이동객체 데이터베이스를 위한 통합 인덱스

한종형, 이준우, 나연묵
단국대학교 전자컴퓨터공학과
e-mail : {jhhan, jwlee}@dblab.dankook.ac.kr, ymnah@dku.edu

A Unified Index for Moving Object Database

Jonghyeong Han, Joonwoo Lee, Yunmook Nah
Dept of Electronics and Computer Engineering, Dankook University

요 약

GPS 및 PDA 등의 발달과 보급으로 인해 위치기반서비스(LBS), 차량항법장치(CNS), 지리정보시스템(GIS)등 공간 데이터를 이용하는 응용프로그램이 급격하게 발달되었다. 응용프로그램의 발달과 더불어 이동객체의 양이 방대해짐에 따라 객체에 대한 효율적 처리가 요구된다. 이동 객체가 시간의 흐름에 따라 연속적으로 발생되는 위치 정보를 신속하게 관리하기 위한 통합된 인덱스 구조를 제안한다. 제안하는 인덱스구조는 이동 객체의 현재, 과거, 미래질의를 처리 할 수 있도록 3 가지의 인덱스를 통합 하였으며, 또한 제안된 인덱스구조에 저장 관리자를 함께 둠으로써 이동 객체정보를 관리 할 수 있다.

1. 서론

최근 GPS 및 PDA 의 사용이 증가하고 위치기반 서비스(LBS), 차량항법장치(CNS), 지리정보시스템(GIS) 등 공간 데이터를 다루는 응용프로그램들이 급속하게 발전하고 보급 되어 왔다[1]. 이러한 응용 프로그램들의 활용증대와 이동객체의 처리를 위해서 다양한 인덱스구조를 필요로 한다. 시간의 흐름에 따라 객체가 이동 하면서 위치를 변경하는 특징을 지닌 이동 객체를 효율적으로 저장하고 관리 할 수 있는 인덱스 기술이 요구된다. 이동 객체의 위치 정보를 인덱스 하기 위해서 현재 위치를 검색하고 관리하기 위한 인덱스들로 R-트리, Hash, Quad-트리가 있으며, 과거궤적 검색을 위한 인덱스들로 3DR-트리, HR-트리, TB-트리 그리고 미래의 정보를 효율적으로 처리 하기 위해 B*-트리, PSI-트리, TPR-트리등이 있다. 현재 인덱스들은 효율성과 성능 면에서 계속적인 변화를 거듭하고 있으며 새로운 알고리즘이 계속 제안 되고 있다.

* 본 논문에서는 R-트리를 기반으로 파생된 트리(R*-트리[2,3,4], MVR-트리[5,6], TPR-트리[7])를 이용하여 이동 객체 정보에 대해 다양한 처리가 가능하도록 인덱스구조를 통합한 형태를 제안한다. 시공간 객체에 대한 정보를 저장하고 관리를 가능케 함으로써 저장된 정보에 대해서 이용성의 증대 즉 현재 과거 그리고 미래에 대한 질의를 가능케 한다.

본 논문의 구성은 다음과 같다. 2 장에서는 본 논문에서 사용되는 인덱스에 대한 관련연구를 기술하였고, 3 장에서는 제안된 인덱스구조와 구성 인덱스의 분할을 제안하고, 4 장에서는 인덱스와 관련된 여러 가지 실험을 하였고, 마지막으로 5 장에는 결론과 향후 연구 방향에 대해 기술한다.

2. 관련연구

기존의 공간 인덱스는 크게 두 가지로 분류 된다. 공간 분할 방법과 데이터 분할 방법이다. 이동 객체는 시간에 따라 계속적인 변화를 나타내기 때문에 관리해야 할 이동 객체 수가 많아진다. 그래서 이동 객체는 관리가 쉽지 않다. 객체의 위치 정보를 관리하기 위해 기존에 제안 된 형태의 인덱스 구조의 변형이 많이 제시 되었다. HR-트리의 변형인 HR*-트리, 3DR-트리와 MVR-트리가 결합된 MV3R-트리등 효율성을 높이기 위해 형태성, 기능성의 변화를 거듭하고 있다.

본 논문은 변형 트리를 이용하지 않고 제안된 대표적인 트리를 이용한다. 현재의 정보를 효율적으로 처리를 위한 접근 기법으로 R-트리, 과거의 정보 처리에 효율적인 접근 기법으로 MVR-트리 그리고 가까운 미

* 본 연구는 정보통신부 및 정보통신 연구 진흥원의 대학 IT연구센터 육성지원사업의 연구 결과로 수행 되었음

래를 예측이 가능한 접근 기법으로 TPR-트리가 있다.

2.1 R-트리

비 정형화된 이동 객체의 CAD 와 지리 데이터 응용 에서 필요한 공간 데이터를 다루기 위해 이동 객체에 대해서 검색이 가능한 인덱스가 필요하다. R-트리[2,3,4]는 B-트리와 유사한 인덱스 레코드들로 구성되는 높이 균형 트리고, 노드의 삽입, 삭제, 질의연산을 혼합하여 사용할 수 있다. 트리의 높이는 (logmN)-1 으로 단말노드가 객체에 대한 포인터를 포함하고 있는 형태이다.

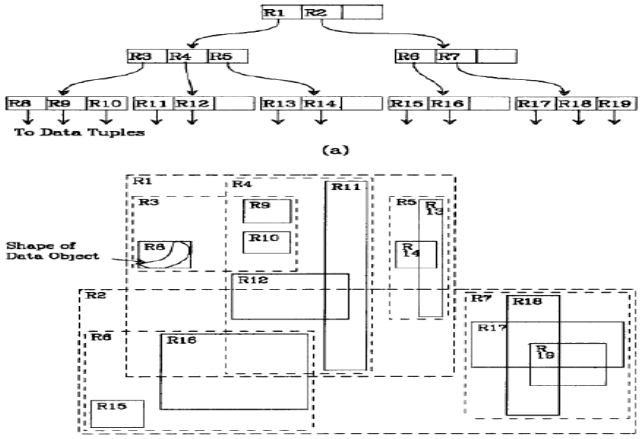


그림 1. R-트리 예

그림 1은 R-트리의 예로 해당레벨의 노드들에 관련된 모든 사각형의 전체 영역을 정의 하고, 두 개 이상의 노드들이 공통적으로 속하는 영역을 재정의 하여 검색을 용이하게 한다.

2.2 MVR-트리

MVR-트리[5,6]는 CAD 그림을 R-트리에 적용하고자 제안 됐다. CAD 체계[6]에 많은 그림이 동시에 존재하며, 크기가 모두 상이한 형태를 보인다. 이들을 저장하기 위하여 일반적으로 R-트리를 이용한다. 그림 2에 보이는 것과 같이 R-트리를 사용하여 저장한다.

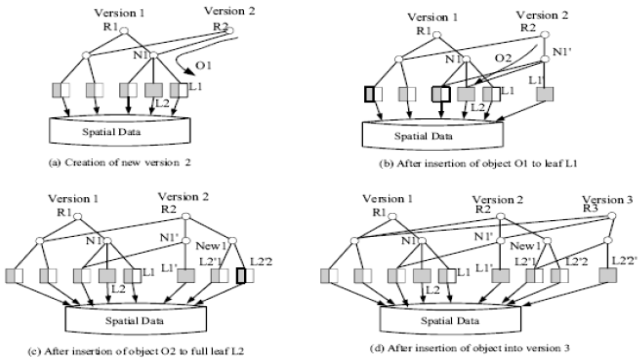
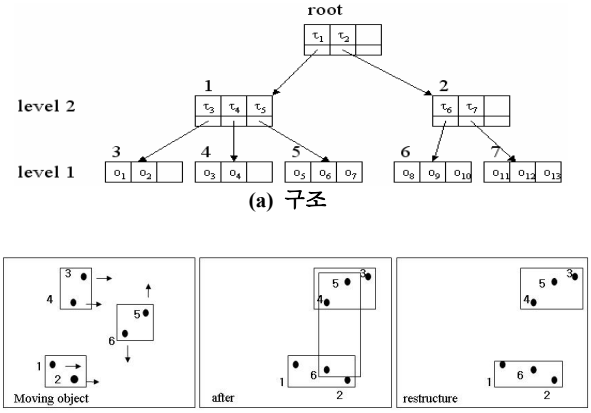


그림 2. MVR-트리 삽입 예

같은 노드를 상호 공유하는 형태를 나타내는 MVR-트리는 timestamp 질의를 처리할 때 효율성에서 좋은 성과를 나타낸다. MVR-트리는 R-트리를 다중으로 사용한 형태이다.

2.3 TPR-트리

TPR-트리[7]는 이동 객체의 현재 및 미래 위치를 검색하기 위한 기존의 R-트리 기반 인덱스 구조의 변형이다. TPR-트리는 단말 노드에 객체의 위치를 저장하고 중간 노드에는 자식 노드에 포함된 영역과 속도 정보 각 차원의 상한 값과 상한 값을 기준으로 자식 노드에 포함된 이동 객체의 속도를 포함하는 정보를 저장 하고 있다. 영역 정보와 속도 정보를 이용하여 객체의 진행 방향성을 예측할 수 있다.



(a) 구조

(b) 이동 객체의 재구성

그림 3. TPR-트리 예

그림 3 (a)의 level 1 에는 객체에 대한 정보를 저장하고 level 2 에 객체의 영역과 속도정보를 저장 하고 있다. 그림 3 (b)는 이동 객체의 속도를 기준으로 현재 시간(그림 3 (b) 좌측)에서 일정시간 이후에 이동된(그림 3 (b) 중간) 객체를 예측하고 노드를 재구성하는 과정이다.

3. 인덱스구성

통합 인덱스 구조는 그림 4 에 보이는 것과 같이 하나의 라이브러리에 디스크관리자, 메모리관리자가 있고, 저장 관리자와 인덱스 구조를 담당하는 인덱스 부분이 있으며 인덱스구성은 R-트리, MVR-트리와 TPR-트리등 현재, 과거, 미래에 대한 정보를 관리할 있도록 구성 되었다.

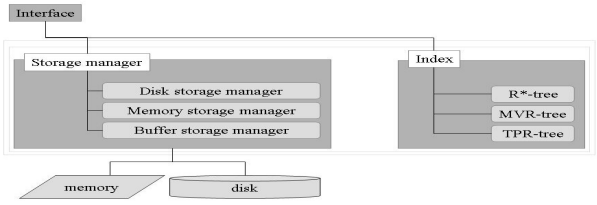


그림 4. 시공간 인덱스

그림 4 에 시공간 인덱스는 캘리포니아 리버사이드 대학교의 Marios Hadjieleftherio 가 제안 했으며[8], 다음과 같은 도구들을 제공한다.

- The core spatialindex utilities
- The storagemanager file
- The spatial-index interfaces
- The r-Tree index

- The mvr-Tree index
- The tpr-Tree index

위의 도구들은 일반적인 메인 메모리와 디스크 기반 저장 관리자를 이용하여 객체의 정보를 저장한다. 인덱스는 내부 메모리와 외부 메모리에 생성도 지원하며, 클러스터 형태로도 지원된다. 그리고 트리에 대한 일반적인 질의(Range, Point Location, Nearest Neighbor, K-nearest Neighbor, Parametric)를 할 수 있으며, 인터페이스를 통하여 객체의 삽입, 삭제, 수정이 가능하다. R-트리의 객체 삽입은 대량 형태의 삽입도 할 수 있다. 또한 Page size, node capacity, Minimum fan-out, splitting algorithm 등 인덱스구조의 특성을 용도별로 변경 할 수 있다.

저장 관리자는 My-SQL 을 이용하였고, 디스크 관리와 메모리 관리 그리고 버퍼에 대한 저장 관리를 모두 수행함으로써 객체의 정보관리를 할 수 있다. 저장된 정보에 대한 현재 및 미래질의 또한 과거질의 처리가 가능하다.

하지만 그림 4 의 시공간 인덱스 구조는 여러 형태의 시공간 질의를 가능하게 하지만 객체의 양이 방대해질 경우 객체를 단일 노드에서 처리하여 시스템의 성능저하를 발생시켜 그 인덱스 구조의 효율성을 점진적으로 저하시킨다.

본 논문에서는 시공간 인덱스 구조의 성능 저하를 막고 처리비용을 분산시켜 그 기능을 유지 시키되 성능 저하를 줄이는 구조를 그림 5 의 구조로 제안하였다.

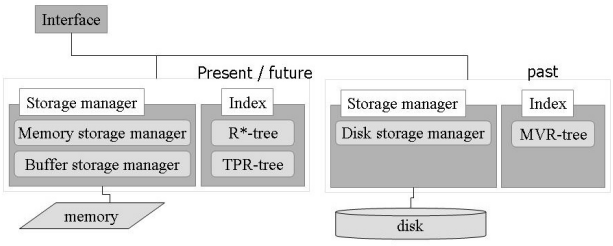


그림 5. 인덱스 분할 구조

인덱스 구조 분할의 기준은 정보의 양과 정보의 이용 빈도에 따라 분할을 하였다. 그림 5 의 오른쪽의 과거 관리자는 MVR-트리와 디스크 관리자로 분할을 하였고, 그림 5 의 왼쪽에 현재와 미래 관리자는 R-트리, TPR-트리 그리고 메모리, 버퍼 관리자로 분할을 하였다.

이동 객체의 과거정보는 질의 처리 과정이 적지만 저장 관리해야 하는 정보의 양이 방대하다. 반면에 객체의 현재 정보는 정보의 양이 과거정보 보다 상대적으로 적으나 질의 처리 부분이 많으며, 미래 질의는 현재정보를 기준으로 질의 처리가 되기 때문에 R-트리와 TPR-트리는 분할 되지 않았다. 질의 처리 및 객체의 저장 관리가 객체 수가 증가될 때 단일 노드에 집중된 형태 보다 분산된 형태로 처리 할 때, 처리비용을 분산하게 되어 효율적이다.

4. 실험

본 논문에서는 기존 인덱스의 질의에 대한 성능비교[2,3,4,5,7]는 제외 하였다. R-트리, MVR-트리 TPR-트리에 대한 질의 성능은 이미 많은 논문에서 그 성능이 입증 되었기 때문이다.

본 논문에서는 제안한 시스템을 기반으로 각 각의 인덱스에 대해 삽입, 노드탐색, 트리 구축에 대한 프로세서 처리 시간 및 시스템 I/O 에 대한 비교를 하였다. 인덱스의 비교를 위해 데이터 생성은 이동 객체 생성기를 이용하였다. 생성된 객체내의 속성을 해당 인덱스 구조에 맞게 실험을 하였다.

실험에 사용한 컴퓨터의 환경은 CPU 2.6Mhz, RAM 1G 이고, 운영체제는 Linux Pedora core 4.0 이였으며, 제안한 시스템은 C 언어로 구현되었다.

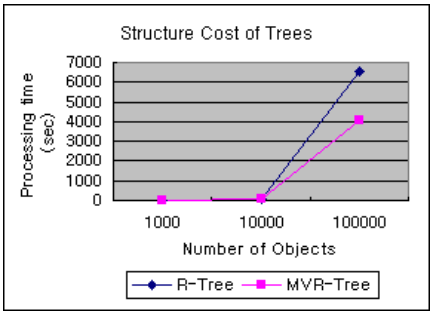
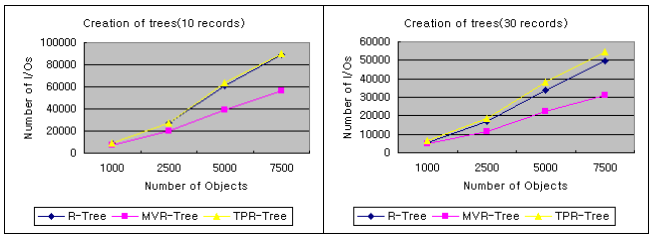


그림 6. 트리 생성

그림 6 은 각 트리가 생성시 걸리는 시간을 측정하였다. 이동 객체수가 십만 개 이상일 때 트리의 생성 시간이 많이 걸리는 것을 알 수 있다. 한번 생성된 트리에 대한 추가 삽입과 수정은 노드 탐색 시간이 짧아 빠른 삽입과 수정이 가능하다.

또한 트리 생성시 73~89%의 노드 사용률을 보이며, 트리의 이용 효율성이 안정적이다.

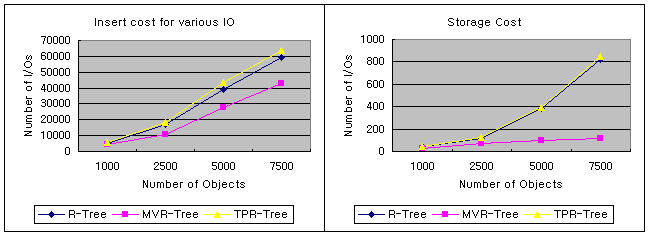


(a) **(b)**

그림 7. 객체 삽입

그림 7 의 (a)와 (b)의 실험은 삽입되는 데이터에 대해 레코드 셋을 10 개의 레코드와 30 개의 레코드로 구분해서 삽입하여 I/O 를 측정 하였다. 레코드 삽입 시 레코드 셋을 30 으로 삽입할 때 I/O 가 50~60%정도 감소한 것을 실험을 통해 살펴볼 수 있었다.

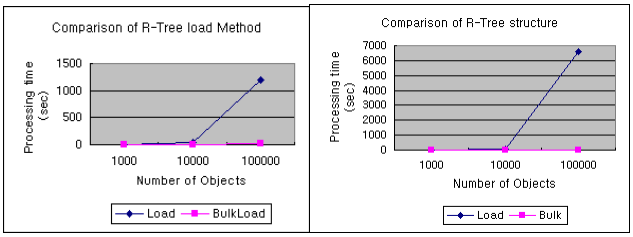
그림 8 의 (a)는 생성된 트리에 이동 객체가 일반적인 형태로 추가 삽입되는 I/O 를 측정한 실험결과이고, (b)는 객체 저장에 대한 I/O 를 측정한 실험이다.



(a) (b)

그림 8. 추가삽입과 저장

객체의 I/O 측정에서 보편적으로 R-트리와 TPR-트리는 비슷한 I/O 수를 보이고 있으며, 수치 증가율은 객체의 수와 비례하여 대각선 상승 곡선 형태를 띠고 있다. MVR-트리의 I/O 수는 앞에서 언급한 2 개의 트리보다 작은 수치이다.



(a) (b)

그림 9. R-트리의 대량삽입과 일반 삽입

그림 9 (a)는 R-트리의 객체삽입 형태별로 대량 형태와 일반적인 객체 삽입 비교이고, (b)는 삽입 형태에 따른 트리의 생성시간 측정이다. 대량형태로 삽입 시 처리 시간이 많이 단축 되었고, 객체에 대해 7~11 배의 증가율을 보였다. 대량형태의 십 만개 객체 삽입에 대한 트리 생성 시간은 약 15 초를 소요했다.

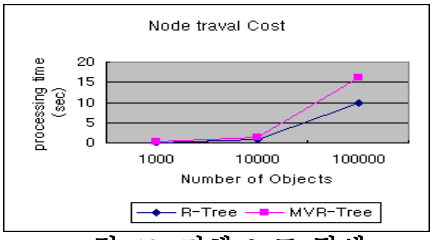


그림 10. 전체 노드 탐색

그림 10 은 생성된 트리의 전체노드 탐색에 대한 기간 측정이다. 전체 노드 탐색 시간은 그림 6 의 트리구조 생성 시간과는 대조적인 특성을 보인다. 트리의 특성상 접근 방법이 다르기 때문에 이와 같은 특성을 보인다.

5. 결론과 향후 연구

본 논문에서는 분산환경에서 사용 가능한 인덱스 구조를 제안 하였다. 이동 객체의 특성상 계속적으로 발생되어 객체의 양이 방대하여 단일 노드에서 이동 객체의 처리를 모두 수행 할 경우 점진적으로 시스템의 성능저하가 예상되기 때문에 인덱스 구조와 저장 형태를 구분하여 각각의 인덱스와 저장 관리자를 분할하였다. 분할된 인덱스 구조가 이동 객체의 정보(현재, 과거, 미래)를 모두 저장 관리함으로써 현재, 과거, 미래에 대한 질의가 모두 가능하다.

본 논문에서 제안한 인덱스 구조는 기존에 제안된 대표적인 인덱스들로 구성하였으며, 이러한 인덱스들이 다른 인덱스들 보다 좋은 성능을 발휘 하는 것은 아니다. 하지만 인덱스 구조의 통합을 통해 복잡한 여러 형태의 질의를 모두 수행할 수 있고, 분할된 구조의 분산효과를 통해 시스템의 성능 향상을 기대 할 수 있는 것이 본 논문에서 제안한 인덱스 구조의 장점이다.

향후 연구로 본 논문에서 제안했던 인덱스 성능과 효율성을 높이기 위한 연구가 필요하다. 변형된 인덱스 적용 그리고 이동 객체 질의 범위가 시공간의 제약 범위를 줄이는 것이다. 그 예로 TPR-트리를 TPR*-트리로, MVR-트리를 MV3R-트리나 MVR*-트리로 변경하여 개선을 하는 것이다. 더불어 분산 환경에 질의 처리 및 이동 객체의 저장 관리에 효율성을 높이는 연구가 지속되어야 할 것이다.

참고문헌

[1] Zhang, j., Zhu, m., Papadias, D., Tao, Y., Lee, D. “Location-Based Spatial Queries. To appear in proceedings of ACM conference on Managerment of Data,” SIGMOD, pp 467-478, June 9-12, 2003

[2] A. Guttman, ”R-Trees: A Dynamic Index Structure for Spatial Searching,” ACM SIGMOD, 1984.

[3] King Lum Cheung, Ada Wai-chee Fu. ”Enhanced nearest Neighbor search on the R-Tree,” SIGMOD, 1998

[4] N. Beckmann, H. Kriegel, R.Schneider, and B.Seeger. “The R*-Tree : An Efficient and Robust Access Method for Points and Rectangles,” ACM SIGMOD, 1990.

[5] Y. Tao, Y, and Papadlas, D. “MV3R-Tree : A Spatiotemporal Access Method for Timestamp and Interval Queries,” VLDB, pp431-440, 2001.

[6] Y. Nakamura and H. Dekihara “Spatial data structures for version management of engineering drawings in cad database,” ICIAP, pp219, 2003.

[7] S. Saltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez “Indexing the Positions of continuously Moving Objects,” SIGMOD, pp331-342, 2000.

[8] E. H. Marios Hadjieleftheriou and V. J. Tsotras “Sail: A spatial index library for efficient application integration,”

[9] Yunmook Nah, K.H. (Kane) Kim, Taehyung Wang, Moon Hae Kim, Jonghoon Lee, Young Kyu Yang, "GALIS: A Cluster-based Scalable Architecture for Location-Based Service Systems," KISS SIGDB, pp66-80. 2002.

[10] Yunmook Nah, K.H.(Kane) Kim, Taehyung Wang, Moon Hae Kim, Jonghoon Lee, Young Kyu Yang, "A Cluster-based TMO-structured Scalable Approach for Location Information Systems," IEEE WORDS 2003F, 2003.