계층적 행정 구역에 기반한 효율적인 위치 정보 표현 방식

(An Efficient Location Encoding Method Based on Hierarchical Administrative District)

이 상 윤 † 박 상 현 †† 김 우 철 † 이 동 원 †††
(SangYoon Lee) (SangHyun Park) (WooCheol Kim) (DongWon Lee)

요 약 최근 이동 통신 기술의 급속한 발달로 인해 휴대폰, PDA등과 같은 휴대용 단말기의 사용이 보편화 되고 있다. 따라서 무선 이동기기의 시간에 따른 공간적인 위치 정보를 활용하여 다양하고 빠른 서비스를 제공하기 위해서 위치 기반 서비스(Location-Based Service)에 관한 많은 연구가 진행되고 있다. 효율적인 위치 기반 서비스의 제공을 위하여 시간에 따라 지속적으로 변하는 이동 객체의 대용량 시공간 정보를 신속하게 저장, 관리, 검색할 수 있는 인덱싱 및 질의 처리 기술이 수반되어야 한다. 본 논문에서는 대용량 이동 객체 데이타베이스를 대상으로 효율적인 인덱스 구축을 위한 위치 정보의 압축 표현 방식에 대하여 논한다. 이를 위해 본 논문에서는 기존의 주요 연구에서 (x, y) 형태의 2차원 공간 좌표로 표현되던 이동 객체의 위치 정보를 계층적 구조를 갖는 행정 구역과 도로 상의 위치를 이용하여 1차원의 위치 정보로 압축 표현하는 방식을 제안한다. 이를 이용해 도로를 따라 움직이는 이동 객체에 대해 위치 정보의 손실 없이 효율적인 위치 기반 서비스를 제공할 수 있다. 또, 일정 공간 내의 객체 분포를 필요로 하는 교통 상황 파악, 근사적(approximate) 공간 정보를 필요로 하는 사람·차량 위치 추적 등에 유용하게 사용할 수 있다.

키워드 : 위치 기반 서비스, 도로 네트워크, 이동 객체, 인덱싱

Abstract Due to the rapid development in mobile communication technologies, the usage of mobile devices such as cell phone or PDA becomes increasingly popular. As different devices require different applications, various new services are being developed to satisfy the needs. One of the popular services under heavy demand is the Location-based Service (LBS) that exploits the spatial information of moving objects per temporal changes. In order to support LBS efficiently, it is necessary to be able to index and query well a large amount of spatio-temporal information of moving objects. Therefore, in this paper, we investigate how such location information of moving objects can be efficiently stored and indexed. In particular, we propose a novel location encoding method based on hierarchical administrative district information. Our proposal is different from conventional approaches where moving objects are often expressed as geometric points in two dimensional space, (x, y). Instead, in ours, moving objects are encoded as one dimensional points by both administrative district as well as road information. Our method is especially useful for monitoring traffic situation or tracing location of moving objects through approximate spatial queries.

Key words : Location-Based Service, road network, moving object, indexing

† 학생회원 : 연세대학교 컴퓨터과학과
sylee@cs.yonsei.ac.kr
twelvepp@cs.yonsei.ac.kr
†† 종신회원 : 연세대학교 컴퓨터과학과 교수
sanghyun@cs.yonsei.ac.kr
††† 정 회 원 : Penn State University Information Sciences and Technology 교수
dongwon@psu.edu
논문접수 : 2005년 3월 4일
심사완료 : 2006년 3월 3일

1. 서 론

최근 이동 통신 기술의 발달로 무선 이동 기기의 사용이 보편화 되면서 무선 이동 기기의 위치 정보를 이용하는 위치 기반 서비스(LBS: Location-Based Service)가 활성화 되고 있다. 예를 들면 친구의 위치를 찾는 “친구 찾기”나 사용자 근처의 맛있는 음식점을 찾는 “음식점 찾기”, 인터넷 쇼핑몰을 통해서 구입한 물건의

현재 위치를 확인하는 “물류 운송정보 조회” 등 다양한 서비스가 제공되고 있다.

위치 기반 서비스는 일정한 시간 간격마다 각 이동 객체의 위치 정보를 파악하여 데이타베이스에 저장하고 이를 이용해 사용자가 원하는 질의를 처리해 결과를 알려주는 서비스이다. 위치 기반 서비스에서 사용하는 질의는 일반적으로 영역 질의, 궤적 질의, 복합 질의로 구분할 수 있다[1]. 영역 질의는 주어진 시간에 특정 영역 안에 있었던 모든 이동 객체를 검색하는 것이며, 궤적 질의는 특정한 시간 간격 동안에 이동 객체가 움직였던 경로를 검색하는 것이다. 복합 질의는 영역 질의와 궤적 질의를 결합한 것으로 특정한 시간 간격 동안에 특정 영역에 있었던 이동 객체의 궤적을 검색하는 것이다.

이러한 위치 기반 서비스에서의 이동 객체는 다음과 같은 특징을 갖는다. 첫째, 시간에 따라 지속적으로 공간적인 위치 정보를 갱신하기 때문에 높은 갱신 비용이 든다. 둘째, 이동 객체는 객체 정보, 시간 정보, 위치 정보 등으로 이루어진 다차원의 데이타로 표현되므로 이동 객체의 저장 시 공간적인 오버헤드가 발생한다. 셋째, 현재 시점뿐 아니라 과거 시점에서의 위치 정보도 저장해야 하므로 시간이 증가함에 따라 데이타의 양이 급속히 증가하여 대용량의 정보가 된다. 넷째, 검색 시 대용량의 정보를 대상으로 하기 때문에 높은 검색 비용을 필요로 한다. 따라서 이러한 이동 객체의 대용량 시공간 정보를 신속하게 저장, 관리, 검색할 수 있는 인덱싱 및 질의 처리 기술이 수반되어야 한다.

본 논문에서는 대용량 이동 객체 데이타베이스를 대상으로 효율적인 인덱스 구축 및 질의 처리를 위한 데이타 압축 표현 방식에 대하여 논한다. 즉, 기존의 연구에서는 2차원의 공간 좌표 (x, y)로 표현되었던 이동 객체의 위치 정보를, 실세계 모델에 부합하는 계층적 행정 구역과 도로 상의 위치를 이용하여 1차원의 공간 좌표로 표현하는 방식을 제안한다. 예를 들어 이동 객체가 ‘서울 시청’ 건물에 있다면, 이를 (동경 126도 58분, 북위 37도 34분)과 같이 2차원 공간 좌표로 표현하는 것이 아니라, 해당 주소를 (행정 구역 이름, 도로 이름, 도로 상의 위치)의 세 개의 필드로 분할하여 (서울시 중구, 태평로1가, 31)로 표현한 후 각 필드를 이진스트링으로 변환하여 순서대로 연결함으로써 하나의 이진스트링으로 표현한다.

제안된 위치 정보 표현 방식은 다음과 같은 세 가지 장점을 가지고 있다. 첫째, 2차원으로 표현되었던 위치 정보를 1차원으로 표현함으로써 객체의 저장 공간을 줄일 수 있으며 인덱스의 차원도 낮출 수 있다. 따라서 질의 처리의 성능 향상을 기대할 수 있다. 둘째, 실세계에서 사람이나 차 같은 이동 객체는 차로나 인도 같은 도로를 따라 이동한다. 하지만, 위치 정보를 (x, y) 같은 공간 좌표로 나타내면 이동 객체가 갈 수 없는 곳까지 표현하게 되므로 위치 정보 표현 방식에 사장 공간(dead space)이 생기게 된다. 그러나 제안된 방식에서는 이동 객체가 실제로 이동할 수 있는 영역만을 위치 정보로 표현하므로 위치 정보 표현 방식의 사장 공간을 없앨 수 있다. 셋째, 변환된 1차원의 위치 정보가 행정 구역의 정보를 내포하고 있기 때문에 질의 결과를 주소 형태로 쉽게 변환할 수 있다. 주소 형태로 변환된 질의 결과는 텍스트 기반의 인터페이스 환경이나 음성 인터페이스 환경에서도 쉽게 표현될 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 좌표 형식의 위치 표현 방식과 도로 네트워크에 대한 관련 연구를 기술하고, 3장에서는 새로운 위치 정보 표현 방법에 대해 설명한다. 4장에서는 새로운 위치 정보 표현 방법을 사용하는 위치 기반 서비스 시스템의 구조를 기술하고, 5장에서는 새로운 위치 정보 표현 방법에 기반한 질의 처리 과정을 설명한다. 6장에서는 실제 행정 구역 및 도로 데이타를 대상으로 제안된 시스템의 효율성을 입증하고, 7장에서는 논문의 결론을 제시한다.

2. 관련 연구

3DR-tree [2], HR-tree [3], STR-tree [4], TB-tree [1], MV3R-tree [5]와 같은 기존의 인덱스들은 이동 객체의 위치 정보를 (x, y)의 2차원 공간 좌표로 표현하고, 이를 3차원 이상의 R-tree [6] 구조를 사용하여 인덱싱한다. 3DR-tree, HR-tree는 영역 질의에 대해, STR-tree, TB-tree, MV3R-tree는 궤적 질의에 대해 효과적인 인덱스이다. 그러나 이러한 인덱스들은 시간이 증가함에 따라 크기가 급속히 커지게 되어 높은 검색 비용을 필요로 하며, 이동 객체가 실제로 도달할 수 없는 위치까지도 (x, y)의 2차원 공간 좌표로 표현하므로 사장 공간이 생겨서 영역 표현의 낭비가 생기게 된다. 따라서 최근 이러한 문제점을 개선하기 위해 실세계의 도로 네트워크(road network)를 이용하여 위치 정보를 표현하려는 연구와 객체의 차원을 분리하여 저장함으로써 질의 검색에 사용되는 인덱스의 차원을 줄이려는 연구가 진행되고 있다.

도로 네트워크를 이용하는 연구 [7,8]은 객체의 이동 가능한 범위를 도로 위로 제한하여 이동 객체의 위치를 (가장 가까운 도로의 식별자, 가장 가까운 도로로부터의 거리)의 쌍으로 표현한다. [7]에서는 이러한 도로 좌표에 적합한 인덱스를 제안하고, 공간 좌표상의 거리를 측정하는 유클리디언 거리(Euclidean distance) 대신 도로를 따라 이동하는 거리를 측정하는 네트워크 거리(network distance)를 이용하여 이동 객체에 대한 질의

를 처리하는 알고리즘을 제시하였다. 예를 들어, “현재 위치에서 가장 가까운 호텔을 찾아라.”라는 질의에 대해 공간 좌표 상에서 가장 가까운 위치에 있는 호텔을 검색 결과로 제시하는 대신에 도로 네트워크를 따라 이동 거리가 가장 짧은 호텔을 검색 결과로 제시한다. 이 방법은 도로 좌표를 사용함으로써 영역 표현의 낭비를 없앨 수 있다는 장점이 있지만, 도로 및 이동 객체의 위치를 여전히 2차원으로 표현한다는 단점이 있다.

[8]에서는 도로 네트워크의 각 노드를 일정한 규칙을 적용하여 이진 스트링으로 변환한 후, 이진 스트링 간의 간단한 연산(예를 들면, 해밍 거리(Hamming distance)의 계산)을 통해 최단 경로 검색 등의 다양한 질의를 처리하는 방법을 제안하였다. 그러나 이 방법은 노드의 수에 비례하여 이진 스트링의 길이가 커진다는 것과, 노드의 이진 스트링만으로는 도로의 상대적 위치를 알 수 없다는 단점을 가지고 있다.

객체의 차원을 분리하여 저장함으로써 인덱스의 차원을 줄이려는 대표적인 연구로 [9-11] 등이 있다. 이들은 사용자의 질의를 효과적으로 처리하기 위하여 2차원의 위치 정보와 1차원의 시간 정보를 각각 다른 인덱스에 저장한다. [9]에서는 공간 영역을 셀 단위로 분할하여 영역 간의 겹침을 제거한 후 각 셀에 포함된 객체들의 시간 정보를 별도의 인덱스에 저장하며, [10]에서는 도로의 2차원 위치 정보로부터 R-tree를 구축한 후 트리의 단말 노드가 1차원 시간 정보를 저장하는 R-tree를 가리키도록 한다. 또한 [11]에서는 도로의 위치 정보를 저장하는 2차원 R-tree와 이동 객체의 도로 내 상대적 위치 및 시간 정보를 저장하는 2차원 R-tree를 별도로 구축하여 관리한다. 이와 같은 연구들은 이동 객체의 위치가 시간에 따라 크게 변하지 않음을 이용하여 위치 정보를 중복 없이 효율적으로 저장한다. 그러나 사용자의 질의를 처리하기 위해서는 여전히 3차원 데이타를 모두 사용해야 한다는 한계가 있다.

[12]에서 제안한 방법은 공간 필링 커브(space filing curve) 기법의 하나인 힐버트 커브(Hilbert curve)를 사용하여 2차원의 위치 정보를 1차원으로 변환하여 저장하기 때문에 본 논문에서 제안한 방법과 가장 유사하다고 할 수 있다. 그러나 위치 정보 변환 시 계층적 행정 구역을 고려하지 않기 때문에 실세계 모델에 적합하다고 할 수 없으며 검색 결과를 행정 구역 단위로 압축하여 표현하는 것이 불가능하다. 그러나 본 논문에서 제안한 위치 표현 방식에서는 이진스트링으로 표현된 위치 정보가 행정 구역에 대한 정보를 내포하고 있기 때문에 이동 객체의 위치를 주소 형태로 쉽게 변환할 수 있으며, 질의 결과를 계층적 행정 구역 단위별로 압축하여 사용자에게 제시하는 것이 가능하다.

3. 제안하는 위치 정보 표현 방법

현재의 주소 체계는 도로를 기반으로 위치를 표현한다. 예를 들어 “서울시 중구 태평로1가 31”인 “서울시청”의 주소는 “태평로1가”라는 도로 이름을 기준으로 도로가 속해있는 행정 구역의 이름인 “서울시 중구”와 도로 상의 위치를 표현하는 “31”로 나눠진다. 이를 일반화하면 주소는 (행정 구역 이름, 도로 이름, 도로 상의 위치)로 나누어 표시할 수 있다. 이러한 주소 체계를 이용하면 이동 객체의 위치를 1차원의 이진스트링으로 쉽게 변환할 수 있다. 즉, 이동 객체가 있는 곳의 주소를 (행정 구역 이름, 도로 이름, 도로 상의 위치)의 세 개의 필드로 분할하여 표현한 후 각 필드를 이진스트링으로 변환하여 순서대로 연결함으로써 이동 객체의 위치를 하나의 이진스트링으로 표현할 수 있다.

이동 객체의 위치를 하나의 이진스트링으로 변환하는 과정을 쉽게 설명하기 위해, 전체적으로 4개의 “시” (A, B, C, D)가 있고 각 “시” 안에 8개의 “구” (a, b, ..., g)가 있어 총 32개의 행정 구역이 존재한다고 가정하자. 이 때, 각 행정 구역을 이진스트링으로 변환하기 위한 간단한 방법으로 행정 구역 이름의 사전식 순서를 이용할 수 있다. 즉, 4개의 “시” 단위 행정 구역을 표현하기 위한 2비트, 각 ″시″안의 8개 “구” 단위 행정 구역을 표현하기 위해 3비트를 사용하면 각각 사전식 정렬 순서로 변환할 수 있다. “A시 a구”는 “00 000″로, ″A시 b구”는 “00 001”로, “B시 a구”는 ″01 000″로 표현할 수 있다.

위와 같은 변환 방법은 간단하지만 이진스트링에서 얻을 수 있는 정보에 한계가 있다는 단점이 있다. 예를 들어 어떤 두 객체가 “00 000”과 “00 001”로 표현되는 두 행정 구역에 각각 위치한다고 가정하자. 두 이진스트링을 비교해보면 “시”를 나타내는 2비트가 같으므로 두 객체가 같은 “시”에 있음을 알 수 있다. 또한 “구”를 나타내는 3비트가 다르므로 두 객체가 서로 다른 “구”에 있음을 알 수 있다. 그러나 이 정보만으로는 두 객체가 속한 행정 구역의 상대적 위치를 판단할 수 없다. 따라서 본 논문에서는 이진스트링이 보다 많은 정보를 표현할 수 있도록 하기 위해서 다차원 공간상의 인접 객체들을 1차원 선형 공간상의 인접 위치로 매핑하는 공간 필링 커브(space filling curve) 기법을 사용한다. 공간 필링 커브 기법은 매핑하는 방법에 따라 Z-ordering [13], R-ordering[14], H-ordering[15] 등으로 구분된다.

3.1 행정 구역의 이진스트링 변환

본 논문에서는 행정 구역간의 상대적 위치 관계를 효과적으로 표현하기 위하여, 행정 구역을 북/남, 서/동 방향으로 분할하면서 북쪽과 서쪽에 ‘0’, 남쪽과 동쪽에 ‘1’

의 이진 비트를 부여한 후, 분할된 영역들을 Z-ordering 방법을 이용하여 좌상단부터 하나씩 순서대로 읽으면서 1차원 이진스트링으로 변환한다. Z-ordering 방법을 사용하여 행정 구역을 이진스트링을 변환하는 알고리즘과 예는 알고리즘 1과 그림 1에 각각 기술된다.

그림 1에서 A시의 8개 구는 그림 2(b)에서 ①을 따라 남/북으로 분할된다. (abcd/efgh) 분할된 영역을 다시 ②를 따라 분할하게 되면 각 영역은 동/서로 분할된다.((ab/cd)/(ef/gh)) 마지막으로 ③을 따라 이동하게 되면 각 영역에 하나의 "구"가 대응된다. 각 영역에 하나의 "구"가 대응되면 z-ordering을 이용하여 각 영역에 이진스트링을 부여함으로서 객 행정 구역을 이진스트링으로 표현할 수 있게 되고, 행정 구역의 이진스트링만으로도 각 행정 구역이 상대적으로 "시" 안에서 어느 위치에 있는지 쉽게 알 수 있다.

이와 같이 Z-ordering 방법을 사용하면 변환된 이진스트링에 더 많은 정보를 표현할 수 있다. 위에서 사용한 예를 다시 살펴보자. 어떤 두 객체가 "00 000"과 "00 001"로 표현되는 두 행정 구역에 각각 위치한다면 다음과 같은 정보를 얻을 수 있다. 처음 2비트가 같으므로 두 객체가 같은 "시"에 있음을 알 수 있을 뿐만 아니라, 다음 2비트인 "00"를 통해서 두 객체 모두 시의 북서쪽에 있는 "구"에 있음을 알 수 있다.

3.2 도로의 이진스트링 변환

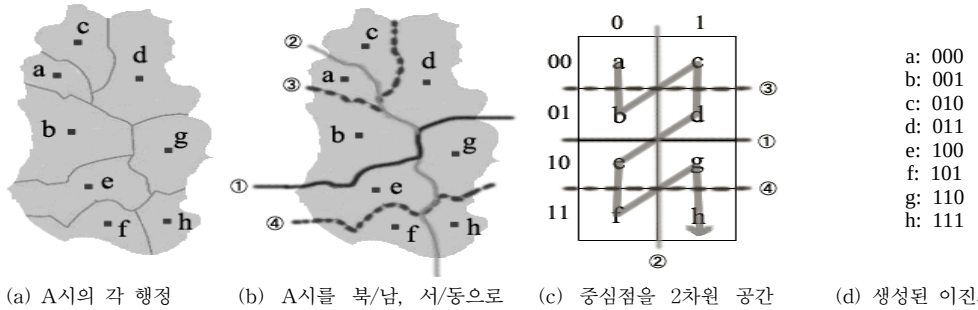
알고리즘 1의 1)에서 "행정 구역의 중심점"을 "도로의 중심점"으로 변경하면 동일한 방식으로 도로를 이진스트링으로 변환할 수 있다. 그러나 도로의 경우 행정 구역처럼 영역으로 표현되는 것이 아니기 때문에 하나의 연속된 도로의 중심점을 구하여 이동 객체의 위치를 나타낼 경우 큰 오차가 발생할 수가 있다. 곡선 형태의 도로의 경우, 도로의 중심점은 도로 위의 한 점으로 표현되지 않을 수도 있다. 따라서 오차를 줄이기 위해 다음과 같은 방법으로 도로를 분할하여야 한다.

1) 교차로를 중심으로 도로를 분할한다. 예를 들면 사거리의 경우 교차점을 중심으로 4개의 도로로 분할된다.
2) 복잡한 곡선으로 표현되는 도로의 경우 도로의 시작점으로부터 최대한 직선이 되게 도로를 분할한다. 이때, 직선의 시작점은 도로의 시작점이거나 이전 도로의 끝점이 된다. 직선의 끝점은 도로의 끝점이거나 곡선의 정점이 된다(그림 2).

그림 2(a)의 경우 교차로를 중심으로 도로를 분할하였고, (b)의 경우 ①,③은 곡선에 대한 분할 지점이고 ②는 교차로에 의한 도로 분할 지점이다. 또, '강변 북로' 같은 여러 행정 구역에 걸쳐서 나타나는 도로에 대해서

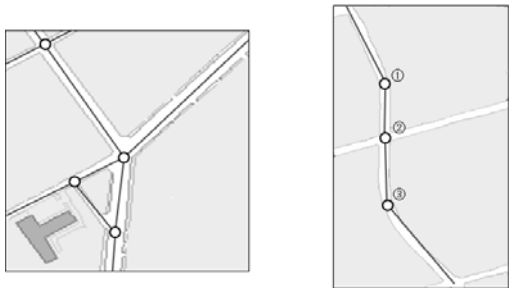
알고리즘 1: 행정 구역들을 분할해 이진스트링으로 매핑하는 알고리즘

1) 각 행정 구역의 중심점들을 찾는다.
2) 각 영역에 중심점의 개수가 1개가 될 때까지 다음 과정을 반복하여 전체 영역을 분할한다.
2-1) 행정 구역들을 북/남 두 개의 영역으로 1)에서 찾은 중심점들의 개수가 비슷하게 분할한다. (분할된 영역 중 북쪽은 '0', 남쪽은 '1'의 이진 비트 부여)
2-2) 남/북으로 나누어진 영역에 대해 다시 서/동 두 개의 영역으로 중심점들의 개수가 비슷하게 분할한다. (분할된 영역 중 서쪽은 '0', 동쪽은 '1'의 이진 비트 부여)
3) 각 행정 구역의 중심점을 전체 영역의 분할 형태를 고려하여 2차원 공간으로 매핑한다.
4) Z-ordering 알고리즘을 통해서 각 행정 구역을 이진스트링으로 변환한다.



(a) A시의 각 행정 구역의 중심점 (b) A시를 북/남, 서/동으로 분할 (c) 중심점을 2차원 공간으로 매핑 (d) 생성된 이진스트링

그림 1 A시의 행정 구역들을 이진스트링으로 표현하는 예



(a) 교차로에 의한 도로 분할 (b) 곡선 도로에 대한 분할

그림 2 도로에 대한 분할 방법

는 우선 행정 구역 단위로 분할한 뒤 위의 방식을 사용하여 도로를 나타낼 수 있다.

도로에 대한 이진스트링 변환 과정은 위의 방식을 사용하여 도로를 분할하고, 분할된 도로에 대해서 행정 구역의 변환처럼 도로의 중심점을 찾아 알고리즘 1을 수행한다. 각 “동”의 도로들에 대해 이진스트링으로 변환이 완료되면, 행정 구역을 나타내는 이진스트링과 도로의 이진스트링을 합쳐 하나의 이진스트링으로 변환한다. 이를 도로의 이진스트링으로 사용한다. 최종 이진스트링은 도로가 어느 행정 구역에 속해 있는지, 행정 구역 내에서의 상대적 위치 등을 쉽게 알 수 있다. 예를 들면, “서대문구 신촌동 서문 2길” 이라는 도로는 ‘서대문구’를 나타내는 “00001”, “신촌동”을 나타내는 “00110”, 신촌동 내에서의 “서문 2길”의 상대적 위치를 나타내는 “10010”을 하나의 이진스트링으로 표현하여 “00001 00110 10010”으로 나타낼 수 있다. 여기서 상위 5비트는 행정 구역인 “서대문구”를 나타내고, 상위10비트는 “서대문구 신촌동(00001 00110)”을 나타낸다. 따라서 도로의 이진스트링은 각 행정 구역과 도로 간의 계층 정보를 가지고 있기 때문에 도로 이진스트링의 접두어를 이용하면 해당 도로가 어느 행정 구역에 속하는 지 쉽게 알 수 있다.

3.3 이동 객체의 위치에 대한 이진스트링 변환

객체의 위치 정보를 이진스트링으로 변환하기 위해서는 행정 구역과 도로 이름을 이진스트링으로 변환한 후 도로 상에서의 객체 위치를 이진스트링으로 표현해야 한다. 이를 위해서 먼저 도로를 2^n-1개의 단위로 분할한 후 각 경계 지점을 n 비트의 이진스트링으로 표현한다. 다음으로 객체의 위치와 가장 가까운 경계 지점의 이진스트링을 이용하여 도로 상의 위치를 표현한다. 이동 객체의 위치는 도로 내의 한 구간으로 매핑 되어지고 매핑된 구간의 이진스트링을 구해 도로를 나타내는 이진스트링과 합쳐서 하나의 이진스트링으로 위치 정보를 나타낸다. 이렇게 표현된 이진스트링은 이동 객체가 어느 행정 구역의 도로에 있는지, 도로 내에서 어느 정도 거리에 위치하고 있는지 쉽게 알 수가 있다.

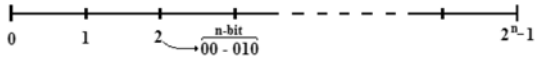


그림 3 도로에서의 상대위치 표현

본 논문에서 제안한 위치 정보 표현 방식은 다음과 같은 특징을 갖는다. 1) 주어진 이진스트링의 집합으로부터 최대 길이의 공통 접두어를 추출하면 해당하는 객체들이 공통으로 속해있는 최소 행정 구역을 파악할 수 있다. 2) 상위 행정 구역은 이진스트링의 범위로 표현할 수 있다. 즉, 위의 예에서 “A시”는 “00000~00111”으로 표현할 수 있다.

4. 시스템 구조

4.1 시스템 흐름도

위치 기반 서비스를 위한 시스템은 1) 이동 객체의 데이타를 수집해서 데이타베이스에 저장하는 부분과 2) 사용자의 질의를 처리하는 부분으로 구분된다. 위치 정보로 이진스트링을 사용하기 위해서는 그림 4와 같이 3가지 위치 정보 변환 모듈 *XY2BS, AD2BS, BS2AD*

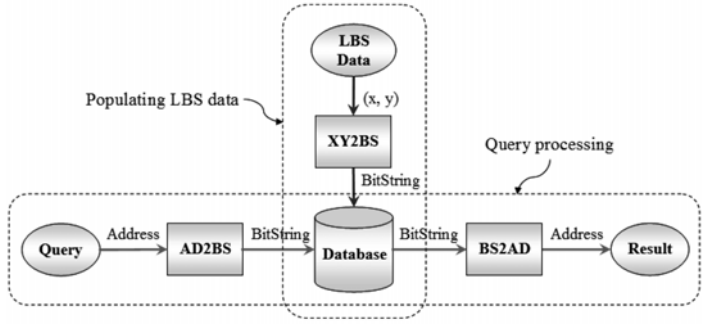


그림 4 LBS 시스템에서의 위치 정보 흐름도

가 필요하다.

4.2 XY2BS

(x, y) 형태로 수집된 이동 객체의 위치 정보를 본 논문에서 제안하고 있는 이진스트링 위치 정보로 변환하는 모듈을 *XY2BS*라 한다. 위치 정보의 신속한 변환을 위해서 *XY2BS*는 R-tree기반의 인덱스를 사용하다. 각 도로를 도로의 시작점과 끝점을 이용해 직선 형태로 R-tree에 저장하고 리프 노드에는 3장에서 설명한 알고리즘을 통해 생성된 도로의 이진스트링을 저장한다. 이렇게 구성된 R-tree와 다음과 같은 알고리즘 2를 통해서 이동 객체의 (x, y) 위치 정보를 이진스트링 위치 정보로 변환할 수 있다.

예를 들어 그림 5(a)의 object의 위치 (x, y)를 이진스트링으로 변환하기 위해서, 먼저 (x, y)를 일정한 크기만큼 확장해서 uMBR을 생성한다. 다음으로 그림 5(b)와 같이 도로들이 저장된 R-tree를 통해서 uMBR을 통과하는 도로인 R1, R2, R3을 검색한 후, (x, y)에 가장 가까운 도로인 R1을 선택한다. 다음으로 그림 5(c)에서와 같이 object를 R1 상에 투영한 후, 투영된 지점의 도로에서의 상대적 위치인 “10”을 구한다. 마지막으로 R1의 이진스트링 “00000”과 투영 지점의 상대적 위치 “10”을 연결하여 object의 이진스트링 위치 정보 “0000010”을 구한다.

4.3 AD2BS와 BS2AD

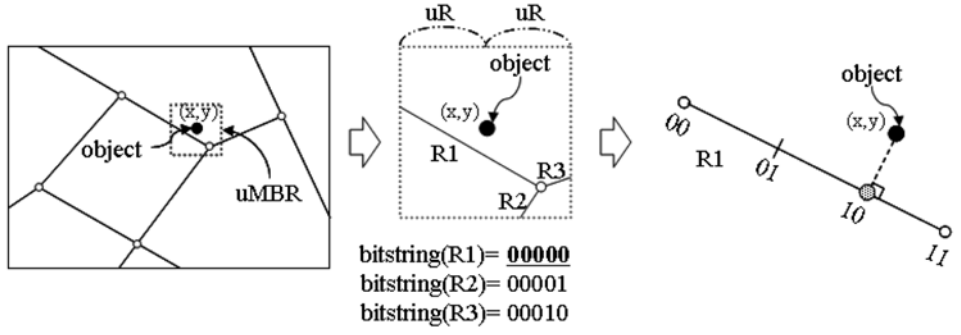
사용자는 주소 형태의 위치 정보를 질의로 이용한다. 또한 질의 결과 역시 주소 형태의 위치 정보로 표현되기를 원한다. 따라서 주소 형태의 위치 정보를 이진스트링 위치 정보로 변환하는 *AD2BS* 모듈과, 이진스트링 위치 정보를 주소 형태의 위치 정보로 변환하는 *BS2AD* 모듈이 필요하다. 위치 정보의 신속한 변환을 위해, *AD2BS* 모듈은 행정 구역 이름과 도로 이름을 연결한 스트링을 키로 이용하여 B-tree를 구축하고 리프 노드에는 해당하는 이진스트링을 저장한다. 또한, *BS2AD* 모듈은 이진스트링을 키로 이용하여 B-tree를 구축하고 리프 노트에는 해당하는 행정 구역 이름과 도로 이름을 저장한다.

5. 질의 처리 방법

이 장에서는 논문에서 제안한 위치 정보 표현 방식을 이용해 위치 기반 서비스의 질의를 처리하는 과정을 설

알고리즘 2. 이동 객체의 (x, y) 위치 정보를 이진스트링 위치 정보로 변환

1) 이동 객체의 위치 정보 (x, y)를 중심으로 X축과 Y축으로 일정한 크기 uR만큼 확장해서 uMBR를 생성한다.
2) 도로가 저장되어 있는 R-tree를 이용해서 uMBR을 통과하는 모든 도로들을 검색한다.
3) 점과 직선 사이의 유클리디언 거리를 이용해서 2)에서 검색된 모든 도로 중에서 (x, y)에 가장 가까운 도로 R을 선택한 후, R에 할당된 이진스트링을 추출한다.
4) 3)에서 선택된 도로에 이동 객체를 투영한 후, 투영된 지점의 도로에서의 상대 위치를 계산해서 이진스트링으로 표현한다.
5) 3)에서 구한 도로 R의 이진스트링과 4)에서 구한 상대 위치의 이진스트링을 연결해서 이동 객체의 이진스트링 위치 정보를 구한다.



(a) object의 위치 (x, y)와 uMBR (b) uMBR를 통과하는 도로 검색 (c) 가장 가까운 도로 선택 및 상대적 위치 계산

그림 5 이동 객체의 위치 정보 변환 과정

명한다.

5.1 영역 질의

영역 질의는 주어진 시간에 특정 영역 안에 있었던 모든 이동 객체를 찾거나 어떤 이동 객체가 특정 영역에 있었던 시간을 구하는 것이다. 영역 질의의 예로 “10시부터 11시 사이에 서울시 서대문구 신촌동에 있었던 모든 이동 객체를 찾아라.”를 생각할 수 있다. 이 질의를 처리하기 위해서는 먼저 이동 객체의 위치를 표현하는 영역을 이진스트링의 범위로 변환해야 한다. 즉, “서울시 서대문구 신촌동”을 *AD2BS* 모듈을 통해서 이진스트링의 최소값과 최대값의 범위로 변환해서 해당 영역에 포함된 모든 도로와 도로 상의 모든 위치를 표현한다. 이진스트링의 범위로 변환된 위치 정보와 “10시부터 11시 사이”의 시간 정보를 이용하여 데이타베이스를 검색하면 조건을 만족하는 이동 객체들이 결과로 나오게 된다.

알고리즘 3은 영역 질의 처리를 위해 위치 정보 변환 과정과 인덱스(R-tree)에서의 질의를 수행하는 과정을 기술한다.

5.2 궤적 질의

궤적 질의는 주어진 시간 간격 동안에 이동 객체가 움직였던 경로를 검색하는 것이다. 궤적 질의의 예로 “철수가 20시부터 21시까지 움직였던 경로를 찾아라.”를 생각할 수 있다. 2차원 공간 좌표로 위치를 표현하는 기존의 시스템에서는 궤적 질의의 결과를 (x, y)의 좌표가 연결된 라인 세그먼트의 연속으로 나타내므로 전자 지도가 없는 환경에서는 유용하지 못하다. 그러나 본 논문에서 제안된 방식에서는 질의의 결과가 계층적 행적 구역의 정보를 내포하고 있기 때문에 텍스트나 음성으로도 쉽게 표현할 수 있다.

궤적 질의의 처리를 위해서는 먼저 알고리즘 4와 같이 객체 정보와 시간 정보를 이용해 데이타베이스를 검색한 후 결과로 나오는 이진스트링들을 시간의 오름차순으로 정렬한다. 다음으로 각 이진스트링을 *BS2AD* 모듈을 통해 행정 구역 이름으로 변환하여 사용자에게 텍스트나 음성의 형태로 보여준다.

이 때, 공통 접두어를 가진 이진스트링들을 행정 구역 단위별로 압축하여 표현함으로써 질의 결과를 계층적으로 보여줄 수 있다. 즉, 최초에는 “구” 단위의 결과를 보여주고 사용자가 자세한 결과를 알고 싶을 때는 “동” 단위의 결과로도 보여줄 수 있다. 예를 들어, 그림 6에서처럼 “Sam의 20:00부터 21:00까지의 궤적을 구해라”라는 질의에 대해, “구” 단위의 결과로 표현할 때는 <“구”에 진입한 시간, “구”에 머물렀던 시간, “구”의 이름>의 리스트로 압축하여 표현할 수 있다. <20:00, 5, “A city a-gu”>의 투플은 “A city a-gu”에 진입한 시간이 20:00이고, “a-gu”에 머물렀던 시간이 5분이었다는 것을 의미한다. 또, “동” 단위로 결과를 표현할 경우, <“동”에 진입한 시간, “동”에 머물렀던 시간, “동”의 이름>의 리스트로 표현할 수 있다.

5.3 기타 질의

또 다른 질의의 유형으로는 “10시에서 11시 사이에 시청 앞을 지난 객체들의 1시간 동안의 궤적을 구해라.”라는 복합 질의가 있다. 이 질의는 “10시에서 11시 사이에 시청 앞을 지난 객체들을 찾아라.”라는 영역 질의와 “객체들의 1시간 동안의 궤적을 구해라.”라는 궤적 질의로 구성되어 있다. 궤적 질의에서 사용되는 객체는 10시에서 11시 사이에 시청 앞을 지난 객체들로써, 그 객체

알고리즘 3. 위치 정보 변환 및 영역 질의 처리

1) 주어진 질의를 시간 정보와 위치 정보로 분할한다.
2) 주소 형태의 위치 정보를 ***AD2BS*** 모듈을 이용하여 주어진 주소에 대응하는 이진스트링으로 위치 정보를 변환한다.
3) 주어진 시간 간격과 변환된 위치 정보(이진스트링)를 이용하여 객체의 데이타를 저장하고 있는 R-tree에서 영역 질의를 수행한다.
4) R-tree의 리프 노드 중 주어진 시간 간격과 변환된 이진스트링과 겹치는 노드들을 결과로 반환한다.

알고리즘 4. 궤적 질의 처리

1) 주어진 질의를 객체 정보와 시간 정보로 구분한다.
2) 주어진 시간 간격을 이용하여 R-tree에서 영역 질의를 수행하고, 리프 노드의 객체 정보와 주어진 객체 정보를 비교한다.
3) 2)의 결과에 대해 시간 순으로 정렬한다.
4) 결과 리스트에 대해 위치 정보를 나타내는 이진스트링을 ***BS2AD*** 모듈을 사용하여 주소 형태의 위치 정보로 변환한다.

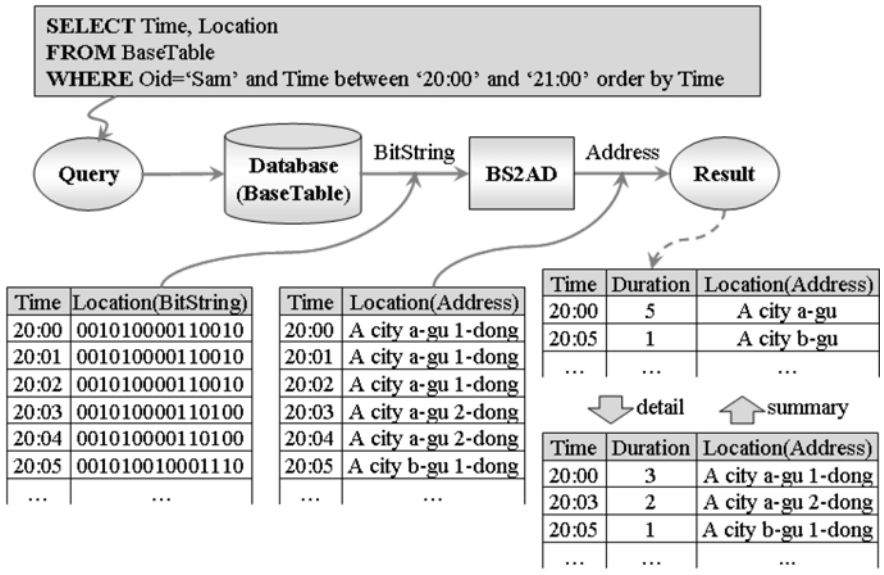


그림 6 궤적 질의 처리 과정 및 결과의 표현 방법

들에 대해 각각 궤적 질의를 수행함에 따라 복합 질의를 처리할 수 있다.

알고리즘 5. 복합 질의 처리
1) 복합 질의를 영역 질의와 궤적 질의로 분할한다.
2) 영역 질의에 대해서 **알고리즘 3**을 수행한다.
3) 2)에 의해 검색된 객체들에 대해 **알고리즘 4**를 각각 수행한다.

그러나 "88도로 위의 객체들을 찾아라."와 같은 영역 질의는 영역 질의에 속하기는 하지만, 하나의 행정 구역에 속한 도로 위를 검색하는 것이 아니라 여러 행정 구역에 걸쳐 나타나는 도로를 대상으로 하고 있다. 이러한 질의를 처리하기 위해서는 강변북로나 88도로 같이 긴 도로에 이름이 붙은 도로들에 대해 별도의 연결 정보를 갖고 있는 인덱스를 구축하여 관리함으로서 위의 질의를 처리할 수 있다.

6. 실 험

본 논문에서 제안한 위치 표현 방식의 효율성을 검증하기 위해서 서울시의 2개 구에 해당하는 영역을 대상으로 실험을 수행하였다. 대상이 되는 2개의 구는 46개 동과 400여개의 도로로 구성되어 있다. 실험에 사용한 이동 객체의 데이타는 다음과 같은 방식으로 생성하였다.

1) 영역 내의 도로 중에서 임의로 하나를 선택하여 그 도로의 시작점을 객체의 최초 위치 정보로 사용한다.
2) 선택된 도로를 따라 객체를 이동시키면서 위치 정보와 시간 정보를 생성한다. 이 때, 단위 시간 당 객체의 이동 거리는 객체의 이동 속도 분포를 고려하여 확률적으로 결정된다.
3) 객체가 도로의 끝에 도달하게 되면 그 도로와 연결되어 있는 도로 중에서 임의로 하나를 선택하여 2)~3) 과정을 반복한다.

위의 방식을 통해 각 이동 객체에 대해 1분 단위로 총 500분 동안 위치 정보를 수집한 후 이를 <시간 정보, x 좌표, y 좌표>의 3차원 데이타로 변환하여 3DR-tree에 저장하거나, <시간 정보, 이진스트링 위치 정보>의 2차원 데이타로 변환하여 2DR-tree에 저장하였다. 이동 객체의 식별자는 키로 사용되어 3DR-tree와 2DR-tree의 리프 노드에 저장되었다. 본 장에서는 3DR-tree와 2DR -tree를 인덱스의 크기 및 질의 처리 성능 면에서 비교함으로써 제안한 위치 표현 방식의 효율성을 검증한다. 실험을 위한 하드웨어 플랫폼으로는 Redhat 계열의 배포판인 Fedora core3를 운영체제로 사용하고, 512MB의 주기억장치와 200GB(7200RPM)의 하드 디스크를 갖는 Pentium Ⅳ 2.6GHz의 PC를 사용하였다.

6.1 XY2BS의 성능

2차원 위치 정보를 이진스트링으로 변환하는데 너무 큰 오버헤드가 든다면 이진스트링 위치 표현 방식이 아무리 효율적이라도 이를 실생활에 적용하기에는 한계가 있다. 따라서 이진스트링 위치 표현 방식의 효율성을 검증하기에 앞서 2차원 위치 정보를 1차원 이진스트링으로 변환하는 *XY2BS* 모듈의 성능을 먼저 측정하였다. 즉, 변환해야 할 2차원 위치 정보의 수를 400,000개로부터 2,000,000개 까지 증가시키면서 이들을 모두 이진스

트링으로 변환하는데 소요되는 시간을 측정하여 *XY2BS* 모듈의 성능 지표로 사용하였다. 그림 7의 결과로부터, 변환에 소요되는 총 시간이 2차원 위치 정보의 수에 비례하여 선형적으로 증가한다는 것과, 초당 7,000,000개 이상의 2차원 위치 정보를 이진스트링으로 변환할 수 있다는 것을 알 수 있었다. 즉, 이 실험을 통해 이진스트링으로의 변환 오버헤드가 크지 않으며, 변환해야 할 2차원 위치의 수가 크게 증가하여도 *XY2BS* 모듈의 성능이 급격이 떨어지지 않는다는 것을 확인할 수 있었다.

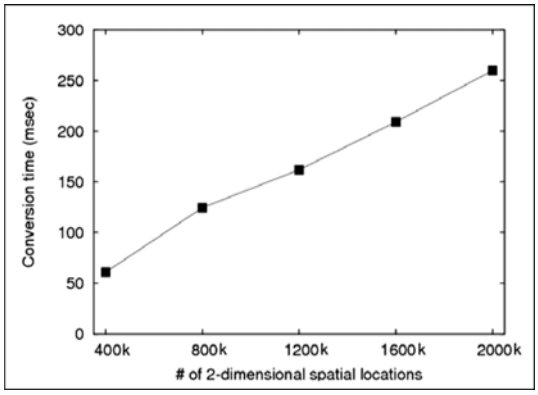


그림 7 2차원 위치 정보를 1차원 이진스트링으로 변환하는 XY2BS 모듈의 성능

6.2 인덱스 크기 비교

제안된 위치 표현 방식에 의해서 인덱스의 크기가 얼마나 감소하는가를 확인하기 위해 이동 객체의 수를 400개로부터 2,000개 까지 증가시키면서 3DR-tree와 2DR-tree의 크기를 비교하였다. 1분 단위로 500분 동안 위치 정보를 수집하였으므로 이동 객체의 수가 400인 경우에는 200,000개의 투플이, 이동 객체의 수가 2,000개인 경우에는 1,000,000개의 투플이 생성된다. 그림 8의 결과에 의하면, 제안된 위치 표현 방식을 이용하는 2DR-tree는 기존의 2차원 위치 표현 방식을 사용하는 3DR-tree에 비하여 약 58% 정도의 공간만을 사용하는 것으로 나타났다. 따라서 인덱스 크기가 약 42% 감소하였으며, 이러한 감소율은 객체의 수가 증가함에 따라 소폭 상승함을 알 수 있었다. 2DR-tree의 경우 AD2BS와 XY2BS 등의 모듈을 처리하기 위한 행정 구역 정보를 저장하는 인덱스가 필요하다. 그러나 각 도로의 위치 정보를 저장하고 있는 인덱스의 크기는 1.1KB로 전체 인덱스 크기에 큰 영향을 미치지 못한다.

6.3 질의 처리 성능 비교

제안된 위치 표현 방식에 의한 질의 처리 성능의 향상을 확인하기 위해 영역 질의와 궤적 질의를 대상으로 2DR-tree와 3DR-tree의 질의 처리 속도를 비교하였다. 각 질의 유형에 대해 1,000개의 질의를 생성하였으며, 이들을 모두 처리하는데 소요된 시간을 질의 처리 성능의 지표로 사용하였다. 본 논문에서 제안한 위치표현 방법은 행정 구역 단위 별로 질의를 처리하는 것을 목적으로 하고 있다. 따라서 본 논문에서 제안한 이진 위치 정보를 이용하게 되면 행정 구역 단위 별로 질의를 수행할 수 있다. 그러나 기존의 위치 정보 표현 방식에서 행정 구역 기반의 질의를 수행하기 위해서는 행정 구역을 포함하고 있는 MBR을 찾아 영역 질의를 수행하고, 질의 행정 구역을 벗어난 객체들에 대해 제거하는 과정을 수행해야 한다. 영역 비교 후, 2DR-tree에서는 질의의 결과가, 3DR-tree에서는 결과의 후보군이 나타나게 된다. 3DR-tree에서는 pruning 과정을 거쳐 질의 결과를 얻어야 하지만, 본 논문에서는 영역 검색에 걸리는 시간만을 실험, 비교하였다.

먼저, “특정 객체가 주어진 ‘동’ 에 있었던 시간을 구하라.”라는 영역 질의(질의 유형 1)와 “특정 객체가 주어진 ‘구’에 있었던 시간을 구하라.”라는 영역 질의 (질의 유형 2)를 수행하였다. 그림 9의 결과를 보면, 객체 수의 증가에 따라 2DR-tree와 3DR-tree의 질의 처리 시간도 선형적으로 증가하지만 2DR-tree의 증가율이 3DR-tree의 증가율보다 작음을 알 수 있다. 3DR -tree에 비해 2DR-tree는 질의 유형 1에 대해 평균적으로 약 98%의 성능 향상을, 질의 유형 2에 대해서는 67%에서 69%까지의 성능 향상을 보였다. 이러한 성능 향상은 제안된 위치 표현 방식에 의해 검색하려는 인덱스의 크기가 줄었으며 검색 영역이 2차원 직사각형에서 1차원 범위로 축소되었기 때문이라고 해석할 수 있다.

다음으로, “처음 250분 동안 주어진 ‘동’ 에 있었던 모

객체 수 (전체 투플의 개수)	3DR-Tree의 크기 (KB)	2DR-Tree의 크기 (KB)	인덱스 크기 감소율 (%)
400 (200,000)	10,973	6,393	41.7
800 (400,000)	22,467	12,779	43.1
1,200 (600,000)	34,342	19,329	43.7
1,600 (800,000)	46,221	25,906	44.0
2,000 (1,000,000)	58,218	32,565	44.1

그림 8 2DR-tree와 3DR-tree의 인덱스 크기 비교

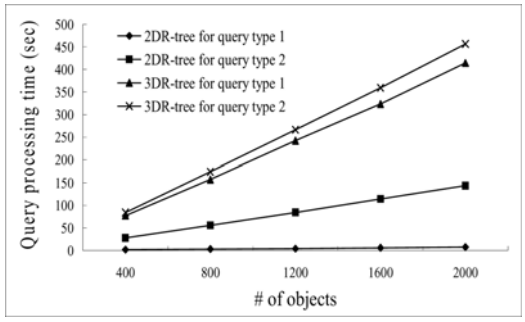


그림 9 “특정 객체가 주어진 ‘동’ 또는 ‘구’에 있었던 시간을 구해라.”에 대한 2DR-tree와 3DR-tree의 처리 시간 비교

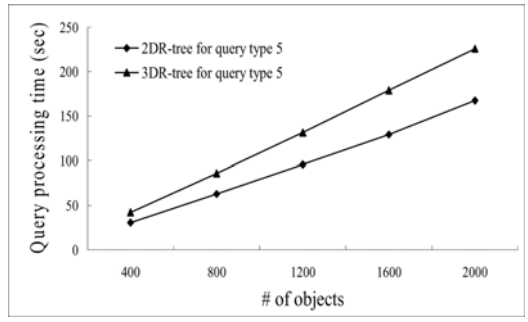


그림 11 “특정 객체 O가 처음 250분 동안 움직였던 경로를 검색하라.”에 대한 2DR-tree와 3DR-tree의 처리 시간 비교

든 이동 객체들을 찾아라.”라는 영역 질의 (질의 유형 3)와 “처음 250분 동안 주어진 ‘구’에 있었던 모든 이동 객체들을 찾아라.”라는 영역 질의 (질의 유형 4)를 수행하였다. 이러한 질의 유형을 처리하기 위해서 2DR-tree는 시간과 이진 위치 정보로 구성된 2차원 영역을 검색해야 하고, 3DR-tree는 시간과 2차원 공간 위치 정보로 구성된 3차원 영역을 검색해야 한다. 그림 10의 결과에 나타나듯이, 객체 수가 증가함에 따라 2DR-tree의 성능 향상이 더욱 커지게 되어 질의 유형 3에 대해서는 최대 98%까지의, 질의 유형 4에 대해서는 최대 64%까지의 성능 향상을 보였다.

마지막으로, “특정 객체 O가 처음 250분 동안 움직였던 경로를 검색하라.”라는 궤적 질의 (질의 유형 5)를 수행하였다. 이러한 질의 유형을 처리하기 위해서 2DR-tree와 3DR-tree 모두 시간에 대한 영역 질의를 수행해야 하지만 3DR-tree에 비해 2DR-tree를 검색하는 것이 CPU 및 I/O 비용 측면에서 유리하다. 그림 11의 결과에 나타나듯이, 3DR-tree에 비해 2DR-tree는 객체 수가 400인 경우에는 약 44%, 객체 수가 2,000 인 경우에는 약 41%의 성능 향상을 보였다.

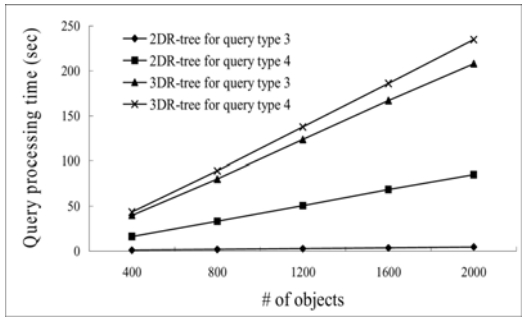


그림 10 “처음 250분 동안 주어진 ‘동’ 또는 ‘구’에 있었던 모든 이동 객체들을 찾아라.”에 대한 2DR-tree와 3DR-tree의 처리 시간 비교

7. 결 론

본 논문에서는, 2차원의 공간 좌표 (x, y)로 표현되었던 이동 객체의 위치 정보를 실세계 모델에 부합하는 계층적 행정 구역과 도로 상의 위치를 이용하여 1차원의 이진스트링으로 표현함으로써, 이동 객체의 저장 공간을 줄이고 질의 처리 성능을 향상시킬 수 있는 효율적인 위치 표현 방식을 제안하였다.

본 논문에서 제안한 위치 표현 방식을 사용하면 위치 기반 서비스를 위한 기존의 인덱싱 및 질의 처리 알고리즘을 단순화시킬 수 있으며, 질의 결과를 행정 구역 단위별로 압축하여 계층적으로 보여줄 수 있다. 또한, 전자 지도가 없는 환경에서도 질의 결과를 텍스트나 음성으로도 쉽게 표현할 수 있다.

위치 기반 서비스를 위해서는 현재 시점뿐만 아니라 과거 시점에서의 위치도 저장해야 하므로 시간이 증가함에 따라 데이타의 양이 급속히 증가하여 대용량의 데이타베이스가 된다. 향후에는, 이진스트링의 공통 접두어를 이용하여 이동 객체의 위치를 행정 구역 단위별로 압축함으로써 전체 데이타베이스의 크기를 줄이는 연구를 수행하고자 한다.

참 고 문 헌

[1] D. Pfoser, C. S. Jensen, and Y. Theodoridis, "Novel Approaches in Query Processing for Moving Objects," In Proc. VLDB Conference, pp. 395-406, 2000.

[2] Y. Theodoridis, M. Vazirgiannis, and T. K. Sellis, "Spatio-Temporal Indexing for Large Multimedia Applications," In Proc. IEEE International Conference on Multimedia Computing and Systems, pp. 441-448, 1996.

[3] M. A. Nascimento and J. R. O. Silva, "Towards Historical R-trees," In Proc. ACM Symposium on Applied Computing, pp. 235-240, 1998.
[4] D. Pfoser, Y. Theodoridis, and C. S. Jensen, "Indexing Trajectories in Query Processing for Moving Objects," Chorochronos Technical Report, CH-99-3, 1999.
[5] Y. Tao and D. Papadias, "MV3R-Tree: A Spatio-Temporal Access Method for Timestamp and Interval Queries," In Proc. VLDB Conference, pp. 431-440, 2001.
[6] A. Guttman, "R-trees: A Dynamic Index Structure for Spatial Searching," In Proc. ACM SIGMOD, pp. 47-54, 1984.
[7] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao, "Query Processing in Spatial Network Databases," In Proc. VLDB Conference, pp. 802-813, 2003.
[8] S. Gupta, S. Kopparty, and C. Ravishankar, "Roads, Codes, and Spatiotemporal Queries," In Proc. ACM PODS, pp. 115-124, 2004.
[9] V. P. Chakka, A. Everspaugh, and J. M. Patel, "Indexing Large Trajectory Data Sets With SETI," In Proc. Conference on Innovative Data Systems Research, pp. 164-175, 2003.
[10] E. Frentzos, "Indexing Objects Moving on Fixed Networks," In Proc. International Symposium on Spatial and Temporal Databases, pp. 289-305, 2003.
[11] V. Almeida, R. H. Gting, "Indexing the Trajectories of Moving Objects in Networks," In Proc. International Conference on Scientific and Statistical Database Management, pp. 115-118, 2004.
[12] D. Pfoser and C.S. Jensen, "Indexing of Network Constrained Moving Objects," In Proc. ACM GIS, pp. 25-32, 2003.
[13] J. A. Orenstein and T. H. Merrett, "A Class of Data Structures for Associative Searching," In Proc. ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, pp. 181-190, 1984.
[14] C. Faloutsos, "Gray Codes for Partial Match and Range Queries," IEEE Trans. on Software Engineering, 14(10), pp. 1381-1393, 1988.
[15] C. Faloutsos and S. Roseman, "Fractals for Secondary Key Retrieval," In Proc. ACM PODS, pp. 247-252, 1989.

이 상 윤
1998년~2004년 B.S. 연세대학교 컴퓨터정보통신공학부. 2004년~현재 연세대학교 컴퓨터과학과 석사과정

박 상 현
1989년 2월 서울대학교 컴퓨터공학과(학사). 1991년 2월 서울대학교 컴퓨터공학과(석사). 2001년 2월 UCLA대학교 전산학과(박사). 1991년 3월~1996년 8월 대우통신 연구원. 2001년 2월~2002년 6월 IBM T. J. Watson Research Center Post-Doctoral Fellow. 2002년 8월~2003년 8월 포항공과대학교 컴퓨터공학과 조교수. 2003년 9월~현재 연세대학교 컴퓨터과학과 조교수. 관심분야는 데이타베이스, 데이타 마이닝, 바이오인포매틱스, XML

김 우 철
1997년~2003년 B.S. 연세대학교 컴퓨터과학과. 2004년~2006년 M.S. 연세대학교 컴퓨터과학과(석사). 2006년~현재 연세대학교 컴퓨터과학과(박사). 관심분야는 바이오인포메틱스, LBS, 데이타베이스 보안, 멀티미디어 데이터베이스 등

이 동 원
2002년~현재 Assistant Professor at College of Information Sciences and Technology / PSU. Assistant Professor at Department of Computer Science and Engineering / PSU (Courtesy). 1997년~2002년 Ph.D. Computer Science, UCLA. 1993년~1995년 M.S. Computer Science, Columbia University. 1988년~1993년 B.S. 고려대학교 컴퓨터과학과