빙산 질의와 빙산 순위 질의의 효율적인 처리를 위한 저장뷰

홍석진 ', 배진욱, 심마로, 이석호
{jinny, oblody, maro}@db.snu.ac.kr, shlee@comp.snu.ac.kr
서울대학교 컴퓨터공학과

Materialized View for Computing Iceberg Query and Iceberg Ranking Query Efficiently

Seokjin Hong ', Jinwook Bae, Maro Shim, Sukho Lee
Dept. of Computer Engineering, Seoul National University

요 약

빙산 질의란 대용량의 데이타들에 대해 집단 함수를 수행하여 임계값 이상인 데이타들을 결과로 반환하는 연산을 의미한다. 빙산 질의는 도메인의 크기가 대단히 큰 대용량의 데이타에 대해 적용되므로 집단 함수의 수행을 위한 카운터를 전부 메모리에 적재할 수 없는 상황이 발생한다. 지난 연구[1]에서는 이러한 빙산 질의를 표본 추출과 해싱을 사용하여 수행하였다. 하지만 많은 수행시간을 필요로 하는 질의를 대용량의 데이타 전체에 대해 매번 수행하여야 하며, 결과를 얻기 위한 후보 수가 커질 수 있다는 문제점이 있다. 이 논문에서는 빙산 질의에 대한 저장뷰를 통해, 사용자의 질의 중 많은 부분을 차지하는 큰 임계값으로 구성된 질의에 대해서는 즉각적인 결과를 돌려주고, 빙산 질의 결과 후보의 수를 감소시키며, 별도의 표본 추출 과정을 생략할 수 있는 방법을 제시한다. 또한 빙산 질의 히스토그램을 통해 빙산 순위 질의를 수행하는 방법을 제시하여 사용자로 하여금 보다 직관적인 질의를 작성할 수 있도록 하였다.

1. 서론

다차원 관계 데이타베이스에서의 group-by를 통한 집단 함수 연산은 OLAP 환경에서 중요하게 사용되는 연산 중의 하나이다. 하지만 이 때, 많은 경우에 있어서 사용자는 질의 결과 중 전체 보다는 어떤 임계값을 넘어가는 데이타에 대해 주로 관심을 갖게 된다. 이처럼 대용량의 데이타들에 대해 group-by를 통한 집단 함수를 수행한 뒤 임계값 이상인 데이타들을 결과로 반환하는 연산을 빙산 질의라 하며 이는 [1]에서 처음으로 제기되었다.

이러한 빙산 질의는 다차원, 대용량의 데이타를 다루는 데이타웨어하우스[2, 3]와 데이타마이닝[4] 등의 분야에서 자주 사용된다.

빙산 질의의 기본적인 형태는 다음 SQL문과 같다.

```
SELECT target_1, target_2, ..., target_k, count(rest1, rest2, ...)
FROM R
GROUP BY target_1, target_2, ..., target_k
HAVING count(rest1, rest2, ...) >= T
```

테이블 R에서 애트리뷰트 target_1, target_2, ..., target_k를 group-by를 위한 차원 애트리뷰트라 하고, 차원 애트리뷰트의 실제 각 레코드 값들을 차원 벡터라 한다. 그리고 나머지 애트리뷰트 rest1, rest2, ... 는 실제 차원 벡터에 대한 값을 명시하는 값 애트리뷰트가 된다. 이러한 R과 임계값 T가 주어졌을 때 빙산 질의는 차원 벡터 각각의 값들에 대한 카운트나 합 또는 평균을 계산한 후 임계값 이상인 것들만 반환하게 된다.

이러한 빙산 질의를 처리하기 위한 가장 기본적인 방법은 각 차원 벡터마다 하나의 카운터를 메모리에 생성한 후 데이타베이스를 스캔하면서 집단 함수를 수행하는 것이다. 하지만 차원이 커지고, 도메인의 크기 또한 대단히 커지게 되는 데이타웨어하우스 환경에서는 이러한 카운터를 메모리에 전부 유지할 수 없는 상황이 발생한다. [1]에서는 이를 표본 추출과 해싱의 방법을 통해 해결하고 있다.

이 논문에서는 집단 함수의 결과가 특정 임계값 이상인 데이타들로 저장뷰[5]를 구성하고, 구성된 저장뷰를 통해 빙산 질의를 효율적으로 수행하는 방법을 제안한다. 한편 기존의 빙산 질의의 경우 사용자가 어느 정도의 임계값을 예측하여 질의를 하여야 하는데, 실제 사용자는 기본 데이타의 분포를 정확히 알 수 없으므로 임계값을 예측하여 질의를 만들기는 어렵다. 따라서 순위를 통해 임계값을 지정하는 빙산 순위 질의가 필요하며, 이 논문에서는 빙산 질의 히스토그램을 통해 빙산 순위 질의를 수행하는 방법을 제시한다.

이 논문의 구성은 다음과 같다. 2절에서는 [1]에서 제시한 빙산 질의를 수행하는 방법을 살펴본다. 3절에서는 빙산 질의를 위한 저장뷰를 소개하고, 저장뷰를 통한 빙산 질의와 빙산

순위 질의의 수행 방법을 알아보며 마지막으로 저장뷰의 갱신에 대해 언급한다. 4절에서 실험 결과를 살펴보고, 5절에서 결론을 맺는다.

2. 관련 연구

[1]에서 제시한 빙산 질의 처리 알고리즘의 기본적인 아이디어는 카운터의 개수가 매우 많아져서 메모리 내에 카운터를 전부 적재할 수 없는 경우에 해싱을 통한 카운터의 공유를 하겠다는 것이다. 하지만 해싱만을 사용하는 경우 다음 두 가지의 문제점이 나타나게 된다. 첫째, 한 카운터 버킷 내에 임계값을 넘는 데이타와 임계값을 넘지 않는 데이타가 함께 들어감으로 인해 후보의 개수가 많아질 수가 있다. 이를 해결하기 위하여 [1]에서는 표본 추출[8]과 해싱[7]을 함께 사용한다. 먼저 기본 데이타베이스에서 표본을 추출하여 임계값이 넘어가는 후보들을 선정하고 그 후보들에 대해서는 별도의 카운터를 통해 개수를 세어, 후보의 개수를 줄인다. 최종적으로 데이터베이스를 스캔하면서 그 후보들에 대한 카운팅을 하여 결과를 얻는다. 이를 위한 기법으로 [1]에서는 Defer Count, Multi Level, Multi Stage의 세 가지 방법을 제시하고 있다. 둘째, 임계값을 넘지 않는 데이타들이 여러 개 모여 임계값을 넘는 경우로, 이 역시 후보 개수를 증가시키는 문제점을 갖고 있다. [1]에서는 이 문제를 여러 개의 해시함수를 사용하여 해결하였다.

그러나, [1]에서 제시한 방법은 후보의 크기가 커질 수 있고, 표본 추출의 과정을 거쳐야 하며, 많은 수행 시간을 필요로 하는 질의를 매번 모든 데이타에 대해 새로 해야 한다는 단점을 갖고 있다.

3. 빙산 질의 저장뷰

3.1. 빙산 질의 저장뷰

빙산 질의 저장뷰는 대용량 데이타에 대한 집단함수의 결과 중 특정 임계값을 넘는 데이타에 대한 값들을 명시적으로 유지하여 빙산 질의 시에 효과적으로 사용할 수 있도록 한다. 저장뷰를 생성하기 위해서는 차원 애트리뷰트들의 집합과 임계값이 인자로 필요하다. 빙산 질의 저장뷰를 생성하기 위한 SQL문의 구조는 다음과 같다.

```
CREATE VIEW 뷰 이름 AS
SELECT target_1, target_2, ..., target_k, count(rest)
FROM 테이블 이름
GROUP BY target_1, target_2, ..., target_k
HAVING count(rest) >= 임계값
```

그림 2는 빙산 질의 저장뷰의 예를 보여주고 있다. 전체 집단 함수 결과 중 임계값 83을 넘는 결과들로 구성된다. 이러한 빙산 질의 저장뷰를 통하여 기존의 빙산 질의를 효율적으로 수행할 수 있으며, 빙산 순위 질의의 수행도 가능하게 된다.

3.2. 빙산 질의 저장뷰를 이용한 빙산 질의의 처리

T를 빙산 질의의 임계값이라 하고, count(rest)를 저장뷰에 저장된 차원벡터의 카운트 값이라고 하자. 먼저, 'T ≥ min(count(rest))' 의 경우는 저장뷰 내에서 빙산 질의 결과를 바로 얻어 낼 수 있다. 예를 들어 그림 1과 같은 저장뷰가 있을 경우 임계값이 98인 질의가 들어온다면 결과 (a, b, 100), (a, c, 98)을 저장뷰를 통해 바로 얻을 수 있다.

Rank	target1	target2	count(rest)
1	a	b	100
2	a	c	98
3	b	b	97
...	...	...	...
20	c	f	83

그림 1 빙산 질의 저장뷰

'T < min(count(rest))'의 경우 빙산 질의 저장뷰에서 min(count(rest))까지의 결과를 얻어 사용자에게 바로 돌려줄 수 있으며, 나머지 T부터 min(count(rest))사이의 결과는 기존 빙산 질의 알고리즘을 사용하여 계산하게 된다. 이 때, 기존의 방법과는 달리, 저장뷰에 포함되어 있는 차원벡터를 제외한 나머지 차원 벡터들에 대해서만 카운팅을 하게 된다. 이로 인해, 한 카운터 버킷 내에 임계값을 넘는 데이타와 임계값을 넘지 않는 데이타가 함께 들어가서 후보의 개수가 커지는 해싱의 문제점을 많이 감소시킬 수 있다. 따라서 이 경우 별도의 표본 추출과정이 생략될 수 있으며, 결과로 나온 후보의 수도 줄어들게 된다. 나머지 T부터 min(count(rest))사이의 결과를 위한 빙산 질의 알고리즘은 [1]에서 제시한 해싱기법과 [6]에서 제시한 동적 분할 기법이 모두 적용될 수 있다.

3.3. 빙산 질의 저장뷰를 이용한 빙산 순위 질의의 처리

지금까지 빙산 순위 질의를 처리하기 어려웠던 이유는 빙산 질의의 특성이 전체 데이타에 대한 카운팅을 한 후에 그 중 임계값을 넘는 대상만 고르는 것이 아니라, 알고리즘 수행중에 임계값을 넘을 가능성이 있는 후보들을 선정한 후 후보들을 다시 카운팅 해야 하기 때문이다. 이 때 임계값을 기준으로 후보를 선정하는 것은 수행 중에 가능하지만, 순위를 기준으로 후보를 선정하는 것은 전체 데이타의 분포를 모르기 때문에 불가능하다. 이 논문에서는 이를 위해, 전체 데이타의 분포 정보를 나타내는 빙산 질의 히스토그램을 사용한다.

Rank	target1	target2	count(rest)
10	a	b	92
20	c	f	83
50	k	m	65
100	s	o	50
200	t	l	27
...	...	...	...

그림 2 빙산 질의 히스토그램

빙산 질의 히스토그램은 그림 2와 같이 구성된다. 전체 데이타 항목 중 일부에 대해 순위와 임계값을 미리 계산하여, 순위로 임계값이 주어지는 빙산 순위 질의를 임계값이 카운트 값인 일반 빙산 질의로 바꾸어 계산할 수 있다. 그림 3에서 순위 45까지의 빙산 순위 질의는 순위 50의 카운트 값에 해당하는 임계값 65인 빙산 질의로 변경하여 계산한다.

R을 빙산 질의의 순위라 하고, Rank를 저장뷰에 저장된 순위 값이라고 하자. 이 경우 역시 'R ≤ max(Rank)'의 경우 빙산 질의 저장뷰에서 빙산 순위 질의 결과를 바로 얻어 낼 수 있으며, 'T > max(Rank)'의 경우는 저장뷰로부터 max(Rank)까지의 질의 결과를 얻고, max(Rank)부터 R까지의 결과는 빙산 질의 히스토그램을 사용하여 일반 빙산 질의로 변경한 후 위의 방법대로 얻을 수 있다.

3.4. 빙산 질의 저장뷰의 갱신

이러한 빙산 질의 저장뷰는 데이타의 변화에 따라서 갱신되어야 한다. 빙산 질의의 특성상 데이타는 삭제나 수정되지 않고 추가만이 일어나며, 데이타웨어하우스의 특성상 일정 주기로 데이타가 추가되기 때문에 데이타웨어하우스 갱신 주기마다 뷰의 갱신을 하면 된다.

매 주기마다 새로 들어온 데이타들에 대해 기존 데이타베이스를 한번 스캔함으로써 저장뷰의 갱신이 이루어 진다. 기존의 저장뷰를 기반으로 새로 들어온 데이타들의 카운트를 참고하여 뷰 내의 순위를 재조정하는 방법으로 갱신을 수행한다.

4. 실험 및 분석

먼저 'T ≥ min(count(rest))' 경우는 저장뷰를 통해 바로 결과를 얻을 수 있으므로 'T < min(count(rest))'인 경우에 대해 기존의 방법과 비교를 하였다. 3차원 데이타 1,000,000개에 대해서 저장뷰가 없는 상황과 저장뷰가 있을 경우에 후보의 개수를 비교하였다. 저장 뷰를 사용하는 경우 표본 추출 과정 없이 수행하였다. 실험에 사용된 기계는 Ultra Sparc II 333MHz 이며, OS는 Solaris 7이다. 먼저 그림 3은 메모리 크기에 따른 후보 개수의 변화를 나타내고 있다. 실험에 사용된 임계값은 56이며, 저장뷰는 임계값 68 이상인 결과를 저장하고 있다. 메모리 크기가 커질수록 한 버킷 내에서 카운팅 되는 차원벡터의 수가 줄어들기 때문에 후보의 개수는 감소하며, 전반적으로 저장뷰가 있을 경우 저장뷰가 없는 경우 보다 후보의 개수가 적게 나타나는 것을 알 수 있다.

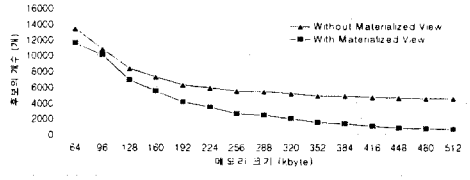

그림 3 메모리 크기에 따른 후보 개수의 변화

그림 4는 임계값에 따른 후보 개수의 변화를 나타내고 있다. 사용된 메모리 크기는 274.62 kbyte이며 저장뷰에서는 임계값 70 이상인 결과를 저장하고 있다. 역시 임계값이 커질수록 후보의 개수가 줄어들고 있으며, 전반적으로 저장뷰를 통해 후보의 개수가 감소된 것을 알 수 있다.

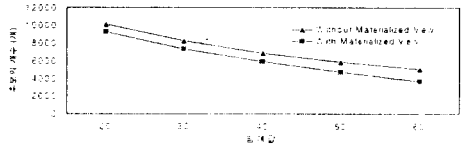

그림 4 임계값에 따른 후보 개수의 변화

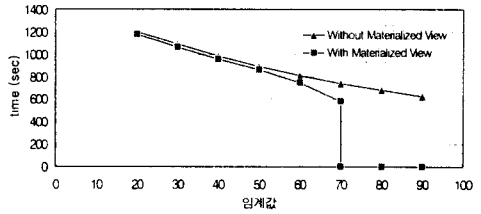

그림 5 임계값에 따른 수행 시간의 변화

그림 5는 임계값에 따른 수행 시간의 변화를 보여준다. 임계값 70이상의 결과는 저장뷰를 통해 바로 돌려주기 때문에 수행시간이 0에 가까워지며, 임계값 70이하에서도 저장뷰가 없는 상황보다는 빠른 수행시간을 보인다.

5. 결론

이 논문에서는 빙산 질의의 저장뷰를 통해 빙산 질의를 효율적으로 수행하는 방법을 제시하였다. 저장뷰를 통해 빙산 질의 수행 시 후보의 개수를 효과적으로 줄일 수 있으며, 별도의 표본 추출 없이 빙산 질의를 수행할 수 있다. 그리고 사용자의 질의중 많은 부분을 차지하는, 큰 임계값으로 구성된 질의에 대해서는 결과를 즉시 돌려줄 수 있으며, 그렇지 않은 경우에도 저장뷰를 통해 결과의 일부를 사용자에게 빨리 돌려줄 수 있기 때문에 연산 수행에 많은 시간이 걸리는 데이타웨어하우스 환경에서 보다 효율적으로 사용될 수 있을 것이다.

또한 빙산 순위 질의를 빙산 질의로 변경하여 수행하는 방법을 제시하여 사용자가 보다 직관적인 질의를 작성할 수 있도록 하였다.

참고 문헌

[1] M. Fang, N. Shivakumar, H. G. Molina, R. Motwani, J. D. Ullman, "Computing iceberg queries efficiently", VLDB 1998

[2] S. Chaudhuri and U. Dayal. "An overview of data warehousing and OLAP technology", ACM SIGMOD Record 26(1), 1997

[3] M.C. Wu, A.P. Buchmann. "Research Issues in Data Warehousing", BTW'97, Ulm. March, 1997

[4] A. Savasere, E. Omiecinski, and S. Navathe, "An efficient algorithm for mining association rules in large databases", VLDB, 1995

[5] N. Roussopoulos. "Materialized Views and Data Warehouses", SIGMOD Record. 27(1), p21-26. March 1998.

[6] 배진욱, 이석호. "빙산 질의 처리를 위한 동적 분할", '99 봄 학술발표논문집(B), 제26권 1호, pp.164-166. 1999.4

[7] J.S. park, M.S. Chen, and P.S. Yu. "An effective hash based algorithm for mining association rules", In Proceedings of ACM SIGMOD Conference. 1995

[8] F. Olken. "Random sampling from databases", PhD thesis, University of California, Berkeley, 1993