고속 질의처리를 위한 MOLAP 큐브 저장구조*

임윤선 양혜영 김 명
이화여자대학교컴퓨터학과 고성능인터넷지식공학연구실
{lys96, purity02, mkim}@ewha.ac.kr

A MOLAP Cube Storage Scheme for Fast Query Processing

Yoonsun Lim Hyeyeong Yang Myung Kim
Dept. of Computer Science & Engineering, Ewha Womans University

요 약

데이터 웨어하우스의 데이터를 다차원적으로 분석하여 그 결과를 온라인으로 사용자에게 제공하는 것을 OLAP 이라고 하고, 이 때 데이터를 큐브라고 불리는 배열에 저장해 두고 데이터를 위치정보를 통해 엑세스하는 시스템을 MOLAP 시스템이라고 한다. OLAP 연산 도중에 디스크로부터 읽어야 하는 데이터의 양을 감소시키기 위해 큐브를 압축된 청크 단위로 저장하는 방안이 이미 제안되어 있으나, 큐브의 데이터 분포, 청크와 디스크 블록의 크기 관계 등을 고려하여 디스크 엑세스를 줄이는 방안에 관한 연구는 아직 소개된 바가 없다. 본 연구에서는 청크들을 밀도를 가준으로하여 군집화 하고, 큐브내의 인접 청크들을 가능한 한 동일한 디스크 블록에 속하게 함으로써, OLAP의 주요 연산인 슬라이스, 다이스와 같은 연산의 속도를 향상시키는 방안을 제시한다. 제안한 저장구조는 실험을 통해 그 효율성을 증명하였다.

1. 서론

대용량 데이터를 다차원적으로 분석하여 그 결과를 신속하게 제공하기 위해 데이터와 사전 연산 결과를 큐브(cube)라고 불리는 다차원 배열에 저장해 두고 질의처리를 하는 시스템을 MOLAP (Multidimensional Online Analytical Processing) 시스템이라고 한다. 다차원 배열이 메모리 상에 존재할 때는 데이터가 위치 정보를 통해 신속하게 엑세스될 수 있으나, 디스크에 저장될 때는 특정 차원을 기준으로 1차원적으로 저장되기 때문에 OLAP의 주요 연산인 슬라이스, 다이스 연산의 경우에 이와 같은 큐브 저장방식은 효율적이지 못하다. 이를 해결하는 한 가지 방법으로 큐브를 디스크 블록 크기 정도의 청크 단위로 나누어 저장하는 방법이 [SS94]에 제안되어 있고, 희박 청크들을 압축하는 방법과 이러한 저장구조를 바탕으로 큐브를 생성하는 방법이 [ZDN97]에 제시되어 있다.

본 연구에서는 이 방법을 개선하여 청크들을 밀도와 인접도에 따라 달리 배치함으로써 OLAP 연산의 속도를 높이는 방안을 제시한다. 디스크 블록 크기와 동일한 크기를 갖는 밀집청크들을 분리하여 1 청크가 1개의 블록에 포함되도록 하였다. 또한 청크 순서를 Z-인덱싱 방식으로 번호를 매겨 저장함으로써 OLAP 큐브의 생성을 수월하게 하고, 인접해 있는 수많은 희박 청크들을 하나의 디스크 블록에 저장하도록 하여 OLAP 연산시에 디스크 엑세스 회수를 줄였다.

본 논문의 구성은 다음과 같다. 2절에서 기존 연구결과를 소개하고, 3절에서 Z-인덱싱을 사용한 새로운 저장구조를 제시한다. 4절에서 저장구조에 대해서 실험한 방법과 결과를 설명하고, 5절에서 결론을 맺는다.

* 본 논문은 2001년도 두뇌한국 21 사업과 SK 텔레콤의 산업체 대응비 지원으로 수행되었음.

2. 관련연구

우선 [ZDN97]에서 제안된 압축청크단위 MOLAP 큐브의 저장방식을 살펴보고, 이를 어떤 면에서 개선 할 것인가를 분석해 보기로 한다.

큐브는 특정차원에 부당하지 않게 각 차원이 동일한 크기인 청크로 나누어 저장한다. 청크 크기는 디스크 블록 크기 정도로 하여 청크 1개를 메모리로 읽어들일 때, 디스크 블록 1개가 엑세스 되도록 한다. 유효 셀이 대략 40% 이상이면 밀집 청크(dense chunk)로 분류하여 배열형태로 저장하고, 밀도가 그 이하인 희박 청크(sparse chunk)들은 유효 셀만 선택하여 그 셀의 청크내의 주소(offset)와 셀 값(value) 쌍으로 저장한다. 청크의 크기가 일정하지 않기 때문에 청크 주소와 크기는 메타데이터로 만들어서 따로 저장한다.

그러나 이와 같은 큐브 저장방식을 잘 살펴보면, 밀집 청크는 그 크기가 비록 디스크 블록의 크기와 같으나, 희박 청크와 함께 저장되므로, 청크의 경계가 디스크 블록의 경계에 align 되어 있지 않다. 따라서, 밀집 청크 1개를 엑세스하기 위해 디스크 블록을 2개 읽어야 하는 경우가 많이 발생한다.

또한 큐브가 생성되는 과정에서 베이스 큐브의 청크들은 row-major 방식으로 번호가 매겨져 번호 순서로 디스크에 저장되기 때문에 OLAP 연산을 할 때 필요한 청크들이 인접해 있지 않아 여러 디스크 블록을 읽어야 하는 문제점을 갖고 있다. 따

라서 본 논문에서는 이러한 디스크 블록을 효율적으로 읽는 저장방식을 제안한다.

첫째, 밀집 청크와 희박청크를 분리하여 저장함으로써, 디스크 데이터 블록 크기와 같은 크기의 밀집 청크를 읽을 때는 항상 1개의 디스크 블록만 읽도록 하고, 1개의 디스크 블록에 가능한 한 많은 수의 희박 청크들을 모아 디스크 엑세스 시간을 줄이는 효과를 보도록 하였다. 둘째, 큐브 생성시 청크들의 순서를 Z-인덱싱 방식으로 저장함으로써, 연관 있는 데이터들을 디스크 블록에 클러스트시켜, OLAP 연산시에 읽을 디스크 블록 수를 줄였다.

3. 청크 밀도와 인접도를 고려한 큐브 저장구조의 설계

인접한 청크들이 동일한 디스크 블록에 모이도록 하기 위해 영상내의 컴포넌트 인덱싱 기법인 Z 인덱싱 기법을 사용한다. $2^n \times 2^n$, $1 \le n$, 크기의 2차원 배열을 Z-인덱싱하는 방법은 다음과 같다. 이 배열을 그림 1(b)와 같이 4개의 배열인 $2^{n-1} \times 2^{n-1}$ 크기로 나눈다. 이들을 각각 NW, NE, SW, SE 블록이라고 하자. Z-인덱싱을 하려면, NW 블록에 속한 원소들을 모두 번호를 매기고 나서, NE 블록에 속한 원소들을 번호 매기고, 그 다음에 SW와 SE 블록 순서로 번호를 매긴다. 이 각 4개의 블록은 다시 동일한 방법으로 순환적으로 분할 (recursively decompose)되어 번호가 매겨진다. 그림 1(c)는 NW 블록을 순환적으로 분할한 것을 보인다. 특정 배열원소의 Z 인덱스는 쉽게 계산될 수 있다. Z-인덱싱은 3차원 이상의 배열에도 확장되어 적용이 가능하다. 그림 2는 Z-인덱싱을 3차원 배열에 적용한 모습이다.

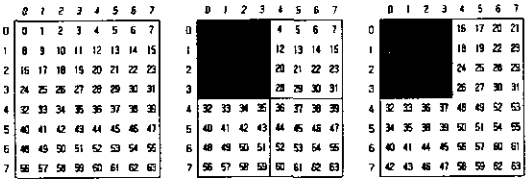

그림 1. 2차원 배열에 Z-인덱싱을 적용한 예제.

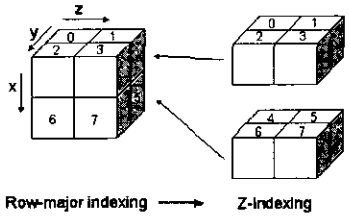

그림 2. 3차원 배열에 z-인덱싱을 사용한 경우.

이제, Z-인덱싱을 사용한 OLAP 큐브 저장구조를 소개한다. 우선 1개의 집계 테이블 (또는 사실 테이블)을 저장하는 방법을 설명 한 후에, 큐브 전체를 저장하는 방법을 설명하기로 한다. 그림 3에 8x8 셀을 갖는 3차원 집계 테이블이 파일에 저장되어 있는 예가 있다. 그림 3(a)의 집계 테이블은 [ZDN97]과 같은 방식으로 청크로 나뉜다. 각 청크는 Z 인덱스 순서로 번호가 매겨진다. 밀집 청크들은 배열 형태를 유지하면서 저장되고, 희박 청크들은 {offset,value} 쌍으로 변환되어 저장된다.

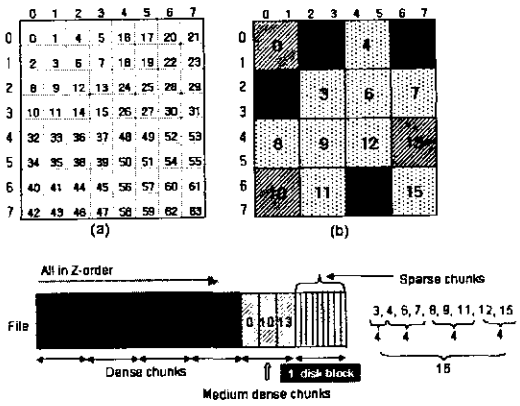

그림 3. 큐브 저장구조의 특징.

파일에 청크들이 저장될 때는 그림 3과 같이 3개의 그룹(D 그룹, M 그룹, S 그룹)으로 나뉘어 저장된다. D(dense) 그룹은 밀도가 높은 밀집 청크들이 저장되는 곳이다. 이 곳에 저장되는 청크들은 배열 형태를 유지하면서 저장된다. 따라서 각 청크의 크기는 디스크 블록의 크기와 같으며, 모두 일정하다.M(medium dense) 그룹과 S(sparse) 그룹에는 밀도가 낮은 희박 청크들이 저장된다. 이 들은 {offset, value} 형태로 변환되어 저장된다. 각 그룹내의 청크들은 Z 인덱스가 증가하는 순서대로 저장된다. 이와 같이 저장함으로써 밀집 청크의 크기가 실제 디스크 블록의 크기와 같지만, 청크가 디스크 블록의 경계에 align 되어 있지 않아서, 청크 1개를 디스크로부터 읽을 때 디스크 블록 2개가 읽히는 경우가 많이 발생한다는 것이었다. 이 문제는 그림 3과 같이 밀집 청크들을 파일의 한 쪽에 몰아 둠으로써 해결된다. 위에서 제안한 방법들은 실험을 통해 빠른 연산 결과를 보이는 것을 입증하였다.

4. 실험 및 평가

[ZDN97]의 방식과 비교해 보았을 때, 본 논문에서 제시한 큐브 저장구조가 질의 처리 속도를 얼마나 향상시켰는가를 분석하는 실험을 하였다. 다양한 형태의 큐브 상에서 슬라이스와 다이스 연산을 실행하였을 때 읽어야 하는 디스크 블록 수를 통해 질의 처리 속도를 측정하였다. 실험에는 MOLAP 큐브의 상당 부분을 차지하면서 희박하여 청크의 크기 변화가 심하고 청크 저장 순서에 영향을 가장 많이 받는 MOLAP 베이스 큐브를 대상으로 하였다.

큐브의 형태는 큐브내의 데이터 분포 형태, 밀도, 차원 수, 차원 멤버의 수를 파라미터로 하여 정하였다. 디스크 블록의 크기는 4K 바이트로 하였고, 큐브를 구성하는 청크는 각 차원이 동일한 멤버 개수를 갖도록 하면서 100% 밀도일 때, 4K 바이트에 가장 근접하도록 정하였다.

실험을 위해 가상 데이터를 생성하는 Storage Scheme Simulator를 개발하였다. 이 프로그램을 통하여 데이터 분포 형태가 랜덤(random)한 경우와 클러스터(cluster)를 이루는 경우의 큐브를 생성하였다.

■ 랜덤 : 데이터가 큐브 전체에 고르게 분포되도록 하였다. 각 청크의 셀 개수는 밀도 편차가 0.1%를 넘지 않도록 하한값과 상한값을 정하여 랜덤함수를 이용하여 각 청크를 생성하였다.

■ 클러스터 : 데이터의 분포가 특정한 공간에 집중되어 있는 형태로 클러스터의 개수, 클러스터의 반경을 다양하게 하여 실생활의 데이터와 유사하게 생성하였다. 셀이 집중되어 있는 청크들도 랜덤함수를 이용하여 셀의 개수를 정하였고, 밀집청크와 희박청크들이 섞여있도록 하였다. 큐브는 표 1과 같은 경우를 만들어 실험하였다.

차원	멤버	청크수	밀도(%)	셀 수
3	320	32,768	0.1	32,768
			0.3	98,304
			1	327,680
	640	262,144	0.1	262,144
			0.3	786,432
			1	2,621,440
4	80	65,536	0.1	40,960
			0.3	122,880
			1	409,600
	160	1,048,576	0.1	655,360
			0.3	1,966,080
			1	6,553,600
5	64	1,048,576	0.1	1,073,741
			0.3	3,221,223
			1	10,737,410
	128	33,554,432	0.1	34,359,738
			0.3	103,079,215
			1	343,597,380

표 1. 실험에 사용한 큐브의 형태.

표 1과 같은 데이터로 Z-인덱싱 방식과 [ZDN97] 방식으로 MOLAP 큐브를 만들어 슬라이스, 다이스 연산을 하였을 때 읽혀지는 디스크 블록 수를 조사하였다.

그림 4(a)는 A, B, C를 차원으로 갖는 3차원 MOLAP 베이스 큐브 상에서 슬라이스 연산을 한 결과이다. [ZDN97] 방식은 차원에 따라 읽혀지는 디스크 블록 수가 큰 차이가 나타나는 것을 보여주고 있다. 반면에 Z-인덱싱 방식은 모든 차원에 대해 디스크 엑세스가 일정하게 일어난다는 것을 알 수 있다.

그림 5와 6은 큐브 데이터 분포가 랜덤한 경우와 클러스터인 경우에 슬라이스, 다이스 연산을 하였을 때 각 차원마다 읽혀지는 디스크 블록 수의 평균값을 Z-인덱싱 방식과 [ZDN97]방식에 대해 비교하며 보인다. 실험을 통하여 본 논문에서 제안하고 있는 Z-인덱싱 저장방식이 효율적이라는 것을 증명하였다.

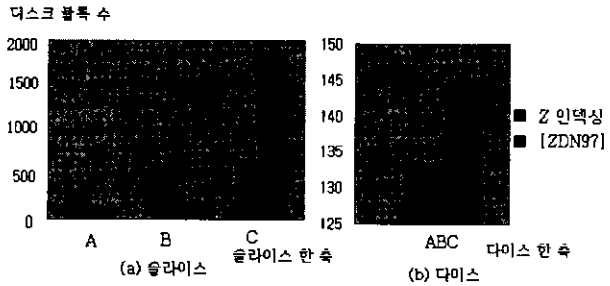

그림 4. OLAP 연산시 읽혀지는 디스크 블록 수
(데이터가 3차원인 경우)

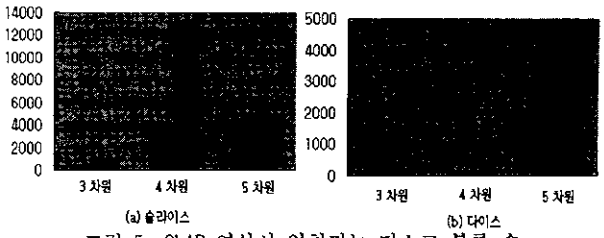

그림 5. OLAP 연산시 읽혀지는 디스크 블록 수.
(데이터 분포가 랜덤 형태인 경우)

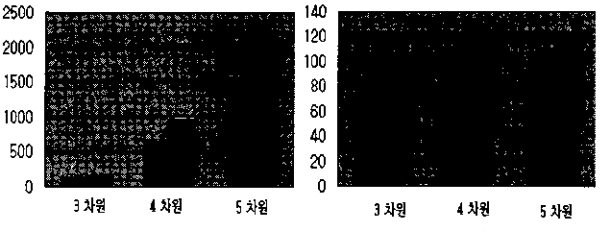

그림 6. OLAP 연산시 읽혀지는 디스크 블록 수.
(데이터 분포가 클러스터 형태인 경우)

5. 결론

MOLAP 큐브는 질의처리 속도를 높이기 위해 청크 단위로 저장되는 것이 바람직하다. 모든 차원 멤버의 조합마다 데이터 셀이 존재하는 것이 아니므로 MOLAP 큐브는 희박하기 쉽고, 이에 따른 저장공간의 낭비를 없애고 질의 처리 속도를 높이기 위해 청크들은 적절하게 압축되어 저장된다. 청크의 크기가 일정하지 않기 때문에 청크의 저장 순서에 질의 처리 시간이 큰 영향을 받게 된다. 본 연구에서는 청크의 밀도와 인접도를 고려한 효율적인 큐브 저장구조를 제안하였다. 본 연구에서 제안한 방법은 밀집 청크들과 희박 청크들을 분리하여 저장함으로써 밀집 청크 엑세스 시간을 줄이고, 희박청크들은 인접한 것들끼리 모아 저장함으로써 디스크 블록의 읽는 회수를 상당히 감소시킬 수 있도록 하였다. 제안한 방법은 데이터의 차원, 밀도, 멤버 수, 분포형태를 다양하게 만들어 실험을 하여 그 효율성을 입증하였다.

6. 참고문헌

[Bus00] Business Intelligence Ltd."The Olap Report : Database Explosion, 2000.

[Pil99] Pilot Software Inc, "An Introduction to OLAP: Multidimensional Terminology and Technology," http://www.pilotsw.com/olap/olap.htm.

[SAG94] S.Sarawagi, R.Agrawal, A, Gupta, "On Computing the Data Cube,"Research Report, IBM Almaden Research Center.

[SS94]S. Sarawagi, M. Stonebraker, "Efficient Organization of Large Multidimensional Arrays," Proc. of 10th Data Engineering Conference, Feb. 1994.

[ZDN97] Yihong Zhao, Prasad Deshpande, Jeffrey Naughton, "An Array-Based Algorithm for Simultaneous Multidimensional Aggregates," Proc. ACM SIGMOD 1997, pp.159-170.