관계 DBMS 상에서 전위 방식의 OLAP 큐브 생성 알고리즘의 성능 평가*

조선화° 김진호 문양세
강원대학교 컴퓨터과학과
{jsunf°, jhkim, jsmoon}@mail.kangwon.ac.kr

Performance Evaluation of Front-End OLAP Cube Generation Algorithms on Relational DBMS

Sunhwa Jo°, Jinho Kim, and Yangsae Moon
Dept. of Computer Science, Kangwon University

요 약

ROLAP 시스템에서는 다차원 OLAP 큐브를 관계 데이터베이스 내에 여러 집계 테이블을 사용하여 저장하며, 관계 DBMS 기능을 그대로 이용하므로 구현이 간단하다. 이들 집계 테이블들은 대용량의 소스 데이타(즉, 사실 테이블)를 정렬한 후 이에 대한 집계 값을 계산하므로 큐브를 생성하는데 많은 시간이 소요된다. 이러한 다차원 큐브를 효율적으로 생성할 수 있는 여러 가지 방법이 제안되었다. 이들 방법들은 큐브 생성 시간이 사실 테이블을 정렬하는데 주로 소요되므로 이 횟수를 줄이는 기법을 주로 제안하였다. 그러나 이러한 큐브 생성 알고리즘의 성능은 실제 DBMS 상에서 평가되지 않았다. 이 연구에서는 기존의 큐브 생성 알고리즘들을 관계 DBMS 상에서 그 성능을 비교.평가하였다.

1. 서 론

OLAP(On-Line Analytical Processing) 시스템은 의사 결정 및 지원에 필요한 분석 결과를 온라인으로 빠르게 제공할 수 있도록 다양한 차원에 따라 사실 값에 대한 집계 값들을 계산하여 저장해 두는 데, 이를 다차원 큐브(multidimensional cube, 이하 큐브)라 한다. 큐브는 정의된 모든 차원들의 조합에 대해 집계 값을 생성하므로 많은 시간이 소요된다. 이러한 큐브는 특별히 고안된 다차원 구조에 저장하거나(MOLAP의 경우), 관계 데이터베이스이 테이블에 저장하는(ROLAP의 경우) 방법이 있다. ROLAP의 경우 관계 DBMS의 기능을 그대로 이용하여 집계 데이터를 저장하며, 별도의 특별한 저장 구조를 사용하지 않는다는 장점이 있다. 그러나 각 차원의 모든 조합들에 대해 집계 값을 저장하는 테이블을 생성하므로 많은 개수의 집계 테이블을 생성해야 한다. 매 집계 테이블을 만들 때마다 방대한 소스 테이블을 정렬해야 하며, 많은 회수의 정렬로 인해 큐브를 생성하는데 많은 시간이 소요된다는 단점이 있다[1,2].

ROLAP에서 큐브를 효율적으로 생성하기 위한 여러 가지 방법이 제안되었다. 이들 방법들은 큐브 생성 시간에 큰 비중을 차지하는 정렬 시간을 줄이기 위해, 한번의 정렬로 여러 집계 테이블을 생성하는 방법을 주로 사용하였다. 그러나 이들 알고리즘을 실제 관계 DBMS 상에서 전위(front-end) 방식으로 구현할 경우, 큐브 생성 알고리즘 자체의 비용 뿐만 아니라 DBMS와의 상호 작용을 위한 통신 오버헤드가 발생한다. 따라서 큐브 생성 시에 정렬 비용 뿐만 아니라 DBMS와의 통신 비용을 함께 고려해야 한다. 따라서 이 연구에서는 대표적인 큐브 생성 알고리즘들을 관계 DBMS 상에서 그 성능을 비교. 평가하였다. 또한 실험을 통해, 이러한 환경에서 큐브를 생성할 경우 성능에 가장 큰 영향을 미치는 원인을 찾아 내고자 한다.

2. 관련 연구

ROLAP 큐브 생성은 사실 테이블의 모든 가능한 차원의 조합의 집계 테이블을 생성하는 과정으로, n개의 차원 요소를 갖는 사실 테이블의 경우, 2^n개의 집계 테이블이 생성된다. 이 큐브를 생성하는 기존 연구들은 사실 테이블을 여러번 정렬한 후 이로부터 집계 데이터를 추출하는 방법과, 사실 데이터를 다차원 배열 형태로 저장한 후 이로부터 집계 데이터를 추출하는 방법으로 구분된다.

정렬에 의한 큐브 생성 방법 중 가장 단순한(naive) 방법은 큐브를 구성하는 모든 집계 테이블을 독립적으로 생성하는 것이다. 이를 위해 각 차원 테이블들을 분석하고자 하는 차원 애트리뷰트 순서로 Group By, 즉 정렬 연산을 수행하는 데 2^n개의 집계 테이블을 생성하기 위해 차원 순서로 사실 테이블을 2^n번 정렬해야 한다. 크기가 방대한 사실 테이블을 정렬하는 데에는 많은 시간과 시스템 오버헤드를 요구하므로, 정렬의 횟수를 줄이기 위한 여러 기법이 제안되었는데, 그 대표적인 방법이

* 본 연구는 첨단정보기술연구센터(AITrc)를 통하여 한국과학재단(KOSEF)의 지원을 받았음.

Pipesort[3]이다.

Pipesort 기법은 집계 테이블 간의 유도 가능 관계를 방향성 그래프로 나타낸 큐브 래티스(lattice)를 집계 테이블 생성 순서 및 계산에 사용한다. 큐브 래티스는 루트 노드에서 리프 노드까지의 모든 노드의 집계 테이블을 생성하도록 하는 큐브 트리로 전환된다. 큐브 래티스로부터 얻어진 큐브 트리를 참고하여 이전 집계 테이블의 정렬 결과를 다음 단계의 집계에 (파이프라인 처럼) 재사용함으로써 방대한 소스 테이블의 반복적인 재정렬을 피하고, 따라서 그로 인한 시스템 오버헤드와 실행 시간을 크게 감소시킬 수 있다.

Pipesort 기법은 큐브 생성 방법으로 가장 널리 사용/연구되는 방법중의 하나이다. 그러나 이 알고리즘은 실제 DBMS 상에서 그 성능을 평가한 것이나 아니라, 화일에 저장된 데이터 집합을 이용하여 평가하였다. 따라서 이 연구에서는 기존의 큐브 생성 알고리즘을 상용 RDBMS를 사용하여 전위 방식으로 구현한 후 그 성능을 비교.분석하고자 한다.

3. 큐브 생성 알고리즘의 구현

큐브 생성기는 DBMS 내부 또는 외부에 그 생성 모듈을 둘 수 있다. DBMS 내부에 큐브 생성 모듈을 밀결합하는 경우는 DBMS 소스 자체를 수정해야 하므로 매우 복잡하고 많은 노력을 필요로 한다. 따라서 일반 ROLAP 시스템에서는 상용 DBMS의 기능과 성능을 그대로 이용하므로 구현이 비교적 간단하고 DBMS의 소스를 수정하는 과정이 필요 없으므로 개발이 매우 용이하다. 그러나 전위 방식으로 개발하는 경우에 DBMS와의 잦은 통신과 이로 인한 오버헤드는 시스템 성능에 영향을 주게 된다. 이러한 오버헤드를 고려하여 큐브 생성 알고리즘의 성능을 향상시킬 수 있는 요소를 탐구하기 위해 기존의 큐브 생성 알고리즘을 전위 방식의 큐브 생성기로 구현하여 그 성능을 평가해 보고자 한다.

큐브 생성 방법으로 가장 기본적인 방법인 Naive와 Pipesort 알고리즘을 SQL Server 2000과 Java J2SE v.1.4.2를 사용하고, Intel Pentium 4 CPU 2.4GHz, 256MB RAM, WindowsXP 시스템 상에 구현하였다.

큐브를 생성하는 가장 기본적인 방법(이하 Naive)은 n개의 차원을 갖는 사실 테이블에서 분석 가능한 2^n개의 차원의 조합에 의한 집계 값을 계산하기 위해 소스 사실 테이블을 2^n번 정렬한다.

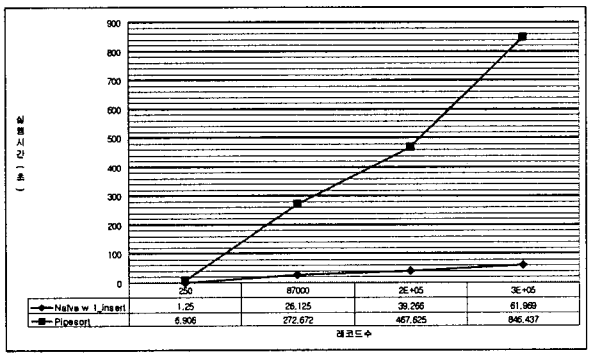
구현하고자 하는 두 번째 큐브 생성 방법인 Pipesort 방법은 각 집계 테이블을 생성하기 위해 소스 테이블을 2^n번 정렬하는 Naive 방법을 개선하는 것이다. 특정 애트리뷰트 순서로 정렬하여 집계 값을 계산하고, 다음 단계의 집계 값을 계산할 때는 소스 테이블을 재정렬하는 대신 캐시에 저장된 정렬 순서를 활용하여 집계값을 계산한다.

4. 성능 비교 및 분석

구현한 각 알고리즘의 큐브 생성 성능을 비교 평가하는 실험을 수행하였다. 실험에 사용하는 데이터는 MS SQL Server 2000에 관계형 테이블로 저장한 FoodMart2000의 Sales_fact 사실 테이블을 사용한다.

4.1. Naive vs. Pipesort 실행 시간 비교

product, time, customer, store 차원의 조합에 대하여 store_sales의 측정값을 집계하는 큐브를 생성하고 저장하는 과정에 소요된 실행 시간을 측정하였다. 아래의 [그림 4]는 같은 실험 환경에서 Naive와 Pipesort 알고리즘을 레크드 수에 따라 그 실행 시간을 측정하여 나타낸 그래프이다. 그래프에 의한 실험 결과는 Naive에 비해 우수한 성능을 기대한 Pipesort 알고리즘의 성능이 매우 낮은 것으로 나타났다.



[그림 4] Naive 와 PipeSort 큐브 생성 알고리즘의 실행 시간 비교

Naive 방법의 구현에서는 1개의 insert 문으로 집계값을 해당 집계 테이블에 삽입하게 되는 데, Pipesort 방법에서는 레코드 단위로 읽고 임시 저장소에 둔 중간 결과값을 테이블에 삽입하므로 Naive에 비해 매우 빈번히 insert 문을 실행하게 된다. 이와 같은 구현 결과로, insert 문의 실행 횟수가 성능에 미치는 영향을 알아보기 위해 실험을 수행하였다.

4.2 성능 분석 실험

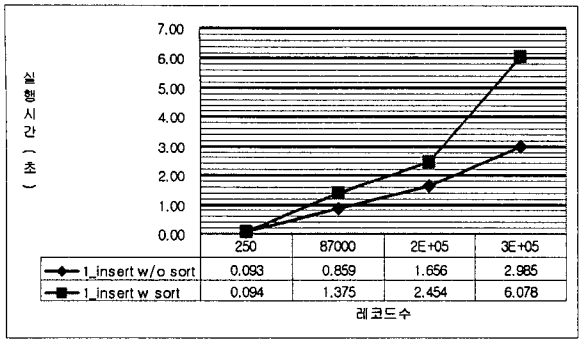
OLAP 큐브 생성 방법의 성능은 정렬 시간이 아닌 insert 문의 횟수 및 실행 시간에 의해 결정됨을 알았다. 이를 좀 더 명확히 분석하기 위해 한 테이블에서 다른 테이블로 레코드를 삽입할 때 서로 다른 조건을 주어 그 실행 시간을 측정하는 실험을 수행하였다. 수행한 실험을 정리하면 [표 1]과 같다.

[표 1] 성능 분석 실험 내용

실 험	내 용	비 교
실험1	한 개의 SQL 문으로 테이블 복사	①, ②, ③
실험2	정렬 후 한 개의 SQL 문으로 테이블을 복사	
실험3	한 투플씩 대상 테이블에 삽입(커서 사용)	

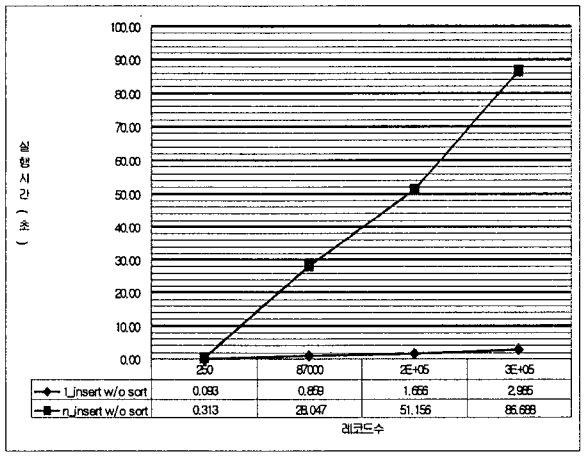
[그림 5]는 정렬 연산을 수행하는 경우 정렬을 위한 시간이 추가로 요구됨을 관찰한 실험으로 첫 번째 실험

과 두 번째 실험의 실행 시간을 비교한 것이다.



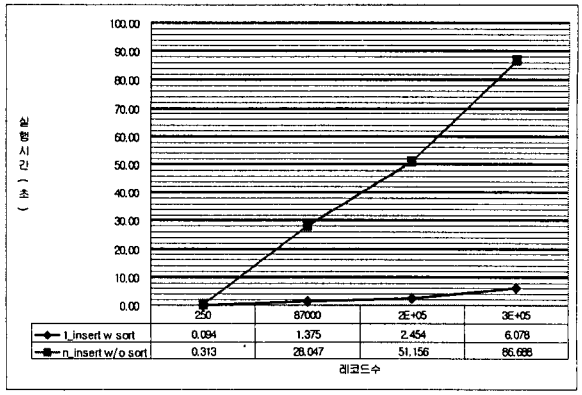
[그림 5] 정렬에 의한 실행 시간의 차이 측정 실험

[그림 6]은 투플 단위의 삽입의 경우 테이블 단위의 삽입에 비해 매우 큰 실행 시간의 차이가 보임을 관찰한 실험으로, 첫 번째 실험과 세 번째 실험 결과를 비교한 것이다. 다음은 정렬과 insert 문의 실행, 두 경우에서 어떤 요인이 실행 시간에 크게 영향을 미치는 지 알아보기 위해 정렬한 후 한 개의 insert 문으로 테이블에 삽입하는 두 번째 실험과, 정렬없이 투플의 개수만큼 insert 문을 실행하여 테이블에 삽입하는 세 번째 실험의 결과를 비교하였다. [그림 7]은 그 결과를 보이고 있는데 그래프에서 보는 바와 같이 RDBMS를 사용한 전위 방식의 OLAP 큐브 생성은 정렬에 의한 비용보다 isnert 문 횟수의 인한 비용의 오버헤드가 성능에 큰 영향을 미침을 알 수 있다.



[그림 6] Insert 문의 횟수에 의한 실행 시간의 차이 측정 실험

이 실험의 결과로 큐브 생성 알고리즘의 효율을 높이기 위해서는 테이블의 정렬 비용을 줄이는 것보다 큐브의 각 집계 테이블에 집계 결과 투플을 삽입하는 insert 문의 호출 횟수를 줄이는 것이 높은 성능을 얻을 수 있는 방법임을 알았다.



[그림 7] 정렬 비용과 insert 문의 실행 비용이 성능에 미치는 영향

5. 결론 및 향후 연구

본 연구에서는 가장 많이 인용되고 연구되고 있는 큐브 생성 방법을 전위 방식으로 관계 DBMS 상에서 구현하고, 실제 데이터 집합으로 큐브를 생성하였다. 구현한 큐브 생성기에 대해 다양한 관점에서 실험을 실시하여 기존 알고리즘의 성능 분석 결과와 다른 결과를 얻었으며, 그 원인을 밝혔다. 전위 방식의 큐브 생성기로 큐브를 생성하고 저장할 때 소요되는 전체 실행 시간은 소스 테이블을 정렬하는데 드는 비용이 아니라, 집계 테이블을 생성하기 위해 실행하는 insert 문의 실행 횟수 (즉, 이를 실행하기 위해 소요되는 DBMS와의 통신 오버헤드)임을 실험을 통해 입증하였다.

향후 연구과제로는 insert 문의 실행과 테이블의 정렬 비용을 최소화함으로써 큐브 생성 효율을 보다 높일 수 있는 방법의 연구가 필요할 것이다.

참고문헌

[1] Pilot Software, "An Introduction to OLAP : Multidimensional Terminology and Technology," http://www.pilotsw.com/olap.olap.htm.

[2] OLAP Counsil, http://www.olapreport.com/DatabaseExplosion.htm.

[3] Sameet Agarwal, Rakesh Agrawal, Prasad M. Deshpande, Ashish Gupta, Jeffrey F, Naughton, Raghu Ramacrishnan, and Sunita Sarawagi, "On the Computation of Multidimensional Aggregates," Proc. of VLDB Conference, pp. 506-521, 1996.