마의 정보를 뷰를 통해서 저장하게 되는데 그러한 뷰의 유지관리를 위한 데이터 웨어하우스 시스템 설계하였다.

시스템 설계에서는 다양하고 서로 다른 저장소들로부터 뷰들을 생성할 수 있다. 그리고 뷰을 유지관리할 수 있는 모듈들을 설계하였다. 시스템 설계의 핵심은 스키마 제어기로서 이 제어기를 사용함으로서 뷰를 유지관리하고 모듈간의 프로세싱에 대한 동시성(Concurrency)을 보장할 수 있게 되었다. 을 설계함으

로서 데이터 웨어하우스에서 사용하게 되는 뷰에 대한 유지관리에 있어서 효과적이고 능동적으로 대처할 수 있게 되었다.

6. 참고문헌

[1] Jennifer Widom, "Research Problems in Data Warehousing", Nov, 1995.

[2] Janet L. Wiener, Himanshu Gupta, Wilburt J. Labio, Yue Zhuge, Hector Garcia-Molina, Jennifer Widom, “A System Prototype for Warehouse View Maintenance", June, 1996.

[3] Y. Zhuge, J. L. Wiener and H. Garcia-Molina, "Multiple View Consistency for Data Warehousing." In Proceedings of the International Conference on Data Engineering, Binghamton, UK, April, 1997.

[4] Y. Zhuge, H. Garcia-Molina, J. Hammer, and J. Widom, "View Maintenance in a Warehousing Environment." In Proceedings of the ACM SIGMOD Conference, San Jose, California, May 1995.

[5] 장동인, “실무자를 위한 데이터 웨어하우스”, Jan, 1999, pp.15-30.

[6] 장동인, “실무자를 위한 데이터 웨어하우스”, Jan, 1999, pp.97-107.

[7] W. J. Labio, D. Quass, B. Adelberg. "Physical Database Design for Data Warehousing." In Proceedings of the International Conference on Data Engineering, Binghamton, UK, April, 1997.

[8] A. Kawaguchi, D. Lieuwen, I. Mumick, D. Quass, K. Ross, "Concurrency Control Theory for Deferred Materialized Views." In Proceedings of the International Conference on Database Theory, Athens, Greece, January 1997.

[9] Y. Zhuge, H. Garcia-Molina, and J. L. Wiener "The Strobe Algorithms for Multi-Source Warehouse Consistency." In Proceedings of the Conference on Parallel and Distributed Information Systems, Miami Beach, FL, December 1996.

[10] Expert systems: true support for the process of decision making; Edgar A. Whitley; Proceedings of the 1990 ACM SIGBDP conference on Trends and directions in expert systems , 1990, pp.123-140.