데이타 웨어하우스 환경에서 조인 비용을 기반으로 한 실체 뷰 선택 알고리즘

(An Algorithm for Selecting Materialized Views based on Join Cost in Data Warehouse Environments)

윤 원 식 † 신 동 천 ††
(WonSik Yoon) (DongCheon Shin)

요 약 데이타 웨어하우스 환경에서 데이타 분석을 위한 OLAP 질의에 대한 응답 시간을 줄이기 위해 실체 뷰 도입을 고려할 수 있다. 데이타 웨어하우스 환경에서 실체 뷰는 차원 테이블과 사실 테이블의 조인으로 구성되어 있는 조인 뷰로 이루어지므로 조인 비용은 실체 뷰 선택에 영향을 미치는 가장 중요한 요소이다. 본 논문에서는 실체 뷰의 차원 속성 레벨을 이용하여 조인 테이블 추적하는 방법을 정형화하고 조인 비용을 기반으로 한 실체 뷰 선택 알고리즘을 제안한다.

Abstract To reduce the response time for the On-Line Analytical Processing(OLAP) query in datawarehouses, introduction of materialized views can be considered. Since the materialized view is a join view among one fact table and many dimension tables, the join cost is one of the most important factors affecting the selection of materialized views. In this paper, we first formalize the method for tracing tables needed for the join by using the level of dimension attribute, then we propose an algorithm for selecting materialized views based on the join cost

1. 서 론

데이타 웨어하우스는 거대한 용량의 이력적 데이타를 가지고 복잡한 집계를 요구하는 의사 결정에 필요한 질의에 대하여 통합된 환경을 제공하는 데이타 저장소라고 할 수 있다[1]. 일반적으로 데이타 웨어하우스에서 데이타 분석을 수행하기 위한 질의문에는 하나 이상의 집계 함수와 group-by 연산자가 포함되므로 OLAP 응용 프로그램(예: 의사결정지원 애플리케이션)의 사용자가 요구하는 응답시간을 충족시키기 어렵다[6,14]. 따라서, 응답 시간을 최소화하기 위하여 사용자가 미리 요구하는 질의를 계산하여 결과를 저장하는 실체 뷰(materialized view)의 도입을 고려할 수 있다.

뷰를 실체화 할 때 고려해야 할 중요한 문제 중 하나가 실체 뷰 선택 문제이다. 실체 뷰는 실제의 저장공간에 결과를 저장해 두기 때문에 저장 할 수 있는 뷰의 개수는 제한되어 있다. 따라서, 사용자의 요구 즉 신속한 응답 시간을 만족시킬 수 있도록 제한된 저장공간에 알맞은 뷰를 선택하여 저장하는 것이 중요한 문제이다[2,4,12].

실체 뷰 선택에 관한 여러 연구가 진행되어 왔으며, 기존의 연구들은 데이타 웨어하우스의 사실 테이블만을 포함한 데이타 큐브에서 실체 뷰를 선택하는 연구이었다[3,6,14]. 일반적으로 데이타 큐브는 소규모 용량의 데이타 마트(data mart)에 사용되고 적은 수의 차원으로 구성되어진다. 이러한 환경에서 실체 뷰는 사실 테이블만으로 이루어지는 단순 격자를 형성하며 실체 뷰 선택에 중요한 기준이 되는 질의 처리 비용은 실체 뷰 크기와 같다고 가정하고 있다. 그러나 현재 실세계에서 운영되는 데이타 웨어하우스들은 전사적인 대규모의 데이타 웨어하우스에 이용되며, 사용자의 다차원적 관점을 지원하기 위해 수 십 개의 차원과 차원계층을 포함하고 있다. 따라서 이러한 데이타 웨어하우스는 하나의 사실 테이블과 여러 개의 차원 테이블을 포함하고 있다. 데이타 웨어하우스에서 생성되는 실체 뷰는 사실 테이블과 차원 테이블의 조

† 학생회원 : 현대정보기술 투신IT실 영업정보팀 연구원
wsyoon@hit.co.kr
†† 종신회원 : 중앙대학교 정보시스템학과 교수
dcshin@cau.ac.kr
논문접수 : 2000년 1월 21일
심사완료 : 2001년 1월 29일

인으로 형성된 복합 격자를 구성하는 뷰이고, 실체 뷰 선택시 가장 중요한 기준이 되는 질의 처리 비용은 조인 비용이다. 따라서 데이타 웨어하우스에서 실체 뷰 선택에 조인 비용을 고려하는 것이 필요하다.

본 논문은 데이타 웨어하우스에서 제한된 저장 공간 하에서 질의 처리 시간을 만족하는 실체 뷰를 선택하는데 있어서 중요한 기준이 되는 조인 비용을 고려하며, 조인 비용을 기반으로 한 실체 뷰 선택 기법을 제안하기 위해 다음의 내용을 전개한다.

- 실체 뷰에 내재된 차원 속성 레벨을 이용한 조인 테이블 추적의 정형화

먼저 데이타 웨어하우스에 정의된 복합 격자를 구성하는 뷰들의 조인 비용을 도출하기 위해 뷰들이 어떤 조인 테이블로 이루어져 있는지를 추적하는 작업이 필요하다. 데이타 웨어하우스의 뷰들은 차원 속성(dimension attribute) 레벨을 명시적으로 내재하고 있으며, 이러한 차원 속성 레벨을 이용하여 정형화된 방법으로 조인 테이블을 추적한다.

- 추적된 조인 테이블에 대한 조인 비용 정형화

추적된 조인 테이블을 기반으로 조인 비용을 정형화한다.

- 조인 비용을 기반으로 한 실체 뷰 선택

실체화 이익은 복합 격자를 구성하는 뷰가 실체화되었을 때 얼마만큼의 이익을 가져다주는지를 의미한다. 가장 큰 이익을 가져다주는 뷰가 실체화되며, 이러한 실체화 이익에 가장 큰 영향을 주는 요소가 조인 비용이다. 따라서 실체 뷰 선택시 실체화 이익에 중요한 요소인 조인 비용을 기반으로 한 알고리즘을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문의 기반 환경인 데이타 웨어하우스 환경에 대하여 기술한다. 3장에서는 조인 비용 모델을 기반으로 조인 비용을 정형화하고 4장에서는 정형화된 조인 비용을 반영해서 복합 격자에서 실체 뷰 선택 알고리즘을 제안한다. 5장에서는 결론을 맺는다.

2. 데이타 웨어하우스 환경

2.1 스타 스키마와 스노우플레이크 스키마

데이타 웨어하우스의 설계에는 차원 모델링이 이용되며 가장 대표적인 것으로 스타 스키마와 스노우플레이크 스키마를 들 수 있다[7,16]. 그림 1은 데이타 웨어하우스를 스타 스키마와 스노우플레이크 스키마로 나타낸 예이다. 스타 스키마로 표현된 데이타 웨어하우스는 데이타를 보는 방법을 나타내는 차원 테이블과 실제 분석하고자 하는 수치 데이타를 포함하는 사실 테이블로 구성되어 있으며, 스노우플레이크 스키마로 표현된 데이타 웨어하우스는 차원 테이블을 정규화한다.

스타 스키마의 차원 테이블 D_i에는 여러 개의 차원 속성 a_{ij}가 존재할 수 있다. 그림 1-a)에서 차원 테이블 D_1은 a_{11}, a_{12}, a_{13}이라는 차원 속성으로 구성되어 있다. 각 차원 속성간에는 계층, 즉 차원 레벨을 형성하고 있으며 1:n의 관계를 가지고 있다. 예를 들면, 그림 1-a)에서 a_{11}은 차원 테이블 D_1의 가장 상세한 차원 정보를 보여주는 즉, 가장 하위 레벨의 차원 속성이며 따라서 a_{11}의 서로 다른 인스턴스(instance)들은 좀더 요약적인 즉, 상위 레벨의 차원 정보를 제공하는 a_{12}의 동일한 하나의 인스턴스와 대응될 수 있다. 마찬가지로, a_{12}의 서로 다른 인스턴스들은 차원 테이블 D_1에서 가장 일반적인 요약정보를 제공하는 a_{13}의 동일한 하나의 인스턴스와 대응된다. 이러한 계층 정보는 사용자가 데이타를 분석할 때 아주 효율적으로 사용될 수 있다.

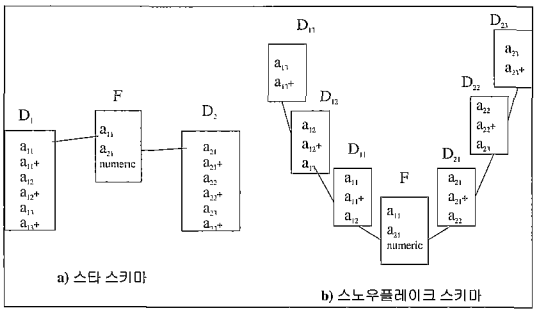

그림 1 데이타 웨어하우스 스키마

스타 스키마가 각 차원 i마다 하나씩 차원 테이블 D_i가 있는 것에 반하여 스노우플레이크 스키마는 그림 1-b)에서 보듯이 차원 i의 차원 속성 $j(a_{ij})$마다 하나의 차원 테이블 D_{ij}을 가지고 있고 이 차원 테이블에는 차원 속성에 대한 정보와 상위 레벨의 차원 테이블 D_{ij+1}로 조인하기 위한 키가 들어 있다. 스노우플레이크 스키마의 차원 테이블은 정규화가 잘 되어 있어서 중복이 안된다. 따라서 차원 테이블의 크기도 작다. 즉 스타 스키마는 스노우플레이크 스키마를 비정규화(denormalization)한 것이라고 생각할 수 있으며 역으로 스노우플레이크 스키마는 스타 스키마의 정규화된 형태라고 생각할 수 있다[16].

본 논문에서 사용하는 데이타 웨어하우스는 중복을 제거한 스노우플레이크 스키마로 구성되어 있다고 가정

한다. 일반적으로 사실 테이블 F와 임의의 차원 테이블 D_{ij}는 다음과 같은 스키마로 구성된다.

$$F\ (\underline{a_{11}*,\ a_{21}*,\ a_{31}*,\ \ldots.\ ,a_{n1}*}\ numeric\ )$$

$$D_{ij}\ (\underline{\ a_{ij},\ a_{ij}^{+},\ a_{ij+1}*}\ )$$

위의 스키마에서 사실 테이블 F는 수치값인 *numeric*을 포함하고 있으며, 각 차원 i의 최하위 차원 테이블 D_{i1}과 관계를 맺어주기 위해 차원 i의 차원 속성 a_{i1}을 외래키로 구성하고 있다. n개의 차원을 갖는 경우 각 차원 속성 a_{11}, a_{21}, ... ,a_{n1}은 사실 테이블 F의 복합 기본키를 구성한다. 차원 테이블 D_{ij}는 i번째 차원의 j번째 차원 속성 a_{ij}를 기본키로 하고 있으며, 상위 차원 테이블 D_{ij+1}과 관계를 형성하기 위해 외래키 a_{ij+1}을 포함한다.

따라서, 최상위 차원 테이블은 외래키를 포함하지 않는다. 속성 a_{ij}^{+}는 차원 속성 a_{ij}에 대한 부가적인 정보를 나타내주는 속성이다. 예를 들어, 차원 속성 a_{ij}가 type_id 라면 a_{ij}^{+}는 type_ name이 될 수 있다. 밑줄로 표시된 속성은 기본키를 나타내며 * 표시된 속성은 외래키를 나타낸다.

2.2 복합 격자(composite lattice)

데이타 웨어하우스에서 정의된 뷰는 또 다른 뷰로부터 계산될 수 있다. 이렇듯 데이타 웨어하우스를 구성하는 뷰들 사이에는 의존 관계가 성립한다. 의존 관계란 두 개의 뷰 V_1과 V_2가 있다고 할 때 V_2의 결과를 가지고 V_1을 계산할 수 있다면 V_1은 V_2에 의존한다고 하고 이것을 $V_1 \leq V_2$ 로 표현한다. 격자 구조는 이러한 관계를 가장 잘 설명해주는 구조이다.

그림 1과 같은 스키마를 가진 데이타 웨어하우스에서 단순 격자는 사실 테이블에 포함되는 차원의 수만을 고려해서 생성된다. 즉, 그림 1-a)에서 고려되는 뷰는 사실 테이블에 포함되는 차원 속성인 a_{11}과 a_{21}에 따라 그룹핑 되는 뷰(V_{11})와 a_{11}, a_{21} 각각에 대해서만 그룹핑 되는 뷰(각각 V_{01}, V_{10}) 그리고 그룹핑을 고려하지 않은 뷰(V_{00})의 단순한 구조를 이룬다 (그림 2-a) 참조). 한편, 복합 격자는 차원 테이블에 존재하는 계층 구조인 차원 속성을 고려해서 단순 격자의 구조를 확장시킨 구조이다 (그림 2-b) 참조). 단순격자에서 복합 격자로 실체 뷰 선택을 확장한 경우 나타나는 특성은 다음과 같다.

- 격자의 수가 기하급수적으로 증가한다.
- 복합 격자를 형성하는 뷰는 사실 테이블과 차원 테이블과의 조인이 필요한 조인 뷰로 형성된다.

본 논문에서 사용되는 데이타 웨어하우스의 차원

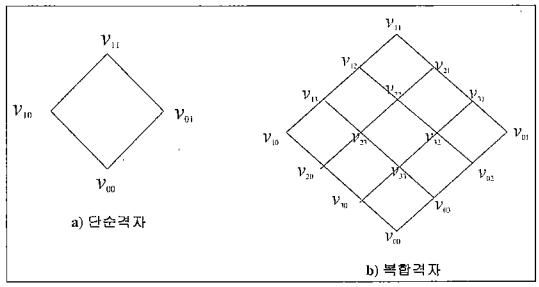

그림 2 데이타 웨어하우스의 격자 구조

속성 레벨은 사실 테이블과 직접적으로 관계를 형성하는 차원 테이블의 차원 속성 레벨을 1로 시작하여 상위 수준의 차원 테이블마다 1씩 증가한다. 예를 들어 그림 1 b)에서 차원 테이블 D_{11}은 레벨이 1, D_{12}는 레벨이 2, D_{13}은 레벨이 3이 되며 두 차원은 모두 차원 속성 레벨이 3까지 있다.

표기 1 : n개의 차원을 갖는 데이타 웨어하우스에서 임의의 실체 뷰를 $V_{d1d2...dn}$ 로 표기한다. 여기서 d_i는 i번째 차원의 차원 속성 레벨을 의미한다.

예를 들어, 그림 2-b)에서 실체 뷰 V_{13}은 첫 번째 차원의 차원 속성 레벨이 1인 차원 테이블 D_{11}의 차원 속성 a_{11}과 두 번째 차원의 차원 속성 레벨이 3인 차원 테이블 D_{23}의 차원 속성 a_{23}으로 그룹핑된 뷰를 의미하며, V_{20}은 첫 번째 차원의 차원 속성 레벨이 2인 차원 테이블 D_{12}의 차원 속성 a_{12}으로만 그룹핑된 뷰를 의미한다. 여기서 차원 속성 레벨이 0 이라는 것은 해당 차원의 차원 속성으로는 실체 뷰를 만들지 않았음을 의미한다.

표기 2 : n개의 차원을 가진 데이타 웨어하우스에서 사실 테이블을 포함하여 각 해당 차원의 차원 속성 레벨까지의 차원 테이블의 집합을 $T_{d1d2..dn}$ 로 표기한다. 여기서 d_i는 i번째 차원의 차원 속성 레벨을 의미한다.

예를 들어, 데이타 웨어하우스에 2개의 차원이 존재하는 그림 1-b)의 경우에 T_{22}는 해당 차원 속성 레벨의 차원 테이블, 즉 첫 번째 차원의 차원 속성 레벨 2까지의 테이블인 D_{12}와 D_{11}, 두 번째 차원의 차원 속성 레벨 2까지의 차원 테이블인 D_{22}, D_{21}을 포함한다. 또한, 사실 테이블인 F 는 데이타 웨어하우스의 전체 실체 뷰에 모두 포함되기 때문에 $T_{22} = \{F, D_{22}, D_{12}, D_{11}, D_{21}\}$ 이 된다. 한편 T_{20}은 첫 번째 차원의 차원 속성 레벨 2까지의 차원 테이블인 D_{12}, D_{11}와 사실 태이블 F가 포함되므로 $T_{20} = \{D_{12}, D_{11}, F\}$ 가 된다.

2.3 데이타 웨어하우스에서의 조인

데이타 웨어하우스에서 사용하는 질의들은 여러 개의

테이블의 조인을 요구하는 경우가 대부분이며, 관계형 데이타베이스는 여러 개의 테이블을 조인해야 하는 질의가 발생하면 이 질의를 순차적인 몇 개의 pair-wise 조인으로 바꾸어 처리한다[9,15]. 데이타 웨어하우스에서의 조인은 스키마가 어떻게 구성되어 있느냐에 따라 크게 2가지 경우로 분류할 수 있다.

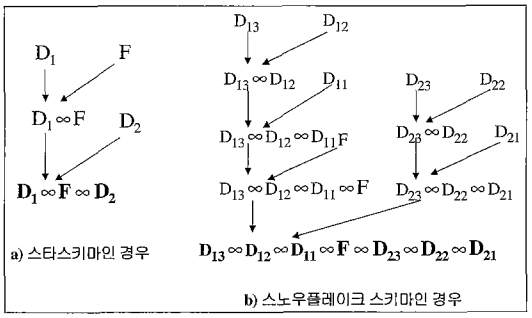

그림 3 데이타 웨어하우스에서의 조인

• 스타 스키마로 구성된 데이타 웨어하우스에서의 조인 : 그림 3-a)와 같이 사실 테이블 F와 차원 테이블 D_1, D_2에 대한 조인이 필요한 질의의 경우 사실 테이블 F와 차원 테이블 D_1에 대한 조인 결과와 차원 테이블 D_2에 대한 조인을 수행하여 결과를 도출할 수 있다.

• 스노우플레이크 스키마로 구성된 데이타 웨어하우스에서의 조인 : 그림 3-b)와 같이 사실 테이블 F와 차원 테이블 D_{11}, D_{12}, D_{13}, D_{21}, D_{22}, D_{23}과 같이 총 7개에 테이블에 대한 조인이 필요한 질의의 경우 먼저 각 차원별 차원 테이블에 대한 조인이 수행된다. 즉, 차원 테이블 D_{11}, D_{12}, D_{13}에 대한 조인과 D_{21}, D_{22}, D_{23}에 대한 조인 후에 조인된 각 차원 테이블 $D_{11}\bowtie D_{12}\bowtie D_{13}$과 $D_{21}\bowtie D_{22}\bowtie D_{23}$은 각각 사실 테이블 F와 조인되어 결과를 도출할 수 있다.

3. 실체 뷰의 조인 비용

3.1 조인 테이블 추적의 정형화

조인 비용을 기반으로 실체 뷰를 선택하기 위해서는 먼저 각 실체 뷰를 생성하는데 소요되는 조인 비용을 구하여야 하므로 조인 연산과 관련된 테이블을 추적하는 것이 필요하다. 그림 2-b)의 복합 격자에서 형성된 V_{21} 뷰는 사실 테이블 F 와 차원 테이블 D_{11}의 조인으로 형성되며 반면에 격자의 최상위에 존재하는 V_{11}뷰는 격자 위에 존재하는 여러 뷰들 중 가장 상세한 정보를 포함하며 조인 연산 없이 사실 테이블 F 로부터 바로 유도되어진다. 복합 격자에서 존재하는 모든 뷰는 반드시 사실 테이블을 포함하고 있다. 왜냐하면 복합 격자에서 존재하는 뷰는 사실 테이블 요약하는 요약 정보를 가지고 있기 때문에 반드시 수치 데이터를 포함하는 사실 테이블을 포함하고 있어야 한다.

실체 뷰 $V_{d1d2..dn}$ 에 나타나는 차원 속성 레벨과 $T_{d1d2...dn}$ 나타나는 차원 속성 레벨사이에는 일정한 패턴을 가지고 있다. n 개의 차원을 가진 데이타 웨어하우스에서 실체 뷰 $V_{d1d2..dn}$ 와 $T_{d1d2...dn}$ 의 집합을 각각 V와 T라 할 때 모든 $V_{d1d2...dn} \in V$ 에 대해 $V_{d1d2...dn}$를 실체화하기 위해 조인에 관련되는 테이블의 집합은 $T_{d1-1d2-1..dn-1}$ 이 된다.

예를 들어, 그림 2-b)의 복합 격자에서 뷰 V_{22}를 보면 이 뷰는 첫 번째 차원의 차원 속성 레벨 2의 차원 속성인 a_{12}와 두 번째 차원의 차원 속성 레벨 2의 차원 속성인 a_{22}로 그룹핑된 뷰이다. 따라서 뷰 V_{22}는 차원 테이블 D_{11}와 D_{21}, 그리고 사실 테이블인 F에 의해 다음과 같이 정의된다.

create view V_{22} **as**
select a_{12}, a_{22}, **sum**(numeric)
from F, D_{11}, D_{21}
where $F.a_{11} = D_{11}.a_{11}$ **and**
$F.a_{21} = D_{21}.a_{21}$
group by a_{12}, a_{22}

따라서 뷰 V_{22}를 얻기 위해서 조인하여야 할 테이블 집합은 $T_{11} = \{F, D_{11}, D_{21}\}$이 된다.

3.2 조인 연산

데이타 웨어하우스의 복합 격자에서 정의된 실체 뷰의 조인 비용을 정형화하기에 앞서 조인이 어떻게 수행되는가를 살펴보는 것이 필요하다. 조인이 어떻게 수행되는가에 따라 조인 비용은 달라지기 때문이다. 본 논문에서는 조인 기법 중 가장 간단한 nested-loop 조인 방법을 이용한다.

Nested-loop 조인 방법은 가장 간단한 조인 방법이다. 조인될 테이블 중 하나는 안쪽 테이블에 위치하고, 나머지 하나는 바깥쪽 테이블에 위치한다. 바깥쪽 테이블의 각 튜플 대하여 안쪽 테이블의 모든 튜플을 읽고 조인 조건을 비교한다. 조인 조건이 만족될 때마다 두 튜플은 연결된다. 효율성을 위하여 많은 튜플의 수 즉, 크기가 큰 테이블을 안쪽 테이블로 선택할 수 있다[8]. 본 논문에서 nested-loop 조인 방법은 조인을 튜플 단위로 처리하기 때문에 조인 비용은 조인 테이블의 튜플의 수와 레코드의 길이의 곱, 즉 조인 테이블의 크기와 같다고 가정한다. 테이블 R의 튜플의 수를 n_r, 테이블

R의 레코드 길이 (단위 byte)를 s_r
이라 하면 테이블 R_1과 R_2의 조인 비용은 다음과 같다.

$$JC(R_1 \bowtie R_2) = n_{r1} \times s_{r1} + n_{r1} \times n_{r2} \times s_{r2}$$

한편, 조인 비용을 정형화 할 때 가장 중요하게 고려해야 할 문제 중 하나가 조인 순서에 대한 문제이다. 일반적으로 조인 순서는 조인 비용에 많은 영향을 주며, 따라서 적은 비용을 가진 조인 순서를 결정하는 문제는 질의 최적화 분야에서 이슈로 대두되고 있다.

여러 개의 차원 테이블과 하나의 사실 테이블로 구성된 데이타 웨어하우스는 크게 두 부분의 조인으로 구분되어 질 수 있다. 2.3절에서 언급한 바와 같이 첫 번째는 각 차원 테이블끼리의 조인이며, 두 번째는 조인된 차원 테이블과 사실 테이블의 조인이다. 스노우플레이크 스키마로 표현된 데이타 웨어하우스에서의 조인은 항상 각 차원 테이블의 조인 후에 조인된 차원 테이블과 사실 테이블의 조인이 수행된다. 두 부분의 조인 모두 기본키와 외래키를 조인 속성으로 하고 있다.

데이타 웨어하우스에서 임의의 실체 뷰를 얻기 위해서는 각 차원에서 해당 차원의 여러 차원 테이블끼리 조인된 각 차원의 결과 테이블들과 사실 테이블의 조인이 필요하다. 이 경우 항상 사실 테이블과의 조인이 먼저 발생하기 때문에 이 두 테이블의 조인으로 발생되는 임시 테이블의 튜플의 수는 사실 테이블의 튜플의 수와 같다. 예를 들어, 그림 2 b)에서 실체 뷰 V_{33}을 얻기 위한 조인 테이블 집합은 T_{22} = {D_{12}, D_{11}, F, D_{22}, D_{21}}이 되므로 첫 번째 차원의 D_{12}, D_{11}의 조인 결과 테이블과 두 번째 차원의 D_{22}, D_{21}의 조인 결과 테이블, 그리고 사실 테이블 F의 조인이 필요하다. 이 경우 F와의 조인 결과 테이블의 크기는 F의 크기와 같게 된다. 따라서 이 두 번째 부분의 조인에 대한 조인 순서는 고려하지 않는다. 본 논문에서는 차원 테이블간의 조인 순서만을 고려한다.

정리 1 : 차원 테이블간의 조인에 대한 결과 테이블의 튜플의 수는 항상 하위 테이블의 튜플의 수와 같다. 따라서 차원 테이블간의 조인은 상위 레벨의 차원 테이블부터 차례로 조인되는 것이 최적의 조인 순서이다.

<증명> : 차원 테이블 D_{ij}와 D_{ij+1}의 조인은 D_{ij+1}의 기본키 a_{ij+1}와 D_{ij}의 외래키 a_{ij+1}사이의 조인으로 이루어지며, 인접 차원 테이블 사이에는 반드시 관계를 갖게 되어 외래키 값은 참조 테이블의 기본키에 반드시 나타나 널 값이 없으므로 조인된 결과 테이블의 크기는 하위 테이블의 크기와 같다. 일반적으로 하나의 차원은 차원 속성 레벨에 따라 여러 개의 차원 테이블을 구성하고 있다, 한편 데이타 웨어하우스에서는 차원 속성 레벨이 깊어질수록 즉 레벨이 커질수록 해당 차원 테이블은 작아지는 것이 일반적이며 최악의 경우 같아질 수 있다. 즉 차원 테이블 D_{ij}는 상위 차원 테이블인 D_{ij-1}보다 항상 크거나 같다는 것을 의미한다. 그러므로 가장 상위 차원 레벨의 차원 테이블부터 하위로 차례로 조인 순서를 정하는 것이 가장 높은 선택도, 즉 가장 작은 임시 테이블을 가지기 때문에 가장 적은 비용을 가진 조인 순서가 된다.

3.3 조인 비용

데이타 웨어하우스에서 실체 뷰 $V_{d1d2...dn}$은 $T_{d1-1d2-1...dn-1}$내의 테이블들의 조인으로 정의되며 $T_{d1d2...dn}$은 다음과 같이 구성되어짐을 앞 절에서 논의하였다.

$T_{d1d2...di...dn}$
$=\{F, D_{11}, D_{12}, ..., D_{1d1}, ..., D_{i1}, D_{i2}, ..., D_{idj}, ..., D_{n1}, D_{n2}, ..., D_{ndn}\}$

데이타 웨어하우스에서 조인은 차원 테이블간의 조인과 조인된 차원 테이블과 사실 테이블의 조인으로 이루어지므로, 먼저 I 차원의 차원 속성 레벨 j에서 차원 테이블 D_{ij}와 D_{ij-1}의 조인 비용을 JC_{ij}라하고 조인 결과 테이블을 R_{ij}라 하면 $JC_{ij} = JC(D_{ij} \bowtie D_{ij-1})$, $JC_{ij-1} = JC(R_{ij} \bowtie D_{ij-2})$가 된다. 따라서, j 레벨까지의 차원 테이블 사이의 조인 총 비용을 TJC_{ij}라고 하면 식(1)과 같다.

$$TJC_{ij} = JC_{ij} + JC_{ij-1} + \cdots + JC_{i2} = \sum_{k=2}^{j} JC_{ik} \quad (1)$$

따라서 i차원의 차원 테이블간의 조인 결과 테이블을 R_{i2}라 하면 R_{i2}를 얻기 위한 조인 비용은 TJC_{i1}가 된다. 한편, n차원의 데이타베이스에서 최종 결과는 $F \bowtie R_{12} \bowtie R_{22} \bowtie \cdots \bowtie R_{n2}$ 로 얻어진다. 한편, 이 부분에 대한 조인 순서는 3.2절에서 언급한 바와 같이 고려하지 않으므로 $F \bowtie R_{12}$의 결과 테이블을 T_1이라하면 최종 결과 테이블 LT는 식(2)와 같이 표현할 수 있다.

$$LT = T_1 \bowtie R_{22} \bowtie R_{32} \bowtie \cdots \bowtie R_{n2} \quad (2)$$

첫 번째 조인 결과 테이블을 JR_1이라 하면 T_1을 산출하기 위한 비용 $JC_{JR1} = JC(F \bowtie R_{12})$가 되며 두 번째 조인 결과 테이블 JR_2를 얻기 위한 조인 비용 $JC_{JR2} = JC(JR_1 \bowtie R_{22})$가 된다. 따라서 LT를 얻기 위한 총 비용은 식(3)과 같다.

$$JC_{LT} = JC_{JR_1} + JC_{JR_2} + \cdots + JC_{JR_{n-1}} = \sum_{k=1}^{n-1} JC_{JR_k} \quad (3)$$

그러므로 n차원으로부터 정의되고 아울러 차원 i에서는 j레벨까지의 차원 테이블로부터 정의되는 실체 뷰 k를 얻기 위한 총 비용을 $JC(k, MD)$라고 하면 식(4)와 같다.

$$JC(k, MD) = \sum_{i=1}^{m} TJC_{ij} + JC_{LT} \quad (4)$$

4. 실체 뷰 선택

데이타 웨어하우스에서 실체 뷰의 크기는 실체 뷰 선택에 중요한 기준이 되기 때문에 실체 뷰의 크기를 추정하는 것은 실체 뷰 선택에 있어서 아주 중요한 문제이다. 따라서 실체 뷰 크기를 추정하는 문제에 대한 연구가 활발히 진행되고 있다[10]. 생성되는 실체 뷰의 크기는 실체 뷰를 정의하는 구문의 group by절에 언급된 속성들의 값들이 갖는 서로 다른 개수에 비례하게 된다. group-by절에 하나의 속성만을 가진 실체 뷰의 크기는 쉽게 추정할 수 있지만 group-by절에 여러 개의 속성을 가진 실체 뷰의 크기를 추정하는 문제는 그렇게 간단한 문제가 아니다. 여러 연구에서 실체 뷰의 크기를 추정하기 위해 샘플링 방법이나 데이타 분포에 대한 통계적 분석을 이용하고 있다[10].

실체 뷰의 크기를 추정하는 방법에는 여러 가지가 있으며, 본 논문에서는 분석적 알고리즘을 이용한다. 이 방법은 데이타베이스에 저장되어 있는 데이타가 고르게 분포되어 있다고 가정하고 단지 데이타 사전에 있는 정보만으로 테이블에 있는 서로 다른 요소의 개수를 계산하는 방법이다. 실체 뷰를 정의하는 구문의 group by절에 나타나는 k개의 속성이 각각 $A_1,A_2,.....A_k$의 서로 다른 요소를 가질수 있다면 실체 뷰의 크기는 $A1 \times A2 \times ..... \times A_k$ 가 된다.

4.1 실체화 이익에 영향을 주는 요소

실체화 이익이란 특정 뷰에 대한 질의가 들어왔을 경우 실체 뷰에서 그 질의를 처리함으로써 얻은 질의 처리비용 감소량 즉, 상대적인 이익을 말한다. 실체화 이익은 실체 뷰 선택의 절대적인 기준이다. 이러한 실체화 이익에 영향을 주는 요소를 살펴보면 다음과 같다.

● 조인 비용

실체화 이익에 많은 부분을 차지하는 요소이다. 스노우플레이크 스키마로 표현된 데이타 웨어하우스에서 정의된 뷰들은 조인을 포함하고 있으며, 실체화 할 뷰들이 어떤 조인 테이블로 이루어져 있는지는 중요하다. 일반적으로 실체화 할 뷰들은 같은 차원의 차원 테이블로 유도되는 뷰와 다른 차원의 차원 테이블로 유도되는 뷰로 구성된다.

- 같은 차원의 차원 테이블로 유도되는 뷰 : 그림 2-b)에서 뷰 V_{13}을 보면 같은 차원의 차원 테이블 D_{21}, D_{22}와 사실 테이블 F로 이루어져 있으며, 이 뷰는 작은 차원 테이블이 먼저 조인되고 상당히 큰 사실 테이블이 맨 마지막에 조인되기 때문에 상대적으로 적은 조인 비용이 든다.

- 다른 차원의 차원 테이블로 유도되는 뷰 : 그림 2-b)에서 뷰 V_{22}와 같이 다른 차원의 차원 테이블 D_{11}, D_{21}과 사실 테이블 F로 이루어져 있는 뷰는 상당히 큰 사실 테이블과 먼저 조인되기 때문에 상대적으로 높은 조인 비용이 든다

이와 같이 다른 차원의 차원 테이블로 유도되는 뷰에 대한 조인 비용이 크기 때문에 이러한 뷰를 먼저 실체화하는 것이 실체화 이익을 크게 한다.

● 실체 뷰의 크기

데이타 웨어하우스의 복합 격자에서 정의된 뷰의 차원 속성 레벨즉, 요약 수준이 높아지면 실체화하려는 뷰의 크기는 작아진다. 예를 들어, 그림 2-b)에서 뷰 V_{11} -> V_{12} -> V_{13} -> V_{10}과 같이 요약 수준이 높아질수록 실체 뷰의 크기는 작아지며, V_{22}, V_{13}과 같이 같은 요약 수준을 갖더라도 실체화하려고 하는 뷰의 크기는 해당 차원의 데이타 분포에 따라 다르게 나타날 수 있다.

뷰 V_{12}와 V_{13}이 정의되는 비용이 똑같이 100이고 V_{12}의 실체 뷰 크기가 30, 더 큰 요약 수준을 갖는 V_{13}의 실체 뷰 크기가 20이라고 가정하면 V_{12}의 실체화 이익은 100 - 30 = 70 이 되고 V_{13}의 실체화 이익은 100 - 20 = 80이 된다. 이 경우 작은 V_{12}를 실체화하는 것이 적은 질의 처리비용 즉 상대적으로 큰 실체화 이익을 갖는다. 따라서 실체화하려고 하는 뷰의 크기가 작은 것 즉, 요약 수준이 높은 뷰를 실체화하는 것이 더 많은 실체화 이익을 가져다준다.

● 공헌도

공헌도는 뷰가 실체화될 경우에 그 뷰가 커버할 수 있는 자식 뷰의 수를 말한다. 그림 2- b)에서 V_{12}는 자신을 제외한 11개의 뷰(V_{22}, V_{13}, V_{10}, V_{23}, V_{32}, V_{20}, V_{33}, V_{02}, V_{30}, V_{03}, V_{00})를 자식 뷰로 하고 있으며, V_{22}는 자신을 제외한 8개의 뷰 (V_{23}, V_{32}, V_{20}, V_{33}, V_{02}, V_{30}, V_{03}, V_{00})를 자식 뷰로 하고 있다. 요약 수준이 높은 실체 뷰는 공헌도가 낮으며, 낮은 요약 수준을 가진 실체 뷰는 공헌도가 높다. 공헌도는 자식 뷰의 수만큼 실체화 이익을 증가시키기 때문에 공헌도가 높은 실체 뷰를 선택하는 것이 실체화 이익을 크게 한다.

4.2 실체화 이익 도출

4.2.1 실체화 이익 b_i 계산

실체화 이익 b_i는 뷰 i가 실체화되었다면 데이타 웨어하우스에 얼마만큼의 이익을 줄 수 있는지를 계산하는 것이다. 이러한 실체화 이익 b_i는 실체 뷰 선택의 기준이 되기 때문에 실체화 이익 b_i가 가장 큰 뷰를 실체화

시킨다. 먼저 복합 격자를 구성하는 모든 뷰들에 대하여 실체화 이익 b_i를 초기화한다.

실체화 이익 b_i는 복합 격자에서 뷰 i 자신이 실체화됨으로써 얻을 수 있는 실체 뷰 i의 자체 이익인 sb_i와 뷰 i가 실체화됨으로써 복합 격자에서 자식 뷰들이 얻는 이익인 공헌도 이익인 cb_i의 두 부분으로 이루어지며, 식(5)와 같다

$$b_i = sb_i + cb_i \quad (5)$$

먼저 실체 뷰 i의 자체 이익 sb_i는 결과를 도출하기 위해 뷰 i가 데이타 웨어하우스의 사실 테이블과 차원 테이블의 조인을 통하여 요구되는 조인 비용에서 자기 자신이 실체화되었다고 가정했을 경우 자기 자신으로부터 직접 요구되는 비용을 뺀 값으로 sb_i를 구할 수 있다.

$$sb_i = JC(i, MD) - C(i, i) \quad (6)$$

식(6)에서 $C(i,i)$는 질의 결과를 실체 뷰 i 자기 자신으로부터 계산하는 비용이며 따라서, $C(i,i)$는 실체 뷰 i의 크기와 같다. 한편, 식(6)으로부터 조인 비용 JC(i, MD)이 큰 뷰를 실체화하는 것이 실체화 이익의 자체 이익 sb_i를 증가시킴을 알 수 있다.

실체 뷰 i의 공헌도 이익 cb_i는 뷰 i의 자식 뷰 j (뷰 i가 실체화되었을 경우 뷰 i로부터 계산될 수 있는 뷰)가 사실 테이블과 차원 테이블의 조인을 통하여 계산되는 조인 비용 $JC(j,MD)$에서 뷰 i가 실체화되었을 경우 자식 뷰 j가 실체 뷰 i를 이용하여 계산되는 비용 $C(j, i)$를 뺀 값으로 cb_i를 구할 수 있다.

조인 비용인 $JC(j,MD)$가 큰 자식 뷰 j를 많이 가진 뷰를 실체화하는 것이 공헌도 이익 cb_i를 증가시킨다. 공헌도 이익 cb_i는 식(7)과 같다.

$$cb_i = \sum_{\{j | j<i \text{ on complex lattice } CL\}} (JC(j, MD) - C(j, i)) \quad (7)$$

따라서 실체 뷰 i의 실체화 이익 b_i는 다음 식(8)로 표현된다.

$$b_i = JC(i, MD) - C(i, i) + \sum_{\{j | j<i \text{ on complex lattice } CL\}} (JC(j, MD) - C(j, i)) \quad (8)$$

4.2.2 실체화 이익 b_i 재계산

실체화 뷰 선택 과정에서 가장 큰 실체화 이익을 가진 뷰 i가 제일 먼저 실체화 되어야 할 첫 실체 뷰로 선택되었다면 그 다음 실체 뷰 선택을 위해 이미 선택된 실체 뷰 i를 제외한 모든 뷰들에 대하여 실체화 이익을 재계산 해주어야 한다. 왜냐하면, 복합 격자를 구성하는 뷰들은 의존 관계를 가지고 있기 때문에 선택된 실체 뷰 i에 의해서 다음 선택 때 나머지 뷰들의 실체화 이익은 영향을 받기 때문이다. 실체 뷰 i에 의해서 영향을 받는 뷰들은 실체 뷰 i의 조상들과 자식 그리고 의존 관계가 없는 뷰이다. 따라서, 이들에 대한 실체화 이익을 재계산하여야 한다. 예를 들어, 그림 4에서 실체 뷰 i가 V_{22}라고 한다면 조상 뷰는 V_{12}, V_{21}가 되며, 자식 뷰는 V_{23}, V_{32}, V_{20}, V_{33}, V_{02}, V_{30}, V_{03}, V_{00}이 되고, 의존 관계가 없는 뷰는 V_{13}, V_{10}, V_{31}, V_{01}이 된다.

• 실체 뷰 i의 조상 뷰 m에 대한 실체화 이익 b_m 계산

복합 격자에서 뷰 i의 조상들 중에서 뷰 i와의 사이에 실체화 되어 있는 뷰가 없는 조상 m은 뷰 i와 뷰 i의 자식을 커버하지 않아도 된다. 그림 4의 a)를 보면 조상 뷰 m의 자식 뷰들은 실체 뷰 i의 자식 뷰들을 포함하고 있다. 하지만 실체 뷰 i의 자식 뷰들은 조상 뷰 m이 실체화되어도 실체 뷰 m으로부터 계산되어지지 않고 적은 비용을 지불하는 실체 뷰 i로부터 계산되기 때문에 실체 뷰 i의 자식 뷰들을 제외한 뷰 v들만이 조상 뷰 m의 공헌도 이익으로 평가된다. 여기서 조상 뷰 m을 V_{12}라고 할 때 뷰 v는 V_{13}, V_{10}이 된다. 따라서, 조상 뷰 m의 실체화 이익 b_m은 재계산 되며, 실체화 이익은 식(9)와 같다.

$$b_m = JC(m, MD) - C(m, m) + \sum_{\{v | v<m,\ v \not< i \text{ complex lattice } CL\}} (JC(v, MD) - C(v, m)) \quad (9)$$

• 실체 뷰 i의 자식 뷰 k에 대한 실체화 이익 b_k 계산

그림 4-b)를 보면 실체 뷰 i의 자식들 중에서 뷰 i와의 사이에 실체화되어 있는 뷰가 없는 자식 k에서 실체화 이익의 자체 이익을 계산할 때, 비교 대상이 가장 가까운 실체화된 조상 뷰가 최상위(top) 뷰에서 실체 뷰 i로 변경된다. 또한 자식 뷰 k의 자식 뷰 v들도 공헌도 이익을 계산할 때 비교 대상이 최상위 뷰에서 실체 뷰 i로 변경된다. 여기서 자식 뷰 k가 V_{23}이라고 하면 뷰 v는 V_{20}, V_{30}, V_{33}, V_{03}, V_{00}이 된다. 따라서 자식 뷰 k의 실체화 이익 b_k는 재계산되며, 실체화 이익은 식 (9)와 같다.

$$b_k = C(k, i) - C(k, k) + \sum_{\{v | v<k \text{ on complex lattice } CL\}} (C(v, i) - C(v, k)) \quad (10)$$

• 실체 뷰 i와 의존 관계를 갖지 않는 뷰 l에 대한 실체화 이익 b_l 계산

실체 뷰 i와 어떠한 의존 관계도 가지고 있지 않은 뷰 (자식 뷰와 조상 뷰를 제외한 뷰) l의 실체화 이익도 재계산 해주어야 한다. 의존 관계가 없는 뷰 l의 자식 뷰가 실체 뷰 i와 의존 관계를 가지고 있기 때문에 선택된 실체 뷰 i에 의하여 의존 관계가 없는 뷰 l의 공헌도 이익이 영향을 받으며 따라서 실체화 이익 b_l을 재계산 해주어야 한다. 그림 4-c)를 보면 실체 뷰 i의 자식 뷰와 의존 관계가 없는 뷰 l의 자식 뷰의 공통 자식 뷰

들중 뷰 l로부터 계산되는 비용이 실체 뷰 i로부터 계산되는 비용보다 적은 자식 뷰 z만이 공현도 이익에 영향을 주기 때문에 의존 관계가 없는 뷰 l의 실체화 이익 b_l은 재계산된다. 여기서 의존 관계가 없는 뷰 l을 V_{13}이라 한다면 v는 l만을 조상으로 하는 뷰 V_{10}이 되고, l과 i의 공통 자식 뷰 z중 (V_{20}, V_{23}, V_{30}, V_{33}, V_{03}, V_{00}) 의존 관계가 없는 뷰 l로 계산될 때 더 적은 비용을 가진 z만을 포함하며 식 (11)로 나타낼 수 있다.

$$b_l = JC(l, MD) - C(l, l) + \sum_{\{v \mid v<l,\ i \not< l,\ v<i \text{ on complex lattice } CL\}} (JC(v, MD) - C(v, l)) + \sum_{\{z \mid z<l,\ z<i \text{ on complex lattice } CL\}} (JC(z, MD) - C(z, l)) \quad (11)$$

이상에서 설명한 실체 뷰 선택 과정은 실체화시킴으로써 가장 큰 이익을 낼 수 있는 실체 뷰를 먼저 선택하고 이득을 재계산하여 다음으로 가장 큰 이익을 낼 수 있는 실체 뷰를 선택하는 과정을 원하는 실체화 뷰 개수만큼 반복하는 greedy 알고리즘으로 요약할 수 있다 (그림 5 참조). 그림 5에서 k는 실체화 하고자 하는 실체 뷰의 개수를 의미한다.

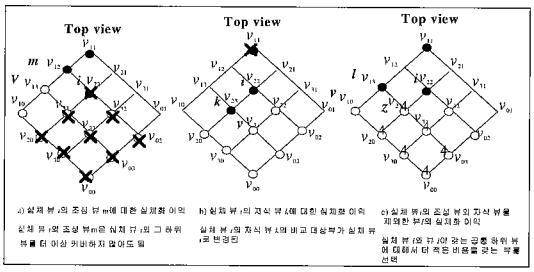

그림 4 실체 뷰 i 선택 후 다음 선택을 위한 실체화 이익 재계산

단계 1: 조인 테이블 산출
- 1-1: 복합 격자의 모든 뷰에 대해 필요한 조인 테이블을 산출한다;

단계 2: 복합 격자의 모든 뷰에 대한 실체화 이익 초기 계산
- 2-1: 모든 뷰에 대한 조인 비용을 계산한다.
- 2-2: 모든 뷰에 대한 실체화 이익을 계산한다.

단계 3: 실체 뷰 선택
- 3-1: 가장 큰 실체화 이익을 갖는 뷰 i를 실체 뷰로 선택한다.
- 3-2: 선택된 실체 뷰가 k개 이면 단계 5로 간다.

단계 4: 실체화 이익 재계산
- 4-1: 실체 뷰 i에 영향을 받는 조상 뷰들의 실체화 이익을 재계산한다.
- 4-2: 실체 뷰 i에 영향을 받는 자식 뷰들의 실체화 이익을 재계산한다.
- 4-3: 실체 뷰 i와 의존 관계가 없는 나머지 뷰들의 실체화 이익을 재계산한다.
- 4-4: 단계 3으로 간다.

단계 5: 종료

그림 5 실체 뷰 선택 알고리즘

4.3 실체 뷰 선택의 예

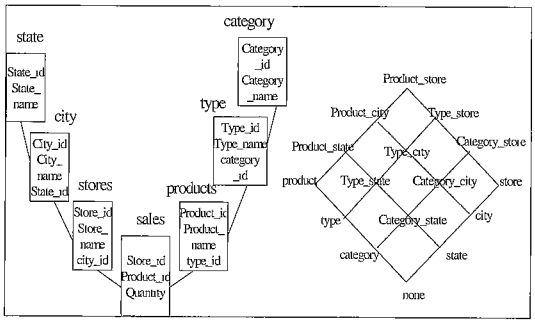

그림 6 예제 데이타 웨어하우스

예제 데이타 웨어하우스는 그림 6과 같이 2개의 차원(product, store)으로 구성되어 있으며, 사실 테이블은 sales 테이블이다. 사실 테이블인 sales 테이블에는 판매액을 의미하는 quantity 속성을 포함하고 있다. product 차원은 3개의 차원 속성 레벨 products, type, category로 이루어져 있으며, store 차원에는 store, city, state로 차원 속성 레벨이 이루어져 있다. 스노우플레이크 스키마로 구성된 예제 데이타 웨어하우스는 각 차원의 차원 속성 레벨마다 차원 속성 테이블을 구성하고 있다. 2개의 차원으로 구성된 데이타 웨어하우스에서 정의되는 뷰들은 의존 관계에 따라 복합 격자를 형성하며 16개의 뷰들로 구성된다. 16개의 뷰들은 서로 다른 요약 수준을 가지고 있다. 예로, type_city라는 뷰는 판매액을 상품의 타입과 상점의 도시별로 판매액을 요약하는 뷰를 의미하고, category_state 뷰는 판매액을 상품의 카테고리와 상점의 주별로 판매액을 요약하는 뷰를 의미한다.

- 차원 속성 테이블을 이용한 조인 테이블 추적

예제 데이타 웨어하우스에의 복합 격자에 정의된 모든 뷰의 조인 테이블을 차원 속성 레벨을 이용하여 추적한다. 표 1은 예제 데이타 웨어하우스의 복합 격자를 구성하는 모든 뷰들에 대한 조인 테이블을 추적한 것이다.

- 추적된 조인 테이블을 기반으로 모든 뷰의 조인 비용을 계산

예제 데이타 웨어하우스의 각 테이블의 튜플의 수와 테이블의 레코드의 크기는 다음과 같다고 하자.

n_{sales} = 100 (단위: row) s_{sales} = 20 (단위: byte)
$n_{products}$ = 10 $s_{products}$ = 15 n_{stores} = 10 s_{stores} = 15
n_{type} = 5 s_{type} = 15 n_{city} = 5 s_{city} = 15 $n_{category}$ = 2
$s_{category}$ = 10 n_{state} = 2 s_{state} = 10

모든 뷰에 대한 조인 비용을 계산하면 표2와 같다.

표 1 예제 데이타 웨어하우스에 정의된 모든 뷰에 대한 조인 테이블

뷰 이름	표기	조인 테이블
product_store	V_{11}	T_{00} = { sales }
type_store	V_{21}	T_{10} = { products, sales }
product_city	V_{12}	T_{01} = { stores, sales }
category_store	V_{31}	T_{20} = { type, products, sales }
type_city	V_{22}	T_{11} = { products, stores, sales }
product_category	V_{13}	T_{02} = { city, stores, sales }
product	V_{10}	T_{00} = { sales }
category_city	V_{32}	T_{21} = { type, products, stores, sales}
type_state	V_{23}	T_{12} = { product, stores, city, sales }
store	V_{01}	T_{00} = { sales }
city	V_{02}	T_{01} = { store, sales }
category_state	V_{33}	T_{22} = { type, products, city, stores, sales }
type	V_{20}	T_{10} = { products, sales }
state	V_{03}	T_{01} = { stores, city, sales }
category	V_{30}	T_{20} = { city, stores, sales }
none	V_{00}	T_{00} = { sales }

표 3 분석적 알고리즘을 이용한 예제 데이타 웨어하우스에 정의된 실체 뷰의 크기

실체 뷰	실체 뷰 크기	실체 뷰	실체 뷰 크기
product_store	10×10×20 = 2000	type_state	5×2×20 = 200
type_store	5×10×20 = 1000	product	10 ×15 = 150
product_city	10×5×20 = 1000	city	5 × 15 = 75
category_store	2×10×20 = 400	category_state	2×2×15 = 60
type_city	5×5×20 = 500	type	5 × 15 = 75
product_state	10×2×20 = 400	state	3 ×15 = 45
store	10×15 = 150	category	2 ×15 = 30
category_city	2×5×20 = 200	none	1 × 10 = 10

● 실체 뷰의 크기 추정

예제 데이타 웨어하우스의 실체 뷰의 크기를 분석적 알고리즘으로 추정하면 표 3과 같다.

● 실체화 이익 도출

표 4는 본 논문에서 제안한 알고리즘을 이용하여 계산되어진 예제 데이타 웨어하우스의 각 뷰에 대한 실체화 이익이다. 실체 뷰의 선택 개수를 3개로 한다면 3번의 반복이 이루어지며, type_city, product_city, type_ store가 선택된다.

표 2 예제 데이타 웨어하우스에 정의된 모든 뷰에 대한 조인 비용

뷰	조인 테이블	조인 비용 (단위: byte)
product_store	sales	$n_{sales}S_{sales}$ = 2000
type_store	sales, products	$n_{products}S_{products}+n_{products}n_{sales}S_{sales}$ = 20150
product_city	sales, stores	$n_{stores}S_{stores}+n_{stores}n_{sales}S_{sales}$ = 20150
category_store	sales, products, type	$n_{type}S_{type}+n_{type}n_{products}S_{products}+n_{products}(S_{type}+S_{products})+n_{products}n_{sales}S_{sales}$ = 21125
type_city	sales, products, stores	$n_{products}S_{products}+n_{products}n_{sales}S_{sales}+n_{stores}S_{stores}+n_{stores}n_{sales}(S_{sales}+S_{products})$ = 55300
product_state	sales, stores, city	$n_{city}S_{city}+n_{city}n_{stores}S_{stores}+n_{stores}(S_{city}+S_{stores})+n_{stores}n_{sales}S_{sales}$ = 21125
store	sales	$n_{sales}S_{sales}$ = 2000
category_city	sales, stores products, type	$n_{type}S_{type}+n_{type}n_{products}S_{products}+n_{products}(S_{type}+S_{products})+n_{products}+n_{sales}S_{sales}+n_{stores}S_{stores}+n_{stores}n_{sales}(S_{type}+S_{products}+S_{sales})$ = 71175
type_state	sales, stores products, city	$n_{city}S_{city}+n_{city}n_{stores}S_{stores}+n_{stores}(S_{city}+S_{stores})+n_{stores}n_{sales}S_{sales}+n_{products}S_{products}+n_{products}n_{sales}(S_{city}+S_{stores}+S_{sales})$ = 71175
product	sales	$n_{sales}S_{sales}$ = 2000
city	sales, stores	$n_{stores}S_{stores}+n_{stores}n_{sales}S_{sales}$ = 20150
category_state	sales, products, stores, city, type	$n_{type}S_{type}+n_{type}n_{products}S_{products}+n_{city}S_{city}+n_{city}n_{stores}S_{stores}+n_{products}(S_{type}+S_{products})+n_{products}n_{sales}S_{sales}+n_{stores}(S_{city}+S_{stores})+n_{stores}n_{sales}(S_{type}+S_{products}+S_{sales})$ = 72250
type	sales, products	$n_{products}S_{products}+n_{products}n_{stores}$=20150
state	sales, stores, city	$n_{city}S_{city}+n_{city}n_{stores}S_{stores}+n_{stores}(S_{city}+S_{stores})+n_{stores}n_{sales}S_{sales}$ = 21125
category	sales, stores, city	$n_{type}S_{type}+n_{type}n_{products}S_{products}+n_{products}(S_{type}+S_{products})+n_{products}n_{sales}S_{sales}$ = 21125
none	sales	$n_{sales}S_{sales}$ = 2000

표 4 예제 데이타 웨어하우스에서 정의된 모든 뷰의 실체화 이익

뷰	1회	2회	3회
V12	298375	37100	
V21	298375	37100	37100
V13	210800	26500	9450
V22	317025		
V31	210800	26500	26500
V10	40700	2100	1100
V23	204875	12800	12800
V32	204875	12800	12800
V01	40700	2100	2200
V20	42675	2975	2975
V33	116260	12435	12435
V02	42675	2975	2975
V30	23065	3015	3015
V03	23065	3015	3015
V00	1999	499	499

4.4 논의

본 연구의 가장 큰 의미는 실체 뷰 선택 문제를 기존의 데이타 큐브 환경의 단순 격자 구조에서 일반적인 데이타 웨어하우스 환경인 복합 격자 구조로 확장한 것이다.

단순격자 구조에서 실체 뷰 선택 : 기존의 연구들은 데이타 웨어하우스의 사실 테이블만을 포함한 데이타 큐브에서 실체 뷰를 선택하는 연구이었다. 일반적으로 데이타 큐브는 소규모 용량의 데이타 마트(data mart)에 사용되고 적은 수의 차원으로 구성되어진다. 이러한 환경에서 실체 뷰는 사실 테이블만으로 이루어지는 단순 격자를 형성하며 실체 뷰 선택에 중요한 기준이 되는 질의 처리 비용은 실체 뷰 크기와 같다고 가정하고 있다.

복합격자 구조에서 실체 뷰 선택 : 실세계에서 운영되는 데이타 웨어하우스들은 전사적인 대규모의 데이타 웨어하우스에 이용되며, 사용자의 다차원적 관점을 지원하기 위해 수 십 개의 차원과 차원계층을 포함하고 있다. 따라서 이러한 데이타 웨어하우스는 하나의 사실 테이블과 여러 개의 차원 테이블을 포함하고 있다. 데이타 웨어하우스에서 생성되는 실체 뷰는 사실 테이블과 차원 테이블의 조인으로 형성된 복합 격자를 구성하는 뷰이고, 실체 뷰 선택시 가장 중요한 요소가 되는 질의 처리 비용은 조인 비용이다.

본 연구에서는 전사적인 대규모의 데이타 웨어하우스 환경에도 적용할 수 있는 실체 뷰 선택 알고리즘을 제시하고 있으며 실체뷰 선택에 있어 가장 중요한 요소인 조인 비용을 고려하였다. 이 과정에서 실체 뷰에 필요한 조인 테이블 추적을 정형화하였으며 실체 뷰 선택 문제를 수학적 모형으로 정형화를 시도하였다.

5. 결 론

본 논문에서는 사실 테이블에 포함된 차원 속성만을 이용하여 실체 뷰를 선택하는 기존의 연구와는 달리 차원 테이블들과 사실 테이블의 조인으로 이루어진 데이타 웨어하우스에서 실체 뷰를 선택하는 문제에 대한 해결 방안을 greedy 알고리즘을 기반으로 제시하였다. 이를 위해, 일차적으로 nested-loop 방식의 조인 비용을 고려했으며, 실체 뷰의 차원 속성 레벨을 이용하여 조인 테이블을 추적하여 추적된 조인 테이블을 기반으로 실체 뷰의 조인 비용을 정형화 하고자 하였다.

본 연구의 완전성을 위해서는 향후 연구가 수반되어야 한다. 첫째, 많은 데이타 웨어하우스환경에서 star- join방식과 hash-join 조인 방식이 많이 사용되고 있으며[13], 이러한 조인 방식을 기반으로 조인 비용을 정형화하는 것이 필요하다. 둘째, 실체 뷰 선택을 단순 격자에서 복합 격자로 확장하면 실체 뷰는 기하급수적으로 증가하므로 탐색 영역을 줄이는 휴리스틱 기법이 알고리즘에 적용되어야 한다.

참 고 문 헌

[1] E. Baraalis, S. Paraboschi, and E. Teniente, "Materialized View Selection in a Multidimensional Database," Proc. of the 23rd VLDB Conference, 1997, pp. 156-165.

[2] S. Chaudhuri, and U. Dayal. "An Overview of Data Warehousing and OLAP Technology," SIGMOD Record 26(1), 1997, pp. 65-74.

[3] H. Gupta, V. Harinarayan, A. Rajaraman, "Index Selection for OLAP," Proc. of the Int. Conf. on Data Engineering, Binghamtor, UK, April, 1997.

[4] J. Hammer, H. Garcia-Molina, J. Widom, W. Labio, and Y. Zhuge, "The Stanford Data Warehousing Project," IEEE Data Eng. Bulletin, Special Issue on Materialized View and Data Warehousing, 18(2), June, 1995, pp. 41-48.

[5] Peter J. Hass, Jeffery F. Naughton, Arun N. Swami. "On the Relative Cost of Sampling for Join Selectivity Estimation," Proc. of the 13th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 1994, pp. 14-24.

[6] V. Harinarayan, A. Rajaraman, and J. D Ullman, "Implementing Data Cube Efficiently," Proc. of the ACM SIGMOD Conf., June 1996, pp. 205-216.

[7] W. H. Inmmon, "Building the Data Warehouse," 2nd Ed. John Wiley & Sons. Inc, 1996.

[8] P. Mishra, Margaret, and H. Eich, "Join Processing in Relational Database," ACM Computing Surveys, Vol. 24, No. 1, March, 1992.

[9] Red Brick Systems, "Star Schema Processing For Complex Queies." white paper 1997.

[10] Amit Shukla, Prasad M. Deshpande, Jeffery F. Naughton, Karthikeyan Ramasamy, "Storage Estimation for Multidimensional Aggregates in the Presence of Hierarchies," Proc. of the 22nd VLDB Conference, Bombay, India, 1996.

[11] Bennet Vance, and David Maier, "Rapid Bushy Join-order Optimization with Cartesian Products." Proc. of the ACM SIGMOD, 1996, pp. 35-46.

[12] J. Widom, "Research Problems in Data Warehousing," Proc. of the 4th Int. Conf. on Information and Knowledge Management(CIKM), 1995, pp. 25-30.

[13] Yihong Zhao, Prasad M.Deshpande, Jeffery F. Naughton, Amit Shulka, "Simultaneous Optimization and Evaluation of Multiple Dimensional Queries," Proc. of ACM SIGMOD, 1998, pp. 271-282.

[14] 서은주, "뷰 이용률을 고려한 효율적인 데이타 큐브 구현 기법." 포항공대, 석사학위논문, 1997.
[15] 오영배, "데이타 웨어하우스 환경에서 hB-tree를 이용한 다중테이블 조인 기법." 서울대학교 전산통계학과, 석사학위논문, 1998.
[16] 장동인, "실무자를 위한 데이타 웨어하우스." 대청출판사, 1998

윤 원 식
1997년 중앙대학교 정보시스템학과 졸업(학사). 2000년 중앙대학교 대학원 정보시스템학과 졸업(석사). 2000년 ~ 현재 현대정보기술 투신IT실 영업정보팀. 관심분야는 데이타 웨어하우스. OLAP, 데이타 마이닝

신 동 천
1985년 서울대학교 컴퓨터 공학과 졸업(학사). 1987년 한국과학기술원 전산학과 졸업(석사). 1991년 한국과학기술원 전산학과 졸업(박사). 1991년 ~ 1993년 한국전산원 선임연구원. 1993년 ~ 현재 중앙대학교 정보시스템학과 부교수. 관심분야는 다중 데이타베이스, 데이타 웨어하우스, 이동 컴퓨팅