대용량 이동체의 색인을 위한 시간축 분할 프레임웍의 설계 및 구현

조대수[O+] 임덕성[++] 박종현[+]
[+]한국전자통신연구원, 공간정보기술센터, GIS 연구팀
{junest[O], jhp}@etri.re.kr
[++]부산대학교 컴퓨터공학과
dsleem@pusan.ac.kr

Design and Implementation of Time Division Framework for Indexing Numerous Moving Objects

Dae-Soo Cho[O+] Duk-Sung Lim[++] Jong-Hyun Park[+]
[+]GIS Research Team, Spatial Information Technology Center, ETRI
[++]Dept of Computer Engineering, Pusan National University

요 약

이동체 데이터베이스에서는 대용량의 위치정보를 효과적으로 저장, 검색하기 위해 이동체 색인을 이용한다. 이동체 색인은 시간에 지남에 따라 검색 영역이 증가하고, 관리해야 하는 위치정보의 양이 커지게 되므로, 삽입, 검색, 삭제 연산의 성능이 계속해서 저하되는 문제가 있다. 이 논문에서는 기존의 이동체 색인을 시간축으로 분할하여 관리하기 위한 시간축 분할 프레임웍을 제안하고, 기존의 TB-tree 및 STR-tree에 대해서 제안한 프레임웍에 적용하였다. 시간축 분할 프레임웍은 전체 시간 도메인에 대해서 하나로 구성되는 색인을 시간 구간별로 쪼개어서 여러 색인으로 구성하여 관리함으로서, 위치정보의 삽입 및 검색 비용을 줄일 수 있으며, 오래된 위치정보에 대한 삭제 연산을 효과적으로 처리할 수 있다.

1. 서론

이동체(Moving Objects)란 시간에 따라 자신의 위치정보가 계속해서 변하는 객체를 의미한다. 이동체의 위치정보는 시간에 따라 계속해서 시간축 도메인이 확장되는 특징과 위치정보의 양이 증가하는 특징을 갖는다. 따라서, 위치정보를 GIS에서 의미하는 공간정보로 다루기 어렵기 때문에 기존의 공간 색인을 그대로 적용하지 못한다. 최근에는 이러한 위치정보를 효과적으로 저장, 검색하기 위해 다양한 종류의 이동체 색인에 대한 연구가 활발히 진행되고 있다.

그런데, 현재 연구되고 있는 대부분의 이동체 색인은 하나의 트리로 구성되는 전체 색인 방식을 따르고 있다. **전체 색인 방식**이란 이동체의 위치가 기록되기 시작한 시각부터 현재까지의 모든 위치정보가 하나의 색인으로 관리되는 것을 의미한다. 이 논문에서는 이동체의 현재 위치만을 다루는 색인은 논의의 대상에서 제외한다. 전체 색인 방식에서는 시간이 지남에 따라서 트리의 깊이(depth)가 깊어지기 때문에 삽입 및 검색(전체 영역에 대한 검색이 아닌 비교적 작은 영역의 질의인 경우) 속도가 저하되는 문제점이 있다. 그리고, 오래된 위치정보에 대해서 삭제 연산을 할 경우에 색인의 전체 구조에 영향을 주기 때문에, 삭제 연산을 처리하기 어려운 문제점이 있다. 오래된 위치정보에 대한 삭제 연산은 다음과 같은 경우에 발생한다. 첫째, 시간이 지남에 따라서 색인의 크기가 무한히 증가하는 것을 방지하기 위해서 가장 과거의 위치정보를 삭제할 수 있다. 둘째, 실제 응용에서 사용되는 위치정보는 저장되어야 할 유효 기간을 갖는데, 유효 기간이 지난 위치정보는 시스템의 성능을 위해서 삭제하거나 백업될 수 있다.

이 논문에서는 전체 색인 방식의 문제점을 해결하기 위해서 **시간 세그먼트 색인 방식**을 제안한다. 이 논문은 새로운 이동체 색인을 제안하는 것이 아니라, 기존의 이동체 색인을 시간 구간별로 저장, 관리하기 위한 **시간축 분할 프레임웍**(TDF, Time Division Framework)을 제안하는 것을 목적으로 한다. 시간축 분할 프레임웍에서 하나의 색인은 시간 구간별로 쪼개어서 여러 색인으로 구성되기 때문에 다음과 같은 장점을 갖는다. 첫째, 시간 구간별로 쪼개어진 **시간 세그먼트 색인**(TSI, Time-Segmented Index)은 전체 시간 구간에 대해서 하나로 구성된 **전체 연결 색인** (WCI, Whole-Connected Index)보다 트리의 깊이가 얕기 때문에, 삽입 및 검색 속도가 높아지는 장점이 있다. 둘째, 유효 기간이 지난 오래된 위치정보에 대해서는 해당 시간 간격대의 TSI를 삭제함으로서 삭제 연산을 효과적으로 처리할 수 있다. 하나의 TSI의 삭제는 다른 시간 간격대의 TSI의 구조에 영향을 주지 않으므로, 비교적 간단한 연산으로 구현될 수 있다.

이 논문의 구성은 다음과 같다. 2장에서는 이동체 색인에 대한 관련연구를 살펴본다. 3장에서 시간축 분할 프레임웍을 자세히 설명하고, 3장에서 간단한 성능평가 결과를 설명한다. 마지막으로 6장에서 결론 및 향후 연구 과제를 설명한다.

2. 관련연구

기존의 GIS에서 공간객체에 대한 색인은 크게 Grid File, Quad-tree와 같은 공간 중심 색인과 R-tree 계열[1,5]의 데이터 중심 색인으로 구분된다. 그런데, 이동체 데이터베이스에서는 시간 도메인이 계속해서 증가하기 때문에, 공간 중심 색인은 사용할 수 없으며, 대부분 R-tree 계열의 데이터 중심 색인을 확장하여 사용하고 있다.

과거의 위치정보를 다루는 이동체 색인의 종류는 효과적으로 처리할 수 있는 이동체 질의 연산의 종류에 따라 구분된다. 이동체 질의 연산의 종류에는 크게, 영역 질의 연산과 궤적 질의 연산이 있다. 영역 질의 연산은 주어진 시공간 영역에 대해서 포함되거나 겹치는 이동체를 검색하는 연산으로, 3DR-tree[4], HR-tree[3], MV3R-tree[6] 등을 통해 효과적으로 처리된다. 궤적 질의 연산은 특정 이동체에 대해서 주어진 시간 간격동안의 궤적을 추적하는 연산으로, TB-tree[2], STR-tree[2] 등을 통해 효과적으로 처리된다.

3. 시간축 분할 프레임웍

기존의 이동체 색인은 모두 전체 색인 방식으로 색인을 관리하기 때문에, 시간이 지남에 따라서 색인의 성능이 저하되는 문제점을 갖고 있으므로, 이 논문에서 제안하는 시간 세그먼트 색인 방식에 따라서 이동체 색인을 관리할 필요가 있다. 이 장에서는 전체 색인 방식의 문제점과 이를 해결하기 위한 시간 세그먼트 색인 방식에 대해 상세히 설명한다.

3.1 전체 색인 방식의 문제점

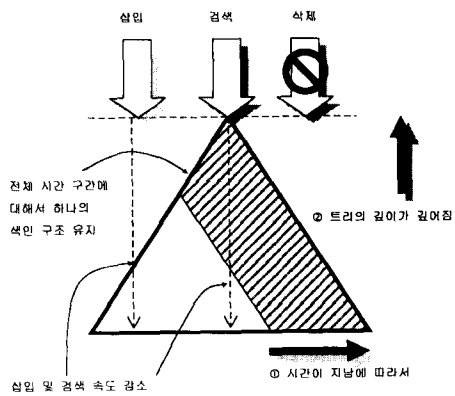


그림 1 전체 색인 방식

그림 1은 주어진 이동체 색인에 대해서 전체 색인 방식으로 관리하는 것을 보이고 있다. 이 논문에서 대상으로 하는 이동체 색인은 현재뿐아니라 과거의 위치정보를 포함하는 색인으로 R-tree 계열의 공간 색인을 확장한 구조이다. 따라서, 이동체 색인은 R-tree의 특징인 균형이 잡힌(height-balanced) 색인 구조를 갖는다.

전체 색인 방식에서는 그림에서 보이는 것과 같이 이동체의 위치가 기록되기 시작한 시각부터 현재까지의 모든 위치정보를 하나의 색인 구조에서 관리하고 있다. 따라서 ①시간이 지남에 따라서 ②트리의 깊이가 깊어지게 되므로, ③삽입 및 검색 연산에 대해서 속도가 감소되는 단점을 갖는다.

그리고, R-tree 계열의 색인에서 삭제 연산은 삭제되는 엔트리가 포함된 노드 N의 엔트리 수가 최소 엔트리 수(m)보다 적게 될 경우에 트리의 재구성(reorganization) 연산을 필요로 한다. 트리의 재구성은 노드 N에 남아있는 m-1개의 엔트리를 재삽입(reinsert)하는 방식으로 이루어지므로, 높은 비용이 발생한다. 따라서, 관리되는 모든 이동체에 대해서 일정 시간 구간에 포함된 모든 위치정보를 삭제하는 연산은 전체 트리의 구조에 영향을 주기 때문에, 실제적으로 처리하기 어려운 문제가 있다. 왜냐하면, 이동체 색인은 계속해서 새로운 위치정보가 삽입되어야 하므로, 삭제 연산을 수행하기 위해서 일정 시간동안 색인에 대한 삽입 연산을 금지할 수 없기 때문이다. 특히, 오랜 시간동안의 위치정보가 축적되어 있는 상황이라면, 더욱 색인의 재구성은 시간이 많이 걸리게 된다.

3.2 시간축 분할 프레임웍의 구조

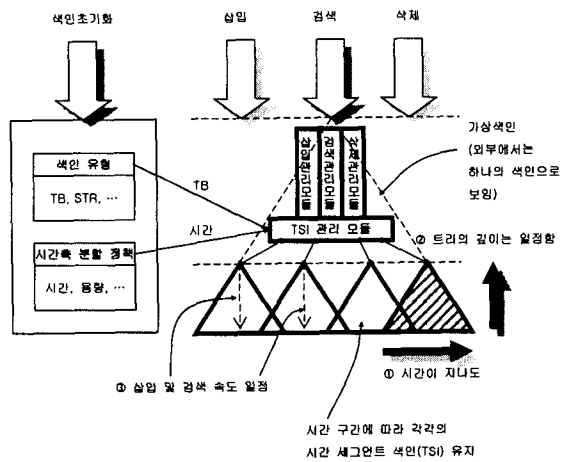


그림 2 시간 세그먼트 색인 방식 (시간축 분할 프레임웍)

그림 2는 주어진 이동체 색인을 시간 세그먼트 색인 방식으로 관리하기 위한 시간축 분할 프레임웍의 구조을 보이고 있다. 시간축 분할 프레임웍에서 색인은 (색인 유형, 시간축 분할 정책)을 매개변수로 색인 초기화에 의해 생성된다. 이때, 생성된 색인은 가상 색인(Virtual Index)으로서 전체 색인 방식에서와 같이 외부에서는 하나의 색인으로 보이게 된다. 가상 색인은 (시간 세그먼트 색인 집합(TSIS), TSI 관리 모듈, 삽입 관리 모듈, 검색 관리 모듈, 삭제 관리 모듈)로 구성된다.

시간 세그먼트 색인 집합 TSIS(TSI Set)은 시간 구간 [tx,ty)별로 쪼개어진 시간 세그먼트 색인 TSI_{tx}의 집합을 의미한다. TSI 관리 모듈은 TSIS을 관리하며, 주어진 시간 t(단, tx≤t<ty, [tx,ty)는 TSI 시간 구간임)에 대해서 TSI_{tx}를 넘겨주는 매핑 함수 $f_{tsi}(t)$를 가지고 있다(그림 3 참조). 따라서, 삽입, 검색, 삭제 관리 모듈로부터 주어진 시간 구간 [ta,tb)에 대해서 겹치는 구간의 시간 세그먼트 집합 $TSIS_{[ta,tb)}=\{f_{tsi}(ta)=TSI_{tl}, f_{tsi}(tb)=TSI_{tn}\}\cup\{TSI_{tm} \mid TSI_{tm}\subset TSIS,\ tl<tm<tn\}$을 매핑 함수를 통해 넘겨줄 수 있다(그림 3 참조). TSI 관리 모듈은 주어진 시간 구간에 겹치는 시간 세그먼트 집합을 매핑 함수를 사용해서 계산하기 때문에 추가의 디스크 접근이 발생하지 않는다. 따라서, 전체적으로 색인의 깊이를 줄여 삽입 및 검색 시간을 단축할 수 있다. 또한, TSI 관리 모듈은 색인의 재구성 비용 없이, 가장 오래된 TSI_{oldest}를 삭제함으로서 이동체 색인에서 관리되는 모든 이동체에 대해서 일정 시간 구간에 포함된 모든 위치정보를 삭제하는 연산을 지원할 수 있다.

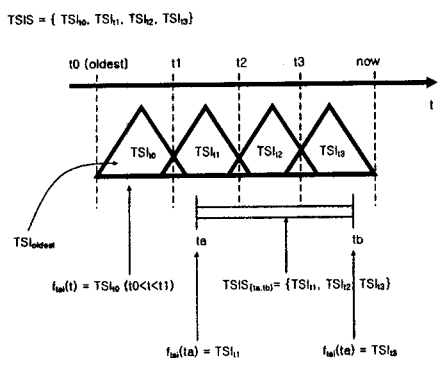


그림 3 시간 세그먼트 색인 집합 및 매핑 함수 예

시간축 분할 프레임웍에서 주어진 이동체 색인을 시간 세그먼트 색인으로 분할하기 위해서는 시간축의 분할 정책이 필요하다. 즉, 시간 세그먼트 색인으로 분할하기 위한 시간 구간을 결정하는 방법이 필요하다. 시간축 분할 정책으로는 고정 시간 간격 방식과 고정 색인 크기 방식이 있다. 이 논문에서는 매핑 함수를 간단하게 구하기 위해서 고정 시간 간격 방식을 사용한다.

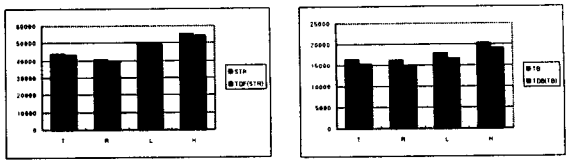
고정 시간 간격 I_{td}은 주어진 이동체 색인 X에 대해서 시간 분할 색인의 깊이가 d가 넘지 않는 최대의 시간 간격으로 설정한다. 이동체 색인 X의 단말노드의 최대 엔트리 수를 n(e), 최소 공간 활용도(space utilization)를 u(X), 관리될 이동체의 개수를 n(mo), 이동체의 단위 시간당 위치 보고 횟수를 n(r)이라고 하자. 이때, 색인의 깊이가 d인 경우 최대 저장 가능한 엔트리 개수 M는 $u(X)*n(e)^d$이고, 임의의 고정 시간 간격 I_{td}동안 보고되는 위치의 개수 L은 $n(mo)*n(r)*I_{td}$이 된다. 따라서, L≤M를 만족하는 최대의 I_{td}는 $floor((u(X)*n(e)^d)/(n(mo)*n(r)))$이다. 단, floor(x)는 x를 넘지 않는 최대의 정수를 의미한다.

그림 2그림 3에서 각각의 TSI가 서로 겹치는 이유는 분할되는 시간축에 대해서 겹치는 이동체의 궤적정보를 중복해서 저장하기 때문이다. 즉, 시간 구간 [t0,t1)를 대표하는 TSI_{t0}와 시간 구간 [t1,t2)를 대표하는 TSI_{t1}에는 분할되는 시간축 t1과 겹치는 이동체의 궤적정보를 중복해서 저장하고 있다. 따라서, 매핑 함수 $f_{tsi}(t)$ (단, tx≤t<ty, tx-ty는 TSI 시간 구간임)에 의해 계산되는 TSI_{tx}는 시간 구간 (tx-ty)에 겹치는 모든 이동체의 위치정보를 포함하게 된다. 삽입, 검색, 삭제 관리 컴포넌트는 가상 색인의 외부에서 요청되는 삽입, 검색, 삭제 연산을 매핑함수와 TSIS을 이용하여 실제로 구현하는 부분이다.

4. 성능평가 실험

이 논문에서는 TB-tree 및 STR-tree에 대해서 시간축 분할 프레임웍을 적용하여, 전체 색인 방식과의 성능을 비교한다. 각 색인의 노드를 위한 페이지는 1024 Kbyte의 크기를 갖는다. 실험을 위한 데이터는 GSTD[7]를 이용하여 생성되었으며, [7]에서 제시한 6가지 매개변수 조합중에서 시나리오 1(T, 이동체들이 왼쪽에서 오른쪽으로 이동하는 분포), 시나리오 2(R, 초기 가우시안 분포에서 북동쪽으로서 퍼지는 분포), 시나리오 5(L, 초기 가우시안 분포에서 느린 속도로 전체 영역으로 퍼지는 분포), 시나리오 6(H, 빠른 속도로 이동하는 분포)을 사용한다.

그림 4는 짧은 시간 간격 질의(공간 조건은 전체 범위임)를 수행할 경우 색인의 페이지 접근 횟수(1000번의 질의에 대한 평균값)를 보이고 있다. 짧은 시간 간격 질의에 대해서는 대부분의 경우에 시간축 분할 프레임웍에서 1.1%~8.3%까지의 페이지 접근이 감소함을 보이고 있다. 그러나, 질의의 시간 간격이 길어질 경우에는 시간 세그먼트 색인간에 중복 저장된 이동체의 궤적정보로 인해 페이지 접근이 증가함을 보였다.



(a) STR-tree 비교 (b) TB-tree 비교

그림 4 짧은 시간 간격 질의에 대한 색인의 페이지 접근

5. 결론 및 향후 연구과제

이 논문에서는 대용량 이동체의 색인을 위한 시간축 분할 프레임웍과 시간 세그먼트 색인을 구성하기 위한 시간축 분할 방법을 제안하였다.

이 논문에서 제안한 시간축 분할 프레임웍의 특징은 다음과 같다. 첫째, R-tree 계열의 데이터 분할 이동체 색인에 대해서 일반적으로 적용할 수 있다. 이 논문에서는 TB-tree 및 STR-tree를 시간축 분할 프레임웍에 적용하였다. 둘째, 시간축 분할 프레임웍을 통해 이동체의 위치 삽입, 검색 속도를 개선할 수 있으며, 전체 색인의 재구성 연산 없이 삭제 연산을 구현할 수 있으므로, 다양한 이동체 색인을 실제 응용에서 활용할 수 있는 기반을 마련하였다. 향후, 시간 분할 프레임에서 최적의 시간 세그먼트 색인 집합을 구하기 위한 시간축 분할 방법에 대한 연구와 다양한 데이터셋, 이동체 색인에 대한 성능 평가 실험이 필요하다.

참고문헌

[1] A, Guttman, "R-trees: A dynamic index structure for spatial searching," ACM SIGMOD Conference, p47-54, 1984

[2] Dieter Pfoser, Christian S. Jensen, and Yannis Theodoridis, "Novel Approaches in Query Processing for Moving Object Trajectories," VLDB 2000, 395-406

[3] M. A. Nascimento and J. R. O. Silva, "Towards historical R-trees," ACM SAC, 1998

[4] Michalis Vazirgiannis, Yannis Theodoridis, and Timos K. Sellis, "Spatio-Temporal Composition and Indexing for Large Multimedia Applications," Multimedia Systems 6(4), 284-298 (1998)

[5] N. Beckmann and H. P. Kriegel, "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles," In Proc. ACM SIGMOD, pp. 332-331, 1990

[6] Y. Tao and D. Papadias, "MV3R-Tree: A Spatio-Temporal Access Method for Timestamp and Interval Queries," VLDB, 2001

[7] Yannis Theodoridis, Jefferson R. O. Silva and Mario A. Naschimento, "On the Generation of Spatiotemporal Datasets," CHOROCHRONOS Technical Report CH-99-01, Proceedings of the 16th Int'l Symposium on Spatial Databases (SSD), 1999

[8] Zhexuan Song 1 and Nick Roussopoulos, "Hashing Moving Objects," MDM 2001, LNCS 1987, pp. 161-17, 2001