효율적인 시공간 영역 질의 처리를 위한 궤적 색인의 노드 재배치 전략

(A Node Relocation Strategy of Trajectory Indexes for Efficient Processing of Spatiotemporal Range Queries)

임 덕 성 † 조 대 수 †† 홍 봉 희 †††
(Duksung Lim) (Daesoo Cho) (Bonghee Hong)

요 약 TB-tree와 같이 시공간에서 궤적 검색을 위한 색인 구조는 단말 노드에 하나의 궤적만을 저장하는 궤적 보존의 특성을 가지기 때문에 궤적의 일부분을 추출하는 항해 질의(Navigational Query)에서 우수한 성능을 보인다. 그러나, 궤적 보존을 위해 공간적 지역성을 완전히 배제하는 구조를 가짐으로써 비단말 노드의 MBR(Minimum Bounding Rectangle)은 큰 사장 영역을 가지는 단점이 있다. 사장 영역 증가는 노드간의 중첩을 높이는 원인을 제공하기 때문에 영역 질의의 성능을 저하시키는 문제가 있다.

이 논문에서는 궤적 검색을 위한 색인 구조에서 항해질의 성능을 유지하면서 영역 질의의 성능을 향상시키기 위한 비단말 노드 분할 정책과 엔트리 재배치 정책을 제시한다. 분할 정책은 비단말 노드의 분할 시 비단말 노드의 MBR을 최대한 감소시키는 최대 영역 축소(Maximal Area Reduction) 정책을 사용하고, 엔트리 재배치 정책은 비단말 노드를 구성하는 다수의 엔트리에서 MBR을 최대로 감소시킬 수 있는 엔트리의 위치를 재배치시키는 방법으로 이 논문에서는 분할 방법에 따라 2가지 재배치 전략을 제시하고 TB-tree와 성능을 비교한다.

키워드 : 시공간 데이타베이스, 궤적 색인, 궤적 질의

Abstract The trajectory preservation property that stores only one trajectory in a leaf node is the most important feature of an index structure, such as the TB-tree for retrieving object's moving paths in the spatio-temporal space. It performs well in trajectory-related queries such as navigational queries and combined queries. But, the MBR of non-leaf nodes in the TB-tree have large amounts of dead space because trajectory preservation is achieved at the sacrifice of the spatial locality of trajectories. As dead space increases, the overlap between nodes also increases, and, thus, the classical range query cost increases.

We present a new split policy and entry relocation policies, which have no deterioration of the performance for trajectory-related queries, for improving the performance of range queries. To maximally reduce the dead space of a non-leaf node's MBR, the Maximal Area Reduction (MAR) policy is used as a split policy for non-leaf nodes. The entry relocation policy induces entries in non-leaf nodes to exchange each other for the purpose of reducing dead spaces in these nodes. We propose two algorithms for the entry relocation policy, and evaluate the performance studies of new algorithms comparing to the TB-tree under a varying set of spatio-temporal queries.

Key words : spatio-temporal database, trajectory indexing, trajectory-based query

· 본 연구는 한국전자통신연구원의 개방형LBS 핵심기술개발 과제에서 일부 지원함
† 정 회 원 : 영진전문대학 컴퓨터정보기술계열 교수
junsung@yjc.ac.kr
†† 종신회원 : 동서대학교 인터넷공학부 교수
dscho@dongseo.ac.kr
††† 종신회원 : 부산대학교 컴퓨터공학과 교수
bhhong@pusan.ac.kr
논문접수 : 2003년 12월 23일
심사완료 : 2004년 9월 14일

1. 서 론

위치 기반 서비스(Location-Based Service)는 무선 통신에 기반 한 서비스로서 최근 그 중요성이 증대되고 있다. 시공간상에서 이동체의 연속적인 이동으로 생성되는 이동체 궤적에 대한 위치 기반 질의를 처리하기 위해서는 이동체의 궤적 정보를 관리하는 이동체 데이타베이스 기술 개발이 선행되어야 한다. 이동체 데이타베

이스는 물류 시스템, 항해 시스템, 기상 정보 시스템, 항공 교통 통제 시스템, 디지털 전장 등과 같은 많은 응용 서비스를 위한 기반 기술로서 이동체의 궤적 정보를 효과적으로 관리하고, 빠른 검색을 제공하는 이동체 색인 방법 필요하다.

이동체의 현재 위치를 공간 상의 2차원 점으로 모델링 하면, 이동체의 이동 경로(궤적)는 시공간상의 3차원 다중선으로 표현된다. TB-tree[1]와 같이 시공간에서 궤적 검색을 위한 색인 구조는 단말 노드에 하나의 궤적만을 저장하는 궤적 보존의 특성을 가지기 때문에 궤적의 일부분을 추출하는 것과 같은 항해 질의(Navigational Query)[1]에서 우수한 성능을 보인다. 그러나, 궤적 보존을 위해 공간적 지역성을 완전히 배제하는 구조를 가짐으로써 비단말 노드에서 큰 사장 영역을 가지는 단점이 있다. 사장 영역 증가는 노드간의 중첩이 높이는 원인을 제공하기 때문에 영역 질의의 성능을 저하시키는 문제가 있다.

이 논문에서는 궤적 검색을 위한 색인 구조에서 항해 질의 성능을 유지하면서 영역 질의의 성능을 향상시키기 위한 분할 방법과 노드 재배치 방법을 제시한다. 분할 정책은 비단말 노드의 분할 시 비단말 노드의 MBR을 최대한 감소시키는 엔트리와 나머지 엔트리로 분할하는 최대 영역 축소(Maximal Area Reduction) 정책을 사용하고, 엔트리 재배치 방법은 비단말 노드의 MBR을 구성하는 다수의 엔트리에서 MBR을 최대로 감소시킬 수 있는 엔트리의 위치를 재배치시키는 방법이다. 재배치 방법은 사장 영역을 감소시켜 비단말 노드간의 중첩을 줄이는 방법으로 분할 방법에 따라 TB-tree의 최근 시간 분할 정책을 사용하는 ERNN(Entry Relocation in New Node)알고리즘과 최대 영역 축소 정책을 사용하는 ERFN(Entry Relocation in Full Node)알고리즘을 제시한다.

R-tree[2]에서 적용되는 삽입 정책인 최소 영역 증가(Least Area Enlargement)정책은 공간적 지역성에 기반한 삽입 정책이다. 이 논문에서 제시하는 분할 정책과 노드 재배치 정책은 기존 궤적 기반 색인에서 궤적 보존의 특성만을 유지하는 단점을 보완하기 위하여 궤적 보존으로 생성된 단말 노드의 위치를 공간적으로 인접한 위치에 동적으로 재배치 시킴으로써 궤적 보존과 공간적 지역성을 함께 고려한 궤적 삽입 정책이라 할 수 있다.

이 논문의 구성은 다음과 같다. 2장에서는 궤적 색인을 위한 관련 연구를 기술하고, 3장에서는 먼저 궤적 색인 구조의 문제점을 정의하고, TB-tree의 영역 질의 성능을 향상을 위한 최대 영역 감소 정책 및 엔트리 재배치 정책에 대한 ERNN과 ERFN알고리즘을 제시한다. 4장에서는 TB-tree와 제시한 알고리즘들에 대한 성능을 비교하고, 마지막으로 5장에서는 결론과 향후 과제를 기술한다.

2. 관련 연구

시공간 환경에서 궤적 검색은 크게 두 가지 문제로서 미래 위치에 대한 궤적 검색에 대한 접근 방법과 과거 위치에 대한 궤적 검색에 대한 접근 방법으로 분류된다. 이 논문에서는 이동체의 과거 궤적만을 다루고 궤적 질의라 함은 이동체의 과거 궤적에 대한 질의를 의미한다.

2.1 과거 궤적을 위한 시공간 색인에 대한 연구

효율적인 과거 궤적 검색을 위한 색인에 대한 현재까지의 연구는 크게 시간 흐름에 따른 연대기성을 고려하는 접근 방법과 시간을 공간의 다른 차원으로 고려하는 접근 방법으로 분류된다[3].

시간 흐름에 따른 연대기성을 고려하는 접근 방법은 시간의 순서에 따라 삽입되는 이동체의 궤적을 2차원의 색인을 이용하여 시간 순서로 저장하는 방법으로 특정 타임스탬프 t에 삽입된 이동체의 궤적을 2차원 색인으로 구성하고, 각 타임스탬프마다 구성된 2차원 색인을 이용하여 이동체의 궤적을 검색한다. 이와 같이 연대기성을 고려하는 접근 방법은 두 가지 개념인 중첩 기법과 다중 버전 기법에 기초로 한다[4].

중첩(Overlapping) 기법은 시간과 공간에 대한 효율적인 파일 공유기법으로 소개되었다. B-tree에 overlapping기법이 적용된 것이 overlapping B-tree(OVB-tree), R-tree에 적용된 것이 Historical R-tree(HR-tree), Linear Quad-tree에 적용된 것이 overlapping Linear Quad-tree이다. 중첩 기법은 타임스탬프 t에서 생성된 색인에서 t+1 시간에 색인 생성시 변화가 없는 노드를 공유하는 방법으로 동일한 질의 성능을 유지하면서 색인의 크기를 감소시키는 효과가 있다.

다중 버전(Multiversion)기법은 WORM(write once read many)구조에서 WOB-tree나 TimeSplit B-tree (TSB-tree)[5]와 같이 버전 데이타에 대한 저장/검색 방법을 위해 소개되었다. 다중 버전 기법은 개념적으로 각 타임스탬프마다 색인을 생성시키는 면에서 중첩 기법과 유사하지만, 시간이 진행성에 따라 객체의 진화를 표현하는 기법으로 전체 색인 크기가 객체의 변화량에 비례하는 특성을 가진다. Multiversion B-tree[6]는 다중 버전 기법을 B-tree에 적용하였고, MVAS[7]에서는 거래 시간 데이타베이스(Transaction-Time DataBases)에서 효율적인 다중 버전 접근 구조를 제시하였다. 시공간 데이타베이스(Spatio-Temporal DataBases)에서 MV3R-tree[8]는 MVB-tree를 기반으로 타임슬라이스 질의 및 영역 질의를 모두 효율적으로 다루기 위한 색인 구조로

서 제시되었고, PPR-tree[9]는 Bitemporal Database에서 R-tree의 시간에 따른 상태를 Multiversion기법을 이용하여 색인의 크기를 줄이는 동시에 검색 성능을 향상시켰다. [4]에서는 다중 버전 기법과 중첩 기법을 고려한 MLTS(Multiple logical-tree structures)의 비용 모델을 제시하였다.

연대기성을 고려하는 접근 방법에서 타임스탬프 사이의 이동체 검색은 높은 비용을 가지는 단점이 있다. 즉, 두 타임스탬프를 획득하여 보간 방법을 이용하여 검색하는 것과 같이 추가적인 질의 비용을 제공하거나 색인의 크기에 대한 비용을 추가하여 정밀한 타임스탬프 간격을 제공하여 검색을 가능하게 하는 것과 같이 추가적인 비용을 수반한다.

시간을 공간의 다른 차원으로 고려하는 접근 방법은 이동체와 같은 시공간 객체의 시간차원을 공간의 또 다른 차원으로 표현하여 2차원 공간의 이동체를 시간 차원을 포함하여 3차원 저장 구조에 저장하여 검색하는 방법이다. 객체의 최소 영역 사각형(MBR)을 저장하여 검색하는 다차원 색인 구조인 R-tree[2]와 R*-tree[10]에서는 n차원 공간을 점유하는 이동체의 시간 간격에 따른 위치를 n+1 차원 MBR로서 표현한다. MBR은 이동체의 시간 간격을 시간축에 대한 높이로서 표현하고 시간 간격 동안의 공간적 위치는 n차원 공간의 영역을 표현한다. R-tree에 시간적 차원을 추가한 시공간 인덱스로서 3D R-tree[11]가 제시되었으나, 색인의 사장 공간이 클 뿐만 아니라 노드간의 중첩이 매우 큰 단점을 가진다. 또한 시공간적 지역성만을 고려하기 때문에 이동체의 이동 경로를 검색하는 궤적 질의의 성능이 낮은 단점이 있다. Octagon Prism-tree(OP-tree)[12]는 영역 및 궤적 질의의 성능을 향상시키기 위해 R-tree계열에서 가장 일반적인 객체의 단순화 방법인 사각형을 사용하는 대신 팔각형을 이용하는 방법을 사용하였다. 팔각형의 단순화 방법은 사각형을 이용한 방법에 비해 사장 영역을 감소시키는 이점을 가지지만 팔각형을 엔트리에 저장하기 위해 추가되는 공간으로 노드의 팬 아웃이 낮아지는 단점을 가진다. SETI[13]는 시공간에서 공간적 지역성과 시간의 연속된 증가를 고려하여 공간적으로 분할된 각 셀에 시간 색인을 두는 복합 색인 구조로서 셀 경계에 있는 객체를 중복 저장함으로서 색인의 크기가 증가하고, 다수의 시간 색인 관리 비용이 높다는 단점이 있다.

2.2 TB-tree

본 논문에서 시공간 영역 질의의 성능 향상을 위해 기초가 되는 색인으로 R-tree를 기반으로 궤적의 효율적인 검색을 위한 Trajectory Bundle-Tree(TB-tree)[1]가 있다. TB-tree는 하나의 궤적을 구성하는 시간 순서화된 부분 궤적을 하나의 단말 노드에 저장하는 방법을 사용한다. 또한 부분 궤적간의 연결을 위해 이전 궤적이 저장된 노드를 가리키는 선행 포인터와 이후 궤적이 저장된 노드를 가리키는 후행 포인터를 저장하여 이동체의 전체 궤적 검색을 효율적으로 처리하는 메커니즘을 가진다. 그림 1과 같이 TB-tree의 구조는 단말 노드에 하나의 궤적만을 저장한다. 이동체 A의 경우 순서적으로 보고되는 궤적 선분 A1, A2, A3는 동일한 단말노드에 저장된다. 또한 이동체 A의 궤적을 저장한 단말 노드인 R1, R2와 같이 동일 궤적의 선분들을 저장한 노드들 간에는 시간적 정렬되어 있으므로 동일 궤적간의 중첩이 없을 뿐만 아니라 이중 연결 구조가 존재하여 논리적으로 하나의 궤적을 표현하는 방법을 제공한다. 따라서 TB-tree에서 궤적 검색시 이중 연결 구조를 따라 부분 궤적을 추출함으로서 R-tree와 같이 Top-down방식으로 검색하는 방법과 비교할 때 불필요한 노드 접근을 줄이는 효과를 가진다. 이와 같은 방법 통해 TB-tree는 궤적 기반 질의 및 복합 질의의 성능을 월등히 향상시키는 기반을 마련하였다.

그러나, 삽입 알고리즘에서 삽입할 부분 궤적을 궤적 보존을 위해 가장 최근에 저장된 노드를 검색하여 검색된 노드에 삽입하기 때문에 R-tree와 같은 시공간적 지역성을 고려하지 않는 정책을 사용함으로써 사장 영역이 증가될 뿐만 아니라 노드간의 중첩이 매우 증가하게 되는 단점이 있다. 즉, R-tree의 가장 중요한 특성인 공간적 지역성을 배제하기 때문에 노드간의 중첩이 증가

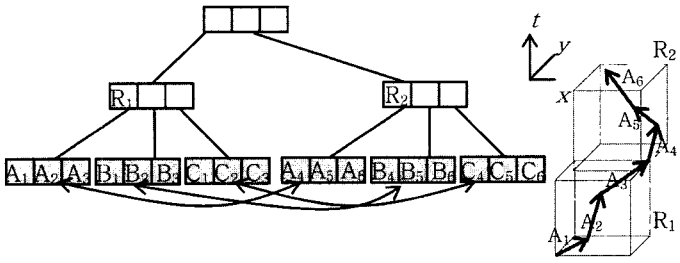


그림 1 TB-tree의 구조

될 뿐만 아니라 공간적 지역성을 감소시켜 영역 질의 비용이 증가하게 되는 단점이 있다[14].

3. TB-tree를 위한 개선된 분할 및 재배치정책

3.1 응용 시나리오

물류시스템이나 항해정보 검색은 정보 시스템에서 중요한 응용이다. 항해 정보 시스템을 고려해보자. 중앙 관제 센터에는 모든 선박의 항해 정보를 관리하는 데이타베이스 시스템이 있다. 각 선박은 GPS 장비를 장착하고 있다. 선박은 GPS장비에 의해 측정된 현재 위치를 획득하고 무선 통신을 이용하여 중앙 관제 센터로 전송한다. 데이타베이스 시스템은 모든 선박의 현재 위치를 저장하여 각 선박의 항해 경로를 관리하고 질의를 처리한다. 이와 같은 응용에서 일어날수 있는 질의는 다음과 같다.

- 오전 10시에 하와이항에 정박한 선박을 검색하라. (timeslice query)
- 오전 10시에서 오후 5시까지 하와이항에 정박했던 선박을 검색하라. (range query)
- 밀수 예상 선박 A의 일주일간의 이전 경로를 검색하라. (navigational query)
- 어제 하와이항을 출항한 선박의 현재까지의 경로를 찾아라. (combined query)

3.2 문제점

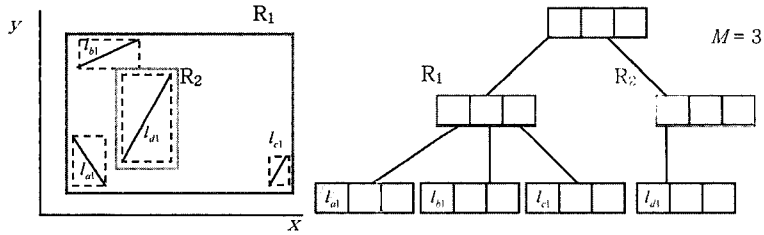
이와 같은 질의를 처리하기 위해 이 논문에서 다루고자 하는 문제점은 다음과 같다. 시공간에서 이동체의 궤적을 색인하기 위한 색인 구조에서 궤적 보존 특성만을 고려할 경우 공간적 지역성을 배제되어 타임슬라이스 및 영역 질의를 처리시 큰 사장 영역으로 인해 질의 성능이 떨어지는 것이다. d 차원의 시공간에서 n개의 이동체 궤적이 있다고 가정하자. i 번째 이동체의 궤적은 다음과 같은 표현한다.

$$Traj_i = \{l_{i1}, l_{i2}, ..., l_{ij}\}$$

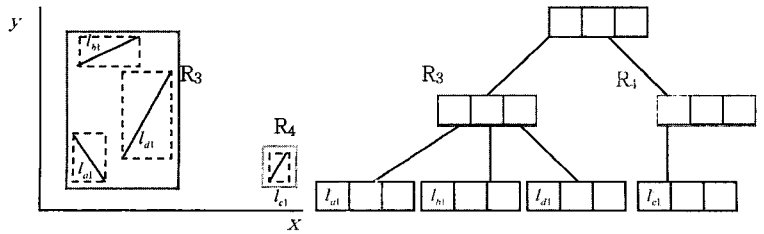
(l : d차원의 선분, i: 궤적 id, j: 부분 궤적(선분)id)

R-tree의 최소 영역 확장(Least Area Enlargement) 정책과 달리 궤적 보존을 속성을 가지는 TB-tree는 시공간적으로 인접한 두 이동체의 궤적을 서로 다른 단말 노드에 저장하고, 비단말 노드에서는 시간 순으로 저장된다. 즉, 각 타임스탬프마다 부분 궤적 정보가 이동체로부터 전송되어 색인의 이전 궤적이 저장된 단말 노드에 저장된다. 또한 단말 노드가 오버플로우 되는 경우 새로운 단말 노드가 생성되고 이를 위한 비단말노드의 엔트리가 생성된다. 생성된 비단말 노드 엔트리는 시간적 순서에 따라 생성될 뿐 공간적 지역성을 고려하지 못한다.

그림 2는 비단말 노드의 팬아웃(fanout)이 3인 TB-tree에 동일한 시간 간격을 가지는 이동체 궤적 a, b, c, d의 궤적 선분 l_{a1}, l_{b1}, l_{c1}, l_{d1} 이 순서대로 삽입되었을 때 공간 도메인상의 위치를 나타낸다. 그림 2(a)는 공간 근접성에 대한 고려 없이 삽입되는 순서대로 색인이 생성되어 비단말 노드 R_1은 l_{a1}, l_{b1}, l_{c1}을 포함한 단말 노드로 구성된다. l_{d1}을 포함한 단말노드가 다른 비단말 노드 R_2를 구성하게 되어, R_2는 R_1에 완전히 포함되어 겹



(a) 공간 근접성 고려 못한 경우



(b) 공간 근접성 고려한 경우

그림 2 삽입 순서에 의한 비단말 노드의 사장 영역과 노드간 중첩의 영향

침이 심하고 R_1의 경우 사장 영역이 매우 크다. 반면 그림 2(b)는 삽입 순서가 아닌 공간 지역성을 고려하여 생성된 경우이다. l_{c1}이 가장 공간적 근접성이 낮으므로 l_{a1}, l_{b1}, l_{d1}이 비단말 노드 R3을 구성하게 되고, l_{c1}이 비단말 노드 R_4를 구성하게 된다. 두 노드 R_3와 R_4의 겹침이 없고 R_3의 사장 영역도 그림 2(a)의 R_1에 비하여 작게 된다. 따라서 이동체의 궤적 선택과 추출을 위한 시공간 색인은 궤적을 보존해야 할 뿐만 아니라 그림 2(b)와 같이 공간 지역성을 고려해야 한다.

3.3. 궤적 색인의 노드 분할 정책

비단말 노드의 사장 영역을 감소시켜 궤적 선택을 위한 질의(예를 들어 시공간 영역질의 또는 타임슬라이스 질의)시 불필요한 노드 접근을 줄이기 위해 정리 1과 같이 사장 영역을 축소시키는 기본 아이디어를 기반으로 한다.

정리 1. 원소의 개수가 M인 집합 N의 Interval을 I(N)라 할 때, N의 최외곽점 하나를 제거할 경우 M-1개를 포함하는 집합 N′의 Interval I(N′)은 I(N)보다 크지 않다.

증명. M개의 원소를 가진 집합 $N=\{o_i \mid o_i \in R\}$과 N에서 가장 큰 값 max(N), 가장 작은 값 min(N)을 뺀 차집합을 N′, N″은 다음과 같이 정의된다.

$$N' = N - \{\max(N)\}$$

$$N'' = N - \{\min(N)\}$$

이 경우 집합 N, N′, N″이 이루는 간격 I(N), I(N′), I(N″)은 다음과 같이 정의된다.

$$\mathrm{I(N)} = \max(N) - \min(N)$$

$$\mathrm{I(N')} = \max(N') - \min(N')$$

$$\mathrm{I(N'')} = \max(N'') - \min(N'')$$

집합 N과 N에서 최대값을 뺀 N′과 최소값을 뺀 N″의 간격차 I(N)-I(N′)과 I(N)-I(N″)은 다음과 같다.

$$\begin{aligned}&\mathrm{I(N)} - \mathrm{I(N')} = \max(N) - \min(N) - (\max(N') - \min(N'))\\ &= \max(N) - \max(N') + \min(N') - \min(N)\\ &= \max(N) - \max(N') \qquad \because \min(N') = \min(N)\\ &\geq 0 \qquad \because \max(N) > \max(N')\end{aligned}$$

$$\begin{aligned}&\mathrm{I(N)} - \mathrm{I(N'')} = \max(N) - \min(N) - (\max(N'') - \min(N''))\\ &= \max(N) - \max(N'') + \min(N'') - \min(N)\\ &= \min(N'') - \min(N) \qquad \because \max(N'') = \max(N)\\ &\geq 0 \qquad \because \min(N'') > \min(N)\end{aligned}$$

정리 1과 같이 중복되지 않는 M개 이상의 원소를 가진 노드에서 최외각의 원소를 제거하면 노드의 MBR은 축소된다. 고차원에서도 노드의 엔트리들을 축에 투영하여 중복되지 않는 최외각의 엔트리를 제거할 경우에는 노드의 MBR은 축소된다.

R-tree와 같은 노드 분할 기법에서는 오버플로우된 노드의 엔트리를 각 노드가 [M/2]개 이상이 되도록 노드를 분할하는 반면 TB-tree에서는 최근 시간 분할 방법으로 가장 최근에 삽입된 엔트리를 새로운 노드에 삽입하는 비균등 분할 방식을 사용한다. 문제 정의에서 언급하였듯이 최근 시간 분할로 인해 비단말 노드의 사장 영역이 큰 문제점을 해결하기 위해 이 논문에서는 궤적 색인을 위한 삽입 방법으로 최대 영역 축소 방법을 적용하는 MAR 분할 정책을 제시한다.

3.3.1 최대 영역 축소(Maximal Area Reduction) 분할 정책

TB-tree는 Right Most Path에 있는 비단말 노드에서만 새로운 엔트리가 생성된다. 예를 들어 새로운 이동체의 궤적이 삽입될 경우 트리의 최우측 Path를 따라 새로운 단말 노드가 생성된다. 이 경우 생성된 단말 노드를 가리키는 엔트리를 포함하는 비단말 노드가 오버플로우인 경우 가장 최근에 생성된 엔트리를 새로운 비단말 노드에 저장한다. 이와 같이 TB-tree에서는 비단말 노드의 분할 방법으로 최근 시간 분할 정책을 사용한다. 그러나 비단말노드의 경우 시간 순으로 정렬되는 특성을 가지는 반면 비단말 노드의 공간적 특성을 고려하지 못한다.

MAR 분할 정책은 비단말 노드에의 공간적 특성을 고려한 분할 정책으로서 사장 영역을 축소하기 위해 오버플로우가 발생한 비단말 노드에서 MBR을 가장 축소할 수 있는 엔트리를 새로운 노드에 저장한다. 그러나, MAR 분할 정책에서 공간적 특성만 고려할 경우 시간 정렬구조를 파괴할 수 있기 때문에 MBR을 축소할 수 있는 엔트리 중 최근 시간에서 주어진 시간 간격 I(α) 이내의 엔트리를 선정한다. 주어진 시간 조건에서 MBR을 가장 축소할 수 있는 엔트리는 정리 1과 같이 노드 MBR의 경계에 위치한 엔트리로서 다음(그림 3)의 SelectVictim 알고리즘에 의해 선택된다.

최대 영역 축소를 기반으로 한 MAR Split알고리즘을 적용한 TB-tree의 수정된 삽입 알고리즘은 다음(그림 4)과 같다.

3.4 엔트리 재배치 정책

엔트리 재배치 정책은 분할 정책이 아닌 트리내의 엔트리들을 서로 교환하는 정책으로 비단말 노드의 엔트리의 위치를 재배치하여 사장 영역을 축소하기 위한 방법이다. 3.3.1에서 언급했듯이 TB-tree에서 비단말 노드에 삽입 되는 엔트리는 트리의 최우측 경로에 있는 비단말 노드에서만 발생하고 이를 제외한 비단말 노드는 모두 Full인 상태이다. 엔트리 재배치는 분할 방법에 따라 두 가지 방법으로 분류된다. 첫째, Mar 분할 정책에 따라 새로운 비단말 노드가 생성될 때, 생성된 비단말 노드의 엔트리와 다른 엔트리간의 1:N관계에서 엔트리

Algorithm MARSplit(N,E)
MAR1 [Select Victim] Let N' be N+{E}. Apply Algorithm **SelectVictim(N')** to choose an entry $E_V$to be the element of new node.
MAR2 [Assign entries to nodes] Add the victim E_V in the new node PP and add the others in the original node N is called P'.
Algorithm SelectVictim(N)
SV1 [Calculate Area of Entries excluding a entry E_i in given time condition]
d_i = area(MBR(N_i)) $N_i = N - \{E_i\}, E_i \in N, E_i.time \subset I(\alpha)$
SV2 [Choose the most wasteful entry] Choose any entry with the minimum d.

그림 3 MAR 분할 알고리즘

```
Algorithm Insert(N,E)
INS1 Invoke FindNode(N,E)
INS2 IF node N' is found,
        IF N' has space,
            insert new segment.
            AdjustTree( N' ); return;
INS3 Let N'' be a leaf node in the right most path
     create new leaf node NC
INS4  AdjustTree'( N'',NC);

Algorithm AdjustTree(N)
ADJ1 IF N is the root
     return;
ADJ2 Let P be the parent node of N
ADJ3 Let En be N's entry in P
        Set En's MBR so that it tightly encloses all entry MBRs in N
ADJ4 AdjustTree(P);

Algorithm AdjustTree'(N,NN)
ADJ'1 Let P be the parent node of N, and let NN be new created node.
ADJ'2 If N is the root and is full
        Create new root with entries pointing to N and NN; return;
ADJ'3 IF P has space
        insert Entry ENN pointing to NN in the P
        Let En be N's entry in P
            Set En's MBR so that it tightly encloses all entry MBRs in N
        AdjustTree(P);
     ELSE
        MARSplit(N, ENN) to produce P' and PP containing all N's old entries and ENN
        AdjustTree( P',PP)
```

그림 4 MAR정책을 기반으로 한 삽입 알고리즘

를 재배치시키는 ERNN(Entry Relocation in New Node)방법과 둘째, TB-tree의 최근 시간 분할 정책을 사용할 경우 비단말 노드가 풀이될 때, 풀이된 단말노드의 엔트리와 다른 엔트리를 M:N 관계에서 엔트리를 재배치시키는 ERFN(Entry Relocation in Full Node) 방법이 있다.

3.4.1 ERNN(Entry Relocation in New Node) 기법

ERNN 기법에서 엔트리 재배치는 MAR정책에 따라 비단말 노드가 분할 된 후, 새로운 비단말 노드가 엔트리를 1개만 가질 경우 적용된다. 즉, 비단말 노드에 포함된 1개의 엔트리가 재배치될 첫 번째 엔트리가 되고, 이를 E_{fr}라 한다. 엔트리 재배치의 목적은 E_{fr}을 동일 레벨의 비단말 노드에 재삽입하여 Overfull된 노드 N에서 엔트리 E_{sr}을 제거하여 MBR을 축소시키는 것이다. ERNN 에서 가장 중요한 것은 E_{fr}와 교환될 엔트리인 E_{sr}을 찾는 것이다.

E_{sr}은 다음의 요건에 만족해야 한다. 첫째, 엔트리 E_{sr}을 포함하는 노드 N은 엔트리 E_{fr}를 포함하여야 한다. 왜냐하면, E_{sr}을 포함하는 노드 N의 MBR을 증가시키지 않게 하기 위해서는 E_{fr}이 노드 N에 포함되어야 하기 때문이다. 둘째, E_{sr}은 Overflull된 노드 N에서 MBR을 축소할 수 엔트리이어야 한다. 셋째, 시간적 중첩을 고

Algorithm SelectSwitchingEntry(E_{fr})

SSN1 [Select Candidate] E_{fr}을 포함하는 후보 노드 CN 들을 검색한다.

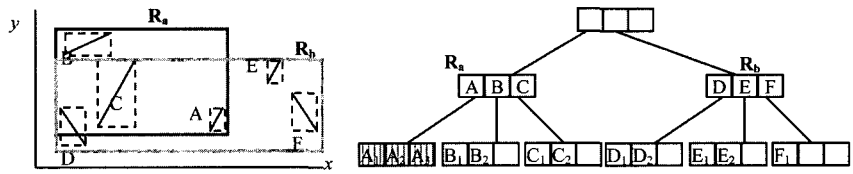
$Candidates = \{CN_i \mid Contain(CN_i, E_{fr}) = true\}$

SSN2 [Choose switching entry E_{sr} by the Most Area reduction in given condition] 후보 노드 집합 {CN}에서 허용 시간 간격 $I(\beta)$에 포함되는 한 엔트리 E_j를 제거할 경우 영역 축소가 가장 큰 엔트리 E_{sr}을 교환 대상 엔트리 SE로 선정한다.

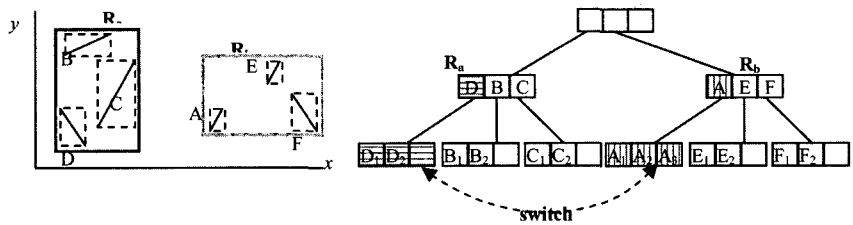
$CN_i^j = CN_i - \{E_j\} + \{v\},\ CN_i \in Candidates\ and\ v.time \subseteq I(\beta)$

$SE = \{E_j \mid \max(area(CN_i^j.mbr) - area(CN_i.mbr)\}$

그림 5 교환 대상 엔트리 선정 알고리즘(ERNN)



(a) F의 삽입으로 Rb노드가 Full인 상태



(b) A와 D 엔트리의 교환

그림 6 엔트리 교환을 통한 사장 영역의 축소

Algorithm SelectSwitchingNode(N,v)

SSN1 [Select Candidate] Full 상태가 된 노드 N에서 희생 엔트리 v를 포함하는 후보 노드 CN 들을 검색한다.

$Candidates = \{CN_i \mid Contain(CN_i, v), CN_i.level = N.level\}$

SSN2 [Choose switching node SN by the Most Area reduction] 노드 N과 후보 노드 집합 {CN}의 쌍에서 희생 엔트리와 후보 노드 CN_i의 한 엔트리 E_j를 교환할 경우 영역 축소가 가장 큰 후보 노드인 선택노드 SN을 교환 대상 노드로 선정한다.

$CPair = \{(N, CN_i) \mid CN_i \in Candidates\}$

$N' = N - \{v\} + \{E_j\}, CN_i^j = CN_i - \{E_j\} + \{v\}$

$CPair' = \{(N', \overline{CN_i^j}) \mid \min(area(N'.mbr) + area(CN_i^j.mbr)), 0 < j \le n(CN_i), CN_i \in Candidates\}$

$SN = \{CN_i \mid \max(area(NP.mbr) + area(CN_i.mbr) - area(NP'.mbr) - area(\overline{CN_i^j}.mbr),$

$(NP, CN_i) \in CPair, (NP', \overline{CN_i^j}) \in CPair'\}$

그림 7 교환 대상 노드 선정 알고리즘(ERFN)

려해야 한다. 재배치시 E_{fr}을 포함하는 노드는 색인에서 가장 최근 시간을 가진 엔트리이다. 그러나 E_{sr}은 재배치 시점 이전의 데이타로서 서로 교환이 발생할 경우 색인의 시간적 중첩을 크게 하는 단점을 가진다. 따라서 E_{sr}의 선정은 재배치시 최근 삽입 시간에서 허용되는 시간 간격($I(\beta)$)이내의 엔트리를 선정한다.

이를 위한 ERNN의 재배치 알고리즘은 그림 7과 같이 SelectSwitchingEntry()알고리즘을 이용한다.

3.4.2 ERFN (Entry Relocation in Full Node)

ERFN 방법은 최우측 단말 노드의 생성으로 인해 비단말 노드가 풀이 되는 경우 동일 레벨의 비단말 노드에서 엔트리를 서로 교환함으로써 사장 영역을 축소시

키는 방법으로 노드 분할은 TB-tree의 최근 시간 분할 방법을 사용한다. 그림 6(a)와 같이 이동체 A~F가 순서적으로 보고할 경우 궤적 데이타가 시간 순서에 따라 저장되기 때문에 R_a와 R_b를 구성하는 MBR은 큰 사장 영역을 가질 뿐만 아니라 MBR간의 중첩이 매우 높다. 엔트리 재배치 방법은 현재 FULL인 노드의 엔트리를 다른 노드의 엔트리와 교환하여 영역 감소를 발생시키는 기법이다. 그림 6(a)와 같이 이동체 F의 삽입시 R_b 노드는 Full인 상태가 될 경우 교환 대상을 검색한다. 교환 대상 노드는 영역을 최대한 축소할 수 있는 노드를 선택하는 것으로서 D노드가 선택되었다고 가정할 경우 A와 D노드의 위치를 교환하여 그림 6(b)와 같이 엔트리를 재분배하여 R_a와 R_b간의 사장 영역 및 중첩을 줄이게 한다.

ERFN 방법은 희생 엔트리 선택과 교환 대상 노드 선정 과정으로 구성된다. 희생 엔트리 선택은 MAR에서 정의한 SelectVictim()알고리즘을 이용하고 교환 대상 노드 선정은 그림 7과 같이 *SelectSwitchingNode*()알고리즘을 이용한다.

4. 성능 평가

이 절에서는 이 논문에서 제시한 MAR 분할 알고리즘과, 노드 재배치 알고리즘인 ERNN, ERFN의 3가지 알고리즘에 대하여 TB-tree와 성능을 비교 평가한다. 먼저 MAR 분할정책에서 희생 엔트리와 교환 대상 엔트리의 시간 조건에 따른 성능을 분석하고, 제시한 알고리즘의 3가지 조합에 대한 성능을 분석한다. 3가지 조합은 다음과 같다.

	분할 방법		재배치 방법	
	MRT	MAR	ERNN	ERFN
MAR		◎		
MARNN		◎	◎	
TBFN	◎			◎

각 알고리즘에 따른 검색 성능을 평가하는 실험에서는 타임 스탬프 질의, 영역 질의 및 복합질의에 대한 노드 접근 횟수를 비교한다.

4.1 실험 데이타 집합

일반적으로 이동체 데이타는 시공간의 궤적 정보를 가지는 데이타로서 이 논문에서는 다양한 분포의 시공간 데이타 집합을 생성할 수 있는 GSTD[15]를 이용하여 생성된 데이타 집합을 기반으로 한다. 생성된 데이타 집합의 경우 시공간상의 점으로서 이동체의 궤적 정보가 저장된다. 성능 평가에 적용하기 위해 이동체의 궤적은 시공간의 선분으로서 색인에 삽입되어야 하므로 생성된 데이타 집합에서 동일 궤적 ID를 가진 시간 순서화 된 두 점을 하나의 선분으로서 사용한다. GSTD의 경우 n개의 이동체에 대해 타임스탬프 1000을 가지고 분포에 따라 LR은 이동체들이 왼쪽에서 오른쪽으로 이동하는 분포, P의 경우에는 초기 분포가 가우시안 분포에서 북동쪽으로 퍼지는 분포, UA1의 경우 초기 가우시안 분포에서 느린 속도로 전체 영역으로 퍼지는 분포, UA2의 경우 UA1과 동일하지만 빠른 속도로 이동하는 분포를 가진다.

4.2 희생 엔트리와 교환 대상 엔트리의 시간 조건

연대기 순으로 입력되는 시공간 데이타를 궤적 색인에 삽입할 경우에는 공간적 근접성뿐만 아니라 시간적 정렬을 고려해야 한다. 예를 들어 ERFN에서 희생엔트리와 교환 대상 엔트리가 공간적으로는 인접하지만 시간 간격이 큰 경우 이를 교환하면 공간적 영역은 축소되지만 시간 영역의 확장이 발생한다. 따라서 시간 영역의 확장을 조정하기 위하여 희생엔트리와 교환대상 엔트리의 시간 간격에 대한 조건을 다음과 같이 추가한다.

희생 엔트리는 시간 조건 $I(\alpha)$ 내에 존재하여야 한다.

교환 대상 엔트리는 시간 조건 $I(\beta)$ 내에 존재하여야 한다.

$I(\alpha)$는 비단말 노드의 가장 작은 시간 간격을 가지는 엔트리의 시간 간격($I(E_s)$과 노드 MBR의 최근 시간(N.MBR.To)을 이용하여 다음과 같이 정의한다.

$$N.MBR.To \geq I(\alpha) \geq N.MBR.To - \omega * I(E_s)$$

$$N.MBR.To \geq I(\beta) \geq N.MBR.To - \sigma * I(E_s)$$

그림 8과 같이 비단말 노드의 페이지 크기가 1024인 경우 팬아웃이 16이므로 ω와 σ는 1, 2, 5, 10, 16의 값을 주고 실험한 결과이다. 각 데이타집합에 대한 타임스탬프, 영역 질의를 각각 1000회씩 수행하여 평균 디스크 접근 회수를 측정하였다. 그래프는 ω=1, σ=1인자의 결과와 각 인자별 실험 결과와의 백분율을 나타낸다. 실험결과 2/16인 경우 전체적 성능이 가장 좋은 것으로 나타

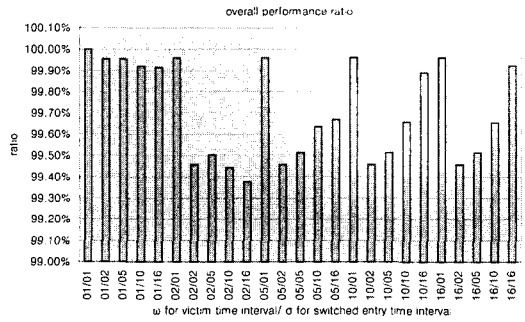


그림 8 시간 간격에 따른 희생 엔트리와 교환 대상 엔트리 선정

났다. 이것은 희생 엔트리와 교환대상 엔트리가 가장 최근 보고된 시간 영역에서 재배치되므로 시간적 중첩을 피할 수 있기 때문이다. 따라서 이후의 실험은 이 인자를 이용하여 다른 색인들과 실험한다.

4.3 영역 질의

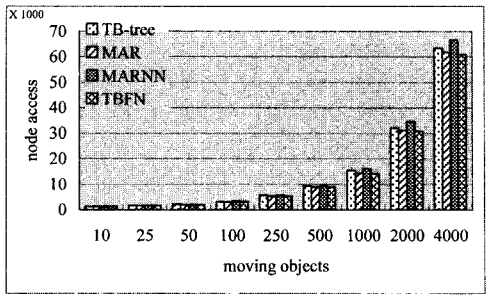
영역 질의는 공간 데이타 뿐만 아니라 시공간 데이타에 대해서도 중요한 질의이다. 이 절에서는 영역 질의를 처리하기 위한 4가지 방법을 비교하기 위해 10개부터 4000개의 이동체가 1000번 보고한 데이타셋(0.6MB~256MB)을 사용한다. 그리고 영역 질의를 위해 3가지 질의셋을 사용한다. 질의셋은 각 축에 대하여 1%, 5%, 25% 범위로서 전체 영역에 대하여 0.0001%, 0.125%, 1.5625%를 나타낸다. 각 질의셋은 1000개의 영역 질의를 포함한다.

그림 9는 다양한 영역질의와 데이타셋에 대한 전체 노드 접근 횟수를 나타낸다. X축은 이동체의 수로 표현한 것으로 2를 공비로 한 등비수열의 형태이고, Y축은 노드 접근 회수의 단위가 1000이다. 영역 질의에서 다음과 같은 성향이 관찰된다. 질의 영역의 크기가 작을수록 각 알고리즘의 차이가 커지고, 질의 영역이 커질수록 성능이 유사한 형태를 가진다. 특히 객체수가 작을 때보다는 객체수가 많은 경우 각 알고리즘의 차이는 뚜렷이 나타난다. 영역 질의시 가장 좋은 성능을 나타내는 것은 TBFN으로 TB-tree보다 최대 10%정도의 노드 접근 횟수가 적다. 또한 TBFN에서는 TB-tree보다 노드 접근 횟수가 많지 않다. 반면 MARNN 알고리즘에서는 TB-tree보다 많은 노드 접근 횟수를 가진다. 이것은 노드 재배치시 재삽입되는 엔트리에 의해 줄어드는 영역보다 재삽입되는 영역에서 희생되는 엔트리에 의한 시간적 정렬을 파괴함으로써 발생하는 시간적 중첩으로 인한 것이다.

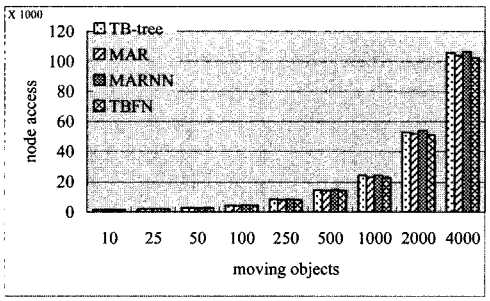
4.4 타임슬라이스 질의

타임슬라이스 질의의 유형은 임의의 시간 t에 대해 공간 영역이 1%, 10%, 50%, 100%인 경우를 비교한다. 4가지 질의 집합은 각각 1000개의 질의를 포함한다. 그림 10에서 보여주는 결과는 TBFN에서 가장 좋은 결과를 보이고, MAR, TB-tree, MARNN순서로 타임 슬라이스 질의 성능을 나타낸다. 성능타임 슬라이스 질의의 공간 영역이 증가할수록 질의 성능의 차이가 작게 되는 성향을 나타낸다.

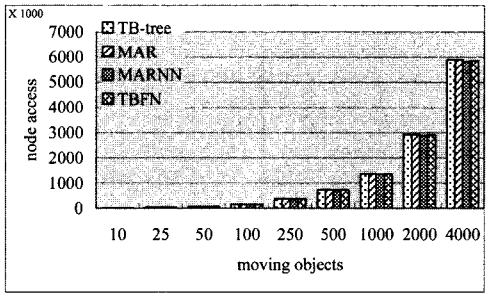
타임슬라이스 질의는 본질적으로 주어진 시각에 이동체의 위치를 검색한다. 따라서 타임슬라이스 질의는 시간 순서만으로 정렬된 TB-tree보다 공간적 지역성을 고려한 분할 방법이 더 효율적이다. 실험 결과에서 이와 같은 현상은 큰 공간 영역을 가진 질의보다 작은 공간 영역 질의에서 분명히 나타난다.



(a)



(b)



(c)

그림 9 Range Queries: varying range, (a) 1%, (b) 5%, and (c) 25% in each dimension

4.5 복합 질의

제시한 알고리즘을 적용한 색인 방법들은 복합질의에서 TB-tree보다 검색 성능이 저하되어서는 안된다. 성능 평가를 위해서 다양한 수의 이동체들에 대한 데이타셋을 사용하고, 복합 질의로서 각 차원(전체 영역)에 대해서 내부 영역의 크기가 1%(0.0001%)이고 외부 영역의 크기가 5%(0.125%), 25%(1.5625%)인 질의가 1000개로 구성된 질의셋을 이용한다.

그림 11의 결과는 TB-tree와 제시한 알고리즘에서 거의 동등한 성능을 나타내고 있다. 복합 질의는 영역질의와 항해질의로 구성되어 있다. 색인 구조는 동일 궤적간의 이중 연결 구조가 있기 때문에 항해 질의에서의

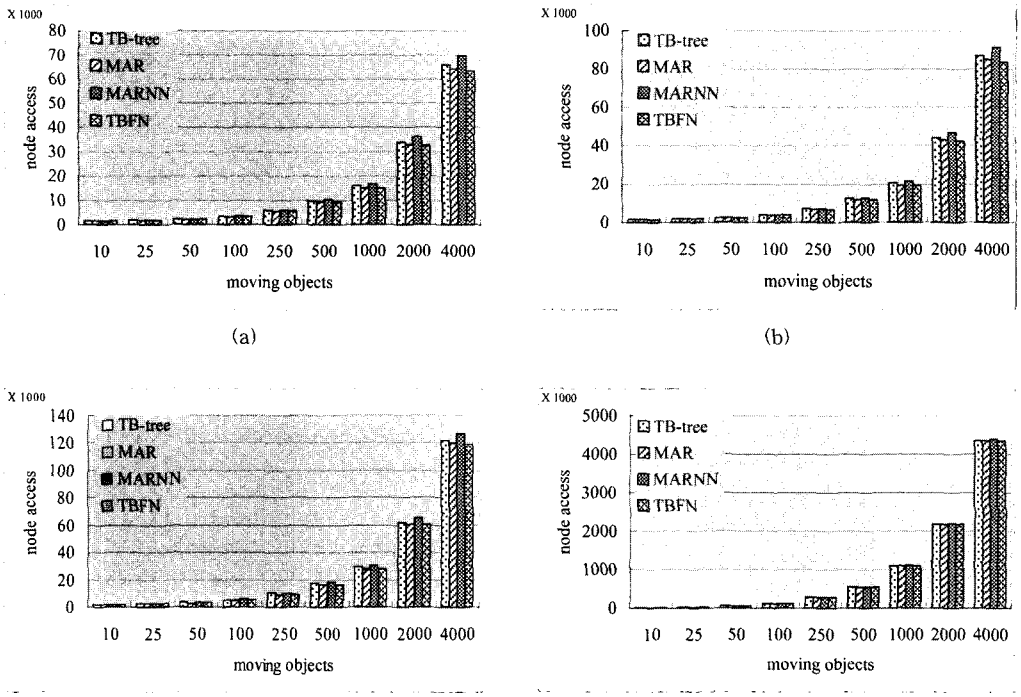


그림 10 Time slice Queries: varying spatial range, (a) 1%, (b) 10%, (c)50%, and (d) 100% in each dimension

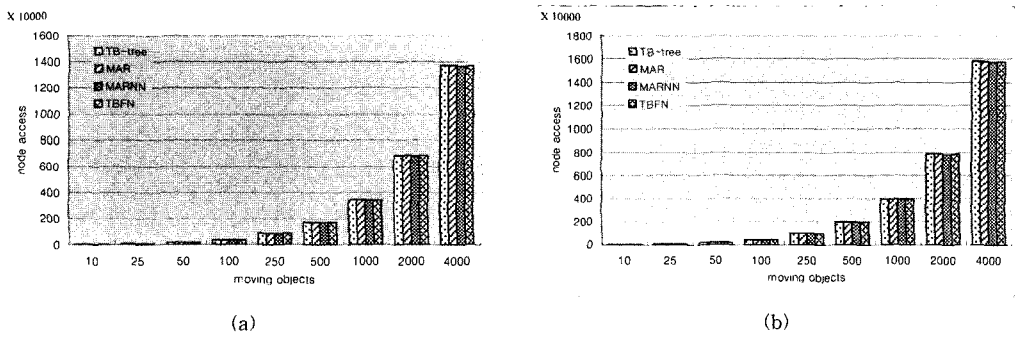


그림 11 Combined Queries: (a) 1% inner-5% outer range and (b) 1% inner-25% outer range, in each dimension

성능을 동일하다. 그러나 영역 질의의 차이로 인해 약간의 성능의 변화가 있다. 그림 11(b)에서 4000개의 이동체를 가진 경우 TBFN 알고리즘은 TB-tree보다 0.4% 정도의 복합 질의 성능 개선이 있다. 그러나 이 논문에서는 복합 질의의 성능을 개선하는 것이 아니라 복합 질의의 성능 저하 없이 영역 질의 성능을 개선하는 것을 목적으로 하였다.

5. 결론 및 향후 연구

궤적 보존을 위해 공간적 지역성을 완전히 배제하는 색인 구조는 비단말 노드에서 큰 사장 영역을 가지기 때문에 노드간의 중첩이 높이는 원인을 제공하여 영역 질의의 성능을 저하시키는 문제를 해결하기 위해 이 논문에서는 궤적 검색을 위한 색인 구조에서 항해질의 성능을 유지하면서 영역 질의의 성능을 향상시키기 위한 비단말 노드 분할 방법과 엔트리 재배치 방법을 제시하였다. 분할 정책은 비단말 노드의 분할시 비단말 노드의 MBR을 최대한 감소시키는 엔트리와 나머지 엔트리로

분할하는 최대 영역 축소(Maximal Area Reduction) 정책을 사용하고, 엔트리 재배치 방법은 비단말 노드의 MBR을 구성하는 다수의 엔트리에서 MBR을 최대로 감소시킬 수 있는 엔트리의 위치를 재배치시키는 방법으로서 최근 시간 분할 정책을 사용하는 ERNN(Entry Relocation in New Node)알고리즘과 최대 영역 축소 정책을 사용하는 ERFN(Entry Relocation in Full Node)알고리즘을 제시하였다.

실험 결과 TB-tree의 최근 시간 분할 정책 정책과 ERFN 알고리즘을 적용한 TBFN 방법이 타임 슬라이스 질의와 영역 질의에서 최대 10% 정도의 성능 향상을 가져왔다. 또한 복합 질의에서는 기존 TB-tree의 항해 질의 성능을 유지하면서 영역 질의의 부분적 성능 향상으로 0.4% 정도 노드 접근 회수를 줄였다.

이 논문에서는 분할 및 재배치 알고리즘으로 기존 궤적 색인의 성능을 개선하였다. 그러나 색인 구조에서 좋은 검색 성능을 가지기 위해서는 높은 노드 팬아웃을 가지고, 높은 공간 활용도와 최소한의 사장 영역을 가져 노드간의 중첩을 줄이는 것이다. 따라서 팬아웃 증가를 위한 데이타 모델링 방법과 궤적을 포함하는 노드 사이의 중첩을 줄이기 위한 삽입 알고리즘의 연구가 필요하다.

참 고 문 헌

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

[2] Antonin Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," In Proc. of SIGMOD, pp.47-57, 1984.

[3] M. Hadjieleftheriou, G. Kollios, V.J. Tsotras, and D. Gunopulos, "Efficient Indexing of Spatiotemporal Objects," In Proc. of EDBT, pp.251-268, 2002.

[4] Yufei Tao, Dimitris Papadias, Jun Zhang, "Cost models for overlapping and multiversion structures," ACM Trans. Database Syst., Vol.27, No.3, pp.299-342, 2002.

[5] David B. Lomet and Betty Salzberg, "Access Methods for Multiversion Data," In Proc. of SIGMOD, pp.315-324, 1989.

[6] Bruno Becker, Stephan Gschwind, Thomas Ohler, Bernhard Seeger, and Peter Widmayer, "On Optimal Multiversion Access Structures," In Proc. of SSD, pp.123-141, 1993.

[7] Peter J. Varman and Rakesh M. Verma, "An Efficient Multiversion Access Structure," TKDE, Vol.9, No.3, pp.391-409, 1997.

[8] Yufei Tao and Dimitris Papadias, "MV3R-Tree: A Spatio-Temporal Access Method for Timestamp and Interval Queries," In Proc. of VLDB, pp.431-440, 2001.

[9] Anil Kumar, Vassilis J. Tsotras, and Christos Faloutsos, "Designing Access Methods for Bitemporal Databases," TKDE, Vol.10, No.1, pp.1-20, 1998.

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

[11] Theoderidis, Y.; Vazirgiannis, M.; Sellis, T., "Spatio-temporal indexing for large multimedia applications," In Proc. of IEEE Multimedia Computing and Systems, pp.441-448, 1996.

[12] Hongjun Zhu, Jianwen Su, and Oscar H. Ibarra, "Trajectory queries and octagons in moving object databases," In Proc. of CIKM, pp.413-421, 2002.

[13] V. Prasad Chakka, Adam Everspaugh, and Jignesh M. Patel, "Indexing Large Trajectory Data Sets with SETI," In Proc. of CIDR, 2003.

[14] Dieter Pfoser, "Indexing the Trajectories of Moving Objects," IEEE Data Eng. Bull. Vol.25, No.2, pp.3-9, 2002.

[15] Theodoridis, Y., Silva, R., and Nascimento, M, "On the Generation of Spatiotemporal Datasets," In Proc. of the 6th Int'l Symposium on Spatial Databases, pp.147-164, 1999.

임 덕 성
정보과학회논문지 : 데이타베이스
제 31 권 제 2 호 참조

조 대 수
정보과학회논문지 : 데이타베이스
제 31 권 제 2 호 참조

홍 봉 희
정보과학회논문지 : 데이타베이스
제 31 권 제 2 호 참조