OLAP 환경에서 다중 존 디스크를 활용한 실체뷰의 효율적 저장 기법

Efficient Storage Techniques for Materialized Views Using Multi-Zoned Disks in OLAP Environment

장재영(Jae-Young Chang)*

초 록

대용량의 데이타를 다루는 OLAP 데이타베이스 환경에서는 기반 디스크 시스템의 구조와 효율적 접근방법이 전체적인 성능을 좌우하는 중요한 요소가 된다. 최근 들어 하드 디스크들은 여러 개의 물리적 존을 갖는 구조로 설계되고 있는데, 각 존들은 그 위치에 따라 다양한 탐색시간과 데이타 전송률을 갖는 특징을 갖고 있다. 그러나 기존 연구에서는 다중 존을 고려하지 않은 하나의 탐색시간과 데이타 전송률을 갖는 단순한 디스크 모델에 기반을 두고 진행되어 왔다. 본 논문에서는 대용량의 데이타를 다루는 OLAP 환경에서 주어진 실체뷰 집합을 다중 존에 효율적으로 저장하는 기법을 제안한다. 이를 위해 각 실체뷰를 접근확률에 따라 디스크 존에 배치하는 알고리즘을 제시하고, 데이타 지속적으로 갱신되는 동적 환경에서의 저장 방법에 대해서도 살펴본다. 마지막으로 실험을 통하여 본 논문에서 제시된 알고리즘을 효율성을 증명한다.

ABSTRACT

In determining the performance of OLAP database applications, the structure and the effective access methods to the underlying disk system is a significant factor. In recent years, hard disks are designed with multiple physical zones where seek times and data transfer rates vary across the zones. However, there is little consideration of multi-zone disks in previous works. Instead, they assumed a traditional disk model that comes with many simplifying assumptions such as an average seek-time and a single data transfer rate. In this paper, we propose a technique storing a set of materialized views into the multi-zoned disks in OLAP environment dealing with large sets of data. We first present the disk zoning algorithm of materialized views according to the access probabilities of each views. Also, we address the problem of storing views in the dynamic environment where data are updated continuously. Finally, through experiments, we prove the performance improvement of the proposed algorithm against the conventional methods.

키워드 : 다중 존 디스크, OLAP, 데이타 큐브, 실체뷰, 질의 처리
Multi-Zoned Disk, OLAP, Data Cube, Materialized View, Query Processing

---

본 연구는 2008년도 한성대학교 교내연구비 지원과제임.
* 한성대학교 컴퓨터공학과 부교수
2009년 01월 20일 접수, 2009년 01월 30일 심사완료 후 2009년 02월 05일 게재확정.

1. 서 론

1990년대 이후 OLAP(On-Line Analytic Processing)을 이용한 정보시스템은 이제 공공기관이나 기업 등에서 의사결정과정에 필수적인 요소로 인식되고 있다. OLAP이 대중화된 요인으로는 대용량의 데이타로부터 정제된 분석정보를 얻으려는 필요성이 점차 확산된 측면도 있지만 이와 더불어 실시간으로 분석정보를 얻을 수 있는 기술의 발달도 큰 역할을 하였다. 대용량의 데이타로부터 분석작업이 이루어지려면 많은 시스템 자원이 요구된다. 따라서 하드웨어나 소프트웨어적으로 성능을 향상시키기 위한 많은 노력들이 진행되어 왔다. OLAP에서 성능향상을 위한 방법 중 가장 대표적인 것이 실체뷰(materialized view)의 도입이다[1~8]. 기존의 OLTP에서는 데이타의 중복을 최소화하기 위해 정규화된 데이타 모델을 사용하였으나 갱신작업이 거의 이루어지지 않는 OLAP 환경에서는 데이타 중복은 큰 문제가 되지 않는다. 따라서 질의 결과를 미리 저장해 놓은 형태인 실체뷰는 대용량의 기본 테이블에 대한 접근 없이 질의 결과를 손쉽게 얻을 수 있도록 도와주므로 성능 향상에 많은 도움이 된다.

OLAP에서는 주로 데이타 큐브(data cube)라고 불리는 다차원 데이타 모델(multidimensional data model)에 기반을 둔다[1, 2, 3, 8]. 데이타 큐브란 사용자가 분석하고자 하는 관심의 대상이 되는 주제(subject)에 대해 이 주제를 설명하는 요소들을 개별 차원(dimension)으로 저장하는 방식이다. 예를 들어 특정 회사의 판매정보는 판매된 “상품”, 판매를 담당한 “사원”, 상품을 구입한 “고객”과 같이 3차원적인 관점으로 데이타를 분석할 수 있으며 각 개별 판매정보당 “판매량”을 측정치(measure)로 저장하게 된다. 결국 판매정보 데이타는 (상품, 사원, 고객)의 3차원 데이타 큐브를 구성하며 각각에 대해 판매량 정보를 담고 있게 된다. 일반적으로 이러한 데이타 큐브에서는 집계된(요약된) 질의도 발생한다. 예를 들어 사원들의 실적을 분석하기 위해 사원별 판매량을 얻으려면 3차원 데이타 큐브로 부터 집계작업을 거쳐 질의의 결과를 얻게 된다. 이와 같이 데이타 큐브에서는 이를 구성하는 차원들의 임의의 부분조합-본 논문에서는 이러한 데이타 큐브의 부분조합을 서브큐브(subcube)라고 한다.-에 대한 질의가 발생할 수 있으며 자주 사용되는 서브큐브에 대해 그 결과를 미리 저장해 놓으면 성능향상을 기대할 수 있다. 결국 미리 저장된 서브큐브들은 앞서 언급한 실체뷰가 되는 것이다. 즉, 가능한 서브큐브 중에서 성능에 도움이 되는 일부를 실체뷰로 저장함으로써 전반적인 질의 성능을 향상시킬 수 있다. 이러한 필요성에 따라 그동안 최적의 실체뷰를 선택하는 많은 연구가 진행되어 왔다. 특히[1, 8]에서는 데이타 큐브 환경에서 휴리스틱(heuristic)을 이용한 실체뷰 선택 방법을 제안하였다.

이와 같이 데이타 큐브 환경에서 실체뷰에 대한 선택에 대한 연구는 많이 이루어져왔으나 정해진 실체뷰를 어떠한 방법으로 저장하느냐에 대한 연구는 거의 이루어지지 않고 있다. 특히 물리적 디스크에 실체뷰의 저장방법에 대한 연구는 거의 없는 실정이다. 최

근 들어 인접한 여러 개의 디스크 실린더들을 존(zone) 단위로 묶어 전체 디스크 영역을 여러 개의 존으로 관리하는 기술이 사용되고 있다[9, 10, 11]. 디스크는 물리적 특성으로 인해 디스크 중심에서 바깥쪽으로 갈수록 트랙(track)의 길이가 더 길어짐에 따라 이론적으로 바깥쪽의 트랙에 더 많은 양의 데이타를 저장할 수 있다. 따라서 디스크를 여러 개의 물리적 존으로 나눈 후에 존마다 탐색 시간(seek time)과 데이타 전송률(transfer rate)을 다양화함으로써, 디스크의 성능 및 저장 공간을 극대화 할 수 있다. 그러나 현재까지 이러한 다중 존(multi-zone)으로 분할된 디스크 모델을 적용하여 데이타 저장 및 접근을 최적화하기 위한 연구는 일부를 제외하고는 거의 이루어지지 않고[12, 13, 14], 모든 트랙이 동일한 탐색 시간과 데이타 전송률을 갖는 전통적인 디스크 모델에 기반을 둔 연구만이 대부분을 차지하고 있다.

본 논문에서는 다중 존 디스크 환경에서 OLAP의 질의 성능을 향상시키기 위해 주어진 실체뷰를 효율적으로 저장하는 기법을 제안한다. 다중 존을 갖는 디스크 환경에서는 자주 접근되는 데이타를 보다 빠른 존에 저장함으로써 성능 향상을 기대할 수 있다. 따라서 저장 대상이 되는 각 실체뷰에 대해서 주어진 질의 집합을 처리하기 위한 접근빈도를 추정하여 접근확률이 높은 실체뷰부터 우선적으로 빠른 존에 저장한다. 질의 성능을 높이기 위한 또 다른 기회는 각 실체뷰를 동일한 존 또는 인접한 존들에 배치하여 실체뷰를 접근할 때 디스크 탐색시간을 최소화하는 것이다. 즉, 각 실체뷰를 존 단위로 부분적인 클러스터링을 하여 질의 처리를 지역화(localization)함으로써 성능 향상을 기대할 수 있다. 본 논문에서는 이러한 기본적인 성능 향상 기법을 적용하여 데이타 큐브 환경에서 서브큐브들로 구성된 실체뷰를 가정하고 이를 다중 존 디스크에 저장하는 알고리즘을 제시한다. 데이타 큐브 환경에서는 이미 저장된 기본 데이타가 변경될 가능성은 희박하나 새로운 데이타가 지속적으로 증가할 가능성은 높다. 따라서 데이타가 지속적으로 증가되는 동적 환경을 가정하여, 각 실체뷰의 페이지들이 추가됨에 따라 위에서 제시한 기본적 저장기준을 충족하는 페이지 배치 알고리즘도 제시한다.

본 논문에서 제안한 기법의 타당성 및 실용성을 검증하기 위해 다양한 환경에서의 실험 결과를 제시한다. 실험은 예제로 주어진 데이타 큐브 환경에 대해서 실제 다중 존 디스크 모델을 이용하여 실시하였고 다중 존 디스크와 기존 디스크 환경에 대해서 성능을 비교 분석하였다. 본 논문에서는 문제를 단순화하기 위해 우선 하나의 디스크만을 가정하였다. 하지만 이 기법은 RAID와 같은 다중 디스크 환경에 대해서도 쉽게 확장될 수 있다.

본 논문의 구성은 다음과 같다. 우선 제 2장에서는 다중 존 디스크 모델과 데이타 큐브에서의 실체뷰에 대해서 간단히 소개한다. 제 3장에서는 실체뷰를 다중 존 디스크에 배치하는 알고리즘과 동적 환경에서의 페이지 저장 알고리즘을 제시한다. 제 4장에서는 실험 결과를 제시하고, 마지막으로 제 5장에서는 결론 및 향후 연구 과제에 대해 논한다.

2. 관련 연구

2.1 다중 존 디스크

최근 자기 디스크 드라이브에서의 중요한 물리적 특징 중 하나가 바로 다중 존 기술이다. 다중 존은 원형으로 구성된 자기 디스크의 물리적 특성을 고려하여 고안되었다. 디스크에서 각 트랙은 중심으로부터 외부 방향으로 갈수록 더 길어진다. 따라서 디스크 상에서 동일한 저장밀도를 갖는다는 가정 하에 이론적으로 내부보다는 외부 트랙에 더 많은 데이타를 저장할 수 있다. 이러한 특성을 기반으로 디스크의 모든 트랙을 여러 개의 배타적 구역으로 분할하여 존으로 구성한다. 즉, 하나의 존은 인접한 디스크 트랙의 집합으로 구성된다. 이때 하나의 존 내에서 각 트랙은 동일한 저장용량을 갖는다. 즉, 같은 존 내의 각 트랙당 섹터(sector) 수는 동일하다. 따라서 중심으로 부터 외부에 위치한 존은 내부의 존보다 보다 많은 섹터수를 갖게 설계할 수 있으므로 외부 존에 더 많은 양의 데이타를 저장할 수 있다. 결과적으로 전체 디스크의 저장용량이 획기적으로 증가하게 된다.

디스크 당 존의 수는 디스크 모델에 따라 다양하다. 예를 들어, Seagate Cheetah X15는 9개의 존으로 구성되며 Seagate Barracuda 7200.7은 15개의 존을 갖는다[15, 16]. 한 디스크 내의 서로 다른 존은 서로 다른 데이타 전송률을 갖는다. 그 이유는 우선 1) 각 존마다 트랙들의 저장 용량이 다르고, 2) 디스크의 회전 속도는 존에 관계없이 일정하기 때문이다. 따라서 디스크 외부 존의 트랙이 내부 존의 트랙보다 더 많은 섹터로 구성되지만 하나의 트랙을 읽는 속도는 위치에 관계없이 일정하므로 외부 존의 데이타 전송률이 더 높게 된다. 예를 들어 <표 1>은 Seagate Cheetah X15에서의 존 별 저장용량과 데이타 전송률을 보여준다[15]. 이 표에서 보는 바와 같이 디스크 외부 존의 용량 및 데이타 전송률이 내부 존보다 높은 것을 알 수 있다.

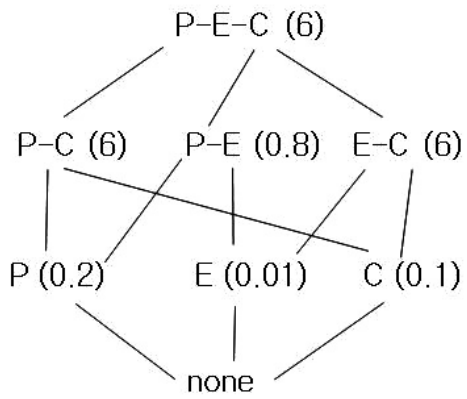
〈표 1〉 Seagate Cheetah X 15의 존 구성

Zone #	Size (GB)	*Read Rate (MB/s)*
0	12	*57.5*
1	3.5	*55.4*
2	3.0	*54.7*
3	4.0	*52.7*
4	3.0	*50.6*
5	2.5	*48.1*
6	3.0	*45.6*
7	2.5	*43.6*
8	*2.5*	*41.9*

2.2 데이타 큐브와 실체뷰

데이타 큐브는 분석 대상이 되는 데이타를 다차원 관점으로 구성하는 방식으로 OLAP의 기본 모델이 되고 있다. 서론에서 예를 든 판매량 분석을 위한 (상품, 사원, 고객)의 3차원 데이타 큐브를 가정하자. 이 예에서 판매량 분석을 위해 차원들의 다양한 조합에 의한 질의가 가능하다. 예를 들어 “상품별, 사원별, 고객별 판매량을 출력하라”와 같은 질의는 별도의 집계과정 없이 데이타 큐브의 내용을 직접 보여주면 된다. 그러나 차원들의

부분조합에 대한 질의는 데이타 큐브로 부터 집계연산을 사용하여 요약된 형태의 데이타 큐브를 생성해야한다. 예를 들어 “상품별, 사원별 판매량을 출력하라”와 같은 질의는 기본 3차원 큐브로 부터 집계연산을 이용하여 2차원 서브큐브를 생성한다. 기본 3차원 데이타 큐브와 이로 부터 계산 가능한 서브큐브들의 관계는 <그림 1>과 같이 격자(lattice) 형태로 표현할 수 있다.



〈그림 1〉 데이타 큐브의 예

이 그림에서 *P, E, C*는 각각 상품, 사원, 고객을 의미한다. 예를 들어, *P-E-C*는 서브큐브(상품, 사원, 고객)을 의미하며 *P-C*는 서브큐브(상품, 고객)을 의미하고 none은 별도의 차원 없이 전체 판매량을 나타내는 서브큐브를 나타낸다. 따라서 이 그림의 각 서브큐브들은 모든 가능한 질의의 종류를 표현한다. 각 서브큐브의 괄호 안에 있는 숫자는 각 실체뷰의 상대적인 row 수의 비율을 나타낸다. 즉, *P-E-C*와 *P-C*는 row의 수가 항상 같으며, *C*는 *E*보다 row의 수가 10배 가량 많다. 하지만 none의 row 수는 어느 경우에든 1이 된다(<그림 1>의 예는 본 논문의 모든 예와 실험에서 사용한다).

이러한 형태의 데이타 큐브를 물리적으로 구축하는 방법으로는 세 가지를 생각해 볼 수 있다. 1) 우선 가장 상단의 기본 데이타 큐브만을 저장하고 나머지 서브큐브들에 대한 질의 결과는 기본 큐브로 부터 집계연산을 통해 얻는다. 2) 두 번째 방법은 모든 서브큐브들을 실체뷰로 저장함으로써 해당 서브큐브에 대한 질의 결과를 집계연산 없이 직접 실체뷰로부터 얻는다. 3) 마지막 방법으로는 성능 향상에 도움이 되는 일부 서브큐브만을 실체화하여 이들로 부터 모든 질의 결과를 계산한다. 우선 첫 번째 방법은 저장 공간은 적게 필요하나 매번 집계연산이 필요하므로 좋은 성능을 기대하기 어렵다. 두 번째 방법은 반대로 집계연산이 전혀 필요하지 않아 좋은 성능을 기대할 수 있다. 그러나 많은 공간이 필요할 뿐만 아니라 데이타가 연속적으로 증가하는 동적 환경에서는 모든 실체뷰를 매번 갱신해야하는 어려움이 존재한다.

마지막 방법은 이전 방법의 장점을 모두 살릴 수 있는 가장 적당한 기법으로 알려져 있다[1, 8]. 예를 들어 <그림 1>에서 *P-E*에 대한 실체뷰를 저장한다면 *P-E* 뿐만 아니라 *P*, *E*, none에 대한 질의도 *P-E*로 부터 질의 결과를 얻을 수 있다. 이때 실체뷰 *P-E*의 크기가 *P-E-C*의 크기보다 매우 작다고 가정하자. 그렇다면 *P*, *E*, none의 결과를 얻을 때 *P-E*를 이용한다면 *P-E-C*로 부터 계산하는 것에 비해 상대적으로 좋은 성능을 기대할 수 있다.[1] 여기서 문제는 최적의 성능을 보

1) 여기서 질의 성능에 대한 비교는 [1]에서 제시한 선형비용모델(linear cost model)에 기반을 두었다.

이기 위해서 어떠한 서브큐브들을 실체뷰로 선택할 것인가에 관한 것이다. 이를 위해서 [1, 2, 3, 8]에서는 데이타 큐브 환경에서 최적의 실체뷰 조합을 선택하는 방법을 제안하였다. 이외에도 실체뷰 선택에 관한 방법은 많은 방법들이 이전 연구에서 제안되어왔다[4, 5, 6, 7]. 이와 같이 실체뷰 선택에 관한 연구는 많이 이루어져 왔으나 선택된 실체뷰를 저장하는 방법에 대한 연구는 이루어지지 않고 있다. 본 논문에서는 제 2.1절에서 언급한 다중 존 디스크 환경에서 데이타 큐브 형태의 실체뷰를 효율적으로 저장하는 방법을 제안한다.

3. 다중 존에서의 실체뷰 저장 기법

3.1 문제 정의

현재 많은 디스크들이 다중 존으로 설계된다는 것을 고려할 때 실체뷰들을 다중 존 환경에 어떻게 저장하느냐에 따라 많은 성능 차이를 보이게 된다. 본 논문에서는 다음과 같은 조건들을 만족하는 실체뷰의 다중 존 배치 기법을 제안한다.

조건 1 : 접근빈도가 높은 실체뷰를 빠른 존에 우선적으로 배치한다.

조건 2 : 하나의 실체뷰는 되도록 같은 존에 클러스터링(clustering)한다.

조건 3 : 실체뷰들을 저장하기위해 선택된 모든 존들이 최대한 동일한 정도로 활용되도록 배치한다.

(조건 1)은 접근빈도가 높은 실체뷰를 빠른 존에 배치하면 그렇지 않은 경우에 비해 좋은 성능을 보인다는 것을 기대할 수 있다. (조건 2)는 저장될 각 실체뷰를 하나의 존이나 인접한 존에 배치함으로써 디스크의 탐색 시간을 최소화 할 수 있다. 마지막으로 (조건 3)은 하나의 큐브를 위해 선택된 실체뷰들을 저장하기 위해 각 존을 균일하게 사용해야한다는 조건으로, 질의처리 성능에는 직접적인 영향을 기대할 수는 없다. 그러나 이 조건은 디스크에 하나의 큐브구조만을 저장하는 것이 아니라 다른 큐브들을 동시에 저장한다는 가정 하에 특정 존에 특정 큐브에 대한 실체뷰들이 집중되거나 전용되는 것을 방지하는데 목적이 있다. 그 이유는 하나의 큐브 구조를 위해 특정 존의 가용 공간을 모두 사용한다면 다른 큐브 구조의 실체뷰들을 저장하려고 할 때 (조건 1)이 만족시키는 배치 방법을 구현하기 어려기 때문이다. (조건 1)~(조건 3)을 모두 만족시키기 위한 알고리즘의 구현은 쉽지 않은 문제이다. 특히 (조건 2)와 (조건 3)은 서로 배치되는 경향이 있다. 따라서 본 논문에서는 (조건 2)를 만족하면서 (조건 3)을 최대한 만족시키기 위한 방법을 제안한다.

본 논문에서 제안하는 실체뷰의 존 디스크 배치 알고리즘을 위해 다음의 기호들을 정의한다. 우선 차원이 N인 큐브구조에 대한 서브큐브의 개수는 2^N이다. 여기서 서브큐브의 집합을 $Cube = \{c_1, \cdots, c_{2^N}\}$이라 하고, 임의의 사용자 질의가 주어졌을 때 이 질의의 결과를 직접적으로 얻을 수 있는 대상이 서브큐브 $c_i (1 \le i \le 2^N)$일 확률을 $pr(c_i)$라 하자. 이 때 관련 연구에서 언급한대로 실체뷰를 선택하는 정책에 따라 집합 $Cube$의 서브큐

브 중에서 n개가 실체뷰로 선택되었다고 가정하고 이를 $M=\{m_1, \cdots, m_n\}$이라 하자. 그렇다면 주어진 임의의 질의의 결과를 실체뷰 $m_j(1 \le j \le n)$로부터 얻을 수 있는 확률 $AP(m_j)$-이 확률을 실체뷰 m_j의 접근확률(access probability)이라 하자. -은 다음의 수식으로 간단히 계산할 수 있다.

$$AP(m_j) = \sum_{k=1}^{l} (pr(c_{jk})) \quad (1)$$

이 식에서 $c_{jk}\,(1 \le k \le l)$는 집합 *Cube*의 각 서브큐브에 중에서 주어진 질의를 실체뷰 m_j로부터 계산하는 것이 가장 좋은 성능을 갖는다고 판단되는 서브큐브들을 집합을 나타낸다. 예를 들어, <그림 1>의 데이타 큐브에서 *P-E-C*와 *P-E*만을 실체뷰로 저장된다고 가정하자. 그리고 각 서브큐브에 대한 질의가 발생할 확률이 동일하다고 가정하자(즉, $pr(c_i)$ = 1/8). 이때 서브큐브 *P-E*를 비롯하여 *P*, *E*, none에 대한 질의는 실체뷰 *P-E-C*보다 *P-E*로 계산하는 것이 가장 유리하며 나머지 서브큐브들은 *P-E*로 계산이 불가능하므로 *P-E-C*로 계산해야한다. 따라서 실체뷰 *P-E-C*와 *P-E*의 접근확률은 각각 0.5가 된다. 만약 모든 서브큐브들이 실체뷰로 구축된다면 모든 실체뷰에 대해서 접근확률은 1/8이 되며, 가장 큰 서브큐브인 *P-E-C* 하나만을 실체뷰로 저장한다면 이 실체뷰에 대한 접근확률은 1이 된다.

다음으로 존 디스크 환경을 가정하기 위해서 다음의 용어와 함수를 정의한다.

NZ : 실체뷰가 저장되도록 할당된 존의 총 수를 나타내며, 데이타 전송 속도가 빠른 존에서 느린 존 순으로 0부터 존 번호가 부여된다(즉, $zid = 0, \cdots, NZ-1$).

Vol : 할당된 존의 총 용량을 나타내며 저장되는 실체뷰를 모두 수용할 만큼 충분히 크다고 가정한다.

Zone Vol[*zid*] : 존 *zid*의 용량을 나타낸다.

Zone R[*zid*] : 존 *zid*의 상대 용량비율로 *Zone Vol*[*zid*]/ *Vol*으로 계산된다.

Zone C[*zid*] : 존 *zid*에 저장된 실제 페이지의 수를 나타낸다.

$npage(m_i)$: 실체뷰 m_i를 구성하는 페이지 수를 나타낸다.

NP : 저장될 실체뷰들의 총 페이지 수를 나타낸다. 즉, $NP = \sum_{i=1}^{n} npage(m_i)$이다.

다음으로 각 존을 균일하게 활용하느냐의 여부를 판단하기 위해서는 각 존의 활용 정도를 나타내는 기준이 필요한데, 존 *zid*에 대한 존 활용도(zone utilization indicator : *ZUI*)는 다음과 같이 정의될 수 있다.

$$ZUI[zid] = Zone\,C[zid]/(NP \times Zone\,R[zid]) \quad (2)$$

이 식에서 $NP \times Zone\,R[zid]$는 모든 존이 균등하게 활용될 경우 존 *zid*에 저장될 페이지의 수를 의미한다. 예를 들어 $ZUI[zid] > 1$일 경우 존 *zid*에는 페이지들이 적정수보다 과도하게 할당되었음을 의미하며 반대로 $ZUI[zid] < 1$은 적정수보다 적게 저장되었음을 의미한다.

3.2 다중 존에서의 실체뷰 저장 알고리즘

실체뷰를 다중 존 디스크에 저장하는 방법은 다음의 두 상황에 따라 나누어 생각해 볼 수 있다. 첫째는 미리 주어진 데이타 집합에 대해서 선택된 실체뷰들을 하나의 알고리즘으로 단시간에 일괄 구축하는 상황이고, 둘째는 개별 데이타의 삽입연산으로 실체뷰의 페이지들을 동적으로 추가해야 하는 상황이다. 본 절에서는 우선 일괄적으로 구축하는 환경에서의 실체뷰 저장 알고리즘을 제시하고, 다음으로 동적 환경에서의 저장 방법을 제시한다.

우선 실체뷰를 일괄 구축해야하는 상황에서 제 3.1절의 (조건 1)~(조건 3)을 만족하는 알고리즘은 비교적 간단하게 구성할 수 있다. 기본적인 아이디어는 주어진 실체뷰 집합에 대해서, 접근확률이 높은 실체뷰 순으로 차례로 빠른 존 부터 배치하는 것이다. 자세한 알고리즘은 <그림 2>와 같다.

이 알고리즘은 우선 실체뷰를 저장하기 위한 존들($zid = 0, \cdots, NZ-1$)에 대해서 저장된 페이지의 수를 나타내는 $ZoneC[zid]$를 초기화하고, 선택된 실체뷰 집합 $M = \{m_1, \cdots, m_n\}$를 접근확률인 $AP(m_i)$값의 내림차순으로 정렬한다. 이후에 정렬된 각 실체뷰 m_i 순으로 페이지들을 빠른 존에 차례로 배치하게 된다. 이 때 각 존에는 상대 용량 비율을 고려하여 $ZUI[zid]$가 1이 되도록 $NP \times ZoneR[zid]$개의 페이지만을 할당하고 그 이후의 페이지들은 다음 존에 배치하게 된다.

이 알고리즘은 제 3.1절에서 제시한 3가지 조건을 모두 만족한다. 접근확률이 높은 실체뷰들을 빠른 존에 순차적으로 배치하므로 (조건 1)을 단연히 만족한다. 그리고 각 실체뷰의 페이지들을 하나의 존이나 인접한 존들에 배치하므로 (조건 2)도 만족된다. 마지막으로 존들을 균일하게 사용하기 위해 각 존에 $NP \times ZoneR[zid]$개의 페이지만을 할당하므로 (조건 3)을 만족하게 된다.

```
알고리즘 ViewZoning
  입력 : 실체뷰 집합 M = {m_1, ···, m_n}
  출력 : 없음

  for each zid = 0, ···, NZ-1
        ZoneC[zid] = 0;
  Sort M = {m_1, ···, m_n} in descending order
  of AP(m_i);
  zid = 0;
  for each sorted m_i {
     for each page p in m_i {
          Write p onto the zone zid;
          ZoneC[zid]++;
          if( ZoneC[zid] == NP × ZoneR[zid] )
              zid++;
     }
  }
```

〈그림 2〉 다중 존 디스크에서의 실체뷰 배치 알고리즘

다음으로 이미 저장된 실체뷰에 대해서 동적으로 페이지들이 추가되는 경우의 배치 방법에 대해 알아본다. 큐브 구조에서 데이타의 삽입은 가장 기본적인 실체뷰에서 발생되고 나머지 실체뷰들에 그 영향이 파급되는 과정으로 이루어진다. <그림 1>에서 가장 기본적인 실체뷰인 P-E-C에서 데이타의 삽입이 이루어지면, 일관성을 유지하기 위해 이 실체뷰의 집계결과인 나머지 실체뷰들에 대해서도 그에 상응하는 데이타가 추가되어야한다. 이러한 데이타 추가 과정에서 기존에 저장된

각 실체뷰의 페이지 영역에 더 이상 공간이 남아있지 않으면 새로운 페이지가 추가된다. 따라서 각 실체뷰에 새로운 페이지가 추가될 경우 어느 존에 이 페이지를 저장할 것인가를 결정해야 한다. 이 때 가장 중요하게 고려해야할 문제가 기존에 저장된 페이지들에 영향을 미치지 않으면서 (조건 1)~(조건 3)을 만족하는 존을 결정하는 것이다.

현재 다중 존 디스크에 <그림 3>의 알고리즘에 따라 실체뷰들이 저장되어 있으며, 이 중 하나의 실체뷰 m_i에 대한 페이지 p를 새롭게 저장해야한다고 가정하자. 이때 기존의 m_i가 저장된 존들 중 하나에 p를 저장하게 되면 (조건 1)과 (조건 2)가 자연스럽게 만족된다. 하지만 이 경우에 모든 존들이 균등하게 활용되어야 한다는 (조건 3)을 위배할 가능성이 높아지게 된다. 앞서 언급한 바와 같이 데이타의 추가는 특정 실체뷰에만 국한된 문제가 아니라 가장 기본적인 실체뷰에 데이타가 추가되면 그 결과가 모든 실체뷰에 파급되게 된다. 따라서 각 실체뷰에 페이지가 추가되는 빈도는 기존에 저장된 실체뷰들의 크기와 비례할 것으로 기대할 수 있다. 예를 들어 <그림 1>에서 초기의 실체뷰 *P-E-C*와 *P-C*의 페이지 수가 각각 80개, 20개라고 가정할 경우, *P-E-C*에 n개의 페이지가 추가된다면 *P-C*에는 평균적으로 $n/4$개의 페이지가 추가 될 것으로 예상할 수 있다. 물론 데이타의 삽입에 따른 각 실체뷰의 상대적 증가 비율은 추가되는 데이타의 특성에 따라 많은 차이를 보일 수 있으나 추가되는 데이타의 종류를 미리 예상할 수 없으므로 가장 일반적인 경우만을 가정할 수밖에 없다-이 문제는 다음 장에서 제시할 실험결과에서 다시 논의한다. -따라서 특정 실체뷰의 새로운 페이지가 추가될 때, 기존에 이 실체뷰가 저장된 존(또는 존들 중 하나)에 저장하는 것이 가장 적합한 방법이라고 판단할 수 있다. 이와 같은 가정 하에 <그림 2>의 알고리즘에 의해 다중 존 디스크에 저장된 $M=\{m_1, \cdots, m_n\}$에 대해서, 실체뷰 m_i의 새로운 페이지 p가 저장될 존은 다음과 같은 방식으로 결정할 수 있다.[2)]

$min_a(F)$를 수식 F를 만족시키는 최소의 α 값이라 할 때, 실체뷰 m_i가 저장된 가장 작은 번호의 존은 다음의 수식을 만족하는 α이다.

$$min_{\alpha}\left(\sum_{j=1}^{i-1} npage(m_j) < \sum_{j=0}^{\alpha} Zone\,C[j]\right) \qquad (3)$$

이 식에서 $\sum_{j=1}^{i-1} npage(m_j)$는 실체뷰 m_1부터 m_{i-1}까지의 총 페이지 수이며 $\sum_{j=0}^{\alpha} Zone\,C[j]$는 존 0부터 존 α까지 저장된 총 페이지 수를 나타낸다. 따라서 실체뷰 m_i가 저장된 가장 작은 번호의 존은 $\sum_{j=0}^{\alpha} Zone\,C[j]$의 값이 $\sum_{j=1}^{i-1} npage(m_j)$보다 큰 값이 되는 α중에서 가장 작은 값으로 나타낼 수 있다. 역으로 실체뷰 m_i가 저장된 가장 큰 번호의 존은 다음의 수식을 만족하는 β이다.

$$min_{\beta}\left(\sum_{j=1}^{i} npage(m_j) \le \sum_{j=0}^{\beta} Zone\,C[j]\right) \qquad (4)$$

2) 단, 이 부분을 포함한 본 논문의 이후 부터는 표현을 단순화하기 실체뷰 집합 $M=\{m_1, \cdots, m_n\}$은 m_1부터 m_n까지 $AP(m_i)(1 \le i \le n)$의 내림차순으로 정렬되었다고 가정한다.

따라서 기존에 실체뷰 m_i가 저장된 존들은 위의 두 수식을 만족하는 α부터 β까지 존들임을 알 수 있다. 결과적으로 실체뷰 m_i에 추가된 새로운 페이지 p는 α부터 β사이의 존 중 하나에 저장하면 된다. 마지막으로 α부터 β 중에서 어떤 존에 저장할지는 각 존의 *ZUI*값을 이용하여 판단 할 수 있다. 즉, (조건 3)을 절대적으로는 만족시킬 수 없지만 p를 *ZUI*값이 가장 작은 존에 저장함으로써 부분적으로나마 존들을 균일하게 활용할 수 있다.

<그림 3>은 지금까지의 과정에 대한 알고리즘을 보여준다. 이 알고리즘에서 npage1과 npage2는 m_1부터 각각 m_{i-1}과 m_i까지 실체뷰들의 총 페이지 수를 나타낸다. CumZoneC는 가장 빠른 존부터 누적된 존들의 총 용량을 저장한다. 이 값들을 바탕으로 첫 번째 while문에서는 식 (3)을 만족하는 α값을 찾는 과정을 나타내며, 두 번째 while문은 식 (4)를 만족하는 β값을 찾는 과정을 보여준다.

알고리즘 DynamicZoning

입력 : 실체뷰 m_i의 새로운 페이지 p

출력 : 없음

```
npage1 = 0;    napge2 = 0;
for j = 1 to i-1 do
        npage1 = npage1 + npage(m_j);
CumZoneC = 0;     j = 0;
while() {
       CumZoneC = CumZoneC + ZoneC[j]
       if(CumZoneC > npage1)   { α = j;    break; }
       else       j = j + 1;
}

npage2 = npage1 + npage(m_i);
while() {
      if(CumZoneC >= npage2)   { β = j;    break; }
      else  {
            j = j + 1
            CumZoneC = CumZoneC + ZoneC[j]
      }
}
```

Write p onto zone *zid* where *ZUI*[*zid*] is the minimum value among *ZUI*[α], …, *ZUI*[β]

〈그림 3〉 동적환경에서의 실체뷰 배치알고리즘

이 알고리즘의 마지막 부분에서 zid는 α부터 β사이의 존 중에서 ZUI가 가장 작은 존을 나타내며, 결과적으로 zid에 페이지 p를 저장하게 된다.

4. 실험 결과

4.1 실험 환경

본 논문에서 제안한 다중 존 디스크 환경에서의 실체뷰 저장 알고리즘들에 대한 우수성 및 실용성을 검증하기 위해 <그림 1>의 데이타 큐브를 대상으로 다양한 조건에서 실험을 실시하였다. 우선 <그림 1>의 데이타 큐브에서 각 서브큐브의 row의 길이는 동일하다고 가정하였다. 이 경우 각 서브큐브를 실체뷰로 저장할 때 그 크기는 <그림 1>에서 나타난 상대적인 row의 수에 비례하게 된다. 데이타 큐브에 대한 질의는 각 차원 조합에 대해 집계 결과를 구하는 상황을 가정하였다. 따라서 각 서브큐브 차제가 질의의 종류와 일치하게 되므로 질의는 총 8개로 구성된다.

실험은 <그림 2>의 알고리즘과 같이 접근 확률을 고려하여 실체뷰를 배치하는 본 논문의 방식과 그렇지 않은 방식의 성능을 비교하였다. 또한 <그림 3>의 알고리즘을 구현하여 동적 환경에서의 실체뷰 배치 방식에서 각 존들 페이지 저장 비율에 대한 변화를 관찰하는 실험을 실시하였다.

본 실험에서는 각 페이지의 디스크 접근시간을 계산하기 위해 <표 2>에 있는 Seagate Barracuda 7200.7의 디스크 모델을 이용하였다[15]. 이 디스크는 2003년 출시 당시에 플래터(platter)당 세계 최초로 100M를 저장할 수 있도록 설계된 고용량 디스크로 현재까지 대중적으로 사용되고 있는 대표적인 다중 존 디스크로 알려져 있다. <표 2>에서 마지막 열은 특정 존에서 디스크 접근이 지역화 되었을 경우 평균적인 페이지의 읽기/쓰기 시간을 millisecond 단위로 나타낸 것이다. 여기서 특정 존 내에서 일정시간 동안 디스크 접근이 발생된다고 가정하자. 물리적 존 0은 가장 빠른 데이타 전송률을 갖고 있다 하더라도 그 존 내에서의 실제 페이지 접근 시간이 가장 빠르다는 것을 보장하지 않는다. 그 이유는 존 0이 가장 많은 수의 실린더로 구성되어 있으므로 탐색 시간이 다른 존에 비해 그 만큼 크기 때문이다. 같은 이유로 가장 작은 용량을 갖는 존이 그만큼 작은 수의 실린더가 할당되므로 실제 페이지 접근시간이 오히려 빠를 수 있다. <표 2>는 이와 같은 존 용량과 페이지 접근 시간과의 관계를 잘 보여주고 있다. 본 실험에서는 디스크의 I/O시간에 중점을 두기 위해 하나의 페이지 버퍼(buffer)만을 가정했으며 프리 패칭(pre-fetching) 역시 가정하지 않았다. 페이지 크기는 8KB로 고정하였다.

4.2 실체뷰 배치 기법의 성능

첫 번째 실험은 <그림 1>의 데이타 큐브 구조에서 실체뷰로 저장할 서브큐브의 수의 변화에 따른 성능을 비교하였다. 각 개수별로 저장될 실체뷰들은 성능을 가장 개선할 것으로 예상되는 것부터 차례로 선택하였으며, 그 순서는 [1]에서의 예와 같이 $P\text{-}E\text{-}C \rightarrow P\text{-}E$

〈표 2〉 실험에 사용된 디스크 모델

NZ(Disk capacity) : 200 GB; Average seek time : 7.5 ms; Average rotationallatency : 4.17 ms; Average data transfer rate of the disk : 50.47 MB/sec.

zid	physical zone number	*zoneVol*(GB)	*zoneR*	average 8KB page read/write time in ms
0	8	6	0.03	4.41270302
1	11	6	0.03	4.454687191
2	4	9	0.045	4.491726772
3	14	6	0.03	4.510334263
4	7	9	0.045	4.526061249
5	12	8	0.04	4.573418958
6	10	9	0.045	4.578541932
7	13	8	0.04	4.595950336
8	5	12	0.06	4.613985193
9	2	14	0.07	4.625634668
10	1	17	0.085	4.703298305
11	6	14	0.07	4.704762063
12	9	13	0.065	4.72599499
13	3	21	0.105	4.892135707
14	0	48	0.24	5.64322693

→ C → E → P → none → P-C → E-C의 순으로 결정하였다. 실체뷰의 크기는 P-E-C의 크기를 기준으로 5GB로 가정하였으며 나머지 실체뷰들은 <그림 1>과 같은 비율로 산정하였다. 그리고 각 실체뷰의 수에 따라 다음과 같이 세 가지 방식에 의해 실체뷰를 저장하는 경우에 대해서 실험을 실시하였다.

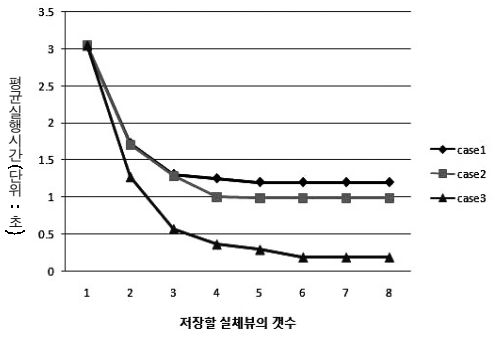
case 1 : 각 존의 접근 성능을 고려하지 않고 임의의 존에 페이지들을 저장

case 2 : 각 서브큐브에 대한 질의가 발생할 확률이 동일하다는 가정 하에 실체뷰의 접근확률을 계산하여 <그림 2>의 알고리즘에 의해 실체뷰의 페이지들을 저장

case 3 : 저장될 실체뷰의 차원이 하나 작아질수록 접근확률이 2배가 된다는 가정 하에 <그림 2>의 알고리즘에 의해 실체뷰의 페이지들을 저장

case 1은 다중 존 디스크를 가정하지 않고 페이지들을 임의의 위치에 배치하는 전통적인 방식을 의미한다. case 2는 각 서브큐브에 대한 질의확률이 동일하다는 가정 하에 저장될 실체뷰 m_i에 대해 $AP(m_i)$를 계산하여, $AP(m_i)$가 높은 실체뷰부터 빠른 존에 저장하는 방식을 나타낸다. 마지막으로 case 3은 실체뷰의 크기가 작을수록 접근확률을 높게 설정한 것으로 case 2에 비해 각 실체뷰에 대한 접근확률의 편차가 클 경우의 성능을 관

찰하기 위한 설정이다. 예를 들어 실체뷰 *P-E-C*에 비해 *P-E*의 접근확률이 2배이며 *P*는 4배의 접근확률을 갖는다. case 1부터 case 3까지 각각의 경우에 대해 10,000개의 질의를 임의로 발생시켜 각 질의의 평균시간을 관찰했으며 그 결과는 <그림 4>와 같다.

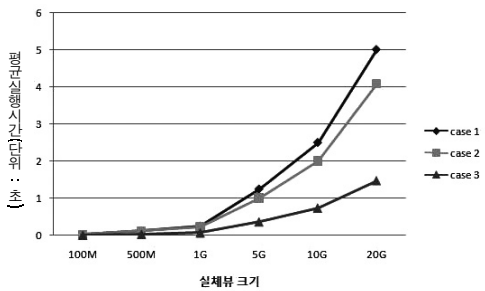


〈그림 4〉 실체뷰 수에 따른 성능 비교

이 그림에서 보는 바와 같이 case 2의 경우 실체뷰의 개수가 3개 이하인 경우에는 각 실체뷰의 접근확률에 큰 차이가 없어 case 1과 거의 유사한 성능을 보인다. 하지만 실체뷰의 개수가 4개 이상인 경우에는 접근확률의 차이가 있고, 접근확률이 높은 실체뷰가 빠른 존에 저장되므로 case 1에 비해 case 2의 성능이 더 좋은 것을 확인할 수 있다. 하지만 실체뷰간의 접근확률의 차이가 크지 않으므로 성능개선의 효과 역시 크기 않음을 알 수 있다. 하지만 case 3의 경우와 같이 각 실체뷰별 접근확률의 차이가 큰 경우에는 case 1과 case 2에 비해 성능이 크게 향상된다는 것을 알 수 있다. 따라서 본 논문에서 제시한 <그림 2>의 알고리즘은 실체뷰간의 접근확률의 차이가 클수록 그 효과가 더 커진다는 것을 이 실험을 통해 확인할 수 있다.

추가적으로 case 1과 case 2의 경우 실체뷰의 개수가 증가할수록 성능이 좋아지지만 4개 이상의 경우는 성능이 거의 일정하다는 것을 알 수 있다. 이 결과는 [1]에서의 실험결과와 동일한 것으로 모든 서브큐브를 실체뷰로 저장할 필요가 없다는 [1]에서의 주장을 뒷받침하고 있다. case 3의 경우도 비슷한 현상을 보이지만 이 경우는 실체뷰의 개수가 6개까지는 꾸준한 성능 향상을 나타내고 있다.

다음 실험은 <그림 1>의 데이타 큐브 구조에서 case 1~case 3의 각 경우에 대해 실체뷰의 크기에 따른 성능의 변화를 측정하였다. 실체뷰는 *P-E-C, P-E, C, E*의 4개만을 저장하였다. 이 실험에서도 case 1부터 case 3까지 각각의 경우에 대해 10,000개의 질의를 임의로 발생시켜 각 질의의 평균 실행시간을 관찰했으며 그 결과는 <그림 5>와 같다.

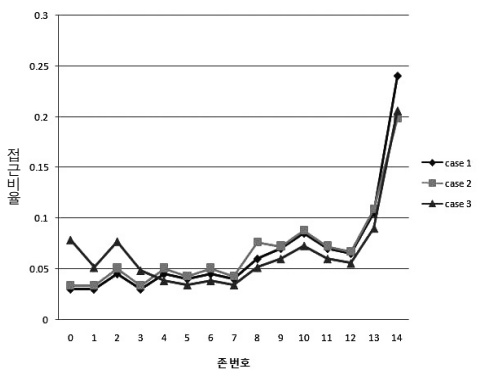


〈그림 5〉 실체뷰의 크기에 따른 성능 비교

이 그림에서 보는 바와 같이 <그림 4>의 경우처럼 case 3이 가장 뚜렷하게 좋은 성능을 보이며, case 2는 case 1에 비해 약간 좋은 성능을 보이고 있다. 또한 모든 경우에 대해서 실체뷰의 크기와 질의 실행시간은 선형적인 관계를 갖고 있다. 그 이유는 기본적으로 제 2.2절에서 언급한 바와 같이 본 논문에

서는 선형비용모델에 기반을 두고 실험을 실시했기 때문이다.

<그림 6>은 case 1~case 3의 각 경우에 대해 각 존의 접근비율을 보여준다. 이 실험에서도 실체뷰는 *P-E-C, P-E, C, E*의 4개만을 저장하였고, 실체뷰 *P-E-C*를 기준으로 5GB 크기를 가정하였으며 10,000개의 임의의 질의를 실행하여 측정하였다. 이 그림에서 보는 바와 같이 모든 경우에 대해 존 번호가 커질수록 접근비율이 높게 나타나게 된다. 그 이유는 <표 2>에서 보는 바와 같이 존 번호가 커질수록 저장용량이 커지므로 그 비율에 맞춰 저장되는 데이타의 양도 많아지기 때문이다. 또한 빠른 존으로 알려진 존 번호 0에서 3까지는 case 3의 접근비율이 가장 높으며 case 1의 접근비율이 가장 낮다는 것을 알 수 있다. 그 이유는 case 3과 case 2의 경우 접근확률이 높은 실체뷰를 빠른 존에 우선적으로 저장하였으므로 접근비율도 case 1에 비해 당연히 높게 나타나기 때문이다. 이 결과는 질의 성능에 직접적인 영향을 미치게 되며, <그림 4>와 <그림 5>의 결과도 이러한 접근비율의 영향으로 나타난 결과라고 해석할 수 있다.



〈그림 6〉 각 존의 접근비율

4.3 동적 환경에서의 실체뷰 배치 기법에 대한 성능

<그림 3>에서 제시한 동적환경에서의 실체뷰 배치 알고리즘은 각 존의 활용정도를 균일하게 유지하는데 목적이 있다. 따라서 본 논문에서는 실체뷰의 크기가 커짐에 따라 각 존에 배치된 페이지들의 변화를 측정하는 실험을 수행하였다. 실험은 case 2의 경우에 대해서 실시하였고, 실체뷰는 *P-E-C, P-E, c, E*의 4개만을 저장하였으며 초기의 실체뷰는 *P-E-C*를 기준으로 5GB 크기를 가정하였다.

첫 번째 실험은 실체뷰 *P-E-C*의 크기가 증가함에 따라 그 내용에 관계없이 나머지 실체뷰들도 같은 비율도 증가하는 경우를 가정하였고 그 결과는 <그림 7(a)>과 같다. 이 그림에서 *x*축은 존 번호를 나타내며 *y*축은 각 존에 대한 *zui*값의 변화를 나타낸다. 각 그래프는 초기의 실체뷰 크기를 기준으로 한 증가비율을 나타낸다. 이 그래프에서 보는 바와 같이 모든 실체뷰들의 크기가 같은 비율로 증가하는 경우에는 *zui*의 값이 꾸준히 1을 유지하는 것을 알 수 있다. 따라서 동적환경에서도 제 3.1절에서 제시한 (조건 3)을 만족한다는 것을 실험을 통해 확인할 수 있다.

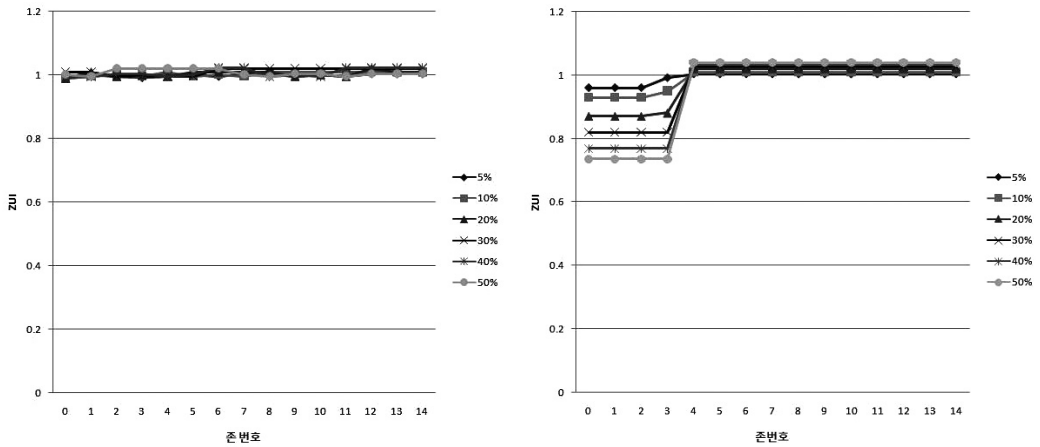
다음은 특정 차원만 증가하는 경우에 대한 실험을 실시하였다. <그림 7(b)>는 실체뷰 *P-E-C*에서 고객 차원(*C*)만이 추가될 경우 *zui*의 변화를 보여준다. 이 실험에서 접근확률이 비교적 높은 *P-E, E, C*는 존 번호 0~3에 배치되고 *P-E-C*는 존 번호 3번의 일부와 나머지 존에 배치되게 된다. 이때 실체뷰 *P-E-C*에서 차원 *C*의 값이 계속 추가된다 하더라도 *P-E*와 *E*의 크기에는 영향이 없어

크기가 그대로 유지하는 반면 *P-E-C*와 *C*의 크기가 증가하게 된다. 더구나 실체뷰 *C*는 크기가 매우 작아서 *zui*값에는 영향을 거의 미치지 않으므로 결과적으로 실체뷰 *P-E-C*의 크기가 증가하면 존 번호 3번 이후의 존들에 페이지가 추가되며 그 이외의 존에는 추가되는 페이지들이 거의 없게 된다.

따라서 이 실험에서는 <그림 7(b)>에서 보는 바와 같이 데이타가 추가됨에 따라 존 번호 4번 이후의 *zui*값은 점점 커지는 반면 그 이외의 존들은 반대로 *zui*값이 상대적으로 작아지게 된다. 문제는 이와 같이 특정 차원의 값들이 계속적으로 증가하게 될 경우 (조건 3)을 점차 위배하는 결과를 낳게 된다는 것이다. 이 경우에는 특정 존에 대한 *zui* 값이 일정 이상 증가하거나 줄어들 경우 디스크 배치를 전면 재구성하는 방안을 고려해야할 것으로 판단된다. 단, 본 논문에서는 제시하지 않았지만 사원(*E*)이나 상품(*P*)차원의 크기가 증가할 경우에는 <그림 7(a)>와 같이 *zui*는 거의 1로 유지된다. 그 이유는 실체뷰 *P-E-C*의 크기가 증가할수록 *P-E*도 같은 비율로 증가하게 되어 전체적으로 대부분의 존들이 *zui*값을 일정하게 유지할 수 있기 때문이다.

5. 결 론

본 논문에서는 OLAP 환경에서 질의 처리 성능을 향상시키기 위해서 실체뷰를 다중 존 디스크에 효율적으로 저장하는 기법을 제안하였다. 제안된 기법은 OLAP의 응용에 따라 정적 환경과 동적 환경을 모두 고려하여 설계되었다. 제안된 방법은 디스크 존들이 균일한 활용도를 가지며 자주 접근되는 실체뷰를 보다 빠른 존에 할당하는 방법을 사용하였다. 또한 이 기법들은 질의 처리 시간에 존 간의 이동을 최소화하는 지역화 개념을 적용함으로써 질의 성능을 극대화할 수 있었다. 제안



(a) 모든 차원이 동등하게 증가하는 경우 (b) 차원 *C*가 증가할 경우

〈그림 7〉 데이타가 증가할 경우 *zui*의 변화

된 기법들은 실제 데이타를 비롯한 다양한 종류의 실험을 통해 기존의 방법에 비해 획기적으로 질의 성능을 높인다는 사실을 확인할 수 있었다. 본 논문의 실험에서는 3차원 데이타 큐브를 이용하여 실시하였다. 비록 3차원에 대해서만 실시하였지만 본 논문에서 제안한 방식은 차원의 수에 관계없이 적용가능하다. 그 이유는 본 논문의 기본적인 아이디어가 실체뷰간의 접근확률에 대한 차이에 기반을 두기 때문이다. 따라서 차원의 수가 높다하더라도 실체뷰간의 접근확률에 대한 차이가 발생하면 어느 정도 성능향상 기대할 수 있으며, 접근확률의 차이가 클수록 성능의 편차도 그 만큼 커지게 된다.

본 논문에서는 데이터 큐브 구조에서 서브큐브들에 대한 질의확률이 사전에 알려져 있다는 가정 하에 실체뷰 배치 방법을 제안하였다. 본 논문의 제안방식은 실체뷰 선택 기법과 밀접한 관계가 있으며, 기존의 실체뷰 선택에 관련된 대부분의 연구에서도 서브큐브들에 대한 질의확률을 사전에 예측할 수 있다는 가정 하에 진행되었다[2, 3, 5, 6, 7, 8]. 따라서 서브큐브들에 대한 질의확률을 사전에 예측하지 못하거나, 도중에 변경되는 경우에는 본 논문의 방식을 적용하기에는 한계가 있으며, 이 경우 다른 방식의 접근이 필요하다.

마지막으로 본 논문에서는 하나의 버퍼만을 가정한 실험 결과를 제시하였다. 하지만 존을 고려한 페이지 저장 기술은 버퍼링 기법과 연관되어 고려될 수 있다. 예를 들어 접근확률이 크면서 크기가 작은 실체뷰는 버퍼에 상주시킬 수 있으므로 가장 느린 존에 저장하는 것이 오히려 가장 좋은 성능을 보여줄 수도 있다. 따라서 본 논문에서 제시한 알고리즘은 실체뷰에 대한 버퍼링 알고리즘이나 예상 버퍼링, 버퍼 사이즈 등의 요소에 따라 여러 가지 방법으로 변화될 수 있으며 이에 대한 후속 연구가 요구된다.

참 고 문 헌

[1] V. Harinarayan, Rajaraman, A., and Ullman, J., “Implementing Data Cubes Efficiently,” Proc. of ACM SIGMOD, 1996.

[2] H. Gupta, “Selection of Views to Materialize in a Data Warehouse,” Proc. of ICDT, 1997, pp. 98-112.

[3] H. Gupta, Harinarayan, V., Rajaraman, A., and Ullman, J., “Index Selection for OLAP,” Proc. of ICDE, 1997, pp. 208-219.

[4] Y. Kotidis and Roussopoulos, N., “Dynamat : A Dynamic View Management System for Data Warehouses,” Proc. of ACM SIGMOD, 1999, pp. 371-382.

[5] H. Mistry, P. Roy, S. Sudarshan, K. Ramamritham, “Materialized View Selection and Maintenance Using Multi-Query Optimization,” Proc. of ACM SIGMOD, 2001, pp. 310-318.

[6] A. Shukla, Deshpande, P. M., and Nau-

ghton, J. F., “Materialized View Selection for Multidimensional DataSet,” Proc. of VLDB, 1998, pp. 488-499,

[7] S. Agrawal, Chaudhuri, S., and Narasayya, V., “Automated Selection of Materialized Views and Indexes for SQL databases,” Proc. of VLDB, 2001, pp. 59-68.

[8] C. A. Dhote and ALi, M. S., “Materialized View Selection in Data Warehouse,” Proc. of International Conference on Information Technology, 2007.

[9] C. Ruemmler. and Wilkes, J., “An Introduction to Disk Drive Modeling,” IEEE Computer, March 1994.

[10] S. W. Ng, “Advances in Disk Technology : Performance Issues,” IEEE Computer Magazine, 1998, pp. 75-81.

[11] Oracle and 3PAR. “Simplified database storage management that lowers management costs and yields high storage utilization”. White paper, http://www.oracle.com/technology/products/database/asm/pdf/oracle_3par_wp_final.pdf 2004.

[12] Yu, B. and Kim, S.-H. “An Efficient Zoning Technique for Multidimensional Access Methods,” Proc. the VLDB Workshop on Trends in Enterprise Application Architecture, LNCS 3888, 2005, pp. 129-143.

[13] Yu, B. and Kim, S.-H. “Zoning Multidimensional Access Methods for Analytical Database Applications,” Proc. the 3rd International Conference on Computer Science and its Applications, 2005, pp. 191-196.

[14] Kim, S.-H., Yu, B., and Chang, J.-Y., “Zoned-partitioning of tree-like access methods,” Information Systems, Vol. 33, 2008, pp. 315-331.

[15] Seagate, Seagate Cheetah X15 FC disk drive ST318451FC/FCV product manual, Vol. 1, Document number 83329486, June 2000.

[16] Seagate, Barracuda 7200.7 Plus Serial ATA Publication number : 100270024, Publication number : 100270024, Rev. N, September 2005.

저 자 소 개

장재영	(E-mail : jychang@hansung.ac.kr)
1992년	서울대학교 계산통계학과 (학사)
1994년	서울대학교 계산통계학과 전산과학전공 대학원 (석사)
1999년	서울대학교 계산통계학과 전산과학전공 대학원 (박사)
2000년~현재	한성대학교 컴퓨터공학과 부교수
관심분야	데이터베이스, 데이터마이닝