데이타 스트림에서 동적 데이타 큐브
(Dynamic Data Cubes Over Data Streams)

서 대 홍 † 양 우 석 †† 이 원 석 †††
(Dae Hong Seo) (Woo Sock Yang) (Won Suk Lee)

요 약 OLAP의 다차원 데이타 모델인 데이타 큐브는 많은 다차원 데이타 분석에 성공적으로 적용되었으며, 데이타 스트림 분석에도 적용하려는 많은 연구가 진행되고 있다. 데이타 스트림은 실시간에 지속적으로 방대하게 생성되며, 데이타의 분포적 특성이 빠르게 변한다는 특징을 가지며, 제한된 메모리 및 처리능력 때문에 한번만 검사하여 처리하는 것을 기본으로 한다. 때문에 데이타 스트림을 메모리에 모두 저장하는 것은 불가능하다. 또한 사용자는 모든 속성 값에 대하여 관심을 두기 보다는 일정 지지율 이상을 가진 속성 값에 더욱 관심을 가지게 된다. 본 논문에서는 이러한 데이타 스트림 환경에서 데이타 큐브를 효과적으로 적용하기 위한 동적 데이타 큐브를 제안한다. 동적 데이타 큐브는 속성 값의 지지율에 따라 사용자 관심 영역을 지정하고, 속성 값을 동적으로 그룹화하여 관리한다. 이를 통해 메모리 및 처리시간을 절약하게 된다. 또한 동적으로 지지율이 높은 속성에 대한 분석 상세도를 높여주기 때문에 사용자의 관심 영역을 효과적으로 보여준다. 마지막으로 실험을 통하여 제한된 메모리에서 동적 데이타 큐브가 효율적으로 동작함을 검증하였다.

키워드 : 데이타 스트림, OLAP, 데이타 큐브

Abstract Data cube, which is multi-dimensional data model, have been successfully applied in many cases of multi-dimensional data analysis, and is still being researched to be applied in data stream analysis. Data stream is being generated in real-time, incessant, immense, and volatile manner. The distribution characteristics of data are changing rapidly due to those characteristics, so the primary rule of handling data stream is to check once and dispose it. For those characteristics, users are more interested in high support attribute values observed rather than the entire attribute values over data streams.

This paper propose dynamic data cube for applying data cube to data stream environment. Dynamic data cube specify user's interested area by the support ratio of attribute value, and dynamically manage the attribute values by grouping each other. By doing this it reduce the memory usage and process time. And it can efficiently shows or emphasize user's interested area by increasing the granularity for attributes that have higher support. We perform experiments to verify how efficiently dynamic data cube works in limited memory usage.

Key words : Data Stream, OLAP, Data Cube

· 이 논문은 교육과학기술부, 지식경제부, 노동부의 출연금 및 보조금으로 수행한 최우수실험실지원사업과 2008년도 정부(교육과학기술부)의 재원으로 한국과학재단의 국가지정연구실사업(No.R0A-2006-000-10225-0)으로 수행된 연구입니다.

† 정 회 원 : Telcoware 네트워크사업부 Framework Solution 팀
sdh@telcoware.com
†† 학생회원 : 연세대학교 컴퓨터과학과
bidol@database.yonsei.ac.kr
††† 종신회원 : 연세대학교 컴퓨터과학과 교수
leewo@database.yonsei.ac.kr

논문접수 : 2008년 2월 27일
심사완료 : 2008년 6월 13일

정보과학회논문지: 데이타베이스 제35권 제4호(2008.8)

1. 서 론

다차원 데이타 분석에서 데이타베이스 및 데이타 웨어하우스 시스템에서의 OLAP은 많은 발전을 이루었다. OLAP은 다차원 데이타에 대한 요약, 통합, 관찰, 공식 적용, 종합의 특성[1]을 갖고 있다. 또한 OLAP에 사용되는 다차원 데이타 모델에 있어[2,3], 데이타 큐브는 차원과 측정치라는 두 요소를 사용하여 데이타 항목의 다양한 특성을 나타낸다. OLAP은 데이타 분석가 및 의사결정권자에게 없어서는 안될 기본적인 도구로 성장하였으며, 데이타 큐브는 다차원 데이타 분석에 성공적으로 적용되었다[4]. 이즈음 정보 기술의 급속한 발전으로 인해 여러 응용 범위에서 생성되는 정보의 양이 이전 어느 때보다 급속히 증가하고 있고, 유비쿼터스 시대가 도래함에 따라 데이타 스트림의 양도 빠르게 증가하고 있다. 데이타 스트림은 다른 응용 범위에서의 데이타와는 다르게 실시간에 지속적으로 방대하게 생성되며, 데이타의 분포적 특성이 빠르게 변한다. 따라서 발생되는 다차원 데이타 스트림을 한정된 메모리 공간에 모두 저장하는 것은 불가능하다. 이런 특성을 고려하여 데이타 스트림을 처리하기 위해서는 다음과 같은 조건을 만족해야 한다[5]. 첫째, 데이타 스트림에서 각 트랜잭션 정보는 단 한번만 읽고 처리해야 한다. 둘째, 새로운 데이타가 지속적으로 생성되더라도 한정된 물리적 메모리 공간에서 처리해야 한다. 셋째, 새롭게 생성된 데이타는 가능한 빠르게 처리되어야 한다. 마지막으로, 데이타 스트림에서 갱신된 최신의 결과는 필요시 즉시 제공되어야 한다. 따라서 데이타 스트림 처리에 기존의 데이타 큐브를 그대로 사용하는 데는 한계가 있다.

최근 들어 이런 한계점을 극복하기 위하여 데이타 스트림에 데이타 큐브를 적용하려는 연구가 진행되고 있다[6]. 스트림 큐브(Stream Cube)는 차원 속성에 대하여 최소 관심 계층(m-layer)와 관찰 계층(o-layer)을 지정하고, 이 두 계층 사이의 차원 속성만을 저장한다. 이를 통해 전체 데이타 큐브가 아닌 관심영역에 대한 데이타 큐브만을 구성하여 처리하는데 초점을 맞추고 있다. 하지만 데이타 스트림에서 의미 있는 정보가 나타나는 차원 속성은 시간에 따라 변하므로, 의미 있는 최소 관심 계층과 관찰 계층을 선택하여 데이타 큐브를 구성하는 것은 쉽지 않다. 또한 데이타 큐브에 속하지 않았던 차원 속성이 시간이 지남에 따라 중요한 의미를 가지게 되는 경우에는 최소 관심 계층과 관찰 계층을 수정하여 데이타 큐브를 다시 구성하지 않는 한 데이타 스트림에 대한 분석을 수행할 수 없는 단점을 가지게 된다.

이와 같이 차원 속성에 대해 사용자 관심 영역만을 저장하기 때문에 생기게 되는 결점을 극복하기 위하여 본 논문에서는 차원 속성 값들을 동적으로 그룹화하는 방식을 이용한 동적 데이타 큐브(dynamic data cube)를 제안 한다. 동적 데이타 큐브에서는 다차원 데이타 스트림에 대하여 정점 큐보이드에서부터 기본 큐보이드까지의 한 경로에 속해있는 큐보이드 전체에 대한 정보를 저장하기 때문에 연산을 통해 전체 데이타 큐브에 대하여 분석을 수행할 수 있다. 또한 사용자의 관심 영역을 벗어나는 속성 값들은 확장 및 축소를 거쳐 동적으로 그룹화시켜 관리함으로써, 메모리 및 처리시간을 절약한다. 관심 영역을 벗어나는 부분이 그룹화되면서 이에 대한 분석상세도를 얼마간 잃어버리겠지만, 관심 영역에 대한 상세도를 높여가기 때문에 사용자에게 유용한 정보를 전달해 줄 수 있다.

2. 관련 연구

2.1 데이타 큐브

데이타 큐브는 데이타를 다차원으로 모델링하며, 차원(Dimension)과 사실(Fact)로 정의된다[7]. 차원이란 조직이 데이타 레코드를 운용하는 이유의 대상이 되는 측면을 의미하고, 사실이란 숫자적으로 표현되는 값을 의미한다. 예를 들어 자동차 Sales에 대하여 Company, Region, Color에 따라 자동차가 어떤 판매 경향을 나타내는지 알아보기 위하여 데이타 큐브를 구성한다면, 그림 1과 같이 나타낼 수 있게 된다. 그림 1(a)사실 테이블은 각 그림 1(b), (c), (d)의 차원 뷰로 나타낼 수 있으며, 각 차원에서의 큐보이드들은 그림 1(e)와 같은 래티스 구조를 가진다. 이 테이블들은 관계 데이타의 집합들을 어떻게 분석 할 지를 나타내게 되며, 가능한 집합의 수는 원래의 데이타를 계층적으로 연결할 수 있는 모든 가능한 방법에 의해 결정된다.

2.2 스트림 큐브

데이타 스트림에 데이타 큐브를 적용하기 위한 방법으로 스트림 큐브(Stream Cube)[2]라는 방법이 제안되었다. 스트림 큐브는 다음 세 가지 특징을 가진다.

첫 번째로, 시간 차원을 요약하기 위하여 경동 시간 구조(Tilted Time Frame)를 이용한다. 기존의 OLAP처럼 요약 데이타를 추가 저장하는 방식은 제한된 메모리를 사용하는 데이타 스트림에는 적합하지 않기 때문에 기존 연구[8,9]와 같이 데이타 스트림의 특성을 기반으로 데이타 큐브를 재구성하였다. 데이타 분석가들은 오래전 데이타보다는 최근의 데이타변화에 더욱 관심이 많기 때문에 최근의 데이타일수록 정밀하게 저장되길 원한다. 아래 그림 2는 경동 시간 구조를 사용하여 최근 1년 동안의 데이타를 저장한 예이다. 현재시간에 가까울수록 보다 정밀하게 데이타를 저장한다.

BMW	America	White	343
BMW	America	Red	330
BMW	America	Blue	297
BMW	Europe	White	132
BMW	Europe	Red	121
BMW	Europe	Blue	85
Ford	America	White	175
Ford	America	Red	145
Ford	America	Blue	103
Ford	Europe	White	143
Ford	Europe	Red	120
Ford	Europe	Blue	135

(a) 사실 테이블

BMW	1308
Ford	821

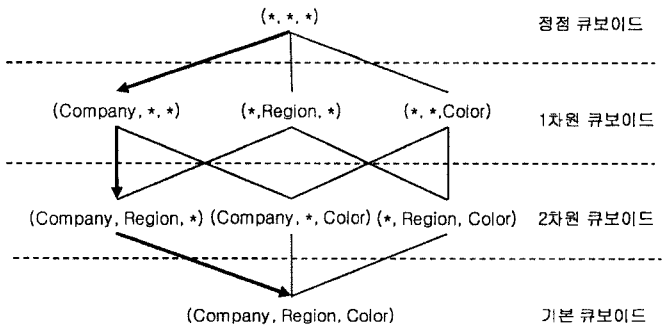
(b) 1차원 뷰

BMW	970	338
Ford	423	398

(c) 2차원 뷰

BMW	343	132	330	121	297	85
Ford	175	143	145	120	103	135

(d) 3차원 뷰



(e) 데이터 큐브 래티스 구조

그림 1 데이타 큐브의 예

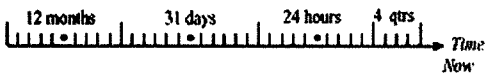
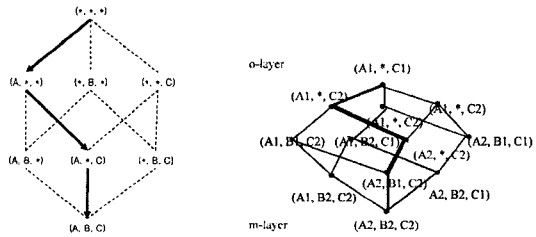


그림 2 경동 시간 구조의 예

매 분마다 데이타가 들어오게 될 경우, 15분이 되는 시점에 데이타를 집계하여 쿼터에 저장하고, 새로 들어온 데이타는 이전처럼 분단위로 저장한다. 이런 방식은 모든 요약 데이타를 분단위로 저장하는 것보다 상당한 메모리 절감 효과를 가져온다. 하지만 이런 저장 방식으로 인하여 과거 데이타로 갈수록 세세한 질의를 할 수 없게 된다.

두 번째로, 스트림 큐브는 사용자 관심 큐보이드들 만을 이용하여 데이타 큐브를 구성한다[10,11]. 빙산 질의(Iceberg Query)[12,13]개념을 차원 속성에 적용하여 사용자 관심 영역을 최소 관심 계층과 관찰 계층으로 정의하고, 최소 관심 계층과 관찰계층 사이에 존재하는 큐보이드만을 이용하여 데이타 큐브를 구성하여 저장공간을 줄인다.

그림 3(a)는 데이타 큐브이다. 여기서 화살표 방향은 정점 큐보이드에서 기본 큐보이드까지의 경로 중 한 개의 경로를 나타낸다. 그림 3(b)는 스트림 큐브에서 사용되는 데이타 큐브로써 사용자가 지정한 관심 영역인 최소 관심 계층과 관찰 계층 사이에 위치한 큐보이드들



(a) (b)

그림 3 스트림 큐브

만으로 구성된다. 따라서 이 경우 일단 스트림 큐브가 구성된 후에는 최소 관심 계층의 아래 계층과 관찰 계층의 위 계층에 대한 질의를 할 수 없다.

세 번째로, 스트림 큐브에서는 최소 관심 계층과 관찰 계층에 해당하는 큐보이드들 중에서 사용자가 이용하는 질의에 가장 많이 노출되는 한 경로의 큐보이드들을 단일 스캔이 가능한 H-tree 기법[14]을 사용하여 저장한다. 저장된 경로 외의 질의에 대해서는 저장된 정보를 활용하여 계산을 통해 응답한다. 여기서 인기 경로는 고정적이며, 통계적 분석이나 경험에 의하여 결정된다. 또한 인기 경로는 도중에 변경할 수 없기 때문에 스트림 데이타의 급격한 변화에 능동적으로 대응하지 못하는 단점이 있다.

3. 동적 데이타 큐브

데이타 스트림은 전통적인 데이타베이스 시스템의 데이타와 달리 실시간에 지속적으로 방대하게 생성된다. 대표적인 예로는 센서 데이타, 증권, 날씨, 콜센터, 웹 페이지 클릭 로그 등을 들 수 있으며, 다음과 같이 정의된다.

1. 데이타 스트림 D^t는 과거부터 현재 시점 t까지의 튜플 T^t를 포함하는 집합을 의미한다. $D^t = \{T^1, T^2, T^3, ..., T^t\}$ 따라서, $|D^t|$는 현재 데이타 집합 D^t에 포함된 튜플의 총 수를 의미한다.
2. 튜플 T는 두 가지 종류의 속성으로 구성된다. 하나는 차원을 나타내기 위한 속성으로 DIM으로 표기하고, 다른 하나는 측정치를 나타내기 위한 속성으로 M으로 표기한다. 즉, $DIM \cup M = T$이고, DIM ∩ M = ϕ 이다.
3. 측정치 속성 M은 데이타 베이스에서 사용하는 COUNT, SUM, AVG 등과 같은 집계 함수로 연산이 가능하다.
4. DIM은 다차원 공간에서 하나의 셀(Cell)을 이룬다.

데이타 스트림은 스키마를 가지는 튜플 즉 셀이 실시간에 지속적으로 방대하게 무한히 생성되는 튜플의 집합으로 정의 될 수 있다.

본 논문에서는 앞서 기술한 스트림 큐브의 단점을 보완하기 위해 동적 데이타 큐브를 제안한다. 동적 데이타 큐브는 전체 큐보이드를 이용해서 데이타 큐브를 구성하는 대신, 차원 속성 값 그룹화를 통하여 사용자 관심 영역에 대한 분석 상세 정도를 유지한다. 사용자 관심 영역은 최소 지지율 S_{min} 이상 최대 지지율 S_{max} 이하인 영역으로 그림 4와 같다.

최소 지지율 S_{min} 이하의 지지율을 가지는 그룹은 축소를 통하여 지지율을 높여 사용자 관심 영역에 속하게 하고, 최대 지지율 S_{max} 이상의 지지율을 가지는 그룹은 확장을 통하여 지지율을 낮추어 사용자 관심 영역에 속하게 한다. 이로써 차원 속성 값 그룹이 최대한 사용자 관심 영역에 있게 유도하며, 동적 데이타 큐브의 메모리 사용을 효율적으로 관리할 뿐 아니라, 사용자 관심 영역에 대하여 상세한 차원 속성 그룹 범위를 제공할 수 있게 된다. 물론 이 과정에서 관심 영역을 벗어나는 부분을 그룹으로 관리하게 되므로 이에 대한 분석 상세도는 줄어들게 된다. 차원 속성 값 그룹의 지지율 및 분석을 위한 정보를 저장하기 위하여 동적 데이타 큐브는 정의 1과 같은 통계 정보를 저장한다.

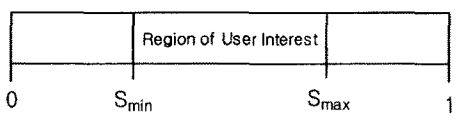


그림 4 사용자 관심 영역

정의 1. 통계 정보

동적 데이타 큐브는 차원 속성 값의 그룹에 대한 차원 속성 정보 및 측정치 속성의 통계 정보를 저장하기 위하여 네 가지의 정보(I, C, R, M)를 노드에 저장한다.

1. 간격(I) : I는 현재 시간 t에서의 차원 속성 값 그룹의 범위에 대한 간격이다.
2. 빈도수(C) : C는 현재 시간 t에서의 차원 속성 값 그룹에 대한 빈도수의 합이다.
3. 차원 속성 값 그룹의 범위(R) : 현재 노드에 저장된 차원 속성 값 그룹의 범위로 $R = \{DIM_1, DIM_2, DIM_3, ..., DIM_n\}$ 로 나타낸다.
4. 평균 측정치 속성 값(M) : M은 현재 시간 t에서의 차원 속성 값 그룹에 속한 측정치 속성 값들에 대한 평균 측정치 속성 값이다.

정의 1에서 정의한 동적 데이타 큐브의 통계 정보는 그림 1(a)사실 테이블을 이용하여 구성한 그림 1(b), (c), (d)에서의 셀들을 1개 이상 그룹화한 차원 속성 값 그룹에 해당하는 정보를 저장한다.

정의 2. 형제 리스트

동적 데이타 큐브의 형제 리스트는 차원 속성 값 그룹들의 차원 속성 정보 및 측정치 속성 통계 정보를 저장하기 위한 리스트이다.

1. 정의 1에서 정의한 노드로 연결된 싱글 링크드 리스트(Single Linked list)이다. ($S = <N_1, N_2, N_3, ..., N_n>$)
2. 노드간의 연결을 위한 다음 차원 속성 그룹 포인터를 유지한다.

정의 2에서 정의한 형제 리스트는 그림 1(a)사실 테이블을 이용하여 구성한 그림 1(b), (c), (d)에서 차원 속성 값 그룹의 집합으로 이루어진 리스트이다. 예를 들어, 차원 속성 값 그룹의 범위 R = {White}를 가지는 그룹과 범위 R = {Red, Blue}를 가지는 그룹으로 이루어진 형제 리스트는 그림 5와 같이 나타낼 수 있다.

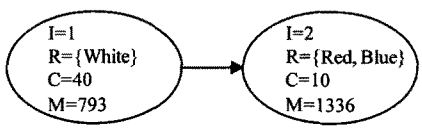


그림 5 동적 데이타 큐브의 형제 리스트 예

정의 3. 1차원 트리

동적 데이타 큐브의 1차원 트리는 1차원 큐보이드들에 대한 차원 속성의 정보 및 측정치 속성의 통계정보를 저장하는 트리이다.

1. 정의 2에서 정의한 형제 리스트 S로 구성된 트리이다.

2. 트리 레벨은 하나의 1차원 큐보이드의 정보를 저장한다.
3. 상위 레벨과 하위레벨은 싱글 링크드 리스트로 구성된다.
4. 현재 레벨의 형제 리스트와 다음 레벨의 형제 리스트를 연결을 위한 다음 레벨 큐보이드 포인터를 유지한다.

정의 3에서 정의한 1차원 트리는 그림 1(a)사실 테이블을 이용하여 구성한 그림 1(e)데이타 큐브 래티스 구조에서 1차원 큐보이드에 해당하는 각 큐보이드들의 형제 리스트로 이루어진 트리이다. 1차원 큐도이드들에 대한 범위 정보 및 카운트 정보만을 간략하게 표시한 1차원 큐보이드 트리는 그림 6과 같이 나타낼 수 있다.

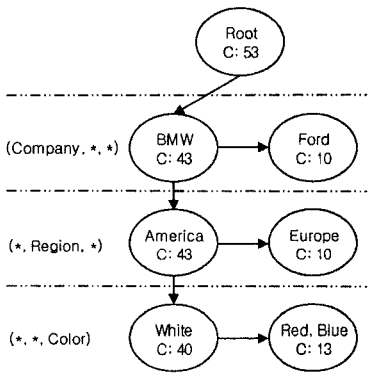


그림 6 1차원 트리의 예

정의 4. 큐브 트리

동적 데이타 큐브의 큐브 트리는 정점 큐보이드부터 기본 큐보이드까지의 한 경로를 구성하는 큐보이드들에 대한 차원 속성의 정보 및 측정치 속성의 통계 정보를 저장 하는 트리이다.

1. 정의 2에서 정의한 형제 리스트로 구성된 트리이다.
2. 트리 레벨 L은 1개 이상의 형제 리스트로 구성되며, 서로 더블 링크드 리스트(Double Linked list)로 연결된다. ($L =< S_1, S_2, S_3, ..., S_n >$)
3. 트리 레벨에 속하는 형제리스트들간의 연결을 위하여, 다음 형제 리스트 헤더 포인터 및 이전 형제 리스트 헤더 포인터를 유지한다.
4. 상위 레벨 과 하위레벨은 싱글 링크드 리스트로 연결된다.
5. 트리에 속하는 레벨들 간의 연결을 위하여 다음 레벨 큐보이드 포인터를 유지한다.

정의 4에서 정의한 큐브 트리는 그림 1(a)사실 테이블을 이용하여 구성한 그림 1(e)데이타 큐브 래티스 구조에서 정점 큐보이드에서부터 기본 큐보이드까지의 한 경로를 구성하는 큐보이드들로 이루어진 트리다. 큐보이드들에 대한 범위 정보 및 카운트 정보만을 간략하게 표시한 큐브 트리는 그림 7과 같이 나타낼 수 있다.

3.1 동적 데이타 큐브의 갱신

데이타 스트림 D^t에서 t시간에 새로운 튜플 T^t이 생성됨에 따라 t-1 시간에서의 큐보이드 트리의 차원 속성 값의 빈도수 C^{t-1} 및 평균 차원 측정치 속성 값 M^{t-1} 또한 갱신되어야 한다. 새로 발생한 튜플의 차원 속성 값이 그룹 범위 R에 속하고 측정치 속성 값이 m^t일 경우, 식 (1)을 통하여 현재 시간 t에서의 차원 속성 값의 빈도수 C^t 및 평균 측정치 속성 값 M^t를 구할 수 있다. 이때 C^{t-1}과 M^{t-1}은 t-1 시간에서의 빈도수와 평균 차원 측정치 속성 값을 나타낸다.

$$C^t = C^{t-1} + 1, \quad M^t = \frac{(M^{t-1} \times C^{t-1}) + m^t}{C^t} \tag{1}$$

동적 데이타 큐브에서는 사용자 관심 영역을 벗어나는 차원 속성 값 그룹에 대하여 재 그룹화하여 최대한 사용자 관심 영역에 위치하도록 그룹의 범위를 조절하는 확장 단계(Expanding Phase) 및 축소(Shrinking Phase)단계를 거친다.

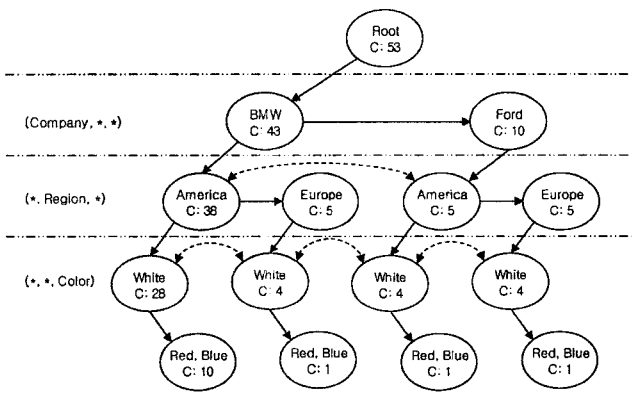


그림 7 큐브 트리의 예

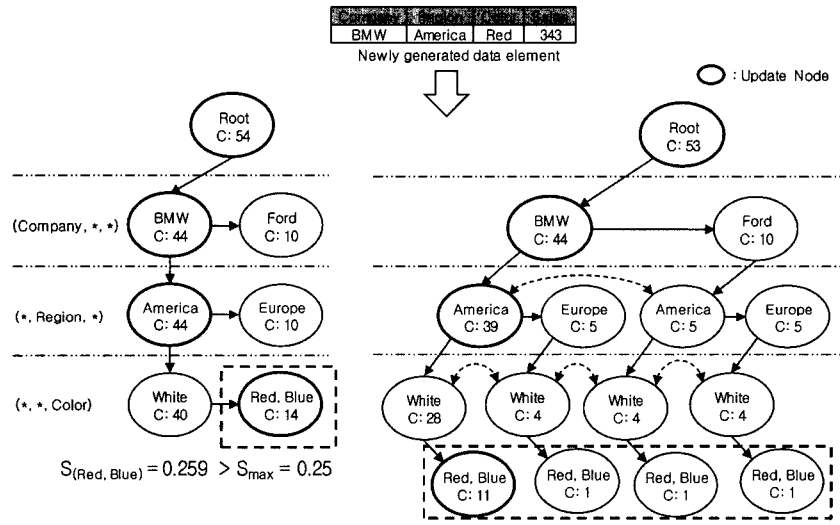
먼저, 확장 단계는 사용자 관심 영역의 최대 지지율인 S_{max}보다 높은 지지율을 가지는 차원 속성 값 그룹에 대하여 범위를 분할하여, 분할된 그룹의 지지율이 최대한 사용자 관심 영역에 속하도록 유도한다. 이 과정은 다음 순서로 수행된다.

1. 1차원 트리에서 차원 속성 값 그룹의 빈도수 지지율이 사용자가 지정한 최대 지지율 S_{max}보다 높은 그룹에 대하여 사용자가 정의한 λ(lambda)만큼 그룹을 분할한다.
2. 큐브 트리는 1차원 트리와 같은 레벨에 있는 차원 속성 값 그룹에 대하여 1차원 트리와 동일하게 그룹을 분할한다.
3. 분할될 차원 속성 값 그룹의 간격이 사용자가 지정한 단위 간격보다 작다면, 그 그룹은 확장되지 않는다.

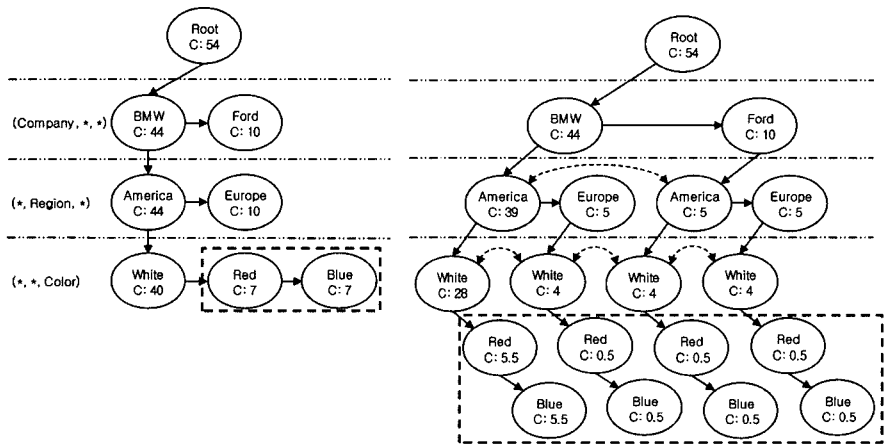
사용자가 지정한 S_{max}보다 높은 빈도수 지지율을 갖는 차원 속성 값 그룹은 사용자의 관심 영역을 벗어나게 되므로, 그룹을 λ만큼 분할하여 분할된 차원 속성 값 그룹이 사용자 관심 영역에 위치할 수 있도록 유도한다. 분할된 그룹의 빈도수와 평균 측정치 속성 값들은 식 (2)를 이용하여 계산된다.

$$C_{new} = C \times I_{new}/I = C \times \frac{1}{\lambda}, \quad M_{new} = M \tag{2}$$

식 (2)에서 분할된 그룹의 간격 I_{new}는 I를 λ등분한 값이다. 분할된 그룹의 빈도수 C_{new} 역시 C의 λ등분한 값을 갖게 된다. 확장된 차원 속성 값 그룹의 평균 측정치 속성 값 M_{new}은 평균 값이기 때문에 확장 단계에서



(a) 확장 전



(b) 확장 후

그림 8 확장 단계 예 (S_{min} = 0.1, S_{max} = 0.25, λ = 2)

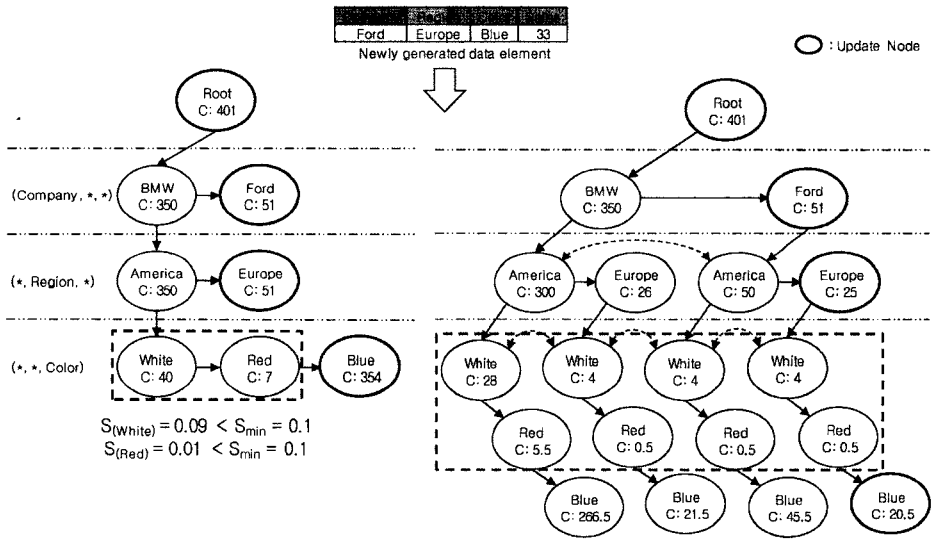
생성된 새로운 그룹의 평균 측정치 속성 값으로 추가적인 계산 없이 사용이 가능하다.

그림 8에서는 점선 박스로 둘러싸인 레드와 블루를 범위로 갖는 차원 속성 값 그룹의 빈도수 지지율이 사용자가 정의한 최대 지지율인 S_{max} = 0.25를 넘어 λ = 2개의 차원 속성 값 그룹으로 분할되는 확장 단계에 대한 예를 보여주고 있다.

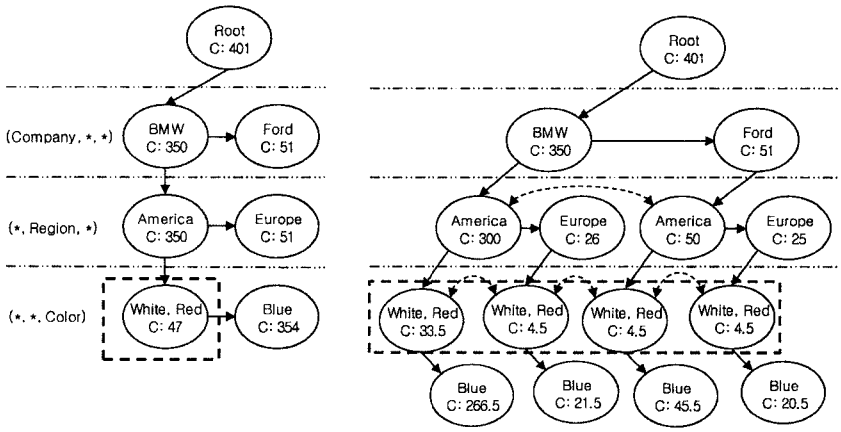
다음으로 축소 단계는 사용자 관심 영역의 최소 지지율인 S_{min}보다 낮은 지지율을 가지는 연속된 차원 속성 값 그룹을 병합하여, 축소된 그룹의 지지율이 최대한 사용자 관심 영역에 속하도록 유도한다. 이 과정은 다음 순서로 수행된다.

1. 1차원 트리에서 차원 속성 값 그룹의 빈도수 지지율이 사용자가 지정한 최소 지지율 S_{min}보다 낮은 연속된 차원 속성 값 그룹들을 하나의 그룹으로 병합한다.
2. 큐브 트리는 1차원 트리와 같은 레벨에 있는 차원 속성 값에 대하여 1차원 트리와 동일하게 그룹을 병합한다.
3. 병합될 차원 속성 값 그룹들이 연속적이지 않다면, 그 그룹들은 병합되지 않는다.

사용자가 지정한 S_{min}보다 낮은 빈도수의 지지율을 갖는 연속된 차원 속성 값 그룹들은 사용자 관심 영역에서 벗어나 있는 그룹들이므로 해당 그룹들을 하나의 차원 속성 값 그룹으로 병합하여, 축소된 그룹이 사용자 관심 영역에 위치하도록 유도 한다. 병합되는 차원 속성 값 그룹들의 빈도수와 평균 측정치 속성 값들은 식 (3)



(a) 축소 전



(b) 축소 후

그림 9 축소 단계 예 (S_{min} = 0.1, S_{max} = 0.25)

을 이용하여 구할 수 있다.

$$C_{new} = \sum_{i=1}^{n} C_i, \quad M_{new} = \sum_{i=1}^{n} (M_i \times C_i) \Big/ C_{new} \qquad (3)$$

연속된 n개의 차원 속성 값 그룹의 병합에 있어 새로운 빈도수는 n개의 그룹의 빈도수의 합으로 계산 가능하다. 또한 평균 측정치 속성 값은 n개의 차원 속성 값 그룹의 평균 측정치를 합으로 환산한 값들의 새로운 빈도수에 대한 평균으로 계산 가능하다.

그림 9에서는 그림 8에서 확장되었던 동적 데이타 큐브에 대하여 축소 단계의 예를 보여주고 있다. 블루를 범위로 갖는 차원 속성 값 그룹의 빈도수가 증가함에 따라, 화이트를 범위로 갖는 그룹과 레드를 범위로 갖는 그룹의 빈도수 지지율이 사용자가 정의한 최소 지지율 S_{min} = 0.1 보다 낮아져 병합되고 있다.

새로운 튜플의 발생에 따른 동적 데이타 큐브의 갱신, 그리고 사용자가 정의한 빈도수 지지율에 따른 확장 및 축소 단계를 수행하는 알고리즘은 아래 그림 10과 같이 구현된다.

3.2 동적 데이타 큐브의 적응적 메모리 사용량 최적화

동적 데이타 큐브에서는 최대 빈도수 지지율 S_{max} 및 최소 빈도수 지지율 S_{min}의 범위를 벗어나는 그룹을 확장과 축소 단계를 거쳐 재 그룹화 함으로써, 차원 속성 값 그룹이 사용자 관심 영역에 위치하도록 유도한다. 하지만 데이타 스트림은 시간 흐름에 따른 변화 가능성이 크므로, 동적 데이타 큐브에서 유지되는 정보의 양도 변한다. 이러한 처리를 제한된 메모리 공간에서 효율적으로 수행하기 위해서는 데이타 스트림의 변화에 따라 초기에 사용자가 지정한 최대 빈도수 지지율 S_{max} 및 최소 빈도수 지지율 S_{min}의 값을 동적으로 적응 시킬 필요가 있다.

동적 데이타 큐브의 메모리 사용량은 사용자가 정의한 지지율인 S_{min}과 S_{max}에 영향을 받는다. 즉, S_{min} 및 S_{max}의 값을 증가 시키게 되면 축소 단계를 통해 한 노드에 그룹화되는 차원 속성 값 그룹이 많아지게 되고, 확장 단계를 통해 분할되는 차원 속성 값 그룹은 줄어들게 되어, 결과적으로 동적 데이타 큐브 전체에서 사용되는 메모리는 감소하게 된다.

따라서, 본 논문에서는 사용자가 정의한 최대 메모리 사용량 M_{max}를 넘지 않으면서, S_{min}과 S_{max}을 조절하여 메모리 공간을 효율적으로 사용하는 적응적 메모리 최적화 방법으로 세 가지 타입의 방법을 제안한다.

첫 번째는 강제 축소 단계 메모리 적응 방법(Force Shrinking Phase Memory Adaptation Method)이다. 동적 데이타 큐브의 갱신시 실제 확장 단계에 들어가기 전에 예상 메모리 사용량을 계산한다. 실제 확장 단계를 수행하였을 경우의 예상 메모리 사용량을 M_{exp}라고 하면, M_{exp}에 대한 계산은 동적 데이타 큐브의 1차원 트리를 탐색하여 확장 예상 노드 수를 조사하여 식 (4)를 이용해 계산할 수 있다.

$$M_{\exp} = [(|EN_1| + |EN_2| + \ldots + |EN_n|) + |EN_1| \times |EN_2| \times \ldots \times |EN_n|] \times N_{size} \qquad (4)$$

$|EN_n|$: 1차원 트리의 n 레벨에서 예상 노드 수

N_{size} : 한 노드의 메모리 크기

예상 메모리 사용량인 M_{exp}가 사용자가 정의한 최대 메모리인 M_{max}보다 클 경우에 강제 축소 단계 메모리 적응 방법이 다음 순서로 수행된다.

1. 사용자가 정의한 초기 S_{min}과 S_{max}값에 대하여 초기 설정 간격을 유지한 채 S_{min}과 S_{max}값을 사용자가 정의한 δ(delta)만큼 증가시킨다.
2. M_{exp}가 M_{max}보다 작아질 때까지, δ 만큼 지지율을 증가 시키면서 반복적으로 축소 단계를 수행한다.
3. 메모리 적응 순서는 가장 많은 수의 노드가 줄어드는 트리 레벨 순으로 진행된다.

두 번째는 지연 확장 단계 메모리 적응 방법(Delaying Expanding Phase Memory Adaptation Method)이다. 강제 축소 단계 메모리 적응 방법은 예상 메모리 사용량이 최대 메모리 M_{max}를 넘었을 경우 사용하는 방법으로써, 동적 데이타 큐브에 대한 전체적인 노드 축소를 사용한다. 따라서 강제 축소 단계 메모리 적응 방법은 즉각적인 메모리 사용량 감소의 효과를 보이지만, 그만큼 많은 처리 시간이 필요하게 된다. 데이타 스트림은 데이타가 무한히 발생하는 특징을 가지기 때문에, 현재 주요하게 발생하는 데이타 스트림에서의 차원 속성 값이 주요한 의미를 가지는 값이라면, 일정 기간이 지난 후에도 주요하게 발생될 확률이 높다. 따라서 현재 확장 단계에서 분할될 차원 속성 값 그룹들 중 중요한 의미를 가지는 그룹들은 일정 시간이 지난 후에도 확장 단계에서 분할될 가능성이 높다. 이를 감안하여 지연 확장 단계 메모리 적응 방법은 이와 같은 그룹에 대해 강제 축소 메모리 적응이 일어날 가능성을 줄이게 된다. 이 방법은 최대 메모리 M_{max}보다 작은 M_{upper}를 정의하고, 예상 메모리 사용량인 M_{exp}가 사용자가 정의한 최대 메모리인 M_{max}보다 작고, M_{upper}보다 클 경우에 확장 단계에서 분할되어야 하는 그룹의 확장시기를 사용자가 정의한 주기인 P 만큼 지연시킨다. 이를 통해 현재 메모리 사용량을 고수하고, 지지율이 사용자가 정의한 S_{min}보다 작아 축소 단계에서 병합되는 차원 속성 값 그룹들의 메모리를 확보하게 된다.

마지막은 지지율 회복 메모리 적응 방법(Support Recovery Memory Adaptation Method)이다. 강제 축소

```
1: function traverse(odc, pc, T^t, k)
/*   odc : the currently traversing 1-D tree node                */
/*   pc: the cuttently traversing cube tree node                 */
/*   T^t : a newly generated tuple                               */
/*   k : the traversing level of cube tree                       */
/*   u : the unit interval of dimensional attribute value        */
/*   h : the expanding phase factor                              */

    /* Updating Phase */
2:  find a 1-D tree node whose interval includes T_k^t   odc.I ∋ T_k^t;
3:  update the distribution statistics of 1-D tree node;
4:  find a cube tree node whose interval includes T_k^t   pc.I ∋ T_k^t;
5:  update the distribution statistics of cube tree node;
   /* Expanding phase */
6:   if (odc.t/|D^t| ≥ S_max and odc.I > u) {
7:        partition odc into odc_1^k,...,odc_h^k;
8:        estimate the distribution statistics of odc_1^k,...,odc_h^k respectively;
9:        partition pc into pc_1^k,...,pc_h^k;
10:       estimate the distribution statistics of pc_1^k,...,pc_h^k respectively;
11:  }
    . /* Shrinking Phase */
12:  else if (odc.c^t/|D^t| ≤ S_min) {
13:       find all the satisfied 1-D tree nodes within the given continuous range;
15:       merge 1-D tree nodes with the satisfied 1-D tree nodes consecutive to it ;
16:       estimate the distribution statistics of odc_1^k,...,odc_h^k respectively;
17:       find all the satisfied cube tree nodes within the given continuous range;
18:       remove their child nodes if any;
19:       merge cube tree nodes with the satisfied cube tree nodes consecutive to it ;
20:       estimate the distribution statistics of pc_1^k,...,pc_h^k respectively;
21:  }
22:  if (odc ≠ null)
         traverse(odc, pc , T^t, k+1);
23:  end // function end
24:  main(odc_root, pc_root, D^t)
/*       odc_root : a root of a 1-D tree                 */
/*       pc_root : a root of a cube tree                 */
25:      for a newly generated data element T^t do
26:           traverse(odc_root, pc_root, T^t, 1);
27:           t=t+1;
28:      end
29:  end
```

그림 10 동적 데이타 큐브 갱신 알고리즘

단계 메모리 적응 방법에서 증가된 지지율이 계속 그대로 유지될 경우, 사용자 초기의 관심 영역과는 거리가 먼 분석을 진행하게 된다. 따라서, 예상 메모리 사용량인 M_{exp}가 사용자가 정의한 M_{upper}보다 작을 경우에 현재의 S_{min}과 S_{max} 값에 대하여 사용자가 정의한 초기 S_{min}과 S_{max}값에 가까워지도록 초기 설정 간격을 유지하면서 현재의 S_{min}과 S_{max}값을 사용자가 정의한 δ만큼 감소시켜 지지율을 회복한다.

3.3 동적 데이타 큐브에서의 OLAP 연산

데이타 큐브에서 의미 있는 데이타를 탐색하기 위하여 OLAP 시스템에서는 Roll-up, Drill-down, Slice, Dice, Pivot등의 연산이 사용된다[15]. Roll-up 연산은 차원의 계층 구조를 한 단계 상승 시키거나 차원을 감소 시키면서 데이타 큐브의 집계(aggregation)를 수행한다. Drill-down 연산은 Roll-up 연산의 역으로, 데이타의 상세 레벨을 탐색하는 것이다. Slice 연산은 주어진 데이타 큐브에서 한 차원을 선택하는 것으로, 결과는 서브 큐브가 된다. Dice 연산은 두 개 이상의 차원을 선택하면서 서브 큐브를 생성하는 것이고, Pivot 연산은 데이타의 축을 회전하여 다른 관점에서 볼 수 있게 하는 시각화를 위한 연산이다.

동적 데이타 큐브에서는 차원의 증가 및 감소에 따른 Roll-up과 Drill-down 연산을 트리의 탐색 레벨 조정을 이용하여 수행한다. 그림 11은 큐브 트리에서의 Roll-up, Drill-down 연산 과정을 보여주고 있다.

큐브 트리의 탐색 레벨을 2레벨에서 3레벨로 조정하면, (Company, *, *) 큐보이드에서 (Company, Region, *) 큐보이드로 Drill-down 연산을 수행할 수 있게 된다. 이와 반대로 탐색 레벨을 3레벨에서 2레벨로 조정하면, (Company, Region, *) 큐보이드에서 (Company, *, *) 큐보이드로 Roll-up 연산을 수행할 수 있게 된다.

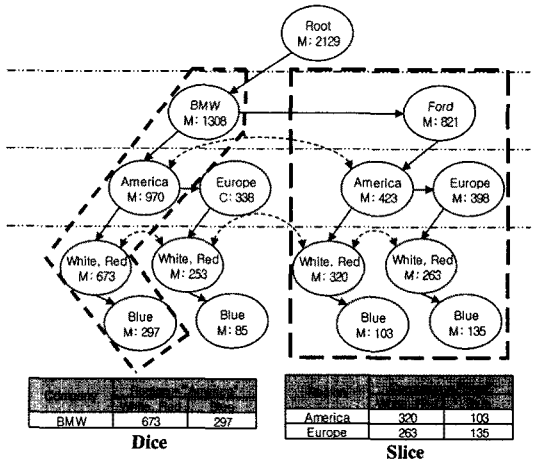


그림 12 큐브 트리를 이용한 Dice, Slice연산 예

Slice와 Dice는 큐브 트리의 부분 탐색을 통하여 사용자가 원하는 결과를 보여준다. 그림 12에서 Slice는 Slice for Company = "Ford"연산의 수행 결과이며, Dice는 Dice for Company = "BMW" and Region = "America" and (Color= "White, Red" or "Blue")연산의 수행 결과이다. 동적 데이타 큐브에서 1차원 큐보이드들은 1차원 트리에 그리고 정점 큐보이드부터 기본 큐보이드까지의 한 경로에 속하는 큐보이드들은 큐브 트리에 저장하기 때문에, 두 트리에 의해 표시되지 않은 큐보이드에 대한 응답은 트리를 순회하면서 계산을 통하여 수행된다.

그림 13은 동적 데이타 큐브에 저장되지 않는 2차원 큐보이드들 중 (Company, *, Color) 큐보이드에 대한 응답 예를 보여주고 있다. 큐브 트리는 4레벨까지 순회하면 기본 큐보이드에 대한 결과를 얻을 수 있게 된다.

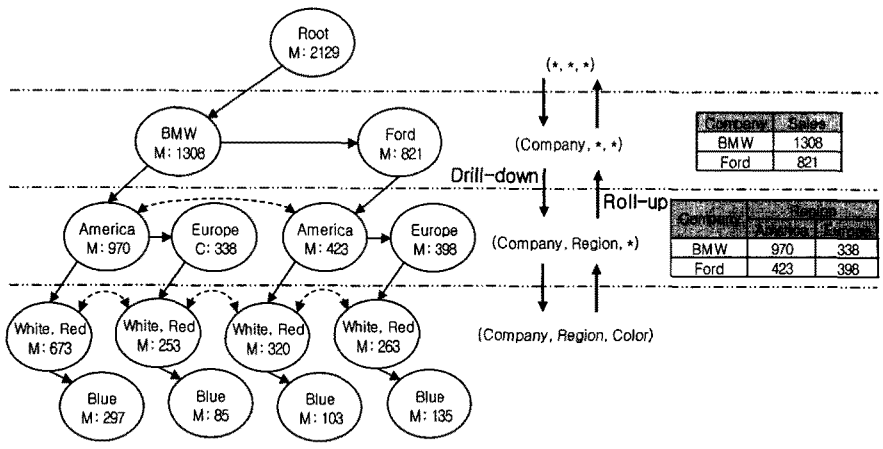


그림 11 큐브 트리를 이용한 Roll-up, Drill-down연산 예

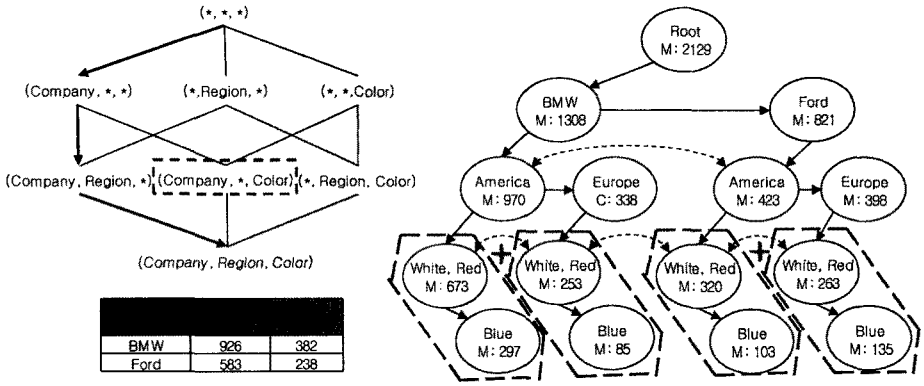


그림 13 큐브 트리를 이용한 (Company, *, Color)큐보이드 응답 예

(Company, *, Color) 큐보이드의 경우 큐브 트리는 4 레벨까지 순회 중 Region 차원의 하위 트리에 대한 합을 통하여 (Company, *, Color)에 대한 결과를 계산하게 된다.

4. 성능 평가

본 장에서는 동적 데이타 큐브의 성능을 분석하기 위하여, 지프 분포(Zipfian distribution)[16]에 의해 생성한 여러 데이타 집합을 이용하였다. 지프 분포는 인구수, 단어 사용의 빈도수, 책의 판매량 등 많은 실 세계 데이타들의 모델링이 가능하며, 1.8에서 3.51까지의 다양한 분포를 형성하고 있다. 본 실험에서 사용한 데이타 집합은 1.8에서 3.4의 지프 분포로써, 차원 속성은 5개에서 8개, 튜플은 십 만개로 구성되어 있다. 또한, 각 차원 속성 값은 서로 다른 100개의 값을 가지고 있다. 데이타 스트림 환경을 모의 실험하기 위해 데이타 집합의 튜플은 순서대로 하나씩 처리되었다.

실험에서 사용된 정확도 A는 데이타 집합을 데이타베이스로 처리한 측정치 R.C와 동적 데이타 큐브로 처리한 측정치 G.C간의 상대 오차율로 식 (5)를 통하여 계산하였다.

$$A = 1 - \frac{\sum_{i=1}^{n}\sum_{k=1}^{m}\left|R.C_i^k - G.C_i^k\right|}{R.C} \quad (5)$$

그림 14는 지프 분포에 따른 데이타 큐브와 동적 데이타 큐브의 메모리 사용량을 도시하였다. 전체 데이타에 대하여 사용자가 정의한 일정 지지율 이하의 차원 속성 값은 그룹화되기 때문에 데이타 큐브보다 동적 데이타 큐브의 메모리가 적게 소비되는 것을 볼 수 있다. 그림 15는 그림 14의 지프 분포 1.8에 대하여 튜플 증가에 따른 메모리 사용량을 도시하였다. 데이타 큐브는 시간이 지나면서 발생되는 새로운 차원 속성 값을 모두 저장하기 때문에 메모리 사용량이 지속적으로 증가하는 반면에, 동적 데이타 큐브는 차원 속성 값이 새로이 발생되어도 지지율이 낮다면, 그룹으로 관리되기 때문에 동일 데이타 분포에서 안정적인 메모리 사용량을 나타낸다.

그림 16은 그림 15 실험에 대한 정확도를 도시하였다. 동적 데이타 큐브는 차원 속성 값의 지지율 변화에 따라 확장 단계와 축소 단계를 거치기 때문에 200,000 구간에서 낮은 정확도를 보이고 있지만, 데이타 스트림의 분포가 안정되어 더 이상의 확장 단계 및 축소 단계가

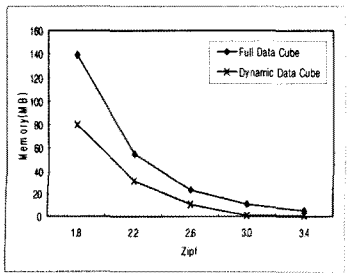


그림 14 메모리 사용량

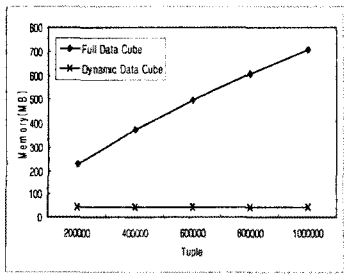


그림 15 메모리 사용량

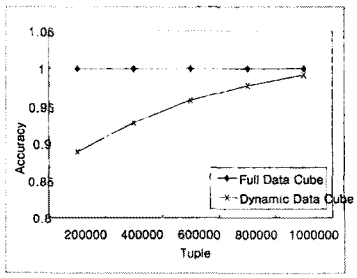


그림 16 정확도

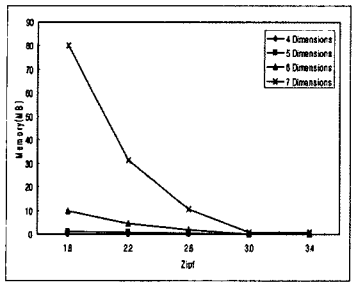


그림 17 메모리 사용량

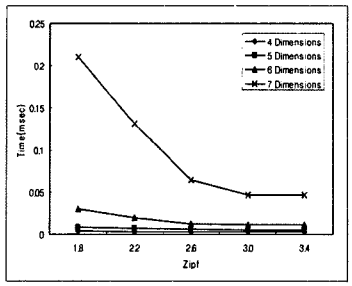


그림 18 처리 시간

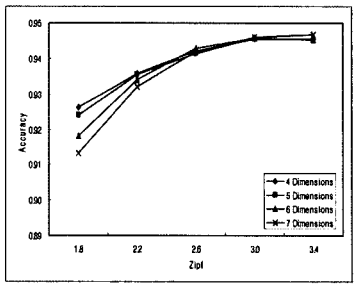


그림 19 정확도

진행되지 않음으로 인해, 정확도가 점점 상승하는 것을 보여준다.

그림 17은 각 지프 분포에서 차원의 수에 따른 메모리 사용량을 도시하였다. 데이타 스트림은 지프 분포를 0.4 구간으로 구분하여 도시하였다. 그림 18은 그림 17의 실험에서 데이타 큐브의 갱신 처리 시간을 도시하였다. 지프 분포는 높은 값을 가질수록 차원 속성 값의 지지율이 한 곳으로 몰리는 현상을 나타내기 때문에, 더 많은 차원 속성 값들이 그룹화되어 동적 데이타 큐브의 메모리 사용량은 감소하는 경향을 보이게 되며, 데이타 큐브의 처리시간 또한 메모리 사용량과 같은 감소 경향을 보이게 된다. 또한 차원수가 증가할수록 트리 높이가 증가하기 때문에 메모리 사용량과 처리 시간이 증가하게 된다.

그림 19는 지프 분포의 각 구간에서의 정확도를 도시하였다. 각 구간에서 정확도는 미세한 차이를 보여준다. 지프 분포가 높은 값을 가질수록 동적 데이타 큐브에서 단위 간격을 가지는 차원 속성 값 그룹과 그룹화되는 차원 속성 값이 많아지기 때문에, 미세하지만 정확도가 높아지는 경향을 보이고 있다.

그림 20은 차원의 각 구간에서 데이타 큐브의 동적 데이타 큐브 구성 경로에 속하는 큐보이드에 대한 평균 질의 응답시간과 데이타 큐브의 동적 데이타 큐브 구성 경로에 속하지 않는 큐보이드에 대한 평균 질의 응답시간을 도시하였다. 동적 데이타 큐브 구성 경로에 속하는 질의 응답시간과 속하지 않는 질의 응답시간이 차이를 보여준다. 구성 경로에 속하지 않는 큐보이드에 대한 질의의 경우 동적 데이타 큐브의 탐색과 집계가 필요하기 때문에, 구성 경로에 속하는 큐보이드의 경우보다 비교적 많은 응답시간이 걸리게 된다.

그림 21은 한정된 메모리 공간에서 동적 데이타 큐브의 메모리 사용량을 도시하였다. 적응적 메모리 최적화를 사용한 계열에서는 초기에 확장 단계의 증가로 인하여 늘어나는 메모리 사용량을 강제 축소 단계 메모리 적응 방법을 통해 억제 하고 있는 모습을 보여주고 있다. 그 이후 적응적 메모리 최적화를 하지 않은 계열에서는 계속적인 확장 단계로 인한 메모리 사용량 증가를 보이는데 반하여, 적응적 메모리 최적화를 사용한 계열에서는 지연 확장 단계 메모리 적응 방법으로 인해 메모리 사용량의 기울기가 완만해 졌음을 알 수 있다. 나중에는 데이타 집합의 분포적 특성이 뚜렷해져, 메모리 사용량이 고정되는 특징을 보이게 된다. 그림 22는 그림 21 실험에서의 정확도를 도시하였다. 적응적 메모리 최적화를 하지 않은 계열에서는 초기의 확장 단계에서의 정확도가 낮은데 반하여, 데이타의 분포적 특성이 나타난 이후에는 데이타의 튜플 수가 증가할수록 정확도가 높아짐을 보이고 있다. 데이타의 분포적 특성이 나타난 후에는 그 데이타의 분포적 특성이 변하지 않는 한, 현재 상태가 지속되므로, 정확도가 증가하게 된다. 적응적 메모리 최적화를 사용한 계열에서는 메모리 적응으로 인하여 확장

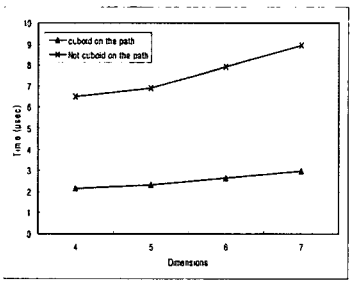


그림 20 질의 응답시간

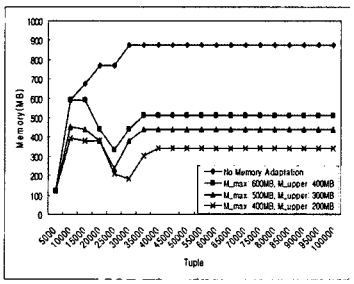


그림 21 메모리 사용량

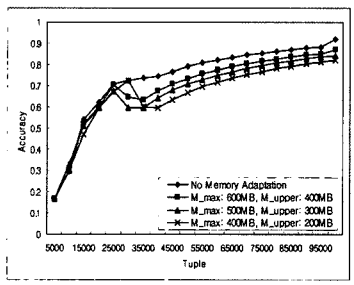


그림 22 정확도

및 축소 단계가 수행되므로, 해당 구간에서는 데이타의 정확도가 떨어지게 된다. 하지만, 메모리 적용 후에는 확장 및 축소 단계를 거치지 않게 되므로 데이타의 정확도가 상당 부분 회복되는 것을 볼 수 있다.

5. 결 론

본 논문에서는 데이타 스트림 환경에서 다차원 분석 데이타 모델인 데이타 큐브를 효과적으로 적용하기 위한 동적 데이타 큐브를 제안하였다. 데이타 스트림의 특성상 발생하는 데이타를 한정된 메모리 공간에 모두 저장하는 것은 불가능 하며 사용자의 요구에 빠르게 응답해야 한다. 따라서 모든 데이타를 저장하고 관리하기보다는 주어진 메모리 공간을 활용하여 사용자가 유용한 정보를 얻도록 도움을 주는 것이 중요하다. 동적 데이타 큐브에서는 메모리 사용량을 효과적으로 관리하기 위하여, 속성 값을 단위 간격으로 분석하지 않고, 속성 값의 지지율에 따라 사용자 관심 영역을 지정하고 속성 값을 동적으로 그룹화하여 관리함으로써 메모리 및 처리시간을 절약하였다. 이를 통해 관심영역 이외의 부분에 대한 정확도를 얼마간 희생하겠지만, 관심영역에 속한 부분에 대한 정확도를 스트림 환경에서 유지할 수 있도록 하였다. 즉 한정된 메모리를 관심영역이 아닌 부분보다는 관심영역인 부분에 최대한 할애하도록 유도한다. 또한 스트림 데이타 입력에 대해 동적으로 그룹화를 수행하기 때문에 최신의 가장 유용한 정보를 전달해 줄 수 있다. 한편, 동적 데이타 큐브는 적응적 메모리 관리 방법을 통하여 제한된 메모리 공간을 최대한 활용하므로 컴퓨터의 자원 사용을 극대화한다.

참 고 문 헌

[1] Inmon, W.H., Building the Data Warehouse, John Wiley, 1992.

[2] The OLAP Council., "MD-API the OLAP Application Program Interface Version 0.5 Specification," 1996.

[3] Rakesh Agrawal, Ashish Gupta, Sunita Sarawagi, "Modeling multidimensional database. In Proc.," the 13th Intl conference on Data Engineering, Birmingham, U.K., pp. 232-243, 1997.

[4] S. Chaudhuri and U. Dayal, "An overview of data warehousing and OLAP technology," SIGMOD Record, Vol.26, pp. 65-74, 1997.

[5] M. Garofalakis, J. Gehrke, and R. Rastogi., "Querying and Mining Data Streams: You Only Get One Look," In tutorial notes of the 28th International Conference on Very Large Data Bases, TUTORIAL SESSION: Tutorial 1, pp. 635-635, 2002.

[6] Jiawei Han, Yixin Chen, Guozhu Dong, Jian Pei, Benjamin W. Wah, Jianyong Wang, Y. Dora Cai, "Stream Cube: An Architecture for Multi-Dimensional Analysis of Data Streams," Distributed and Parallel Databases, Vol.18, No.2, pp. 173-197, 2005.

[7] George Colliat, "OLAP, relational and multidimensional database systems," ACM SIGMOD Record, Vol.25, No.3, pp. 64-69, 1995.

[8] Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang, "Multi-dimensional regression analysis of time-series data streams," Proceedings of the 28th international conference on VLDB, pp. 323-334, 2002.

[9] Jiawei Han, Jian Pei, Guozhu Dong, Ke Wang. "Efficient computation of iceberg cubes with complex measures," ACM SIGMOD Record, Vol.30, No.2, pp. 1-12, 2001.

[10] V. Harinarayan, A. Rajaraman, and J.D. Ullman, "Implementing data cubes efficiently," ACM SIGMOD Record, Vol.25, No.2, pp. 205-216, 1996.

[11] K. Beyer and R. Ramakrishnan, "Bottom-up computation of sparse and iceberg cubes," ACM SIGMOD Record, Vol.28, No.2, pp. 359-370, June 1999.

[12] Z. Shao, J. Han, and D. Xin, "MM-Cubing: Computing iceberg cubes by factorizing the lattice space," Proceedings of the 16th International Conference on Scientific and Statistical Database Management, pp. 213-222, June 2004.

[13] D. Xin, J. Han, X. Li, and B.W. Wah, "Star-cubing: Computing iceberg cubes by top-down and bottom-up integration," Proceedings of the 29th international conference on Very large data bases, Vol.29, pp. 476-487, 2003.

[14] Jiawei Han, Jian Pei, Guozhu Dong, Ke Wang, "Efficient Computation of Iceberg Cubes with Complex Measures," SIGMOD Conference, Vol.30, No.2, pp. 1-12, 2001.

[15] J. Gray, S.Chaudhuri, A.Bosworth, A.Layman, D. Reichart, M.Venkatrao, "Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals," Data Mining and Knowledge Discovery, Vol.1, pp. 29-53, 1997.

[16] M.E.J. NEWMAN, "Power laws, Pareto distributions and Zipf's law," Contemporary Physics, Vol.46, No.5, pp. 323-351, 2005.

서 대 홍

2005년 2월 경희대학교 전자 정보학과 학사. 2007년 8월 연세대학교 컴퓨터 과학과 석사. 2007년 8월~현재 (주)텔코웨어. 관심분야는 Data warehouse, OLAP, Data Mining

양 우 석

2007년 8월 연세대학교 컴퓨터과학과 학사. 2007년 9월~현재 연세대학교 컴퓨터과학과 석사 과정. 관심분야는 Data warehouse, OLAP, Data Mining

이 원 석

1985년 7월 Boston University 컴퓨터과학 공학학사. 1987년 7월 Purdue University 컴퓨터과학 공학석사. 1990년 7월 Purdue University 컴퓨터과학 공학박사. 2004년 3월~현재 연세대학교 컴퓨터과학과 정교수. 관심분야는 Database, Data Mining