韓國컴퓨터情報學會 論文誌
第16卷 第10號, 2011. 10.

2011-16-10-1-4

도로 네트워크 환경에서 이동 객체의 현재 위치 관리를 위한 효율적인 색인 기법

김 태 규*, 신 숭 선*, 정 원 일**, 배 해 영*

Effective indexing of moving objects for current position management in Road Networks

Tae-Gyu Kim*, Soong-Sun Shin*, Weo-nil Chung**, Hae-Young Bae*

요 약

최근 위치 확인 기술과 이동 통신의 발전에 따라 이동 객체의 위치에 기반한 서비스의 요구가 증가하고 있다. 이에 따라, 이동 객체의 위치 정보를 관리할 수 있는 다양한 색인에 대한 연구가 진행되고 있다. 유클리드 공간의 색인은 이동의 제약이 없기 때문에 이동의 제약이 강한 실세계에 적용하기 어렵다. 도로 네트워크 공간의 색인의 경우에는 인접도로의 연결 정보 알기 위해 추가적인 비용이 드는 문제점이 있다. 또한, 기존의 연구들은 건물, 병원과 같은 위치가 고정되어 있는 객체는 고려하지 않고 있다. 본 논문에서는 도로 네트워크 공간에서 이동 객체의 현재 위치를 효율적으로 관리하기 위한 색인을 제안한다. 제안하는 색인 구조는 도로망을 격자로 분할하여 도로 연결 정보를 색인 구조에 포함하였고 고정 객체를 위해 별도의 색인을 유지한다. 그리고 실험을 통하여 기존의 기법보다 제안 색인의 성능이 우수함을 보인다.

▸ Keyword : 이동 객체, 색인 구조, 도로 네트워크, 도로 연결 정보

Abstract

Recently, advances in mobile communication and location identifying technology of the moving object is evolving. Therefore, the location-based services based on request for service have increased and a variety of the indexing for the position management of moving objects has been studied. Because the index based on Euclidean space are no restriction of movement, it is difficult

• 제1저자 : 김태규 • 교신저자 : 정원일
• 투고일 : 2011. 07. 08, 심사일 : 2011. 08. 01, 게재확정일 : 2011. 08. 13.
* 인하대학교 컴퓨터정보공학과(Dept. of Computer Science & Information Technology, Inha University)
** 호서대학교 정보보호학과(Department of Information Security, Hoseo University)

to apply to the real world. Also, there is additional cost to find adjacent road segments in road networks-based indexing. Existing studies of fixed static objects such as buildings or hospitals are not considered. In this paper, we propose an efficient road networks-based indexing for management of current positions. The proposed indexing partitions road networks by grids and has integrated road connection informations and manage separated extra indexing for fixed static objects. Through the experiment, we show that the proposed indexing based on road networks improves the performance of operation for search or update than existing indexing.

▸ Keyword : moving object, indexing, road network, road connection

I. 서 론

이동 통신 기술이 발달함에 따라 GPS 기술의 무료 개방 및 무선 인터넷 망의 개방과 세계 각국의 정책적 지원에 힘입어 위치 기반 서비스(Location-bsed service)가 이동통신 분야에서 새로운 기반 기술로 각광받고 있다. 위치 기반 서비스는 GPS와 같은 기술을 사용하여 이용자의 위치를 파악하고 이와 관련된 다양한 응용 서비스를 지원한다. 위치 기반 서비스에서 적용 가능한 서비스 분야에는 자산 추적, 차량 추적, 교통량 감시 시스템, 주변정보 조회, 내비게이션과 같은 생활 편의를 위한 서비스들이 있다. 또한, 위치 기반 서비스는 하천 모니터링, 화재 감시 모니터링과 같은 u-방재 서비스의 기반 기술로도 사용되고 있다. 이와 같은 서비스를 지원하기 위해서는 시간의 흐름에 따라 위치가 변화하는 시공간 데이터(spatio-temporal data)의 특성을 지니는 이동 객체의 위치 정보를 효과적으로 저장 및 검색하기 위한 기술이 필수적이다[1-7][10].

이동 객체는 주기적으로 자신의 위치 정보를 서버로 보내게 되고 서버는 이동 객체의 갱신되는 위치 정보를 저장하게 된다. 이 때 저장되는 데이터는 각 이동 객체에 대해 전송되는 위치 정보를 연속적으로 저장하거나(과거궤적정보), 현재 위치 정보만을 저장한다(현재위치정보). 또한 이동 객체의 위치 정보와 함께 속도, 방향 등과 같은 부가적인 정보를 함께 저장(미래위치예측)하거나 각 응용에 맞게 위치 정보를 혼합하여 저장한다. 이렇게 저장된 과거, 현재, 미래 위치 정보는 범위 질의(range query), k-최근접 질의(k-nearest neighbor query), 궤적 질의(trajectory query)와 같은 질의를 효율적으로 처리할 수 있는 색인 구조에 대한 많은 연구가 진행 되고 있다[4, 5, 7, 20]. 연구 되고 있는 색인은 크게 색인에 적용되는 공간 모델링을 사용하여 하늘과 같이 이동 객체의 움직임에 제약이 없는 공간인 유클리드 공간과 움직임의 제약이 강한 도로 네트워크 공간에서 사용되는 색인으로 나눌 수 있다[5][13-14][16].

지금까지 연구 되었던 색인의 대부분은 대량의 이동 객체에 대한 갱신 시의 비용 문제를 여전히 가지고 있다. 유클리드 공간 색인의 경우에는 갱신 문제를 갱신 지연 기법 등을 적용하여 해결하였지만 대량의 이동 객체에 대한 갱신 요청이 극히 짧은 주기로 발생한 경우 성능이 저하되는 문제점이 존재한다. 그리고 움직임의 제약이 없는 유클리드 공간에서 연구가 진행되었기 때문에 건물, 도로와 와 같이 이동 객체의 움직임에 제약이 강한 실세계에 적용하기 어렵고 이동 객체의 위치 정보를 갱신하는 경우, 이미 저장되어 있는 이동 객체를 빠르게 찾기 위해 해싱(hashing)과 같은 보조 색인을 사용하게 되는데, 이 때 사용되는 보조 색인의 유지비용이 크다는 문제점이 있다. 또한, 유클리드 공간에서의 색인에서 관리 되는 이동 객체는 인접한 도로로 이동해야 하는 특성을 갱신 연산이나 질의 처리 시에 활용하지 못하여 비효율적이다. 도로 네트워크 공간 색인의 경우에도 빈번한 갱신 문제, 보조 색인의 유지비용 문제나 도로의 연결정보를 활용하지 못한다는 문제점이 존재한다.

본 논문에서는 위와 같은 문제점들을 해결하기 위해 이동 객체의 현재 위치를 효율적으로 저장할 수 있는 색인을 제안한다. 제안하는 색인은 빈번한 갱신 문제와 보조 색인의 유지비용 문제를 해결한 LUGrid[15]를 실세계에 적용할 수 있는 도로 네트워크 공간으로 확장하였다. 갱신 연산 비용을 줄인 LUGrid는 갱신 연산을 할 경우 격자(grid)에 저장되는 객체의 수에 따라 격자들이 합병(merge)되거나 분할(split)된다는 문제점과 주유소, 병원과 같이 움직임이 전혀 없는 정적 객체를 동적인 이동 객체와 같은 색인에서 관리를 하여 정적 객체의 위치가 변하지 않는다는 특성을 활용할 수 없다는 문제점이 있다. 제안하는 색인 구조는 위와 같은 LUGrid의 문제점들을 해결하였다.

본 논문의 장점은 다음과 같다. 첫 번째로 격자(grid) 기

반의 도로 네트워크 모델을 제안한다. 두 번째로 LUGrid를 확장하여 격자의 분할 및 합병 문제를 해결하고 도로의 연결 정보를 사용하여 이동 객체의 검색 처리 성능을 향상시켰다. 또한, 건물과 같이 고정되어 있는 정적 객체 관리를 위한 색인 구조를 추가하여 질의를 처리할 경우 제안 색인에서 정적 객체 정보를 사용할 수 있도록 한다. 마지막으로 실험을 통해 기존의 색인보다 제안 색인의 성능이 우수함을 보인다.

논문의 구성은 다음과 같다. 2장에서는 이동 객체의 현재 위치 색인과 관련된 기법 및 도로 네트워크 모델에 대해 알아본다. 그리고 3장에서는 제안하는 색인 기법의 구조와 아이디어, 관련 알고리즘 및 제안하는 도로 네트워크 모델에 대해 알아본다. 4장에서는 실험을 통해 제안하는 색인 기법의 성능을 평가하고 5장에서는 결론 및 향후 연구 과제에 대해 정리한다.

II. 관련 연구

2.1 도로 네트워크 기반의 색인 기법

유클리드 공간의 색인은 이동 객체의 제약이 전혀 없기 때문에 실세계에 적용하기 어려워 도로망의 제약적인 특성을 반영한 도로 네트워크 공간에서의 색인에 대한 연구가 활발히 진행되어 왔다. 도로망에서의 특성을 반영할 경우, 이동 객체의 이동 방향을 쉽게 예측할 수 있다. 또한, 이동 객체는 도로 위에서만 이동하므로 도로를 분할하여 도로 세그먼트 단위로 이동 객체들을 관리 할 수 있기 때문에 갱신 비용을 줄이고 검색 성능을 향상 시킬 수 있다[18].

IMORS[13]는 동적 색인(dynamic index)의 경우, 갱신 시에 중간 노드(internal node)가 합병과 분할을 반복하게 된다는 문제점과 도로는 거의 변하지 않는다는 점에 착안하여 색인을 정적 색인과 동적 색인으로 나누었다. 정적 색인은 도로망을 교차점을 기준으로 하여 폴리라인(poly line)으로 분할하여 도로 세그먼트를 만들고 R*-tree의 단말 노드(leaf node)인 MBR에 대응되는 로드 섹터 블록(Road sector block)으로 설정하여 해당 도로 세그먼트 위의 이동 객체들을 저장하였다. 그리고 이동 객체의 실제 위치 정보는 이동 객체 데이터 블록(data block for moving object)에 저장한 뒤, 갱신 시에 검색 비용을 줄이기 위해 로드 섹터 블록, 이동 객체 데이터 블록에 저장된 동일 이동 객체 간에 양방향 포인터를 두었다. IMORS의 갱신 정책에 의해 이동 객체의 위치 정보는 로드 섹터 블록을 벗어날 경우에만 검색 비용이 발생하게 된다. IMORS은 색인의 구조가 거의 변하지 않는 정적 색인으로 도로망을 색인하고 갱신 정책에 의해 갱신 비용을 줄여 유클리드 공간의 색인보다 성능을 향상시켰다. 하지만 갱신 연산을 한 개씩 처리한다는 점, 위치 정보 관련 데이터를 로드섹터 블록과 이동 객체 데이터 블록 두 곳에 저장, 도로의 연결 정보를 색인에 반영하지 못하였다는 점과 같은 문제점을 가지고 있다[14, 15, 17].

IORN-tree[14]는 IMORS에서 사용한 도로 네트워크 모델을 확장하여 도로의 연결 정보를 색인 구조에 포함시켰다. 분할한 도로 세그먼트들은 데이터 노드에 저장하고 데이터 노드들에 대해 R-tree를 구성한다. 또한, 도로의 연결 정보를 데이터 노드에 포함시켜 갱신 연산이나 검색 시에 이를 사용할 수 있게 하였다. 그리고 이동 객체의 갱신 연산 수행 시, 검색을 빠르게 처리하기 위해 이동 객체의 위치 정보를 저장한 데이터 노드 접근을 위한 해시 테이블을 구성하였다. IORN-tree는 모든 이동 객체에 대해 해시 테이블을 유지하는데 고비용이 소요된다는 문제점과 이동 객체의 위치가 다른 도로 세그먼트로 이동한 경우의 갱신 연산을 수행할 시, 저장될 새로운 도로 세그먼트를 찾기 위한 검색 비용 및 IMORS와 마찬가지로 갱신 연산을 한 개씩 처리한다는 문제점이 있다.

2.2 갱신 연산 비용을 최소로 하는 색인 기법

2.2, 2.3 절에서 살펴본 바와 같이 이동 객체 환경에서의 색인은 대량의 이동 객체들의 갱신 연산 비용이 크다는 문제점과 빠른 검색을 위해 모든 이동 객체에 대한 보조 색인의 유지비용이 크다는 문제점을 가지고 있다. 이를 해결하기 위해 유클리드 공간 상에서 최소 갱신 비용 기법을 적용한 LUGrid[15]가 연구되었다.

LUGrid[15]는 Grid File[19]을 기반으로 확장한 기법으로 갱신 연산이 삭제와 삽입으로 구성된다는 점에 착안하여 지연 삽입(Lazy-insertion)과 지연 삭제(Lazy-deletion) 기법을 적용하여 갱신 비용을 줄였다. Grid File과 다른 점은 격자 디렉토리(grid directory)를 디스크가 아니라 메모리에 유지한다는 점이다. Grid File의 격자 디렉토리에 해당하는 LUGrid의 메모리격자(MG:Memory Grid)는 객체의 갱신 정보를 디스크에 바로 저장하지 않고 임시 저장(buffering)한다. 디스크의 버켓 페이지(bucket page)에 해당하는 디스크 격자(DG:Disk Grid)는 이동 객체의 위치 정보를 저장하고 있다. 또한, 해시 기반의 삭제 대상 메모(MDM:Miss-Deletion Memo)를 유지하여 삭제 대상이 되어 쓸모가 없게 된 위치 정보들(obsolete entries)을 저장한다. 그리고 MDM에 저장되는 삭제 대상 엔트리의 수를 제한하기 위해

특정 갱신 횟수에 도달하게 되면 Cleaner 수행하여 삭제 대상 위치 정보를 MG와 DG에 반영하게 한다. [그림 1]은 이와 같은 정책들이 적용된 LUGrid의 전체적인 구조이다. LUGrid에서 질의를 처리할 경우에는 먼저, 질의 범위(query range)와 MG의 격자 중에 겹침이 발생하는 격자들을 선별한 뒤, MG에 저장되어 있는 최신의 위치 정보 중에 질의 범위에 포함되는 것을 질의 결과 집합에 포함한다. 그리고 선별된 MG의 격자들에 대응되는 DG의 격자들을 한 개씩 읽어오면서 DG의 각 격자에 저장된 이동 객체의 위치 정보가 MDM에서 삭제대상으로 설정되어 있는 지와 질의 범위에 포함되는 지를 판단하여 질의 결과 집합에 포함하여 질의 처리를 종료한다. LUGrid의 경우, 대량의 이동 객체들이 군집하여 빠르게 이동할 경우에는 DG의 격자가 언더플로우와 오버플로우를 반복하게되어 DG와 MG의 합병과 분할이 많이 발생하게 되는 문제점과 도로망의 정보가 포함되어 있지 않은 MG와 질의 범위간의 겹침이 되는 범위에 대해 질의를 처리하여 질의 대상 범위가 넓다는 문제점이 존재한다.

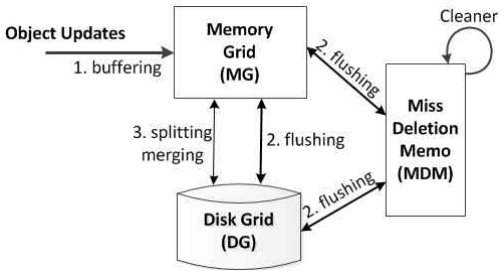

그림 1. LUGrid 전체 구조
Fig 1. Overview LUGrid

III. 본 론

3.1 제안 도로 네트워크 공간 모델

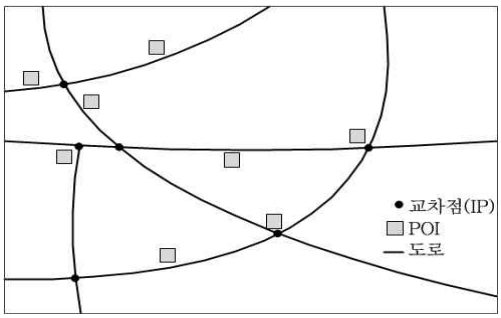

그림 2. POI를 포함하는 실제 도로망
Fig 2. Road networks included poi

도로 네트워크는 도로망이 라인으로 표현되고 도로망이 교차하는 지점에서 교차점이 생성된다. 라인은 인접한 공간으로 뻗어 나가게 되는데 이와 같은 도로의 인접성을 표현하면서 LUGrid[15]를 도로 네트워크 공간 모델로 확장하기 위해 Grid 기반의 도로 네트워크 공간 모델을 사용하였다.

제안하는 도로 네트워크 공간 모델은 Grid File[19]을 기반으로 IRON-tree[14]에서 사용된 모델을 확장하였다. [그림 2]는 건물, 병원, 주유소와 같은 POI(Position-Of-Interest)가 포함된 실제 도로망이다. LUGrid[15]를 도로 네트워크 공간으로 확장하기 위해 실제 도로망을 교차점을 기준으로 격자 분할하여 교차점(IP), 연결점(CP)과 같은 도로의 연결 정보를 메모리 상(MG:Memory Grid)에 저장하게 된다. [그림 3]의 차례대로 교차점인 IP(Intersection point)를 기준으로 격자(grid)를 나눈다. 이 때, IP는 각 격자당 1개씩 저장되게 하고 격자의 경계에 포함된 연결점인 CP(Connection point) 역시 격자 정보를 저장하는 메모리 격자(MG)에 같이 저장되게 한다.(자세한 구성은 다음절에서 살펴본다)

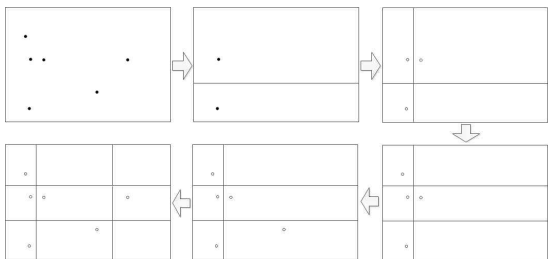

그림 3. 도로망 분할 과정
Fig 3. Road networks partitioning

POI에 대한 정보는 R*-tree[6]을 사용하여 [그림 4]와 같이 색인을 구성한다.

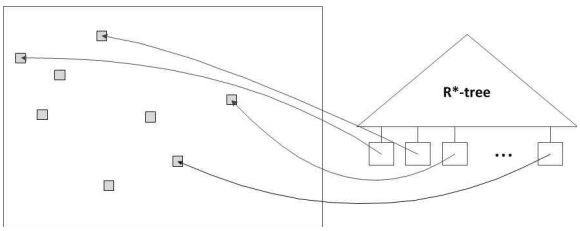

그림 4. POI 색인 구성
Fig 4. POI Index building

[그림 5]는 실제 도로망인 [그림 2]를 본 논문에서 제안하는 도로 네트워크 공간 모델로 구성한 색인 구조의 간단한 예시이다.

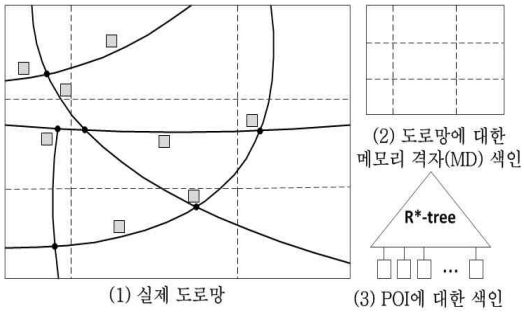

그림 5. 제안 색인의 예
Fig 5. Example of proposed index

3.2 제안 색인 구조

본 절에서는 제안 색인 구조에 대해 자세히 살펴본다.

먼저, 본 논문에서 제안하는 색인 구조는 LUGrid[15]를 도로의 연결 정보를 포함하는 도로 네트워크 공간으로 확장하였고 갱신 연산중에 발생하는 합병 및 분할되는 문제점을 해결하였다. 또한, 건물, 병원, 주유소와 같은 POI(Point-Of-Interest)에 대한 검색을 처리할 수 있게 하기위한 색인 구조를 추가하였다.

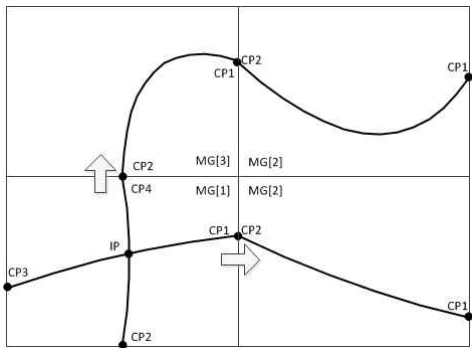

그림 6. 도로 연결 정보
Fig 6. The road connection info.

도로 연결 정보를 포함하는 도로 네트워크 공간으로 확장하기 위해 실제 도로망을 분할할 경우에는 Grid File[19]을 사용하여 교차점(IP)를 기준으로 분할하고 연결점(CP)이 포함되게 한다. Grid File에서는 메인 메모리 상에 해시 테이블과 같은 선형 눈금자(linear scale)를 두어 디스크의 해당 격자의 페이지 아이디(page id)를 저장하여 둔다. 제안 색인 구조는 이를 확장하여 메인 메모리 상에 MG에 격자 디렉토리(grid directory)와 도로 연결 정보 및 LUGrid에서 사용하는 속성 값을 저장한다. 이와 같이 구성할 경우의 이점은 Grid File의 분할된 격자를 이용하여 도로의 연결성을 확인할 수 있다. 즉, 인접한 격자에 해당 도로가 이어져 있다. 또한, 도로 연결성의 정확한 사용을 위해 격자에 저장된 CP와 인접한 격자의 CP를 사용하여 도로가 연결되었음을 검증할 수 있다[14]. [그림 6]과 같은 경우, MG[1]의 도로는 MG[1]에 인접한 격자인 MG[2]와 MG[3]의 도로로 연결되어 있고 이는 MG[1].CP1 = MG[2].CP2와 MG[1].CP4 = MG[3].CP2인 것으로 검증할 수 있다.

위와 같이 구성할 경우, 분할된 격자는 IP를 포함하는 격자와 IP 없이 CP만 포함하는 격자로 구분할 수 있다. [그림 7]에서 이것을 확인할 수 있다[13, 14]. 교차점이 포함된 격자에 대해서는 [그림 8]-(1)([그림 7]-(2)에 대응)과 같이 격자 디렉토리에 IP, CP 정보를 포함 시킨다. 여기서, Grid Info는 선형눈금자(linear scale), IP는 교차점의 위치 정보, CP count는 총 연결점 수, CP List는 CP count만큼 저장된 연결점 정보이다. 그리고 LUGrid의 갱신 연산 처리 중에 발생하는 격자의 합병 및 분할을 방지하기 위해 해당 MG의 격자에서 사용하는 DG의 디스크 페이지의 수와 디스크 페이지에 포함되는 이동 객체의 정보를 추가하였다. 갱신되는 이동 객체의 가장 최근의 위치 정보만을 저장하는 MG의 각 격자에서 엔트리 수의 언더플로우(underflow)는 무시하고 오버플로우(overflow)가 발생하면 오버플로우 페이지를 추가하여 사용한다([그림 8]-(4)). 사용되는 속성 값은 N_u(해당 격자 안에 버퍼링된 갱신 위치 정보 수), D_{cnt}(해당 격자에서 사용 중인 디스크 페이지 수), D_{id}(해당 격자에 연결된 DG의 디스크 페이지 ID), N_E(디스크 페이지에 저장된 이동 객체의 위치 정보 엔트리 수), E_1, ..., E_m(MG 격자 안에 버퍼링된 엔트리의 object id, 최신 위치 정보)가 있다. 교차점이 없는 격자에 대해서는 [그림 7]-(2)([그림 7]-(3)에 대응)과 같이 [그림 8]-(1)에서 IP정보와 CP count 정보만 없는 구조이다. [그림 8]-(5)는 삭제 대상 메모(MDM:Miss-Deletion Memo)의 구조를 나타낸다. OID는 Object ID, OLoc은 위치 정보, D_{vm}은 OID에 대한 삭제 대상 수(miss deletion number)이다.

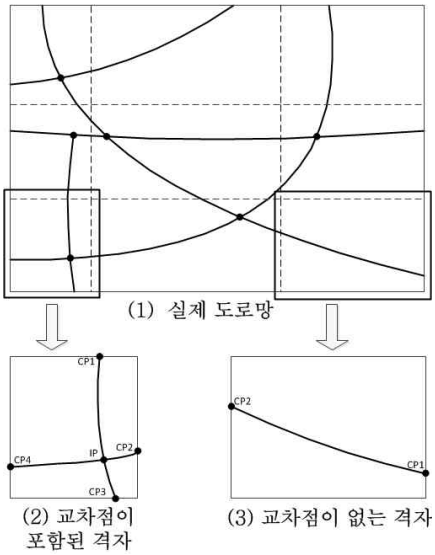

그림 7. 격자(grid)에 저장된 두 가지 종류의 도로 세그먼트
Fig 7. two type of road segment

MG(Memory Grid)의 격자에 대응되는 DG(Disk Grid)의 디스크 페이지 구조는 [그림 8]의 (3), (4)와 같이 두 가지로 구분할 수 있다. [그림 8]-(3)은 도로 정보와 POI 정보를 포함한 기본 디스크 페이지 구조로서 D_{id}는 디스크 페이지 ID, Poly-Line은 도로의 Poly-Line 정보이다. POI List는 건물, 병원과 같은 POI의 정보가 저장되어 있는 색인의 단말 노드를 가리키는 포인터이다. N_E는 해당 페이지에 저장되어 있는 위치 정보의 엔트리 수, Entry set은 DG의 디스크 페이지에 저장되어 있는 이동 객체의 ID와 위치 정보를 저장한다. [그림 8]-(4)는 오버플로우 페이지의 구조이다.

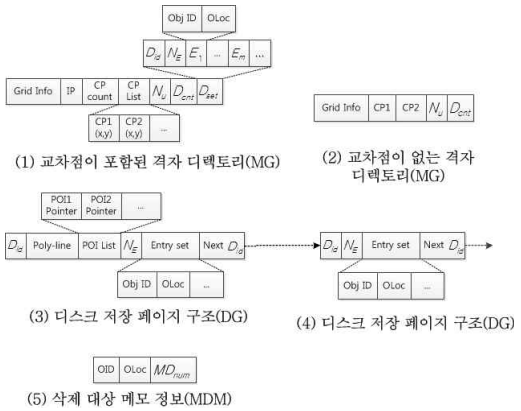

그림 8. 확장된 MG, DG, MDM 구조
Fig 8. Expanded MG, DG and MDM structure

3.3 색인 생성 과정 (Indexing construction)

제안 색인은 다음의 세단계로 구축된다.

먼저, 색인 생성은 도로망에 대한 색인을 3.2절에서 제안한 구조로 생성한다. 도로망에 대한 색인과 IP, CP와 같은 도로 연결 정보는 고정되어 있고 거의 변하지 않는다. 또한, 색인 구성 시에 모든 IP와 CP의 수를 알 수 있으므로 처음 생성되는 비용을 제외하고는 이동 객체의 갱신 시에 별도의 비용이 필요하지 않다. 다음으로 건물, 병원과 같은 POI에 대해 R*-tree의 색인을 생성한다. 도로망에서와 같이 POI에 대한 색인도 고정적인 객체를 대상으로 하고 추가되는 객체가 많지 않으므로 처음 생성 비용 외에는 추가 비용이 거의 필요하지 않다. 마지막으로 생성된 2개의 색인을 통합해 준다. 동일 공간을 대상으로 도로망과 POI에 대한 색인이 구성되었으므로 도로망 색인의 각 분할된 격자에는 POI에 대한 색인인 R*-tree의 단말 노드가 포함된다. 그러므로 도로망 색인의 각 격자에 중첩되거나 포함되는 R*-tree의 단말노드에 대한 포인터를 유지한다. [그림 9]는 통합된 색인의 예이다.

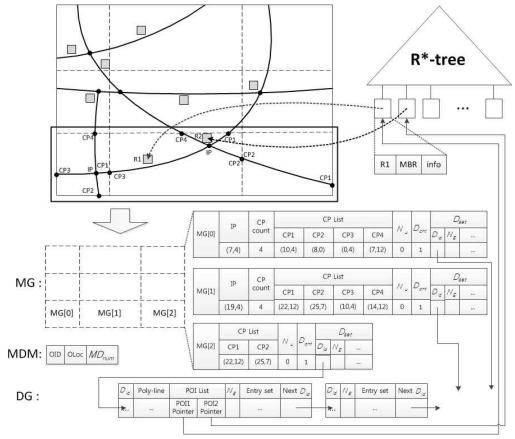

그림 9. 통합된 색인 예
Fig 9. Example of integrated index

3.4 갱신

제안하는 색인의 갱신 연산은 LUGrid[15]의 연산을 확장하여 기존 기법에서 분할 및 합병 과정을 제거하였다. 분할 및 합병 과정을 제거하기 위해 MG의 격자(grid) 대응되는 DG의 격자에 오버플로우(overflow) 페이지를 추가할 수 있도록 확장하였다. 2.3절에서 정리한 바와 같이 LUGrid의 갱신 연산은 이동 객체의 위치 정보를 메모리 격자(MG)에 임시 저장(buffering)하는 과정과 임시 저장된 정보를 디스크 격자(DG)에 쓰는 과정으로 구성된다.

표 1. 갱신 정보 임시 저장 알고리즘
Table 1. Update data buffering Algorithms

```
Algorithms 1
input : u(oid, loc)
output : buffered u in grid of MG
1.  MG m = searchMG(u.oid);
2.  if ( isNull(m) = false ) then
3.  deleteMG(m); m.Nu —; usedSlots—;
4.  else
5.  m = searchCoveredMG(u.loc);
6.  endif
7.  isfull = true;
8.  foreach Dset e in m.Dset
9.  if ( NE != full ) then
10. insertMG(u, e.Did); isfull=false;
11. endif
12. endfor
13. if ( isfull ) then
14. d = createOverflowPage();
15. Dcnt++; addMG(d, m); insertMG(u);
16. endif
17. u.Nu++; usedSlots++;
18. if ( u.Nu >= MaxUpdPerMGCell ) or
19. ( usedSlots >= MaxSlots ) then
20. FlushingUpdates(m);
21. endif
```

[표 1]은 갱신 정보를 MG에 임시 저장하는 확장된 알고리즘이다. 갱신 위치 정보 u가 입력 될 경우, u에 대해 디스크에 쓰여지지 않는 최신 위치 정보가 MG에 저장되어 있는지 검색하여 만일 존재한다면, MG에 저장되어 있는 위치 정보를 삭제하고 존재하지 않을 경우에는 u가 저장될 수 있는 MG의 격자를 검색한다(라인 1-6). MG의 격자 정보 D_{id}를 조회하여 저장 가능한 DG의 격자를 찾는다. 만일, DG의 모든 페이지에 저장할 공간이 없다면, 오버플로우 페이지를 생성하여 저장한다(라인 8-16). MG의 격자의 총 엔트리 수와 사용 슬롯 수를 검사하여 DG의 격자에 쓰기를 할지 결정한다(라인 18-21).

MG 내 임의의 격자 한 개의 위치 정보를 DG의 격자에 쓰기위한 알고리즘은 DG의 격자가 여러 페이지에 걸쳐 저장되어 있는 경우, 페이지를 읽기 위해 MG 내 격자 정보 중에서 D_{set}에 저장되어 있는 여러 페이지 정보를 하나씩 읽으면서 처리하는 것을 제외하고는 기존 LUGrid의 기법과 크게 다르지 않다. MG의 격자에 저장되어 있는 DG의 격자 페이지 정보(D_{set}) 중 한 페이지를 읽어 오면서 다음의 과정을 반복한다. 먼저, 삭제 대상 메모(MDM)에 저장되어 있는 위치 정보를 가져온 DG의 격자 페이지에서 삭제하여 준다. 그 뒤, MG 내 격자에 저장되어 있는 최신 위치 정보를 읽어온 DG의 격자 페이지에 저장하고 마지막으로 디스크에 쓰기를 실행한다.

3.5 질의 수행

제안 색인 구조에서 범위 질의(range query) 처리 알고리즘은 [표 1]과 같다. LUGrid의 범위 질의 처리 알고리즘을 도로의 연결 정보를 사용하도록 확장하였다. 먼저, 질의 영역(range)와 겹치는 메모리 격자(MG)의 집합 Set_{mg}를 구한다(라인 2). 그 뒤에 MG에 포함된 도로망 정보를 사용하여 Set_{mg} 중에 도로가 연결되어 있는 MG인 Set_{input}을 구한다(라인 3). MG에 저장되어 있는 최신 위치 정보를 결과 집합에 포함 한다(라인 4-10). 마지막으로 라인 11-18에서는 Set_{mg}에 대응되는 디스크 격자(DG)의 페이지를 읽어 온 뒤, 해당 페이지 안의 위치 정보를 하나씩 읽어 온다. 읽어온 이동 객체의 위치 정보는 IdentifyingEntry 함수를 실행하여 삭제 대상 메모(MDM)에 표시되어 있는지 검사하여 삭제 대상이 아니라면 결과 집합에 포함한다.

표 2. 범위 질의(Range query) 처리 알고리즘
Table 2. Range query processing Algorithms

```
Algorithms 2
input : range (query range)
output : ResultSet (object set)
1.  ResultSet = null;
2.  Setmg = getOverlapInMG(range);
3.  Setinput = getRoadConnet(Setmg);
4.  foreach MG MGcell in Setinput
5.  foreach object obj in MGcell
6.  if obj.OLoc ∈ range then
7.  ResultSet = ResultSet ∪ obj;
8.  endif
9.  endfor
10. endfor
11. foreach diskpageid Did in Setinput
12. buffer buf = getLoadPage(Did);
```

```
13.  foreach object obj in buf
     if (obj.OLoc ∈ range) and
14.  (IdentifyingEntry(obj)
     = CURRENT) then
15.  ResultSet = ResultSet ∪ obj;
16.  endif
17.  endfor
18.  endfor
```

주변 정보 검색 알고리즘은 [표 2]와 같다. 라인 3에서 질의 영역(range)과 겹치는 메모리 격자(MG)의 집합 Set_{mg}를 구한다. 그 뒤에, Set_{mg}의 값을 한 개 씩 가져오면서 해당 격자 MG_{cell}에 저장되어 있는 POI 색인의 단말 노드 포인터 정보를 가져온다(라인 4-8). 마지막으로 Set_{input}에 저장되어 있는 POI 색인 단말 노드 포인터를 하나 씩 가져오면서 단말 노드의 MBR이 질의 영역과 겹치는(overlap)지 확인하고 건물, 병원 등의 type과 비교하여 만족하는 단말 노드를 결과 집합에 포함한다.

표 3. 주변 정보 검색 알고리즘
Table 3. Point-Of-Interest search Algorithms

```
Algorithms 3
input : range (query range), type
output : ResultSet (object set)
1.   ResultSet = null;
2.   pointer Set_input = null;
3.   Set_mg = getOverlapInMG(range);
4.   foreach MG MG_cell in Set_mg
5.   foreach pointer POI in MG_cell
6.   Set_input = Set_input ∪ POI
7.   endfor
8.   endfor
9.   foreach diskpageid D_id in Set_input
10.  buffer buf = getLoadPage(D_id);
11.  foreach object obj in buf
12.  if (obj.MBR ∩ range) and
13.  (obj.info = type) then
14.  ResultSet = ResultSet ∪ obj;
15.  endif
16.  endfor
17.  endfor
```

IV. 성능 평가

4.1 실험 환경

본 논문에서 제안하는 색인 구조의 우수성을 보이기 위해 기존의 색인 기법인 LUGrid[15]와 성능 평가를 수행한다. 실험에 사용된 시스템의 하드웨어 및 소프트웨어 환경은 [표 3]과 같다. 실험에 사용된 데이터는 Brinkhoff가 제안한 데이터 생성기(Generating Network-Based Moving Object)[20, 21]를 사용하여 생성하였다. 실험에 사용된 데이터 셋은 6105개의 노드와 7035개의 에지 및 교차점의 노드 수는 2238개이고 도로망의 크기는 23854 ×30851이다.

표 3. 성능 평가에 사용된 시스템 환경
Table 3. system environment for experiments

구 분	설 정
CPU	Intel Core Quad Q6600, 2.4GHz
메인 메모리	4GB
하드 디스크	160G
운영 체제	Window XP Professional
개발 언어	Java (JDK 1.6)

4.2 실험 결과

다음은 이동 객체의 갱신에 대한 성능 평가이다. 이동 객체의 데이터는 2,000개에서 최대 10,000개 까지 생성하며, 색인의 성능은 디스크 I/O에 가장 큰 영향을 받으므로 성능 비교는 디스크 I/O 수로 정한다.

[그림 10]은 기존 색인과 제안 색인의 갱신 연산에 대한 실험 결과이다. 이동 객체의 수가 적을 경우, 제안 색인은 기존 색인 보다 갱신 성능이 조금 떨어지는 것을 알 수 있다. 하지만 이동 객체의 수가 증가 할수록 기존 기법보다 성능이 향상됨을 확인할 수 있다. 이와 같은 결과는 기존 LUGrid의 이동 객체 수가 증가 할수록 갱신 처리 중에 디스크 격자(DG)의 분할 및 합병의 수가 증가하게 되에 디스크 I/O가 증가하게 되지만 제안 색인은 오버플로우 페이지에 저장되어 있는 이동 객체의 위치 정보를 가져오기 위한 비용만이 추가되어 기존 색인보다 이동 객체의 증가에 따른 디스크 I/O 비용이 상대적으로 적기 때문이다.

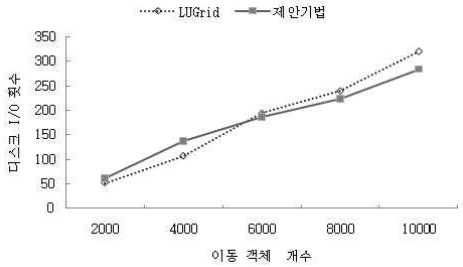

그림 10. 갱신 연산에 대한 성능 비교
Fig 10. Updating cost

[그림 11]은 범위 질의(range query)에 대한 성능 평가이다. 이동 객체의 수는 총 10,000개이고 질의 대상 범위(range)는 전체 공간의 10%에서 90%에서 실험을 진행하였다.

[그림 11]의 결과와 같이, 제안 색인이 LUGrid보다 질의 처리 성능이 우수함을 알 수 있다. 기존의 색인에서는 질의 대상 범위가 넓어질수록 질의 범위와 겹치는 메모리 격자(MG)의 모든 격자와 해당 디스크 격자(DG)의 페이지를 검색해야 한다. 하지만 제안 색인에서는 도로의 연결정보를 사용하여 질의 대상 범위를 줄일 수 있다.

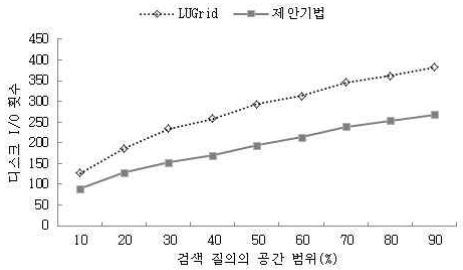

그림 11. 질의 처리에 대한 성능 비교
Fig 11. Query processing cost

V. 결론 및 향후 연구

본 논문에서는 도로 네트워크 공간에서 이동 객체의 현재 위치를 효율적으로 관리하기 위한 색인 구조를 제안하였다. 이를 위해, 교차점을 기반으로 도로망을 격자(grid)로 분할 및 도로의 연결 정보를 색인 구조에 통합하여 질의 처리 중에 사용할 수 있게 하였다. 기존 색인 기법을 갱신 연산중에 격자가 분할 및 병합이 되지 않도록 확장하였고 또한, 건물, 병원과 같은 정적 객체를 관리하는 색인을 따로 두어 질의 처리 시에 검색될 수 있게 하였다. 성능 평가 결과 제안 색인은 기존 기법보다 대량의 이동 객체에 대한 갱신 연산의 성능과 질의 처리 성능이 우수함을 확인할 수 있었다.

향후 연구로는 실험에서 확인된 문제점인 이동 객체 수에 따라 성능이 저하되는 문제점을 개선할 수 있는 연구가 필요하다. 이 문제점은 제안 색인 구조에서 이동 객체의 수에 따라 이동 객체의 위치 정보가 여러 디스크 페이지에 저장되어 발생하는 것이므로 도로의 폭, 너비에 의해 물리적인 이동 객체의 수가 제한된다는 점에 착안하여 분할된 도로 세그먼트가 저장할 수 있는 최대 객체 수까지 저장 가능한 효율적인 디스크 페이지 저장 기법을 연구하여 제안 색인 구조를 확장할 계획이다.

Acknowledgment

본 연구는 건설교통부 첨단도시기술개발사업 - 지능형국토정보기술혁신 사업과제의 연구비지원(07국토정보C05)에 의해 수행되었습니다.

참고문헌

[1] K. M. Yeo, J. H. Ahn, “Location Based Service Technologies and Standards,” Telectronics and Telecommunications Trends, Vol. 25, No. 6, pp.11-19, 10 2010.

[2] S. H. Lee, K. W. Min, J. C. Kim, J. W. Kim, J. H. Park, “Technical Trend of Location-Based Service,” Telectronics and Telecommunications Trends, Vol. 20, No. 3, pp.33-42, 6 2005.

[3] D. H. Ryu, H. G. Lee, K. H. Kim, “Trend of Service and Technology for u-Disaster Prevention”, Telectronics and Telecommunications Trends, Vol. 25, No. 4, pp.142-153, 8 2010.

[4] X. Meng, and J. Chen, "Moving Objects Management," Springer, pp.69-80, 2010

[5] H. M. Yu et al., "2008 Database white paper," Korea Database Promotion Center, pp.133-143, 2008

[6] M. Yannis et al., "R-Trees: Theory and Applications," Springer, pp.18-20, 2004

[7] LV Nguyen-Dinh et al., "Spatio-Temporal Access

Methods : part 2 (2003-2010)," Data Engineering, Vol. 33, No. 2, pp.46-55, June 2010

[8] S. C. Lim, "An Indexing Schema for Predicting Future-time Positions of Moving Objects with Frequently Varying Velocities," The Korea Society of computer and information Journal, Vol. 15, No. 5, pp.23-31 , 5 2010.

[9] M. H. Jeong et al., "A Unified Index of Moving Objects in Road Networks," The Korean Institute of Information Scientists and Engineers Korea Computer Congress 2004, Vol. 31, No. 2, pp.4-6, 2004

[10] J. H. Cheon et al., "UCN-Tree: A Unified Index for Moving Objects in Constrained Networks," Korea Spatial Information Society journal, Vol. 8, No. 1, pp.37-57, 6 2006

[11] Yuni Xia, S. Prabhakar, "Q+Rtree: efficient indexing for moving object databases," Database Systems for Advanced Applications 2003. , pp.175-182, March 2003.

[12] Y. Tao et al., "The TPR*-tree: an optimized spatio-temporal access method for predictive queries," VLDB Journal, Vol. 29, pp.790-801 ,2003.

[13] K. S. Kim, S. W. Kim, T. W. Kim, K. L. Li, "Fast indexing and updating method for moving objects on road networks," Web Information Systems Engineering Workshops 2003, pp.34-42, 2004.

[14] K. S. Bok, H. W. Yoon, D. M. Seo, S. M. Jang, M.-H. Kim, and J. S., Yoo."Indexing the Current Positions of Moving Objects on Road Networks" In APWeb/WAIM Workshops, pp.247-252, HuangShan, China, June 2007.

[15] Xiaopeng Xiong , Mohamed F. Mokbel , Walid G. Aref, "LUGrid: Update-tolerant Grid-based Indexing for Moving Objects," Mobile Data Management 2006, p.13, Nara, Japan, May 2006.

[16] J. Chen and X. Meng., "Update-efficient indexing of moving objects in road network," GeoInformatica, Vol. 13, No. 4, pp.397-424, December 2009.

[17] Y.N. Silva, X. Xiong and W.G. Aref, "The RUM-tree: supporting frequent updates in r-trees using memos," VLDB Journal, Vol. 18, No. 3, pp.719-738, 2009

[18] K. S. Bok, J. S. Yu, "Trend of Moving object indexing," The Korean Institute of Information Scientists and Engineers Korea journal, Vol. 24, No. 12, pp.38-46, 12 2006

[19] J. Nievergelt, Hans Hinterberger, Kenneth C. Sevcik, "The Grid File: An Adaptable, Symmetric Multikey File Structure," ACM Transactions on Database Systems, Vol. 9, No. 1, pp.38-71, March 1984.

[20] Thomas Brinkhoff, "A Framework for Generating Network-Based Moving Objects," Geoinformatica, Vol. 6, No. 2, pp.153-180, June 2002

[21] Generating Network-Based Moving Objects, http://iapg.jade-hs.de/personen/brinkhoff/generator/

저 자 소 개

김 태 규

2009 : 인하대학교 컴퓨터정보공학과 공학사

2010~현재 : 인하대학교 컴퓨터정보공학과 공학석사과정

관심분야 : 데이터스트림, 이동객체, 시공간 데이터베이스

Email : neoprog@gmail.com

신 승 선

2006 : 서원대학교 컴퓨터교육학과 이학사

2008 : 인하대학교 컴퓨터정보공학과 공학석사

2008~현재 : 인하대학교 컴퓨터정보공학과 공학박사과정

관심분야 : 공간 데이터베이스, 그리드 데이터베이스, 데이터스트림

Email : hermit@dblab.inha.ac.kr

정 원 일

1998 : 인하대학교 전자계산학과 공학사
2004 : 인하대학교 컴퓨터정보 공학과 공학박사
2004~2006 : 한국전자통신연구원 선임연구원
2007~현재 : 호서대학교 정보보호학과 전임강사
관심분야 : 데이터스트림, 이동객체, 데이터베이스
Email : wuchung@hoseo.edu

배 해 영

1974년 : 인하대학교 학사학위
1978년 : 연세대학교 석사학위
1989년 : 숭실대학교 박사학위
1985년 : 휴스턴 대학교 객원교수 역임
1992~1995 : 인하대학교 전사 계산소 소장
2004~2006 : 인하대학교 정보통신대학원원장
2006~2009 : 인하대학교 대학원장
1982~현재 : 인하대학교 컴퓨터공학부 교수
1999년~현재 : 지능형 GIS연구센터 센터장
2000년~현재 : 중국 중경우전대학교 대학원 명예교수로 재직
관심분야 : 분산 데이터베이스, 공간 데이터베이스, 지리정보 시스템, 멀티미디어 데이터베이스
Email : hybae@inha.ac.kr