데이터 웨어하우스에서 클러스터링 기법을 이용한 실체화 뷰 선택 알고리즘

양 진 혁† · 정 인 정††

요 약

데이터 웨어하우스에서 실체화 할 뷰들을 알맞게 선택하는 것은 분석적인 질의에 대한 정확하고 신속한 응답을 얻기 위해서 대단히 중요한 문제이다. 기존의 뷰 선택 알고리즘들에서는 릴레이션 전체가 실체화 뷰들로서 고려되었다. 그러나, 릴레이션의 부분 대신 전체를 실체화한다는 것은 시간과 공간 비용측면에서 좋지 못한 성능을 초래한다. 따라서, 우리는 기존 뷰 선택 알고리즘들에서의 문제점을 극복하기 위해서 개선된 실체화 뷰 선택 알고리즘을 제안한다. 제안된 알고리즘 ASVMRT(Algorithm for Selection of Views to Materialize using Reduced Table)에서는 먼저 속성-값들의 농도에 기반을 둔 자동 클러스터링을 사용하여 축약 테이블들을 데이터 웨어하우스에서 생성한 다음, 원래의 베이스 릴레이션들의 조합 대신에 축약 테이블들의 조합을 실체화 뷰들로 고려한다. 제안한 알고리즘의 타당성 검증을 위하여 우리는 실험결과에서 시간 및 공간 모두에서 기존 알고리즘들보다 약 1.8배의 성능향상이 있음을 보인다.

Materialized View Selection Algorithm using Clustering Technique in Data Warehouse

Jin-Hyuk Yang† · In-Jeong Chung††

ABSTRACT

In order to acquire the precise and fast response for an analytical query, proper selection of the views to materialize in data warehouse is very crucial. In traditional view selection algorithms, the whole relations are considered to be selected as materialized views. However, materializing the whole relations rather than a part of relations results in much worse performance in terms of time and space cost. Therefore, we present an improved algorithm for selection of views to materialize using clustering method to overcome the problem resulted from conventional view selection algorithms. In the presented algorithm, ASVMRT(Algorithm for Selection of Views to Materialize using Reduced Table), we first generate reduced tables in data warehouse using automatic clustering based on attribute-values density, then we consider the combination of reduced tables as materialized views instead of the combination of the original base relations. For the justification of the proposed algorithm, we show the experimental results in which both time and space cost are approximately 1.8 times better than the conventional algorithms.

1. 서 론

은행업무와 같은 트랜잭션(transaction)을 위주로 설계된 관계형 데이터 베이스에서는 사용자가 요구하는 분석적이고 시계열적인 질의어에 응답하는데 시간이 많이 소요된다. 따라서, 시장분석을 통하여 경영자에게 의사결정을 지원하기 위해서 기존의 OLTP(On-Line Transaction Processing) 위주의 관계형 데이터 베이스

† 준 회 원 : 고려대학교 대학원 전산학과
†† 종신회원 : 고려대학교 전산학과 교수
논문접수 : 2000년 5월 24일, 심사완료 : 2000년 8월 18일

로부터 새로운 개념인 주제 중심적(subject oriented)이고 자료가 통합적이며(integrated) 비휘발성인(non-volatile) 성질을 지니고 시간 변이적인(time variant) 특징을 갖는 데이터 웨어하우스의 구축이 활발해지고 있는 추세이다.

데이터 웨어하우스에서 뷰란 베이스 릴레이션이나 다른 뷰로부터 파생되는 릴레이션으로 참조되어질 때마다 재계산되는 가상의 릴레이션으로서, 이러한 뷰 튜플들을 요약 및 저장한 것을 실체화 뷰(materialized view)라고 한다. 실체화 뷰를 사용하는 이유는 시계열적인 데이터를 저장하고 있는 데이터 웨어하우스에서 분석적인 질의어에 대한 처리를 신속하게 처리하기 위함이다. 그러나, 실체화 뷰를 많이 사용하면 할수록 데이터 웨어하우스의 공간이 크지게 된다. 따라서, 실체화 뷰를 효율적으로 선택한다는 것은 복잡한 질의어에 대한 응답시간과 저장공간이라는 요소를 적절히 만족시켜야 한다.

관련된 뷰 선택 알고리즘을 위한 연구들로는 [1-3]이 있다. [1]에서는 단지 집계함수(aggregate function)들만을 고려했고, [2]에서는 AND, OR 및 AND-OR 그래프로써 실체화 뷰 선택을 휴리스틱(heuristic) 기반의 탐욕적(greedy) 방법으로 제안했으나 그에 대한 평가가 없었다. [3]에서는 HA_{MVD}라는 알고리즘을 제안했으나 HA_{MVD}의 입력변수인 다중 뷰 처리계획 생성과정에 있어서 너무 많은 시간이 소요되는 문제점이 있다. 따라서, 본 논문에서는 기존 알고리즘이 안고 있는 저장공간과 속도에서의 문제점을 개선시킨 알고리즘을 제안한다.

데이터 웨어하우스에서 신속한 질의응답을 위한 실체화 뷰를 선택하기 위해서 클러스터링 기법을 이용한 본 논문의 제안된 알고리즘에서는 릴레이션 차원들의 상대적인 농도를 근거로 클러스터들을 자동으로 찾아낸 후 생성된 클러스터들을 참조해서 축약 테이블(reduced table)을 만든다. 생성된 축약 테이블들은 ASVMRT (Algorithm for Selection of Views to Materialize using Reduced Table)에서 최적의 다중 뷰 처리계획(Multiple View Processing Plan : MVPP) 생성을 위해서 사용되어지는 릴레이션들이다. 우리는 생성된 축약 테이블을 이용하여 다중 뷰 처리계획을 생성하고, 제안한 알고리즘 ASVMRT를 사용하여 생성된 다중 뷰 처리계획에서 뷰들을 신속하게 효과적으로 처리하고 선택할 수 있다. 제안한 알고리즘의 타당성을 보이기 위하여 데이터 베이스 교육용으로 많이 사용되고 있는 'Pubs'에서의 실험결과와 한국전자통신연구원(ETRI : Electronics and Telecommunications Research Institute)의 정보통신 기술기준 정보시스템(http://tris.etri.re.kr)에서 사용중인 충분히 큰 크기의 데이터 베이스에서 실험한 결과를 보인다. 실험결과에서 기존 알고리즘들에서 보다 시간 및 공간에서 모두 평균 1.8배 정도의 향상을 볼 수 있다.

본 논문의 구성은 다음과 같다. 2장에서 데이터 웨어하우스와 뷰 실체화에 대한 관련연구들에 대해서 살펴보고, 3장에서 ASVMRT를 보인다. 4장에서 실험결과를 통해서 ASVMRT를 기존 알고리즘과 비교하고, 끝으로 5장에서 결론을 맺음과 동시에 향후 실체화 뷰 선택을 위한 이슈들을 제시한다.

2. 데이터 웨어하우스와 뷰 실체화 관련 연구

2장에서는 데이터 웨어하우스에 대해서 간략하게 살펴보고 데이터 웨어하우스의 효율성을 증대시키는 방법으로 사용되고 있는 실체화 뷰 및 관련 알고리즘들에 대한 기존 연구들을 살펴본다.

2.1 데이터 웨어하우스

데이터 웨어하우스는 기업의사결정을 지원하기 위한 주제 중심적이고, 통합적이며 비휘발성인 동시에 시간 변이적인 특징을 가지는 데이터 저장장소로서 정의된다[4]. OLTP 목적을 위해서 ER(Entity-Relationship) 모델에 근거를 두고 설계된 관계형 데이터 베이스에서는 통계적이며 분석적인 질의를 통한 기업경영의 의사결정 지원기능이 없다. 이러한 OLAP(On-Line Analytical Processing) 기능을 필요로 하는 사용자의 요구사항을 만족시키기 위해서 데이터 웨어하우스가 구축된다. 다음 <표 1>은 데이터 웨어하우스와 OLTP와의 차이를 나타낸 것이다[6].

<표 1> OLTP와 데이터 웨어하우스와의 차이점

	OLTP	데이터 웨어하우스
주 사용자	사무원	전문적 분석가
시스템 사용목적	기본적인 기업경영	기업경영활동의 분석
사용자와의 상호작용	정 적	유동적
기본 처리단위	트랜잭션	질 의
처리 업무의 특징	읽기/쓰기	읽 기
접근 레코드 수	수 십	수백만
사용자 수	수 천	수 백
시스템의 주안점	데이터 유입	정보추출

2.2 기존 실체화 뷰 선택 관련 알고리즘

뷰란 베이스 릴레이션이나 다른 뷰로부터 파생되는 릴레이션으로 참조되어질 때마다 재계산되는 가상의 릴레이션이다. 이러한 뷰의 튜플들을 데이터 베이스에 요약 및 저장한 것을 실체화 뷰라고 한다[5]. 실체화 뷰들에 대해서 인덱싱을 함으로써 분석적인 질의어에 응답하기 위해 뷰들을 재계산하는 것보다 훨씬 빠른 속도로 질의어를 처리할 수 있다.

2.2.1 격자(lattice) 구조 데이터 큐브에서의 실체화 뷰 선택 알고리즘

이 연구에서는 데이터 큐브(data cube)를 격자 구조로 변환 후 거기에서 실체화 할 뷰들을 선택하고 있다. 참조 문헌[1]에서 사용되고 있는 수식 중 $B(v, S)$는 뷰 v를 선택함으로서 생기는 총 이득으로서, 알고리즘은 $B(v, S)$의 총 이득이 극대화하는 방향으로 실체화 뷰들을 선택하고 있다. 그리고 모든 뷰들의 선택이 끝나면 알고리즘은 종료되고 실체화 뷰들을 반환한다.

2.2.2 AND-OR 그래프를 이용한 뷰 선택 알고리즘

AND-OR 뷰 그래프에는 하나의 질의 처리 계획을 가지는 AND 뷰 그래프와 여러 개의 질의 처리 계획을 가지는 OR 뷰 그래프가 있다[2]. AND 뷰 그래프에서는 다중질의 최적기(Multiple Query Optimizer)를 사용하여 주어진 질의들을 위한 AND 뷰 프래프에 해당하는 글로벌 계획을 생성한다. 질의계획이 생성된 후에는 질의처리계획을 구성하고 있는 노드(뷰)들을 실체화할 대상으로 삼는다. 글로벌 질의 처리계획은 여러 개의 작은 질의들로 나누어진 후 각각의 질의들이 처리되고 나면 합쳐진다.

이 알고리즘은 뷰들에 대한 갱신비용이 없고, 주어진 공간제약 S를 가지는 상태에서 실체화 뷰들의 집합 M을 선택하는 탐욕적 알고리즘(greedy algorithm)으로서, 실체화 뷰 공간 제약 $S(M)$을 넘지 않는 범위 내에서 가장 이득이 많이 생기는 뷰들을 차례대로 선택한다. 주어진 공간 제약 S를 넘으면 알고리즘은 멈추고 실체화 뷰 집합 M을 반환한다.

데이터 큐브에서 AND-OR 뷰 그래프는 OR 뷰 그래프가 된다. 왜냐하면 데이터 큐브에서는 다른 뷰들로부터 그 뷰를 만들 수 있는 방법이 여러 개 존재하기 때문이다. 데이터 큐브 환경에서 뷰들을 선택하는 문제 해결방법은 데이터 큐브 환경에서의 실체화 뷰를 선택하는 알고리즘을 제안하고 있는 참고문헌 [1]의 일반화된 형태이다.

2.2.3 다중 뷰 처리계획을 이용한 뷰 선택 알고리즘

다중 뷰 처리계획[3]은 질의들을 루트 노드로 단말 노드를 베이스 릴레이션으로 표시한 DAG(Directed Acyclic Graph)으로서 데이터 웨어하우스에서 뷰에 대한 질의어 처리계획을 나타낸다. 다중 뷰 처리계획은 다섯 개의 원소를 성분으로 가지는 $M=(V, A, C_a, C_m, f_q, f_u)$으로 구성되어 있다. 여기서 V는 노드들이고, A는 노드들 사이의 선후관계를 나타내는 방향을 가진 연결선이다. C_a, C_m은 각각 해당 노드의 질의처리비용과 유지비용이다. 그리고, f_q, f_u는 각각 해당 노드의 질의접근빈도와 갱신빈도이다.

그리고 이 연구에서는 탐색공간을 줄이기 위해서 다음과 같은 휴리스틱을 제공하고 있다. 관련된 뷰 v_1과 v_2에서 v_1이 v_2의 자식일 경우, 만약 v_1을 실체화했을 경우에 이득이 생기지 않는다면 v_2를 실체화 할 대상으로 삼지 않는다는 휴리스틱이다. 이러한 휴리스틱은 데이터 마이닝 기법 중 연관규칙탐사에서 사용되는 Apriori[11] 및 DHP[12]에서 사용되는 닫힘성(closure property)과 유사하다. 알고리즘은 모든 노드들을 포함하고 있는 집합 LV와 실체화 뷰 대상 집합 M을 입력으로 받아서 비용에 이득이 생기는 뷰들을 실체화한다. 알고리즘은 고려해야할 대상 뷰가 없을 때까지 즉, LV가 공집합이 될 때까지 수행한다. 알고리즘의 결과는 실체화 대상 뷰들의 집합인 M을 반환한다.

2.2.4 기타 관련 연구들

기타 연구들에는 데이터 큐브와 데이터 큐브에서 사용할 수 있는 연산자를 보인 연구[13], 처음으로 다중 뷰 관리 문제를 언급한 연구[14], 데이터 큐브에서 인덱스를 부과시켜 뷰 선택을 할 수 있는 알고리즘을 제안한 연구[15], 다차원 데이터 베이스에서 뷰 선택 알고리즘을 제안하고 있는 연구[16] 등이 있다.

3. ASVMRT(Algorithm for Selection of Views to Materialize using Reduced Table)

본 장에서는 기존 뷰 선택관련 알고리즘과는 달리 데이터 마이닝[7-10] 기법중의 하나인 클러스터링 기법을 이용하여 축약 테이블을 만든 후 실체화할 뷰들을 선택하는 알고리즘을 제안한다.

3.1 동기 및 예제

시계열적인 데이터의 저장소인 데이터 웨어하우스에

서 분석적인 질의어를 신속하게 처리하기 위해서 우리는 실체화 뷰를 선택한다. 그러나 이러한 실체화 뷰들을 구성하고 있는 전체 튜플들 중에서는 실제로 주어진 질의어에 응답하는 데 있어서 관계없는 튜플들도 존재한다. 이에 우리는 질의어와 관계있는 튜플들만 추출하여(클러스터링하여) 실체화 뷰로 저장한다. 제안하는 실체화 뷰 선택 알고리즘은 모든 튜플들을 저장하고 있는 기존 실체화 뷰 선택 알고리즘들보다 튜플들의 계산에서 더 빠르며 저장공간 또한 절약된다. 다음은 이러한 생각을 뒷받침할 수 있는 예제이다.

6개의 차원을 가지는 월급 릴레이션(700개의 튜플이 존재한다고 가정)과 8개의 차원을 가지는 나이 릴레이션(500개의 튜플이 있다고 가정)이 있다고 가정하자. 데이터 웨어하우스를 사용하여 기업 경영인은 아래와 같은 질의를 통하여 경향을 분석 및 예측할 수 있으며, 예측된 결과로부터 새로운 경영전략을 세울 수 있다.

질의 : 월 평균 소득 300만원 이상인 20대가 선호하는 차의 기종은?

기존의 방법에서는 700×500에 해당하는 튜플들의 조인 결과로부터 다시 선택(select)연산을 취한다. 그러나, 월급 릴레이션 및 나이 릴레이션에서 축약 테이블을 만든다면(가정하기를 월급 릴레이션에서 300만원 이상의 소득자가 350명이고, 나이 릴레이션에서 20대인 사람들이 250명이라면) 350×250에 해당하는 튜플들에서만 선택연산을 하면 된다. 이 예제에서 보듯이 릴레이션 전체를 실체화 대상으로 삼을 경우보다 축약 테이블을 이용하는 경우가 속도측면에서 4배($\frac{700\times500}{350\times250}$) 향상됨을 알 수 있다. 뿐만 아니라 실체화 뷰들의 저장공간도 2배($\frac{700+500}{350+250}$) 향상됨을 알 수 있다. 보기에서는 두 릴레이션에서만 언급되었지만 실제의 데이터 웨어하우스 환경에서는 수많은 뷰들이 존재하게 된다. 이러한 수십 기가바이트에 이르는 데이터 베이스 시스템 환경에서는 수많은 뷰들이 존재하게 된다. 이러한 환경에서 속도와 저장공간을 모두 2배에 가깝게 향상시키는 것은 데이터 웨어하우스 성능 면에서 아주 중요하다.

3.2 ASVMRT

본 절에서는 제안하는 알고리즘 ASVMRT에 대한 알고리즘의 단계를 기술하고 알고리즘의 각 단계를 예제를 통하여 보인다. 제안된 알고리즘은 다음과 같이 크게 4개의 단계로 이뤄져있다.

- Step 1 : k차원의 릴레이션들에서 고농도의 클러스터들을 찾는다.
- Step 2 : 구해진 클러스터들의 상·하한 값을 이용하여 축약 테이블을 생성한다.
- Step 3 : 생성된 축약 테이블을 이용하여 다중 뷰 처리계획들을 수립한다.
- Step 4 : 질의 처리속도 향상과 뷰 유지비용을 고려하여 효율적인 실체화 뷰를 선택한다.

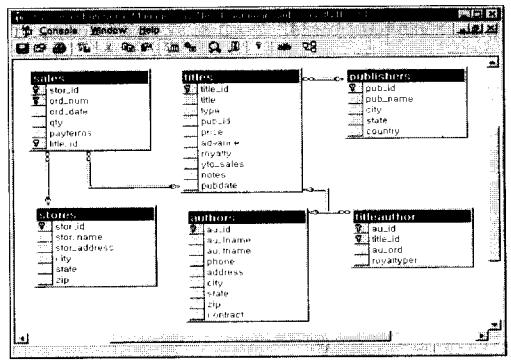
3.2.1 ASVMRT

```
ASVMRT( τ , n , T, Q, SC, UDT, UET) {
/* 사용자가 입력하는 임계치 τ */
/* 질의 개수 및 질의를 구성하는 테이블들의 개수에 관한 정보 n */
/* 대상 테이블들의 집합 T */
/* n 개 질의에 대한 정보를 가지는 집합 Q */
/* 사용자가 입력하는 공간제약 SC */
/* 사용자가 입력하는 클러스터링 차원의 정보를 가지는 집합UDT
   (클러스터링을 해야만 하는 차원에 대한 정보) */
/* 사용자가 입력하는 클러스터링 차원의 정보를 가지는 집합 UET
   (클러스터링을 하지 말아야 하는 차원에 대한 정보) */

   C = ∅; /* 클러스터들에 대한 정보를 가지는 집합 */
   RT = ∅; /* 축약 테이블에 대한 정보를 가지는 집합 */
   VP = ∅;/* 질의 처리계획에서 사용된 뷰들에 대한 정보를 가지는
             집합 */
   MV = ∅; /* 실체화 할 뷰들에 대한 정보를 가지는 집합 */

   for (i = 0; i<n; i++) {
         C = C ∪ find_cluster( τ , n, Ti, UDT, UET);
   }
   for (i = 0; i<n; i++) {
       RT = RT ∪ generate_reduct_table(Ci, Ti, RTi);
   }
   make_mvpp(n, Q, RT);
   select_view(VP);

   return MV;
}

/* Step 1: k차원의 릴레이션들에서 고농도의 클러스터들을 찾는다. */
find_cluster( τ , n, Ti, UDT) {
    T = Ti;
    target = 0; /* 속성들의 투영 농도를 비교하기 위한 값 */
    for (i = 0; i<n; i++) {
        for (j = 0; j<n; j++) {
            /* 주 키, 외래 키 및 사용자가 입력한 클러스터링을 위하
               지 않는 테이블의 차원은 클러스터링 대상에서 제외한
               다. */
            if (Ti.dj == primary_key || Ti.dj == foreign_key || Ti.dj ==
               UETi.dj) continue;
            /* 차원이 사용자가 입력한 차원일 경우에는 무조건 반영
               시킨다. */
            if (Ti.dj == UDT.dj) {
```

```
                for (k=0; Ti.di.low[k] != NULL; k++) {
                    /* 차원에 대해 고농도 클러스터 상·하한 구간
                       선정 */
                    Ci = Ti.di.low[k], Ti.di.high[k];
                }
                break; /* 다음 테이블로 이동 */
            }
            /* 차원 i의 차원 j에 대한 투영이 dense한가?
               기존 투영 농도보다 진한가? */
            /* 'Π' 연산자는 첫 번째 원소를 두 번째 원소에 대해 투영
               을 시켜서 투영 농도를 반환하는 연산자이다. */
            else if (Π(Ti.di, Tid) > τ && |Ci| > target) {
                target = |Ci|;
                for (k=0; Ti.di.low[k] != NULL; k++) {
                    Ci = Ti.di.low[k], Ti.di.high[k];
                }
            }
        }
    }
    return C;
}

/* Step 2: 구해진 클러스터들의 상·하한 값을 이용하여 축약 테이블을
   생성한다. */
generate_reduct_table(Ci, Ti) {
    /* '←' 연산자는 인덱스를 반환하는 연산자이다. */
    tmp ← Ti.Ci.low[0];
    for (k=0; Ti.Ci.low[k] != NULL; k++) {
        /* '[tmp]'는 tmp 인덱스가 가리키는 값을 반환하는 연산자이
           다. */
        while ([tmp] >= Ti.Ci.low[k] && [tmp] <= Ti.Ci.high[k]) {
            copy tuple from Ti to RT;
            tmp++;
        }
    }
    return RT;
}

/* Step 3: 생성된 축약 테이블을 이용하여 다중 뷰 처리계획을 수립한
   다. */
make_mvpp(n, Q, RT) {
    for (i = 0; i<n; i++) {
        /* 축약 테이블을 베이스 릴레이션으로 사용하여 n 개의 뷰
           처리계획을 생성한다. */
        make vpi using Q and RT as base relation instead Ti;
        count the number of nodes in vpi and save into NNi;
        /* NNi은 각각의 vpi에 대한 노드들의 수에 대한 정보를 가지는
           집합이다. */
    }
    /* n개의 뷰 처리계획들을 모두 합친다. 만약 중복 베이스 릴레이션
       및 중복 뷰들이 존재 할 경우 공통 뷰들을 가지는 질의 계획들을
       합치고, 해당 뷰나 릴레이션의 질의빈도 수를 증가한다. */
    for (i = 0; i<n; i++) {
        for (j=0; j<NNi; j++) {
            for (k=0; k<NNk; k++) {
                VP = VP ∪ vpi;
                /* 공통된 노드가 발견될 경우 질의빈도를 증가시킨
                   다. */
                if (vpi.nodej == VPi.nodek) {
                    VPi.nodek.fq ++;
                }
            }
        }
    }
    return VP;
}
/* Step 4: 뷰 처리 시간비용 및 뷰 유지비용을 고려하여 실체화 뷰를
   선택한다. */
select_view(VP) {
    /* n 개의 질의들에 대해 VP 노드들의 질의처리시간비용(Ca), 질의
       유지비용(Cm) 및 실체화했을 경우 총비용(Ct)을 계산한다. */
    for (i = 0; i<n; i++) {
        for (j=0; j<n; j++) {
            VPi.Ca = VPi.Ca + VPj.nodei.Ca;
            VPi.Cm = VPi.Cm + VPj.nodei.Cm;
            VPi.Ct = VPi.Ct + VPi.Ca + VPi.Cm;
        }
        VP.Ca = VP.Ca + VPi.Ca;
        VP.Cm = VP.Cm + VPi.Cm;
        VP.Ct = VP.Ct + VP.Ca + VP.Cm;
    }

    /* Ct순으로 오름차순으로 VP를 정렬한다. */
    sort(VP);

    /* 공간 제약 SC를 넘지 않도록 실체화 뷰를 선택한다. */
    for (i = 0; i<n; i++) {
        /* 연산자 ∑nMV는 MV의 전체 노드들이 차지하는 저장공간의
           반환하는 연산자이다. */
        if (∑nMV < SC) {
            MV = MV ∪ VPi;
            MV.Ct = MV.Ct + VPi.Ct;
        }
        else
            break;
    }
    return MV;
}
```

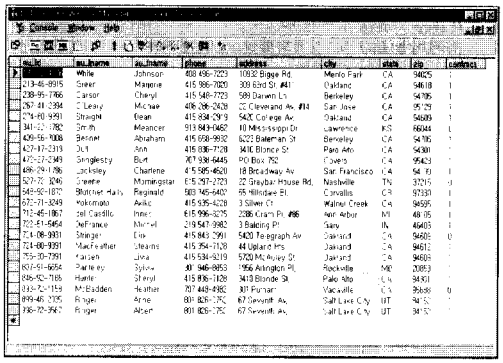
3.2.2 ASVMRT 예제

다음은 ASVMRT의 각 단계를 예제를 통하여 살펴본다. 우선 예제 테이블은 데이터 베이스 교육용으로 가장 많이 사용되고 있는 SQL Server 7.0의 Pubs 데이터 베이스의 'authors' 테이블로 한다. Pubs 데이터 베이스에 대한 스키마는 (그림 1)과 같다.



(그림 1) Pubs 데이터 베이스 스키마

다음 (그림 2)는 Pubs 데이터 베이스를 구성하고 있는 테이블들 중의 하나인 authors 테이블을 나타낸다.



(그림 2) authous 테이블

(그림 2)의 authors 테이블을 구성하는 속성들에 대한 상대적인 농도를 구하면 <표 2>와 <표 3>을 얻는다. <표 2>와 <표 3>에서 범위는 상대 농도의 계산을 개념적으로 쉽게 하기 위하여 수치정보의 범위를 가지는 차원에 대해서 0부터 9까지 10개의 구역으로 분류한 값이고, 문자정보의 범위를 가지는 차원에 대해서 a부터 z까지 26개의 구역으로 분류한 값이다. 실제의 데이터 웨어하우스 내부에 존재하는 레코드들의 개수가 상당히 많은 릴레이션들에 있어서는 보다 많은 구역으로 분류한다. 농도 계산은 테이블의 전체 레코드들 중 해당 범위에 속하는 레코드들의 개수를 정형화(normalization)시킨 값으로 계산한다. 예를 들어 전체 레코드들 중에서 zip 차원의 값이 9xxxx인 튜플들의 상대 농도는 전체 튜플들의 개수(23)에 대한 비율(16/23)에서 수치정보를 가지는 차원들을 정형화 시키기 위해서 사용되는 정형화 인수(10)를 나눈 값(0.06957)의 백분율

<표 2> authors 테이블의 수치 정보를 가지는 차원들에 대한 상대농도

범위	au_id	농도	phone	농도	address	농도	zip	농도	범위	contract	농도
0					1	0.435			0	4	8.696
1	1	0.435			4	1.739			1	19	41.304
2	4	1.739	1	0.435	3	1.304	1	0.435			
3	1	0.435	1	0.435	6	2.609	1	0.435			
4	4	1.739	13	5.652	1	0.435	2	0.87			
5	1	0.435	1	0.435	5	2.174					
6	2	0.87	2	0.87	3	1.304	1	0.435			
7	5	2.174	2	0.87							
8	4	1.739	2	0.87			2	0.87			
9	1	0.435	1	0.435			16	6.957			

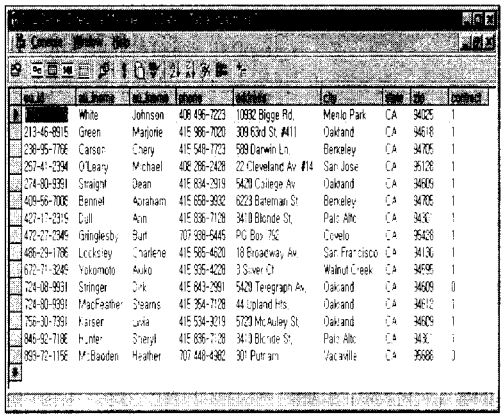
<표 3> authors 테이블의 문자 정보를 가지는 차원들에 대한 상대농도

범위	au_lname	농도	au_fname	농도	city	농도	state	농도
a			5	0.835	2	0.334		
b	2	0.334	1	0.167	2	0.334		
c	1	0.167	2	0.334	2	0.334	15	2.505
d	3	0.501	2	0.334				
e								
f								
g	3	0.501			1	0.167		
h	1	0.167	1	0.167				
i			1	0.167			1	0.167
j			1	0.167				
k	1	0.167					1	0.167
l	1	0.167	1	0.167	1	0.167		
m	2	0.334	5	0.835	1	0.167	2	0.334
n					1	0.167		
o	1	0.167			5	0.835	1	0.167
p	1	0.167			2	0.334		
q								
r	2	0.334	3	0.501	1	0.167		
s	3	0.501	1	0.167	4	0.668		
t							1	0.167
u							2	0.334
v								
w	1	0.167			1	0.167		
x								
y	1	0.167						
z								

로 표시한 것이다. 이경우 zip 차원에서 9xxxx값을 가지는 튜플들의 전체 테이블에 대한 농도는 69.57% (6.957*10)가 된다. 만약 사용자가 입력하는 클러스터링의 농도 임계 변수 τ를 60%라고 입력했을 경우 zip 차원은 선택 고려 차원들 중 하나의 후보로 선택된다. 여기서 우리는 사용자가 입력변수 *UDT*와 *UET*에 대한 정보로서 각각 zip 차원과 contract 차원을 입력했다고 가정한다.

<표 2>와 <표 3>에서 우리는 contract 차원의 상대적인 농도가 가장 높다는 것을 알 수 있다. 그러나, 사용자가 입력한 변수 *UET*에 contract 차원의 정보가 존재하므로 우리는 이 차원을 클러스터링 할 대상 차원으로 고려하지 않는다. 또한, au_id 차원은 주 키이므로 클러스터링 차원의 대상에서 제외된다. 그리고, 사용자가 입력변수 *UDT*에서 zip 차원을 클러스터링 대상차원으로 명시하였기 때문에 우리는 authors 테이블에서 zip 차원을 클러스터링 대상차원으로 선택한다. 만약 앞에서 가정한 사용자 입력변수인 *UDT*가 없다면 차원들의 상대농도가 6.957로서 가장 높은 zip 차원을 클러스터링의 대상 차원으로 선택하게 된다. 클러스터링 대상차원이 zip 차원으로 선택되고 난 후에는

해당 범위에 속하는 튜플들로만 구성된 (그림 3)과 같은 rt_authors 테이블을 얻는다.



(그림 3) authors 테이블의 rt_authors 테이블로의 축약

이와 같은 방법으로 데이터 웨어하우스에 존재하는 모든 릴레이션에 대해서 알고리즘의 두 번째 단계까지 수행하면 축약 테이블들이 만들어진다. 세 번째 단계에서는 축약 테이블들을 이용하여 다중 뷰 처리계획을 수립하게 된다.

알고리즘의 세 번째 단계를 위해서 다음과 같은 4개의 질의들이 있다고 가정하자.

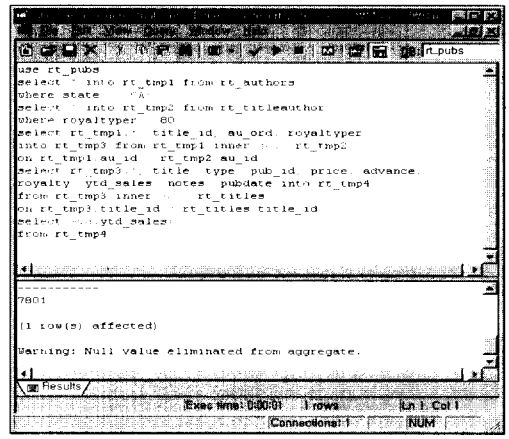
질의1 : royaltyper가 80이상이고 CA지역 거주자들의 year-to-date sales 평균은?

질의2 : 미국 CA지역 서점들 중 최근 3년 동안 (1993-1995) 가장 많이 팔린 책들 중 상위 3개의 책 종류는?

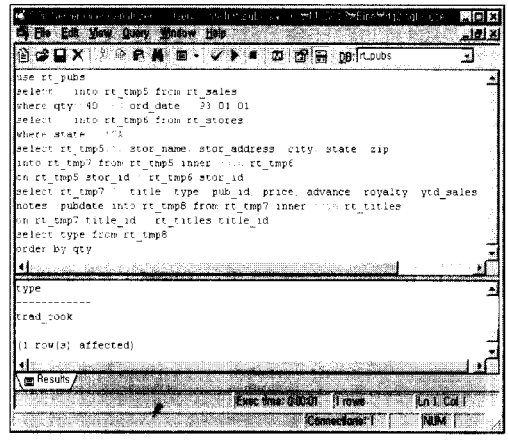
질의3 : royaltyper가 높은 책들 중에서 가격이 $15 이상인 경제관련 책제목에는 어떠한 것들이 있는가?

질의4 : 미국 발행 기관들이 발행한 심리학관련 책 중에서 CA 거주지역 저자가 쓴 책들에는 어떤 것들이 있는가?

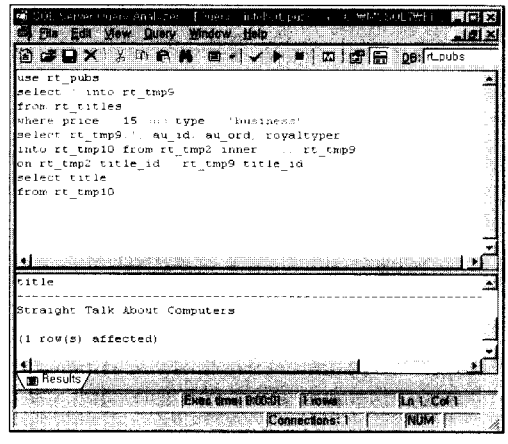
위 네 개의 질의에 대한 SQL 문장의 표현은 (그림 4), (그림 5), (그림 6) 및 (그림 7)에 나타나 있고 이것을 이용하여 다중 뷰 처리계획을 생성한 결과는 (그림 8)에 나타나 있다.



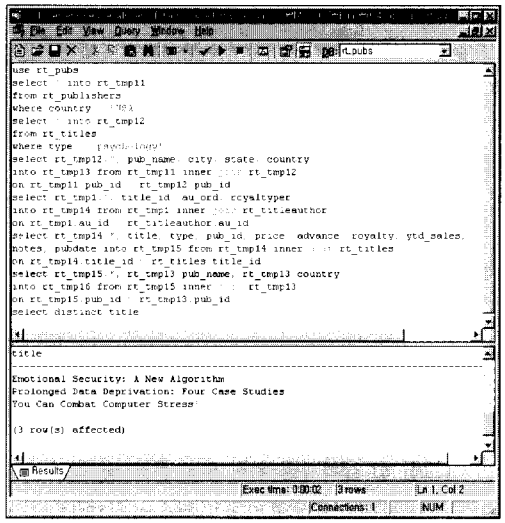
(그림 4) 질의 1을 위한 SQL 문장 및 결과



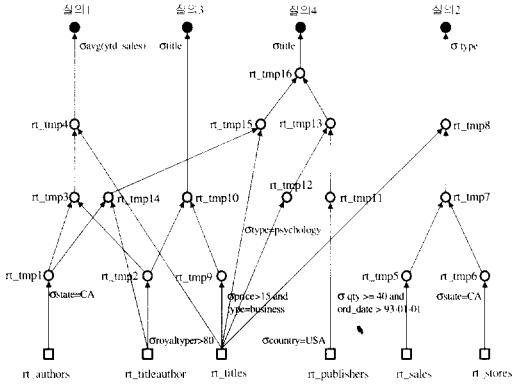
(그림 5) 질의 2를 위한 SQL 문장 및 결과



(그림 6) 질의 3을 위한 SQL 문장 및 결과



(그림 7) 질의 4를 위한 SQL 문장 및 결과



(그림 8) 질의 1, 2, 3, 4를 위한 다중 뷰 처리계획

위 (그림 8)에서 베이스 릴레이션은 □로 나타내고 있고 질의를 위한 중간값을 나타내기 위해서는 ○를 사용하고 있으며 질의는 ●로서 표시하고 있다.

(그림 8)과 같이 다중 뷰 처리계획이 수립되면 비용을 고려해서 실체화 할 뷰들을 선택한다. 본 논문의 전반에서 사용하고 있는 비용 측정의 기본 단위는 가장 흔히 사용하는 튜플의 수에 근거한다[1, 3].

(그림 8)에서 만약 rt_tmp3 릴레이션을 실체화 뷰로 선택할 경우의 총 비용 C_t는 질의 1을 답할 때 rt_tmp3와 rt_tmp4 릴레이션만 사용하면 되므로 rt_tmp3가 가지는 튜플들의 수(6)와 rt_tmp4가 가지는 튜플들의 수(6)를 더한 뷰 처리 시간비용 C_a(12)와 rt_tmp3를 실체화 뷰로 저장하고 있을 경우 rt_tmp3를 유지하는데 필요로하는 비용 rt_tmp3(6), rt_tmp1(15), rt_tmp2(10), rt_authors(15) 및 rt_titleauthors(10) 릴레이션들이 가지는 튜플들의 총합의 2배인 뷰 유지비용 C_m(2×56=112)을 더한 값(124)이 된다. 여기서 2를 곱해주는 이유는 실체화 뷰 rt_tmp3의 갱신이 발생할 경우 rt_tmp3를 포함하여 rt_tmp3의 자식노드들(rt_tmp1, rt_tmp2, rt_authors, rt_titleauthors) 모두를 재계산해야 하는 이유에서 비롯된다. 전체비용 T는 질의 1, 2, 3 및 질의 4에서의 총비용을 더한 값(124)이다. 이와 같은 방식으로 (그림 8)에서 우리는 <표 4>를 생성한다.

〈표 4〉 질의 1, 2, 3, 4에 대한 실체화 뷰 선택을 위한 비용 계산(축약 테이블 사용)

	f_a	t#	C_a				C_m				C_t				T
			Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	
rt_authors	1	15	42			54	0			0	42			54	96
rt_tijtleauthor	2	10	64		42	68	0		0	0	64		42	68	174
rt_titles	3	18	120	95	105	180	0	0	0	0	120	95	105	180	500
rt_publishers	1	6				29				0				29	29
rt_sales	1	11		14				0				14			14
rt_stores	1	3		8				0				8			8
rt_tmp1	2	15	54			78	60			60	114			138	252
rt_tmp2	2	10	44		22		40		40		84		62		146
rt_tmp3	1	6	12				112				124				124
rt_tmp4	1	6	6				124				130				130
rt_tmp5	1	1		3				24				27			27
rt_tmp6	1	3		5				12				17			17
rt_tmp7	1	1		2				38				40			40
rt_tmp8	1	1		1				40				41			41
rt_tmp9	1	2			3				40				43		43
rt_tmp10	1	1			1				81				82		82
rt_tmp11	1	6				23				24				47	47
rt_tmp12	1	5				22				46				68	68
rt_tmp13	1	5				17				80				97	97
rt_tmp14	1	6				24				92				116	116
rt_tmp15	1	6				18				140				158	158
rt_tmp16	1	12				12				220				232	232

<표 4>에서 보듯이 알고리즘의 3번째 단계에서 적용되고 있는 수식을 사용한 결과 공간 제약이 10일 경우 임시값을 지니는 뷰들 rt_tmp6, rt_tmp5, rt_tmp7, rt_tmp8, rt_tmp9를 선택하게 된다. 이 경우 뷰 실체화에 필요한 부가적인 공간은 8이다.

<표 4>, <표 5>, <표 7> 및 <표 8>의 첫 번째 열은 다중 뷰 처리계획에 사용된 릴레이션들이고, 두 번째 열은 각각의 릴레이션에 대한 질의빈도(f_q)이다. 세 번째 열은 각 릴레이션들이 가지고 있는 튜플들의 수($t\#$)이고, 네 번째, 다섯 번째 그리고 여섯 번째의 열은 각각 질의 1, 질의 2, 질의 3, 질의 4에 대한 각각의 뷰 처리 시간비용(C_a), 뷰 유지비용(C_m) 및 총비용이다 (C_t). 마지막 열은 모든 질의들에 대한 총 비용(T) 값

을 나타낸다.

〈표 5〉 질의 1, 2, 3, 4에 대한 실체화 뷰 선택을 위한 비용 계산(축약 테이블 사용하지 않음)

	f_q	f_u	C_a				C_m				C_r				T
			Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	
authors	1	23	50			89	0			0	50			89	139
tijtleauthor	2	25	94		72	152	0		0	0	94		72	152	318
titles	5	18	120	105	105	290	0	0	0	0	120	105	105	290	590
publishers	1	8				36				0				36	36
sales	1	21		30				0				30			30
stores	1	6		15				0				15			15
tmp1	2	15	54			132	72			72	126			204	330
tmp2	2	10	44		22		70		70		114		92		206
tmp3	1	6	12				158				170				170
tmp4	1	6	6				170				176				176
tmp5	1	3		9				48				57			57
tmp6	1	3		9				18				27			27
tmp7	1	3		6				72				78			78
tmp8	1	3		3				78				81			81
tmp9	1	2			3				10				13		13
tmp10	1	1			1				112				113		113
tmp11	1	6				28				28				56	56
tmp12	1	3				27				46				73	73
tmp13	1	3				22				84				106	106
tmp14	1	17				57				180				237	237
tmp15	1	17				34				250				284	284
tmp16	1	17				17				464				481	481

3.3 알고리즘 ASVMRT의 분석 및 특징

본 절에서는 앞의 3.2절에서 언급하고 있는 ASVMRT의 각 단계가 가지고 있는 특징들에 대해서 살펴본다.

알고리즘의 첫 번째 단계에서는 데이터 마이닝 기법 중 클러스터링을 사용하는 부분으로서 대상 베이스 릴레이션들에 대해서 고농도의 클러스터를 찾아내는 단계이다. 테이블의 각 차원들에 대해서 사용자가 입력한 임계치 τ를 넘는 최대 차원을 선택한다. 발견된 차원에 대해서 클러스터의 상·하한 값을 저장한다. 이 정보는 알고리즘의 두 번째 단계에서 사용된다. 이러한 기법은 기존 뷰 선택 알고리즘들에서는 찾아볼 수 없는 기법으로서 데이터 웨어하우스에서 사용자가 간과하고 있는 묵시적으로 중요한 정보를 제공할 수 있는 기회를 제공할 수 있다는 의미에서 중요하다. 또한 클러스터링 기법을 사용함으로써 잠재적으로 유용한 정보를 제공할 수 있을 뿐만 아니라 질의 수행 시간향상과 뷰 저장공간 절약이라는 이득을 볼 수 있다. 사용자는 임의로 클러스터링정보에 포함시키고자하는 차원을 알고리즘에 반영시키기 위해서 입력할 수 있다. 입력된 차원은 임계치를 넘는 다른 차원에 항상 우선한다. 이러한 기능의 부여는 외관적으로 작은 양의 데이터일지라 하더라도 중요한 정보를 담고 있는 차원의 정보일 경우, 클러스터링에 반영할 수 있다는 것을 나타낸다. 이러한 사용자 차원 입력기능으로 말미암아 중요한 정보를 소실할 가능성을 배제하는 것이다.

알고리즘의 두 번째 단계에서는 테이블마다 저장된 클러스터에 대한 상·하한 값을 이용하여 해당되는 튜플들만 저장하는 축약 테이블을 만든다. 기존의 알고리즘들에서는 베이스 릴레이션 전체의 튜플들에 대해서 실체화 대상 뷰로 삼고 있으나 본 알고리즘에서는 축약 테이블의 튜플들에만 국한함으로서 알고리즘이 추구하고 있는 질의응답 시간향상과 뷰 저장공간 절약의 목표를 달성할 수 있다. 그러나, 질의응답 시간향상과 저장공간 절약 모두를 성취하기 위해서는 기존 알고리즘의 방법보다 더 많은 임시 저장장소를 요구하고, 클러스터링을 수행하는데 부가적인 시간이 필요하다. 그러나, 수 테라 바이트에서 수십 테라 바이트에 이르는 막대한 정보가 저장되어 있는 데이터 웨어하우스 환경에서는 뷰 갱신과 같은 작업을 온라인으로 할 수 없다. 이러한 이유로 오프라인으로 행해지는 베이스 릴레이션들에 대한 클러스터링 작업과 축약 테이블 생성과정은 기존 알고리즘들에서 사용하는 방법보다 효율적인 면에서 저하되지는 않는다.

알고리즘의 세 번째 단계에서는 두 번째 단계에서 생성된 축약 테이블을 이용하여 다중 뷰 처리계획을 생성한다. 기존의 알고리즘[3]에서는 다중 뷰 처리계획을 수립하기 위한 방법으로서 0-1 정수프로그램(integer programming) 방법과 HA_{mvpp}를 제안하고 있다. 0-1 정수프로그램 방법은 최적의 다중 뷰 처리계획을 생성하지만 실제 환경에서는 구현하기에 너무나 많은 시간을 필요로 하고, HA_{mvpp}는 구현 가능한 알고리즘이지만 최적의 다중 뷰 처리계획을 생성하지 못하는 문제점을 가지고 있다. 이에 본 알고리즘에서는 이러한 문제점을 해결할 수 있도록 다중 뷰 처리계획의 수립에 있어서 질의 빈도를 이용하여 오프라인으로 구현 가능한 프로시저를 제공하고 있다.

알고리즘의 네 번째 단계에서는 생성된 다중 뷰 처리계획에서 뷰 처리 시간비용과 뷰 유지비용을 고려하여 해당 뷰를 실체화했을 경우에 이득이 생기는 뷰들을 사용자가 입력한 공간제약을 넘지 않는 범위 내에서 선택한다. 기존 알고리즘에서는 선택연산자들에 대한 비용 고려를 하지 않고 단지 조인(join) 연산만 고려했을 뿐만 아니라 질의 빈도를 질의 자체에만 국한시켰다. 그러나 이러한 방법에 근거한 비용계산은 비용계산 측면에서 모든 요소를 포함시키지 않고 있다.

따라서 ASVMRT에서는 이러한 선택연산자의 사용에 대한 비용까지 포함하고 있는 비용계산식을 사용하고 있다. 그리고 질의 빈도를 질의 자체에 부과하지 않고 질의를 구성하고 있는 모든 뷰들에 대해서 부과한다. 이것은 질의를 구성하고 있는 뷰들이 다른 질의에서 사용될 수 있다는 것을 고려한 것이다.

4. 실험결과 및 분석

본 장에서는 3장에서 언급한 pubs 데이터 베이스를 이용한 실험 및 결과를 살펴보고, 데이터 베이스 크기가 상대적으로 큰 한국전자통신연구원의 정보통신 기술기준 정보시스템 검색방법중의 하나인 조항별 키워드 검색의 응답시간을 향상시키기 위해서 ASVMRT를 적용한 실험결과를 보인다.

4.1 Pubs 데이터 베이스에서의 실험 및 결과

3장에서 사용된 4개의 질의((그림 3), (그림 4), (그림 5) 및 (그림 6))에 대한 다중 뷰 처리계획을 축약 테이블을 생성하지 않고 릴레이션 전체를 대상으로 삼는 기존 알고리즘의 경우에 대해서 비용을 계산하면 <표 3>과 같다.

전체 질의로그를 참조해서 <표 4>와 <표 5>를 정리하여 기존 알고리즘들에서 취하고 있는 방법과 제안하는 알고리즘에서 사용하는 방법과의 차이를 정리하면 <표 6>과 같다. 두 종류를 비교하기 위해서 ASVMRT의 사용자 입력 변수인 공간제약 SC가 주어지지 않았다고 가정한다.

〈표 6〉 Pubs 데이터 베이스에 대한 기존 알고리즘과 제안 알고리즘의 비교

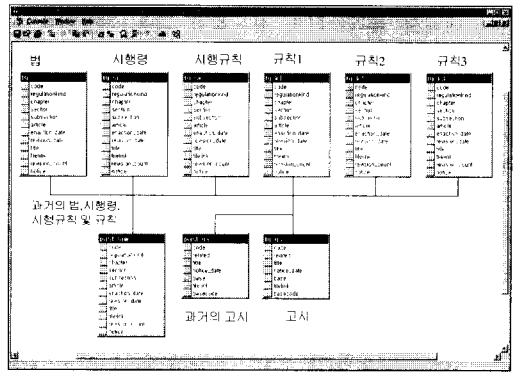
		기존 알고리즘들	제안된 알고리즘
부분적으로 뷰를 실체화 했을 경우	실체화 뷰	tmp5, tmp6, tmp7, tmp8	rt_tmp5, rt_tmp6, rt_tmp7, rt_tmp8
	전체 비용	243	125
	저장 공간	12	6
전체 뷰를 실체화 했을 경우	실체화 뷰	ALL	ALL
	전체 비용	3,646	2,441
	저장 공간	220	149

<표 6>에서 알 수 있듯이 부분적으로 뷰들을 실체화했을 경우 기존 알고리즘들에서 사용하고 있는 방법보다 제안된 알고리즘 방법이 질의응답시간 요소에서 1.944(243/125)배의 효율성을 보이고 있으며 뷰 저장공간 요소에서 2(12/6)배의 효율성을 보이고 있다. 뿐만 아니라, 해당 뷰들의 적절한 선택 알고리즘의 적용이 없을 경우 즉, 모든 임시 뷰들을 실체화했을 경우에 있어서도 기존 알고리즘들에서 사용하고 있는 방법보다 제안된 알고리즘에서 질의응답시간 및 뷰들의 저장공간에서 각각 1.493(3,646/2,441), 1.476(220/149)배의 효율성을 지닌다.

기존 알고리즘과의 전체평균 1.5배에 해당하는 향상은 4.2절에서 보이고 있는 평균 1.8배의 향상과의 차이를 보이고 있다. 그 이유는 pubs 데이터 베이스에 존재하는 릴레이션들의 레코드 수가 충분히 크지 않기 때문이다. 따라서, 테이블들이 가지는 레코드 수가 충분히 많은 데이터 베이스를 대상으로 실험한 결과를 4.2절에서 보인다.

4.2 정보통신 기술기준 정보시스템 데이터 베이스에서의 실험 및 결과

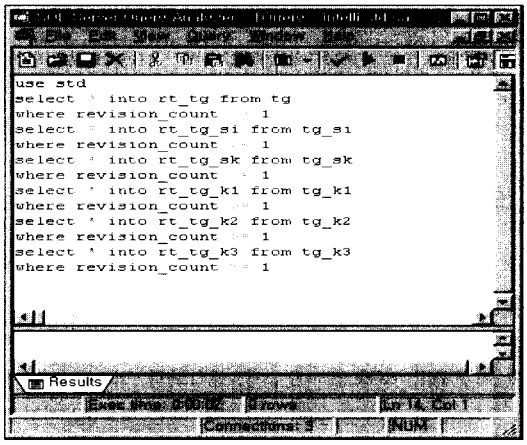
본 절에서는 pubs와 같은 작은 데이터 베이스가 아니라 실제적으로 사용되고 있는 많은 양의 데이터를 가지고 있는 데이터 베이스를 대상으로 한 실험 및 결과를 보인다. 본 절에서 사용하고 있는 데이터 베이스는 한국전자통신연구원의 정보통신 기술기준 정보시스템에서 사용하고 있는 데이터 베이스이다. 본 시스템은 정보통신 기술기준 관련 정보를 위해서 구축된 사이트이다. 정보시스템에서 사용하고 있는 데이터 베이스 스키마는 (그림 9)와 같다.



(그림 9) 정보통신 기술기준 정보시스템의 데이터 베이스 스키마

본 논문에서는 대한민국 정보통신법을 이루고 있는

14개의 법령 중에서 가장 많은 튜플 수를 가지고 있는 전파법에 대해서 실험을 했다. 정보관리 시스템에서 제공하고 있는 4가지 검색방법중의 하나로서 사용자가 입력한 키워드와 법 조항의 제목을 비교하여 검색결과를 반환하는 조항별 키워드 검색이 있다. 조항별 키워드 검색의 응답시간을 향상시키기 위해서 알고리즘의 첫 번째 단계인 클러스터링 단계에서 관련 고시가 있는 법 조항들을 클러스터링할 대상으로 삼는다. 전파법에 대해서 ASVMRT의 첫 번째 단계와 두 번째 단계를 (그림 10)에서 보는바와 같이 수행한다.



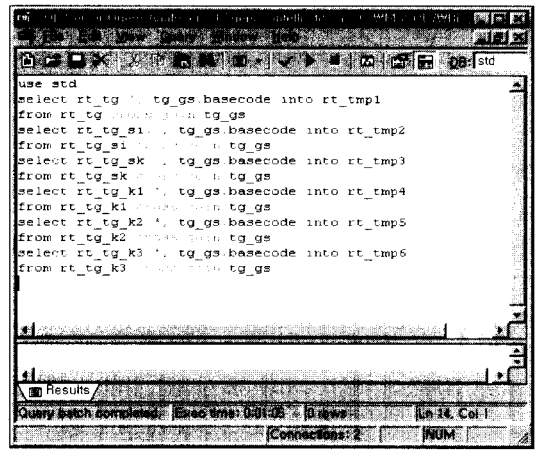
(그림 10) 질의 5를 위한 축약 테이블 생성

전파법에 대해서 축약 테이블 생성과정이 끝나게 되면 ASVMRT의 세 번째 단계인 질의로그들을 참조해서 다중 뷰 처리계획을 생성하게 된다. 다음과 같은 질의가 있다고 가정한다.

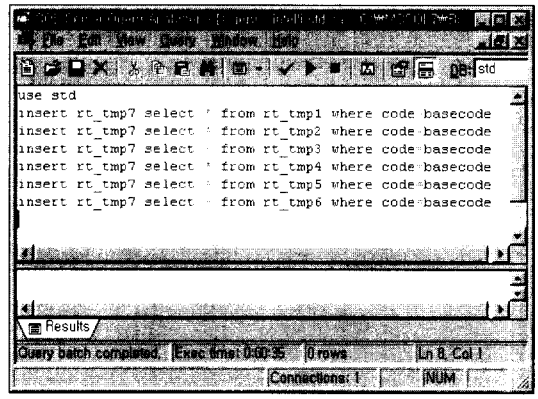
질의 5 : 전파법 관련 법 조항들 중에서 전파법 관련 고시의 근거가 되는 법 조항들을 모두 나열하시오.

위 질의 5는 관련 고시와 관련된 모든 질의를 검색하는 질의로서 사용자가 어떠한 키워드를 입력하더라도 위 질의에 포함된다. 따라서 고시관련 정보검색을 위해서 반환하는 레코드의 모든 가능한 범위를 포함하고 있는 질의이기 때문에 고시관련 정보검색을 위한 모든 질의들을 대표한다고 할 수 있다.

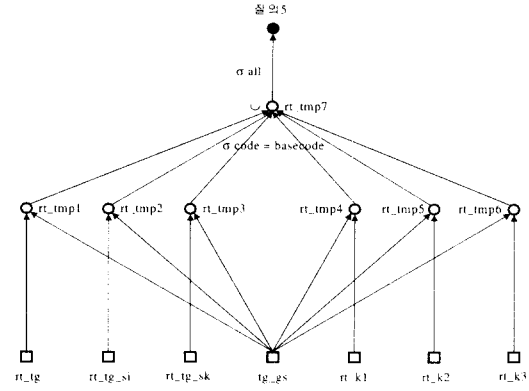
위 질의 5에 대해서 ASVMRT의 세 번째 단계를 (그림 11) 및 (그림 12)와 같이 수행한 후 주어진 질의를 처리하기 위해 (그림 13)과 같이 다중 뷰 처리계획을 생성한다.



(그림 11) 질의 5를 위한 다중 뷰 처리계획 생성 과정 1



(그림 12) 질의 5를 위한 다중 뷰 처리계획 생성 과정 2



(그림 13) 질의 5를 위한 다중 뷰 처리계획

기존 알고리즘들에서 취하고 있는 방법을 사용한 결

과와 ASVMRT를 적용한 후의 결과가 각각 <표 7> 및 <표 8>에 나타나 있다. 만약 ASVMRT의 사용자 입력 변수인 SC를 30,000이라고 입력을 한 경우에 <표 7>에서 rt_tmp6, rt_tmp5, rt_tmp1, rt_tmp2를 순서대로 실체화 뷰로 선택하게 된다. 이 경우 뷰 저장공간에 사용되는 공간은 23,612이다. <표 9>에서 보듯이 기존 방법을 적용한 것보다 질의응답 시간 측면에서 1.754(127,417/72,632))배, 뷰 저장공간 측면에서 1.768(41,769/23,612)배 더 좋은 결과를 보이고 있다.

〈표 7〉 질의 5에 대한 실체화 뷰 선택을 위한 비용계산 (축약 테이블 사용)

	f_i	t#	C_a	C_m	C_i
			Q5	Q5	Q5
tg_gs	1	119	46,810	0	47,405
rt_tg	1	71	8,682	0	8,682
rt_tg_si	1	72	8,752	0	8,752
rt_tg_sk	1	116	14,032	0	14,032
rt_tg_k1	1	77	9,352	0	9,352
rt_tg_k2	1	29	3,592	0	3,592
rt_tg_k3	1	26	3,232	0	3,232
rt_tmp1	1	8,499	8,611	17,378	25,989
rt_tmp2	1	8,568	8,680	17,518	26,198
rt_tmp3	1	13,804	13,916	28,078	41,994
rt_tmp4	1	9,163	9,275	18,718	27,993
rt_tmp5	1	3,451	3,563	7,198	10,761
rt_tmp6	1	3,094	3,206	6,478	9,684
rt_tmp7	1	112	112	94,402	94,514

〈표 8〉 질의 5에 대한 실체화 뷰 선택을 위한 비용계산 (축약 테이블 사용하지 않음)

	f_q	t#	C_a	C_m	C_i
			Q5	Q5	Q5
tg_gs	1	119	84,039	0	84,039
tg	1	121	14,634	0	14,634
tg_si	1	132	15,954	0	15,954
tg_sk	1	219	26,394	0	26,394
tg_k1	1	134	16,194	0	16,194
tg_k2	1	49	5,994	0	5,994
tg_k3	1	49	5,994	0	5,994
tmp1	1	14,399	14,513	29278	43,791
tmp2	1	15,708	15,822	31918	47,740
tmp3	1	26,061	26,175	52798	78,973
tmp4	1	15,946	16,060	32398	48,458
tmp5	1	5,831	5,945	11998	17,943
tmp6	1	5,831	5,945	11998	17,943
tmp7	1	114	114	169426	169,540

<표 7>과 <표 8>을 정리 및 비교하면 <표 9>와 같다. ASVMRT의 사용자 입력 변수인 공간제약 SC가 주어지지 않은 환경에서 평균적으로 계산할 때, <표 9>에서 알 수 있듯이 기존 알고리즘들에서 사용하는 방법보다 제안된 알고리즘에서 질의 응답시간에서 1.786(593,591/332,180)배, 실체화 뷰를 위한 저장공간에서 1.794(84,713/47,201)배 더 좋은 효율성을 보인다.

〈표 9〉 std 데이터 베이스에 대한 기존 알고리즘과 제안 알고리즘의 비교

		기존 알고리즘들	제안된 알고리즘
부분적으로 뷰를 실체화 했을 경우	실체화 뷰	tmp1, tmp2, tmp5, tmp6	rt_tmp1, rt_tmp2, rt_tmp5, rt_tmp6
	전체 비용	127,417	72,632
	저장 공간	41,769	23,612
전체 뷰를 실체화 했을 경우	실체화 뷰	ALL	ALL
	전체 비용	593,591	332,180
	저장 공간	84,713	47,201

5. 결론 및 향후과제

제안된 알고리즘 ASVMRT는 첫 번째 단계에서 주어진 테이블들의 차원으로 고농도의 클러스터들을 발견하고, 두 번째 단계에서 발견된 클러스터들의 정보를 이용하여 축약 테이블들을 생성한다. 세 번째 단계에서는 축약 테이블들을 이용하여 다중 뷰 처리계획을 생성하고 마지막 단계인 네 번째 단계에서 생성된 다중 뷰 처리계획을 이용하여 비용계산에 근거하여 실체화 할 대상 뷰들을 선택하는 실체화 뷰 선택 알고리즘이다.

시장 경향 분석을 통하여 최고 경영자에게 기업의 나아갈 지표를 제공할 수 있는 OLAP 기능을 가지는 데이터 웨어하우스에서는 질의 응답시간의 향상을 위해서 뷰 실체화 기법이 요구된다. 이러한 뷰 실체화 기법으로 본 논문에서는 데이터 마이닝에서 사용되는 기법들 중의 하나인 클러스터링 기법을 적용하여 ASVMRT를 제안하였다. 제안된 알고리즘에서는 중요한 정보를 소실할 가능성을 배제하는 사용자 입력 차원 기능, 클러스터링의 압축 정도를 결정할 수 있는 사용자 입력 임계치 부여기능 및 주어진 공간제약을 넘지 않는 범위 안에서 실체화 뷰들을 선택하도록 하는 사용자 입력 공간제약기능이 있다. 이러한 사용자와의 인터페이스는 기존 알고리즘들에서는 찾아볼 수 없다.

실험결과에서 알 수 있듯이 축약 테이블을 사용하고

있는 ASVMRT의 경우 기존 알고리즘들에서 사용되는 방법보다 질의 응답 시간적인 측면과 실체화 뷰 저장 공간적인 측면 모두에서 평균 1.8배의 향상을 보인다. 뿐만 아니라, ASVMRT의 사용자 입력 변수인 공간 제약 SC가 주어지지 않을 경우, 즉 공간제약이 없다고 가정할 경우에서도 Pubs 데이터 베이스 경우 약 1.5배의 향상을 보였으며, 한국전자통신연구원의 정보통신 기술기준 정보시스템의 데이터 베이스 경우 약 1.8배의 향상을 보였다.

데이터 웨어하우스의 실체화 뷰와 관련하여 두 가지 이슈가 있다. 그 중 하나는 실체화 뷰 선택 문제이고 나머지 하나는 실체화 뷰 유지 문제이다. 본 논문에서 제안된 ASVMRT는 실체화 뷰 선택을 위한 해결책이다. 두 번째 이슈에 대해서 우리는 소스 데이터 베이스의 갱신이 있을 경우 데이터 웨어하우스의 일관성을 유지하기 위해서 ASVMRT에서 실체화 뷰 선택에 반영된 축약 테이블들을 어떻게 갱신할 것인지에 대한 연구를 할 것이다.

참 고 문 헌

[1] V. Harinarayan, A. Rajaraman, and J. Ulman, Implementing data cubes efficiently. In Proc. of the ACM SIGMOD International Conference of Man agement of Data, Canada, June 1996.

[2] H. Gupta, Selection of views to materialized in a data warehouse, in ICDT, 1997

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

[4] W. H. Inmon, Building the Data Warehouse, Second Edition, John Wiley and Sons. Inc., 1996.

[5] A. Gupta, I. S. Mumick, Maintenance of Materialized Views : Problems, Techniques, and Applications, IEEE Data Engineering Bulletin, Special Issue on Materialized Views and Data Warehousing, 18(2), pp.3-18, June 1995.

[6] Red Brick System, Ins & Outs(and everything in between) of Data Warehousing, Red Brick Systems white paper, 1996.

[7] M.-S. Chen, J. Han and P. Yu, Data Minning : An Overview from Database Perspective, IEEE Trans. on Knowledge and Data Engineering, 1997.

[8] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami, Database Mining : A Performance Perspective, IEEE Transactions on Knowledge and Data Engineering, Vol.5, No.6, pp.914-925, December 1993.

[9] Berson, J. Smith, Data Warehousing, Data Mining, & OLAP, McGraw-Hill, 1997.

[10] Fayyad U. M., Piatetsky-Shapiro G., Smyth P and Uthurusamy R., Advances in Knowledge Discovery and Data Mining., Cambridge Ma : AAAI Press/MIT press 1996.

[11] R. Agrawal and R. Srikant, Fast algorithms for mining association rules, In Processings of the 20th VLDB Conference, Santiago, Chile, Sept. 1994.

[12] J. S. Park, M. S. Chen, and P. S. Yu, An effective hash-based algorithm for mining association rules, In Preceedings of ACM SIGMOD Conference on Management of Data, pp.175-186, SanJose, California, May, 1995.

[13] J. Gary, A. Bosworth, A. Layman, H. Pirahesh, Data Cube : A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals, Micro soft Technical Report No. MSR-TR-95-22.

[14] K. A. Ross, D. Srivastava, and S. Sudarshan, Materialized View Maintenance and Integrity Constraint Checking : Trading Space for Time, In Proc. ACM SIGMOD '96, pp.447-458, Montreal, June 1996.

[15] H. Gupta, V. Harinarayan, A. Rajaraman, J. D. Ullman, Index Selection for OLAP, Proceedings of the International Conference on Data Engineering, pp.208-219, Binghamton, UK, April, 1997.

[16] E. Baralis, S. Paraboschi, E. Teniente, Materialized View Selection in a Multidimensional Database, Proc. VLDB '97, pp.156-165.

양 진 혁

e-mail : grjinh@tiger.korea.ac.kr

1998년 고려대학교 전산학과 졸업 (이학사)

2000년 고려대학교 대학원 전산학과(이학석사)

2000년~현재 고려대학교 대학원 전산학과 박사과정 재학중

관심분야 : 데이터 웨어하우스, 데이터 마이닝, 정보 검색, 지능형 에이전트

정 인 정

e-mail : chung@tiger.korea.ac.kr

1978년 서울대학교 계산통계학과 (이학사)

1980년 한국과학원 전산학과 (공학석사)

1989년 미국 University of Iowa 전산학과(전산학박사)

1980년~1983년 삼성전자 컴퓨터 사업부연구원

1981년~1984년 홍익대학교 및 동국대학교 전자계산학과 강사

1983년~1984년 이화여자대학교 전자계산학과 전임강사

1996년~1998년 고려대학교 서창 계산소장

1990년~현재 고려대학교 전산학과 조교수, 부교수, 교수

관심분야 : 전문가 시스템, 데이터 웨어하우스 및 데이터 마이닝, 정보검색 및 디지털 라이브러리, 에이전트 시스템