# 이동체를 위한 R-트리 기반 색인에서의 궤적 클러스터링 정책

반 재 훈[†] · 김 진 곤[††] · 전 봉 기[†††] · 홍 봉 희[††††]


## 요 약

이동체 데이터베이스를 위한 과거 궤적 색인으로 R-tree계열이 많이 사용되었다. 그러나 R-tree계열의 색인은 공간 근접성만을 고려하였기 때문에 동일 궤적을 검색을 할 때 많은 노드 접근이 필요하다. 즉 기존의 이동체 색인들은 공간 근접성과 궤적 연결성이 서로 상반된 특징을 가지므로 함께 고려하지 못했다. 이동체 색인에서 영역 질의의 성능개선을 위해서는 노드 간의 심한 중복과 사장 공간(Dead space)을 줄여야 하고, 궤적 질의의 성능 개선을 위해서는 이동체의 궤적 보존이 이루어져야 한다. 이와 같은 요구 조건을 만족하기 위해, 이 논문에서는 R-tree 기반의 색인 구조에서 궤적 클러스터링 정책을 제안한다. 노드 분할 정책에서는 궤적 클러스터링을 위해서 동일 궤적을 그룹화해서 분할하는 공간 축 분할 정책과 공간 활용도를 높이는 시간 축 분할 정책을 제안한다. 또한 비단말 노드의 연결 정보를 저장하여 개선된 복합 질의 알고리즘을 제안하였다. 이 논문에서는 제안한 R-tree기반 색인 구조의 구현 및 성능 평가를 통해서 검색성능이 우수함을 보였다.


**키워드 : 이동체 데이터베이스, 시공간색인, 궤적질의, 분할정책**

# Policies of Trajectory Clustering in Index based on R-trees for Moving Objects

ChaeHoon Ban[†] · JinGon Kim[††] · BongGi Jun[†††] · BongHee Hong[††††]


## ABSTRACT

The R-trees are usually used for an index of trajectories in moving-objects databases. However, they need to access a number of nodes to trace same trajectories because of considering only a spatial proximity. Overlaps and dead spaces should be minimized to enhance the performance of range queries in moving-objects indexes. Trajectories of moving-objects should be preserved to enhance the performance of the trajectory queries. In this paper, we propose the TP3DR-tree(Trajectory Preserved 3DR-tree) using clusters of trajectories for range and trajectory queries. The TP3DR-tree uses two split policies: one is a spatial splitting that splits the same trajectory by clustering and the other is a time splitting that increases space utilization. In addition, we use connecting information in non-leaf nodes to enhance the performance of combined-queries. Our experiments show that the new index outperforms the others in processing queries on various datasets.


**Key Words : Moving Object Database, Spatiotemporal Database, Trajectory Query, Split Policy**

## 1. 서 론

이동체는 시간에 따라 연속적으로 위치 정보가 변경되는 시공간 객체이다. 이러한 이동체의 위치를 검색하는 것은 위치 기반 서비스에서 중요한 응용 중의 하나이며, 현재 국내에서도 이동 통신사를 중심으로 친구 찾기, 물류 관리 시스템, 교통 정보 서비스, 최단 거리 서비스등과 같은 다양한 LBS 응용들이 개발되고 있다. 이와 같은 LBS 응용의 개발을 위해서는 시간의 흐름에 따라 증가하는 이동체의 위치 정보를 효과적으로 관리하고, 빠른 검색을 제공하는 이동체 색인의 개발이 필요하다.

이동체 색인에 있어서 질의 종류로는 크게 영역 질의, 타임스탬프(timestamp) 질의, 궤적 질의, 복합 질의로 나누어진다. 영역 질의는 주어진 시간 간격 동안에 공간 윈도우(spatial window)에 속하는 이동체들을 검색하는 질의이다. 타임스탬프 질의는 특정 시간에 주어진 공간 윈도우에 속하는 이동체들을 검색하는 질의이다. 궤적 질의는 특정한 이동체의 궤적을 검색하는 질의이다. 복합 질의는 "특정 시간에 주어진 영역을 지나간 객체의 궤적을 검색하라"와 같이 시공간 도메인의 영역 질의와 궤적 질의가 복합된 질의이다.

† 정 회 원 : 경남정보대학 인터넷응용계열 조교수
†† 정 회 원 : 삼성전자 소프트웨어랩 무선 4그룹
††† 정 회 원 : 신라대학교 컴퓨터정보공학부 전임강사
†††† 정 회 원 : 부산대학교 공과대학 컴퓨터공학과 교수
논문접수 : 2005년 3월 15일, 심사완료 : 2005년 6월 23일

이동체의 궤적을 3D R-tree[1]에 저장하면 보편적으로 TB-tree[2]보다 영역 질의에서는 우수한 성능을 보인다. 그러나, 노드 간의 중복이 많아 검색 성능이 저하되고, 궤적의 삽입 순서가 시간 축으로 증가하는 특징으로 인하여 공간 활용도가 낮다는 문제점이 있다. 3D R-tree는 궤적의 보존(Trajectory preservation)에 관한 개념을 고려하지 않기 때문에, 궤적 질의를 처리하기 위해서 점 질의(point query)를 이용하여 반복 검색을 하므로 질의 처리 비용이 크다는 문제점이 있다.

이 논문에서는 R-tree의 색인 구조에서 한 노드 내에 동일 궤적을 최대한 많이 저장하기 위한 동일 궤적 그룹화에 따른 단말 노드 클러스터링 정책을 제안한다. 단말 노드의 클러스터링 정책을 사용하더라도 궤적을 검색하기 위해서는 루트 노드에서 탐색을 하는 점 질의를 수행해야 한다. 이러한 점 질의는 많은 노드 접근 횟수를 유발시키는 원인이 된다. 따라서 점 질의 수행을 최소한 줄일 수 있는 다른 방법으로 비단말 노드에 추가 정보를 저장하여 이 정보를 이용해서 동일 궤적을 탐색하는 방법을 제안한다.

성능 평가를 위한 실험 데이터는 대표적인 위치 데이터 생성 도구인 GSTD 데이터셋 생성기[3]를 사용하여 생성하였다. 구체적인 실험 평가 항목으로는 색인의 공간 활용도, 영역 질의 성능 비교, 복합 질의 성능 비교, 심한 중첩의 임계값에 따른 색인 성능 비교 등이다.

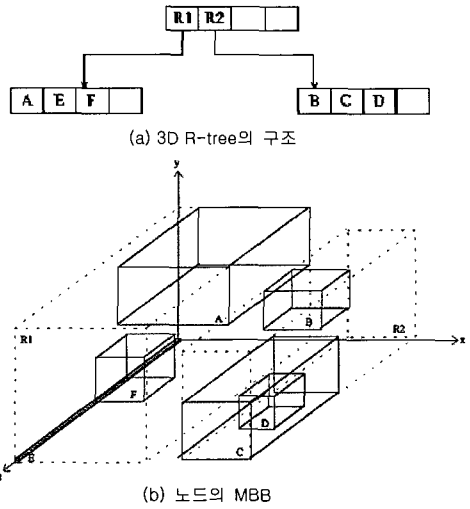
실험 평가를 위해 이동체 궤적의 영역 질의 성능이 우수한 TA3D R-tree[4]와 TB-tree[2]를 C++로 구현하여 실험하였다. 제안하는 TP3D R-tree(Trajectory Preserved 3DR-tree)는 동일 궤적 그룹화에 따른 분할 정책으로 인하여 단말 노드 내에 동일 이동체의 궤적이 많이 저장되기 때문에 TA3D R-tree 보다 복합 질의의 성능이 뛰어 나다. 또한 단말 노드의 중첩이 심하지 않기 때문에 TB-tree보다 영역 질의에서 성능이 좋다.

이 논문의 구성은 다음과 같다. 2장에서 관련 연구를 기술하고, 3장에서는 3D R-tree의 문제점을 분석한다. 4장에서 동일 궤적 그룹화에 따른 단말 노드 분할 정책을 제안하고, 5장에서는 비단말 노드의 추가 정보를 이용한 복합 질의 알고리즘을 제안한다. 6장에서는 실험을 수행하여 TP3D R-tree의 성능을 비교한다. 마지막으로 7장에서 결론 및 향후 연구를 기술한다.

## 2. 관련 연구

이동체의 궤적 색인에 관련된 연구는 크게 3가지 부분으로 나누어 볼 수가 있다. 첫째, 현재와 미래 위치를 검색하기 위한 색인에 관련된 연구들로서 R-tree, 해쉉, Quad-tree를 기반으로 하는 색인이 있다. 둘째, 시공간 색인에 대한 연구로서 3D R-tree, HR-tree가 있다. 셋째, 이동체의 궤적과 같은 과거 이력 색인에 대한 연구로서 STR-tree, TB-tree가 있다.

3D R-tree[1]는 공간 색인인 R-tree에서 한 축에 시간 도



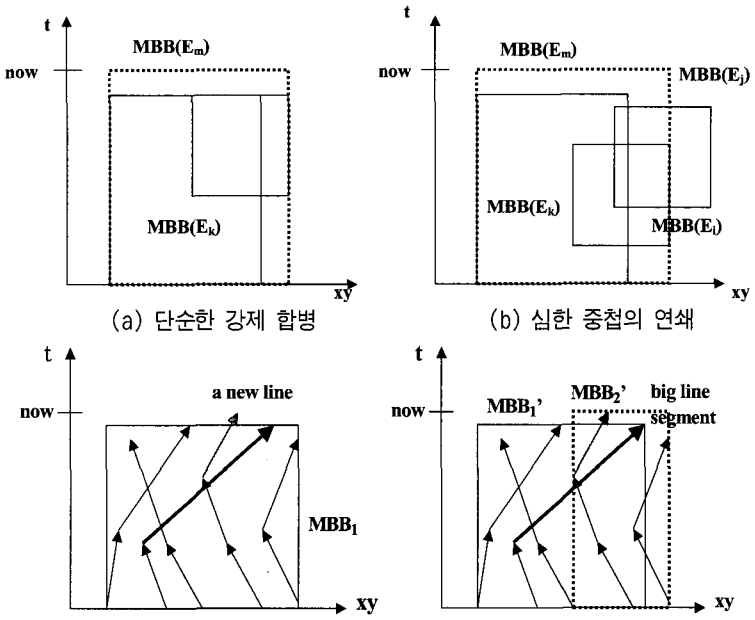

(그림 1) 3D R-tree의 노드 구조와 노드의 MBB

메인을 추가함으로써 시공간 색인으로의 확장이다. 3D R-tree는 기존의 다차원 색인인 R-tree의 3차원 버전으로서, 현재 또는 UC(until changed)에 대해 처리할 수 없는 문제가 있기 때문에 객체의 모든 이동을 시간에 대하여 닫힘(closed)되어 있다고 가정한다. 즉 닫힌-선(closed-line)들만이 삽입 될 수 있다. 특정 시간과 영역에 대한 질의인 영역 질의에서 우수한 성능을 보이지만, 이동체의 궤적 보존이 이루어지지 않기 때문에 궤적 질의에서의 검색 성능은 비효율적인 문제가 있다. (그림 1)은 3D R-tree의 구조와 노드의 MBB을 나타낸 예이다.

TA3DR-tree(Tagged Adaptive 3DR-tree)[4]는 이동체의 이동 특징을 고려하여 3D R-tee를 기반으로 다음과 같은 정책을 이용하여 영역 질의 성능을 개선 하였다. 첫째, (그림 2)의 (a), (b)와 같이 노드간의 중첩을 조사하여 중첩이 심한 노드들을 심한 중첩(High Overlap) 노드로 규정하고, 강제 합병 후 재분할(re-split)을 수행하여 중첩을 최소화 하였다. 둘째, 그림 2의 (c), (d)와 같이 노드간의 심한 중첩을 유발하는 선분을 긴 선분(Big Line Segments)으로 규정하여 절단(Clipping)하여 중첩을 최소화 하였다. 셋째, 시간 축 분할을 비균등 분할하여 추가 삽입이 없는 과거 노드의 공간 활용도를 높였다.

HR-tree[5]는 R-tree에 거래시간 개념을 추가하여 이력 정보를 표현할 수 있으며 연속적인 상태를 표현하기 위하여 R-tree에 중첩(overlapping) 개념을 추가한 것이다. (그림 3)과 같이 HR-tree는 타임스탬프마다 다른 색인 인스턴스로 구성된다. 기본적인 알고리즘은 원래의 트리는 유지하고 상태가 변한 루트와 가지를 대체함으로써 현재와 과거를 유지하는 것이다. 상태가 변하지 않는 가지들은 복제되지 않는다. HR-tree는 이동체들의 이동이 빈번하지 않은 경우에는 효율적이지만, 이동이 많이 발생하는 경우 단말 노드 및 비단말 노드를 새로 생성해야 하고, 특히 영역 질의의 성능이 저하되는 문제를 가진다.

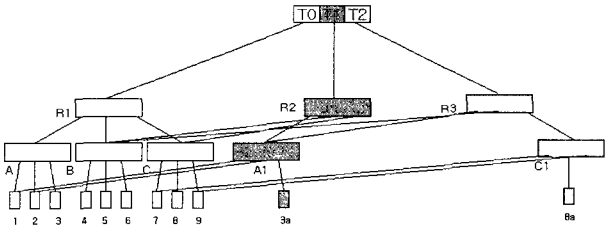
Theodoridis[6]는 HR-tree가 타임스탬프마다 색인 인스턴



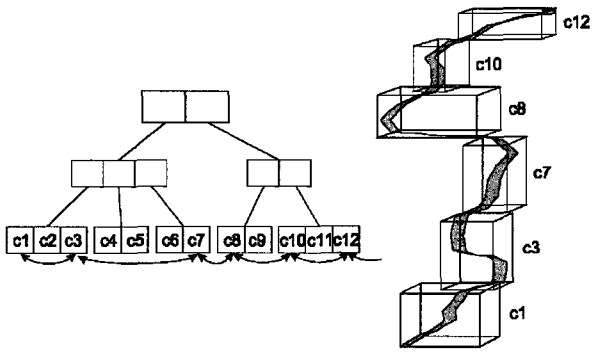

(a) 단순한 강제 합병 (b) 심한 중첩의 연쇄

(c) MBB1에서 오버플로우 발생 (d) 2개로 분할된 MBB들이 심한 중첩인 경우

(그림 2) TA3D R-tree의 강제합병 후 재분할 및 긴 선분 절단 정책




(그림 3) HR-tree의 구조




(그림 4) TB-tree의 구조

스를 생성함으로써 이동체를 저장하기에는 색인의 크기가 크다는 문제점을 소개한다. 이를 해결하기 위해 2개 또는 3개의 타임스탬프를 합쳐서 색인화 함으로써 색인의 크기를 줄였고, 검색 성능이 저하되지 않음을 실험으로 보였다.

STR-tree[2]는 이동체들의 궤적을 선분의 집합으로 나타내고, 보존 파라미터(preservation parameter)를 사용하여 같은 객체의 궤적을 근접한 페이지에 저장하도록 유도 하였다. STR-tree의 부분적인 궤적 보존 정책은 3D R-tree에 비해 궤적 질의 성능 향상 효과가 낮을 뿐만 아니라, 삽입 정책에서 공간 근접성과 같은 공간 구별성이 낮아지기 때문에 영역 질의의 성능이 나쁘다.

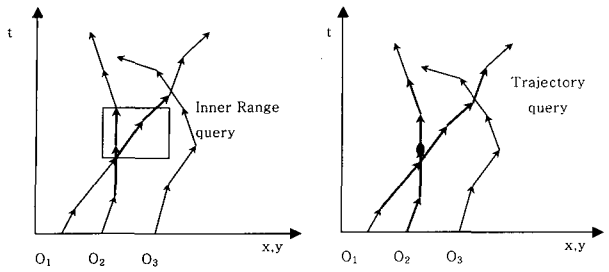
TB-tree[2]는 궤적 질의 및 복합 질의 성능 향상을 위하여 극단적인 궤적 보존 정책을 사용하였다. 하나의 단말 노드는 오직 동일한 궤적에 속하는 선분들만 저장할 수 있다. 공간적으로 근접한 선분들이 같은 궤적이 아니면, 서로 다른 노드에 저장된다. TB-tree는 이동체의 궤적을 3 차원 최소경계박스에 저장하고, (그림 4)와 같이 최소경계박스의 연결로 이동체의 모든 궤적을 표현한다. 단말 노드에서 이동체의 궤적은 양방향 연결 리스트(linked list)로 구성되어 있으므로 궤적에 대한 질의를 처리할 때는 뛰어난 성능을 보인다. 그러나 TB-Tree는 궤적에 대한 질의 성능은 좋은 반면 시간 간격에 대한 질의나 공간 간격에 대한 질의에는 좋지 않은 성능을 보인다.

## 3. 문제 정의

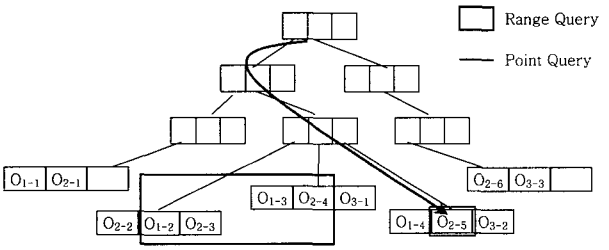
이 장에서는 대표적인 이동체 색인인 3D R-tree와 TB-tree의 복합 질의 처리에 대해 비교 분석을 하고, 3D R-tree 복합 질의 처리를 수행할 경우에 발생하는 문제점을 소개한다.

### 3.1 3D R-tree및 TB-tree 복합 질의 처리에 대한 분석

이동체 데이터베이스에서 복합 질의는 1절에서 소개했던 것과 같이 영역 질의와 궤적 질의를 합친 것과 같다. 즉, "어제 저녁 8시에서 9시까지 부산대학교 앞을 지나간 차량들이 오늘 아침 9시까지의 궤적을 검색하라"와 같은 질의이다. 위의 질의에서 영역 질의는 (그림 5)와 같이 "어제 저녁




(그림 5) 복합 질의의 예




(그림 6) 3D R-tree의 복합 질의 처리

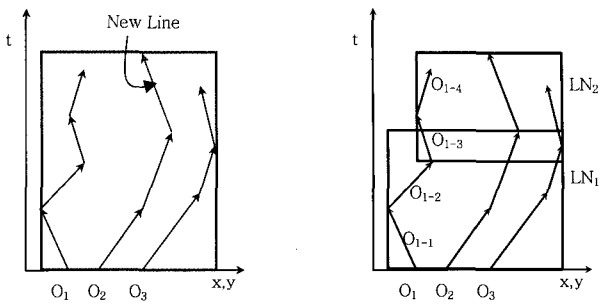
8시에서 9시까지 부산대학교 앞을 지나간 차량들"이 해당되고, 궤적 질의는 영역 질의에 해당되는 차량들 중에서 (그림 5)와 같이 "오늘 아침 9시까지의 궤적을 검색하라"가 해당된다.

3D R-tree에서 복합 질의를 수행하기 위해 두 단계의 처리 과정이 필요하다. 첫 번째 단계는 특정한 영역에 대한 영역 질의(Range query)를 수행한다. 3D R-tree는 공간적으로 근접해 있는 객체를 한 노드에 저장하기 때문에 영역 질의에서 보편적으로 우수한 성능을 보인다. 두 번째 단계는 영역 질의 결과에 대한 이동체의 궤적을 점 질의(Point query)를 수행하여 동일 궤적을 검색해야 한다. 이동체의 궤적 보존이 이루지지 않기 때문에 궤적에 대한 선분을 찾기 위해서는 점 질의를 수행할 수 밖에 없다. 따라서, (그림 6)과 같이 $O_{2\text{-}4}$와 연결되는 궤적 $O_{2\text{-}5}$를 찾기 위해 루트 노드에서부터 검색을 해서 $O_{2\text{-}5}$가 저장되어있는 단말 노드까지 검색을 수행해야 한다.

### 3.2 3D R-tree의 복합 질의 처리에 대한 문제점

3D R-tree의 분할 정책은 최소 영역 증가분 정책(Least Area Enlargement Policy)에 따라 분할된다. (그림 7)의 (a)와 같이 새로운 선분이 삽입됨으로써 오버플로우가 발생하게 되고, 이 노드는 두 개의 노드로 분할이 이루어진다. 첫 번째 단계로써는 노드 내에서 공간상으로 가장 멀리 떨어져 있는 엔트리(Entry)를 선택하여 두 개의 그룹으로 나누는 PickSeed 단계이다. 다음 단계는 나머지 엔트리들을 나누어진 두 개의 그룹에 분배를 하는 단계이다. 엔트리들을 분배할 때에는 두 개의 그룹 중에 영역이 최소로 증가되는 그룹으로 분배를 하게 된다. 따라서 (그림 7)의 (b)와 두 개의 노드로 분할이 이루어진다.

(그림 7)의 (b)와 같이 분할된 두 개의 노드에서 $O_1$의 궤적을 검색할 경우에 먼저 $O_{1\text{-}1}$을 탐색했다고 가정을 한다면 $O_{1\text{-}2}$의 궤적은 같은 노드에 저장되어 있기 때문에 더 이상의 노드 접근 없이 검색을 할 수 있다. 그러나 $O_{1\text{-}3}$의 궤적을 검색하기 위해서는 더 이상 $LN_1$의 단말 노드에는 $O_1$의 궤적이 저장되어 있지 않기 때문에 $O_{1\text{-}2}$의 궤적에 대한 점 질의를 수행해야만 한다. 이러한 점 질의는 루트에서부터 탐색을 해야 하기 때문에 많은 노드 접근 횟수가 필요로 한다. 또한, $O_1$의 궤적들이 서로 다른 단말 노드에 각각 저장되어 있다고 한다면 궤적을 찾기 위해 점 질의를 유발 시켜 노드 접근 횟수를 더욱더 증가 시킬 것이다.




(a) 단말 노드의 오버플로우 (b) 단말 노드 분할정책의 문제점

(그림 7) 3D R-tree의 분할 정책의 문제점

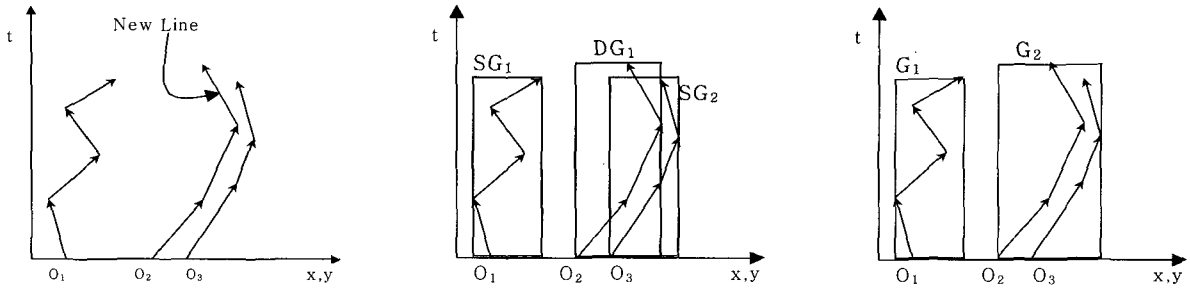
따라서, 3D R-tree에서 궤적 질의를 수행 할 경우에는 노드 접근 횟수를 줄일 수 있는 방법으로써는 점 질의 수행을 최소화 해야만 한다. 점 질의 처리 횟수를 줄일 수 있는 방법으로써는 하나의 단말 노드 내에 동일 이동체의 궤적이 최대한 많이 저장되어야 한다. 즉, 단말 노드 내에 동일 이동체의 궤적을 클러스터링 할 수 있는 정책이 필요하다.

## 4. 동일 궤적 그룹화에 의한 분할 정책

3D R-tree에서의 분할 정책은 단말 노드 내에 동일 이동체의 궤적들이 서로 다른 노드에 저장될 확률이 많다는 문제점이 있었다. 이 논문에서 이러한 문제점을 해결하고자 동일 궤적을 그룹화하여 분할하는 공간 축 분할 정책과 이동체의 특징상 시간에 따라 보고를 하기 때문에 시간 축으로 비 균등 분할을 하여 공간 활용도를 높이고자 하였다.

### 4.1 단말 노드의 공간 축 분할 정책

단말 노드의 공간 축 분할은 궤적을 보존을 하기 위한 분할 정책이다. 새로운 선분의 삽입에 따른 단말 노드의 오버플로우가 발생하면 두 개의 노드로 분할을 한다. 분할 과정은 다음과 같이 세 단계 과정으로 이루어진다. 첫 번째 단계로는 동일 궤적 그룹화 과정이다. 여기에서 동일 궤적은 같은 이동체가 시간에 따라 보고한 선분(Line Segment)이다. 두 번째 단계로써는 여러 동일한 이동체의 궤적들로 그룹화된 그룹들 중에 가장 멀리 떨어져 있는 그룹을 시드 그룹(SG: Seed Group)이라 정의 하고, 그 시드 그룹을 선택하는 과정이다. 세 번째 단계로써는 시드 그룹을 제외한 나머




(a) 노드의 오버플로우 (b) 동일 궤적 그룹화 및 시드 그룹선택 (c) 노드 분할의 결과

(그림 8) 단말 노드 공간 축 분할 정책

지 그룹을 분배 그룹(DG: Distribution Group)이라고 정의하고 그 분배 그룹들을 각각의 시드 그룹에 공간 근접성에 따라 분배를 하는 과정이다.

(그림 8)은 동일 궤적을 그룹화에 따른 분할 정책을 보여주고 있다. (그림 8)의 (a)는 새로운 선분이 삽입 됨으로써 단말 노드가 오버플로우가 된 것을 보여주고 있다. (그림 8)의 (b)에서는 $O_1$, $O_2$, $O_3$의 궤적들 별로 그룹화를 하고, 그룹들 중에 공간상으로 가장 멀리 떨어져 있는 그룹을 시드 그룹으로 선택한 것을 보여 주고 있다. (그림 8)의 (c)는 분배 그룹인 $DG_1$을 시드 그룹 $SG_2$에 분배되어 두 개의 노드로 분할된 결과이다. 동일 궤적 그룹화 의한 분할을 함으로써 하나의 단말 노드 내에 동일 이동체의 궤적들을 최대한 많이 저장 시킴으로써 궤적 질의 수행 시에 점 질의의 횟수를 줄일 수 있다.

### 4.2 궤적 분리

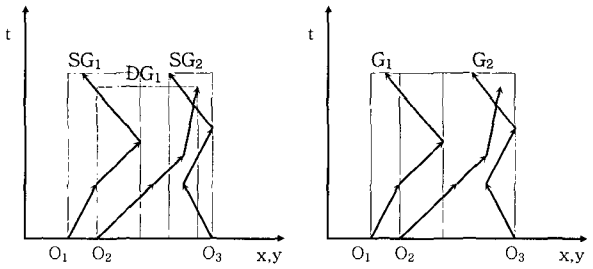
**정의 1. 분할 노드 간의 중첩률**

궤적 그룹화 과정에서 분할되어 생성되는 두 개의 노드를 $G_i$, $G_j$ 라 한다. 노드내의 궤적을 모두 포함하는 최소경계박스(MBB)를 그룹의 MBB라 하면, 분할 노드 간의 중첩률은 다음과 같다.

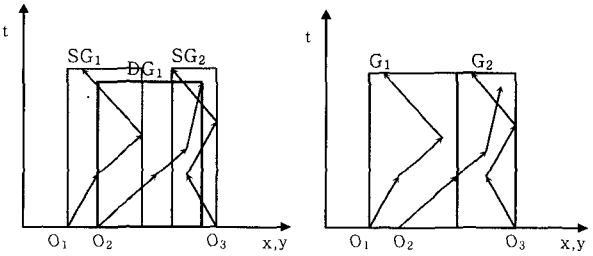
$$OverlapRate(G_i, G_j) \leftarrow area(G_i.MBB \cap G_j.MBB) / \min(area(G_i.MBB), area(G_j.MBB))$$

공간 축 분할 정책에서 두 개의 시드 그룹을 선택한 후에 분배 그룹을 분배한다. (그림 9)와 같이 두 개의 시드 그룹($SG_1$, $SG_2$)에 분배그룹($DG_1$)을 분배한 결과 두 노드 간의 심한 중첩이 발생할 수 있다. 즉, 분할 노드 간의 중첩률에 임계값을 두어 이 보다 더 클 경우에 심한 중첩이라 정의하고, 이러한 심한 중첩이 발생할 경우에는 분배 그룹을 궤적의 구성 요소인 선분으로 분리를 하여 분배한다. 노드 간의 중첩률의 임계값은 실험을 통하여 최적의 임계값을 구하고자 한다.

(그림 10)은 분배 그룹($DG_1$)의 심한 중첩에 의해서 분배 그룹을 궤적별로 분리해서 각 그룹에 중첩이 최소화 되도록 분배된 예이다. 이렇게 분할 됨으로써 영의 질의 및 점 질의에서 중첩이 최소화 되어 노드 접근 횟수를 줄일 수 있다.



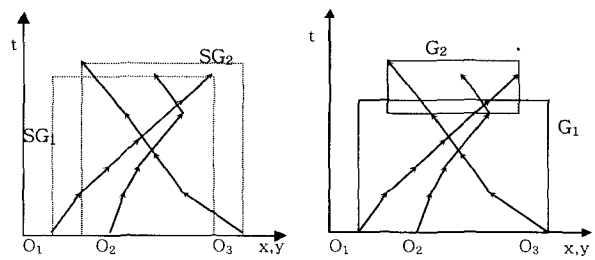

(그림 9) 분배 그룹의 심한 중첩




(그림 10) 궤적 분리에 의한 분배

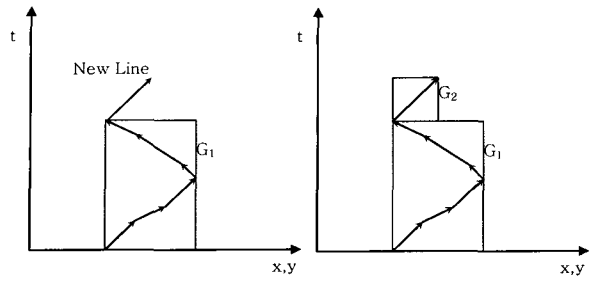
### 4.3 단말 노드의 시간 축 분할 정책

공간 축 분할일 경우, 첫 번째 단계로서 동일 궤적끼리 그룹을 생성하고 생성된 그룹들 중에서 시드 그룹을 선택한다. 다음 단계에서 시드 그룹(SG) 간에 심한 중첩이 발생할 수 있다. 심한 중첩은 영역 질의 및 점 질의시에 많은 노드 접근을 유발 시키기 때문에 성능에 좋지 않다. 이러한 경우에는 분배 그룹(DG)를 분배를 하더라도 심한 중첩이 발생한다.




(a) 시드 그룹 간의 심한 겹침 (b) 시간 축 분할의 결과

(그림 11) 시간 축 분할 정책의 예(1)




(a) 시간 축 분할 조건 (b) 시간 축 분할 결과

(그림 12) 시간 축 분할 정책의 예(2)

따라서, 시드 그룹간의 심한 중첩이 발생 할 경우에는 시간 축 분할을 해서 심한 중첩을 제거하고자 하고, 또한 시간 축으로 비균등 분할을 하여 공간 활용도를 높이고자 하였다. (그림 11)의 (b)에서 $G_1$의 노드에는 더 이상 궤적이 삽입되지 않는 노드이기 때문에 공간 활용도를 높였고, 이동체는 최근 시간 값에 따라 계속 보고를 하기 때문에 $G_2$의 노드에는 최근 보고한 궤적만을 저장하고 있다.

(그림 12)와 같이 단말 노드 내에 하나의 이동체의 궤적들만이 저장 될 경우가 생긴다. 즉, TB-tree 단말 노드와 동일하게 저장 될 수 있다. 이러한 경우에도 시간 축 비 균등 분할을 하여 공간 활용도를 높이고자 하였다.

### 4.4 노드 분할 알고리즘

4.1절에서 4.3절까지 공간 축 분할 정책, 궤적 분리, 시간 축 분할 정책을 제안 하였다. 이 절에서는 단말 노드 분할 정책에 대한 알고리즘을 제시한다. 단말 노드 분할 알고리즘은 크게 공간 축 분할과 시간 축 분할로 나누어 진다. 공간 축 분할에서 다시 조건에 따라 궤적 분리 여부를 결정하게 된다.

〈표 1〉 단말 노드 분할 정책의 조건

| 분할축 선정 | 축선정 조건 | 세부정책 |
|---|---|---|
| 공간 축 분할 | SG간의 중첩률 < 임계값 | SG와 DG간의 중첩률 >= 임계값 → 궤적 분리 |
| 시간 축 분할 | SG간의 중첩률 >= 임계값 or 그룹의 개수가 1 | |

(알고리즘 1)은 동일궤적 그룹화에 의한 분할 정책에 대한 알고리즘이다. Grouping-trajectory() 함수에서 동일 이동체의 궤적별로 그룹화한다. PickSeed() 함수는 그룹들 중에 공간상으로 가장 멀리 떨어져 있는 두 개의 그룹을 선택한다. OverlapRate() 함수는 두 그룹간의 중첩율을 구하는 알고리즘이고, 심한 중첩(HRO : Highly Rate of Overlap)일 경우 시간 축으로 분할을 한다. 심한 중첩의 임계값은 실험을 통해서 최적의 임계값을 구하고자 한다.

(알고리즘 2)는 공간 축 분할 정책에 대한 알고리즘이다. 시드 그룹(SG)을 제외한 나머지 그룹 즉, 분배 그룹(DG)을 분배하기 위한 알고리즘이다. PickNext() 함수는 R-tree의 PickNext() 함수와 동일하다. PickNext() 함수에 의해 선택된 $DG_i$ 의 그룹을 두 개의 시드 그룹 중 영역이 적게 증가되는 그룹에 분배를 하게 된다. 또한 $DG_i$와 $SG_1$, $SG_2$과의 중첩율을 검사(HighOverlapCheck() 함수)해서 임계값보다 클 경우에는 $DG_i$를 각각의 선분으로 분리(IsolatedTrajecoty() 함수)를 해서 각 선분을 분배를 분배한다.

(알고리즘 3)은 시간 축 분할 정책에 대한 알고리즘이다. 가장 최근에 보고한 이동체의 궤적을 $G_1$에 저장하고, 이전에 보고한 이동체 궤적에 대해서는 $G_2$에 저장함으로써 $G_2$의 공간 활용도를 높였다. 이동체는 시간에 따라 보고되기 때문에 과거 궤적만을 저장하고 있는 $G_2$에는 더 이상의 궤적이 저장되지 않는다. 따라서 공간 활용도를 최대한 높여줌으로써 검색 성능을 높이고자 하였다.

(알고리즘 1) 그룹화에 의한 단말 노드 분할 알고리즘

```
LN_m is an overfull node, Let LS_new be a new line segment.
PROCEDURE  LeafNodeSplit(LN_m, LS_new)
Line_Set   = the collection of all line segments within LN_m
Group_Set = Groupping_trajectory(Line_Set)
  // Groupping that trajectories of same moving objects within Line_Set
IF (Group_Set.Size() == 1)
   TimeAxisSplit(Group_Set)
ELSE
   PickSeed(SG_1, SG_2, Group_Set)
    // Choose SG(Seed Group)
   IF OverlapRate(SG_1, SG_2) > HRO
     TimeAxisSplit(Group_Set)
   ELSE
     SpatialAxisSplit(SG_1, SG_2, Group_Set)
```

(알고리즘 2) 공간 축 분할 알고리즘

```
G_1, G_2 is Seed Group.
Group_Set is Groups that trajectories of same moving objects
PROCEDURE  SpatialAxisSplit(G_1,G_2, Group_Set)
While ( ! Group_Set.empty())
  DG_i = PickNext(Group_Set) // DG is Distribution Group
  //calculate d1 = the area increase required in the covering rectangle
  //of G_1 to include Group_Set_i . Calculate d2 similarly for G_2
  //Choose any Group whth maximum difference between d_1 and d_2
  IF (HighOverlapCheck(G_1.mbb, G_2.mbb, DG_i.mbb) > HRO)
    IsolatedTrajectory(G_1, G_2, DG_i)
  ELSE
    G1 or G2 ← DG_i
    // Adding the DG_i to the Group with smaller area
```

(알고리즘 3) 시간 축 분할 알고리즘

```
Group_Set is Groups that trajectories of same moving objects
PROCEDURE  TimeAxisSplit(Group_Set)
G1  // UCMBB(Until Changed MBB)
G2 // FixedMBB
While(! Group_Set.empty())
   DGi = Group_Seti
   IF (DGi is UCMBB)
     G1 = DGi
   ELSE
     G2 = DGi
```

## 5. 비단말 노드의 추가정보

3D R-tree의 경우에는 이동체의 궤적은 서로 다른 단말 노드에 저장 될 수 있다. 동일한 궤적을 서로 다른 노드에 저장되어 있는 노드들을 연결 노드(Connected Node)라 한다. 연결 노드에 대한 정보를 표현하기 위해 같은 부모를 가진 자식 노드 간의 궤적 연결성을 (그림 13)과 같이 CNInfo(Connected Node Information)에 저장한다. CNInfo는 자식 노드만큼의 비트가 필요하고, 하나의 선분이 삽입될 때 갱신된다. 단말 노드들의 궤적 연결성을 저장함으로써 비단말 노드 분할에서 궤적 클러스터링을 가능하게 한다. 궤적 질의에서 너비우선 탐색(Breadth first search)을 수행함으로써 루트 노드에서 단말 노드까지 반복 검색하는 것을 최소화하여 불필요한 노드 방문을 줄일 수 있다. 비단말 노드의 구조는 (그림 13)의 (a)와 같다.

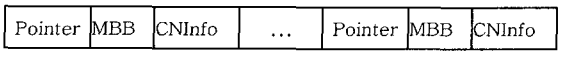
(그림 13)의 (c)에서 $O_2$의 궤적을 찾고자 할 경우에 $O_{2-1}$에서 기존에는 점 질의를 통해 루트 노드에서부터 탐색을 수행했지만, $O_{2-1}$을 저장하고 있는 단말 노드인 $LN_1$에서 부모 노드인 $NN_1$으로 가서 $NN_1$의 연결노드정보(CNInfo)를 검사한다. $NN_1$의 두 번째 엔트리인 $LN_2$가 연결노드로 검사될 것이다. 따라서 루트 노드를 통하지 않고 바로 연결 노드인 $LN_2$을 탐색을 하여 $O_{2-2}$와 $O_{2-3}$을 찾을 수 있다.

기존의 색인에서 비단말 노드는 하위노드를 가리키는 포인터와 3차원 MBB인 ($cp$, $x$, $x'$, $y$, $y'$, $t$, $t'$) 형태의 엔트리를 가진다. 이 경우 각 정보의 크기를 4byte라하고 node size를 1024byte라하면 비단말 노드는 1024/28 = 약 36개의 엔트리를 포함할 수 있다. 반면에 비단말 노드에 위와 같이 연결정보를 추가하는 경우에는 2차 방정식의 형태인 $1024/(24+x)=x$로 나타내어지며 약 19개의 엔트리를 포함할 수 있다. 따라서 연결정보를 추가하는 경우에는 그렇지 않은 경우보다 색인이 약 1.89배 증가되지만 실험결과 복합질의의 성능이 매우 빨라짐을 알 수 있었다. 이것은 6장 실험에서 언급한다.

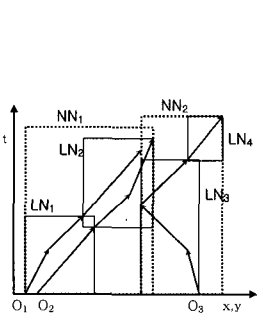
(알고리즘 4)는 비단말 노드의 추가정보에 따른 복합 질의 알고리즘이다. Inner Range Query에 대한 결과로 $RS_{SQ}$라고 한다면 $RS_{SQ}$의 부모 노드의 연결노드정보를 통해서 단말 노드를 찾아서 연결되어진 궤적을 찾는다.

(알고리즘 4) 비단말 노드 추가정보에 따른 복합 질의 알고리즘

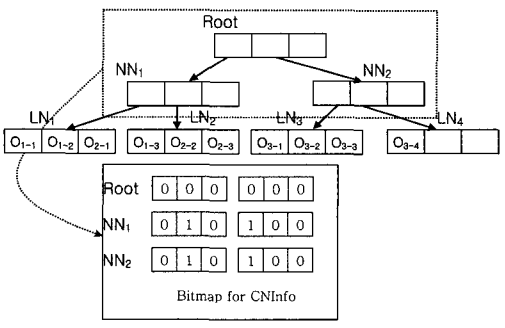
```
RSRQ is Result Set of the inner Range Query
Period is the Value of Outer Range Query
Algorithm SearchCombinedQuery (RSRQ, period1, period2)
While(! RSRQ.empty())
  LSi = RSRQ-i // LS is Line Segment
  IF(LSi < period1 || LSi > period2)
    RSRQ-i.pop()
  ELSE
    PN ← parent Node including LSi
    CNn = PN.ConnectedNode(i)
    For Each i IN n
      LSnew = ConnectedTrajectory(LSi, CNi) //search connected trajectory
      RSRQ-n+1.push(LSnew)
    End For Each
End while
```




(a) 비단말 노드의 자료 구조




(b) 비단말 노드의 MBB




(c) 비단말 노드의 추가 정보의 예

(그림 13) 비단말 노드의 구조

## 6. 실 험

이 장에서는 논문에서 제안하는 색인의 성능을 평가하기 위해 이동체의 색인 중에서 TB-tree, 3D R-tree을 개선한 TA3DR-tree와 성능을 비교하고자 한다. 색인의 성능 비교의 기준은 색인의 크기, 질의 처리 성능을 실험을 통해 비교하였다. 색인의 크기는 공간 활용도와 색인의 총 노드 수를 기준이 된다. 검색 질의의 경우 CPU 연산 비용이 작기 때문에, 영역 질의 처리와 복합 질의 처리는 질의 수행 동안에 방문한 노드의 수가 기준이 된다.

### 6.1 실험 데이터 및 환경

제안한 TP3DR-tree, TA3DR-tree, TB-tree는 각각 C++ 언어로 구현하였고, 윈도우 2000에서 256MB 메인 메모리, CPU Pentium IV 1.7Ghz를 장착한 PC를 이용하여 실험하였다.

일반적으로 색인의 성능 검증을 위해서는 다양한 환경에서의 색인 성능을 평가할 수 있는 테스트 환경이 필요하다. 이 논문에서는 이러한 연구 중 이동체 생성에 대한 대표적인 도구인 GSTD 생성기[3] 사용하여 다양한 조건에서 이동체의 위치 데이터를 생성하였다. 생성된 이동체 데이터 집합을 이용하여 구축된 색인을 이용해서 다양한 질의에 대한 성능 평가를 수행 하였다.

#### 6.1.1 실험 데이터

색인의 성능을 평가하기 위해 색인의 공간 활용도, 영역 질의 및 복합 질의 성능을 평가하였다. 영역 질의와 복합 질의의 성능은 노드 접근 횟수로 측정한다. <표 2>와 같이 이동체의 수($N_{MO}$)와 이동체의 보고 횟수($N_{RE}$)를 달리하여 성능을 평가하였으며 영역 질의의 크기(RQS)는 각 도메인 크기의 5%, 10% 이다. 영역 질의 데이터는 RQS 값 마다 1000개의 영역을 생성하며 도메인의 범위를 벗어나지 않게 임의의 위치를 중점을 가지도록 생성한다. 또한 복합 질의의 크기(CQS)는 Inner Range 1%→ Outer Range 5%, 1%→10%, 5%→10%에 해당하는 1000개의 질의에 대한 성능 평가를 하였다.

이동체 데이터로는 <표 3>과 같이 랜덤 데이터와 비대칭인 데이터를 사용하였다. 랜덤 데이터는 이동체 수가 1000개이고 보고 횟수가 각각 500, 1000, 2000인 데이터를 사용하였다. 비대칭 데이터도 이동체 수가 1000개로 동일하고, 보고 횟수가 각각 500, 1000, 2000인 데이터를 사용하였다.

〈표 2〉 성능 평가 인자

| 성능평가 인자 | 설 명 |
|---|---|
| $N_{MO}$ | 이동체의 수 |
| $N_{RE}$ | 이동체 보고 횟수 |
| RQS | 영역 질의 크기 |
| CQS | 복합 질의 크기 |

〈표 3〉 이동체 데이터

| 약어 | 이동 방향 | 이동 간격 |
|---|---|---|
| G-Normal | 임의 | 보통 |
| G-Skewed | 동북쪽 | 짧다 |

TP3DR-tree는 <표 4>와 같이 중첩률의 임계값을 정의하여 색인을 구축하였다.

〈표 4〉 중첩률에 따른 임계값 정의

| 인자 | 설 명 | 값 |
|---|---|---|
| SGSOR | 시드 그룹(SG)간의 중첩률의 임계값 | 5%, 10%, 20%, 35% |
| SGDGOR | 시드 그룹(SG)과 분배 그룹(DG)간의 중첩률의 임계값 | 5%, 10%, 25% |
| CSINOR | 단말노드 선택(ChooseSubtree)시 중첩인 노드(Intersected Node)간의 중첩률의 임계값 | 1%, 5%, 10%, 20% |

실험 인자에 따른 색인 이름은 <표 5>와 같다.

〈표 5〉 색인 약어(1)

| 인자 | 인자 | 색인 약어 |
|---|---|---|
| SGSOR = 5 % | SGDGOR = 5 % | TP3DR-tree SG5DG5 |
| SGSOR = 5 % | SGDGOR = 10 % | TP3DR-tree SG5DG10 |
| SGSOR = 5 % | SGDGOR = 25 % | TP3DR-tree SG5DG25 |
| SGSOR = 10 % | SGDGOR = 5 % | TP3DR-tree SG10DG5 |
| SGSOR = 10 % | SGDGOR = 10 % | TP3DR-tree SG10DG10 |
| SGSOR = 10 % | SGDGOR = 25 % | TP3DR-tree SG10DG25 |
| SGSOR = 20 % | SGDGOR = 5 % | TP3DR-tree SG20DG5 |
| SGSOR = 20 % | SGDGOR = 10 % | TP3DR-tree SG25DG10 |
| SGSOR = 20 % | SGDGOR = 25 % | TP3DR-tree SG20DG25 |
| SGSOR = 35 % | SGDGOR = 5 % | TP3DR-tree SG35DG5 |
| SGSOR = 35 % | SGDGOR = 10 % | TP3DR-tree SG35DG10 |
| SGSOR = 35 % | SGDGOR = 25 % | TP3DR-tree SG35DG25 |

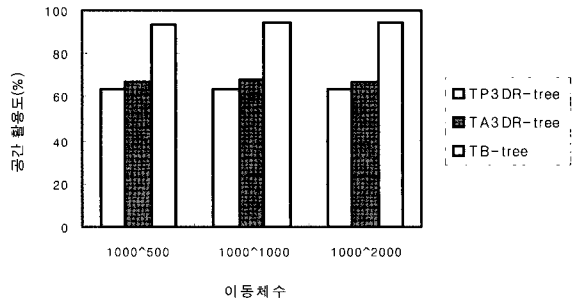
또한, 3D R-tree에서 $CSIN_{OR}$만을 추가 했을 경우에 질의 성능이 어떠한 변화를 이루는지에 대해서 알아보고자 하였다. 따라서 <표 6>과 같이 색인을 구축을 했다.

〈표 6〉 색인 약어(2)

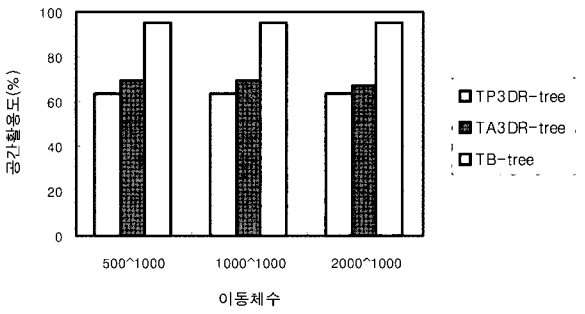
| 인 자 | 색인 약어 |
|---|---|
| CSINOR = 0 % | 3D R-tree CS0 |
| CSINOR = 1 % | 3D R-tree CS1 |
| CSINOR = 5 % | 3D R-tree CS5 |
| CSINOR = 10 % | 3D R-tree CS10 |
| CSINOR = 20 % | 3D R-tree CS20 |

### 6.2 공간 활용도의 비교

높은 공간 활용도는 일반적으로 트리의 높이와 같이 질의 비용을 감소시킨다. TA3DR-tree는 시간 축에 따른 비균등




(그림 14) 공간활용도(G-Normal)




(그림 15) 공간 활용도(G-Skewed)

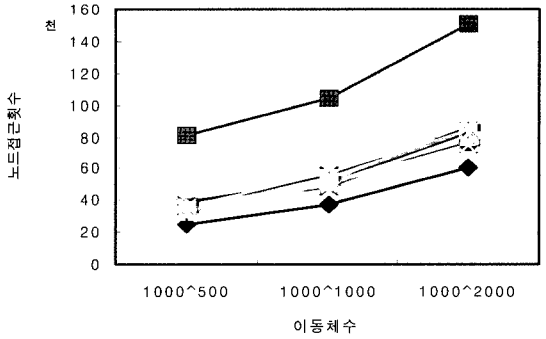
분할 정책과 합병 및 재분할 때문에 공간 활용도가 약 67%이다. TB-tree는 같은 객체의 궤적을 모아서 데이터 페이지에 저장하기 때문에, 90% 이상의 공간 활용도를 가진다. 제안하는 TP3DR-tree는 시간 축에 의한 비균등 분할을 하기 때문에 3D R-tree보다는 좋은 63%이상의 공간 활용도를 가진다.

(그림 14)는 GSTD 알고리즘으로 생성한 G-Normal 데이터 집합에서 이동체 수에 따른 공간 활용도 변화를 보여 준다. TP3DR-tree는 시간 축의 비균등 분할의 효과로 63%~64%의 공간 활용도를 가진다. (그림 15)는 GSTD 알고리즘으로 생성한 G-Skewed 데이터 집합에서 이동체 수 따른 공간 활용도 변화를 보여 준다. TP3DR-tree는 시간 축의 비균등 분할의 효과로 63%~64%의 공간 활용도를 가진다.

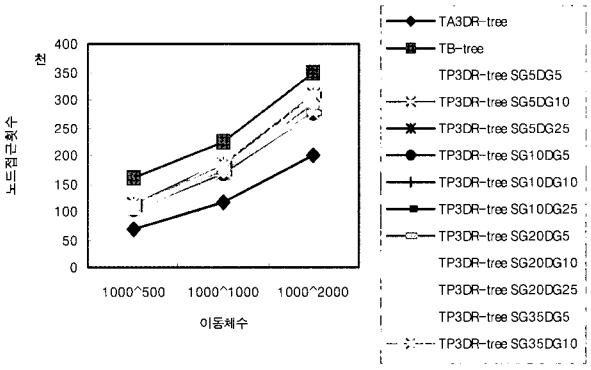
### 6.3 영역 질의의 성능 평가

영역 질의의 성능은 RQS 값에 따라 2가지 종류의 영역 질의를 각각 1000개 생성하여 수행하였다. 영역 질의는 색인의 전체 도메인을 벗어나지 않는 범위에서 임의 생성하였다. 방문 횟수는 1000번의 영역 질의를 수행하는 동안 방문한 노드의 총 횟수이다.

(그림 16)은 정규 분포 데이터에 대한 영역질의 실험 결과이다. 그림에서 TB-tree는 노드간의 중첩 현상으로 인하여 TP3DR-tree에 비해 성능이 나쁘다. TA3DR-tree는 강제 합병 정책과 긴 선분의 절단 정책을 사용하여 노드간의 중첩을 최소화하기 때문에 성능이 우수하다. TP3DR-tree의 경우 여러 인자를 두어 실험한 결과 영역 질의에서는 TA3DR-tree와 TB-tree사이의 성능을 유지 한다.

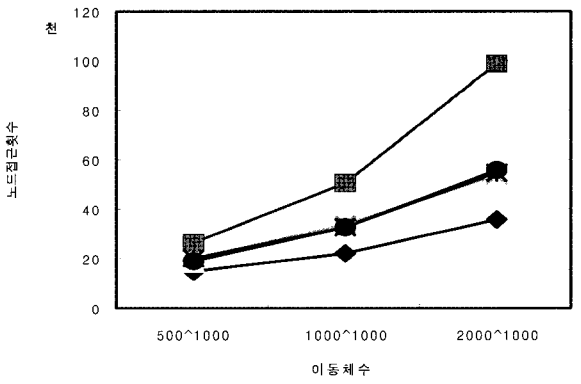



(a) 5% 영역질의

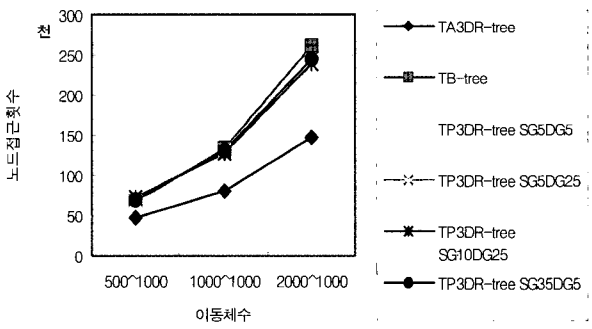



(b) 10% 영역질의

(그림 16) 영역 질의 성능 비교(G-Normal 데이터)




(a) 5% 영역질의




(b) 10% 영역질의

(그림 17) 영역 질의 성능 비교(G-Skewed 데이터)

비대칭 분포 데이터의 영역 질의 성능 평가는 TP3DR-tree의 분할 정책에 따른 인자를 두어 색인을 구축하였고, 또한 3DR-tree에서의 개선된 단말 노드 선택 알고리즘을 추가하여 색인을 구축하고, 실험을 수행하였다. 비대칭 분포 데이터는 (그림 17)과 같이 TP3DR-tree의 분할 정책에 따른 성능은 TA3DR-tree와 TB-tree의 사이에 위치한다.

### 6.4 복합 질의의 성능 평가

영역 질의의 성능은 CQS 값에 따라 3가지 종류의 영역 질의를 각각 1000개 생성하여 수행하였다. 영역 질의는 색인의 전체 도메인을 벗어나지 않는 범위에서 임의 생성하였다. 방문 횟수는 1000번의 복합 질의를 수행하는 동안 방문한 노드의 총 횟수이다.

정규분포 데이터의 복합 질의는 TP3DR-tree의 노드 분할 정책에 따른 인자를 두어 색인을 구축하였고, 또한 6절에서 제안한 비단말 노드 추가정보를 이용해서 색인을 구축하였고, 그에 따른 개선된 복합 질의 알고리즘을 사용하여 복합 질의를 수행하였다.

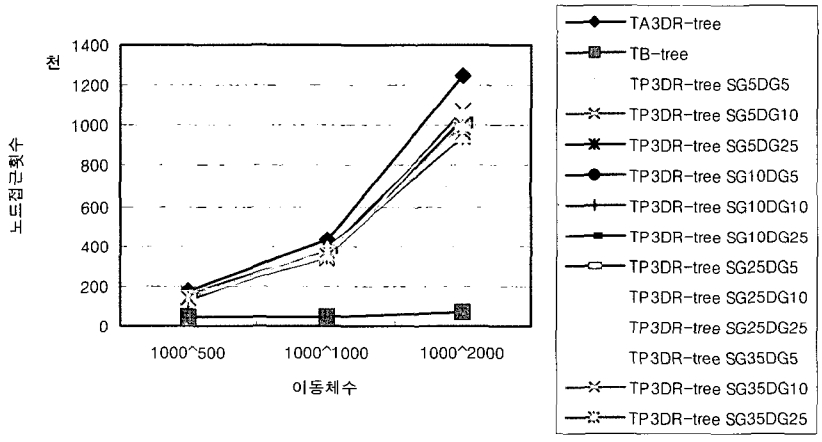
(그림 18)은 TP3DR-tree의 분할 정책에 따라 인자를 두어 구축한 색인의 복합 질의 성능 평가이다. 단말 노드 분할 정책에 따른 동일 궤적 클러스터링 효과에 의해서 TP3DR-tree는 TA3DR-tree보다 15%~30%의 성능 향상을 보인다.

(그림 19)는 TP3DR-tree의 분할 정책에 따라 인자(SG25DG5)를 두고, 비단말 노드 추가정보를 저장해서 색인을 구축하였고, 새로운 복합 질의 처리 알고리즘을 사용하여 질의 처리를 수행한 결과이다. TP3DR-tree의 성능이 TB-tree와 유사한 성능 결과가 나왔으며, 원인은 궤적 질의를 하기 위해 점 질의를 중복해서 처리를 해야 했던 기존의 복합 질의를 비단말 노드의 추가 정보를 통해 점 질의를 하지 않고, 부모노드에서 연결된 단말 노드를 바로 검색 할 수 있기 때문이다.

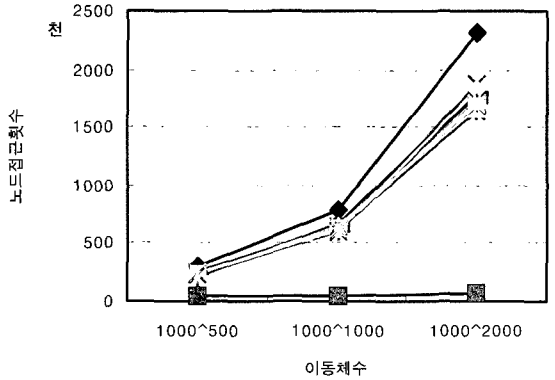
비대칭분포 데이터의 복합 질의는 TP3DR-tree의 노드 분할 정책에 따른 인자를 두어 색인을 구축하였고, 또한 3DR-tree에서의 개선된 단말 노드 선택 알고리즘을 추가하여 색인을 구축하고, 실험을 수행하였다.

(그림 20)은 TP3DR-tree의 분할 정책에 따라 인자를 두어 구축한 색인의 복합 질의 성능 평가이다. 단말 노드 분할 정책에 따른 동일 궤적 클러스터링 효과에 의해서 TP3DR-tree는 TA3DR-tree보다 15%~30%의 성능 향상을 보인다.

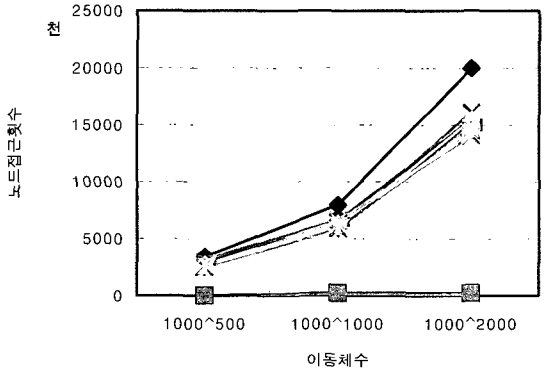
(그림 21)은 TP3DR-tree의 분할 정책에 따라 인자(SG25DG5)를 두고, 비단말 노드 추가정보를 저장해서 색인




(a) Inner 1% → Outer 5%의 복합질의

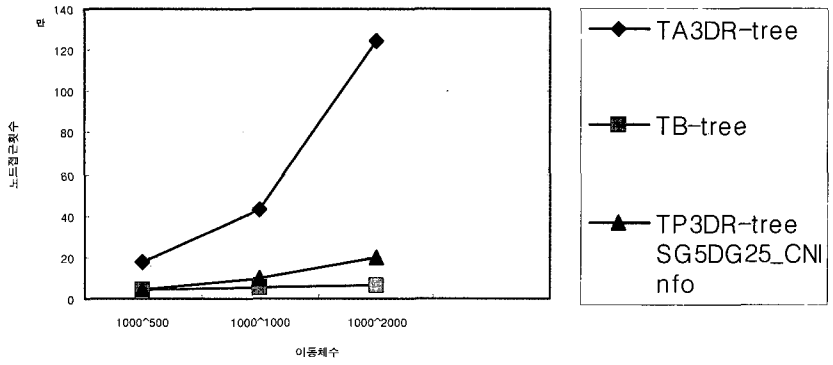



(b) Inner 1% → Outer 10%의 복합질의

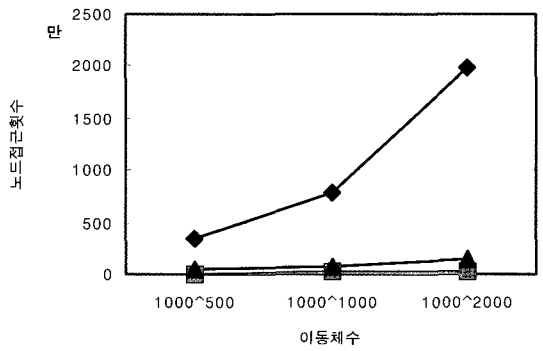



(c) Inner 5% → Outer 10%의 복합질의

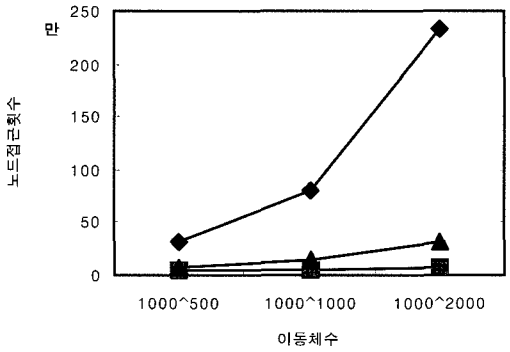
(그림 18) TP3DR-tree의 분할 정책에 따른 복합 질의 성능평가(G-Normal)




(a) Inner 1% → Outer 5%의 복합질의

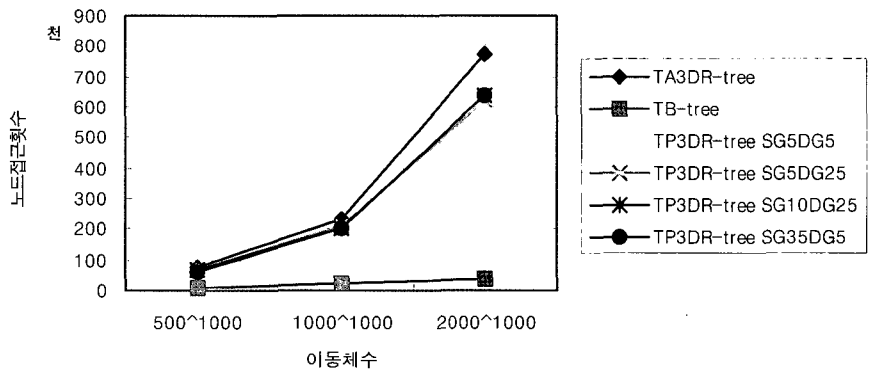



(b) Inner 1% → Outer 10%의 복합질의

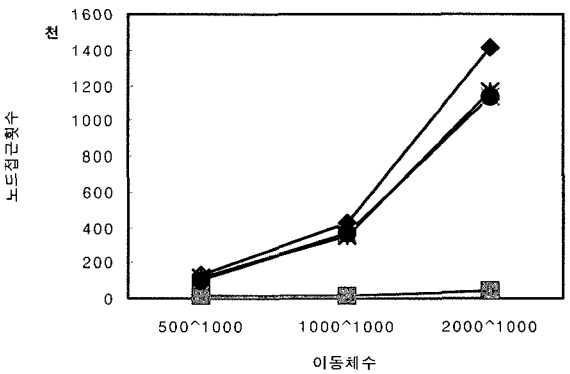



(c) Inner 5% → Outer 10%의 복합질의

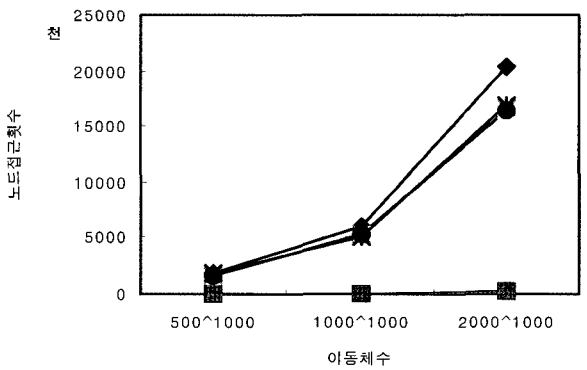
(그림 19) 비단말 노드의 연결 정보를 추가한 복합질의 성능평가(G-normal)




(a) Inner 1% → Outer 5%의 복합질의

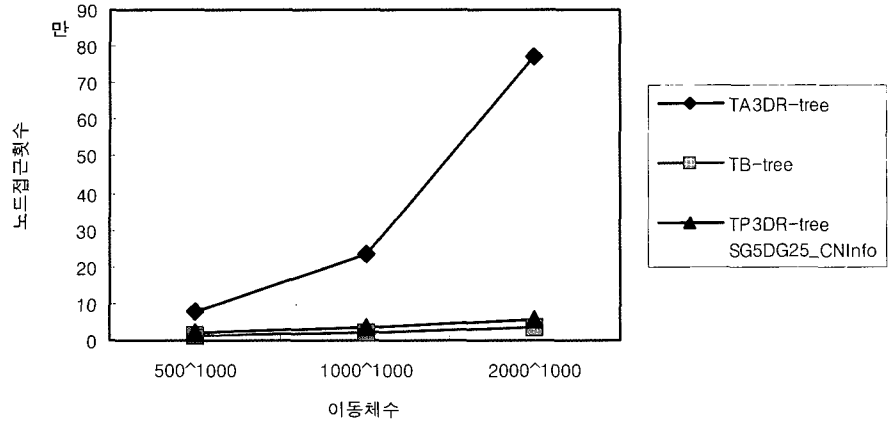



(b) Inner 1% → Outer 10%의 복합질의

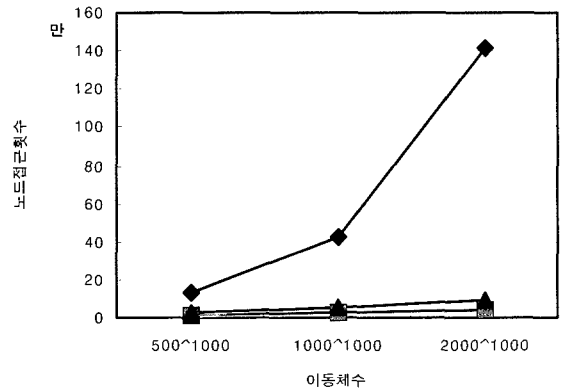



(c) Inner 5% → Outer 10%의 복합질의

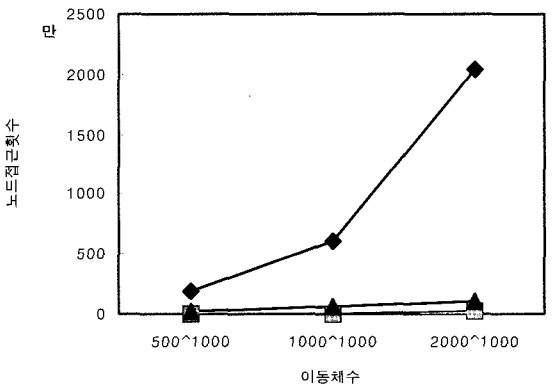
(그림 20) TP3DR-tree의 분할 정책에 따른 복합 질의 성능 비교(G-Skewed)




(a) Inner 1% → Outer 5%의 복합질의




(b) Inner 1% → Outer 10%의 복합질의




(c) Inner 5% → Outer 10%의 복합질의

(그림 21) 비단말 노드의 연결 정보를 추가한 복합질의 성능평가(G-Skewed)

을 구축하였고, 새로운 복합 질의 처리 알고리즘을 사용하여 질의 처리를 수행한 결과이다. TP3DR-tree의 성능이 TB-tree와 유사한 성능 결과가 나왔으며, 원인은 궤적 질의를 하기 위해 점 질의를 중복해서 처리를 해야 했던 기존의 복합 질의를 비단말 노드의 추가 정보를 통해 점 질의를 하지 않고, 부모노드에서 연결된 단말 노드를 바로 검색 할 수 있기 때문이다.

### 6.5 데이터의 분포에 따른 실험결과의 평가

일반적으로 2차원 공간색인에서는 실험데이터의 분포가 고른 경우에 최적의 성능을 나타낸다. 즉, 실험데이터가 한쪽으로 치우친 경우에는 노드의 과도한 겹침이 발생하여 질의의 성능이 저하된다.

본 논문에서 사용한 실험데이터는 공간적으로 고른 분포인 G-Normal 데이터와 공간적으로 한쪽으로 치우진 G-Skewed 데이터를 사용하여 실험평가를 수행하였다. 그런데 2차원 공간색인에서와는 G-Normal 데이터보다는 G-Skewed 데이터에서 영역질의와 복합질의의 성능이 우수함을 발견하였다. 이러한 이유는 공간상으로는 데이터가 치우쳤다 하더라도 이동하기 때문에 시간축으로는 데이터가 겹치지 않으므로 노드가 겹치는 현상이 줄어든다. 특히 복합질의인 경우에는 영역에 포함된 궤적을 찾고 그 궤적들과 연속된 다른 궤적을 찾아야 하므로 데이터가 몰려 있는 경우에는 질의 수행시 노드의 접근이 상대적으로 작아진다.

## 7. 결론 및 향후 연구

이 논문에서는 3D R-tree기반의 색인에서 궤적을 보존하기 위한 방법을 제안하였다. TA3DR-tree는 영역 질의에서 우수한 성능을 보이는 색인이다. 즉, R-tree 계열의 색인은 영역 질의에서 우수한 성능을 보이지만 궤적을 찾기 위해서는 많은 점 질의를 수행하기 때문에 궤적 질의에서 성능이 떨어지는 결과가 나왔다. 제안한 색인은 궤적 질의에서 TB-tree보다는 성능이 좋지 않지만 R-tree계열 색인에서 월등한 성능을 보인다.

이동체의 궤적 보존을 위해 단말 노드에서는 동일 궤적 그룹화에 따른 분할 정책을 제안하였다. 동일 궤적을 그룹화 함으로써 단말 노드에 저장되는 궤적들을 클러스터링 하였고, 또한 그룹화 되어진 상태에서 R-tree와 유사한 공간 축 분할을 하기 때문에 영역 질의에서도 성능이 우수하다. 공간 활용도를 높이기 위해 시간 축 비균등 분할을 하였다.

비단말 노드에는 동일 이동체 궤적이 서로 다른 단말 노

드에 저장되어지는 정보를 비단말 노드의 자료구조를 변경하여 저장하였다. 비단말 노드의 연결 정보를 추가 함으로써 색인의 크기는 증가하지만 복합 질의 처리할 때 루트 노드에서부터 단말 노드로의 탐색 줄였기 때문에 복합 질의 성능이 향상되었다.

제안한 색인은 몇 가지의 한계점을 보이고 있다. 첫째는 색인의 크기이다. 단말 노드의 연결 노드 정보를 저장하기 위해선 비단말 노드의 각 엔트리 당 노드 팬아웃 만큼의 추가 비트가 필요하기 때문이다. 두 번째는 비단말 노드 정보를 이용해서 연결 노드인 단말 노드를 검색한 결과 찾고자 하는 궤적이 없을 경우도 있다는 것이다. 즉, 연결된 노드라도 사용자가 찾고자 하는 이동체의 궤적이 아닌 다른 이동체의 궤적이 연결되어져 있을 수도 있기 때문이다. 이러한 경우에는 또 다른 연결 노드를 탐색을 해야 하고, 또한 찾고자 하는 궤적이 없을 경우에는 점 질의를 수행해야 하는 한계점을 가지고 있다.

## 참 고 문 헌


[1] Y. Theodoridis, M. Vazirgiannis, and T. K. Sellis, "Spatio-temporal indexing for large multimedia applications," *IEEE Int'l Conf. on Multimedia Computing and Systems*, pp.441-448, 1996.

[2] D. Pfoser, C. S. Jensen, and Y. Theodoridis, "Novel approaches in query processing for moving objects," *Proc. of Int'l Conf. on Very Large Data Bases*, pp.395-406, 2000.

[3] Y. Theodoridis, J. R. O Silva, and M.A Nascimento, "On the generation of spatiotemporal datasets," *Proc. of Int'l Symposium on Spatial Databases*, pp.147-164, 1999.

[4] B. Jun, B. Hong, and B. Yu, "Dynamic splitting policies of the adaptive 3DR-tree for indexing continuously moving objects," *Int'l Conf. on Database and Expert Systems Applications*, September, 2003.

[5] B.-U. Pagel, H.-W. Six, H. Toben, and P. Widmayer, "Towards an analysis of range query performance in spatial data structures," *Proc. of the ACM Symposium on Principles of Database Systems*, pp.214-221, 1993.

[6] D. Pfoser, Y. Theodoridis, and C. S. Jensen, "Indexing trajectories in query processing for moving objects," Chorochronos Technical Report, CH-99-3, October, 1999.

[7] N. Beckmann and H. P. Kriegel, "The R*-tree: An efficient and robust access method for points and rectangles," *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, pp.332-331, 1990.

[8] S. Berchtold, D. A. Keim, and H. P. Kriegel, "The X-tree : An index structure for high-dimensional data," *Proc. of Int'l Conf. on Very Large Data Bases*, pp.28-39, 1996.

[9] A. Guttman, "R-trees: A dynamic index structure for spatial searching," *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, pp.47-54, 1984.

[10] M. Hadjieleftheriou, G. Kollios, V. J. Tsotras, and D. Gunopulos, "Efficient indexing of spatiotemporal objects," *Int'l Conf. on Extending Database Technology*, pp.251-268, 2002.

[11] M. A. Nascimento, J. R. O Silva, and Y. Theodoridis. "Access structure for moving points," TimeCenter Technical Report TR-33, 1998.

[12] M. A. Nascimento, J. R. O. Silva, and Y. Theodoridis, "Evaluation of access structures for discretely moving points," *Proc. of Int'l Workshop on Spatio-Temporal Database Management*, pp.171-188, 1999.

[13] D. Pfoser, "Indexing the trajectories of moving objects," *IEEE Data Engineering Bulletin*, Vol.25, No.2, pp.3-9, 2002.

[14] T. K. Sellis, N. Roussopoulos, and C. Faloutsos, "The R+-Tree: A dynamic index for multi-dimensional objects," *Proc. of Int'l Conf. on Very Large Data Bases*, pp.507-518, 1987.

[15] A. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao, "Modeling and querying moving objects," *Int'l Conf. on Data Engineering*, pp.422-432, 1997.

[16] Y. Theodoridis, T. K. Sellis, A. Papadopoulos, and Y. Manolopoulos, "Specifications for efficient indexing in spatiotemporal databases," *Int'l Conf. on Scientific and Statistical Database Management*, pp.123-132, 1998.



**반 재 훈**

e-mail : chban@kit.ac.kr

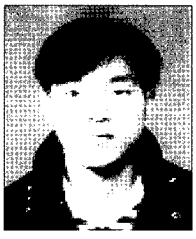
1997년 부산대학교 컴퓨터공학과(공학사)

1999년 부산대학교 컴퓨터공학과(공학석사)

2001년 부산대학교 컴퓨터공학과 박사수료

2002년~현재 경남정보대학 인터넷응용계열 조교수

관심분야 : 데이터베이스, 공간 데이터베이스, 이동체 데이터베이스, RFID


**김 진 곤**

e-mail : jingon11.kim@samsung.com

2002년 부산외국어대학교 컴퓨터공학과 (공학사)

2004년 부산대학교 컴퓨터공학과 (공학석사)

2004년~현재 삼성전자 소프트웨어 랩 무선 4 그룹

관심분야 : 데이터베이스, 공간 데이터베이스, 이동체 데이터베이스

## 전 봉 기

e-mail : bgjun@silla.ac.kr
1991년 부산대학교 컴퓨터공학과(공학사)
1993년 부산대학교 컴퓨터공학과(공학석사)
1993년~1998년 한국통신 연구소 전임연구원
2003년 부산대학교 컴퓨터공학과(공학박사)
2003년~현재 신라대학교 컴퓨터정보공학부 전임강사
관심분야 : 데이터베이스, 공간 데이터베이스, 이동체 데이터베이스

## 홍 봉 희

e-mail : bhhong@pusan.ac.kr
1982년 서울대학교 컴퓨터공학과(학사)
1984년 서울대학교 대학원 컴퓨터공학과(석사)
1988년 서울대학교 대학원 컴퓨터공학과(박사)
1987년~현재 부산대학교 공과대학 컴퓨터공학과 교수
관심분야 : 데이터베이스, 공간 데이타베이스, 이동체 데이터베이스, 이동체 색인, 트랜잭션