RFID 태그 객체를 위한 구간 색인 구조의 설계 및 구현

반재훈* · 홍봉희**

*고신대학교 · **부산대학교

Design and Implementation of Index for RFID Tag Objects

Chae-Hoon Ban* · Bong-Hee Hong**

*Kosin University · **Pusan National University

E-mail : chban@kosin.ac.kr* · bhhong@pusan.ac.kr**

요 약

RFID 시스템에서 태그의 위치를 추적하기 위해서는 태그의 궤적을 모델링하고 색인으로 구성해야 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 않는 경우에 태그의 궤적은 인식영역에 들어올 때만 보고된 점으로 표현되며, 질의 처리 시 이러한 태그를 찾기 위해 질의영역을 확장해야하는 문제가 발생한다.

이러한 문제를 해결하기 위하여 이 논문에서는 RFID 태그의 궤적을 위한 구간 데이터 모델을 정의한다. 또한 구간 데이터 모델에 적합한 R-tree 기반 색인 구조인 IR-tree(Interval R-tree)를 제시하며 효율적인 질의처리를 위해 시간에 종속적인 동적 구간의 특성을 고려한 새로운 삽입 및 분할 알고리즘을 제안한다. 마지막으로 다양한 데이터 집합에서 제안된 색인과 기존 알고리즘을 사용하는 색인과의 성능비교를 통하여 색인의 우수성을 입증한다.

ABSTRACT

For tracing tag locations, a trajectories should be modeled and indexed in radio frequency identification (RFID) systems. The trajectory of a tag can be 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 and we should extend the region of a query to find the tag that remains in a reader.

In this paper, we propose an interval data model of tag's trajectory in order to solve the problem. For the interval data model, we propose a new index scheme called the IR-tree(Interval R-tree) and algorithms of insert and split for processing query efficiently. We also evaluate the performance of the proposed index scheme and compare it with the previous indexes.

키워드

RFID, Tag, Trajectory, Data Model, Index, R-trees

Ⅰ. 서론

RFID(Radio Frequency IDentification)는 각종 물품에 소형 칩인 태그(Tag)를 부착하고 판독기(Reader)를 통해 인식하는 시스템으로 항만/물류/유통, 군사, 식품/안전 등 비즈니스 영역에 킬러 애플리케이션으로서 막대한 파급 효과를 끼칠 전망이다[1].

RFID 응용에서는 태그를 부착한 실 개체(이하 태그)의 위치를 추적하기 위하여 태그가 판독기에 들어올 때와 나갈 때 보고되는 시공간 위치를 이용하여 태그의 궤적을 선분으로 표현하고 색인을 구성한다[2]. 그러나 이러한 궤적의 모델링 기법을 사용하는 경우에 판독기에 머무는 태그의 궤적은 태그가 판독기에 들어갈 때 보고한 시공간 점으로만 구성되어 태그가 판독기에 머문다는 정보를 표현할 수 없다. 따라서 이러한 태그를 찾기 위해 질의영역을 확장해야 하는 문제가 발생한다.

이 논문에서는 위와 같은 문제를 해결하기 위하여 구간 데이터 모델을 정의한다. 이 모델에서는 태그의 궤적을 시간에 종속적인 선분인 동적 구간(dynamic interval)과 시간에 고정적인 정적 구간(static interval)으로 표현한다. 따라서 판독기에 머무는 태그의 궤적은 시간에 종속적인 동적 구간으로 표현하여 태그를 효율적으로 찾을 수 있게 한다. 또한 구간 데이터 모델에 적합한

R-tree 기반 색인 구조인 IR-tree(Interval R-tree)를 제시하며 효율적인 질의처리를 위해 시간에 종속적인 동적 구간의 특성을 고려한 새로운 삽입 및 분할 알고리즘을 제안한다. 마지막으로 다양한 데이터 집합에서 제안된 색인과 기존 알고리즘을 사용하는 색인과의 성능비교를 통하여 색인의 우수성을 입증한다.

II. 구간 데이터 모델

태그가 인식영역에 들어오면 Enter 이벤트가 발생하며 반대로 태그가 인식영역을 빠져나가면 Leave 이벤트가 발생한다[1]. 이 논문에서는 태그의 단일궤적을 다음과 같이 정의한다. 정의에서 *tid, rid, t*는 3차원 공간의 각 축을 의미하며 tid_i는 태그의 식별자, rid_j는 판독기의 식별자, t_{enter}, t_{leave}는 각각 Enter와 Leave 이벤트의 발생 시간을 의미한다.

정의 1: 판독기 rid_j에서의 태그 tid_i의 궤적 tr
$tr = \{(tid,rid,t)R^3 \mid tid=tid_i,\ rid=rid_j,\ t_{enter} \le t \le t_{leave}\}$

태그가 호출 영역 안으로 들어가면, Enter 이벤트가 발생하고, 이때 t_{enter} 값을 알 수 있다. 그러나 언제 호출 영역 밖으로 나올지 모르기 때문에 t_{leave} 값은 알 수 없다. 이러한 태그는 [정의 1]에 의해 t_{enter}로 구성된 점으로 표현된다. 이 경우, 태그가 현재 어느 판독기의 호출 영역에 들어 있는지를 알기 위해서는 질의영역을 확장해야하는 문제가 발생한다. 이 논문에서는 이러한 문제를 해결하기 위하여 시간에 따른 태그의 궤적을 정적 구간과 동적 구간으로 분류하며 아래와 같이 정의한다. 정의에서 t_{now}를 현재시간을 의미한다.

정의 2: 동적 구간(dynamic interval) dI = $\{(tid,rid,t)R^3 \mid tid=tid_i,\ rid=rid_j,\ t_{enter} \le t \le t_{now}\}$.
정의 3: 정적 구간(static interval) sI = $\{(tid,rid,t)R^3 \mid tid=tid_i,\ rid=rid_j,\ t_{enter} \le t \le t_{leave}\}$.

태그가 판독기에 들어오면 동적 구간이 생성되며 시간 구간의 끝 값은 현재시간을 나타내는 t_{now}로 변경된다. 따라서 동적 구간의 시간 축 길이 $dI_t=\{t \mid t_{enter} \le t \le t_{now}\}$가 되며 t_{now}에 의해 시간 축 길이가 계속해서 변하게 된다. 판독기에 들어와 아직 빠져나가지 않은 태그의 궤적이 시간에 종속적인 선분으로 표현되므로, 질의영역의 시간 구간 최소값을 t_{query}라 하고 $t_{enter} \le t_{query} \le t_{now}$이라 하면 항상 $dI_t \ni t_{query}$를 만족하여 질의를 처리할 수 있다.

만약 태그가 판독기를 빠져 나가게 되면 시간 구간의 끝 값은 t_{leave}로 변경되어 정적 구간이 된다. 이 경우 정적 구간의 시간 축 길이 $sI_t=\{t \mid t_{enter} \le t \le t_{leave}\}$가 되며 질의영역의 시간구간 최소값이 $t_{enter} \le t_{query} \le t_{leave}$인 경우 $sI_t \ni t_{query}$를 만족하여 질의를 처리할 수 있다.

예를 들어, 그림 1-(a)와 같이 태그가 판독기에 t_{enter}시간에 들어와 t_{leave}시간에 빠져 나간다고 가정하자. $t_{enter} \le t_{now} < t_{leave}$인 경우에 태그의 궤적은 그림 1-(b)와 같이 태그가 아직 판독기를 빠져 나오지 않았으므로 시간 축 범위가 $[t_{enter}, t_{now}]$인 동적 구간이 되어 질의 처리가 가능하다. $t_{enter} \le t_{leave} < t_{now}$인 경우에는 태그가 그림 1-(c)와 같이 판독기를 빠져 나왔으므로 궤적은 시간 축 범위가 $[t_{enter}, t_{leave}]$인 정적 구간이 되어 질의 처리가 가능하다. 따라서 위와 같이 태그의 궤적을 정적 구간, 동적 구간으로 정의하면 언제나 질의처리가 가능하다.

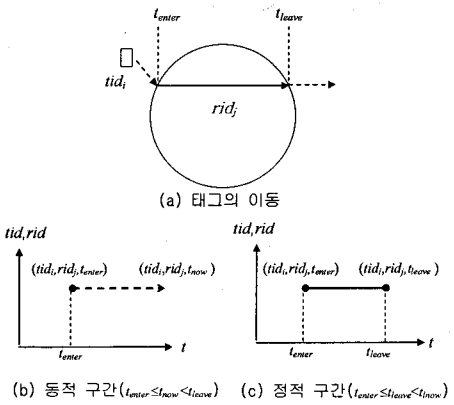


그림 1. 구간 데이터 모델

III. IR-tree(Interval R-tree)

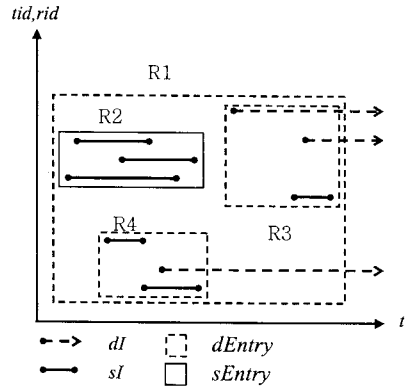
이 논문에서는 구간 데이터 모델에 적합한 색인 구조인 IR-tree(Interval R-tree)를 제안한다. 또한 효율적인 질의처리를 위해 시간에 종속적인 동적 구간의 특성을 고려한 새로운 삽입 및 분할 알고리즘을 제안한다.

3.1 데이터 구조 및 탐색

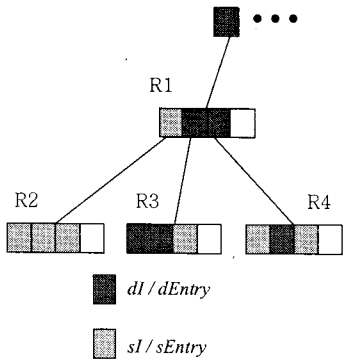
IR-tree의 단말 노드는 <*MBB*>의 형태의 구간을 엔트리로 포함한다. *MBB*는 3차원 최소경계박스로서 <*tid, rid,* $[t^{\vdash},t^{\dashv}]$>의 형태를 가진다. 동적 구간(이하 *dI*)은 시간의 시작 값만이 보고되고 시간의 끝 값은 보고되지 않았으므로 <*tid, rid,* t_{enter}, t_{now}>의 형태로 단말 노드에 삽입된다. 정적 구간(이하 *sI*)은 판독기에 들어온 시간 값과 나간 시간 값이 모두 보고되므로 <*tid, rid,* t_{enter}, t_{leave}>의 형태로 단말 노드에 삽입된다.

비단말 노드는 <*cp, state, MBB*>의 형태를 갖는 엔트리를 포함한다. 여기서 *cp*는 자식 노드를 가리키는 포인터, *state*는 엔트리의 상태를 나타내며 *MBB*는 <$[tid^{\vdash},tid^{\dashv}]$, $[rid^{\vdash},rid^{\dashv}]$, $[t^{\vdash},t^{\dashv}]$>형태의 3차원 최소경계박스를 나타낸다. 엔트리의 상태 *state*는 *sEntry*와 *dEntry*의 두 가지로 분류된다. 먼

저 *sEntry*는 자식 노드가 포함하는 엔트리가 모두 *sEntry*이거나 *sI*인 경우이며 이 경우에 *MBB*는 자식 노드의 모든 엔트리들을 포함하는 두 개의 3차원 시공간 점으로 구성된다.



(a) 3차원 공간의 구간



(b) IR-tree의 구조

그림 2. IR-tree의 구성 예

자식 노드가 포함하는 엔트리가 하나 이상의 *dEntry*나 *dI*인 경우에 *state*는 *dEntry*가 된다. 이 경우에 *MBB*는 자식노드가 단말노드인 경우에는 *sI* 선분 전체와 *dI*의 시작점만을 포함하는 영역이며, 비단말 노드인 경우에 *MBB*는 *dEntry*와 *sEntry*를 포함하는 영역이 된다. 따라서 이러한 *dEntry*는 하위 노드에 *dI*를 포함하므로 질의 처리 시에 현재 시간 값으로 시간 구간을 확장해서 처리해야 한다.

예를 들어 그림 2와 같이 3차원 공간에 *sI*와 *dI*가 존재한다고 가정하자. 단말 노드 R2는 그림 2-(a)와 같이 세 개의 *sI*를 포함하므로 이것은 *sEntry*가 되며 모든 *sI*를 포함하는 *MBB*를 가지게 된다. 따라서 부모 노드 R1에서 R2를 가리키는 엔트리의 *state*는 *sEntry*가 된다. R3는 하나의 *sI*와 두 개의 *dI*를 포함하므로 *dEntry*가 되며 이것의 *MBB*는 *sI* 전체와 각 *dI*의 시작점만을 포함하게 된다. 이와 같은 방식으로 R4와 R1은 모두 *dEntry*가 된다.

탐색은 기존의 R-tree와 유사하게 색인의 루트에서부터 하위방향으로 탐색해 나간다. 그러나 IR-tree에서는 노드의 *state* 값에 따라 탐색 방법이 달라진다. 먼저 *sEntry*의 경우에는 단말 노드가 나올 때까지 기존 탐색과 동일하게 *MBB*가 질의 영역과 교차하면 하위 노드로 탐색을 수행한다. 그러나 *dEntry*의 경우에는 먼저 *MBB*의 시간 축을 현재 시간인 t_{now}까지 확장하여 질의 영역과 교차하는지를 검사하여 교차하면 하위 노드로 탐색을 수행한다.

단말 노드가 발견되면 포함된 *sI*와 *dI*를 검사하여 질의의 결과로 반환하게 된다. 이 경우에도 *dI*는 그 시간 축을 현재 시간인 t_{now}까지 확장하여 질의 영역과 교차하는지를 검사한다. 이처럼 *dEntry*와 *dI*의 시간 축을 확장하는 이유는 IR-tree에서 단말 노드를 구성할 때 *dI*의 경우 시작점만으로 노드를 구성하기 때문이다. 따라서 질의 처리 시에는 이러한 *dI*와 이것을 포함하는 *dEntry*를 현재 시간 값으로 확장하여 질의를 처리해야 한다.

3.2 삽입

데이터 삽입은 삽입될 노드를 찾는 정책이 중요하다. R-tree에서는 삽입될 노드를 최소영역확장(least area enlargement) 정책을 이용하여 찾게 된다. 그러나 IR-tree에 이러한 정책을 사용하게 되면 *dEntry*가 질의 처리 시에 확장되는 특성을 고려하지 않기 때문에 노드들의 면적과 겹침이 증가되어 질의 처리가 비효율적이다.

따라서 구간을 삽입하는 경우에 첫째, 삽입되는 구간의 종류, 둘째 삽입의 대상이 되는 엔트리의 속성 변화, 마지막으로 삽입으로 인해 확장된 엔트리의 면적을 동시에 고려한 삽입 방법이 필요하다. 이를 위하여 이 논문에서는 먼저 *Local Fixed MBB(LFMBB)*를 아래와 같이 정의한다.

> **정의 4**: *dEntry*의 *LFMBB*는 *dEntry*를 포함하는 노드가 가지는 최대 시간 값으로 *dEntry*의 시간 값을 고정시킨 *MBB*

이것은 *dEntry*가 확장되는 성질을 일정으로 보장해 주므로 삽입 수행 시 이러한 *LFMBB*를 사용하여 최소영역확장 방법을 사용한다.

표 1. 엔트리들의 영역확장

삽입 구간	선택 엔트리	영역확장
sI	*sEntry*	Area(*MBB'*)-Area(*MBR*)
	dEntry	Area(*LFMBB'*)-Area(*LFMBB*)
dI	*sEntry*	Area(*LFMBB'*)-Area(*MBB*)
	dEntry	Area(*LFMBB'*)-Area(*LFMBB*)

표 1은 삽입되는 구간의 타입에 따라 삽입 시 선택되는 엔트리들의 영역확장 계산 방법을 제시

하고 있다. 표에서 Area( )는 영역을 계산하는 함수이며 *MBB'*와 *LFMBB'*는 각각 삽입되는 구간을 포함한 후에 엔트리들의 *MBB*와 *LFMBB*를 나타낸다. *sI*가 삽입되는 경우 *sEntry*의 영역확장 계산은 기존 알고리즘과 동일하게 삽입 후의 *MBB* 면적에서 삽입 전 *MBB*를 감산하여 계산한다. 그러나 *dEntry*의 경우에는 앞에서 언급한 것처럼 *LFMBB*를 사용하여 삽입 후의 *LFMBB* 면적에서 삽입 전 *LFMBB*를 감산하여 계산한다.

3.3 분할

노드에 오버플로우가 발생하면 기존 방법에서는 마진을 사용하여 분할 축을 설정하고 겹침이 최소화되도록 노드를 분할하였다. 그러나 IR-tree에 이 정책을 사용하게 되면 *dEntry*와 *dI*가 질의 수행 시 확장되는 특성을 고려하지 않았기 때문에 노드들 간에 겹침이 증가되어 비효율적이다. 따라서, 이 논문에서는 먼저 아래와 같이 *Local Fixed dI(LFdI)*를 정의한다.

정의 5: *LFdI*는 단말 노드의 최대 시간 값으로 *dI*의 시간 구간을 고정 시킨 구간

IR-tree에서의 분할은 앞에서 정의한 *LFMBB*와 *LFdI*를 이용하여 수행한다. 먼저 비단말 노드의 분할 시에 *sEntry*는 *MBB*를 *dEntry*는 *LFMBB*를 사용하여 R*-tree에서의 분할 방법과 동일한 방법으로 비단말 노드를 분할한다. 또한 단말 노드 분할의 경우에 *sI*는 그대로, *dI*는 *LFdI*로 확장하여 R*-tree에서의 분할 방법과 동일한 방법으로 분할한다.

Ⅳ. 성능평가

논문에서 제안하는 색인의 성능을 평가하기 위하여 기존 R-tree 계열의 삽입 및 분할 방법과 비교한다. 색인의 성능 비교의 기준은 각 색인의 구축 비용과 각 질의의 처리 비용이다. 이 논문에서 제안하는 새로운 색인의 실험을 위하여 태그의 위치 데이터를 생성하기 위한 태그 데이터 생성기(Tag Data Generator, TDG)를 개발하여 사용하였다.

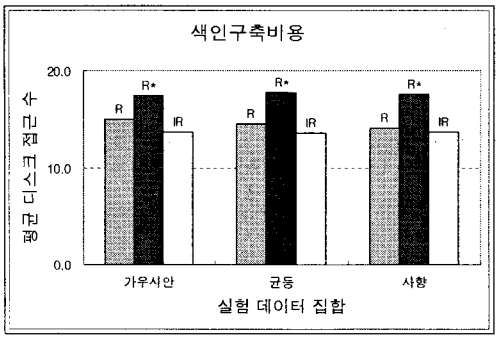


그림 3. 데이터집합에 따른 색인 구축 비용

그림 3과 같이 데이터 집합에 따른 색인 구축 비용은 IR-tree<R-tree<R*-tree의 순으로 나타났다. IR-tree의 경우 질의 처리 능력이 우수하므로 갱신 시 이전 정적 구간을 찾을 때 노드의 접근이 상대적으로 작다. R*-tree는 노드 오버플로우가 발생할 시에 노드를 분할하지 않고 데이터를 재삽입하기 때문에 색인 구축비용에서 성능이 가장 낮다.

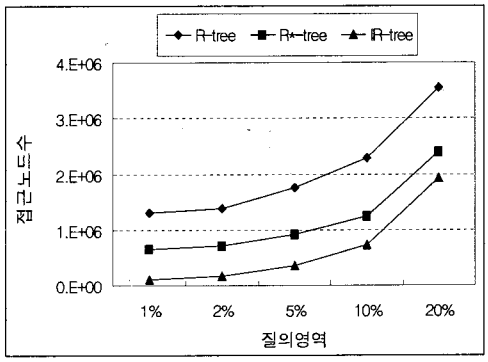


그림 4. 가우시안분포에서 FIND 질의 성능

그림 4는 1,000개의 태그로 100,000번의 Enter와 Leave 이벤트를 발생시켜 생성한 가우시안 분포에 대하여 FIND 질의를 수행한 결과이다. 그림에서와 같이 IR-tree의 성능이 가장 우수했고 R-tree가 가장 낮은 성능을 보였다. IR-tree가 우수한 성능을 보이는 이유는 질의 처리 시 확장되는 *dI*와 *dEntry*의 특성을 고려한 삽입 및 분할 알고리즘을 적용하였기 때문이다.

Ⅴ. 결론 및 향후 연구

이 논문에서는 태그의 위치 추적을 위해 궤적을 표현하는 방법을 제시하였다. 기존 궤적의 표현방법의 문제점을 소개하고 문제 해결을 위해 구간 데이터 모델을 정의하였다. 또한 구간 데이터 모델에 적합한 IR-tree를 제시하고 효율적인 질의처리를 위해 시간에 새로운 삽입 및 분할 알고리즘을 제안하였으며 다양한 데이터 집합에서 성능비교를 통하여 색인의 우수성을 입증하였다.

향후 연구로서는 태그의 이동 특성을 고려한 삽입, 분할 알고리즘의 개발과 고급질의에 대한 연구가 필요하다.

참고문헌

[1] K. Romer, et al., "Smart Identification Frameworks for Ubiquitous Computing Applications" Proc. of Pervasive Computing and Communications, pp. 256-262, 2003

[2] Y. Theodoridis, et al., "Spatio-Temporal Indexing for Large Multimedia Applications", In International

[3] 반재훈, 홍봉희, "RFID 태그 객체의 위치 추적을 위한 구간 데이터 모델", 한국해양정보통신학회 추계종합학술대회, pp. 578-581, 2007.