韓國 컴퓨터情報學會 論文誌
第11卷 第6號, 2006. 12

2006-11-6-1-2

이동체의 위치 정보를 저장하기 위한 해쉬 기반 색인의 성능 분석

전 봉 기*

On the performance of the hash based indexes for storing the position information of moving objects

Bong-Gi Jun*

요 약

이동체 데이터베이스 시스템은 연속적으로 위치와 방향을 바꾸는 이동체 집합을 관리해야 한다. 전통적인 공간색인 기법은 정적인 공간 데이터를 관리하는 목적으로 사용되기 때문에 이동체를 위한 색인으로 부적합하다. 이동체의 위치는 연속적으로 변하기 때문에, 이동체의 색인은 변경된 위치 정보를 유지하기 위하여 빈번한 갱신 연산을 수행해야 한다. 이 논문에서는 이동체들의 이동을 처리하기 위한 삽입/삭제 비용을 분석하였다. 실험 결과에서 제안하는 동적 해싱 색인이 기존의 R-트리와 고정 그리드 보다 성능이 우수하였다.

Abstract

Moving objects database systems manage a set of moving objects which changes its locations and directions continuously. The traditional spatial indexing scheme is not suitable for the moving objects because it aimed to manage static spatial data. Because the location of moving object changes continuously, there is problem that expense that the existent spatial index structure reconstructs index dynamically is overladen. In this paper, we analyzed the insertion/deletion costs for processing the movement of objects. The results of our extensive experiments show that the Dynamic Hashing Index outperforms the original R-tree and the fixed grid typically by a big margin.

▸ Keyword : 이동체 데이터베이스(Moving objects databases), 사공간 색인(Spatio-temporal indexing), 해쉬 함수(Hash function), 지리정보시스템(Geographic information system)

• 제1저자 : 전봉기
• 접수일 : 2006.10.19, 심사일 : 2006.12.01, 심사완료일 : 2006. 12.25
* 신라대학교 컴퓨터정보공학부 조교수

Ⅰ. 서론

이동 통신 기술의 발달로 인해 국내에서는 3,800만 명에 해당하는 휴대폰 가입자를 갖고 있고, 개인 휴대 단말기(PDA) 사용자도 급격히 증가하고 있다. 또한 세계 어느 나라보다 상세한 국가 지도를 웹상에서 서비스하고 있다. 따라서 위치기반 서비스(LBS: Location Based Service) 서비스에 있어 선결 요건인 컨텐츠로서 전국 규모의 상세한 전자지도를 이미 확보하고 있다. 이동통신사와 휴대폰을 개발하는 제조사에서 GPS(Global Positioning System)칩을 내장한 휴대폰이 출시되어 있으며, PDA에서는 내·외장형으로 GPS 수신기를 연결할 수 있다. 또한 국내에서 시판되는 PDA나 외국산 PDA도 이미 CDMA 모뎀을 내장형 또는 외장형으로 장착할 수 있어, PDA에서 위치 데이터를 위치 확인 서버에 송신하는 것은 이미 기술적으로 해결된 상태이며 일부 통신사에서는 단순 위치 확인 서비스를 실시하고 있다.

이동체는 GPS 수신기를 갖고 있는 무선단말기에서 자신의 위치를 위치 확인 서버에 전송하는 차량이나 개인으로 시간이 지남에 따라 자신의 위치가 변경되는 객체를 말한다[1]. 이동체 데이터베이스는 저장된 이동체의 정보를 지속적으로 변경해야 한다는 점이 기존 공간 데이터베이스와의 가장 큰 차이점이다. 이동 데이터베이스의 가장 중요한 요구조건은 데이터베이스에 저장된 이동체의 위치 데이터가 지속적으로 변경되어야 하고, 변경 처리는 다음 이동 위치가 보고되기 전에 처리되어야 하는 시간 제약(time-constrained) 연산이라는 것이다. 그리고 주어진 지역의 이동체를 찾는 질의(영역 질의)가 중요하기 때문에 이동체 위치 데이터에 대한 공간 색인 구조[2]를 유지하는 것이 필수적이다.

이동체의 현재 위치는 2차원의 점으로 표현될 수 있으며, 가장 최근에 보고된 위치를 현재 위치로 가정할 수 있다. 공간 색인을 이용하여 이동체의 현재 위치를 색인할 경우에 다음과 같은 문제가 발생한다. 고정 격자 파일은 이동체의 위치를 키 값으로 해싱하는 방법을 사용하여 비교적 간단한 과정으로 색인이 가능하다는 장점이 있다. 그러나, 데이터 집합이 비정규 분포일 경우, 특정 영역 셀에 지속적인 오버플로우가 발생하여 색인의 성능이 저하되는 문제가 발생한다. 이동체는 이동성으로 인해 이동체 밀집 지역을 빈번히 발생시킨다. 이로 인해 고정 격자 파일의 성능 저하가 발생한다. R-tree는 높이 균형 트리 구조이기 때문에, 데이터의 분포와 관계없이 우수한 성능을 나타낸다. 그러나, 이동체의 계속되는 위치 이동으로 인해 색인의 변경이 발생하고, 색인의 빈번한 변경으로 전체적인 색인의 성능 저하가 발생한다. 이 논문에서는 이동체의 위치 변경에 따른 이동체 색인의 비용을 구하여 성능을 분석하고 성능 평가 실험을 통하여 이를 검증하였다.

이 논문의 구성은 다음과 같다. 2장에서는 이동체 데이터베이스 색인의 관련 연구를 기술하고, 3장에서는 셀의 크기에 따른 이동체 색인 비용을 기술한다. 4장에서는 이동체의 위치 변경에 따른 비용을 기술한다. 5장에서는 실험을 수행하여 제안하는 동적 해싱 색인의 성능을 비교한다. 마지막으로 6장에서는 결론을 기술한다.

Ⅱ. 관련 연구

이동체 데이터베이스를 위한 공간 색인을 기존의 정적인 공간데이터[9]를 위해 사용되던 공간 색인인 R-tree, grid-file, quad-tree, KDB-tree 등으로 구현하기가 어렵다. 이유는 이동체의 위치가 지속적으로 변하기 때문에 공간 색인 구조가 동적으로 재구성되는 경우에 비용이 너무 크다는 문제가 있다. R-tree 계열의 공간색인은 이동체의 위치가 변경되면 기존 위치에서 이동체를 삭제하고 새로운 위치에서 새로 삽입해야 하며, 이는 노드의 분할과 합병을 빈번하게 요구하며 이를 처리하는 연산 비용이 많이 소요되어 성능이 현저히 떨어지는 단점이 있다.

이동체의 색인에 관련된 연구는 크게 3가지 부분으로 나누어 볼 수가 있다. 첫째, 현재와 미래 위치를 검색하기 위한 색인에 관련된 연구들로서 TPR-tree, 해싱, Quad-tree를 기반으로 하는 색인이 있다. 이동체의 현재 위치 또는 미래 위치를 예측하는 것을 목적으로 한다. 둘째, 시공간 색인에 대한 연구[10]로서 3D R-tree, HR-tree, MV3R-tree가 있다. 이동체의 이동 경로를 저장하는 것을 목적으로 한다. 셋째, 이동체의 궤적과 같은 과거 이력 색인에 대한 연구로서 STR-tree, TB-tree가 있다. 과거 궤적 질의의 성능 향상을 목적으로 하는 색인이다. 이 논문에서는 이동체의 현재 위치를 검색하기 위한 색인을 연구하였다.

해싱 기법을 이용한 이동체 색인은 이동체의 위치를 2차원 공간 상의 점으로 모델링한다. 관련 연구[3, 11]는 그림 1과 같이 해쉬 기반 색인에서 이동체가 셀을 벗어나지 않으면 색인 갱신이 없다는 아이디어를 사용한다. 해싱 함수를 사용하여 빠른 검색이 가능하나, 이동체의 분포에 따른 노드의 오버플로우 문제를 해결해야 한다.

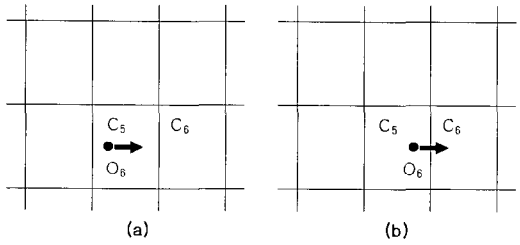


그림 1. 이동체의 이동과 색인의 변화.
(a) 셀 내에서의 이동, (b) 다른 셀로의 이동
Fig. 1. The movement of a moving object and the update of index. (a) the movement in a cell, (b) the movement across cells

그림 1(a)와 같이 셀 내에서의 위치 변경은 색인 구조에서는 변경이 없다. 그림 1(b)와 같이 이동체의 위치가 다른 셀로 이동해도 해쉬 기반의 색인 구조에서는 삭제와 재 삽입 연산 비용이 트리 계열보다 작다.

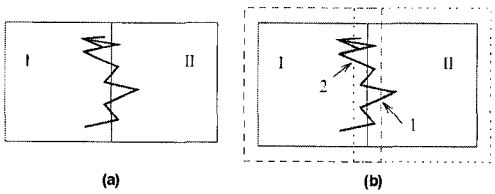


그림 2. Zigzag 문제.
(a) 셀 경계를 이동하는 이동체, (b) 중첩된 확장 MBR들
Fig. 2. Zigzag problem. (a) an moving object along the border, (b) overlaped expansion MBRs

그림 2와 같이 이동체가 셀의 경계 상에서 이동할 경우, 셀 간의 이동으로써 연속적인 색인의 갱신이 발생하는 문제가 있다. 이 경우 그림 2(b)처럼 영역 간에 약간의 중첩(overlap)을 허용하여 갱신 횟수를 줄일 수 있다[3].

이동체는 연속적으로 위치가 변하기 때문에 시간에 따라 분포성이 변한다. 관련 연구[3]에서는 Quad-tree를 색인으로 사용하기 때문에 그림 3과 같이 비균등 분포에서 경사 트리(Skewed tree)가 될 수 있다. 특히, 이동체는 특정시간에 도심지, 교차로에서 밀집될 수 있다. 해쉬 기반 인덱스에서 이동체의 밀집화는 셀 버켓(bucket)의 오버플로우를 유발하여 색인의 성능을 저하시킨다.

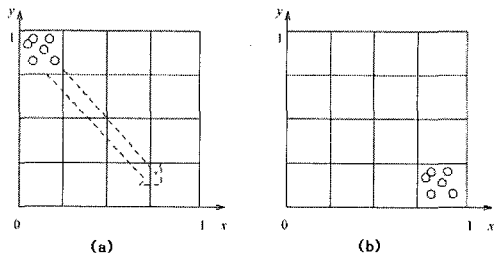


그림 3. 이동체의 비대칭 분포.
(a) 시각 t0에서의 이동체들, (b) 시각 t1에서의 이동체들
Fig. 3. Skewed distribution of objects. (a) moving objects at t0, (b) moving objects at t1

LUR-tree(Lazy Update R-tree)[4]는 다차원 색인인 R-tree을 이용한 이동체 색인으로 갱신 비용을 줄이는 것을 목적으로 한다. 기존의 R-tree는 해쉬 기반의 색인에서와 같이 같은 셀 내의 이동을 판단할 수 없기 때문에 이동체의 모든 위치 변경은 그림 4와 같이 삭제 및 삽입 연산을 수행해야 한다.

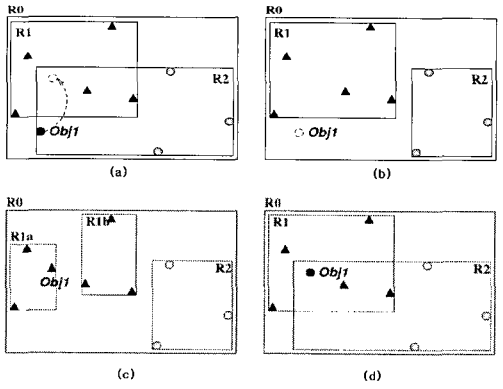


그림 4. R-tree에서의 기존 갱신 방법과 Lazy 갱신 방법의 예.
(a) 위치 이동, (b) 기존 방법: Obj1 삭제, (c) 기존 방법: Obj1 삽입과 노드 분할 (d) Lazy 갱신: R2의 MBB 확장
Fig. 4. Example of traditional update algorithm and lazy update algorithm. (a) a movement of the position, (b) traditional update: delete Obj1, (c) traditional update: insert Obj1 and split the node, (d) Lazy update: expand the MBB of R2

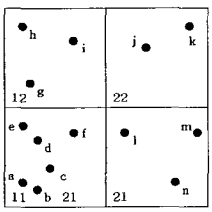
그림 4(a)와 같이 객체 Obj1이 이동하게 되면 우선 그림 4(b)와 같이 객체 Obj1의 이전 위치를 키값으로 노드 R2에서 삭제한다. 그러면 R2의 MBR이 조정된다. 객체 Obj1의 새로운 위치를 키값으로 삽입하면 R1 노드는 오버플로우가 발생하기 때문에 그림 4(c)와 같이 2개의 노드로 분할된다.

그림 4와 같이 기존 방법에서는 삭제 및 삽입 연산으로

최적의 검색이 가능한 색인을 구성할 수 있으나, 갱신 연산은 노드의 분할 또는 합병을 유발하기 때문에 비효율적이다. Lazy 갱신은 이러한 문제점을 해결하기 위하여 이동체의 위치 변경으로 발생하는 갱신 연산을 노드 내에서의 갱신으로 제한함으로써 노드의 구조 변경에 따른 비용을 감소시켰다.

III. 셀의 크기에 따른 이동체 색인 비용

이동체의 이동성을 고려할 때, 이동체 데이터베이스를 위한 색인 구조는 빈번한 갱신 연산을 효율적으로 처리하기 위하여 위치 데이터의 갱신 연산 비용이 최소화된 공간 색인 구조이어야 한다.



(a)

Cell_ID	Data_Bucket	Overflow_Bucket
11	a, b, c	d, e, f
12	h, i, g	
21	l, m, n	
22	j, k	

(b)

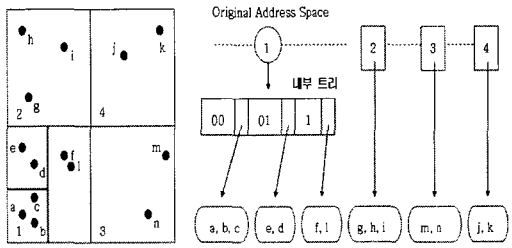
그림 5. 고정 그리드 색인의 예.
(a) 지동상의 그리드 셀과 이동체, (b) 고정 그리드의 색인 구조
Fig. 5. An example of the fixed grid indexing. (a) moving objects and grid cells on a map, (b) the index structure of a fixed grid

이 논문에서는 그림 1(a)와 같이 셀 내의 이동에 대해 색인의 변경이 발생하지 않는다는 특징을 고려하여 이동체 색인의 비용 모델을 구하고자 하였다. 고정 그리드에서 셀이 크면 보다 많은 이동체의 이동에 셀 내의 이동이기 때문에 색인의 변경이 작게 발생한다. 하지만 셀의 크기를 크게 하면 그림 5의 셀11과 같이 오버플로우가 발생할 확률이 증가하기 때문에 색인의 성능이 감소하게 된다.

고정 그리드의 성능은 색인 대상 데이터의 분포에 민감하게 영향을 받는다[5]. 즉, 정규 분포의 데이터 집합에 대해서 우수한 색인 성능을 나타내지만, 특정 지역에 객체가 집중된 비정규 분포의 데이터 집합에 대해서는 지속적인 오버플로우 페이지의 생성 및 접근을 발생시켜 색인 성능이 떨어진다.

이동체를 차량이나 휴대단말기로 가정한다면, 이동체들은 도심지역에 밀집되어 있으며 시간에 따라 이동하게 된다. [6] 이동체의 이동을 셀 내의 이동으로 유도하기 위하여 셀의 크기를 증가시키면 오버플로우 노드를 증가시켜 성능을 저하시킨다.

고정 그리드에서 가장 이상적인 셀의 크기는 이동체가 전체 도메인에서 균등 분포로 분산되어 있을 때, 하나의 셀이 n/b개의 이동체를 포함하는 크기이다. 이 때 전체 이동체의 수는 n이고, 데이터 페이지의 최대 엔트리 수는 b이다.

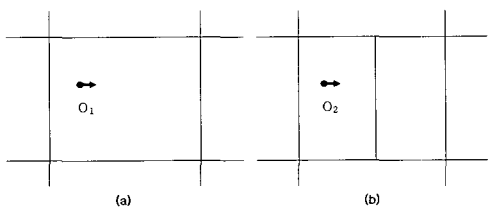


(a) (b)

그림 6. 동적 해싱 색인 구조.
(a) 지도 상의 분할 예, (b) 동적 해싱 색인의 구조
Fig. 6. An example of a dynamic hashing index. (a) an example of the split on a map, (b) the structure of a dynamic hashing index

그림 5와 같이 고정 그리드는 이동체가 밀집된 지역에서 오버플로우 버켓이 발생하면, 삽입/삭제/검색시 순차검색(데이터 버켓 + 오버플로우 버켓수)을 수행해야 하기 때문에 성능 저하를 초래한다. 이러한 문제점을 해결하기 위하여, 이 논문에서 제안하는 동적 해싱 구조는 공간 분할을 수행하여 그림 6과 같이 내부 트리로 분할된 셀들을 관리하여 검색 및 변경 연산에서 I/O 비용을 감소시켰다.

제안하는 동적 해싱 구조는 그림 6(b)와 같이 1차원 색인에서의 전통적인 동적 해싱과 같이 1차 주소를 얻기 위해 해쉬 함수를 사용하고, 오버플로우 버켓들을 트리 구조의 색인을 사용하여 색인화하였다. 오버플로우 버켓은 Z-ordering 방법을 사용하여 분할번호를 부여한다.



(a) (b)

그림 7. 셀의 크기와 색인의 변경의 관계.
(a) 그리드 셀 내에서의 이동, (b) 1/2크기로 분할된 셀 내에서의 이동.
Fig. 7. The relationship between the size of a cell and updating index. (a) The movement in a grid cell, (b) The movement in a half-size cell.

공간 분할을 수행하면 그림 7(b)와 같이 이동체가 속하는 셀 공간이 작아지게 된다. 분할로 인하여 셀 공간이 감소하면 이동체의 셀 내의 이동이 감소하고 셀 간의 이동이 증가하여 이동체의 위치 변경으로 색인의 변경이 증가하는 문제점이 발생한다. 분할로 인해 셀 공간이 1/2로 줄어 들면 셀 내의 이동이 대략 1/2로 줄어들게 된다. 이는 셀 내의 이동시 색인의 변경이 없는 장점이 상쇄되게 된다.

다음 장에서 소개할 검색 성능 분석에서와 같이 공간 분할로 동적 해싱 구조를 사용하면 검색에서는 좋은 성능을 보이는 것은 분명하지만 위치 변경으로 색인의 변경이 많이 발생하는 이동체 색인의 특성상 변경 연산 비용을 줄이는 것이 중요하다.

이러한 문제점을 해결하기 위하여 이 연구에서는 색인 변경 비용을 줄이는 방법으로 문제를 해결하였다. 해결책은 앞에서 설명한 오버플로우가 발생한 셀에 존재하는 내부 트리를 메인 메모리에 캐싱하는 것이다.

IV. 이동체의 위치 변경에 따른 비용

공간 분할 방식의 색인인 고정 그리드와 동적 해싱 구조는 이동체의 위치 변경이 셀 내의 이동인지, 셀 간의 이동인지 알 수 있다. 이는 고정 그리드에서 메모리에 상주하는 디렉토리를 검사하면 I/O 발생 없이 가능하다. 또한 동적 해싱에서도 오버플로우가 발생한 셀의 내부 트리를 메모리에 캐싱하고 있으면 I/O 발생 없이 가능해 진다.

4.1 변경 연산의 비용 모델

이동체의 위치 변경에 따른 색인들의 비용을 비교하기 위하여 우선 고정 그리드와 동적 해싱 구조의 변경 연산에 대한 비용을 구한다. 이동체의 셀 간의 이동은 삭제 후 삽입 연산으로 구현된다.

이동체 분포의 집중화가 심하여 오버플로우가 많이 발생할수록 고정 그리드의 성능이 저하된다. 성능 비교를 간단히 하기 위하여 이 절에서는 셀 분할이 1번 발생한 경우에 대해 성능을 비교한다. 고정 그리드는 1개의 오버플로우 버켓을 가지게 된다.

다음은 오버플로우가 발생한 셀에서 다른 오버플로우가 발생한 셀로의 이동시 고정 그리드와 동적 해싱 구조의 변경 연산의 비용이다.

㉠ 고정 그리드에서는

> 삭제 연산에 4번의 I/O 발생 : read에 2번, 2번의 write
> 삽입 연산에 3번의 I/O 발생 : read에 2번, 1번의 write

삭제 연산에서 데이터 페이지에 해당 객체가 있으면 Read I/O 1번이 발생하고, 해당 객체의 삭제로 인하여 나머지 객체들이 앞으로 옮겨지기 때문에 오버플로우 페이지를 읽는 Read I/O 1번과 Write I/O 2번이 발생한다. 삽입 연산에서는 항상 맨 뒤에 객체가 삽입되기 때문에 오버플로우 페이지를 찾기 위하여 2번의 Read I/O와 1번의 Write I/O가 필요하다.

㉡ 동적 해싱 구조에서는

> 삭제 연산에 2번의 I/O 발생 : read에 1번, 1번의 write
> 삽입 연산에 2번의 I/O 발생 : read에 1번, 1번의 write

동적 해싱 구조에서 내부 트리를 메모리에 캐싱함으로서 추가적인 I/O 없이 데이터 페이지를 바로 찾을 수 있게 하여 성능 향상을 도모 하였다.

그림 8은 평균 0.5, 분산 0.1의 정규분포에서 각 셀의 데이터 페이지의 수를 보이고 있다. 고정 그리드와 동적 해싱 방법은 둘 다 공간 분할 방식이기 때문에 셀 당 페이지 수는 동일하다.

1	1	1	1	1	1	1	1	1	1
1	1	1	2	2	2	2	1	1	1
1	1	2	2	2	2	2	2	1	1
1	2	2	2	2	2	2	2	2	1
1	2	2	2	2	2	2	2	2	1
1	2	2	2	2	2	2	2	1	1
1	1	2	2	2	2	2	2	1	1
1	1	1	1	2	2	1	1	1	1
1	1	1	1	1	1	1	1	1	1

그림 8. 정규 분포에서의 페이지 수
Fig. 8. The number of pages in normal distribution

고정 그리드와 동적 해싱 구조에서는, 이동체의 이동시에 셀 간의 이동에서 색인의 변경이 발생한다. 표 1은 고정 그리드와 동적 해싱 구조에서 색인 변경의 경우에 따른 I/O 횟수를 보인다. 오버플로우 발생한 셀은 C_o로, 오버플로우가 발생하지 않은 셀은 C_g로 표시하였다. 그림 9에서 C_{22}와 C_{32}는 C_o이다.

표 1. 이동하는 셀의 유형에 따른 I/O 횟수
Table 1. The number of I/O versus the cell types of the movement of objects

이동 유형	설명	고정 그리드	동적 해싱 방법
M1	Co 내부 이동	0	4
M2	Co -> Co 이동	7	4
M3	Cg -> Co 이동	5	4
M4	Co -> Cg 이동	6	4
M5	Cg -> Cg 이동	4	4

셀 간의 이동체의 이동은 삭제 후 삽입 연산으로 구현되기 때문에, 표 1은 4.1절에서 설명한 비용 모델을 사용하여 이동 유형에 따른 I/O 수를 구하였다.

그림 6과 같이 동적 해싱 구조는 오버플로우가 발생하면 셀 공간을 1/2로 분할한다. 고정 그리드는 그림 5와 같이 오버플로우가 발생하면 셀 공간을 분할 하지 않는다. 하지만 새로운 버켓을 추가하여 검색 비용을 증가시킨다.

동적 해싱 구조에서 오버플로우 발생에 따른 셀 공간 분할의 단점은 그림 7과 같이 공간이 1/2로 감소하였기 때문에 오버플로우가 발생한 셀(Co)에서 내부 이동시 I/O가 발생할 수 있다는 것이다. 고정 그리드에서는 셀 내부 이동은 색인 변경이 없기 때문에 I/O가 발생하지 않는다.

제안하는 동적 색인 구조의 성능 분석을 위해서는 이동체의 이동과 셀 유형에 따른 I/O 횟수를 구해야 한다. 표 1에서와 같이 Co 셀에서 외부로의 이동(M2, M4)시 30% 이상의 성능 향상이 있다.

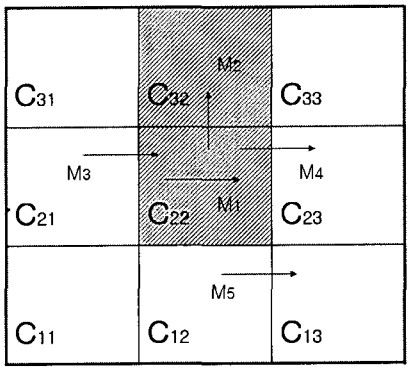


그림 9. 오버플로우 셀을 지나는 객체의 이동의 예
Fig. 9. An example of movement of objects across overflow cells

그림 8에서와 같이 오버플로우가 최대 1번만 발생한 비교적 균등한 분포에서, 표 1의 이동체의 이동에 따른 I/O 수를 이용하여 위치 변경 연산의 비용을 계산할 수 있다. 위치 변경 연산에 따른 비용에서 20% 이상의 성능 향상을 예측할 수 있다.

4.2 영역 질의 비용 모델

영역 질의 비용 모델은 비교적 간단히 예측할 수 있다. 영역 질의는 검색 연산이기 때문에 Write I/O는 발생하지 않기 때문에 질의 영역에 겹치는 셀의 해당 페이지 수와 같다.

그림 7과 그림 8에서 사용한 제약 조건과 같은 조건에서 질의 영역을 작게 하여 셀 들간의 경계에서 질의를 하면 표 2과 같은 영역 질의에 따른 대략적인 I/O 수를 구할 수 있다. 오버플로우 발생한 셀은 Co로, 오버플로우가 발생하지 않은 셀은 Cg로 표시하였다.

표 2는 영역 질의가 2개의 셀에만 겹쳐질 경우만을 계산하였다. 하지만 실제로는 4개의 셀과 겹칠 수도 있다. 전체적으로 50% 이상의 성능 향상이 예측된다. 이는 기존의 공간 색인에서 고정 그리드가 가지고 있는 단점을 이동체 색인에서도 보이고 있기 때문이다.

표 2. 영역 질의의 종류에 따른 I/O 횟수
Table 2. The number of I/O versus the types of range queries

	고정 그리드	동적 해싱 방법
Co 내부	2	2 또는 1
Co + Co	4	2
Co + Cg	3	2
Cg + Cg	2	2
Cg 내부	1	1

V. 실험

이 장에서는 색인의 성능 비교의 기준을 제시한 관련 연구[7]를 바탕으로 논문에서 제안하는 동적 해싱 구조의 성능을 평가하기 위해 이동체의 색인 중에서 고정 그리드, R*-tree와 성능을 비교하고자 한다. R*-tree는 색인의 빈번한 삽입 삭제 연산으로 인하여 해싱 기반의 공간 분할 방식의 색인에 비해 상대적으로 연산비용이 높았다.

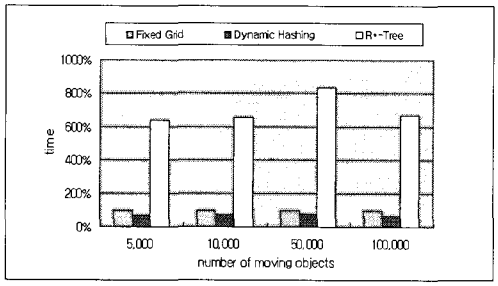


그림 10. 이동체의 개수에 따른 색인 변경 비용 비교
Fig. 10. A comparison of the costs of updating indexes for varying the number of moving objects

그림 10과 같이 R*-tree는 색인 변경 연산을 위하여 비단말 노드의 변경 전파로 인한 노드 변경 연산의 증가로 색인의 현재위치를 저장하는 이동체 색인으로 적합하지 않았다. 이 연구에서는 이후의 실험에서는 고정 그리드와 동적 해싱 색인만을 비교한다. 이동체 생성에 대한 대표적인 도구인 GSTD 생성기[8]를 사용하여 다양한 조건에서 이동체의 위치 데이터를 생성하여 실험하였다.

5.1 저장 공간의 비교

고정 그리드와 제안하는 동적 해싱 구조는 1차 주소를 해싱 함수를 사용하는 것이 동일하기 때문에 우선 공간을 고정 격자로 나누어야 한다. 셀의 갯수는 n/b가 이상적이다. 이 때 전체 이동체의 수는 n이고, 데이터 페이지의 최대 엔트리 수는 b이다.

고정 그리드를 셀의 크기를 증가시켜 실험하였으나, 오버플로우가 증가하여 성능 향상이 없었다. 동적 해싱 구조는 1차 주소를 해싱 함수를 사용하고, 오버플로우가 발생한 셀 내에서만 트리 구조를 가지기 때문에 셀 크기에 따른 성능 분석을 위하여 다음과 같은 셀 크기에 따른 2가지로 색인을 (DH-1, DH-2)를 만들어 실험한다. DH-1과 DH-2는 색인을 생성할 때 색인 생성 파라미터가 다릅니다.

㉠ DH-1
- 실험에 사용되는 고정 그리드와 같은 조건이다.

㉡ DH-2
- DH-1 보다 셀의 면적이 4배 크다.
- 각 축당 셀의 수가 DH-1의 2배이다.

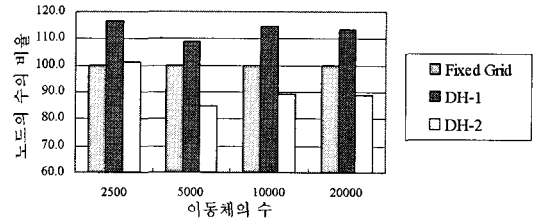


그림 11. 색인 저장 공간의 비교
Fig. 11. A comparison of space utilization.

그림 11과 같이 고정 그리드의 오버플로우 페이지를 포함하는 데이터 페이지와 동적 색인 방법의 데이터 페이지를 비교하면, DH-1의 경우에 내부 트리를 저장하는 페이지가 추가로 필요하기 때문에 페이지의 수가 많다. 동적 색인 구조는 그림 6에서와 같이 해싱 주소 2, 3, 4번이 가리키는 데이터 페이지와 내부 트리가 가리키는 데이터 페이지가 있다. DH-2의 경우에는 셀을 크게 만들었기 때문에 초기 셀의 수가 1/4이기 때문에 페이지 수가 작다.

5.2 색인 구축 및 변경 연산의 비교

이 절에서는 제안하는 동적 해싱 구조의 색인 구축와 변경 연산에 관련하여 노드 접근 횟수를 비교하여 성능 비교를 한다. 그림 12는 이동체가 초기에 보고한 위치를 기반으로 색인 구축 성능을 비교하였다. 동적 해싱 방법이 비용 모델에서와 같이 30% 정도의 성능 향상이 있었다. DH-2 방법이 근소하나 DH-1보다 I/O가 5% 정도 작았다. 이동체의 수가 증가하여도 색인의 셀의 수가 증가하기 때문에 이동체 수에 따른 성능 차이는 없었다.

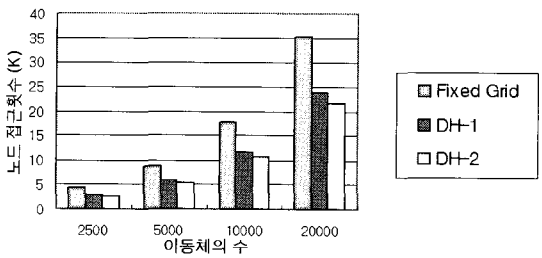


그림 12. 색인 구축을 위한 노드 접근 횟수
Fig. 12. The number of node accesses for building indexes

이동체의 이동으로 발생하는 색인 변경 연산도 동적 색인 방법이 30% 이상의 성능 향상이 있었다. 이동체의 위치 변경은 100번을 수행하였다.

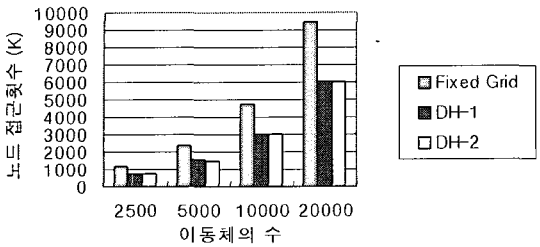


그림 13. 색인 변경을 위한 노드 접근 횟수
Fig. 13. The number of node accesses for updating indexes

제안하는 동적 해싱 구조에서 색인 변경 연산의 성능은 4장에서 예측한 성능 향상이 있었다. 색인 변경 연산에서 DH-1과 DH-2는 성능 차이가 거의 없었다. 그림 8과 같은 정규 분포에서는 동적 해싱 구조에서 셀의 크기를 증가시키면 셀에 포함되는 객체의 수가 증가되기 때문에 오버플로우가 발생하여 셀의 분할하기 때문이다.

5.3 영역 질의의 성능 평가

주어진 특정 지역의 이동체들을 검색하는 영역 질의의 성능은 이동체 색인에서 중요하다. 영역 질의의 크기(RQS)는 각 도메인 크기의 5%, 10%, 20%이다. 질의 영역은 랜덤으로 주어진 도메인 내에서 생성하였으며, 각 색인들은 미리 생성된 영역 질의 집합 데이터를 사용하여 동일한 질의에 대한 처리 성능을 비교하였다.

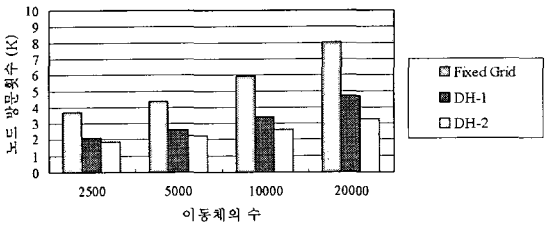


그림 14. 5% 영역 질의 성능 비교
Fig. 13. Performance evaluation for processing 5% range queries in each dimension

그림 14와 같이 5% 영역 질의는 실제 도메인에 따라 차이가 있겠지만 작은 크기의 질의는 아니다. 동적 해싱 방법의 성능 향상은 이동체 수가 증가하여도 같았다. DH-2가 DH-1 보다 예측한 것과 같이 질의 성능이 우수하다. 이는 그림 11에서와 같이 데이터 페이지의 수가 작기 때문에 상대적으로 페이지 당 셀의 크기가 크다. 이는 질의와 겹치는 데이터 페이지 수의 감소를 의미하는 것이다.

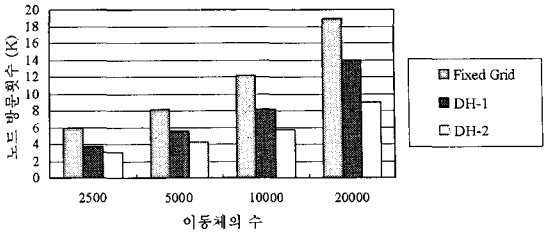


그림 15. 10% 영역 질의 성능 비교
Fig. 15. Performance evaluation for processing 10% range queries in each dimension

그림 15와 같이 10% 질의에서는 검색 성능 향상 폭이 작아진다. 이는 질의 영역이 넓어지기 때문에 5% 질의에 비해 상대적으로 겹치는 영역이 증가하여 셀을 쪼개서 관리하는 효과가 작아지기 때문이다. 또한 DH-1의 데이터 페이지가 많은 것에 대한 비용 증가가 영향을 미친다.

20% 영역 질의는 매우 큰 질의로 대부분의 색인에서 저장 공간의 효율성에 영향을 많이 받는 것으로 알려져 있다. 그림 16에서도 5% 질의에서 보였던 동적 해싱 방법 중 DH-1의 성능 향상 효과가 보이지 않는다. 하지만 DH-2는 고정 그리드에 비해 20% 이상의 성능 향상이 있었다.

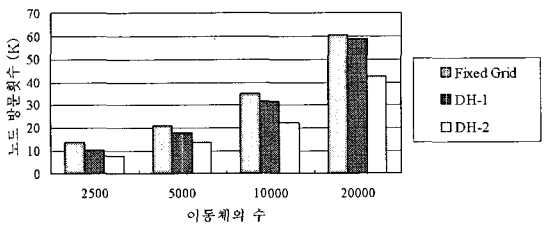


그림 16. 20% 영역 질의 성능 비교
Fig. 16. Performance evaluation for processing 20% range queries in each dimension

V. 결론

전통적인 색인 기법은 정적인 위치 데이터를 관리하는 목적으로 사용되기 때문에 이동체를 위한 색인으로 부적합하다. 이동체의 위치는 연속적으로 변하기 때문에, 이동체의 색인은 변경된 위치 정보를 유지하기 위하여 빈번한 갱신 연산을 수행해야 한다. 갱신 연산은 색인의 구조를 재구성해야 하기 때문에 빈번한 위치 변경은 색인 변경 과부하를 초래한다.

이 논문에서는 이동체들을 관리하기에 적합한 새로운 색인 기법인 동적 해싱 색인을 제안하고, 이 기법의 색인 비용을 분석하였다. 동적 해싱 색인 구조는 해쉬와 트리를 결합한 동적 해싱 기술을 공간 색인에 적용한 것이다. 이 논문에서 이동체의 빈번한 위치 변경에 대한 비용 모델과 동적 색인 구조를 분석하였고, 성능 평가 실험을 통하여 검증하였다. 실험 결과에서 새로이 제안하는 색인 기법(동적 해싱 색인)은 R-tree와 고정 그리드 보다 성능이 우수하였다.

이동체 데이터베이스의 비용 분석 및 성능 평가는 이동체 질의 처리, 이동체 저장 모델, 위치 기반 질의 처리, 시공간 이동체 저장 시스템과 같은 이동체 데이터베이스의 관련 분야 연구에 기초가 될 수 있다.

참고문헌

[1] O. Wolfson, B.Xu, S. Chanmberlain, and L. Jiang. "Moving objects database: issues and solutions," Proc. of Int'l Conf. on Scientific and Statistical Database Management, pp. 111-122, 1998.

[2] H. Samet, The Design and Analysis of Spatial Data Structures, Addision-Wesley, Reading, MA, 1990.

[3] Z. Song and N. Roussopoulos, "Hashing moving object," Int'l. Conf. on Mobile Data Management, pp. 161-172, 2001.

[4] D. Kwon, S. Lee, and S. Lee, "Indexing the current positions of moving objects using the lazy update R-tree," Int'l Conf. on Mobile Data Management, pp. 113-120, 2002.

[5] J. Nievergelt, H. Hinterberger, and K. C. Sevcik, "Then Grid Files: An adaptive, symmetric multikey file structure," ACM Transactions on Database Systems, vol.9, no. 1, pp. 38-71, 1984.

[6] T. Brinkhoff "A framework for generating network-based moving objects," Proc. of the Int'l Conf. on Scientific and Statistical Database Management, pp. 253-255, 2000.

[7] J. Zoble, A. Moffat, and K. Ramamohanarao, "Guildlines for presentation and comparison of indexing techniques," ACM SIGMOD Record, vol. 25, no. 3, pp. 10-15, 1996.

[8] Y. Theodoridis, J. R. O Silva, and M.A Nascimento, "On the generation of spatio-temporal datasets," Proc. of Int'l Symposium on Spatial Databases, pp. 147-164, 1999.

[9] 이기영, 노경택, "공간 데이타 변환 시스템의 설계 및 구현," 한국컴퓨터정보학회 논문지, 제8권, 제4호, 2003.

[10] 전봉기, "이동체의 현재 위치 색인을 위한 동적 해싱 구조의 설계 및 구현," 한국해양정보통신학회 논문지, 제8권, 제6호, pp. 1266-1272, 2004.

[11] 전봉기, "3차원 R-트리를 이용한 이동체 색인에 관한 연구," 한국컴퓨터정보학회 논문지, 제10권, 제4호, 2005.

저 자 소 개

전 봉 기

1991년 부산대학교 컴퓨터공학과(학사)

1993년 부산대학교 컴퓨터공학과(석사)

2003년 부산대학교 컴퓨터공학과(공학박사)

1993년~1998년 한국통신연구소 전임연구원

2000년~2001년 동명대학교 정보공학부 전임강사(기간제)

2003년 ~현재 : 신라대학교 컴퓨터정보공학부 조교수

〈관심분야〉 데이터베이스, 공간 데이터베이스, 이동체 데이터베이스