제 26 회 한국정보처리학회 추계학술발표대회 논문집 제 13 권 제 2 호 (2006. 11)

데이터 스트림 정보 요약 기법

한상길, 이원석
연세대학교 컴퓨터과학과
e-mail : girihan@database.yonsei.ac.kr
leewo@database.yonsei.ac.kr

A Summarization Method for Data Streams

Sang-Gil Han, Won Suk Lee
Dept. of Computer Science, Yonsei University

요 약

최근까지 데이터웨어하우스와 OLAP 에 관한 연구와 더불어 데이터 큐브(data cube)는 많은 다차원 데이터웨어하우스에서 데이터 분석과 의사 결정 지원을 위해 빠르게 OLAP 연산을 처리하기 위한 중요한 역할을 수행해 왔다. 최근에는 빠른 속도로 생성됨과 동시에 지속적으로 발생되는 연속적인 데이터로 구성된 데이터 스트림이 네트워크 트래픽 모니터링, 증권, 날씨, 콜 센터 등과 같은 많은 분야에서 생성된다. 데이터 스트림은 무한의 집합이기 때문에 기존의 데이터 큐브 방법은 처리시간과 저장공간의 문제 때문에 데이터 스트림에 적용하기 어렵다. 이에 본 논문에서는 기존의 데이터 큐브와 같은 데이터의 요약 정보를 데이터 스트림 환경에서 제한된 메모리를 이용하여 관리할 수 있는 전위트리를 이용한 데이터 스트림 요약 기법을 제안하고, 실험을 통해 본 논문에서 제안한 방법이 데이터 스트림 환경에서 적응적으로 동작함을 증명한다.

1. 서론

최근까지 데이터웨어하우스(data warehouse)와 on-line analytical processing(OLAP) 시스템[1, 2]에 대한 연구개발과 더불어 많은 응용프로그램에 데이터웨어하우스와 데이터 큐브가 개발되고 적용되었다. 데이터 큐브(data cube)는 데이터분석과 지능적인 의사결정 지원을 위한 대부분의 데이터웨어하우스와 관계형 데이터베이스 시스템에서 중요한 역할을 수행해 왔다.

데이터 웨어하우스 시스템은 사용자에게 데이터 분석과 의사결정에 필요한 정보를 제공하며 OLAP 은 복잡한 분석질의에 대한 빠른 응답을 제공하기 위한 접근 방법으로, 판매나 마케팅, 자원관리를 위한 비즈니스 보고와 데이터 마이닝에 널리 사용된다. OLAP 시스템에 사용되는 다차원 데이터 모델에 있어서, 데이터 큐브는 빠른 OLAP 연산을 처리하기 위한 중요한 역할을 수행한다. 데이터 큐브는 차원(dimension)과 척도(measure)라는 두 요소에 의해 데이터 항목의 다양한 특성을 나타내는데 이용된다. 차원은 데이터 큐브의 축을 이루며 계층구조를 갖는다. 척도는 수치 값으로 데이터 큐브의 각 차원을 구성하는 항목들의 조합에 해당하는 데이터들의 대표값(*ex. COUNT, SUM* 등)을 나타낸다.

최근 많은 응용 도메인에서 생성되는 데이터들은 네트워크 트래픽 모니터링, 증권, 날씨, 콜 센터, 웹 클릭 데이터 등과 같이 빠른 속도로 생성됨과 동시에 지속적으로 발생되는 연속적인 데이터로 구성된 데이터 스트림 형태이다. 데이터 스트림 역시 다양한 차원과 척도 값을 갖기 때문에 데이터 큐브로 요약될 경우, 기존의 OLAP 시스템에서 수행했던 작업들을 데이터 스트림 환경에 빠르게 수행할 수 있다. 그러나 무한한 데이터 스트림의 정보를 요약하기 위해서는 많은 처리시간과 저장공간이 요구된다. 또한 데이터 스트림을 처리하기 위해서는 다음과 같은 제약 사항을 만족해야 한다[3]. 첫째 데이터 스트림을 분석하기 위해 각 데이터 항목을 최대 한번은 검사되어야 한다. 둘째 데이터 스트림을 분석하기 위해 사용되는 메모리는 유한하게 제한된다. 셋째 새로 생성되는 데이터 객체들은 가능한 빨리 처리되어야 한다. 마지막으로 데이터 스트림에 대한 최신 결과는 필요 시 즉시 사용가능 해야 한다. 위의 요구사항을 만족시키기 위해서 데이터 스트림의 처리 결과는 약간의 오차를

포함하지만 제한된 컴퓨팅 자원을 이용하여 근사적인 답을 제공한다. 따라서 무한한 데이터 스트림을 요약해서 저장하기 위해서는 한 번의 데이터 접근으로 빠르게 데이터 스트림의 정보를 요약하여 메모리 상에서 관리할 수 있는 데이터 구조와 알고리즘이 필요하다. 본 논문에서는 데이터 스트림 환경에서 제한된 메모리를 이용하여 데이터 스트림의 요약 정보를 생성하고 관리하는 방법을 제안하였다.

본 논문의 구성은 다음과 같다. 2 장에서는 관련 연구에 대해 알아보고, 3 장에서는 전위트리를 이용하여 데이터 스트림의 정보를 요약하는 방법에 대해 기술하고 4 장에서는 실험결과 및 성능에 대해 논의하고 5 장에서 결론을 기술한다.

2. 관련연구

데이터웨어하우스와 OLAP 에 관한 연구가 진행되면서 데이터 큐브(data cube)는 빠르게 OLAP 연산을 처리하기 위한 중요한 요소로써 인식되었다[2]. 실시간으로 다차원 분석을 수행하기 위해 데이터 큐브를 생성하는 방법은 다차원 집계 값을 미리 계산하고 저장하는 것이다. 이러한 데이터 큐브를 생성하기 위해 ROLAP-based multi-dimensional aggregate computation[4], BUC[5], H-Cubing[6] 등 다양한 알고리즘들이 제안되었다. 전체 데이터 큐브를 생성하기 위해서 많은 처리시간과 많은 수의 큐브 셀(cube cell)들을 저장하기 위한 저장공간이 요구된다. 따라서 사용자가 정의한 임계값 이상의 셀들만을 저장하는 빙산 큐브(iceberg cube) 알고리즘[5, 6] 이 제안되었다. 데이터 스트림의 크기는 무한대에 가까우므로 데이터 스트림의 모든 요약정보를 메모리에서 관리하기 어렵다. 따라서 본 논문에서도 빙산 큐브와 같이 사용자가 미리 정의한 빈발도 임계값을 넘는 데이터만을 메모리 상에서 관리하게 된다.

빙산 큐브와 같이 사용자가 미리 정의한 임계값을 넘는 항목집합을 찾는 것은 데이터 마이닝 분야에서 빈발항목집합을 찾는 것과 깊은 연관성이 있다. 데이터 스트림에서 의미 있는 지식을 탐색하기 위해 다양한 데이터 마이닝 방법들이 제안되었다. 이러한 방법들 중에서 *sticky sampling, Lossy Counting* 알고리즘[7] 및 *estDec* 알고리즘[8]은 데이터 스트림 환경에서의 빈발항목집합 탐색에 중점을 두고 있다. *Lossy Counting* 알고리즘[7]에서는 빈발항목집합 탐색 과정에서 메모리 사용량을 일정 범위로 한정하기 위해서 출현 빈도수 관리 대상 항목집합들을 보조 저장장치에 관리하며 메모리 상에는 일괄 연산 수행을 위한 버퍼만을 유지한다. 이 알고리즘에서는 보조기억 장치를 사용하기 때문에 수행시간이 늦어지는 단점이 있다. *estDec* 알고리즘[8]에서는 데이터 스트림에서 발생한 항목집합들 중에서 사전 정의된 지연 추가 및 전지 임계값 $S_{sig}(S_{sig} \leq S_{min})$ 이상의 지지도를 갖는 항목집합들을 가까운 미래에 빈발항목집합이 될 수 있는 중요 항목집합으로 간주하여 이 항목집합들만을 메모리 상에서 관리한다. 이를 통해 온라인 데이터 스트림에 대한 빈발항목집합 탐색 과정에서 메모리 사용량을 감소 시킨다. *estDec* 방법에서 출현 빈도수 관리 대상 항목집합들은 전위트리 래티스 구조로 메모리상에서 관리된다. 본 논문에서는 [8]에서 제안된 *estDec* 알고리즘을 기반으로 데이터 스트림 환경에서 제한된 메모리를 이용하여 데이터 큐브와 같은 정보를 저장하는 전위트리를 생성한다.

3. 데이터스트림 요약 기법

데이터웨어하우스나 OLAP 시스템에서 테이블 $R(d_1, d_2, \ldots, d_k, m)$과 같은 형식의 데이터를 입력으로 하여 큐브 연산자(cube operator)로 데이터 큐브를 생성하기 위한 질의는 아래와 같다.

```
[질의 1]
CREATE VIEW  cube_table  AS
SELECT       d1, d2, ... , dk, COUNT(*), SUM(m)
FROM         R
CUBE BY      d1, d2, ... , dk
```

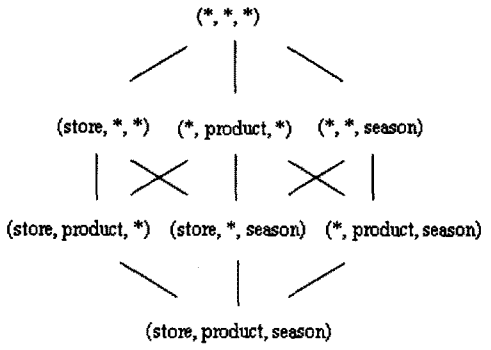
[질의 1]에 따라서 데이터 큐브를 생성할 때, 질의에 사용된 차원 수와 각 차원의 도메인 수에 따라서 데이터 큐브의 크기는 기하급수적으로 증가하게 된다. 또한 데이터 큐브를 생성하기 위해서 많은 처리시간과 저장 공간이 요구된다. 그러나 많은 응용 프로그램에서 사용자가 데이터 큐브의 모든 셀에 관심을 갖지 않는다. [질의 2]와 같이 큐브 연산자에 의해 생성된 데이터 큐브내의 셀의 빈발도가 사용자가 정의한 임계값 이상인 셀들이 주된 관심대상이 된다.

```
[질의 2]
CREATE VIEW  cube_table  AS
SELECT       d1, d2, ... , dk, COUNT(*)
FROM         R
CUBE BY      d1, d2, ... , dk
HAVING       COUNT(*) ≥ T
```

[질의 2]에 의해 생성되는 데이터 큐브는 임계값에 따라서 사용자가 관심을 갖는 빈발도가 사용자가 정의한 임계값(T) 이상인 셀들만을 포함하게 되므로 [질의 1]보다 훨씬 적은 수의 셀들을 저장하게 된다. 따라서 데이터 큐브를 생성하기 위해 필요한 처리시간과 저장공간을 효율적으로 관리할 수 있다. 만약 입력되는 데이터가 테이블 R 과 같은 형식의 데이터 스트림 형태라면 제한된 메모리와 컴퓨팅 파워를 가지고 빠르게 데이터 스트림의 정보를 데이터 큐브 형태로 요약해야 하므로 [질의 1]에 의해 생성되는 데이터 큐브는 스트림 환경에서 생성하기 어렵다. 본 논문에서 데이터 큐브의 셀의 빈발도를 기반으로 하여 [질의 2]에 의해 생성되는 데이터 큐브와 같이 사용자가 정의한 임계값 이상의 빈발도를 갖는 셀들의 요약 정보를 전위트리를 이용하여 관리하는 방법을 제안한다.

그림 1 은 기본 테이블 *R(store, product, season)*을 기반으로 데이터 큐브를 생성하였을 때, 데이터 큐브를 구성하는 큐보이드(*Cuboid*)를 나타낸다. 큐보이드는 주어진 기본 테이블의 차원들의 부분집합을 말하며,

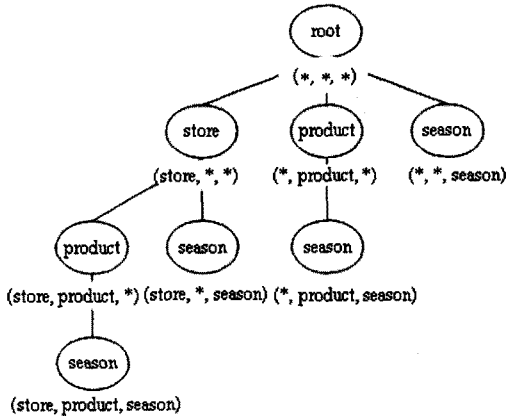
큐보이드는 차원들의 부분집합에 따라 집계함수 (*COUNT, SUM* 등)를 수행하여 생성된 요약된 정보를 갖는 셀들로 구성된다.



(그림 1) 데이터 큐브를 이루는 큐보이드 래티스

그림 1 과 같은 데이터 큐브를 생성하기 위해 사용된 각 차원의 도메인의 범위가 0<store≤ 1000<product ≤ 2000<season≤ 3000 와 같다면 그림 1 의 큐보이드 래티스는 그림 2 와 같은 전위트리로 매핑될 수 있다. 전위트리의 각 노드는 데이터 큐브의 큐보이드를 구성하는 셀과 1:1 로 매핑된다. 예를 들어, 하나의 차원에 의해 집계된 셀을 갖는 큐보이드 (*store*, *, *) (*, product, *) (*, *, season)의 한 셀은 각각 전위트리의 1 레벨의 노드(1-항목 노드)인 <*store*>, <product>, <season>과 매핑되고, 두 개의 차원에 의해 집계된 셀을 갖는 큐보이드 (store, product, *), (store, *, season), (*, product, season) 은 각각 전위트리의 <store → product>, <store→season>, <product→season>의 경로를 갖는 2 레벨 노드들 <product>, <season>, <season>으로 매핑된다. 마찬가지로 세 개의 차원 (store, product, season)에 의해 집계된 셀을 갖는 큐보이드는 전위트리에서 <store→product→season>의 경로를 갖는 3 레벨 노드인 <season>과 매핑된다.

본 논문에서는 데이터 큐브가 가지고 있는 요약 정보를 저장하는 전위트리를 생성하기 위해서 [10]에서 제안된 *estDec* 알고리즘을 사용한다. *estDec* 알고리즘은 데이터 스트림 마이닝을 위해 제안된 알고리즘으로써 지연추가와 전지 작업을 통하여 가까운 미래에 빈발항목집합이 될 가능성이 있는 항목집합만을 메모리에서 관리한다. *estDec* 알고리즘은 크게 추가 지연과 전지 작업 두 단계로 구분된다. 추가 지연은 데이터 스트림을 구성하는 트랜잭션의 항목들로 구성 가능한 한 모든 큐보이드들에 속하는 셀들의 빈발도가 가까운 미래에 빈발한 셀이 될 수 있을 때까지(S_{sig} 이상) 전위트리에 추가되는 작업이 지연된다. 즉 각 셀들의 지지도가 사전에 미리 정의한 추가 지연 임계값 이상 되는 시점에서 전위트리에 추가된다. 추가 지연 작업에서는 현재 트랜잭션에서 발생한 차원 값들로 생성할 수 있는 모든 큐보이드에 속하는 셀들의 빈발도가 추가 지연 임계값 이상인지 여부를 판단함과 동시에 이미 전위트리에서 관리되고 있는 빈발한 셀들의 경우 집계 값에 대한 갱신이 이루어 진다.



(그림 2) 전위트리로 표현한 큐보이드 래티스

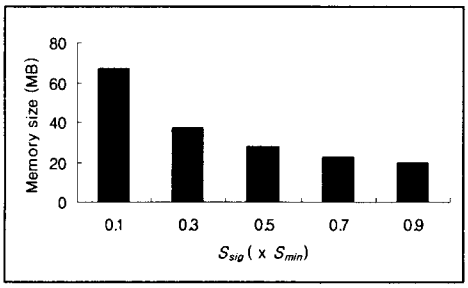
또한 효율적인 전위트리 내의 노드 전지 방법을 통해 이전 데이터 집합에서 빈발한 셀이었더라도 해당 셀의 지지도가 사전에 정의된 최소 지지도 임계값 미만으로 감소할 때 해당 셀을 앞으로 빈발한 셀이 될 가능성이 상대적으로 낮은 중요하지 않은 셀로 간주하여 빈발항목집합의 안티모노톤(*antimonotone*)성질에 의해 해당 셀을 표현하는 노드와 그 노드의 모든 자손 노드들을 전위트리에서 삭제한다. 이 과정을 전지 작업이라 한다. 이렇게 지연 추가와 전지 작업 과정을 통해 사용자가 관심을 갖는 빈발한 셀들로만 이루어진 데이터 큐브가 가지고 있는 요약 정보를 전위트리를 이용하여 표현할 수 있다.

4. 실험 및 성능 평가

제한된 메모리와 컴퓨팅 성능을 가지고 빠른 속도로 생성됨과 동시에 지속적으로 발생되는 데이터 스트림을 처리하기 위해서는 데이터 스트림의 생성 속도와 크기에 따라서 적응적으로 전위트리에서 관리하는 요약 정보의 양을 적절히 조절 할 수 있어야 한다.

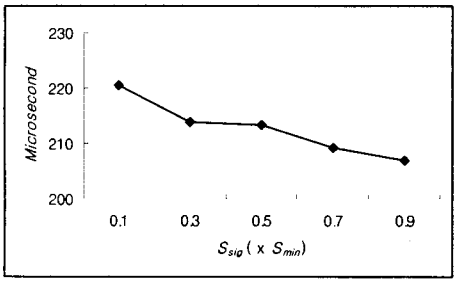
본 장에서는 Zipf 분산을 1.5 로 하여 생성한 데이터 집합을 가지고 본 논문에서 제안한 방법을 검증하였다. 각각의 데이터 집합은 1,000,000 개의 트랜잭션으로 구성되어 있으며, 실험에 따라서 데이터 집합의 차원은 2, 4, 6, 8 개로 구성되어 있고, 각 차원의 도메인 수는 20 이다. 실험에서 데이터 집합의 트랜잭션은 순서대로 하나씩 처리되었으며 전위트리는 각 *n-d* 셀들의 빈발횟수를 기반으로 생성되었다. 모든 실험은 1GB 의 메모리를 가진 2.8GHz 팬티엄 컴퓨터와 리눅스 7.3 환경에서 실험되었으며 모든 프로그램은 C 언어로 구현되었다.

그림 3 과 4 는 7 개의 차원을 가진 데이터 집합을 이용하여 지연 추가/전지 작업 임계값(S_{sig})에 따른 전위트리의 메모리 사용량과 평균 수행시간을 측정한 것이다. 최소 지지도(S_{min})는 0.0001 로 설정되었다.



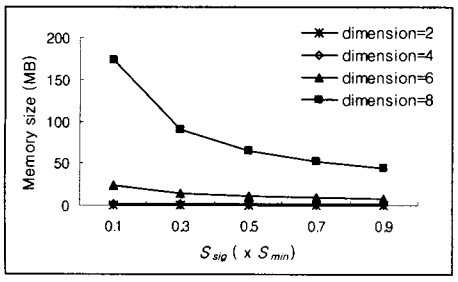
(그림 3) S_{sig} 변화에 따른 메모리 사용량

그림 3 에서 S_{sig} 값이 커질수록 전위트리의 크기가 작아지면 메모리 사용량이 감소한다. 그러나 S_{sig} 값이 증가할수록 사용되는 메모리의 양은 적어지지만 전위트리에서 관리하는 스트림 데이터의 요약정보의 정확도는 감소하게 된다



(그림 4) S_{sig} 변화에 따른 평균 처리시간

그림 4 는 트랜잭션당 평균 처리시간을 보인다. 지연 추가 임계값이 증가할수록 전위트리 내에서 관리하는 노드의 수가 감소하므로 처리시간이 감소한다. 그림 3 과 4 에서와 같이 S_{sig} 값에 따라서 전위트리에서 관리하는 데이터 스트림의 요약정보의 크기가 달라진다. S_{sig} 값이 작을수록 많은 요약정보를 저장하기 때문에 사용되는 메모리의 크기가 커지며 평균 처리시간은 증가하게 된다.



(그림 5) 차원 수에 따른 메모리 사용량

그림 5 는 차원의 수가 2, 4, 6, 8 인 데이터 집합을 이용하여 S_{sig} 값의 변화에 따른 메모리 사용량을 나타낸다. 데이터 집합의 차원 수가 증가할수록 생성 가능한 큐보이드의 셀의 수가 증가하기 때문에 메모리 사용량이 증가하지만 S_{sig} 값에 따라 메모리 사용량을 적응적으로 조절할 수 있다. 따라서, 데이터 처리를 위한 메모리 사용량이 제한되는 스트림환경에서 적응적으로 동작할 수 있음을 보인다.

5. 결론

데이터웨어하우스나 OLAP 시스템에서의 데이터 큐브는 데이터 분석이나 의사결정을 위해 빠른 온라인 처리를 지원하기 위한 중요한 요소이다. 그러나 최근 발생하는 데이터 스트림 환경에서 기존의 데이터 큐브를 적용하기는 처리시간과 저장공간의 제약이 따른다. 따라서 본 논문에서는 *estDec* 방법을 이용하여 사용자가 정의한 임계값에 따라서 데이터 스트림을 전위트리로 요약하는 방법을 제시하였다. 데이터 스트림의 요약된 정보를 관리하는 전위트리는 메모리 상에서 관리되기 때문에 사용자의 요구 따라 빠른 응답이 가능하다.

참고문헌

[1] S. Shaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. pp. 65-77, In Proc. of the 1996 ACM SIGMOD Intl. Conf. on Management of data.

[2] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab and sub-totals. pp.29-54, In Intl. conf. on Data Ming and Knowledge Discovery, 1997.

[3] 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, 2002.

[4] S. Agarwal, R. Agrawal, P. M. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan and S. Sarawagi. On the computation of multidimensional aggregates. pp. 506-521, In Proc. of the 22th Intl. Conf. on Very Large Data Bases, 1996.

[5] K. Beyer and R. Ramakrishnan. Buttom-up computation of sparse and iceberg cubes, pp. 359-370, In Proc. of the 1999 ACM SIGMOD Intl. Conf. on Management of data.

[6] J. Han, J. Pei, G. Dong, and K. Wang. Efficient computation of iceberg cubes with complex measures, pp. 1-12, In Proc. of the 2001 ACM SIGMOD Intl. Conf. on Management of data.

[7] G. S. Manku and R. Montwal. Approximate Frequency Counts over Data Streams. pp 346-357, In Proc. of the 28th Intl. Conf. on Very Large Data Bases.

[8] J. H. Jang and W. S. Lee. Finding recent frequent itemsets adaptively over online data streams. pp. 255-264, In proc. of the 2003 ACM SIGMOD Intl. Conf. on Management of data.

[9] R. C. Agarwal, C. C. Agrawal, and V.V.V. Prasad. Depth First Generation of Long Patterns. pp. 108-118, In Proc. of the 2000 ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining.