데이터 웨어하우징 환경에서 질의 처리를 위한 새로운 기법

김윤호[U*] 김진호[**] 김상욱[*]
[*] 강원대학교 컴퓨터정보통신공학부,
[**] 강원대학교 전자계산학과
anna95@dreamwiz.com, {jhkim, wook}@cc.kangwon.ac.kr

A New Method for Processing Queries in Data Warehouse Environment

Yun-Ho Kim[U*] Jin-Ho Kim[**] Sang-Wook Kim[*]
[U*] Dept. of Computer, Information and Telecommunication, Kangwon University
[**] Dept. of Computer Science, Kangwon University

요 약

대용량의 데이터가 저장되는 데이터 웨어하우징 환경에서는 조인이나 집계 함수와 같은 고비용의 연산의 효율적인 처리는 매우 중요하다. 본 논문에서는 집계 함수(aggregate function)와 조인이 모두 포함된 질의를 처리하는 새로운 기법을 제안한다. 제안하는 기법은 먼저 차원 테이블(dimension table)을 미리 그룹핑한 후, 비트맵 조인 인덱스(bitmap join index)를 이용하여 조인을 처리하는 방식을 사용한다. 이 결과, 사실 테이블만을 접근하여 집계 함수를 처리함으로써 기존 기법이 가지는 성능 저하의 문제점을 해결할 수 있다. 기존 기법과 제안하는 기법에 대한 비용 모델(cost model)을 정립하고, 이를 기반으로 시뮬레이션을 수행함으로써 제안된 기법의 우수성을 규명한다.

1. 서론[1)]

데이터 웨어하우스(data warehouse)에는 매우 많은 데이터가 저장되므로 의사결정을 위한 복잡하고 분석적인 질의들의 처리를 위하여 많은 데이터에 접근하여야 한다[Inm96]. 따라서 데이터 웨어하우스에서 사용자의 요구를 빠르게 처리할 수 있는 기법의 개발은 중요한 의미를 갖는다[GHQ95][YL95][CS96]. 특히, 데이터 웨어하우스 환경에서 의사결정시 빈번하게 사용되는 집계 함수(aggregate function) 및 조인은 기본적인 관계 연산인 실렉트, 프로젝트 연산과는 달리 질의처리 비용이 매우 크므로 이에 대한 연구는 매우 중요하다.

본 논문에서는 비트맵 조인 인덱스(bitmap join index)[OQ97][C198][WB98][WU99]를 이용하여 집계 함수와 조인을 포함하는 질의를 효과적으로 처리하기 위한 새로운 기법을 제안하고자 한다. 제안하는 기법은 조인을 처리하기 전에 차원 테이블을 미리 그룹핑한 후 비트맵 조인 인덱스를 이용하여 조인을 처리함으로써 사실 테이블만을 검색하여 집계함수를 처리할 수 있는 장점을 갖는다. 정량적인 성능분석을 위하여 기존의 기법과 제안하는 기법에 대한 비용모델을 정립하고 시뮬레이션을 수행함으로써 기존의 방법보다 적은 수의 디스크 액세스만으로 질의를 처리할 수 있음을 보였다.

2. 데이터 웨어하우징 환경의 연구 배경

2.1. 스타 스키마

다차원 데이터 모델(multidimensional data model)은 데이터 웨어하우스 환경에 적합한 모델링 기법으로 알려져 있다[Kim96]. 다차원 모델은 사용자의 관점에서 중요한 데이터를 포함하는 테이블을 스키마의 중심에 놓고, 이를 설명하기 위한 테이블을 주변에 배치한다. 중심에 있는 테이블을 사실 테이블(fact table)이라고 하고, 주변에 있는 테이블을 차원 테이블(dimension table)이라고 한다. 이러한 형태의 스키마는 별 모양을 가지므로 스타 스키마(star schema)라고 부른다. 일반적으로 사실 테이블은 차원 테이블에 비해 그 크기가 매우 크다. 그리고 사실 테이블의 기본 키(primary key)는 모든 차원 테이블의 기본 키들의 결합된 형태로 구성된다. 이러한 결합 키로 인하여 사실 테이블과 차원 테이블간의 조인은 질의 처리시 매우 빈번하게 발생된다.

2.2. OLAP 질의

데이터 웨어하우징 환경에서 널리 사용되는 OLAP 질의에서는 조인 연산이 빈번하게 사용된다. 이러한 조인은 차원 테이블의 기본 키와 사실 테이블의 외래 키를 주 대상으로 수행된다[OQ97]. 데이터 웨어하우징 환경에서는 사실 테이블과 차원 테이블의 크기가 매우 크므로 조인 처리비용이 매우 크다. 또한, OLAP 질의는 분석적인 질의를 처리하기 위하여 집계함수를 빈번하게 사용한다. 이러한 집계함수는 사실 테이블과 차원 테이블간의 조인 후에 처리되어지며, 집계 속성은 대부분의 경우 사실 테이블에 있는 속성 값이 된다.

3. 기존의 접근 방법

3.1. 비트맵 조인 인덱스

비트맵 조인 인덱스(bitmap join index)는 데이터 웨어하우스에서 많이 사용되는 비트맵 인덱스(bitmap index)를 조인 비용을 줄이기 위하여 응용한 인덱스이며, 데이터 웨어하우스와 같은 갱신이 거의 일어나지 않는 환경에 적합하다[OQ95]. 특히, 스타 스키마의 경우 차원 테이블의 크기는 사실 테이블에 비해 매우 작으므로 사실 테이블에 존재하는 각 차원의 조인 속성에 대해서 비트맵 인덱스를 사용하면 일반적인 RID(record identifier)를 이용한 인덱스보다 훨씬 적은 공간을 차지하게

1) 본 연구는 첨단 정보기술 연구 센터를 통한 한국과학재단의 지원과 한국과학재단 99 해외 Post-Doc 방문 연구 프로그램의 지원을 받았습니다.

된다. 또한, AND, OR, NOT과 같은 논리 연산에 대해서도 각 비트맵 인덱스에 대해 비트 연산을 수행함으로써 그 결과를 빠르게 얻을 수 있다. 따라서 비트맵 조인 인덱스는 비트맵 인덱스의 장점을 이용하여 스타 스키마에서 조인을 효율적으로 처리한다.

비트맵 조인 인덱스는 사실 테이블의 각 RID에 대해서 해당되는 차원 테이블의 기본키 값을 비트맵 인덱스의 형태로 표현을 한다. 비트맵 조인 인덱스는 이차원 배열의 형태로서 표현이 되며, 이차원 배열의 행은 사실 테이블의 RID 리스트이고 열은 차원 테이블의 기본 속성 값이 된다. 만약, 이차원 배열에서 행과 열이 교차하는 부분이 1로 저장되어 있으면 그 위치에 해당하는 사실 테이블의 한 레코드와 차원 테이블의 그 기본 속성 값을 갖는 레코드가 조인된다는 의미이다. 각 행과 열이 각각 사실 테이블과 차원 테이블의 어느 레코드와 대응되는 지에 대한 정보는 시스템 카탈로그에 별도로 저장된다.

3.2. 비트맵 조인 인덱스를 이용한 기존의 처리 기법

비트맵 조인 인덱스는 조인의 처리 성능을 효과적으로 하기 위한 인덱스이므로 이 인덱스를 이용하여 집계함수를 처리하는 방법에 대한 연구는 진행되어 있지 않다. 그러나 일반적인 데이터베이스 환경에서 사용하는 집계함수 처리 방법을 적용하여 사용자의 질의에 포함되어 있는 집계함수를 처리할 수 있다 [Gra93][Mur92].

사용자의 질의에 차원 테이블과 사실 테이블간의 조인과 SUM, MAX, MIN, COUNT, AVG와 같은 집계함수가 포함되어 있는 경우의 처리는 다음과 같이 수행된다. 먼저, 비트맵 조인 인덱스를 이용하여 조인 연산을 처리하여 최종 결과를 임시 테이블에 저장한다. 또한, 집계함수를 처리하기 위해서 먼저 SQL의 GROUP BY절에 포함되어 있는 속성 값에 의해 그룹핑을 한다. 그룹핑을 하기 위해서는 먼저 임시 테이블을 해당되는 그룹핑 속성 값으로 정렬하고 그 속성 값에 따라 그룹핑을 한다. 끝으로, 집계함수를 처리하기 위해서 그룹핑되어 있는 임시 테이블의 레코드를 검색한다.

4. 제안하는 방법

집계함수와 조인이 포함된 질의를 제 3장에서 설명한 기존의 방법을 이용하여 처리하는 경우 성능저하가 심각하다. 그 이유는 스타 스키마 환경에서 사실 테이블의 크기는 차원 테이블에 비해 상당히 크므로 집계 함수를 처리하기 위해서 조인 후의 임시 테이블을 정렬하는 과정은 매우 큰 처리비용을 요구하기 때문이다. 이러한 정렬은 외부정렬에 해당되므로 디스크 액세스 수는 임시 테이블의 크기에 비례하게 되고 조인 후의 임시 테이블은 사용자의 질의에서 차원 테이블에 대한 질의 조건이 존재하지 않는 경우에는 사실 테이블의 크기보다도 커지게 된다.

또한, 임시 테이블을 그룹핑 속성으로 정렬 한 후에 사용자 질의에 포함된 집계 함수를 처리하기 위해서는 임시 테이블을 한 번 검색해야 한다. 이 비용 또한 사실 테이블보다 크기가 커진 임시 테이블을 검색함으로써 사실 테이블을 검색하는 디스크 액세스 수보다 더 많은 비용을 요구하게 된다.

기존 방법의 근본적인 문제점은 조인의 결과로서 큰 임시 테이블을 만들고 그 테이블을 그룹핑하기 위해서 외부정렬을 수행함으로써 많은 처리비용을 유발한다는 것이다. 이러한 문제점을 해결하기 위하여 본 연구에서는 임시 테이블보다 상대적으로 크기가 작은 차원 테이블을 정렬을 통하여 미리 그룹핑하고, 이를 기반으로 조인을 수행함으로써 디스크 액세스 수를 줄일 수 있는 방법을 제안하고자 한다.

OLAP 질의에서 집계 속성은 사실 테이블의 한 속성이 되며, 그룹핑 속성은 차원 테이블의 한 속성이 된다. 따라서 기존의 방법에서는 조인 결과를 저장하는 임시 테이블에서 차원 테이블의 그룹핑 속성에 대해서 그룹핑을 하고, 그룹핑 결과로부터 다시 사실 테이블의 집계 속성에 대해서 집계함수 연산을 처리한다. 그러나 제안하는 방법에서는 차원 테이블에 대해서 그룹핑 속성이 존재하므로 조인을 처리하기 전에 크기가 작은 차원 테이블을 미리 그룹핑하여 처리함으로써 작은 처리비용을 가지고 임시 테이블을 그룹핑하는 것과 동일한 결과를 얻을 수 있다.

1. 차원 테이블을 탐색하여 선택조건을 만족하는 레코드들을 임시 테이블에 저장한다.
2. 임시 테이블을 그룹핑 속성으로 정렬하여 그룹핑한다.
3. 비트맵 조인 인덱스를 검색하여 각 그룹마다 하나의 비트맵을 얻는다.
4. 각 그룹에 대해서, 비트맵에서 1로 저장된 위치에 있는 사실 테이블의 레코드들을 검색하여 집계함수를 계산한다.

<그림 1> 제안하는 방법의 처리 알고리즘.

<그림 1>는 사용자의 OLAP 질의가 사실 테이블과 차원 테이블의 조인뿐만 아니라 사실 테이블의 속성 값에 대하여 집계함수 연산을 포함한 경우의 처리 알고리즘을 개략적으로 나타낸 것이다. 단계 4에서, 각 그룹마다 얻어진 비트맵에서 어떤 한 비트가 1로 저장되어 있다는 것은 그것에 대응되는 사실 테이블의 한 레코드가 그 그룹에 속한다는 것을 의미한다. 따라서 실제적인 집계함수의 계산은 각 그룹마다 얻은 비트맵을 조사하여 1로 저장되어 있는 것과 대응되는 사실 테이블의 레코드를 검색하여 처리를 한다. 그러나 COUNT 집계함수인 경우에는 각 그룹마다 얻은 비트맵에서 1로 저장되어 있는 비트 수를 계산함으로써 사실 테이블을 검색하는 처리비용을 제거 할 수 있다.

5. 성능평가

5.1. 비용모델

데이터 웨어하우스에서 질의 처리의 주요 비용은 디스크 액세스 시에 발생하므로 본 논문에서는 디스크 액세스 수를 기반으로 성능 평가를 수행하고자 한다. 먼저, 디스크 액세스 수를 분석하기 위하여 필요한 인자를 정의하고, 기존의 방법과 제안하는 방법의 디스크 액세스 수를 추정한다.

인 자	인자의 의미
n_d	차원 테이블의 레코드 수
n_f	사실 테이블의 레코드 수
s_d	차원 테이블의 레코드 크기
s_f	사실 테이블의 레코드 크기
b	디스크 블록의 크기
r_{sel}	선택률
r_{parti}	조인 참여 비율
k	외부 정렬 시 사용될 디스크 블록 수

<표 1> 비용모델에서 사용되는 인자들.

다음은 기존의 방법과 제안하는 방법의 디스크 액세스 수를 추정하기 위한 비용모델에 대해서 설명한다. 우선, 사용자의 질의에 선택조건이 포함되어 있는 경우에는 차원 테이블에 선택률(selectivity)을 적용하여야 한다. 그러나 기존의 방법이나 제안하는 방법에서 첫 단계로서 공통적으로 필요한 부분이므로 본 연구에서는 질의의 선택조건을 만족하는 레코드를 찾기 위한 처리비용은 고려하지 않는다. <표 1>은 디스크 액세스 수를 분석하기 위하여 필요한 인자들을 정리한 것이다.

비트맵 조인 인덱스를 이용하여 기존의 방법으로 질의를 처리할 때 발생하는 총 디스크 액세스 수는 다음과 같다.

$$\text{Total Cost} = \lceil (r_{sel} \times r_{parti} \times n_d) \rceil \times \lceil \frac{n_f}{8 \times b} \rceil + (n_f \times r_{sel}) + \lceil \frac{(s_d + s_f) \times (n_f \times r_{sel})}{b} \rceil + 2 \times \lceil \frac{(s_d + s_f) \times (n_f \times r_{sel})}{b} \rceil \times \log_k \lceil \frac{(s_d + s_f) \times (n_f \times r_{sel})}{b} \rceil + \lceil \frac{(s_d + s_f) \times (n_f \times r_{sel})}{b} \rceil$$

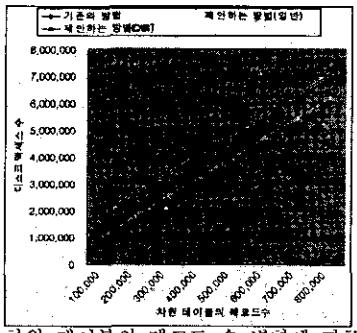
비트맵 조인 인덱스를 이용하여 제안하는 방법으로 질의를 처리할 때 발생하는 총 디스크 액세스 수는 다음과 같다.

$$\text{Total Cost} = \lceil \frac{s_d \times (n_d \times r_{sel})}{b} \rceil + 2 \times \lceil \frac{s_d \times (n_d \times r_{sel})}{b} \rceil \times \log_k \lceil \frac{s_d \times (n_d \times r_{sel})}{b} \rceil + \lceil (r_{sel} \times r_{parti} \times n_d) \rceil \times \lceil \frac{n_f}{8 \times b} \rceil + (n_f \times r_{sel})$$

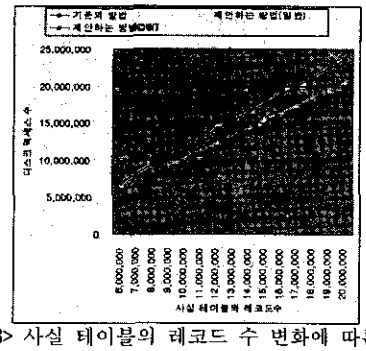
5.2. 시뮬레이션 결과

본 시뮬레이션은 TPC-H 벤치마크[TPC99]에서 사용하는 스타 스키마 환경을 참조하여 수행하였다. TPC-H 벤치마크는 대량의 데이터들에 대한 복잡하고 긴 질의들을 필요로 하는 데이터 웨어하우스 시스템과 의사 결정 지원 시스템의 처리 능력을 테스트하는 벤치마크이다.

시뮬레이션을 위하여 차원 테이블의 레코드 크기와 사실 테이블의 레코드 크기는 각각 112와 144로 지정하였다. 그리고 차원 테이블의 레코드 수와 사실 테이블의 레코드 수는 각각 800,000과 6,000,000으로 지정하였다. 그리고 선택률과 참여율은 0.05와 0.8로 지정하였으며 시스템 버퍼의 수와 디스크 블록의 크기는 100과 4K로 지정하였다.



<그림 2> 차원 테이블의 레코드 수 변화에 따른 성능변화.



<그림 3> 사실 테이블의 레코드 수 변화에 따른 성능변화.

첫 번째 시뮬레이션에서는 차원 테이블의 레코드 수를 100,000부터 800,000까지 변화시킴으로써 사실 테이블에 비해 차원 테이블의 레코드 수가 상대적으로 작아지는 경우의 성능상 경향을 관찰하였다. <그림 2>는 이 시뮬레이션 결과를 나타낸 것이다. 결과에 의하면, 제안된 기법은 기존 기법과 비교하여 모든 경우에 나은 성능을 보였으며, 특히 집계함수가 COUNT인 경우에는 기존 방법의 38%의 처리비용으로 질의를 처리하는 것으로 나타났다.

두 번째 시뮬레이션에서는 사실 테이블의 레코드 수를 6,000,000부터 20,000,000까지 변화시킴으로써 사실 테이블의 레코드 수가 차원 테이블의 레코드 수에 비해 상대적으로 커지는 경우의 성능 변화를 관찰하였다. <그림 3>은 이 시뮬레이션 결과를 나타낸 것이다. 결과에 의하면, 사실 테이블의 레코드 수가 증가하더라도 제안하는 기법은 모든 경우에 나은 성능을 보였으며 특히, 집계함수가 COUNT인 경우에는 기존 방법의 82%의 처리비용으로 질의를 처리할 수 있었다.

6. 결론

본 논문에서는 차원 테이블에 대하여 먼저 그룹핑을 적용함으로써 조인 후의 결과 테이블을 정렬하기 위한 디스크 액세스 수를 줄이는 방법을 제안하였다. 제안된 기법은 스타 스키마 환경에서 집계함수에 대한 그룹핑 속성이 차원 테이블에 존재하고 집계 속성이 사실 테이블에 존재한다는 특성을 이용하여 테이블의 레코드 수가 작은 차원 테이블을 먼저 그룹핑한다. 이러한 기법은 조인을 효과적으로 처리하기 위한 인덱스를 사용하여 집계함수도 효과적으로 처리가 가능하다.

제안된 기법의 성능을 평가하기 위하여 대표적인 데이터웨어하우스 벤치마크인 TPC-H의 스타 스키마를 기반으로 기존의 기법과 시뮬레이션을 통한 성능 비교를 수행하였다. 시뮬레이션 결과에 의하면, 제안된 기법은 기존의 기법과 비교하여 모든 경우에서 나은 성능을 보였다. 본 연구 결과는 데이터 웨어하우징 환경에서 OLAP 질의의 처리 과정에 효과적으로 적용될 수 있다.

7. 참고 문헌

[CI98] C.Y. Chan and Y.E. Ioannidis, "Bitmap Index Design and Evaluation," In Proc. Int'l. Conf. on Management of Data, ACM SIGMOD, pp. 355-366, Seattle, Washington, USA, June 1998.

[CS96] S. Chaudhuri and K. Shim, "Optimizing Queries with Aggregate Views," In Proc. Int'l. Conf. on Extending Database Technology, pp. 167-182, Avignon, France, March 1996.

[GHQ95] A. Gupta, V. Harinarayan, and D. Quass, "Aggregate-Query Processing in Data Warehousing Environments," In Proc. Int'l. Conf. on Very Large Data Bases, pp. 358-369, Zurich, Switzerland, September 1995.

[Inm96] W.H. Inmon, Building the Data Warehouse, John Wiley & Sons, March 1996.

[Kim96] R. Kimball, The Data Warehouse Toolkit, John Wiley & Sons, 1996.

[OQ97] P. O'Neil and D. Quass, "Improved Query Performance with Variant Indexes," In Proc. Int'l. Conf. on Management of Data, ACM SIGMOD, pp. 38-49, Tucson, Arizona, USA, May 1997.

[TPC99] Transaction Processing Performance Council(TPC), TPC Benchmark H (Decision Support), Standard Specification Revision 1.2.1, 1999.

[WB98] M.C. Wu and A. Buchmann, "Encoded Bitmap Indexing for Data Warehouses," In Proc. Int'l. Conf. on Data Engineering, IEEE, pp. 220-230, Orlando, Florida, USA, February 1998.

[Wu99] M.C. Wu, "Query Optimization for Selections using Bitmaps," In Proc. Int'l. Conf. on Management of Data, ACM SIGMOD, pp. 227-238, Philadephia, Pennsylvania, USA, June 1999.

[YL95] W.P. Yan and P.A. Larson, "Eager Aggregation and Lazy Aggregation," In Proc. Int'l. Conf. on Very Large Data Bases, pp. 345-357, Zurich, Switzerland, September 1995.