RFID 태그의 이력 추적을 위한 시간 간격 색인 : SLR-트리

(A Time Interval Index for Tracking Trajectories of RFID Tags : SLR-Tree)

류 우 석 † 안 성 우 † 홍 봉 희 †† 반 재 훈 ††† 이 세 호 ††††
(Wooseok Ryu) (Sungwoo Ahn) (Bonghee Hong) (Chaehoon Ban) (Seho Lee)

요 약 RFID 시스템에서의 태그의 궤적은 태그가 리더의 인식영역에 들어왔을 때와 벗어날 때의 시공간 위치를 선분으로 연결하여 표현한다. 그러나 태그가 리더의 인식영역을 벗어난 후 다음 리더의 인식영역에 들어올 때까지는 태그의 위치를 파악할 수 없으므로 태그의 궤적은 연결되어 있지 않고 단절된 간격의 집합으로 표현된다. 그러므로 태그의 이력을 검색하기 위해서는 전체 색인을 검색해야 하는 문제가 발생한다. 이 논문에서는 높은 궤적 검색 비용문제를 해결하기 위해 전자태그의 간격을 연결하기 위한 기법을 제시하고 이 기법을 적용한 색인인 SLR-tree를 제안한다. 또한, 연결 정보의 추가로 인한 노드의 공간 활용도의 저하를 최소화하기 위하여 두 간격간의 연결정보를 공유하기 위한 기법을 제안하고 노드의 분할 시 공유정보를 유지하기 위한 분할 정책을 제안한다. 마지막으로 제안된 색인에 대한 성능을 비교평가 함으로써 이력검색 성능의 우수성을 입증한다.

키워드 : RFID, 태그색인, 태그궤적, 이력질의

Abstract The trajectory of a tag in RFID system is represented as a interval that connects two spatiotemporal locations captured when the tag enters and leaves the vicinity of a reader. Whole trajectories of a tag are represented as a set of unconnected interval because the location of the tag which left the vicinity of a reader is unknown until it enters the vicinity of another reader. The problems are that trajectories of a tag are not connected. It takes a long time to find trajectories of a tag because it leads to searching the whole index. To solve this problem, we propose a technique that links two intervals of the tag and an index scheme called SLR-tree. We also propose a sharing technique of link information between two intervals which enhances space utilization of nodes, and propose a split policy that preserves shared-link information. And finally, we evaluate the performance of the proposed index and prove that the index processes history queries efficiently.

Key words : RFID, tag indexing, tag trajectory, history query

1. 서 론

RFID(Radio Frequency Identification)[1]는 무선 주파수를 이용하여 전자태그를 부착한 객체를 리더가 자동으로 인식하고 전자태그의 정보를 읽는 기술이다. 기존의 바코드를 대체하여 유비쿼터스 환경을 위한 차세대 인식 기술로 주목을 받고 있으며 재고관리, 물류의 공급망 관리, 공장 자동화 등의 다양한 응용분야에서 활용되고 있다. 이러한 응용에서 생산자는 상품의 판매 상황을 조회하며 소비자는 전자태그의 위치를 추적하여 상품의 유통 경로를 조회할 수 있다.

전자태그를 부착한 제품이 이동 중에 리더의 인식영역에 들어오면 리더가 전자태그를 판독하여 전자태그의 위치를 보고한다. 이때 전자태그의 위치는 곧 리더의 위치가 된다. 전자태그는 시간에 따라 위치를 이동한다는 측면에서 이동체(Moving Object)와 유사한 특징이 있

· 이 논문 또는 저서는 2007년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(지방연구중심대학육성사업/차세대물류IT기술연구사업단)
† 학생회원 : 부산대학교 컴퓨터공학과
wsryu@pusan.ac.kr
swan@pusan.ac.kr
†† 종신회원 : 부산대학교 컴퓨터공학과 교수
bhhong@pusan.ac.kr
††† 정 회 원 : 경남정보대학 인터넷응용계열 교수
chban@kit.ac.kr
†††† 정 회 원 : 삼성전자 정보통신총괄 연구원
hose.lee@samsung.com
논문접수 : 2006년 5월 2일
심사완료 : 2006년 11월 14일

다. 이동체에서는 연속적으로 보고되는 시공간 위치를 선분으로 연결함으로써 이동체의 궤적을 표현할 수 있으며 전자태그도 마찬가지로 전자태그가 리더의 인식영역에 들어올 때와 나갈 때의 시공간 위치를 선분으로 표현함으로써 이동 궤적을 표현할 수 있다. 이 때의 선분을 구성하는 두 시공간 위치는 리더와 태그가 동일하므로 시간 축에 평행한 간격(interval)으로 표현된다.

그러나 이동체와 달리 전자태그의 위치보고는 리더의 인식 영역 안에서만 이루어지며 리더들의 인식영역이 모든 공간 영역에 걸쳐 있지 않으므로 태그가 특정 리더의 인식영역을 벗어난 후 다른 리더의 인식영역에 들어올 때까지 시간의 차이가 발생하며 해당 시간에는 전자태그의 위치를 확인할 수 없다. 이때 이전 리더의 시공간 위치와 다음 리더의 시공간 위치를 연결할 수 없으며, 오직 리더의 인식영역에 들어갈 때와 나갈 때의 두 시공간 위치만을 연결할 수 있다. 그러므로, 전자객체의 궤적은 이산적인(discrete) 선분의 집합으로 표현되는데, 이는 다중 선(polyline)으로 궤적을 표현하는 이동체 모델과 서로 차이점이 있다.

공급망 관리와 같은 RFID 응용 시스템에서 "Tag#9가 지난 하루 동안 이동하였던 경로는?"과 같은 이력 질의는 빈번하게 발생되는 질의이며, 또한 빠른 처리 속도를 필요로 한다. 그러나, RFID와 이동체의 특성 차이로 인하여 이동체 색인 모델을 바로 적용하였을 때 이력 질의를 처리할 수 없는 문제점이 있다. 그러므로, 이력 질의를 효율적으로 처리하기 위한 새로운 방법이 제시되어야 한다.

이 논문에서는 전자태그의 위치 추적 서비스에서 사용되는 이력 질의를 효율적으로 처리하기 위한 전자태그 간격의 연결기법과 색인 구조를 제시한다. 전자태그를 위한 데이타 모델로서 식별자가 주어지는 질의를 처리하기 위해서 전자태그 식별자를 새로운 도메인으로 추가한다. 그리고 실좌표계 대신에 리더의 식별자를 도메인으로 사용하였으며, 시간 도메인을 추가한 3차원 모델로 전자태그의 위치 데이타를 표현한다. 그리고, 전자태그의 이동 간격을 연결하고 효율적인 이력 검색을 지원하기 위해 R*-tree[2] 기반 색인 구조인 SLR-tree (Shared Link R-tree)를 제시하고 색인에서의 삽입 정책 및 이력 검색 기법을 제안한다. 이 색인을 적용하면 전자태그에 대한 궤적 수행 속도를 향상시키게 되어 객체의 이력 추적을 필요로 하는 다양한 분야에 활용할 수 있다.

이 논문의 구성은 다음과 같다. 먼저 2장에서는 관련 연구를 소개하고, 3장에서는 대상 환경 및 전자태그의 특성으로 인한 문제점을 정의한다. 4장에서는 전자태그의 위치정보를 연결하는 방법을 제안하고 전자태그의 위치추적을 위한 색인 구조와 삽입 정책 및 궤적 검색 기법을 제시한다. 5장에서는 실험을 통하여 기존 색인과의 성능을 비교한다. 마지막으로 6장에서 결론 및 향후 연구를 기술한다.

2. 관련 연구

2.1 이동체를 위한 색인

이동체는 시간에 따라 위치가 변화하기 때문에 시공간 객체로 볼 수 있으며, 시공간 데이타를 저장하는 색인 중에서 R-tree[3]에 기반한 3DR-tree[4]가 이동체를 위한 색인으로 사용될 수 있다. 이동체를 위한 색인으로는 TB-tree[5] 등이 있다.

R-tree[3]는 대표적인 공간 색인으로서 공간 객체를 최소경계사각형(MBR)을 사용하여 표현하고 저장하는 균형 트리 구조의 색인이다. 삽입 시 최소 영역 확장 정책을 통해 삽입 및 분할함으로써 영역 검색에서 높을 성능을 나타내어 시공간 색인 등 많은 종류의 색인에서 기본 색인으로서 활용되고 있다[2,4-7]. 그리고, R*-tree [2]는 R-tree의 단점인 노드간의 중첩과 사장공간으로 인한 성능 저하를 개선하기 위해 영역확장 및 영역 크기, 노드간의 중첩을 모두 고려하는 새로운 삽입 알고리즘과 분할 알고리즘, 그리고 재 삽입 알고리즘을 제시하여 영역질의에서 R-tree보다 뛰어난 성능을 나타낸다.

3DR-tree[4]는 R-tree를 기반으로 하여 시간을 또 다른 하나의 축으로 추가하였다. 이동체가 시간 $[t_i,t_j)$ 동안 위치 (x_i,y_i)에 있다면, 3차원 시공간에서 선분 $\overline{[(x_i,y_i,t_i),(x_i,y_i,t_j))}$으로 표현할 수 있으며, 3차원 R-tree를 사용하여 색인함으로써 특정 시간과 공간 영역이 주어지는 영역 질의에 높은 성능을 보이고 있다.

TB-tree[5]는 궤적 질의 및 복합 질의의 성능 향상을 위하여 극단적인 궤적 보존 정책을 사용하였다. 하나의 단말 노드는 오직 동일한 이동체에 대한 선분들만을 저장하고, 이 단말 노드들에 대해 양방향 연결 리스트를 구성하여 궤적 질의 성능을 향상 시킬 수 있다. 그러나 하나의 단말 노드에 하나의 이동체에 대한 선분만을 저장하기 때문에 공간 근접성이 무시되어 있으므로 영역질의 성능이 떨어지는 문제점이 있으며, 특히 리더를 따라 이동하는 RFID환경에서는 태그가 직전에 인식되었던 리더와 이후 인식된 리더의 위치가 근접하여 있지 않으므로 노드의 중복이 매우 증가하여 영역질의 성능이 더욱 저하되는 문제가 있다.

2.2 간격을 위한 색인

간격(interval)은 $[l, r](l \le r)$인 형태를 가지며 어떤 속성이 색인 상에서 유효한 기간(duration)을 나타내는 순서쌍이다. 대표적인 간격 색인인 SR-tree[8]는 메인 메

모리 거주 색인 구조인 Segment Tree[9]를 디스크 기반으로 확장한 색인이다. 색인 구조는 R-tree와 동일한 구조를 가지며 기존의 삽입 정책과 분할 정책을 그대로 사용한다. 차이점은 자식 노드를 완전히 걸치는 긴 간격 데이타가 삽입될 경우, 데이타를 단말 노드에 저장하지 않고 걸쳐지는 노드의 부모 노드에 저장한다는 점이다. SR-tree는 비 단말 노드에도 데이타를 저장하므로, 노드의 크기가 루트 노드로 갈수록 커지는 단점이 있다.

TPIR-tree[10]는 RFID 전자태그 간격을 위한 색인으로 전자태그를 매개 인자로 하는 질의에서 빠른 검색을 지원하기 위하여 x-y 좌표와 시간에 전자태그 식별자를 도메인으로 추가하였으며, 리더 인식영역을 벗어난 간격(*FixedI*)과 리더 인식 영역에 머물러 있는 간격(*NowI*)을 정의하였다. 특정 리더에 현재 머물러 있는 태그를 *NowI*로 모델링하고 이를 R*-tree를 개선한 색인 알고리즘을 제안함으로써 현재 위치 질의를 효율적으로 지원한다. 그러나 TPIR-tree에서는 각각의 간격을 연결하고 있지 않으므로, 특정 전자태그의 이력을 추적하기 위해서는 해당 전자태그의 식별자를 이용하여 전체 색인을 검색하여야 하는 문제점이 발생한다.

3. 대상환경 및 문제정의

이 논문에서 RFID 시스템을 사용하는 응용에서 다양한 질의 처리 및 전자태그의 위치 추적을 위한 데이타 모델과 색인 구조를 제안한다. 이 장에서는 먼저 RFID 전자태그를 부착한 객체의 위치 데이타를 수집하는 방법과 기존 이동체 데이타 모델과 색인 구조의 문제점에 대해서 기술한다.

RFID 시스템은 그림 1과 같이 전자태그와 리더, 그리고 서버로 구성된다. 물류환경이나 일반적인 위치 추적 같은 응용에서는 이동하는 제품이나 사람에게 전자태그를 부착하여 전자태그의 위치를 이를 부착한 사물로 판단하는 것이다. 리더기는 전략적으로 전자태그들이 드나드는 주요 지점에 고정 설치되며, 사용되는 전자태그는 스스로 자신의 위치를 판단할 수 없다.

따라서, 리더는 자신의 전자태그 인식영역 안으로 전자태그가 존재할 경우 리더는 이에 대하여 전자태그의 위치를 자신의 위치로 맵핑하여 서버로 보고하게 된다. 리더의 인식영역 안으로 태그가 들어오면 이에 대하여 *In_Event*를 발생하여 판독한 전자태그의 정보와 함께 서버로 전송하고, 이와 반대로 전자태그가 리더의 인식영역을 나갈 때는 *Out_Event*를 발생하여 그 정보를 서버에 전송하게 된다. 리더의 인식영역은 물리적인 리더의 인식영역이 아닌 논리적 인식영역으로 예를 들어서, 창고의 입구와 출구에 있는 물리적 리더가 입구에서 전자태그를 읽으면 *In_Event*를, 출구에서 읽히면 *Out_Event*를 발생하여 창고에 부착된 물리적 리더를 논리적 리더로 맵핑하고 창고를 하나의 논리적 리더의 인식영역으로 판단할 수 있다.

RFID 시스템에 저장되어 있는 태그 이벤트 데이타에 대한 위치 및 이력 검색을 위한 질의는 다음과 같이 4종류로 분류된다[11]. FIND질의는 "18:00~23:00까지 #13 태그가 위치한 리더는?"과 같이 전자태그 식별자와 시간이 주어지면 해당 전자태그가 해당 시간에 존재했던 리더 식별자를 반환한다. LOOK 질의는 "18:00~23:00까지 리더 #13, #14에 위치한 태그는?"과 같이 리더 식별자와 시간이 주어지면 FIND질의처럼 조건에 만족하는 전자태그 식별자를 반환한다. HISTORY질의 경우는 전자태그 식별자가 주어지는 경우 해당 전자태그가 이동한 리더와 시간을 반환한다. 그리고, WITH 질의는 복합질의로서 "18:00~23:00까지 #13 태그와 같이 있었던 모든 전자태그를 찾아라."와 같이 FIND 질의 후에 결과 집합인 리더 식별자와 FIND 질의의 시간을 사용하여 다시 LOOK질의를 수행하는 질의로서 특정 시간에 특정 태그와 함께 있었던 태그들을 반환한다.

전자태그의 궤적은 그림 2와 같이 전자태그가 리더의 인식영역 안으로 들어갈 때와 나갈 때 보고된 두 시공간 위치를 연결한 선분으로 표현된다. 보고되는 전자태그의 위치는 리더의 위치를 전자태그의 위치로 보고하기 때문에 리더의 인식영역으로 들어가고 나갈 때까지

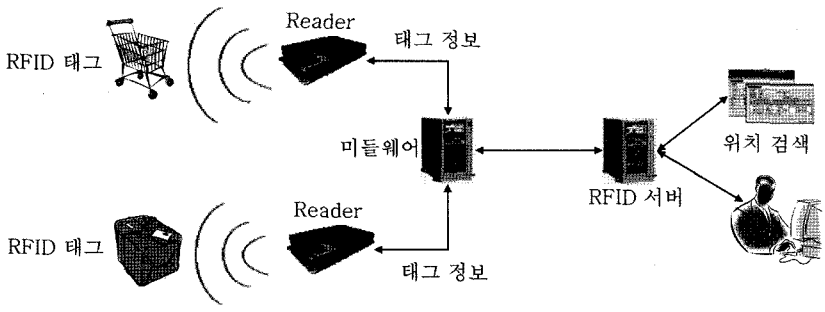


그림 1 RFID 시스템

의 리더의 인식영역 안에서는 전자태그의 위치는 변하지 않는다. 따라서, 전자태그의 궤적은 tr_{tag} = {(*readerID, tag, time*)$\in R^3$ | *readerID* = *r*, *tag* = *t*, $time_{in} \le time \le time_{out}$}인 시간 축에 평행한 간격(Interval)으로 표현된다. 전자태그의 위치는 이를 판독한 리더의 위치이므로 각 보고위치가 이산적이며, 전자태그가 리더의 인식영역 안으로 들어 왔을 때만 그 위치를 파악할 수 있기 때문에 전자태그의 궤적은 시공간적으로 연결되어 있지 않은 간격의 집합으로 나타난다.

전자태그는 이동체와 유사하게 시간에 따라 연속적으로 이동하므로 위치를 모델링 하고 효율적인 질의를 하기 위해서 이동체를 위한 시공간 색인을 적용할 수 있으나 전자태그는 이동체와 비교하여 다음과 같은 차이점을 보인다. 첫째, RFID 시스템에서 질의는 "Reader#4에 14:00부터 22:00까지 있었던 전자태그들은?"과 같이 이동체 색인의 실제 좌표와 달리 논리적 위치를 그 인자로 한다[12]. 즉, 실 좌표계인 x-y좌표계가 아닌 리더의 논리적 위치와 전자태그의 식별자를 도메인으로 가지며, 이 도메인들은 연속적이지 않고 이산적이다. 둘째, 전자태그의 궤적은 연결되어 있지 않다. 전자태그의 이동표현은 시간에 평행한 간격으로 표현되므로 공간적으로 연결되어 있지 않고, 또한 전자태그의 위치는 리더에 의해서 판독이 되므로 리더의 인식영역 밖에 있을 경우에는 전자태그의 위치를 알 수 없는 구간이 발생하게 된다. 따라서, 전자태그의 궤적은 시간적으로도 연결되어 있지 않다.

그림 3(a)와 같이 이동체의 궤적은 각 보고 지점을 잇는 선분의 집합인 다중 선으로 형태로 나타난다. 이동체의 경우 이동의 표현은 각 보고지점을 연결한 선분으로 표현되며, 궤적은 각 선분을 연결한 다중 선으로 표현하게 된다. 이 경우, 각 선분의 시작점과 그 이전의 선분의 끝점은 같기 때문에 궤적 검색 시 타임 슬라이스(time slice)질의를 통해 해당 이동체의 선분을 찾아낸 고 연결된 각 지점에 대하여 점 질의를 수행하여 궤적 추적이 가능하지만 그림 3(b)처럼 전자태그의 궤적은 이산적인 간격의 집합으로 표현되기 때문에 전후 간격을 찾기 위해 간격을 끝이나 시작점으로 점 질의를 수행하여도 다음 간격을 찾을 수 없는 문제가 있다. 또한 전자태그정보가 유효한 기간 중에서 리더의 인식영역밖에 존재하여 시간 도메인 상에서 위치정보를 표현하지 못하는 구간이 생기기 때문에 이동체 색인처럼 타임 슬라이스 질의 시 원하는 전자태그의 데이타 검색을 보장할 수 없다. 즉, 전자태그의 이동 데이타는 시공간적으로 어떠한 연결도 없기 때문에 궤적 추적 시 검색 영역의 급격한 증가로 낮은 검색 효율을 가진다.

이 논문에서는 궤적의 단절로 인한 이력 질의의 비효율성을 해결하기 위해 전자태그의 이력 질의 처리 시 전자태그의 각 간격을 탐색하기 위하여 시공간적으로

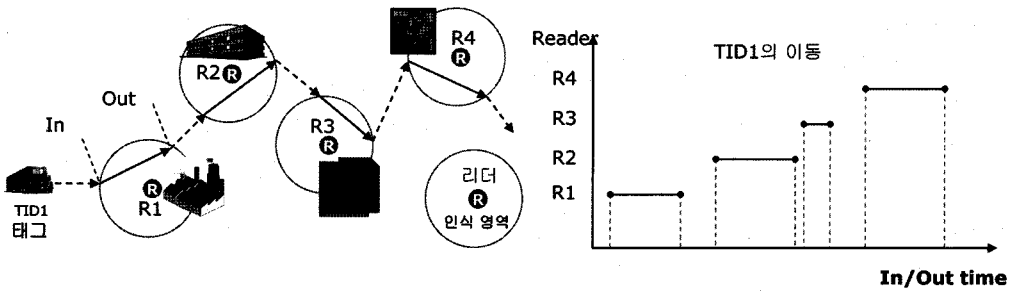
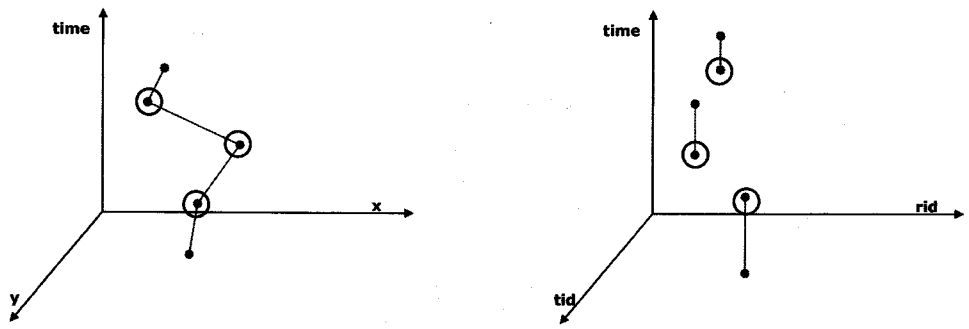


그림 2 전자태그의 이동 표현



(a) 이동체 궤적의 표현 (b) 전자태그 궤적의 표현

그림 3 이동체와 전자태그의 위치 표현 비교

연결되지 않은 간격의 연결 정보를 유지하는 색인 구조를 제안하고, 태그 객체의 식별자가 주어지는 질의와 복합 질의의 이력 질의를 처리하기 위해서 R-tree에 기반한 RFID 태그 객체의 위치 추적을 위한 색인 구조를 제안한다. 제안된 색인은 과거 위치 검색 및 이력 검색을 효율적으로 처리하며 새로운 단말 노드 분할 정책을 사용한다.

4. SLR-tree(Shared Link R-tree)

이 장에서는 (*readerID, tag, time*)으로 표현되는 전자태그의 다차원 모델에 기반하여, 전자태그의 위치추적을 위해 각 간격의 연결정보를 유지하는 색인구조를 제안한다. 이 색인 구조는 R*-tree 기반 색인 구조로서 단말 노드는 간격 데이타와 간격들의 연결정보를 동시에 저장할 수 있으며, 과거 및 현재 위치 및 연결정보를 통한 태그 이력 검색을 지원한다. 4.1절에서는 간격을 연결하기 위한 기법을 제안하고 4.2절에서는 단말 노드 구조, 4.3, 4.4, 4.5절에서는 각각 연결정보를 고려한 데이타 삽입, 갱신, 분할 방법을 제안한다.

4.1 간격 연결 기법

간격을 연결하기 위한 첫 번째 방법은 엔트리 연결 기법으로써 각 간격 엔트리마다 전(prev)/후(next) 간격을 포함하는 단말 노드의 포인터를 저장하는 방법이다. 이 방법은 그림 4와 같이 모든 엔트리에 두 개의 링크를 추가하므로 링크 탐색을 통해 이력 검색을 간단하게 수행할 수 있다. 그러나 모든 간격 엔트리마다 두 개의 링크가 추가되므로 한 노드에 포함할 수 있는 엔트리의 개수가 줄어들어서 전체적인 색인의 크기가 증가하는 문제점이 있다. 두 번째 방법은 노드 연결 기법으로써 단말 노드 내의 각 간격들이 연결되는 단말 노드를 모두 포함하는 하나의 상위 노드의 포인터를 유지하는 방법이다. 이 경우 연결 정보는 단말 노드당 전/후 두 개의 링크만으로 표현이 가능하여 단말 노드의 전체적인 공간 활용도 저하를 최소화하는 장점이 있지만 링크가 가리키는 노드의 레벨이 올라갈수록 검색영역이 급격히 넓어지고 또 이미 검색된 영역에 대하여 중복 검색이 이루어질 수 있는 단점이 있다. 그림 5처럼 단말 노드의 간격들의 다음 간격의 위치를 R1같은 상위노드를 가리킬 경우 단말 노드 안의 간격들의 다음 간격을 찾기 위해서는 R1의 하위 노드를 전부 검색하여야 한다.

본 논문에서 제안하는 공유 연결 기법은 첫 번째 방법과 같이 각 엔트리마다 두 개의 링크를 생성하는 것

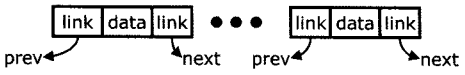


그림 4 엔트리 연결 기법의 단말 노드 구조

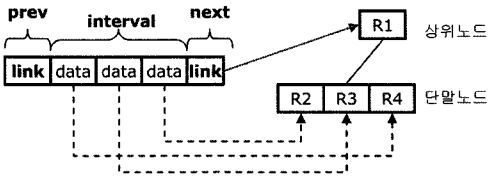


그림 5 노드 연결 기법의 단말 노드 구조

을 기반으로 하되 단말 노드의 여러 엔트리에 있는 각각의 링크들이 서로 동일하면 이를 따로 생성하는 것이 아니라 하나만 생성하여 서로 공유하는 방법이다. 동일한 링크를 서로 공유함으로써 불필요한 링크의 생성을 억제하여 첫 번째 방법의 문제점을 해결하고 있다. 그러나, 단말 노드에 있는 간격들의 연결정보가 서로 다르다면 단말 노드의 간격의 개수에 비례하여 링크의 개수 또한 증가하게 되고 이는 단말 노드의 용량 감소로 나타나며, 전체적인 색인의 구축비용 및 검색 성능이 떨어지는 단점이 발생한다. 이때는, 링크데이타들이 가리키는 노드들의 상위 노드가 동일한 경우 링크를 각각 개별 단말 노드로 설정하지 않고 상위 노드로 설정함으로써 링크데이타를 공유할 수 있다. 링크가 가리키는 노드의 레벨이 올라갈수록 링크를 공유할 가능성이 높아지게 되어 노드의 공간 활용도는 높아지나 검색 시 하위 노드 전체를 검색해야 하므로 검색 범위가 늘어나는 문제가 있다. 그러므로 본 논문에서는 링크데이타가 가리키는 노드를 단말 노드 또는 바로 상위 노드로 한정함으로써 검색 영역의 증가를 제한한다. 공유 연결 기법을 적용한 SLR-tree의 단말 노드 구조는 4.2절에서 설명하고 있다.

4.2 단말 노드 구조

단말 노드는 전자태그의 간격 데이타와 다른 노드의 위치정보를 저장하고 있는 링크 데이타, 그리고 각 간격 데이타가 링크 데이타를 가리키기 위한 플래그로 구성된다. 간격 데이타와 전/후 플래그는 서로 대응되는 관계이며 이 플래그를 통해서 전자태그의 전/후 간격이 있는 노드의 위치를 알 수 있다. 단말 노드의 위치정보를 저장하는 링크 데이타에 비해 그 링크 데이타를 가리키는 플래그는 그 크기가 상대적으로 작다. 따라서, 링크 정보를 많이 공유할수록 노드의 공간 활용도는 높아진다. 단말 노드의 구조는 그림 6과 같이 Data1이라는 간격은 회색으로 된 prev값 1을 가지는 전 플래그와 회색으로 next값 3을 가지는 후 플래그로 구성된다. 따라서, 간격 데이타 Data1의 이전 간격은 전 플래그가 가리키는 link1이 가리키는 노드에 존재하고 간격 데이타 Data1의 이후 간격은 후 플래그가 가리키는 link3을 따라가면 찾을 수 있다.

플래그는 링크 데이타의 수를 줄이기 위한 것으로

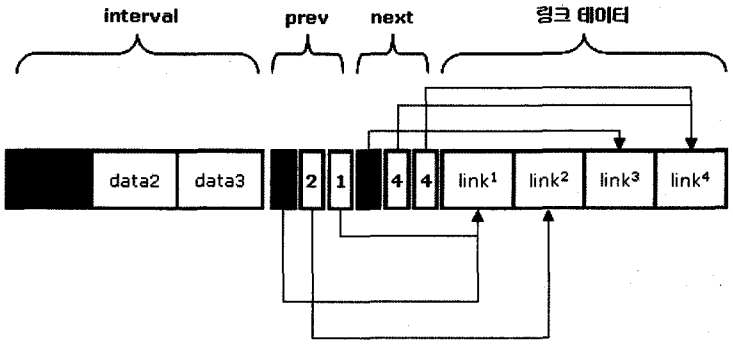


그림 6 Shared Link R-tree의 단말 노드 구조

*None, Self, Link*와 같은 3가지 상태를 가진다. *None*은 자신이 처음이나 마지막 간격으로 다음 간격이 존재하지 않을 때 설정된다. 그리고, *Self*는 자신과 동일한 단말 노드에 간격이 존재할 때 설정되며, 다른 노드에 있을 때 *Link*로 설정된다. 따라서 링크데이타는 플래그가 *Link*일 때 즉, 간격의 전/후 간격이 다른 노드에 존재할 경우에만 사용되며 *None, Self*인 경우에는 추가의 링크 데이타를 가질 필요가 없다.

4.3 데이타 삽입

데이타 삽입과정은 R*-tree와 동일하며 새로운 NI (*NewI*)가 들어갈 단말 노드를 찾는 ChooseSubtree와 삽입 후의 노드 MBB를 조절하고, 분할을 상위 노드로 전파하는 AdjustTree 알고리즘을 그대로 적용한다. 그런데 SLR-tree에서는 단말 노드에 링크 정보를 가지고 있으므로 삽입이 완료되면 새로 삽입된 NI의 이전 FI (*FixedI*)를 찾아서 두 간격에 대한 연결정보를 갱신하여야 한다.

새로운 간격에 대한 이전 간격이 존재하는 경우에는 이전 간격을 포함하는 단말 노드를 찾아야 한다. 이전 간격을 포함하는 단말 노드가 새로운 간격과 동일한 노드이면 플래그를 *Self*로 설정하고 종료한다. 그렇지 않은 경우에는 새로운 링크를 생성해야 하는데 이미 해당 노드를 가리키는 링크가 존재한다면 추가로 링크를 만들지 않고 존재하는 링크를 공유하도록 설정한다. 연결정보를 갱신하기 위한 AdjustLink 알고리즘은 그림 7과 같다.

Algorithm AdjustLink(NI I_{new})

```
1  IF previous interval is not exist
2    I.prev = None;
3  ELSE IF previous interval is in same node
4    I.prev = Self;
5  ELSE
6    Find node n that contains previous interval p
7    IF n has link that pointing here
8      n.p point n.link;
9    ELSE
10   Create link to here and update n.p point to n.link;
11 Repeat 7-10 process with this.node
12 Repeat with next link
```

그림 7 AdjustLink 알고리즘

4.4 데이타 갱신

이전에 삽입된 NI 데이타를 갱신하는 기존의 방법은 NI를 삭제하고, FI를 재 삽입하는 정책을 사용하였다. 이 방법은 삭제를 하기 위해 삭제할 데이타가 있는 단말 노드까지 검색하는 것과, 새로운 데이타를 삽입하는데 추가적인 I/O를 필요로 하기 때문에 색인 구축에 있어서 비효율적일 뿐만 아니라 NI데이타의 삭제로 인하여 연결정보의 재구성 비용과 FI의 삽입 시 다시 연결정보를 재구성하는 비용이 추가적으로 존재한다. 그리고 노드에서 삭제된 FI데이타를 제외한 하나 이상의 다른 NI가 존재하는 경우, 다시 삽입되는 FI가 삭제된 노드로 재 삽입될 가능성이 아주 높다. 그림 8에서 NI 데이타를 삭제하고, FI 데이타를 삽입하더라도 노드 MBB가 변하지 않은 것을 볼 수 있다. 다른 노드로 삽입되는 경우는 그림 9에서 볼 수 있듯이 NI 데이타의 삭제로 인하여 노드 MBB가 축소되는 경우이다. 따라서 FI 데이타의 삽입 시 이전 데이타인 NI의 삭제는 이전 데이타인 NI의 삭제로 인한 단말 노드의 MBB축소가 되는 경우에만 시행하고 그 이외의 경우에는 이전 NI데이타를 FI 데이타로 갱신한다.

데이타의 갱신 알고리즘은 이전 NI를 찾는 FindLeaf와 NI를 제외한 단말 노드의 MBB변화를 측정하는 ChangeMBB로 구성된다. 갱신을 위한 UpdateData 알고리즘은 그림 10과 같다.

4.5 노드 분할

기존의 노드 분할 정책은 공간 객체 색인에서 균등

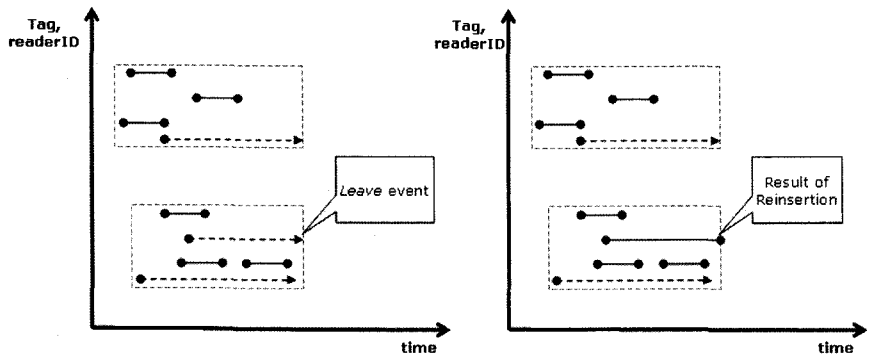


그림 8 같은 노드로 재 삽입 되는 경우

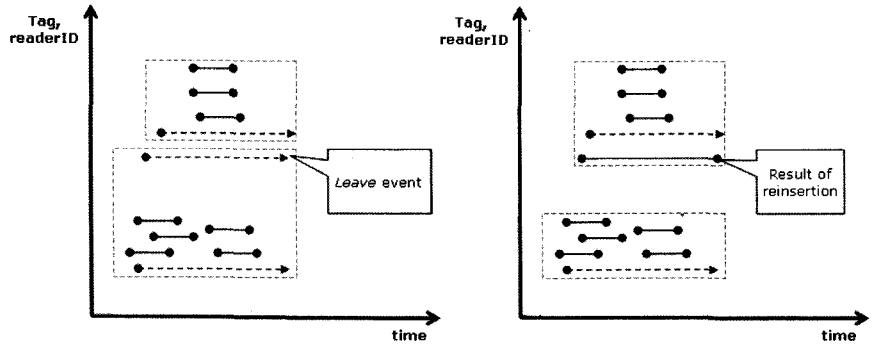


그림 9 다른 노드로 재 삽입 되는 경우

```
Algorithm UpdateData(NI, FI)
1   Node l := FindLeaf(NI);
2   IF l is NULL
3     Return FALSE;
4   ELSE
5     Find entry e in l that contains NI;
6     IF ChangeMBB(NI)
7       DeleteEntry(NI);
8       InsertData(FI);
9     ELSE
10      Update e.I into FI
11      IF l is not root node
12        AdjustTree(l);
13    Return TRUE;
```

그림 10 UpdateData 알고리즘

분할을 수행하여 공간 근접성을 높이는 것에 중점을 두고 있었다. 그러나, 노드 분할에서 균등 분할을 적용할 시 단말 노드에서 구축된 전/후 간격의 링크 데이타의 공유가 더 이상 유효하지 않게 되어 링크 데이타의 증가로 인한 노드의 공간 활용도가 떨어지고 검색 성능이 떨어지는 문제가 발생한다. 연결 정보의 공유를 최대한 유지하면서 분할을 하기 위해서는 분할 될 단말 노드의 링크 데이타 공유 상태를 이용하여야 하며 링크 데이타를 공유하는 엔트리들을 가능한 동일한 노드에 배치하여 분할해야 한다. 단말 노드가 분할 되어 두 개의 노드로 분리 될 때 분할될 단말 노드는 ON, 새로 생성되는 단말 노드는 NN으로 정의하며, ON의 형제 노드를 CN으로 정의한다. ON이 분할될 때 CN의 수가 오버플로우를 일으켜 상위 노드의 분할이 일어나는 경우가 아닐 때에는 ON과 NN이 같은 형제 노드로 구성되며, 이때에는 링크 데이타의 변경이 발생하지 않는다. 따라서, ON의 엔트리를 링크데이타가 CN을 가리키는 간격(L_{cN})과 CN이외의 노드를 가리키는 간격(L_{cAN})으로 구분하여 ON에는 L_{cN}을 할당하고 NN에는 L_{cAN}을 할당한다. 그러나 L_{cN}과 L_{cAN}의 비(L_{ratio})가 극단적으로 차이가 날 경우에는 분할이 제대로 이루어지지 않을 수 있다. 이때는 pr 값에 따라 링크 데이타끼리 집합을 이루어 분할 한다. pr값은 L_{cn}과 L_{cAN}의 L_{ratio}를 나타낸다. pr값에 따라 R^*-tree을 사용할 경우에 L_{cN}과 L_{cAN}중 수가 적은 간격은 R^*-tree의 분할 시 포함하지 않고 분할 된 노드 중 엔트리의 수가 적은 노드로 같이 삽입을 한다.

```
Algorithm SplitLeafNode(GI I, Group g1, Group g2)
1  Set LNum as the number of LcN in node I;
2  Set LANum as the number of LcAN in node I;
3  IF L_ratio < pr
4    Insert LcN into g1;
5    Insert LcAN into g2;
6    For each entry e of g1 and g2
7    Invoke AdjustLink(e);
8  ELSE
9    Invoke LinkSplit(I_new, g1, g2);
```

그림 11 SplitLeafNode 알고리즘

```
Algorithm LinkSplit(GI I_new, Group g1, Group g2)
1  Set L as larger Group;
2  Set S as smaller Group;
3  Invoke RstarSplit(I_new, g1, g2);
4  Insert S to smaller g;
5  For each entry e of g1 and g2
6    Invoke AdjustLink(e);
```

그림 12 LinkSplit 알고리즘

위 분할 방법은 단말 노드에 대한 분할방법이며 비 단말 노드의 분할은 R*-tree의 분할 방법을 그대로 적용한다. 그림 11은 단말 노드의 분할 알고리즘이며, LinkSplit 알고리즘은 그림 12와 같다.

5. 성능 평가

이 장에서는 논문에서 제안하는 색인의 성능을 평가하기 위하여 기존 색인과의 삽입 및 검색 성능을 비교 평가한다. SLR-tree는 R*-tree를 기반으로 하고 과거 이력 검색을 위해 이를 확장한 색인이며, 비교대상이 될 수 있는 기존의 RFID색인인 TPIR-tree 또한 알고리즘은 리더인식영역에 머물러 있는 태그 간격인 현재 간격(*nowI*)에 대한 고려를 제외하면 R*-tree와 동일하다. 그러므로, 본 논문에서 제안하는 SLR-tree의 간격 연결 기법을 평가하기 위해 TPIR-tree의 기반 색인인 R-tree계열 색인과의 비교 평가를 통해 기존의 연구와 성능비교를 수행한다.

비교 평가되는 기존의 색인은 데이타 모델이 서로 상이하므로 간격 데이타 모델에 대해 각 색인 고유의 삽입 및 분할 알고리즘을 적용하여 평가한다. 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. 이 논문에서 제안한 색인과 실험에 사용된 색인의 종류는 표 1과 같다.

표 1 실험에 사용된 색인과 색인 인자

색인 종류	색인 약어
R-tree	QUADRATIC
R*-tree	RSTAR
SLR-tree	SLR

5.1 실험 환경

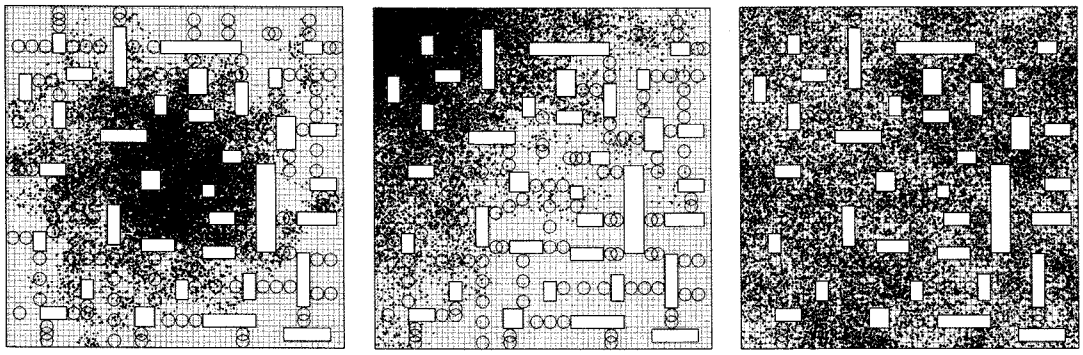
이 논문에서 제안한 SLR-tree는 Microsoft MFC 라이브러리와 Visual C++ 6.0을 사용하여 구현하였고, Windows XP 에서 1GB 메인 메모리, CPU Pentium IV 2.6GHz를 장착한 PC를 이용하여 실험하였다.

색인의 성능을 검증하기 위해서 다양한 환경에서 색인의 성능을 평가할 수 있는 실험 환경이 필요하다. 이동체의 색인 구축 및 실험을 위한 GSTD를 이용하여 다양한 조건에서 실험을 수행한다. 그러나 전자태그의 경우, 이동체와 달리 위치보고가 리더의 인식 영역 안에서만 위치를 보고한다는 차이점이 있으므로 새로운 실험 환경이 필요하다. 이 논문에서는 RFID 환경을 반영하기 위하여 전자태그의 위치보고를 여부를 결정하는 리더를 추가한 전자태그 데이타 생성기(Tag Data Generator, TDG)를 사용하였다. 기본적인 이동체의 위치 이동 알고리즘은 GSTD알고리즘을 사용하지만 위치보고 시점이 리더의 인식영역 안으로 들어갈 때와 밖으로 나갈 때만 데이타를 생성한다. 그림 13은 이 논문에서 제시한 색인의 실험을 위해 사용한 전자태그 위치 생성기에 실험에 사용된 데이타 시공간 분포를 표시한 그림이다. 사각형은 건물의 벽 등 전자태그가 이동할 수 없는 지역이며, 원은 리더의 호출 지역을 나타낸다. 공간상으로 리더의 인식영역의 겹침은 없다고 가정하며 리더는 전자태그가 이동하는 전략적 위치에 설치된다. 그리고, 태그의 위치는 검은색 점으로 표시되어 있는데 각각 균등 분포(Uniform distribution), 사향 분포(Skewed distribution), 가우시안 분포(Gaussian distribution)이며 인자를 조절하여 다양한 전자태그의 위치 데이타를 생성할 수 있다

실험에서 사용된 데이타 집합의 종류는 표 2와 같다. 그리고, 실험에 사용된 질의는 태그 이벤트 데이타를 검색하기 위해서 FIND, LOOK, HISTORY 세가지 질의

표 2 실험에 사용된 데이타 집합의 종류

종류	초기 분포	전자태그 수	ENTER 수
G1	Gaussian	100	10,000
G2	Gaussian	200	10,000
G3	Gaussian	500	10,000
G4	Gaussian	1,000	23,000
R1	Uniform	1,000	24,000
S1	Skewed	1,000	24,000



(a) 가우시안 분포 (b) 사향 분포 (c) 균등 분포

그림 13 실험에 사용된 3가지 데이타 분포

를 선정하였다.

5.2 색인 구축 성능 비교

색인 구축 성능을 비교하기 위하여 전자태그의 이동을 간격의 형태로 저장하는 R-tree와 R*-tree를 같이 비교하였다. 그림 14에서는 TDG로 생성한 데이타의 종류에 따른 색인 구축 비용을 보여준다. R-tree계열의 각 색인에서 노드의 크기는 1024바이트이고 단말 노드의 최대 엔트리 수는 26, 비 단말 노드의 최대 엔트리 수는 18이다. SLR-tree 색인의 단말 노드의 최대 엔트리 수는 21에서 26까지의 가변크기를 가지며 비 단말 노드의 최대 엔트리 수는 18이다. 구축 비용은 데이타 삽입과 갱신에서 발생한 노드 접근 횟수를 합한 값이다. SLR-tree는 각 간격의 삽입 시 전/후의 간격과의 연결 정보를 갱신하고 분할 시에는 분할되어 새롭게 생성되는 노드에 대한 연결 정보 갱신 비용 때문에 상대적으로 높고 R*-tree는 노드 오버플로우가 발생할 시에 노드를 분할하지 않고 데이타 재 삽입을 수행하기 때문에 색인 구축 비용이 R-tree보다 높다.

5.3 질의 성능 비교

이 논문에서 제안한 SLR-tree의 간격 연결 정보 공유 정책의 성능을 측정하기 위해서 기존 R-tree의 quadratic 분할 정책과 R*-tree의 분할 정책을 비교한다. Find와 Look 질의가 전자태그 식별자와 리더 식별자가 범위가 아닌 점 질의를 가정하였고 균등, 사향 분포를 사용하였고 각 데이타 집합에서 질의 성능을 비교한다.

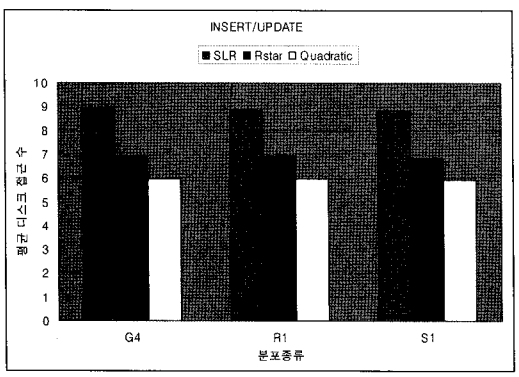


그림 14 데이타 분포에 따른 색인 구축 비용

5.3.1 FIND 질의의 성능 비교

그림 15는 Find 질의를 수행한 결과이며 R*-tree의 성능이 가장 우수했고 R-tree가 가장 낮은 성능을 보였다. R*-tree가 높은 성능을 보이는 이유는 분할 축을 선택할 시에 데이타 겹침이 없는 식별자 축을 선택하기 때문이다. 반면에 R-tree의 경우 전자태그 식별자축과 시간 축보다 공간 축에서 분할이 자주 일어나기 때문에 낮은 성능을 보인다. SLR-tree의 경우 기본적으로 R*-tree의 분할정책을 사용하기 때문에 거의 비슷한 성능을 가진다.

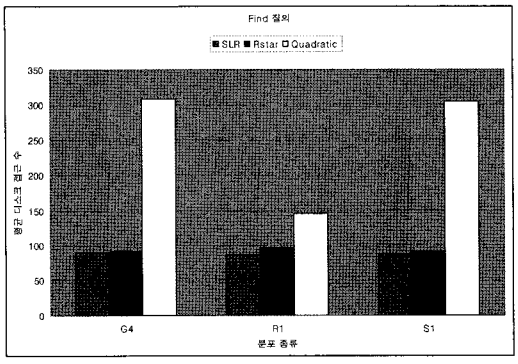


그림 15 각 분포 형태에서의 FIND 질의 성능 비교

5.3.2 LOOK 질의의 성능 비교

LOOK 질의 시의 성능 평가는 FIND의 질의의 성능과 비슷한 향상을 보인다. 그림 16을 보면 성능이 가장 좋게 나오는 것이 R*-tree이고, SLR-tree가 다음으로

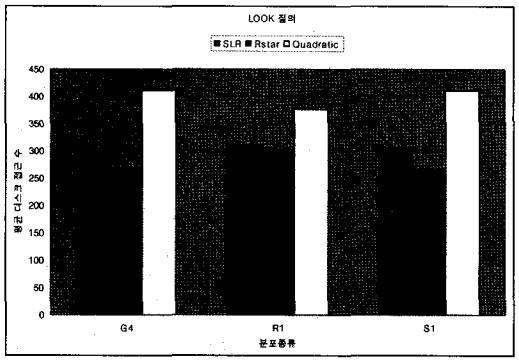


그림 16 각 분포 형태에서의 LOOK 질의 성능 비교

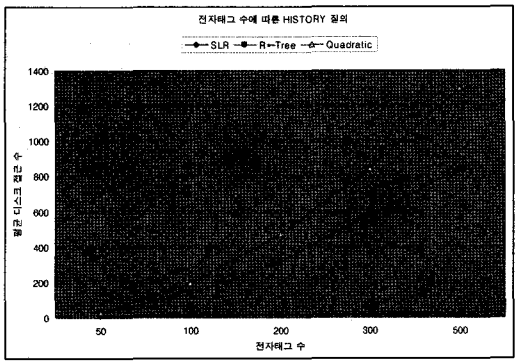


그림 18 전자태그 수에 따른 HISTORY 질의 성능 비교

성능이 좋은 것으로 평가되었다. 이는 같이 이동하는 전자태그의 경우, 노드의 오버플로우 발생 시 분할 할 때, 같은 연결 정보를 가지는 것끼리 분할하는 정책을 사용하여 전자태그의 식별자 값에 관계없이 같은 노드에 존재하기 때문이다.

5.3.3 HISTORY 질의의 성능 비교

그림 17과 같이 HISTORY 질의 시의 성능 평가는 SLR-tree가 월등히 성능이 좋게 나타난다. R-tree와 R*-tree가 HISTORY 질의 시 시간 및 리더의 도메인 전체로 질의를 수행하는 방식에 비하여 SLR-tree의 경우 각 단말 노드의 연결 정보를 이용하여 바로 전/후의 전자태그의 간격 데이타가 있는 단말 노드로 접근이 가능하다. 또한 가우시안 분포와 사향 분포처럼 방향성을 가지고 같이 이동하는 경우에는 R*-tree가 R-tree보다 좋은 성능을 보이고 있다.

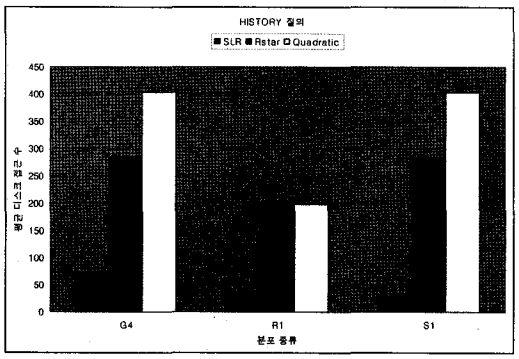


그림 17 각 분포 형태에서의 HISTORY 질의 성능 비교

5.3.4 전자태그 수에 따른 비교

전자태그의 수가 많아지면 데이타가 증가하게 되고 따라서, 이를 저장하는 노드의 수가 늘어나게 된다. 그림 18은 전자태그 수에 따른 HISTORY 질의의 성능을 보여준다. SLR의 경우 는 간격의 연결 정보를 이용하여 단말 노드를 직접 접근하여 데이타를 검색하기 때문에 데이타의 양과는 별개로 일정한 성능을 보임을 알 수 있다. R-tree, R*-tree의 경우 태그 객체 수가 증가할수록 급격하게 성능이 떨어짐을 알 수 있다.

6. 결론 및 향후 연구

이 논문에서는 기존 이동체와 RFID 시스템의 전자태그 환경의 특징에 대해 분석하고 기존 색인의 높은 이력 검색 비용문제를 해결하기 위해 전자태그의 간격간의 연결정보 유지 기법과 색인 구조를 제시하였다. 전자태그 이동은 그 궤적이 연결되어 있지 않은 시간에 평행한 간격의 형태로 나타나게 되고 따라서, 전자태그의 전/후의 위치를 검색하기 위해서는 간격 전/후의 시간 도메인과 리더 도메인 전체에 대하여 검색을 수행하여야 하는 문제점이 발생한다. 전자태그의 이력 추적을 지원하기 위해서 각 전자태그 간격의 전/후 간격의 연결정보를 유지하는 기법을 제시하였으며 이 기법에서 연결정보의 추가로 인한 노드의 공간 활용도의 저하를 최소화하기 위하여 플래그를 사용한 연결정보의 공유 기법을 제안하였다. 플래그를 적용함으로써 연결정보를 공유하고 또한 불필요한 연결정보 생성을 억제함으로써 노드 공간 활용도를 높였다. 그리고, 이 연결정보의 공유기법을 이용한 R*-tree 기반 색인인 SLR-tree를 제안하였으며, 간격 삽입 시의 연결정보 유지 기법 및 단말 노드에서 공유 정보의 유지를 위한 분할 정책을 제안하였다.

이 논문에서 제안하는 SLR-tree의 성능을 검증하기 위해 기존의 RFID색인이 기반하였던 R-tree계열 색인과 실험을 통해 성능을 비교하였다. 실험결과 색인 구축 성능은 연결정보를 유지하는 만큼 다소 높게 나타났지만 FIND 및 LOOK질의는 R*-tree와 유사한 성능을 보인 반면 전자태그의 이력 검색을 위한 HISTORY질의에서는 매우 높은 성능을 나타내었다. 또한, 연결정보를 통해 이력을 바로 탐색할 수 있으므로, 전자태그 수가

증가하더라도 고른 성능을 보였다. RFID관련 응용에서는 태그 이력 질의는 아주 빈번하게 요구되는 질의인 만큼 제안한 색인이 기존 연구보다 우수하다는 사실을 실험을 통해 검증하였다.

향후 연구로서 색인의 구축 비용을 절감을 위한 삽입 정책과 FIND 및 LOOK 질의 성능을 보다 향상시키기 위한 데이타 삽입 정책과 분할 정책에 대한 연구가 필요하다. 그리고 특정 리더에 오래 머물러 있어서 긴 시간 간격을 가지는 데이타는 심한 노드 겹침을 일으키고 질의의 성능을 저하시키므로 효율적으로 저장할 수 있는 방법이 필요하다.

참 고 문 헌

[1] S. E. Sarma, S. A. Weis, and D. W. Engels, "RFID Systems and Security and Privacy Implications," *Springer-Verlag*, pp.454-469, 2002.

[2] N. Beckmann and H. P. Kriegel, "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles," *Proc. ACM SIGMOD*, pp.332-331, 1990.

[3] A. Guttman, "R-Trees: A dynamic index structure for spatial searching," *In Proc. of the 1984 ACM SIGMOD Intl. Conf. on Management of Data*, pp.47-57, June 1984.

[4] Y. Theodoridis, M. Vassilakopoulos, 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.

[5] D. Pfoser, C. S. Jensen, and Y. Theodoridis, "Novel Approaches to the Indexing of Moving Objects," *In Proc. of the 26th VLDB Conf*, pp.395-406, 2000.

[6] M. A. Nascimento and J.R.O. Silva, "Towards historical R-Trees," In Proc. of the 1998 ACM Symposium on applied Computing, pp.235-240, February 1998.

[7] M. A. Nascimento, J. T. O. Silca, and Y. Theodoridis, "Evaluation of access structures for discretel moving points," Proc. Of Int'l Workshop on Spatio-Temporal Database Management, pp.171-188, 1999.

[8] C. Kolovson and M. Stonebraker, "Segment Indexes: Dynamic Indexing Techniques for Multi-Dimensional Interval Data," *Proc. ACM SIGMOD*, pp.138-147, 1991.

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

[10] B. CheaHoon. BongHee Hong, "Time Parameterized Interval R-tree for Tracing Tags in RFID Systems," *16th International Conference, DEXA 2005* (pp. 503-513).

[11] K. Romer, T. Schoch, "Infrastructure Concepts for Tag-Based Ubiquitous Computing Applications," Workshop on Concepts and Models for Ubiquitous Computing, Ubicomp 2002, Goteborg, Sweden, September 2002.

[12] Mark Harrison, "EPCTM Information Service-Data Model and Queries," 2003.

류 우 석

1997년 부산대학교 컴퓨터공학과 졸업(공학사). 1999년 부산대학교 대학원 컴퓨터공학과 졸업(공학석사). 2002년~현재 부산대학교 대학원 컴퓨터공학과 박사과정. 관심분야는 지리정보시스템, 시공간 데이타베이스, RFID 미들웨어, 플래시 메모리 파일시스템

안 성 우

1999년 부산대학교 컴퓨터공학과 졸업(공학사). 2001년 부산대학교 대학원 컴퓨터공학과 졸업(공학석사). 2001년~현재 부산대학교 대학원 컴퓨터학과 박사과정. 관심분야는 지리정보 시스템, RFID 미들웨어, 모바일 GIS, 이동체 색인

홍 봉 희

1982년 서울대학교 전자계산기공학과 졸업(학사). 1984년 서울대학교 전자계산기공학과 졸업(석사). 1988년 서울대학교 전자계산기공학과 졸업(박사). 1987년~현재 부산대학교 공과대학 컴퓨터공학과 교수/부산대학교 컴퓨터 및 정보통신연구소 책임연구원. 관심분야는 이동체 데이타베이스, 모바일 데이타베이스, 공간 데이터베이스

반 재 훈

1997년 부산대학교 컴퓨터공학과 졸업(공학사). 1999년 부산대학교 대학원 컴퓨터공학과 졸업(공학석사). 2006년 부산대학교 대학원 컴퓨터공학과 졸업(공학박사). 2002년~현재 경남정보대학 인터넷응용계열 교수. 관심분야는 데이타베이스, 공간 데이타베이스, 이동체 데이타베이스, RFID

이 세 호

2004년 부산대학교 컴퓨터공학과 졸업(공학사). 2006년 부산대학교 대학원 컴퓨터공학과 졸업(공학석사). 2006년~현재 삼성전자 정보통신총괄 무선사업부 연구원 관심분야는 RFID 미들웨어, 이동체 데이타베이스, 메인 메모리 데이타베이스