RFID/LBS 환경에서 태그 이동 객체의 위치 추적을 위한 색인 구조

안준환° 임덕성 안성우 홍봉희
부산대학교 컴퓨터공학과
{ynoser°, dsleem, swan, bhhong}@pusan.ac.kr

Index Structure for Tracing of Tag Moving Objects in RFID/LBS environment

Junhwan An°, Duksung Lim, Sungwoo Ahn, Bonghee Hong
Department of Computer Engineering, Pusan National University

요 약

RFID를 이용하는 물류 시스템에서 태그 객체의 위치 추적을 위해서는 태그 객체의 궤적을 모델링하고 효율적으로 검색하기 위한 색인 구성이 필수적이다. 태그 객체의 궤적은 리더의 인식 영역에 들어오고 나가는 두 점을 연결한 시공간 선분으로 표현할 수 있다. 그러나 태그 객체가 리더의 인식영역 밖으로 벗어나게 되면 태그 객체의 궤적을 표현할 수 없으므로 위치 추적이 불가능하게 되는 문제를 가진다. 이러한 문제를 해결하기 위하여 리더의 비 인식 영역에서 GPS를 이용한 태그 객체의 위치 추적을 병행할 필요가 있다.

본 논문에서는 RFID 시스템과 LBS 시스템을 연동한 환경에서 태그 궤적을 표현하기 위한 위치 추적 시스템의 모델을 제시하고, 제시된 모델에서 태그 객체의 위치 추적을 효율적으로 처리하기 위한 색인 구조를 제안한다. 제안된 색인 구조는 태그 객체의 현재 위치뿐만 아니라 과거 궤적을 효율적으로 처리하기 위한 새로운 삽입 및 분할 알고리즘을 제안하여 노드가 차지하는 영역을 최소화한다.

1. 서 론

RFID(Radio Frequency Identification)는 태그를 부착한 객체를 리더에서 무선으로 인식하는 기술로써 태그를 장착한 객체의 위치 추적이나 객체의 현재 상태를 감시하는 자동화생산, 재고관리, 공급망관리 등과 같은 다양한 응용분야에서 사용된다[1].

RFID 시스템을 이용한 위치추적은 리더의 인식능력의 한계로 인해 태그 객체가 리더의 인식영역 밖으로 벗어나게 되면 위치 추적이 불가능한 문제점을 가지고 있다. 따라서 비 인식 영역에서 LBS 시스템을 이용한 태그 객체의 연속적인 위치추적이 필요하다.

RFID/LBS 통합 환경에서의 태그 객체의 궤적은 단절된 형태의 간격 데이터와 연속된 선분의 집합으로 나타난다. 단절된 간격 데이터의 궤적 검색을 위해서는 간격들의 순차적인 검색이 필요하기 때문에 질의 비용이 큰 문제점이 있으며 최근 보고시점과 질의를 수행하는 현재시점 사이의 질의를 처리하는데 드는 비용이 큰 문제가 있다.

본 논문에서는 RFID 시스템과 LBS 시스템을 연동한 환경에서 효율적인 태그 객체의 위치 추적을 위해 위치 데이터의 종류를 분석하고 이를 기반으로 궤적 데이터 모델과 색인 구조를 제시한다. 데이터 모델은 태그 객체의 상태에 따라 동적인 상태와 정적인 상태를 나누어 표현하고 색인 구조에서는 단절된 간격 데이터 간의 효율적인 궤적 검색을 위해 링크 기법을 적용한다.

논문의 구성은 다음과 같다. 2장에서는 관련 연구를 소개하며 3장에서는 대상 환경과 문제를 정의한다. 4장에서는 데이터 모델과 질의를 제안하고, 5장에서는 태그 객체를 위한 색인 구조 및 삽입정책, 분할정책, 검색기법에 대해서 기술한다. 마지막으로 6장에서는 결론 및 향후 연구를 기술한다.

2. 관련연구

RFID 시스템에서 태그 객체의 위치 추적을 위한 색인으로는 TPIR-Tree[2]가 있으며, LBS 시스템에서 이동체의 위치추적을 위한 대표적인 시공간 색인으로는 3DR-tree[3]가 있다.

TPIR-Tree는 태그 객체가 리더의 인식영역 안에 있을 경우 검색이 불가능한 문제를 해결하기 위해 동적인 간격 데이터를 이용하여 태그 객체의 현재 위치 및 궤적 질의를 효율적으로 처리하기 위한 색인구조이다. 그러나 간격데이터들이 서로 단절되어 있어 궤적 검색 시 연결된 궤적들을 순서 따라 검색하는 비용이 높은 단점을 가진다.

이동 객체의 궤적을 시간과 공간을 별도의 색인에 분리해서 저장하지 않고 시간과 공간을 하나의 3차원 공간에 저장하는 3DR-tree는 시공간 영역 질의 성능이 우수하지만 현재 위치의 검색 및 궤적 검색 비용이 높은 단점을 가진다.

3. 대상환경 및 문제정의

3.1 대상환경

RFID 시스템은 그림 1과 같이 태그(tag)와 리더(reader), 그리고 서버로 구성된다. 태그는 물류환경에서의 컨테이너와 같은 물품에 장착되고, 리더는 응용 환경에 따라 전략적으로 중요한 지점에 설치된다. 설치된 리더는 리더의 인식영역에 들어오거나 나가는 태그의 정보를 수집한다. 이때 태그가 인식영역에 들어오게 되면 리더는 태그에 대해 Enter 이벤트를 발생하여 서버에 전송하게 되고, 인식영역을 벗어날 때에는 Leave 이벤트를 발생하여 서버에 전송하게 된다[2]. 사용자는 서버에 전송된 이벤트 데이터들을 이용해 각 태그 객체의 위치정보를 검색할 수 있게 된다.

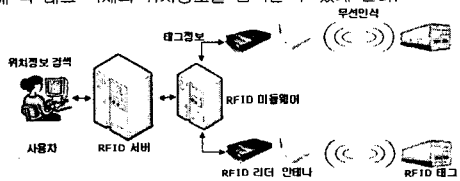

그림 1 RFID 시스템의 구성

3.2 문제정의

RFID 시스템의 주요 질의에는 "10번 컨테이너가 13시에 있었던 곳은?"과 같이 태그 식별자와 시간이 주어질 때 위치를 찾는 Temporal Tag Query와 "11번 톨게이트를 13시에 지나간 컨테이너는?"과 같이 특정 위치와 시간을 주고 태그의 식별자를 찾는 Temporal Reader Query가 있다. 그러나 리더의 인식능력의 한계로 인해 태그 객체가 인식영역을 벗어난 구간에 대한 질의가 주어질 경우 태그의 위치를 알 수 없는 문제가 발생한다. 따라서 응용 환경의 필요에 따라 리더의 인식영역 밖에서는 기존의 LBS 시스템을 이용해 태그 객체의 위치추적을 병행할 필요가 있다.

RFID/LBS 통합 시스템에서는 태그 객체의 위치를 각 구간에서 발생한 이벤트를 이용해서 인식한다. 각 구간에서 발생하는 이벤트들은 다음과 같이 정의된다.

[정의 1] 태그 객체가 리더의 인식영역으로 들어올 때 발생하는 이벤트를 Enter라고 한다.
[정의 2] 태그 객체가 리더의 인식영역 밖으로 나갈 때 발생하는 이벤트를 Leave라고 한다.
[정의 3] 태그 객체가 리더의 인식영역 밖에서 이동 중 방향이나 속도가 변경될 때마다 발생하는 이벤트를 Moving이라고 한다.

이런 이벤트들을 이용한 태그 객체의 궤적은 그림 2와 같이 RFID구간과 LBS구간에서 서로 다르게 표현된다. RFID구간의 궤적은 Enter 이벤트가 발생한 지점과 Leave 이벤트가 발생한 지점을 연결한 간격 데이터로 표현된다. LBS구간의 궤적은 Moving 이벤트가 발생한 지점들을 연결한 연속적인 선분의 집합으로 표현된다.

이와 같은 위치 데이터에 대한 태그 객체 추적 시 다음과 같은 문제점을 가진다. 첫째, RFID구간의 간격 데이터는 Enter 이벤트 발생 후 Leave 이벤트 발생 전까지의 데이터는 점(point)으로 표현되어 Enter 이벤트 발생 시점부터 현재(now) 사이의 질의 검색 비용이 높다. 둘째, RFID 시스템 구간의 간격 데이터는 서로 단절되어 있어 궤적 검색 시 이전 궤적의 검색비

이 논문은 교육인적자원부 지방연구중심대학육성사업(차세대물류IT기술연구사업단)의 지원에 의하여 연구되었음.

용이 높아진다. 셋째, LBS 시스템 구간에서 최근 보고 시점과 현재 사이의 태그 객체의 위치에 대한 질의 처리 시 높은 검색 비용을 가진다.

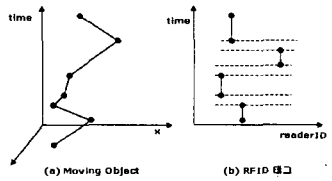

그림 2 이동체와 태그의 위치 표현 방법

본 논문에서는 이러한 문제점들을 해결하기 위해서 RFID/LBS 통합 환경에서 태그 객체의 데이터 모델을 정의하고 새로운 색인 구조를 제시한다. 그리고 태그 객체와 관련된 질의를 효율적으로 처리하기 위해서 새로운 삽입 정책과 분할 정책 그리고 검색 기법을 제시한다.

4. 데이터 모델

4.1 RFID/LBS 통합 환경에서의 질의 종류

표 1은 RFID/LBS 통합 시스템에서 사용되는 기본 질의 종류를 보여준다. Temporal Tag Query는 태그 객체 식별자(tid)와 시간(time)이 주어질 때 태그 객체의 위치(rid or x-y)를 찾는 질의이다. Temporal Reader Query는 리더 식별자(rid)와 시간이 주어질 때 주어진 리더의 인식 영역에 있는 태그 객체 식별자를 찾는 질의이며 Temporal Location Query는 공간 좌표(x-y)와 시간이 주어질 때 주어진 좌표상에 위치하는 태그 객체 식별자를 찾는 질의이다.

표 1 기본 질의 종류

질의 종류	입 력	결 과		질의 예
		RFID 구간	LBS 구간	
Temporal Tag Query	tid, time	rid	x-y	tid2가 2시에 있었던 위치는?
Temporal Reader Query	rid, time	tid set	-	2시에 rid에 있었던 tid는?
Temporal Location Query	x-y range, time	-	tid set	2시에 x-y에 있었던 tid는?

Temporal Reader Query와 Temporal Location Query는 질의에 따른 결과가 {tid}로 명확하다. 하지만, Temporal Tag Query는 그 결과가 태그 객체의 해당 시점의 위치에 따라 달라진다. 만약 태그 객체가 RFID구간 즉, 리더의 인식영역 내부에 있다면 rid가 결과가 되지만 태그 객체가 LBS구간 즉, 리더의 인식영역 외부에서 이동 중이라면, 질의의 결과는 x-y 좌표가 될 것이다. 하지만, 질의 결과는 tuple의 데이터에 표현 되므로 색인에 나타날 필요는 없다.

기본 질의 외에도 자주 사용되는 질의로 궤적 검색 질의가 있다. 궤적 검색 질의를 기본질의로 수행하기 위해서는 Temporal Tag Query를 전체 시간과 전체 공간으로 수행하고 그 결과를 시간순으로 정렬해야 하므로 그 비용이 너무 높은 문제점이 있다.

4.2 RFID/LBS 통합 환경에서의 데이터 모델

RFID/LBS 통합 시스템에서 사용되는 기본 질의인 Temporal Tag Query, Temporal Reader Query, Temporal Location Query를 처리하기 위해서 태그 객체의 궤적은 다음과 같이 정의된다.

[정의 4] RFID 시스템 구간에서 태그 객체의 궤적은 간격(interval) 데이터의 집합으로 정의한다.

[정의 5] LBS 시스템 구간에서 태그 객체의 궤적은 연속적인 선분(line segment)의 집합으로 정의한다.

이러한 데이터는 객체 식별자(tid), 리더 식별자(rid), 공간 좌표 (x, y), 시간(time)으로 구성되고 하나의 다차원 색인에 저장된다. 이때 리더 식별자와 공간좌표는 두 도메인 모두 특정 위치를 뜻하므로 의미상으로는 동일하다고 할 수 있다.

R-tree[4] 기반 색인은 차원이 올라갈수록 성능이 급격히 저하되는 문제가 있으므로[5] 공간좌표와 의미상으로 동일한 리더 식별자를 공간좌표에 매핑(mapping) 하여 한 개 차원을 줄여 성능을 향상시킨다. 단, 리더의 실제 좌표를 공간좌표에 매핑(mapping)함을 가정한다.

RFID 구간의 간격 데이터는 다시 다음과 같이 분류되어 정의될 수 있다.

[정의 6] 태그 객체가 리더 인식영역에 들어왔다가 다시 나간 상태의 궤적을 SI(Static Interval)이라고 정의한다.

[정의 7] 태그 객체가 리더 인식영역에 들어와 머물러 있는 상태의 궤적을 DI(Dynamic Interval)이라고 정의한다.

각 간격 데이터는 색인에서 <tid, x, y, prev_link, next_link, t_{enter}, t_{leave}>로 표현되며 시간간격 [t_{enter}, t_{leave}] 동안 tid를 태그 식별자로 하는 태그 객체가 공간좌표 (x, y) 의 위치에 있는 리더 인식 영역에 존재했음을 의미한다. 이때 DI는 t_{leave} 가 아직 정해지지 않았으므로 t_{now}로 나타낼 수 있으며 이는 현재까지 리더의 인식영역 안에 존재함을 나타낸다. 그리고 prev_link와 next_link는 단절되어있는 직전, 직후 간격 데이터들과의 연결 정보이며 이는 효율적인 궤적 검색 질의를 위해 사용된다.

LBS 구간의 선분은 다시 다음과 같이 분류되어 정의될 수 있다.

[정의 8] 태그 객체가 이동 중 속도 또는 방향이 변경되어 위치를 보고한 두 개의 지점을 잇는 궤적을 SL(Static Line segment)이라고 정의한다.

[정의 9] 최근의 보고지점에서 태그 객체의 속도, 방향으로 현재 위치를 예측해 연결한 궤적을 DL(Dynamic Line segment)이라고 정의한다.

SL은 색인에서 <tid, x1, x2, y1, y2, t1, t2>로 표현되며 tid를 태그 식별자로 하는 태그 객체가 공간좌표 (x1, y1)에서 또 다른 공간좌표 (x2, y2)로 시간간격 [t1, t2] 동안 이동했음을 의미한다. DL은 색인에서 <tid, x, y, velocity, direction, t>로 표현되며 tid를 태그 식별자로 하는 태그 객체가 공간좌표 (x, y)의 위치에서 direction 방향을 향해 velocity의 속도로 이동 중임을 의미한다. 이때 direction과 velocity를 이용해 질의 시점의 현재 위치를 유추해 낸다.

DI와 DL는 데이터가 갱신될 때까지 시간이 지남에 따라 계속해서 성장하게 된다.

SI, DI, SL, DL은 각각 Tag ID, X, Y, Time 4차원 공간상에 그림 3과 같이 나타난다.

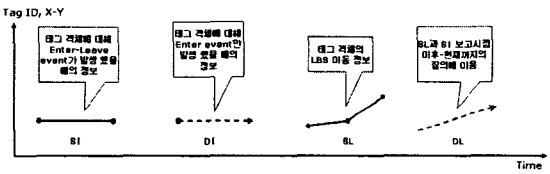

그림 3 이벤트별 궤적 정보 모델링

5. RFID/LBS 통합 환경에서 태그 객체를 위한 색인

5.1 색인구조

본 논문에서 제안하는 색인 구조의 단말 노드는 <state, tuple-identifier, MBB> 형태의 엔트리를 가진다. tuple-identifier는 실제 tuple 데이터의 위치를 의미한다. 단말 노드 엔트리의 MBB(Minimum Bounding Box)구조는 state에 의해 결정되며 state는 SI, DI, SL, DL 중 하나로 설정된다. 만약 state가 SI또는 DI이면 MBB의 구조는 <tid, x, y, prev_link, next_link, t_{enter}, t_{leave}>형태를 가진다. 이때 state가 DI이라면 t_{leave}는 t_{now}가 된다. 만약 state가 SL이라면 MBB의 구조는 <tid, x1, x2, y1, y2, t1, t2>로 표현된다. 그리고 state가 DL이라면 MBB의 구조는 <tid, x , y, velocity, direction, t>형태를 가진다.

비 단말 노드는 <state, MBB, child-pointer>형태의 엔트리를 가진다. child-pointer는 트리에서 자식 노드를 가리킨다. state는 해당 엔트리의 단말 노드에 DI 또는 DL의 state를 가지는 엔트리가 있는지 확인하는데 사용되며 만약 존재한다면 state는 D(Dynamic entry)가 되고, 존재하지 않는다면 state는 S(Static entry)가 된다. MBB는 <[$tid^{\downarrow}$,$tid^{\uparrow}$], [$x^{\downarrow}$,$x^{\uparrow}$], [$y^{\downarrow}$,$y^{\uparrow}$], [$t^{\downarrow}$,$t^{\uparrow}$], $Vx^{\downarrow}$, $Vx^{\uparrow}$, $Vy^{\downarrow}$, $Vy^{\uparrow}$>의 구조를 가진다. tid, x, y, t는 각각 MBB의 크기 및 위치를 나타내는데 사용되고, $Vx^{\downarrow}$, $Vx^{\uparrow}$, $Vy^{\downarrow}$, $Vy^{\uparrow}$는 해당 엔트리의 단말에 존재하는 DL들의 velocity와 direction을 종합했을 때 MBB의 확장 속도를 의미한다.

그림 4는 위의 색인 구조를 이용한 색인 구성의 예를 보여 준다.

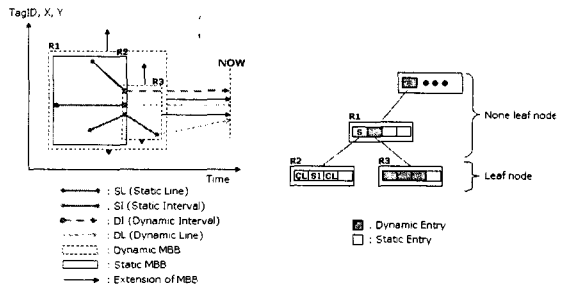

그림 4 색인구성의 예

5.2 삽입 정책

태그 객체가 이동 중 Enter 이벤트가 발생하면 색인에 DI가 삽입된다. 이 때 해당 태그 객체의 직전 이벤트의 종류가 Moving라면 직전 이벤트 지점과 Enter 이벤트 지점을 잇는 SL의 삽입이 선행 되어야한다. 삽입된 DI는 해당 태그 객체의 직전 궤적과 연결 정보를 유지하게 된다.

태그 객체가 리더의 인식영역 밖으로 벗어나 Leave 이벤트가 발생하면 색인에 이전에 삽입되어있던 DI를 SI로 업데이트 시키고 DL을 삽입해 업데이트 된 SI의 현재 질의를 가능하게 한다.

태그 객체가 리더의 비 인식영역에서 이동 중 Moving 이벤트가 발생하면 이전에 현재질의 처리를 위해 삽입되어있던 DL을 SL으로 업데이트 시키고 DL을 삽입하여 Moving 이벤트 발생시점 이후의 현재질의를 가능하게 한다.

SI, DI, SL, DL의 삽입은 기존 R-tree와 유사한 기법으로 진행되며 그 알고리즘은 표 2와 같다. 이때 서브 알고리즘인 UpdateConnection은 SI와 DI의 궤적 연결정보를 유지한다.

그 외에 삭제 알고리즘은 기존 R-tree와 동일하며, 업데이트 알고리즘은 삭제와 삽입을 순서대로 하여 수행한다.

표 2 Algorithm Insert

```
Algorithm Insert(Entry entry)
1   Set rootNode to root node
2   Node node := ChooseSubtree(rootNode, entry)
3   Insert entry into node
4   IF node overflow
5     Invoke SplitNode to obtain L and R
6     IF node is root node
7       Create a new root node whose children are L and R
8       AdjustTree(L, R)
9   ELSE
10    IF node is not root node
11      AdjustTree(node)
12  UpdateConnection(node, entry)
```

5.3 분할 정책

분할 알고리즘은 표 3과 같으며 그 정책은 엔트리의 구성에 따라 크게 3가지로 분류된다. Static엔트리만으로 구성되어 검색 시에도 MBB의 확장이 없는 노드는 기존 R*-tree[6]와 동일한 분할 정책을 사용한다. Dynamic 엔트리만으로 구성되어 검색 시 하위노드의 모든 MBB의 확장이 일어나는 노드의 분할 정책은 time을 제외하고 tid, x, y공간으로만 R*-tree와 동일한 분할 정책을 적용한다. 마지막으로 Dynamic 엔트리와 Static 엔트리가 혼재되어있는 노드는 Dynamic 엔트리를 MBB 내부에서 최대한 확장한 뒤에 R*-tree의 분할 정책을 적용한다.

표 3 Algorithm SplitNode

```
Algorithm SplitNode(Node Nold, Node Nnew1, Node Nnew2)
1   IF Nold has Dynamic entry
2     IF Nold has Static entry
3       Extend MBB of Nold's entries
4       SplitNodeImpl(Nold, Nnew1, Nnew2, null)
5     ELSE
6       SplitNodeImpl(Nold, Nnew1, Nnew2, time)
7   ELSE
8     SplitNodeImpl(Nold, Nnew1, Nnew2, null)

Algorithm SplitNodeImpl(Node Nold, Node Nnew1, Node Nnew2, Axis garbageAxis)
1   /* SplitNodeImpl is similar to R*-tree's split algorithm
2   except that garbageAxis is not considered */
3   invoke ChooseSplitAxis
4   invoke ChooseSplitIndex
5   distribute N+1 entries into Nnew1 and Nnew2
```

5.4 검색 기법

제안된 색인구조에서의 검색 기법은 R-tree를 기반으로 한다. 그러나 동적 데이터를 검색하기 위해서 MBB 확장기법을 사용한다. 검색 중 state가 D, DI, DL 중 하나인 엔트리와 만나게 되면 해당 엔트리의 MBB를 확장속도에 따라 확장하여 검색을 수행하여 최근 보고 지점과 검색 시점 사이의 질의를 가능하게 한다. 그 알고리즘은 표 4와 같다.

기존 검색 알고리즘을 이용한 궤적 검색은 그 성능에 문제가 있다. 따라서 본 논문에서는 표 5과 같이 궤적 검색 알고리즘을 사용한다. 궤적 검색은 RFID 시스템 구간과 LBS 시스템 구간을 나누어 처리한다. RFID 시스템 구간으로만 이루어진 태그 객체의 궤적 검색은 각 SI들에 존재하는 링크를 따라가는 것으로 수행되며, LBS 시스템 구간과 혼합된 태그 객체의 궤적 검색은 SL의 끝점을 이용한 점 질의와 SI의 링크를 따라가는 기법을 병행하여 궤적 검색의 성능을 향상시킨다.

표 4 Algorithm Search

```
Algorithm Search(Node root, MBB mbb)
1   Set N to be the root (for first call)
2   IF N is a leaf
3     FOR EACH entry e
4     IF e.state is SL OR SI AND e intersects mbb
5       return e
6     ELSE IF e is (DL OR DI) AND extension of e intersects mbb
7       return e
8   ELSE
9     FOR EACH entry e
10    IF e.state is S AND e intersects mbb
11      Search(e, mbb)
12    ELSE IF e.state is D AND extension of e intersects mbb
13      Search(e, mbb)
```

표 5 Algorithm SearchTrajectory

```
Algorithm SearchTrajectory (Node root, MBB mbb)
1   Set of entry result := {}
2   Entry entry := Search(root, mbb)
3   DO WHILE e intersects mbb
4     Add e to result
5     IF e.state is SI
6       e := e.next_link
7     ELSE IF e.state is SL
8       e := Search(root, e.mbb)
9   Return result
```

6. 결론 및 향후 연구

태그 객체의 이동 궤적은 RFID 시스템 구간에서는 연속된 선분의 집합으로 나타나고, LBS 시스템 구간에서는 연속된 선분의 집합으로 나타난다. 단절된 간격 데이터로 이루어진 RFID 시스템 구간에서는 간격 데이터 간에 연결정보가 없어 궤적 검색에 비효율적이고, 연속된 선분으로 나타나는 LBS 시스템 구간에는 최근 보고 시점부터 현재 시점까지의 질의를 처리하는 비용이 높은 문제가 있다. 이와 같은 문제를 해결하기 위해 본 논문에서는 RFID/LBS 환경에서 태그 이동 객체의 효율적인 현재 질의, 궤적 검색을 위한 데이터 모델과 색인 및 삽입, 분할, 검색 기법을 제안하였다.

향후 태그객체 간 포함관계를 고려한 색인 구조의 연구가 필요하고, 기존 색인구조와의 성능평가를 통한 검증이 필요하다.

참고문헌

[1] K.Romër, T.Schoch, F.Mattern and T. Dubendorfer, "Smart Identificatrion Frameworks for Ubiquitous Computing applications", In Pervasive Computing and Communications Proc. of the First IEEE International Conf., pp.256-262, 2003

[2] C. H. Ban, B. H. Hong, and D. H. Kim, "Time Parameterized Interval R-tree for Tracing Tags in RFID Systems", 16th Int. Conf. on DEXA, pp.503-513, 2005.

[3] Y. Theodoridis, M. Vazirgiannis and T. Sellis, "Spatio-Temporal Indexing for Large Multimedia Applications", In Proc. of the 3rd IEEE Conf. on Multimedia Computing and Systems, pp.441-448, June 1996

[4] A. Guttman, "R-tree: A Dynamic Index Structure for Spatial Searching", ACM SIGMOD Conf., pp47-54, 1984

[5] S. Berchtold, D. A. Keim, and H. P. Kriegel, "The X-tree: An Index Structure for High-Dimensional Data", In Proc. of Int. Conf. on 22nd VLDB, pp.28-39

[6] N. Beckmann and H. P. Kriegel, "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles", In Proc. ACM SIGMOD Int. Conf. on Management of Data, pp.322-331, 1990