# 집계큐브트리: 효율적인 범위-집계 질의의 수행을 위한 큐브트리

홍석진 °
서울대학교 전기.컴퓨터공학부
jinny@db.snu.ac.kr

송병호
상명대학교 소프트웨어학과
bhsong@pine.sangmyung.ac.kr

이석호
서울대학교 컴퓨터공학부
shlee@cse.snu.ac.kr

# Aggregate Cubetree : Cubetree for Efficient Execution of Range-Aggregate Query

Seokjin Hong °
School of Electrical Engineering and Computer Science,
Seoul National University

Byoungho Song
Dept. of Software Science,
Sangmyung University

Sukho Lee
School of Computer Science and Engineering,
Seoul National University

**요 약**

데이타웨어하우스 환경에서는 범위-집계 질의를 효율적으로 수행하기 위해 데이타큐브로 저장뷰를 구성한다. 큐브트리란 이러한 저장뷰를 R-Tree형태로 구성하는 기법으로, 효율적인 데이타 접근성을 제공하지만 범위-집계 질의의 수행을 위해 질의 범위 내의 모든 노드를 접근해야 하는 단점이 있다. 이 논문에서는 중간노드의 MBR에 자식 노드 레코드들의 집단함수 값을 저장하여, 질의 범위에 포함되는 중간노드의 경우 단말노드를 접근하지 않고 효율적으로 범위-집계 질의를 수행할 수 있는 집계큐브트리를 제안하였다. 집계큐브트리는 기존의 큐브트리에 비해, 항상 적은 수의 노드 접근으로 질의를 수행하며 질의 범위의 크기가 커질수록 좋은 성능을 보인다.

## 1. 서론

데이타웨어하우스[1]는 데이타의 분석을 목적으로 하는 데이타베이스로서, OLTP 데이타베이스로부터 데이타를 추출, 통합, 정제하여 분석에 알맞은 형태로 저장한다. 관계형 데이타베이스에서는 스타스키마(Star schema)[2]를 통해 분석에 적합한 형태로 데이타를 표현한다. 스타스키마의 중심이 되는 릴레이션은 사실테이블(fact table)이며, 사실테이블은 각 차원테이블(Dimension table)과 연관되는 차원애트리뷰트들과, 수치 값으로 구성되는 값애트리뷰트들로 이루어진다. 사실테이블에 대해 수행되는 질의는 차원애트리뷰트에 대한 그룹 연산과 값애트리뷰트에 대한 집계연산으로 구성된다. 많은 경우, 데이타웨어하우스의 특성상 사실테이블의 크기는 매우 크며, 차원애트리뷰트의 수가 많으므로 질의 수행시간이 매우 길어지게 된다.

이러한 문제를 해결하기 위해 많은 연구들이 진행되어 왔으며, 그 중 하나가 사실테이블에 대한 저장뷰(materialized view)[3]를 데이타큐브(Data Cube)[4]를 통해 구성하는 방법이다. 데이타큐브란 차원애트리뷰트로부터 가능한 모든 그룹에 대해 집단함수를 수행하는 연산이다. 데이타큐브의 결과를 저장뷰로 유지함으로써, 사실테이블에 대한 요약 정보를 통해 빠른 질의 수행이 가능하게 된다.

이러한 저장뷰는 릴레이션 형태로 저장되며 크기가 매우 크다. 따라서 저장뷰를 통해 질의를 효율적으로 수행하기 위해서는 저장뷰 자체에 인덱스를 구성하여야 하는 오버헤드가 있다. [7][8]에서는 이러한 저장뷰를 R-Tree[5] 형태의 큐브트리로 구성하여 이를 해결하고 있지만, 범위-집계(Range-Aggregate) 질의[6]를 수행하기 위해 해당 범위에 속하는 모든 노드를 접근해야 하는 단점이 있다.

이 논문에서는 큐브트리에 부가적인 정보를 저장하여 범위-집계 질의에 대해 기존 큐브트리보다 적은 수의 노드 접근으로 빠른 처리를 할 수 있는 집계큐브트리를 제안한다.

이 논문의 구성은 다음과 같다. 2절에서는 [7][8]에서 제안한 큐브트리에 대해 살펴본다. 3절에서는 큐브트리를 개선한 집계큐브트리를 소개하고, 집계큐브트리의 구조와, 알고리즘을 알아본다. 4절에서 실험 결과를 살펴보고 5절에서 결론을 맺는다.

## 2. 관련연구

n차원의 차원 애트리뷰트를 갖는 사실 테이블에 대해 데이타큐브를 구성하면 n차원부터 0차원까지의 초월평면(hyper plane)으로 이루어진 큐브가 생성된다. [7][8]의 기본 아이디어는 이러한 n차원부터 0차원까지의 초월평면을 n차원 공간으로 사상시켜 n차원 공간데이타로 구성한 뒤 이를 R-tree에 저장하겠다는 것이다. 이 때 n차원의 데이타를 제외한 나머지의 데이타로 구성된 큐브를 Dataless 큐브트리라 하며, Dataless 큐브트리를 여러 개의 n-1차원 트리로 구성한 것을 Reduced 큐브트리라 한다. 큐브를 이러한 Reduced 큐브트리 형태로 구성함

으로써 큐브의 크기를 감소시키며, 데이타의 응집성을 높인다. 또한 데이타의 응집성을 높이기 위한 다른 방법으로 정렬과 벌크로딩(bulk loading) 기법을 사용한다.

하지만 [7][8]에서 구성한 큐브트리는 범위-집계 질의를 수행하기 위해 범위 내의 모든 단말 노드를 접근해야 한다는 문제가 있으며, 이는 질의 영역의 크기가 클수록 질의의 성능을 떨어뜨리는 요인이 된다. 데이타웨어하우스의 특성상 넓은 범위에 대한 집계 질의가 빈번하게 발생하며, 극단적인 경우 거의 모든 단말노드를 접근해야 하는 경우도 발생한다. 이 경우 일반적인 큐브트리는 순차적으로 데이타를 탐색하는 것보다 오히려 안 좋은 성능을 내기도 한다.

## 3. 집계큐브트리

### 3.1 집계큐브트리

큐브트리에서 범위-집계 질의는 기존 R-tree의 범위 질의와 마찬가지의 방법으로 수행된다. R-tree에서의 범위 질의는 질의의 범위와 중첩되는 노드들을 찾고, 그 자식 노드에 대해 다시 범위 질의를 수행하는 재귀적인 구조로 수행된다. 최종적으로 선택된 단말 노드에 포함되어 있는 데이타가 결과로 출력되며, 범위-집계 질의의 경우에는 단말노드에 들어있는 값들에 대해 집단함수가 수행되어 결과로 출력된다. 따라서 큐브트리를 통해 범위-집계 질의를 수행하기 위해서는 질의의 범위에 포함되는 모든 단말 노드를 접근해야 한다.

하지만 집계큐브트리는 중간 노드의 MBR에 해당 MBR에 포함되는 자식 노드에 대한 포인터와 함께 MBR에 포함되는 자식 노드 레코드들의 집단 함수 값을 저장하여, 범위 질의에 완전히 포함되는 중간 노드에 대해서는 자식 노드를 순회할 필요없이 질의를 수행할 수 있도록 하였다.

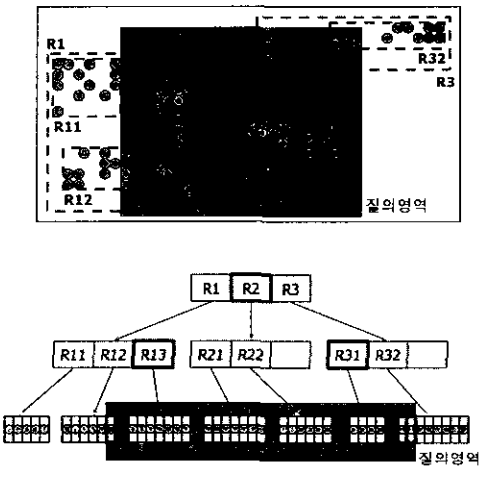


그림 1 집계큐브트리

그림 1은 집계큐브트리의 예이다. R1은 자식 노드 R11, R12, R13에 대한 집단함수 값을 저장하고 있으며, R11, R12, R13역시 자신의 자식 노드인 단말 노드의 데이타에 대한 집단함수 값을 저장하고 있다. 따라서 그림 1에서와 같은 범위-합 질의가 들어오면, 범위에 포함되는 중간노드의 MBR R2, R13, R31의 경우 자식 노드를 접근할 필요없이 해당 MBR이 갖고 있는 집단함수 값을 이용하게 되며, 나머지 단말 노드들만 접근하여 범위-합 질의를 수행하게 되므로 노드 접근 횟수가 줄어든다.

### 3.2 알고리즘

#### 3.2.1 삽입 알고리즘

그림 2는 집계큐브트리의 삽입 알고리즘이다. R-Tree의 삽입 알고리즘과 비슷하나, AdjustTree를 통해 부모 노드의 집단함수 값을 갱신하며, 필요한 경우 MBR을 재조정하게 된다.

```
Algorithm Insert(record)
Input:
        record: 삽입될 투플
Begin
        leafNode = chooseLeaf(record)
        If (leafNode.AddRecord(record) = SUCCESS) Then
          leafNode.UpdateMBR()
          AdjustTree(leafNode)
        Else
          SplitNode(leafNode, record)
        End If
End
```

그림 2 삽입 알고리즘

#### 3.2.2 범위-합 질의 알고리즘

그림 3은 범위-합 질의 알고리즘이다. 재귀적으로 수행되는 함수이며, 노드의 모든 레코드에 대해 MBR이 질의 영역에 포함되면 SUM을 결과에 더하고, MBR이 질의 영역과 중첩되면 레코드가 가리키는 자식노드에 대해 범위-합 질의를 수행하여 그 값을 결과에 더한다.

범위-카운트 질의나, 범위-평균, 범위-최대, 범위-최소 질의 등의 나머지 범위-집계 질의도 이와 비슷한 방법으로 수행할 수 있다.

```
Algorithm Range_SUM(Rect, node)
Input:
        Rect: 질의 영역
        node: 질의의 시작 노드
Output:
        result: 질의 영역 내에 포함되는 모든 단말 노드의 SUM
Begin
        result = 0
        For Each record in node
          If (Rect.inclusion(record.mbr) == TRUE) Then
            result = result + record.SUM
          Elseif (Rect.overlap(record.mbr) == TRUE) Then
            result = result + Range_SUM(Rect, record.node)
          End If
        End For

        return result;
End
```

그림 3 범위-합 질의 알고리즘

## 4. 실험

데이타큐브를 큐브트리와 집계큐브트리로 구성하고 각각에 대해 범위-합 질의를 수행하여, 발생하는 노드의 접근 횟수를 측정하고 이를 비교하였다. 2차원부터 5차원까지의 데이타 1만개, 10만개, 100만개를 임의로 생성하여 실험하였다. 트리 노드의 크기는 4kbyte이며, 노드 내에 들어가는 레코드의 최대 개수는 차원에 따라 달라진다. 2차원 데이타의 경우 한 노드 안에 들어가는 레코드의 최대 개수는 340개 이며, 5차원의 경우 169개이다. 실험에 사용된 기계는 Ultra Sparc II 333MHz이며, OS는 Solaris 7이다.

### 4.1 질의 영역의 크기에 따른 노드의 접근 횟수 비교

구성된 각각의 트리에 대해 질의 영역의 크기를 변화시켜 가면서 질의를 수행하여 노드의 접근 횟수를 측정하였다. 질의 영역의 크기는 각 차원의 전체 도메인 크기에 대한, 질의 영역 각 차원의 크기의 비율이며, 0.1부터 0.9까지의 값을 갖는다. 각 질의는 100번씩 수행하였으며, 그 평균값을 사용하였다. 그림 4는 2차원의 데이타 100만개에 대해 질의영역의 크기를 0.1부터 0.9까지 변화시켜가면서 노드의 접근 횟수를 측정한 그림이다.

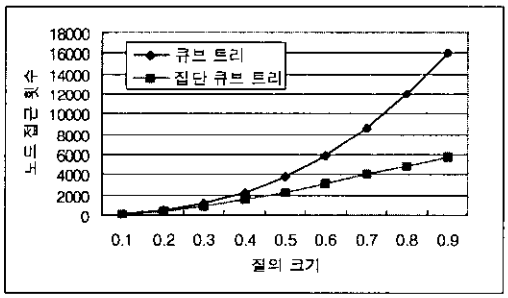


그림 4 질의 크기에 따른 노드 접근 횟수

전반적으로 집계큐브트리가 큐브트리에 비해 노드 접근 횟수가 작은 것을 알 수 있다. 큐브트리는 질의 크기의 증가에 따라 노드 접근 횟수가 급격히 증가하는 반면, 질의 영역에 포함되는 단말노드를 전부 접근하지 않아도 되는 집계큐브트리의 경우는 거의 선형적인 증가를 하고 있다. 질의 영역이 크기가 클수록 성능차이가 두드러지는 것을 알 수 있다.

### 4.2 레코드 수에 따른 노드의 접근 횟수 비교

레코드 수를 변화시켜 가면서 노드 접근 횟수를 조사하였다. 2차원 데이타에 대해 질의 크기를 0.5로 하여 실험하였으며, 역시 100개의 질의를 수행하여 평균 노드 접근 수를 측정하였다. 그림 5는 큐브트리의 평균 노드 접근 수를 1로 하였을 때, 집계큐브트리의 평균 노드 접근 수를 보여주고 있다.

레코드 수가 증가함에 따라 R-Tree에 비해 접근해야할 노드 수가 크게 줄고 있는 것을 알 수 있으며, 100만개 데이타의 경우 큐브트리에 비해 60%의 노드 접근만으로 질의를 수행할 수 있다.

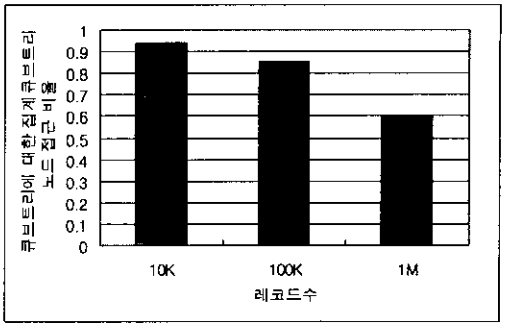


그림 5 레코드 수에 따른 노드 접근 횟수

## 5. 결론

이 논문에서는 데이타큐브를 저장뷰로 유지하는 방법으로 집계큐브트리를 제안하였다. 집계큐브트리는 중간 노드의 MBR에 해당 MBR에 포함되는 자식 노드의 집단함수 값을 저장하여, 질의 범위에 포함되는 모든 단말노드를 접근하지 않고도 효율적으로 범위-집계 질의를 수행할 수 있다. 또한 실험 결과에서 알 수 있듯이 어떠한 경우에도 기존의 큐브트리에 비해 적은 수의 노드 접근으로도 질의가 가능하며, 질의 범위의 크기가 커질수록 좋은 성능을 보인다.

## 6. 참고 문헌

[1] S. Chaudhuri and U. Dayal, "An overview of data warehousing and OLAP technology", ACM SIGMOD Record 26(1), 1997

[2] R. Kimball. "The Data Warehouse Toolkit", John Wiley & Sons, 1996

[3] N. Roussopoulos, "Materialized Views and Data Warehouses", SIGMOD Record, 27(1), p21-26, March 1998.

[4] J. Gray, A. Bosworth, A. Layman, and H.Piramish, "Data Cube: A Relational Aggregation Operator Generalizing, Group-By, Crosstab, and Sub-Totals", Int. Conference on Data Engineering p152-159, 1996

[5] A. Guttman. "R-Trees: A Dynamic Index Structure for Spatial Searching", ACM SIGMOD 1984

[6] C.Ho, R.Agrawal, N.Megiddo, R.Srikant. Range Queries in OLAP Data Cubes , ACM SIGMOD 1997

[7] Nick Roussopoulos, Yannis Kotdis, Mena Roussopoulos, "Cube Tree: Organization of and Bulk Incremental Update on the Data Cube, ACM SIGMOD, 1997

[8] Yannis Kotdis, Nick Roussopoulos, "An Alternative Storage Organization for ROLAP Aggreagte Views Based on Cubetrees", ACM SIGMOD, 1998