위치 기반 질의 처리를 위한 궤적 보존 색인의 설계 및 구현

Design and Implementation of Trajectory Preservation Indices for Location Based Query Processing

임덕성* / Duk-Sung Lim, 홍봉희** / Bong-Hee Hong

요 약

위치 기반 서비스(Location-Based Service)는 무선 통신에 기반 한 서비스로서 최근 그 중요성이 증대되고 있다. 차량, 선박과 같이 시간에 따라 위치를 변경하는 이동 객체(moving object)의 이동 경로는 궤적(trajectory)으로 표현된다. 이동 객체의 궤적 모니터링을 위한 데이터베이스에서는 이동 객체의 위치를 추적할 뿐만 아니라 이동 경로를 감시하기 위한 궤적 질의를 효율적으로 지원해야 하므로 이동 객체의 궤적 정보를 효과적으로 관리하고, 빠른 검색을 제공하는 이동 객체 색인 방법이 필요하다. 이 논문에서는 먼저 기존 궤적 색인 구조에서 사장 영역 문제를 정의한다. 궤적 색인의 사장 영역은 궤적 보존 속성으로 인해 공간적 지역성을 고려되지 않기 때문에 발생한다. 이를 해결하기 위해 이 논문에서는 사장 영역 및 비단말 노드간의 중첩을 줄이기 위해 엔트리 재배치 기법을 제시하고, 제안된 색인과 기존 알고리즘을 사용하는 색인과의 성능비교를 통하여 제시한 색인의 우수성을 입증한다.

Abstract

With the rapid development of wireless communication and mobile equipment, many applications for location-based services have been emerging. Moving objects such as vehicles and ships change their positions over time. Moving objects have their moving path, called the trajectory, because they move continuously. To monitor the trajectory of moving objects in a large scale database system, an efficient indexing scheme to processed queries related to trajectories is required.

In this paper, we focus on the issues of minimizing the dead space of index structures. The Minimum Bounding Boxes (MBBs) of non-leaf nodes in trajectory-preserving indexing schemes have large amounts of dead space since trajectory preservation is achieved at the sacrifice of the spatial locality of trajectories. In this thesis, we propose entry relocating techniques to reduce dead space and overlaps in non-leaf nodes. we present performance studies that compare the proposed index schemes with the TB-tree and the R*-tree under a varying set of spatio-temporal queries.

† 이 논문은 2008년 정부(교육과학기술부)로부터 지원받아 수행된 연구임(지역거점연구단육성사업/차세대물류IT기술연구사업단)

■ 접수일 : 2008.06.03 ■ 수정일 : 2008.07.23 ■ 심사완료일 : 2008.07.23

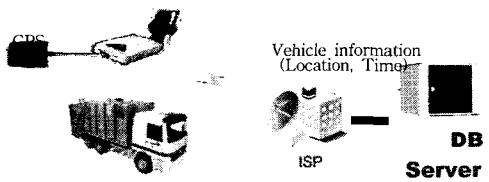
* 영진전문대학 컴퓨터정보계열 교수(junsung)@yjc.ac.kr

** 교신저자 부산대학교 공과대학 컴퓨터공학과 교수(bhhong)@pusan.ac.kr

주요어 : 위치 기반 서비스, 이동체 데이터베이스, 궤적 색인
Keyword : *location-based services, moving object databases, trajectory index*

1. 서론

위치 기반 서비스(Location-Based Service)는 무선 통신에 기반한 서비스로서 최근 그 중요성이 증대되고 있다. 예를 들어 산업폐기물 운반 차량이 폐기물을 지정된 처리장소에서 처리하지 않고 불법 매립을 위해 정해진 경로를 벗어나는 것을 감시하거나, 지정된 수거 영역을 벗어나 추가적인 폐기물을 수집하는 것을 감시하기 위해 차량의 위치를 모니터링 할 필요가 있다. 이때 폐기물 운반차량과 같이 시간에 따라 위치를 변경하는 객체를 이동 객체(moving object)라고 한다[6].



<그림 1> 위치 모니터링을 위한 환경

<그림1>과 같이 이동 객체의 위치를 모니터링 하기 위해서는 GPS와 같은 위치 측정을 위한 장비와 위치 정보를 저장할 수 있는 데이터베이스, 그리고 위치를 전송할 TRS와 같은 통신 시스템이 필요하게 된다. 이동 객체는 속도나 방향이 바뀔 때마다 3차원 시공간 상의 점으로 위치를 보고한다. 그리고 보고된 두 점을 선분으로 연결하여 이동 경로를 나타내며, 이러한 경로들을 모아 다중선(polyline)인 궤적(trajectory)으로 표현한다[7].

궤적 검색을 위한 질의는 크게 위상 질의(topological query), 항해 질의(navigational query), 복합 질의(combined query)로 분류된다[11]. 복합 질의는 1)궤적의 선택과 2)궤적의 부분을 추출하는 질의로서 "오전 7:00시에서 8:00시 사이에 도축장 A에서 오염된 육류를 적재한 차량의 다음 1시간내의 이동 경로를 구하라"와 같이 "오전 7:00시에서 8:00시 사이에 도축장 A에서 오염된 육류를 적재한 차량"에서 궤적을 선택하는 부분과 "다음 1시간내의 이동경로"에서 궤적의 일부를 추출하는 부분으로 구성된다.

궤적에 대한 복합 질의는 영역내의 궤적 선택과 궤적의 일부분을 추출하는 과정을 포함하기 때문에 인접한 데이터를 같은 노드에 저장하는 지역성 기반 색인 구조(locality-based indexing scheme)는 궤적의 일부를 추출하기 위한 비용이 높은 단점을 가진다. 반면, 동일 궤적을 같은 노드에 저장하는 궤적 보존 색인 구조(trajectory-preserving indexing scheme)의 경우 노드 간의 중첩이 매우 높아 영역내의 궤적 검색 비용이 높은 단점이 있다. 따라서 시공간 도메인의 동적 확장을 지원하고 대용량 궤적 정보에 대한 복합 질의를 효율적으로 처리하기 위한 이동 객체 궤적 색인이 필요하다.

이 논문에서는 궤적 질의 성능은 우수하지만 공간적 지역성을 고려하지 않은 궤적 보존 색인 구조를 기반으로 영역 질의 및 복합 질의에 성능이 우수한 색인구조를 제시한다. 먼저 사장 영역 및 비단말 노드간의 중첩을 줄여 영역 질의를 효율적으로 처리하기 위한 방법으로 비단말 노드 분할시 최대 영역 감소 정책을 지원하는 MAR-tree와 비단말 노드간의 엔트리를 서로 교환 함으로써 사장 영역을 줄이는 엔트리 재배치 정책을 사용하는 MAR+-tree를 제시한다.

이 논문의 구성은 다음과 같다. 2장에서는 궤적 색인을 위한 관련 연구를 기술하고, 3장에서는 기존 궤적 색인 구조의 문제점을 기술한다. 4장에서는 비단말 노드 분할시 비단말 노드의 MBB를 최소화 할 수 있는 최대 영역 감소 정책을 사용하는 MAR-tree의 색인 구조를 제시한다. 5장에서는 엔트리의 재배치를 통해 비단말 노드의 사장 영역을 감소시키는 MAR+-tree를 제시한다. 6장에서는 궤적 색인의 성능을 비교하고, 마지막으로 7장에서는 결론과 향후 과제를 기술한다.

2. 관련연구

현재까지 이동 객체의 궤적 검색을 위한 색인의 근간은 Guttman의 R-trees[1]로서 이를 기반으로 이동 객체와 같은 시공간 객체의 시간차원을 공간의 또 다른 차원으로 표현하는 다양한 종류의 색인들이 연구되어 왔다.

객체의 최소 영역 사각형(MBB)을 저장하여 검색하는 다차원 색인 구조인 R-tree[1]와 R*-tree[2]에서는 n차원 공간을 점유하는 이동 객체의 시간 간격에 따른 위치를 n+1차원 MBB로서 표현한다. MBB는 이동 객체의 시간 간격을 시간축에 대한 높이로서 표현하고 시간 간격 동안의 공간적 위치는 n차원 공간 영역을 표현한다. R-tree에 시간적 차원을 추가한 시공간 인덱스로서 3DR-tree[13]가 제시되었다. 3DR-tree는 궤적을 구성하는 각 부분 궤적을 선분으로 표현하여, 3차원 R-tree를 사용한다. 3DR-tree는 특정 시간과 공간 영역이 주어지는 영역 질의에 높은 성능을 보이지만, 이동 객체의 궤적 저장시 색인의 사장 공간이 클 뿐만 아니라 노드간의 중첩이 매우 큰 단점을 가진다.

SETI[4]와 ACAR-tree[16]는 시공간에서 공간적 지역성과 시간의 연속된 증가를 고려하여 공간적으로 분할된 각 셀에 시간 색인을 두는 복합 색인 구조로서 셀 경계에 있는 객체를 중복 저장함으로써 색인의 크기가 증가하고, 다수의 시간 색인 관리 비용이 높다는 단점이 있다.

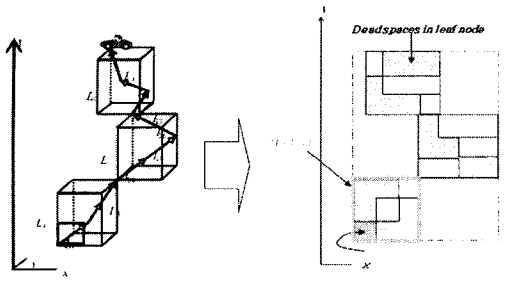
STR-tree[11]는 이동 객체들의 궤적을 선분의 집합으로 나타내고, 보존 파라미터(preservation parameter)를 사용하여 같은 객체의 궤적을 근접한 페이지에 저장하도록 유도 하였다. STR-tree의 부분적인 궤적 보존 정책은 3DR-tree에 비해 궤적 질의 성능 향상 효과가 낮을 뿐만 아니라, 삽입 정책에서 공간 근접성과 같은 공간 구별성이 낮아지기 때문에 영역 질의의 성능이 좋지 못한 단점이 있다.

R-tree를 기반으로 궤적의 효율적인 검색을 위해 Trajectory Bundle-Tree(TB-tree)[11]가 제시되었다. 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 이동 객체의 전체 궤적 검색을 효율적으로 처리하는 메커니즘을 가진다. 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.

3. 궤적 색인 구조의 사장 영역 문제

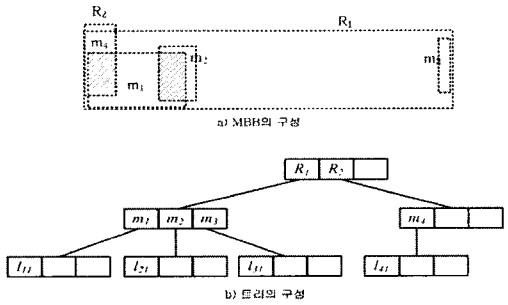
R-tree의 최소 영역 확장(Least Area Enlargement) 정책과 달리 궤적 보존 속성을 가지는 TB-tree는 시공간적으로 인접한 궤적일지라도 서로 다른 궤적을 서로 다른 단말 노드에 저장하고, 비단말 노드에서는 단말 노드의 MBB를 시간 순으로 저장한다. 또한 단말 노드가 오버플로우 되는 경우 새로운 단말 노드가 생성되고 단말 노드의 MBB를 저장하기 위한 비단말 노드의 엔트리가 생성된다. 생성된 비단말 노드 엔트리는 시간적 순서에 따라 생성될 뿐 공간적 지역성을 고려하지 못한다.

<그림2>와 같이 노드를 구성할 경우 부분 궤적 l_{01}, l_{12}, l_{23}는 그룹핑되어 하나의 단말 노드 L_1을 구성하게 된다. 단말 노드의 구성시 단말 노드의 MBB는 단말 노드를 구성하는 엔트리를 모두 포함하는 최소 영역 사각형을 구성하게 된다. <그림2>와 같이 하나의 단말 노드를 구성할 경우 엔트리에 의해 점유되는 공간을 제외한 사장 영역(dead space)이 매우 큰 것을 볼 수 있다. 이와 같이 사장 영역이 크게 되는 주요 원인은 궤적 보존 색인이 단말 노드에 동일한 궤적의 부분 궤적만을 저장함으로써 시간에 종속적인 저장 구조를 가지기 때문이다.



<그림 2> 단말 노드의 사장 영역(dead space)

<그림3>과 같이 노드 용량이 3인 색인에서 이동 객체 O_1, O_2, O_3, O_4의 궤적 데이터가 저장됨에 따라 시간 순서에 의해 O_1, O_2, O_3의 궤적 데이터를 포함한 단말 노드가 하나의 비단말 노드에 저장되고, O_4의 궤적 데이터가 삽입될 때, 기존 비단말 노드가 Full인 상태이므로 분할에 의해 새로운 데이터인 l_{41}의 궤적데이터는 새로 생성된 비단말 노드에 저장된다. 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 못하여 큰 사장 영역을 가지는 단점을 가진다.



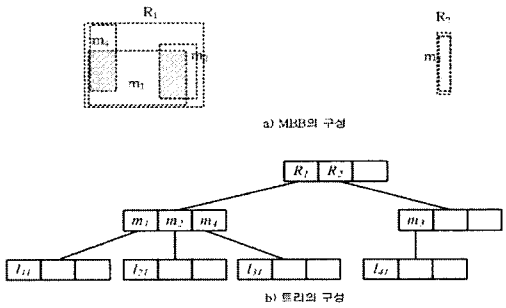
<그림 3> 시간 분할 정책(TB-tree)

4. MAR-tree

4.1 최대 영역 축소(Maximal Area Reduction) 분할 정책

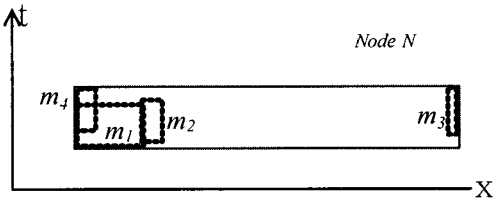
비단말 노드의 분할시 시간 순서가 아닌 공간 근접성을 고려하여 분할 할 경우 사장 영역이 축소되고 비단말 노드간 중첩도 제거할 수 있다. 따라서 MAR 분할 정책에서는 분할 되는 영역을 최소화할 수 있는 방법을 제시한다.

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.



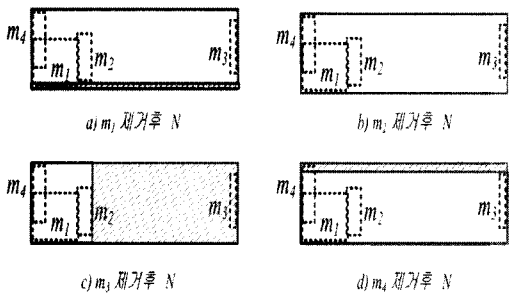
<그림 4> 공간근접성을 고려한 분할 방법

예를 들어 <그림4>와 같이 m_4의 삽입으로 오버플로우 되어 분할되어야 하는 경우, <그림5>와 같이 각 엔트리를 하나씩 제거한 다음의 노드 영역을 측정한다.



<그림 5> 오버플로우된 비단말 노드

이 경우 <그림6>의 (c)와 같이 m_3가 제거 되었을 때 N의 영역이 가장 최소가 되므로 m_3가 분할 대상으로 선정되어 새로운 노드에 저장된다.



<그림 6> 엔트리 제거후 사장 영역의 감소

희생 엔트리 선정을 위해 SelectVictim을 호출한다. SelectVictim은 주어진 시간 조건 I_α에 속한 엔트리 E_i 중에서 E_i를 제외한 나머지 엔트리를 포함한 MBB를 계산한다(SV1). MBB의 크기가 가장 작을 경우 희생 엔트리로서 E_i를 선정한다(SV2). 두번째 단계는 희생 엔트리 E_i를 선정한 후, E_i는 새로운 노드에 저장하고, 나머지 노드는 이전 노드에 저장한다(MAR2).

Algorithm MARSplit(N)
MAR1 [Select Victim] Apply Algorithm SelectVictim(N) to choose an entry EV to be the element of the new node.
MAR2 [Assign entries to nodes] Add the victim EV to the new node N' and add the others in the original node N, called p' .

Algorithm SelectVictim(N)
SV1 [Calculate the Area of Entries excluding an entry Ei in the given time condition] di=area(MBB(Ni)), Ni = N - {Ei}, N ⊃ Ei, Ei.time ⊂ I_α.
SV2 [Choose the most wasteful entry] Choose any entry with the minimum d.

<그림 7> MAR 분할 알고리즘

4.2 MAR 분할 알고리즘

MAR 분할 정책은 사장 영역을 줄이기 위한 목적을 가진다. 이를 위해 <그림7>의 분할 알고리즘은 크게 두 단계로 나누어 진다. 첫번째 단계(MAR1)는 오버플로우된 노드에서 사장영역을 가장 축소할 수 있는 희생 엔트리를 선택하는 것이다.

4.3 삽입 알고리즘

최대 영역 축소를 기반으로 한 MAR 분할 알고리즘을 적용한 MAR-tree의 삽입 알고리즘은 <그림8>과 같다.

Algorithm Insert(N,E)
INS1 Invoke FindNode(N,E) to select a leaf node N' which a predecessor of E.
INS2 IF node N' is not found, Create the new leaf node N'
INS3 Insert new segment E in N' .
INS4 AdjustTree(N')

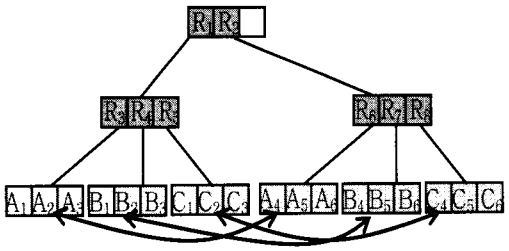
Algorithm AdjustTree(N)
ADJ1 IF N is the root, return;
ADJ2 If N is overflow, Invoke MARSplit(N)
ADJ3 Let P be the parent node of N, Let En be entries in N
Adjust MBB(En) so that it tightly encloses all entry MBBs in N
ADJ4 Set N=P and repeat from ADJ1

<그림 8> MAR정책을 기반으로 한 삽입 알고리즘

5. MAR+-tree

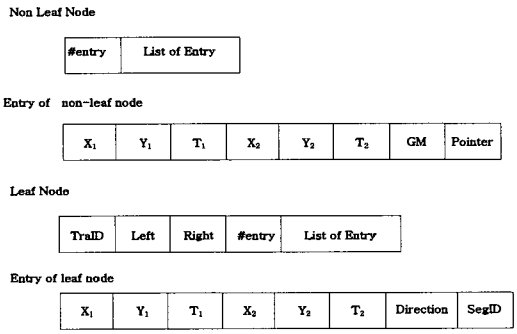
5.1 색인 구조

MAR+-tree는 MAR-tree의 단점을 보완하기 위해 재배치 정책을 적용한 색인 구조이다. 색인 구조는 <그림9>와 같이 단말노드에 동일 궤적 데이터만 저장하는 궤적 클러스트링을 지원하고, 동일 궤적을 저장하는 단말 노드간의 연결정보를 가지는 TB-tree, MAR-tree와 유사하다.



<그림 9> MAR+-tree 색인 구조

MAR+-tree의 비단말 노드의 구조는 <그림10>과 같이 저장된 엔트리의 개수인 #entry정보와 하위 노드의 MBB를 저장하기 위한 비단말노드 엔트리의 집합으로 구성된다. 3차원 시공간에서 MBB를 저장하기 위하여 비단말 노드의 엔트리는 공간 영역 저장을 위해 X_1, X_2, Y_1, Y_2 정보를 저장하고 시간의 간격 저장을 위해 T_1, T_2 정보를 저장한다. 또한 비단말 노드의 속성 정보를 위해 GM이라는 속성을 사용한다. 그리고, MBB와 연관된 하위 노드에 대한 포인터 정보를 포함한다.

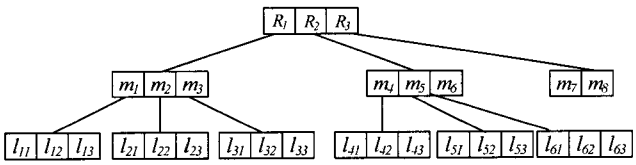


<그림 10> 단말 노드와 비단말 노드의 구조

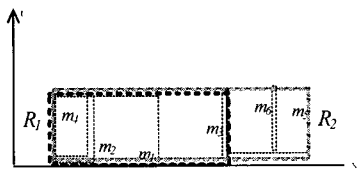
5.2 엔트리 재배치 정책

엔트리 재배치 정책은 분할 정책이 아닌 노드내의 엔트리들을 서로 교환하는 정책으로 비단말 노드의 엔트리의 위치를 재배치하여 사장 영역을 축소하기 위한 방법이다. TB-tree에서 비단말 노드에 삽입 되는 엔트리는 트리의 최우측 경로에 있는 비단말 노드에서만 발생하고 이를 제외한 비단말 노드는 모두 Full인 상태이다.

<그림11>의 (a)는 이동 객체 O_1에서 O_6의 위치 정보가 각각 3회씩 보고 되었을 때 단말 노드의 엔

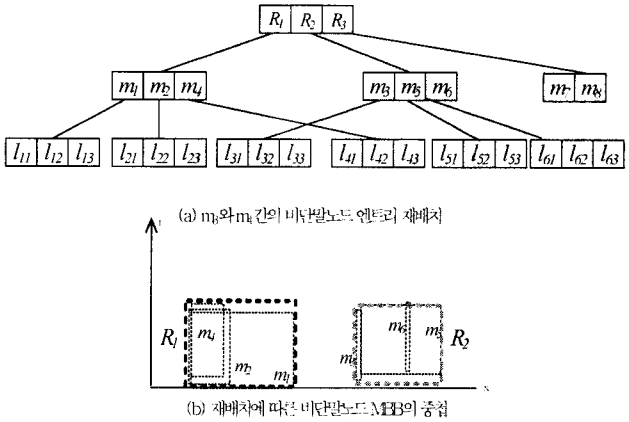


(a) 이동체 O1~O6의 3회 보고시 색인 구조



(b) 색인 구조에 따른 비단말노드 MBB의 중첩

<그림 11> 궤적 저장에 따른 색인 구성 및 MBB



<그림 12> 재배치에 따른 색인 구조 및 MBB 구성

트리가 Full인 상태를 보여준다. 각 단말 노드는 동일한 객체의 궤적을 저장하기 때문에 이동 객체 O_1의 부분 궤적 l_{11}, l_{12}, l_{13}가 하나의 단말 노드에 저장되어 있다. 3개의 부분 궤적은 하나의 단말노드를 구성하고 단말 노드의 MBB는 상위의 비단말 노드에 R_1의 엔트리로서 저장된다. 비단말 노드는 R_1은 m_1, m_2, m_3의 3개의 단말노드의 MBB를 저장하고 있다. R_1과 R_2는 비단말 노드의 MBB로서 <그림11>의 (b)에서 볼 수 있다. 이 경우 시간 순서에 의해 생성되는 노드들에 의해 MBB가 결정됨으로써 비단말 노드의 MBB가 유사한 시간대에 속하는 특징을 가지는 반면 xy평면상에서는 지역성이 결여되는 단점을 가진다. 또한 <그림11>의 R_1과 R_2의 MBB와 같이 두 MBB간의 겹침이 심해서 질의 처리시 비용이 증대되는 단점을 가진다

<그림12>와 같이 비단말 노드인 m_1와 m_4간의 재배치를 통해 비단말 노드의 MBB의 크기를 감소시킬 뿐만 아니라 비단말 노드간의 중첩도 줄일 수 있게 된다.

5.3 엔트리 재배치 알고리즘

중첩된 비단말 노드간의 사장영역을 줄이기 위한 가장 직관적인 방법은 엔트리를 연속적으로 재배치시켜 더 이상 사장영역이 줄지 않을 때까지 반복하는 것이다. 엔트리 재배치 알고리즘은 다음 3단계의 과정을 수행한다. 첫번째 이벤트 노드와 중첩되는 후보 노드를 찾는다. 다음으로 각 후보 노드에서 하나의 엔트리와 이벤트 노드에서 하나의 엔트리를 교환하는 모든 그룹 중 가장 작은 사장 영역을 가지는 그룹을 선택한다. 세 번째 단계로서 두 엔트리를 실제로 교환한다. 반복을 위해 두번째 단계로 이동한다.

Algorithm RelocatingEntry(N)

RE1 [Find Candidate Nodes] Find candidate nodes {Ci} that overlap with an event node N. Stop if no candidate node is found.

RE2 [Select Target Node] Select a target node T containing an entry Ei that could maximally reduce dead spaces by exchanging with an entry Ej in node N. (Ei∈T, Ej∈N, T ∈{Ci}). Stop if no target node is found.

RE3 [Exchange Entries] Exchange entry Ei with entry Ej in nodes T and N. Repeat from RE2

<그림 13> 엔트리 재배치 알고리즘(Exhaustive entry relocation)

6. 성능 평가

이 장에서는 <표1>과 같이 기존 시공간 색인으로 잘 알려진 R*-tree와 대표적인 궤적 색인인 TB-tree, 이 논문에서 제시한 최대 영역 축소 분할 정책을 적용한 MAR-tree, 엔트리 재배치 방법을 적용한 MAR+-tree등 총 4가지 색인 구조를 비교 평가한다. TB-tree는 분할 방법에서 시간 순서 분할 기법을 사용한다. R*-tree는 분할의 방법으로 오버플로우된 노드를 분할 및 재삽입 기법에 의해 처리한다. 이때 분할 비율은 50%로 설정하고 재삽입될 엔트리의 비율을 30%로 설정한다. MAR-tree는 분할 시 희생엔트리로 선정되는 엔트리의 시간 간격을 노드의 최대 시간에서 40%이내의 시간간격으로 설정한다. MAR+-tree는 엔트리 재배치를 통해 더 이상의 MBR의 축소가 일어나지 않는 Exhaustive entry relocation 방법을 이용한다.

<표 1> 비교 대상 색인 구조

Index	Node capacity(page: 1kB)		Parameters
	non-leaf node	leaf node	
TB tree	36	36	Time order split
R* tree	36	36	Reinsert factor: 0.3 Split distribution factor: 0.5
MAR tree	35	34	Time Interval Iα: 0.4
MAR+ tree	35	34	Exhaustive entry relocation

각 알고리즘에 따른 검색 성능을 평가하는 실험에서는 영역 질의 및 복합 질의에 대한 노드 접근 횟수를 비교한다.

6.1 실험 데이터 집합

일반적으로 이동 객체 데이터는 시공간의 궤적 정보를 가지는 데이터로서 이 논문에서는 다양한 분포의 시공간 데이터 집합을 생성할 수 있는 GSTD[14]를 이용하여 생성된 데이터 집합을 기반으로 한다. 생성된 데이터 집합의 경우 시공간상의 점으로서 이동 객체의 궤적 정보가 저장된다. 성능 평가에 적용하기 위해 이동 객체의 궤적은 시공간의 선분으로서 색인에 삽입되어야 하므로 생성된 데이터 집합에서 동일 궤적 ID를 가진 시간 순서화 된 두 점을 하나의 선분으로서 사용한다. GSTD의 경우 n개의 이동 객체에 대해 1000개의 타임스탬프를 생성시키는데, 초기 분포가 가우시안 분포에서 랜덤한 방향으로 퍼지는 데이터 집합을 사용한다. 실험에 사용하는 데이터집합은 <표 2>와 같이 이동 객체의 개수는 10 ~ 2000개를 사용한다. 이와 같은 이동 객체의 이동으로 생성되는 데이터 집합의 엔트리 수는 10K에서 2000K개이다.

<표 2> 성능 평가를 위한 데이터 집합

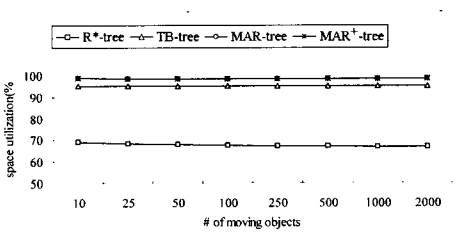
Type	Initial distri.	# object	# snapshot	# segments
DS1	Gaussian	10	1,000	10,000
DS2	Gaussian	25	1,000	25,000
DS3	Gaussian	100	1,000	100,000
DS4	Gaussian	250	1,000	250,000
DS5	Gaussian	500	1,000	500,000
DS6	Gaussian	1,000	1,000	1,000,000
DS7	Gaussian	2,000	1,000	2,000,000

질의를 위해서는 <표3>과 같이 영역, 타임슬라이스, 복합질의를 위해 축당 비율로서 다양한 질의 영역을 가진 질의 집합을 사용한다. 각 질의 집합의 질의의 개수는 1000개이다.

<표 3> 성능평가를 위한 질의 집합

Query type	range	# query
Range	1%, 5%, 10%, 20%	1,000
Timeslice	10%, 50%	1,000
Combined	(5%,10%),(5%,20%)	1,000

6.2 공간 활용도



<그림 14> 공간 활용도 비교

<그림14>에서와 같이 공간 활용도 비교를 위한 실험에서 R*-tree는 다양한 데이터 집합에서 70~67% 정도의 공간 활용도를 나타내는 반면, TB-tree는 96%정도의 공간 활용도를 나타내었고, MAR-tree와 MAR+-tree의 경우 99%정도의 공간활용도를 나타냄을 볼 수 있다.

이와 같이 공간 활용도의 가장 중요한 요인은 분할 방법이다. R*-tree의 오버플로우가 발생한 노드의 엔트리 M+1개를 2개로 나누어서 최소 노드에 M/2개 이상의 엔트리가 존재하지만, 궤적 보존 색인 구조에서는 새로운 궤적이 저장되고 있는 노드를 제외하면 100%의 공간 활용도로 데이터를 저장하는 노드 분할 방법을 사용하기 때문이다.

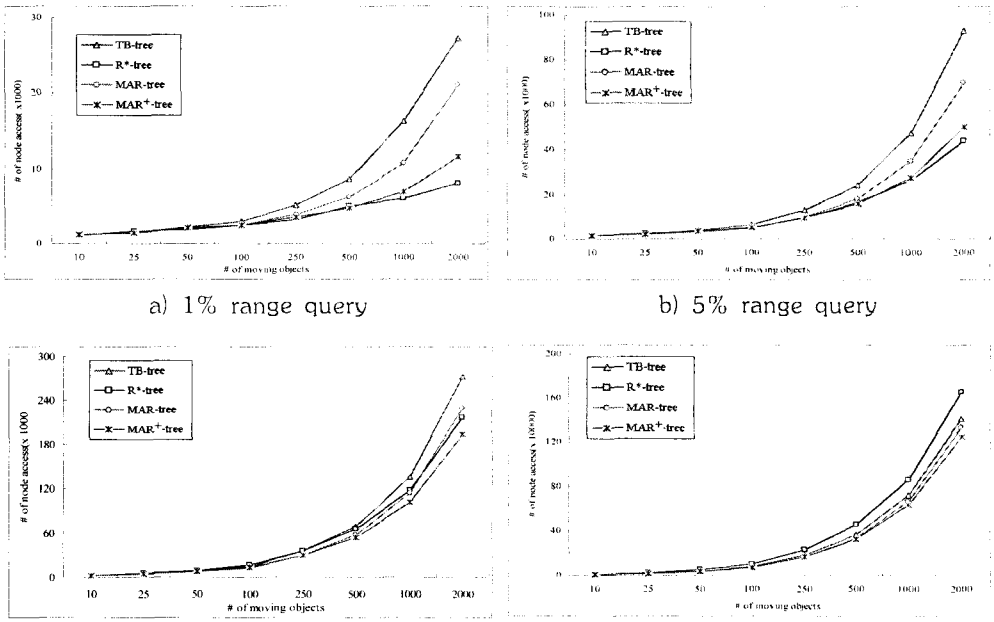
6.3 영역 질의

영역 질의를 위해 4가지 질의 집합을 사용한다. 질의 집합은 각 축에 대하여 1%, 5%, 10%, 20% 범위로서 전체 영역에 대하여 0.0001%, 0.0125%, 0.1%, 0.8%를 나타낸다. 각 질의 집합은 1000개의 영역 질의를 포함한다.

<그림15>는 영역질의시 전체 노드 접근 횟수를 나타낸다. X축은 이동 객체의 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. 작은 영역 질의시 가장 좋은 성능을 나타내는 것은 R*-tree인 반면 질의 영역이 커질수록 낮은 공간 활용도를 가지는 R*-tree의 질의 영역내의 궤적을 검색하는 비용이 높아짐을 알 수 있다.

궤적 보존 색인인 TB-tree, MAR-tree, MAR+-tree의 영역 질의 성능은 유사한 패턴을 보인다. 큰 영역 질의에서 MAR+-tree가 다른 색인 구조보다 질의 성능이 우수하다. 이를 정리하면 다음과 같은 결론을 얻을 수 있다.

- MAR+-tree는 영역 질의 처리를 위해서 가장



a) 1% range query　　b) 5% range query

c) 10% range query　　d) 20% range query

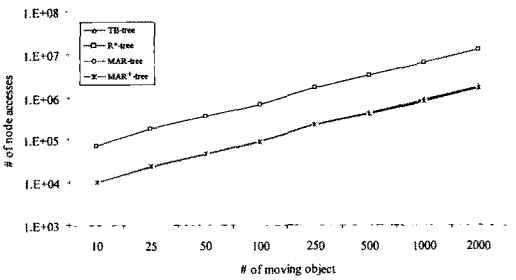
<그림 15> 영역 질의 성능 비교

효율적이다.

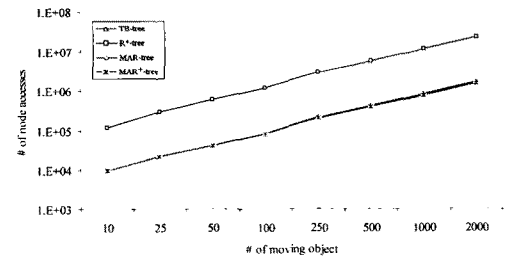
- MAR+-tree는 영역 질의 처리를 하는데 있어 TB-tree보다 우수하다.
- 사장 영역이 작아짐에 따라 영역 질의를 처리하기 위한 비용은 낮아진다.
- 효율적인 영역 질의 처리를 위한 주요 요인중 하나는 사장 영역의 크기이다.

6.4 복합 질의

제시한 알고리즘을 적용한 색인 방법들은 복합질의에서 TB-tree보다 검색 성능이 저하되어서는 안된다. 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 질의 집합을 이용한다.



(a) 5% inner-10% outer range



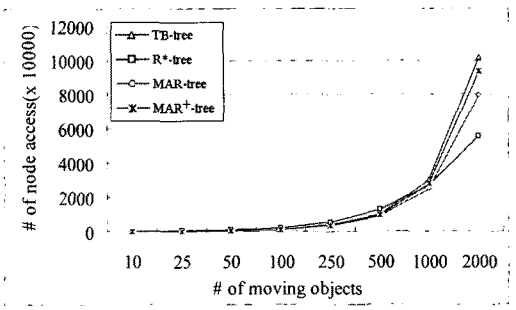
(b) 5% inner-20% outer range

<그림 16> 복합 질의 성능 비교

<그림16>의 결과는 궤적 보존 색인 구조가 R*-tree의 복합 질의 성능에 비해 월등히 우수함을 보여준다. 성능 차이는 이동 객체의 수가 증가함에 따라 더 커짐을 볼 수 있다. 이와 같이 R*-tree와 궤적 보존 색인이 차이를 보이는 것은 단말 노드에서 동일 궤적 간의 연결정보가 존재하기 때문이다.

6.5 삽입 비용

이절에서는 4가지 색인 구조를 구축할 때 소요되는 노드 접근 회수와 평균 삽입 및 검색비용을 비교한다. <그림17>에서는 TB-tree, R*-tree, MAR-tree, MAR+-tree를 구축하는 동안 소요되는 비용을 보여준다.



<그림 17> 색인 구축 비용

7. 결론 및 향후 연구

시간에 따라 위치를 변경하는 이동 객체의 궤적을 모니터링 하기 위한 대용량 데이터베이스에서 궤적을 모니터링하고 추적하기 위한 질의를 효율적으로 처리하기 위해서는 영역 질의뿐만 아니라 항해질의를 포함하는 복합 질의를 효율적으로 처리할 수 있는 색인 구조가 필요하다.

항해질의 처리를 효율적으로 처리하기 위해 공간적 지역성을 완전히 배제하고 궤적 보존 특성을 가지는 색인 구조는 비단말 노드에서 큰 사장 영역을 가지기 때문에 노드간의 중첩을 높이는 원인을 제공하여 영역 질의의 성능을 저하시키는 문제를 발생시킨다. 이를 해결하기 위해 이 논문에서는 궤적 검색을 위한 색인 구조에서 항해질의 성능을 유지하면서 영역 질의의 성능을 향상시키기 위해 엔트

리 재배치 기법을 제시하였다.

MAR-tree는 비단말 노드의 분할시 비단말 노드의 MBB를 최대한 감소시키는 엔트리와 나머지 엔트리로 분할하는 최대 영역 축소(Maximal Area Reduction) 정책을 정의하였고, MAR+-tree는 MAR-tree에서 발생한 공간적 사장 영역 확장의 문제를 해결하기 위해 비단말 노드의 MBB를 구성하는 다수의 엔트리에서 사장 영역을 최대로 감소시킬 수 있도록 엔트리의 위치를 재배치시키는 방법을 제시하였다.

사장 영역을 축소 시킴으로써 MAR+-tree는 TB-tree에 비해 시공간 영역 질의 및 타임슬라이스 질의에서 최대 40% 정도의 성능 향상을 가져왔다. 또한 복합 질의에서는 기존 TB-tree의 항해 질의 성능을 유지하면서 영역 질의의 부분적 성능 향상으로 0.4%의 노드 접근 회수를 줄였다. 또한 영역 질의에 우수한 R*-tree보다 MAR+-tree는 큰 질의 영역에서 10%의 노드 접근 회수를 줄여 영역 질의 성능을 향상시켰다.

참고문헌

1. Abdelguerfi M., Givaudan J., Shaw K., and Ladner R., "The 2-3TR-tree, A trajectory-oriented index structure for fully evolving valid-time spatio-temporal datasets," Proc. of the ACM Int' l symposium on advances in geographic information systems, 2002, pp.29-34.
2. Beckmann N. and Kriegel H., "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles," In Proc. ACM SIGMOD, 1990, pp.332-331.
3. Burton F., Kollias J., Matsakis D., and Kollias V., "Implementation of Overlapping B-Trees for Time and Space Efficient Representation of Collections of Similar Files," Comput. J. 33(3) , 1990, pp.279-280.
4. Chakka V., Everspaugh A., and Patel J., "Indexing Large Trajectory Data Sets with SETI," CIDR, 2003.
5. Hadjieleftheriou M., Kollios G., Tsotras V., and Gunopulos D., "Efficient indexing of spatiotemporal objects," Int' l Conf. on Extending Database Technology, 2002, pp. 251-268.
6. Kollios G., Gunopulos D., and Tsotras V., "On indexing mobile objects," Proc. of the ACM Symposium on Principles of Database Systems, 1999, pp. 261-272.
7. Nascimento M., Silva J., and Theodoridis Y., "Evaluation of access structures for discretely moving points," Proc. of Int' l Workshop on Spatio-Temporal Database Management, 1999, pp. 171-188.
8. Nascimento M. and Silva J., "Towards historical R-trees," Proceedings of the ACM symposium on Applied Computing, 1998.
9. Ni J., Ravishankar C., "PA-Tree: A Parametric Indexing Scheme for Spatio-temporal Trajectories," SSTD, 2005, pp. 254-272.
10. Patel J., Chen Y., and Chakka V., 10. "STRIPES: An Efficient Index for Predicted Trajectories," In SIGMOD Conference, 2004, pp. 637-646.
11. Pfoser D., "Indexing the trajectories of moving objects," IEEE Data Engineering Bulletin, vol. 25, no. 2, 2002, pp. 3-9.
12. Tao Y., Faloutsos C., Papadias D., 12. and Liu B., "Prediction and Indexing of Moving Objects with Unknown Motion Patterns," In SIGMOD Conference, 2004, pp. 611-622.
13. Theodoridis Y., Vazirgiannis M., and Sellis T., "Spatio-temporal indexing for large multimedia applications," Multimedia Computing and Systems, 1996, pp.441-

448.

14. Theodoridis Y., Silva R., and Nascimento M., "On the Generation of Spatiotemporal Datasets," In Proc. of the 6th Int'l Symposium on Spatial Databases, 1999, pp.147-164.
15. 김경숙, 이기준, "도로 위에 존재하는 이동객체의 궤적에 대한 네트워크 기반의 색인 방법," 한국정보처리학회논문지, vol. 13-D, no. 7, 2006, pp. 879-888.
16. 이응재, 이양구, 류근호, "이동객체의 현재 위치정보 관리를 위한 셀 기반 색인 기법," 한국정보처리학회 논문지, vol. 11-D, no. 6, 2004, pp.1221-1230.
17. 조형주, 최용진, 민준기, 정진완, "시공간 데이타베이스에서 영역 합 질의를 위한 색인 기법," 한국정보과학회 논문지, vol. 32, no. 2, 2005, pp.129-141.

임덕성(Duksung Lim)
1998년 : 동아대학교 컴퓨터공학과 졸업 (학사).
2000년 : 부산대학교 컴퓨터공학과 졸업 (공학석사).
2008년 : 부산대학교 컴퓨터공학과 졸업 (공학박사).
2003년~현재 : 영진전문대학 컴퓨터정보계열 교수.
관심분야 : 데이터베이스, 이동체 데이터베이스, RFID 시스템.
e-mail : junsung@yjc.ac.kr

홍봉희 (Bonghee Hong)
1982년 : 서울대학교 전자계산기공학과 졸업 (학사).
1984년 : 서울대학교 전자계산기공학과 졸업 (공학석사).
1988년 : 서울대학교 전자계산기공학과 졸업 (공학박사).
1987년~현재 : 부산대학교 공과대학 컴퓨터공학과 교수.
관심분야 : 이동체 데이터베이스, 공간 데이터베이스, RFID 시스템, RFID 데이터베이스
e-mail : bhhong@pusan.ac.kr