# 이동 객체의 위치 처리 기법

김진덕 · 진교홍

동의대학교

# A Method for Managing Positions of Moving Objects

Jin-Deog Kim · Kyo-Hong Jin

Dongeui University

E-mail : {jdk, khjin}@dongeui.ac.kr

## 요 약

이동 객체는 시간에 따라 연속적으로 위치정보가 변하는 현실 세계의 모든 객체를 말한다. 기존의 공간 색인에 관한 연구는 효율적인 검색 방법을 제시하였지만 이동체 데이터베이스에서는 질의처리의 효율성보다 이동 객체의 최근 위치를 빨리 저장하는 것이 더 중요하다. 그러므로 기존의 색인은 가능한 한 정확한 현재 위치 정보를 제공해야 하는 이동체 데이터베이스에는 적용할 수 없다.

이 논문에서는 이동 객체의 색인 구축비용을 줄이기 위한 처리 기법을 제안한다. 구체적으로 각 이동 객체의 위치 변경 보고 즉시 변경하는 방법과 일정 주기마다 전체 객체에 대해 재색인하는 방법의 장단점을 분석하고, 데이터베이스 변경 횟수의 최소화를 위해 이동체의 특성을 감안한 새로운 선택적 즉시 변경 방법을 제안한다.

## ABSTRAÇT

The objects which continuously change their locations in the real world are called moving objects. The works on the existing spatial indices have been proposed the retrieval methods. However, keeping track of the terminal location of moving objects is more important than the efficiency of the query processing in the moving object database. Therefore, many pure spatial indices are not applicable to the moving objects database which should maintain the object's current location as precise as possible.

This paper proposes a method for reducing the construction time of indexing moving objects. We analyze the characteristics of the method to re-index all the objects after each time period and the method to update immediately the locations on reporting their locations. We also newly propose a selective immediate update method using the properties of moving objects in order to minimize the number of database updates.

키워드

이동 객체, 선택적 변경 연산, 버켓 분할

## 1. 서 론

최근 들어 무선 통신 기술의 발달, GPS(Global Positioning System)의 활용도가 높아짐에 따라 PDA(Personal Digital Assistants), 휴대폰 등의 무선 이동 기기들로부터 이동체의 위치 정보를 수집하기가 용이해져 위치기반 서비스(LBS : Location Based Service)가 활성화되고 있다.

위치기반 서비스를 위한 이동 객체의 데이터베이스는 효율적인 데이터 관리와 검색 성능의 향상을 위해 2차원 공간 색인의 도입이 필수적이다. 그러나 계속 변하는 위치 정보를 갖는 이동 객체를 전통적인 공간 색인에 저장 및 관리할 경우 심각한 시스템 성능 저하를 초래하게 된다. 왜냐하면 기존의 공간 색인은 정적 데이터의 효율적인 검색을 위주로 고안된 것이기 때문에 이동체의 고유한 특성을 반영하지 못하고 있다[1]. 그러므로, 빈번한 데이터 변경이 발생하는 이동체에 대한 효율적인 색인 구축 기법의 대한 연구가 필요하다.

이와 같이 이동 객체의 색인화는 빠른 검색보다 오히려 색인 구축 시간을 최소화함으로써 이동체의 가장 최근 위치를 저장하는 것이 더욱 더 중요한 의미를 갖는다. 즉, 각 이동체의 위치 보고 주기를 보다 짧게 하는 것이 최신의 위치 정보를 갖는 것이며, 이를 위해 색인 구축 시간을 단축하는 것이 이동 객체의 처리에서는 매우 중요하다.

따라서 이 논문에서는 이동체를 공간 데이터베이스에 효율적으로 색인하기 위한 위치정보의 처리 기법을 제안한다. 구체적으로 각 이동 객체로부터 위치 정보를 보고 받은 즉시 위치 정보를 변경하는 방법과 보고 받은 위치 정보를 보관한 뒤 일정 주기마다 전체 객체에 대해 재색인하는 방법의 장단점을 분석한다. 그리고 이를 근거로 데이터베이스 변경 횟수를 최소화 방법을 이용하는 새로운 선택적 즉시 변경 방법을 제안한다. 이는 이동체의 움직임 특성을 파악하여 위치변경에도 불구하고 색인 재구성이 불필요한 경우에는 변경 연산을 수행하는 않는 방법을 택하는 것이다. 이를 위해 버켓 모양 변형과 같은 방법을 이용한다.

이 논문의 구성은 다음과 같다. 제 2장에서는 이동 객체의 공간 색인에 관한 관련연구에 대해 알아보고, 제 3장에서는 전술한 즉시 변경 방법과 주기적 재색인 방법의 장단점을 분석한 뒤, 이동체를 위한 새로운 공간 색인 기법을 제안하며, 간단한 실험을 통해 제안한 기법의 효율성을 살펴본다. 그리고 제 4장에서 결론을 맺는다.

## II. 관련 연구

기존의 데이터베이스는 정적인 상태를 표현하는 고정된 값을 관리하는 반면 이동체 데이터베이스는 지속적으로 변화하는 동적인 값을 다룬다. 지금까지의 이동체를 표현하기 위한 연구는 위치 정보를 수식으로 표현하는 방법[2], 이동 객체의 과거 이력 정보를 검색하기 위한 궤적으로 표현하는 방법[3,4,5], 미래의 위치를 표현하고 검색하기 위한 방법[6], 데이터베이스 변경횟수를 줄이기 위한 방법[7] 등으로 나뉘어진다.

관련연구 [2]에서는 이동 객체의 이동 속도와 방위각, 시간 축을 이용하여 수식으로 현재 및 미래 위치를 계산하는 연구이다. 객체 검색은 수학적인 연산으로 이루어진다. 그리고 이 연구는 하부구조로 색인을 사용하지 않고 있다.

관련 연구 [3]은 이동체의 궤적을 선분으로 표현한 색인으로 R-Tree의 변형인 STR-Tree, TB-Tree를 제시하였다. 그러나 이 색인들은 현재위치 검색이 고려되지 않고 영역질의 성능이 좋지 않다. 3DR-Tree[4]는 R-Tree를 3차원으로 확장한 구조로 이동객체의 궤적 질의를 지원하는 색인이다.

이동체의 과거궤적 및 현재 위치를 동시에 지원하기 위한 연구로 2+3DR-Tree[5]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 시간이 소요되어 최신의 현재 위치를 정확하게 표현하기 어렵다.

관련 연구 [6]은 이동체를 시간에 대한 선형함수로 표현하는 색인구조인 TPR-Tree를 제시하였다. 이동체의 위치를 단순한 좌표로 저장하지 않고 방향과 속도를 나타내는 벡터를 저장하여 특정 임계값 이하의 변화일 경우 변경하지 않아 그 횟수를 줄일 수 있지만, 시간에 대한 선형 함수를 사용하여 현재 및 가까운 미래위치에 대한 변환 연산에 많은 시간이 소요된다.

관련연구 [7]은 이동객체의 데이터베이스 변경 횟수를 줄이기 위한 방안을 제시하여 현재 이동체의 위치 정보를 검색하기 위한 방법을 제시하였다. 그러나 이 연구에서는 즉시 변경정책과 전체 재색인 방법의 대한 평가가 없으며, 이동체의 특성을 고려한 색인 기법이 제시되지 않았다.

## III. 이동체의 색인을 위한 변경 정책

### 3.1 즉시 변경과 재색인 정책

공간 색인에서 색인 구축방법은 크게 즉시 변경과 재색인 정책과 같은 두 가지가 구분된다. 즉시 변경 정책은 공간 객체의 위치정보가 변하는 즉시 해당 객체를 기존의 색인에서 삭제한 후 새로운 위치정보를 삽입하는 연산으로 이루어진다. 전체 객체의 재색인 정책은 주기적으로 공간 색인에서 관리하고 있는 전체 객체로부터 변화된 위치 정보를 보고 받은 후 새로운 색인을 구축한다.

위와 같은 두 방법은 각각 장단점이 존재한다. 우선, 즉시 변경은 항상 삭제 및 삽입 연산이 이중으로 발생하고, 각 이동 객체가 삽입되는 순서에 따라 R-tree 계열의 공간 색인은 그 모양이 다양해지고 이웃 노드간의 겹침 영역이 많이 발생하여 전체적인 색인 성능이 떨어진다. 반면, 전체 재색인 정책은 일정기간의 주기 동안 각 이동 객체의 위치 정보를 정리한 후 한꺼번에 색인을 구축하므로 삭제 과정이 생략되므로 색인 구축시간이 단축되며, 삽입 순서를 조정하므로 노드의 크기가 줄어든다.

그렇지만 위치 정보의 평균 오차율 측면은 다음과 같은 양상을 보인다. 그림 1은 즉시 변경 정책에 의한 위치보고 주기와 유효한 질의시간을 그림으로 나타낸 것이다.


그림 1. 즉시 변경 정책

그림에서 $\alpha$는 하나의 객체에 대한 삽입 연산시간이다. 위치 보고 주기를 $T_0$~$T_1$이라 할 때, 유효한 질의 시간은 $(T_0+\alpha)$~$(T_1+\alpha)$이다. 따라서 최소 오

차, 최대 오차, 평균오차는 다음과 같다.

최소 오차 : $\alpha$

최대 오차 : $(T_1 - T_0) + \alpha$

평균 오차 : $(T_1 - T_0)/2 + \alpha$

위 수식에서 알 수 있는 것은 즉시 변경이라 하더라도 최대 오차는 '위치보고 주기 + 한 객체의 삽입 시간'임을 알 수 있다. 그리고 평균 오차는 '위치 보고 주기의 절반 + 한 객체의 삽입 시간'이다. 예를 들어 전체 객체의 위치 보고 주기가 100초이고 한 객체의 평균 삽입 시간이 0.5초이면 평균 오차 시간은 50.5초가 된다.

그림 2는 전체 재색인 정책에 의한 위치보고 주기와 유효한 질의시간을 그림으로 나타낸 것이다. 그림에서 위치 보고 주기와 색인 구축 시간이 계속 중복되면서 진행됨을 알 수 있다. 이 정책을 사용할 경우 위치 보고 주기(a)를 기준으로 한 최소 오차, 최대 오차, 평균오차는 다음과 같다.

최소 오차 : $T_1 - T_0$

최대 오차 : $T_3 - T_0$

평균 오차 : $T_2 - T_0$

따라서 전체 객체의 위치 보고 주기가 100초이면 평균 오차는 200초이며, 위치 보고 주기가 25초일 때 평균 오차는 50초이다.

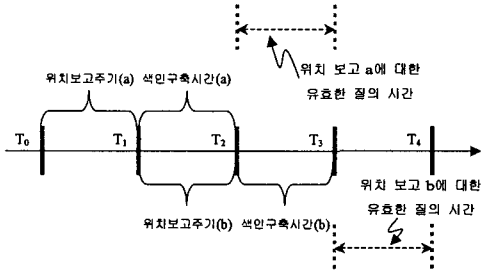



그림 2. 재색인 정책

그러므로 $\alpha$값이 크지 않는 한 재색인 정책을 택할 경우 위치 보고 주기를 1/4로 단축해야 즉시 변경 정책을 사용할 경우와 같은 오차를 기록함을 알 수 있다. 즉, 재색인 정책을 사용할 경우 4배나 많은 색인 구축이 필요하다. 일반적으로 $\alpha$값은 전체 재색인 시간에 비해 극히 작은 시간값을 갖는다. 따라서 즉시 변경 정책이 훨씬 좋은 성능을 보임을 분석을 통해 알 수 있다.

### 3.2 이동체의 위치 처리 기법

이 논문에서는 적은 횟수의 색인 구축으로 좋은 오차율을 보이는 즉시 변경 정책을 채택하여 보다 효율적으로 이동체를 색인화하는 성능향상 기법을 제안한다. 논문의 이동체는 점으로 표현되며, 일정한 주기로 현재 위치를 보고하는 것으로 가정한다.

(1) 변경 연산의 최소화

지금까지 공간 색인은 객체의 상태 변화가 발생하면 항상 색인을 변경하는 방법을 택했지만, 이동체 데이터베이스는 많은 수의 객체가 빈번하게 변경되는 것을 빠른 시간내에 색인에 반영해야 하므로 가능하면 변경이 불필요한 객체에 대해서는 변경 연산을 하지 않음으로써 전체 색인 구축시간을 단축하는 것이 효과적인 방법이다. 예를 들어 위치 변경 보고 주기 동안 전혀 움직이지 않았거나 일정 거리 이내만 움직인 경우[8]와 주어진 경계 내에서만 움직인 경우[7]에는 색인 구조의 변경이 불필요하다. 이 논문에서는 두 번째 방법을 도입하여 이동객체의 검색을 위해 공간 색인을 사용하며 색인 변경의 최소화를 위해 이동 객체가 이동 후 해당 버켓 내에 존재하면 데이터 파일 내에서의 값은 최신 값으로 변경되지만 색인에서의 포인터 정보는 변경하지 않는 방법을 이용한다.

(2) 이동체를 위한 공간 색인

공간 색인은 공간 분할 방식의 사분트리(Quad-tree), 해쉉(그리드) 구조와 객체 경계 방식의 R-Tree가 있다. R-Tree는 중간 노드에서 하위노드의 포함하는 최소 경계 사각형을 유지하기 때문에 변경 연산은 이 사각형의 확장을 초래하므로 중복 검색이 발생한다. 또한 이동 객체의 위치 변경을 빠른 시간 내에 처리되어야 하지만 R-tree는 그 복잡한 구조로 인해 색인 구축에 많은 시간이 소요되므로 이동체를 색인 구조로는 부적합하다.

반면, 사분트리나 해쉬 기반의 공간 색인은 그 구조가 간단하여 빠른 색인 구축이 가능하므로 이동체를 위한 공간 색인으로 적합하지만, 밀집 지역에서 버켓 오버플로우가 발생한다. 이와 같은 오버플로우는 버켓의 분할을 야기하고, 이동체의 방향 특성을 무시한 채 사분트리나 해쉬 기반 공간 색인의 단순히 공간적인 정규분할을 한다.

그러나 지금까지 연구된 공간 색인은 이동체의 특성을 전혀 고려하지 않았다. 그래서 빠른 색인 구축이 가능하고 이동체의 특성을 감안하여 변경 연산의 최소화를 유도하는 2차원 동적 해쉉(dynamic hashing) 파일 구조를 새롭게 제안한다.

그림 3은 이동체를 위한 동적 해쉉 파일 구조이다. 오버플로우가 발생했을 경우 다시 2개의 버켓으로 분할하여 하나의 버켓에 들어가는 객체의 수를 일정하게 유지하도록 하였다. 그림 3에서 5,9번 버켓은 이동객체가 밀집되어 있으므로 추가적인 분할이 발생하였음을 보여준다. 다음은 해쉉 함수를 나타낸 것이다.

· Hx(x) = int[(x - Xmin)/(Xmax-Xmin)*Nx]

· Hy(y) = int[(y - Ymin)/(Ymax-Ymin)*Ny]

단, Nx, Ny는 Y축 버켓의 수

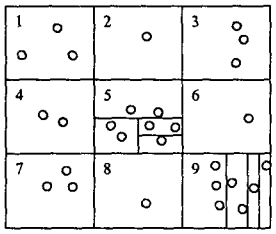



그림 3. 2차원 동적 해쉬 파일 구조

그런데 5번과 9번 버켓의 분할은 조금 다른 양상을 보여준다. 5번 버켓은 최초 Y축으로, 그 뒤에는 X축, Y축으로 연속적인 분할이 되었지만, 9번 버켓은 X축으로 세 번 연속 분할되었다.

이 논문에서 객체들을 두 개의 버켓으로 분산시킬 때 전술한 변경 연산 횟수를 최소화하기 위해 X축과 Y축을 선택할 수 있다. 기존의 색인에서는 분할된 버켓이 정사각형에 가깝도록 X축으로 분할하거나 X와 Y축을 번갈아가며 분할한다. 예를 들어 그림 4에서 하나의 버켓을 X와 Y축을 기준으로 각각 분할할 수 있다. 그러나 이동체는 항상 움직인다는 특성을 감안하면 그림 4의 예에서는 Y축을 기준으로 분할하는 것이 변경연산을 최소화할 수 있다. 왜냐하면 버켓내의 객체들은 X축으로 따라 이동하기 때문에 Y축을 기준으로 분할하면, 이동체의 다음 위치 보고 시간에 해당 버켓내에 존재할 확률이 높아 전술한 바와 같이 변경연산을 생략할 수 있기 때문이다.

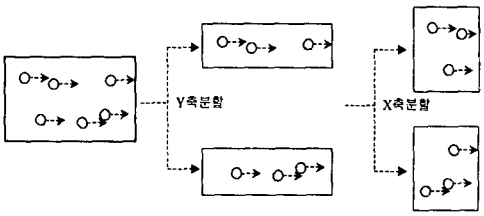



그림 4. 버켓 분할

이 논문에서는 분할하고자 하는 축을 선택할 때 다음과 같은 규칙을 차례대로 적용한다.

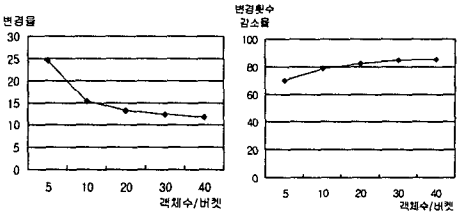
① 버켓내의 객체들의 방향성의 평균값이 X축이면 Y축을 기준으로, Y축이면 X축 기준 분할
② 현재 벡터의 방향성이 없을 경우 도로, 항로 등과 같은 인프라의 방향성을 기준으로 분할
③ 이동 객체의 움직임에 관한 통계정보(평상시의 이동경로 등)를 근거로 방향성 기준으로 분할

단, 분할될 두 버켓중 한 버켓에 극단적으로 많은 버켓들이 배분될 경우 두 버켓에 골고루 배분될 수 있는 축을 택한다.

(3) 실험 고찰

실험 평가를 위해 사용한 데이터는 이동객체의 Benchmark 데이터로 이용되는 GSTD[5]로서 Gaussian 분포상태의 방향성(directed) 이동을 하는 1000개의 이동 객체로 구성하였다.

그림 5(a)는 해쉬 버켓의 수에 따른 색인 변경 횟수의 감소효과를 그래프로 나타낸 것이다. 이 논문에서 제안한 변경 연산의 최소화 기법을 사용한 경우 전체적으로 75%이상의 감소효과가 있었으며, 버켓의 크기가 클수록 변경 횟수는 더욱더 감소함을 보였다. 그래프의 Y축은 변경횟수율로 기존 방법에 비해 이 논문에서 제안한 방법의 변경 횟수를 백분율로 나타낸 것이다.




(a) 변경 연산 감소율 (b) 버켓 분할 효과

그림 5. 변경횟수 비교

그림 5(b)는 이 논문에서 제시한 2차원 동적 해쉬의 버켓 분할의 효과를 그래프로 나타낸 것이다. 기존의 X, Y축 교차 분할에 비해 이동체의 특성을 감안했을 경우 데이터베이스 변경 횟수가 평균 20%이상 줄어들었다.

## IV. 결 론

이 논문에서는 이동 객체의 색인 구축 비용을 줄이기 위한 처리 기법을 제안하였다. 구체적으로 각 이동 객체의 위치 변경 보고 즉시 변경 방법과 일정 주기마다 전체 객체에 대한 재색인 방법 중 즉시 변경 방법이 좋은 성능을 보임을 알 수 있었고, 즉시 변경 방법을 이용할 경우의 데이터베이스 변경 횟수를 최소화하는 방법과 이동체의 특성을 고려한 버켓 분할을 수행하는 색인으로서 2차원 동적 해쉬 파일을 제시하였다. 실험 결과 이 논문에서 제안한 방법은 75% 이상의 색인 변경 연산 감소 효과가 있었으며, 제안한 버켓 분할 방식이 기존의 방법에 비해 20% 이상 좋은 성능을 보였다.

앞으로는 수많은 이동체가 짧은 주기마다 위치 정보를 보고하는 이동체 데이터베이스에서의 색인 속도 향상을 위해서 다중 프로세서나 쓰레드에 의한 병렬 처리 기법을 연구하고자 한다.

이 논문은 2002년도 한국학술진흥재단의 지원에 의하여 연구되었음(KRF-2002-003-D00277)

## 참고문헌

[1] O. Wolfson, B. Xu, S.Chamberlain, and L. Jiang. "Moving Objects Databases: Issues and Solutions," Proc. of SSDBM Conf., pp 111-122, 1998

[2] G. Kollios, D. Gunopulos, and V. J. Tsotras. "On Indexing Mobile Objects," Proc. of PODS Conf. pp. 261-272, 1999

[3] D. Pfoser, C. Jensen, Y. Theodoridis, "Novel Approaches to the Indexing of Moving Object Trajectories" , Proc. of VLDB Conf., pp. 395-406, 2000.

[4] Y. Theodoridis, "Spatio-Temporal Indexing for Large Multimedia Applications" , Proc. of Multimedia Computing and Systems Conf, pp. 441-448, 1996

[5] M. Nascimento, J. Silva, Y. Theodoridis, "Evaluation of Access Structures for Discretely Moving Points." Proc. of STDBM Conf., pp. 171-188, 1999

[6] S. Saltenis, C. Jensen, S. Leutenegger, M. Lopez, "Indexing the Positions of Continuously Moving Objects." , Proc. of ACM SIGMOD Conf., pp. 331 - 342, 2000.

[7] Z. Song, N. Roussopoulos, "Hashing Moving Objects", LNCS, pp. 161-172, 2001

[8] O. Wolfson, L. Jiang, A. Sistla, S. Chamberlain, N. Rishe, M. Deng, "Databases for Tracking Mobile Units in Real Time", Proc. of ICDT Conf., pp. 169-186, 1999