과거 궤적 색인을 위한 TB-트리의 시공간 중첩 최소화 정책

The Policy of Minimizing Spatio-Temporal Overlaps on the TB-tree for Trajectories Index

조대수*, 임덕성**, 홍봉희***
Dae-Soo Cho, Duk-Sung Lim, Bong-Hee Hong

요 약 차량과 같이 시간의 흐름에 따라 위치가 변경되는 객체를 이동체라 한다. 이동체의 과거 궤적은 시간이 지남에 따라 누적되므로 대용량 정보가 된다. 대용량 궤적 정보를 저장하는 이동체 데이터베이스에서 효율적으로 궤적을 검색하기 위해서는 색인이 필요하다. 특히 궤적을 선택하는 과정(영역 질의)과 선택된 궤적의 일부분을 추출하는 과정(궤적 질의)으로 이루어진 복합 질의를 처리하기 위해서는 궤적 보존을 지원하는 TB-tree와 같은 색인 구조가 적합하다. 그러나 TB-tree는 비단말 노드에서의 공간적인 특성을 고려하지 못하여 영역 질의시 불필요한 노드 접근이 발생하는 문제가 있다.
이 논문에서는 영역 질의를 효율적으로 처리하기 위하여, TB-tree의 비단말 노드의 사장 영역을 감소시킬 수 있는 분할 정책을 제안하고 이를 TB-tree에 적용하여 구현한다. 이 논문에서 제안하는 분할 정책은 높은 공간 활용도, 효과적인 궤적 추출과 같은 TB-tree의 장점을 유지하면서 비단말 노드의 사장 영역을 줄임으로써 영역 질의에 효과적인 특징이 있다. 제안된 분할 정책은 성능평가를 통하여 기존의 TB-tree보다 영역 질의에서 우수함을 보인다.

Abstract Objects, which change their positions over time such as cars, are called moving objects. Trajectories of a moving object have large volumes because trajectories are accumulated. Efficient indexing techniques for searching these large volumes of trajectories are needed in the moving object databases. Especially the TB-tree which supports bundling trajectories is suitable for processing combined queries which have 2 steps: first step is selecting trajectories (range search), next is selecting the parts of each trajectory (trajectory search). But the TB-tree has unnecessary disk accesses cause of lack of spatial discrimination in range queries.

In this paper, we propose and implement the splitting policy which can reduce dead spaces of non-leaf node in order to process range queries efficiently. The policy has better performance about range queries than the TB-tree as well as the advantages of the TB-tree, such as highly space utilization and efficient trajectory extraction. This paper shows that the newly proposed split policy has better performance in processing the range queries than that of the TB-tree by experimental evaluation.

주요어 : 분할 정책, 이동체 데이터베이스, 이동체 색인
KeyWords : TB-tree, Split Policy, Moving Objects Databases, Moving Objects Indexes.

* 동서대학교 인터넷공학부 dscho@dongseo.ac.kr
** 영진전문대학 컴퓨터정보기술계열 junsung@yjc.ac.kr
*** 부산대학교 전자전기정보컴퓨터공학부 bhhong@pusan.ac.kr

1. 서 론

건물이나 지형과 같은 지리 객체와 달리 차량, 선박은 시간의 흐름에 따라 자신의 위치가 변경되는 이동체(moving object)의 특성[1]을 가진다. 이동체는 이동점(moving point)과 이동영역(moving region)으로 나눌 수 있다[2]. 자동차, 비행기 등과 같이 이동체의 위치 변화만 의미를 가지는 경우 이동점으로 이동체를 표현한다. 반면에, 이동체의 위치뿐만 아니라 크기, 모양 등이 중요한 의미를 가지는 경우는 이동체를 이동영역으로 나타내며 태풍의 이동, 지번의 변화 등이 있다. 이 논문에서는 이동점으로 표현될 수 있는 이동체의 색인 방법에 대해서 다룬다.

이동점은 시간에 따라 변경된 이동체의 위치정보를 저장하고 있으므로, 시공간 차원에서하나의 궤적을 갖는다. 따라서 이동체 데이터베이스에서는 시공간 영역 내에 포함되는 이동체를 검색하는 영역 질의 뿐 아니라, 특정 이동체의 궤적정보를 검색하는 궤적 질의가 매우 빈번하게 발생되므로, 영역 질의와 궤적 질의를 효과적으로 처리할 필요가 있다. 특히, 이동체의 위치를 일정 시간 간격 또는 이동체의 속도나 방향의 변경이 있을 때 마다 저장할 경우 다수의 이동체로부터 획득된 궤적 정보의 양은 시간이 흐름에 따라 누적되어 대용량의 정보가 되므로, 이동체의 효과적인 검색을 위해서는 이동체 색인이 필요하다.

효과적인 질의처리를 위해 사용되는 이동체 색인 방법으로는 3D R-tree[3], TB-tree[5] 등이 있다. 3D R-tree는 이동체의 위치정보를 시간(1차원)과 공간(2차원)에 대해서 3차원 MBB(Minimum Bounding Box) 형태로 표현한다. 따라서 시간과 공간에 대한 색인이 각각 존재하는 것보다 3차원 영역 질의에 효과적이다. 그러나 3D R-tree와 같이 R-tree[4] 계열의 색인에서 분할 정책은 연대순(chronology)으로 삽입되는 이동체의 특성을 고려하지 않기 때문에, 동일한 이동체의 궤적이 많은 노드에 분산 저장되어 높은 궤적 추출비용이 필요하므로 궤적 질의에 효율적이지 못하다. TB-tree는 궤적 추출을 효과적으로 하기 위하여 단말 노드에 동일한 이동체의 궤적을 모아두는 궤적 번들 기법을 사용한다. 또한 서로 다른 단말 노드에 저장된 동일한 궤적간에 연결 구조가 존재하여 다시 비단말 노드를 접근하지 않고 다음 궤적이 있는 단말 노드로 이동할 수 있다. 그러나 궤적 번들 기법으로 인하여 R-tree와 같은 공간적인 특성을 고려하지 못하여 비단말 노드에서의 겹침(overlap)과 사장 영역을 증가시킴으로써 영역 질의시 노드 방문 회수를 증가시키는 문제점이 있다.

이 논문에서는 TB-tree에 대해서 효율적으로 영역 질의를 처리하기 위하여 비단말 노드의 사장 영역을 감소시킬 수 있는 새로운 분할 정책을 제안한다. 제안된 정책은 궤적 추출에 효과적인 TB-tree의 특성을 유지시킬 뿐 아니라, TB-tree에 비해서 비단말 노드의 사장 영역을 줄임으로써 영역 질의의 성능을 개선할 수 있다. 성능 평가 실험을 위하여 기존 TB-tree의 최근 시간 분할 정책과 이 논문에서 새롭게 제안하는 노드 분할 정책을 구현함으로써, 기존 TB-tree에 비하여 성능이 개선됨을 검증한다.

이 논문의 구성은 다음과 같다. 2장에서 관련 연구를 기술하고, 3장에서는 TB-tree의 특징과 공간 또는 시간 도메인 위주로 분할하였을 때 발생하는 문제점을 기술한다. 4장에서는 TB-tree의 문제점을 개선하기 위한 노드 분할 정책을 제시한다. 5장에서는 실험을 통하여 이 논문에서 제안하는 정책과 기존의 최근 시간 분할 정책에 대해서 영역 질의에 대한 성능평가를 보인다. 마지막으로 6장에서는 결론 및 향후 연구를 기술한다.

2. 관련 연구

이 장에서는 이동점을 표현하기 위한 이동체 모델링에 관한 연구를 먼저 기술하고, 다음으로 이동체에 관한 질의에 대한 연구를 기술한다. 마지막으로, 이동체 궤적 색인 방법에 대한 연구를 기술한다. 이동체 색인 방법은 다시 HR-tree[11] 등과 같이 시간 도메인을 공간 도메인과 별도로 처리하는 이동체 색인 방법과, 3D R-tree[3] 등과 같이 시공간 도메인을 동시에 처리하는 이동체 색인 방법으로 나누어 기술한다.

2.1 이동체 모델링

이동체 모델링은 크게 궤적 선분(trajectory segment)과 MBB(Minimum Bounding Box)로 이동체를 표현하는 방법과 시간 함수로 이동체를 표현하는 두 가지 방법으로 분류할 수 있다. 첫째, 궤적 선분을 사용하여 이동체를 모델링하는 방법에서는 이동체의 위치를 시공간의 점으로 표현하고, 시간적으로 연속적인 두 위치를 선형 보간법(linear interpolation)을 사용하여 3차원 선분으로 표현한다. R-tree[4] 계열의 색인에서는 궤적 선분은 3차원 MBB로 색인에 저장된다. 이 논문에서는 이동체 모델을 위해서 이 방법을 사용한다.

둘째, 시간 함수로 이동체를 모델링하는 방법에서는 시간에 대한 함수 f(t)를 사용하여 이동체를 표현한다. 시간함수로는 일반적으로 선형 함수가 많이 사용된다. 예를 들어 시간이 t0일 때 이동체의 위치가 x0이고, 이동 속도가 v라고 한다면, 시간 t에서 이동체의 위치는 f(t)= x0 + v(t-t0)으로 표현될 수 있다. 즉, 임의의 시간 tk가 주어지면 f(tk)를 계산함으로써 이동체의 위치를 찾을 수 있다. 이 방법은 이동체의 미래 위치까지 예측할 수 있으므로, TPR-tree[6]와 같이 미래 질의를 위한 색인에서 많이 사용한다.

2.2 이동체 질의

Pfoser[5]는 이동체 색인에서 사용되는 질의를 좌표기반 질의(coordinate-based query)와 궤적기반 질의(trajectory-based query)로 구분하여 정의하였다. 좌표기반 질의는 3차원 시공간 영역에서의 점 질의(point query), 영역 질의(range query), 최근접 질의(nearest neighbor query) 등이 있다. 궤적기반 질의는 enter, leave, cross, bypass와 같은 위상 질의(topological query)와 움직인 거리, 속도, 방향과 같은 항해 질의(navigational query)가 있다.

또한 Pfoser는 두 가지 종류의 질의를 복합적으로 적용한 복합질의를 정의하였다. 복합질의는 특정 궤적을 선택하고, 선택된 궤적에 대해서 궤적선분(궤적의 일부분)을 추출하는 두 단계로 구분된다.

※ 복합 질의의 예 : "2003년 11월 29일 10:00 에서 10:30 까지 부산은행 장전동지점 앞 사거리를 지난 차량들의 다음 한 시간까지의 궤적을 추출하라"

- "2003년 11월 29일 10:00에서 10:30까지 부산은행 장전동지점 앞 사거리" : 영역 질의를 통하여 궤적을 선택하는 과정
- "다음 한 시간까지 궤적을 추출하라" : 궤적을 추출하는 과정

2.3 이동체 색인

이동체 색인은 각 타임스탬프마다 2차원의 공간 색인을 생성함으로써, 시간 도메인을 공간 도메인과 별도로 취급하는 색인 방법과 시공간 도메인을 동시에 3차원으로 처리하는 색인 방법으로 나눌 수 있다[8].

각 타임스탬프마다 2차원의 공간 색인을 사용하는 기존의 궤적 색인에는 대표적으로 HR-tree[11]가 있다. HR-tree는 각 타임스탬프 마다 2차원 R-tree가 존재한다. 연속되는 두 타임스탬프간에 변화가 있는 경우에는 새로운 노드를 생성하고, 그렇지 않는 경우에는 과거의 노드를 공유한다. 따라서 HR-tree는 시간의 흐름에 따라 객체의 변화가 적은 경우에는 적합하나 객체의 위치 변화가 빈번한 경우 색인의 크기가 커지는 문제점이 있다. HR-tree는 특정 시점에 대한 점질의를 효율적으로 처리할 수 있으나, 영역 질의에 대해서는 성능이 우수하지 못하다.

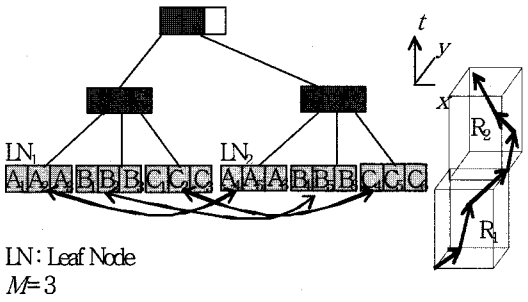
시간 도메인과 공간 도메인을 3차원 시공간 도메인으로 처리하는 이동체 궤적 색인으로써, 영역 질의와 궤적 질의를 동시에 고려한 이동체 색인으로 STR-tree[5], TB-tree[5], OP-tree[7] 등이 있다. STR-tree는 R-tree 기반이며 궤적을 처리할 수 있도록 확장하였다. STR-tree는 공간 근접성(spatial closeness)뿐만 아니라 궤적 보존(trajectory preservation)을 시도하는 색인 구조이며, 보존 파라미터(preservation parameter) p를 사용하여 공간적인 특성과 궤적 보존성을 조절할 수 있다. 보존 파라미터가 작을수록 궤적 보존성은 낮아지고 공간 근접성을 높아진다. TB-tree는 STR-tree에 비해 궤적 질의의 성능을 개선하기 위하여 단말 노드에 동일한 궤적만

을 저장하고 있다.

OP-tree는 객체 단순화 방법으로 기존의 일반적으로 사용하는 MBB 대신 MBOP (Minimum Bounding Octagon Prism)을 사용한다. MBOP가 MBB보다 더 세밀하게 객체를 단순화 시킬 수 있으므로 MBOP는 MBB에 비하여 사상 영역이 더 적다. 그러나 MBOP는 MBB보다 더 많은 정보를 사용해야 하므로 노드의 팬아웃이 낮아지는 단점이 있다.

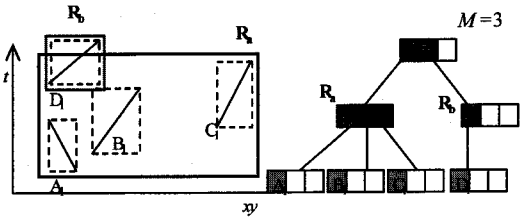
3. 문제 정의

이 장에서는 TB-tree의 특징과 TB-tree의 최근 시간 분할 정책으로 인하여 발생하는 문제점에 대하여 기술한다. TB-tree는 이동체의 위치를 샘플링하여 선형 보간법을 사용하여 궤적을 표현한다. TB-tree는 <그림 1>과 같이 하나의 단말 노드에 동일한 이동체의 궤적만을 저장하는 궤적 보존 정책을 사용하며, <그림 1>의 LN1과 LN2와 같이 동일한 이동체의 궤적을 저장하고 있는 단말 노드간의 연결 구조가 존재한다. 연결 리스트를 사용하는 논리적 궤적 표현 구조를 사용하여 궤적 추출시 상위 노드를 접근할 필요가 없이 연결 리스트를 통하여 이전 또는 다음 궤적이 있는 곳으로 이동할 수 있으므로 궤적 추출을 효과적으로 할 수 있다.

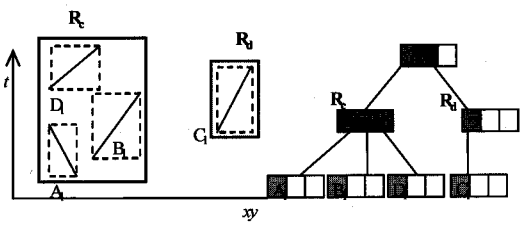


<그림 1> TB-tree의 구조

TB-tree는 분할 정책으로 최근 시간 분할 정책을 사용한다. 최근 시간 분할 정책은 단말 노드 N이 오버플로우 되었을 때, 가장 최근 시간의 엔트리만 새로운 노드에 저장한다. 따라서 <그림 1>의 LN1이 가득 차 있을 때 A4가 삽입되어 오버플로우가 발생하면, 시간적으로 가장 최근의 엔트리인 A4만 새로운 노드(LN2)에 저장된다.



(a) 시간 도메인만 고려한 분할 방식



(b) 공간 도메인만 고려한 분할 방식

<그림 2> 비단말 노드의 겹침과 사상 영역

단말 노드에서 최근 시간 분할 정책은 궤적 보존을 위해 반드시 적용되어야 한다. 이 논문에서 제시하는 TB-tree의 문제점은 이러한 최근 시간 분할 정책이 비단말 노드에서도 동일하게 적용됨으로 인해 비단말 노드에서 노드간의 겹침과 사상 영역이 증가한다는 사실이다. <그림 2>(a)는 비단말 노드의 팬아웃(fanout)이 3인 TB-tree에 시간대가 비슷한 이동체 A, B, C, D의 궤적 선분 A1, B1, C1, D1이 순서대로 삽입된 것을 표현한 것이다. 비단말 노드 Ra는 궤적 선분 A1, B1, C1을 포함한 단말 노드들로 구성되고, 비단말 노드 Rb는 최근에 들어온 궤적 선분 D1을 포함한 단말 노드를 구성하게 되어, Rb는 Ra에 겹침이 심하고 Ra는 사상 영역이 매우 크다.

반면 <그림 2>(b)는 삽입 순서가 아닌 공간 근접성을 고려하여 비단말 노드를 생성된 경우이다. 궤적 선분 C1이 가장 공간 근접성이 낮으므로 궤적 선분 A1,

B1, D1이 비단말 노드 Rc를 구성하게 되고, 궤적 선분 C1이 비단말 노드 Rd를 구성하게 되어, 두 노드 Rc와 Rd의 겹침이 없고, Rc의 사장 영역도 <그림 2>(a)의 Ra에 비하여 작으므로 영역 질의에 효율적이다. 따라서 TB-tree에 대해서 <그림 2>(b)와 같이 공간 근접성을 고려할 수 있도록 비단말 노드를 구성함으로써 기존 TB-tree에 비해 궤적 추축 성능의 저하 없이 영역 질의의 성능을 개선할 수 있다.

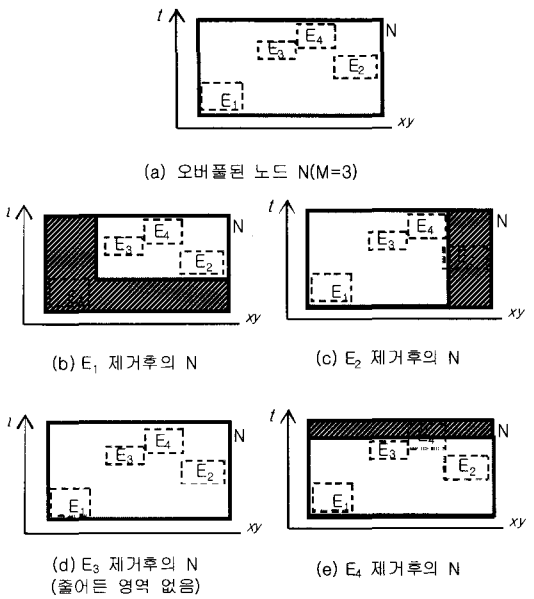
4. 노드 삽입 정책

이 장에서는 TB-tree의 궤적 보존 및 높은 공간 활용도의 장점은 유지하면서 3장에서 제시한 비단말 노드에서의 겹침과 사장 영역 문제를 개선할 수 있는 삽입 정책을 제시한다.

4.1 최대 영역 감소 정책

최대 영역 감소(MAR : Maximum Area Reduction) 정책은 TB-tree의 비단말 노드의 사장 영역과 겹침 문제를 해결하고자 한 비단말 노드 분할 정책이다. MAR 정책은 비단말 노드 N이 오버플로우 되었을 때, 분할 후의 N이 가장 작은 영역을 가지도록 분할한다. MAR 정책의 비단말 노드 분할 방법은 비단말 노드 N이 오버플로우 되었을 때, N의 엔트리를 하나씩 제거한 다음 N의 면적(부피)을 계산하여, N의 면적을 가장 최소로 만드는 엔트리를 분할 대상으로 선정하여 가장 우측에 있는 새 노드에 저장한다. 단말 노드의 경우 TB-tree와 동일한 최근 시간 분할 정책을 사용한다.

따라서 <그림 3>의 경우 <그림 3>(a)에서 E4의 삽입으로 오버플로우 되어 분할되어야 하는 경우, 그림 3(b)~(e)와 같이 각 엔트리를 하나씩 제거한 다음의 노드 영역을 측정한다. 이 경우 <그림 3>(b)와 같이 E1이 제거 되었을 때 N의 영역이 가장 최소가 되므로 E1이 분할 대상으로 선정되어 새로운 노드에 저장된다.



(a) 오버플된 노드 N(M=3)

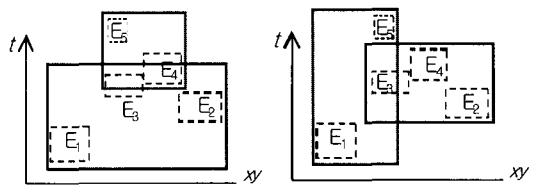
(b) E_1 제거후의 N

(c) E_2 제거후의 N

(d) E_3 제거후의 N
(줄어든 영역 없음)

(e) E_4 제거후의 N

<그림 3> MAR 정책의 분할 방법



(a) 최근 시간 분할 (b) 최대 영역 감소 분할

<그림 4> MAR 정책의 문제점

MAR 정책은 비단말 노드에 오버플로우가 발생하여 분할이 일어날 경우에 비단말 노드간의 겹침을 최소화함으로써 사장영역을 줄이는 방법이므로, 분할된 비단말 노드에 엔트리가 추가될 경우에는 MAR 정책으로 인해 오히려 시간 도메인의 겹침과 사장 영역이 커지는 문제가 발생할 수 있다. <그림 4>는 최근 시간 분할 정책에 대해서 MAR 정책의 문제점을 보이고 있다. <그림 4>(a)는 최근 시간 분할 정책에 따라 엔트리 E4를 분할 대상으로 선정하여 분할한 뒤, 새로운

엔트리 E5가 삽입되었을 때의 TB-tree를 나타낸 것이다. 반면, <그림 4>(b)는 MAR 정책에 따라 <그림 3>(a)에서 전체 영역을 가장 최소로 만드는 엔트리 E1이 분할 대상으로 선정된 후 새로운 엔트리 E5의 삽입 후를 나타낸 것이다. <그림 4>(b)가 <그림 4>(a)에 비하여 시간 도메인의 겹침이 심하고 사장 영역이 커 영역 질의의 성능을 저하시킨다.

시간 도메인 겹침 심화 문제는 <그림 4>(b)의 E1과 같이 노드의 과거 영역이 분할 대상으로 선정되면 발생할 확률이 커진다. 따라서, 시간과 공간을 고려해서 영역을 최소화하기 위해서 MAR 정책을 개선할 필요가 있다.

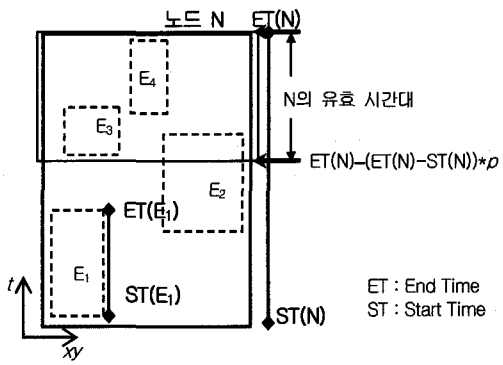
4.2 개선된 최대 영역 감소 정책

개선된 최대 영역 감소 정책(eMAR : enhanced Maximum Area Reduction)은 MAR 정책의 시간 도메인 겹침 심화 문제<그림 4>를 해결하기 위하여 MAR 정책의 분할 대상 선정시에 시간적 제약 조건을 추가한 것이다. 시간적 제약 조건은 과거 영역의 엔트리가 분할 대상이 되지 않도록 하기 위하여, 유효 시간을 만족하는 엔트리만 분할 대상 후보가 될 수 있도록 한다.

유효 시간대(effective time span)는 일정 시간 범위로써, 임의의 엔트리에 대해서 시작 시간이 유효 시간 이내에 있다면 그 엔트리는 분할 대상의 후보가 된다. 유효 시간대를 설정하기 위하여 유효 시간대 파라미터 p를 사용하여 노드 N의 전체 시간에 대한 비율로 지정한다. 관련된 용어의 정의는 <표 1>과 같다.

<표 1> 용어 정리

용어	설명
N	분할이 필요한 노드
E_i	N의 i번째 엔트리
p	유효 시간대 파라미터 ($0 \le p \le 1$)
$ET(N\|E_i)$	노드 또는 엔트리의 최종 시간(end time)
$ST(N\|E_i)$	노드 또는 엔트리의 시작 시간(start time)

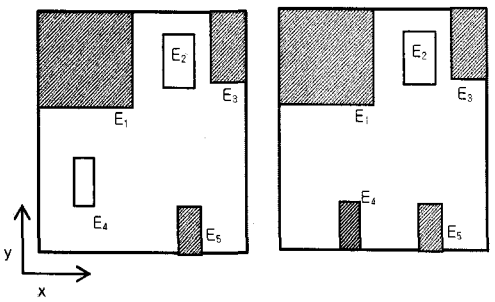


<그림 5> 유효 시간대

노드 N의 유효시간은 ET(N) - (ET(N)-ST(N))*p에서 노드의 최종 시간(ET(N))까지를 의미한다. MAR 정책과 달리 eMAR은 $ST(Ei) \ge ET(N) - (ET(N) - ST(N))*p$의 조건을 만족하는 엔트리만 후보로 하여 분할 대상을 선정한다. <그림 5>의 경우 E3와 E4만 유효시간 이내에 있으므로 분할 대상 후보가 된다.

만약, 유효 시간대 파라미터 p가 1이라면, 노드 N에 포함된 모든 엔트리 Ei에 대해서$ST(Ei) \ge ST(N)$로써 모든 엔트리가 분할 대상으로 선정될 수 있으므로, MAR 정책과 동일하다. 반면 유효 시간대 파라미터 p가 0인 경우에는 노드 N에 포함된 모든 엔트리 Ei에 대해서 $ST(Ei) \ge ET(N)$로써 분할 대상을 선정할 수 없으므로, TB-tree와 같이 최근 시간 분할 정책을 따른다. 따라서 유효 시간대 파라미터 p가 0인 경우는 TB-tree와 동일하다. 이와 같이 eMAR 정책은 MAR 정책과 기존 TB-tree의 분할 정책을 모두 포함할 수 있는 정책이다.

그런데, MAR 정책과 eMAR 정책은 분할 발생시 기존 TB-tree의 최근 시간 분할 정책에 비하여 연산량이 많으므로 삽입 속도가 저하되는 문제가 있다. 따라서 분할 대상 엔트리를 선정하기 위해 필요한 연산량을 줄여 삽입 속도를 개선할 필요가 있다. 이를 위하여 MBB 면(face)에 접하지 않는 엔트리는 제거하더라도, MBB의 면적(부피)이 줄어들지 않는다는 특성을 이용한다.



(a) {E1, E3, E5}　　(b) {E1, E3}

<그림 6> 분할 대상 엔트리 후보

<그림 6>(a)에서 E1, E3, E5는 제거하면 면적이 줄어들지만, E2, E4는 MBB 면에 접하지 않으므로 제거하여도 MBB의 면적은 줄어들지 않으므로, 분할 대상 엔트리 후보는 {E1, E3, E5}이다. 그러나 MBB 면에 접한 엔트리를 제거하였다고 해서 반드시 MBB의 면적이 줄어드는 것은 아니다. <그림 6>(b)의 E4(E5)는 MBB면에 접하고 있지만, E4(E5)를 제거하더라도 E5(E4)로 인하여 MBB의 면적은 줄어들지 않으므로, 이 경우에 분할 대상 엔트리 후보는 {E1, E3}이다. 이와 같이 영역에 대한 특성을 이용하여 모든 엔트리를 대상으로 후보를 선정하는 것이 아니라 MBB 면에 접하지 않은 엔트리를 제외한 나머지 엔트리만 분할 대상의 후보로 설정하여, 후보의 수를 줄여 삽입 성능을 개선할 수 있다.

4.3 알고리즘

이 절에서는 4.1과 4.2에서 제안된 최대 영역 감소 정책과 개선된 최대 영역 감소 정책의 알고리즘을 제시한다. 개선된 최대 영역 감소 정책과 최근 시간 분할 정책의 차이점은 비단말 노드 분할시 분할 대상 선정 방법에 있다. TB-tree의 최근 시간 분할정책은 가장 최근에 삽입된 엔트리를 분할대상으로 설정하는 반면, 개선된 최대 영역 감소 정책의 경우는 유효 시간대 내에서 임의의 엔트리를 제거하였을 때 노드의 영역을 가장 최소로 만드는 엔트리를 분할 대상으로 선정한다.

Algorithm Insert(N, E)	
INS1	Invoke FindNode(N, E)to select a leaf node N′ which has a predecessor of E
INS2	If node N′ is found, Insert E else Create a new leaf node N′′ for E Insert E
INS3	Invoke AdjustTree(N′)

(알고리즘 1) 삽입 알고리즘

알고리즘 1 Insert는 색인에 새로운 엔트리 E를 삽입하는 알고리즘이다. 먼저 FindNode를 수행하여 이전에 보고된 위치가 있는지 확인한 다음, 보고된 위치가 있다면 그 노드에 삽입을 수행하여 궤적을 보존한다. 이전에 보고된 위치가 없다면 새로운 이동체가 처음으로 위치를 보고한 것이므로 새로운 이동체를 위하여 새로운 단말 노드를 생성한다. 삽입한 뒤에는 AdjustTree를 호출하여 변경된 MBB를 상위 노드로 전파해야 하며 필요한 경우 오버플로우를 처리하도록 한다. 여기서 FindNode는 TB-tree의 것과 동일한 알고리즘이며, 재귀호출을 사용하여 각 노드에 대하여 주어진 엔트리 E와 겹침이 발생하는 노드를 순회하면서 삽입될 궤적 선분의 선행 위치(predecessor)를 찾는 역할을 한다.

Algorithm AdjustTree(N)	
AT1	If N is the root Return
AT2	If N is overflow Invoke Split(N)
AT3	Let P be the parent node of N, and let EN be N's entry in P
AT4	Adjust MBB(EN) so that it tightly encloses all entry boxes in N
AT5	Set N = P and repeat from AT1

(알고리즘 2) 전파 알고리즘

알고리즘 2 AdjustTree는 삽입이나 분할로 인하여 변경된 MBB를 상위 노드에 전파하는 알고리즘이며 N이 루트 노드에 도달할 때까지 스택 정보를 이용하

여 상위 노드로 전파한다. 오버플로우가 발생하여 분할을 해야 할 경우 Split을 호출하여 분할을 처리하도록 한다.

Algorithm Split(N)	
S1	If N is a leaf node Create new leaf node Make trajectory link between old node and new one. else Invoke SplitNonleafNode(N)

(알고리즘 3) 분할 알고리즘

알고리즘 3 Split은 분할이 발생한 노드 N이 단말 노드 또는 비단말 노드인지 확인하여 단말 노드인 경우 기존의 TB-tree와 마찬가지로 최근 시간 분할 정책을 사용하여 분할하고 분할 전후의 두 노드에 대하여 궤적 연결 리스트를 생성한다. 비단말 노드인 경우 단말 노드와 다른 분할 정책을 사용하므로 SplitNonleafNode를 호출하여 처리한다.

Algorithm SplitNonleafNode(N)	
SNN1	For each E which is satisfied MBB area property, calculate d = volume(N) volume(N-E)
SNN2	Select E′ which makes largest d
SNN3	If rightmost node N′′ whose level is same as N has space Insert E′ to N′′ else Create a new non-leaf node for E′
SNN4	Invoke AdjustTree(N′′)

(알고리즘 4) 비단말 노드 분할 알고리즘(MAR)

알고리즘 4는 최대 영역 감소(eMAR) 정책의 SplitNonleafNode 알고리즘이다. 최근 시간분할 정책과 최대 영역 감소 정책 및 개선된 최대 영역 감소 정책의 실질적인 차이가 존재하는 알고리즘이며 비단말 노드 N이 오버플로우 되었을 때 호출된다. MBB 영역 속성을 만족하는 엔트리 E에 대하여 노드 N의 면적과 후보 엔트리 E를 제거한 노드 N′의 면적 차 d를 계산한다. 영역 감소가 가장 큰 d를 가지는 엔트리를 분할 대상으로 선정한다. 선정된 분할 대상에 대해서는 TB-tree와 동일하게 색인의 가장 오른쪽 패스(rightmost path)에 저장한다.

Algorithm SplitNonleafNode(N)	
SNN1	For each E which is satisfied MBB area property and effective time span, calculate d = volume(N) volume(N-E)
SNN2	Select E′ which makes largest d
SNN3	If rightmost node N′′ whose level is same as N has space Insert E′ to N′′ else Create a new non-leaf node for E′
SNN4	Invoke AdjustTree(N′′)

(알고리즘 5) 비단말 노드 분할 알고리즘(eMAR)

알고리즘 5는 개선된 최대 영역 감소(eMAR) 정책의 SplitNonleafNode 알고리즘이다. 알고리즘 5는 알고리즘 4에 비하여, 분할 대상 후보 선정시 유효 시간 이내에 있는지를 검사하는 부분이 추가되었다. MBB 영역 속성을 만족하는 엔트리중 유효 시간 이내에 있는 엔트리를 후보로 설정한 뒤, 각 후보 엔트리 E에 대하여 노드 N의 면적과 후보 엔트리 E를 제거한 노드 N′의 면적 차 d를 계산한다. 영역 감소가 가장 큰 d를 가지는 엔트리를 분할 대상으로 선정한다.

선정된 분할 대상에 대해서는 TB-tree와 동일하게 색인의 가장 오른쪽 패스(*rightmost path*)에 저장하여 분할이 발생한 노드 N의 공간 활용도는 100%를 유지하므로 TB-tree와 마찬가지로 동일하게 높은 공간 활용도를 유지하여 색인의 높이를 낮추는데 유리하다.

5. 성능 평가

이 장에서는 논문에서 제안하는 eMAR 정책의 성능을 평가하기 위하여 eMAR 정책을 TB-tree에 적용하여 TB-tree의 최근 시간 분할 정책과 성능을 비교하고자 한다. 성능 평가는 노드에 접근할 때마다 노드 I/O 횟수를 계산하였으며, 캐쉬(cache)는 없는 것으로 가정하였다.

5.1 실험 데이터

이동체 생성을 위한 대표적인 도구로 GSTD(Generate Spatio-Temporal Data) 생성기[14]와 Brinkhoff의 네트워크 데이터 생성기[15] 그리고 IBM의 CitySimulator[16]가 있다. 이 논문에서는 GSTD와 네트워크 데이터 생성기를 사용하여 생성된 데이터셋(dataset)에 대하여 실험을 하였다.

<표 2> 데이터셋의 종류(GSTD)

분포	이동체수	보고 회수
LR	100	1,000
P	100	1,000
UA1	100	1,000
UA2	100	1,000

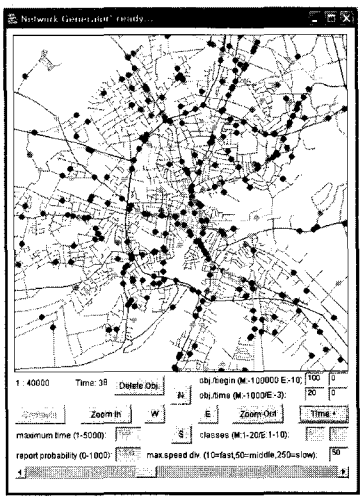
GSTD로 생성한 데이터셋은 이동체의 위치 정보를 시공간의 점으로 표현되지만, 성능 평가를 위해서 색인을 생성할 경우에, 이동체의 궤적은 시공간의 선분으로서 표현되어야 한다. 이 논문에서는 GSTD에 의해 생성된 데이터셋에서 동일 궤적 ID를 가지는 점들에 대해서 시간 순서상 연속되는 두 점을 하나의 선분으로 변환하여 색인에 삽입하였다. GSTD를 사용하여 생성한 데이터셋은 <표 2>와 같으며, 각각의 데이터셋은 이동체의 이동 분포에 따라 구분된다. 데이터셋 LR은 이동체들이 왼쪽에서 오른쪽으로 이동하는 분포이며, P는 초기 데이터가 점차 북동쪽으로 퍼지는 분포를 의미한다. 그리고, UA1은 초기 데이터가 느린 속도로 전체 영역으로 퍼지는 분포이며, UA2는 UA1과 유사하지만 빠른 속도로 이동하는 분포를 의미한다.

<표 3> 네트워크 데이터셋 생성 파라미터

분포	네트워크 데이터
초당 객체 생성수	20
이동체 속도	중간
보고 확률	50%

네트워크 데이터셋은 [15]에서 제안한 방법에 의해 생성되었으며, 사용한 파라미터는 <표 3>과 같다.

<그림 7>은 파라미터에 따라 네트워크 데이터셋을 생성하는 과정을 보이고 있다.

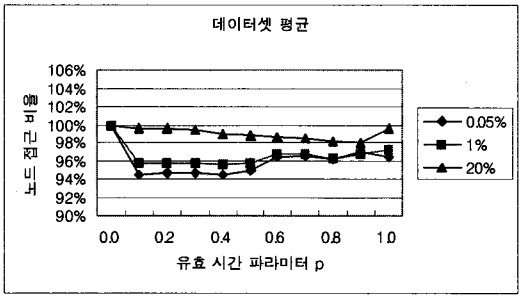


<그림 7> 네트워크 데이터 생성기

5.2 질의 성능 비교

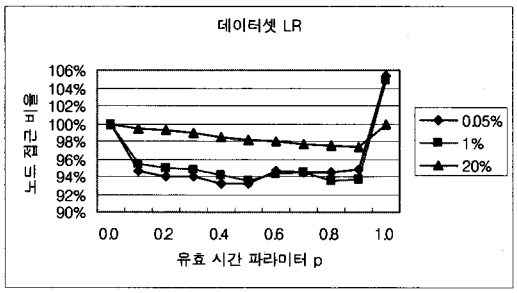
이 논문에서 생성된 실험 데이터셋들에 대해서 기존 TB-tree의 최근 시간 분할 정책과 이 논문에서 제안하는 eMAR 정책에 따라 각각 색인을 생성하고, 성능을 비교한다. 성능 평가 실험은 eMAR 정책을 통해 색인을 생성할 때, 유효 시간대 파라미터 $p(0 \le p \le 1)$에 따른 질의 처리 성능을 비교한다. 질의 처리 성능은 각 질의를 처리하기 위해 필요한 노드접근 횟수로써 평가하며, 기존의 최근 시간 분할 정책에 의해 생성된 색인의 노드 접근 횟수에 대해서 제안된 방법으로 생성된 색인의 노드 접근 횟수의 비율로써 표시한다. 즉, 노드접근 비율이 100%라는 것은 기존 방법과 동일한 성능을 의미하며, 80%라는 것은 기존 방법에 비해 20%의 성능이 개선되었음을 의미한다.

유효 시간 변화에 따른 영역 질의 성능을 비교하기 위하여 각 데이터셋에 대하여 시공간의 각 축에 대하여 0.05%, 1%, 20% 의 시공간 영역 질의를 각각 1,000회 수행하여 노드 접근 횟수를 측정하였다.

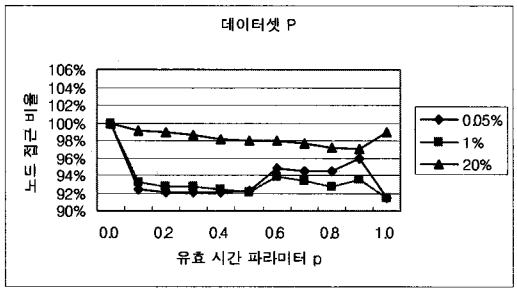


<그림 8> 영역 질의 성능(GSTD평균)

<그림 8>은 GSTD로 생성된 데이터셋의 영역 질의 처리 성능의 평균값을 보이고 있다. 작은 영역 질의인 시공간 축당 0.05% 영역 질의와 시공간 축당 1% 영역 질의에 대해서는 유효 시간대 파라미터 p=0.4에서 각각 평균 약 6%, 5%의 질의 성능 향상이 있었다. 반면에, 큰 영역 질의인 시공간 축당 20% 영역 질의에서는 유효 시간대 파라미터 p=0.9 일때 가장 좋은 성능을 보였다.



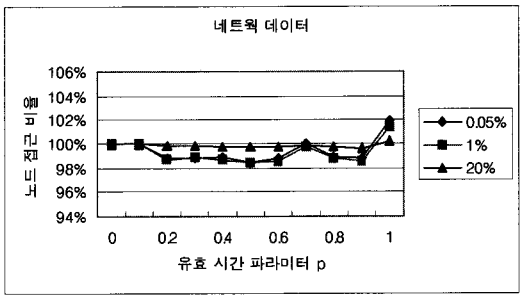
(a) 데이터셋 LR에 대한 영역 질의



(b) 데이터셋 P에 대한 영역 질의

<그림 9> 영역 질의 성능(LR, P)

<그림 9>는 데이터셋 LR과 데이터셋 P에 대한 성능 평가 결과를 보이고 있다. 데이터셋 LR<그림 9>(a)에서는 유효 시간대 파라미터 p=1에서 기존 방법에 비해 성능이 저하됨을 알 수 있다. 즉, 과거 영역의 엔트리가 분할 대상으로 선정됨으로 인해 시간 도메인의 겹침 심화 현상이 나타난 것으로 판단된다. 따라서 이 경우에는 유효 시간을 줄여 과거 영역이 분할 대상으로 선정되지 않도록 해야 한다. 시공간 축당 0.05% 영역 질의의 경우 유효 시간대 파라미터 p=0.4에서, 시공간 축당 20% 영역 질의의 경우 유효 시간대 파라미터 p=0.9에서 가장 좋은 성능을 보였다. 데이터셋 P<그림 9>(b)에서는 유효 시간대 파라미터 p=1.0에서 시공간 축당 0.05%, 1% 질의의 경우에 모두 약 9%의 성능 향상이 있었다.



<그림 10> 네트워크 데이터셋

<그림 10>은 네트워크 데이터셋에 대하여 동일한 질의를 수행한 결과이다. 네트워크 데이터셋은 시간이 흐를수록 기존의 이동체는 삭제되고, 새로운 이동체가 계속 보고되는 특징으로 인하여 성능 향상의 정도는 GSTD 보다 적지만 LR 데이터셋과 비슷한 결과를 나타낸다. 시공간 축당 0.05%와 1%와 같이 작은 영역 질의에 대해서는 유효 시간대 파라미터 p=0.5에서, 시공간 축당 20%와 같이 큰 영역 질의에 대해서는 유효 시간대 파라미터 p=0.9에서 가장 좋은 성능을 보였다.

6. 결론 및 향후 연구

이 논문에서는 TB-tree에 대해서 비단말 노드에서의 겹침과 사장 영역이 커지는 문제를 정의하고, 이

문제를 해결하기 위하여 새로운 노드 분할 정책을 제안하였다. 이 논문에서 제안한 eMAR 정책은 궤적을 번들하는 TB-tree의 특성을 유지하고 있으며, 공간 활용도가 높아 대용량의 이동체 궤적 데이터 색인으로 적합하다. 또한, 기존 TB-tree의 최근 시간 분할 정책에 비하여 비단말 노드의 사장 영역 및 겹침을 줄여 영역 질의의 비용이 줄어드는 특징이 있다. 시공간 축당 0.05%, 1%와 같은 작은 영역 질의에 대해서는 유효 시간대 파라미터 p가 0.4 일 때 가장 우수한 성능을 보이며, 시공간 축당 20%와 같은 큰 영역 질의에 대해서는 유효 시간대 파라미터 p를 0.9로 설정하는 것이 영역 질의에 효과적인 것을 실험을 통하여 확인하였다.

향후 연구로써 이동체의 수가 색인 생성과정에서 동적으로 변하는 데이터에 대한 성능 평가 실험과 서로 다른 이동체 분포에 적합한 유효 시간대 파라미터를 결정하는 방법에 대한 연구가 필요하다. 또한 시간 도메인과 공간 도메인의 상관관계에 대한 수학적 질의 모델에 대한 연구가 필요하다.

참고문헌

[1] Zhexuan Song, Nick Roussopoulos, "Hashing Moving Objects", International Conference on Mobile Data Management, pp. 161-172, 2001.

[2] Martin Erwig, Ralf Hartmut Gutting, Markus Schneider, Michalis Vazirgiannis, "Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases", GeoInformatica vol. 3(3), pp. 269-296, 1999.

[3] Yannis Theodoridis, Michael Vazirgiannis, Timos Sellis, "Spatio-Temporal Indexing for Large Multimedia Applications", IEEE International Conference on Multimedia Computing and Systems, pp. 441-448, 1996.

[4] Antonm Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching", Proceedings of ACM-SIGMOD Conference on the Management of Data, pp. 47-57, 1984.

[5] Dieter Pfoser, Christian S. Jensen, Yannis Theodoridis, "Novel Approaches to the Indexing of Moving Objects", Proceedings of International Conference on Very Large Data Bases, pp. 395-406, 2000.

[6] Simonas Saltenis, Christian S. Jensen, Scott T. Leutenegger, Mario A. Lopez, "Indexing the Positions of Continuously Moving Objects", Proceedings of ACM-SIGMOD Conference on the Management of Data, pp. 331-342, 2000.

[7] Hongjun Zhu, Jianwen Su, Oscar H. Ibarra, "Trajectory Queries and Octagons in Moving Object Databases", Proceedings of the ACM Conference on Information and Knowledge Management, pp. 413-421, 2002.

[8] Marios Hadjieleftheriou, George kollios, Dimitrios Gunopulos, Vassilis J. Tsotras, "Efficient Indexing of Spatiotemporal Objects", Proceedings of Extending Database Technology, pp. 251-268, 2002.

[9] Dongseop Kwon, Sangjun Lee, Sukho Lee, "Indexing the Current Positions of Moving Objects Using the Lazy Update R-tree", International Conference on Mobile Data Managemnet, pp. 113-120, 2002.

[10] Norbert Beckmann, Hans-Peter Kriegel, "The R*-tree: An Efficient and Robust Access Method for Point and Rectangles", Proceedings of ACM990.

[11] Mario A. Nascimento, Jefferson R. O. Silva, "Towards Historical R-trees", Proceedings of ACM Symposium on Applied Computing, pp. 235-240, 1998

[12] Yufei Tao, Dimitris Papadias, "MV3R-Tree: A Spatio-Temporal Access Method for Timestamp and Interval Queries", Proceedings of International Conference on Very Large Data Bases, pp. 431-440, 2001.

[13] V. Prasad Chakka, Adam C. Everspaugh, Jignesh M. Patel, "Indexing Large Trajectory Data Sets With SETI", Proceedings of the 2003 CIDR Conference, 2003.

[14] Yannis Theodoridis, Jefferson R. O. Silva, Mario A. Nascimento, "On the Generation of Spatiotemporal Datasets", Advanced in Spatial Databases, pp. 147-164, 1999.

[15] Thomas Brinkhoff, "A framework for Generating Network-Based Moving Objects," GeoInfomatica vol.6(2), pp. 153-180, 2002.

[16] http://www.alphaworks.ibm.com/tech/citysimulator

조대수

1995년 부산대학교 컴퓨터공학과 졸업 (공학사)

1997년 부산대학교 컴퓨터공학과 졸업 (공학석사)

2001년 부산대학교 컴퓨터공학과 졸업(공학박사)

2001년 ~ 2004년 한국전자통신연구원 텔레매틱스연구단 선임연구원

2004년 ~ 현재 동서대학교 인터넷공학부 전임강사

관심분야 : GIS, 공간데이터베이스, LBS, 이동체 데이터베이스 등

임덕성

1998년 동아대학교 컴퓨터공학과 졸업 (학사)

2000년 부산대학교 컴퓨터공학과 졸업 (공학석사)

2002년 부산대학교 컴퓨터공학과 박사수료

2003년 ~ 현재 영진전문대학 컴퓨터정보계열 전임강사

관심분야 : GIS, 시공간 데이터베이스, 이동체 데이터베이스

홍봉희

1982년 서울대학교 컴퓨터공학과 졸업 (공학사)

1984년 서울대학교 컴퓨터공학과 졸업 (공학석사)

1988년 서울대학교 컴퓨터공학과 졸업(공학박사)

1987년 ~ 현재 부산대학교 컴퓨터공학과 교수

2002년 ~ 현재 한국공간정보시스템학회 상임이사

2004년 ~ 현재 차세대 물류 IT 사업단 단장

관심분야 : 데이터베이스, GIS, RFID 미들웨어, USN (유비쿼터스 센서 네트워킹)