계층 최대 R-트리를 이용한 범위 상위-k 질의의 효율적인 수행

(Efficient Execution of Range Top-k Queries using a Hierarchical Max R-Tree)

홍 석 진 † 이 상 준 † 이 석 호 ††
(Seokijn Hong) (Sangjun Lee) (Sukho Lee)

요 약 범위 상위-k 질의는 질의 범위 내의 다차원 데이타 중 값 애트리뷰트를 기준으로 상위 k개의 레코드를 반환하는 질의로 공간 데이타베이스와 데이타 웨어하우스에서 분석을 위해 많이 사용되는 유용한 질의 형태이다. 이 논문에서는 계층 최대 R-트리의 선택적인 탐색을 통해 범위 상위-k 질의를 효과적으로 수행하는 기법을 제시한다. 이 기법은 단말 노드의 일부만을 접근하여 질의를 수행할 수 있으며, 질의 범위의 크기에 관계없이 거의 일정한 성능을 보인다. 또한 이 기법은 우선순위 큐를 효율적으로 관리함으로써 큐의 유지비용을 최소화 하며, 기존 R-트리와 같은 팬아웃을 보장할 수 있다.

키워드 : 범위 상위-k 질의, 계층 최대 R-트리

Abstract A range top-k query returns top k records in order of a measure attribute within a specified region on multi-dimensional data, and it is a powerful tool for analysis in spatial databases and data warehouse environments. In this paper, we propose an algorithm for answering the query via selective traverse of a Hierarchical Max R-Tree(HMR-tree). It is possible to execute the query by accessing only a small part of the leaf nodes in the query region, and the query performance is nearly constant regardless of the size of the query region. The algorithm manages the priority queue efficiently to reduce cost of handling the queue and the proposed HMR-tree can guarantee the same fan-out as the original R-tree.

Key words : Range Top-k Query, Hierarchical Max R-Tree

1. 서 론

범위 질의란 다차원 데이타 중 특정 범위 내에 있는 데이타를 반환하는 것으로, 공간 데이타베이스[1]에서 많이 사용되는 질의의 형태이다. 범위 질의의 경우 질의를 수행하기 위해 범위 내의 모든 레코드를 접근해야 하므로, 데이타의 수가 많아지고 질의 범위의 크기가 커지면 수행시간이 크게 증가하게 된다. 일반적으로 사람들은 오랜 수행시간을 통해 전체 결과를 얻는 것 보다는 짧은 수행시간을 통해 결과 중 상위 일부를 원하는 경우가 많다. 이렇게 범위 질의의 결과 중 데이타의 값을 기준으로 상위-k개를 반환하는 질의를 범위 상위-k 질의라고 한다.

이러한 범위 상위-k 질의는 다양한 형태로 적용될 수 있다. 가장 간단한 예로, 각 상점에 대한 정보로 상점의 위치와 상점의 매출량을 저장하고 있는 공간 데이타베이스에 대해 "특정 지역의 상점 중 매출액 기준 상위 100개의 상점을 구하라"와 같은 질의가 이에 속한다. 어떤 지역의 기온과 강수량 데이타나, 교통량 데이타 등도 공간 데이타베이스 상의 범위 상위-k 질의의 좋은 대상이 될 수 있다. 범위 상위-k 질의는 데이타 웨어하우스[2]에도 적용될 수 있다. YEAR, STATE, CUSTOMER_AGE를 차원 애트리뷰트, SALES_AMOUNT를 값 애트리뷰트로 하는 사실 테이블이 있다고 하면, "2000년에서 2002년 사이에 모든 주에서 발생한 20대의 구매 기록 중 가장 매출이 높은 50개의 (YEAR, STATE, CUSTOMER_AGE)와 해당 SALES_AMOUNT를 구하라"와 같은 질의가 그 예가 될 수 있다.

· 본 연구는 2003년도 두뇌한국21사업과, 정보통신부의 대학 IT연구센터(ITRC) 지원을 받아 수행되었습니다.
† 학생회원 : 서울대학교 전기컴퓨터공학부
jinny@db.snu.ac.kr
freude@db.snu.ac.kr
†† 종신회원 : 서울대학교 전기컴퓨터공학부 교수
shlee@cse.snu.ac.kr
논문접수 : 2003년 7월 21일
심사완료 : 2003년 11월 3일

범위 상위-k 질의를 수행하는 일반적인 방법은 질의 범위 내의 모든 데이타를 검색하여 값에 의해 정렬한 후, 그 중 상위-k개를 반환하는 것이다. 하지만 이 방법은 범위 내의 모든 데이타를 접근해야 하므로, 데이타의 개수가 많아지고 질의 범위가 커짐에 따라서 질의 성능이 급격히 떨어지는 문제가 있다. 따라서 질의 범위 내의 모든 데이타를 접근하지 않고 범위 상위-k 질의를 수행하는 방법이 필요하다. 범위 상위-k 질의와 비슷한 질의로 특정 범위 내에 있는 데이타에 대한 집계 함수를 수행하여 단일 집계 값을 구하는 범위 집계 질의가 있다. 범위 집계 질의의 경우 미리 계산해 놓은 집계값을 통해 효율적으로 질의를 수행하는 기법이 많이 연구되었다. 하지만 범위 상위-k 질의의 경우, 질의의 범위 뿐 아니라 질의 결과의 크기까지 질의마다 다르며, 질의 결과가 단일 값이 아니라 상위-k개의 데이타라는 점 등의 이유 때문에, 질의 결과를 미리 계산해 놓는 방법은 적용할 수 없으며, 다른 식의 접근이 필요하다.

이 논문에서는 질의 영역내의 데이타 중 일부만을 접근하여 범위 상위-k질의를 효율적으로 수행하는 알고리즘을 제시한다. 이 알고리즘은 계층 최대 R-트리(Hierarchical Max R-Tree, HMR-tree)라는 트리구조를 사용한다. 계층 최대 R-트리는 기존 R-트리[3,4]를 변형한 것으로 트리의 각 노드에 하위 노드에 대한 최대값을 저장하는 구조이다. 알고리즘에서는 계층 최대 R-트리의 각 중간 노드에 저장된 최대값을 기준으로 우선순위 큐를 통해 큰 값이 저장되어 있는 단말 노드부터 접근하여, 상위-k개의 결과를 점증적으로 찾아낸다. 질의 영역 내의 모든 데이타를 접근하지 않고 질의 수행이 가능하며, 점증적으로 질의 결과를 반환할 수 있으므로 파이프라인 형태의 질의 수행이 가능하다. 또한 이 논문에서는 우선순위 큐를 효율적으로 관리하는 기법과, 트리노드를 효율적으로 구성하여 트리의 팬아웃을 유지하는 기법을 제안한다.

이 논문의 구성은 다음과 같다. 2절에서는 다차원 데이타의 저장 방법 및 다차원 데이타에 대한 범위 집계 질의의 연구에 대해 살펴보고 범위 상위-k 질의와의 차이점을 알아본다. 3절에서 범위 상위-k 질의를 수행하기 위한 자료구조인 계층 최대 R-트리와 범위 상위-k 질의 알고리즘을 소개한 후, 4절에서는 우선순위 큐와 트리 노드의 효율적인 구성에 대해 살펴본다. 5절에서 실험 결과를 분석하고, 6절에서 결론을 맺는다.

2. 관련 연구

다차원 데이타를 저장하는 방법은 배열을 사용하는 방법과 트리를 사용하는 방법으로 크게 나누어 볼 수 있다. 배열을 사용하는 방법에서는 다차원 공간을 큰 하나의 다차원 배열로 구성한 다음, 다차원 공간상의 각 데이타를 해당 위치의 셀에 저장한다. 다차원 배열의 각 셀은 1차원 배열의 각 셀로 매핑할 수 있으며, 다차원 데이타의 좌표에 대해 해당하는 셀의 오프셋을 바로 구할 수 있으므로, 원하는 다차원 데이타를 한 번에 접근할 수 있다는 장점이 있다. 반면 데이타가 성긴(sparse) 경우 공간의 낭비가 심하다는 큰 단점이 있다. 배열은 비연속적인 성질 때문에 공간내의 개별적인 데이타를 저장하는 용도보다는 공간내의 데이타에 대한 요약 정보를 나타내는데 많이 사용된다. 공간 데이타베이스에서 공간을 격자 형태로 분할하여, 격자의 각 셀에 영역내의 데이타에 대한 요약 정보를 저장하거나, 비연속적인 도메인으로 구성되는 다차원 OLAP 환경에서 데이타 큐브[5,6]를 구성하는 데 배열 구조가 사용되고 있다.

트리를 사용하는 방법에서는 다차원 데이타를 릴레이션 형태로 저장한 후 다차원 트리 형태의 인덱스 구조를 통해 각 다차원 데이타를 효율적으로 접근할 수 있도록 한다. 다차원 데이타는 일반적으로 성기기 때문에 트리 구조를 사용하면 공간을 효율적으로 사용할 수 있으며, 점이나 선, 면, 공간 등으로 이루어진 다양한 형태의 개별적인 다차원 데이타를 직접 저장할 수 있다. 트리 구조는 공간 데이타베이스에서 개별적인 데이타를 저장하거나, 성긴 데이타를 처리하는 OLAP에서 주로 사용된다.

다차원 데이타의 범위 집계 질의 대해서는 많은 연구가 진행되어왔다. 배열 기반의 범위 집계 질의는 누적합(prefix-sum) 기법[7]을 시작으로 하여 다양한 변형[8-11]들이 연구되었으며, 주로 질의 범위 내의 모든 셀을 접근하지 않고 범위 내의 셀에 대한 집계값을 구하는 방법에 초점을 맞추고 있다. 트리 기반의 범위 집계 질의에서는 트리의 중간 노드에 집계값을 저장하여 트리의 단말노드를 접근하지 않고 범위 내의 데이타에 대한 집계값을 구하는 여러 기법[12,13]들이 연구되었다. 이러한 트리 구조는 트리 노드의 각 엔트리마다 부가적인 정보를 저장함으로써 엔트리의 크기가 커져, 트리의 팬아웃이 줄어드는 문제점을 안고 있다.

범위 상위-k 질의는 기존의 범위 집계 질의와는 달리 다음과 같은 다른 특징이 있다. (a) 기존 범위 질의에서는 범위 내의 데이타에 대한 하나의 집계값을 반환하는데 비해, 범위 상위-k 질의는 데이타가 갖는 값에 대해 상위-k개를 순서대로 반환한다. (b) 또한 상위-k개 데이타의 값 뿐 아니라 실제 데이타까지 함께 반환한다.

이러한 특징 때문에 범위 상위-k 질의에서는 질의 결과를 미리 계산해서 저장하기 어려우며, 따라서 질의 결과를 구하기 위해서는 반드시 실제 데이타 노드에 접근해야 한다. 하지만 일반 범위 질의와는 달리 질의 범위

내의 모든 단말 노드 중 일부 노드만 상위-k개의 결과를 포함하기 때문에, 결과를 포함하는 노드만을 선택적으로 접근할 수 있다면 노드 접근 수를 최소화하여 질의를 수행할 수 있을 것이다. 이를 위해 이 논문에서는 각 노드마다 자식 노드에 대한 최대값을 유지하며, 우선순위 큐를 이용하여 큰 최대값을 갖는 노드부터 선택적으로 접근하는 기법을 제시한다. 또한 최대값을 노드 자신에 저장함으로써 기존 R-트리와 같은 팬아웃을 보장할 수 있도록 하였다

3. 범위 상위-k 질의를 위한 자료구조와 알고리즘

3.1 자료 구조

범위 상위-k 질의의 대상이 되는 데이타 D는 ($location$, $value$, $pointer$)의 형태의 다차원 점 데이타로 구성된다. $location$은 다차원 데이타의 위치를 나타내고, $value$는 순위의 기준이 되는 다차원 데이타의 값을 나타낸다. $pointer$는 다차원 레코드의 부가 정보를 카리키는 포인터 값으로 다차원 데이타가 $value$만을 유지하는 경우에는 생략이 가능하다. 모든 점 데이타는 $location \in R_{space}$, $value \in V$를 만족한다. R_{space}는 점 데이타를 이루는 도메인으로, $R_{space} \subseteq R^d$를 만족하는 d차원 공간의 일부이다. V는 $value$를 구성하는 도메인이다. 사용자의 질의는 (R_Q, k)으로 구성된다. R_Q는 질의의 대상이 되는 영역을 나타내며, $R_Q \subseteq R_{space}$, $k \in \{i | i=1,2,3,\ldots\}$의 조건을 만족한다. R_Q 내에 포함된 데이타 중 $value$가 큰 순서대로 상위 k개가 결과로 반환된다.

이 논문에서는 계층 최대 R-트리(Hierarchical Max R-tree, HMR-트리)라는 자료구조를 사용한다. 계층 최대 R-트리는 R-트리를 기반으로 하며, R-트리의 각 노드에 자식 노드에 대한 최대값을 계층적으로 저장하는 구조이다. 계층 최대 R-트리는 최대값을 저장하는 방법에 따라 여러 가지 형태로 구성할 수 있다. 엔트리 내 저장(*In Entry*) 기법은 자식 노드의 최대값을 해당하는 부모 노드의 각 엔트리에 저장하는 방법이고, 노드 내 저장(*In Node*) 기법은 노드의 최대값을 노드 자신에 저장하는 방법이다. 정렬된 노드 내 저장(*In Sorted Node*) 기법은 노드 내 저장 기법의 각 노드의 엔트리를 최대값이 큰 순서대로 정렬하는 기법이다. 3절에서는 엔트리 내 저장 기법을 중심으로 알고리즘을 설명하며, 노드 내 저장 기법과 정렬된 노드 내 저장 기법의 자세한 설명은 4절에서 하기로 한다.

그림 1은 엔트리 내 저장 기법으로 구성한 2차원 계층 최대 R-트리의 예이다. 엔트리 E_{11}은 자식 노드 N_{11}의 최대값인 79를 저장하며, E_{12}는 N_{12}의 최대값인 85를 저장한다. E_1은 자식 노드 N_1의 엔트리 E_{11}, E_{12}의 최대값인 85를 저장한다.

3.2 범위 상위-k 질의 알고리즘

이 절에서는 계층 최대 R-트리를 사용하여 범위 상위-k 질의를 수행하는 효율적인 알고리즘을 제시한다. 알고리즘은 우선순위 큐를 통해서 수행된다. 노드의 각 엔트리가 우선순위 큐의 아이템으로 사용된다. 아이템은 ($entry$, $score$)의 형태로 구성된다. 아이템의 $score$가 클수록 큐에서 높은 우선순위를 갖게 된다. 중간 노드의 경우에는 각 엔트리의 최대값이 아이템의 $score$로 사용되며, 단말 노드에서는 실제 데이타의 $value$를 아이템의 $score$로 사용하게 된다.

알고리즘은 크게 초기화(initialize)와 드릴다운(drill down)의 두 과정으로 구성된다. 초기화 과정에서는 우선순위 큐를 초기화한다. 질의 영역에 포함되거나 질의 영역에 겹치는 루트 노드 엔트리를 우선순위 큐에 삽입한다. 두 번째 드릴다운 과정에서는 우선순위 큐에서 최상위 아이템을 하나 뽑아 해당되는 하위 노드 엔트리 중 질의영역에 포함되거나 겹치는 엔트리들을 우선순위 큐로 다시 넣는 과정을 상위 k개의 결과가 나올 때까지 반복한다. 드릴다운 과정은 다음과 같이 구성된다.

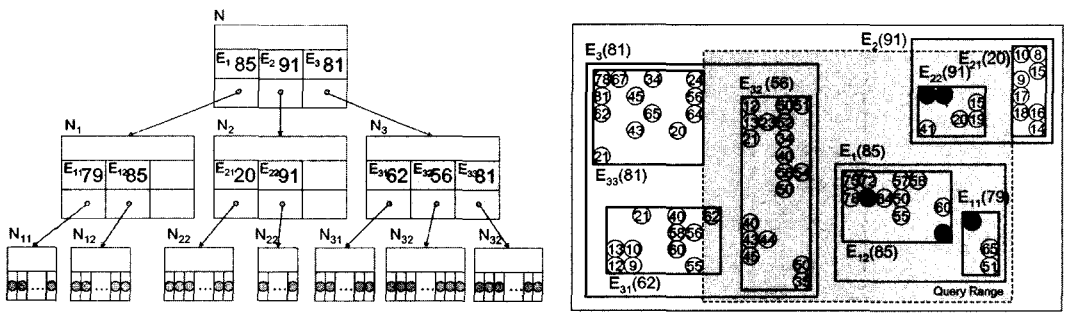


그림 1 엔트리 내 저장 기법으로 구성된 계층 최대 R-트리의 예

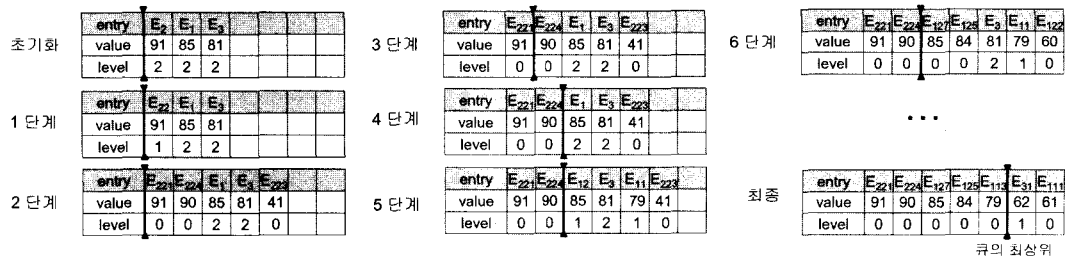


그림 2 범위 상위-k 질의의 수행 과정

- 우선순위 큐에서 아이템 하나를 뽑는다.
- 해당 아이템의 레벨이 0이면 결과로 출력한다.
- 레벨이 1이상이면, 해당하는 하위 노드의 모든 아이템을 읽어서 질의 영역에 포함되거나 겹치는 아이템들을 우선순위 큐에 삽입한다.
- 위의 과정을 상위-k개의 결과가 나올 때까지 반복한다.

그림 2는 그림 1의 계층 최대 R-트리 위에서 범위 상위-5 질의를 수행하는 예이다. 각 단계는 질의 수행 중의 큐의 상태를 나타낸다. 초기화 과정에서는, 루트 노드의 엔트리인 E_1, E_2, E_3가 질의 영역과 모두 겹치므로 큐에 삽입된다. 1 단계부터는 드릴다운 과정을 나타낸다. 1단계에서는 최상위 엔트리인 E_2를 큐에서 뽑은 후, 그 자식 노드인 N_2를 접근하여 그 엔트리 중, 질의 영역에 포함되거나 겹치는 엔트리인 E_{22}를 큐에 넣는다. 2 단계에서는 역시 최 상위 엔트리인 E_{22}를 큐에서 뽑은 후 자식 노드 N_{22}의 엔트리 중 E_{221}, E_{224}, E_{226}를 큐에 삽입한다. 3 단계에서는 최 상위 엔트리인 E_{221}을 뽑게 되며, 이는 단말 노드의 엔트리이므로 결과로 바로 반환한다. 4 단계 역시 뽑혀진 최 상위 엔트리 E_{224}가 단말 노드의 엔트리이므로 결과로 반환하게 된다. 5 단계에서는 최 상위 엔트리 E_1을 뽑고 자식 노드 엔트리 E_{11}, E_{12}를 큐에 다시 삽입한다. 이러한 과정을 상위 5개의 결과가 반환될 때까지 반복하여 질의를 수행한다. 알고리즘 1은 범위 상위-k 질의를 수행하는 알고리즘이다.

```
1  Algorithm Range_Topk(HMRT, range, k)
2  Input:
3    HMRT: 계층 최대 R-트리
4    range: 질의 영역
5    k: 상위 결과의 개수
6  Output:
7    result: 질의 영역 내에 포함되는 다차원 데이타 중 상위-k개
8  Begin
9    // step 1: initialize
10   For each entry in HMRT.root
11     If (range.overlap(entry.mbr) == TRUE) Then
12       queue.insert(entry);
13     End If
14   End For
15   // step 2: drill down
16   While (|result| < k)
17     entry = queue.pop();
18     If (entry.level == 0) Then
19       output entry
20     Else
21       For each child_entry in entry.child_node
22         If (range.overlap(child_entry.mbr) == TRUE) Then
23           queue.insert(child_entry);
24         End If
25       End For
26     End If
27   End While
28 End
```

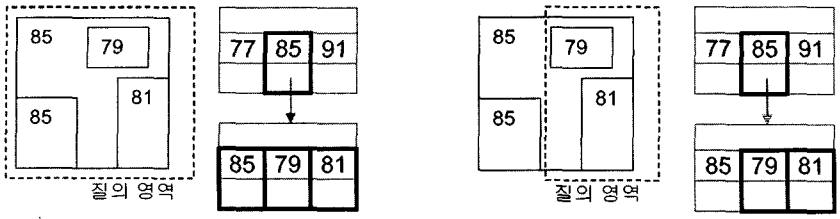
알고리즘 1 범위 상위-k 질의 알고리즘

4. 우선순위 큐와 트리 노드의 최적화

4.1 우선순위 큐의 관리

질의 수행 과정이 진행될수록 큐에 삽입되는 엔트리의 수는 계속해서 증가하게 된다. 하지만 그 중 상위 k개의 결과를 얻기 위해 실제로 사용되는 엔트리는 상위 일부분에 불과하다. 따라서 질의 수행에 반드시 필요한 상위 일부분만을 유지할 수 있다면 큐를 유지하는 비용을 최소화할 수 있을 것이다.

그림 3은 질의 영역과 엔트리와의 관계를 나타내고 있다. 질의 영역에 완전히 포함되는 엔트리의 경우, 앞으로 더 찾아야 하는 결과의 개수를 r ($r \le k$)이라고 했을때, 큐에 상위 r개만을 유지해도 결과가 누락되지 않음을 보장할 수 있다. 그림 3(a)에서와 같이 질의 수행 중에 부모 노드의 엔트리가 큐에서 뽑혀 나오더라도, 자식 노드의 엔트리들이 모두 질의 영역내에 포함되어 있으므로 최대값 85를 갖는 엔트리는 큐로 다시 삽입된다. 만일 큐에서 뽑힌 엔트리가 단말노드의 엔트리인 경우라면, 해당 엔트리는 결과로 바로 반환되고 r은 하나 감소하기 때문에 역시 결과가 누락되지 않음을 보장할



(a) 완전히 포함되는 경우 (b) 일부만 겹치는 경우

그림 3 엔트리와 질의 영역과의 관계

수 있다. 하지만 그림 3(b)와 같이 질의 영역에 부분적으로 겹치는 엔트리의 경우에는 큐에서 삭제되었을 경우, 최대값 85를 갖는 엔트리를 포함한 자식노드 엔트리의 일부가 질의 영역 바깥에 있기 때문에 최대값 85는 다시 큐에 삽입되지 않아서 결과의 누락이 발생할 수도 있다.

따라서 질의 영역에 완전히 포함되는 엔트리는 상위 r개만을 큐에 유지하고, 질의 영역에 부분적으로 겹치는 엔트리의 경우는 큐에 유지된 완전히 포함되는 엔트리의 *score*의 최소값보다 크거나 같은 *score*를 갖는 엔트리만 유지한다. 이를 통해 결과의 누락을 허용하지 않으면서 큐의 크기를 최소화 할 수 있다.

4.2 트리 노드의 효율적 구성

엔트리 내 저장 기법은 MRA-트리, 집계 큐브트리처럼 부모노드의 각 엔트리에 자식 노드의 최대값을 저장한다. 이러한 기법은 count, sum, max 등의 집계 질의를 자식 노드를 방문하지 않고 수행하는데 적합한 구조이나 노드의 모든 엔트리가 최대값을 저장해야 하므로 트리의 팬아웃이 감소하는 단점이 있다. 팬아웃 감소는 노드 수의 증가와 이에 따른 트리 높이의 증가를 가져올 수 있으며, 이는 R-트리 기반의 기존 알고리즘 수행에 나쁜 영향을 미치게 된다.

d를 데이타의 차원, n을 노드의 크기, m, p, v를 각각 엔트리에 저장되는 한 차원의 크기, 포인터, 최대값이라고 하면 기존 R-트리 엔트리의 크기 e, MRA-트리 엔트리의 크기 e'를 다음과 같이 구할 수 있다.

$$e = 2 \times d \times m + p$$

$$e' = 2 \times d \times m + p + v$$

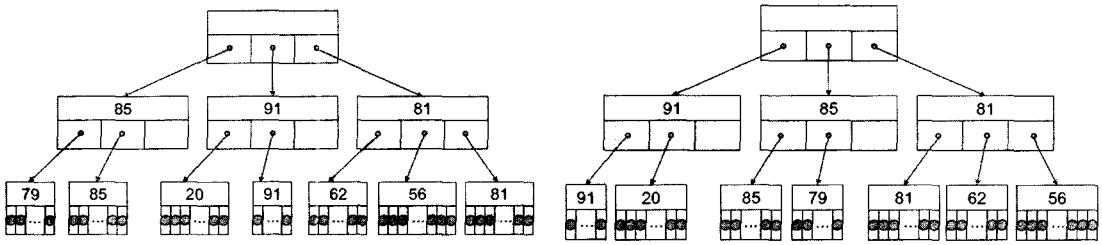
$n = 4KBytes$, $d = 2$, $m = p = v = 4Bytes$일때, 기존 R-트리 노드에 들어갈 수 있는 엔트리의 최대 개수 c, MRA-트리 노드에 들어갈 수 있는 엔트리의 최대 개수 c'를 구하면 다음과 같다.

$$c = \left\lfloor \frac{n}{e} \right\rfloor = \left\lfloor \frac{4096}{20} \right\rfloor = 204$$

$$c' = \left\lfloor \frac{n}{e'} \right\rfloor = \left\lfloor \frac{4096}{24} \right\rfloor = 170$$

위의 경우에서 MRA-트리의 팬아웃은 기존 R-트리에 비해 20%나 감소하게 된다.

엔트리의 크기를 절약하기 위해 이 논문에서는 노드의 최대값을 노드 자신에 저장하는 노드 내 저장(*In Node*) 기법을 제안한다. 그림 4(a)는 노드 내 저장 기법으로 구성된 계층 최대 R-트리의 예를 보여주고 있다. 노드의 각 엔트리는 더 이상 추가적인 정보를 저장하지 않으며, 각 노드당 오직 하나의 최대값을 저장한다. 따라서 엔트리의 크기가 기존 R-트리와 같게 함으로써 기존 R-트리와 같은 팬아웃을 보장하게 된다. 하지만 노드 내 저장 기법은 모든 자식 노드의 최대값을 구하기 위해 자식 노드들을 일일이 다 접근해야 하므로



(a) 노드 내 저장 기법 (b) 정렬된 노드 내 저장 기법

그림 4 계층 최대 R-트리의 다른 기법

노드 접근 수가 증가하는 문제점이 있다.

노드의 모든 자식 노드를 접근해야 하는 것을 막기 위해 이 논문에서는 정렬된 노드 내 저장(*In Sorted Node*) 기법을 제안한다. 정렬된 노드 내 저장 기법에서는 노드 내 저장 기법으로 구성된 트리의 각 노드를 최대값이 큰 순으로 정렬함으로써, 필요한 노드만 접근하여 질의를 수행할 수 있다. 자식 노드를 방문하던 중 현재 자식 노드의 최대값이 큐에 저장된 최대값 중 가장 작은 값 이하로 떨어지면 더 이상의 방문을 중단하면 되기 때문에 엔트리 내 저장 기법과 거의 비슷한 효과를 얻을 수 있다. 따라서 정렬된 노드 내 저장 기법으로 계층 최대 R-트리를 구성하면, 트리의 팬아웃을 보장하면서, 노드 접근 회수 감소 효과를 동시에 얻을 수 있다. 그림 4(b)는 정렬된 노드 내 저장 기법으로 구성된 계층 최대 R-트리의 예이다.

5. 실험 및 분석

범위 상위-k 질의 알고리즘의 성능을 검증하기 위해, 범위 상위-k 질의를 계층 최대 R-트리와 기존 R-트리에 대해 수행하였다. 계층 최대 R-트리에서는 범위 상위-k 질의 알고리즘을, 기존 R-트리에서는 기존 범위 질의 알고리즘을 사용하여 범위 상위-k 질의를 수행하였다. 실험에 사용된 컴퓨터의 사양은 펜티엄 III 1GHz CPU와 512MB 메모리, 40GB 하드디스크이며, 운영체제는 리눅스를 사용하였다. 디스크 페이지의 크기는 1KB로 하였으며, 2차원부터 5차원까지의 균등 분포를 갖는 데이타 셋과, TPC-H 데이타 셋[14], 타이거라인(Tiger Line) 데이타 셋[15]에 대해 질의를 100번 반복 수행하여 평균 노드 접근 횟수를 측정하였다.

그림 5는 균등분포를 갖는 2차원 데이타 셋 3,000,000개에 대해 범위 상위-100 질의를 질의 영역의 크기를 10%에서 90%까지 변화시켜가면서 수행한 결과이다. 그림 5(a)에서 알 수 있듯이, 계층 최대 R-트리는 질의 영역의 크기에 상관없이 노드 접근 횟수가 거의 일정한 것을 알 수 있다. 이는 질의 영역내의 모든 단말 노드를 접근해야 하는 R-트리에 비해, 계층 최대 R-트리의 경우 질의 영역의 크기와는 무관하게 필요한 만큼의 단말 노드를 선택적으로 접근할 수 있기 때문이다.

계층 최대 R-트리의 노드 접근 횟수는 노드 구성에 따라서 달라진다. 그림 5(b)에서 알 수 있듯이 엔트리 내 저장기법과 정렬된 노드 내 저장 기법이 노드 내 저장 기법에 비해 성능이 좋음을 알 수 있다. 엔트리 내 저장 기법에서는 자식 노드의 최대값이 부모 노드의 엔트리마다 저장되어 있는 반면, 노드 내 저장 기법에서는 자식 노드의 최대값을 얻기 위해 자식 노드를 모두 방문해야 하기 때문이다. 정렬된 노드 내 저장 기법에서는 각 노드를 최대값이 큰 순서대로 정렬해 놓았기 때문에 자식 노드의 상위 일부분만을 접근하고 질의를 수행할 수 있다.

그림 6은 계층 최대 R-트리의 여러 기법과, MRA-트리, 기존 R-트리 등에 대해 팬아웃과 트리 높이를 비교한 결과이다. 2차원 트리에 대해 노드 크기를 1024 byte로 고정하였을 경우 R-트리, 노드 내 저장 기법, 정렬된 노드 내 저장 기법은 노드 내 각 엔트리의 크기가 20 byte인 반면, 엔트리 내 저장 기법은 엔트리마다

	노드 크기 (bytes)	엔트리 크기 (bytes)	노드내의 엔트리 개수 (팬아웃)	트리 높이
기존 R-트리	1024	20	50	4
MRA-트리	1024	24	42	5
HMR-트리(In Entry)	1024	24	42	5
HMR-트리(In Node)	1024	20	50	4
HMR-트리(In Sorted Node)	1024	20	50	4

그림 6 계층 최대 R-트리와 기존 트리의 팬아웃과 트리 높이 비교

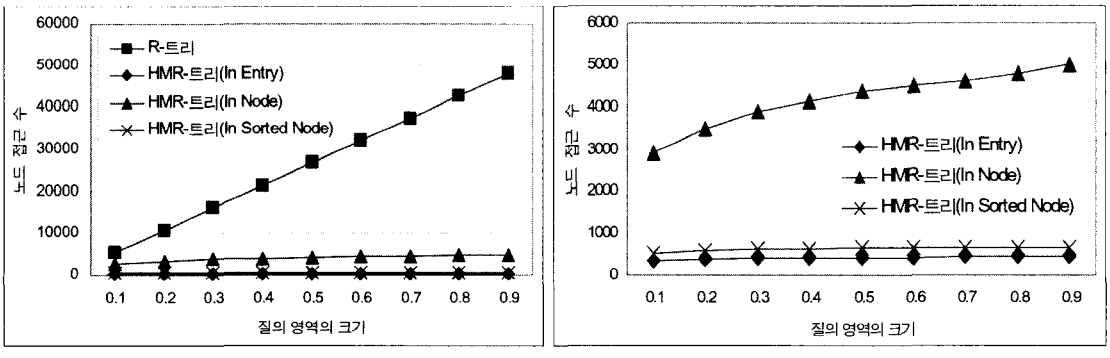


그림 5 질의 영역의 크기 변화에 따른 노드 접근 회수 비교

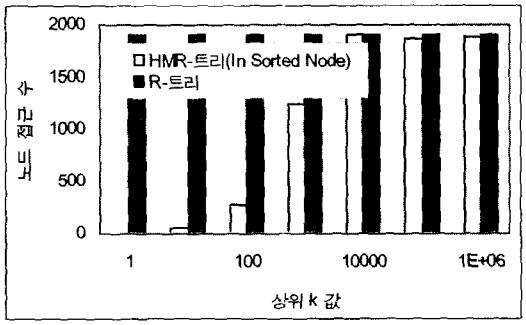
최대값을 저장해야 하므로 24 byte 만큼이 공간이 필요하다. 따라서 기존 R-트리와 노드 내 저장기법, 정렬된 노드 내 저장 기법이 노드 내 엔트리의 수가 50개인 반면 엔트리 내 저장 기법은 42개로 줄어들며, 이로 인해 3,000,000개의 데이타를 저장하는 경우 기존 R-트리와 노드 내 저장기법, 정렬된 노드 내 저장 기법의 높이가 4인 반면, 엔트리 내 저장기법은 높이가 5로 늘어나게 된다.

따라서 정렬된 노드 내 저장 기법으로 구성한 계층 최대 R-트리는 기존 R-트리와 비슷한 팬아웃을 보이는 노드 내 저장 기법의 장점과, 더 적은 수의 노드 접근으로 질의 수행이 가능한 엔트리 내 저장 기법의 장점을 절충한 기법이라 할 수 있다.

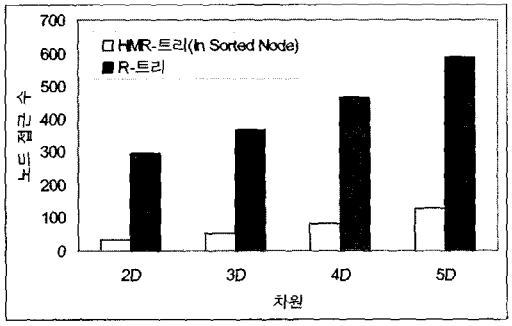
그림 7(a)는 질의 결과 크기인 k를 변화해가며 질의를 수행한 결과이다. 정렬된 노드 내 저장 기법의 계층 최대 R-트리를 사용했으며, 100만개의 2차원 데이타에 대해 질의 영역을 10%로 하여 질의를 수행하였다. 역시 기존 R-트리가 k값에 상관없이 항상 큰 노드 접근 횟수를 보이는 반면, 계층 최대 R-트리는 k값이 작은 경우 적은 수의 노드 접근만으로 질의 수행이 가능하며, k값이 커짐에 따라 노드 접근 횟수가 증가하게 된다. 그림 7(b)는 차원을 변화시켜가면서 질의를 수행한 결과이다. 10만개의 데이타에 대해 질의 영역을 30%로 하여 질의를 수행하였다. 차원을 증가시켜도 기존 R-트리에 비해 항상 적은 수의 노드 접근만으로 질의 수행이 가능한 것을 알 수 있다.

그림 8은 실제 데이타를 사용하여 질의를 수행한 결과이다. 그림 8(a)는 TPC-H 데이타에 대한 결과를 그림 8(b)는 타이거라인 데이타 셋에 대한 결과를 보여주고 있다. TPC-H 데이타는 1,500,000개의 레코드를 갖는 orders 테이블을 사용하였으며, 타이거라인 데이타는 2002년 뉴욕의 데이타를 사용하였다. 각 데이타의 차원은 2차원이며, 상위 100개를 검색하였다. 실제 데이타에 적용한 결과도 임의로 생성한 데이타에 대한 결과와 마찬가지로 기존 기법에 비해 질의 영역의 크기에 상관없이 일정하게 좋은 성능을 보이고 있다.

6. 결 론

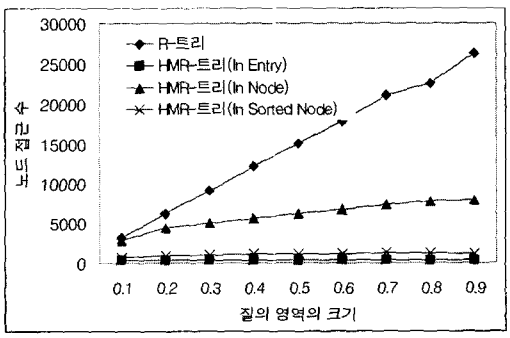


(a) k의 변화에 따른 노드 접근 회수

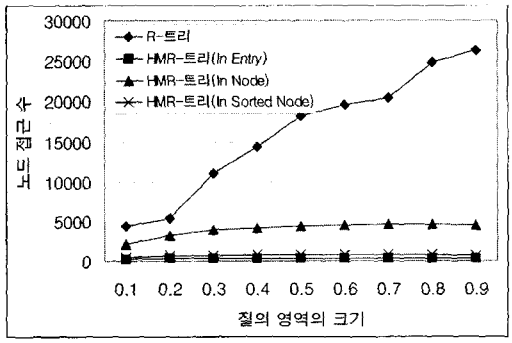


(b) 차원의 변화에 따른 노드 접근 회수

그림 7 k값과 차원의 변화에 따른 노드 접근 회수의 비교



(a) TPC-H 데이타 셋



(b) 타이거라인 데이타 셋

그림 8 실제 데이타 셋에 대한 범위 상위-k 질의의 수행 결과

이 논문에서는 범위 상위-k 질의를 효율적으로 수행하는 알고리즘과, 자료구조인 계층 최대 R-트리를 제안하였다. 계층 최대 R-트리는 각 중간 노드마다 최대값을 저장하며, 알고리즘은 질의 영역내의 모든 단말노드를 접근하지 않고도, 우선순위 큐를 사용하여 최대값의 순서대로 단말노드를 접근하여 질의를 수행할 수 있다. 우선순위 큐의 상위 일부분만을 유지함으로써 큐의 유지 비용을 최소화 하고, 정렬된 노드 내 저장 기법을 통해 기존 R-트리와 같은 팬아웃을 보장할 수 있다.

실험 결과에서 알 수 있듯이, 제안된 기법은 기존 R-트리에 기반한 기법에 비해 항상 좋은 성능을 보이며, 질의 영역의 크기에 상관없이 항상 일정한 성능을 나타낸다. k값이 작은 경우에는 기존 기법에 비해 15%의 노드 접근만으로도 질의를 수행할 수 있으며, k값이 큰 경우에도 여전히 좋은 성능을 나타낸다.

참 고 문 헌

[1] Ralf Hartmut Guting, "An Introduction to Spatial Database Systems," VLDB Journal vol.3, No.4, pp.357-399, 1994.

[2] Surajit Chaudhuri, Umeshwar Dayal, "An Overview of Data Warehousing and OLAP Technology," SIGMOD Record vol.26, No.1, pp.65-74, 1997.

[3] Antonin Guttman: R-Trees, "A Dynamic Index Structure for Spatial Searching," SIGMOD Conference, pp.47-57, 1984.

[4] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, Bernhard Seeger, "The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles," SIGMOD Conference, pp.322-331, 1990.

[5] 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, pp.152-159, 1996.

[6] Venky Harinarayan, Anand Rajaraman, Jeffrey D. Ullman, "Implementing Data Cubes Efficiently," SIGMOD Conference, pp.205-216, 1996.

[7] Ching-Tien Ho, Rakesh Agrawal, Nimrod Megiddo, Ramakrishnan Srikant, "Range Queries in OLAP Data Cubes," SIGMOD Conference, pp.73-88, 1997.

[8] Steven Geffner, Divyakant Agrawal, Amr El Abbadi, Terence R. Smith, "Relative Prefix Sums: An Efficient Approach for Querying Dynamic OLAP Data Cubes," Int. Conference on Data Engineering, pp.328-335, 1999.

[9] Chee Yong Chan, Yannis E. Ioannidis, "Hierarchical Prefix Cubes for Range-Sum Queries," VLDB, pp.675-686, 1999.

[10] Sin Yeung Lee, Tok Wang Ling, Hua-Gang Li, "Hierarchical Compact Cube for Range-Max Queries," VLDB, pp.232-241, 2000.

[11] Seok-Ju Chun, Chin-Wan Chung, Ju-Hong Lee, Seok-Lyong Lee, "Dynamic Update Cube for Range-sum Queries," VLDB, pp.521-530, 2001.

[12] Iosif Lazaridis, Sharad Mehrotra, "Progressive Approximate Aggregate Queries with a Multi-Resolution Tree Structure," SIGMOD Conference, 2001.

[13] Seokjin Hong, Byoungho Song, Sukho Lee, "Efficient Execution of Range-Aggregate Queries in Data Warehouse Environments," ER, pp.299-310, 2001.

[14] TPC-H, http://www.tpc.org/tpch, 2003.

[15] U.S.Census Bureau - TIGER/LINE, http://www.census.gov/geo/www/tiger, 2003.

홍 석 진

1998년 서울대학교 컴퓨터공학과 졸업(공학사). 2000년 서울대학교 대학원 컴퓨터공학과 졸업(공학석사). 2000년~현재 서울대학교 대학원 전기컴퓨터공학부 박사과정. 관심분야는 데이타웨어하우스, OLAP, 시공간 데이타베이스, XML

이 상 준

정보과학회논문지 : 데이타베이스
제 31 권 제 1 호 참조

이 석 호

정보과학회논문지 : 데이타베이스
제 31 권 제 1 호 참조