이동 객체의 현재 위치 색인 기법†

On Indexing Method for Current Positions of Moving Objects

박현규(朴顯奎)*, 강성탁(姜成卓)**, 김명호(金命鎬)*, 민경욱(閔庚旭)***
Hyun-Kyoo Park, Sung-Tak Kang, Myoung-Ho Kim, Kyoung-Wook Min

요 약 위치 기반 서비스는 이동 통신 단말기의 위치를 GPS 등을 이용하여 실시간 위치 정보와 관련된 정보를 제공하는 응용 서비스로서 이동 통신망의 고도화에 따라 교통, 물류, 전자 상거래 등의 분야에서 시공간 데이터베이스의 주요한 응용 영역으로 부각되고 있다.

속도와 이동 방향과 같은 궤적 정보가 제공되지 않는 이동 통신 환경에서 지속적으로 변화하는 이동 객체의 위치를 효율적으로 색인하는 방법으로 본 논문에서는 *A-Quadtree*를 제시한다. *A-Quadtree*는 2차원 위치 정보를 실시간 갱신을 지원하는 구조로서 객체 식별자에 대한 색인과 통합하여 .Net 콤포넌트로서 구현함으로써 다양한 플랫폼에 적용이 가능하도록 하였다. 또한 실험을 통하여 *A-Quadtree*는 기존의 다차원 색인 구조보다 갱신과 검색 성능이 효율적임을 보인다.

ABSTRACT Location-based service is an important spatiotemporal database application area that provides the location-aware information of wireless terminals via positioning devices such as GPS. With the rapid advances of wireless communiation systems, the requirement of mobile application areas including traffic, mobile commerce and supply chaining management became the center of attention for various research issues in spatiotemporal databases.

In this paper we present the A-Quadtree, an efficient indexing method for answering location-based queries where the movement vector information (e.g., speed and velocity) is not presented. We implement the A-Quadtree with an index structure for object identifiers as a .Net component to apply the component to multiplatforms. We present our approach and describe the performance evaluation through various experiments. In our experiments, we compare the performance with previous approaches and show the enhanced efficiency of our method.

주요어 : 위치 기반 서비스, 이동 객체, 시공간 데이터베이스, 색인, 공간 분할
Key word : Location-based Service, Moving Object, Spatiotemporal Database, Indexing, Space Partitioning

1. 서 론

급속한 이동 통신의 발달과 실시간 위치 정보 제공 기술이 보편화되면서 위치 기반 서비스(LBS)가 요구되는 다양한 응용 영역이 제기되고 있으며, 시공간(Spatiotemporal) 데이터베이스는 이러한 서비스를 위한 기반 기술로 최근 많은 연구가 이루어 지고 있다.

특히 GPS에 의한 실시간 위치 정보의 획득이 저렴한 비용으로 가능해 짐에 따라 위치 정보를 제공하는 이동 통신 단말기가 늘어나면서 이동 객체의 위치 정보를 다루기 위한 데이터베이스를 이동 객체 데이터베이스(MODB)라고 한다 [6]. 이동 객체 데이터베이스는 현재 또는 과거 위치 정보를 이용하여 전자상거래, 교통 관리, 군사 목적의 지휘통제 시스템 등에서 활용

† 본 연구는 한국전자통신연구원의 개방형 LBS 핵심기술 개발과제의 부분적인 지원을 받아 수행되었음
* 한국과학기술원 전자전산학과 전산학 전공 {hkpark, mhkim}@dbserver.kaist.ac.kr
** LG전자 GSM연구소 stkang@dbserver.kaist.ac.kr
*** 한국전자통신연구원 GIS연구팀 kwmin92@etri.re.kr

할 수 있으며, 지속적인 위치 변화를 실시간에 반영할 수 있는 시공간 색인 기법이 요구된다 [1, 2].

본 논문에서 다루고자 하는 위치 정보와 시공간 질의는 이동 객체의 현재 위치를 대상으로 하는 점(Point) 객체 질의와 특정 영역(Range)에 존재하는 객체를 검색하는 영역 질의로 구분된다. 예를 들어 현재 나의 위치에서 1Km 이내에 존재하는 주유소를 검색과 같은 질의와 시청으로부터 반경 1Km 이내에 존재하는 순찰차를 검색의 질의를 처리하기 위해서는 이동 객체의 식별자(ID)에 대한 색인과 위치 정보 색인이 필요하며, 각각의 색인은 갱신과 검색이 동시에 이루어 질 수 있어야 한다.

본 논문에서 제시한 색인 구조는 객체 식별자 색인을 위한 B-tree와 위치 정보 색인 구조 *A-Quadtree*로 이루어져 있으며, 각 색인은 지속적으로 변화하는 위치 정보의 갱신과 검색 질의 처리를 동시에 처리하도록 하였다. 또한 객체 식별자에 대한 색인과 위치 정보에 대한 시공간 색인은 이식성과 융통성을 높일 수 있도록 .Net 프레임워크에서 적용 가능한 컴포넌트(Component)로서 구현하였다.

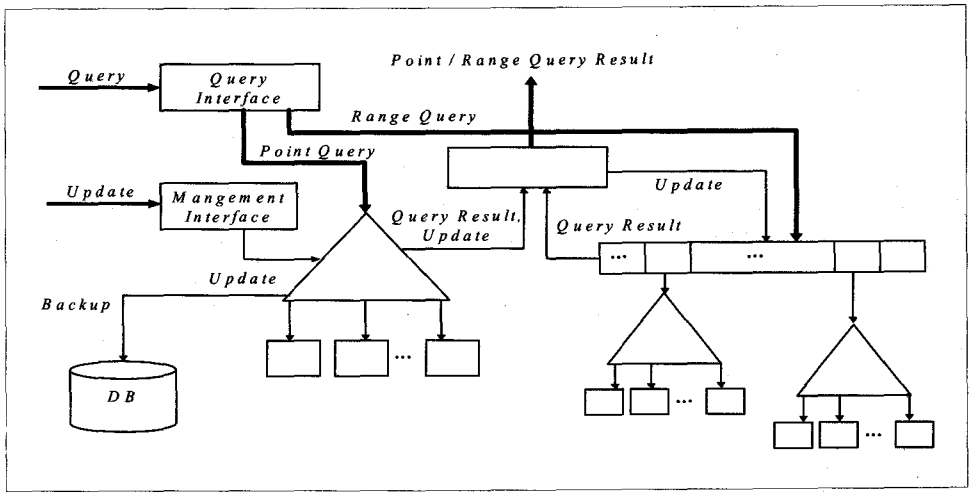
논문의 구성은 다음과 같다. 2장에서는 문제 정의와 연구 동기를 기술하고, 3장은 기존의 관련 연구 및 차이점, 4장은 현재 위치 색인 컴포넌트 구조와 이동 객체에 대한 색인 및 질의 처리 모델을 설명하였다. 5장은 *A-Quadtree* 색인 구조 및 기능, 6장은 제시된 색인의 실험에 의한 기존 연구와의 비교 및 성능 분석, 그리고 7장에서는 결론 및 향후 연구 과제를 제시하였다.

2. 문제 정의

위치 기반 서비스를 위한 위치 관리 모델(Location Management Model)은 점 위치 관리(Point Location Management)와 궤적 위치 관리(Trajectory Location Management)로 구분할 수 있으며[3], 본 논문에서는 이동 객체의 방향과 속도 벡터 정보가 제공되지 않으며 형태의 변화가 없는 점 위치 정보에 대한 색인과 질의 처리를 다룬다.

객체의 속도와 방향 등의 이동 정보를 이용하여 현재 위치를 추정하는 방법이나, 추정에 의한 질의 처리는 본 연구에서는 배제하며, 점 정보로 표현되는 이동 객체의 현재 위치는 데이터베이스에서 마지막 위치 정보가 저장된 시간에서의 정보를 현재 위치로 가정하므로 갱신 주기를 짧게 해야 위치 정보의 불확실성(Uncertainty)을 줄일 수 있다. 따라서 위치 기반 시스템은 실시간 응답이 요구되는 분야이므로 점 정보로 표현되는 현재 위치 정보에 대한 색인은 지속적으로 변화하는 위치 정보를 데이터베이스에 실시간에 반영하며, 영역 검색 질의가 효율적으로 처리될 수 있어야 한다.

이동 객체의 위치 정보를 관리하기 위한 방법으로 공간 분할 방법이 다차원 색인 방법보다 갱신 문제에 대해서는 효율적으로 알려져 있으나[4, 5], 다양한 객체의 이동 형태와 질의 처리를 반영하기 위해서는 색인 방법이 효율적인 경우가 있으며 기존 색인 방법의 개선이 필요하다. 따라서 본 논문에서 제시하는 방법과 기존 연구 결과와 다양한 데이터 셋과 질의 처리에 따른 비교가 필요하다. 그리고 색인 구조는 위치 기반



〈그림 1〉 현재 위치 정보 색인 및 질의 처리 컴포넌트

시스템에서 바로 적용 가능해야하므로 기존 시스템이나 데이터베이스에 추가(Add-On)할 수 있는 컴포넌트 소프트웨어로 구현할 필요가 있다[6].

3. 관련 연구

위치 기반 서비스를 위한 이동 객체 관리 기술은 위치 획득, 위치 저장 그리고 위치 검색 기술로 구분될 수 있으며[6], 이동 객체의 위치 정보에 대한 색인은 갱신 빈도를 줄이기 위한 방법으로 객체의 위치 정보를 선형 함수로 표현하여 색인하는 방법을 제시하고 있다[1, 2]. 이러한 방법은 이동 객체의 이동 방향과 속도 등이 알려져 있는 경우 또는 궤적 위치 관리를 이용한 미래 질의를 다루는 분야 등에 적용 가능하며, *R-tree*기반의 대표적인 색인 구조로서 *TPR-tree*[7] 등이 제시되고 있다.

이동 객체의 위치 정보를 기반으로 하는 시공간 질의는[3, 8] 등에서 제시되고 정형화되었으며, 공간(Volume) 정보가 없는 점 객체에 대한 영역 질의는 현재 객체의 위치 정보의 불확실성을 최소화시키는 것이 주요한 과제이다. 그러나 [1]에서 제시한 방법은 데이터베이스 갱신 문제를 부분적으로 해결하는 것으로 실제 환경에서 객체의 이동을 적절히 표현할 수 있는 이동 함수를 찾는 것은 매우 어렵다.

궤적 정보가 없는 이동 객체의 현재 위치는 점 정보이며, 점 위치 관리를 다루는 현재 위치 정보에 대한 *R-tree* 기반 색인 방법으로는 [9]에서 제시한 LUR-tree가 있다. 그러나 *LUR-tree*는 MBR 설정과 분할 방법을 개선함으로써 갱신 효율을 향상시켰으나, 검색 성능을 오히려 *R*-tree* 보다 떨어지는 것으로 알려져 있다.

따라서 이동 객체에 대한 색인에 의한 방법보다 분할 및 *Hashing*에 의한 방법 [4, 5] 등이 제시되었으며, 특히 이동 객체의 정확한 위치 정보를 색인하지 않고 객체가 속하는 공간 정보를 이용하는 *Hashing* 기반의 방법 [4]은 저장 공간 활용과 검색에서 색인 방법보다 효율적이다. 그러나 2차원 공간에서의 최적 공간 분할은 NP-Hard 문제로서 알려져 있다 [10]. 이진 공간 분할(Binary Space Partition)의 경우 계산 복잡도는 $O(n \log n)$ 이지만, 갱신이 일어나는 경우 분할된 공간내의 객체 수가 변화하므로 공간 분할의 특성을 반영하지 못하거나 분할 공간의 조정의 복잡도가 증가하게 된다. 이러한 방법은 공간상의 객체들의 분포가 치우치는 경우 분할된 공간의 오버 플로우가 발생하므로 검색 성능 또한 저하된다.

본 연구에서 제시하는 방법은 공간 분할에 의한 색인을 다루는 *Quad-tree* 기반의 색인으로, 이동 객체의 궤적을 *Quad-tree* 기반의 색인 방법을 이용한 [11]의 연구가 있다. 본 논문에서 제시하는 색인은 [11, 12, 13]에서 다루어진 *Quad-tree*를 이용하여 공간 분할과 색인 관리를 개선함으로써 동일 위치에 존재하는 객체들로부터 발생하는 오버플로우(Overflow), 갱신 방법을 개선하였다.

4. 시스템 구조 및 색인 모델

본 논문에서 다루고자 하는 이동 객체는 (o-id, x_i, y_i)의 3가지 속성으로 이루어진 점 객체이며, 속성 값은 각각 객체의 식별자, x좌표, y좌표이다. 위치 기반 서비스를 위한 질의는 객체의 식별자에 대한 현재 위치 질의와 공간 영역에 존재하는 객체 질의가 있으며, 이를 위하여 객체 식별자에 대한 색인과 좌표에 대한 색인이 동일한 정보를 유지해야 한다. 그러므로 위치 정보의 갱신은 객체 식별자를 이용하여 과거 좌표 정보를 검색한 후 이를 이용하여 공간 색인의 자료를 현재 좌표로 갱신해야 한다.

4.1 현재 위치 색인 시스템

색인 구조는 외부 인터페이스를 통하여 삽입, 갱신, 삭제를 비롯하여 질의 처리가 이루어진다. 즉, 객체의 위치를 검색하는 점 질의 경우 객체 식별자를 이용한 위치 정보를 검색하는 색인을 검색하고, 영역 질의의 경우에는 2차원 공간 영역에 대한 색인을 검색한다. 그러므로 검색 결과는 객체의 위치 정보 또는 일정한 영역내에 존재하는 객체의 식별자 집합이 된다.

색인에 대한 삽입과 갱신 요구는 객체의 식별자에 대한 색인을 검색하여 갱신 전 좌표를 확인한 후 객체의 위치 정보를 현재 좌표로 갱신한다. 그리고 검색한 갱신 전 좌표를 이용하여 위치 정보 색인의 좌표위치 정보를 갱신한다.

질의 인터페이스 및 색인 구조는 모두 실시간 처리를 위하여 주 기억장치에 상주하고 페이지 단위로 정보를 저장, 검색한다. 주 기억장치를 이용한 컴포넌트는 시스템 장애에 대한 대비가 필요하므로 색인 내용의 디스크 기반 백업(Backup) 기능이 필요하며 백업 시스템은 동일한 시스템 또는 네트워크를 이용한 별도 시스템을 이용할 수 있다.

컴포넌트가 적용되는 시스템은 *Microsoft .Net* 프레임워크를 이용하며, 현재 색인 컴포넌트를 이용하여 얻어지는 질의 처리 결과는 .Net을 이용하는 데이터

베이스 또는 분산 컴포넌트에서 사용할 수 있다.

4.2 이동 객체 모델 및 색인

점으로 표현되는 이동 객체의 현재 위치 정보는 2차원 점 정보이며 [14], 다차원 색인 구조를 이용하여 색인하거나, 공간 분할과 *Hashing*을 이용하면 갱신이 이루어 질 때마다 색인을 잠금(Locking)함으로써 갱신 빈도가 높아지면 검색 성능이 떨어지는 비효율이 발생한다.

다차원 색인을 이용하여 색인에 대한 잠금 없이 갱신을 허용하기 위해서는 적절히 공간을 분할하여 검색을 최대한 허용할 수 있도록 색인 구조의 비균형(Unbalance)을 허용할 필요가 있다 [10]. 비균형 기반 색인은 이동 객체의 위치 변화가 임의 발생하는 일반적인 환경에서 색인의 재구성을 매번 갱신 때 수행하지 않고 적절한 기준 범위를 벗어날 때 색인을 재구성 하도록 함으로써 갱신 오버헤드를 줄일 수 있다.

2차원 점 정보를 색인하는 *Region Quad-tree* [12]는 *Quad-tree*영역을 적당한 객체를 포함할 때까지 분할하여 최종적으로 분할된 공간을 단말 노드에 할당한다. 이 경우, 〈그림 2〉와 같이 공간을 분할하여 각 형제(Sibling) 노드는 동일한 크기의 영역을 표현한다.

일반적으로 *Quad-tree*의 높이(Height)는 $O(\log n)$이 되지만[12], 공간을 분할하는 *Quad-tree*의 경우 데이터의 분포에 따라 트리의 높이가 변화하게 된다. 〈그림 2 (d)〉와 같이 (2, 3, 4) 영역에 객체가 많이 존재하고 다른 영역은 희박하게 존재하는 경우 *Quad-tree*의 높이는 매우 증가할 수 있다.

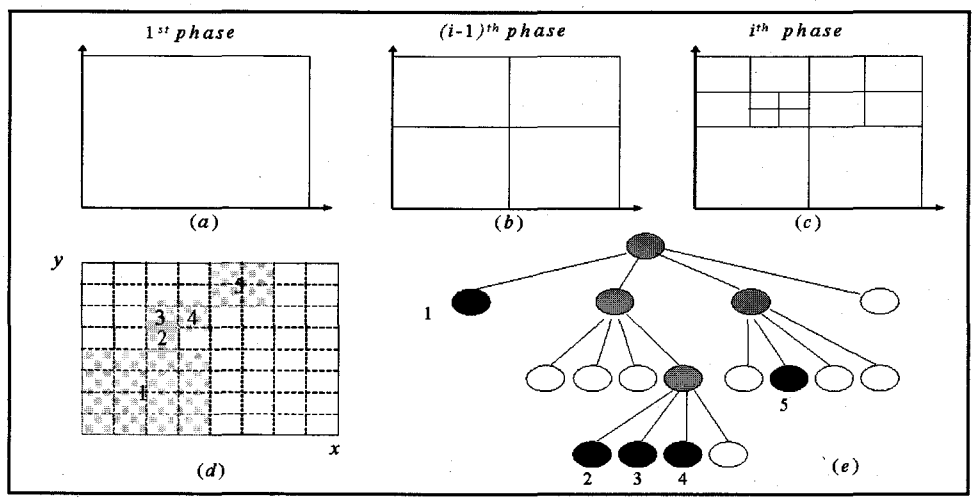
5. 색인 구조

5.1 색인 구조 및 질의 처리

4장에서 설명되어진 이동 객체의 특성을 반영하는 색인 구조로서 논문에서는 객체 식별자 색인을 위한 *B-tree*와 위치 색인 *A(Adaptive)-Quadtree*를 제시하고, Hash 백업을 현재 위치 기반 색인 컴포넌트로 구현하였다. 이동 객체를 다루는 *A-Quadtree*는 색인의 높이와 압축을 Quad-tree의 알고리즘을 적응적(Adaptive)으로 수행함으로써 색인 관리 성능을 향상시켰다.

객체 삽입과 갱신을 위해서 객체 식별자 색인인 *B-tree*와 *A-Quadtree*의 삽입과 갱신이 순차적으로 일어나며, *A-Quadtree*를 이용한 영역 질의는 영역을 검색한 후 질의 조건을 만족하는 객체를 검색하는 하향(Top-Down) 형태로 이루어진다.

색인 내용은 필요시 실시간 백업이 동시에 이루어지도록 하며 백업의 수행은 *B-tree*의 내용과 *A-Quadtree*의 내용이 동일하므로 *A-Quadtree*에 대한 동작과 동시에 수행될 수 있다. 또한 색인은 컴포넌트화 되어 있으므로 셀(Cell)로 분리된 PCS 가입자망과 같은 환경에서는 다수의 컴포넌트가 동작할 수 있도록 색인 구조를 클래스(Class)화 함으로써 공간 분할에 의한 병행성을 증가시킬 수 있다.



〈그림 2〉 Quad-tree에 의한 공간 분할 및 색인

5.2 삽입 및 오버플로우(Overflow) 처리

객체 식별자에 의한 색인은 고유한 키 값을 가지므로 단말 노드에서 동일한 키 값에 의한 오버플로우가 발생하지 않지만, 위치 정보에 대한 색인은 위치 해상도(Resolution)가 낮은 경우 또는 고층 빌딩 내부의 객체들의 경우 동일한 좌표로 표시되는 객체가 많아짐으로써 노드 분할을 하지 못하는 오버플로우가 발생한다.

*A-Quadtree*에서는 동일한 키 값에 의한 노드 오버플로우를 처리할 수 있도록 <그림 3>과 같이 단말 노드에서는 객체 식별자를 C# 언어가 제공하는 *ArrayList* 타입의 *Abstraction Set*으로 함으로써 데이터와 별도로 관리할 수 있도록 하였다. 위치 기반 시스템의 특성은 객체의 위치 정보 색인과 식별자 색인이 별도로 운용되므로 이러한 방법을 사용함으로써 객체의 갱신과 검색 성능을 기존의 노드를 링크(Link)하는 방법보다 효율적으로 개선할 수 있다.

5.3 삭제 및 압축(Packing) 처리

이동 객체의 위치 정보에 대한 갱신은 삭제와 삽입이 동시에 일어나므로 삭제가 발생하는 노드의 하위 구조는 재구성이 필요하며, 단위 시간의 위치 변화가 큰 경우 단말 노드의 병합을 통한 빈 노드를 제거는 색인의 효율을 매우 감소시킨다. 예를 들어 출퇴근 시간의 도심과 주거 지역 사이의 이동 객체의 변화는 특정 공간에서는 대부분 삭제가 발생하고 다른 공간에서는 삽입이 집중적으로 발생한다. 이러한 경우 색인 구조에 큰 변화를 가져오며 노드 병합과 분할에 따른 갱신 부담이 증가한다.

*A-Quadtree*의 단말 노드는 전체 객체의 수 N, 노드 크기 B일 때, $\lceil (N/B) \rceil$ 페이지가 필요하며, 병합은 $\lceil \log \lceil (N/B) \rceil$의 수행이 일어난다. 마지막 병합이 일어나는 페이지는 병합에 의한 삽입이 같은 페이지에서 일어나므로 실제 병합 과정은 $\lceil \log \lceil (N/B) \rceil \rceil - 1$ 이 된다. 또한 모든 페이지에 대하여 일기와 쓰기 과정이 일어나므로 색인 갱신에 의한 재구성 비용은 재구성이 일어나는 최상위 노드로부터 단말 노드까지 다음과 같다.

재구성비용(*Reorganization* cost)

$$= \lceil (N/B) \rceil \times (\lceil \log \lceil (N/B) \rceil \rceil) \times 2 \qquad (1)$$

이동 객체의 위치 변화가 임의 변화라고 가정하면 4.2절에서와 같이 색인의 높이는 $\lceil \log_B N \rceil$ 이 되므로 전체 노드의 페이지 비용은

$$\text{페이지 비용} = \sum_{h=1}^{\lceil \log_B N \rceil} 4 \times \lceil N/B^h \rceil \qquad (2)$$

따라서 전체 비용은 공식 (1)과 (2)로부터 다음과 같다.

$$\text{전체 비용} = = \lceil (N/B) \rceil \times (\lceil \log \lceil (N/B) \rceil \rceil - 1) \times 2 + \sum_{h=1}^{\lceil \log_B N \rceil} 4 \times \lceil N/B^h \rceil \qquad (3)$$

이동 객체 위치 정보에 대한 갱신은 연속적으로 수행되는 갱신에 의한 노드의 효율성이 일정 수준으로 감소될 때 압축을 통하여 효율성을 회복하는 것이 필요하며, 이 때 적응적으로 전체 비용을 최소화할 수 있도록 해야한다. 객체의 분포 정도에 따라 노드의 압축은 갱신이 일어날 때 처리되지 않고, 적절한 조건을 달성할 때 이루어지도록 함으로써 전체 성능을 향상시

```
Insert (QuadTree /* Parent */, Data /* 삽입 객체 */)
        if (localIndex>=0)  /* 동일 위치 객체 존재 시 */
                Insert IDSet; /* 동일 노드의 ID Set에 객체 추가 */
        else /* 동일 위치 객체 미존재 : 노드 분할 또는 단말 노드 삽입 */
                if (Node Full) /* 노드 분할 */
                        Node Split; /* Split 이후 Insert 호출 */
                        if ( (height+1) > parent.height )
                                (Parent.Height)++;
                else    /* 단말 노드의 끝 부분에 삽입 */
                        Insert Data;
```

<그림 3> Quad-tree 삽입 알고리즘

킬 수 있으며 압축이 일어나는 시간을 선정하는 것은 조정 변수(Tuning Parameter)가 된다.

5.4 회복 및 장애 처리

현재 색인 컴포넌트는 주기억장치 상주 시스템이므로 장애(Failure)에 대한 대비가 필요하지만, 장애 및 회복이 시스템의 성능에 미치는 영향을 최소화시켜야 한다. 따라서 색인된 객체 정보는 디스크 기반의 온라인(On-line) 백업(Backup)이 필요하며, 갱신 또한 백업 내용에 실시간으로 반영될 수 있어야 한다.

갱신을 실시간에 백업 내용에 반영하기 위해서는 객체 식별자에 의한 검색과 갱신이 필요하며 이러한 목적으로 시스템에서 사용할 수 있는 *Hash* 기반의 백업 기능을 구현하였다. *Hash* 백업은 이동 객체의 레코드에서 키 값인 객체 식별자를 *Hash* 함수를 이용하여 *Hash* 테이블에 저장한 후 이를 파일 스트림(Stream)으로 직렬화(Serialization)하여 이진 파일(Binary File)로서 디스크에 실시간 백업이 이루어지도록 하였다.

Hash 함수를 이용한 백업은 디스크의 객체를 키 값을 이용하여 검색하므로 빠른 검색과 갱신이 가능하며 다수의 갱신이 디스크에 반영되는 경우 직렬화된 이진 스트림을 처리하므로 시스템의 부담이 감소되고, 별도의 시스템을 이용하여 백업을 할 수 있는 융통성을 제공한다.

6. 실험 결과

6.1 구현 환경 및 실험 데이터

논문에서 제시한 *Quad-tree* 및 *R*-tree*, *B-tree* 등은 모두 *Visual Studio .NET*에서 C# 언어를 이용하여 구현하였고, 펜티엄 IV 2.4GHz CPU, 512MB 주기억장치를 사용하는 *Windows XP* 시스템에서 실험을 수행하였다. 실험에 사용된 데이터는 *citySimulator* [15]를 이용하여 생성하였으며, 조건은 도로 48, 건물 71, 교차로 6, 공원 1로 구성된 환경에서 500,000의 이동 통신 사용자를 모의하도록 하였다. 생성된 데이터는 *x*, *y*축에 대하여 각각 최초 0~1,000 미터 사이의 값을 가지므로 동일한 데이터를 0~100,000 미터의 공간으로 확대하여 각각의 데이터 셋에 대한 2가지 밀집도를 가지는 데이터 셋을 구성하였다.

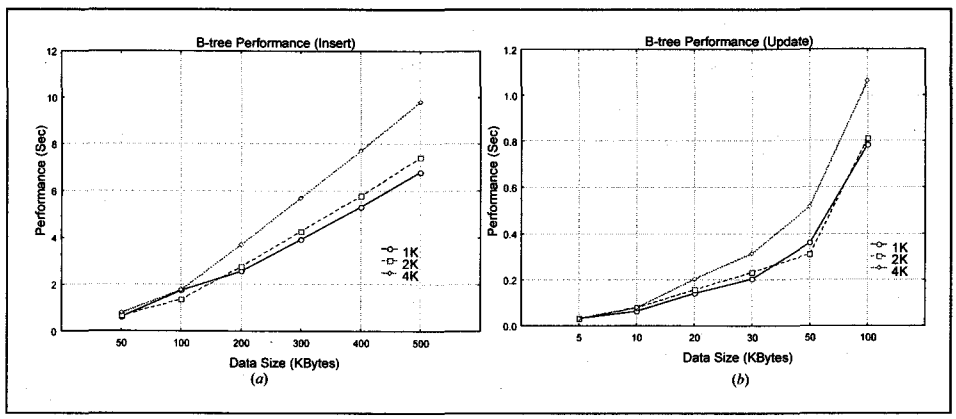
6.2 색인 삽입 및 갱신 성능

R-tree*는 위치 정보를 표현하는 대표적인 다차원 색인 구조로서 대부분의 공간 데이터베이스에서 색인 구조로 사용되므로 *R*-tree*와의 성능 비교를 통하여 논문에서 제시한 *A-Quadtree*의 성능을 평가하는 적절한 척도가 될 수 있다.

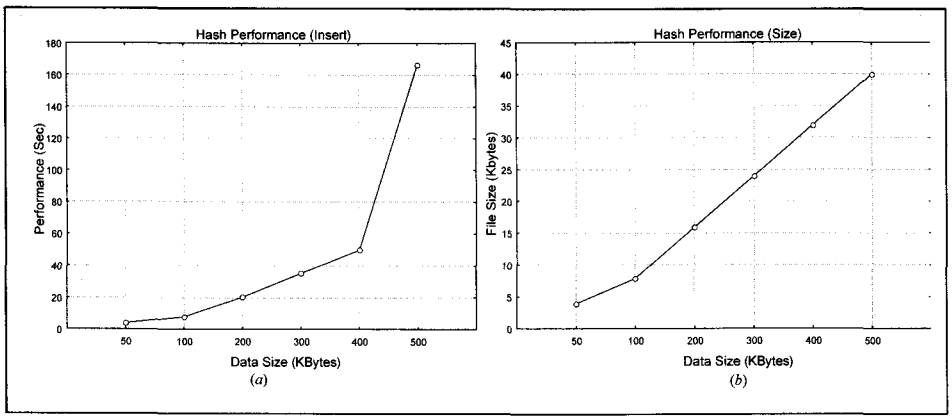
실험에서는 노드(Node) 크기를 1K~4K 바이트로 변경하여 실험을 하였으며, citySimulator 에서 생성한 유사한 형태의 3가지 데이터 셋들로부터 얻어진 값들의 평균 값을 제시하였다.

객체 식별자에 대한 색인은 *B-tree*를 이용하여 수행하며 *B-tree*의 성능은 〈그림 4〉와 같고, 대부분의 삽입 및 갱신 시간은 위치 정보 색인에 비하여 무시할 수 있고, 검색은 수 밀리초 단위에 수행되므로 실험 결과는 배제하였다.

색인에 대한 백업은 〈그림 5〉에서와 같이 *B-tree*에서 삽입과 갱신이 일어날 때 동일하게 수행되며,



〈그림 4〉 B-tree 성능 평가를 위한 삽입 및 갱신 실험 결과

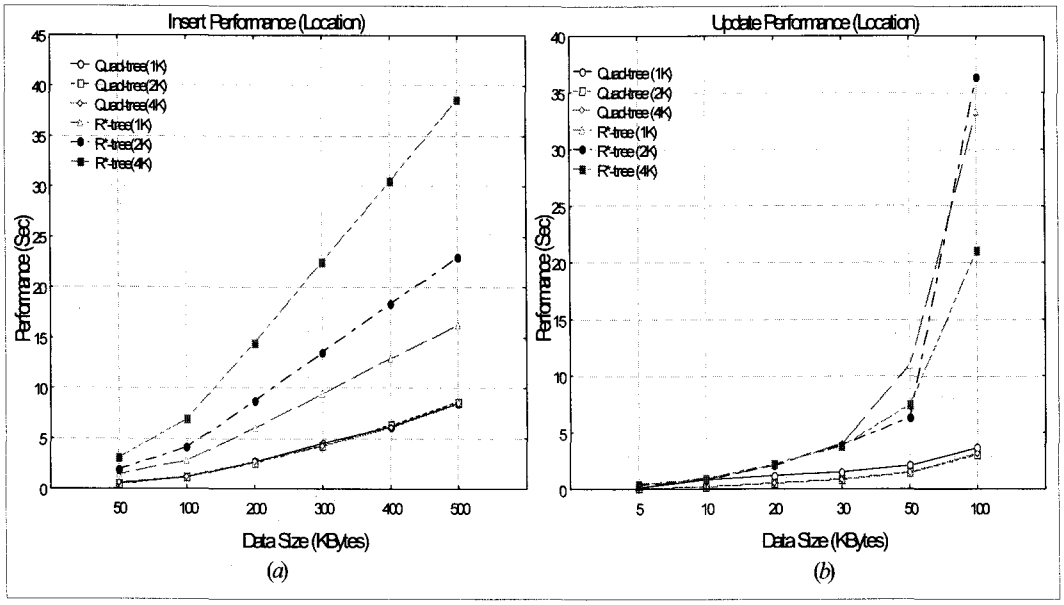


〈그림 5〉 Hash 성능 평가를 위한 삽입 및 파일 크기 실험 결과

Hashing 함수는 *.Net* 라이브러리 함수를 사용하였다. 데이터를 이용한 실험 결과 *Hashing*에 의한 데이터 백업은 40만개 이상의 객체를 저장하는 경우 급격한 시간 증가가 일어나므로 색인의 백업을 40만개 이내에서 제한하거나, 분할하여 각각 다른 파일로 저장하는 것이 바람직한 것으로 나타났다.

이동 객체의 위치를 색인하는 *A-Quadtree*는 성능 평가를 위하여 *R*-tree*와 동일한 조건에서 삽입과 갱신을 실험하였고, 결과를 〈그림 6〉에 제시하였다.

실험 결과 *A-Quadtree*의 높이는 500,000개의 데이터에 대하여 8~10 사이의 값을 가진다. 대부분의 데이터 셋에서 색인의 높이는 100,000개 객체에 대하여 높이가 7이며 그 이상의 데이터에 대해서는 급격한 높이의 증가를 가져오지 않고, 데이터 셋의 분포와 동일한 위치에 중복되는 객체의 수에 따라 공간 분할의 형태가 변화된다. 그러므로 일반적인 도시 환경과 같은



〈그림 6〉 위치 정보 색인 성능 평가를 위한 삽입 및 갱신 실험 결과

객체 분포 환경에서는 급격한 높이의 증가가 없으며, 노드 크기가 작은 것이 삽입과 갱신 면에서 유리하다.

또한 *A-Quadtree*의 검색 성능은 *B-tree*와 거의 유사한 시간에 수행되며 전체 공간의 5% 범위 이내의 질의 처리는 300밀리 초 이내에 수행되므로 실험 결과의 비교는 생략하였고, 대체적으로 검색 성능은 패러미터의 변화에 거의 영향을 받지 않고 수행되었다.

6.3 검색 및 공간 효율성

*A-Quadtree*에서 객체가 저장되는 단말 노드는 노드 크기와 데이터 분포에 따라 효율성이 변화하며 실험을 통하여 대부분의 데이터 셋에서 0.4~0.5% 정도의 효율을 보인다. 〈그림 7 (a)〉에서 저장 공간의 효율성은 대체로 *R*-tree*와 유사하지만, 노드 크기가 충분히 크거나 작은 경우에 보다 효율적이다.

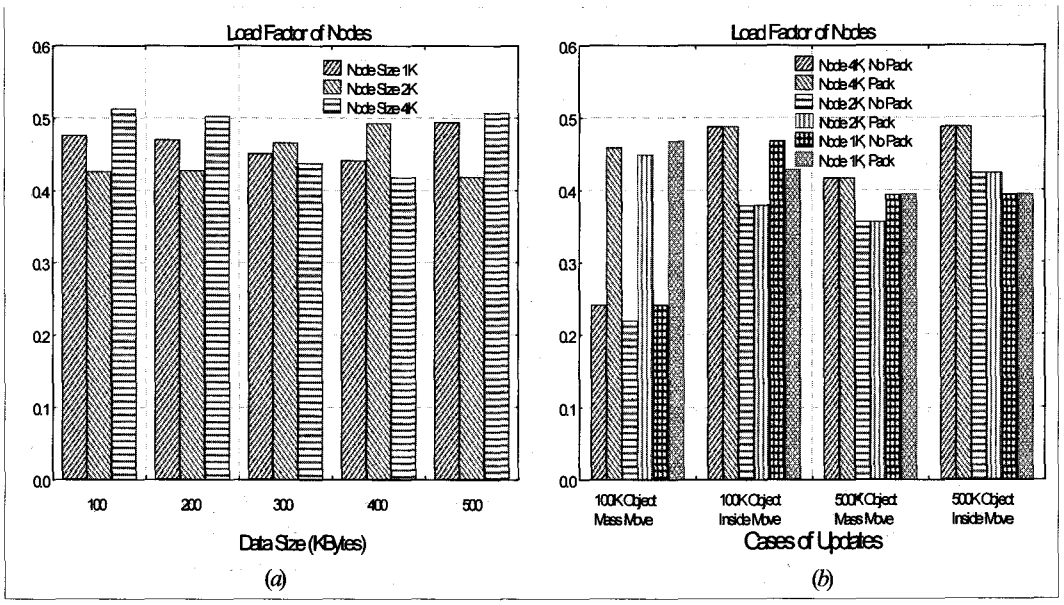
또한 갱신에 의한 저장 공간 효율성의 감소는 〈그림 7(b)〉에서와 같이 100,000개 이하의 객체와 500,000개의 객체 색인에서 대부분의 객체 위치가 크게 변화하는 경우(Mass Move)와 동일한 공간에서 작은 위치 변화가 있는 경우에 대한 평가는 적응적 압축이 필요한 이유를 제시한다. 적응 압축을 통하여 위치 변화가 큰 경우, 즉 공식 (3)에서 N이 작고 위치 변화가 큰 경우와 N이 크고 위치 변화가 큰 경우 노드 크기에 따라 압축 시간을 설정할 필요가 있다. 기타의 경우 갱신이 발생할 때 색인의 조정이 없어도 노드 효율성은 크게 감소하지 않으므로 논문에서 제시한 적응형 관리 방법을 사용하는 것이 보다 효율적임을 알 수 있다.

7. 결 론

일반적으로 데이터베이스에서는 이동 객체의 위치 정보와 같은 지속적으로 변화하는 데이터를 다룰 수 있는 기능을 제공하지 않고 있으며, 위치 정보를 효율적으로 색인하기 위해서는 실시간에 갱신을 색인에 반영할 수 있도록 해야 한다. 특히 위치 정확도가 요구되는 경우 현재 색인은 기존의 *R-tree* 기반의 색인 방법보다 처리 시간을 줄일 수 있어야 하며, 본 논문에서는 이러한 목적의 위치 정보 색인 구조로서 *A-Quadtree*를 제시하였다.

본 논문에서 제시하는 색인 구조는 주 기억장치 상주형 B-tree와 통합하여 위치 기반 서비스에서 요구되는 현재 위치 질의 처리 및 색인 관리를 효율적으로 다룰 수 있는 컴포넌트 소프트웨어로서 구현하였으며, 실험 결과 점 질의와 영역 질의 모두에서 R-tree 기반의 색인보다 효율적임을 보였다.

또한 이동 객체의 위치 정보의 특성을 반영할 수 있는 적응 압축 방법을 사용함으로서 색인 관리에 따른



〈그림 7〉 A-Quadtree노드 효율성 및 적응 압축 성능

시간 소모 감소시키고, 실시간 처리를 가능토록 하였다.

향후 연구 과제는 제시한 색인 구조를 기반으로 최기(Nearest Neighbor) 질의 및 집계(Aggregation) 질의 등과 같은 다양한 질의에 대한 연구가 필요하며, 도시 환경이 아닌 데이터 셋에 대한 실험을 통하여 위치 기반 서비스에서 요구되는 기능을 실제로 지원할 수 있는 상용 시스템으로 개선이 요구된다.

참고문헌

[1] Wolfson, O., Sistla, P., Chamberlain, S., Yesha, Y., Updating and Querying Databases that track Mobile Units, J. of Distributed and Parallel Databases, Vol. 7, 1999, pp257-287.

[2] Park, H, Son, J., Kim, M., An Efficient Spatiotemporal Indexing Method for Moving Objects in Mobile Communication Environments, Proceedings of MDM, LNCS 2574, 2003, pp78-91.

[3] Forlizzi, L., Guting, R., Nardelli, E., Schneider, M., A Data Model and Data Structures for Moving Objects Databases, Proceedings of SIGMOD, 2000, pp. 319-330.

[4] Song, Z., Roussopoulos, N., Hashing Moving Objects, Proceedings of MDM, LNCS 1987, 2001, pp. 161-172.

[5] Chon, H., Agrawal, D., Abbadi, A., Query Processing for Moving Objects with Space-Time Grid Storage Model, Proceedings of MDM, 2002, pp. 121-128.

[6] 조대수, 남광우, 이종훈, M-커머스를 위한 위치 기반서비스 응용 기술, 한국정보과학회지, 2002. pp. 45-51.

[7] Saltenis, S., Jensen, C., Indexing of Moving Objects for Location-Based Services, Proceedings of ICDE, 2002, pp. 463-472.

[8] Pfoser, D., Jensen, C., Theodoridis, Y., Novel Approaches in Query Processing for Moving Objects, Proceedings of VLDB, 2000, pp. 395-406.

[9] Kwon, D., Lee, S., Lee, S., Indexing the Current Positions of Moving Objects Using the Lazy Update R-tree, Proceedings of MDM, 2002.

[10] Muthukrishnan, S., Poosala, V., Suel, T., On Rectangular Partitioning in Two Dimensions: Algorithms, Complexity and Applications, Proceedings of ICDT, LNCS 1540, 1998, pp. 236-256.

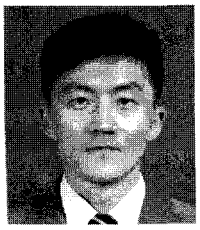
[11] Tayeb, J., Ulusoy, O., Wolfson, O., A Quadtree-based Dynamic Attribute Indexing Method, The Computer Journal 41(3), 1998, pp. 185-200.

[12] Gaede, V., Gunter, O., Multidimensional Access Method, ACM Computing Surveys 30(2), 1998, pp. 170-231.

[13] Aboulnaga, A., Aref, W., Window Query Processing in Linear Quadtrees, Distributed and Parallel Databases, Vol. 10, 2001, pp. 111-126.

[14] Theodoridis, Y., Sellis, T., Papadopoulos, A., Manolopoulos, Y., Specifications for Efficient Indexing in Spatiotemporal Databases, Proceedings of SSDBM, 1998, pp. 123-132.

[15] citySimulator, http://alphaworks.ibm.com/tech/citysimulator



박현규

1987년 육군사관학교 전산학과 학사

1992년 Naval Postgraduate School 전산학과 석사

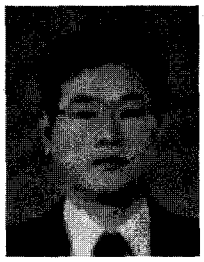
2003년 한국과학기술원 전자전산학과 박사

1987년 ~ 현재 육군 전산장교

관심분야 : 시공간 데이터베이스, 위치 기반 시스템, C4I시스템, Pervasive Computing

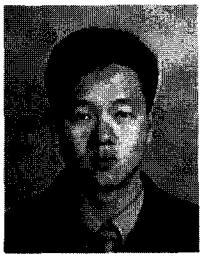
강성탁

1996년 한국과학기술원 전산학과 학사
1998년 한국과학기술원 전산학과 석사
2003년 한국과학기술원 전자전산학과 박사과정 수료
2003년 3월 ~ 현재 LG전자 GSM연구소 연구원
관심분야 : Temporal 데이터베이스, 시공간 데이터베이스, 데이터 웨어하우스, OLAP



김명호

1982년 서울대학교 컴퓨터공학과 학사
1984년 서울대학교 컴퓨터공학과 석사
1989년 MICHIGAN 주립대 전산학과 박사
1989년 MICHIGAN 주립대 연구원
1989년 ~ 1993년 한국과학기술원 조교수
1993년 ~ 1999년 한국과학기술원 부교수
1999년 ~ 현재 한국과학기술원 교수
1992년 ~ 1993년 개방형 컴퓨터 통신 연구회 (OSIA) 분산 트랜잭션처리 분과위(TG-TP) 의장
1996년 ~ 1998년 한국정보과학회 DB연구회지 책임편집위원
관심분야 : 데이터베이스, 분산트랜잭션, 분산시스템, 워크플로우



민경욱

1996년 부산대학교 전자계산학과 졸업(학사)
1998년 부산대학교 전자계산학과 졸업(석사)
2001년 ~현재 한국전자통신연구원 LBS 연구팀 연구원
관심분야 : 공간데이터베이스, 이동객체데이터베이스, LBS, GIS