이동 객체의 궤적 검색을 위한 인덱스 구조

이계영[0] 권동섭 송병호[+] 이석호[$]
서울대학교 전기컴퓨터공학부
[+]상명대학교 소프트웨어학과
[$]서울대학교 컴퓨터공학부
{zolid, subby}@db.snu.ac.kr bhsong@sangmyung.ac.kr shlee@cse.snu.ac.kr

An Index Structure for Searching Moving Object Trajectories

Gyeyoung Lee[0] Dongseop Kwon Byoungho Song[+] Sukho Lee[$]
School of Electrical Engineering and Computer Science, Seoul National University
[+]Division of Software Science, Sangmyung University
[$]School of Computer Science and Engineering, Seoul National University

요 약

이동 기기의 발달로 이동 객체의 위치 정보에 대한 연구가 진행되고 있다. 그러나 시공간 데이타베이스에서 요구하는 질의를 효과적으로 처리하기 위한 인덱스 구조는 많지 않다. 사용자는 먼저 범위 질의를 던져 이동 객체를 뽑아낸 후 궤적 질의를 던지게 되는데 이것을 결합 질의라 한다. 기존 방법인 TB-tree는 MBR의 겹침이 심해 성능이 저하된다.

본 논문에서는 결합 질의를 효과적으로 처리할 수 있는 CR-tree 인덱스 구조를 제안한다. CR-tree는 R-tree와 연결노드로 구성된다. R-tree 부분은 MBR의 겹침을 최소화해 범위 질의를 효과적으로 처리하고 연결 노드 부분은 이동 객체별로 궤적의 부분을 차례로 저장해서 궤적 질의를 빠르게 처리한다. 즉 결합 질의시 R-tree를 이용해 범위에 속하는 이동 객체를 검색하고 연결 노드를 이용해 그 객체의 궤적을 찾는다.

1. 서론

PDA나 휴대폰 등의 이동 기기와 GPS의 발달로 공상 영화에나 볼 수 있었던 위치정보 서비스가 실현되고 있다. 이런 시스템에서 차량이나 휴대폰, PDA 등을 이동 객체라 한다. 전통적인 데이타베이스 관리 시스템은 고정된 객체에 대한 저장 및 관리에 초점이 맞추어져 왔기 때문에 기존의 시스템을 가지고 이동 객체에 활용할 수 없다. 현재는 데이타 타입의 정의 및 질의 정의와 같은 모델링과 이동 객체의 저장과 검색을 위한 인덱스 구조 등이 연구되고 있다.

인덱스 구조 연구는 현재만의 위치를 저장하는 분야와 과거부터 현재까지의 모든 위치를 저장하는 분야 두 가지로 분류할 수 있다. 전자는 현재의 위치 정보를 제공할 뿐만 아니라 미래의 정보를 예측한다. 후자의 경우 연속된 이동을 일정한 시간 간격으로 쪼개 불연속적인 데이타로 만들어 저장한다. 이런 이동 객체의 연속된 이동을 궤적이라 한다. 이런 시스템에서 많이 사용되는 질의는 궤적에 관한 것(어떤 이동객체가 어디로 움직였는가?, 이동한 거리는 얼마인가?, 두 객체의 위상은 어떤가?)이다.

궤적을 저장하고 검색하기 위해 제안된 구조로는 3D R-tree[1]와 Quadtree를 이용한 방법[2], 그리고 R-tree를 변형한 방법들[3]이 있다. 이런 방법들은 기존의 공간 데이타베이스 시스템에서 사용하던 방법들을 단순히 시간 차원을 추가하거나 시간별로 생기는 공간 인덱스를 연결한 것에 불과하다. 그래서 궤적에 관한 질의를 효과적으로 처리하기 어렵다.

본 논문에서는 R-tree와 연결 노드가 합쳐진 CR-tree를 제안한다. CR-tree는 범위질의와 궤적 질의가 합쳐진 결합질의를 보다 빠르게 처리한다. 본 논문의 구성은 다음과 같다. 2장에서는 기존의 인덱스 구조에 대해 알아보고 3장에서는 여기서 사용할 데이타 타입과 질의에 대해 설명한다. 4장에서는 제안하는 CR-tree의 구조와 알고리즘에 대해 살펴보고 5장에서는 기존의 인덱스 구조와 성능을 비교한다. 마지막으로 6장에서 결론과 향후 연구 방향을 제시한다.

2. 관련 연구

시공간 데이타는 공간 데이타와는 다르게 시간축으로 계속 증가하는 성격을 가지고 있다. 질의 또한 공간 데이타베이스 시스템과는 다르게 궤적에 대한 질의가 중요하다. 그래서 [4]에서는 질의를 새로이 분류하고 그것을 보다 효율적으로 처리할 수 있는 TB-tree(Trajectory-Bundle tree)를 제시하였다. TB-tree는 R-tree의 구조와 같지만 삽입 알고리즘이 다르다. R-tree는 단말 노드의 MBR(Minimum bounding Rectangle)이 가장 작게 증가하도록 삽입되지만 TB-tree는 궤적 보존(Trajectory preservation) 방법을 이용해서 하나의 단말 노드에 같은 이동 객체의 세그먼트만 삽입된다. 즉 한 단말 노드에 같은 이동 객체의 세그먼트가 시간 순서로 삽입이 된다. 포화 상태면 분할이 일어나지 않고 새로운 단말 노드를 만들어 새로 들어온 세그먼트만 그 노드에 삽입하고 포인터로 연결한다. 궤적 보존 방법을 이용하면 객체의 전후 세그먼트를 쉽게 찾을 수 있게 해준다. 이것은 궤적에 대한 질의를 빠르게 처리할 수 있게 해주지만 단말 노드의 MBR이 커지게 된다. [4]에서의 실험 결과를 보면 각 축의 1% 범위 질의에 대해 1000개의 객체인 경우 TB-tree가 R-tree보다 3~4배 접근 노드 수가 많음을 알 수 있다. 또한 객체가 많아질수록 MBR의 겹침 현상이 두드러져 밀도가 높을수록 성능이 나빠짐을 알 수 있다.

3. 데이타 타입 및 질의

R-tree를 이용한 방법들은 궤적을 일정한 시간 간격에 따라 나누어 세그먼트로 저장한다. 보통 세그먼트는 $\{(x_i, y_i, t_i),$

$(x_{i+1}, y_{i+1}, t_{i+1})$로 구성된다. 세그먼트 안에서 방향을 나타내는 필드가 추가되기도 한다.[4]

질의는 아직 표준화가 안되어 약간씩 논문마다 다르게 기술하고 있다. 공간 데이타베이스에서 사용되던 점 질의나 범위 질의가 있다. 시간축에 대한 질의를 별도로 생각해서 time slice, time interval 질의를 분류하기도 한다. 하지만 이것은 범위 질의의 일종이므로 여기서는 범위 질의로 분류하겠다. [7]에서 분류한 것을 보면 범위 질의와 궤적 질의가 있다. 궤적 질의란 이동객체의 움직임을 가지고 알 수 있는 질의를 말한다. 예를 들어 어떤 범위에서의 이동 거리나 두 객체사이의 위상 같은 것이 궤적 질의이다. 이것은 위상 질의(topological queries)와 이동 질의(navigational queries)로 나뉘는데 더욱 자세한 것은 이 논문을 살펴보기 바란다. 궤적 질의는 범위 안에 속하는 특정 이동 객체의 궤적을 모두 찾는 것이므로 범위 질의로 처리하기 위해서는 범위에 속하는 다른 이동 객체의 궤적까지 찾게 된다. 따라서 궤적 질의를 범위 질의로 찾기 위해서는 접근 노드 수가 많아진다.

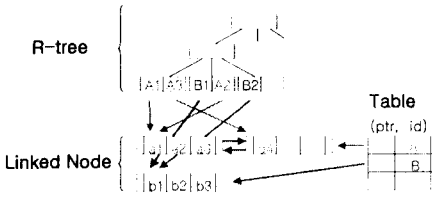
수많은 이동 객체가 있을 때 모든 이동 객체의 궤적을 조사할 수는 없다. 사용자는 일정 범위를 주고 그 범위에 속하는 이동 객체들에 대한 움직임을 알고 싶을 것이다. 예를 들면 "오늘 4시와 5시 사이에 신림 사거리를 지나는 택시의 오늘 행적을 알고 싶다." 이런 질의는 처음에 범위 질의가 행해져서 그 범위에 속하는 이동 객체를 찾아 궤적을 찾게 된다. 이것을 결합 질의라 한다.[4] 처음 이동 객체를 찾는 범위를 inner range라고 하고 다음 궤적을 찾을 범위를 outer range라 한다. 위의 예에서 inner range는 "오늘 4시와 5시 사이에 신림 사거리"이고 outer range는 "오른 행적"이므로 "오늘"이다.

기존의 R-tree는 결합 질의를 처리하기 위해 inner range뿐만 아니라 outer range에 속하는 모든 이동 객체의 궤적을 찾게 된다. TB-tree는 inner range로 범위 질의를 처리하고 여기에 속하는 이동 객체의 단말 노드만을 따라가 궤적을 찾기 때문에 접근 노드 수를 줄일 수 있다. 하지만 TB-tree의 삽입 알고리즘은 단말 노드의 MBR을 크게 만들어 범위 질의의 성능을 저하시킨다. 따라서 TB-tree는 이동 객체의 수가 많고 밀도가 높은 응용에서는 결합 질의의 성능이 떨어진다.

4. CR-tree(Combined R-tree)

CR-tree는 결합 질의를 효율적으로 처리하기 위해 R-tree와 이동 객체별 연결 노드(linked node)를 결합한 형태이다. 기본적으로 R-tree의 방법을 그대로 이용하며 삽입시 단말 노드와 연결 노드를 연결한다. R-tree부분은 범위 질의를, 연결 노드는 trajectory preservation을 해서 궤적 질의를 빠르게 처리 할 수 있다.

4.1 구조



<그림 1> CR-tree의 구조 예

<그림 1>처럼 CR-tree는 크게 세 부분으로 나눌 수 있다. 윗부분의 R-tree 부분과 아랫부분의 연결 노드 그리고 테이블로 구성된다.

R-tree부분은 [5]에서 제안한 것을 3차원으로 만든 것이다. 단말 노드의 엔트리는 {객체아이디, (x_i, y_i, t_i) , $(x_{i+1}, y_{i+1}, t_{i+1})$, 포인터}로 구성되고 엔트리의 포인터는 연결 노드를 가리킨다. 비단말 노드의 엔트리는 {(x_i, y_i, t_i) , $(x_{i+1}, y_{i+1}, t_{i+1})$, 포인터}로 구성되고 포인터는 하위 노드를 가리킨다. 연결 노드는 각 이동 객체별로 저장하는데 세그먼트로 들어온 데이타를 중복을 피하기 위해 점으로 저장한다. 시간 순서대로 점을 저장하므로 궤적 질의를 연결 노드만을 이용해서 처리할 수 있다. 그리고 한 노드에 세그먼트로 저장하는 것보다 거의 두 배로 저장하므로 궤적 질의를 처리할 때 TB-tree보다 접근 노드 수를 줄일 수 있다. 테이블은 이동 객체 아이디와 포인터의 리스트로 구성된다. 테이블은 이동 객체 별로 가장 최근에 삽입된 연결 노드를 포인터로 연결해서 삽입시 이용된다.

4.2 삽입 알고리즘

```
Algorithm insert(S)
S : segment
    LN := pre_insert(S)
    N := ChooseLeaf(S)
    if(NOI full)
        Split_node(S,N,LN)
    else
        N에 S 삽입
        N과 LN을 연결
```

<그림 2> CR-tree의 삽입 알고리즘

알고리즘은 기존의 R-tree와 유사하다. <그림 2>는 R-tree의 삽입 알고리즘을 수정한 것이다. 삽입할 때 먼저 테이블을 보고 연결 노드에 삽입을 한다. pre_insert()에서 연결노드에 삽입을 하고 삽입된 노드가 반환되서 나중에 R-tree부분과 연결된다. pre_insert()에서의 알고리즘은 만약 테이블에 없으면 처음 삽입된 객체의 세그먼트이므로 새로운 연결 노드를 만들어 삽입하고 테이블에 등록한다. 연결 노드가 포화상태인 경우 새로운 노드를 만들어 삽입하고 연결 노드와 연결하고 테이블의 포인터를 수정한다.

4.3 질의 처리 알고리즘

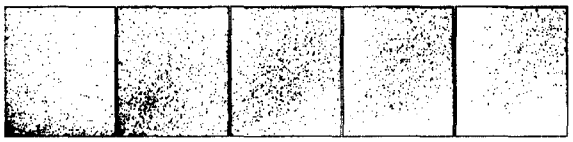
3장에서 언급했듯이 시공간 데이타베이스에서의 질의는 세 가지로 분류할 수 있다. CR-tree의 질의 처리 방법은 기존의 R-tree나 TB-tree와 크게 다르지 않다.

범위 질의인 경우 CR-tree의 윗부분의 R-tree를 이용하므로 R-tree와 질의 처리 방식이 같다. 궤적 질의인 경우 연결 노드를 따라가면 된다. 즉 결합 질의인 경우는 윗 부분의 R-tree를 이용하여 inner range에 속하는 객체를 찾아서 아이디를 저장하고 연결 노드를 이용해 outer range에 속하는 궤적을 찾으면 된다.

5. 성능 평가

5.1 실험 환경

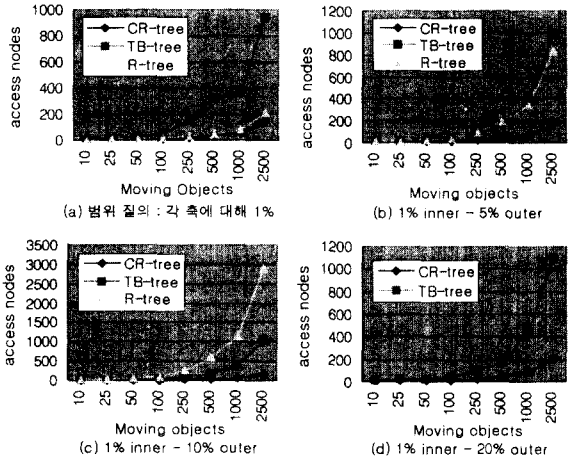
공간 데이타와는 다르게 시공간 데이타는 공개된 것이 없다. 비록 실제 데이타는 아니지만 시공간 데이타베이스에서 많이 사용하는 GSTD(Generate SpatioTemporal Data)를 이용하여 데이타 셋을 만들어 사용하였다. [6]에서 GSTD를 만들고 시공간 데이타 셋에 대한 몇 가지 시나리오를 제공한다. <그림 3>에서 보듯이 데이타 셋은 남서쪽에서 북동쪽으로 이동하는 것을 택했고 하나의 이동 객체당 1500개의 세그먼트를 생성했다.



<그림 3> 데이타 셋 분포와 움직임

실험 조건은 [4]와 같다. 한 페이지의 크기는 1KB이고 R-tree의 비단말노드와 단말노드의 팬아웃은 36, 28이고 TB-tree의 팬아웃은 36,31 CR-tree의 팬아웃은 36, 25 이다. 공간 오버헤드는 CR-tree가 더 크다. 이동 객체가 2500개일 때 CR-tree의 저장 공간 크기는 TB-tree에 비해서 2.4배이고 R-tree에 대해서는 1.3배이다.

5.2 실험 결과



<그림 4> 질의 결과 : 범위 질의(a), 결합 질의{(b), (c), (d)}

<그림 4>의 (a)는 각각의 축에 1% 범위 질의 결과이다. CR-tree는 범위 질의에 대해 R-tree와 비슷한 성능을 낸다. 단말 노드의 팬아웃 차이에 의해서 약간의 차이가 난다. TB-tree는 R-tree보다 단말노드의 MBR이 크거나 같다. 실험 결과 이동 객체의 특성상 앞으로 나가기 때문에 MBR이 상당히 커진다는 것을 알 수 있었다. (a)를 보면 TB-tree는 겹침이 많이 일어나 다른 것보다 접근 노드 수가 훨씬 많다. 객체가 2500개일때는 5배정도 많다. 게다가 객체의 수가 많아질수록 접근 노드수가 많아진다.

<그림 4>의 (b), (c), (d)는 결합 질의에 대한 접근 노드 수를 비교한 것이다. R-tree는 결합 질의의 경우 outer range에 속하는 모든 노드를 방문해야 하기 때문에 접근 노드 수가 매우 많아진다. (d)는 R-tree의 접근 노드 수가 너무 많아 그래프에서 제외했다. (b)는 1% inner-5% outer range를 갖는 결합 질의의 결과이다. R-tree와 TB-tree가 비슷한 성능을 보인다. 왜냐하면 1% inner range 질의에 대해 TB-tree의 접근 노드수가 R-tree보다 훨씬 많지만 5% 궤적 질의를 수행하는 TB-tree가 5% 범위 질의를 해야 하는 R-tree보다 접근 노드 수가 적기 때문에 상쇄되기 때문이다. 즉 TB-tree는 outer range가 커져서 궤적 질의의 범위가 커져야 성능이 향상된다. 하지만 CR-tree는 궤적 질의의 범위가 작아도 성능이 좋고 outer range가 커져도 TB-tree보다 1/5정도의 접근 노드 수를 보였다. 이것은 일단 inner range에 의한 범위 질의에서 효과적일 뿐 아니라 궤적 질의에서도 연결 노드가 TB-tree의 단말노드보다 더 많은 점(세그먼트)을 저장하므로 궤적 질의 처리에도 효과적이기 때문이다. 또한 밀도가 높아질수록 접근 노드의 비율(CR-tree/TB-tree)이 작아진다. 이것은 이동객체의 수가 많은 응용에 매우 적합하다는 것을 말해 준다.

6. 결론 및 향후 연구 방향

본 논문에서는 이동 객체의 움직임을 저장하는 시스템에서 결합 질의를 보다 빠르게 처리할 수 있는 인덱스 구조를 제안하였다. CR-tree는 R-tree부분과 연결 노드를 유지하기 위해 저장 공간이 상대적으로 크다. 하지만 충분한 성능 향상을 가져 오고 특히 이동 객체가 많을수록 유리하다. 이동 객체에 대한 위치 서비스가 대중화되면서 이것을 통한 유용한 정보를 뽑고자 하는 연구가 진행중이다. 여러 가지 이동 객체에 대해 결합 질의가 많이 사용되는 응용에 적용될 수 있다.

향후 연구 과제로는 실제 데이타를 이용한 실험과 실제 시스템을 구현하고자 한다. 그리고 이동 객체의 모델링과 다른 질의에 대한 연구도 많이 필요하다.

참 고 문 헌

[1] Theodoridis, Y., Silva, R., and Sellis, T., "SpatioTemporal Indexing for Large Multimedia Applications", In Proc. of the 3rd IEEE Int'l Conference on Multimedia Computing and Systems, pp. 441-448, 1996.

[2] Tzouramanis, T., Vassilakopoulos, M., and Manolopoulos, "Overlapping Linear Quadtrees: A Spatio-Temporal Access Method", In Proc. of the 6th Int'l Symposium on Advances in Geographic Information Systems, pp. 1-7, 1998.

[3] Nascimento, M., Silva, J., and Theodoridis, Y., "Evaluation of Access Structures For Discretely Moving Points", In Proc. of Int'l Workshop on Spatio-Temporal Database Management, pp. 171-188, 1999.

[4] Pfoser, D., Jensen, C., Theodoridis Y., "Novel Approaches to the Indexing of Moving Object Trajectories", In Proc. of the 26th Int'l Conference on VLDB, pp. 395-406, 2000.

[5] Guttman, A., "R-tree: a Dynamic Index Structure for Spatial Searching", In Proc. of ACM-SIGMOD Conference on the Management of Data, pp. 47-57, 1984.

[6] Theodoridis, Y., Silva, R., and Nascimento, M., "On the Generation of Spatiotemporal Datasets", In Proc. of the 6th Int'l Symposium on Spatial Databases, pp. 147-164, 1999.

[7] Guting R., Bohlen, M., Erwig, M., "A Foundation for Representing and Querying Moving Objects", ACM Transactions on Database Systems, pp. 1-42, 2000.