이동체 데이터베이스를 위한 색인 구조의 비용 모델

전 봉 기*

Cost Model of Index Structures for Moving Objects Databases

Bong-Gi Jun*

이 논문은 2004년도 한국학술진흥재단의 지원에 의하여 연구되었음 (KRF-2004-003-D00303)

요 약

본 논문에서는 이동체들을 관리하기에 적합한 새로운 색인 기법을 개발하고, 이 기법의 비용 모델을 제안한다. 또한 삽입/삭제 비용이 적은 동적 해싱 색인을 제안한다. 동적 해싱 색인 구조는 해쉬와 트리를 결합한 동적 해싱 기술을 공간 색인에 적용한 것이다. 본 논문에서는 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.

ABSTRACT

In this paper, we are going to develop a newly designed indexing scheme which is compatible to manage the moving objects and propose a cost model of the scheme. We propose a dynamic hashing index that insertion/delete costs are low. The dynamic hashing structure is that apply dynamic hashing techniques to combine a hash and a tree to a spatial index. We analyzed the dynamic index structure and the cost model by the frequent position update of moving objects and verified through a performance assessment experiment. The results of our extensive experiments show that the newly proposed indexing schemes(Dynamic Hashing Index) are much more efficient than the traditional the fixed grid and R-tree.

키워드

Moving objects databases, Geographic Information System, Spatio-temporal indexing, Hash

Ⅰ. 서 론

이동체는 GPS 수신기를 갖고 있는 무선단말기에서 자신의 위치를 위치 확인 서버에 전송하는 차량이나 개인으로 시간이 지남에 따라 자신의 위치가 변경되는 객체를 말한다[1]. 이동체 데이터베이스(Moving Objects Databases)는 이동체들의 집합으로 정의되며 저장된 이동체들의 위치 데이터는 지속적으로 변경되는 특징을 갖고 있다. 이동체의 위치 데이터는 주어진 시간 간격으로 무선단말기(PDA나 GPS칩이 내장된 핸드폰)에서 위치 확인 서버로 전송됨을 가정한다. 위치 데이터를 전송하는 시간은 수초 내지 수분 또는 수십분이 걸릴 수 있음

* 신라대학교 컴퓨터정보공학부 조교수

접수일자 : 2006. 9. 19

을 가정한다.

연속적으로 이동하는 이동체의 위치를 검색하는 것은 위치 기반 서비스(LBS : Location Based Service)에서는 중요한 응용 중의 하나이다. 이동체의 위치를 기반으로 하는 LBS 응용은 다양한 분야에서 사용될 수 있으며, 특히 물류 및 수송관리, 교통정보 서비스, 디지털 전장과 같은 대표적인 응용 서비스의 개발이 요구되고 있다. 이와 같은 LBS 응용의 개발을 위해서는 시간의 흐름에 따라 증가하는 이동체의 위치 정보를 효과적으로 관리하고, 빠른 검색을 제공하는 이동체 색인의 개발이 필요하다.

본 논문은 이동체 데이터베이스를 위한 공간 색인(Spatial Indexing)에 관한 연구로서 이동체의 빈번한 위치 변경에 따른 동적 색인 구조와 비용 모델을 해석적 접근으로 분석하고 성능 평가 실험을 통하여 이를 검증한다.

기존의 공간 색인은 정적인 공간 데이터에 대한 빠른 검색을 위한 색인 구조로 설계되었다. 그러나 이동체의 위치 변경은 공간 색인 구조에서 삭제와 삽입으로 인해 공간 색인 구조의 빈번한 재구성으로 인해 성능이 현저히 떨어지는 단점이 있다.

본 논문에서는 이동체 데이터베이스를 위한 새로운 동적 공간 색인 구조를 제시하고, 동시 다발로 모든 이동체가 삭제되고 삽입될 때의 변경 연산 알고리즘, 그리고 동시다발로 변경되는 이동체 데이타베이스에서의 이동 객체의 검색 알고리즘을 설계하고 해석적 모델로 성능을 분석하고 구현된 동적 공간 색인 구조와의 성능 평가 실험으로 이를 검증하는 것을 목표로 한다. 성능 평가 실험 비교는 R*-tree[2], 고정 그리드 등에 대하여 수행한다.

이 논문의 구성은 다음과 같다. 2장에서는 이동체에 관한 관련 연구를 기술하고, 3장에서는 이동체 색인의 비용 모델을 기술한다. 4장에서는 실험을 수행하여 제안하는 동적 해싱 색인의 성능을 비교하였다. 마지막으로 5장에서는 결론을 기술한다.

II. 관련 연구

이동체가 능동적으로 보고한 위치는 시간이 지남에 따라 변하게 된다. 이러한 이유에서 이동 방향을 이용한 미래 위치 예측에 관한 연구[3, 4, 11]가 진행되었다. 하지만 이러한 연구는 이동체의 위치 에러에 관한 연구와 병행되어야 하기 때문에 이 연구의 주제를 벗어난다. 이 논문에서는 이동체의 현재 위치를 이동체가 보고한 가장 최근의 위치로 가정하였다.

최근에 이동체의 과거 궤적에 관한 연구[5]가 활발히 진행되었다. 이 논문은 이동체의 현재 위치를 저장하는 색인 구조에 관한 연구이기 때문에 소개하지 않겠다.

이동체의 위치는 2차원의 점으로 표현될 수 있으며, 공간 색인을 이용하여 색인할 경우에 다음과 같은 문제가 발생한다. 고정 격자 파일[6]은 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-tree[7]는 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. 이와 같은 문제의 원인은 공간 색인이 정적 데이터를 기반으로 설계되고, 검색이나 삽입과 같이 색인의 변경을 위한 연산이 별도로 정의되어 있지 않기 때문이다.

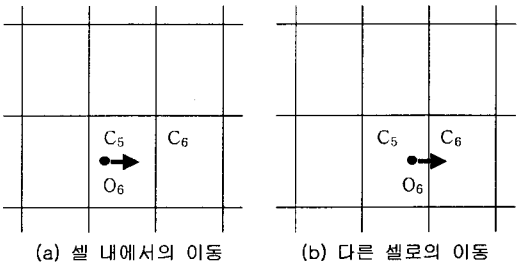


그림 1 이동체의 이동과 색인의 변화
Fig. 1. The movement of a moving object and the update of index. (a) the movement in a cell, (b) the movement across cells

이동체의 위치 변경으로 인한 빈번한 색인의 갱신 문제를 해싱(Hashing) 함수를 이용하여 줄일 수 있다는 접근 방법을 이용한 이동체 색인 연구[8, 12]가 제안되었

다. 이동체의 위치 변경 연산은 대부분 기존 객체의 변경으로 볼 수 있다. 기존의 공간 색인은 대부분 변경 연산보다는 검색 연산을 중점적으로 고려하였다.

이동체 데이터베이스에서는 빈번한 갱신 연산을 고려한 색인 구조가 필요하다. 공간 객체의 위치 변경은 트리 기반의 색인구조에서 삭제 후 재 삽입 연산으로 이루어진다. 삭제와 재 삽입 연산은 색인에서 병합과 분할을 초래하기 때문에 위치 변경이 빈번한 이동체 색인에서는 적합하지 않다.

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.

이동체가 셀의 경계 상에서 이동할 경우, 셀 간의 이동으로써 연속적인 색인의 갱신이 발생하는 문제가 있다. 이 경우 영역 간에 약간의 중복(overlap)을 허용하여 갱신 횟수를 줄일 수 있다. 이와 유사한 개념으로 LUR-tree[9]에서는 확장 MBR를 사용하여 경계에서 이동하는 이동체의 갱신 연산을 줄인다.

이동체는 연속적으로 위치가 변하기 때문에 시간에 따라 분포성이 변한다. 관련 연구[8]에서는 Quad-tree를 색인으로 사용하기 때문에 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.

LUR-tree(Lazy Update R-tree)[9]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 삭제 및 삽입 연산을 수행해야 한다.

기존의 대표적인 공간색인인 R-tree에서는 삭제 및 삽입 연산으로 최적의 검색이 가능한 색인을 구성할 수 있으나, 갱신 연산은 노드의 분할 또는 합병을 유발하기 때문에 비효율적이다. Lazy 갱신[9]은 이러한 문제점을 해결하기 위하여 이동체의 위치 변경으로 발생하는 갱신 연산을 노드 내에서의 갱신으로 제한함으로써 노드의 구조 변경에 따른 비용을 감소시켰다.

관련 연구[4, 11]은 미래 예측이 가능한 TRP-tree를 기본 구조로 하여 새로운 색인 구조를 제안하였다. 관련 연구[11]에서는 이동체 삽입에 따른 새로운 오버플로우 처리 기법을 제안하였고, 관련 연구[4]에서는 이중 구조를 이용한 메모리 버퍼링 기법을 통한 색인 성능 향상을 제안하였다. 이동체의 새로운 위치 삽입으로 인한 노드 중첩과 색인 구조 변경 비용이 발생하는 R-tree 기반 색인 구조는 비용 분석에서 제외한다.

Ⅲ. 이동체 색인의 비용 모델

이동체의 빈번한 위치 변경은 이동체를 저장하는 데이터베이스의 색인 구조 변경을 초래한다. 이 장에서는 이동체의 위치 변경에 따른 색인 변경 비용 모델을 구하여 고정 그리드와 제안하는 동적 해싱 색인의 성능을 계산해 보겠다.

3.1 비용 모델을 위한 기호

색인의 비용 모델을 구하기 위하여 기본적인 색인 구조를 자세히 살피고, 비용 모델을 위한 기호들을 정의한다. 고정 그리드[6]는 오버플로우 버켓을 사용하기 때문에 이동체의 밀집도가 성능에 영향을 미친다. 1차원 색인에서는 해쉬 함수를 사용하여 키들을 분산하여 저장할 수 있지만 2차원 좌표를 사용한 공간에서는 이동체의 위치에 따라 오버플로우가 많이 발생할 수 있다.

이러한 문제를 해결하기 위하여 1차원 색인에서 균형 트리와 해쉬 함수를 이용한 색인 방법과 같이 R-tree가 평균적인 검색 성능을 보장해 준다. 하지만 R-tree는 앞에서 설명한 것과 같이 이동체의 이동이 색인의 변경을 초래하는지 검색을 해 보지 않으면 알 수 없기 때문에 본 논문에서는 고정 셀 크기를 사용하는 색인 기법만을 고려한다.

고정 그리드는 데이터 페이지를 고정된 위치에 저장함으로서 디렉토리를 가지지 않아도 되지만 이럴 경우 빈 페이지를 만들어야 하는 단점이 있다. 본 논문에서는 페이지가 디스크 또는 메모리에 존재할 수 있기 때문에 디렉토리를 사용하는 방법을 사용한다. 디렉토리는 고정 그리드와 동적 해싱 방법 모두에서 같이 사용되기 때문에 성능 비교에서 고려하지 않는다. 또한 구현에서는 디렉토리를 메모리에 상주시켰다.

표 1. 비용 모델을 위한 기호
Table 1. Symbols of Cost Model

기호	의미
n	이동체의 총 개수
b	페이지 용량(페이지 당 이동체의 수)
P(i, j)	셀 Cell(i, j)의 확률
Num(i, j)	셀 Cell(i, j)의 이동체 수
k	셀 Cell(i, j)의 페이지 수

표 1은 이동체의 밀집도에 따른 색인들의 성능을 비교하기 위한 색인의 비용 모델을 분석하기 위한 기호들이다.

3.2 이동체 검색 비용 모델

공간에서 키 값은 좌표가 될 수 있다. 이 절에서는 좌표 값을 이용한 이동체 검색에서 고정 그리드와 동적 해싱 방법의 비용 모델을 비교한다. 앞에서 기술한 것과 같이 고정 그리드와 동적 해싱 방법을 위한 디렉토리를 메인 메모리에 상주시키기 때문에 데이터 페이지를 위한 I/O 회수가 성능 분석의 기준이 된다. 따라서 이동체 색인의 비용 모델은 I/O 회수를 구하는 것으로 한다.

1) 고정 그리드의 검색 비용 모델

한 페이지에 저장될 수 있는 최대 엔트리 수가 b라 하고, 셀 Cell(i, j)의 엔트리 수는 Num(i, j)이라 하자. 만약 Num(i, j) <= b이면 고정 그리드에서 오버플로우 버켓이 없고 데이터 버켓만 있기 때문에, I/O 수는 1번이다(데이터 버켓 읽기 1번). 만약 Num(i,j) > b and Num(i,j) <= 2b 이면, 최악의 I/O 수는 데이터 버켓 1번, 오버플로우 버켓 1번이다. 평균 검색 I/O 수는 (b + 2 * (Num(i,j) - b)) / Num(i,j)가 된다. 이를 정리하면 다음과 같다.

```
If Num(i,j) <= b ,
    then a point query requires one I/O.
    // only one for read datapage.
If Num(i,j) > b and Num(i,j) <= 2b,
    then one or two I/O for read datapage.
    // average => ( b + 2 * (Num(i,j) - b) ) / Num(i,j)
If Num(i,j) > 2b and Num(i,j) <= 3b.
    // average => ( b+2b+3* (Num(i,j)-2b) ) / Num(i,j)
```

데이터 페이지에 저장될 수 있는 최대 엔트리가 b라 하자. 셀 Cell(i, j)의 페이지 수는 수식 (1)과 같다.

$$k = Num(i,j) / b \qquad \text{- (1)}$$

주어진 이동체의 총 개수가 n개이라 하자. 셀 Cell(i, j)의 이동체 수는 수식 (2)와 같다.

$$Num(i,j) = P(i,j) * n \qquad \text{- (2)}$$

수식 (1)과 수식 (2)를 이용하여 평균 검색 I/O 수를 구하면 수식 (3)과 같다.

$$\begin{aligned}
&(\sum_{i}^{k-1}(b*i) + k(Num(i,j) - (k-1)b)) / Num(i,j) \\
&= (\frac{k(k-1)*b}{2} + k(Num(i,j) - (k-1)b)) / Num(i,j) \\
&= (k*Num(i,j) - \frac{k(k-1)*b}{2}) / Num(i,j) \\
&= k - \frac{k(k-1)*b}{2*Num(i,j)} \approx \frac{k+1}{2} = \frac{P(i,j)*n/b+1}{2} \qquad (3)
\end{aligned}$$

복잡한 수식으로 구하여 졌지만 수식 (3)에서와 같이 이동체가 밀집되어 있으면 평균적인 이동체 검색 I/O 수가 선형적으로 증가함을 알 수 있다.

2) 동적 해싱 구조의 검색 비용 모델

한 페이지에 저장될 수 있는 최대 엔트리 수가 b라 할 때, 셀 Cell(i, j)의 엔트리 수는 Num(i, j)이다. 만약 Num(i, j) <= b이면 고정 그리드에서 오버플로우 버켓이 없고 데이터 버켓만 있기 때문에, I/O 수는 1번이다(데이터 버켓 1번). 이것은 동적 해싱 방법에서 Num(i, j) <= b이면 고정 그리드 셀이기 때문이다. 고정그리드와 동일하다.

만약 Num(i,j) > b이면, I/O 수는 트리 색인 1번, 데이터 버켓 1번이다. 평균 I/O가 2가 된다. 동적 해싱 구조의 평균 검색 I/O 수를 정리하면 다음과 같다.

```
If Num(i,j) <= b,
    then a point query requires two I/O.
    // only one for read datapage.
If Num(i,j) > b,
    then two I/O for read datapage.
    // one is search tree, others is read datapage
```

위의 식에서와 같이 동적 해싱 구조는 이동체의 밀집화로 분할 될 셀에서 최대 I/O수를 2번이라 하였다. 이는

전제조건이 2차 주소인 트리가 루트가 분할되지 않았을 때의 성능이다. 공간을 1차로 고정 그리드로 분할 하였기 때문에 2차 주소에서 추가적인 I/O를 유발할 정도의 분할이 이루지지는 않는다.

고정 그리드와 동적 해싱 구조의 좌표 키를 이용한 검색 성능은 밀집화가 심하여 오버플로우가 많이 발생할수록 고정 그리드의 성능이 저하됨을 알 수 있다. 또한 동적 해싱 구조의 성능에서 트리 색인의 I/O가 성능 저하의 요인이 됨을 알 수 있었다. 오버플로우의 발생으로 분할 된 셀을 가리키기 위한 트리의 내용이 셀 내로 제한적이기 때문에 정보의 크기가 작았다. 이와 같은 이유로 본 논문에서는 트리를 메모리에 상주시키고, 변경이 발생한 경우에는 Write I/O가 발생하도록 하여 성능 향상을 도모하였다.

실험을 통하여 성능을 분석해 보면 이동체의 빈번한 위치 변경은 데이터 페이지의 많은 Write I/O를 유발하지만 트리의 변경은 상대적으로 작았다. 트리의 생성은 초기에 밀집화로 인하여 셀의 분할 시 발생하는 구조적인 변화이기 때문에, 밀집화가 사라지거나 다른 지역의 밀집화가 발생하였을 경우에 트리의 생성 및 삭제가 발생한다.

3.3 색인 구축 비용 모델

색인 구축 비용은 초기에 이동체들이 삽입 연산으로 색인에 저장되는 비용이다. 고정 그리드와 동적 해싱 방법은 R-트리 구조의 색인과 달리 오버플로우가 발생하기 전까지는 1번의 Write I/O로 이동체의 좌표를 저장하게 된다.

고정 그리드와 동적 해싱 방법은 오버플로우가 발생하지 않은 셀에서는 동일한 구조를 가지기 때문에 성능이 같다. 하지만 오버플로우가 발생한 셀에서는 서로 다른 성능을 보인다.

먼저 색인 구축 비용을 구하기 구해서는 삽입 연산을 살펴 보아야 한다. 색인을 초기에 구축하는 것은 공간 객체를 삽입하는 것과 같기 때문이다.

1) 고정 그리드의 삽입 연산 비용

오버플로우가 발생하지 않은 셀에서는 삽입 연산을 위한 I/O는 2번(read & write)이다. 오버플로우가 발생한 셀에서는 공간 객체의 삽입은 마지막 오버플로우 버켓의 마지막 엔트리에 삽입하기 때문에 항상 k+1(k번 read, 1번 write)이다. 이 때 k는 셀 Cell(i, j)의 페이지 수이고, k > 1이다.

2) 동적 해싱 방법의 삽입 연산 비용

오버플로우가 발생하지 않은 셀에서는 삽입 연산을 위한 I/O는 2번(read & write)이다. 오버플로우가 발생한 셀에서는 공간 객체의 삽입은 앞에서 설명한 것과 같이 트리 탐색 1번, 데이터 페이지 변경 I/O수 2번(1번 read & 1번 write) 이다.

고정 그리드와 동적 해싱 방법의 색인 구축 비용은 이동체의 밀집도에 따라 성능 비교할 수 있다. 만약, 고정 그리드의 셀 등 중에서 k(i, j) > 2인 셀이 많을 수록 동적 해싱 방법이 색인 구축 비용에서 성능이 우수하다.

k(i, j) > 2인 셀이 존재하지 않으면 동적 해싱 방법은 고정 그리드와 색인 구축 비용이 동일하다. 동적 해싱 구조의 검색 비용 모델에서 설명한 것과 같이 삽입 연산 비용에서도 트리 탐색 비용은 메모리에 캐싱(Caching) 함으로서 I/O 수를 감소 시킬 수 있다. 트리를 메모리에 캐싱하여 I/O 수를 1 줄이면 동적 해싱 방법은 고정 그리드에 비해 오버플로우가 발생한 셀에서 30% 이상 성능 향상이 있다.

3.4 이동체의 위치 변경에 따른 비용 모델

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

1) 셀의 크기에 따른 비용 모델

본 논문에서는 셀 내의 이동에 대해 색인의 변경이 발생하지 않는다는 특징을 고려하여 이동체 색인의 비용 모델을 구하고자 하였다. 고정 그리드에서 그림 31과 같이 셀이 크면 셀 간의 이동이기 때문에 색인의 변경이 발생하지 않는다. 하지만 셀의 크기를 크게 하면 오버플로우가 발생할 확률이 증가 하기 때문에 색인의 성능이 감소하게 된다. 이는 검색 비용 모델의 수식 (3)에서와 같다.

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.

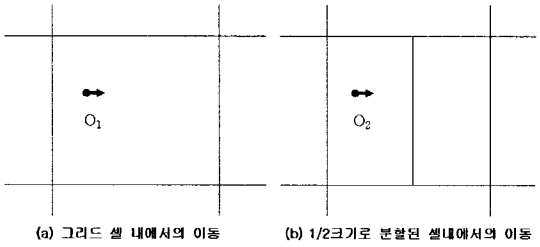


그림 2. 셀의 크기와 색인의 변화
Fig. 2. The size of a cell and the update of index. (a) The movement in a grid cell, (b) The movement in a half-size cell

고정 그리드는 이동체가 밀집된 지역에서 오버플로우 버켓이 발생하여 성능 저하를 초래한다. 이러한 문제점을 해결하기 위하여 동적 해싱 방법은 공간 분할을 수행하여 내부 트리로 분할된 셀들을 관리하여 검색 및 변경 연산에서 I/O 비용을 감소시켰다. 하지만 공간 분할을 수행하면 그림 2와 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 절에서 소개할 검색 성능 분석에서와 같이 공간 분할로 동적 해싱 방법을 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 본 논문에서는 공간 해싱 방법에서 색인 변경 비용을 줄이는 방법으로 문제를 해결하였다. 해결책은 앞에서 설명한 오버플로우가 발생한 셀에 존재하는 내부 트리를 메인 메모리에 캐싱하는 것이다.

3.5 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 3에서와 같은 조건에서 질의 영역을 작게 하여 셀 들간의 경계에서 질의를 하면 표 2와 같은 영역 질의에 따른 대략적인 I/O 수를 구할 수 있다. 오버플로우 발생한 셀은 Co로, 오버플로우가 발생하지 않은 셀은 Cg로 표시하였다.

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

영역질의의 종류	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

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.

Ⅳ. 실 험

이 장에서는 논문에서 제안하는 동적 해싱 색인 구조의 성능을 평가하기 위해 이동체의 색인 중에서 고정 그리드, R*-tree와 성능을 비교하고자 한다. 색인의 성능 비교를 위하여 색인의 생성 비용과 색인 변경 비용, 질의 처리 성능을 실험을 통해 비교하였다.

본 논문에서는 이동체 생성에 대한 대표적인 도구인 GSTD 생성기[11]를 사용하여 다양한 조건에서 이동체의 위치 데이터를 생성하여 실험하였다.

4.1 색인 구축을 위한 인자

고정 그리드와 동적 해싱 방법은 1차 주소를 해싱 함수를 사용하는 것이 동일하기 때문에 우선 공간을 고정 격자로 나누어야 한다. 3장에서 설명하였듯이 셀의 갯수는 n/b가 이상적이다. 이 때 전체 이동체의 수는 n이고, 데이터 페이지의 최대 엔트리 수는 b이다.

표 3. 이동체의 수에 따른 셀의 수
Table 3. The number of cells versus the number of moving objects

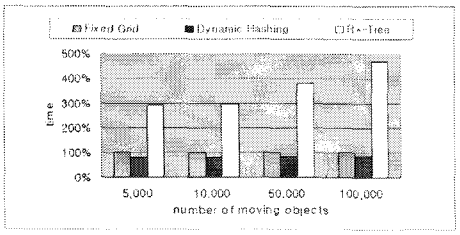
이동체수	2,500	5,000	10,000	20,000
셀의 총 개수	50	100	200	400
각 축당 셀의 수	7	10	14	20

데이터 페이지의 크기는 고정되어 있다. 따라서 페이지 당 최대 허용 엔트리 수도 고정되어 있다. 그래서 이동체의 수가 증가하면 셀의 수도 증가하게 되고, 셀의 면적도 줄어들게 된다. 표 3은 페이지 당 최대 허용 엔트리 수가 50일 때 이동체 수에 따른 색인에서의 셀의 수와 각 축당 셀의 수로 보인다.

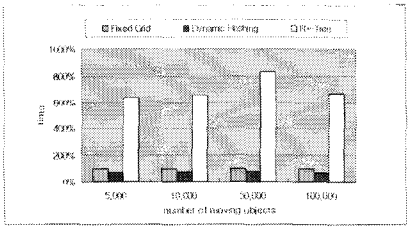
동적 해싱 방법은 오버플로우를 내부 트리를 사용하여 처리하기 때문에, 오버플로우에 의한 성능 저하 요인이 작다. 이 장에서는 동적 해싱 방법을 2가지로 색인 구조를 만들어 실험한다.

㉠ DH-1
- 표 3과 같이 셀을 생성한다.
- 실험에 사용되는 고정그리드와 같은 조건이다.

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



(a) 초기 색인 구축 비용



(b) 색인 변경 비용

그림 3. 이동체의 개수에 따른 성능 비교
Fig. 3. Performance evaluation for varying the number of moving objects. (a) Costs for building indexes, (b) Costs for updating indexes

4.2 색인 성능의 시간 비교

3장에서 동적 해싱 방법을 내부 트리를 메인 메모리에 캐싱(Caching)하여 성능 향상을 도모한 것을 제시하였다. 이 절에서는 강제적인 내부 트리의 메모리 캐싱 방법이 아닌 운영체제의 캐싱 효과를 측정하기 위하여 시간만을 측정하여 보았다.

그림 3은 이동체의 수를 5000에서 10만까지 달리하여 이동체의 이동에 따른 색인의 성능을 비교하였다. R*-tree는 색인의 빈번한 삽입 삭제 연산으로 인하여 해싱 기반의 공간 분할 방식의 색인에 비해 상대적으로 연산비용이 높았다. 제안하는 동적 해싱 방법은 색인 구축 비용은 고정 그리드에 비해 10%-20% 성능 향상이 있었다. 나머지 실험에서는 20%-30% 성능 향상이 있었다.

그림 3과 같이 R*-tree는 색인 변경 연산을 위하여 비단말 노드의 변경 전파로 인한 노드 변경 연산의 증가로 색인의 현재위치를 저장하는 이동체 색인으로 적합하지 않았다. 이후의 실험에서는 고정 그리드와 동적 해싱 색인만을 비교한다.

4.3 색인 구축 및 변경 연산의 비교

이 절에서는 동적 해싱 방법의 색인 구축와 변경 연산에 관련하여 노드 접근 회수를 비교하여 성능 비교를 한다. 4.2절에서는 오버플로우가 발생한 셀에서 내부 트리를 디스크에서 읽어 운영체제의 캐싱을 이용한 성능 비교를 하였지만, 이 절 부터는 내부 트리를 메모리에 직접 캐싱하여 동적 해싱 방법의 최대 성능 향상을 검증한다.

그림 4는 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.

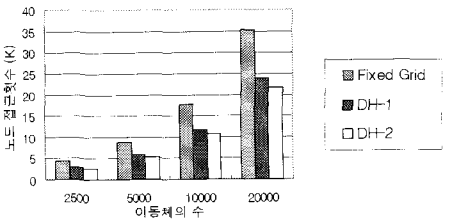


그림 4. 색인 구축을 위한 노드 접근 횟수
Fig. 4. The number of node accesses for building indexes

4.4 영역 질의의 성능 평가

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

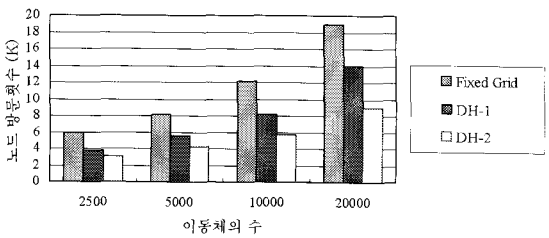


그림 5. 영역 질의 성능 비교
Fig. 5. Performance evaluation for processing range queries in each dimension

그림 5에서와 같이 10% 질의에서는 검색 성능 향상폭이 작아진다. 이는 질의 영역이 넓어지기 때문에 작은 질의(5%)에 비해 상대적으로 겹치는 영역이 증가하여 셀을 쪼개서 관리하는 효과가 작아지기 때문이다. 또한 DH-1의 데이터 페이지가 많은 것에 대한 비용 증가가 영향을 미친다.

V. 결 론

제안하는 동적 해싱 색인에서는 이동체의 현재 위치를 2차원의 점으로 정의하고, 빈번한 이동체의 위치 변경으로 발생하는 색인의 갱신 연산 비용을 해쉬 기법을 사용하여 줄였으며, 해쉬 기법에서 발생하는 오버플로우 처리를 새로운 트리 구조와 알고리즘을 제시하여 갱신 성능을 향상시켰다. 실험 평가에서 기존의 고정 그리드에 비해 20% 이상의 성능 향상이 있었다.

이동체 색인의 색인 구축 비용 모델, 변경 연산 비용 모델, 영역 질의 비용 모델을 분석하여 빈번한 위치 변경에 적합한 이동체 색인 구조를 제시하고, 이를 검정하기 위하여 성능 평가를 실시하였다.

실험 평가에서 현재 이동체 실험 데이터 생성에 주로 사용되는 GSTD 알고리즘을 이용한 도구를 이용하여 대량의 위치 데이터를 생성하여 실험하였다.

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] 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.

[3] S. Saltenis, C. S. Jensen, S.T. Leutenegger, and M. A. Lopez, "Indexing the positions of continuously moving objects," Proc. of the ACM SIGMOD Int'l Conf. on Management of Data, pp. 331-342, 2000.

[4] B. Cui, D. Lin, and K. L. Tan, "IMPACT: A Twin-index Framework for Efficient Moving Object Query Processing," Data & Knowledge Engineering, Vol. 59, pp. 63-85, 2006.

[5] B. Jun, B. Hong, and B. Yu, "Dynamic splitting policies of the adaptive 3DR-tree for indexing continuously moving objects," Int'l Conf. on Database and Expert Systems Applications, September, 2003.

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

[6] A. 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.

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

[8] 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.

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

[10] 복경수, 유재수, "이동 객체의 미래 위치 검색을 위한 공간 분할 방식의 색인 구조, "충북대학교 컴퓨터정보통신연구소 논문집, 제13권, 제1호, 2005.

[11] 전봉기, 홍봉희, "이동체 데이타베이스를 위한 해쉬 기반의 공간 색인," 한국정보과학회 학술발표논문집, 제28권, 제2호, pp. 205-207, 2001.

저자소개

전 봉 기(Bong-Gi Jun)

1991년 부산대학교 컴퓨터공학과 공학사
1993년 부산대학교 컴퓨터공학과 공학석사
2003년 부산대학교 컴퓨터공학과 공학박사
1993년 ~ 1998년 한국통신 연구소 전임연구원
2003년 ~ 현재 신라대학교 컴퓨터정보공학부 조교수
※관심분야 : 데이터베이스, 공간 데이터베이스, 이동체 데이터베이스.