모바일 응용에서 이동 객체의 궤적-기반 질의를 위한 색인 구조

심춘보* · 주재흠*

An Index Structure for Trajectory-based Query of Moving Objects in Mobile Applications

Choon-Bo Shim* · Jae-Heum Joo*

요 약

무선 통신 및 모바일 기술의 급속한 발전으로 인해 위치 기반 서비스 및 모바일 응용 서비스와 같은 이동 객체에 기반한 다양한 서비스에 대한 요구가 증가하게 되었다. 이를 위해, 본 논문에서는 모바일 응용에서 이동 객체에 대한 다양한 질의 중에서 특히 궤적-기반 질의에 대한 성능을 향상시킬 수 있는 색인 구조를 제안한다. 제안하는 색인 구조는 이동 객체의 궤적-기반 질의를 위해 제안되었던 기존의 TB-tree의 삽입 및 검색 성능의 효율성을 높이기 위해 연결테이블(LinkTable:L-Table)을 유지한다. L-Table은 선행 노드를 한번에 직접 접근하기 위해 이동 객체의 처음 세그먼트와 마지막 세그먼트가 저장된 단말 노드를 가리키는 포인터 정보와 디스크에서의 페이지를 가리키는 페이지 번호를 저장한다. 아울러 시스템의 가용메모리가 제약을 받는 경우에도 일정한 크기의 버퍼를 유지해 색인의 일부만을 메모리에 상주시킬 수 있도록 설계한다. 마지막으로 제안하는 색인 구조의 성능을 평가하기 위하여 다양한 실험 데이타를 이용해 성능 평가를 수행한 결과, 삽입 및 궤적-기반 질의 측면에서 기존의 색인 구조들에 비해 더 나은 성능을 보인다.

ABSTRACT

With the rapid development of wireless communications and mobile technologies, requirements of various services based on moving objects like location-based services and mobile applications services have been increased. In this paper, we propose an index structure which can improve the performance on trajectory-based query especially, one of the various query types for moving objects in mobile applications. It maintains link table(L-Table) to obtain good efficiency on retrieval and insertion performance of the existing TB(Trajectory Bundle)-tree proposed for trajectory-based query of moving objects. The L-Table contains page number in disk and memory pointers pointing the leaf node with the first and last line segment of moving objects in order to directly access preceding node. In addition, we design to reside a part of whole index in main memory by preserving a fixed size of buffer in case of being restricted by available main memory. Finally, experimental results with various data sets show that the proposed technique is superior to the existing index structures with respect to insertion and trajectory-based query.

키워드

Mobile Applications, Moving Objects, Trajectory-based Query, Index Structures

*부산가톨릭대학교

접수일자 : 2004. 8. 24

Ⅰ. 서 론

무선 인터넷 및 데이터 전송능력을 지닌 휴대용 단말기의 급속한 성장으로 인해 모바일 응용 서비스나 위치기반 서비스(LBS)가 무선 인터넷 시장에서 중요한 이슈로 급부상하고 있으며, 2004년까지 약 10억대의 모바일 폰 시장이 형성될 것으로 예상하고 있다[1,2]. LBS가 점차 보편화 되어감에 따라 다양한 서비스들이 일정한 장소에서 뿐만 아니라, 휴대용 전화기, PDA, 노트북과 같은 이동성을 지닌 단말기를 이용하여 이동 중에도 지속된다. 따라서 시간의 흐름에 따라 객체가 이동하면서 그 위치를 연속적으로 변경하는 특징을 지닌 이동 객체(Moving Objects)[3]를 위한 다양한 질의 처리와 효율적인 색인[4,5]을 가능하게 하는 기술이 요구된다.

본 연구에서는 모바일 응용에서 이동 객체에 대한 다양한 질의 중에서 특히 궤적-기반 질의에 대한 성능을 향상시킬 수 있는 색인 구조를 제안한다. 이동 객체의 궤적-기반 질의를 위해 기존에 제안되었던 TB-tree는 이동 객체의 궤적 세그먼트를 삽입할 때마다 선행 세그먼트를 가지고 있는 단말 노드를 찾기 위해 루트 노드부터 단말 노드까지 순회해야만 하기 때문에 불필요한 노드 접근으로 인한 오버헤드가 발생한다. 이를 위해 제안하는 색인 구조는 선행 노드를 직접적으로 접근하기 위해 이동 객체의 처음 세그먼트와 마지막 세그먼트가 저장된 단말 노드를 가리키는 포인터 정보와 더불어 디스크에서의 페이지를 가리키는 페이지 번호를 별도의 연결테이블(L-Table)에 같이 유지한다. 따라서 저장의 경우 동일한 이동 객체의 선행노드를 빨리 검색할 수 있고, 궤적-기반 질의 처리의 경우 직접 디스크에 접근해 해당 객체의 궤적들을 검색함으로써 검색 성능을 향상시킬 수 있다. 아울러 색인의 크기가 커질수록 시스템의 가용메모리의 제한으로 인해 성능에 영향을 미칠 수 있기 때문에 일정한 버퍼를 유지함으로써 해당 버퍼를 교체해 나가는 버퍼링 정책을 사용해 가용 메모리를 효율적으로 사용할 수 있다.

Ⅱ. 관련 연구

R*-tree[6]는 R-tree계열에서 가장 성능이 우수하다는 평가를 받고 있으며, 최소 중복 증가 정책(Least Overlap Enlargement Policy)를 이용한 삽입 노드 선택 정책과 아울러 가장자리 값과 중복 값을 고려한 새로운 분할 기법을 제시한 색인 구조로 알려져 있다. 또한 재삽입(reinsert) 정책을 사용하여 기존의 R-tree의 공간 오버헤드를 줄일 수 있도록 설계되었다.

CR-tree[7]는 결합 질의를 효율적으로 처리하기 위해 R-tree와 이동 객체별 연결노드를 결합한 형태이다. 기본적으로 R-tree의 방법을 그대로 이용하며 삽입시 단말 노드와 연결 노드를 연결한다. R-tree부분은 범위질의를, 연결노드는 궤적 보존(trajectory preservation)을 해서 궤적 질의를 빠르게 처리 할 수 있는 구조로 되어 있다.

TB-tree[8]는 극단적으로 이동 객체의 궤적에만 초점을 맞추어 제안된 색인 기법이다. 즉, 시간에 따라 연속적으로 변화하는 위치 정보를 표현하기 위해 이동 객체의 궤적을 라인 세그먼트(line segment)로 저장하는 방법이다. 이 기법은 동일한 이동 객체에 속하는 라인 세그먼트들을 같은 단말 노드에 삽입하여 유지하는 궤적 보존 전략(trajectory preservation strategy)을 채택하고 있으며, 이로 인해 이동 객체의 궤적-기반 질의 처리시 매우 좋은 성능을 올릴 수 있다.

MOTB-tree[9]는 복합질의의 성능을 개선하기 위해 TB-tree를 변형시킨 색인이다. MOTB-tree는 단말 노드에 궤적별 클러스터링을 지원하고, 궤적의 연결성을 위해 단말 노드간을 링크로 연결하고 있다. 단말 노드의 부모 노드의 엔트리는 단말노드의 MBB를 두 개 이상 분할된 형태로 가질 수 있다는 것이 TB-tree와 다른 점이다.

Ⅲ. 제안하는 색인 구조

3.1 색인 구조의 설계

이동 객체의 궤적 정보를 색인하기 위해 기존에 제안되었던 TB-tree는 궤적-기반 질의의 검색 성능을 올리기 위해 동일한 이동 객체에 속하는 라인 세그먼트들을 반드시 같은 단말 노드에 삽입하도록 하는 궤적 보존 전략을 채택하고 있다. 이는 기존에 궤적-기반 질의 처리를 위해 제안되었던 다른 기법들에 비해 좋은 성능을 나타내도록 하지만, 반면에 이동 객체의 새로운 라인 세그먼트를 삽입할

때마다 궤적 보존 전략의 특성상 동일한 객체의 선행 라인 세그먼트가 삽입된 단말 노드를 찾기 위해 루트 노드에서부터 단말 노드까지 트리 전체를 순회해야 하기 때문에 삽입시 불필요한 많은 노드를 접근해야 하는 큰 단점을 지니고 있다. 특히 이동 객체의 라인 세그먼트들이 계속해서 증가함에 따라 색인의 크기가 커질수록 삽입될 객체의 선행 노드를 검색하는 데 많은 비용이 들게 된다. 따라서 본 연구에서는 이동 객체의 궤적을 구성하는 라인 세그먼트를 삽입할 때의 삽입 성능을 개선하고 저장 공간 측면과 궤적-기반 질의 처리의 효율성을 높이기 위해 기존의 TB-tree를 변형시킨 구조의 색인 기법을 제안한다. 제안하는 색인 구조는 그림 1과 같이 기존의 TB-tree의 구조와 전체적으로 유사하며, 단지 선행 노드를 직접적으로 접근할 수 있도록 하기 위해 각 이동 객체의 처음 세그먼트와 마지막 세그먼트가 저장된 단말 노드를 가리키는 메모리 포인터 정보와 노드를 저장하기 위한 디스크 페이지 번호를 유지하는 연결테이블(L-Table)이 별도로 존재한다. L-Table 구조는 헤더 정보와 단말 노드에 대한 포인터 정보와 디스크의 페이지 번호로 구성된다. L-Table 헤더는 total_rec과 id_len으로 구성되며, total_rec은 저장된 전체 이동 객체의 수를 나타내며, id_len은 이동 객체의 부가적인 세부 정보를 가리키는 식별자를 위한 식별자 길이 정보를 의미한다. L-Table의 각 레코드에는 oid, s_ptr, e_ptr, s_pid, e_pid 필드가 있으며, 각각은 이동 객체의 식별자, 이동 객체의 첫 번째 및 마지막 라인 세그먼트를 저장하고 있는 단말 노드의 메모리 포인터와 각각 디스크 상에서의 페이지 번호들을 나타낸다.

제안하는 색인 구조는 궤적에 대한 라인 세그먼트를 삽입할 때 기존의 TB-tree처럼 트리의 루트 노드에서부터 단말 노드까지 순회하지 않고 L-Table에서 해당 객체의 마지막 라인 세그먼트를 포함하고 있는 단말 노드를 가리키는 포인터(e_ptr)와 디스크 상에서의 페이지 번호(e_pid)를 이용하여 한 번에 직접적으로 단말 노드에 접근함으로써 삽입 성능의 효율성을 향상시킬 수 있다. 또한 이동 객체의 궤적 질의 처리시 해당 객체의 선행 궤적 정보를 페이지 번호를 통해 직접 디스크로부터 읽기 때문에 페이지 접근 횟수를 줄일 수 있다. 디스크 기반의 구조를 가지고 있어 새로운 이동 객체의 라인 세그먼트를 삽입시 해당 객체의 궤적을 바로 디스크에 반영해 메모리 상의 트리 구조와 디스크 상의 트리 구조 사이의 일관성을 유지한다.

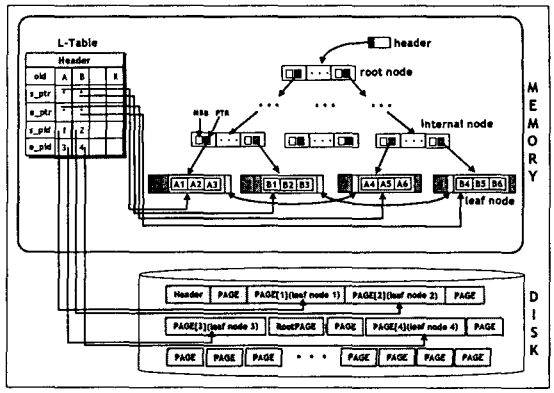

그림 1. 제안하는 색인 기법의 전체 구조
Fig. 1. Overall Architecture of Proposed Index Structure

3.2 색인 구조의 구현

제안하는 색인 구조는 그림 2에서와 같이 크게 6개의 클래스 즉, cETBTree 클래스, cETBTree Buffer 클래스, cETBTreeLinkTable 클래스, cETB DirNode 클래스, cETBLeafNode 클래스, cETB Entry 클래스로 구성되어 있다. 표 1은 각 클래스에 대한 기능 설명 및 세부 함수들을 나타낸다.

표 1. 각 클래스에 대한 기능 설명
Table. 1 Description of each Class

클래스명	기능 설명
cETBTree	트리의 생성 및 세그먼트 삽입과 질의 처리를 위한 함수를 포함하는 클래스
cETBTreeBuffer	버퍼링에 관련된 함수를 포함하는 클래스
cETBTreeLinkTable	L-Table에 대한 자료 구조 및 레코드 관리에 대한 함수를 포함하는 클래스
cETBDirNode	트리의 비단말 노드에 대한 자료 구조 및 MBR에 관련된 함수를 포함하는 클래스
cETBLeafNode	트리의 단말 노드에 대한 자료 구조 및 이동 객체의 세그먼트를 저장하기 위한 함수를 포함하는 클래스
cETBEntry	트리의 비단말 노드를 구성하는 엔트리들을 위한 함수를 포함하는 클래스

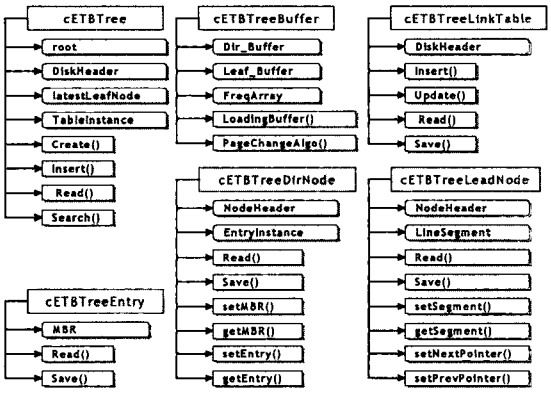

그림 2. 클래스 구성도
Fig. 2. Class Configuration

IV. 실험 환경 및 성능 평가

본 논문에서는 제안하는 색인 구조를 구현하여 삽입 성능과 검색 성능에 대해서 성능 평가를 수행하였으며, 특히 검색 성능의 경우에는 궤적-기반 질의에 대해서 성능을 평가하였다. 성능 평가 대상으로는 R*-tree, TB-tree, 그리고 제안하는 색인 구조를 선정했다. 실험 환경은 펜티엄-IV 2.8GHz CPU와 메인 메모리 1GB, 윈도우 2000에서 C++ 언어를 이용하여 구현 및 실험하였다. R*-tree, TB-tree, 제안하는 색인 구조 모두 디스크 페이지의 크기는 4KB이다. 성능 평가를 위해 사용한 실험 데이터는 GSTD[10]와 Runtime21[11]을 이용하여 생성한 데이타로서 특히 Runtime21은 로드맵(loadmap)에 따라 이동하는 자동차들과 같이 지리 네트워크 정보(TIGER/Line)를 이용해 리얼 데이타와 거의 유사한 데이타 집합을 생성할 수 있는 도구이다. GSTD는 이동 객체를 위한 랜덤 데이타를 생성하기 위해 가장 많이 이용하는 데이타 생성 도구로서 다양한 형태의 데이타 분포도를 고려하여 생성할 수 있는 장점을 가지고 있다. 따라서 본 논문에서는 랜덤 데이타를 위해서는 표 2와 같이 데이타 분포도를 고려하여 GSTD 알고리즘을 이용해 각 분포도에 따라 객체의 수가 1,000(1K)에서부터 10,000(10K)까지 100초 동안 이동한 데이터를 toriod 좌표계를 사용해 생성하였다. 아울러 리얼 데이타를 위해서는 Runtime21을 이용하여 표 3과 같이 생성하였다.

표 2. GSTD를 이용한 랜덤 데이타 집합
Table. 2 Random Data Set using GSTD

	초기 분포도	이동 방향성
U-R dataset	Uniform	Random
U-D dataset	Uniform	Directed
G-R dataset	Gaussian	Random
G-D dataset	Gaussian	Directed

표 3. Runtime21을 이용한 리얼 데이타 집합
Table. 3 Real Data Set using Runtime21

	# of moving object	time
50,000(50K)	500	100 sec.
200,000(200K)	2,000	100 sec.
500,000(500K)	5,000	100 sec.

본 논문에서는 다양한 형태의 성능 평가를 위해 버퍼의 크기에 따른 성능 평가와 데이타의 분포도에 따른 성능 평가로 나누어 실험을 수행하였다.

먼저, 버퍼의 크기에 따른 성능 평가는 Runtime21의 리얼 데이타 집합을 이용했으며, 제안하는 색인 구조만을 평가 대상으로 하였다. 그림 3은 버퍼 크기의 변화에 따른 삽입 성능에 대한 디스크 페이지 접근 횟수(I/O)를 측정한 결과로 버퍼 크기가 클수록 삽입 성능이 향상됨을 알 수 있다. 특히 데이타 집합의 크기가 커질수록 버퍼 크기가 페이지 접근 횟수에 큰 영향을 미치는 것을 확인할 수 있다. 데이타의 크기가 500K일 때 약 2배 정도의 페이지 접근 횟수의 감소가 일어남을 알 수 있다.

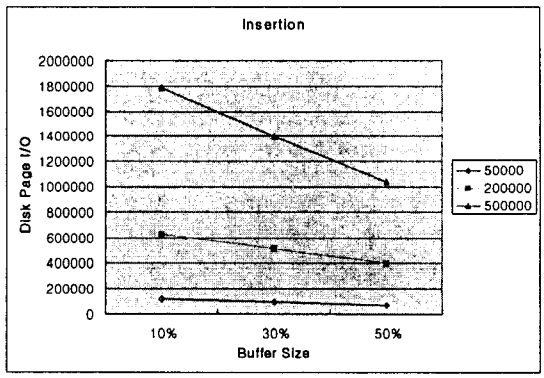

그림 3. 버퍼 크기에 따른 삽입 실험 결과
Fig. 3 Experimental Result for Insertion with Buffer Size

그림 4는 버퍼 크기의 변화에 따른 궤적-기반 질의에 대한 검색 성능에 대해 디스크 페이지 접근 횟수(I/O)를 측정한 결과를 보여준다. 버퍼 크기가 10%일 때는 500K일 경우 가장 큰 페이지 접근 횟수를 보이다가 버퍼 크기가 50%로 감소할 때 가장 적은 수의 페이지 접근 횟수를 나타내고 있다. 이는 50K의 경우와 같이 데이타의 크기가 적은 경우에는 버퍼 크기에 큰 영향을 받지 않지만, 데이타의 크기가 큰 경우 즉, 500K의 경우 버퍼 크기가 작은 경우에는 색인의 일부만이 메모리에 상주되기 때문에 페이지 접근 횟수가 잦아지는 반면에 버퍼 크기가 큰 경우에 색인의 많은 부분이 메모리에 상주될 수 있기 때문에 페이지 접근 횟수가 적어 검색 성능이 향상됨을 알 수 있다.

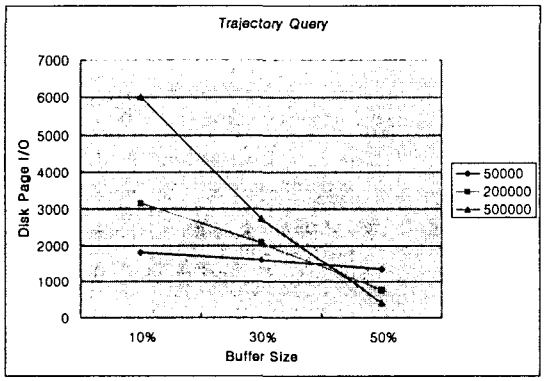

그림 4. 버퍼 크기에 따른 궤적 질의 실험 결과
Fig. 4 Experimental Result for Trajectory-based Query with Buffer Size

그림 5는 UD(Uniform-Directed) dataset을 이용하여 궤적-기반 질의에 대한 페이지 접근 횟수를 측정한 결과이다. 궤적-기반 질의는 1000개의 이동 객체의 궤적을 검색하도록 한 후 평균을 구한 것이다. 대체적으로 R*-tree는 궤적-기반 질의에 대한 특별한 정책이 없기 때문에, 이동 객체의 수가 증가할수록 페이지 접근 횟수가 점진적으로 증가함을 알 수 있으며, TB-tree나 제안하는 색인 구조보다 훨씬 성능이 떨어짐을 알 수 있다. 이는 원래 R*-tree가 MBR을 이용하여 이동 객체의 영역 질의(range query)에 대한 성능 향상을 위해 제안되었던 색인 구조이기 때문이다.

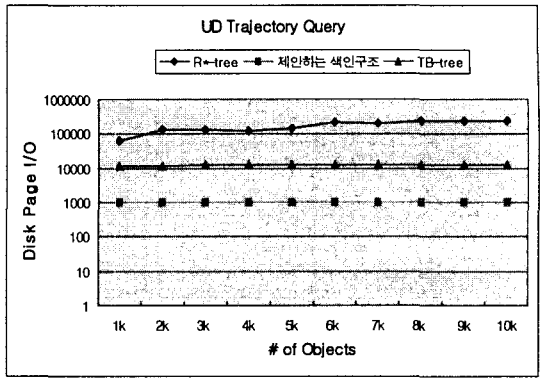

그림 5. UD dataset일 때의 궤적 질의 실험 결과
Fig. 5 Experimental Result for Trajectory-based Query with UD dataset

TB-tree와 제안하는 색인 구조는 객체 수의 변화에 큰 영향을 받지 않으며, 제안하는 색인 구조는 TB-트리보다 약 10배 정도의 성능이 향상됨을 알 수 있다. 이는 제안하는 색인 구조의 경우 L-Table을 이용하여 원하는 궤적의 첫번째 세그먼트를 거의 한 번의 노드 접근으로 검색할 수 있기 때문이다. 아울러 궤적-기반 질의의 경우 제안하는 색인 구조나 TB-tree 모두 데이타가 어떤 형태의 분포도를 가지고 있느냐와는 큰 상관관계가 없음을 알 수 있다.

그림 6은 UR(Uniform-Random) dataset을 이용하여 궤적-기반 질의에 대한 페이지 접근 횟수를 측정한 결과를 보이고 있다. R*-tree의 경우 UD dataset보다는 다소 나은 성능을 보이나 큰 차이는 없음을 알 수 있으며, TB-tree와 제안하는 색인 구조는 거의 비슷한 결과를 나타내고 있다.

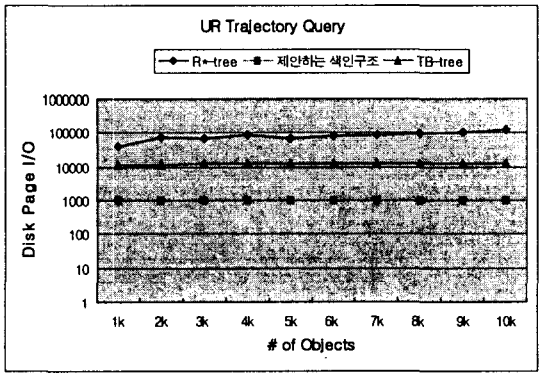

그림 6. UR dataset일 때의 궤적 질의 실험 결과
Fig. 6 Experimental Result for Trajectory-based Query with UR dataset

그림 7은 GD(Gussian-Directed) dataset을 이용하여 궤적-기반 질의에 대한 페이지 접근 횟수를 측정한 결과를 보이고 있다. R*-tree, TB-tree, 제안하는 색인 구조 모두 UD dataset과 거의 비슷한 결과를 보인다.

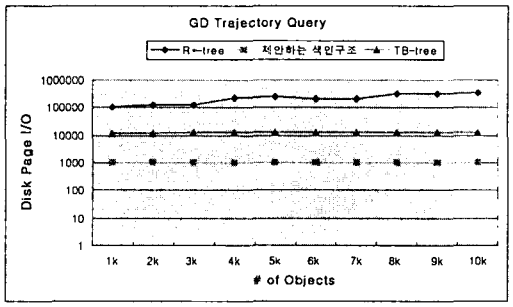

그림 7. GD dataset일 때의 궤적 질의 실험 결과
Fig. 7 Experimental Result for Trajectory-based Query with GD dataset

그림 8은 GR(Gussian-Random) dataset을 이용하여 궤적 질의에 대한 페이지 접근 횟수를 측정한 결과를 보이고 있다. R*-tree의 경우만 약간의 차이를 보이고 있으며, TB-tree와 제안하는 색인 구조는 거의 차이가 없음을 알 수 있다.

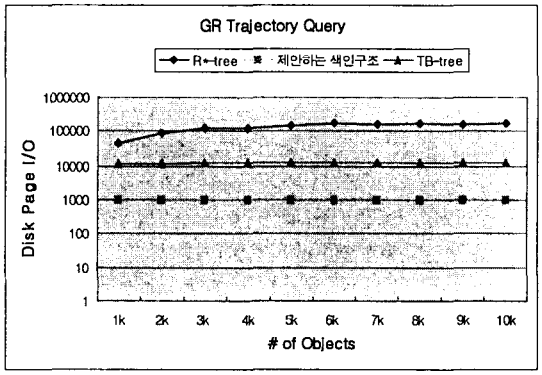

그림 8. GR dataset일 때의 궤적 질의 실험 결과
Fig. 8 Experimental Result for Trajectory-based Query with GR dataset

V. 결 론

본 연구에서는 모바일 응용에서 이동 객체의 궤적-기반 질의에 대한 성능을 향상시킬 수 있는 색인 구조를 설계 및 구현하였다. 제안하는 색인 구조는 선행 노드를 직접적으로 접근하기 위해 이동 객체의 처음 세그먼트와 마지막 세그먼트가 저장된 단말 노드를 가리키는 포인터 정보와 더불어 디스크에서의 페이지를 가리키는 페이지 번호를 별도의 연결테이블(L-Table)에 같이 유지하도록 설계하였다. 따라서 저장의 경우 동일한 이동 객체의 선행노드를 빨리 검색할 수 있고, 궤적-기반 질의 처리의 경우 직접 디스크에 접근해 해당 객체의 궤적들을 검색함으로써 검색 성능을 향상시킬 수 있었다. 아울러 색인의 크기가 커질수록 시스템의 가용메모리의 제한으로 인해 성능에 영향을 미칠 수 있기 때문에 일정한 버퍼를 유지함으로써 해당 버퍼를 교체해 나가는 버퍼링 정책을 사용해 가용 메모리를 효율적으로 사용할 수 있도록 하였다. 마지막으로, GSTD 알고리즘과 Runtime21을 이용하여 랜덤 데이타와 리얼 데이타를 사용하여 버퍼 크기의 변화에 따른 실험과 다양한 형태의 데이타 분포도를 고려한 실험을 수행하였다. 성능 평가 결과, 삽입 성능은 기존의 TB-tree에 비해 약 6배 정도, 검색 성능은 궤적-기반 질의에 대해서 약 10배 정도의 성능 향상을 올릴 수 있었다.

참고문헌

[1] S. Saltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez, "Indexing the Positions of Continuously Moving Objecs", ACM SIGMOD on Management of data, pp. 331-342, 2000.

[2] Y. Tao, D. Papadias, "MV3R-Tree: A Spatio-Temporal Access Method for Timestamp and Interval Queries", International Conference on VLDB, pp. 431-440, 2001.

[3] O. Wolfson, B. Xu, S. Chamberlaina, and L. Jiang, "Moving objects databases: Issues and solutions", In Proc. of 10th Int'l. Conf. on Scientific and Statistical Database Management, pp. 111-122, 1998.

[4] D. Kwon, S. Lee, and S. Lee, "Indexing the Current Positions of Moving Objects Using the Lazy Update R-Tree", Conference on

Mobile Data Management, pp. 113-120, 2002.
[5] D. Pfoser, Y. Theodoridis, and C. S. Jensen, "Indexing Trajectories in Query Processing for Moving Objects", Chorochronos Technical Report, CH-99-3, 1999.
[6] N. Beckman, H. P. Kriegel, "The R*-tree:An Efficient and Robust Access Method for Points and Rectangles", In Proc. of ACM SIGMOD, pp. 302-331, 1990.
[7] 조형주, 정진완, "시공간 질의를 위한 인덱싱 기법", 한국정보과학회 한국데이타베이스 학술대회 논문집, Vol. 18, No. 2, pp. 93-100, 2002.
[8] D. Pfoser, C. S. Jensen, and Y. Theodoridis, "Novel Approaches in Query Processing for Moving Objects", International Conference on VLDB, pp. 395-406, 2000.
[9] 임덕성, 전봉기, 홍봉희, "이동체를 위한 궤적 색인의 분할 정책", 개방형 GIS 학술회의 논문집, Vol. 5, No. 2, pp.173-176, 2002.
[10] Y. Theodoridis, J. R. O. Silva, and M. A. Nascimento, "On the generation of spatiotemporal datasets", International Symposium on Spatial Datasets, pp. 147-164, 1999.
[11] T. Brinkhoff, "A Framework for Generating Network-Based Moving Objects", GeoInformatica, Vol. 6, No. 2, pp 153-180, 2002.

저자소개

심춘보(Choon-Bo Shim)

1996년 전북대학교 컴퓨터공학과(공학사)
1998년 전북대학교 컴퓨터공학과(공학석사)
2003년 전북대학교 컴퓨터공학과(공학박사)
2004~현재 부산가톨릭대학교 컴퓨터정보공학부 전임강사
※관심분야 : 멀티미디어 데이타베이스, 멀티미디어 정보검색, Mobile & Moving DB, LBS 등

주재흠(Jae-Heum Joo)

1988년 부산대학교 전자공학과(공학사)
1990년 부산대학교 전자공학과(공학석사)
2000년 부산대학교 전자공학과(공학박사)
2000~현재 부산가톨릭대학교 컴퓨터정보공학부 부교수
※관심분야 : 영상인식, 영상처리, 컴퓨터비전 등