데이타 웨어하우스 환경에서 최적 실체뷰 구성을 위한 효율적인 탐색공간 생성 기법

(An Efficient Search Space Generation Technique for Optimal Materialized Views Selection in Data Warehouse Environment)

이 태 희 † 장 재 영 †† 이 상 구 †††

(Tae-Hee Lee) (Jae-young Chang) (Sang-goo Lee)

요 약 데이타 웨어하우스에서의 분석 질의는 대체로 복잡한 연산을 포함하고 있기 때문에 질의 처리 과정이 매우 중요하다. 성능 향상을 위해서 데이타 웨어하우스에서 보편적으로 쓰이고 있는 방법은 실체뷰를 구축하는 것이다. 어떤 실체뷰를 구축하느냐 하는 문제는 데이타 웨어하우스 전체의 질의처리 성능과 유지보수 비용에 중요한 영향을 미친다. 실체뷰 구성 문제란 이러한 질의처리 비용과 유지보수비용을 고려하여 최적의 실체뷰를 선택하는 것이다. 본 논문에서는 이러한 최적의 실체뷰를 구성하는 효율적인 해결방안을 제시한다. 최적 실체뷰의 구성문제는 일반적으로 NP-hard 문제이지만, 본 논문에서는 관계형 데이타베이스에서 사용되는 조인, 선택, 그룹, 집계 연산의 특성을 고려하여 문제해결을 위한 탐색 공간을 획기적으로 줄이는 방법을 제안한다.

키워드 : 데이타 웨어하우스, 질의처리, 뷰 관리, 실체뷰

Abstract A query processing is a critical issue in data warehouse environment since queries on data warehouses often involve hundreds of complex operations over large volumes of data. Data warehouses therefore build a large number of materialized views to increase the system performance. Which views to materialized is an important factor on the view maintenance cost as well as the query performance. The goal of materialized view selection problem is to select an optimal set of views that minimizes total query response time in addition to the view maintenance cost. In this paper, we present an efficient solution for the materialized view selection problem. Although the optimal selection of materialized views is NP-hard problem, we developed a feasible solution by utilizing the characteristics of relational operators such as join, selection, and grouping.

Key words : data warehouse, query optimization, view maintenance, materialized view

1. 서 론

현재 생산되는 정보의 양은 폭발적으로 증가하고 있으며 이러한 데이타들을 체계적으로 관리하여 의사 결정 등에 필요한 자료의 분석 정보를 빠르게 얻고자 하는 요구도 이에 따라 증가하고 있다. 그러나 기존의 데이타베이스 기술로는 이러한 사용자의 분석 요구를 만족시키지 못하였으며, 과거와 현재의 분산된 데이타들을 주제와 시간에 따라 정리하여 저장할 필요성이 대두되었다. 데이타 웨어하우스(data warehouse)는 이러한 사용자들의 요구를 충족시키기 위하여 오랫동안 축적되어 온 이질적인 데이타들을 통합하여 저장한 대용량의 정보 저장고이다.

데이타 웨어하우스 구축의 목적은 이러한 분산되고 이질적인 기존의 수많은 데이타 원천들로부터 새로운 분석 정보를 얻는 것이다. 데이타 웨어하우스의 분석 작업은 OLAP(On-Line Analytic Processing) 등을 통하여 이루어지는데, 이러한 분석 작업에 사용되는 질의들은 많은 양의 데이타를 탐색하고 복잡한 집계 연산을 필요로 하는 것이 일반적이다. 실시간 분석 작업에 있어서 이렇게 시간이 오래 걸리는 것을 원하지 않기 때문에 많은 질의 최적화 방법들이 연구되어 왔다. 예를 들어, 분석 질의에 반드시 사용되는 집계 연산을 효율적으

· 본 연구는 2004년도 한성대학교 교내연구비 지원과제임

† 비 회 원 : 서울대학교 컴퓨터공학부
thlee@europa.snu.ac.kr
†† 종신회원 : 한성대학교 컴퓨터공학부 교수
jychang@hansung.ac.kr
††† 종신회원 : 서울대학교 컴퓨터공학부 교수
sglee@europa.snu.ac.kr

논문접수 : 2002년 2월 25일
심사완료 : 2004년 9월 14일

로 처리하기 위한 방법들이나 비트-맵 인덱스(bitmap index)나 조인 인덱스(join index) 등을 사용하여 질의 처리 성능을 높이는 방법들이 그 예이다[15].

또 다른 방법은 자주 사용되는 질의의 결과를 미리 저장하여 두는 실체뷰(materialized view)를 구축하는 방법이다[11]. 실체뷰는 데이타 웨어하우스에 저장되어 있는 기본 테이블과는 별도로 뷰를 실체화시켜 저장하여 둔 것으로, 자주 사용되는 질의의 결과를 미리 계산하여 저장해 둠으로써 오랜 시간을 필요로 하는 복잡한 질의에 대한 결과를 빠른 시간 내에 얻을 수 있다는 장점을 갖고 있다. 그러나 어떤 질의에 대한 결과를 미리 계산하여 두어야 전체적인 질의 처리 성능을 높일 수 있는가를 판단하는 것은 매우 어려운 문제이다. 왜냐하면 하나의 실체뷰가 그 자체로는 자주 사용되지 않는 질의의 결과라 하더라도 다른 질의들의 수행 결과의 부분 또는 전체를 가지고 있다면 전체적으로 시스템의 질의 처리 성능이 향상된 결과를 낳을 수 있기 때문이다. 또한 실체뷰의 구축은 데이타들을 쓰기 쉬운 형태로 중복 저장하여 두는 것이기 때문에 실체뷰를 사용하는 질의의 결과가 기본 테이블을 이용한 질의의 결과와 일관된 상태가 되도록 기본 테이블의 변경 사항을 실체뷰에 반영하여야 하는 실체뷰의 유지(view maintenance) 문제가 생기게 된다. 따라서 질의 처리에 직접적으로 사용되지 않더라도 이런 실체뷰의 유지비용을 줄이는 것이 전체적인 비용을 줄이는 데 기여할 수 있는 상황도 생길 수 있다. 이러한 실체뷰를 구축하는데 있어서 어려운 문제중의 하나는 주어진 사용자 질의에 대해서 구축 가능한 실체뷰의 수가 매우 방대하여 모든 가능한 실체뷰들을 대상으로 최적의 실체뷰를 구성한다는 것은 사실상 불가능하다는 것이다. 따라서 이들로부터 성능을 최대화 할 수 있는 최적의 실체뷰 집합을 찾는데 있어서 탐색 공간을 최소화하는 것은 매우 중요한 문제라고 볼 수 있다.

본 논문에서는 최적의 실체뷰를 구축하는 과정에서 위에서 언급한 질의처리 비용과 실체뷰의 유지비용을 모두 고려하여 최적 실체뷰 탐색을 위한 효율적인 탐색 공간 생성 기법을 제시한다. 이미 데이타 큐브(data cube)[7]나 다중 뷰 처리 계획(multiple-view-processing plan)[21] 등을 이용하여 실체뷰 탐색 및 구축에 관한 방법들이 제안되었으나 질의 모델의 단순성 또는 알고리즘의 복잡도 때문에 실제 상황에 적용하기 어려운 방법들이 대부분이었다. 따라서 본 논문에서는 일반적인 SQL로 질의 모델을 확장함과 동시에, 기존의 방법들이 간과하고 있는 관계형 데이타베이스의 특성을 이용하여 알고리즘의 복잡도를 줄여 실제 상황에 이용할 수 있는 새로운 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 정리하고 3장에서는 실체뷰 구성의 문제를 정의한다. 4장에서는 최적 실체뷰에 포함될 가능성이 있는 실체뷰를 정의하며, 5장에서는 최적 실체뷰 탐색을 위한 탐색 공간 생성 알고리즘을 제안하고 실험을 통해 이 알고리즘의 효율성을 검증한다. 마지막으로 6장에서는 결론과 향후 연구계획을 기술한다.

2. 관련 연구

실체뷰의 구축 문제에 있어서 가장 중요한 점은 실체뷰의 유지비용과 질의처리비용을 최소화하는 것으로 지금까지의 연구들도 이 문제의 해결에 집중되어 있다. 실체뷰의 구축은 질의 처리 성능을 향상시키는 장점이 있지만 실체뷰를 유지하는 데 추가적인 비용이 들게 되는데 이러한 유지비용을 줄이는 방법들이 [5,13,16] 등에서 제안되었다. 이 방법들은 데이타 원천(data source)이 변경될 때마다 실체뷰를 다시 만들지 않고 원천의 변경 사항만을 이용하여 실체뷰의 변경 사항을 계산해 내는 방법을 사용하고 있다. 특히 [16]에서는 비용을 디스크 엑세스의 양으로 추론하는 방법을 이용하여 최초로 여러 개의 실체뷰가 구축되어 있을 때 유지비용을 최소화 문제에 대해서 형식적인 기반을 마련하였다. [13]에서는 여기서 더 나아가 구축된 실체뷰의 유지비용을 줄이는 것이 아니라 유지비용을 최소로 하는 실체뷰와 인덱스를 구해내는 A*알고리즘을 제안하고 있다.

질의 처리비용을 최소로 하는 실체뷰들을 선택하는 문제에 대한 해결책들은 [7,8,10-12,14,20,21] 등에서 제안되었다. [7]에서 제안한 큐브 연산자는 이런 노력의 출발점이 되었다. 큐브 연산자를 사용했을 때 생기게 되는 격자(lattice)의 각 노드를 실체뷰 구축의 대상으로 보게 된 것이다. 이런 격자내의 실체뷰들 중 적은 공간을 차지하면서 최적과 근사한 질의 처리 성능을 보일 수 있는 실체뷰의 집합을 찾아내는 알고리즘이 [8,11] 등에서 제안되었다. [11]에서는 사용할 수 있는 디스크의 크기에 제한이 있을 때 격자 내의 노드들 중에서 실체뷰로 구축할 것을 찾아내는 그리디 알고리즘(greedy algorithm)을 제안하였고, [8]에서는 실체뷰 뿐 아니라 인덱스까지 선택의 대상으로 확장하였다. 그러나 이 방법들에서는 공통적으로 실체뷰의 유지비용을 고려하지 않았고 큐브 연산자의 특성상 GROUPBY 연산만을 이용하여 질의 모델을 만들었다는 문제가 있었다.

데이타 큐브를 사용하는 위 방법들의 문제점을 해결하고자 [10]에서는 보다 일반적인 질의 모델과 유지 모델 사용하여 최적 실체뷰 문제의 해결하는 방법에 대한 기반을 확립하였다. 이 방법에서는 질의 처리 계획을 AND-OR 그래프로 표현하여 그래프의 각 노드를 실체

뷰로 만들 수 있는 대상으로 보았다. [9,14]에서도 AND-OR 그래프를 사용하였으며 실체뷰의 유지비용을 최적화 시키는 방법이 제안되었다. 이 방법들에서 AND-OR 그래프의 각 노드들은 질의 처리 계획에서의 한 노드로 표현될 수 있는 것들인데, 이 방법들에서의 실체뷰로 만들 수 있는 대상은 주어진 질의 집합에 대한 다중 질의 처리 계획[19] 중 생기는 중간 결과, 즉 다중 질의에 대한 일부 결과(sub-expression)로 제한되고 있다.

[2,18,20]등에서도 여러 가지 환경에서의 실체뷰 구성 방법을 제안하고 있는데 [2]에서는 함수적 종속(functional dependency)과 계층 구조, 그리고 실체뷰의 크기를 예상해내는 휴리스틱(heuristic)을 사용하여 실체뷰를 찾는 알고리즘의 탐색 공간을 줄이는 방법이 제안되었고, [18]에서는 모든 질의가 실체뷰만으로 처리될 수 있을 때 유지비용을 최소화하는 방법을 해결하였다. 또한 [20]에서는 다시 데이타 큐브에 대한 고찰이 이루어져 데이타 큐브 격자 중 크기의 제한이 있는 특수한 격자에 대해서 실체뷰를 선택하는 방법의 성능을 향상시켰다. 최근의 연구들로 [1]에서는 Microsoft SQL Server에서의 실체뷰와 인덱스의 구성 방법에 대하여 제안하고 있는데 질의 비용을 이용하여 실체뷰 대상 공간을 프러닝(prunning)하고 두 개의 실체뷰를 합하여 하나의 실체뷰로 구성하는 방법을 제안하고 있으나 이러한 실체뷰들은 같은 릴레이션을 조인하고 있어야하는 제한사항이 있다. [4]에서는 이러한 실체뷰 선택 문제의 문제 복잡도를 비용 모델의 완전성에 따라 이론적으로 분석하고 있다.

이들 외에 데이타 웨어하우스 환경에서의 캐싱(caching)에 관한 연구들에서도[6,12] 주어진 질의 집합을 알지 못할 경우에 대해 캐쉬에 담아두어야 할 질의 결과들에 관해 연구되었는데 캐쉬 내에 구축할 실체뷰를 사용자의 질의에 따라 동적으로 변화시키는 방법을 제안하고 기존의 정적인 실체뷰 구축 방법과 비교하고 있다. 그러나 이 연구들은 주로 제한된 저장 공간에서 이전 질의 결과들을 새로운 질의 결과로 대체하는 방법에 관한 연구들로 실체뷰의 유지비용을 고려하지 않았고 실체뷰 대상이 질의 결과 전체 또는 일부분으로 제한되고 있다.

지금까지 언급한 기법들은 질의 모델이 비교적 단순하여 실제 SQL 기반의 환경에서는 제한적으로 적용될 수밖에 없었다. 또한 여러 개의 질의 결과를 모두 포함하는 실체뷰를 만들어 내는 방법을 고려하지 않고 주로 질의 결과의 일부만을 실체뷰의 대상으로 삼아 이용 가능한 실체뷰를 제한적 방법으로 탐색하는 반면 본 논문에서는 데이타베이스의 특성을 활용하여 일반적인 질의 모델을 기반으로 최적의 실체뷰를 효율적으로 탐색할 수 있는 탐색공간 생성 기법을 제안한다.

3. 최적 실체뷰 구성 문제의 정의

3.1 실체뷰 구성의 예

데이타 웨어하우스 스키마가 그림 1과 구성되어 있다고 가정하자. 이 다차원 스키마(multidimensional schema)에서는 차원 테이블(dimension table)로 각 상점에 대한 정보를 가지고 있는 *Store* 릴레이션, 상점에서 취급하는 상품 정보에 대한 *Product* 릴레이션, 그리고 시간을 가지고 있는 *Time* 릴레이션이 있다. 차원에 대한 사실값을 가지고 있는 사실 테이블(fact table)로는 상점에서의 상품의 판매량을 가지고 있는 *Sales* 릴레이션이 있다. 이러한 스키마에 대해서 자주 쓰이는 질의로서 다음과 같은 Q_1과 Q_2가 있다고 하자.

Q_1 : SELECT StoreName, Day, SUM(Quantity)
FROM Store S, Sales A, Time T
WHERE S.StoreKey = A.StoreKey AND A.TimeKey = T.TimeKey
GROUP BY StoreName, Day

Q_2 : SELECT ProductName, SUM(Quantity)
FROM Product P, Sales A
WHERE P.ProductKey = A.ProductKey
GROUP BY ProductName

Q_1은 각 상점에서의 날짜별 판매량의 합을 구하는 질의이고, Q_2는 상품별 판매량의 합을 구하는 질의이다. 이때 다음과 같은 질의로 실체뷰 V_1을 구축했다고 가정하자.

V_1 : SELECT StoreName, ProductName, Day, SUM(Quantity) as SUM_QUANTITY
FROM Store S, Sales A, Product P, Time T
WHERE S.StoreKey = A.StoreKey
AND A.ProductKey = P.ProductKey
AND A.TimeKey = T.TimeKey
GROUP BY StoreName, ProductName, Day

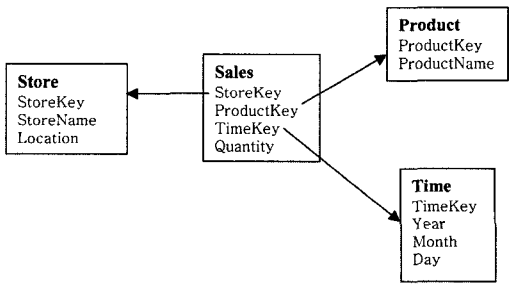

그림 1 다차원 스키마의 예

그러면 질의 Q_1과 Q_2는 각각 다음의 질의 Q_1'과 Q_2'와 같이 V_1만으로 결과를 얻을 수 있다.

```
Q1' : SELECT   StoreName, Day,
                SUM(SUM_QUANTITY)
      FROM     V1
      GROUP BY StoreName, Day
Q2' : SELECT   ProductName,
                sum(SUM_QUANTITY)
      FROM     V1
      GROUP BY ProductName
```

이와 같이 V_1을 실체뷰로 구축함으로써 Q_1'과 Q_2'는 각각 Q_1과 Q_2와 비교하여 조인의 필요성이 없어지므로 질의처리 비용이 줄어들 것으로 기대할 수 있다. 또 다른 예로써 V_1과 함께 다음과 같은 V_2를 실체뷰로 구축했을 경우를 살펴보자.

```
V2 : SELECT   S.StoreKey, S.ProductKey, T.Day,
              SUM(Quantity) AS SUM_QUANTITY
     FROM     Sales S, Time T
     WHERE    S.TimeKey = T.TimeKey
     GROUP BY S.StoreKey, S.ProductKey, T.Day
```

V_2는 FROM절에 *Store*와 *Product*를 포함하고 있지 않으므로 질의 Q_1과 Q_2 중 어느 질의의 결과도 포함하고 있지 않다. 그러나 *Store*와 *Product* 릴레이션에 빈번한 수정이 일어나는 경우를 생각해 보자. 이 경우 V_2가 실체뷰로 구축되어 있지 않다면 V_1을 최신의 상태로 유지하기 위해서 수정이 일어난 튜플과 *Sales* 릴레이션, *Time* 릴레이션을 조인하여 V_1에서 갱신해야 될 내용들을 생성해야한다. 그러나 V_2가 실체뷰로 구축되어 있으므로 이런 조인이 필요 없이 V_2와의 조인만으로 V_1의 갱신 내용을 생성해낼 수 있으므로 전체적인 유지비용이 줄어들 가능성이 있다. 이와 같이 질의 처리에 직접적으로 사용될 수 없더라도 유지비용을 최소화하는 데 실체뷰가 쓰일 수 있으므로 최적 실체뷰의 구성 문제를 풀 때에는 이러한 경우까지도 모두 고려하여야 한다.

3.2 최적 실체뷰 구성 문제의 형식적 정의

앞서 말한 바와 같이 최적 실체뷰 구성 문제는 질의처리 비용과 유지비용을 모두 고려하여야 한다. 데이타 웨어하우스에 실체뷰의 집합 $\boldsymbol{V} = \{V_1,...,V_n\}$가 구축되어 있을 때 임의의 질의 Q를 처리하는데 드는 질의처리 비용은 구축된 실체뷰의 집합 $\boldsymbol{V}$ 중 질의처리에 이용할 수 있는 실체뷰들에 따라 결정된다. 또한 실체뷰의 유지비용도 실체뷰의 유지에 쓰일 수 있는 실체뷰들에 따라 결정된다. 이것은 전체적으로 보면 구축되어 있는 실체뷰에 따라 질의처리 비용과 실체뷰의 유지비용이 결정된다고 보는 것이다. 따라서 자주 쓰이는 질의의 집합 $\boldsymbol{Q}$에 대해서 질의처리 비용은 $\boldsymbol{Q}$와 $\boldsymbol{V}$의 함수인 $C_q(\boldsymbol{Q}, \boldsymbol{V})$로 나타낼 수 있고, 유지비용은 $C_m(\boldsymbol{V})$로 나타낼 수 있다. 따라서 데이타 웨어하우스의 스키마 $\boldsymbol{D}$에 대하여 자주 쓰이는 질의의 집합 $\boldsymbol{Q}$가 주어졌을 때 최적 실체뷰는 다음과 같이 정의될 수 있다.

정의 1. 최적 실체뷰(optimal materialized views)

데이타 웨어하우스의 스키마 $\boldsymbol{D}$와 질의의 집합 $\boldsymbol{Q}$에 대하여 질의처리 비용을 $C_q(\boldsymbol{Q}, \boldsymbol{V})$라 하고 실체뷰의 유지비용을 $C_m(\boldsymbol{V})$라 할 때 다음과 같은 전체 비용을 최소로 하는 실체뷰의 집합 $\boldsymbol{V}$를 최적 실체뷰라 한다.

$$C(\boldsymbol{Q}, \boldsymbol{V}) = C_q(\boldsymbol{Q}, \boldsymbol{V}) + t * C_m(\boldsymbol{V})$$

단, t는 질의처리 비용에 대한 실체뷰의 유지비용의 중요도를 나타내는 상수이다. □

3.3 질의 모델

데이타 웨어하우스의 분석 질의는 SQL로 표현했을 때 대개 GROUPBY와 집계 함수를 많이 필요로 한다. 예를 들어 그림 1과 같은 스키마에서 상품별 판매량의 시간에 따른 변화, 각 상점의 월별 판매량의 변화 등의 분석 질의는 모두 GROUPBY에 이은 집계 함수로 표현되는 질의들이다. 본 논문에서는 GROUPBY와 집계 함수가 포함된 단일 블록(single block)의 집계 질의(aggregate query)를 가정한다. 이러한 질의들을 표현하기 위해 다음과 같은 관계대수 연산자를 정의한다.

$\pi[P]R$: 릴레이션 R에 속성 집합 P를 프로젝션하는 연산이다.

$F[P]R$: 릴레이션 R에 속한 속성 P에 대하여 집계 함수 F를 적용하는 연산이다. F는 집계 함수의 집합으로서 $F=\{f_1,, f_n\}$과 같이 정의되고 각 $f_i(1 \le i \le n)$는 {SUM, COUNT, MAX, MIN} 중의 하나이다. 이 연산의 결과는 P에 속한 속성들을 이용하여 각 f_i 값들을 계산한 것을 나타낸다. 예를 들어, $P = \{a_1, a_2, a_3\}$일 때 각 f_i는 MIN(a_1), COUNT(a_3), SUM(a_1) 등과 같은 함수가 될 수 있다.

$\mathbf{G}[P]R$: 릴레이션 R에 대하여 속성 집합 P로 SQL의 GROUPBY 연산을 수행한 결과이다.

$\sigma[S]R$: 릴레이션 R에 대하여 선택(selection) 연산 S를 수행한 결과이다. S는 WHERE절에 나오는 선택 술어들 중 조인 조건이 아닌 선택 연산 술어들만을 포함한다.

$\bowtie[\boldsymbol{R}]$: $\boldsymbol{R}$은 하나 이상의 릴레이션으로 구성된 집합으로, 하나의 릴레이션일 경우 아무런 연산도 수행하지 않은 릴레이션이 그대로 결과가 되고, 여러 개의 릴레이션을 포함하고 있는 집합일 경우 $\boldsymbol{R}$에 속한 각 릴레이션들을 자연 조인(natural join)한 것이 결과가 된다.

추가적으로 이후에 쓰일 대수 연산식을 위해 위에서

정의한 연산자들 이외에 아래와 같은 표기법을 정의한다.

- *attr*(***R***) : 릴레이션 집합 ***R***에 있는 모든 속성의 집합.
- *common*(R_1, R_2): 릴레이션 집합 R_1과 R_2의 공통 속성.
- *subattr*(*A*, ***R***): 속성의 집합 *A* 중에서 릴레이션 집합 ***R***에 포함된 속성의 집합
- *pred*(*S*, ***R***) : *S*는 선택 술어의 집합. *S*에 속한 술어들 중에서 릴레이션 집합 ***R***의 속성에 대한 술어들의 집합.

4. 질의를 이용한 실체뷰의 구축 방법

본 장에서는 주어진 질의에 대한 실체뷰를 어떻게 생성할 것인가에 대해 논한다. 우선 주어진 질의의 결과를 모두 포함하는 실체뷰를 정의하고, 그 다음으로 질의의 일부분만을 포함하는 실체뷰를 정의하는 방법에 대해 논한다.

4.1 질의의 결과를 포함하는 실체뷰의 구성

질의의 결과를 모두 포함하는 실체뷰를 구축하는 방법에 대해 살펴보자. 직관적으로 질의 $Q_1,...,Q_n$의 결과를 모두 포함하는 실체뷰를 *V*라고 할 때, *V*는 모든 $Q_i(1 \le i \le n)$에서 조인되고 있는 릴레이션들을 조인한 결과를 갖고 있어야 한다. 또한 *V*의 GROUPBY 속성은 모든 Q_i의 GROUPBY 속성들을 포함하여 *V*에 Q_i의 GROUPBY 연산을 다시 한번 수행함으로써 Q_i의 GROUPBY 결과를 얻을 수 있어야 한다. 그리고 *V*의 선택 연산은 Q_i의 선택 연산들을 OR함으로써 Q_i의 결과를 모두 포함하고 있어야 하며, 마지막으로 집계 연산도 Q_i에 속한 집계 연산의 결과를 포함할 수 있도록 구성되어야 한다.

이때 가장 문제가 되는 것은 조인 연산이다. *V*로부터 Q_i의 결과를 추출해 낼 수 있으려면 Q_i에 속한 릴레이션의 조인의 결과가 *V*에는 속하지만 Q_i에 속해 있지 않은 릴레이션과의 조인에 의해 중복되거나 누락되는 튜플이 발생해서는 안 된다. 만약 조인에 의해 중복이나 누락이 발생한다면 질의 처리에 실체뷰를 사용할 수 없다. 이러한 튜플의 중복과 누락이 생기는 것을 결정하는 요인은 함수적 종속성(*functional dependency*)이다. 질의 Q_i의 릴레이션 집합이 $\boldsymbol{R}_Q$이고 이 릴레이션들을 포함하는 *V*의 릴레이션 집합이 $\boldsymbol{R}_V$라고 가정하자. 이때 *V*에서 $attr(\boldsymbol{R}_Q) \rightarrow attr(\boldsymbol{R}_V)$가 만족되면 조인에 의해 $\boldsymbol{R}_Q$의 튜플이 중복되거나 누락되지 않으므로 *V*로부터 Q_i의 결과를 얻어낼 수 있다.

결국 Q_i의 결과는 $attr(\boldsymbol{R}_Q)$에 의해 함수적으로 결정(functionally determine)되는 속성으로 구성된 어떤 릴레이션과 조인되더라도 조인 결과에 대한 프로젝션 연산으로 다시 Q_i의 결과를 얻어낼 수 있음을 보장할 수 있다. 그러나 하나의 질의를 처리하기 위한 실체뷰는 질의의 결과를 그대로 실체뷰로 구축하면 가장 최적의 실체뷰가 되며, 질의에 쓰인 릴레이션에 함수적으로 결정되는 릴레이션이라 하더라도 질의에 쓰이지 않았을 경우 조인하게 되면 질의처리 비용이나 유지비용에 있어서 이득이 없다. 이것은 두개 혹은 그 이상의 질의의 결과를 가지고 있는 실체뷰를 구축하고자 할 때에도 그대로 적용되므로 질의에 쓰이지 않는 릴레이션을 조인할 필요가 없다. 따라서 본 논문에서 제안하고 있는 방법에서 필요로 하는 것은 질의에 쓰이지 않는 릴레이션을 정보의 손실 없이 조인할 수 있는 조건이 아니라, 기본적으로 두개의 질의에 대해서 이 두 질의의 결과를 포함하고 있는 하나의 실체뷰를 만들기 위한 조건이다. 이러한 조건을 위해 다음과 같이 조인 무손실 개념을 정의한다.

정의 2. 조인 무손실(*join-lossless*)

릴레이션 R_1과 R_2의 자연조인인 $\bowtie[R_1, R_2]$에서 다음의 함수적 종속이 항상 만족되면, R_1과 R_2는 조인 무손실하다.

$$attr(R_1) \cap attr(R_2) \rightarrow attr(R_1)$$
$$attr(R_1) \cap attr(R_2) \rightarrow attr(R_2) \quad \square$$

이러한 조인 무손실의 성질을 이용하여 두 질의의 결과를 포함하고 있는 실체뷰의 구축 조건을 제한할 수 있다. 제한된 조건은 다음의 결합-뷰에 대한 정의로 표현할 수 있다.

정의 3. 결합-뷰(*union-view*)

질의 Q_1과 Q_2에 대하여 $Q_1 = \pi[G_1]F_1[A_1]\mathbf{G}[G_1]\sigma[s_1]\bowtie[\boldsymbol{R}_1]$ 이고, $Q_2 = \pi[G_2]F_2[A_2]\mathbf{G}[G_2]\sigma[s_2]\bowtie[\boldsymbol{R}_2]$일 때, $\boldsymbol{R}_1$과 $\boldsymbol{R}_2$가 조인 무손실하면 다음과 같은 질의 Q_3을 Q_1과 Q_2의 결합-뷰이라고 한다.

$$Q_3 = \pi[G_1 \cup G_2]F''[A_1 \cup A_2]\mathbf{G}[G_1 \cup G_2]\sigma[s_1 \vee s_2]\bowtie[\boldsymbol{R}_1 \cup \boldsymbol{R}_2]$$

단, $F'' = \{ f_k(b_k) \mid f_k(b_k)$는 표 1에서 정의된 F_1과 F_2의 각 집계 함수 $f_i(b_i)$에 대하여 A부분의 $f_k(b_k)$에 해당하는 집계 함수$\}$ 이다. $\square$

표 1은 질의처리 과정에서 하나의 집계함수를 두 개의 집계함수로 분할하여 처리하는 규칙을 나타낸다. 즉, 집계함수 $f_i(b_i)$는 $f_j(b_j)$와 $f_k(b_k)$로 분할할 수 있음을 의미한다. 예를 들어, 3.1절에서 질의 Q_1과 Q_1'에서 Q_1의 집계함수 SUM(Quantity)는 Q_1'의 집계함수 SUM(SUM_QUANTITY)와 Q_1'에서 사용되는 실체뷰 V_1에서 정의된 SUM(Quantity)로 분할된 것으로 해석할 수 있다. 집계함수 분할 규칙에 대한 자세한 설명과 증명은 [3,17] 등에서 참조할 수 있다.

이러한 결합-뷰 연산에 의해 구해진 실체뷰를 이용하여 항상 질의의 결과를 얻을 수 있음을 다음의 정리에서 증명하고 있다.

정리 1. 정의 3의 Q_1, Q_2, Q_3 에 대해서 Q_3의 결과가

표 1 집계함수 분할 규칙

	$f_i(b_i)$	$f_j(b_j)$	$f_k(b_k)$
A	SUM (b_i)	SUM (N)	SUM (b_j)
	COUNT (b_i)	SUM (N)	COUNT (b_j)
	MIN (b_i)	MIN (N)	MIN (b_j)
	MAX (b_i)	MAX (N)	MAX (b_j)
B	SUM (b_i)	SUM(N* b_j)	COUNT (N)
	COUNT (b_i)	SUM (N)	COUNT (N)
	MIN (b_i)	MIN (b_j)	X
	MAX (b_i)	MAX (b_j)	X

X : 필 요 없 음
N : $f_k(b_k)$의 결 과
n : 임 의 의 속 성

실체뷰 V라면 다음의 Q_1'과 Q_2'의 결과는 각각 Q_1과 Q_2의 결과와 동일하다.

$Q_1' = \pi[G_1]F_1'[A_1]\mathbf{G}[G_1]\sigma[s_1]\bowtie[V]$

$Q_2' = \pi[G_2]F_2'[A_2]\mathbf{G}[G_2]\sigma[s_2]\bowtie[V]$

단, F_1' = { $f_j(b_j)$ | $f_j(b_j)$는 표 1에서 정의된 F_1의 각 집계 함수 $f_i(b_i)$에 대하여 A부분의 $f_j(b_j)$에 해당하는 집계 함수}

F_2' = { $f_j(b_j)$ | $f_j(b_j)$는 표 1에서 정의된 F_2의 각 집계 함수 $f_i(b_i)$에 대하여 A부분의 $f_j(b_j)$에 해당하는 집계 함수}

증명. Q_1'에서 V 대신에 Q_3의 대수식을 이용하면 Q_1'은 다음과 같이 표현할 수 있다.

$Q_1' = \pi[G_1]F_1'[A_1]\mathbf{G}[G_1]\sigma[s_1]$

$(\pi[G_1 \cup G_2]F''[A_1 \cup A_2]\mathbf{G}[G_1 \cup G_2]\sigma[s_1 \vee s_2]\bowtie[\boldsymbol{R}_1 \cup \boldsymbol{R}_2])$

$=\pi[G_1]\pi[G_1 \cup G_2]F_1'[A_1]\mathbf{G}[G_1]F''[A_1 \cup A_2]\mathbf{G}[G_1 \cup G_2]$

$\sigma[s_1]\sigma[s_1 \vee s_2]\bowtie[\boldsymbol{R}_1 \cup \boldsymbol{R}_2]$

여기서 $\sigma[s_1]=\sigma[s_1]\sigma[s_1 \vee s_2]$이고, $\pi[G_1] = \pi[G_1]\pi[G_1 \cup G_2]$이므로 위의 식은 다음과 같이 나타낼 수 있다.

$Q_1' = \pi[G_1]F'[A_1]\mathbf{G}[G_1]F''[A_1 \cup A_2]\mathbf{G}[G_1 \cup G_2]\sigma[s_1]\bowtie[\boldsymbol{R}_1 \cup \boldsymbol{R}_2]$

이때 집계 함수의 분할 법칙에 의하면

$F_1'[A_1]\mathbf{G}[G_1]F''[A_1 \cup A_2]\mathbf{G}[G_1 \cup G_2] = F_1[A_1]\mathrm{G}[G_1]$

이므로 Q_1'은 다시

$Q_1' = \pi[G_1]F_1[A_1]\mathbf{G}[G_1]\sigma[s_1]\bowtie[\boldsymbol{R}_1 \cup \boldsymbol{R}_2]$

가 된다. 결합-뷰의 정의에 의하면 $\boldsymbol{R}_1$과 $\boldsymbol{R}_2$는 조인 무손실 성질을 만족하므로 $\pi[G_1]\bowtie[\boldsymbol{R}_1 \cup \boldsymbol{R}_2]$는 $\pi[G_1]\bowtie[\boldsymbol{R}_1]$과 같다. 따라서

$$Q_1' = \pi[G_1]F_1[A_1]\mathbf{G}[G_1]\sigma[s_1]\bowtie[\boldsymbol{R}_1]$$

이고, 이것은 Q_1과 같다. Q_2에 대해서도 위와 같은 방법으로 Q_2'이 Q_2와 같음을 증명할 수 있다. □

결국, 질의 Q_1과 Q_2의 결과를 포함하는 실체뷰 V를 구축하기 위해서 V는 결합-뷰 조건을 만족해야하고 정리 1에서 증명한 바와 같이 이 실체뷰를 이용해서 Q_1이나 Q_2의 결과를 얻어낼 수 있다.

4.2 질의 결과의 일부분을 포함하는 실체뷰의 구성

질의 결과의 일부분을 갖는 실체뷰를 구축하는 방법에 대해 살펴보자. 직관적으로 질의 Q의 결과의 일부분만을 포함하는 실체뷰를 V라고 할 때, V는 Q에서 조인되고 있는 릴레이션들 중 일부분을 조인하고 있어야 한다. 또한 Q의 GROUPBY 속성들 중 V와 Q에 공통되는 릴레이션의 속성들을 GROUPBY 속성으로 포함하면 된다. 이 조건이 만족되어야만 다른 릴레이션과의 조인 후에 다시 한번 Q의 GROUPBY 연산을 수행함으로써 Q의 결과를 얻을 수 있기 때문이다. V의 선택 연산은 V와 Q에 공통되는 릴레이션에 대한 선택 연산들에 의미적으로 포함되는 연산이어야 한다. 집계 연산은 V와 Q에 공통되는 릴레이션에 속한 속성에 대한 집계 연산의 결과만을 포함하고 있으면 된다.

이와 같이 질의 결과의 일부분만을 포함하고 있는 실체뷰의 경우에는 질의에 쓰인 릴레이션들 중 어느 일부분을 조인하고 있지 않으면 질의처리 비용은 커질 수 있지만, 유지비용은 작아질 수 있고, 결과적으로 전체 비용이 작아질 수 있다. 그러므로 이러한 실체뷰가 최적 실체뷰에 포함될 수 있는 가능성도 있다. 따라서 질의에서 조인하고 있는 릴레이션들 중 임의의 릴레이션을 조인에서 제외시켜 실체뷰를 구축하고, 이 실체뷰를 이용하여 원래 질의의 결과를 계산해 낼 수 있는 조건이 필요하다. 이러한 실체뷰는 다음과 같이 부분-뷰의 정의로 표현할 수 있다.

정의 4. 부분-뷰(partial-view)

질의 Q_1이 $Q_1 = \pi[G_1]F[A_1]\mathbf{G}[G_1]\sigma[s_1]\bowtie[\boldsymbol{R}_1 \cup \boldsymbol{R}_2]$와 같이 정의될 때, 다음과 같은 질의 Q_2의 결과를 Q_1에 대한 $\boldsymbol{R}_1$의 부분-뷰라고 한다.

$Q_2 = \pi[subattr(G_1, \boldsymbol{R}_1) \cup (\mathrm{common}(\boldsymbol{R}_1, \boldsymbol{R}_2)]F'[subattr(A_1, \boldsymbol{R}_1)]$

$\mathbf{G}[subattr(G_1, \boldsymbol{R}_1) \cup (\mathrm{common}(\boldsymbol{R}_1, \boldsymbol{R}_2)]\sigma[pred(s_1, \boldsymbol{R}_2)]$

$\bowtie[\boldsymbol{R}_1]$

단, F'' = { $f_k(b_k)$ | $f_i(b_i) \in F[A_1]$인 각 $f_i(b_i)$에 대해서 $b_i \in \mathrm{attr}(\boldsymbol{R}_1)$이면 $f_k(b_k)$는 표 1에서 정의된 $f_i(b_i)$에 대하여 A부분의 $f_k(b_k)$에 해당하는 집계 함수이고, $b_i \in \mathrm{attr}(\boldsymbol{R}_2)$이면 $f_k(b_k)$는 B부분의 $f_k(b_k)$에 해당하는 집계 함수} □

Q_1에 대한 $\boldsymbol{R}_1$의 부분-뷰란 Q_1에서 $\boldsymbol{R}_2$에 관한 연산을 모두 제외시키는 것이다. 여기서 주의할 점은 부분-뷰에 대한 질의의 GROUPBY 속성에 $\mathrm{common}(\boldsymbol{R}_1, \boldsymbol{R}_2)$이 포함된 것이다. 이 속성은 부분-뷰 질의의 결과를 이용하여 다시 $\boldsymbol{R}_2$와 조인함으로써 Q_1의 결과를 얻어내기 위하여 조인에 필요한 속성을 GROUPBY에 사용한 것이다. 또 한가지 주의할 점은 부분-뷰 정의에 의해 만들어진 질의 Q_2는 $\boldsymbol{R}_2$와 관련된 연산을 제외한 Q_1의

모든 연산을 수행하는 질의라는 점이다. 이렇게 만들어진 실체뷰는 R_2와의 조인 후에 다시 계산되어야 할 연산을 이미 처리해 놓은 것이므로, 이를 이용한 질의처리 비용은 Q_1을 직접 수행하는 것보다 적을 것으로 예상할 수 있다. 뿐만 아니라 선택 연산이나 GROUPBY 연산에 의해 실체뷰의 크기가 작아질 수 있으므로 유지비용 또한 작아질 것을 기대할 수도 있다. 다음의 정리는 부분-뷰를 이용하여 만들어진 질의를 다시 다른 릴레이션과 조인함으로써 원래의 질의의 결과를 얻을 수 있음을 증명하고 있다.

정리 2. 정의 4의 Q_1과 Q_2에 대해서 Q_2의 결과가 실체뷰 V라면, 다음의 질의 Q_1'의 결과는 Q_1과 동일하다.

$$Q_1' = \pi[G_1]F'[A_1]\mathbf{G}[G_1]\sigma[s_1]\bowtie[V, R_2]$$

단, $F' = \{ f_j(b_j) \mid f_i(b_i)\in F[A_1]$인 각 $f_i(b_i)$에 대해서 $b_i\in$attr(R_1)이면 $f_j(b_j)$는 표 1에서 정의된 $f_i(b_i)$에 대하여 A부분의 $f_j(b_j)$에 해당하는 집계 함수이고, $b_i\in$attr(R_2)이면 $f_j(b_j)$는 B부분의 $f_j(b_j)$에 해당하는 집계 함수$\}$

증명. Q_1'의 대수식에서 V 대신에 Q_2의 정의를 이용하면 Q_1'은 다음과 같이 표현할 수 있다.

$$\begin{aligned} Q_1' = \; & \pi[G_1]F'[A_1]\mathbf{G}[G_1]\sigma[s_1]\bowtie[R_2, \\ & \pi[subattr(G_1, R_1)\cup(\text{common}(R_1, R_2)] \\ & F'[subattr(A_1, R_1)] \\ & \mathbf{G}[subattr(G_1, R_1)\cup(\text{common}(R_1, R_2)] \\ & \sigma[pred(s_1, R_2)]\bowtie[R_1]] \end{aligned}$$

여기서 $\pi[subattr(G_1, R_1)\cup(\text{common}(R_1, R_2)]$은 이후의 연산에서 더 이상 쓰이지 않고, $\pi[G_1]$에 의해 최종적으로 프로젝션되므로 생략할 수 있다. 따라서 Q_1'은 다음과 같이 쓸 수 있다.

$$\begin{aligned} Q_1' = \; & \pi[G_1]F'[A_1]\mathbf{G}[G_1]\sigma[s_1]\bowtie[R_2, \\ & F''[subattr(A_1, R_1)] \\ & \mathbf{G}[subattr(G_1, R_1)\cup(\text{common}(R_1, R_2)] \\ & \sigma[pred(s_1, R_2)]\bowtie[R_1]] \end{aligned}$$

또한, $\sigma[pred(s_1, R_2)]$는 조인 연산과 순서를 교환해도 관계없으므로 다음과 같이 표현할 수 있다,

$$\begin{aligned} Q_1' = \; & \pi[G_1]F'[A_1]\mathbf{G}[G_1]\sigma[s_1]\sigma[pred(s_1, R_2)]\bowtie[R_2, \\ & F''[subattr(A_1, R_1)] \\ & \mathbf{G}[subattr(G_1, R_1)\cup(\text{common}(R_1, R_2)] \\ & \bowtie[R_1]] \end{aligned}$$

이때 $\sigma[s_1]\sigma[pred(s_1, R_2)] = \sigma[s_1]$이므로 Q_1'은 다음과 같다.

$$\begin{aligned} Q_1' = \; & \pi[G_1]F'[A_1]\mathbf{G}[G_1]\sigma[s_1]\bowtie[R_2, F''[subattr(A_1, R_1)] \\ & \mathbf{G}[subattr(G_1, R_1)\cup(\text{common}(R_1, R_2)] \\ & \bowtie[R_1]] \end{aligned}$$

여기서 집계 함수의 분할 규칙에 따르면

$$\begin{aligned} & F'[A_1]\mathbf{G}[G_1]\sigma[s_1]\bowtie[R_2, (F''[subattr(A_1, R_1)] \\ & \qquad \mathbf{G}[subattr(G_1, R_1)\cup(\text{common}(R_1, R_2))] \\ & \qquad \bowtie[R_1])] \\ = \; & F[A_1]\mathbf{G}[G_1]\sigma[s_1]\bowtie[R_1 \cup R_2] \end{aligned}$$

이므로,

$Q_1' = \pi[G_1]F[A_1]\mathbf{G}[G_1]\sigma[s_1]\bowtie[R_1\cup R_2] = Q_1$이다. □

결국 실체뷰가 질의 결과의 일부분만을 포함할 경우에도 질의처리에 이용될 수도 있고 유지비용도 절감하는 효과를 얻을 수 있다. 예를 들어 3.1절의 실체뷰에 대한 예에서 V_2는 V_1의 유지비용을 줄일 뿐만 아니라, 다음과 같이 *Store* 릴레이션과의 조인으로 Q_1과 동일한 결과를 생성할 수 있다.

```
SELECT  S.StoreName, V2.Day, SUM(V2.SUM_
        QUANTITY)
FROM    Store S, V2
WHERE   S.StoreKey = V2.StoreKey
GROUP BY S.StoreName, V2.Day
```

물론 이 질의는 조인 연산을 포함하므로 V_1만을 이용하는 Q_1'에 비해서 질의처리 비용이 더 소요될 가능성이 높으나, V_1의 유지비용을 절감하는 효과 때문에 최적 실체뷰에 포함될 가능성을 배제할 수 없다.

5. 최적 실체뷰 구성 기법

최적 실체뷰 구성 과정은 크게 두 단계로 나뉠 수 있다. 첫 번째 단계는 실체뷰 탐색 공간의 생성이다. 실체뷰 탐색 공간이란 최적 실체뷰에 포함될 가능성이 있는 실체뷰들의 집합을 의미한다. 모든 질의의 결과를 계산하여 저장하면 실체뷰가 될 수 있지만, 모두가 최적 실체뷰에 포함될 수 있는 것은 아니다. 이런 모든 실체뷰들 중에서 최적 실체뷰에 포함될 가능성이 있는 질의들의 범위를 결정하는 것이 실체뷰 탐색 공간의 생성이 해결하고자 하는 문제이다. 그 다음 단계는 탐색 공간에서 최적 실체뷰를 결정하는 탐색 알고리즘 부분이다. 앞 단계가 탐색의 범위를 결정하는 부분이라면 탐색 알고리즘은 탐색 공간에 포함된 실체뷰들을 하나씩 찾아나가면서 최적 실체뷰를 구성하는 부분이다.

5.1 실체뷰 탐색 공간의 생성

질의를 이용한 실체뷰 탐색 공간의 생성은 앞 절에서 정의한 결합-뷰와 부분-뷰를 이용한다. 결합-뷰와 부분-뷰를 이용하면 질의의 결과를 포함하는 실체뷰뿐만 아니라 질의 결과의 일부분만을 포함하는 실체뷰도 정의할 수 있으므로, 질의 최적화와 실체뷰의 유지에 사용되는 실체뷰들을 모두 생성할 수 있게 된다.

실체뷰 탐색 공간의 생성은 우선 기본 릴레이션에 대해 부분-뷰를 생성하고, 이렇게 정의된 실체뷰의 각 쌍에 대해서 결합-뷰를 생성한다. 이와 같이 부분-뷰를 생성함으로써 유지비용이 적은 실체뷰를 구할 수 있고, 결합-뷰를 생성함으로써 두 질의의 결과를 포함하는 실

```
Algorithm CreateViewSpace(Q)
INPUT : Q        // 최적화 하고자 하는 질의의 집합
OUTPUT : S       // 가능한 실체뷰의 집합
BEGIN
  S = ∅
  candidates = Q
  while (candidates is not empty) {
        new_candidate = ∅
        for each query Q in candidates
            for each relation R in Q
                new_candidates = new_candidates ∪ (Q에서 R을 제외한
                                 릴레이션의 부분-뷰)
        S = S ∪ candidates
        candidates = new_candidates
  }
  for each query Q1, Q2 in S
        if Q1 and Q2 are join-lossless
              S = S ∪ (Q1과 Q1의 결합-뷰)
END
```

그림 2 실체뷰 탐색공간 생성 알고리즘

체뷰를 만들어 낼 수도 있다. 결합-뷰와 부분-뷰를 이용한 실체뷰 공간의 생성 알고리즘은 그림 2와 같다. 여기서 입력은 사용자 질의의 집합인 $\boldsymbol{Q}$이고 최종 출력은 구축 가능한 실체뷰의 집합 S이다. 이 알고리즘에서 보는바와 같이 우선 $\boldsymbol{Q}$의 각 질의에 대해서 가능한 부분-뷰를 생성하고 생성된 각 부분-뷰에 대해서도 반복적으로 같은 작업을 수행한다. 그러면 주어진 질의 집합 $\boldsymbol{Q}$에 대해서 가능한 모든 부분-뷰들을 생성할 수 있다. 그 다음 단계로는 지금까지 생성된 실체뷰들의 각 쌍에 대해서 가능한 결합-뷰들을 생성하게 된다.

시간복잡도

질의 집합 $\boldsymbol{Q}$의 각 질의에 포함된 릴레이션의 개수를 k라 하고 질의의 개수를 $|Q|$라 하면 그림 2의 알고리즘 중 질의 집합인 $\boldsymbol{Q}$에 속한 모든 질의 Q에 대하여 부분-뷰를 생성하는 복잡도는 $O(|Q| \circ 2^k)$이다. 여기서 다시 각 부분-뷰에 대하여 결합-뷰를 만드는 복잡도는 $O(2^{|Q| \cdot 2^k})$이다.

[4]에 의하면 데이타베이스의 통계 정보를 완전히 알고 있다는 가정 하에 실체뷰 선택 문제에서의 탐색 공간의 생성에는 최악의 경우 질의에 사용된 릴레이션의 이중 지수(double-exponential) 만큼의 최소 임계값(lower bound)을 갖는 것으로 증명되었다. 여기서 말하는 최악의 경우는 모든 릴레이션의 조합으로 조인이 가능한 경우이다. 그림 2의 알고리즘도 앞서 설명한 바와 같이 최악의 경우 [4]에서 증명한 바와 같은 복잡도를 가지고 있다. 그러나 일반적인 복잡한 관계형 데이타베이스 스키마에서는 이처럼 모든 릴레이션의 조인이 가능하지 않으며, 본 논문에서 제안하는 조인 무손실 등의 조건을 이용하여 이러한 복잡도는 실제 상황에서 크게 줄어들 수 있다. 이러한 경우에 대하여 5.2절에서 실험으로 그 타당성을 검증하였다.

지금까지 최적 실체뷰에 대한 개념적인 정의를 기반으로 실체뷰의 탐색공간을 생성하는 방법을 제시하였지만, 질의처리 비용과 실체뷰 유지비용에 대한 구체적인 정량적 비용 모델은 제시하지 않았다. 그러나 다음과 같은 제한된 비용 모델을 적용할 경우에 그림 2의 알고리즘은 최적 실체뷰에 속한 실체뷰들을 모두 생성한다는 사실을 정리 3과 4에서 증명한다. 아래의 비용 모델은 주어진 질의를 처리하는 데 소요되는 비용은 질의를 계산하는데 필요한 릴레이션의 크기와 비례한다는 선형 비용 모델(linear cost model)을 기반으로 한다[11].

비용모델 (cost model)

질의 $Q_1 = \pi[G_1]F_1[A_1]\mathbf{G}[G_1]\sigma[s_1]\bowtie[\boldsymbol{R}_1]$과 $Q_2 = \pi[G_2]F_2[A_2]\mathbf{G}[G_2]\sigma[s_2]\bowtie[\boldsymbol{R}_2]$에 대해서 다음의 조건을 모두 만족하면 Q_1의 질의처리 비용은 Q_2보다 적다.

- $\boldsymbol{R}_1 \subseteq \boldsymbol{R}_2$
- $F_1[A_1] \subseteq F_2[A_2]$
- $G_1 \subseteq G_2$
- $s_1 \supseteq s_2$ □

정리 3. 질의의 집합 $\boldsymbol{Q} = \{Q_1,..,Q_i,..,Q_k,...,Q_n\}$에 대한 최적 실체뷰의 집합 $\boldsymbol{V}$에 대해서, $V_m \in \boldsymbol{V}$인 V_m으로부터 다른 릴레이션과의 조인 없이 $Q_i,..,Q_k$의 결과와 동일

한 결과를 생성하는 질의들을 구성할 수 있다면 V_m은 $Q_i,..,Q_k$에 대한 결합-뷰로 정의할 수 있다.

증명. V_m이 $Q_i,..,Q_k$에 대한 결합-뷰로 정의된 실체뷰가 아니라고 가정하자. 그러면 V_m은 결합-뷰의 정의를 따르지 않도록 정의된 실체뷰이므로 우선 $Q_i,..,Q_k$의 릴레이션들이 조인 무손실 조건을 만족하지 않는 상황을 가정할 수 있다. 그렇다면 조인 무손실 정의에 따라 이들 릴레이션들의 조인 결과에는 $Q_i,..,Q_k$의 튜플들이 누락되거나 중복될 수밖에 없다. 따라서 V_m으로부터 $Q_i,..,Q_k$의 결과와 동일한 질의를 구성할 수 있다는 가정에 위배된다. 다음으로 V_m이 조인 무손실 조건을 만족하지만 릴레이션, GROUPBY 속성, 집계 연산, 선택 연산들이 정의 3의 결합-뷰에 대한 질의 구성과 같지 않다고 가정하자. 이를 위해 $V_j \in \boldsymbol{V}$이면서 $j \neq m$인 임의의 실체뷰 V_j를 생각해 보자. V_j로부터 V_m과 똑같이 $Q_i,..,Q_k$의 결과를 구해낼 수 있고, V_j는 $Q_i,..,Q_k$의 결합-뷰로 생성된 실체뷰가 아니라면 V_j는 V_m에서 정의된 릴레이션, GROUPBY 속성, 집계 연산을 포함한 더 많은 연산이 정의되었거나 더 적은 선택 연산이 정의된 결과이어야 한다. 이러한 V_j는 위에서 정의한 비용 모델에 의해 동일한 결과를 생성하면서도 V_m보다 더 많은 비용을 필요로 하므로 최적 실체뷰에 속할 수 없다. 따라서 이와 같이 질의의 결과를 모두 포함하는 실체뷰의 경우 최적 실체뷰에 속한 실체뷰는 질의들의 결합-뷰로 정의할 수 있다. □

정리 4. 질의 Q에 대하여, Q를 처리하는 과정에서 생성되는 중간 결과 중에서 최적 실체뷰의 집합에 포함될 가능성이 있는 것은 모두 Q의 부분 뷰로 정의할 수 있다.

증명. 부분-뷰의 정의에 따르면 Q의 부분-뷰로 표현할 수 있는 질의는 Q에서 조인된 릴레이션의 부분 집합에 대해서 Q의 선택 연산, GROUPBY 연산, 집계 연산 중 부분-뷰에 포함된 릴레이션에 관한 연산을 모두 수행한 질의이다. 이러한 부분-뷰로 표현할 수 있는 질의들 중에서 임의의 질의를 Q_d라 했을 때, Q_d에서 조인하고 있는 릴레이션들과 같은 릴레이션들을 모두 조인하고 있는 임의의 질의를 Q_d'라 하자. 만약 Q_d'이 Q_d보다 적은 GROUPBY 연산과 집계 연산을 수행하고 많은 선택 연산을 수행한 결과라면 Q_d'을 이용하여 Q를 계산해 낼 수 없으므로 Q_d'은 Q의 중간결과가 될 수 없다. 다음으로 Q_d'이 Q_d보다 많은 GROUPBY 연산과 집계 연산을 수행하고 적은 선택 연산을 수행한 결과라고 가정하자. 만약 실체뷰를 유지할 때마다 실체뷰의 결과를 재계산한다고 가정하면 위의 비용 모델은 실체뷰 유지비용에도 적용할 수 있다. 따라서 Q_d'을 실체뷰로 구축했을 때 Q_d보다 질의처리 뿐만 아니라 실체뷰 유지에도 더 많은 비용을 필요로 하므로 이 질의로 구축한 실체뷰는 최적 실체뷰의 집합에 포함될 수 없다. 따라서 최적 실체뷰의 집합에 포함될 가능성이 있는 중간 결과는 모두 Q의 부분-뷰로 표현할 수 있다. □

다음으로 해결해야하는 문제는 이렇게 생성된 실체뷰의 탐색 공간에 대해서 최적 실체뷰 찾는 것이다. 이 문제를 해결하는 가장 단순한 방법은 그림 2에서 생성된 실체뷰들의 모든 가능한 부분집합에 대해서 정의 1의 최적 실체뷰 조건을 만족하는 것을 찾는 것이다. 그러나 실체뷰의 개수에 따라 부분 집합의 개수는 지수 함수적으로 늘어나므로 이 방법은 때로 수행이 불가능할 수도 있다. 따라서 다양한 휴리스틱을 이용하여 탐색 공간을 획기적으로 줄이는 방법을 사용할 수도 있다. 그러나 최적 실체뷰의 탐색에 대한 문제는 본 논문의 범위를 벗어나고, 또한 [2,8,11,13,18] 등에서 많은 방법들이 제시되었으므로 더 이상 이 문제에 대해서는 논하지 않는다.

5.2 알고리즘의 성능

본 논문에서 제안한 실체뷰의 탐색 공간의 생성 알고리즘에 대해 그림 3과 같은 스키마에서 성능을 평가였다. 각 릴레이션 R_i는 3개의 GROUPBY 속성과 하나의 집계 연산이 가능한 속성을 가지고 있고, 화살표 방향으로 키에 의해 조인되어 있다. 예를 들어 R_2는 R_1, R_3, R_5의 키를 외래 키로 가지고 있어서 이 속성으로 조인 연산을 수행한다. 이러한 스키마에서 무작위로 한 개부터 다섯 개의 릴레이션을 선택하고 이 릴레이션들에 속한 GROUPBY 속성과 집계 연산 속성들도 임의로 선택하여 질의들을 생성하였다. 이렇게 생성된 질의들에 대하여 최적 실체뷰를 구하고자 할 때 고려하게 되는 실체뷰의 개수를 비교하여 본 논문이 제시한 방법의 탐색 공간이 어떻게 줄어드는 지를 설명하려 한다.

비교의 대상이 되는 방법은 [2]에서 제안한 데이타 큐브를 이용한 방법이다. [2]를 비교의 대상으로 한 이유는 이 방법이 기존의 다른 방법들 중에서 비교적 작은 탐색 공간을 갖기 때문이다. 이 방법에서는 실체뷰 탐색 공간의 크기를 줄이기 위해 차원 계층의 구조 정보 등

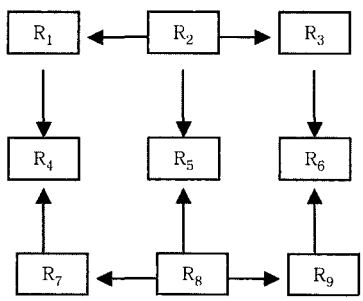

그림 3 성능 평가를 위한 스키마

을 이용한다. 생성되는 질의의 개수를 변화시켜 가면서 최적 실체뷰를 만들기 위해 고려하는 실체뷰의 개수를 비교하면 그림 4와 같다.

그림 4에 의하면 실체뷰 탐색 공간이 기존의 방법에서는 지수 함수적으로 증가하지만 본 논문에서 제안한 방법에서는 거의 선형으로 늘어나고 있다. 이것은 본 논문이 제안한 방법이 조인 무손실의 성질을 이용하기 때문이다. 데이타 큐브를 이용한 방법들은 테이블의 조인 관계와는 상관없이 GROUPBY 속성들을 모두 합하여 격자를 구성한다. 그러나 본 논문이 제안한 방법을 사용하면 조인 무손실하지 않은 릴레이션의 속성들에 대해서는 조인에 이은 GROUPBY 연산을 수행하더라도 질의의 결과를 얻을 수 없음을 알고, 실체뷰 탐색 공간에서 제외시킨다. 이런 차이가 그림 3과 같이 조인 관계가 서로 얽혀있는 스키마에서 실체뷰 탐색 공간을 줄이도록 만든 것이다. 물론 [2]에서 제안한 방법은 데이타 큐브만을 대상으로 하고 있으므로 본 논문에서 제안한 방법과의 성능 비교는 큰 의미가 없을 수 있다. 그러나 본 논문이 제안한 방법은 데이타 큐브라는 제한된 환경뿐만 아니라 관계형 모델의 일반 질의 환경에서도 적용할 수 있다는 점에서 그 의의를 찾을 수 있다.

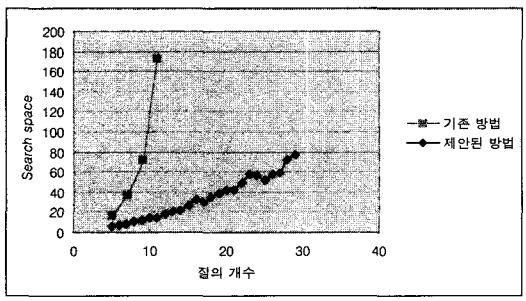

그림 4 실체뷰 탐색공간의 비교

6. 결 론

본 논문에서는 데이타 웨어하우스에서 최적 실체뷰를 구성하기 위한 실체뷰의 탐색 공간을 효율적으로 생성하는 방법을 제시하였다. 논문에서 제시하고 있는 방법은 광범위한 질의를 표현할 수 있는 일반적인 SQL 질의 모델을 사용하고 있으므로 복잡한 질의가 많거나 질의의 사소한 차이로 처리비용이 급격히 바뀔 수 있는 실제 환경에서도 적용할 수 있다. 데이타 웨어하우스 환경에서의 실체뷰 구성 문제는 NP-hard 문제이다. 따라서 실체뷰 구성 문제를 풀기 위해서는 모든 실체뷰의 집합에 대해서 최적을 찾는 방법보다는 탐색의 대상이 되는 실체뷰들을 제한하는 방법이 중요하다. 본 논문에서는 관계형 데이타베이스의 연산들의 특징을 활용하여 실체뷰의 탐색 공간을 줄일 수 있는 방법으로 조인 무손실, 결합-뷰, 부분-뷰 등을 정의하여 사용하였다. 이 방법은 데이타베이스의 스키마, 특히 조인 관계가 복잡할수록 다른 방법에 비해 좋은 성능을 보인다. 이것은 관계형 연산들의 특성을 이용하여 실체뷰 구성 방법을 제한한 결과이다. 향후에는 본 논문이 제안한 실체뷰의 탐색 공간 생성 문제를 발전시켜 다양한 휴리스틱을 이용한 최적 실체뷰 탐색 알고리즘을 개발할 계획이다.

참 고 문 헌

[1] S. Agrawal, S. Chaudhuri, V. Narasayya, Automated Selection of Materialized Views and Indexes for SQL databases, Materialized View Selection in Multidimensional Database, In Proc. of VLDB, pp.59-68, 2000.

[2] E.Baralis, S.paraboschi, E.Teniente, Materialized View Selection in Multidimensional Database, In Proc. of VLDB, pp.156-165, 1997.

[3] J.-y. Chang, S.-g. Lee, Extended Conditions for Answering an Aggregate Query Using Materialized Views, Information Processing Letters, Vol. 72 No. 5-6, pp. 205-212. 1999.

[4] R. Chirkova, A. Y.Halevy, D. Suciu, Formal Perspective on the View Selection Problem, In Proc. of VLDB, pp.59-68, 2001.

[5] Y. Cui, J. Widom. Lineage Tracing for General Data Warehouse Transformations, In Proc. of VLDB, pp. 471-480, 2001.

[6] P. M. Deshpande, K. Ramasamy, A. Shukla, J., Naughton, Caching Multidimensional Queries Using Chunks, In Proc. of ACM SIGMOD, pp.259-270, 1998.

[7] J.Gray, A.Bosworth, A.Layman, H.Piramish, Data Cube : A Relational Aggregatioon Operator Generalizing Group-By, Cross-Tab, and Sub-Totals, In Proc. of ICDE, pp. 152-159, 1996.

[8] H.Gupta, V.Harinarayan, A.Rajaraman, J.Ullman, Index Selection for OLAP, In Proc. of ICDE, pp.208-219, 1997.

[9] H.Gupta, I.S.Mumick, Selection of views to materialize under a Maintenance cost constraint, In Proc. of ICDT, pp. 453-460, 1999.

[10] H.Gupta, Selection of Views to Materialize in a Data Warehouse, in Proc. of ICDT, pp.98-112, 1997.

[11] V.Harinarayan, A.Rajaraman, J.Ullman, Implementing Data Cubes Efficiently, In Proc. of ACM SIGMOD, pp. 205-216, 1996.

[12] Y.Kotidis, N.Roussopoulos, Dynamat : A Dynamic View Management System for Data Warehouses, In Proc. of ACM SIGMOD, pp 371-382, 1999.

[13] W.J.Labio, D.Quass, B.Adelberg, Physical Database

Design for Data Warehouses, In Proc. of VLDB, pp. 277-288, 1997.
[14] H. Mistry, P. Roy, S. Sudarshan, K. Ramamritham, Materialized View Selection and Maintenance Using Multi-Query Optimization, In proc. of ACM SIGMOD, pp 310-318, 2001.
[15] P. O'Neil, D. Quass, Improved query performance with variant indexes, In Proc. of ACM SIGMOD, pp.38-49, 1997.
[16] K. A. Ross, D. Srivastava, S. Sudarshan, Materialized View Maintenance and Integrity Constraint Checking : Trading Space for Time, In Proc. of ACM SIGMOD, pp.457-448, 1996.
[17] D. Srivastava, S. Dar, H. V. Jagadish, and A. Y. Levy. Answering Queries with Aggregation Using Views. In Proc. of VLDB, pp. 318-329, 1996.
[18] A.Shukla, P.M.Deshpande, J.F.Naughton, Materialized View Selection for Multidimensional Data-Set, In Proc. of VLDB, pp 488-499, 1998.
[19] Timos K. Sellis, Multiple-query Optimization, ACM Transactions on Database Systems, pp. 23-52, 1988.
[20] D.Theodoratos and T.Sellis, Data Warehouse Configuration, In Proc. of VLDB, pp.126-135, 1997.
[21] J.Yang, K.Karlapalem, Q.Li, Algorithms for Materialized View Design in Data Warehousing Environment, In Proc. of VLDB, 1997.
[22] J. Yang and J. Widom. Making Temporal Views Self-Maintainable for Data Warehousing, In Proc of EDBT, pp 395-412, 2000.

이 태 희
1998년 KAIST 전산과학과 졸업(학사) 2000년 서울대학교 전산학과 졸업(석사) 2000년 3월~현재 서울대학교 컴퓨터공학부 박사과정. 관심분야는 데이타 웨어하우스, e-Catalog, 데이타 마이닝

장 재 영
1992년 서울대학교 계산통계학과 학사 1994년 서울대학교 계산통계학과 전산과학전공(석사). 1999년 서울대학교 계산통계학과 전산과학전공(박사). 2000년~현재 한성대학교 컴퓨터공학부 조교수. 관심분야는 데이타 웨어하우스, 데이타 마이닝

이 상 구
서울대학교 학사. 미 Northwestern University 석사/박사. University of Minnesota 전임강사, EDS 연구원 등 역임 현 서울대학교 컴퓨터공학부 교수, 서울대학교 e-비즈니스 기술연구센터장. 관심분야는 e-비즈니스 기술, e-Catalog, 데이터베이스