이동체 데이터베이스를 위한 해쉬 기반의 공간 색인

전 봉 기°, 홍 봉 희
부산대학교 컴퓨터공학과
{bgjun, bhhong}@hyowon.pusan.ac.kr

Hash based Spatial Indexing for Moving Object Database

Bong-Gi Jun° and Bong-Hee Hong
Dept. of Computer Engineering Pusan National University

요 약

이동 객체는 현실 세계의 차량과 같이 시간에 따라 움직이는 모든 객체를 말한다. 이동체 데이터베이스는 이동 객체가 연속적으로 위치 정보가 변하기 때문에 기존의 데이터베이스 기법과 차이가 있다. 이동 객체의 위치 변경은 빈번한 갱신 연산을 수행해야 하는 문제점이 있고, 이동체의 움직임을 모두 저장할 수 없으므로 이동 객체의 움직임을 이산적 표현 방법을 사용하여 이동 객체의 현재와 미래 위치를 계산해야 한다. 이동 객체의 빈번한 위치 변경은 색인 처리 비용이 증가를 초래한다.

이 논문에서는 빈번한 갱신으로 발생하는 색인의 과부하를 최소화하는 해쉬 기반의 공간 색인 구조를 사용한 이동 객체의 위치 변경 처리 기법을 제안한다. 또한 이동 객체의 이산적 표현을 위한 자료 구조를 제시하며, 이동 객체의 밀집화 문제를 해결하기 위하여 오버플로우 처리 방법을 기존의 해쉬 오버플로우 처리 방법을 응용하여 3가지 방법으로 제안한다. 버퍼질의를 이용하여 이동 객체의 영역 질의 방법을 제안한다.

1. 서론

이동 객체는 차량과 같이 시간에 따라 연속적으로 위치 정보가 변하는 객체이다. GPS와 이동 통신을 통하여 서버에 전달된 위치 정보를 저장하여 이동체들의 현재 위치를 검색할 수 있게 한다. 이동 객체는 공간적으로 두가지 특징이 있다. 첫째, 연속적으로 이동하기 때문에 빈번한 갱신 연산이 수행된다. 위치 변경은 색인 구조의 변경을 초래하기 때문에 색인 구조가 변하지 않는 색인 구조가 적합하다. 둘째, 이동 객체를 차량이나 사람이라고 본다면 도심이나 교차로 등에서 밀집되는 특징이 있다

이동 객체를 효율적으로 검색하기 위해서는 공간 색인 구조가 필요하다. 공간 색인 구조는 해쉬 기반의 그리드 구조와 트리 기반의 R-Tree가 대표적인 색인 구조라 할 수 있다. 트리 기반에는 Quad-Tree, K-D-B Tree 등도 있으나, 평형 트리(Balanced Tree) 구조가 아니기 때문에 밀집 지역에서 성능 저하를 초래한다. R-Tree는 균형 트리 구조를 가지고 있지만 중간 노드에서 하위노드의 포함하는 최소 경계 사각형(Minimum Boundary Rectangle)을 유지하기 때문에 이동 객체의 위치 변경이 이동체의 방향성으로 인하여 MBR의 확장을 초래하여 중복 검색의 원인이 된다[5].

해쉬 기반의 공간 색인은 밀집 지역에서 버켓 오버플로우가 발생한다. 버켓 오버플로우는 색인에서 분할을 초래하여 성능을 저하한다. 이 논문에서는 기존의 해쉬 기법에서 사용되는 오버플로우 처리 기법을 공간 인덱스에 응용한 3가지 방법을 제안한다.

이 논문의 구성은 다음과 같다. 2장에서는 이동 객체에 관한 관련 연구를 기술하고, 3장과 4장에서는 빈번한 갱신연산을 효과적으로 처리할 수 있는 색인 구조와 변경 연산 기법을 제시한다. 5장에서는 이동 객체의 밀집화 문제와 오버플로우의 처리 방법들에 대해 기술한다. 6장에서는 이동 객체를 2차원 상의 점으로 표현 했을 때의 영역 질의 처리 방법을 기술한다. 마지막으로 7장에서는 결론과 향후 과제를 기술한다.

2. 관련 연구

기존의 데이터베이스 기술에서는 저장 객체는 고정된 값을 저장하는데 목적을 두고 있다고 한다면 이동 객체는 변화하고 있는 값을 저장한다. 이동 객체에 관한 연구는 이동 단말체의 위치 정보 전송에 관한 연구와 이동 객체의 저장 방법에 관한 연구로 나뉘어 지며, 정확한 위치 정보를 저장하기 위한 효율적인 이동 객체의 색인 구조에 관한 연구가 활발히 진행 중에 있다.[1, 2, 5]

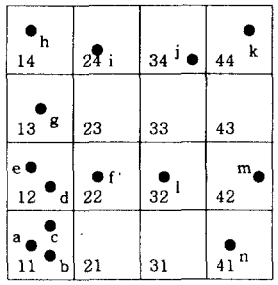
시간에 따라 연속적으로 변화하는 위치 정보를 표현하기 위하여 이동 객체 표현 방법은 수식으로 표현하는 방법[1]이 초기에 연구되었다. 이동 객체의 이동 속도와 방위각을 이용하여 시간 축을 이용하여 수식으로 현재 및 미래 위치를 계산하는 연구이다. 이후 이동 객체를 공간 객체로 표현하는 연구[3]가 진행되었으며, 크게 이동 객체를 선으로 표현하는 방법과 점으로 표현하는 방법 두 가지가 있다. 선으로 표현하는 방법은 이동 객체의 미래 궤적을 선으로 표현한다. 점으로 표현하는 방법은 이동 객체를 시간의 스냅샷(snapshot)에서 좌표상의 점으로 표현한 것이다.

이동 객체의 표현 방법에 따라 여러 가지 색인 방법들이 제시되었다. 수식을 이용하는 방법[1]에서는 색인이 없는 구조가 없다. 객체 검색은 수학적인 연산으로 이루어 진다. 선으로 표현하는 방법은 PMR 사분트리를 이용하여

구현하였다[2]. 이동 객체를 선으로 저장하여 객체 중복이 많고 일정 주기 마다 인덱스를 새로 생성해야 하는 문제점이 있다. 이동 객체를 점으로 표현한 연구는 TPR-tree 방법[5]을 제안하였다. 기존의 R-Tree의 문제점을 보안하였으나 위치 정보 변경의 색인 구조 변경의 비용은 크게 감소하지는 않았다. 점으로 표현하는 경우에는 질의 처리 시 버퍼 질의를 수행해야 하는 특징이 있다.

3. 이동 객체 색인 구조

이 논문에서 이동 객체를 2차원 좌표 상의 점으로 표현한다. 이동 객체의 위치 정보 수집은 GPS와 이동 통신을 통하여 이동 단말체에서 이동 객체 관리 서버로 전송되며 일정한 시간 간격으로 위치 정보를 전송한다. 이때 위치 정보를 보고하는 시점의 좌표와 시간이 데이터베이스에 저장된다(그림 1(c)).



(a) 고정 그리드 셀과 이동 객체

(b) 고정 그리드 인덱스

Cell_Id	Geometry_ids
11	a, b, c
12	e, d
13	g
14	h
....	
42	m
43	
44	k

(c) 이동 객체 공간 구조

Geom_id	Position	Start_time	Speed	Orient
a	x1, y1			
b	x2, y2			
...			...	
n	x12, y12			

그림 1. 이동 객체 고정 그리드 색인 구조

이동 객체의 영역 질의(Region Query) 처리를 위하여 공간 인덱스가 필요하다. 이 논문에서는 빈번한 갱신을 저비용으로 처리하기 위해서 고정 그리드 인덱스를 사용하였다(그림 1(b)). 이동 객체의 위치 정보 갱신 문제는 다음 장에서 다룬다.

이동 객체는 시간에 따라 이동하고 있기 때문에 이동 객체를 표현하기 위해서는 그림 1(c)과 같이 벡터로 표현하여 이동 객체의 현재 위치와 미래 위치를 예측하는데 사용한다. 이동 객체의 이동성을 표현하기 위하여 점 좌표, 보고 시간, 이동 방위각, 이동 속도를 저장한다.

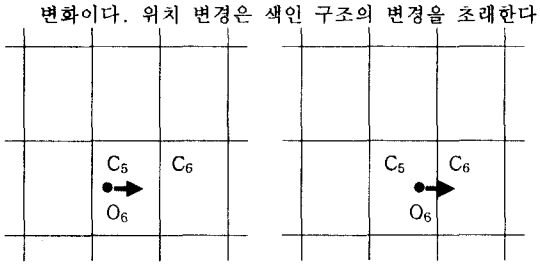
4. 이동 객체의 위치 변경 처리 기법

이동 객체는 위치를 파악하기 위해서는 변경된 이동 객체의 위치 정보를 이동 객체 관리 서버에 보고 되어야 한다. 변경된 위치 정보를 보고하는 것은 통신 비용이나 서버에서의 갱신 연산을 수행해야 하기 때문에 전송 횟수를 줄이는 방법들에 대해서 연구가 되었다. 하지만 이러한 방법들을 사용하더라도 여전히 이동 객체의 위치 갱신은 고비용의 연산이다.

갱신 연산의 종류는 다음과 같다.

1) 새로운 객체의 삽입
2) 객체의 삭제
3) 기존 객체의 변경(relocation)

이동 객체의 변경은 위치, 속도, 진행 방향의 변화이다. 위치 변경은 색인 구조의 변경을 초래한다.



(a) 셀 내에서의 이동 (b) 다른 셀로의 이동

그림 2. 이동 객체의 위치 이동과 색인의 변화

이동 객체의 갱신 연산은 대부분 기존 객체의 변경으로 볼 수 있다. 기존의 공간 인덱스는 대부분 변경 연산 보다는 검색을 중점으로 고려하였다. 이동 객체 데이터베이스에서는 빈번한 갱신 연산을 고려한 색인 구조가 필요하다. 공간 객체의 위치 변경은 트리 계열에서 삭제 연산과 재 삽입으로 이루어 진다. 삭제와 재 삽입은 색인에서 병합과 분할을 초래하기 때문에 위치 변경이 빈번한 이동 객체에는 적합하지 않다.

그림 2(a)와 같이 셀 내에서의 위치 변경은 색인 구조에서는 변경이 없다. 그림 2(b)와 같이 이동 객체가 다른 셀로 위치가 변경되어도 해쉬 기반의 색인 구조에서는 삭제와 재 삽입 연산 비용은 트리 계열보다 작다.

5. 이동객체의 밀집화 및 오버플로우 처리

5.1 이동객체의 밀집화 문제

이동 객체는 연속적으로 위치가 변화하기 때문에 시간에 따라 분포성이 변한다. 특히, 특정시간에 도심지, 교차로에서 밀집될 수 있다. 이동 객체의 밀집화는 해쉬 기반 인덱스에서 셀 버켓(bucket)이 오버플로우가 발생한다. 이 논문에서는 셀 버켓의 오버플로우를 해쉬의 오버플로우 처리 기법을 기반으로 3가지 방법으로 제안한다.

5.2 해쉬(hashing)에서의 오버플로우 처리

이 논문에서 고려하는 해쉬 방법은 버켓을 사용하는 해쉬 방법이다. 버켓(bucket)을 사용하는 해쉬 방법에서 오버플로우를 처리하는 방법은 별도의 오버플로우 영역(Sepaarate Overflow area)을 사용 방법, 확장 해쉬(Extensible hashing) 방법, 동적 해쉬(Dynamic Hashing) 방법이 있다.

해쉬의 오버플로우 처리 기법을 공간 인덱스에 적용하면 다음과 같다. (1)별도의 오버플로우 영역을 사용하는 방법은 2 개의 그리드 파일을 사용하는 트윈 그리드 화일(Twin Grid File)[7]을 응용할 수 있다. (2)확장 해쉬은 오버플로우가 발생하면 버켓 분할과 디렉토리를 확장하는 방법으로 그리드 화일[7]에 해당된다. (3) 동적 해쉬은 오버플로우가 발생하면 트리로 분할하는 방법으로 이 논문에서는 고정 그리드와 K-D-B 트리의 혼합으로 제안한다.

5.3 해쉬 기반 공간 인덱스에서의 오버플로우 처리

5.3.1 트윈 그리드 화일

트윈 그리드 화일은 스케일이 다른 두개의 그리드 파일을 사용한다. 그림 3 은 이동 객체를 알파펫 순서대로 삽입이

되었을 때의 예이다. 트윈 그리드 화일은 일반적인 그리드 파일 보다 공간 활용도(space utilization)가 높다. 오버플로우 영역을 사용함으로서 셀 분할이나 병합이 작지만 이동 객체 검색을 위해서는 두개의 그리드 파일을 검색해야 하는 단점이 있다. 셀 분할, 병합 시에도 두개의 그리드 화일에서 모두 수행해야 한다.

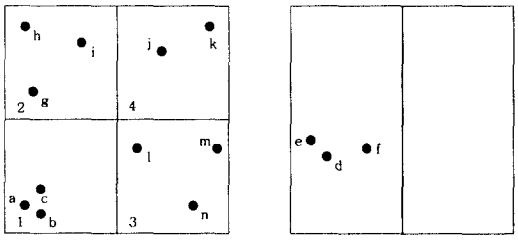


그림 3. Twin Grid File을 이용한 오버플로우 처리 예

5.3.2 그리드 화일

그림 4(b)은 이동 객체 l 이 셀3에서 셀1로 이동했을 경우로서 오버플로우가 발생하면 셀을 분할 한다. 그리드 화일은 트윈 그리드 화일 보다 셀 분할, 병합이 자주 일어난다. 셀 분할과 병합은 전체 디렉토리의 락(Lock)을 초래하여 성능 저하를 초래한다.

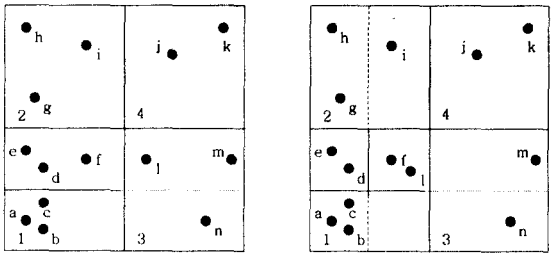


그림 4. 그리드 화일의 오버플로우 처리 예

5.3.3 동적 해쉬 방법

동적 해쉬 방법은 공간 인덱스에서 고정 그리드와 K-D-B 트리로 구성된다. 그림 5와 같이 오버플로우가 발생하면 K-D-B 트리 알고리즘으로 셀 분할한다. 셀 분할, 병합이 셀 내에서만 이루어 지기 때문에 그리드 화일에서 같이 전체 디렉토리를 변경하지 않아도 되며, 셀 별로 락을 수행할 수 있다.

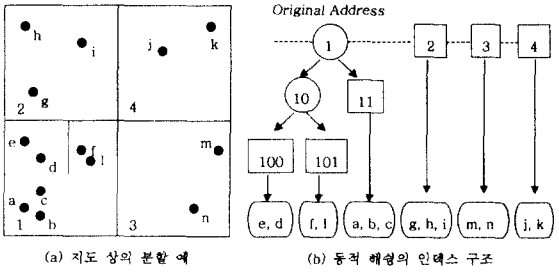


그림 5. 동적 해쉬 방법의 예

6. 이동 객체의 질의 및 질의 처리

이동 객체의 질의는 이동 객체의 식별자를 이용한 질의와 특정 영역에 있는 이동 객체를 검색하는 영역질의가 있다. 식별자를 이용한 질의는 객체 식별자를 이용하여 이동 객체의 위치를 검색한다. 영역 질의는 공간 색인을 이용하여 해당 지역에 있는 이동 객체들을 검색한다. 이때 이동 객체는 연속적으로 위치가 변하기 때문에 버퍼 질의를 수행해야 한다

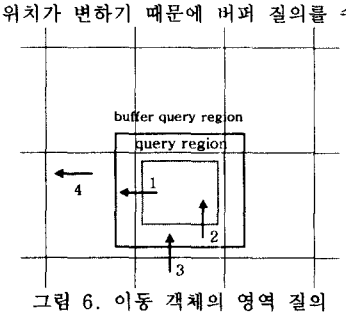


그림 6. 이동 객체의 영역 질의

질의 처리는 여과(filtering) 단계와 정제 (refinement) 단계로 진행된다. 여과 단계에서는 후보 객체들을 찾아 준다. 그림 6 과 같이 버퍼 질의를 수행하여 후보 객체들의 점을 찾는다. 그림 6 에서는 객체 1, 2 가 후보 객체이다. 정제 단계에서는 객체의 이동 속도와 보고 시간을 이용하여 이동 객체의 현재 위치를 계산한다. 정제 단계를 수행하면 객체 2 만이 질의 영역 내에 있음을 알 수 있다.

7. 결론

이 논문에서는 해쉬 기반의 인덱스 구조를 이용하여 빈번한 이동 객체의 위치 변경을 하기 위한 자료구조, 위치변경 방법, 오버플로우 처리 기법, 영역 질의 방법을 제안하였다. 이동 객체를 시작 위치, 방향, 속도를 이용하여 벡터로 표현하였으며, 공간 색인 만을 점으로 표현하여 색인의 갱신 성능을 향상 시켰다. 향후 성능 평가를 통하여 5 장에서 제안한 오버플로우 처리 방법들을 비교하고자 한다.

이동 객체에 관한 연구로 현재 및 미래위치를 추적하는 기능 외에 이동 객체의 이동 경로를 저장하여 검색하는 연구가 활발히 진행되고 있다. 이동 경로 검색은 시공간 데이터베이스 분야로서 해쉬 기반의 인덱스 구조를 발전시켜 향후 과제로 진행할 계획이다.

참고문헌

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

[2] J. Tayeb, O. Ulusoy, and O. Wolfson. " A Quadtree Based Dynamic Attribute Indexing Method." , The Computer Journal, 41(3), pp185-200, 1998

[3]. L. Forlizzi, R. H. Guting, E. N. and M. Schneider, "A data model and data structures for moving objects databases," Proceedings of the 2000 ACM SIGMOD on Management of data, pp. 319 - 330, 2000

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

[5]. S. Saltenis, C. S. Jensen, S.T. Leutenegger, M. A. Lopez, " Indexing the Positions of Continuously Moving Objects," Proceedings of the 2000 ACM SIGMOD on Management of data, pp. 331 - 342, 2000

[6] V. Gaede and O. Gunther, " Multidimensional Access Methods." , Computing Surveys 30(2), pp 170-231, 1998.