이동 객체의 미래 위치 검색을 위한 시공간 색인 구조

서동민, 복경수, 유재수
충북대학교 정보통신공학과
{dmseo, ksbok}@netdb.chungbuk.ac.kr
yjs@cbucc.chungbuk.ac.kr

Spatio-Timporal Index Structure for Retrieving Future Positions of Moving Objects

Dong Min Seo, Kyoung Soo Bok, Jae Soo Yoo
Dept. of Computer and Communication Engineering, Chungbuk National University

요 약

최근 위치기반 기술의 급속한 발전으로 인하여 이동 객체를 효율적으로 관리하기 위한 색인 구조의 필요성이 증가하고 있다. 본 논문에서는 KDB-트리를 기반으로 하는 새로운 형태의 시공간 색인 구조인 TPKDB-트리 (Time Parameterized KDB-Tree)를 제안한다. 제안하는 색인 구조는 갱신 비용을 최소화하여 이동 객체 검색의 효율성을 증가시키고 노드 내에 포함되어 있는 이동 객체의 변화를 시간에 대한 파라미터로 유지함으로서 효율적으로 이동 객체의 미래 위치 검색을 지원한다. 또한, 공간활용도를 최대화하기 위해 EFP 분할 (Enhanced First Division Splitting) 기법을 제안한다. 제안하는 색인 구조의 우수성을 입증하기 위해 실험을 통해 다른 색인 구조와의 성능 비교를 수행한다.

1. 서론

최근 정보통신사업, 위성사업 그리고 지능형 교통 시스템 (ITS : Intelligent Transport System) 사업 등의 위치 기반 서비스에 대한 급속한 발전으로 이동 객체의 위치를 모니터링 할 수 있게 되면서 위성 위치 확인 시스템 (GPS : Global Positioning System)과 같은 위치기반 서비스를 제공하는 응용 서비스들이 여러 분야에서 광범위하게 제공되고 지속적으로 움직이는 이동 객체의 위치를 추적하고 관리하기 위한 연구들을 필요로 하게 되었다.

이동 객체 데이터베이스 (Moving Object Databases : MOD)에서는 동적인 속성을 가지는 이동 객체에 대한 데이터 타입을 과거 데이터, 현재 데이터 그리고 미래 데이터로 구분한다. 지금까지는 이동 객체의 과거와 현재 위치에 대한 데이터를 관리하는데 중점을 두고 많은 연구가 진행되었으며, 최근에 이동 객체의 미래 위치에 대한 데이터 관리의 중요성이 증가하였고 그와 관련된 연구가 활발히 진행 중이다.

지속적으로 움직이는 이동 객체의 위치를 추적하고 저장하기 위해서는 새로운 기술의 색인 구조가 요구된다. 기존의 색인 구조는 특정 시점의 객체 위치를 관리하기 때문에 이동 객체의 위치 정보와 같이 동적인 속성을 가지는 정보를 표현, 저장 그리고 질의하는데 있어서 비효율적이다. 그 예로, R-트리[1] 기반의 기존 공간 색인 구조는 지속적으로 변경되는 이동 객체의 위치 정보를 색인하기 위해서 색인 구조에 있어 많은 갱신 비용을 요구하고 성능을 크게 감소시키는 문제점을 가진다.

이러한 문제점을 해결하기 위해 새로운 색인 구조들이 제안되었다[2, 3]. 그러나 이러한 색인 구조들 또한 객체의 동적 변화에 따른 많은 갱신 비용을 소요할 뿐만 아니라 이로 인해 검색 성능을 저하시키는 문제점이 있다.

본 논문에서는 이동 객체의 현재 및 미래 위치 검색을 효과적으로 수행하기 위한 TPKDB-트리라는 새로운 색인 구조를 제안한다. 제안하는 색인 구조에서는 KDB-트리를 기반으로 객체의 미래 위치를 검색하고 갱신 비용을 최소화하기 위한 기법을 제안한다. 또한 공간활용도를 향상시키기 위한 새로운 분할 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존에 제안된 이동 객체를 색인하는 색인 기법들에 대해 살펴본다. 3장에서는 TPKDB-트리의 구조와 삽입, 삭제, 분할 그리고 검색 기법에 대해 기술한다. 4장에서는 제안하는 색인 구조와 기존 색인 구조들과의 비교 분석한 내용을 기술한다. 마지막으로, 5장에서는 결론 및 향후 연구 방향에 대해 기

술한다.

2. 관련 연구

기존의 공간 색인 기법에서는 위치가 고정된 객체의 위치를 저장하고 관리하기 위한 많은 연구들이 진행되었으나, 최근에는 이동 객체를 효과적으로 저장하고 관리하기 위한 많은 연구들이 진행되고 있다. 대용량의 이동 객체에 대한 검색을 효과적으로 처리하기 위해서는 이동 객체를 관리하기 위한 시공간 색인 구조가 필수적이다. 초기의 시공간 색인 구조는 대부분 과거에서 현재까지 객체의 이동 경로 즉, 궤적을 관리하거나 현재의 위치만을 관리하였다. 3DR-트리, STR-트리, HR-트리 등은 객체의 과거 위치와 현재 위치를 공간적인 위치와 시간 정보를 사용하여 표현한다. 이러한 색인 구조들은 이동 객체의 궤적을 효과적으로 표현하지 못하기 때문에 이를 해결하기 위해 TB-트리, SETI-트리, SEB-트리 등이 제안되었다.

기존의 색인 구조들은 이동 객체의 미래 위치에 대한 검색을 수행하지 못하는 문제점이 있다. 이를 해결하기 위해 최근 이동 객체에 대한 현재 및 미래 위치에 대한 검색을 지원하는 TPR-트리[2], VCI-트리[3] 등이 제안되었다. 미래 위치에 대한 검색을 수행하기 위한 색인 구조들은 색인 구조 내에 현재 객체들의 공간적인 위치 정보와 함께 이동 객체의 방향과 속도 정보를 표현한다.

그러나 제안된 색인 구조들은 이동 객체의 불확실성을 해결하기 위해서 이동 객체의 동적인 변화나 새로운 객체의 삽입이나 삭제로 인해 전체 색인 구조를 순회하면서 갱신 작업을 수행하기 때문에 많은 갱신 비용을 필요로 할 뿐만 아니라 이로 인해 검색 성능을 저하시킨다.

3. TPKDB-트리 (Time Parameterized KDB-Tree)

3.1 색인 구조

제안하는 TPKDB-트리는 불필요한 갱신 비용을 최소화하고 미래 위치에 대한 성능을 향상시키기 위해 그림 1과 같이 KDB-트리와 보조 색인 구조의 혼합형 색인 구조로 구성되어 있다.

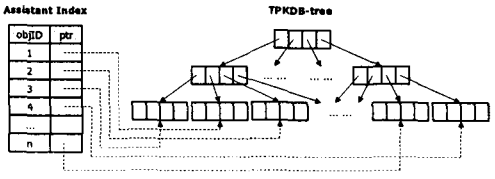
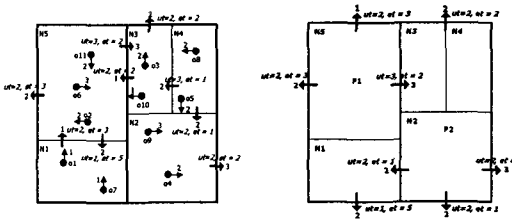


그림 1. TPKDB-트리의 구조

일반적으로, 이동 객체의 위치 데이터를 색인하는 방법은 분할 방법에 따라 DPAM (Data Partitioning Access Method)과 SPAM (Space Partitioning Access Method)로 나누어 살펴 볼 수 있다. SPAM 분할 기법은 데이터의 분포를 고려하여 분할하기 때문에 빈번한 색인 구조의 변경을 피할 수 있어 DPAM 보다 성능이 우수하다[4]. 따라서 본 논문에서는 이동 객체를 색인하는 기존 색인 구조들의 많은 갱신 비용을 줄이고 검색 성능을 향상시키기 위해 이동 객체의 위치 데이터를 3차원 상에서 점으로 모델링하고 SPAM인 KDB-트리을 사용한다.

보조 색인 구조는 이동 객체 검색 비용을 줄임으로서 갱신 연산을 줄이기 위해 이동 객체의 ID를 별도로 관리하는 색인 구조이다. 보조 색인 구조는 어떤 종류의 색인 구조를 사용해도 무관하나 이동 객체 ID 검색시 디스크 I/O 횟수가 최소인 색인 구조를 사용하고 보조 색인 구조에 색인된 이동 객체 ID는 해당 객체가 색인된 TPKDB-트리의 단말 노드와 연결 (Link) 정보를 가진다.

그림 2는 TPKDB-트리의 노드 구조를 보여주고 있다. 각 노드들은 미래 위치 검색을 지원하고 검색 비용을 최소화하기 위해서 영역 내에 있는 이동 객체나 노드들과의 관계를 시간에 대한 파라미터로 유지한다.



(a) 단말 노드 (b) 중간 노드

그림 2. TPKDB-트리의 노드 구조

이동 객체는 $<OID, t, d, v, x_{ref}, y_{ref}>$로 표현되고 이동 객체를 이동 객체 ID($OID$), 색인 요청 시간($t$), 이동 방향($d$), 이동 속도($v$) 그리고 t 시점에서의 위치 (x_{ref}, y_{ref}) 정보들로 구성함으로서 미래 위치 검색 질의시 별도의 갱신 없이 위 정보들로 구성된 시간에 대한 선형함수를 사용하여 질의를 처리할 수 있다. 그림 2의 (a)에서는 이동 객체를 포함하는 단말 노드의 구조를 보여준다. 단말 노드는 $<RS, t_{upd}, t_{esc}, v, objPtr_1, ..., objPtr_n>$와 같은 구조를 가지고 노드의 영역 정보($RS$), 갱신 시간($t_{upd}$), 노드 탈출 시간($t_{esc}$), 노드 속도($v$) 그리고 이동 객체 포인터($objPtr$) 정보로 구성된다. 그림 2의 (a)에서 각 점은 단말 노드 안에 포함된 이동 객체를 이동 객체 ID와 함께 나타내므로 단말 노드 N5에 포함된 이동 객체 o6은 동쪽으로 속도 3을 가지고 이동하는 것을 나타낸다.

단말 노드의 v는 TPR-트리[2]의 노드와 유사한 방법으로 미래 위치 검색 질의를 위해 미래 시간에 대해 노드를 확장하는데 사용되는 정보이며 t_{upd}, t_{esc}는 노드 확장시 불필요한 노드의 확장을 줄임으로서 노드들간의 겹침 (Overlap)을 줄이고 검색 성능을 향상시키기 위해 사용되는 정보이다. v는 노드에 포함되어 있는 객체들 중 각각의 대응되는 방향에 대해 가장 큰 값을 t_{upd}와 t_{esc}는 가장 작은 값을 취함으로서 노드에 포함된 모든 이동 객체들을 고려하여 노드를 확장할 수 있게 한다. 그리고 v는 노드의 최소 확장을 보장하기 위해서 대응되는 방향에 대한 이동

객체가 없고 대응되는 반대 방향에 대한 υ가 있을 경우 그 값의 음의 값을 취한다. N5에 대해 왼쪽으로 향하는 화살표와 '2'는 노드 확장시 왼쪽으로 단위 시간에 대해 2만큼 영역을 확장한다는 것을 나타내고 $up(t_{upd})$=2와 $et(t_{esc})$=3는 미래 질의 처리를 위해 노드 확장시 N5는 질의 시간 5까지는 노드 확장이 불필요하다는 것을 나타낸다.

그림 2의 (b)는 중간 노드를 나타내며 <RS, t_{upd}, t_{esc}, υ, $childNodePtr_1$, ..., $childNodePtr_n$>와 같이 표현된다. 중간 노드를 구성하는 정보들은 단말 노드가 노드 안에 있는 이동 객체에 의해 각각의 값이 결정되었다면 중간 노드는 노드 안에 있는 자식 노드가 유지하고 있는 값에 의해 각각의 값이 결정된다.

3.2 삽입 및 삭제 알고리즘

TPKDB-트리에서의 삽입·삭제 알고리즘은 KDB-트리의 삽입·삭제 알고리즘과 유사하다. 단지, 보조 색인 구조에 색인된 이동 객체 ID들은 해당 객체가 삽입된 TPKDB-트리의 단말 노드와 연결 정보를 유지하기 때문에 이동 객체의 삽입과 삭제시 변경되는 연결 정보의 변경 연산이 수반된다.

이동 객체의 갱신 처리를 위해 수반되는 삽입·삭제 연산시 보조 색인구조를 통한 빠른 객체 검색과 갱신 전과 갱신 후의 이동 객체의 위치가 동일 노드이고 이동 객체의 궤적을 구성하는 파라미터들의 변경이 없는 경우 갱신을 피하는 방법을 사용함으로서 디스크 I/O 횟수를 줄여 갱신 성능을 향상시킨다.

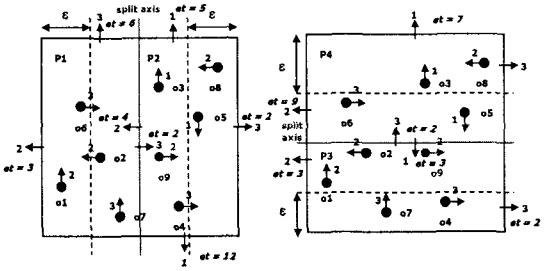
3.3 분할 알고리즘

[5]에서는 KDB-트리의 중간 노드 분할 정책인 FS 분할 (Forced Splitting)과 FD 분할 (First Division Splitting)을 정의하고 두 분할 정책에 관한 성능 비교를 하였다. 성능 비교 결과 FD 분할이 강제로 중간 노드의 하위 노드까지 분할이 전파되는 문제를 발생하는 FS 분할 보다 단말 노드의 공간활용도가 높아지고 검색 성능도 향상됨을 보였다.

[4]에서는 KDB-트리의 단말 노드 분할 정책으로 데이터 종속적인 분할 (Data dependent split strategies)과 분할 종속적인 분할 (Distribution dependent split strategies)에 대해 정의하고 분할 종속적인 분할 정책보다 데이터 종속적인 분할 정책이 균등하게 데이터를 분할하고 공간활용도가 개선됨을 보였다.

그러나 기존의 KDB-트리의 분할 정책은 이동 객체의 특성을 고려하지 않은 분할 정책이고 FD 분할 정책, 데이터 종속적인 분할 정책 그리고 분할 종속적인 분할 정책 모두 단말 노드에 저장된 데이터 분포 편차가 클 경우 불균등한 데이터 분할이 일어나 단말 노드의 공간 활용도가 저하되는 문제점을 가진다. 그래서 본 논문에서는 이동 객체를 색인하는 색인 구조에 효율적이면서 기존의 분할 정책이 가지는 중간 노드와 단말 노드의 공간활용도를 개선시키기 위한 EFD 분할 (Enhanced First Division Splitting) 정책을 제안한다.

그림 3과 같이 EFD 분할 정책에서는 단말 노드 오버플로우 발생시 데이터 종속적인 분할을 사용하여 모든 도메인에 대한 분할 위치를 선정한다. 분할 위치를 선정할 때는 분할되는 두 단말 노드가 수용해야 하는 최소 엔트리 수에 참여되는 데이터를 제외한 데이터들의 도메인 평균값을 분할 위치로 선정한다. 분할에 참여하는 단말 노드들의 최소 엔트리 수를 보장함으로서 편향된 데이터를 포함하는 오버플로우된 단말 노드의 공간활용도 저하 문제를 해결할 수 있다. 각 도메인에 대한 분할 위치를 선정 후, 각 단말 노드의 t_{upd}, t_{esc} 그리고 υ을 구한다. 그리고 임의의 t (>각 단말 노드의 t_{esc} 중 가장 큰 값)에 대해 각 단말 노드를 확장한 후, 상대적으로 작게 확장되는 단말 노드를 포함하는 분할 도메인을 분할 도메인으로 선정한다.



(a) x축으로 분할한 예 (b) y로 단말 노드 분할 예

그림 3. EFD 분할로 인한 단말 노드 분할

EFD 분할은 중간 노드 분할시 단말 노드의 강제 분할을 막기 위해 FD 분할을 사용하고 공간 활용도를 개선하기 위해서 분할 축을 오버플로우된 중간 노드의 첫 분할 도메인으로 선정하는 것이 아니라, 첫 분할 도메인을 기준으로 분할했을 때보다 공간활용도를 개선시키는 분할 도메인을 분할 축으로 선정한다. 첫 분할 도메인에 비해 공간활용도를 개선시키는 분할 도메인이 없다면 FD 분할과 마찬가지로 첫 분할 도메인을 분할 축으로 선정한다.

3.4 검색 알고리즘

TPKDB-트리는 이동 객체의 현재 위치에 대한 검색 처리뿐만 아니라 미래 위치에 대한 검색도 처리하는 색인 구조이다. TPKDB-트리에서 제공되는 검색의 종류는 두 가지로 객체 질의와 범위 질의가 있다. 객체 질의는 특정 이동 객체에 대한 현재와 미래 위치에 대한 질의이고 범위 질의는 현재 또는 미래 시간에 대해 정의된 영역 안에 있는 모든 이동 객체에 대한 질의이다.

객체 질의는 보조 색인 구조를 사용하여 빠르게 검색할 수 있고 현재 시간에 대한 범위 질의는 기존 KDB-트리에서 일정 질의 영역 안에 있는 객체를 검색하는 방법과 동일하다. 미래 시간에 대한 범위 질의를 처리하는 방법은 그림 4에서와 같이 각 노드의 t_{upd}와 t_{esc} 같은 시간

정보와 속도 정보 v를 가지고 질의 시간 t_{query}까지 노드를 확장한다. 그림 4에서와 같이 노드 확장 후 질의 영역과 겹침이 발생하는 노드 안의 이동 객체에 대한 궤적을 구한 후 질의 시간에 대해 질의 영역 안에 포함되는 이동 객체를 반환함으로서 미래 위치 질의를 처리할 수 있다.

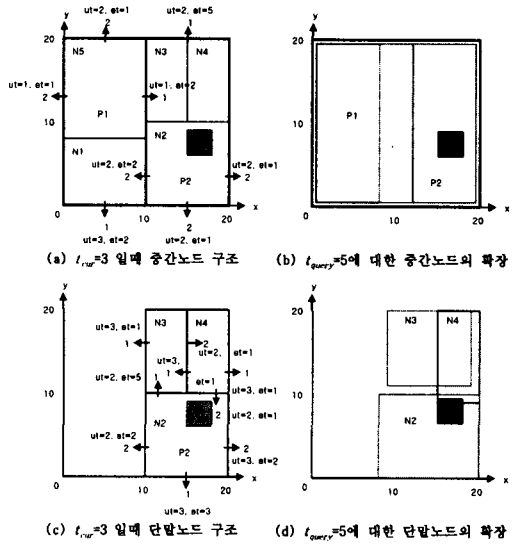


그림 4. TPKDB-트리에서 미래 위치 검색

4. 성능 평가

성능평가에 사용된 시스템은 펜티엄-IV 1.7GHz 프로세서에 256Mbyte의 메모리를 가지며, 운영체제는 윈도우 2000을 사용하였다. 성능평가에 사용된 파라미터로는 트리 노드 크기를 4Kbytes로 설정했고 시뮬레이션으로 사용된 데이터는 임의의 속도와 방향을 가지고 1000×1000㎞의 2차원 영역을 지속적으로 이동하는 균등 분포된 100,000개의 데이터을 사용하였다.

그림 5는 동일한 환경을 설정하고 TPR-트리와의 삽입 성능을 비교한 것으로 삽입되는 객체의 수가 증가할수록 TPKDB-트리의 성능이 향상됨을 볼 수 있다.

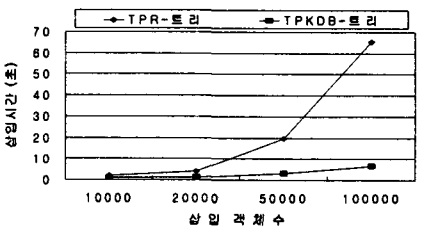
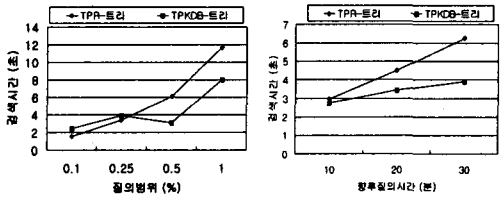


그림 5. TPKDB-트리와 TPR-트리의 삽입시간

그림 6의 (a)는 100,000개의 데이터를 삽입하고 현재 시간에 대해 질의 범위가 다른 1000개의 범위 질의 처리 성능을 비교한 것이고 (b)는 전체 영역 1%에 해당하는 질의 범위를 가지고 각기 다른 미래 시간에 대해 범위 질의 1000개를 처리한 것을 나타낸다. 현재 시간에 대한 범위 질의 처리시 질의 범위가 작은 경우 KDB-트리의 사장 영역문제로 인한 성능저하 부분을 제외하고는 검색 성능이 향상됨을 볼 수 있다.



(a) 질의 범위에 따른 검색 시간 (b) 미래 질의 시간에 따른 검색시간

그림 6. TPKDB-트리와 TPR-트리의 검색시간

5. 결론

본 논문에서는 효율적으로 이동 객체를 관리하고 미래 위치를 검색할 수 있는 색인 구조를 제안하였다. 제안된 TPKDB-트리는 이동 객체의 갱신 비용을 줄이기 위해 성능이 개선된 KDB-트리와 보조 색인 구조의 혼합형 색인 구조이다. 그리고 빠른 이동 객체의 미래 위치 검색을 위해 노드와 노드 안에 포함되어 있는 이동 객체의 관계를 시간에 대한 파라미터로 유지하는 방법을 제안하였다. 또한, 색인 구조의 공간활용도를 향상시키기 위해서 EFD 분할을 제안하였고 성능 평가를 통해 기존 색인구조에 비해 갱신과 검색 성능이 향상되었음을 보였다. 향후 연구 방향으로 다양한 실험을 수행하고 다양한 유형의 검색 기법들에 대한 연구들을 진행할 예정이다.

5. 참고 문헌

[1] A. Guttman, "R-trees : A dynamic index structure for spatial searching", Proc. ACM SIGMOD, pp.47-57, 1984.

[2] S. Simonas, Christian S. Jensen, Scott T. Leutenegger and Mario A. Lopez, "Indexing the Positions of Continuously Moving Objects", Proc. ACM SIGMOD, pp.331-342, 2000.

[3] S. Prabhakar, Yuni Xia, Kalashnikov, D.V., W.G. Aref and S.E. Hambrusch, "Query indexing and velocity constrained indexing : scalable techniques for continuous queries on moving objects", IEEE Transactions on Computer, Vol.51, pp.1124-1140, 2002.

[4] A. Henrich, H.-S. Six and P. Widmayer, "The LSD Tree : Spatial Access to Multidimensional Point and Non-point Objects", Proc. VLDB, pp.45-53, 1989.

[5] Ratko Orlandic and Byunggu Yu, "Implementing KDB-Trees to support High-Dimensional Data", IDEAS, pp.58-67, 2001.