이동체 데이터베이스 에서 과거, 현재 및 미래위치 질의를 위한 통합 색인

전희철°, 안성우, 김진덕*, 홍봉희
부산대학교 컴퓨터공학과°, 동의대학교 컴퓨터공학과*,
e-mail : {hcjeon°, starnme}@pusan.ac.kr ,jdk@dongeui.ac.kr*, bhhong@pusan.ac.kr

A Unified Index for Querying Past, Current and Future Positions in Moving Object Databases

Heechul Jeon°, Sungwoo Ahn, Jindeog Kim*, Bonghee Hong
Dept. of Computer Engineering. Pusan National University
Dept. of Computer engineering. Dongeui University*

요 약

이동체 색인에 관한 기존 연구는 시간도메인에 따라 과거궤적에 대한 색인과 현재 및 미래위치색인 등으로 구분된다. 그러나 실 세계 응용에서는 과거 궤적뿐만 아니라 현재 및 미래위치를 모두 필요로 하는 경우가 많기 때문에 각 시간도메인 별로 두 개의 색인을 유지해야 하며 이 방법은 불필요한 비용을 필요로 한다. 따라서 과거, 현재, 미래에 대한 위치 데이터를 통합하여 불필요한 비용을 줄인 새로운 색인을 개발할 필요가 있다. 이 논문에서는 이동체의 과거궤적과 현재 및 미래위치데이터를 추상데이터 타입으로써 통합한 새로운 색인을 제안한다.

1. 서론

이동체 데이터베이스 연구는 최근 무선이동통신 기술과 GPS 기술의 발달로 PDA(Personal Digital Assistants), 휴대폰 등의 무선이동기기들이 보편화 됨에 따라 더욱 활발히 진행되고 있다. 이동체 데이터베이스는 사용자들로부터 전송되어 오는 수많은 위치정보를 효율적으로 저장, 검색, 관리할 수 있는 색인연구가 그 핵심이라 할 수 있다.

기존 연구는 대상 시간도메인에 따라 3가지로 분류할 수 있는데 첫째, 과거궤적 검색을 위해 이동체의 위치를 선분(Line Segment)으로 모델링한 3DR-tree[3]와 TB-tree, STR-tree[2]등이 있고 둘째, 위치데이터를 점(Point)로 모델링한 R-tree와 해쉬 기반 색인을 들 수 있고 마지막으로 현재 및 가까운 미래위치 검색을 위해 시간에 대한 선형 함수를 사용하는 TPR-tree[1]와 PMRQuad-tree 등이 있다.

실 세계의 서비스에서 이동체 데이터베이스는 다양한 질의 타입을 제공하며 대표적인 질의 타입으로 영역질의[3]를 들 수 있다. 영역질의는 다시 질의시간 범위에 따라 과거 혹은 현재 그리고 미래에 대한 질의로 나눌 수 있고 각각 다양한 서비스에 사용되고 있다. 이동체 색인의 기존연구에는 이러한 세가지 시간 도메인에 대한 질의를 모두 제공하는 예가 없으며 세 가지 시간 도메인에 대한 질의를 제공하는 서비스를 제공하기 위해서는 두 개 이상의 색인을 별도로 유지해야 한다. 예를 들어 "지금부터 5 분전까지 서울역 광장 앞을 지난 차량을 검색하라" 와 같은 질의에 응답하기 위해서는 3DR-tree 와 같은 과거위치 색인과 TPR-tree 와 같은 현재위치 색인에 대한 접근이 필요하다. 그러나 이와 같은 방법은 비효율적인 노드탐색 경로와 두 개의 색인을 유지함에 따른 별도의 부가비용을 필요로 하는 문제가 있기 때문에 과거, 현재, 미래 시간에 대한 질의를 제공함으로써 이러한 문제로 인한 손실을 줄인 새로운 색인을 개발할 필요가 있다.

이 논문에서는 이동체의 과거위치를 표현하는 선분과 현재위치정보를 나타내는 시간에 대한 선형함수를 통합하여 하나의 색인에 유지 함으로써 두 개의 색인을 유지할 때 발생하는 불필요한 노드탐색과 부가비용이 발생하는 문제를 해결한 새로운 색인을 제안한다.

2장에서 관련연구에 대해 조사하고 3장에서는 과거궤적을 위한 색인과 현재 및 미래위치를 위한 색인을 별도로 유지할 때의 문제점에 대해 살펴본다. 4장에서는 이 논문에서 제안하는 색인에 대해 설명하고 5장에서 성능평가를 수행한 후 6장의 결론으로 끝을 맺는다.

2. 관련연구

STR-Tree, TB-Tree[2] 는 이동체의 궤적을 선분으로 표현한 색인으로 R-Tree의 변형이다. 이 색인은 현재위치 검색이 고려되지 않았고 궤적을 보호하기 위해 공간근접성을 고려하지 않기 때문에 영역질의 성능이 좋지 않다. 3DR-Tree[3]는 R-Tree를 3차원으로 확장한 구조로써 역시 현재위치 검색이 고려되지 않았고 분할 시 시간도메인의 특성에 대해 고려되지 않았기 때문에 공간 효율성이 떨어진다.

이동체의 과거궤적 및 현재 위치에 대한 기존 연구로 2+3DR-Tree[6]가 있다. 2DR-Tree를 사용하여 현재 위치를 점 객체로 표현하고 3DR-Tree를 이용하여 과거궤적을 선분으로 표현하는 방법이지만 이 구조에서 새로운 위치보고가 있을 때 2DR-Tree에서 삭제와 삽입 연산이 발생하고 3DR-Tree에서 다시 삽입연산이 발생하므로 삽입에 많은 부하가 걸리고 2DR-Tree의 현재위치 색인에서는 이동체가 최

근 보고한 위치를 현재 위치라고 정의하고 있으나 이는 실제 현재위치와 오차가 크다고 할 수 있다. 또한 과거 궤적에 대한 3DR-Tree의 경우 위에서 설명했던 시간도메인에 대한 고려가 없다는 단점을 그대로 가진다.

TPR-Tree[1]는 이동체를 시간에 대한 선형함수로 표현하는 색인구조이다. 이동체의 위치를 단순한 좌표로 저장하지 않고 방향과 속도를 나타내는 벡터를 저장하여 특정 임계값 이하의 변화일 경우 업데이트 하지 않음으로써 업데이트 횟수를 획기적으로 줄였고 시간에 대한 선형 함수를 사용하여 현재 및 가까운 미래위치에 대한 검색이 가능한 구조이지만 과거궤적에 대한 고려가 없다는 단점이 있다.

3. 두 개의 색인 유지 시의 문제점

이동체의 과거위치와 현재 및 미래위치를 위해 별도의 색인을 유지하는 경우 "5분 전부터 현재까지 서울역 광장 앞을 통과한 차량을 찾아라" 와 같이 동시에 두 색인에 대한 접근이 필요한 질의가 요청되면 [그림 1]처럼 각 색인에 대한 노드탐색 비용과 그 결과를 병합하는 부가적인 비용을 필요로 하게 된다.

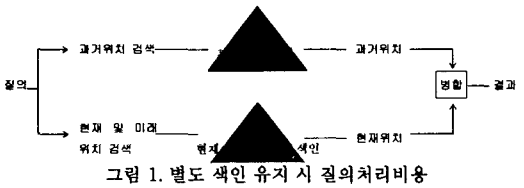

그림 1. 별도 색인 유지 시 질의처리비용

시간이 지남에 따라 현재위치 색인은 과거위치 색인에 대해 상대적으로 크기가 작아지지만 색인의 깊이는 크게 차이가 나지 않으므로 ($N = P^D$, N : 노드개수, P : 팬아웃, D : 깊이) 질의처리 시 두 색인에 대해 루트노드로부터 탐색하는 것은 효율적이지 못한 방법이다. 또한 이동체의 과거위치와 현재위치는 연결되어 있고 서로간에 공간근접성이 존재하므로 이 두가지 데이터를 각각의 색인에 유지하게 되면 이러한 공간근접성을 이용할 수 없게 되고 질의, 삽입등의 처리시 불필요한 I/O를 필요로 하는 문제가 있다. 따라서 과거궤적 데이터와 현재 및 미래위치데이터를 통합한 새로운 색인을 개발할 필요가 있다.

4. HCR-Tree(History and Current R-tree)

4 장에서는 이 논문이 제안하는 새로운 색인에 대해 용어정의, 과거데이터와 현재 및 미래를 나타내는 선형함수를 통합한 추상데이터 타입 E 와 함께 색인의 구조를 설명한다.

4.1 용어정의

이 논문에서는 현재 및 미래위치를 표현하기 위해 시간에 대한 선형함수[1]를 사용하고 과거위치를 저장하기 위해 선분을 사용한다. 이 두 가지 형태의 위치정보를 하나의 색인에 유지하기 위해서 다음과 같이 시간과 위치정보 형식을 정의한다.

[정의 1]
T_{uc} : 이동체의 최근 보고시간
Closed-line(CL) : 과거궤적에 해당하는 선분
$(x_1, y_1, t_1, x_2, y_2, t_2)$
Opened-line(OL) : 이동체의 이동속성을 나타내는 벡터
(x, y, v_x, v_y, t)
Bounding rectangle(BR) : 노드 내 혹은 하위 노드들의 *CL*과 *OL*을 포함하는 시간에 대해 동적인 영역
$(BR_T, BR_B, BR_L, BR_R, v_T, v_B, v_L, v_R, t_1, t_2)$

CL 은 3 차원상의 두 점을 연결한 선분으로써 이동체의 과거궤적을 표현하고 OL 의 x, y 는 T_{uc}시간의 이동체의 위치를 나타내고 v_x와 v_y는 이동체의 이동속성을 표현한다. 이 논문에서 제안하는 색인의 한 노드는 OL 과 CL 을 동시에 저장해야 하므로 [1]에서 제시한 TPBR 을 [정의 1]의 BR 과 같이 재정의 한다. 즉 TPBR 과는 달리 이 논문에서 제시하는 BR 은 OL 과 CL 모두를 포함할 수 있다. [그림 2]는 BR 이 시간이 지남에 따라 OL 의 이동방향과 속도에 대해 확장되는 것을 보여준다. 예를 들어 BR_T는 노드내의 OL 들의 v_y중 가장 큰 값을 가지는 v_T의 속도로 시간이 흐름에 따라 확장하고 나머지 BR_L, BR_R, BR_B 등도 같은 원리에 따라 확장한다. BR 의 t_1 은 노드 내에 있는 이동체의 가장 오래된 보고시간으로 설정되고 t_2는 삽입 혹은 질의발생시간으로 결정된다.

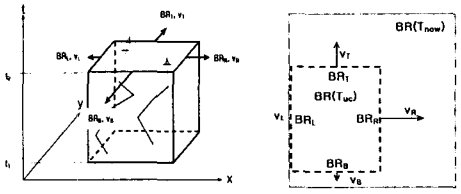

그림 2. Bounding rectangle

4.2 추상데이터 타입

CL 과 OL 은 서로 표현이 다를 뿐만 아니라 필요로 하는 저장 공간도 다르다. 그러나 이동체들이 위치를 보고함에 따라 전체 색인에서 OL 의 비중이 줄어들게 되므로 필요로 하는 저장공간의 차이로 인한 손실은 극히 작아지게 된다.

OL 과 CL 의 표현의 차이에 대해 이 논문에서는 OL 과 CL 을 통합하는 추상데이터 타입 E 를 다음과 같이 정의함으로써 해결한다.

[정의 2]
T_{now} : 현재 시간(질의 혹은 삽입시간)
*E(OL) = (x, y, x + (vx * tnow − t), (vy * tnow − t), t, tnow)*
E(CL) = (x1, y1, x2, y2, t1, t2)
E(BR) = (BRT + vT(tnow − tc), BRB + (tnow − tc), BRL + (tnow − tc) . BRR + (tnow − tc). tc. t)

[정의 2]는 [정의 1]의 3 가지 데이터 타입이 추상데이터 타입 E 로 통합 될 수 있음을 보여준다. 즉 동적 데이터(OL, BR)를 T_{now}에 대한 정적 데이터로 변환함으로써 모든 데이터를 정적 데이터 E 로의 참조가 가능하게 된다.

4.3 색인구조

앞 절에서는 CL, OL, BR 등의 데이터가 E 라는 통합된 데이터타입으로 참조될 수 있음을 보였다. 4.3 절에서는 CL, OL 등을 색인 내에서 유지하는 두 가지 방법에 대해 설명한다.

4.3.1 OL 과 CL 을 같은 노드에 저장

3 장에서 언급하였듯이 이동체의 과거위치와 현재위치 간에는 공간관련성이 존재한다. OL 과 CL 을 공간 근접성에 근거하여 한 노드에 저장하는 방법은 이러한 특징을 최대한 활용하는 방법이다.

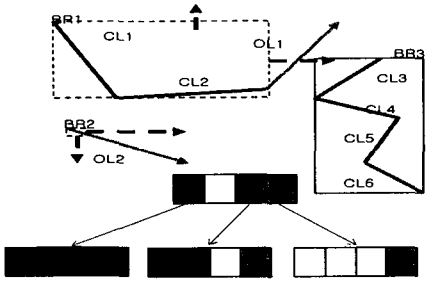

그림 3. OL 과 CL 을 한노드에 저장

[그림 3]은 OL 과 CL 에 따른 색인구조를 보여준다. CL1, CL2, OL1 은 하나의 노드에 포함되어 있고, 이 노드의 BR 인 BR1 은 OL1 의 속도와 방향에 따라 확장된다. BR2 는 OL 만 저장된 경우이고 BR3 은 CL 만으로 구성된 노드를 보여주며, 이 때 BR 은 확장되지 않는다.

이 구조에서 삽입이 발생하면 동일 이동체의 직전 위치 검색하여 OL 을 CL 로 변환(Logical update) 한 후[그림 4] 새로운 OL 을 삽입하게 된다.

[그림 4]에서 볼 수 있듯이 직전위치에 해당하는 OL 이 CL 로 변환되는 것은 동일 노드에서 이루어 진다.

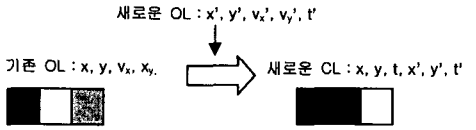

그림 4. Logical update

4.3.2 OL 과 CL 을 구분하여 다른 노드에 저장

OL 과 CL 을 하나의 색인에 유지하는 두번째 방법으로써 OL 이 저장된 노드를 색인의 우측에 구성하는 방법이 있다. 즉 과거위치색인과 현재 및 미래위치 색인이 루트노드를 공유하는 형태가 된다. 이 구조는 OL 과 CL 의 공간근접성을 무시하고 과거, 현재 및 미래에 대한 개별적인 질의처리에 초점을 맞춘 구조이다.

4.3.3 질의처리 시의 노드 탐색

앞절에서 OL 과 CL 을 통합하는 두가지 방법에 대해 설명하였다. 이 절에서는 이러한 구조에서 질의처리 시 노드탐색 경로가 어떻게 되는지 살펴본다. 이 논문에서 주로 관심을 가지는 질의를 [정의 3]과 같이 정의한다.

[정의 3]
Range Query : $Q = (R, T_{q1}, T_{q2})$
R : 공간상의 질의영역 (x_1, y_1, x_2, y_2)
T_{q1}, T_{q2} : 시간도메인 상의 질의영역
a. $(T_{q1} < T_{q2} < T_{uc})$
b. $(T_{q1} < T_{uc} < T_{q2})$
c. $(T_{uc} < T_{q1} < T_{q2})$

[정의 3]의 a 는 질의가 요구하는 정보가 모두 과거의 위치정보라는 것을 의미한다. 4.3.1 에서 제시한 방법의 경우 CL 들만 저장된 노드뿐만 아니라 OL 과 CL 이 함께 저장된 노드도 탐색해야 하지만 4.3.2 의 방법에서는 OL 과 CL 이 저장된 노드가 각각 분리되어 있으므로 불필요한 노드탐색은 없다. [정의 3]의 b 는 질의가 요청하는 정보가 과거와 현재 및 미래 위치데이터라는 것을 의미한다. 즉 색인 내의 OL 과 CL 이 모두 참조되어야 하며 이때 4.3.1 의 방법은 공간근접성을 보존하였으므로 4.3.2 의 방법보다 노드탐색 경로가 짧아지게 된다. [정의 3]의 c 는 질의 대상이 현재위치 및 예측된 미래위치 일 경우이며 이때는 OL 들이 한곳에 집중된 구조인 4.3.2 의 방법이 4.3.1 의 방법처럼 OL 이 전체 색인에 흩어져 있는 경우보다 효율적이다.

노드내에 OL과 CL이 섞여있는 경우 질의에서 요구하는 시간 범위에 따라 필요한 엔트리만을 비교하게 되며 CL들만 저장된 노드인 경우는 [그림2]의 우측의 $E(BR(T_{uc}))$(확장하지 않는 BR)로써 참조된다.

5. 성능평가

5.1 실험데이터

HCR-tree 는 이동체의 위치를 OL 의 형태로 보고한다. 따라서 GSTD(Generating Spatio-Temporal Data)를 사용하여 생성한 데이터를 OL 의 형태로 가공하여 사용한다. 생성된 데이터는 주기적으로 응집과 확산을 반복하며 각각 2000 번의 보고 횟수를 가진다.

5.2 질의성능평가

[그림 5]는 이동체 수가 증가함에 따른 영역질의 시 필요한 디스크 I/O 를 보여준다. 그림에서 [1]은 4.3.1 에서 설명한 방법이며 [2]는 4.3.2 의 방법을 구현한 것이다. 공간근접성을 고려한 [1]의 방법이 질의의 시간 범위가 T1 < Tun < T2 인 경우 [2]의 방법보다 좋은 성능을 나타냈으며 과거, 현재 및 미래에 대한 질의를 개별적으로 수행 했을 때는 [2]의 방법이 [1]의 방법보다 성능이 좋은 것을 볼 수 있다.

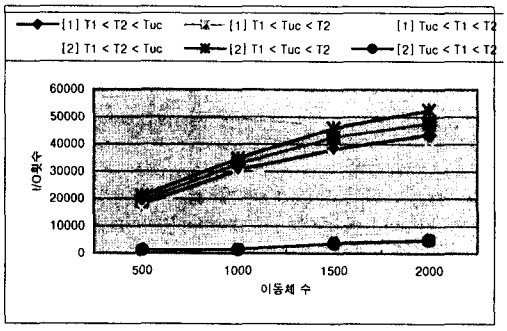

그림 5. 이동체 수에 따른 영역질의성능

6. 결론

이 논문은 과거위치 색인과 현재 및 미래위치 색인을 추상 데이터 타입 E 를 통해 통합하여 모든 시간영역에 대한 영역질의를 지원하는 새로운 색인인 HCR-tree 를 제안하였다. 또한 색인을 구성하는 두 가지 방법에 대해 설명했으며 구현 및 실험을 통하여 질의의 시간영역에 따라 제안한 두 가지 구조가 다른 성능을 나타내는 것을 보였다.

7. 참고문헌

[1] S. Saltenis, C. S. Jensen, S.T. Leutenegger, and M. A. Lopez," Indexing the Positions of Continuously Moving Objects," , In Proc. ACM SIGMOD on Management of data, p331 - 342, 2000.

[2] 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.

[3] Yannis Theodoridis, " Spatio-Temporal Indexing for Large Multimedia Applications" , In Proc. Of the 3rd IEEE Conf. on Multimedia Computing and Systems, pages 441-448, June 1996

[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 1990; 322-331

[5] Betty salzberg, vassilis j. tsotras, " Comparison of Access Methods for Time-Evolving Data" ACM Computing Surveys, Vol.31, No.2, pp.158-221, 1999

[6] Mario A. Nascimento, Jefferson R. O. Silva, Yannis Theodoridis" Evaluation of Access Structures for Discretely Moving Points." Spatio-Temporal Database Management,171-188 1999