RFID 태그 객체의 위치 추적을 위한 색인 구조의 설계 및 구현†

Design and Implementation of Index Structure for Tracing of RFID Tag Objects

김동현*, 이기형**, 홍봉희**, 반재훈***
Dong-Hyun Kim, Gi-Hyoung Lee, Bong-Hee Hong, Chae-Hoon Ban

요 약 RFID 시스템에서 태그의 위치를 추적하기 위해서 궤적은 모델링되고 색인되어야 한다. 궤적은 태그가 판독기의 인식영역으로 들어갈 때와 나갈 때 보고되는 두 개의 시공간 위치를 연결한 선분으로 표현될 수 있다. 만약 태그가 판독기의 인식영역에 들어와 나가지 않으면 시공간 위치는 오직 태그가 인식영역에 들어올 때만 보고된다. 따라서 판독기에 머물고 있는 태그는 궤적을 표현할 수가 없으므로 질의 시 이러한 태그를 검색할 수 없다. 이러한 문제를 해결하기 위하여 이 논문에서는 태그의 궤적을 간격으로 정의하고 새로운 색인인 Interval R-tree를 제안한다. 또한 효율적인 질의처리를 위한 새로운 삽입 및 분할 알고리즘을 제안한다. 마지막으로 제안된 색인을 구현하여 다양한 데이터 집합에서 R-tree와 R*-tree와 성능을 비교하여 우수성을 입증한다.

Abstract For tracing tag locations, the trajectories should be modeled and indexed in a radio frequency identification (RFID) system. The trajectory of a tag is represented as a line that connects two spatiotemporal locations captured when the tag enters and leaves the vicinity of a reader. If a tag enters but does not leave a reader, its trajectory is represented only as a point captured at entry. Because the information that a tag stays in a reader is missing from the trajectory represented only as a point, it is impossible to find the tag that remains in a reader. To solve this problem we propose the data model in which trajectories are defined as intervals and new index scheme called the Interval R-tree. We also propose new insert and split algorithms to enable efficient query processing. We evaluate the performance of the proposed index scheme and compare it with the R-tree and the R*-tree. Our experiments show that the new index scheme outperforms the other two in processing queries of tags on various datasets.

주요어 : RFID 태그 객체, RFID 위치 추적, RFID 시스템, 이동체, 이동체 데이터베이스
KeyWords : RFID tag object, tracing tag object, RFID systems, moving object, moving object database

1. 서 론

RFID(Radio Frequency IDentification)는 무선 주파수를 이용하여 태그(tag)를 장착한 객체를 판독기(reader)가 자동으로 인식하고 확인하는 기술로서 태그를 장착한 객체의 위치 추적이나 객체의 현재의 상태를 감시하는 자동화 생산(automated manufacturing), 재고 관리(inventory tracking), 공급망 관리(supply

† 이 논문은 2004년도 한국학술진흥재단의 지원에 의하여 연구되었음(KRF-2004-002-D00319).
* 동서대학교 인터넷공학부 pusrover@dongseo.ac.kr
** 부산대학교 컴퓨터공학과 {lievebejust, bhhong}@pusan.ac.kr
*** 경남정보대학 인터넷응용계열 chban@kit.ac.kr

chain management) 등과 같은 다양한 응용분야에서 사용된다[1][2].

태그는 이동체와 유사하게 시간에 연속적으로 이동하므로 태그의 궤적을 추적하기 위해서 이동체를 위한 시공간 색인을 적용할 수 있다. 즉, 이동체는 이동하면서 일정한 시간에 위치를 보고하므로 보고된 두 개의 시공간 위치를 연결하는 선분으로 궤적을 표현할 수 있다[3][4][5]. 이와 유사하게 태그가 판독기의 호출지역(interrogation zone)에 들어올 때와 나갈 때 보고되는 시공간 위치를 이용하여 태그의 궤적을 표현할 수 있다.

그러나 호출지역 입출력을 기반으로 궤적을 표현하면 판독기의 인식영역에 들어와 머무는 태그를 찾을 수 없는 문제가 발생한다. 태그의 궤적은 태그가 호출지역에 들어올 때와 나갈 때 보고하는 두 개의 시공간 위치를 연결한 선분으로 표현된다. 만약 태그가 이동하지 않고 판독기의 호출지역 안에 계속 머무는 경우에 태그 객체의 궤적은 태그가 호출지역에 들어갈 때 보고한 시공간 위치로만 구성된다. 따라서 태그가 판독기의 호출지역에 머문다는 정보를 표현할 수 없기 때문에 호출지역 안에서 이동하지 않는 태그를 찾을 수 없다.

이 논문에서는 시공간 환경에서RFID 태그 객체를 검색하는 다양한 질의를 효율적으로 처리하기 위한 데이터 모델과 색인 구조를 제시한다. 태그 객체를 위한 데이터 모델은 현재 위치 질의를 위해서 태그 객체의 현재 위치를 성장 간격(Growing Interval)으로 표현하여 호출지역 안에서 벗어나지 않는 태그에 대한 검색을 지원한다. 또한 태그 객체의 식별자가 검색 조건인 질의를 처리하기 위해서 태그 식별자를 새로운 도메인으로 추가한다. 그리고 색인 구조는 현재 및 과거 다차원 간격 데이터를 저장하는 R-tree 기반 색인 구조를 제안하며, 새로운 단말 노드 분할 정책을 사용하여 기존 분할 정책과 실험을 통해서 비교한다.

이 논문의 구성은 다음과 같다. 먼저2장에서는 관련 연구를 기술하고, 3장에서는 대상환경과 기존 연구의 문제점을 기술한다. 4장에서는 RFID를 사용하여 위치 추적 서비스를 제공하는 응용에서 필요한 질의와 데이터 모델을 기술한다. 5장에서는 RFID 태그 객체의 간격 데이터를 위한 색인 구조를 기술하고 6장에서는 실험을 통하여 기존 색인과의 성능을 비교한다. 마지막으로 7장에서는 결론 및 향후 연구를 기술한다.

2. 관련 연구

2.1 간격 데이터를 위한 색인 구조에 관한 연구

간격(interval)은 [l, r](l≤r)인 형태를 가지며 어떤 속성의 기간(duration)을 나타내는 순서쌍이다. 지금까지 연구된 간격 데이터를 위한 색인 구조는 크게 메인 메모리 거주 색인 구조와 디스크 기반 색인 구조로 구분된다.

메인 메모리 거주 색인 구조는 모두 이진 검색 트리 계열과 유사한 구조를 가지며, Segment Tree[6], Interval Tree[7], Priority Search Tree[8] 등이 있다. 이러한 색인의 데이터 구조는 이진 검색 트리 구조이기 때문에 모든 데이터는 메모리에 거주하게 된다. 따라서 디스크 페이지로 저장할 수 없는 문제점이 있다. 그리고 1차원 간격 데이터를 위한 색인 구조이고 backbone tree를 먼저 생성한 후 데이터를 삽입한다. 그래서 다차원 간격 데이터를 저장할 수 없으며, 동적인 환경에서는 사용할 수 없는 문제점이 있다.

디스크 기반 색인 구조로서 대표적인 색인은 R-tree 계열[9][10]이 있다. R-tree[9]는 데이터 분할 방법의 대표적인 공간 색인으로서 공간 객체를 최소경계사각형(Minimum Bounding Rectangle)으로 표현하며 단말 노드에 모든 데이터를 저장한다. 색인에 데이터를 삽입할 때 최소영역확장정책(Least Area Enlargement Policy)을 사용하며 노드 분할 시 분할되는 2개의 노드가 최소 영역을 갖도록 분할한다. R*-tree[10]는 겹침(overlap)과 가장자리(margin)를 추가적으로 고려한 색인 구조이다. 단말 노드 선택시에 최소겹침확장정책(Least Overlap Enlargement Policy)을 사용하며 노드를 분할할 때 노드의 가장자리가 최소가 되는 분할 축을 선택하고 형제 노드와의 겹침 값이 최소가 되도록 엔트리를 분배한다. 또한 재삽입 정책을 사용하여 R-tree의 공간 활용도가 낮은 문제점을 보완하였다.

SR-tree[11]는 메인 메모리 거주 색인 구조인 Segment Tree를 디스크 기반으로 확장한 색인이다. R-tree와 동일한 구조를 가지며 기존의 삽입 정책과 분할 정책을 그대로 사용한다. 그러나 자식 노드를 완전히 걸치는 긴 간격 데이터가 삽입될 경우 데이터를 단말 노드에 저장하지 않고 걸쳐지는 노드의 부모 노드에 저장한다. SR-tree는 비단말 노드에도 데이터를 저장하므로, 노드의 크기가 루트 노드로 갈수록 커지

는 단점이 있다. 그리고 아주 긴 간격 데이터가 삽입될 경우, 선분을 잘라서 다시 삽입해야 하고, 삽입과 분할로 인하여 노드의 크기가 변할 경우 비단말 노드에 저장되어 있는 간격 데이터를 다른 노드로 이동해야 한다. 태그 객체의 매우 긴 성장 간격은 비단말 노드와 단말 노드에 절단 되어서 저장되기 때문에 SR-tree는 태그 객체를 위한 색인으로 적합하지 않다.

2.2 이동체의 색인 구조에 관한 연구

이동체는 시간에 따라 위치가 변화하기 때문에 시공간 객체로 볼 수 있으며, 시공간 데이터를 저장하는 색인 중에서 3DR-tree[4], 2+3R-tree[12], HR-tree[5]를 이동체를 위한 색인으로 사용할 수 있다.

기존 2차원 공간 데이터를 저장하는 R-tree에 기반한 3DR-tree[4]는 시간을 또 다른 하나의 축으로 고려한다. 이러한 원리를 이용하여, 객체가 시간 [ti,tj) 동안 위치 (xi,yi)에 있고, 시간 [tj,tk) 동안 위치 (xj,yj)에 있다면, 3차원 시공간에서 선분 $\overline{[(x_i,y_i,t_i),(x_i,y_i,t_j))}$와 선분 $\overline{[(x_j,y_j,t_j),(x_j,y_j,t_k))}$으로 표현할 수 있으며, 3차원 R-Tree를 사용하여 색인 할 수 있다. 3DR-tree는 두 끝점이 알려져 있는 선분만 저장할 수 있기 때문에 현재 위치를 저장할 수 없다. 특정 시간과 공간 영역이 주어지는 영역 질의에 높은 성능을 보이지만, 노드 분할 시에 시간 도메인에 대한 고려가 없기 때문에 공간 활용도 크게 떨어지는 단점이 있다.

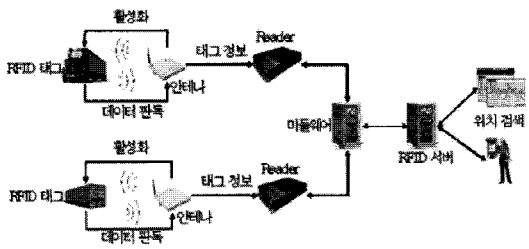
2+3 R-tree[12]는 3D R-tree의 문제점을 2차원 점과 3차원 선분에 대해 각각 따로 저장하는 두 개의 다른 R-tree를 사용함으로써 해결한다. 2차원 점은 현재 시점에서 객체의 공간 상의 점을 표현하며, 3차원 선분은 객체의 과거 이력을 표현한다. 2+3 R-tree에서 객체의 위치에 대한 끝 시간을 아직 모른다면, 그 데이터는 2차원 R-tree에서 객체의 위치에 대한 시작 시간과 객체식별자를 유지하면서 저장된다. 열린(opened) 객체의 현재 상태에서 끝 시간을 알게 될 경우, 2차원 R-tree에서 해당 엔트리를 삭제하고, 3차원 시공간 상의 선분을 구성하여 3차원 R-tree에 삽입한다. 그러나 2+3 R-tree는 두 개의 R-tree에서 질의를 수행해야 하는 문제점이 있다.

HR-tree[5]는 R-tree에 트랜잭션 시간 개념을 추가하여 객체의 이력 정보를 표현하며, 타임스탬프마다 다른 색인 인스턴스를 구성한다. 기본 원리는 원래의 트리는 유지하고 상태가 변한 루트와 가지를 대체함으로써 현재와 과거를 유지한다. 상태가 변화하지 않은 가지들은 복제되지 않는다. HR-tree는 이동체의 이동이 빈번하지 않은 경우에는 효율적이지만, 이동이 많이 발생하는 경우에는 단말 노드와 비단말 노드를 새로 생성해야 하는 문제점을 가진다.

3. 대상환경 및 문제정의

3.1 대상환경

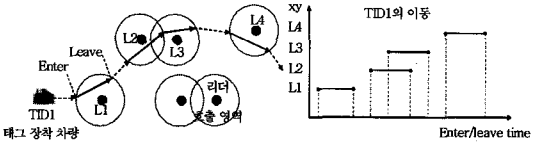
RFID 시스템은 <그림 1>과 같이 태그와 판독기, 그리고 안테나로 구성된다[1]. 태그는 무선 주파수로 판독기와 통신할 수 있는 안테나와 객체 식별자(tid)를 저장하기 위한 메모리로 구성된다. 판독기는 응용 환경에 따라 전략적으로 중요한 지점에 설치되고, 고정되어 있으며, 미리 계산된 위치 좌표(x,y,z)를 가질 수 있다. 또한, 판독기는 태그와 통신할 수 있는 3차원 공간 상의 영역을 가지며 이것을 판독기의 호출 지역[13]이라고 한다.



<그림 1> RFID 시스템 구성도

RFID태그 객체가 호출 지역 안으로 들어가서 처음으로 인식되면 enter 이벤트가 발생하고, 호출 지역 밖으로 나가서 인식되지 않으면 leave이벤트가 발생한다[2]. 이벤트가 발생할 때마다 판독기는 자신의 위치와 태그 객체의 식별자를 서버로 전송한다. 예를 들어서, 식별자 tid를 가지는 태그 객체가 판독기와 연결된 안테나의 호출 지역(loc)으로 이동할 때, 시간 t에서 인식 영역 안으로 들어가면 enter 이벤트가 발생하고 (tid, loc, t_{enter})가 서버로 전송된다. 반대로 객체가 호출 지역을 벗어나면 leave 이벤트가 발생하며 (tid, loc, t_{leave})가 서버로 전송된다. 이때 두 이벤트의 tid와 loc는 동일하지만 t_{leave}는 t_{enter}보다 항상 크다. 이렇게

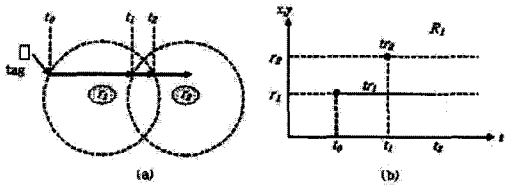
수집된 데이터는 하나의 다차원 색인에 저장되고 검색된다. <그림 2>는 2차원 공간 상에서 RFID 태그 객체가 이동하는 모습과 각 이벤트가 발생하여 생성된 데이터를 보여준다. <그림 2>에서와 같이 태그 객체는 다차원 간격(Interval)으로 표현된다.



<그림 2> RFID 태그 객체의 이동 및 생성된 간격의 예

3.2 문제정의

태그를 위해 기존 연구에서 사용한 궤적의 표현 방법을 사용하면 판독기에 머무는 객체를 찾을 수 없는 문제가 발생한다. 만약 태그가 판독기에 들어와 나가지 않는 경우에는 질의 수행 시에 질의의 결과임에도 불구하고 질의의 후보가 될 수 없다. t_{now}를 현재시간이라고 가정하고 t_{enter}와 t_{leave}를 Enter 이벤트와 Leave 이벤트의 발생시간이라 가정하자. 만약 $t_{enter} \leq t_{now} < t_{leave}$이면 Enter 이벤트만 발생하고 Leave 이벤트는 발생하지 않게 된다. 따라서 궤적은 Enter 이벤트 시에 보고된 시공간 위치인 점으로만 표현되므로 판독기에 머물러 있다는 정보를 표현할 수 없다. 따라서 질의 수행 시 이러한 태그를 찾을 수 없는 문제가 발생한다.



<그림 3> 판독기에 머무는 태그의 문제발생

예를 들어 <그림 3>(a)와 같이 판독기 r_1에 태그가 t_0에 들어와 t_2에 빠져나간다고 가정하자. 궤적은 <그림 3>(b)의 tr_1과 같이 두 개의 시공간 위치를 연결한 선분으로 표현된다. 만약 태그가 판독기 r_2에 들어와 빠져나가지 않으면 태그의 Leave 이벤트는 발생하지 않게 된다. 따라서 궤적은 tr_2와 같이 Enter 이벤트 발생시에 보고된 시공간 위치인 점으로 표현되며 이것이 색인에 삽입되게 된다. <그림 3>(b)의 R_1과 같이 특정 시간에 판독기 r_1과 r_2에 위치하는 태그를 찾는 질의를 수행한다고 가정하자. 질의가 수행되면 R_1에 포함되는 궤적을 찾는데, tr_1은 포함되나 tr_2는 포함되지 않으므로 태그가 판독기 r_2에 머물러 있음에도 불구하고 그 정보가 표현되지 않으므로 tr_2는 질의의 결과에서 제외된다. 따라서 판독기에 들어와 Enter 이벤트만 발생되고 머무는 태그를 표현할 수 있는 방법이 필요하다.

이 논문에서는 이러한 문제를 해결하기 위하여 태그의 궤적을 간격으로 정의한다. 간격은 고정간격과 성장간격으로 구분되어지며 현재 및 과거 위치 질의를 가능하게 한다. 또한 R-tree에 기반한 RFID 태그 객체를 위한 색인 구조를 제안한다. 제안된 색인은 과거 및 현재 위치 질의를 처리하며 새로운 단말 노드 분할 정책을 사용한다.

4. 질의와 데이터 모델

4.1 질의

RFID 시스템에서 필요한 질의는 특정 시공간 지역에 포함되는 궤적을 추출하는데 다음과 같이 네 종류로 분류된다. 4가지 질의 중 FIND 질의와 LOOK 질의가 기본 질의이다. $[a^{\vdash}, a^{\dashv}]$ 는 좌표 축에 투영한 값의 범위를 나타낸다.

Type 1 :
FIND 질의: Q = (tid, $[t^{\vdash}, t^{\dashv}]$) - 시간 $[t^{\vdash}, t^{\dashv}]$에 태그 식별자 tid 가 이동한 판독기의 위치 반환.

Type 2 :
LOOK 질의: Q = ($[x^{\vdash}, x^{\dashv}]$, $[y^{\vdash}, y^{\dashv}]$, $[t^{\vdash}, t^{\dashv}]$) - 시간 $[t^{\vdash}, t^{\dashv}]$에 특정 위치 $[x^{\vdash}, x^{\dashv}]$, $[y^{\vdash}, y^{\dashv}]$에 위치한 판독기를 지나간 태그의 식별자 반환.

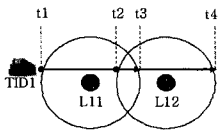
Type 3 :
HISTORY 질의: Q = (tid) - 태그 식별자 tid가 지나간 모든 판독기의 위치 반환.

Type 4:
WITH 질의: Q = (tid, $[t^{\vdash}, t^{\dashv}]$) - 시간 $[t^{\vdash}, t^{\dashv}]$에 태그 식별자 tid와 같이 이동한 모든 태그들의 식별자 반환.

4.2 데이터 모델

리더의 호출 지역에서만 태그 객체의 데이터를 수집할 수 있다. 태그가 머물고 있는 판독기의 현재 위치에 대한 질의를 처리하기 위해서는 각 이벤트가 발생할 때마다 데이터를 생성해야 한다. 그림 4는 태그 식별자 TID1을 가지는 태그 객체의 이동을 이벤트가 발생할 때마다 생성한 예를 보여준다. 태그 객체가 enter와 leave 이벤트를 발생하는 시간 동안에는 호출 지역에 존재했음을 의미하기 때문에 태그 객체를 (tid, loc, t_{enter}, t_{leave})와 같이 표현할 수 있다. 따라서 이 논문에서는 태그 객체를 정의 1과 같이 표현한다.

정의 1: RFID 태그 객체 tag는 시간 간격 [li, ri) 동안 호출 지역 loc에 존재하는 태그 객체이다. 식별자 tid를 가지는 tag의 이력은 (tid, loci, li, ri)의 집합으로 표현된다. 여기서 ri는 li보다 항상 크다.



tid	loc	t_{enter}	t_{leave}	state
TID1	L1	t1	-	Enter
TID1	L2	t2	-	Enter
TID1	L1	-	t3	Leave
TID1	L2	-	t4	Leave

<그림 4> TID1의 이벤트 발생시마다 데이터 생성 예

태그 객체가 호출 지역 안으로 들어가면, enter 이벤트가 발생하고, 이때 l_i 값을 알 수 있다. 그러나 언제 호출 지역 밖으로 나올지 모르기 때문에 r_i 값은 아직 알 수 없다. 현재 위치 질의를 수행하기 위해서는 enter 이벤트가 발생할 시에 r_i 값을 표현할 방법이 필요하게 된다. 아직 값이 결정되지 않은 ri 를 표현하기 위하여 r_i 를 시간 도메인의 가장 큰 값(MAX TIME)으로 표현하고 leave이벤트가 발생할 시에 r_i 값을 갱신한다. 호출 지역에서 enter 이벤트가 발생한 이후로 ri 값은 시간이 지남에 따라 계속해서 증가하기 때문에, 이 논문에서는 leave이벤트가 발생하지 않은 태그 객체를 성장간격(Growing Interval, GI)이라 한다. 반대로 호출 지역을 벗어난 태그 객체는 r_i 값이 결정되어 있고, 값이 변화하지 않기 때문에 고정간격(Fixed Interval, FI)라 한다.

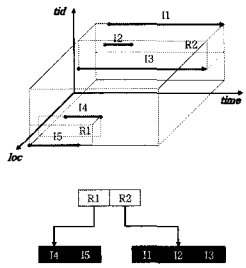
정의 2: 호출 영역 loc에서 enter와 leave 이벤트를 모두 발생시킨 태그 객체 tag의 간격을 고정 간격이라 하며 다음과 같이 표현된다.
FI_{tag} = (tid, loc, l, r)

정의 3: 호출 영역 loc에서 enter 이벤트만 발생시킨 태그 객체 tag의 간격을 성장 간격이라 하며 다음과 같이 표현된다.
GI_{tag} = (tid, loc, l, now)

5. RFID 태그 객체를 위한 색인 구조

5.1 색인 구조 및 검색

다차원 간격 데이터로 정의된 태그 객체를 색인하기 위한 IR-tree(Interval R-tree)는 R-tree 기반 색인 구조로서 간격 데이터를 저장하고 검색하기 위한 색인이다. 단말 노드는 FI와 GI를 동시에 저장하고, 과거 및 현재 위치를 검색할 수 있다. <그림 5>는 IR-tree의 예를 보여준다.



<그림 5> Interval R-tree의 구조

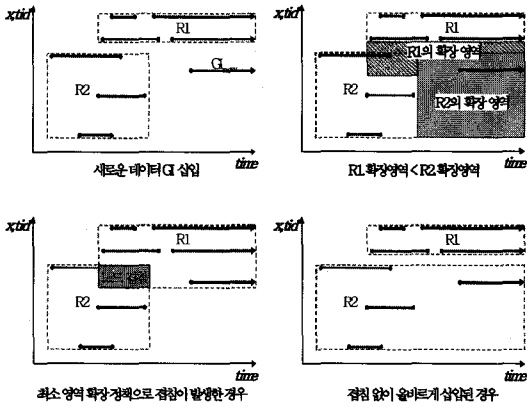
IR-tree의 단말 노드는 (I) 형태의 엔트리로 구성되며 I는 (tid, loc, l, r)로써 태그 객체의 간격 데이터를 나타낸다. 위치 loc는 n차원의 공간 상의 점을 나타낸다. 비단말 노드는 (cp, rect) 형태의 엔트리를 가지며, cp는 자식 노드를 가리키는 포인터이고, rect는 자식 노드의 모든 엔트리를 포함하는 n차원의 공간에서 n+2차원의 최소 경계 박스를 나타낸다. 예를 들어서, 2차원 공간에서는 4차원 최소 경계 박스를 나타내며 (tid_{low}, x_{low}, y_{low}, t_{low}, tid_{high}, x_{high}, y_{high}, t_{high})의 형태를 가진다.

색인을 변경하는 주요 연산은 enter 이벤트가 발생할 때 수행되는 삽입 연산이며, GI를 단말 노드에 삽입한다. 태그 객체가 호출 지역을 벗어나는 leave 이벤트 시에는 이전에 삽입된 GI 데이터를 FI로 갱신하는 연산이 수행된다. 이 때 단말 노드의 엔트리에서 r값이 큰 값에서 작은 값으로 수정되기 때문에 단말 노드에서부터 루트 노드까지 노드의 MBB를 조절해야 한다.

IR-tree에서 검색 알고리즘은 기존 R-tree와 유사하다. 차이점은 비단말 노드의 엔트리 (tid_{low}, x_{low}, y_{low}, t_{low}, tid_{high}, x_{high}, y_{high}, t_{high})에서 t_{high} 값은 호출 지역을 벗어난 시간이기 때문에 질의에는 포함되지 않는다.

5.3 데이터 삽입

IR-tree에서 데이터 삽입은 새로운 GI가 들어갈 단말 노드를 찾는 ChooseSubtree와 삽입 후의 노드 MBB를 조정하고, 분할을 상위 노드로 전파하는 AdjustTree 알고리즘으로 구성된다. R-tree에서 삽입은 최소 영역 확장을 가지는 엔트리를 선택하여 단말 노드를 찾아 내려 간다. 그러나 최소 영역 확장 정책은 간격 데이터에서 노드 겹침을 증가 시킬 수 있는데, 이것은 노드가 아주 긴 GI와 짧은 FI를 동시에 가질 수 있기 때문이다. <그림 6>은 기존 R-tree의 최소 영역 확장 정책을 사용하여 ChooseSubtree를 수행할 시에 노드 겹침이 발생하는 경우를 보여준다. <그림 6>에서 GInew가 삽입될 시에 R1과 R2의 영역 확장이 작은 노드를 선택하면, 노드 겹침이 발생하는 것을 알 수 있다. 노드 겹침은 검색 성능을 저하시키기 때문에 데이터 삽입은 최소 겹침 확장을 가지는 노드를 선택함으로써 수행된다.



<그림 6> 최소 영역 확장 정책으로 인한 노드 겹침

(알고리즘 1)은 GI 데이터를 삽입하는 알고리즘이다. (알고리즘 4)에서 2번째 라인의 InsertDataImpl(p, r.cp, r.rect)는 비단말 노드에 자식 노드의 데이터를 삽입하는 함수로서 (알고리즘 2)와 동일하다.

Algorithm InsertData(GI I_{new})

```
1 Node rootNode := ReadNode(root id);
2 Node node := ChooseSubtree(rootNode, Inew);
3 InsertDataImpl(node, Inew);
```

(알고리즘 1) InsertData

Algorithm InsertDataImpl(Node node, GI I_{new})

```
1 IF node have a room
2  Insert Inew into node
3  IF node is not root node
4      AdjustTree(node);
5 ELSE
6  Invoke SplitNode to obtain L and R
7  IF node is root node
8     Create a new root node whose children are L and R
9     AdjustTree(L, R);
```

(알고리즘 2) InsertDataImpl

Algorithm ChooseSubtree(Node n, GI I_{new})

```
1 IF n is leaf
2  Return n
3 IF entries of n point to non-leaf node
4  Find entry e in n whose rectneeds least area
   enlargement to include Inew;
5 ELSE
6  Find entry e in n whose rectneeds least overlap
   enlargement to include Inew;
7 Node child:= ReadNode(e.cp);
8 ChooseSubtree(child, Inew);
```

(알고리즘 3) ChooseSubtree

Algorithm AdjustTree(Node n)

```
1 Node p := ReadParentNode(n);
2 Find entry e in p points to n
3 IF p.rect do not contains n.rect OR p.rect touches
  e.rect
4  Adjust p.rect so that it tightly encloses all entry
   rectangles in p;
5 IF adjusted AND p is not root node
6  AdjustTree(p);
```

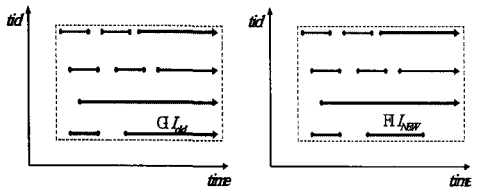
Algorithm AdjustTree(Node l, Node r)

```
1 Node p := ReadParentNode(l);
2 InsertDataImpl(p, r.cp, r.rect);
3 Find entry e in p points to l;
4 IF p.rect do not contains l.rect OR p.rect touches
  e.rect
5  Adjust p.rect so that it tightly encloses all entry
   rectangles in p;
6 IF adjusted AND p is not root node
7  AdjustTree(p);
```

(알고리즘 4) AdjustTree

5.4 데이터 갱신

이전에 삽입된 GI 데이터를 갱신하는 기존의 방법은 GI를 삭제하고, FI를 재삽입하는 정책을 사용하였다. 이 방법은 삭제를 하기 위해 삭제할 데이터가 있는 단말 노드까지 검색하는 것과, 새로운 데이터를 삽입하는 데 추가적인 I/O를 필요로 하기 때문에 색인 구축에 있어서 비효율적이다. 그리고 노드에서 삭제된 GI데이터를 제외한 하나 이상의 다른 GI가 존재하는 경우, 다시 삽입되는 FI가 삭제된 노드로 재삽입될 가능성이 아주 높다. <그림 7>에서 GI 데이터인 I_{old}를 삭제하고, FI 데이터인 I_{new}를 삽입하더라도 노드 MBB가 변하지 않은 것을 볼 수 있다.



<그림 7> GI 데이터를 FI 데이터로 갱신

이전 GI 데이터를 FI로 바꾸는 갱신 알고리즘 UpdateData는 이전 GI를 찾는 FindLeaf와 필요 시에 노드 MBB를 조절하는 AdjustTree 알고리즘으로 구성된다.

Algorithm UpdateData(FI I_{new})

```
1 GI I_old := I_new;
2 I_old.t_high := now;
3 Node l := FindLeaf(I_old);
4 IF l is NULL
5  Return FALSE;
6 ELSE
7  Find entry e in l that contains I_old
8  Update e .I into I_new;
9 IF l is not root node
10  AdjustTree(l);
11 Return TRUE;
```

(알고리즘 5) UpdateData

5.5 노드 분할

RFID 태그 객체는 시간 순서에 따라 삽입이 이루어지고, 과거 데이터에 대한 변화가 거의 없기 때문에 비균등 분할이 공간 활용도를 높이는 방법이다. RFID 태그 객체인 다차원 간격을 객체 식별자와 공간 축으로 투사하면 점이 되지만, 시간 축으로 투사하면 선분이 된다. 그래서 식별자와 공간 도메인을 분할 축으로 선정하면 겹침이 발생하지 않지만, 시간 도메인을 분할 축으로 선정하는 것은 데이터 간의 겹침을 발생시키므로 검색 성능을 저하시키는 요인이 된다. 시간 축에서 분할이 일어나기 위해서는 동일 tid를 가지는 태그 객체가 많아야 한다. 즉, 노드에 존재하는 각 태그 객체가 다수의 enter/leave를 발생시켰을 때에 시간 축에서 분할이 일어나야 노드 겹침 영역이 적어진다. 반대로 각 태그 객체가 소수의 enter/leave를 발생시킬 경우 시간 축이 아닌 다른 축에서 분할이 일어나야 한다.

이 논문에서는 단말 노드 분할 정책으로 분할 축을 임계 값에 따라 선택한다. GI와 FI는 색인의 단말 노드에 동시에 존재할 수 있으므로, 노드의 영역을 크게 만드는 GI보다는 FI를 많이 가짐으로써 이웃 노드와의 겹침을 최소화해야 한다. 호출 지역의 겹침이 없을 경우, 단말 노드 분할 시에 GI의 수는 태그 객체 수와 같거나 작기 때문에 태그 객체 수가 작으면 FI 수가 높아진다. 이러한 원리를 이용하여 단말 노드 분할은 먼저 tid축에서 일어나며, 임의의 p개의 태그 객체를 가질 때까지 tid 축에서 분할이 계속 일어나고, p보다 작은 태그 객체를 가질 경우 다른 축을 분할 축으로 선택한다. 여기서p값은 tid 분할 인자(tid split factor, tsf)에 의해서 아래와 같이 계산된다.

$$p = tsf \times N,\ 0 < tsf < 1$$
$$N = leaf\ \ node\ \ capacity$$

시간 축 비균등 분할을 수행하는 시점은 공간 활용도를 높이기 위해 최소한의 k개의 GI 데이터가 존재할 때 일어난다. 여기서 k의 값은 시간 분할 인자(time distribution factor, tdf)에 의해서 아래와 같이 계산된다.

$$k = tdf \times N,\ tdf = tsf / S$$
$$N = leaf\ \ node\ \ capacity$$
$$S = the\ number\ of\ \ interrogation\ zone's\ \ overlaps$$

IR-tree의 비단말 노드의 분할은 기존의 R*-tree의 방법을 사용하지만 단말 노드의 분할은 비균등 분할에 기반한 새로운 단말 노드 분할 알고리즘을 사용한다. (알고리즘 6)은 단말 노드 l에서 N+1개의 데이터를 두 그룹으로 분할하는 SplitLeafNode 알고리즘을 기술한다.

```
Algorithm SplitLeafNode(GI Inew, Group g1, Group g2)
1 Set tidNum as the number of tid in node l;
2 Set giCount as the number of GI in node l;
3 IF tidNum > p
4  Invoke RFSplitTID(Inew, g1, g2);
5 ELSE IF giCount> k
6  Invoke RFSplitSpatioTemporal(Inew, g1, g2);
7 ELSE /* IF tidNum ≤ p AND giCount ≤ k */
8 Invoke RFSplitTime(Inew, g1, g2);
```

(알고리즘 6) SplitLeafNode

```
Algorithm RFSplitTID(GI Inew, Group g1, Group g2)
1 Make k lists each list includes same tid value;
2 Sort k lists in increasing order of tidvalue;
3 Insert first k/2 lists to g1 and insert the remaining
  lists to g2;
```

(알고리즘 7) RFSplitTID

```
Algorithm RFSplitSpatioTemporal
(GI Inew, Group g1, Group g2)
/*RFSplitSpatioTemporal is similar to R*-tree's split
  algorithm except that tid domain is not considered*
1 Invoke ChooseSplitAixs;
2 Invoke ChooseSplitIndex;
3 Distribute N+1 entries into g1 and g2;
```

(알고리즘 8) RFSplitSpatioTemporal

```
Algorithm RFSplitTime(GI Inew, Group g1, Group g2)
1 FOR EACH entry e IN l and Inew
2  IF e is FI
3   Insert e to g1;
4  ELSE
5   Insert e to g2;
```

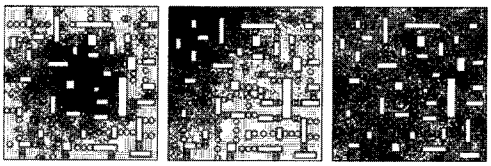
(알고리즘 9) RFSplitTime

6. 성능 평가

이 장에서는 논문에서 제안하는 색인의 성능 평가를 기술한다. 색인의 성능 비교의 기준은 각 색인의 구축 비용과 각 질의의 처리 비용이다. 실험에 사용된 색인은 디스크 기반이기 때문에 각 실험의 비용은 디스크 I/O로 평가된다.

6.1 실험 데이터

이 태그 객체의 위치 데이터를 생성하기 위하여 태그 데이터 생성기(Tag Data Generator, TDG)를 개발하여 사용하였다. 기본적인 알고리즘은 GSTD알고리즘[19]과 유사하지만, 차이점은 위치 보고 시점이 호출 지역 안으로 들어갈 때와 밖으로 나갈 때만 데이터를 생성한다. <그림 8>은 이 논문에서 제시한 색인의 실험을 위해 개발한 태그 객체 위치 생성기이다. <그림 8>에서 원은 리더의 호출 지역을 나타내며 격자 모양이 나타나지 않는 곳은 태그 객체가 이동할 수 없는 지역이다.



a) 가우시안 분포 b) 사향 분포 c) 균등 분포

<그림 8> TDG로 생성된 3가지 데이터 분포

실험에서 사용된 데이터 집합의 종류는 <표 1>과 같다. 그리고 각 데이터 집합에서 색인을 구성하는 도메인의 범위는 <표 2>와 같다.

<표 1> 실험에 사용된 데이터 집합의 종류

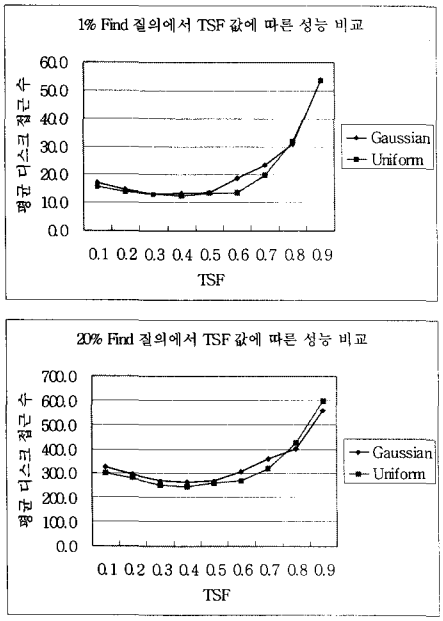
종류	초기 분포	태그 객체 수	Enter 수	Leave 수
G1	Gaussian	100	11,262	11,253
G2	Gaussian	200	22,334	22,295
G3	Gaussian	500	57,152	57,058
G4	Gaussian	1,000	90,709	90,512
R1	Uniform	1,000	83,530	53,348
S1	Skewed	1,000	64,581	64,405

<표 2> 데이터의 색인 도메인 범위

종류	객체 식별자	공간(x, y)	시간
G1	[1,100]	X:[0,1], Y[0,1]	[0,10]
G2	[1,200]	X:[0,1], Y[0,1]	[0,10]
G3	[1,500]	X:[0,1], Y[0,1]	[0,10]
G4	[1,1000]	X:[0,1], Y[0,1]	[0,10]
R1	[1,1000]	X:[0,1], Y[0,1]	[0,10]
S1	[1,1000]	X:[0,1], Y[0,1]	[0,10]

실험은 크게 두 가지 분야에 대하여 수행하였다. 첫 번째는 IR-tree에서 분할 축 선정 기준으로 활용하는 *tsf*값의 변화에 따른 Find 질의와 Look 질의의 성능 실험을 하였다. 그리고 각각의 질의에 대한 적절한 *tsf* 값을 측정하였다. 두 번째는 IR-tree의 단말노드 분할 정책의 성능평가를 위하여 각 데이터 분포 집합에 대하여 IR-tree, R-tree 그리고 R*-tree에서 Find와 Look 질의를 수행하였을 때 각 색인의 성능을 비교하였다.

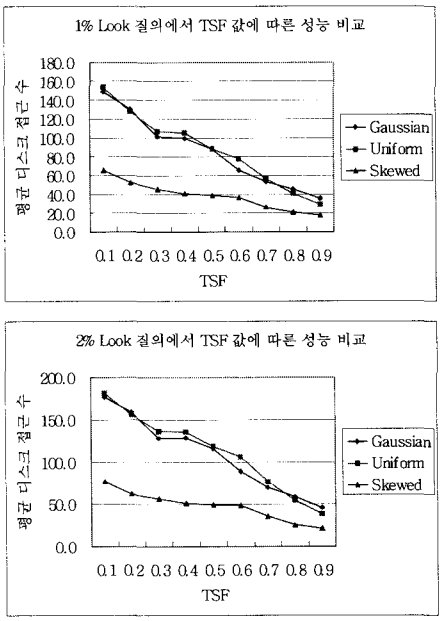
6.2 TSF 값에 따른 성능 비교



<그림 9> 데이터 집합 G4와 R1에서 tsf 값에 따른 Find 질의 성능 비교

IR-tree는 단말 노드 분할 인자로 사용되는 식별자 분할 인자와 시간 분할 인자의 값에 따라 다양한 결과를 보여준다. <그림 9>는 가우시안 분포와 균등 분포에서 *tsf* 값의 변화에 따라 질의의 성능을 보여준다. 1%와 20% Find 질의에서 *tsf* 값이 0.4일 때 가장 좋은 성능을 보였다.

<그림 10>은 1%와 20% Look 질의에서 *tsf* 값에 따른 질의 성능 비교를 보여준다. 두 실험에서 모두 *tsf* 값이 증가할수록 Look 질의의 성능이 좋아졌다. 이것은 *tsf* 값이 증가할수록 식별자 축에서 보다 공간 축에서 분할이 자주 일어나기 때문이다. *tsf* 값에 따라 Find 질의와 Look질의는 서로 반대되는 성능을 보였다. Find 질의는 *tsf* 값이 높을수록 식별자 축 분할 수가 적어지기 때문에 성능이 낮아진다.

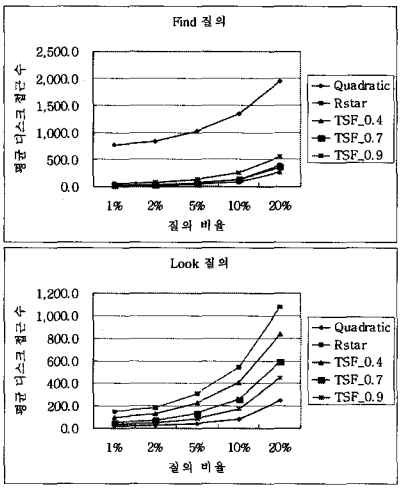


<그림 10> 데이터 집합 G4, R1에서 tsf 값에 따른 Look 질의 성능 비교

6.3 질의 성능 비교

이 논문에서 제안한 IR-tree의 단말 노드 분할 정책의 성능을 측정하기 위해서 기존 R-tree의 quadratic 분할 정책과 R*-tree의 분할 정책을 비교한다. Find와 Look 질의가 동일한 빈도로 사용된다는 가정하에 IR-tree의 *tsf* 값은 0.4, 0.7, 0.9를 사용했다. 사용된 데이터 집합은 가우시안, 균등, 사향 분포를 사용하였고 각 데이터 집합에서 질의 성능을 비교한다. 실험은 과거 및 현재 위치 질의를 동시에 수행하였다.

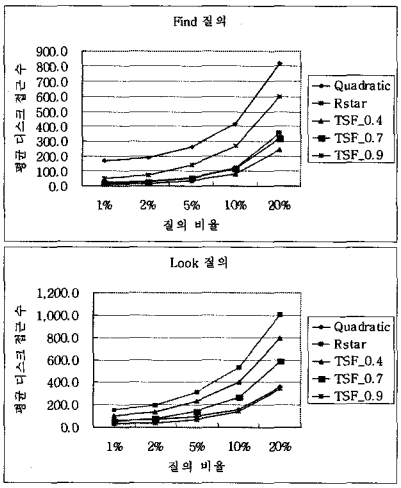
<그림 11>은 Find 질의를 수행한 결과이며 IR-tree의 성능이 가장 우수했고 R-tree가 가장 낮은 성능을 보였다. R*-tree가 IR-tree와 비슷한 성능을 보이는 이유는 분할 축을 선택할 시에 데이터 겹침이 없는 식별자 축을 선택하기 때문이다. 반면에 R-tree의 경우 식별자 축과 시간 축보다 공간 축에서 분할이 자주 일어나기 때문에 Find 질의에서 낮은 성능을 보인다.



<그림 11> 데이터 집합 G4에서 질의 성능 비교

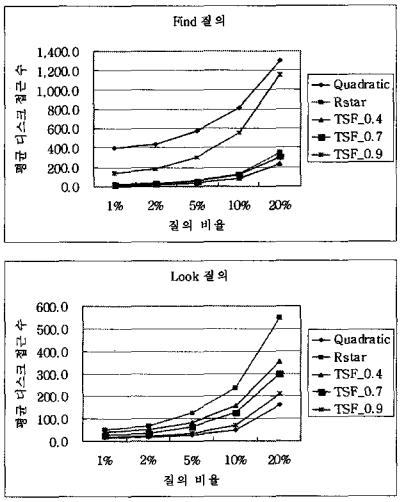
Look 질의에서 IR-tree는 R*-tree보다 성능이 우수하지만 R-tree보다 좋지 않다. 이것은 R-tree가 IR-tree보다 공간 축에서 더 많은 분할이 발생하기 때문이다. 두 질의에서 R-tree와 R*-tree가 서로 반대의 결과를 보여주지만 두 질의가 동일한 빈도로 들어올 경우에는 IR-tree의 성능이 가장 우수하다.

<그림 12>에서 보듯이 균등 분포의 데이터 집합에서 실험한 결과는 가우시안 분포와 비슷하다. Find와 Look 질의를 동시에 수행했을 때, 가우시안 분포에서는 R*-tree가 R-tree보다 항상 좋았지만 균등 분포에서는 질의 비율이 증가할수록 낮은 성능을 보였다.

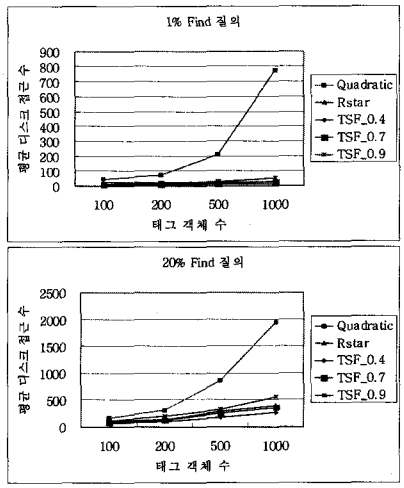


<그림 12> 데이터 집합 R1에서 질의 성능 비교

<그림 13>은 사향 분포에서 Find, Look 질의 결과를 보여주며, 가우시안 분포와 비슷한 성능을 보임을 알 수 있다. 3가지 데이터 분포에서 Find 질의는 IR-tree가 가장 우수한 성능을 보였으며, Look 질의에서는 R-tree가 가장 좋았다.



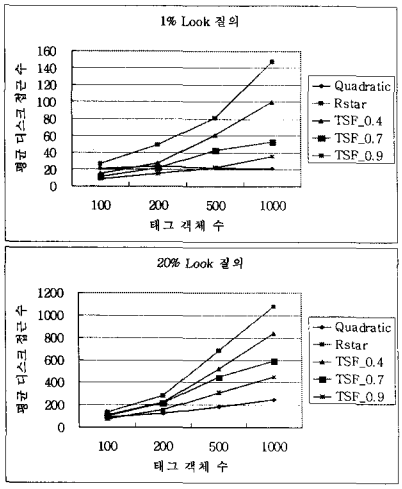
<그림 13> 데이터 집합 S1에서 질의 성능 비교



<그림 14> 태그 객체 수에 따른 Find 질의 성능 비교

태그 객체의 수가 많으면 리더의 수가 정해져 있기 때문에 하나의 호출 지역 안에 존재하는 태그 객체 수가 증가한다. 따라서 동일한 위치를 가지는 태그 객체 수가 많이 존재하게 된다. <그림 14>는 태그 객체 수

에 따른 Find 질의의 성능을 보여준다. R-tree의 경우 태그 객체 수가 증가할 수록 다른 색인에 비해 급격하게 성능이 떨어짐을 알 수 있다. <그림 15>는 태그 객체 수에 따른 Look 질의의 성능을 보여 주며 Find 질의와 반대의 결과를 보였다.



<그림 15> 태그 객체 수에 따른 Look 질의 성능 비교

7. 결론 및 향후 연구

이 논문에서는 RFID 태그를 장착한 객체의 위치를 추적하기 위해서 태그 객체의 데이터 모델과 색인 구조를 제시하였다. 태그 객체의 이동을 이산적 모델에 적용했을 경우, Find 질의와 Trace 질의를 처리할 수 없는 점과 현재 시간과 관련된 질의를 처리할 수 없는 문제점이 있었다. Find 질의와 Trace 질의를 처리하기 위해서, 태그 객체의 식별자가 색인에 사용될 수 있도록 색인 도메인에 포함시킴으로써 문제를 해결하였다. 그리고 현재 시간과 관련된 질의를 처리하기 위해서 현재 위치를 성장 간격(GI)으로 표현하고 과거 위치를 고정 간격(FI)으로 표현하였다.

태그 객체의 위치 추적을 위한 색인 구조로는 R-tree기반의 색인 구조를 변경한 IR-tree를 제안하였으며, Find 질의와 Look 질의를 효율적으로 처리하기 위해서 새로운 단말 노드 분할 정책을 제안하였다. 기존 R-tree의 분할 정책은 Look 질의에 유리하며 R*-tree의 분할 정책은 Find 질의에 유리함을 실험을 통해서 알 수 있었다. 태그 객체의 식별자와 위치는 서로 관련성이 없기 때문에 두 태그 객체의 근접성을 고려하기가 어렵다. 이 논문에서는 색인의 노드에서 데이터를 클러스터링하는 방법으로 최소 태그 객체 수에 이를 때까지 식별자 축에서 분할하고 그렇지 않을 경우 공간 축에서 분할하는 방법을 사용하였다. 그리고 노드에 FI 데이터 수가 임의의 GI 데이터 수보다 많을 경우 시간 축 비균등 분할을 수행함으로써 공간 활용도를 높였다.

실험을 위하여 태그 객체의 위치 데이터 생성기를 GSTD 알고리즘에 기반하여 설계 및 구현하였으며, 생성된 데이터를 사용하여 성능 평가를 수행하였다. 이 논문에서 제안하는 IR-tree가 기존 색인들보다 Find 질의와 Trace 질의에서 우수한 질의 성능을 보였으나, Look 질의에서는 R-tree보다 좋지 못한 성능을 보였다. 이것은 R-tree가 공간 축에서 분할이 많이 일어나기 때문이고, Find 질의의 성능이 극히 나빠지는 원인이었다.

향후 연구로서 Find 질의와 Look 질의의 성능을 동시에 향상시키는 데이터 삽입 정책과 분할 정책에 대한 연구가 필요하다. 그리고 긴 시간 간격을 가지는 데이터는 심한 노드 겹침을 일으키고 질의의 성능을 저하시키므로 효율적으로 저장할 수 있는 방법이 필요하다.

참고문헌

[1] S. E. Sarma, S. A. Weis, and D. W. Engels, "RFID Systems and Security and Privacy Implications ," Springer-Verlag, 2002, pp454-469.

[2] K. Romer, T. Schoch, "Infrastructure Concepts for Tag-Based Ubiquitous Computing Applications ," Workshop on Concepts and Models for Ubiquitous Computing, 2002.

[3] D. Pfoser, C. S. Jensen, and Y. Theodoridis, "Novel Approaches to the Indexing of Moving Objects ," Proc. of the 26th VLDB Conf., 2000, pp395-406.

[4] Y. Theodoridis, M. Vassilakopoulos, and T. Sellis, "Spatio-temporal indexing for large multimedia applications ," Proc. of the 3rd IEEE

Conf. on Multimedia Computing and Systems, 1996, pp441-448.

[5] M. A. Nascimento and J.R.O. Silva, "Towards historical R-Trees ," Proc. of the 1998 ACM Symposium on applied Computing, 1998, pp235-240.

[6] J. L. Bently, "Algorithms for Klee's Rectangle Problems ," Computer Science Department, Carnegie-Mellon University, 1977.

[7] H. Edelsbrunner, "A New Approach to Rectangle Intersections ," Int. Journal of Computer Mathematics, 1983, pp.209-229.

[8] E. M. McCreight, "Priority Search Trees ," SIAM Journal of Computing, 14(2), 1985, pp257-276.

[9] A. Guttman, "R-Tree: A dynamic index structure for spatial searching ," Proc. of the 1984 ACM SIGMOD on Management of Data, 1984, pp47-57.

[10] N. Beckmann and H. P. Kriegel, "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles ," Proc. of ACM SIGMOD, 1990, pp332-331.

[11] C. Kolovson and M. Stonebraker, "Segment Indexes: Dynamic Indexing Techniques for Multi-Dimensional Interval Data ," Proc. of ACM SIGMOD, 1991, pp138-147.

[12] M. A. Nascimento, J.R.O. Silva and Y, Theodoridis, "Evaluation of Access Structures for Discretely Moving Points ," Proc. of the Intl. Workshop on Spatiotemporal Database Management, 1999, pp171-188.

[13] W. E. Daniel and E, S, Sanjay, "The Reader Collision Problem ," IEEE, 2002.

[14] A. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao, "Modeling and Querying Moving Objects ," 13th Int. Conf. on Data Engineering, 1997.

[15] O. Wolfson, S. Chamberlain, and L. Jiang, "Moving Objects Databases: Issues and Solutions ," Conf. Statistical and Scientific Database Management, 1998.

[16] M. Erwig, R. H. Guting, M. Schneider, and M. Vazirgiannis, "Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases ," GeoInformatica, 3, 1999, pp265-291.

[17] M. Erwig, R. H. G?ting, M. Schneider, and M. Vazirgiannis ,"Abstract and Discrete Modeling of Spatio-Temporal Data Types ," 6th ACM Symposium on Geographic Information Systems , 1998, pp131-136.

[18] J. Waldrop, D. W. Engels, and S. E. Sarma, "An Anticollision Algorithm for The Reader Collision Problem ," IEEE International Conference on Communication, 2003.

[19] Y. Theodoridis, J. R. O Silva, and M.A Nascimento, "On the generation of spatiotemporal datasets ," Proc. of Int. Symposium on Spatial Databases, 1999, pp147-164.

김동현

1995년 부산대학교 컴퓨터공학과 졸업 (학사)

1997년 부산대학교 대학원 컴퓨터공학과 (석사)

2003년 부산대학교 대학원 컴퓨터공학과(박사)

2004년 ~현재 동서대학교 소프트웨어전문대학원 전임강사

관심분야 : 이동체 색인, 모바일 GIS, 지리정보시스템, 이동체 데이타베이스, 공간 데이터베이스

이기형

2003년 부산대학교 컴퓨터공학과 (공학사)

2005년 부산대학교 컴퓨터공학과 (공학석사)

관심분야 : 이동체 색인, 모바일 GIS, 지리정보시스템, 이동체 데이타베이스, 공간 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)

반재훈

1997년 부산대학교 컴퓨터공학과 (공학사)
1999년 부산대학교 컴퓨터공학과 (공학석사)
2001년 부산대학교 컴퓨터공학과 박사수료
2002년 ~ 현재 경남정보대학 인터넷응용계열 조교수
관심분야 : 개방형 GIS, 이동객체, 객체지향데이터베이스