의미 정보와 실체뷰를 활용한 효율적 질의 재구성 기법

장 재 영†

요 약

실체뷰는 뷰의 연산 결과를 미리 저장한 형태로서 질의 성능을 향상시키기 위해 질의 처리과정에 활용될 수 있다. 주어진 질의를 처리하는데 있어서 어떠한 실체뷰를 어떻게 이용할 것인가는 쉽게 해결될 수 있는 문제가 아니며 지금까지도 많은 연구가 이루어지고 있다. 질의를 처리하는데 실체뷰가 이용가능한가의 여부는 주어진 질의와 실체뷰간의 관계에 따라 결정된다. 본 논문에서는 질의처리에 실체뷰를 활용하는 새로운 방법을 제안한다. 우선 기존의 질의와 실체뷰간의 문법적 관계를 확장하여 의미적 관계를 이용한 활용 방안을 제시한다. 또한 원래의 질의에는 포함되지 않는 릴레이션을 갖는 실체뷰의 활용 방안에 대해서도 논의한다. 이를 위해 본 논문에서는 실체뷰의 활용을 위한 조건들을 제시하고 이 조건들을 검증하고 질의를 재구성하는 알고리즘을 제시한다.

An Efficient Query Rewriting Technique Utilizing Semantic Information and Materialized Views

Jae-young Chang†

ABSTRACT

Materialized views which are stored views of the database offer opportunities for significant performance gain in query evaluation by providing fast access to pre-computed data. The question of when and how to use a materialized view in processing a given query is a difficult one attracting a significant amount of research. Whether a materialized view can be used in answering a query depends on the relationship between the view and the query. Proposed in this paper are new ways of utilizing materialized views in answering a query. Semantic relationships are used in addition to syntactic ones. We also utilize a materialized view in answering a query even if it has relations unrelated to the query. We first show the conditions for testing whether a materialized view can be utilized in answering a query and then present the algorithms for testing the conditions and reformulating a query with a materialized view.

키워드 : 실체뷰(Materialized View), 질의 처리(Query Processing), 질의 재구성(Query Rewrite), 의미정보(Semantic Information)

1. 서 론

실체뷰(materialized view)는 뷰의 연산 결과를 미리 저장한 형태로서 질의 처리의 성능을 향상시키는데 유용하게 이용될 수 있다. 실체뷰가 주어진 질의에 대해 일부 연산을 수행한 결과를 포함하고 있으면 해당 연산을 실행할 필요없이 실체뷰를 이용하여 그 결과를 얻음으로써 질의 처리 성능을 향상시킬 수 있다. 실체뷰를 이용한 질의 재구성 방법은 데이타 웨어하우스(data warehouse)나 분산 정보 시스템(distributed information system) 등에 유용하게 이용되고 있다. 예를 들어 데이타 웨어하우스에서는 주어진 질의를 실행하는데 있어서 규모가 큰 기본 테이블(base tables)을 접근하는 대신 요약 테이블(summary tables)을 이용하여 질의 결과를 얻는 방법에 응용될 수 있으며, 분산 정보 시스템에서는 근접 지역(local site)에 캐쉬(cache)한 이전 질의의 결과를 이용하여 원격 지역(remote site)에 접근하지 않고 현재의 질의를 처리하는 데 이용될 수 있다.

실체뷰를 효율적으로 활용하기 위해서는 다음과 같은 두 가지 문제가 해결되어야한다. 첫째는 실체뷰의 관리(view maintenance) 문제로 실체뷰를 캐쉬와 같이 기본 테이블로부터 추출된 일종의 중복된 테이블들로 볼 수 있으므로 실체뷰와 기본 테이블의 일관성을 유지하는 것이 중요한 문제가 된다[1, 5, 8, 15, 16]. 둘째는 주어진 질의에 연관된 활용 가능한 실체뷰를 효율적으로 탐색하고 찾아진 실체뷰를 이용하여 질의를 재작성(rewrite)하는 문제이다[4, 6. 7, 9-13]. 특히 질의 처리 과정에서 실체뷰의 활용은 기본 테이블 만을 이용하는 방법에 비해 현격한 성능의 향상을 가져올 수 있다.

현재까지 질의 처리에 실체뷰를 활용하는 다양한 방법들이 제시되었지만[2-6, 10-13], 실체뷰의 선택에 있어서 공통적인 몇 가지 제약이 존재한다. 첫째는 주어진 질의를 처리

※ 본 연구는 2003학년도 한성대학교 공학연구센터 특별연구비 지원과제임.
† 종신회원 : 한성대학교 컴퓨터공학부 교수
논문접수 : 2003년 3월 5일, 심사완료 : 2003년 6월 9일

하는데 실체뷰의 사용가능 여부를 판단하기 위해서 주어진 질의와 실체뷰간의 문법적 관계만을 고려하였다. 따라서 문법적으로는 사용가능 여부를 판단할 수 없지만 실제로는 사용 가능한 많은 실체뷰들이 고려대상에서 제외되었다. 둘째는 실체뷰가 질의에 대해서 포함 대응(containment mapping)의 관계가 성립할 경우만을 고려하였다는 점이다. 따라서 실체뷰가 질의에 언급되지 않은 테이블을 참조하거나 질의 수행에 필요한 속성을 포함하지 않는다면 그 실체뷰는 질의 재구성에 이용될 수 있음에도 불구하고 고려 대상에서 제외되어 왔다. 이러한 제약들은 실체뷰의 사용 가능 여부의 판단을 단순화시킨다. 그러나 이와 같은 제약들이 만족되지 않더라도 여전히 질의 처리에 이용 가능한 많은 실체뷰들이 존재한다.

본 논문에서는 질의 처리과정에 활용될 수 있는 유용한 실체뷰들을 탐색하는 확장된 방법을 제안한다. 이 기법은 기존의 방법에서는 제외되었던 유용한 실체뷰들을 탐색하여 활용할 수 있다. 본 논문이 제안하는 방법을 요약하면 다음과 같다.

- 유용한 실체뷰의 탐색과정에서 의미적 정보를 활용
  - 모든 데이타베이스의 인스턴스(instance)들이 의미적 정보들을 만족한다는 가정하에 이러한 의미적 정보들은 질의와 실체뷰의 문법적 관계에서는 탐색이 불가능했던 유용한 관계의 도출을 가능하게 한다.
- 질의에 언급되지 않은 릴레이션이 포함되어 있더라도 질의 처리에 이용 가능한 실체뷰들을 탐색
  - 실체뷰가 질의에 정의되지 않은 릴레이션을 갖는다 하더라도 질의 처리 과정에 필요한 정보를 갖고 있다면 그 실체뷰는 질의 처리 과정에 이용될 수 있다.

본 논문에서는 실체뷰가 주어진 질의의 처리 과정에 활용될 수 있는 지를 검증하는 조건들을 실체뷰의 유용성(usability) 정도에 따라 단계적으로 제시한다. 우선 의미적 정보를 활용하여 사용 가능한 실체뷰를 탐색하는 방법을 제시한다. 그 다음으로 질의에 언급되지 않은 테이블이 포함되어 있는 실체뷰까지 탐색 가능하도록 위의 조건을 확장한다. 이 조건들을 기반으로 선형시간(polynomial time) 내에 질의 재구성에 사용 가능한 실체뷰인가를 검증하는 알고리즘을 제시한다. 또한 실험을 통하여 각 알고리즘의 성능을 관찰하여 제안된 방법의 유용성을 증명한다. 질의와 실체뷰를 표현하기 위해서 본 논문에서는 질의 모델로서 관계 대수(relational algebra)를 사용하고 질의와 실체뷰를 비교하기 위해서 관계 대수의 동치 특성(equivalence properties)들을 활용한다.

본 논문의 구성은 다음과 같다. 2장에서는 질의 재구성 예제와 질의 모델 및 기호들을 정의한다. 3장에서는 질의 재구성에 사용 가능한 실체뷰를 탐색하는 조건을 단계적으로 제시한다. 4장에서는 실체뷰의 사용 가능성을 검증하는 알고리즘과 질의를 재구성하는 알고리즘을 각각 제시하고, 5장에서는 실험 결과를 제시한다. 마지막으로 6장에서는 결론과 향후 계획에 대해 논한다.

2. 질의모델 및 개념정의

2.1 질의 재구성 예

본 절에서는 실체뷰가 질의 처리에 이용되는 예를 제시한다. 다음의 스키마(schema)와 제약들(constraints)은 본 논문에서 제시되는 모든 예에 이용되며(주 키는 밑줄로 표시), 제약들은 모든 유효한 데이타베이스 인스턴스에 대해서도 만족된다고 가정한다.

- 데이타베이스 스키마
  - *sales*(*eid*, *item*, *vol*, *date*)
  - *emp*(*eid*, *name*, *dept*, *salary*, *age*)
  - *dept*(*dept*, *manager*, *loc*)
  - *item*(*item*, *category*, *size*, *weight*)
- 외래키 제약(foreign key constraints)
  - *sales.eid* 는 *emp*를 참조하는 외래키
  - *sales.item* 은 *item*을 참조하는 외래키
  - *emp.dept* 는 *dept*를 참조하는 외래키
- 암시 제약(implication constraint)
  - *item.item* = 'doll' → *item.category* = 'toy'

[예 2.1]

다음과 같이 SQL로 표현된 질의 Q와 실체뷰 *emp_toy_sales*를 가정하자.

Q :	Select	*E.name, E.dept*
	From	*emp E, sales S*
	Where	*E.eid* = *S.eid* and *S.item* = 'doll'
emp_toy_sales :	Select	*E.eid, E.name, E.dept, I.item*
	From	*emp E, sales S, item I*
	Where	*E.eid* = *S.eid* and
		S.item = *I.item* and
		I.category = 'toy'

우선 위에서 주어진 암시제약에 의해 *emp_toy_sales*에서 *I.item*이 'doll'인 모든 튜플들은 *I.category*가 'toy'이다. 따라서 질의 실행과정 중에 조건식 *I.category* = 'toy'에 의해서 *I.item*이 'doll'인 어느 튜플들도 삭제되지 않는다. 또한 릴레이션 *sales*는 릴레이션 *item*과 외래키에 대해서 조인이 이루어지므로, 릴레이션 *item*은 조인 연산으로 인해 릴레이션 *sales*의 어떤 튜플(tuple)들도 중복시키거나 제거하지 않는다. 따라서 *emp_toy_sales*는 질의 Q에서 *S.item* = 'doll'을 만족하는 모든 *Sales*의 튜플들을 유일하게 포함하게 된다. 결국 *emp_toy_sales*로부터 질의 Q의 결과를 얻어낼 수 있으

며 다음과 같은 질의 Q'로 질의 Q와 동일한 결과를 얻는다.

Q' : Select *ET.name, ET.dept*
From *emp_toy_sales ET*
Where *ET.item* = 'doll'

Q와 Q'중 어느 것이 더 좋은 성능을 보일지는 현재의 상태에서는 알 수 없지만, Q'는 조인 연산이 필요 없으므로 Q를 처리하는 것이 훨씬 효율적일 것으로 예상할 수 있다. □

2.2 질의 모델 및 기호 정의

본 논문에서는 문제를 단순화하기 위해서 질의와 실체뷰는 합접 질의(conjunctive query) 형태로 구성된다고 가정하며, 합접 질의는 다음과 같은 관계 대수 연산자들로 표현한다.

- p [P] r : 릴레이션 r을 속성집합 P로 프로젝션(projection)
- s [S] r : 조건식 집합 S를 만족하는 릴레이션 r의 모든 튜플들을 선택(select)
- c [R] : 집합 R에 포함된 모든 릴레이션들에 대해 카티션 프로덕트(Cartesian product)

예를 들어 예 2.1의 질의 Q는 다음과 같이 표현할 수 있다.

p [*emp.name, emp.dept*] s [*emp.eid* = *sales.eid, sales.item* = 'doll'] c [*emp, sales*]

그리고 다음과 같은 함수들을 정의한다.

- *proj*(Q) : 질의 Q에 대해서 프로젝션되는 속성의 집합
- *cond*(Q) : 질의 Q에 대한 조건식 집합
- *rel*(Q) : Q에서 참조되는 릴레이션의 집합
- *subproj*(Q, R) : *proj*(Q)중에서 릴레이션 집합 R에서 정의된 속성
- *subcond*(Q, R) : *cond*(Q)중에서 릴레이션 집합 R에서 정의된 속성으로 구성된 조건식의 집합
- *attr*(R) : 릴레이션 집합 R에서 정의된 속성 집합
- $t(r)$: 튜플 t가 질의 Q의 결과에 속한 튜플일 때 t를 구성하는데 필요한 $r \in rel(Q)$인 릴레이션 r의 튜플

또한 실체뷰의 속성 이름은 그 속성이 본래 속한 기본 릴레이션에서 정의된 속성 이름을 그대로 따른다고 가정한다. 다음으로 의미적 질의 재구성을 위한 의미적 암시(semantic implication)에 대해 정의한다. 조건식 집합 C가 또 다른 조건식 집합 C'을 논리적으로 암시(logically imply)한다면 C를 만족하는 모든 튜플들은 C'를 당연히 만족한다. 예를 들어 C = {*sales.item* = *item.item*, *sales.item* = 'doll'}이고 C' = {*item.item* = 'doll'}이라면 C는 C'를 논리적으로 암시한다. 또 다른 경우로 C = {*sales.item* = *item.item*, *sales.item* = 'doll'}이고 C' = {*sales.item* = *item.item*, *item.category* = 'toy'}이라면 C는 C'을 논리적으로 암시하지 못하므로 C를 만족하는 모든 튜플들이 C'를 만족하는지 알 수 없다. 그러나 *item.item* = 'doll' → *item.category* = 'toy'이라는 암시 제약이 모든 유효한 데이타베이스 인스턴스에 대해서 만족된다면 C를 만족하는 모든 튜플들은 C'를 만족하게 된다. 의미적 제약들은 이와 같이 질의와 실체뷰 간에 문법적으로 찾을 수 없는 중요한 관계들을 도출할 수 있는 기반을 제공해준다. 본 논문에서는 이와 같은 특징을 다음과 같이 의미적 암시라는 용어로 정의한다.

[정의 2.1] 의미적 암시(semantic implication)

C와 C'를 조건식 집합이라 하고, S를 모든 데이타베이스 인스턴스에 대해서도 만족되는 의미적 제약들이라고 가정하자. 이때 $C \cup S$가 C'를 논리적으로 암시한다면 C는 C'를 의미적으로 암시한다고 정의한다. □

위의 정의와 마찬가지로 C가 C'를 의미적으로 암시하고 그 역도 만족된다면 C와 C'는 의미적 동치(*semantically equivalent*)라고 한다. 이 개념은 질의간의 동치에 대한 정의로 확장할 수 있다. 즉, 두 개의 질의 Q와 Q'에 대해서, 이들의 결과가 주어진 의미적 제약들을 만족하는 모든 데이타베이스 인스턴스에 대해서도 동일하다면 이 두 질의는 의미적 동치라고 한다. 의미적 제약들은 종류와 표현 방법들이 다양하나 본 논문에서는 이러한 가능한 의미적 제약 중에서 범위 제약(domain constraints)이나 암시 제약(implication constraints)과 같이 무결성 제약(integrity constraints)으로 표현 가능한 간단한 의미적 제약들만을 가정한다[14].

2.3 튜플보존의 개념

본 절에서는 질의 처리에 실체뷰를 사용할 수 있는가를 판단하는 조건에 사용되는 기본 개념인 튜플보존(tuple preservation)에 대해서 소개한다.

[예 2.2]

다음의 질의를 고려하자.

E_1 : p [P] s [S] c [J] 단, P = {*sales.item*},
S = {*sales.item* = *item.item*},
J = {*sales*, *item*}.

이 질의에서 릴레이션 *sales*와 *item*의 조인은 *sales*가 *item*을 참조하는 외래키에 의해 이루어지므로 *sales*의 각 튜플은 *item*과의 조인 결과로 중복되거나 삭제되지 않는다. 따라서 모든 *sales*의 각 튜플들은 질의 결과에 유일하게 포함되게 된다. 다시 말해서, 위의 질의는 다음의 질의와 의미적으로 동치이다.

E_2 : p [P] c [K] where K = {*sales*}. □

[예 2.2]의 질의 E_1의 결과는 E_2을 실행하는데 필요한 튜

플들을 포함한다. 따라서 이 E_1이 실체뷰에 대한 정의라면 E_1으로 부터 E_2의 결과를 얻어낼 수 있다. 이와 같이 실체뷰가 비록 주어진 질의에서 정의되지 않는 테이블과 조인이 되었다 하더라고(위의 예에서 실체뷰 E_1에서 정의된 테이블 *item*은 질의 E_2에는 포함되어 있지 않다) 그 조인이 외래키에 의한 것이라면 그 실체뷰를 질의의 실행과정에 활용할 수 있다. 그 이유는 외래키에 의한 조인은 조인의 결과로 튜플들을 중복시키거나 삭제시키지 않기 때문이다. 따라서 본 논문에서는 조인연산을 실행해도 본래 테이블의 튜플들이 유일하게 남아 있다는 의미로 이와 같은 개념을 튜플보존(*tuple preservation*)이라는 용어로 정의한다. 또한 이 정의를 이용한 실체뷰의 재구성 조건을 3장에서 제시한다.

[정의 2.2] 튜플보존(Tuple Preservation)

주어진 질의 Q에 대해서 R을 $rel(Q)$의 부분집합이라 하고 C를 R에 대한 조건식 집합이라고 하자. 이때 만약 다음의 표현식 E_1과 E_2가 의미적 동치이면, 질의 Q는 조건식 C에 대해 R을 튜플보존한다고 한다.

E_1 : p [*subproj*(Q, R)] s [C] c [R]

E_2 : p [*subproj*(Q, R)] s [*cond*$(Q) \cup C$] c [*rel*(Q)] □

E_2는 질의 Q의 결과로부터 조건식 C를 만족하는 튜플을 선택하고 R에 대한 속성만을 프로젝션한 결과이다. 따라서 위의 정의는 결국 C를 만족하는 R의 모든 튜플들은 질의 Q의 선택이나 조인 연산자에 의해서 삭제되거나 중복되지 않음을 의미한다. 따라서 Q가 실체뷰에 대한 정의라면 E_1의 결과에 존재하는 모든 튜플들은 Q의 결과에 포함된다는 것을 알 수 있다. 다음의 정리는 튜플보존을 위한 조건을 나타낸다.

[정리 2.1] 튜플보존을 위한 조건

[정의 2.2]와 같이 R을 $rel(Q)$의 부분집합이라 하고 C를 R에 대한 조건식의 집합이라 하자. 이때 다음 조건을 만족하는 $s \in rel(Q) - R$인 모든 s에 대해서 s부터 $r \in R$까지 조인 경로(join path)가 유일하게 존재(one and only one)한다면, 질의 Q는 조건식 C에 대해 R을 튜플보존한다.

조건 : $r = s_0, s_1, \cdots, s_n = s$일때 $\{s_1, \cdots, s_{n-1}\} \subset (rel(Q) - R)$이고 각 $s_i (1 \le i \le n)$는 다음의 두 조건을 만족한다.

1. s_i와 s_{i-1}의 조인은 s_{i-1}이 s_i의 키를 참조하는 외래키로 조인되는 동등조인(equijoin)이다.
2. $C \cup \{s_0$부터 s_i까지의 조인 조건식들$\}$이 *subcond* $(Q, R \cup \{s_1, \cdots, s_i\})$를 의미적으로 암시한다.

[증 명]

이 정리는 위의 조건들이 만족되었을 때 [정의 2.2]의 E_1과 E_2가 의미적 동치임을 보이면 된다. 따라서 $t \in E_1$이면 $t \in E_2$임을 먼저 보이고 다음으로 이의 역을 증명한다.

- Case 1 : $t \in E_1 \Rightarrow t \in E_2$

R과 S가 다음과 같이 주어졌다고 가정하자.

$$R = \{r_1, \cdots, r_n\}, \quad S = R - rel(Q) = \{s_1, \cdots, s_m\}$$

그러면 튜플 t를 생성한 R의 각 릴레이션 $r_1, \cdots, r_n$의 튜플들 $t(r_1), \cdots, t(r_n)$가 존재한다. 그렇다면 $t_2(r_1), \cdots, t_2(r_n)$, $t_2(s_1), \cdots, t_2(s_m)$ (단, $t(r_p) = t_2(r_p)$ $(1 \le p \le n)$)로 구성된 $t_2 \in E_2$가 존재함을 보여야한다. 위의 조건에 따라 각 $s_i (2 \le i \le m)$에 대해서 s_i로부터 R의 특정 릴레이션까지의 조인 경로가 존재함을 쉽게 알 수 있다. 따라서 s_1은 R의 특정 릴레이션과 조인되고, 각 s_i는 R의 특정 릴레이션과 조인되던지 아니면 $\{s_1, \cdots, s_{i-1}\}$ 중 하나의 릴레이션과 조인된다고 가정할 수 있다. 따라서 이 정리는 다음과 같이 귀납법(induction)으로 증명할 수 있다.

◦ 기본 단계(base step) : $S = \{s_1\}$

s_1은 $r_j \in R$과 조인되며 정리 2.1의 조건 1과 조건 2를 만족한다고 가정하자. 그러면 $t_2(s_1)$는 $t_2(r_j)$과 외래키에 의해 조인되고, $C \cup \{s_1$과 r_j의 조인 조건식$\}$은 $cond(Q)$와 동치인 $subcond(Q, R \cup s_1)$를 의미적으로 암시한다. 따라서 $t_2(r_1), \cdots, t_2(r_n)$는 C를 만족하므로 $t_2(r_1), \cdots, t_2(r_n)$, $t_2(s_1)$는 $cond(Q)$를 만족한다.

◦ 귀납 단계(induction step)

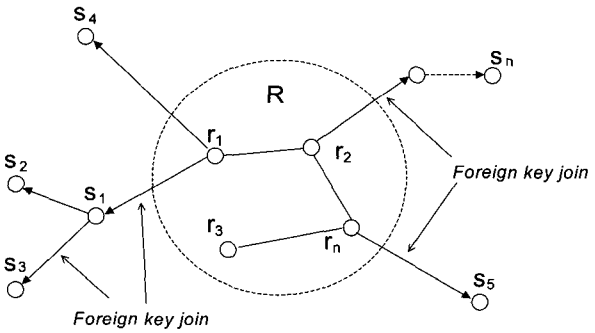
S에 존재하는 릴레이션의 수가 $k-1$일 때 위의 정리가 만족한다고 가정하자. 그러면 $C \cup subcond(Q, R \cup \{s_1, \cdots, s_{k-1}\})$를 만족하는 $t_2(r_1), \cdots, t_2(r_n), t_2(s_1), \cdots, t_2(s_{k-1})$가 존재한다. 이때 $S = \{s_1, \cdots, s_k\}$이고 $r_j = s_{k0}, s_{k1}, \cdots, s_{kq} = s_k$(단, $\{s_{k1}, \cdots, s_{kq}\} \subset \{s_1, \cdots, s_k\}$)인 r_j와 s_k에 대해서 s_k부터 $r_j \in R$까지의 조인 경로가 존재한다고 가정하자. 그렇다면 $t_2(s_{k(q-1)})$와 외래키로 조인되는 $t_2(s_{kq})$가 존재하고, $C \cup \{s_{kq}$부터 r_j까지의 조인 조건식$\}$은 $subcond(Q, R \cup \{s_{k1}, \cdots, s_{kq}\})$를 의미적으로 암시한다. 따라서 $t_2(r_1), \cdots, t_2(r_n), t_2(s_1), \cdots, t_2(s_k)$는 $C \cup subcond(Q, R \cup \{s_1, \cdots, s_{k-1}\})$를 만족하고, 또한 $subcond(Q, R \cup \{s_{k1}, \cdots, s_{kq}\})$도 역시 만족한다. 따라서 $rel(Q) = R \cup \{s_1, \cdots, s_{k-1}\} \cup \{s_{k1}, \cdots, s_{kq}\}$이므로 $t_2(r_1), \cdots, t_2(r_n), t_2(s_1), \cdots, t_2(s_k)$는 $C \cup cond(Q)$를 만족한다.

- Case 2 : $t \in E_2 \Rightarrow t \in E_1$

$t \in E_2$이고, $t(r_1), \cdots, t(r_n), t(s_1), \cdots, t(s_m)$는 t를 구성하는 튜플들이라고 가정하자. 그러면 $t(r_1), \cdots, t(r_n)$는 당연히 C를 만족한다. 따라서 $t_2(r_1), \cdots, t_2(r_n)$ (단, $t(r_p)$

$= t_2(r_p)(1 \le p \le n)$) 로부터 구성되고 t와 동일한 $t_2 \in E_1$이 존재한다. □

(그림 1)은 R과 다른 릴레이션간의 관계를 보여준다. 이 그림에서 보는 바와 같이 $rel(Q) - R$에 존재하는 각 릴레이션은 R과 유일한 조인 경로를 가지며 다른 릴레이션과는 조인 관계를 갖지 않는다. 정리의 첫째 조건은 릴레이션 R의 튜플들이 $rel(Q) - R$의 릴레이션과의 조인에 의해서 중복되거나 삭제되지 않아야 된다는 것을 의미한다. 외래키는 각 튜플들이 참조되는 릴레이션의 오직 하나의 튜플과 조인 관계가 성립하도록 제약하기 때문이다. 따라서 $s \in rel(Q) - R$로부터 $r \in R$까지의 모든 조인 경로가 외래키로만 이루어진다면 R의 모든 튜플들은 조인연산 후에도 삭제되지 않고 유일하게 남게 된다. 그러나 이러한 조인 경로 이외에 $rel(Q) - R$간의 조인 연산이 정의되어 있으면 이 연산에 의해 R의 튜플들을 삭제될 수 있으므로 R의 모든 튜플들이 Q의 결과에 남게 된다는 것을 보장하지 못한다. 더구나 C를 만족하는 R의 튜플들은 $cond(Q)$에 존재하는 선택 연산자에 의해 삭제될 수도 있다. 정리의 두 번째 조건은 $subcond(Q, R)$와 $cond(Q) - R$에 존재하는 선택 연산들이 C를 만족하는 튜플들을 삭제하지 않는다는 것을 보장하기 위해 필요한 조건이다.



(그림 1) R과 $rel(Q) - R$의 관계

3. 질의 재구성을 위한 실체뷰의 조건

본 절에서는 질의를 처리하는 과정에서 실체뷰를 활용할 수 있는가의 여부를 판단하는 조건을 제시하고, 이 조건을 만족할 경우 실체뷰를 이용한 재구성된 질의의 형태를 제시한다. 우선 질의와 실체뷰간의 문법적 관계로만 판단되었던 기존의 조건을 제시하고 이를 기반으로 본 논문에서 확장한 방법 즉, 의미적 정보의 활용과 튜플보존 개념을 이용한 질의 재구성 조건 및 재구성된 질의의 형태를 단계적으로 제시한다.

3.1 문법적 관계만을 고려한 기존의 방법

실체뷰가 질의 처리과정에서 활용가능한가를 판단하는 가장 간단한 방법은 문법적 관계만을 고려하는 것이다. 이 방법은 기존의 연구에서 많이 다루어졌지만 본 논문에서 제안하는 기법이 이 방법을 기반으로 확장되므로 본 절에서 간단히 살펴본다.

[정리 3.1]

Q와 v가 각각 주어진 질의와 실체뷰일 때 다음의 조건들이 만족된다고 가정하자.

C1.1. $rel(v) \subset rel(Q)$
C1.2. $subproj(Q, rel(v)) \subset proj(v)$
C1.3. $cond(v) \subset cond(Q)$
C1.4. $cond(Q) - cond(v)$의 모든 속성이 $(proj(v) \cup attr(rel(Q) - rel(v)))$에도 존재한다.

그러면 다음의 질의 E_1은 Q와 동치이다.

$$E_1 : \mathrm{p}\,[proj(Q)]\ \mathrm{s}\,[cond(Q) - cond(v)]\ \mathrm{c}\,[rel(Q) - rel(v), v]$$ □

위의 정리에서 C1.1은 v에 정의된 릴레이션으로부터 Q에 정의된 릴레이션으로 일대일 대응되어야한다는 것을 명시하고 있다. C1.2와 C1.4는 v가 Q의 조건식과 질의 결과에 필요한 속성들을 모두 포함하고 있어야한다는 조건을 나타낸다. 마지막으로 C1.3은 v의 조건식들이 Q의 조건식의 부분 집합이 되어야한다는 것을 나타낸다. E_1은 v를 이용하여 재구성된 질의를 나타낸다. 여기서 $rel(v)$에 존재하는 모든 릴레이션은 v를 생성하는 과정에서 이미 조인되었으므로 Q의 $rel(v)$는 단순히 v로 대치되었고, $cond(v)$도 v에서 이미 평가되었으므로 $cond(E_1)$은 $cond(Q) - cond(v)$로만 구성된다. 또한 $proj(Q)$의 모든 속성들은 $proj(E_1)$에도 역시 존재하게 된다. 결과적으로 본래의 질의 Q를 대신하여 실체뷰 v를 이용하여 구성한 E_1을 실행할 수 있다. 여기서 Q와 E_1 중에서 어느 질의가 더 효율적인지 판단하려면 인덱스의 생성 여부나 여러 가지 통계정보를 이용하여 판단하여야 한다. 그러나 직관적으로 살펴보더라도 일반적인 경우에 v는 Q의 중간 결과를 생성하기 위한 중간 결과를 이미 포함한 것이므로 Q를 실행하는 것보다 E_1을 실행하는 것이 더욱 효율적일 것이라는 것을 예상할 수 있다.

본 논문에서는 [정리 3.1]과 다음절의 [정리 3.2]의 조건들은 3.3절에서 제시하게 될 [정리 3.3]의 특별한 경우이므로 이에 대한 증명들은 [정리 3.3]에 대한 증명으로 대치한다.

3.2 의미적 제약의 활용

[정리 3.1]의 C1.3은 $cond(Q)$와 동치인 여러 가지 조건식들의 변화들을 고려하지 않고 단순히 부분집합의 개념만을 활용하므로 사용 가능한 실체뷰인지를 판단하는데 매우 제한적이다. 예를 들어 $cond(Q)$ = {*sales.item* = *item.item*, *item.item* = 'doll'}이고 $Cond(v)$ = {*sales.item* = 'doll'}일 때 $cond(v)$는 $cond(Q)$의 부분집합이 아니다. 그러나 $cond(Q)$는 {*sales.*

item = *item.item*, *sales.item* = 'doll'}과 동치이고, 따라서 *cond*(v)를 포함하게 된다. 더구나 데이타베이스의 의미적 제약들을 활용한다면 3.1절의 문법적 관계에서는 찾지 못했던 여러 가지 유용한 사실들을 도출해 낼 수가 있다. 의미적 제약은 릴레이션의 범위(domain)나 데이타베이스 인스턴스에 대한 여러 가지 무결성 제약 등을 포함한다. 이러한 의미적 제약들은 주로 기존에 많이 연구되었던 의미적 질이 최적화(semantic query optimization)에서 주로 활용되었지만[14], 이 분야에서 연구되었던 기술이나 알고리즘들을 실체뷰를 이용한 질의 재구성에 활용할 수 있다. 예를 들어 두개의 조건식 {*sales.item* = *item.item*, *item.item* = 'doll'}과 {*sales.item* = *item.item*, *item.item* = 'doll', *item.category* = 'toy'}는 문법적으로는 동치가 아니나 암시제약 *item.item* = 'doll' → *item.category* = 'toy'가 만족된다는 가정 하에서는 의미적으로 동치가 된다.

[정리 3.2]

Q와 v가 각각 주어진 질의와 실체뷰일 때 다음의 조건들이 만족된다고 가정하자.

C2.1 $rel(v) \subset rel(Q)$

C2.2 $subproj(Q, rel(v)) \subset proj(v)$

C2.3 $C \cup cond(v)$가 $cond(Q)$와 의미적으로 동치이고 C의 모든 속성이 $(proj(v) \cup attr(rel(Q) - rel(v)))$에 존재하는 조건식 C가 존재한다.

그러면 다음의 질의 E_2는 Q와 의미적 동치이다.

$$E_2 : \mathrm{p}\,[proj(Q)]\mathrm{s}\,[C]\mathrm{c}\,[rel(Q) - rel(v), v] \qquad \square$$

E_2는 v를 이용한 재구성된 질의를 나타낸다. C2.1과 C2.2는 C1.1, C1.2와 각각 동일하다. C2.3은 의미적 동치 개념을 활용함으로서 C1.3과 C1.4보다 완화된 조건으로 볼 수 있다. 즉, C를 $cond(Q) - cond(v)$로 제한한다면 C2.1~2.3은 C1.1~C1.4와 동일한 조건이 된다. 결국 C는 Q와 동일한 결과를 얻기 위해서 실체뷰를 이용한 재구성된 질의에서 실행되어야할 조건식의 집합을 의미한다. 따라서 C는 $rel(Q) - rel(v)$에서의 조건들과 v에서는 아직 평가되지 않았던 $rel(v)$에 대한 조건식들로 구성되게 된다.

3.3 튜플보존 개념의 활용

지금까지는 질의 처리과정에서 활용 가능한 실체뷰를 찾을 때, 실체뷰에 정의된 각 릴레이션이 질의에 정의된 릴레이션에 일대일 대응되는 것만을 고려하였다. 그러나 실체뷰 v와 질의 Q에 대해서, $rel(v)$가 $rel(Q)$에 존재하지 않는 릴레이션을 포함한다 하더라도 여전히 Q를 실행하는 과정에서 v를 사용할 수 있는 가능성이 존재하게 된다. 예를 들어, $rel(v) = \{a, b\}$이고 $rel(Q) = \{a\}$라면 릴레이션 b는 $rel(Q)$에 포함되지 않으므로 v는 [정리 3.2]의 조건들을 만족하지 않는다. 그러나 v가 Q를 실행하는 과정에서 필요한 튜플들을 모두 갖고 있다면 v는 여전히 질의 실행과정에 사용될 수 있다. 이러한 조건을 만족하기 위해서는 a와 b의 조인연산이 a의 튜플들을 삭제하거나 중복시키지 말아야하며, a와 b의 선택 연산이 Q를 실행하는데 필요한 튜플들을 삭제하지 말아야한다. 본 논문에서는 이러한 개념을 2.3절에서 튜플보존이란 개념으로 정의하였다. 즉, v는 $cond(Q)$에 대해서 $\{a\}$를 튜플보존해야 한다.

[정리 3.3]

Q와 v가 각각 주어진 질의와 실체뷰일 때 다음의 조건들이 만족된다고 가정하자.

C3.1. $R = rel(v) \cap rel(Q)$는 공집합이 아니다.

C3.2. v는 $cond(Q)$에 대해서 R을 튜플보존한다.

C3.3. $subproj(Q, R) \subset subproj(v, R)$

C3.4. $C \cup subcond(v, R)$가 $cond(Q)$와 의미적 동치이고 C의 각 속성들이 $(subproj(v, R) \cup attr(rel(Q) - R))$에 존재하는 조건식 C가 존재한다.

그러면 다음의 질의 E_3은 Q와 의미적 동치이다.

$$E_3 : \mathrm{p}\,[proj(Q)]\mathrm{s}\,[C]\mathrm{c}\,[rel(Q) - R, v]$$

[증 명]

증명은 튜플 t에 대해서 $t \in Q$이면 $t \in E_3$임을 증명하고 그의 역도 성립함을 보이면 된다.

- Case 1 : $t \in Q \Rightarrow t \in E_3$

$R = \{r_1, \cdots, r_n\}$이고 $rel(Q) - R = \{s_1, \cdots, s_m\}$이라고 가정하자. 이때 $t \in Q$이면, t를 구성하는 튜플들 $t(r_1), \cdots, t(r_n), t(s_1), \cdots, t(s_m)$이 존재한다. 그러면 C3.4에 의해 $t(r_1), \cdots, t(r_n)$은 $subcond(v, R)$를 만족하고, C3.2에 의해 $t(r_1), \cdots, t(r_n)$로부터 생성되는 $t_v \in v$가 존재한다. 따라서 C3.4에 의해 $t(r_1), \cdots, t(r_n), t(s_1), \cdots, t(s_m)$는 C를 만족하므로, $t_v, t(s_1), \cdots, t(s_m)$는 C를 만족한다. 결과적으로 $t_v, t(s_1), \cdots, t(s_m)$로부터 구성되는 $t' \in E_3$가 존재하고 이 튜플은 t와 같은 튜플들로부터 생성되었으므로 t와 일치한다.

- Case 2 : $t \in E_3 \Rightarrow t \in Q$

$t \in E_3$가 $t(v), t(s_1), \cdots, t(s_m)$로부터 생성되었다고 가정하면 이 튜플들은 C를 만족한다. 그리고 $t(v)$가 $t(r_1), \cdots, t(r_n)$로부터 생성되었다면 $v.attr$과 같은 v의 각 속성들은 $r_i.attr$로 표현할 수 있다($attr$은 $r_i \in R$의 속성이라고 가정). 따라서 $t(r_1), \cdots, t(r_n), t(s_1), \cdots, t(s_m)$는 C를 만족한다. 또한 $t(r_1), \cdots, t(r_n)$는 $subcond(v, R)$를 만족하므로 $t(r_1), \cdots, t(r_n)$는 $subcond(v, R)$를 만족한다는 사실을 쉽게 알 수 있다. 따라서 C3.4에 의해서 $cond(Q)$는 $C \cup subcond(v, R)$와 의미적으로 동치이므로 $t(r_1), \cdots, t(r_n), t(s_1), \cdots,$

$t(s_m)$는 $cond(Q)$를 만족한다. 결과적으로 $t(r_1),\cdots,t(r_n)$, $t(r_1),\cdots,t(r_n)$로부터 생성되는 튜플 t'가 존재하고 이 튜플은 t와 같은 튜플들로부터 생성되었으므로 t와 일치한다. □

C3.1은 $rel(v)$와 $rel(Q)$간에 공통된 릴레이션이 존재하기만 하면 되므로 C2.1에 비해서 완화된 조건이다. 만약 R이 공집합이라면 v는 Q와 관련된 어떠한 정보도 포함하고 있지 않으므로 v는 Q의 실행과정에 이용될 수 없다. C3.2는 $rel(v)-R$에 대한 조건식이 Q에서 필요한 R의 튜플들을 삭제하지 말아야한다는 조건을 나타낸다. 다시 말해서, $rel(v)-R$은 $subcond(Q, R)$을 만족하는 R의 튜플들에 대해서 어떠한 영향을 미쳐서도 안 된다. C3.3과 C3.4는 v에서 R에 관련된 부분만을 고려하도록 C2.2와 C2.3을 약간 수정한 조건이다. 따라서 C2.1~C2.3은 C3.1~C3.4의 특별한 경우로 볼 수 있다. E_3에서 R에 대한 조인과 선택 연산은 v에서 이미 수행되었으므로, R의 릴레이션들은 v로 대치되었다. 따라서 E_3에서는 Q와 동치인 결과를 얻기 위해, Q에서 정의된 그 이외의 조인과 선택 연산만을 필요로 하게 된다.

[예 3.1]

다음의 질의와 실체뷰를 고려하자.

Q : p [*sales.item, sales.vol*] s [*sales.item* = 'doll'] c [*sales*]

v : p [*sales.item, sales.vol*] s [*sales.item* = *item.item*, *item.category* = 'toy'] c [*sales, item*]

여기서 $R=\{sales\}$이고 v는 $\{sales.item = \text{'doll'}\}$에 대해서 R을 튜플보존한다. 또한 $subproj(Q, R) = subproj(v, R) = \{sales.item, sales.vol\}$이므로 C3.3이 만족된다. 마지막으로 C를 $\{sales.item = \text{'doll'}\}$로 정의하면 C3.4도 역시 만족된다. 따라서 v는 Q의 실행 과정에 사용될 수 있으며 v를 이용하여 재구성된 질의 Q'는 다음과 같다.

Q' : p [*v.item, v.vol*] s [*v.item* = 'doll'] c [*v*] □

4. 질의 재구성 알고리즘

본 장에서는 실체뷰를 이용하여 질의를 재구성하는 알고리즘을 제시한다. 알고리즘은 [정리 3.3]의 조건 C3.1~조건 C3.4를 검증하는 부분과 재구성된 질의 E_3를 생성하는 부분으로 구성된다.

4.1 알고리즘

실체뷰를 이용하여 질의를 재구성하기 위해서는 우선 실체뷰의 사용가능성 여부를 판단해야한다. 사용 가능성은 실체뷰가 [정리 3.3]의 C3.1~C3.4를 만족하는 가를 검증하면 된다. 특히 C3.2는 튜플보존을 검증하는 과정을 필요로 한다. 따라서 실체뷰의 사용가능성 여부를 판단하는 알고리즘을 제시하기 전에 튜플보존을 검증하는 함수를 먼저 제시한다.

튜플보존을 검증하는 함수는 (그림 2)에 제시되어있다. 이 알고리즘은 실체뷰 v가 조건식 C에 대해서 R을 튜플보존하는 가를 검증한다. 알고리즘 첫 부분에 *Rset*과 *Jset*은 각각 R과 $subcond(v, R)$에 존재하는 조인 조건들로 구성된다. 그런 후에 ⑥에서 제시된 조건이 만족되면, ②의 각 while 루프마다 *Rset*에 존재하는 릴레이션과 조인되는 $rel(v)$ - *Rset*의 릴레이션이 *Rset*에 추가되고, 그 때의 조인 조건이 *Jset*에 추가된다. 마지막으로 v의 모든 릴레이션과 조인 조건들이 각각 *Rset*과 *Jset*에 추가된다면 이 함수는 TRUE를 반환하고 그렇지 않으면 FALSE를 반환한다.

```
Preserve(v, R, C)
 {
   ① Rset = R, Jset = join conditions in subcond(v, R)
   ② While (TRUE)
   ③     If Rset = rel(v) and Jset = join conditions in cond(v)
                 return TRUE
   ④   Among cond(v) - Jset, pick a join condition c of r_i
       and r_j where r_i ∈ Rset and r_j ∈ rel(v) - Rset
   ⑤   If such a condition does not exist
                 return FALSE
   ⑥   If r_i is joined with r_j on a foreign key and subcond(v,
       Rset) ∪ C ∪ {c} semantically implies subcond(v, r_j).
                 Rset = Rset ∪ {r_j}
                 Jset = Jset ∪ {c}
       else      return FALSE
 }
```

(그림 2) 튜플보존 여부를 검증하는 알고리즘

(그림 2)의 알고리즘을 이용하면 [정리 3.3]에 따라 질의 Q를 처리하는 과정에 실체뷰 v를 이용할 수 있는 가를 검증하는 알고리즘을 비교적 쉽게 구성할 수 있다. (그림 3)에 보여지는 알고리즘은 C3.1~C3.4가 만족된다면 TRUE를 반환하고 그렇지 않으면 FALSE를 반환한다. 이 알고리즘에서 ①과 ②는 C3.1을 검증하는 부분이고, ③은 (그림 2)의 함수 *Preserve*를 이용하여 C3.2를 검증하는 부분이다. ④에서 C는 [14]에서 정의된 질의 그래프(query graph)를 이용하여 생성될 수 있다. 즉, C는 $subcond(v, R)$와 동치인 질의 그래프의 일부를 삭제하고 남게되는 조건들로 구성하면 된다. 따라서 $cond(Q)$가 $subcond(v, R)$를 의미적으로 암시하면, $C \cup subcond(v, R)$가 $cond(Q)$와 의미적으로 동치가 되는 C를 항상 생성할 수 있다.

```
Is_Usable(Q, v)
 {
        /* Testing C3.1 */
   ①  R = rel(Q) ∩ rel(v)
   ②  If R is empty,  return FALSE
        /* Testing C3.2 */
   ③  If Preserve(v, R, subcond(Q, R)) = FALSE)
                 return FALSE
        /* Testing C3.4 */
```

```
        /* Testing C3.4 */
    ④ If there does not exist a set of conditions C such
       that C ∪ subcond(v, R) is semantically equivalent to
       cond(Q)
                return FALSE
    ⑤ if there exists an attribute referenced in C but not in
       (subproj(v, R) ∪ attr(rel(Q) - R))
                return FALSE
       /* Testing C3.3 */
    ⑥ if (subproj(Q, R) - subproj(v, R)) is not empty
                return FALSE
       return TRUE
}
```

(그림 3) 실체뷰의 사용가능성을 검증하는 알고리즘

```
RewriteQuery(Q, v)
{
    ① If Is_Usable(Q, v)
    ②    rel(Q') = (rel(Q) - R) ∪ v
    ③    proj(Q') = proj(Q)
    ④    cond(Q') = C
}
```

(그림 4) 질의 재작성 알고리즘

지금까지의 알고리즘들을 기반으로 실체뷰를 이용하여 질의를 재구성하는 알고리즘은 (그림 4)와 같이 비교적 간단하게 구성할 수 있다. 주어진 질의 Q와 v에 대해서, 함수 *Is_Usable*을 이용하여 질의 Q를 실행하는데 실체뷰 v가 이용가능한 가를 검증한다. 만약 이용이 가능하다면 재구성된 질의 Q'의 각 구성 요소인 $rel(Q')$, $proj(Q')$, $cond(Q')$를 구성한다.

4.2 알고리즘의 시간 복잡도(time complexity)

알고리즘 *RewriteQuery*의 시간 비용은 알고리즘 *Is_Usable*의 시간 비용에 따라 결정된다. *Is_Usable* 또한 알고리즘 *Preserve*의 시간 비용과 조건식 집합 C를 결정하는데 필요한 비용에 의해 결정된다. *Preserve*에서 의미적 암시는 Q의 각 조인 조건에 대해서 검증해야한다. 따라서 c를 데이타베이스에서 정의된 의미적 제약의 수이고 Q에 존재하는 조인 조건의 수를 상수(constant)로 가정할 때, [14]에서 정의된 질의 그래프를 이용하면 $O(c^3)$내에 의미적 암시 여부를 판단할 수 있다. 따라서 알고리즘의 *Preserve*의 시간 비용은 $O(c^3)$가 된다. 또한 C를 생성하려면 의미적 동치를 검증하는 과정을 필요로 한다. 따라서 알고리즘 *Is_Usable*의 시간 비용은 $O(c^3)$가 된다. 이러한 시간 비용은 실체뷰의 활용 가능성을 최대화시킨 [정리 3.3]의 경우에 대한 시간 비용이다. 하지만 문법적 관계만을 고려한 [정리 3.1]의 조건에 대해서는 릴레이션(C1.1), 속성(C1.2와 C1.4), 조건식들(C1.3)의 집합 포함 관계만을 검증하는 작업만이 필요하며, 각각의 요소들은 질의나 실체뷰의 정의에 나타난 것이므로 상수로 간주할 수 있다. 따라서 시간 비용은 $O(1)$이다.

결론적으로 활용 가능한 실체뷰를 검증할 때 문법적 관계만을 고려한다면 알고리즘 *RewriteQuery*의 성능은 단순히 질의와 실체뷰의 정의가 얼마나 복잡한 가에 따라 결정되지만, 의미 정보나 튜플보존의 개념을 활용하여 실체뷰의 탐색 범위를 확장한다면 성능은 데이타베이스에서 정의된 의미적 제약의 수에 영향을 받게 된다. 그러나 실제 상황에서 의미적 제약의 수가 많다 하더라도 주어진 질의와 실체뷰에 관련된 의미적 제약의 수는 많지 않으며 또한 선형 시간 내에 탐색이 가능하다. 따라서 본 논문에서 제안한 실체뷰의 확장된 활용 기법으로 인한 추가적 시간비용은 실체뷰의 활용으로 인해 기대되는 실질적인 질의의 성능 향상에 비해 상당히 미미한 수준으로 볼 수 있다.

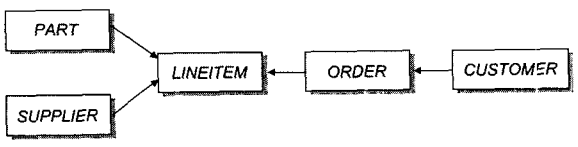
5. 성능 분석

본 장에서는 본 논문에서 제안한 질의 재구성 조건과 알고리즘의 유용성을 검증하기 위해 TPC-H 데이타베이스[17]를 이용한 실험 결과를 기술한다. 실험은 3장에서 제안된 확장된 질의 재구성 조건을 활용한 최적 질의와 기존의 조건 즉, 실체뷰의 테이블들이 질의의 테이블에 포함된 경우만을 고려한 조건을 활용한 질의에 대한 성능을 비교한다. 단, 3.2절에서 제안된 의미정보를 이용한 질의 재구성에 대한 성능은 인위적으로 의미정보를 생성하기가 어려울 뿐만 아니라 설령 생성이 가능하다 하더라도 실험 결과가 실제 환경을 대변할 수 있는 지에 대한 평가가 사실상 불가능하여 실험 대상에서 제외하였다. 따라서 실험은 3.3절에서 제안한 튜플보존 개념을 이용한 질의 재구성기법에 대한 실험만을 실시하였다. 실험을 위하여 4장에서 제시한 알고리즘들을 256MB의 주기억 장치를 장착한 SUN Ultra2 워크스테이션에서 구현하였고 데이타베이스는 Oracle 8i를 이용하였다.

5.1 실험 환경

5.1.1 데이터베이스

본래 TPC-H는 집계 질의(aggregate query)의 성능을 평가하기 위해서 정의된 것이고 본 논문은 집계 질의에 대한 가정은 하지 않았으나, 이 스키마는 집계질의 뿐만 아니라 일반 질의에 대한 질의 생성이 과정이 복잡하지 않아 실험 대상으로 선택하였다. 데이터베이스는 (그림 5)에서 보는바와 같이 PART, SUPPLIER, LINEITEM, ORDER, CUSTOMER의 5개의 테이블로 구성된다. 그림에서 화살표는 일대다의 관계를 표현한다. TPC-H 벤치마크 규정에 따라 전



(그림 5) TPC-H 데이타베이스

체 데이터베이스의 크기는 20M를 구축하였으며 각 테이블의 기본키에 인덱스를 구축하였다.

5.1.2 실체뷰와 질의

실체뷰를 구축하기 위해 각 실체뷰는 무작위로 한 개에서 다섯 개까지의 테이블이 포함되도록 정의하였고 테이블 LINEITEM은 모든 실체의 정의에 포함되도록 하였다. 실험을 단순화를 위해 선택 술어(selection predicate)는 가정하지 않았으며 조인 술어(join predicate)는 기본키와 기본키를 참조하는 속성들 사이의 등식 술어(equal predicate)만을 가정하였다. 프로젝트 속성들은 실체뷰에 포함된 각 테이블에서 10%~40%의 속성들을 무작위로 선택하였다. 모든 실체뷰는 기본 테이블들과 같이 저장하였으며 각 실체뷰의 정의에서 기본 테이블의 기본키가 프로젝트되면 그 키에 인덱스를 구축하였다. 이는 실체뷰를 이용하여 재구성된 질의와 그렇지 않은 질의에 대한 성능을 비교할 때 인덱스의 영향을 되도록 적게 받게 하기 위해서다. 질의는 실체뷰 정의와 거의 유사하나 정의에 포함되는 각 테이블의 속성들 중에서 0%~30%의 속성들을 무작위로 선택하여 프로젝트 속성에 포함하였다. 그 이유는 각 실체뷰가 질의의 재구성에 활용될 수 있는 가능성을 높이기 위해서다.

5.1.3 비용 모델

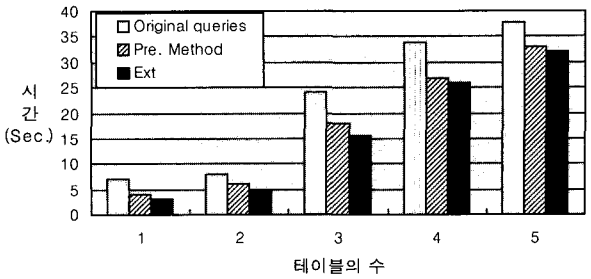
본 논문에서는 실험을 위하여 [9]에서 제안한 선형 비용 모델(linear cost model)을 약간 수정한 비용 모델을 이용하였다. 이 비용 모델은 선형 비용 모델에 테이블의 수를 고려한 요소를 추가한 형태로 질의 Q에 대한 선형 비용이 *Linear Cost*일 경우 본 실험에서 활용한 비용식 *Mod. Linear Cost*는 다음과 같다.

$$Mod.\ Linear\ Cost = Linear\ Cost \times n(rel(Q))$$

이 식에서 $n(rel(Q))$는 테이블의 집합 $rel(Q)$에 존재하는 테이블들의 수를 나타낸다.

5.2 실험 결과

실험을 위하여 20개의 실체뷰를 무작위로 정의하여 구축하였고 역시 무작위로 정의된 700여개의 질의에 대해 알고리즘을 실행하였다. (그림 6)은 질의에 포함된 테이블의 수에 따른 질의의 평균 실행 시간을 보여준다. *Pre.-method*는 문법적 관계만을 고려한 기존의 방법을 나타내며 *Ext*는 3장에서 제안한 확장된 질의 재구성 조건을 적용한 결과이다. 이 그림에서 보는 바와 같이 질의에 포함된 테이블의 수에 관계없이 *Ext*에 의해 재구성된 질의는 본래의 주어진 질의(그림에서 *Original queries*) 뿐만 아니라 *Pre.-method*에 의해 재구성된 질의에 비해 좋은 성능을 보이고 있다. (그림 6)의 결과는 무작위로 생성된 모든 질의에 대한 평균 실행 시간을 나타낸 것이다. 따라서 이 결과는 각 재구성 기법별로 재구성 가능한 질의뿐만 아니라 그렇지 못한 질의를 포함한 평균값을 의미한다.



(그림 6) 테이블 수에 따른 질의 실행시간

<표 1>은 다음의 값들을 보여주고 있다.

- *Avg. Ratio (all)* : *Original queries*에 대한 비용의 평균 비율
- *Diff. Query Ratio* : *Original queries*와 다르게 재구성된 질의의 비율
- *Avg. # of Candidates* : 최적 질의를 찾기 위해 재구성된 후보 질의의 평균 수

〈표 1〉 질의 성능의 비교

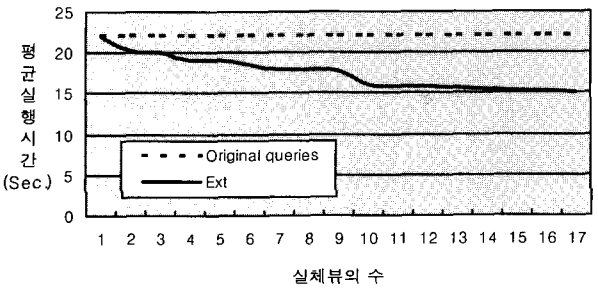
	Avg. Ratio(all)	Diff.Query Ratio	Avg. # of Candidates
Pre.-Method	77%	41%	0.8
Ext	69%	49%	1.9

*Original queries*의 성능에 대해 *Ext*에 의한 질의 비용의 평균 비율은 69%로 *Pre.-method*의 비율(77%)에 비해 좋은 성능을 보이고 있다. *Diff. Query Ratio*는 본 논문이 제시한 기법이 *Pre.-method*에 비해 *Original Queries*와 다른 많은 최적 질의를 찾는다는 사실을 보여주고 있다. 마지막으로 *Avg. # of Candidates*는 *Ext*에서 최적 질의를 찾기 위해 생성되는 후보 질의의 수가 *Pre.-method*에 생성되는 것들에 비해 크게 증가하지 않음을 보여주고 있다. 이는 본 논문이 제시한 방법이 기존의 방법에 비해 탐색 공간을 크게 증가시키지 않고도 많은 유용한 실체뷰를 탐색할 수 있음을 보여주는 것이다.

마지막으로 (그림 7)은 *Ext*에 의해 재구성된 질의에 대해 실체뷰의 수에 따른 평균 실행 시간을 보여준다. 이 그림에서 9~10개까지의 실체뷰에 대해서는 실체뷰의 수가 증가함에 따라 전체 질의의 성능도 비슷한 비율로 향상됨을 확인할 수 있다. 그러나 그 이상의 실체뷰에 대해서는 실체뷰의 수가 증가하여도 전체적인 질의의 성능은 거의 향상되지 않는다. 그 이유는 실체뷰의 수가 증가할수록 유사한 실체뷰들이 정의될 수 있기 때문이다.

결론적으로 이 실험에서는 본 논문이 제시한 확장된 질의 재구성 방법이 기존의 방법에서는 찾지 못했던 많은 유용한 실체뷰들을 탐색할 수 있음을 보여주고 있다. 비록 실제 환경에서의 질의 성능은 실체뷰의 수나 질의 패턴과 같은 여러 가지 요인에 따라 다양하게 나타날 수 있으나 대부분의 경우 본 논문에서 제시한 확장된 조건에 의한 질의 재구성

방법은 기존의 방법에 비해 좋은 성능을 보일 것으로 예상할 수 있다.



(그림 7) 실체뷰의 수에 따른 질의 성능

6. 결 론

질의와 실체뷰간의 문법적 관계만을 고려한 기존의 연구에서는 실제 질의 처리과정에 사용 가능함에도 불구하고 검증의 편의를 위해 많은 유용한 실체뷰들을 활용하지 못하는 문제가 있었다. 본 논문에서는 이러한 기존이 연구 결과를 바탕으로 되도록 많은 실체뷰들이 질의 처리과정에서 활용될 수 있는 기법을 확장 정도에 따라 단계적으로 제시하였다. 우선 문법적 관계뿐만 아니라 의미 제약과 같은 의미 정보를 활용한 방법을 제시하였으며, 그 다음으로 튜플 보존 개념을 활용하여 질의에 정의되지 않은 릴레이션을 포함하는 실체뷰의 활용 가능성도 제시하였다. 이러한 개념들을 기반으로 실체뷰의 활용 가능성을 검증하는 조건들을 찾을 수 있었고 이러한 조건들을 검증하고 질의를 재구성하는 알고리즘을 제안하였으며, 실험을 통하여 알고리즘의 효율성을 입증하였다.

본 논문에서는 주어진 질의에 대해서 하나의 실체뷰만을 활용하는 방법을 제안하였으나 향후에는 두개 이상의 실체뷰가 질의의 재구성에 활용될 수 있는 방법을 연구할 계획이다. 이 문제가 해결된다면 하나의 실체뷰만을 활용하는 방법 보다 더욱 효율적인 질의의 재구성이 가능할 것으로 기대된다.

참 고 문 헌

[1] D. Agrawal, A. E. Abbadi, A. Singh and T. Yurek, Efficient View Maintenance at Data Warehouses, In Proc. of ACM SIGMOD, pp.417-427, 1997.

[2] S. Abiteboul and O. M. Duschka, Complexity of Answering Queries Using Materialized Views, In *Proc. ACM SIGMOD-SIGACT-SIGART Symp. on Principles of Database Systems*, pp.254-263, June, 1998.

[3] J. Albrecht, W. Hummer, W. Lehner, L. Schlesinger, Query Optimization by Using Derivability in a Data Warehouse Environment, In *Proc. of DOLAP*, 2000.

[4] S. Chaudhuri, Krishnamurthy, S. Potamianos and K. Shim, Optimizing Queries with Materialized Views, In *Proc. of ICDE*, pp.190-200, 1995.

[5] Y. Cui, J. Widom, Lineage Tracing for General Data Warehouse Transformations, In *Proc. of VLDB*, Rome, Italy, September, 2001.

[6] J. Gray, A. Bosworth, A. Layman and H. Pirahesh, Data Cube : A Relational Operator Generalizing Group-By, Cross-Tap and Sub-Totals, In *Proc. of ICDE*, pp.152-159, 1996.

[7] A. Gupta, V. Harinarayan and D. Quass, Aggregate-Query Processing in Data Warehousing Environments, In *Proc. of VLDB*, pp.358-369, 1995.

[8] C. Hurtado, A. Mendelzon, A. Vaisman, Maintaining Data Cubes under Dimension Updates, in *Proc. of ICDE*, 1999.

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

[10] A. Y. Levy, A. O. Mendelzon, Y. Sagiv and D. Srivastava, Answering Queries Using Views, In *Proc. of ACM PODS*, pp.95-104, 1995.

[11] C. -S. Park, M. H. Kim and Y. -J. Lee, Rewriting OLAP Queries Using Materialized Views and Dimension Hierarchies in Data Warehouses, In *Proc. of ICDE*, 2001.

[12] K. A. Ross and K. A. Zaman, Optimizing Selections over Datacubes, in *Proc. of the 2000 SSDBM Conference*, July, 2000.

[13] 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.

[14] S. T. Shenoy and Z. M. Ozsoyoglu, Design and Implementation of a Semantic Query Optimizer, *IEEE Transactions on Knowledge and Data Engineering*, Vol.1, No.3, pp. 344-361, Sept., 1989.

[15] J. Yang and J. Widom, Making Temporal Views Self-Maintainable for Data Warehousing, In *Proc. of the 7th International Conference on Extending Database Technology*, Germany, March, 2000.

[16] Y. Zhuge, H. Garcia-Molina, J. Hammer and J. Widom, View Maintenance in a Warehouse Environment, In *Proc. of ACM SIGMOD*, pp.316-327, 1995.

[17] Transaction Processing Performance Council, TPC benchmark (tm) H (Decision Support), Revision 2.0, April, 2002.

장 재 영

e-mail : jychang@hansung.ac.kr

1992년 서울대학교 계산통계학과(학사)

1994년 서울대학교 계산통계학과 대학원 (이학석사)

1999년 서울대학교 계산통계학과 대학원 (이학박사)

2000년~현재 한성대학교 컴퓨터공학부 조교수

관심분야 : 질의처리, 데이타 웨어하우스 등