■ 2007년도 학생논문 경진대회 수상작

RFID 태그 데이타 색인의 질의 성능 향상을 위한 불균형 삽입 정책

(Disproportional Insertion Policy for Improving Query Performance in RFID Tag Data Indices)

김 기 홍 † 홍 봉 희 †† 안 성 우 †
(Gihong Kim) (Bonghee Hong) (SungWoo Ahn)

요 약 RFID 기술을 기반으로 한 자동화 제조, 재고 관리, 공급망 관리와 같은 응용에서 RFID 태그를 부착한 객체의 위치를 추적하는 질의는 가장 중요한 요구사항 중의 하나이다. 태그의 위치추적 질의를 지원하기 위해서 기존의 연구에서는 태그 아이디, 리더 아이디, 시간을 도메인으로 하는 색인을 제시하고 있으며 이는 이동체 색인을 기반으로 한다. 이동체 색인에서와 달리 RFID 태그를 위한 색인의 도메인은 도메인 간의 크기 차이가 매우 크며 질의 영역이 차지하는 크기의 비율이 리더 아이디 도메인에 편중되는 특징이 있다. 그러나, 기존의 RFID 태그를 위한 색인에서는 이동체 색인과는 다른 도메인의 특징을 고려하지 않음으로써 질의 영역과 색인 노드 간의 불필요한 겹침을 유발시키며 이로 인해 태그 객체 검색 시 많은 노드 접근이 발생하게 되는 문제점이 있다. 본 논문에서는 이러한 문제점을 해결하기 위해 R*-tree를 기반으로 한 RFID 태그 데이타 색인의 불균형 삽입정책과 분할 정책을 제안한다. 제안된 방법은 각 도메인의 가중치와 노드의 가장자리 정보를 사용하여 가중치가 적용된 가장자리 값을 구한다. 데이타를 삽입할 때 이를 사용함으로써 데이타가 삽입될 하위트리를 선택하며 노드 분할 방법을 선택한다. 제안된 불균형 삽입 정책은 질의 영역과 MBR 간의 겹침을 줄임으로써 영역질의 수행 시 노드 접근 비용을 감소시켜 준다. 실험 결과 이 논문에서 제안된 불균형 삽입 정책을 적용한 색인은 기존의 삽입 정책을 사용하는 색인에 비하여 우수한 질의 성능을 보여줌을 확인할 수 있다.

키워드 : RFID, 전자태그, 위치추적, 다차원 색인

Abstract Queries for tracing tag locations are among the most challenging requirements in RFID based applications, including automated manufacturing, inventory tracking and supply chain management. For efficient query processing, a previous study proposed the index scheme for storing tag objects, based on the moving object index, in 3-dimensional domain with the axes being the tag identifier, the reader identifier, and the time. In a different way of a moving object index, the ranges of coordinates for each domain are quite different so that the distribution of query regions is skewed to the reader identifier domain. Previous indexes for tags, however, do not consider the skewed distribution for query regions. This results in producing many overlaps between index nodes and query regions and then causes the problem of traversing many index nodes. To solve this problem, we propose a new *disproportional insertion and split policy* of the index for RFID tags which is based on the

· 이 논문 또는 저서는 2007년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(지방연구중심대학육성사업/차세대물류IT기술연구사업단)

† 학생회원 : 부산대학교 컴퓨터공학과
buglist@pusan.ac.kr
swan@pusan.ac.kr
†† 종신회원 : 부산대학교 컴퓨터공학과 교수
bhhong@pusan.ac.kr

논문접수 : 2007년 7월 26일
심사완료 : 2008년 5월 15일

Copyright@2008 한국정보과학회 : 개인 목적이나 교육 목적인 경우, 이 저작물의 전체 또는 일부에 대한 복사본 혹은 디지털 사본의 제작을 허가합니다. 이 때, 사본은 상업적 수단으로 사용할 수 없으며 첫 페이지에 본 문구와 출처를 반드시 명시해야 합니다. 이 외의 목적으로 복제, 배포, 출판, 전송 등 모든 유형의 사용행위를 하는 경우에 대하여는 사전에 허가를 얻고 비용을 지불해야 합니다.
정보과학회논문지 : 데이타베이스 제35권 제5호(2008.10)

R*-tree. For efficient insertion of tag data, our method derives the weighted margin for each node by using weights of each axis and margin of nodes. Based the weighted margin, we can choose the subtree and the split method in order to insert tag data with the minimum cost. Proposed insertion method also reduces the cost of region query by reducing overlapped area of query region and MBRs. Our experiments show that the index based on the proposed insertion and split method considerably improves the performance of queries than the index based on the previous methods.

Key words : RFID, RFID tag, track and trace, multidimensional index

1. 서 론

RFID(Radio Frequency Identification)는 각종 물품에 소형 칩을 부착해 사물의 정보와 주변 환경정보를 무선주파수로 전송·처리하는 비접촉식 자동 인식시스템이다. 이러한 RFID 는 빠른 인식 속도, 인식률, 데이타 저장능력으로 인해 여러 응용분야에서 주목 받고 있는 기술이다. RFID의 응용 분야는 재고관리, 물류의 공급망 관리, 공장 자동화 등으로 다양하게 확산되고 있다[1-3]. RFID 기술은 생산자에게는 상품의 판매 상황을 실시간으로 정확한 파악이 가능하게 하며 소비자에게는 상품의 유통 경로나 상품정보에 대한 검색을 가능하게 한다. 또한 현재 상품의 재고를 파악하여 부족한 물품을 자동 재입고하여 기업의 이익을 확대시킨다. 이러한 여러 응용환경에서는 RFID 리더(reader)로부터 수집되어 저장되는 대량의 RFID 태그 이벤트를 저장, 검색이 가능한 시스템이 필요하며 저장과 검색을 효율적으로 처리하기 위해 RFID 시스템 내부에 효율적인 색인 기법이 필요하다[4].

이러한 RFID 태그 이벤트의 효율적인 저장과 검색을 위한 기존 색인으로 TPIR-tree(Time Parameterized Interval R-tree)[4]가 있다. TPIR-Tree는 기존 이동체의 시간의 흐름에 따른 위치 정보를 색인 하는 방법을 기반으로 RFID 태그 객체의 위치 특성을 고려하여 저장 및 질의 처리를 효과적으로 처리하기 위한 색인이다. TPIR-Tree의 경우 삽입 및 분할 정책은 R*-Tree[5]의 방식을 기본적으로 따르고 있으며 RFID 데이타를 간격 데이타로 모델링하여 RFID 환경에 적합한 색인 구조를 제시하고 있다.

기존의 이동체 색인에서는 공간 좌표(x, y)와 시간(t)으로 구성된 3차원 도메인 공간 상에 연속된 세그먼트로 이동체의 시간에 따른 위치 정보를 저장하지만 RFID를 위한 색인인 TPIR-Tree의 경우 태그 식별자 tid, 리더 식별자 rid, 시간 t로 구성된 3차원 도메인에서 모델링된 태그 궤적을 저장한다. RFID 색인에서는 이동체 색인의 실제좌표계인 x-y 좌표계와 달리 리더가 가지는 논리적 위치를 질의의 인자로 가진다. 또한 "Tag#10이 있었던 리더는?"과 같이 특정 전자태그 값으로 질의가 수행된다. 즉, 실좌표계가 아닌 리더의 번호와 전자태그의 번호를 도메인으로 가진다[4].

이러한 RFID 색인 도메인은 다음과 같은 특징을 가진다. 첫째, RFID 색인의 도메인인 rid와 tid 도메인은 크기차가 크다. EPCglobal의 태그 코드 체계인 EPC(Electronic Product Code)[6]의 경우 96-bit, 128-bit 또는 256-bit로 이루어져 있다. 96bit 코드 체계일 경우 표현되는 도메인의 크기는 296까지 가능하다. 그에 비해 rid의 경우 tid의 크기에 비해 훨씬 작은 수가 된다. 둘째, 이러한 특징으로 인해 영역 질의의 모양이 한쪽 축으로 긴 불균형 영역 질의의 발생이 빈번하다. 기존 이동체나 공간 색인을 이루는 R-Tree[7] 방식의 경우 정사각형(3차원에서는 정육면체) 질의에 최적화된 성능을 보인다[8]. 그러나 RFID 환경에서는 질의를 정사각형이라고 볼 수 없고 질의 영역의 불균형한 특징이 나타난다.

따라서, 최소 면적 확장 정책을 사용하는 기존의 이동체 색인의 방법을 그대로 사용했을 경우 질의 영역과 생성된 노드간의 겹침이 심하게 되어 색인 검색 비용이 증가하는 문제가 발생한다. 또한 둘레의 길이를 최소화하는 분할 정책 즉 정사각형에 가까운 MBR을 만드는 분할 축 선정방법 및 겹침 최소화 정책을 RFID 환경에 그대로 적용했을 경우 똑같은 문제가 발생한다. 본 논문에서는 이러한 데이타 모델의 변경으로 인해 발생하는 문제를 분석하고 R*-tree의 삽입 기법을 기본적으로 따르는 TPIR-tree의 삽입 및 분할 방법을 개선한 불균형 확장 정책을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구를 소개하고, 3장에서는 대상 환경 및 기존 삽입 방법의 문제점을 정의한다. 4장에서는 도메인의 차이를 고려한 삽입 정책 및 분할 정책을 제시한다. 5장에서는 실험결과에 대해 살펴보고 마지막으로 6장에서 결론 및 향후 연구를 기술한다.

2. 관련 연구

이 장에서는 이동체 색인의 모델을 RFID 색인의 모델로 적용했을 때 문제점을 기술하기 위해 이동체 색인을 기반으로 한 RFID 색인인 TPIR-tree[4]에 대해 기술한다. 본 논문에서 대상으로 하는 이동체 도메인과 RFID 도메인의 차이점과 그에 따른 문제점을 분석하기 위해 TPIR-tree에 관한 연구를 기술하도록 한다.

2.1 RFID 색인에 관한 연구

TPIR-tree[4]는 이동체 색인을 기반으로 RFID 환경

의 데이타 모델을 적용한 색인이다. 태그와 이동 객체는 시간에 따라 위치가 변하는 유사한 특징을 가지므로 태그의 궤적을 추적하기 위한 색인은 이동 객체의 색인을 사용하여 구축할 수 있다. 기존 이동체 색인의 방법에서는 x, y를 색인의 축으로 사용했지만 TPIR-tree에서는 x, y를 rid로 사상하여 사용한다. 이동체 색인에서는 이동체가 유클리디언 공간상에 이동하는 것에 비해 RFID 환경에서는 태그가 어떤 리더에 읽혔을 때만 기록되게 된다. 이동체 색인에서는 (x1, y1, t1)에서 (x2, y2, t2)로 이동체가 움직였을 때 그 사이에 t1<t3<t2, x1<x3<x2, y1<y3<y2인 (x3, y3, t3)가 의미가 있지만 RFID 환경에서 만약 x, y 공간을 사용하게 되면 1번 리더에 읽힌 후 2번 리더에 읽혔다고 했을 때 그사이에 있는 3번 리더에 꼭 있었다라고 말할 수 없다. 이러한 특징은 항상 태그의 위치를 알 수 없고 리더에 읽혔을 당시에만 그 위치를 확인 할 수 있는 RFID 환경의 특징에서 기인한다[13]. 이러한 특징으로 인해 RFID 환경에서는 x, y 대신에 rid를 축으로 사용하고 그림 1과 같이 태그 이벤트를 정의한다[4]. 그림 1의 (a)의 경우 리더의 인식영역 안에 태그가 들어오는 이벤트를 *Enter Event*로 정의하고 태그가 나가는 이벤트를 *Leave Event*로 정의하고 있다. 그러한 이벤트를 기반으로 그림 1의 (b)와 (c)와 같이 하나의 태그 궤적으로 표현하고 있다. 그림 1의 (b)의 경우 질의 시간이 태그의 *Leave Event*가 발생하기 전일 경우 동적 구간(dynamic interval)으로 표현하고 리더에서 나오는 *Leave Event*가 발생했을 때 그림 1의 (c)와 같이 정적 구간(static interval)으로 표현하였다. 이때 동적 구간은 *Leave Event*가 발생하지 않아 선분의 끝점을 알 수가 없어 한쪽으로 열리게 된다. 그에 비해 정적 구간은 *Leave Event*가 발생하여 선분의 오른쪽 끝점이 닫힌 선분을 의미한다.

아래 그림 2는 TPIR-tree에서 정적 구간과 동적 구간 데이타를 이용하여 색인을 구성하는 예제이다. 그림 2를 보면 정적 구간 데이타만을 포함하는 비단말 노드

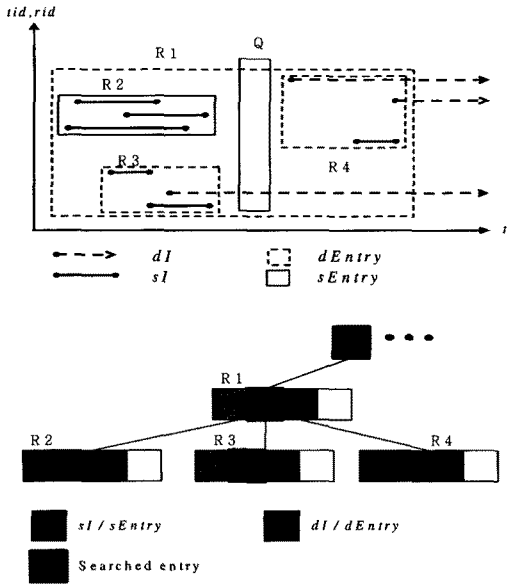


그림 2 TPIR-tree의 구성 예

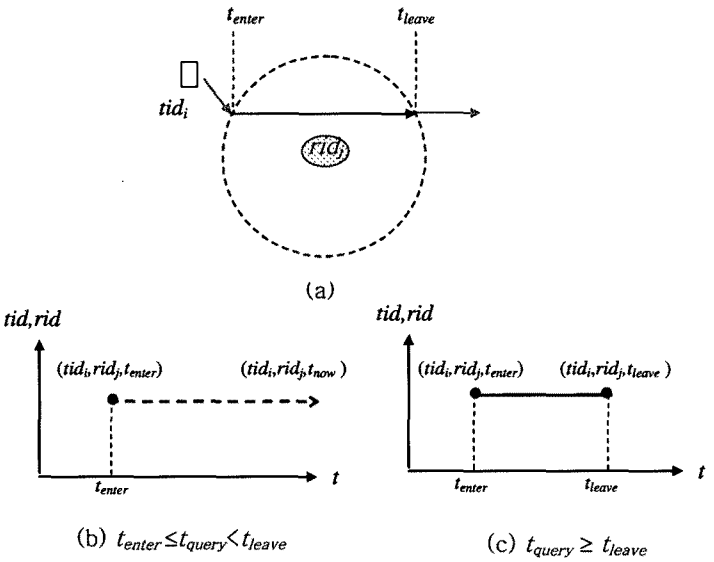


그림 1 TPIR-tree의 동적 구간과 정적 구간 데이타

를 *sEntry*로 정의하고 있으며 동적 구간 데이타를 하나라도 포함하고 있으면 *dEntry*로 표현하고 있다. 또한 이러한 데이타가 색인 상에서 표현되는 방법을 보여 주고 있다.

2.2 영역질의에 대한 비용에 관한 연구

R-tree에서 영역질의 $q = q_x \times q_y$에 대해 검색 확률 또는 검색 비용을 다음과 같은 식 (1)으로 나타낸다[8].

$$P(q_x, q_y) = TotalArea + q_x * L_y + q_y * L_x + N * q_x * q_y \quad (1)$$

q_x는 영역 질의 q의 x축 변 q_y는 y축 변을 의미하면 L_x 및 L_y는 각각 R-tree를 이루는 MBR들의 x축 및 y축의 총합의 의미한다. 여기서 확인 할 수 있는 점은 면적 외에도 노드의 둘레길이(margin)를 최소화 하는 것이 검색 비용을 감소시키며 특히 질의 영역이 커질수록 둘레길이를 최소화 하는 것이 중요해진다[8]. 이러한 점은 R*-tree에서 둘레길이를 적용하여 대부분의 경우에 좋은 성능을 나타내는 것을 실험적으로 보여준 바가 있다[5]. 또한 공간 질의에 대한 비용 모델에 대한 논문인 [11]의 경우에도 비용 모델을 세우는 과정에 모든 노드는 정방형일 때 가장 좋은 성능을 내는 것으로 가정하고 있다. 이때 식 (1)에 따라 질의 영역 역시 질의 영역의 면적 $q_x * q_y$가 작을수록 P는 작아지고 질의 영역의 둘레길이를 이루는 L_x, L_y가 작을수록 P가 낮아지는 것을 알 수 있다. 즉 정방형 영역질의에 대해 적은 검색 비용이 든다. 결론적으로 R-tree 계열 색인들은 정방형의 노드를 구성해야 하며 질의영역이 정방형일 때 가장 좋은 성능을 낼 수 있다. 그러나 이러한 정방형의 노드 구성일 경우 RFID 환경의 도메인 특징으로 인한 질의 영역의 불균형으로 인해 검색 비용이 증가하게 된다.

3. 대상환경 및 문제정의

3장에서는 본 논문의 대상인 RFID 환경 및 응용에 설명하고 기존 TPIR-tree의 태그 이벤트 모델링에 대해 소개한다. 그리고 이러한 대상환경을 토대로 기존 연구에서 발생하는 문제점을 제시한다.

3.1 대상환경

RFID 시스템은 태그, 리더, 서버로 구성되며 리더는 인식 영역에 들어오거나 나가는 태그의 정보를 수집하여 서버로 전송하고 이러한 정보를 사용하여 자동화 생산(automated manufacturing), 재고 관리(inventory tracking), 공급망 관리(supply chain management) 등과 같은 다양한 응용분야에 적용이 가능하다[1-3]. 예를 들어 그림 3의 경우처럼 물류 관리를 위한 회사에 RFID 시스템을 적용시킬 수 있다. 태그를 배송하고자 하는 제품에 부착하고 각 창고의 입구에 리더를 설치하여 창고 출입 관리가 가능하다.

이러한 응용분야에서는 태그를 장착한 객체의 위치 추적이나 객체의 현재의 상태를 감시하는 서비스를 제공해야 한다. 즉, "제품 1이 현재 어느 창고에 있는가?"라는 태그의 현재 위치 추적이나, "두 시간 전에 창고 B를 나간 제품은 무엇인가?"라는 태그의 과거 위치 추적을 통해 제품의 위치 및 현재 상태를 감시하는 서비스를 제공해야 한다.

이러한 서비스를 제공하기 위한 RFID 색인으로 TPIR-

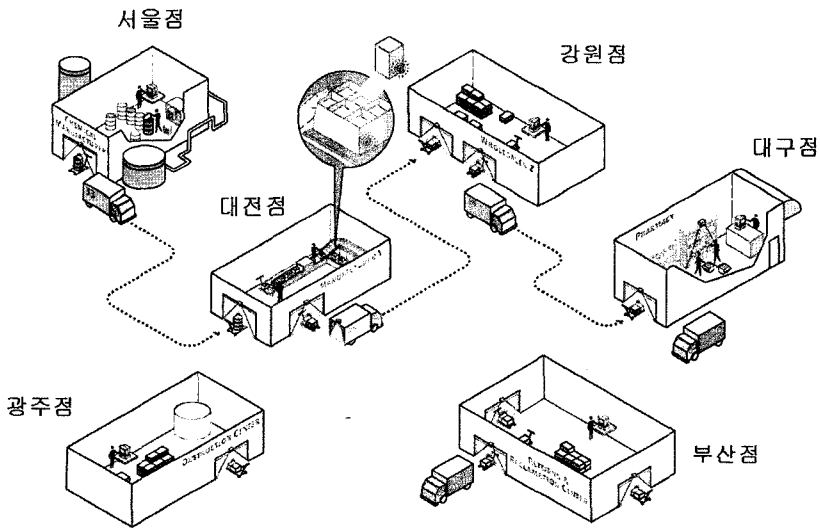


그림 3 RFID 응용 예제 환경

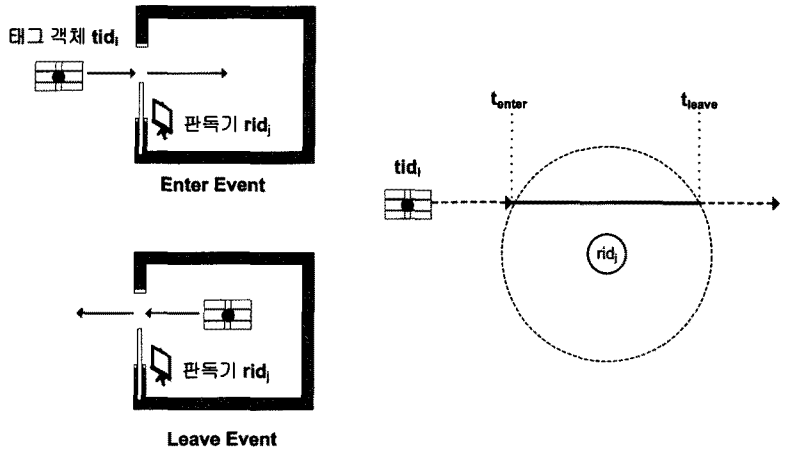


그림 4 태그 이벤트 종류

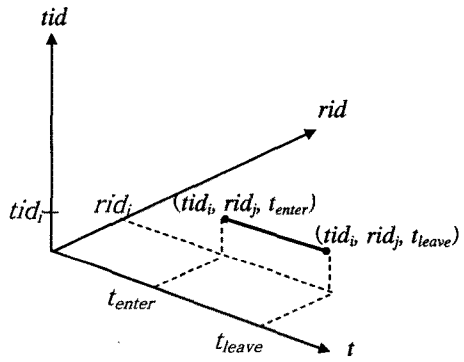


그림 5 TPIR-tree의 태그 이벤트 데이타 모델링

tree[4]가 있으며 그림 4에서 보는 바와 같이 태그의 위치를 색인하기 위해 태그 이벤트를 정의하고 있다. 태그가 리더의 인식영역에 들어갈 때 *Enter Event*가 발생하고 인식 영역을 빠져 나올 때 *Leave Event*가 발생한다.

또한 이러한 태그 이벤트를 기반으로 태그의 궤적을 그림 5와 같이 표현하고 있다. *tid, rid, time*은 3차원 공간의 각 축을 의미하며 *tid*는 태그 객체의 식별자, *rid*는 리더의 식별자, t_{enter}, t_{leave}는 각각 *Enter*와 *Leave Event*의 발생 시간을 의미한다.

3.2 문제정의

기존 이동체 색인을 구성하는 데이타 모델은 공간(*x, y*)과 시간(*time*) 도메인이다. 본 논문의 기반 모델이 되는 RFID 색인인 TPIR-Tree는 RFID 환경에 이동체 색인의 방법을 적용하였다. RFID 색인의 경우 "과거 10:00부터 현재까지 *tid*#10의 이동 경로는?"과 같이 태그 식별자(*tid*)에 대한 질의가 중요해 *tid*가 하나의 축으로 추가되었다. 또한 RFID 환경에서는 공간의 개념이 x, y와 같은 유클리디언 공간(Euclidean space)이 아닌 리더(*rid*)로 이루어진 공간으로 도메인인 x, y를 리더 식별자(*rid*) 축으로 사상한다[4].

이러한 도메인의 변환 과정에서 고려되지 않은 점이 있다.

첫째, *tid* 도메인의 전체크기는 *rid* 도메인의 전체 크기에 비해 훨씬 크다. 이동체의 공간 도메인인 *x, y*는 동일한 유클리디언 공간으로 최대 도메인의 크기가 같지만 그에 비해 RFID 색인의 도메인인 *tid, rid*의 경우 각 도메인의 최대값의 크기 차이가 크다. *tid*의 경우 EPC(Electronic Product Code)[6]로 이루어져 있으며 전체 도메인의 크기는 EPC의 종류에 따라 2^{96}또는 2^{128} 이상의 값을 가진다. 반면 *rid*의 경우 물리적 리더(physical reader)가 아닌 논리적 리더(logical reader)로 정의되며 논리적 리더는 실제 환경에서 하나의 창고 또는 건물로 사상된다[12]. 그러므로 논리적 리더에 해당하는 *rid*의 경우 도메인의 크기가 *tid*에 비해 상대적으로 매우 작은 크기이다. 극단적으로 하나의 물리적 리더가 하나의 논리적 리더로 사상될 때 1m × 1m 크기의 리더를 지구전체에 펼쳐 놓아도 2^{50}을 넘지 못한다. 실제 환경에서의 *rid*의 개수는 *tid*에 비해 훨씬 더 적은 수가 될 것이다.

둘째, 영역 질의 시 *tid*의 질의 영역에 비해 *rid*의 영역 질의의 크기가 훨씬 크다. 두 번째 특징은 첫 번째 특징이 원인이 되어 발생하는 특징이다. 다음과 같은 "100.100.*의 *tid*가 오전 11:00~12:00시 사이에 1~2의 *rid*중 어떤 *rid*에 존재했는가?"라는 질의가 있을 때 *tid*

표 1 *tid, rid*의 불균형 영역 질의 예

	전체도메인	질의범위	정규화된 공간상의 질의 영역이 차지하는 비율 (질의범위/전체도메인)
tid	2^{96}	2^{36} (=100.100.*)	$2^{36}/2^{96} = 2^{-60}$
rid	2^{50}	2^{1} (=2)	$2^{1}/2^{50} = 2^{-49}$

와 *rid* 도메인의 질의영역을 비교할 수 있다. 이 때 크기가 다른 도메인들을 비교하기 위해 각 도메인을 $[0,1]^3$의 정규화된 공간(Normalized space)에 표현하면 표 1과 같다.

tid 질의 영역의 경우 전체 *tid* 전체 도메인에서 2^{-60}만큼의 비율을 차지하는 것을 알 수 있으며 *rid* 질의 영역의 비율은 *rid*전체 도메인에서 2^{-49}만큼을 차지하는 것을 알 수 있다. *tid*와 *rid*의 질의 영역의 크기 차이는 2^{11}만큼 나타남을 알 수 있다. 이러한 영역 질의를 *tid, rid, time* 공간상에 표현하면 그림 6과 같다.

그림 6에서 $[0,1]^3$으로 정규화된 공간상에서는 *rid*의 질의 영역의 크기가 *tid* 질의 영역의 크기에 비해 2^{11}배 크다는 것을 알 수 있다. 이처럼 RFID 환경에서는 질의 영역이 정방형이 아닌 질의 영역이 *rid*축으로 긴 직육면체 모양으로 불균형 질의 영역으로 나타남을 알 수 있다.

하지만 기존 방법은 질의 영역이 정방형일 때 최적의 검색 비용이 나타난다[8]. TPIR-Tree는 R*-tree의 색인 기법을 기본적으로 따르고 있으며 R*-Tree 의 경우 면적(area), 중첩(overlap) 및 둘레길이(margin)를 작게 하는 방식을 선택한다. 면적이 작을수록 질의 시 MBR이 질의영역과 겹칠 확률을 줄여 주며 상위 노드에서의 중첩(overlap)이 작을수록 서브트리를 순회해야 하는 비용을 줄여 검색 성능이 높아진다. 또한 둘레길이(margin)을 적게 하면 각 MBR이 정사각형에 가까운 모양이 나타나게 한다. 같은 면적의 MBR이라도 둘레길이가 적은 MBR 즉 정사각형에 가까운 MBR 일수록 질의 영역이 정사각형에 가깝게 커질 때 질의 검색 비용을 감소 시킨다. 즉 R*-tree의 둘레길이(margin)을 적게 하는 정책은 질의가 영역 질의이며 정사각형에 가까운 질의라는 것을 가정하고 있다[8,9]. 이렇게 기존 R*-tree 기반의 공간색인 또는 이동체 색인의 경우 질의 영역을 정방형이라는 가정을 하고 있으며 정방형의 질의 영역에 최적화된 성능을 나타낸다. 반면 RFID 환경에서는 도메인간의 상이성으로 인해 *tid, time*에 비해 *rid* 방향으로 긴 직육면체 모양 즉 불균형 영역 질의로 나타난다.

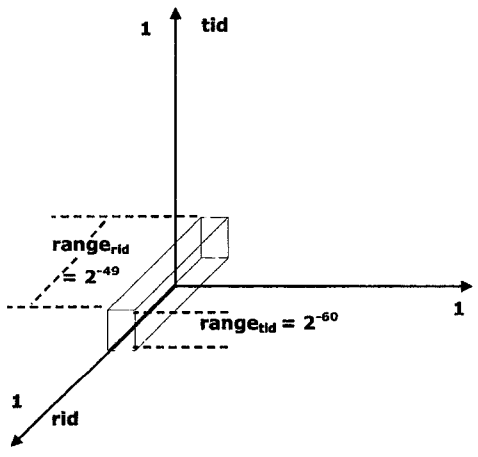


그림 6 RFID 환경의 불균형 질의 예

그림 7은 *tid*와 *rid*축의 2차원으로 간단하게 표현하였다. *rid* 방향으로 긴 불균형 질의에 대해 거의 유사한 면적을 가지는 그림 7의 (a), (b), (c) 3가지 모양의 MBR에 대해 노드 접근을 비교하였다. 그림 7의 질의는 rid와 tid 영역에서 균등 발생(uniform distribution)이라고 가정했을 때 그림 7의 (a)에서는 질의 영역과 MBR의 겹침이 5회 발생하며 유사하게 (b)에서는 10번, (c)에서는 25번 발생하게 된다. 질의영역과 평행한 MBR인 (a)의 경우에 대해 가장 적은 노드 접근을 보여주며 질의영역과 교차하는 (c)의 경우에 가장 많은 노드 접근이 일어난다. 기존 R*-tree의 방법인 정방형에 가까운 모양을 만드는 (b)의 경우는 면적이 4로 (a) 경우의 MBR의 면적인 5에 비해 더 작은데도 불구하고 (a)의 방법에 비해 많은 노드 접근이 일어남을 알 수 있다.

이러한 도메인의 불균형과 불균형 영역 질의의 특징으로 인해 기존 정방형 질의에 맞춰져 있는 R*-tree의 색인 구축 방법을 따르는 TPIR-tree의 삽입 및 분할 정책은 수정되어야 한다. 기존 방법을 따를 경우 검색 시 MBR과 질의 영역의 겹침이 심해져 노드 접근 횟수가 많아지는 문제점이 있다. 본 논문의 뒷장에서 상이한 특성을 갖는 두 도메인의 특징을 고려하여 불균형 삽입 및 분할 정책을 제안한다.

4. 삽입 및 분할 정책

이 장에서는 전자태그의 데이타 모델에 기반하여, 불균형 영역 질의에 적합한 삽입 및 분할 정책을 제안한다. 삽입 시 질의 영역을 고려한 MBR을 구성하기 위한 방법으로 MBR을 구성하는 육면체의 변의 길이를 이용하는 방법을 제안한다. 또한 유사한 접근 방법으로 분할 정책을 제안한다.

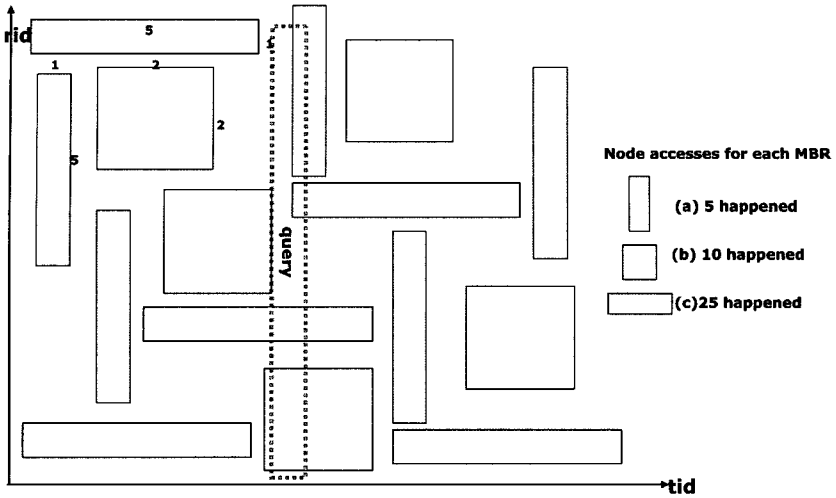


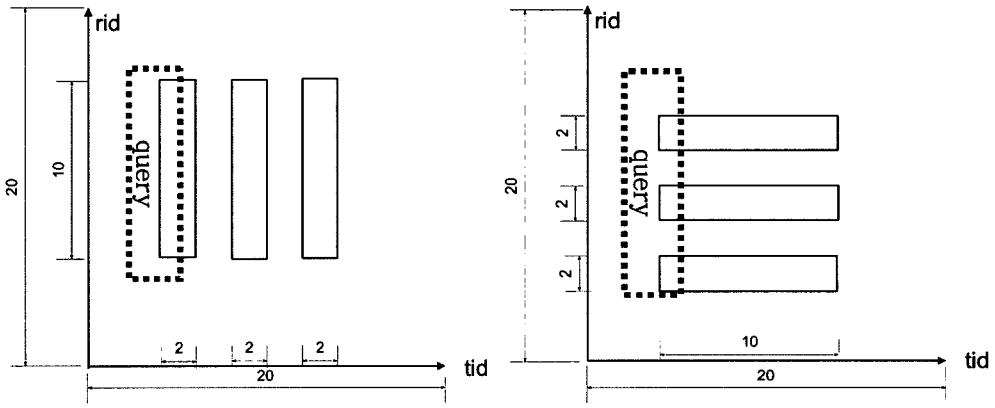
그림 7 불균형 질의에 대한 MBR 모양 별 노드 접근

4.1 기본 접근 방법

그림 8과 같이 질의 영역이 *tid*에 비해 *rid*에 대해 긴 영역일 경우 MBR의 모양에 따라 노드 접근 횟수가 달라진다. 쉬운 비교를 위해서 그림 8의 (a)와 같이 *rid* 방향으로 긴 MBR과 그림 8의 (b)와 같이 *tid*방향으로 긴 MBR의 두 가지 케이스에 대해 표현하였다.

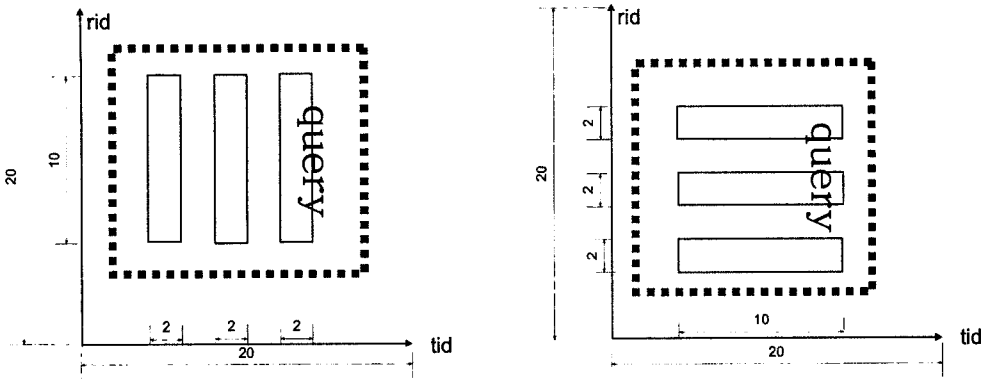
그림 8의 (a)에서와 같이 *tid* 방향으로 긴 MBR의 경우 질의 영역이 *rid* 방향으로 긴 모양일 경우 MBR과 평행한 것을 알 수 있다. 그림 8의 (b)의 경우 *rid* 방향으로 긴 MBR의 경우 질의 영역 역시 *rid* 방향으로 길어 질의 영역과 MBR이 교차한다. 질의 영역과 MBR이 교차하는 경우 한번의 질의에 대해 최소 MBR 3개와 모두 겹치는 것을 알 수 있다. 그에 비해 MBR이 질의 영역과 평행한 경우 같은 크기의 영역 질의에 대해 최소 1개의 MBR이 겹치는 것을 알 수 있다. 하지만 질의가 점 질의일 경우 두 경우의 MBR의 면적은 같으므로 질의에 겹칠 확률은 같게 된다. 또한 그림 9와 같이 질의 영역이 각 경우의 MBR을 다 포함할 정도로 클 경우에는 각 경우에 최대 3번의 노드 접근이 일어나므로 같다고 할 수 있다. 즉 점 질의이거나 질의영역이 전체 도메인에 걸치게 넓은 경우에는 두 경우 모두 최대 3번이 겹치게 되어 질의 영역과 노드의 겹침 확률은 같다고 할 수 있다. 그 외의 질의 영역의 경우에는 질의 영역이 MBR과 평행할 때 최소 1번으로 질의 영역과 MBR이 교차하여 최소 3번일 때 보다 노드 접근 횟수가 줄어드는 것을 알 수 있다.

위에서 살펴본 바와 같이 질의 모양과 닮은꼴 즉 질의 영역과 MBR이 평행한 경우 MBR이 같은 면적을



(a) 질의 영역과 평행한 MBR (b) 질의 영역과 교차하는 MBR

그림 8 질의 영역이 작을 때



(a) 질의 영역과 교차하는 MBR (b) 질의 영역과 평행한 MBR

그림 9 질의 영역이 MBR을 모두 커버할 때

갖더라도 질의와 겹칠 확률이 낮은 것을 알 수 있다. 삽입 시 이러한 특성을 이용하여 MBR을 구성할 때 질의 영역과 평행한 모양의 MBR이 생성되도록 데이타를 삽입하도록 한다. 분할 시에도 비슷한 접근 방법으로 MBR간의 중첩(overlap) 영역의 질의 영역과 평행한 모양이 되도록 MBR을 분할하도록 한다.

4.2 삽입 정책

이번 절에서는 불균형 영역 질의에 대해 질의 영역과 MBR이 평행한 모양이 구성되는 삽입 정책을 제안한다. 먼저 데이타를 삽입 시 서브트리를 찾아가는 방법인 *ChooseSubtree*를 제안하고 *ChooseSubtree* 알고리즘의 내부 알고리즘으로 기존 방법인 최소 면적 확장(Least Area Enlargement) 정책이 아닌 불균형 확장(Disproportional Enlargement) 정책을 제안한다.

4.2.1 ChooseSubtree

TPIR-tree의 삽입 정책은 R*-tree의 삽입 방법을 기본적으로 따르고 있다. 삽입 알고리즘 중에 *ChooseSubtree* 알고리즘을 아래 알고리즘 1과 같이 변경하였다. 기존 *ChooseSubtree* 알고리즘은 비단말 노드를 선택할 때 면적 확장값이 가장 적은 쪽 서브트리를 선택하는 최소 면적 확장(Least Area Enlargement) 정책을 사용한다. 이 방법을 RFID 환경에 그대로 적용했을 경우 한 축으로 긴 MBR을 구별할 수가 없다. 즉 면적만을 고려하므로 질의 영역과 교차하는 MBR을 구성할 수 있어 질의 시 질의 영역과 MBR의 겹침이 심해진다.

따라서 MBR을 구성하는 각 축의 변의 길이에 가중치(weight)를 부여하고 둘레길이(margin)을 이용하는 불균형 확장 정책(Disproportional Enlargement Policy)을 사용한다. 가중치는 질의영역의 *tid*와 *rid* 비율을 이용해 실험으로 구할 수 있다. 다음 절에서 불균형 확장 정책에 대해 기술한다.

Algorithm of ChooseSubtree

CS1 Set N to be the root
CS2 If [Leaf check]
N is a leaf,
return N
else [determine the minimum overlap cost]
If the child pointers in N point to leaves, choose the entry in N whose rectangle needs **least overlap enlargement** to include the new data rectangle Resolve ties by choosing the entry whose rectangle needs least area enlargement,
then
the entry with the rectangle of smallest area
if [determine the minimum Disproportional Enlargement cost]
the child pointers in N do not point to leaves, choose the entry in N whose rectangle needs **Disproportional Enlargement** to include the new data rectangle. Resolve ties by choosing the entry with the rectangle of smallest area
end
CS3 Set N to be the child node pointed to by the child pointers of the chosen entry and repeat from CS2

알고리즘 1 *ChooseSubtree*

ChooseSubtree 알고리즘은 기존 방법과 유사하다. 노드 *N*에서 선택 후보가 되는 엔트리를 선택할 때 하위 엔트리가 단말 노드를 가리킬 경우 기존의 방법과 같이 중첩(overlap) 영역이 가장 작은 하위 엔트리에 삽입하게 된다. 하위 엔트리가 비단말 노드를 가리킬 경우에 기존에는 넓이 확장값이 가장 작은 하위 엔트리에 삽입하였으나 본 논문에서는 불균형 확장값이 적은 하위 엔트리에 삽입하게 된다. 불균형 확장값을 구하는 방법은 다음절에서 살펴보도록 한다.

4.2.2 불균형 확장 정책(Disproportional Enlargement Policy)

삽입 시 질의 영역의 모양과 유사한 닮은꼴 MBR 즉 질의 영역과 MBR이 평행한 MBR을 선택할 때 불균형 확장값이 적은 MBR을 선택한다. 그림 10과 같이 새로 삽입된 데이타를 삽입할 MBR을 선택하기 위해 가중치(weight)를 부여한 각 축의 변의 길이의 합인 MBR의 둘레길이(margin) 즉 가중치를 부여한 둘레길이의 확장값이 적은 MBR을 선택한다. 면적 대신 둘레길이(mar-

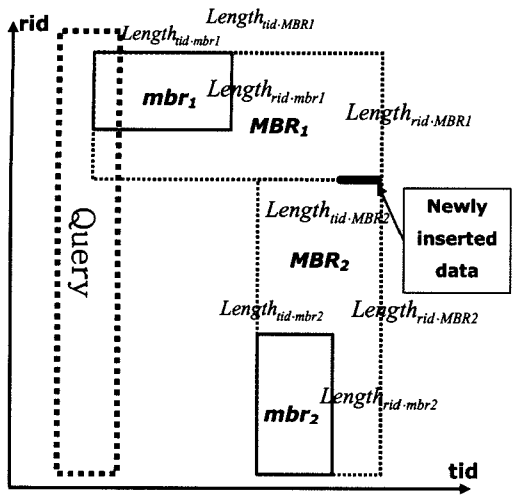


그림 10 데이타 삽입시 *ChooseSubtree*를 위한 변수

gin)를 이용하는 이유는 일반적으로 둘레길이는 면적이 작을수록 작은 값을 가지게 되기 때문이다. 제안된 삽입 정책은 R-tree 계열에서 검색비용을 결정하는 기존의 방법인 면적에 대한 고려 역시 포함하게 된다. 둘레길이를 선택하는 이유는 면적만으로는 MBR의 모양에 대한 정보를 알 수 없기 때문이다.

다음으로 불균형 확장값을 계산하는 방법에 대한 설명이다. 그림 10과 같이 MBR 육면체를 이루는 각 변 중에 *rid* 축에 평행한 변의 길이를 $Length_{rid}$라고 *tid*축에 평행한 변의 길이를 $Length_{tid}$라고 정의 할 수 있다. 또한 *tid*에 대한 가중치를 $Weight_{tid}$라고 하고 *rid*에 대한 가중치를 $Weight_{rid}$라고 정의할 수 있다. 이때 *rid*로 긴 모양의 MBR을 구성하기 위해 *rid*에 부여하는 가중치를 *tid*에 부여하는 가중치값보다 작게 준 뒤에 각 변의 길이와 각 변의 가중치값을 곱한값을 더한 값을 비교하여 더 작은 확장값의 MBR을 선택하는 방법이다.

각 MBR에서 각 도메인과 평행한 변의 길이와 각 도메인의 가중치를 도메인 별로 곱한값을 모두 합한값을 *WeightedMargin*으로 아래 식 (2)와 같이 정의 할 수 있다.

$$WeightedMargin(mbr,d) = \sum_{i=1}^{d} Weight_i \times Length_i$$

이때 MBR에 데이타를 삽입 시 MBR이 확장하게 되는데 이전 MBR을 MBR_{before}라고 하고 확장후의 MBR을 MBR_{after}라고 할 수 있다. 이때 각 MBR의 *WeightedMargin*값을 식 (2)를 통해 구할 수 있고 *WeightedMargin*값의 확장값을 *WeightedMarginEnlargement*라고 정의 할 수 있으며 식 (3)과 같이 표현할 수 있다. 이때 *d*는 차원(dimensionality)를 의미한다.

WeightedMarginEnlargement =
WeightedMargin(MBR_{after}, *d*) - *WeightedMargin*(MBR_{before}, *d*) (3)

그러므로 그림 10의 각 MBR의 *WeightedMarginEnlargement*를 식 (3)을 이용해 표현하면 다음 식 (4) 및 (5)와 같다.

$WeightedMarginEnlargement_{mbr1}$
= *WeightedMargin*(*MBR1*,3) - *WeightedMargin*(*mbr1*,3) (4)

$WeightedMarginEnlargement_{mbr2}$
= *WeightedMargin*(*MBR2*,3) - *WeightedMargin*(*mbr2*,3) (5)

다음으로 그림 11의 예를 사용하여 실제 삽입 시 MBR을 선택하기 위해 *WeightedMargin*을 구하는 방법을 설명한다.

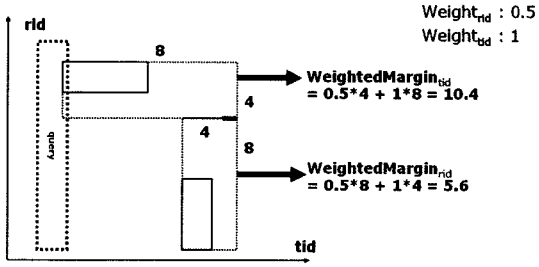


그림 11 *WeightedMargin* 계산 예제

$Length_{tid}$ = $Length_{tid.mbr}$ - $Length_{tid.mbr}$이라고 정의하고 $Weight_{tid}$ = 1, $Weight_{rid}$ = 0.5이라고 가정하면 각 노드의 *WeightedMargin*은 표 2와 같다.

표 2 각 MBR별 *WeightedMargin*의 값 계산

	$Length_{tid}$	$Length_{rid}$	*WeightedMarginEnlargement*
mbr1	8	4	4×0.5+8×1=10
mbr2	4	8	8×0.5+4×1=8

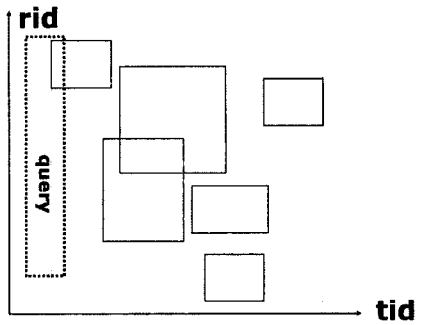
mbr1의 *WeightedMarginEnalrgement*값은 10이고 mbr2의 *WeightedMarginEnlargement*값은 8이므로 mbr1에 비해 더 적게 확장된 mbr2가 삽입할 서브트리가 된다. 즉 면적이 같은 두 MBR에서도 *WeightedMarginEnlargement* 방법을 사용하여 *rid* 방향으로 긴 MBR을 선택할 수 있다.

4.3 분할 정책

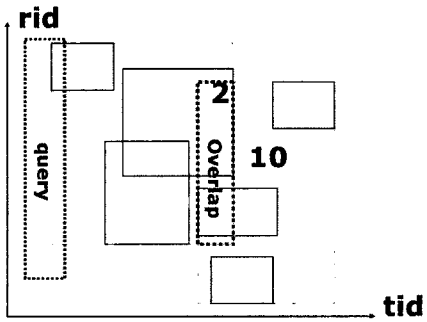
기존의 노드 분할 정책은 분할 시 MBR간의 중첩(overlap)영역의 넓이가 작아지도록 하는데 중점을 두고 있었다. 중첩영역의 넓이를 작게 할수록 검색 시에 질의 영역과 중첩(overlap) 영역이 겹치는 확률을 줄여주는 효과를 가지고 있다. 그렇지 않으면 질의 영역과 중첩영

역이 겹침으로 인해 서브트리를 모두 순회하게 되어 검색 비용이 증가하는 문제가 있다. 한편 앞 절에서 살펴보았듯이 RFID 환경의 질의는 불균형 영역 질의이므로 중첩(overlap) 영역의 넓이가 작기만 해서는 안 된다. 삽입과 마찬가지로 분할 시에도 중첩(overlap) 영역의 넓이가 작을수록 좋고 또한 중첩(overlap) 영역이 질의 영역과 평행한 모양일수록 질의 영역과 중첩영역이 겹칠 확률이 줄어든다.

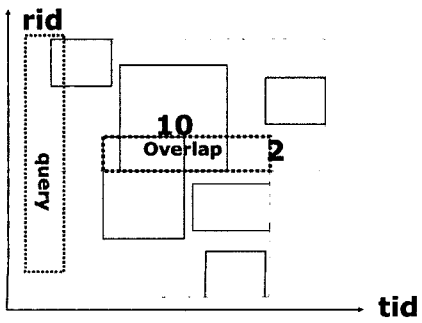
삽입 때 살펴보았던 그림 8의 사각형을 중첩영역이라고 가정하고 살펴보면 그림 8의 (a)의 경우 중첩영역이 *tid* 방향으로 길고 질의 영역이 *rid* 방향으로 긴 모양이므로 중첩영역과 직각으로 교차하는 것을 알 수 있다. 그림 8의 (b)의 경우에는 반대로 *rid* 방향으로 긴 중첩영역의 경우 질의 영역 역시 *rid* 방향으로 길어 질의 영역과 중첩영역이 평행한 것을 알 수 있다. 질의 영역과 중첩영역이 교차하는 경우 한번의 질의에 대해 3개의 중첩영역과 모두 겹치는 것을 알 수 있다. 그에 비해 중첩영역이 질의영역과 평행한 경우 같은 크기의 영역 질의에 대해 최소 1개의 중첩영역과 겹치는 것을 알 수 있다.



(a) overflow 되어 분할될 노드



(b) rid 방향으로 긴 중첩영역



(c) tid 방향으로 긴 중첩영역

그림 12 분할 시 중첩영역의 모양

삽입 시 접근 했던 방법과 마찬가지로 질의 모양과 닮은꼴 즉 질의 영역과 중첩영역이 평행한 경우 중첩영역의 면적이 갖더라도 질의와 겹칠 확률이 낮은 것을 알 수 있다. 분할 시에도 이러한 특성을 이용하여 분할된 두 MBR의 중첩영역이 질의 영역과 평행한 모양의 MBR이 생성되도록 분할되어야 한다.

그림 12의 (a)는 분할되기 전의 MBR이고 그림 12의 (b)는 분할 후 두 MBR의 중첩영역이 질의 영역과 평행하게 분할한 것이다. 그림 12의 (c)는 분할 후 두 MBR의 중첩영역이 질의 영역과 수직이 되게 분할된 것이다. 위에서 살펴보았듯이 그림 12의 (b)의 경우와 같이 분할했을 시에 질의 영역과 중첩영역이 겹칠 확률이 줄어들게 된다.

알고리즘 2는 분할 알고리즘이다. 분할 알고리즘은 기본적으로 R*-tree의 방식과 유사하다. 먼저 기존과 같은 방법의 둘레길이를 이용하여 *ChooseSplitAxis*를 통해 분할이 일어날 축을 선정하고 *ChooseSplitIndex*를 통해 앞의 *ChooseSplitAxis*에서 선정된 분할 축에 따라 가장 좋은 분할을 찾아내게 된다.

Algorithm of Split

S1 Invoke **ChooseSplitAxis** to determine the axis, perpendicular to which the split is performed
S2 Invoke **ChooseSplitIndex** to determine the best distribution into two groups along that axis
S3 Distribute the entries into two groups

알고리즘 2 *Split*

다음은 *ChooseSplitIndex* 알고리즘이다. 기존의 *ChooseSplitIndex* 방법에서는 분할 후 두 그룹의 MBR로 나뉘어졌을 경우 두 MBR간에 중첩영역이 가장 작은 분포가 되는 분할을 수행하였다. *ChooseSplitIndex* 알고리즘은 알고리즘 3과 같이 표현된다. 본 논문에서는 불균형 중첩(disproportional overlap) 영역이 가장 작은 분할이 일어나는 분포를 선택하였다. 불균형 중첩영역의 크기가 같을 경우에는 기존 방법과 마찬가지로 사장영역의 면적이 더 작은 분할을 선택한다.

Algorithm of ChooseSplitIndex

CSI1 Along the chosen split axis, choose the distribution with the **minimum disproportional overlap-value**. Resolve ties by choosing the distribution with minimum area-value

알고리즘 3 *ChooseSplitIndex*

그림 13과 같이 중첩영역을 이루는 사각형에서 각 도메인에 평행한 변의 길이를 $Overlap_{tid}$, $Overlap_{rid}$라고 정의하고 *tid*의 가중치를 $Weight_{tid}$, *rid*의 가중치를 $Weight_{rid}$라고 정의하면 *WeightedOverlapMargin*값은 다음 식 (6)과 같이 정의 한다.

WeightedOverlapMargin(*mbr*,*d*) =

$$\sum_{i=1}^{d} Weight_i \times Overlap_i \quad (6)$$

이때 *rid*로 긴 모양의 중첩영역으로 분할되기 위해 *rid*에 부여하는 가중치를 *tid*에 부여하는 가중치값보다 작게 준 뒤에 각 중첩영역의 변의 길이와 각 변의 가중치값을 곱한값을 더한 후 비교하여 가장 작은 값 즉 가장 작은 중첩영역이 되는 분할을 선택하는 방법이다.

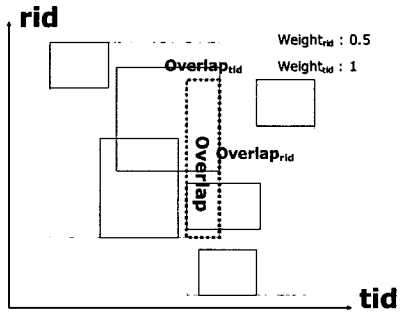


그림 13 분할 방법

이때 MBR에 데이타를 삽입 시 MBR이 분할하게 되는데 각 분할의 중첩영역의 *WeightedOverlapMargin* 값을 식 (6)을 통해 구할 수 있다. 이때 *d*는 차원(dimensionality)를 의미한다. 제안한 분할 방법은 각 분할의 중첩영역의 *WeightedOverlapMargin*값을 비교하여 *WeightedOverlapMargin*값이 가장 적은 분할을 선택한다.

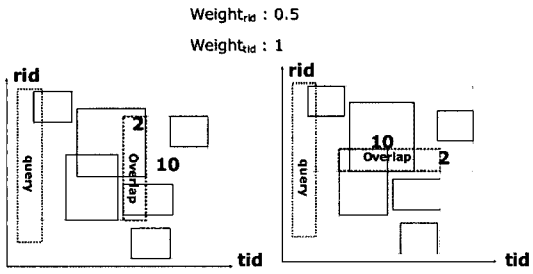


그림 14 분할 예제

그림 14에서 실제 분할 시에 *WeightedOverlapMargin*을 구하는 방법을 예시하고 있다. $Weight_{tid}$ = 1, $Weight_{rid}$ = 0.5이라고 가정했을 때 각 mbr의 *WeightedOverlapMargin*은 표 3과 같다.

표 3 각 분할별 *WeightedOverlapMargin*의 값 계산

	$Overlap_{tid}$	$Overlap_{rid}$	*WeightedOverlapMargin*
SplitCase1	2	10	10×0.5+2×1=7
Splitcase2	10	2	2×0.5+10×1=11

표 3에서 그림 14의 (a)에 해당하는 것이 SplitCase1이며 그림 14 의 (b)에 해당하는 분할이 SplitCase2이다. 이때 SplitCase1의 경우 *WeightedOverlapMargin* 값은 7이 되며 SplitCase2의 경우 *WeightedOverlapMargin*값은 11이 되어 SplitCase1의 경우보다 *WeightedOverlapMargin*의 값이 더 커지게 된다. 따라서 중첩영역이 더 작아지는 SplitCase1을 선택하여 분할이 일어나게 된다. 즉 중첩영역의 면적이 같은 두 분할 방법에서도 *WeightedOverlapMargin*을 이용한 불균형 분할 방법을 사용하여 *rid* 방향으로 긴 중첩영역이 되도록 하여 검색 시 질의영역과 노드간의 겹침을 줄인다.

5. 성능 평가

이 장에서는 논문에서 제안하는 색인의 성능을 평가하기 위하여 기존 연구인 TPIR-tree와 본 논문에서 제안한 삽입 및 분할 정책을 적용한 색인을 비교한다. 색인의 성능 비교의 기준은 각 색인의 구축 비용과 각 질의 처리 비용이다. 실험에 사용된 색인은 디스크 기반이기 때문에 각 실험의 비용은 디스크 I/O로 평가된다. 이 논문에서 제안한 색인과 실험에 사용된 색인의 종류는 표 4와 같다.

표 4 실험에 사용된 색인기법과 약어

색인 기법	약어
TPIR-tree Scheme	TPIR
Disproportional Scheme	Disproportion

5.1 실험 환경

5.1.1 색인 구현

기존의 TPIR-tree에 본 논문에서 제안한 불균형 삽입 및 분할 정책을 적용한 색인은 프로그래밍 언어로는 Java를 썼으며 JDK 5.0라이브러리를 사용 구현하였다. 실험 플랫폼으로는 Windows XP 에서 1GB 메인 메모리, CPU Pentium IV 2.6GHz를 장착한 PC를 이용하여 실험하였다.

색인의 성능을 검증하기 위해서 다양한 환경에서 색

인의 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 위치 태그 데이타 생성기(Tag Data Generator, TDG)를 사용하였다.

5.1.2 실험 데이타

태그 데이타 생성기를 통해 본 논문에서 제안하는 새로운 색인의 실험을 위한 전자태그의 위치 데이타를 생성한다. 기본적인 이동체의 위치 이동 알고리즘은 GSTD 알고리즘과 유사하지만 위치보고 시점이 리더의 인식영역 안으로 들어갈 때와 밖으로 나갈 때만 데이타를 생성한다. TDG를 이용하여 전자태그의 이동 간격, 위치보고 시간 간격 및 보고 횟수를 일정 범위 내에서 랜덤이 되도록 설정한다. 표 5는 색인 구축에 사용된 전자태그의 수와 그때 발생하는 태그 이벤트의 수이다.

표 5 실험에 사용된 전자태그의 수와 그에 따른 태그 이벤트의 수

전자태그 수	태그 이벤트 수
30,000	100,000
60,000	200,000
90,000	300,000
120,000	400,000
150,000	500,000

표 6은 태그 이벤트가 10만개일 때 색인의 삽입 및 분할 정책에 사용되는 가중치를 변경하면서 각각 색인을 구축하였다. 가중치의 경우 *tid*, *time*의 가중치는 1로 고정시키고 *rid*의 가중치를 표 6과 같이 0.5~0.001로 변경하며 색인을 구축했다. 동일한 방법으로 태그 이벤트의 개수를 표 5에서 볼 수 있듯이 10만개에서 50만개로 변경시키며 각 가중치 별로 색인을 구축했다.

표 6 태그 이벤트가 10만개 일 때 각 도메인별 가중치 비율

태그 이벤트	$Weight_{rid}$: $Weight_{tid}$: $Weight_{time}$
100,000	0.5 : 1 : 1
	0.1 : 1 : 1
	0.05 : 1 : 1
	0.01 : 1 : 1
	0.001 : 1 : 1

표 7은 실험에 사용될 영역질의의 비율을 변경을 표현하고 있다. 영역 질의에서 *rid* 도메인 쪽의 질의 영역을 $range_{rid}$라고 표현하였으며 그 때 각 질의 영역의 비율을 표 7와 같이 변경하였다. 예를 들어 *rid* 질의 영역의 크기가 1%일 때 *tid*, *time* 질의 영역의 크기 비율을 표 7에서 보는 바와 같이 10:1:1에서 100000:1:1까지 변경하였다. 각 영역질의에 대해 천 번의 질의를 수행하였다. 동일한 방법으로 *rid*의 질의 영역의 크기 즉 $ragne_{rid}$를 1%, 5%, 10%, 20%, 30%, 50%로 변경하며 각 도메인의 영역 질의의 비율을 같은 방법으로 변경하여 색인에 질의를 수행했다.

표 7 *rid*의 질의 영역이 1%일 때 각 도메인의 질의 영역의 크기 비율

$range_{rid}$	$range_{rid}$: $range_{tid}$: $range_{time}$	Count
1%	10 : 1 : 1	1000
	100 : 1 : 1	1000
	1000 : 1 : 1	1000
	10000 : 1 : 1	1000
	100000 : 1 : 1	1000

5.2 색인 구축 성능 비교

색인 구축 성능을 비교하기 기존 연구인 TPIR-tree와 비교하였다. 아래 그림 15는 TDG로 생성한 태그이벤트의 크기와 각 도메인별 가중치의 비율에 따른 색인 구축 비용을 보여준다. 각 색인의 노드의 크기는 1024바이트이고 단말 노드의 최대 엔트리 수는 26, 비 단말 노드의 최대 엔트리 수는 18이다. 색인 구축 비용은 데이타 삽입과 갱신에서 발생한 노드 접근 횟수를 합한 값이다. 그림 15는 30만개 태그 이벤트에 대해 TPIR-tree 색인과 본 논문에서 제안한 방법에 따라 가중치를 변경시킨 값들을 적용한 색인 구축 비용을 비교하고 있다. TPIR-tree 방법에 비해 본 논문에서 제안한 가중치가 적용된 모든 경우에 대해서 색인의 구축비용이 반 이하인 것을 알 수 있다. 색인에 데이타를 삽입 할 경우 데이타의 삽입 될 자리를 찾아가기 위해 검색이 빈번하게 일어나게 되어 색인의 구축비용이 줄어들게 된다.

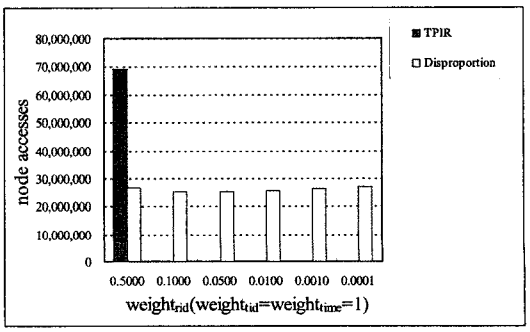


그림 15 색인 구축 성능 비교

5.3 가중치 변경에 따른 질의 성능 비교

본 논문에서 제안한 삽입 및 분할정책의 성능을 측정하기 위해 기존 TPIR-tree와 질의 성능을 비교한다. 논문의 앞부분에서 문제로 삼고 있듯이 기존 이동체 색인의 정방형 질의와는 다르게 $[0,1]^3$의 공간상에서 질의 영역이 *rid* 방향으로 긴 육면체 모양으로 검색을 수행하였다.

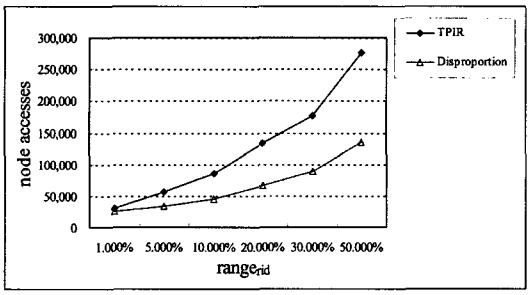


그림 16 $Weight_{rid}$ = 0.5일 때 검색성능

색인 구축 시에 사용한 가중치가 *tid, time*의 경우 1로 고정하고 *rid*는 0.5에서 0.05까지 변경하였다. 그림 16는 *rid*의 가중치 즉 $Weight_{rid}$가 0.5일 때의 검색 성능이다. 가로축은 *rid* 질의 영역의 크기이다. 표 7에서 본 바와 같이 *rid* 질의 영역이 1%일 경우 나머지 *tid*와 *time* 도메인의 질의 영역의 크기를 변경하며 각각에 대해 천 번의 질의를 수행한 결과이다. *rid*의 질의 영역의 크기가 전체에서 1%를 차지할 경우 TPIR-tree와 본 논문에서 제안한 방법간에는 큰 차이가 나지 않는다. 하지만 질의 영역이 커질수록 격차가 늘어나는 것을 알 수 있다.

그림 17은 *tid* 및 *time* 도메인의 가중치는 1로 고정하고 *rid*의 가중치를 0.05로 변경해서 구축한 색인의 검색 비용이다. *rid*의 가중치가 0.5일 때와 의 유사한 모양을 가진다. 가중치의 변경에도 검색 성능의 변화가 없는 이유는 질의 영역을 표 7과 같이 다양한 비율로 조합했기 때문이다. 즉 가중치에 따라 도메인간의 질의 영역의 크기 비율이 최적인 질의가 존재하게 되어 다양한 질의를 조합했을 경우 각 영역질의에 대해 가감이 일어나게 되어 각 가중치에 대해 유사한 검색 성능이 나오게 된다.

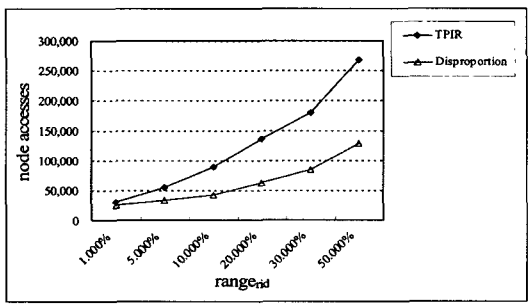


그림 17 $Weight_{rid}$ = 0.05일 때 검색성능

가중치 값은 질의 영역의 크기에 매우 의존적이다. 질의 영역의 크기를 미리 알 수 있는 환경에서는 적절한 가중치 값을 찾을 수 있지만 일반적으로 질의 영역의 크기를 미리 알 수 없는 환경에서는 적절한 가중치 값을 선택하기 위한 방법으로 크게 두 가지 방법이 있을 수 있다. 첫번째로 색인이 구축된 환경에서 실험적으로 가중치를 찾아내는 방법이 있을 수 있다. 이 방법은 미리 여러 번의 실험을 통해 질의의 크기의 분포를 조사하여 통계적인 방법으로 처리 하는 방법이다. 두번째로 질의 영역의 크기 변경에 따라 가중치 값을 계속해서 변경하는 방법이다. 이 방법은 질의 영역의 크기가 일정하지 않고 변하므로 그 시점에서 가장 적합한 가중치를 찾는 방법이다. 현 시점의 질의가 많다고 했을 경우에는 효율적이지만 과거에 대한 질의에는 부적합한 단점이 있다.

6. 결론 및 향후 연구

이 논문에서는 RFID 환경에 적합한 기존 연구인 TPIR-tree를 기반으로 RFID 환경에서 *tid, rid* 도메인의 특징에 따른 삽입 및 분할 정책에 대해서 제시하였다. RFID 환경에서는 도메인의 크기의 차이가 크고 그에 따라 영역질의 시 질의 영역의 각 도메인의 크기 비율이 차이가 심한 불균형 영역 질의가 발생하게 된다. 즉 질의 영역의 모양이 정방형이 아닌 *rid* 도메인 쪽으로 긴 직육면체 모양으로 나타난다. 이동체 색인을 기반으로 하는 기존의 RFID색인의 경우 이러한 도메인의 특성을 반영하지 못한 삽입 정책으로 인해 질의 시에 불균형 영역 질의와 MBR간의 겹침이 심해져 노드 접근 횟수가 증가하는 문제점이 있었다.

이러한 문제를 해결하기 위해 RFID 도메인의 특성을 고려한 삽입 및 분할 방법을 제시하였다. 데이타를 색인에 삽입 시 질의 영역과 닮은꼴의 MBR을 선택하는 불균형 확장 정책을 제안하였다. 즉 질의 영역과 MBR이 교차하지 않고 평행하게 되도록 MBR을 구성하여 검색 시 MBR과 영역 질의간의 겹침을 줄이는 방법이다. 이 방법은 MBR의 모양을 판단하기 위해 면적이 아닌 MBR의 둘레길이(margin)를 사용한 방법이다. 둘레길이는 MBR의 각 도메인에 해당하는 변의 길이에 가중치를 부여하여 더하는 방법으로 구한다. 이러한 둘레길이를 *WeightedMargin*이라고 정의하였고 삽입 시에 이 값을 이용하여 선택될 후보 서브트리의 MBR 중 *WeightedMarginEnlargement* 즉 *WeightedMargin*의

확장값이 가장 작은 서브트리를 선택하도록 하였다. 이러한 삽입 방법을 통해 영역질의와 MBR간의 겹침을 줄여 검색 시 노드 접근 횟수를 줄일 수 있었다.

또한 본 논문에서 R*-tree의 중첩(Overlap) 영역 최소화 정책을 변경한 분할 정책을 제시하였다. 분할 정책은 제안된 삽입 정책과 유사한 접근 방법으로 중첩(overlap)영역이 작고 동시에 중첩(Overlap)영역의 모양이 질의 영역의 모양과 가장 유사한 분할 후보를 선택하도록 하였다. 이 방법은 중첩영역의 모양을 판단하기 위해 중첩영역의 면적이 아닌 중첩영역의 둘레길이(margin)를 사용한 방법이다. 둘레길이는 중첩영역의 각 도메인에 해당하는 변의 길이에 가중치를 부여하여 더하는 방법으로 구한다. 이러한 둘레길이를 *WeightedOverlapMargin*이라고 정의하였고 분할 시에 이 값을 이용하여 분할되는 후보 중 *WeightedOverlapMargin* 값이 가장 작은 분할을 선택하도록 하였다. 이러한 분할 정책을 통해 영역질의와 중첩영역간의 겹침을 줄여 검색 시 노드 접근 횟수를 줄일 수 있었다.

실험을 위하여 태그 객체의 위치 데이타 생성기(TDG)를 통해 RFID환경의 다양한 변수를 고려한 실험 데이타를 생성하였다. 생성된 데이타를 통해 본 논문에서 제안한 삽입 및 분할 정책을 적용한 색인과 기존 연구인 TPIR-tree 색인의 삽입 비용 및 검색 비용을 비교하였다. 불균형 영역 질의가 일어나는 RFID 환경에서 검색 시 2배에 가까운 성능 향상을 볼 수 있었다. 부가적으로 삽입 비용 또한 검색 비용의 감소로 인해 줄어드는 효과가 있었다. 이러한 결과가 나오는 이유는 삽입 연산은 역시 검색의 *ChooseSubtree* 알고리즘을 바탕으로 하기 때문이다. 또한 삽입 시 현재 시간의 데이타를 찾은 연산이 필요하다. 이 연산은 극단적인 불균형 영역 질의이기 때문에 성능 향상에 많은 영향을 끼쳤던 것으로 확인된다.

향후 연구로써 RFID 색인의 도메인인*tid* 도메인과 *rid* 도메인 외에 *time* 도메인에 대한 고려가 필요하다. *time* 도메인은 *tid, rid* 도메인과 다르게 계속해서 증가하는 성질이 있고 *time* 축 방향으로 색인이 커지게 된다. 또한 검색 시에 현재 질의가 중요하여 다른 도메인과의 분석을 통해 검색 성능을 높일 수 있는 방법이 필요하다.

참 고 문 헌

[1] K. Romer, T. Schoch, F. Mattern and T. Dubendorfer, "Smart Identification Frameworks for Ubiquitous Computing Applications," *Proc. of Pervasive Computing and Communications*, pp. 256-262, 2003.

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

[3] K. Romer, T. Schoch. "Infrastructure Concepts for Tag-Based Ubiquitous Computing Applications," *Proc. of Ubicomp*, pp. 253-262, 2002.

[4] 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.

[5] 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.

[6] EPC Tag Data Standard Work Group, "EPC Tag Data Standards Version 1.23," EPC Global, 2005.

[7] 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.

[8] I. Kamel, C. Faloutsos, "On packing R-trees," *Int'l Conf. on Information and Knowledge Management*, pp. 490-499, 1993.

[9] 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.

[10] 전봉기, 홍봉희, "이동체의 색인을 위한 시간 기반 R-트리의 설계 및 구현", *한국정보과학회 논문지 D-데이타베이스, 30권 3호*, pp. 320-335, 2003.

[11] Y. Theodoridis, E. Stefanakis, and T. K. Sellis. "Efficient cost models for spatial queries using r-trees. *Transactions on Knowledge and Data Engineering (TKDE),*" 12(1), pp. 19-32, 2000.

[12] EPCglobal, "The Application Level Events (ALE) Specification, Version 1.0," 2005.

[13] 김동현, 홍봉희, 안성우, "RFID 시스템에서 태그 추적을 위한 리더 모델링 기법", *한국정보과학회 데이타베이스연구회, Korean Database Conference 2006 (KDBC2006), ISSN: 1598-9798, pp. 49-56.*

김 기 홍

2005년 부산대학교 컴퓨터공학과 졸업(공학사). 2007년 부산대학교 대학원 컴퓨터공학과 졸업(공학석사). 2007년~현재 부산대학교 대학원 컴퓨터공학과 박사과정. 관심분야는 지리정보 시스템, RFID 미들웨어, 모바일 GIS, 이동체 색인

홍 봉 희

1982년 서울대학교 컴퓨터공학과 졸업(공학사). 1984년 서울대학교 대학원 컴퓨터공학과 졸업(공학석사). 1988년 서울대학교 대학원 컴퓨터공학과 졸업(공학박사). 1987년~현재 부산대학교 컴퓨터공학과 교수. 관심분야는 이동체 데이타베이스, 공간 데이타베이스, RFID 미들웨어

안 성 우

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