Shared nothing 구조하에서 Chunk-based Caching 전략에 의한 OLAP Query response time의 속도향상.

엄준식* 정병수
경희대학교 전자계산공학과

Speed Up of OLAP Query response time By Chunk-based Caching Scheme In Shared nothing Structure

Jun Sik Eom*, Byung Soo Jeong
Dept. of Computer Science, Kyunghee University

요 약

현재 의사결정 시스템을 위한 데이터 웨어하우스는 데이터베이스 분야에서 비약적인 발전을 해오고 있다. 이 분야에서 중요한 사항은 사용자의 질의에 대한 응답시간이다. 사용자로부터 질의가 요구되면 과거의 많은 데이터를 분석하여 적절한 시간 내에 분석결과를 되돌려 줄 수 있어야 하는데 데이터 웨어하우스의 특성상 대용량의 데이터를 저장하고 분석 시에 많은 데이터를 검색해야 하므로 질의 응답시간에 많은 시간이 소요된다. 이 논문에서는 Chunk based caching기법에 의해서 새로이 요구되는 질의에 대해 이미 메모리에 캐시 되어진 내용을 이용하는 방식을 통해 디스크의 I/O 횟수를 줄임으로써 질의 응답시간을 단축시키는 기법을 소개한다. 또한 chunk miss에 대한 처리를 병렬로 수행함으로써 메모리에 캐시되지 않은 내용을 디스크로부터 로드하는 시간을 단축시키는 방법도 아울러 소개한다.

1. 서 론

최근들어 각 조직내에서 의사결정을 위한 데이터베이스의 사용이 급격히 증가하고 있다. 이러한 현상은 데이터 웨어하우스와 OLAP이라는 대용량의 데이터를 저장하고 추출해내는 기술의 결과로써 이 두 기술은 의사결정을 위해서 과거의 데이터에 대한 저장소와 데이터 추출도구를 제공한다. OLTP가 트랜잭션의 처리율에 초점을 맞춘 반면에 OLAP은 의사결정을 지원하고 긴 기간동안 과거의 데이터를 포함하도록 설계되었다.[1]

일반적으로 데이터 웨어하우스에 저장된 데이터는 다차원 성격을 띠고 있어 다양한 측면에서 데이터를 분석할 수 있도록 한다. 기존의 논문들 중에는 자주 검색되어지는 테이블에 대해서 미리 요약된 테이블을 작성하여 사용자의 질의에 대한 응답시간을 단축시키는 방법[2]을 제시하였고 또한 인덱스와 데이터를 하나의 데이터 구조로 저장함으로써 질의 처리율을 높이는 방법[5]과 데이터 웨어하우스에 적합한 인덱스 구조를 제시하여 데이터에 대한 효과적인 검색을 할 수 있는 방안[6][8]을 제시하였다. 이 논문에서는 chunk-based caching기법[3]을 통해 다차원 질의에 대한 응답시간을 단축시키는 전략을 근거로 하여 이 전략을 shared nothing구조로 확장시킴으로써 cache miss에 대한 처리시간을 단축시키는 방법을 소개한다.

2. 관련연구

Caching에 대한 기존의 방법은 질의결과 전체를 caching하는 query level caching이었다. 그러나 이러한 전략은 사용자로부터 요구되어진 새로운 query에 대해 캐시되어진 이전 query의 결과를 이용할 수 있는지의 여부를 판단하기 위해 query들간의 포함관계를 사용하는데 새로운 질의가 이전 질의에 완전히 포함되지 않지만 부분적으로 겹쳐지는 부분이 존재할 때 그러한 부분을 재사용하지 못하고 새로이 질의 결과를 재 계산해야하는 단점을 갖는다. 이러한 단점을 극복하기 위해서 부분적으로 겹쳐지는 부분을 인지하고 재 사용할 수 있도록 하는 메커니즘이 필요한데 이러한 전략을 chunk based caching[3]이라고 한다. Chunk based caching은 다차원 질의공간을 일정한 크기의 chunk들로 나누고 이 chunk들을 caching하는 전략을 갖는다. 따라서 query level caching에 비해 더 작은 caching단위를 제공하므로 새로운 질의는 이전 질의와 겹쳐지는 부분을 재 사용

할 수 있게 된다. 예를 들면 [그림1]에서 query1과 query2의 결과가 이미 cache되어진 상태에서 query3가 사용자로부터 새로이 요구되었을 때 chunk based caching 전략에서는 query3가 query1, query2와 겹쳐지는 부분에 대해서는 재사용하고 겹치지 않는 부분(빗금친 부분)은 디스크로부터 메모리에 cache되어야 한다.

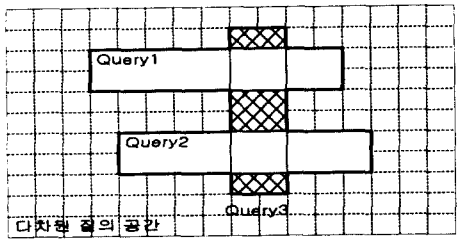

그림 1. Cache되어진 chunk의 재사용

3. Chunk에 의한 Multi-dimensional space의 분할

각각의 차원테이블은 차원이 가진 고유의 계층구조를 반영하여 chunk region과 chunk number를 부여받게 된다. 차원테이블에 대해서 chunk number가 부여되면 사실테이블은 차원테이블의 chunk number에 의해서 chunk number가 부여된다. 일반적으로 다차원 질의공간은 사용자가 요구하는 질의의 group by 리스트에 의해서 질의공간의 차원이 결정되기 때문에 각 질의공간에서 캐시되어진 chunk와 새로이 캐시되어질 chunk를 구분하기 위해서 [그림2]와 같이 다차원 질의공간에 대해서 chunk number를 서로 다르게 부여한다.

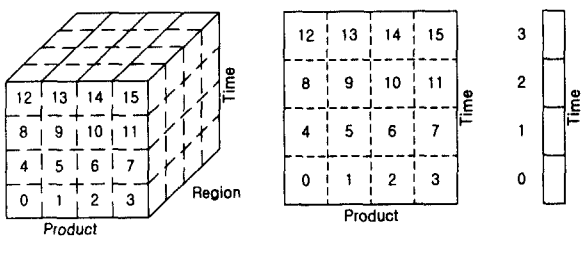

그림 2. Chunk에 의한 다차원 질의공간의 분할

4. Chunk를 이용한 query processing

Chunk based caching에서는 cache에 저장되는 질의결과들은 chunk들로 나뉘어지고 chunk 단위로 cache에 저장된다. 사용자로부터 새로운 질의가 요구되었을 경우에 해당 질의에 응답하기 위해서 필요한 chunk들이 결정된다. 이때 2개의 리스트가 만들어지는데 하나는 cache로부터 응답할 수 있는 chunk들의 리스트이고 다른 하나는 cache에 존재하지 않기 때문에 재 계산되어져야 하는 missing chunk들에 대한 리스트가 된다. 따라서 missing chunk들에 해당하는 부분만을 디스크로부터 읽어들여 cache에 저장한 후 이미 캐시되어진 chunk들과 함께 질의를 처리하게 된다.

5. Chunk number가 추가된 스키마의 구성

이 논문에서는 기존의 일반적인 star-schema 릴레이션에 chunk number를 저장할 attribute를 추가하여 chunk miss에 대한 효율적인 처리를 할 수 있도록 한다. 따라서 query를 처리하는 과정에서 발생하는 chunk miss에 대해서는 해당 릴레이션에서 해당 chunk number에 해당하는 레코드를 디스크로부터 읽어들여 caching하게 된다. 따라서 각 차원테이블에는 chunk number를 저장할 하나의 attribute만 추가되며, 사실테이블에는 발생 가능한 group by 리스트에 대해서 [그림3]과 같이 chunk number를 저장할 attribute들이 추가된다.

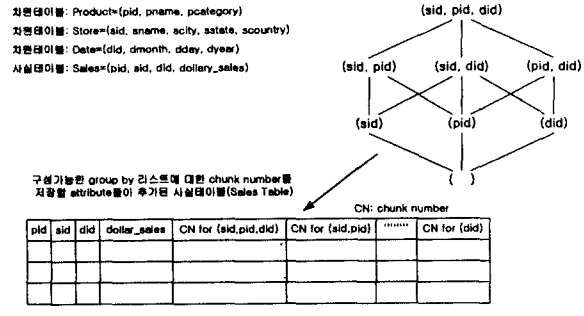

그림 3. 사실테이블(fact table)에 chunk number 할당

6. 병렬처리를 위한 시스템 구성

앞 절에서 기술된 chunk based caching전략을 병렬처리를 통한 속도향상을 위하여 시스템 구성을 3-tier 구조로 구성한다. 먼저 Front-tier에서는 사용자가 GUI환경으로 구성된 OLAP 도구를 통해 질의를 작성할 수 있도록 하며, 작성된 질의를 Middle-tier로 전달하는 역할을 수행한다. Middle-tier에서는 Front-tier에서 보낸 질의를 받아 처리하게 되는데 앞에서 기술된 chunk based caching전략을 이용하여 질의를 처리한다. Backend-tier는 shared-nothing구조로 구성되며 기존의 star-schema 릴레이션을 균등하게 분할하여 저장하게 된다.

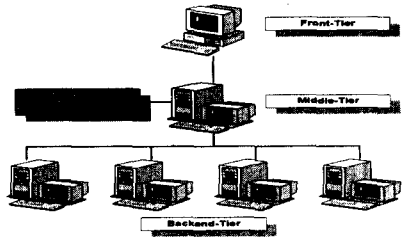

그림 4. 3-Tier구조에 의한 시스템 구성

7. 데이터의 분할과 Query processing

chunk number를 저장할 attribute가 추가된 차원테이블과 사실테이블은 [그림7]에서 처럼 Backend-tier를 구성하는 전체 노드수에 의해서 균등하게 분할되어 각 노드에 저장된다. 이러한 상황에서 사용자로부터 질의가 요구되어지면 middle-tier에서는

질의를 분석하여 2개의 리스트를 작성한다. 작성된 리스트중에서 첫 번째 리스트는 요구된 질의에서 재 사용 가능한 chunk들의 리스트이고 두 번째 리스트는 cache에 저장되어있지 않기 때문에 Backend-tier로부터 캐시되어져야 할 chunk들의 리스트가 된다. 예를 들면 [그림5]와 같은 경우 missing chunk 리스트가 chunk number (9, 10, 11, 15, 16, 18, 19, 24, 25, 27)로 구성되게 된다. Middle-tier에서는 missing chunk 리스트에 해당하는 chunk number들을 [그림7]에서와 같이 Backend-tier의 각 노드에 병렬로 요구한다. Backend-tier에서는 product 차원테이블의 chunk number에 대해 구성된 인덱스를 나타내는 그림[6]과 같이 각 노드에 균등하게 분할되어진 차원테이블과 사실테이블에 대해서 각각 chunk number에 대한 B트리 인덱스를 구성하고 구성된 인덱스를 통해 해당 chunk를 검색하게 된다. 이러한 과정을 통해 리턴 되어진 chunk들은 middle-tier의 cache manager에 의해서 다차원 질의공간에 cache되어지고 이전에 캐시되어진 chunk들의 내용과 함께 사용자에게 질의결과로서 리턴 된다.

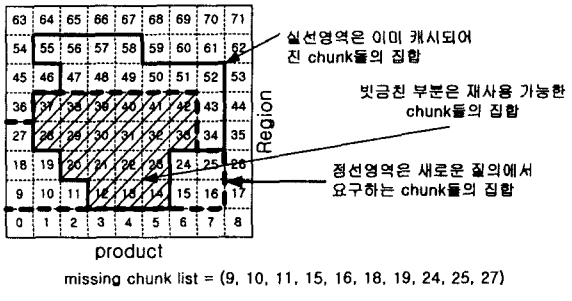

그림 5. missing chunk들에 대한 리스트 구성

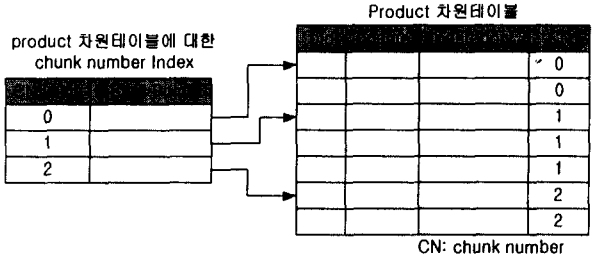

그림 6. product 차원테이블의 chunk number에 대해서 구성된 index

8. 결론 및 향후발전 방향

데이터 웨어하우스 환경에서는 많은 데이터를 저장하고 검색하기 때문에 사용자에 대한 질의 응답속도가 중요한 부분을 차지한다. 이 논문에서는 이러한 목표를 달성하기 위해서 데이터 웨어하우스 환경에 적합한 caching 전략에 병렬처리 기법을 적용하여 효과적인 caching 전략을 통해 사용자의 질의에 대한 응답시간을 줄일 수 있도록 시스템을 구성하였다. 향후 발전계획은 다양한 사용자 질의에 대해서 효과적으로 응답할 수 있는 데이터 분할방법과 질의를 계산하는데 소용된 비용에 근거한 cache replacement정책을 적용하여 보다 효율적인 caching 전략이 될 수 있도록 보완할 계획이다.

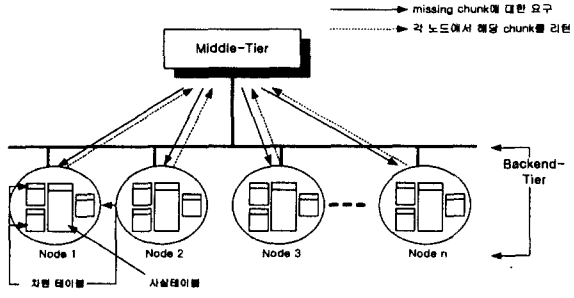

그림 7. 데이터 분할 및 chunk miss에 대한 parallel processing

참고문헌

[1] Surajit Chaudhuri and Umeshwar Dayal, "An Overview of Data Warehousing and OLAP Technology", stanford university, 1996.

[2] D. Quass. Materialized Views in Data Warehouses. PhD thesis, Stanford University, Department of Computer Science, 1997.

[3] Prasad M. Deshpande and Karthikeyyan Ramasamy and Jeffrey F. Naughton and Amit Shukla, "Caching Multidimensional Queries Using Chunks", SIGMOD '98 Seattle, WA, USA, 1998.

[4] Anindya Datta and Bongki Moon, "A Case for Parallelism in Data Warehousing and OLAP", http://loochi.bpa.arizona.edu. 1998.

[5] Yannis Kotidis and Nick Roussopoulos, "An Alternative Storage Organization for ROLAP Aggregate Views Based on Cubtrees", 1998.

[6] P. O'Neil and D. Quass, "Improved query performance with variant indexes", Proc. ACM SIGMOD, volume 26(2) of SIGMOD Record, pages 38-49, Tucson, Arizona, May 13-15 1997.

[7] P. Scheuerman, J. Shim and R. Vingralek, "A Data warehouse Intelligent Cache Manager", *Proc. of the 22nd Int. VLDB Conf.*, 1996.

[8] I. Viguier, A. Datta, and K.Ramamritham. "have your data and index it, too", efficient storage and indexing for data warehouses. Technical Report GOOD-TR-97-02, Dept. of MIS, University of Arizona, June 1997. Publication URL: http://loochi.bpa.arizona.edu.