위치 기반 서비스에서 이동 객체의 궤적을 위한 HR-트리의 확장

우성현°* 김미경* 전현식** 박현주***
한밭대학교 *정보통신공학과, **전파공학과, ***정보통신컴퓨터공학부
wsungh123°*@naver.com, {mkkim922*, hsjeon**, phj***}@hanbat.ac.kr

Extension of HR-Tree for Trajectory of Moving Objects in Location-Based Services

Sung-hyun Woo°*, Mi-kyeng Kim*, Hyun-sik Jeon**, Hyun-ju Park***
*Dept. of Information Communication Engineering, Hanbat National University
**Dept. of Radio-Wave Engineering, Hanbat National University
***Div. of Infomation Communication Computer Engineering, Hanbat National University

요 약

시간의 흐름에 따라 그 위치가 빈번히 변화하는 이동 객체의 특성으로 인해 실시간으로 증가하는 이동 객체의 연산 정보를 효과적으로 관리할 수 있는 효율적인 연산 기법이 요구된다. 따라서 본 논문에서는 이동 객체의 시공간 연산을 위해 기존에 제안되었던 HR 트리의 성능을 개선시킬 수 있는 확장된 HR-트리(Extended HR-Tree : EHR-Tree)를 제안한다. 기존의 HR 트리는 삽입, 삭제, 갱신과 같은 연산이 빈번한 경우에 단말 노드 및 비단말 노드를 새로 생성해야 함으로 인하여 성능이 떨어지고, 공간의 낭비가 있게 된다. 이 문제를 해결하기위해 본 논문에서 제안하는 EHR-트리는 연산이 발생할 때마다 새로운 HR-트리를 생성하는 것이 아니라 시간 구간을 두어서 새로 발생한 연산이 같은 시간 구간에 있을 경우 그 단위 시간에 생성된 HR-트리에 그대로 삽입, 삭제, 갱신과 같은 연산을 수행하게 된다. 따라서 기존 HR-트리에서 단말 노드 및 비단말 노드를 새로 생성해야함으로 발생되던 많은 저장 공간 요구를 감소시킴으로, 즉 전체 연산 크기를 작게 하여 디스크 I/O수를 감소시킴으로써, 시공간 질의의 처리 속도를 향상시켜 효율적인 질의가 가능하도록 한다.

1. 서 론

최근 GPS 및 무선 데이터 전송능력이 있는 휴대용 단말기의 등장과 이동 무선 컴퓨팅 기술의 발달로 인해 위치기반 서비스(Location-Based Services: LBS)가 무선 인터넷 시장에서 중요한 이슈로 대두되고 있다. LBS란 이동통신망을 기반으로 사람이나 사물의 위치를 정확하게 파악하고 이를 활용하는 응용시스템 및 서비스를 말한다. LBS가 보편화 되어감에 따라 다양한 서비스들이 일정한 장소에서 뿐만 아니라, 휴대용 전화기, PDA, 노트북과 같은 이동성을 지닌 단말기를 이용하여 이동 중에도 지속된다. 따라서 시간의 흐름에 따라 객체가 이동하면서 그 위치를 연속적으로 변경하는 특징을 지닌 이동 객체를 효율적으로 저장 및 관리 할 수 있는 색인 기술이 요구된다. 이동 객체의 위치 정보를 색인하기 위한 기존의 연구는 크게 세 가지가 있는데 그 중에서도 현재 및 과거 정보를 효율적으로 처리하기 위한 시공간 접근 기법은 본 논문과 밀접한 관련이 있으며, 그 중 HR-트리가 여기에 속한다. 하지만 HR-트리는 연산이 발생할 때마다 새로운 HR-트리를 생성해야하므로, 빈번한 연산이 발생할 경우 성능이 떨어지고, 공간의 낭비가 있게 된다. 따라서 본 논문에서는 이동 객체의 시공간 연산을 위해 기존에 제안되었던 HR 트리의 성능을 개선시킬 수 있는 확장된 HR-트리(Extended HR-Tree:EHR-Tree)를 제안한다. EHR-트리는 연산이 발생할 때마다 새로운 HR-트리를 생성하는 것이 아니라 시간 구간을 두어서 새로 발생한 연산이 같은 시간 구간에 있을 경우 그 단위 시간에 생성된 HR-트리에 그대로 삽입, 삭제, 갱신과 같은 연산을 수행하게 된다. 따라서 EHR-트리는 기존 HR-트리에서 단말 노드 및 비단말 노드를 새로 생성해야함으로 발생되던 많은 저장 공간 요구를 감소시킨다. 그리고 전체 연산 크기를 작게 하여 디스크 I/O수를 감소시킴으로 시공간 질의의 처리 속도를 향상시켜 효율적인 질의가 가능하도록 한다.

본 논문의 구성은 다음과 같다. 2장에서는 이동객체를 위한 색인 기법과 관련된 기존의 연구들을 살펴보고 3장에서는 기존 HR-트리를 확장한 EHR-트리에 대해서 설명한다. 그리고 마지막 4장에서 결론을 맺는다.

2. 관련 연구

기존의 공간 색인은 Quad-트리, KDB-트리와 같은 공간-분할 방법과 R-트리 계열의 데이터-분할 방법으로 나뉜다. 이동 객체의 분포는 시간에 따라 변하기 때문에, 공간-분할 방법은 적합하지 않다. 또한 시간 도메인은 동적으로 증가하기 때문에, 공간-분할 방법에서 시간 도메인을 관리하기가 어렵다는 문제가 있다. 대부분의 이전 연구에서도 이와 같은 문제로 인하여 R-트리의 변형(variants)을 사용한다.

이동 객체의 위치 정보를 색인하기 위한 기존의 연구는 크게 세 가지로 분류된다. 먼저 이동 객체의 현재 및 과거 정보를 효율적으로 처리하기 위한 시공간 접근 기법과 예측된 미래 시간을 처리하기 위한 시공간 접근 기법, 마지막으로 이동 객체의 궤적 기반 질의를 효과적으로 처리하기 위한 시공간 접근 기법이다. 그 중에서도 현재 및 과거 정보를 효율적으로 처리하기 위한 시공간 접근 기법은 본 논문과 매우 밀접한 관련이 있으며, HR-트리가 여기에 속한다. 지금부터는 본 논문과 직접적인 관련이 있는 HR-트리의 개념 및 특징, 장·단점에 대해 알아본다.

HR-트리(Historical R-tree)는 R-트리에 거래시간 개념을 추가하여 이력정보를 표현하고 있으며 연속적인 상태를 표현하기 위하여 R-트리에 중복 개념을 추가한 것이다. 모든 타임스탬프마다 현재의 상태에 대한 2차원 R-트리를 만들어 유지하는 방법으로 모든 이전의 상태를 2차원 R-트리로 유지하는 구조를 가지고 있으며, 새롭게 생성되는 노드들의 수가 기존에 제공된 트리들에 비해 적게 유지되도록 한다. 그림 1은 HR-트리의 일반적인 구조를 나타낸 그림이다.

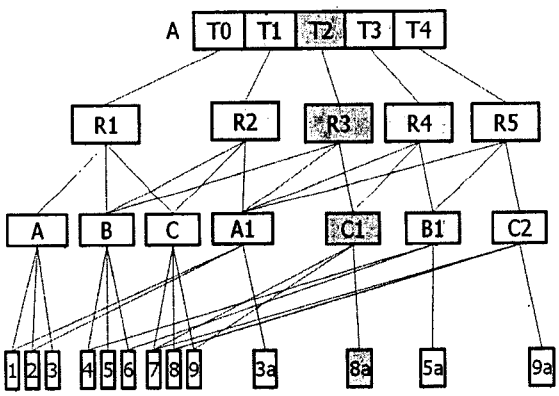

그림 1. HR 트리의 일반적인 구조

이 그림에서 나타나는 바와 같이 연산이 발생한 시간을 저장하고 있는 타임스탬프, 즉 T0, T1, T2, T3, T4에서 완전한 R-트리들이 유지되고 있다. 그림1에서 살펴본 바와 같이 시간이 경과함에 따라 변하지 않는 객체들의 정보는 공유해서 연결하고, 변화하는 객체들에 대한 정보만 저장하게 된다. 따라서 HR-트리는 이동객체들의 이동이 빈번하지 않은 경우에는 매우 효율적이라 할 수 있다. 하지만 계속적인 추가, 삭제, 갱신이 일어나는 경우에는 연산이 발생할 때마다 R-트리를 새로 생성해야 한다. 따라서 그에 따른 많은 저장 공간을 요구하기 때문에 비효율적이고, 실제적인 응용에는 무리가 따른다.

3. 제안하는 EHR-트리

본 논문에서는 기존에 제안되었던 HR-트리의 문제점인 계속적인 추가, 삭제, 갱신이 있는 객체들에서 발생되던 비효율성을 해결하고 저장 공간 측면과 질의 처리의 효율성을 높이기 위해서 확장된 HR-트리인 EHR-트리를 제안한다.

3.1 구조

EHR-트리의 일반적인 구조는 그림 2와 같다. 그림 2를 살펴보면 EHR-트리의 구조가 기존의 HR-트리와 유사하다는 것을 알 수 있다. 하지만 연산이 발생할 때마다 새로운 R-트리를 생성하는 기존의 HR-트리와 달리 EHR-트리는 시간구간(time-interval)을 두게 된다. 따라서 새로 발생한 연산이 현재 지속되고 있는 시간구간 범위 내에서 발생하게 되면, 새로운 R-트리를 생성하지 않고 해당하는 단위 시간에 내에 생성되어 있던 기존의 R-트리에서 삽입, 삭제, 갱신의 연산이 이루어지게 된다.

그림 2를 살펴보면 $TS_0=[t_0,t_1)$, $TS_1=[t_1,t_2)$, $TS_2=[t_2,t_3)$과 같이 시간구간(time-interval)이 설정 되어있는 것을 볼 수 있다. 그림 2에 나타난 EHR-트리가 완성되는 과정을 살펴보자. $TS_1=[t_1,t_2)$의 시간 구간에서 3의 값의 갱신이 발생하게 된다. 따라서 R2의 트리가 생성하게 되게 된다. 그리고 $TS_1=[t_1,t_2)$의 시간 구간 안에서 8의 값의 갱신이 발생되게 된다. 기존의 HR-트리는 그림 1에서 보면 알 수 있듯이 연산이 발생시 마다 R-트리를 생성하는 것을 볼 수 있다. 하지만 EHR-트리는 현재 진행되고 있는 $TS_1=[t_1,t_2)$의 시간 구간 범위 안에서 새로운 연산이 요구되었기 때문에 기존의 R2 트리 내에서 갱신이 일어나게 되는 것을 볼 수 있다. 그리고 지정된 $TS_1=[t_1,t_2)$범위의 T_2시간이 지나게 되면 다음에 발생되는 연산은 $TS_2=[t_2,t_3)$에서 처리되게 된다. 따라서 5의 값의 갱신은 R3 트리를 새로 생성하게 되고, $TS_2=[t_2,t_3)$의 시간 구간 내에서 발생된 9의 값의 갱신 역시 R3에서 발생하게 된다. 그리고 다음에 발생될 연산을 now인 현재 시간 구간 $TS_2=[t_2,t_3)$에서 T_3의 시간이 지날 때까지 대기하게 되고, 만약 T_3의 시간이 지나게 되면 새로운 TS_3 시간 구간에서 다음으로 발생될 연산을 기다리게 된다. 만약 TS_3 시간 구간에서 연산이 발생하지 않을 경우에는 TS_3 시간 구간은 사라지게 되고, 다음의 시간 구간인 TS_4 시간 구간에서 대기하게 된다.

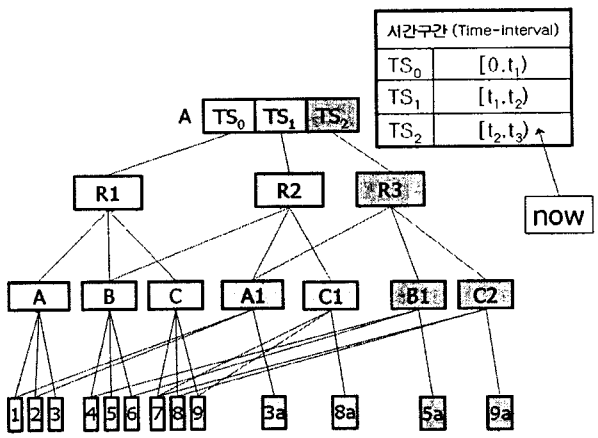

그림 2. EHR-트리의 구조

앞에서 살펴본 바와 같이 기존 HR-트리에서는 계속적인 연산으로 인하여 단말 노드 및 비단말 노드를 새로 생성해야했다. 하지만 EHR-트리는 계속적인 연산으로 인해 발생되던 많은 저장 공간 요구를 감소시키고, 또한 전체 트리 크기를 작게 하여 디스크 I/O수를 감소시킴으로써, 시공간 질의의 처리 속도를 향상시켜 효율적인 질의가 가능하도록 한다. 하지만 지금까지 살펴본 EHR-트리는 정확한 시간을 검색하는데 문제점을 가지고 있다.

정보이용자로부터 '12시 33분 출발했던 기차의 정보'의 검색을 요구 받았다고 가정하자. 기존의 HR-트리는 연산이 발생한 시간, 즉 기차의 출발 시간을 타임스탬프에서 저장하고 있기 때문에 쉽게 원하는 정보를 찾을 수 있다. 하지만 EHR-트리는 타임스탬프를 두는 대신 시간 구간(Time-interval)을 두게 됨으로 시간 구간을 30분 단위 지정했다고 가정했을 시 원하는 정보가 12시 30분에서 1시까지의 시간 어느 부분에 있다는 것만 알 수 있을 뿐 정확한 데이터를 찾기는 쉽지 않다. 이 같은 문제를 해결하기 위해 EHR-트리는 기존의 HR-트리와 다른 단말 노드의 구조를 가지고 있다. 그림 3은 EHR-트리의 단말 노드의 구조를 나타내고 있다.

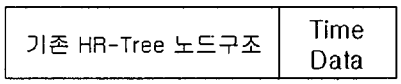

그림 3. EHR-트리 단말노드의 구조

그림 3을 살펴보면 EHR-트리의 단말 노드는 기존의 HR-트리 노드 구조에 노드가 생성된 시간 데이터가 추가로 포함되었다는 것을 볼 수 있다. 따라서 시간 정보의 질의를 받게 되면 해당되는 시간 구간을 먼저 검색하게 되고, 깊이 탐색을 시행함으로 단말 노드에 저장되어 있는 정확한 시간 정보를 찾을 수 있게 된다.

3.2 알고리즘

EHR-트리의 삽입 알고리즘을 살펴봄으로써 EHR-트리의 구조를 이해하도록 한다. 아래에서 살펴보는 바와 같이 EHR-트리는 기존의 HR-트리의 삽입 알고리즘과 유사하다. 단지 삽입 전에 시간 구간을 더해서 불필요한 저장 공간의 낭비를 감소시킨다.

```
Algorithm Insert(On, OR)
1. If A[pt] + C < now // 만약 A[pt:가장 최근 엔트리]
                      // + C(시간구간)가 현재시간
                      // 보다 작다면
   then 현재 생성된 EHR-트리에 삽입한다.
2. else
3. { create a new state in the R-tree }
4. { create a root NR to insert on }
     Invoke CreateBranch to create a new logical
          R-tree rooted at NR. The new logical R-tree
          contains a leaf node L in which to place On;
5. { insert On in L }
     If L has room for another entry
       Then insert on in L;
       Else remove L created by CreateBranch
                invoke SplitNode to obtain a new L
                  and LP containing On and all the
                other entries of L removed;
6. { propagate split upwards }
     If a split was performed
          then Invoke AdjustTree on L, also passing LP;
7. { grow tree taller }
     If node split propagation caused a root split
          then create a new root NR whose children are
               the resulting nodes resulting from the
               root split;
     adjust the entry in A to point to the new root NR;
```

4. 결 론

본 논문에서는 이동 객체의 궤적을 삽입, 삭제, 갱신하기 위해 기존에 제안되었던 EHR-트리의 성능을 개선시킬 수 있는 확장된 EHR트리를 제안하였다. 본 논문에서 제안하는 EHR-트리는 연산이 발생할 때마다 새로운 HR-트리를 생성하는 것이 아니라 시간 구간을 두어서 새로 발생한 연산이 같은 시간 구간에 있을 경우 그 단위 시간에 생성된 HR-트리에 그대로 삽입, 삭제, 갱신과 같은 연산을 수행하게 된다. 따라서 기존 HR-트리에서 단말 노드 및 비단말 노드를 새로 생성해야함으로 발생되던 많은 저장 공간 요구를 감소시킴으로, 즉 전체 연산 크기를 작게 하여 디스크 I/O수를 감소시킴으로써, 시공간 질의의 처리 속도를 향상시켜 효율적인 질의가 가능하도록 한다.

5. 참고문헌

[1] Mario A.Nascimento, Jefferson R.O.Silva, "Towards Historical R-Trees," Acm Press New York, NY, USA, pp.235-240, 1998

[2] Yufei Tao, Dimitris Papadias, Tao,Efficient "Historical R-trees," IEEE Computer Society, pp.223-232, 2001

[3] Antonin Guttman, "R-trees: a dynamic index structure for spatial searching," ACM Press, pp. 47-57, 1984

[4] 한기준, "위치 기반 서비스(LBS)의 표준화와 연구동향", 정보화정책, 10권, 4호, pp.3~17, 2003

[5] 전봉기, 임덕성, 홍봉희, "이동체 데이터베이스를 위한 색인 기법", 데이타베이스연구, 18권 4호, pp23~35, 2002