다차원 온라인 분석처리에서 분리-포함 분할 다차원 파일 구조를 사용한 원-패스 집계 알고리즘

(A One-Pass Aggregation Algorithm using the Disjoint-Inclusive Partition Multidimensional Files in Multidimensional OLAP)

이 영 구 † 문 양 세 †† 황 규 영 †††
(Young-Koo Lee) (Yang-Sae Moon) (Kyu-Young Whang)

요 약 다차원 온라인 분석처리(Multidimensional On-Line Analytical Processing: MOLAP)에서 집계 연산은 중요한 기본 연산이다. 기존의 MOLAP 집계 연산은 다차원 배열 구조를 기반으로 한 파일 구조에 대해서 연구되어 왔다. 이러한 파일 구조는 편중된 분포를 갖는 데이타에서는 잘 동작하지 못한다는 단점이 있다. 본 논문에서는 편중된 분포에도 잘 동작하는 다차원 파일 구조를 사용한 집계 알고리즘을 제안한다. 먼저, 새로운 **분리-포함 분할**이라는 개념을 사용한 집계 연산 처리 모델을 제안한다. 집계 연산 처리에서 분리-포함 분할 개념을 사용하면 페이지들의 액세스 순서를 미리 알아 낼 수 있다는 특징을 가진다. 그리고, 제안한 모델에 기반하여 **원-패스 버퍼 크기**(*one-pass buffer size*)를 사용하여 집계 연산을 처리하는 원-패스 집계 알고리즘을 제안한다. 원-패스 버퍼 크기란 페이지 당 한 번의 디스크 액세스를 보장하기 위해 필요한 최소 버퍼 크기이다. 또한, 제안한 집계 연산 처리 모델 하에서 제안된 알고리즘이 최소의 원-패스 버퍼 크기를 갖는다는 것을 증명한다. 마지막으로, 많은 실험을 통하여 이론적으로 구한 원-패스 버퍼 크기가 실제 환경에서 정확히 동작함을 실험적으로 확인하였다. 이 알고리즘은 미리 알려진 페이지 액세스 순서를 이용하는 버퍼 교체 정책을 사용함으로써 최적의 원-패스 버퍼 크기를 달성한다. 제안하는 알고리즘은 여러 집계 질의가 동시에 요청되는 다사용자 환경에서 특히 유용하다. 이는 이 알고리즘이 정규화된 디스크 액세스 횟수를 1.0으로 유지하기 위해 반드시 필요한 크기의 버퍼만을 사용하기 때문이다.

Abstract Aggregation is an operation that plays a key role in multidimensional OLAP (MOLAP). Existing aggregation algorithms in MOLAP have been proposed for file structures such as multidimensional arrays. These file structures do not work well with skewed distributions. In this paper, we present an aggregation algorithm that uses multidimensional files adapting to a skewed distribution. We first present an aggregation computation model that uses the new notion of disjoint-inclusive partition. Using the *disjoint-inclusive partition* allows us to compute the order of accessing data pages in advance. Based on this model, we then present the one-pass dynamic aggregation algorithm. This algorithm computes aggregations using the *one-pass buffer size*, which is the minimum buffer size required for guaranteeing one disk access per page. We prove that our aggregation algorithm is optimal with respect to the one-pass buffer size under our aggregation computation model. Finally, we demonstrate that the one-pass buffer size theoretically derived is indeed correct in real environments. We note that our algorithm achieves the optimal one-pass buffer size by using a buffer replacement policy that exploits the page access order computed in advance. Our algorithm is effective especially in a multiuser environment, where many aggregation queries are requested concurrently, because it uses only those buffer pages that are essential to maintain the normalized I/O access of 1.0.

* 본 연구는 첨단정보기술연구센터를 통하여 한국과학재단의 지원을 받았음
† 종신회원 : 한국과학기술원 전자전산학과
yklee@mozart.kaist.ac.kr
†† 학생회원 : 한국과학기술원 전자전산학과
ysmoon@mozart.kaist.ac.kr
††† 종신회원 : 한국과학기술원 전자전산학과 교수
첨단정보기술연구센타 소장
kywhang@cs.kaist.ac.kr
논문접수 : 2000년 12월 29일
심사완료 : 2001년 4월 11일

1. 서 론

온라인 분석처리(On-Line Analytical Processing: OLAP)는 사용자가 의사 결정에 필요한 지식을 찾아내기 위해 대량의 데이타를 쉽게 분석할 수 있도록 도와주는 데이타베이스 응용이다[1]. 의사 결정에 있어서는 개별적인 레코드들보다 레코드들을 요약한 경향이 중요하기 때문에 상당수의 OLAP 질의들이 데이타를 요약하는데 사용되는 집계(aggregate) 연산을 포함하고 있다. 그런데, 집계 연산들은 처리 비용이 매우 큰 연산이기 때문에 집계 연산의 처리 성능은 OLAP 시스템의 성능에 큰 영향을 미치는 중요한 요소이다[2,3,4,5].

OLAP에서는 데이타를 다차원 배열로 모델링하는 다차원 데이타 모델을 사용한다[1]. 다차원 데이타 모델은 데이타를 분석의 대상이 되는 **측정값**(*measure*)들과 측정값을 결정하는 **차원**(*dimension*)으로 구분한다. 그리고, 각 차원은 다차원 배열의 하나의 축(axis)으로 대응시키고 측정값은 배열의 셀(cell)에 저장된 값으로 대응시킨다. 다차원 데이타 모델은 차원들의 값의 변화에 따른 측정값의 변화를 분석하는 OLAP 사용자의 논리적 사고 방식에 적합하다고 알려져 있다[1].

OLAP 시스템은 OLAP 데이타의 저장 방법에 따라 크게 관계형 OLAP(Relational OLAP: ROLAP)과 다차원 OLAP(Multidimensional OLAP: MOLAP)으로 구분된다[1]. ROLAP은 관계 데이타베이스 관리 시스템(Relational DBMS: RDBMS)을 기반 시스템으로 사용하는 것으로서, 테이블을 이용하여 OLAP 데이타를 저장한다. 반면에, MOLAP에서는 다차원 데이타를 효율적으로 저장, 관리할 수 있는 다차원 파일 구조를 사용하여 OLAP 데이타를 저장한다. 최근에는 다차원 파일 구조의 장점을 인식하고 ROLAP에서도 이를 이용하려는 연구가 진행되고 있다[3,5].

ROLAP에서의 집계 연산 처리에 대해서는 RDBMS에서의 기존 연구 결과를 포함하여 많은 연구 결과가 발표되었다[2,6]. 반면에, MOLAP에서의 집계 연산 처리에 대해서는 많은 연구가 이루어지지 않았으며, 대부분의 기존 연구는 선계산된(precomputed) 집계 연산 결과를 저장하여 두는 정적인 방법을 주로 사용한다. 그런데, 이 방법은 사용될 집계 연산들의 결과를 모두 저장해두기 때문에 저장 공간 오버헤드와 주기적인 갱신에 따른 오버헤드가 크다는 단점이 있다. 이러한 단점을 극복하기 위해서는 결과를 미리 저장하지 않고 집계 연산이 요구될 때마다 처리하는 동적인 방법이 필요하다.

MOLAP에서의 동적인 집계 연산 처리에서는 다차원 배열[5]과 압축된 다차원 배열[4]을 대상으로 한 집계 연산 처리 방법이 연구되었다. 그러나, 다차원 배열은 편중된(skewed) 분포를 갖는 데이타를 잘 처리하지 못하는 단점이 있으며, 압축된 다차원 배열은 클러스터링을 파괴하여 영역 질의 등 다른 OLAP 연산들의 성능이 저하되는 단점이 있다. 본 논문에서는 다차원 클러스터링 특성을 유지하면서 편중된 분포의 데이타들을 잘 처리할 수 있는 다차원 파일 구조를 사용하는 동적인 집계 알고리즘을 제시한다.

본 논문에서는 먼저 다차원 파일 구조에서 분리-포함 분할이라는 개념을 사용한 새로운 집계 연산 처리 모델을 제시한다. 그리고, 이 모델을 기반으로 **원-패스 버퍼 크기**(*one-pass buffer size*)를 사용하여 집계 연산을 처리하는 원-패스 동적 집계 알고리즘을 제안한다. 원-패스 버퍼 크기란 페이지 당 한 번의 디스크 액세스를 보장하기 위해 필요한 최소 버퍼 크기이다. 다음으로, 제안한 알고리즘의 원-패스 버퍼 크기를 유도하고, 제안한 집계 연산 처리 모델 하에서 이 원-패스 버퍼 크기가 최소임을 증명한다. 즉, 제안한 처리 모델을 따르는 어떤 알고리즘도, 버퍼 교체 정책 및 페이지 액세스 순서에 상관 없이, 제안한 알고리즘보다 작은 원-패스 버퍼 크기를 가질 수 없다. 마지막으로, 이 알고리즘을 다차원 파일 구조인 계층 그리드 파일(Multilevel Grid File: MLGF)[7,8]을 이용하여 구현하고, 많은 실험을 통하여 이론적으로 구한 원-패스 버퍼 크기가 실제 환경에서 정확히 동작한다는 것을 실험적으로 확인한다. MLGF는 편중된 분포의 데이타들을 잘 처리하며 [7,8], 본 논문에서 제안하는 분리-포함 분할 조건을 만족하는 다차원 파일 구조이다.

본 논문의 구성은 다음과 같다. 제2절에서는 연구 배경으로서 다차원 파일 구조와 버퍼 교체 정책을 간략히 소개한다. 제3절에서는 다차원 파일 구조를 이용한 집계 연산 처리에 대한 동기를 설명하고, 다차원 파일 구조를 이용한 일반적인 집계 연산 처리 방법을 제시한다. 제4절에서는 집계 연산 처리 모델을 제시하고, 제5절에서는 이 모델에 따른 원-패스 집계 알고리즘을 제안한다. 제6절에서는 원-패스 버퍼 크기에 대한 실험 결과를 제시한다. 마지막으로, 제7절에서는 결론을 내린다.

2. 연구 배경

2.1 다차원 파일 구조

먼저 다차원 파일 구조에 관련된 용어들을 정의한다[7]. **파일**은 **속성**들의 리스트로 구성된 **레코드**들의 모

임이다. 파일의 구조는 레코드를 구성하는 속성들 중에서 일부에 의해서 결정되며, 이와 같이 파일을 구성하는데 참여하는 속성들을 **구성 속성**(*organizing attribute*)이라 한다. 그리고, 두 개 이상의 구성 속성을 가지는 파일 구조를 **다차원 파일 구조**라 한다. **도메인**은 한 속성이 취할 수 있는 모든 값들의 집합이며, 다차원 파일 구조 내에서 하나의 축에 해당된다. 모든 속성에 대한 도메인들의 카티젼 곱(Cartesian product)을 **도메인 공간**(*domain space*)이라 정의하고, 도메인 공간의 일부분을 **영역**(*region*)이라 한다. 그리고, 도메인 공간에서 페이지 P가 나타내는 영역을 **페이지 영역**(*page region*)이라 정의하고, $\bar{P}$로 표시한다.

다차원 파일 구조는 다차원 클러스터링을 지원함으로써 여러 속성에 의한 검색, 즉 다중키 액세스를 효율적으로 처리한다. 다차원 클러스터링이란 각 구성 속성의 값이 서로 유사한 레코드들을 인접하게 저장하는 것을 말한다. 다차원 클러스터링을 지원하기 위하여 다차원 파일 구조는 도메인 공간을 여러 개의 영역으로 분할하고 각 영역에 속하는 레코드들을 하나의 페이지에 저장한다. 그리고, 도메인 공간의 분할 상태는 디렉토리에 저장한다.

다차원 파일 구조는 영역의 분할시 분할 경계를 정하는 기준에 따라 크게 두 가지로 분류된다. 하나는 영역에 포함된 레코드의 개수를 균등하게 양분하는 속성 값을 분할 경계로 하는 **레코드 기준 분할**(*record-oriented splitting*) 전략을 사용하는 파일 구조로서, 분할 경계가 레코드의 분포에 따라 유동적이다. 또다른 하나는 영역의 크기를 반분하는 위치를 분할 경계로 하는 **영역 기준 분할**(*region-oriented splitting*) 전략을 사용하는 파일 구조로서, 분할 경계가 레코드의 분포에 상관없이 고정적이다. 영역 기준 분할 전략을 사용하는 파일 구조는 영역 분할의 경계가 항상 고정되므로, 파일 구조가 레코드의 입력 순서에 관계없이 일정하게 유지되고, 영역 분할 전략에 따라 영역의 모양을 쉽게 조정할 수 있는 특징이 있다.

2.2 버퍼 교체 정책

버퍼 교체 정책은 새로운 페이지를 버퍼에 읽어오기 위해 버퍼에 상주하고 있는 페이지들 중에서 쫓아낼 희생자(victim)를 선택하는 정책이다[9]. 현재까지 제안된 대표적인 버퍼 교체 정책들은 LRU[10], CLOCK[9], LRU-k[11] 등이 있다. 이들은 공통적으로 버퍼 폴트율을 최소화하기 위해서 앞으로 예상되는 액세스 시점이 가장 먼 페이지를 희생자로 선정하는 교체 전략을 사용한다. LRU의 경우 가장 마지막으로 발생한 액세스 시점을 사용하여 앞으로 액세스될 시점을 예측하며[10], LRU-k는 LRU의 일반화된 알고리즘으로서 가장 마지막 k개의 액세스 시점을 사용한다[11]. CLOCK은 LRU의 단순화된 알고리즘으로서 참조 비트[9]를 활용한다. 이러한 교체 정책들은 향후 페이지 액세스 순서를 모르는 상황에서의 버퍼 교체 정책이다.

페이지들의 액세스 순서가 미리 알려져 있으면, 그 순서를 이용해서 버퍼의 효과를 높일 수 있다. 미리 알려진 페이지들의 액세스 순서를 이용하는 버퍼 교체 정책에는 Belady의 B_0 정책[10]과 즉석-토스 정책[12]이 있다. B_0 정책은 페이지 액세스 순서를 모두 알 수 있을 때 사용할 수 있으며, 최적의 버퍼 교체 정책으로 증명되었다[10]. 이 정책은 앞으로 액세스 시점이 가장 먼 페이지를 희생자로 선정하는 교체 전략을 사용한다. 반면에, 즉석-토스 정책은 페이지 액세스 순서를 모두 알지는 못하더라도 주어진 시점에서 앞으로 액세스되지 않을 페이지를 알 수 있을 때 사용할 수 있다. 이 정책은 페이지를 액세스 한 후 해당 페이지가 더 이상 액세스되지 않는다면 그 페이지를 즉시 버퍼에서 쫓아낸다. 더 이상 액세스되지 않는 페이지가 없는 경우는 나머지 페이지들 중에서 희생자를 선정하여야 하는데, 이 경우를 위해 즉석-토스 정책은 LRU, CLOCK 등과 같은 일반적인 버퍼 교체 정책과 함께 사용된다. 그런데, 일반적으로는 페이지 액세스 순서를 미리 알 수 없기 때문에 B_0 또는 즉석-토스 정책은 이론적으로만 의미가 있을 뿐 실제 응용에서는 적용이 어려운 단점이 있다. 본 논문에서는 이들 정책이 집계 연산을 처리함에 있어 유효하게 사용될 수 있음을 보인다.

3. 다차원 파일 구조를 이용한 집계 연산 처리

본 절에서는 다차원 파일 구조를 이용한 집계 연산 처리에 대한 연구 동기를 설명하고, 다차원 파일 구조를 이용한 일반적인 집계 연산 처리 방법을 제시한다. 제3.1절에서는 필요한 용어들을 정의한다. 제3.2절에서는 다차원 파일 구조를 이용한 일반적인 집계 연산 처리 방법을 제시한다. 제3.3절에서는 집계 연산 처리에서 버퍼의 필요성에 대해 논의한다.

3.1 용어

집계 연산은 데이타들을 주어진 속성들의 값에 따라 여러 개의 그룹으로 나눈 후 주어진 집계 함수를 적용하여 각 그룹당 하나씩의 값을 구하는 연산이다[6]. 구성 속성 중 집계 연산에서 레코드들을 여러 개의 그룹으로 나누는 기준이 되는 속성을 **그룹화 속성**(*grouping*

attribute)이라 정의하고, 집계 함수가 적용되는 속성, 즉 분석의 대상이 되는 측정값을 나타내는 속성을 **집계 속성**(*aggregated attribute*)이라 정의한다. 그리고, 그룹화 속성들의 도메인의 카티젼 곱을 그룹화 **도메인 공간**(*grouping domain space*)이라 정의하며, 그룹화 도메인 공간의 일부분을 **그룹화 영역**(*grouping region*)이라 정의한다. 다음으로, 집계 연산을 위하여 그룹화 도메인 공간을 한 개 이상의 그룹화 영역으로 분할한 경우, 이를 **집계 윈도우**(*aggregation window*)라 정의한다. 집계 윈도우를 대상으로 한 집계 연산을 **부분집계 연산**(*partial aggregation operation*)이라 정의한다.

페이지 P의 페이지 영역 $\tilde{P}$를 그룹화 속성들의 집합이 G인 그룹화 도메인 공간으로 프로젝션한 결과를 G에 대한 P의 **페이지 그룹화 영역**(*page grouping region*)이라 정의하고, 이를 $\Pi_G\tilde{P}$라 표시한다. 그리고, 페이지 영역 P가 영역 Q와 겹칠 때 **간단히 페이지 P와 영역 Q가 겹친다**고 말한다. 또한, 페이지 그룹화 영역 $\Pi_G\tilde{P}$가 집계 윈도우 W와 겹칠 때 페이지 **P와 집계 윈도우 W가 겹친다**고 말한다. 마지막으로, 집계 윈도우 W와 겹치는 페이지를 W의 **부분집계 페이지**(*partial aggregation page*)라 정의한다.

예 1: 그림 1은 X, Y, Z 세 개의 구성 속성으로 이루어진 다차원 파일 구조에 대한 집계 연산의 예를 나타낸다. 도메인 공간은 모두 여섯 개의 영역으로 분할되어 있으며, 각 영역에 속하는 레코드들은 A, B, C, D, E, F의 페이지에 저장되어 있다. 그림에서, X와 Y는 그룹화 속성이고, 그룹화 도메인 공간은 X:[0,99]×Y:[0,99]이며, 이 공간의 일부분이 그룹화 영역이다. 그리고, 집계 연산을 위해 분할된 X:[0,49]×Y:[0,49], X:[0,49]×Y:[50,99], X:[50,99]×Y:[0,49], X:[50,99]×Y:[50,99]의 네 개의 그룹화 영역이 집계 윈도우이다. 그림 1에서

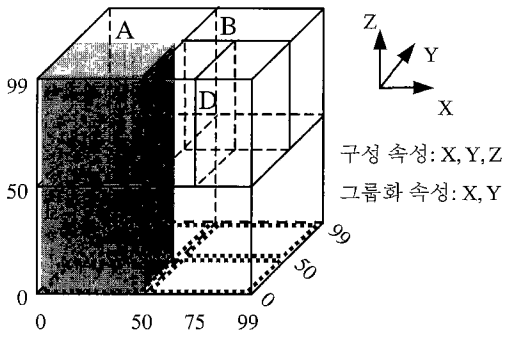


그림 1 집계 연산 처리 방법의 예

집계 윈도우 X:[0,49]×Y:[0,49]의 부분집계 페이지들은 A와 E이다.

3.2 일반적인 집계 연산 처리 방법

집계 연산 처리의 가장 간단한 방법은 그룹화 도메인 공간을 하나의 집계 윈도우로 보고 집계 연산을 처리하는 방법으로, 그 절차는 다음과 같다. 먼저 주기억장치에 (그룹화 속성 값, 결과 값)의 엔트리로 구성되는 결과 테이블을 하나 준비한다. 그리고, 파일을 한번 스캔하면서, 검색된 각 레코드에 대해 그룹화 속성들의 값을 키로 하여 결과 테이블의 해당 엔트리를 찾아 집계 속성의 값을 엔트리의 결과 값에 집계한다. 이때, 해당 엔트리가 없으면 새로운 엔트리를 하나 삽입한다. 이 방법은 파일을 한번만 스캔하면서 집계 연산을 처리한다는 장점이 있다. 그러나, 주기억장치 크기의 한계로 인해 대용량 데이타베이스에는 유용하지 못하다.

주기억장치 크기의 한계 문제를 해결하기 위해서, 그룹화 도메인 공간을 여러 개의 집계 윈도우로 나누어 집계 연산을 처리하는 방법을 사용할 수 있다. 즉, 그룹화 도메인 공간을 대상으로 하는 집계 연산을 집계 윈도우를 대상으로 하는 부분집계 연산들로 나누어 수행하는 것이다. 이것이 가능한 이유는 서로 다른 집계 윈도우에 속하는 레코드들은 그룹화 속성들의 값이 달라서 서로 다른 그룹에 속하므로, 하나의 부분집계 연산의 결과는 다른 부분집계 연산의 결과에 영향을 미치지 않기 때문이다. 집계 윈도우의 크기가 작아지면, 부분집계 연산의 결과 크기도 작아진다. 따라서, 결과 테이블의 크기가 주어졌을 때, 부분집계 연산들의 결과가 결과 테이블에 들어가도록 집계 윈도우들을 선택하여 집계 연산 처리에 사용할 수 있다.

다차원 파일 구조는 집계 윈도우에 속하는 레코드들을 효율적으로 검색할 수 있다. 그 이유는 다차원 파일 구조들은 다차원 클러스터링 특성을 지원함으로써 영역 질의들을 효과적으로 처리하기 때문이다. 반면에, 관계형 데이타베이스에서 사용되는 테이블 저장 구조는 다차원 클러스터링을 지원하지 않기 때문에, 집계 윈도우에 속하는 레코드들을 검색하는데 비효율적이다. 따라서, 본 논문에서는 다차원 파일 구조의 영역 질의를 사용하여 집계 연산을 효율적으로 처리하는 집계 알고리즘을 제안한다.

그림 2는 위 아이디어에 기반한 집계 알고리즘 General_Aggregation을 나타낸다. 단계 1은 그룹화 도메인 공간을 여러 개의 집계 윈도우로 분할한다. 단계 2는 각 집계 윈도우를 하나씩 순회하면서 부분집계 연산을 수행한다. 단계 2.1은 부분집계 연산에 사용될 영역

질의를 구성한다. 사용되는 영역 질의는 그룹화 속성들에 대해서는 해당 집계 윈도우를 대상으로 하고, 나머지 속성들에 대해서는 해당 속성의 전체 도메인을 범위로 한다. 단계 2.2와 2.3에서는 영역 질의로 집계 윈도우에 속하는 레코드들을 탐색하면서 부분집계 연산을 수행한다. 부분집계 연산의 중간 결과는 주기억장치의 결과 테이블에 유지된다.

알고리즘 General_Aggregation
입력: (1) OLAP 데이타를 저장하고 있는 다차원 파일 구조 *md-file*
(2) 그룹화 속성들의 집합 G
(3) 집계 속성 A
출력: 집계 연산 결과
알고리즘:
1 그룹화 도메인 공간을 집계 윈도우들로 분할한다.
2 각 집계 윈도우에 대해서 다음을 수행한다.
2.1 영역 질의를 구성한다. 여기에서 질의 영역은 그룹화 속성들에 대해서는 집계 윈도우를 대상으로 하고, 다른 속성에 대해서는 전체 도메인을 대상으로 한다.
2.2 영역 질의로 *md-file*을 검색한다.
2.3 검색된 각 레코드에 대해 그룹화 속성들(G)의 값을 키로 하여 결과 테이블의 해당 엔트리를 찾아 집계 속성(A)의 값을 엔트리의 결과 값에 집계한다.

그림 2 다차원 파일 구조를 이용한 집계 알고리즘 General_Aggregation

예 2: 집계 연산 처리 방법을 그림 1의 예를 사용하여 설명하면 다음과 같다. 우선, 그룹화 도메인 공간 X:[0,99]×Y:[0,99]를 X:[0,49]×Y:[0,49], X:[0,49]×Y:[50,99], X:[50,99]×Y:[0,49], X:[50,99]×Y:[50,99]의 네 개의 집계 윈도우로 분할한다. 그리고, 영역질의를 사용하여 각 집계 윈도우에 대한 부분집계 연산을 수행한다. 예를 들어, 집계 윈도우 X:[0,49]×Y:[0,49]에 해당하는 부분집계 연산은 X:[0,49], Y:[0,49], 그리고 Z:[0,99]로 구성된 영역 질의를 사용하여 수행된다. 즉, 그림 1에서 음영으로 표시된 막대에 속하는 레코드들을 액세스함으로써 X:[0,49]×Y:[0,49]에 해당하는 부분집계 연산이 처리된다.

알고리즘 General_Aggregation의 단계 1에서 집계 윈도우들을 결정할 때, 부분집계 연산들의 결과 크기가 비슷해지도록 집계 윈도우들을 선택하는 것이 바람직하다. 그 이유는 각 부분집계 연산의 결과를 결과 테이블에 저장하기 위해서는 결과 테이블의 크기가 부분집계 연산의 결과 크기 중 최대치가 되어야 하는데, 부분집계 연산들의 결과 크기를 비슷하게 함으로써 이 최대치를 줄일 수 있기 때문이다. 본 논문에서는 히스토그램을 사용하여 집계 윈도우를 선택하는 방법을 사용한다.[1] 본 논문의 이후 논의에서는 집계 윈도우들은 결정되어 있다고 가정하고 단계 2에서의 효율적인 집계 연산 처리에 대해 다룬다.

3.3 집계 연산 처리에서의 버퍼의 필요성

알고리즘 General_Aggregation으로 집계 연산을 처리하면, 동일 페이지에 대하여 여러 번의 디스크 액세스가 발생할 수 있다. 알고리즘 General_Aggregation의 단계 2.2에서 각 페이지는 자신과 겹치는 집계 윈도우에 대한 집계 연산을 수행할 때마다 액세스된다. 그리고, 다차원 파일 구조에서는 페이지 영역들의 크기가 데이타 분포에 따라 다르므로 다양한 크기의 페이지 영역들이 존재한다. 따라서, 페이지 영역이 큰 페이지들은 여러 집계 윈도우들과 겹치게 되어 여러 번의 디스크 액세스가 발생한다.

디스크 액세스 횟수를 줄이기 위해서는 일반적으로 버퍼를 사용한다. 즉, 집계 연산 시에 읽은 페이지들을 버퍼에 관리함으로써 동일한 페이지에 대한 여러 번의 디스크 액세스를 줄일 수 있다. 버퍼를 사용할 때 버퍼의 효과를 높이기 위해서는 여러 번 액세스되는 페이지들이 가능한 짧은 간격으로 액세스되어야 한다. 따라서, 동일한 페이지와 겹치는 집계 윈도우에 대한 부분집계 연산들이 인접해서 일어날 수 있도록 하는 방법이 요구된다. 한편, 버퍼 관리 시 버퍼 교체 정책은 버퍼의 성능을 좌우하는 중요한 요소 중의 하나이다. 현재 데이타베이스 시스템에서 널리 사용되고 있는 LRU나 CLOCK과 같은 교체 정책들은 페이지 액세스 순서를 미리 알 수 없을 때 사용하는 일반적인 방법이다. 그런데, 다차원 파일 구조를 이용한 집계 연산에서는 집계 윈도우와 페이지 그룹화 영역들 간의 관계를 이용하면 페이지 액세스 순서들을 미리 알 수 있어 B_0나 즉석-토스와 같은 더 나은 교체 정책을 사용할 수 있다. 제 4절에서는 이러한 이슈들을 논의한다.

4. 분리-포함 분할 다차원 파일 구조를 이용한 집계 알고리즘

일반적으로 다차원 파일 구조의 페이지 영역들의 모양은 다양하며, 이로 인해 페이지 그룹화 영역들의 포함 관계는 복잡한 양상을 갖는다. 그러나, 페이지 그룹화 영역들의 포함 관계가 어떤 규칙적인 특성을 갖는 경우

1) 이에 대한 자세한 내용은 부록 A를 참조한다.

가 있으며, 이러한 경우에는 그 특성을 이용하여 집계 연산 처리의 성능을 높일 수 있다. 본 절에서는 분리-포함 관계 및 분리-포함 분할의 개념을 제안하고, 이 개념들을 다차원 파일 구조에 적용한 집계 연산 처리 모델을 제안한다. 그리고, 원-패스 버퍼 크기를 사용한 원-패스 집계 연산 처리 알고리즘을 제안한다. 또한, 이 알고리즘의 원-패스 버퍼 크기가 제안된 집계 연산 처리 모델 하에서 최적임을 증명한다.

4.1 분리-포함 분할 다차원 파일 구조

정의 1 두 개의 영역 R과 S가 조건식 (1)을 만족하면 R과 S는 **분리-포함 관계**(*disjoint-inclusive relationship*)를 갖는다라고 정의한다.

$$R \cap S \neq \emptyset \Rightarrow (R \supseteq S \vee R \subseteq S) \quad (1)$$

정의 2 구성 속성이 A_1, A_2, ..., A_n인 도메인 공간 D에서 영역들의 집합 $\mathbb{Q}=\{Q_1, Q_2, ..., Q_k\}$가 다음 두 가지 조건을 만족하면 $\mathbb{Q}$를 도메인 공간 D의 **분리-포함 분할**(*disjoint-inclusive partition: DIP*)이라 정의한다.

(1) $(\bigcup_{i=1}^{k} Q_i = \mathrm{D}) \wedge (Q_i \cap Q_j = \emptyset,\ i \neq j)$

(2) $\forall G \subseteq \{A_1, A_2, ..., A_n\}$, $\forall i, j (1 \leq i, j \leq k)$, $\Pi_G Q_i$와 $\Pi_G Q_j$는 분리-포함 관계를 갖는다.

그리고, 페이지 영역들이 분리-포함 분할을 구성하는 다차원 파일 구조를 **분리-포함 분할 다차원 파일 구조**라 정의한다.

정의 1의 분리-포함 관계는 두 영역 간에 서로 겹침이 발생한다면 반드시 한 영역이 다른 영역을 포함한다는 것을 의미한다. 정의 2에서 조건 (1)은 $\mathbb{Q}$가 도메인 공간 D의 분할이 되기 위한 필요충분 조건이다. 그리고, 조건 (2)는 $\mathbb{Q}$의 두 영역들을 임의의 G에 대한 Π_GD 공간으로 프로젝션 했을 때, 프로젝션된 두 영역이 분리-포함 관계를 만족해야 한다는 것이다.

예 3: 그림 3은 X, Y, Z 세 개의 구성 속성을 가진 다차원 파일 구조에서 발생할 수 있는 도메인 공간의 분할 예를 나타낸다. 그림 3(a)에서는 도메인 공간이 A~F의 여섯 개 영역으로 분할되어 정의 2의 조건 (1)을 만족한다. 또한, 그림에서와 같이 영역들을 G={X, Y}인 Π_GD 공간으로 프로젝션했을 때, 모든 프로젝션된 영역들이 분리-포함 관계를 만족한다. G 값으로 {X, Y} 대신 임의의 속성들의 조합을 사용하더라도 이러한 관계가 성립하므로, 그림 3(a)는 분리-포함 분할이다. 그림 3(b)에서는 A~E의 다섯 개 영역으로 분할되어 정의 2의 조건 (1)을 만족한다. 그러나, 그림에서와 같이 G={X, Y}일 때, Π_GD 공간에서 Π_GA와 Π_GD, 그리고 Π_GA와 Π_GE는 겹침이 발생하나 포함 관계는 성립하지 않는다. 따라서 3(b)는 분리-포함 분할이 아니다.

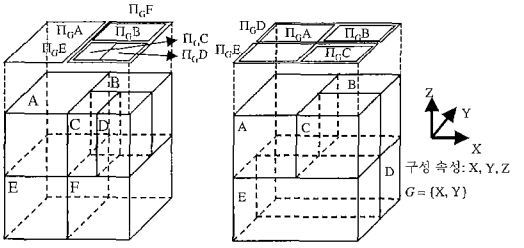


그림 3 분리-포함 분할의 예와 분리-포함 분할이 아닌 예

보조정리 1은 특정 조건을 만족하는 다차원 파일 구조들은 분리-포함 분할 특성을 가짐을 보여준다.

보조정리 1 다차원 파일 구조가 다음 분할 규칙 (1)과 (2)를 만족하면, 그 결과로 생기는 도메인 공간의 영역들의 집합은 분리-포함 분할이다.

(1) 영역 기준 분할 전략을 사용한다.

(2) 도메인 공간의 두 영역을 Q_i와 Q_j라 하고, Q_i가 만들어지기까지 사용된 분할 축들의 다중집합(multiset)을 SplitAxes(Q_i)라 할 때, SplitAxes(Q_i)⊆SplitAxes(Q_j) 또는 SplitAxes(Q_i)⊇SplitAxes(Q_j)가 성립한다.

증명: 부록 B 참조 □

보조정리 1의 분할 규칙 (2)의 특별한 경우로서, SplitAxes(Q_i)를 Q_i가 만들어지기까지 사용된 분할 축들의 시퀀스(sequence)라 할 때, SplitAxes(Q_i)와 SplitAxes(Q_j)에 대해 하나가 다른 하나의 접두어(prefix)가 되는 조건을 고려할 수 있다. 이 조건은 분할 규칙 (2)의 충분조건이 된다. 이러한 조건을 만족하는 예로는 분할 축을 순환적으로 선택하는 순환 분할(cyclic splitting)이 있다.

4.2 집계 연산 처리 모델

정의 3 다음 네 가지 조건을 만족하는 집계 연산 처리 모델을 *DIP-Computation-Model*이라 정의한다.

(1) 분리-포함 분할 다차원 파일 구조를 사용한다.

(2) 그룹화 도메인 공간에 대한 집계 연산을 집계 윈도우들에 대한 집계 연산으로 나누어 수행한다.

(3) 집계 윈도우들은 페이지 그룹화 영역들과 분리-포함 관계를 만족한다.

(4) 각 부분집계 연산을 다차원 파일 구조의 영역 질의를 사용하여 처리한다.

DIP-Computation-Model은 다차원 클러스터링 및 분리-포함 분할 특성을 갖는 다차원 파일 구조를 이용하여 집계 연산을 처리하기 위한 모델이다. 조건 (2)와 (4)는 다차원 클러스터링 특성을 이용하여 집계 연산을 처리하기 위한 조건이다. 이 조건을 이용한 집계 연산 처리 방법은 제3.2절에서 논의하였다. 조건 (1)과 (3)은

분리-포함 분할 특성을 이용하는 것으로서, 효율적인 집계 알고리즘의 개발에 유용하며 알고리즘의 이론적 분석을 위한 토대를 제공한다. 분리-포함 분할 다차원 파일 구조의 페이지 그룹화 영역들은 서로 분리-포함 관계를 만족한다. 그리고, 각 집계 윈도우를 한 개 이상의 페이지 그룹화 영역을 합한 영역으로 결정한다면 이들 집계 윈도우는 페이지 그룹화 영역들과 분리-포함 관계를 만족한다. 따라서, 조건 (3)과 같이 집계 윈도우들이 페이지 그룹화 영역들과 분리-포함 관계를 만족하도록 하는 것은 항시 가능하다.

4.3 원-패스 버퍼 크기의 하한

DIP-Computation Model을 따르는 원-패스 집계 알고리즘을 제시하기 전에, 본 절에서는 먼저 DIP-Computation-Model을 이용하여 집계 연산을 처리할 때 원-패스 버퍼 크기의 하나의 하한값(a lower bound)을 구한다. 이 하한값이란 DIP-Computation-Model을 따르는 어떤 알고리즘도 페이지 당 한번의 디스크 액세스를 보장하기 위하여 가져야 하는 최소한의 버퍼 크기보다 작거나 같은 하나의 버퍼 크기를 말한다. 이 하한값은 본 논문에서 제안한 알고리즘의 원-패스 버퍼 크기가 최적임을 증명하기 위해서 제5.1절에서 사용된다.

정의 4 페이지 P의 페이지 그룹화 영역을 R이라 하자. 이때, R이 최소한 하나 이상의 집계 윈도우를 진포함(proper inclusion)하면, P를 **L-페이지**(*large page*)라 정의한다. 그리고, 집계 윈도우 W가 R에 진포함되면($W \subset R$), P를 ***W*의 L-페이지**라 정의한다.

정리 1 DIP-Computation-Model을 이용하여 집계 연산을 처리한다고 하자. $\mathbb{W}=\{W_1, W_2, \cdots, W_k\}$가 집계 윈도우들의 집합이라고 하면, $\max_{W_i, 1 \le i \le k}\{(W_i$의 L-페이지 개수$)+\alpha\}$는 원-패스 버퍼 크기의 하나의 하한값이다. 여기에서, α는 W_i의 부분집계 페이지가 모두 L-페이지이면 0이고, L-페이지가 아닌 것이 한 개 이상 있으면 1이다.

증명: 먼저, 집계 윈도우 W_i의 모든 L-페이지는 또 다른 집계 윈도우 W_i의 L-페이지임을 증명한다. 집계 윈도우 W_i의 L-페이지들을 $P_1, P_2, \ldots, P_L$이라 하고, $P_l(1 \le l \le L)$의 페이지 그룹화 영역을 R_l이라 하자. 그러면, L-페이지의 정의에 의해 모든 R_l에 대해 $W_i \subset R_l$이 성립한다. 그리고, 페이지 그룹화 영역들이 분리-포함 관계이므로, 각 $R_{l_1}(1 \le l_1 \le L)$과 $R_{l_2}(1 \le l_2 \le L)$ 사이에는 $R_{l_1} \subseteq R_{l_2}$ 혹은 $R_{l_2} \subseteq R_{l_1}$이 성립한다. 따라서, $\{1, \ldots, L\}$에 대한 한 순열 $\{a_l\}$이 있어서 $W_i \subseteq R_{a_1} \subseteq R_{a_2} \subseteq \cdots \subseteq R_{a_L}$의 관계가 성립한다. 이때, 집계 윈도우들은 그룹화 도메인 공간의 분할인데 R_{a_1}은 W_i보다 크기 때문에 R_{a_1}은 또 다른 집계 윈도우 W_j와 겹친다. 또한, 집계 윈도우들이 페이지 그룹화 영역들과 분리-포함 관계이므로 $W_j \subset R_{a_1}$이 성립한다. 그리고, $R_{a_1} \subseteq R_{a_2} \subseteq \cdots \subseteq R_{a_L}$이므로, W_i의 모든 L-페이지 P_l은 W_j의 L-페이지가 된다.

위 성질을 이용하면 다음과 같이 하나의 하한값을 구할 수 있다. 페이지 당 한번의 디스크 액세스를 보장하기 위해서는 한번 버퍼에 올라온 페이지는 더 이상 액세스되지 않을 때까지 버퍼에 있어야 한다. 따라서, W_i의 L-페이지는 W_i에 대한 부분집계 연산 도중 버퍼에 올라와 W_j에 대한 부분집계 연산에서 해당 페이지가 처리되기 전까지 버퍼에 남아 있어야 한다(W_i와 W_j의 순서가 바뀌어도 마찬가지임). 이를 위해서는 W_i에 대한 부분집계 연산을 처리할 때는 W_i의 L-페이지 수 만큼의 버퍼가 필요하다. 또한, 집계 윈도우에서 L-페이지들 이외의 페이지들은 해당 부분집계 연산에서만 액세스되는 페이지들로서 이러한 페이지들을 위해 버퍼 하나가 필요하여 $\alpha=1$이 된다. 결국, $\mathbb{W}$의 모든 집계 윈도우를 고려해야 하므로 $\max_{W_i, 1 \le i \le k}\{(W_i$의 L-페이지 개수$)+\alpha\}$만큼의 버퍼가 필요하다. □

DIP-Computation-Model은 버퍼 교체 정책과 페이지 액세스 순서에 대한 제한이 없다. 따라서, DIP-omputation-Model 하에서 버퍼 교체 정책과 페이지 액세스 순서를 다르게 채택함으로써 다양한 알고리즘의 개발이 가능하다. 정리 1은, 버퍼 교체 정책과 페이지 액세스 순서에 상관없이, 어떤 가능한 알고리즘도 원-패스 버퍼 크기가 정리 1의 결과보다 작을 수 없음을 의미한다.

4.4 페이지 액세스 순서

본 논문에서 제안하는 알고리즘은 페이지 액세스 순서를 제어함으로써 여러 번 액세스되는 페이지는 가능한 인접한 시점에 액세스되도록 한다. 페이지가 가능한 인접한 시점에 액세스되게 하는 이유는 동일한 페이지가 한꺼번에 액세스되고 더 이상 액세스되지 않아 버퍼에 더 이상 보관할 필요가 없게 되게 하기 위함이다. 제5.1절에서는 이렇게 페이지 액세스 순서를 제어하면 제안한 알고리즘이 DIP-Computation-Model 하에서 최적의 원-패스 버퍼 크기를 갖는 것을 증명한다.

DIP-Computation-Model을 사용한 집계 연산 처리에서, 여러 번 액세스되는 페이지가 인접해서 액세스되도록 하기 위해서는 이들 페이지와 겹치는 집계 윈도우들이 인접해서 순회되도록 집계 연산을 처리해야 한다. 이러한 집계 윈도우들의 순회 순서를 구하기 위해 영역들의 집합이 임의로 주어졌을 때 이 집합으로부터 유도

되는 공간 채움 곡선[13]을 정의한다. □

정의 5 다차원 공간 D에서 분리-포함 관계를 가지는 영역들의 집합 S=$\{S_1, S_2, \cdots, S_n\}$이 주어졌을 때, 다음의 조건을 만족하는 공간 채움 곡선을 **유도된 공간 채움 곡선**(*Induced Space Filling Curve: ISFC*)이라 정의한다.

조건: 다차원 공간 D상의 점 p의 ISFC 값을 ISFC(p)라 할 때, 임의의 S_i에서 $\min_{p\in s_i}\{\text{ISFC}(p)\} \le v \le \max_{p\in s_i}\{\text{ISFC}(p)\}$를 만족하는 모든 ISFC 값 v에 대하여, ISFC(p)=v를 만족하는 점 p는 S_i에 속한다.

여기에서 S를 **ISFC 근저**(*basis*)라 정의한다. 그리고, $\max_{p\in s_i}\{\text{ISFC}(p)\}$를 영역 S_i의 ISFC 값으로 정의하고, 간략히 ISFC(S_i)라 표기한다.

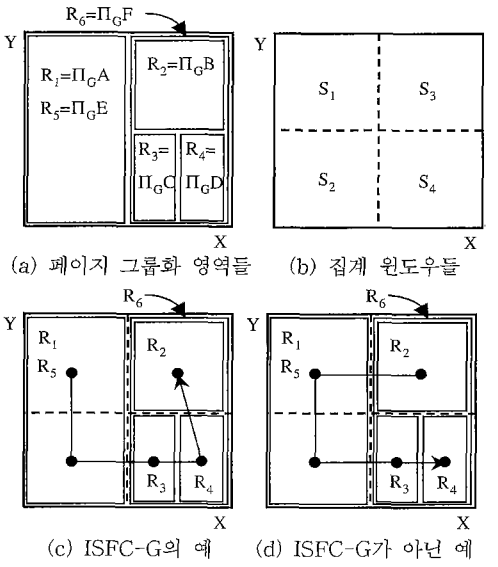
보조정리 2 분리-포함 관계를 가지는 영역들의 임의의 집합 S=$\{S_1, S_2, \cdots, S_n\}$이 주어졌을 때, S를 근저로 하는 ISFC가 적어도 하나 존재한다.

증명: 부록 C 참조 □

정의 5는 ISFC 순서가 한 영역에 있는 점들을 모두 순회한 후에 다른 영역에 있는 점들을 순회함을 의미한다. 영역들의 관점에서 보면, ISFC 순서는 큰 영역에 포함되는 작은 영역들을 모두 순회한 후에 큰 영역과 겹치지 않는 다른 영역을 순회함을 의미한다. 이를 집계 연산 처리에 적용하면, 한 페이지 그룹화 영역과 이에 포함된 여러 집계 윈도우들이 있을 때, 겹치는 집계 윈도우들이 인접해서 순회되도록 ISFC 순서를 사용할 수 있다.

이제, 집계 연산 처리에서 ISFC를 집계 윈도우의 순회 순서로 사용하는 방법과, 이에 따른 특성에 대해 논의한다. 분리-포함 분할 다차원 파일 구조의 페이지 그룹화 영역들의 집합을 R이라 하고 R과 분리-포함 관계를 갖는 집계 윈도우들의 집합을 W라 할 때, R∪W를 근저로 사용한 그룹화 도메인 공간에서의 ISFC를 *ISFC-G*라 부른다.

예 4: 그림 4는 ISFC-G의 예와 ISFC-G가 아닌 예를 보여준다. 그림 4(a)와 (b)는 각각 페이지 그룹화 영역들(R_i)과 집계 윈도우들(W_i)을 나타낸다. 그림 4(c)와 (d)는 그림 4(b)위에 그림 4(a)를 겹친 그림 위에 공간 채움 곡선을 나타낸 것이다. 그림 4(c)는 ISFC-G의 조건을 만족하는 반면에 그림 4(d)는 영역 R6의 점들을 순회하는 도중에 영역 R_1과 R_5의 점들을 순회하기 때문에 ISFC-G의 조건을 만족하지 않는다. R∪W를 근저로 한 ISFC-G를 집계 윈도우들의 순회 순서로 사용하면, 다음 보조정리 3에서와 같은 특성을 갖는다. 이 특성은 보조정리 4를 증명하는데 이용된다.



(a) 페이지 그룹화 영역들 (b) 집계 윈도우들

(c) ISFC-G의 예 (d) ISFC-G가 아닌 예

그림 4 ISFC-G의 예와 ISFC-G가 아닌 예(G={X,Y})

보조정리 3 DIP-Computation-Model을 이용하여 집계 연산을 처리한다고 하자. W=$\{W_1, W_2, \cdots, W_k\}$를 ISFC-G 순으로 나열된 (즉, $i<j$이면 ISFC-G(W_i) <ISFC-G(W_j)임) 집계 윈도우들의 집합이라 하면, 하나의 특정 L-페이지와 겹치는 집계 윈도우들은 연속적이다. 즉, $W_l, W_{l+1}, \ldots, W_h$ ($1 \le l < h \le k$)이다.

증명: 부록 D 참조. □

보조정리 3에 의하여, 집계 윈도우를 ISFC-G 순서로 순회하면서 부분집계 연산을 수행하면 특정 L-페이지와 겹치는 집계 윈도우들의 처리는 그 순서가 연속적이다. 다시 말해서, 여러 번 액세스되는 페이지는 연속적인 집계 윈도우들에서 액세스된다.

실제 구현에서의 구체적인 ISFC-G는 다차원 파일 구조의 고유의 특성에 따라 결정된다. 본 논문에서는 보조정리 1의 분할 규칙에 기반하여 구한 ISFC-G를 사용한다. R에 포함된 모든 페이지 그룹화 영역은 보조정리 1의 분할 규칙에 따라 생성된 영역이다. 또한, W에 포함된 집계 윈도우는 페이지 그룹화 영역들과 분리-포함 관계를 가져야 하기 때문에, 보조정리 1의 분할 규칙에 따라 생성될 수 있는 영역들 중에서 선택된다. 그런데, 보조정리 1의 분할 규칙을 만족하는 다차원 파일 구조들은 도메인 공간을 재귀적으로 반분하여 분할하기 때

문에, 그룹화 도메인 공간에 대해 재귀적으로 반분하여 분할되는 그룹화 영역들 만이 생성된다. 따라서, 다차원 파일 구조의 분할 축 선택 순서를 따르며 재귀적인 특성을 가지는 공간 채움 곡선을 ISFC-G로 사용할 수 있다. 예를 들어, 분할 축을 순환적(cyclic)으로 선택하는 순환 분할을 사용할 경우, ISFC-G는 Z-order[13]가 되며 본 논문의 실험에서는 이러한 Z-order를 사용한다.

5. 원-패스 집계 알고리즘

본 절에서는 DIP-Compuation-Model에 기반한 원-패스 집계 연산 알고리즘을 제안한다. 제5.1절에서는 버퍼 교체 정책으로 즉석-토스 정책을 사용했을 경우의 원-패스 버퍼 크기를 구한다.[2] 제5.2절에서는 원-패스 버퍼 크기를 사용하여 집계 연산을 처리하는 원-패스 집계 알고리즘을 제안한다.

5.1 즉석-토스 정책 사용 시 원-패스 버퍼의 크기

즉석-토스 정책을 사용하기 위해서는 주어진 시점에서 앞으로 액세스되지 않을 페이지를 알 수 있어야 한다. DIP-Computation-Model에서는 다음 보조정리 4를 이용하여 앞으로 액세스되지 않을 페이지를 찾을 수 있다.

보조정리 4 DIP-Computation-Model을 이용하여 집계 연산을 처리하고, $\mathbb{W}=\{W_1, W_2, \cdots, W_k\}$가 집계 윈도우들의 집합이라 하자. 그리고, W_i를 ISFC-G 순으로 순회하면서 부분집계 연산을 수행한다고 하자. 이 때, 집계 윈도우 W_{curr}에 대한 부분집계 연산 도중 액세스된 한 페이지 P_{curr}에 대해, ISFC-G($\Pi_G \widetilde{P}_{curr}$)≤ISFC-G($W_{curr}$)이 성립하면, P_{curr}는 더 이상 액세스되지 않는다.

증명: 부록 E 참조.

다음 정리 2에서는 즉석-토스 정책을 사용한다고 가정했을 경우의 원-패스 버퍼 크기를 구한다.

정리 2 DIP-Computation-Model을 따라 집계 연산을 처리한다고 하자. 그리고, $\mathbb{W}=\{W_1, W_2, \cdots, W_k\}$를 집계 윈도우들의 집합이라 하고, W_i를 ISFC-G 순서로 순회하면서 집계 연산을 처리한다고 하자. 이 때, 버퍼 교체 정책으로 즉석-토스 정책을 사용한다면, 이 경우 원-패스 버퍼 크기는 정리 1에서 구한 원-패스 버퍼 크기의 하한값 ($\max_{W_i, 1\le i\le k}\{(W_i$의 L-페이지 개수)+$\alpha\}$)과 동일하다. 즉, 이 값은 최소값이다.

증명: 우선, NP(t)를 페이지 액세스 시점 t에 "시점 t-1까지 액세스된 페이지들 중 앞으로 추가로 액세스될 페이지 개수+β"로 정의한다. 여기에서, β는 시점 t에 액세스할 페이지가 버퍼에 없는 페이지이면 1이고, 그렇지 않으면 0이 된다. 페이지 당 한번의 디스크 액세스를 보장하기 위해서는 한번 버퍼에 올라온 페이지는 더 이상 액세스되지 않을 때까지 버퍼에 있어야 한다. 그런데, 즉석-토스 정책은 앞으로 액세스되지 않는 페이지를 먼저 희생자로 선정하기 때문에, 한번 버퍼에 올라온 페이지를 더 이상 액세스하지 않을 때까지 버퍼에 유지하기 위해 필요한 버퍼 크기, 즉 원-패스 버퍼 크기는 t가 처음부터 마지막까지의 페이지 액세스에 걸쳐 변할 때 NP(t) 값들 중 최대값이 된다. 한편, 지금까지 액세스된 페이지 중 앞으로 추가로 액세스될 페이지는 현재 처리 중인 집계 윈도우의 L-페이지이다. 그 이유는 집계 윈도우들이 페이지 그룹화 영역들과 분리-포함 관계를 만족하고, 집계 윈도우들을 ISFC-G 순서로 순회하며 집계 연산을 처리하므로, 보조정리 3 — 하나의 특정 L-페이지는 연속적인 집계 윈도우들에서 액세스된다 — 에 의해서 현재 집계 윈도우의 L-페이지들 중 앞으로 추가로 액세스되는 페이지는 다음 집계 윈도우에서도 L-페이지가 되기 때문이다. 또한, "시점 t에 버퍼에 없는 페이지"는 해당 집계 윈도우에서만 액세스되는 페이지 혹은 L-페이지로서 처음 액세스되는 페이지이다. 따라서, NP(t)는 L-페이지들이 모두 버퍼에 올라온 상태에서 L-페이지가 아닌 페이지가 액세스되는 시점에서 최대가 되며, 이 때 β는 정리 1의 α와 동일한 값을 갖는다. 따라서, 최대값은 $\max_{W_i, 1\le i\le k}\{(W_i$의 L-페이지 개수)+$\alpha$)가 되어, 정리 1에서 구한 하한값과 동일하다.

5.2 원-패스 버퍼 크기를 사용한 집계 연산 알고리즘

그림 5는 원-패스 집계 연산 알고리즘 One_Pass_Aggregation을 나타낸다. 이 알고리즘은 그림 2의 알고리즘 General_Aggregation에 분리-포함 관계, ISFC-G, 원-패스 버퍼 크기의 개념을 추가하여 만든 것이다. 단계 1에서는 집계 윈도우들이 페이지 그룹화 영역들과 분리-포함 관계를 만족하도록, 그룹화 도메인 공간을 집계 윈도우들로 분할한다. 단계 2.1에서는 정리 2의 결과를 이용하여 원-패스 버퍼 크기 BUFSIZE를 구한다. 다차원 파일 구조들은 디렉토리에 페이지들에 대한 영역 정보를 가지고 있다. 따라서, 디렉토리만을 검색함으로써 원-패스 버퍼 크기를 구할 수 있다.[3] 단계 2.2는 BUFSIZE 만큼의 메모리를 할당받아 버퍼 관리자를 초기화한다. 단계

2) DIP-Computation-Model을 사용할 경우 모든 페이지 액세스 순서를 미리 알 수 있고, 따라서 즉석-토스보다 더 우수한 B_0 정책도 사용할 수 있다. 그러나, 모든 페이지의 액세스 순서를 계산하는 것은 복잡하며 본 논문의 초점이 아니므로, B_0 정책을 사용하는 알고리즘은 다른 논문에서 소개하기로 한다. 즉석-토스 정책에서는 앞으로 액세스되지 않을 페이지 만을 찾아내면 되므로 간단하게 사용할 수 있다. 이 두 알고리즘의 경우 원-패스 버퍼 크기는 동일하며, 이 크기 이하인 경우는 B_0 정책이 즉석-토스 정책보다 우수한 성능을 가진다.

3.1에서는 부분집계 연산에 사용될 영역 질의를 구성하며, 단계 3.2에서 이 질의를 수행한다. 영역 질의 수행 도중 한 페이지(P_{curr})에 대한 처리가 끝났을 때, 그 페이지 그룹화 영역이 현재 처리중인 집계 윈도우(W_{curr})와 ISFC-G 값이 같거나 작으면 더 이상 액세스되지 않는 페이지이므로 즉석-토스 정책에 의하여 그 페이지를 버퍼에서 쫓아낸다. 단계 3.3에서는 영역 질의로 검색된 레코드들을 대상으로 결과 테이블에서 집계 값을 구한다. 정리 2에 의하여 알고리즘 One_Pass_Aggregation의 원-패스 버퍼 크기는 DIP-Computation-Model을 따르는 알고리즘들의 원-패스 버퍼 크기의 최소값이다.

알고리즘 One_Pass_Aggregation
입력: (1) OLAP 데이타를 저장하고 있는 분리-포함 분할 다차원 파일 구조 *md-file*
(2) 그룹화 속성들의 집합 G
(3) 집계 속성 A
출력: 집계 연산 결과
알고리즘:
1 그룹화 도메인 공간을 집계 윈도우들로 분할한다. 이 때, 집계 윈도우들이 페이지 그룹화 영역들과 분리-포함 관계를 만족하도록 분할한다.
2 버퍼 관리자를 초기화한다.
2.1 전체 디렉토리를 스캔하여 원-패스 버퍼 크기 BUFSIZE를 구한다. BUFSIZE는 각 집계 윈도우에 대한 (L-페이지들의 개수+α) 값들 중 최대값이다.
2.2 BUFSIZE 크기의 메모리를 버퍼 관리자에 할당한다.
3 집계 윈도우들을 ISFC-G 순서로 순회하면서, 각 집계 윈도우 W_{curr}에 대해 다음을 수행한다.
3.1 영역 질의를 구성한다. 여기에서 질의 영역은 그룹화 속성들에 대해서는 집계 윈도우를 대상으로 하고, 다른 속성에 대해서는 전체 도메인을 대상으로 한다.
3.2 영역 질의로 *md-file*을 검색한다. 영역 질의 수행 중 한 페이지(P_{curr})에 대한 처리가 끝난 후, ISFC-G($\prod_G \tilde{P}_{curr} \leq$ ISFC-G(W_{curr})이 성립하면 P_{curr}를 버퍼에서 쫓아낸다.
3.3 검색된 각 레코드에 대해 그룹화 속성들(G)의 값을 키로 하여 결과 테이블의 해당 엔토리를 찾아 집계 속성(A)의 값을 엔트리의 결과 값에 집계한다.

그림 5 분리-포함 분할 다차원 파일 구조를 이용한 집계 연산 알고리즘 One_Pass_Aggregation

3) 실제로는 디렉토리 검색없이 알고리즘을 구현한다. 단계 2에서는 버퍼를 전혀 할당하지 않고, 대신에 단계 3.2에서 버퍼에 없는 새로운 페이지가 액세스되는데 가용한 버퍼가 없는 경우 동적으로 버퍼 하나를 할당한다. 이와 같은 구현 방법을 사용해도 원-패스 버퍼 크기는 동일하며 메모리를 동적으로 관리함으로써 오히려 성능을 개선시킬 수 있다. 그러나, 본 논문에서는 이해를 돕기 위해 원-패스 버퍼 크기를 구하는 과정을 포함하여 알고리즘을 기술하였다.

6. 성능 평가

본 절에서는 제안한 One_Pass_Aggregation의 성능 평가 결과를 설명한다. 실험의 목표는 다음과 같다. 첫째, 제안한 알고리즘의 원-패스 버퍼 크기가 정리 2의 결과와 동일함을 실험을 통해 확인하는 것이다. 둘째, 즉석-토스 버퍼 교체 정책을 사용하는 알고리즘 One_Pass_Aggregation이 일반적으로 널리 사용되는 버퍼 교체 정책인 CLOCK 정책(LRU의 근사법)을 사용했을 때 보다 우수함을 보이는 것이다. 제 5.1절에서는 실험 데이타와 실험 환경을 소개하고, 제 5.2절에서는 실험 결과를 설명한다.

6.1 실험 데이타 및 실험 환경

실험 데이타는 여섯 개의 속성으로 구성된 레코드이다. 여섯 개의 속성 중 다섯 개는 차원을 나타내는 구성 속성이고, 나머지 한 개는 측정값을 나타내는 속성이다. 모든 속성은 도메인이 $[-2^{31}, 2^{31}-1]$인 4바이트 정수형 타입이다. 구성 속성 값의 분포에 따라 세 가지 종류의 데이타를 사용하여 실험을 수행한다. 각 분포는 5백만개의 레코드(120MB)를 생성한다.

UNIFORM-DATA 도메인 공간에서 균일 분포를 갖는 데이타이다.

CLUSTERED-DATA 5,000개의 레코드를 생성하는 다변량 정규 분포(multivariate normal distribution) 1,000개를 합쳐서 사용한 데이타로, 다변량 정규 분포는 i번째 속성이 정규 분포 $N(\mu_i, \sigma^2)$을 따른다. 여기서, 평균 μ_i는 랜덤하게 선정하였고, 여러 개의 클러스터로 구성된 분포를 시뮬레이션하기 위하여 표준 편차 σ는 2^{20}으로 고정하였다. 두 속성 간의 상관 계수는 0이 되도록 결정하였다.

CORRELATED-DATA CLUSTERED-DATA에서 두 속성 간의 상관 계수가 0.5가 되도록 변경한 데이타이다.

UNIFORM-DATA는 도메인 공간이 그리드 셀 모양으로 균일하게 분할된 경우의 성능 분석을 위하여 사용한다. CLUSTERED-DATA는 데이타들이 클러스터들을 형성하는 OLAP 데이타의 특성을 반영한 것이다. CORRELATED-DATA는 차원들 간에 상관 관계가 있는 데이타의 성능 분석을 위한 것이다.

본 실험에서는 위의 데이타를 대상으로 알고리즘 One_Pass_Aggregation을 사용하여 집계 연산을 수행한다. 사용된 집계 연산은 그룹화 속성으로 세 개를 사용한다. 데이타를 저장하는 분리-포함 분할 다차원 파일 구조로는 보조정리 1의 분할 규칙들을 만족하는 MLGF

[7,8]를 사용하고, 페이지의 크기는 4KB로 한다. 먼저, 집계 윈도우들은 80KB 크기의 결과 테이블에 부분집계 연산 결과가 들어가도록 결정하여 사용한 후, 결과 테이블의 크기를 변화시켜가며 주기억장치 요구량(=결과 테이블 크기+원-패스 버퍼 크기)을 분석한다. 버퍼 크기는 크기의 영향을 살펴 볼 수 있도록 버퍼 개수를 5 페이지부터 5 페이지 단위로 증가시키며 실험한다.

버퍼 크기가 원-패스 버퍼 크기보다 작을 때는 그림 5의 알고리즘 One_Pass_Aggregation의 작동이 불가능하므로, 이 경우에는 즉석-토스 정책과 CLOCK을 병행(혼동이 없는 한 간단히 즉석-토스라 부른다)하여 사용한다. 즉, 단계 3.2에서 버퍼에서 쫓아낼 페이지가 없는 경우 CLOCK 알고리즘을 사용하여 희생자를 선정한다. 성능을 평가하기 위한 척도로는 집계 연산 처리 도중 발생하는 디스크 액세스 횟수를 사용한다. 그 이유는 집계 연산의 성능이 디스크 입출력에 가장 큰 영향을 받기 때문이다[6].

또한, 원-패스 버퍼 크기의 정확성을 보이기 위해서 원-패스 버퍼 크기에서 디스크 액세스 횟수가 파일 전체의 페이지 개수와 동일한지를 확인하기 위함이다.

6.2 실험 결과

그림 6은 UNIFORM-DATA에 대한 실험 결과이다. 그림에서 가로 축은 버퍼 크기를 나타내고, 세로 축은 페이지 당 평균 디스크 액세스 횟수, 즉 디스크 액세스 횟수를 전체 페이지 개수로 정규화한 값을 나타낸다. 이 값이 1.0일 때가 이론적으로 최적의 성능임을 뜻하며, 1.0 값을 가지는 최소 버퍼 크기가 바로 원-패스 버퍼 크기이다. 그림 6의 경우 즉석-토스 정책의 원-패스 버퍼 크기는 20으로 나타났는데, 이 값은 정리 2의 $\max_{W_i, 1 \le i \le k}\{(W_i$의 L-페이지 개수$)+\alpha\}$을 이용하여 구한 값(20)과 동일함을 확인하였다.

그림 6을 보면, 모든 구간에서 즉석-토스 정책을 사용한 경우가 CLOCK 정책을 사용한 경우보다 디스크 액세스 횟수가 적음을 알 수 있다. 즉, 즉석-토스 정책을 사용하면 CLOCK 정책보다 훨씬 더 좋은 성능을 얻을 수 있다. 이는 본 논문에서 언급한 페이지 액세스 순서 정보를 미리 알아내고 이를 이용하여 희생자를 선택하는 버퍼 교체 정책이 매우 우수함을 보이고 있는 것이다. 즉석-토스 정책은 버퍼 크기가 증가함에 따라 디스크 액세스 횟수가 줄어들다가, 원-패스 버퍼 크기인 20에 이르러서는 최적의 성능을 보이고 있다. CLOCK 정책의 경우 버퍼 크기가 증가하더라도 디스크 액세스 횟수는 거의 변하지 않고 있다. 이는 부분집계 페이지 개수들이 실험에 사용된 버퍼 크기에 비해 커서 버퍼가 효과를 보지 못하기 때문이다.

그림 7은 CLUSTERED-DATA에 대한 실험 결과이다. 그림 6과 마찬가지로 모든 구간에서 즉석-토스 정책을 사용한 경우가 CLOCK 정책을 사용한 경우보다 성능이 훨씬 우수함을 알 수 있다. 그림 7을 보면, CLOCK 정책은 UNIFORM-DATA와 달리 버퍼 크기가 커질수록 성능이 개선됨을 알 수 있다. 이는 CLUSTERED-DATA의 경우 UNIFORM-DATA와 달리 집계 윈도우에 따라 부분집계 페이지들의 개수가 다른데, 부분집계 페이지들의 개수가 적은 부분집계 연산들에서는 버퍼 효과를 보기 때문이다. 즉석-토스 정책을 사용한 경우 그림 6은 버퍼 크기가 20에서 갑자기 성능이 좋아지는 반면 그림 7에서는 서서히 좋아지고 있음을 알 수 있다. 이는 그림 6은 균일 분포로서 각 집계 윈도우에 대한 L-페이지들의 개수가 비슷하며, 버퍼 크기가 그 개수(=20)에 이르면 원-패스가 되어 성능이 갑자기 좋아지기 때문이다. 반면에, 그림 7은 불균일 분포로서 집계 윈도우에 따라 L-페이지들의 개수가 다르게 되어, 버퍼 크기가 증가함에 따라 디스크 액세스 횟수가 천천히 감소한다. 즉석-토스 정책의 원-패스 버퍼 크기는 그림 7에서 40으로 나타났으며, 이는 정리 2의 결과를 이용해서 구한 값(40)과 동일하다.

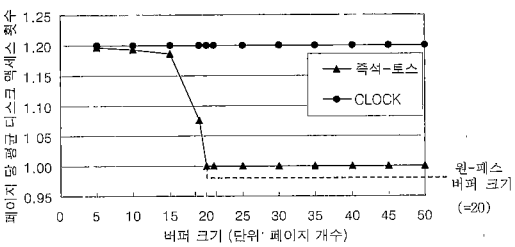


그림 6 UNIFORM-DATA에 대한 정규화된 디스크 액세스 횟수

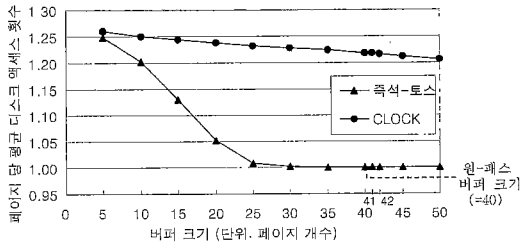


그림 7 CLUSTERED-DATA에 대한 정규화된 디스크 액세스 횟수

CORRELATED-DATA에 대한 실험 결과는 CLUSTERED-DATA와 매우 비슷하게 나타났다. 이는 MLGF가 불균일 데이타를 잘 처리하기 때문이다. 지면 관계상 CORRELATED-DATA의 자세한 실험 결과는 생략한다.

실험 결과, 제안한 집계 알고리즘이 CLUSTERED-DATA와 CORRELATED-DATA에서와 같이 편중된 분포의 데이타에 대해서도 잘 동작함을 알 수 있다. 그 이유는 두 가지로 요약할 수 있다. 첫째, 편중된 분포의 데이타를 잘 저장하는 다차원 파일 구조를 사용함으로써 생성된 데이타베이스 크기가 레코드 개수에 선형적으로 비례하기[7,8] 때문이다. 둘째, 제안한 집계 알고리즘을 사용함으로써 이 데이타베이스를 단지 한번만 읽어서 집계 결과를 구할 수 있기 때문이다.

그림 8은 CLUSTERED-DATA에서 결과 테이블의 크기에 따른 주기억장치 요구량을 나타낸다. 그림에서 가로 축과 세로 축은 각각 결과 테이블 크기와 주기억장치 요구량을 데이타베이스 크기로 정규화한 값을 나타낸다. 그림 8을 보면, 결과 테이블 크기가 커지면 원-패스 버퍼 크기는 감소함을 알 수 있다. 이것은, 결과 테이블 크기가 커지면 집계 윈도우의 크기가 커지게 되어 L-페이지의 개수가 줄어들기 때문이다. 그리고, 결과 테이블 크기가 0.1% 이하의 구간에서 전체 데이타베이스의 0.2%의 주기억장치 크기만으로 최적의 성능을 얻을 수 있음을 알 수 있다. 따라서, 실제적인 환경에서 이 구간에 속한 값을 결과 테이블 크기로 사용할 수 있다.

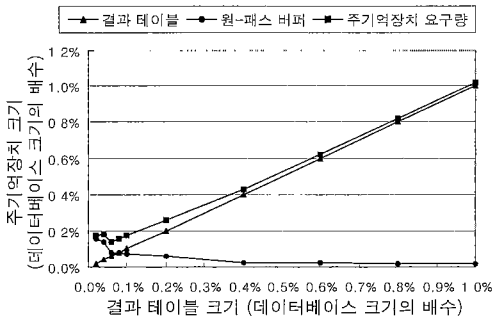


그림 8 CLUSTERED-DATA에 대한 주기억장치 요구량

7. 결 론

집계 알고리즘은 OLAP 시스템의 성능에 큰 영향을 미치는 중요한 요소이다. 본 논문에서는 편중된 분포의 데이타를 잘 처리할 수 있는 다차원 파일 구조를 사용하는 동적인 집계 알고리즘을 제안하였다. 먼저, 분리-포함 분할이라는 새로운 개념을 제시하였으며, 이 개념을 기반으로 하는 새로운 집계 연산 처리 모델인 DIP-Computation-Model을 제안하였다. 그리고, 이 모델에 기반하여 원-패스 버퍼 크기를 사용하는 원-패스 집계 알고리즘인 One_Pass_Aggregation을 개발하였다. 여기에서, 원-패스 버퍼 크기란 페이지 당 한번의 디스크 액세스를 보장하기 위해 필요한 최소 버퍼 크기이다. 그리고, 알고리즘 One_Pass_Aggregation을 분리-포함 분할 조건을 만족하는 MLGF를 이용하여 구현하여 본 알고리즘이 균일한 분포의 데이타 뿐만 아니라 편중된 분포의 데이타들도 잘 처리함을 보였다.

본 논문에서는 DIP-Computation-Model에서 액세스되는 페이지들의 순서를 제어함으로써 최소의 원-패스 버퍼 크기로 집계 연산을 처리할 수 있는 정형적인 프레임워크(formal framework)를 제안하였다. 먼저, 정리 1을 통해 DIP-Computation-Model에서 원-패스 버퍼 크기의 하나의 하한값을 정형적으로 유도하였다. 그리고, 보조 정리 3을 통해 ISFC를 사용하여 페이지들의 액세스 순서를 제어하면 여러 번 액세스되는 페이지들이 인접해서 액세스될 수 있음을 증명하였다. 또한, 보조 정리 4를 통해 페이지들의 액세스 순서를 이용한 즉석-토스 버퍼 교체 정책을 사용할 수 있음을 증명하였다. 그리고, 정리 2에서 즉석-토스 정책을 사용하면 원-패스 버퍼 크기가 최소가 됨을 증명하였다. 마지막으로, 이러한 원-패스 버퍼 크기를 사용하는 집계 알고리즘을 제안하였다.

제안하는 One_Pass_Aggregation의 성능 평가를 위해 다양한 분포의 데이타에 대해 많은 실험을 수행하였다. 실험 결과, 제안하는 알고리즘의 원-패스 버퍼 크기는 정리 2의 결과와 동일하였으며, 일반적으로 널리 사용되는 버퍼 교체 정책인 CLOCK 정책을 사용했을 때와 비교해서 즉석-토스 정책을 사용했을 때 성능이 항상 우수함을 보였다.

제안한 방법은 페이지 액세스 순서를 미리 알아 낼 수 있어, 페이지 액세스 순서를 이용하는 버퍼 교체 정책을 사용함으로써 최적의 원-패스 버퍼 크기를 달성한다. 특히, 여러 집계 질의가 동시에 요청되는 다사용자 환경에서는, 질의 최적화기가 제한된 페이지 버퍼들을 각 질의들에 대해 최적으로 할당해야 성능을 크게 향상시키므로, 제안하는 이론의 중요성은 매우 크다. 이러한 점에서 제안하는 원-패스 알고리즘은 대량의 데이타에 대해 다수의 집계 연산이 처리되는 환경에서 효율적으

로 동작할 것으로 예상한다.

제안한 집계 처리 모델은 MOLAP에서의 집계 연산 처리 분야, 예를 들어 데이타 큐브 계산에서 유용하게 사용될 우수한 정형적인 기초를 제공할 것으로 예상되며, 이 부분에 대한 자세한 내용은 향후 연구로 남겨 둔다.

참 고 문 헌

[1] Chaudhuri, S. and Dayal, U., "An Overview of Data Warehousing and OLAP Technology," *ACM SIGMOD Record*, Vol. 26, No. 1, pp. 65-74, Mar. 1997.

[2] Agarwal, S., Agrawal, R., Deshpande, P.M. et al., "On the Computation of Multidimensional Aggregations," In *Proc. Int'l Conf. on Very Large Data Bases*, pp. 506-521, Mumbai (Bombay), India, Sept. 1996.

[3] Kotidis, Y. and Roussopoulos, N., "An Alternative Storage Organization for ROLAP Aggregate Views Based on Cubetrees," In *Proc. Int'l Conf. on Management of Data*, pp. 249-258, ACM SIGMOD, Seattle, Washington, June 1998.

[4] Li, J., Rotem, D., and Srivastava, J., "Aggregation Algorithms for Very Large Compressed Data Warehouses," In *Proc. Int'l Conf. on Very Large Data Bases*, pp. 651-662, Edinburgh, Scotland, UK, Sept. 1999.

[5] Zhao, Y., Deshpande, P.M., and Naughton, J.F., "An Array-Based Algorithm for Simultaneous Multidimensional Aggregates," In *Proc. Int'l Conf. on Management of Data*, pp. 159-170, ACM SIGMOD, Tucson, Arizona, June 1997.

[6] Graefe, G., "Query Evaluation Techniques for Large Databases," *ACM Computing Surveys*, Vol. 25, No. 2, pp. 73-170, June 1993.

[7] Whang, K. and Krishnamurthy, R., Multilevel Grid Files, IBM Research Report RC 11516(51719), 1985.

[8] Whang, K. et al., "Dynamic Maintenance of Data Distribution for Selectivity Estimation," *The VLDB Journal*, Vol. 3, No. 1, pp. 29-51, Jan. 1994.

[9] Effelsberg, W. and Haerder, T., "Principles of Database Buffer Management," *ACM Trans. on Database Systems*, Vol. 9, No. 4, pp. 560-595, Dec. 1984.

[10] Coffman, E.G. Jr. and Denning, P.J., *Operating Systems Theory*, Prentice-Hall, 1973.

[11] O'Neil, E.J., O'Neil, P.E., and Weikum, G., "The LRU-K Page Replacement Algorithm for Database Disk Buffering," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, Washington, DC, May 1993.

[12] Korth, H.F. and Silberschatz, A., *Database System Concepts*, McGraw-Hill, New York, Second Ed., 1991.

[13] Gaede, V. and Gunther, O., "Multidimensional Access Methods," *ACM Computing Surveys*, Vol. 30, No. 2, pp. 170-231, June 1998.

부록 A: 집계 윈도우 선택 방법

집계 윈도우를 선택하기 위해서는 각 부분집계 연산들의 결과 크기가 결과 테이블의 크기와 비슷해지도록 그룹화 도메인 공간을 집계 윈도우로 나누어야 한다. 따라서, 집계 윈도우로 선택될 영역에 포함되는 서로 다른 그룹화 속성의 값의 개수에 대한 정보가 필요하다. 이를 위하여 그룹화 속성들을 대상으로 한 다차원 히스토그램을 사용한다. 히스토그램은 집계 연산 이외의 질의 처리에도 유용하게 사용되므로, 본 논문에서는 OLAP 데이타베이스 구축시 히스토그램도 함께 계산하여 유지한다고 가정한다. 그런데, 조합 가능한 모든 그룹화 속성들에 대해 히스토그램을 유지하는 것은 큰 오버헤드가 될 수 있다. 따라서, 본 논문에서는 그룹화 영역에 속한 그룹화 속성의 값이 서로 다른 레코드 개수의 추정치로서 간단히 그룹화 영역에 속한 레코드의 개수를 사용함으로써[4] 전체 구성 속성을 대상으로 한 하나의 히스토그램만을 유지하는 방법을 사용한다.

그림 9는 보조정리 1을 만족하는 다차원 파일 구조들에서 히스토그램을 사용하여 집계 윈도우들을 선택하는 알고리즘이다. 입력은 구성 속성들을 대상으로 한 히스토그램(H), 다차원 파일 구조에서 분할 축을 선택할 때 사용한 순서 중에서 그룹화 속성들 만에 대한 순서 ($SplittingOrder=A_1A_2A_3\cdots$), 결과 테이블 크기($R_{max}$, 단위는 레코드 개수임), 그리고 그룹화 영역(W)이다. W를 그룹화 도메인 공간으로 설정하여 알고리즘 SelectAggregationWindows를 수행하면 그룹화 도메인 공간을 분할하는 집계 윈도우들을 구할 수 있다. 단계 1에서는 W에 속한 그룹화 속성 값이 서로 다른 레코드 개수의 추정치($no_records(W)$라 하자)로서, 그룹화 속성들에 대해서는 W를 대상으로 하고 다른 속성에 대해서는 전체 도메인을 대상으로 하는 영역에 속한 레코드의 개

4) 이 추정치는 최대치로서 실제 값과의 오차는 레코드 당 그룹화 속성의 값이 같은 레코드 개수, 즉 중복도가 높을수록 커지게 된다. 좀 더 정확한 계산을 위해서는 통계 정보로서 중복도를 구하여 파라메터로 사용하는 방법을 사용할 수 있을 것이나 본 논문의 범위를 벗어나므로 여기서는 다루지 않는다.

수를 히스토그램을 이용하여 구한다. 단계 2에서는 no_records(W)가 R_{max} 이하이면 W를 집계 윈도우로 선택하고 종료한다. no_records(W)가 R_{max}보다 큰 경우를 처리하기 위하여 단계 3에서는 W를 A_1을 분할 축으로 하여 W_1과 W_2로 나눈다. 그리고, 단계 4에서는 W를 W_1으로, *SplittingOrder*를 A_2A_3…로 설정하고 SelectAggregationWindows를 재귀적으로 호출한다. 그리고, 단계 5는 W를 W_2로, *SplittingOrder*를 A_2A_3…로 설정하고 SelectAggregationWindows를 재귀적으로 호출한다.

알고리즘 SelectAggregationWindows
입력: (1) 구성 속성들을 대상으로 한 히스토그램 H
(2) 사용된 분할 축 선택 순서 중 그룹화 속성들만에 대한 순서 $SplittingOrder=A_1A_2A_3$…
(3) 결과 테이블의 크기 R_{max}
(4) 그룹화 영역 W
출력: 집계 윈도우들
알고리즘:
1 히스토그램(H)을 이용하여 W에 속한 그룹화 속성 값이 서로 다른 레코드 개수의 추정치 no_records(W)를 구한다.
2 IF no_records(W)≤ R_{max} THEN W는 집계 윈도우이므로 리턴한다.
3 W를 축 A_1을 분할 축으로 사용하여 W_1과 W_2로 분할한다.
4 분할 대상 그룹화 영역 W를 W_1으로, *SplittingOrder*를 A_2A_3…로 설정하고 SelectAggregationWindows를 호출한다.
5 분할 대상 그룹화 영역 W를 W_2로, *SplittingOrder*를 A_2A_3…로 설정하고 SelectAggregationWindows를 호출한다.

그림 9 집계 윈도우들을 선택하는 알고리즘 SelectAggregationWindows

부록 B: 보조정리 1의 증명

먼저 도메인 공간을 구성하는 속성들을 A_1, …, A_n이라 하고, 분할로 생긴 도메인 공간의 영역들의 집합을 $Q=\{Q_1, \cdots, Q_k\}$라 하자. 그러면, 임의의 두 영역 Q_i와 Q_j에 대하여 다음이 성립한다.

(1) 영역 기준 분할 전략은 영역의 크기를 반분하는 위치를 분할 경계로 선택한다. 이 경우, 영역들을 임의의 속성 하나(A_l이라 하자)에 대하여 프로젝션했을 때 생기는 영역들은 속성 A_l의 도메인을 재귀적으로 반분함으로써 생길 수 있는 영역들로 국한된다. 따라서, 두 영역 Q_i와 Q_j를 속성 A_l에 대하여 프로젝션 한 $\Pi_{A_l}Q_i$와 $\Pi_{A_l}Q_j$는 분리-포함 관계를 만족한다. 또한, $NS_{A_l}(Q_i)$를 Q_i가 만들어지기까지의 속성 A_l에 대한 분할 횟수라 하면, $\Pi_{A_l}Q_i$와 $\Pi_{A_l}Q_j$가 겹치는 경우 $NS_{A_l}(Q_i)$와 $NS_{A_l}(Q_j)$ 중 작은 값을 갖는 영역이 다른 영역을 포함한다.

(2) $SplitAxes(Q_i)\subseteq SplitAxes(Q_j)$ 또는 $SplitAxes(Q_i)\supseteq SplitAxes(Q_j)$를 만족하므로, $\forall_{A_l}(NS_{A_l}(Q_i)\leq NS_{A_l}(Q_j))$이거나 혹은 $\forall_{A_l}(NS_{A_l}(Q_i)\geq NS_{A_l}(Q_j))$이 성립한다.

이제, 임의의 $G\subseteq\{A_1, \cdots, A_n\}$에 대하여 $\Pi_G Q_i$와 $\Pi_G Q_j$가 겹친다고 하자. 그러면, G에 속하는 모든 A_l에 대해 $\Pi_{A_l}Q_i$와 $\Pi_{A_l}Q_j$는 서로 겹침을 의미한다. 그런데, 위 (1)과 (2)에 의해 $\forall_{A_l\in G}(\Pi_{A_l}Q_i\supseteq\Pi_{A_l}Q_j)$ 혹은 $\forall_{A_l\in G}(\Pi_{A_l}Q_i\subseteq\Pi_{A_l}Q_j)$가 성립한다. 다시 말해서, $\Pi_G Q_i\supseteq\Pi_G Q_j$ 혹은 $\Pi_G Q_i\subseteq\Pi_G Q_j$가 성립하고, 이는 한 영역이 다른 영역을 포함함을 의미한다. 따라서, Q는 분리-포함 분할이다.

부록 C: 보조정리 2의 증명

S에 속한 영역들은 서로 분리-포함 관계를 만족하므로, 다음과 같이 S를 부분집합 U_1, U_2, …, U_n으로 분할할 수 있다. 먼저, U_1을 $\{S_i\in S \mid \forall_{S_j\in S}(S_i\not\subset S_j)\}$라 하자. 즉, U_1의 영역 $S_{1,i}$는 S에 속한 영역들 중에서 자신을 진포함하는 영역 $S_j\in S$가 하나도 없는 영역이다. 다음으로, U_2를 $\{S_i\in S-U_1 \mid \forall_{S_j\in S-U_1}(S_i\not\subset S_j)\}$라 하자. 즉, U_2의 영역 $S_{2,i}$는 $(S-U_1)$에 속한 영역들 중에서 자신을 진포함하는 영역 $S_j\in(S-U_1)$가 하나도 없는 영역이다. 그러면, U_2에 속한 각 영역은 U_1에 속한 영역에 진포함된다. 그 이유는, 만일 U_2에 속한 영역 $S_{2,i}$가 U_1에 속한 어떤 영역에도 진포함되지 않는다면, $S_{2,i}$는 U_2가 아닌 U_1에 속하기 때문이다. 이와 같은 방법으로, $U_m(1<m\leq n)$을 $\{S_i\in S-(U_1\cup\ldots\cup U_{m-1}) \mid \forall_{S_j\in S-(U_1\cup\ldots\cup U_{m-1})}(S_i\not\subset S_j)\}$라 하자. 즉, U_m의 영역 ($S_{m,i}$는 $S-(U_1\cup\ldots\cup U_{m-1})$)에 속한 영역들 중에서 자신을 진포함하는 영역 $S_j\in(S-(U_1\cup\ldots\cup U_{m-1}))$가 하나도 없는 영역이다. 그러면, U_2의 경우와 마찬가지로, U_m에 속한 각 영역 $S_{m,i}$는 U_{m-1}에 속한 하나의 영역에 진포함된다.

이제 ISFC를 하나 생성해 보임으로써 S를 ISFC 근저로하는 ISFC가 존재함을 증명한다. 먼저, $S_{n,i}\in U_n$에 대해서 $S_{n,i}$의 좌하점에서 시작해서 우상점에서 끝나도록 $S_{n,i}$에 포함된 점들을 순회하는 임의의 순서를 정해, 이 순서를 $S_{n,i}$에 대한 공간 채움 곡선으로 하고, 이를 $SFC_{n,i}$라 한다. 다음으로, $S_{n-1,i}\in U_{n-1}$에 대해서도 좌하점에서 시작해서 우상점에서 끝나는 $SFC_{n-1,i}$를 할당하되, $S_{n-1,i}$에 포함된 $S_{n,k}(\in U_n)$와 겹치는 부분에서는 $SFC_{n,k}$를 사용한다. 마찬가지로, $S_{m,i}\in U_m(1\leq m<n)$에 대해서도 좌하점에서 시작해서 우상점에서 끝나는 $SFC_{m,i}$를 할당하되, $S_{m,i}$에 포함된 $S_{m+1,k}(\in U_{m+1})$와 겹치는 부분에서는 $SFC_{m+1,k}$를 사용한다. 이를 $S_{1,i}\in U_1$에

대해서까지 반복한 후, 영역 $S_{1,i}$에 대해 만들어진 $SFC_{1,i}$들을 모두 연결하여 SFC를 만들면, SFC는 S를 근저로 하는 ISFC의 조건을 자연히 만족한다. 따라서, S를 근저로 하는 ISFC가 존재한다.

부록 D: 보조정리 3의 증명

L-페이지 P의 페이지 그룹화 영역을 R이라 하고, R과 겹치는 집계 윈도우들을 $W' = \{W_{i_1}, W_{i_2}, \cdots, W_{i_m}\}(1 \leq m \leq k)$이라고 하자. 그러면, L-페이지의 정의에 의해 $W_{i_j}(1 \leq j \leq m)$는 $W_{i_j} \subset R$을 만족한다. 그리고, 이로부터 $\bigcup_{j=1}^{m} W_{i_j} \subset R$이 성립한다. 한편, 집계 윈도우들은 그룹화 도메인 공간의 분할이므로, R은 R과 겹치는 집계 윈도우들의 합집합에 포함되어야 한다. 즉, $R \subseteq \bigcup_{j=1}^{m} W_{i_j}$이 성립한다. 결국, 두 식 $\bigcup_{j=1}^{m} W_{i_j} \subset R$과 $R \subseteq \bigcup_{j=1}^{m} W_{i_j}$이 성립하므로, $\bigcup_{j=1}^{m} W_{i_j} = R$이 성립한다. 한편, ISFC-G는 페이지 그룹화 영역들과 집계 윈도우들의 합집합을 근저로 하는 ISFC이고, ISFC는 큰 영역(R)에 속하는 작은 영역들($W_{i_j}(1 \leq j \leq m)$)을 연속적으로 순회하므로 R에 속하는 집계 윈도우 W_{i_j}들은 ISFC-G 값이 연속적이다.

부록 E: 보조정리 4의 증명

S와 p를 각각 다차원 공간 $\boldsymbol{D}$상의 영역과 점이라 할 때, $\min_{p \in S}\{\text{ISFC-G}(p)\}$와 $\max_{p \in S}\{\text{ISFC-G}(p)\}$를 간단히 $\text{ISFC-G}_{\min}(S)$와 $\text{ISFC-G}_{\max}(S)$로 표기한다. 또한, $\Pi_G \tilde{P}_i$를 R_i로 $\Pi_G \tilde{P}_{curr}$를 R_{curr}로 각각 표시한다. 페이지 P_i가 집계 윈도우 W_j에 대한 부분집계 연산에서 액세스되기 위해서는 P_i와 W_j가 겹쳐야 한다. P_i와 W_j가 겹치기 위한 조건은 다음 식(2)와 같다. 그 이유는 ISFC의 정의에 의해 P_i와 W_j가 겹치기 위해서는 [시작점,끝점]이 $[\text{ISFC-G}_{\min}(R_i), \text{ISFC-G}_{\max}(R_i)]$와 $[\text{ISFC-G}_{\min}(W_j), \text{ISFC-G}_{\max}(W_j)]$인 두 선분이 겹쳐야 하기 때문이다.

$$(\text{ISFC-G}_{\min}(R_i) \leq \text{ISFC-G}_{\max}(W_j)) \wedge (\text{ISFC-G}_{\max}(R_i) \geq \text{ISFC-G}_{\min}(W_j)) \quad (2)$$

그런데, 현재 처리 중인 집계 윈도우 W_{curr} 다음에 순회되는 집계 윈도우를 W_{next}라 하면 $\text{ISFC-G}_{\max}(W_{curr}) < \text{ISFC-G}_{\min}(W_{next})$가 성립한다. 그리고, 보조정리 4의 충분 조건(즉, $\text{ISFC-G}(R_{curr}) \leq \text{ISFC-G}(W_{curr})$)에 의해 $\text{ISFC-G}_{\max}(R_{curr}) \leq \text{ISFC-G}_{\max}(W_{curr})$가 성립한다. 따라서, 부등식 $\text{ISFC-G}_{\max}(R_{curr}) < \text{ISFC-G}_{\min}(W_{next})$가 성립한다. 즉, R_{curr}는 W_{next}와 겹칠 수 없다. 따라서, P_{curr}는 더 이상 액세스되지 않는다.

이 영 구
1992년 2월 한국과학기술원 과학기술대 전산학과 학사. 1994년 2월 한국과학기술원 전산학과 석사. 1994년 3월 ~ 현재 한국과학기술원 전자전산학과 전산학 전공 박사과정. 관심분야는 OLAP, 데이타 웨어하우스, Access Method.

문 양 세
정보과학회논문지 : 데이타베이스 제 28 권 제 1 호 참조

황 규 영
정보과학회논문지 : 데이타베이스 제 28 권 제 1 호 참조