프랙탈을 이용한 시공간 데이타웨어하우스

최원익[O] 이석호
서울대학교 공과대학 전기컴퓨터공학부
styxii@db.snu.ac.kr[O], shlee@comp.snu.ac.kr

Spatio-Temporal Data Warehouses Using Fractals

Wonik Choi[0] Sukho Lee
School of Electrical Engineering and Computer Science,
Seoul National University

요 약

최근 시공간 데이타에 대한 OLAP연산 효율을 증가시키기 위한 여러 가지 연구들이 행하여지고 있다. 이들 연구의 대부분은 다중트리구조에 기반하고 있다. 다중트리구조는 공간차원을 색인하기 위한 하나의 R-tree와 시간차원을 색인하기 위한 다수의 B-tree로 이루어져 있다. 하지만, 이러한 다중트리구조는 높은 유지비용과 불충분한 질의 처리 효율로 인해 현실적으로 시공간 OLAP연산에 적용하기에는 어려운 점이 있다.

본 논문에서는 이러한 문제를 근본적으로 개선하기 위한 접근 방법으로서 힐버트큐브(Hilbert Cube, H-Cube)를 제안하고 있다. H-Cube는 집계질의(aggregation query) 처리 효율을 높이기 위해 힐버트 곡선을 이용하여 셀들에게 완전순서(total-order)를 부여하고 있으며, 아울러 전통적인 누적합(prefix-sum) 기법을 함께 적용하고 있다. H-Cube는 적응적이며, 완전순서화되어 있으며, 또한 누적합을 이용한 셀 기반의 색인구조이다. 본 논문에서는 H-Cube의 성능 평가를 위해서 다양한 실험을 하였으며, 그 결과로서 유지비용과 질의 처리 효율성면 모두에서 다중트리구조보다 높은 성능 향상이 있음을 보인다.

1. 서 론

시공간 데이타에 대한 OLAP연산의 중요성에도 불구하고 현재 진행된 연구는 한정적이다. 이 연구의 대부분은 다중트리구조(*multi-tree structure*)[1]를 기반으로 하고 있다. 다중트리구조는 공간차원을 색인하기 위해서 R*-tree[2](이하 R-tree로 칭함)를 이용한다. 이 다중트리구조에서 R-tree는 단 1개만이 필요하며 전집계(pre-aggregation)기법[3]이 적용되어 있다. 시간차원에 대해서는 R-tree의 엔트리 개수에 해당하는 수많은 B^+-tree들을 이용하여 각 엔트리의 시간정보를 유지하고 있다. 이러한 다중트리구조는 다음과 같은 단점이 있다.

첫째, 다중트리구조는 하나의 R-tree와 다수의 B-tree들의 조합으로 인해 그 유지비용이 너무 크다. R-tree가 갱신될 때 마다 그 엔트리의 개수에 해당하는 모든 B-tree들도 함께 갱신되어야 한다는 것을 의미하며 이는 시스템에 막대한 양의 오버헤드를 일으키게 된다.

둘째, 다중트리구조는 질의 수행과정에서 많은 디스크 접근을 필요로 하며 이러한 사실은 다중트리구조를 이용한 온라인(*on-line*) 질의 처리가 불가능하다는 것을 의미한다. R-tree는 데이타분할(data-partitioning) 방식의 구조적인 특성상 잠재적으로 많은 오버랩(overlap)이 발생할 수 있으며 그에 따라 이러한 오버랩들은 방문해야 할 B-tree의 개수를 급격히 증가시키게 된다. 더욱이 시간이 진행됨에 따라 즉, 더 많은 객체가 삽입되고 갱신되는 과정에서 R-tree의 성능이 저하된다는 사실은 이러한 문제점을 더욱 악화시키게 된다.

셋째, 다중트리구조는 막대한 양의 저장장치를 필요로 한다. 예를 들어 약 10,000개의 객체에 대해 1,000시간스탬프 동안 진화된 다중트리구조의 크기는 약 100MB[1]에 달한다.

본 논문에서는 이러한 문제점을 해결하기 위해 시공간 데이타웨어하우스를 위한 색인구조로서 H-Cube(Hilbert Cube)를 제안한다. H-Cube는 유지비용과 저장 공간의 최소화와 함께 집합질의(aggregation query), 예를 들어 "시간간격 q_t동안 질의 영역 q_s내에 존재한 객체의 총 개수를 검색하라"와 같은 질의를 효율적으로 처리하는 것을 목표로 하고 있다. H-Cube는 적응적이며 완전순서화된 그리고 누적합(prefix-sum)[4] 기법을 적용한 셀 기반의 색인 구조이다. H-Cube는 대상 시공간을 일정크기의 셀로 분할하고 시간 축으로 배열하여 큐브형태를 이루고 있다. H-Cube의 각 셀들은 그 중심좌표에 대한 힐버트 값이 부여되고 그 값 순서대로 디스크에 저장된다. 특히 이 셀들이 담고 있는 집계값(aggregated value)들은 누적합 형태로 저장되어 질의의 시간간격의 길이에 독립적으로 상수시간에 원하는 집계값을 계산할 수 있다. 이러한 H-Cube의 셀들은 기본적으로 형태가 변화하지 않으며 대신, 데이타가 편중되는 셀들에 한해서 적응적으로 정제하는 기법을 제공하고 있다.

H-Cube의 설계의 근간은 공간분할방법 측면에서는 사분트리(quadtree)의 장점을, 그리고 접근방법 측면에서는 격자화일(gridfile)의 장점만을 결합하고자 하는 것이며 아울러 이를 바탕으로 힐버트 곡선 및 전집계기법의 적용으로 한층 강화된 구조를 제안하고자 하는 것이다. 본 논문에서는 이 H-Cube가 다중트리구조 보다 더 시공간 데이타웨어하우스에 적합하다는 것을 보이고자 한다.

2. 관련연구

본 논문에서 제안하고 있는 기법과 같이 정적인 공간 차원에 초점을 두고 있는 기법인 a3DR-tree와 aRB-tree를 살펴본다. a3DR-tree는 aR-tree(Aggregation R-tree)[3]를 시간 축으로 확장함으로써 얻을 수 있다.

aR-tree는 R-tree에 전집계기법 즉, 각 노드의 MBR에 속한 객체들에 해당하는 집계값을 노드에 함께 가지고 있는 기법을 적용시켜 강화시킨 것이다. 그림 1은 aR-tree를 보여주고 있다. 그림 1의 aR-tree의 단말 노드 중 첫 번째 엔트리 <R_1, 6>은 시간스탬프 T_1에 구역 R_1에 6개의 객체가 있었다는 것을 의미한다. 하지만 aR-tree는 시간정보를 유지할 수 있는 기능을 가지고 있지 않으므로 이 기능을 제공하기 위해서는 시간 축으로

확장해야 하며, aR-tree를 시간 축으로 확장한 구조가 a3DR-tree(그림 2)이다. a3DR-tree는 집계값이 변할 때 마다 새로운 엔트리가 생성되면서 같은 MBR정보들이 매번 반복 저장된다. 이러한 단점으로 인해 색인의 크기가 급격히 늘어나고 질의처리 성능도 함께 저하된다.

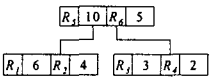

그림 1 aR-tree의 예

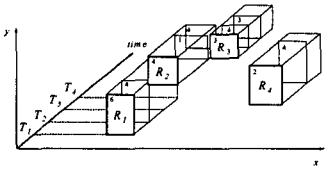

그림 2 a3DR-tree의 예

Papadias외 3인은 [1]에서 시공간 데이타웨어하우스를 위한 프레임웍을 최초로 제안하였다. 그들은 그림3과 같이 하나의 R-tree와 그 R-tree의 엔트리 개수만큼의 B-tree들로 이루어진 다중트리구조를 제안하였다.

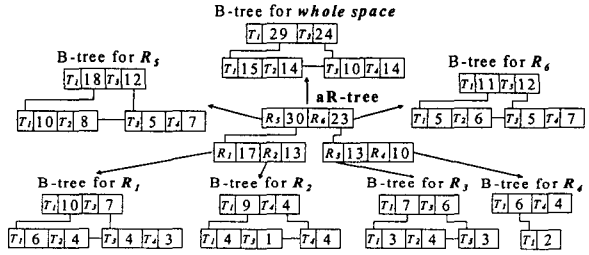

그림 3 aRB-tree의 예

aRB-tree(aggregate R- B- tree)로 불리는 이 트리는 aR-tree의 각 엔트리에 대한 시간 정보를 B-tree에 저장한다. aR-tree의 구역 R_5에 해당하는 B-tree의 단말 노드의 첫 번째 엔트리 <T_1, 10>는 시간스탬프 T_1에 10개의 객체가 존재한 것을 의미한다. aRB-tree의 단점은 질의 수행과정에서 너무 많은 디스크 접근을 필요로 하고 있다는 것과 aR-tree의 크기가 커질수록 즉, 엔트리의 개수가 많아질수록 방문해야 될 B-tree의 수는 그만큼 증가하게 된다는 것이다. 더욱이 이러한 B-tree들은 디스크 내의 산재된 페이지 내에 저장될 수밖에 없기 때문에 많은 탐색시간을 소요하게 되어 성능의 저하를 가져오게 된다. 또한 aRB-tree는 aR-tree의 엔트리 개수만큼의 B-tree가 필요하기 때문에 시간이 지남에 따라 막대한 양의 저장 공간을 필요로 하게 될 것은 자명한 일이다. 이러한 사실은 aRB-tree가 특정 저장 공간 또는 유지비용 제한 조건을 위반할 가능성이 높다는 것을 설명한다. 이러한 문제점들은 다중트리구조를 시공간 데이타웨어하우스에 적용하기에는 현실적으로 어렵다는 것을 말해주고 있으며, 새로운 접근 방법이 요구되고 있다는 것을 암시해 주고 있다.

3. 힐버트큐브(H-Cube)

3.1 H-Cube의 구조

이 절에서는 H-Cube를 소개하고 질의 처리 알고리즘을 다룬다. H-Cube는 공간 채움 곡선(space filling curve) 즉, 프랙탈(fractals)을 이용하고 있다. 프랙탈 중 힐버트 곡선(Hilbert curve)을 이용하여 셀들에게 선형적인 순서를 부여하며 이 순서에 맞추어 디스크에 셀들을 저장한다. 힐버트 곡선이 가장 이상적으로 클러스터(cluster)의 개수를 최소화 할 수 있다고 널리 알려져 있다.[5][6][7]. 이 문헌들은 모두 일관성 있게 다른 어떤 공간 채움 곡선보다도 힐버트 곡선이 가장 작은 개수의 클러스터를 만들어낸다고 밝혀내고 있다. 이러한 사실은 2차원 이상의 공간상에 존재하는 셀들에게 힐버트 값(Hilbert value)을 이용하여 일차원 순서를 부여하고 그 순서대로 디스크에 저장하게 되면 추가의 탐색시간을 최소화하면서 연속적인 디스크 페이지를 읽음으로써 셀들을 접근 할 수 있다는 것을 말해주고 있다.

H-Cube는 대상 공간을 N개의 고정크기의 셀들로 분할한다. 각 셀들은 힐버트 값을 부여받은 후 그 순서로 디스크에 저장된다. 이 셀들이 N개가 모인 것을 셀블럭(cellblock)이라 부르며 이러한 셀블럭들이 시간 축으로 배열되어 H-Cube를 형성하게 된다. 이러한 H-Cube의 구조가 그림 4에 나타나 있다. 힐버트 순서와 시간 순서가 함께 셀들에게 완전순서(total-order)를 부여함으로써 접근하고자하는 셀들은 상수 시간(constant time)에 접근할 수 있는 메카니즘을 제공하고 있다. 이러한 접근 방법은 집계질의 처리시 많은 성능 이득을 얻게 해주고 있다.

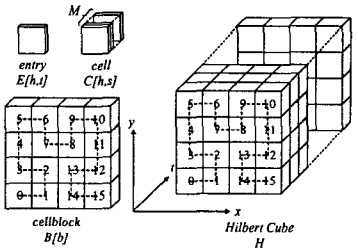

그림 4 H-Cube의 구조

3.2 적응적 셀 분할 기법

이동객체는 동적이다. 또한 그 분포는 예상 불가능하며 편중될 가능성이 크다. 이러한 데이타 편중은 셀의 크기를 증가시키고 따라서 셀 접근시 효율을 떨어뜨리게 된다. 이러한 문제를 다루기 위해 H-Cube는 셀의 객체 수에 따라 적응적으로 셀을 정제한다. 즉 만약 셀 내에 속한 객체의 수가 임계값 V를 초과하면 이 셀들은 모든 서브셀(sub-cell)들이 V개 이하의 객체를 갖게 될 때까지 재귀적으로 분할된다.

셀이 분할된 경우에는 그 엔트리에 누적합이 저장되어있는 것이 아니라 서브큐브(Subcube)라 불리는 구조체를 가리키는 포인터를 대신 담고 있다. 서브큐브는 분할된 셀들에 해당하는 각 엔트리들을 역시 힐버트 값 순서로 저장하고 있는 연속된 저장 공간을 말한다. 이 서브큐브 구조체는 질의윈도우가 분할된 셀들을 부분적으로 오버랩하고 있는 경우 해당 서브큐브 구조체를 접근하여 결과값을 보다 더 정제할 수 있는 기법을 제공하고 있다.

3.3 H-Cube의 질의 알고리즘

완전순서화된 셀 기반의 구조인 H-Cube는 집계 질의 처리과정 또한 효율적으로 이루어진다. 더욱이 누적합 형태로 구성된 H-Cube는 질의로 주어진 시간간격의 양 끝에 해당하는 엔트리만 접근해서 원하는 결과를 얻어 낼 수 있다. 즉 시간간격의 끝 엔트리의 누적합에서 시작 엔트리의 누적합을 빼면 그 셀의 SUM값을 얻을 수 있다.

H-Cube의 질의 처리 알고리즘은 기본적으로 두 단계로 이루어져 있다. 첫 번째 단계에서는 집계질의의 중간 결과(intermediate result)값이 계산된다. 먼저 q_s와 오버랩되는 모든 셀들을 찾아내 시간간격 q_t에 해당하는 값들을 계산한 후 모두 합하면 중간 결과값을 얻을 수 있다

그림 5를 통해 예를 들어보자. 임계값 V는 4이며 질의의 시간간격 q_t은 $[T_1, T_3]$이라고 가정한다. 먼저 질의윈도우 q_s와 겹쳐있는 셀들을 구한다. 이 예의 경우 q_s는 두 개의 셀 즉, C_1과 C_2과 오버랩하고 있다. 이 두 셀들의 해당 엔트리들을 이용하여 중간 결과값을 계산할 수 있다. C_1의 경우 T_3에 해당하는 값은 9이고 T_1에 해당하는 값은 0이라고 가정하면 9-0=9를 얻을 수 있다. 마찬 가지로 C_2는 13-0=13을 얻을 수 있다고 가정하면 중간결과값은 이 둘을 합한 22가 된다.

한편 첫 번째 단계에서 C_2는 그 값이 13/3=4.3으로 임계값 4보다 크기 때문에 C_2는 큐에 삽입된다. 이어서 두 번째 단계가 진행되는데 이 과정에서 서브큐브를 방문하면서 중간 결과값을 점진적으로 정제한다. 큐로부터 C_2을 삭제하여 엔트리들을 살펴보면 그림 5의 중간부분의 그림에서 나타나 있듯이 C_2가 4개의 셀로 분할 된 것을 알 수 있다. 이 4개의 셀 중 2번과 3번 셀 즉, 그림에서 음영처리 된 두개의 셀은 질의윈도우 q_s와 오버랩되어 있지 않기 때문에 2번과 3번 셀에 해당하는 값은 중간 결과값에서 감해주어야 한다. 따라서 22-3=19가 최종 결과값이 된다.

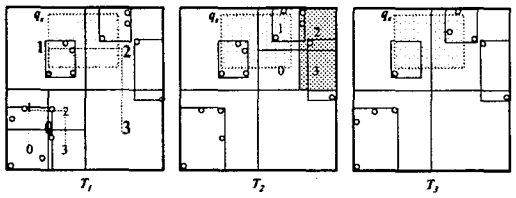

그림 5 H-Cube의 집계질의 처리 예

4. 성능 평가

데이타셋은 GSTD[8]를 이용하여 100,000개의 객체가 1,000시간스탬프동안 이동한 결과를 생성하였으며 $D[\alpha_{obj}, z]$로 나타낸다. 여기서 α_{obj}는 객체활동률(object activity)로서 각 시간스탬프마다 전체 객체 중 α_{obj}%만이 이동하였다는 의미를 가진다. z는 데이타의 편중도를 나타낸다. 작업부하는 $Q[q_s, q_t]$로 표기한다.

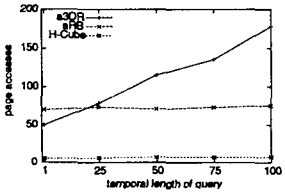

그림 6 $D[30,1], Q[0.25, q_t]$

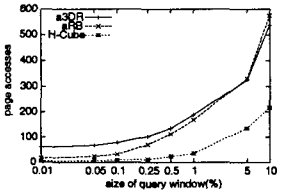

그림 7 $D[30,1], Q[q_s, 50]$

그림 6은 데이타셋 $D[30,1]$에 대해 q_t를 변화시켰을 때의 평균페이지 접근수를 보여주고 있다. H-Cube는 q_t에 영향을 받지 않으면서 월등한 성능을 보여주고 있으며 이는 H-Cube가 누적합형태를 유지하고 있기 때문에 나타나는 의미 있는 결과이다.

aRB-tree 역시 q_t에 영향을 받지 않고 있는데 이는 방문해야 될 B-tree의 개수가 거의 일정하고 더욱이 노드가 질의윈도에 완전히 포함된 경우 해당 B-tree의 루트만 방문해도 되기 때문이다. 반면에 a3DR-tree는 점점 증가되는 크기와 높이로 인해 q_t가 증가함에 따라 성능이 급격히 나빠지는 것을 알 수 있다. 그림 7은 q_s의 크기를 변화시켰을 때의 성능 평가 결과를 보여주고 있다. 이 경우도 역시 H-Cube가 월등한 성능 우위를 보여 주고 있다. aRB-tree의 경우 q_s에는 많은 영향을 받고 있는데 이는 q_s가 커질수록 방문해야 할 B-tree의 개수도 함께 증가하기 때문이다. 그 결과 5%보다 큰 질의윈도우에 대해서는 aRB-tree가 a3DR-tree보다 더 나쁜 성능을 보여주고 있다.

5. 결론

본 논문에서는 시공간 데이타웨어하우스를 위한 유지비용을 최소화하고 집계질의를 효율적으로 수행하는 새로운 접근 방법을 제시하고 있다. 이 접근 방법은 계층구조 방식(hierarchical access method)을 기반으로 한 구조가 아닌 셀 기반 구조로서의 최초의 시도이다. 본 논문에서 제안하고 있는 힐버트큐브(Hilbert Cube, H-Cube)는 적응적이며, 완전순서화되었으며 또한 누적합 기법을 적용한 셀 기반 색인 구조이다.

H-Cube는 힐버트 곡선을 이용하여 셀들에게 완전순서를 부여한 후 이 순서를 이용하여 디스크에 클러스터링(clustering)하고 있으며, 아울러 누적합(prefix-sum) 기법을 적용함으로써 집계질의(aggregation query) 효율성을 높이고 있다. 광범위한 실험을 통해 H-Cube는 저장 공간과 유지비용 그리고 질의처리의 효율성 모든 면에서 기존 접근 방법보다 월등히 우세하다는 결과를 보였다. 이 사실은 H-Cube가 시공간 데이타웨어하우스에 적합한 구조라는 것을 말해주고 있다.

추후 연구과제로서 H-Cube에 대해 선택도 분석(selectivity estimation) 및 근사오차(approximation error)등을 예측할 수 있는 이론적인 근거를 제시하고자 한다.

참고문헌

[1] Dimitris Papadias, Yufei Tao, Panos Kalnis, and Jun Zhang, "Indexing Spatio-Temporal Data Warehouses," In Proc. of the 8th Int'l Conf. on Data Eng., pp. 166-175, 2002.

[2] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger, "The R*-Tree: An efficient and Robust Access Method for Points and Rectangles," In Proc. of the 1990 ACM SIGMOD, pp. 322-331, 1990.

[3] Dimitris Papadias, Panos Kalnis, Jun Zhang, and Yufei Tao, "Efficient OLAP Operations in Spatial Data Warehouses," In Advances in Spatial and Temporal Databases, 7th Int'l Symp., SSTD 2001, 2001.

[4] Ching-Tien Ho, Rakesh Agrawal, Nimrod Megiddo, and Ramakrishnan Srikant, "Range Queries in OLAP Data Cubes," In Proc. of the 1997 ACM SIGMOD, pp. 73-88, 1997.

[5] H. V. Jagadish, "Linear Clustering of Objects with Multiple Attributes," In Proc. of the 1990 ACM SIGMOD, pp. 332-342, 1990.

[6] Christos Faloutsos and Shari Roseman, "Fractals for secondary key retrieval," In Proc. of the Eighth ACM SIGACT-SIGMOD-SIGART, pp. 247-252, 1989.

[7] Bongki Moon, H. V. Jagadish, Christos Faloutsos, and Joel H. Saltz, "Analysis of the Clustering Properties of the Hilbert Space-Filling Curve," TKDE, Vol.13, No.1, pp. 124-141, 2001.

[8] Yannis Theodoridis, JeRerson R. O. Silva, and Mario A. Nascimento, "On the Generation of Spatiotemporal Datasets," In Proc. of Int'l. Symp. on Spatial Databases, pp. 147-164, 1999.