효율적인 디스크 관리와 질의 처리 시간 향상을 위한 개선된 QSF-Tree

강소영, 양성봉
연세대학교 컴퓨터과학과

An Improved QSF-Tree for Efficient Disk Utilization and Query Processing Time

So-Young Kang, Sung-Bong Yang
Dept. of Computer Science, Yonsei Univ.

CAD나 Cartography 등 여러분야에서 많이 사용되고 있는 거대한 다차원 공간 데이터를 효율적으로 구축하고 관리하는 일은 매우 중요하다. 본 논문에서는 기존에 공간데이터를 구축하던 대표적인 R-Tree계열의 방법을 탈피하여 새로운 데이터 구조와 공간상의 데이터간의 관계를 효율적으로 표현할 수 있는 QSF-Tree를 바탕으로 하여 QSF-Tree의 disk utilization을 높일 수 있는 방법을 제안하고 새로운 split 방법을 이용하여 실 공간에서 가까이에 있는 object들을 되도록 하나로 묶어 search 속도를 향상시키고 disk access 숫자를 줄이는 방법을 제안한다.
또한 조건을 만족하는 MBR들을 찾은 후 이에 해당하는 데이터들의 실제값을 비교하는데 많은 시간이 소요된다는 사실에 의거하여 MBR들간의 topological relation을 바탕으로 실제 object의 값을 비교하지 않아도 실제 데이터간의 topological relation을 알 수 있는 MBR들의 위치관계를 이용하여 질의 처리 속도를 향상시켰다.

1. 서론

이전에 존재했던 Database Management System(DBMS)은 다차원 공간 속의 데이터들, 즉 사각형, 다각형 또는 다차원 공간 속의 포인트들을 다루는데 있어 효율적이지 못하였다. 하지만 이러한 다차원 데이터를 사용하는 여러 application들 예를 들어 지리적 질의에 대해 신속하게 응답하여야 하는 지도 Cartography나 여러 기계나 건축 도면을 설계하는 Computer-Aided Design(CAD)등이 나타나면서 이러한 다차원 데이터들을 신속히 저장하고 찾고 불러들이기 위한 방법이 절실히 필요로 되었다. 이와 관련하여 공간 데이터를 다루는 방법으로 대표적으로 Spatial Access Method(SAM)가 시도되었는데 이는 다차원 공간 속의 복잡한 다각형, 원형의 데이터 등을 Minimal Bounding Rectangle(MBR)로 표현하고 이들을 Rectangle로 묶어 page단위별로 이를 disk에 저장하고 관리하는 방법이다[5]. 이에 대표적인 R-Tree계열은 차원이 높아질수록 MBR들의 overlap과 margin으로 인하여 serach할 때 많은 disk access를 요구하므로 고차원 data를 위한 새로운 방법이 모색되었다.

2. 관련연구

공간데이터 구조를 위한 대표적인 구조인 R-TREE는 공간상의 POINT 뿐만이 아니라 spatial data를 지원하는 인덱스 구조이다. 데이터를 저장하기 위한 point transformation을 필요로하지 않으며 다른 기존의 tree보다 나은 공간 clustering을 지원한다.

R-tree는 다차원에서의 확장된 B-Tree이며 데이터 구조는 high- balanced tree로 데이터들은 leaf nodes에 저장된다.

R-Tree의 단점은 실제 데이터의 MBR을 포함하는 rectangle을 구성할 때 이들 간의 겹쳐지는 부분과 실제 데이터를 포함할 때 남는 여백이 page의 낭비를 유발하고 overlap되는 rectangle사이에 있는 data search시 여러 가지 방법의 검색경로가 나올 수 있다는 것이다[2]. R*-Tree는 overlap을 없애기 위해 데이터를 clipping 하기 때문에 데이터가 여러 rectangle에 나누어 저장될 수 있어 여러 탐색경로가 나오게 되므로 data search 시 R-Tree보다도 query 처리 성능이 떨어질 수 있다[3].

R*-Tree는 복잡한 split 방법을 통해 data의 overlap을 최소화 하고 data를 포함하는 MBR의 dead space와 margin을 최소화하는데 그 중점을 두었다. 무엇보다도 reinsert라는 factor를 두어 insert시 효율적으로 data를 저장할 수 있어 page utilization을 높였으며 현재 spatial data를 다루는 많은 application들이 R*-Tree구조를 많이 사용하고 있다[1].
하지만 위의 R-Tree 계열은 dimension이 증가할수록 query 속도가 현저히 떨어지며 또한 공간상의 object간의 relation을 표현하는데에 있어 부족함이 있다[4].

이러한 다차원 공간 데이터를 처리하기 위한 TV-Tree, SS-Tree등이 등장하였고 이중 하나인 X-tree의 아이디어는 overlap-minimizing split과 supernode를 두어 tree의 높은

level에서의 split을 피하고 만일 overflow가 발생하여 split을 요구하는 경우 split의 결과가 나쁜 overlap을 유발한다면 현재 level의 기본 노드 사이즈보다 2배로 증가한 supernode를 만들어 같이 저장한다는 것이다[6]. 하지만 이러한 방법은 낮은 dimension에서는 non-overlapping data를 가질 수 있지만 dimension이 높아질수록 결국 tree의 모양은 하나의 거대한 root-supernode로 되기때문에 data list처럼 되어 tree로서의 의미가 없어질수도 있다.

· QSF-Tree

최근에 발표된 QSF-Tree에서는 결국 데이터를 포함하는 MBR들의 overlap이 나쁜 query 결과와 index 구조를 만든다는 것에 그 문제점이 있다고 보고 이를 해결하기 위한 방안을 제시하였다[7]. 대부분의 PAM 방법들이 region overlap을 일으키지 않는 다는 데에 착안 그중 K-D-B Tree[9]를 기본구조로하는 Tree를 만들었다.

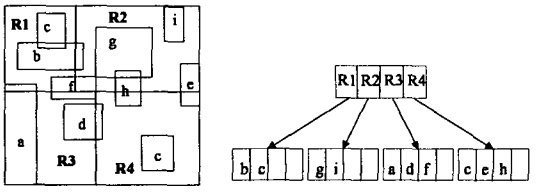
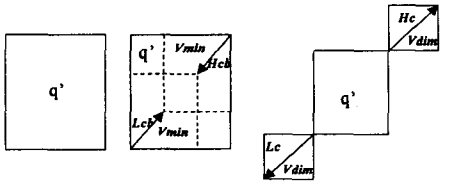


그림1 KDB-Tree에 기반을 둔 QSF-Tree

모든 객체를 구성하는 dimension의 최소값(Lp)과 최대값(Hp)을 가지고 객체를 표현하되 이중 각 dimension의 최소값을 기준으로 하여 KDB-Tree에 바탕을 두어 tree를 생성한다.
따라서 QSF-Tree에서는 결국 non-overlaping 구조를 만들 수 있어 overlap 때문에 야기되었던 기존 문제를 해결할 수가 있다. 하지만 tree에는 하나의 포인트에 대한 정보만 있기 때문에 이를 보완하기 위한 function들이 제공된다.
각 dimension에서의 가장 큰 interval 값과 작은 interval값을 Mi, mi로 기록한다. 이 값과 function으로 Contain과 Overlap등의 topological relation을 알아내기 위하여 L-region, H-region이라는 개념을 만들어 내었다. L-region은 주어진 조건을 만족하는 MBR들의 모든 low endpoint를 포함하는 공간을 L-region으로 이와 마찬가지로 주어진 조건을 만족하는 MBR들의 *high endpoint*를 포함하는 *region*을 *H-region*이라 한다.



Equal(r', q') Contained_by(r', q') Contain(r', q')

그림2 Relation에 따른 검색 범위

주어진 조건에 따라 각 관계에 따른 relation의 L-region에 해당하는 MBR들을 검색 후 이들중 high point가 H-region에 있는 MBR들을 추출하여 실제 데이터를 비교 검색 후 그 결과를 출력하게 된다. 그림2는 relation에 따른 검색 범위를 나타낸 것이다.

3. 기본 개념

KDB-Tree에 기반을 두고 있는 QSF-Tree는 page utilization이 60% 내외이다. utilization이 떨어지면 그만큼 많은 leaf node들이 생성되며 결국 search나 insert할 때 더 많은 disk access를 요구하게 된다. 이와 함께 overflow시 발생하는 split을 실행하는 과정에서 만들어지는 Time cost를 줄이기 위해 되도록 split을 피하면서 disk utilization을 높이기 위한 data들의 migration방법을 제안한다.
기본적인 insert방법은 원래의 QSF-tree와 유사하지만 여기서 insert시 migration이라는 과정을 거치게 한다.

· *migration()*

1) root로부터 시작하여 새로운 data가 삽입될 node를 선택하게 된다.
2) 만일 새로운 data를 삽입함으로써 overflow가 발생할 경우 이 leaf node를 포함하는 parent node의 region 정보를 읽어 형제 node를 찾는다.
3) 형제 node가 full이 아니면 삽입될 node에서 형제 node와 split한 축의 경계에 가장 가까운 data (1~최대갯수 * 10%)를 골라 형제 node에 보내고 이에 따라 region 정보를 수정한다.
4) 만일 형제 node역시 full인 경우 split을 실행한다.

· *Interval -split()*

tree구조는 low point를 기준으로 만들지만 가까이 있는 data들이 되도록 같은 node에 위치할 수 있도록 split할때의 축과 값의 선택은 각 interval을 중심으로 split한다. split할 때 기준점과 걸쳐지는 interval들이 가장 값을 선택하여 실제 공간상에 가까운 위치에 있는 data들이 같은 node에 속할 수 있도록 한다.

· *Search*

검색을 하는 방법은 다음과 같다.
1) data를 구성하는 각 dimension의 최대 최소 값을 구한다.
2) 조건을 만족하는 MBR들을 선택한다.
3) MBR을 만족하는 data들을 대상으로 실제 값을 비교하여 결과를 출력한다.

Dimitris Padias[8] 연구 결과에 따르면 7가지로 요약될 수 있는 toplogical relation을 가진 MBR과 실제 데이터들을 비교해본 결과 MBR끼리 만족하는 relation들은 실제 data 비교시 성립하지 않는 case가 많다는 것을 보여주고 있다. 실제로 사용되는 application들은 not_disjoint와 disjoint의 관계에 집중되어 있는 경우가 많으며 여기서 MBR들의 overlap되는 위치에 따라 실제 data 값을 비교해 보지 않아도 data들이 overlap되는 경우를 알아낼 수 있어 실 data를 비교하는데 걸리는 시간을 단축시킬수 있는 경우를 보여주었다. 그림3는 각 dimension에서의 data의 interval(Lp, Hp)간의 있을 수 있는 overlap 관계를 보여주는 것이다. 또한 그림 4는 2차원 데이터일 경우

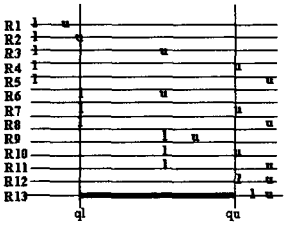


그림3 각 차원에서의 Relation

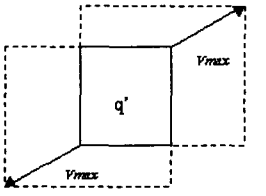


그림4 Overlap의 범위

overlap relation에서 data 검색 범위를 보여준다.

dimension	overlap
2	(R9,R4) (R9,R8) (R9,R5)
3	(R4,R7,R9) (R7,R8,R9) (R5,R7,R9) (R7,R7,R9)

표1 실제 data값을 비교하지 않아도 overlap관계임을 알 수 있는 MBR Relation

위의 표는 data의 각 dimension에서의 interval끼리의 관계에 따라 2, 3차원 data에 대하여 실제 데이터를 비교하지 않아도 overlap relation을 알수 있는 경우이다.

또한 어느 한 dimension에서라도 R1 혹은 R13인 관계가 있는 경우 이들의 MBR은 물론 실제 data들은 disjoint하다.

4. 실험 및 연구 결과

disk utilization을 높이기 위한 방법으로 split을 실행하기전 형제 node의 포화상태를 보고 data를 migration하는 방법을 실행하였을때의 disk utilization을 비교해보았다.

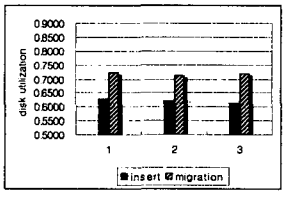


그림5 insert방법에 따른 disk utilization

회수	insert	migration
1	0.6266	0.7203
2	0.6219	0.7128
3	0.6158	0.7187

표2 다른 data로 실행해본 disk utilization 값

표2 에는 나타나 있지 않지만 branch의 개수가 증가함에 따라 utilization도 약간씩 증가하는 경향을 보였다.

다차원 data를 split할 때 우선 기준이 되는 것은 dimension의 선택이다. 두가지의 축 선택방법으로 실험을 하였다.

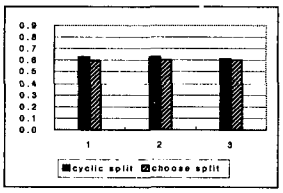


그림6 split방법에따른 disk utilization

회수	cyclic split	choose split
1	0.6266	0.5974
2	0.6219	0.5995
3	0.6158	0.6002

표3 split방법에 따른 utilization

그림6은 아래에서 제시한 방법으로 split하였을 때의 disk utilization의 비교이다. split하는 방법은 우선 cyclic한 방법으로 dimension0->.....->dimensionN -> dimension0....의 순서로 하는 방법, 그리고 각 dimension에서 split을 하는 경우 경계값과 걸쳐지는 data 갯수가 적은 축을 골라 우선 split하는 방법을 비교하였다. split방법에 따른 time cost비교는 다음과 같다. 실험 결과 cyclic한 방법으로 하였을 때 더 좋다는걸 보여준다

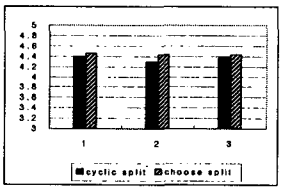


그림7 Time cost

회수	cyclic split	choose split
1	4.3847	4.4556
2	4.2936	4.4325
3	4.3839	4.4331

표4 time cost 값

query의 경우 contain, contained_by, cover, covered_by, equal인 관계를 만족하는 object를 찾고자하는 경우 조건에 맞는 MBR을 가지는 data들의 실제값들을 비교하여야 하지만 overlap관계에서는 일부 위의 표에서와 같은 조건을 가진 MBR들은 실제 data를 구성하는 값을 일일이 비교하지 않아도 된다. overlap relation을 검색시 절약할 수 있는 time cost는 실제 data를 이루고 있는 data의 구성에 따라 약간씩 다르긴하겠지만 대략 15%정도의 속도가 향상된다.

5. 향후 계획

QSF-Tree는 tree에 들어가는 실제 data들이 모두 일정한 크기를 가지는 경우나 small data인 경우 매우 빠른 query 처리 속도를 보여주지만 다양한 사이즈의 data들이 대상인 경우 몇개의 아주 큰 large data로 인하여 Mi값이 증가하게 되고 따라서 data의 topological relation을 만족하는 다른 data 검색시 검색 범위가 커지게되므로 따라서 많은 disk access를 유발하는 단점이 있다. 향후에는 이러한 Mi값으로 낭비되는 disk access 수를 감소 시킬수 있는 search 범위값을 줄일 수 있는 기준을 만들어 보고자 한다.

6. 참고 문헌

[1]Robert Beckmann, Hans-Peter Kriegel Ralf Schneider, Bernhard Seege : "The R*-tree:An efficient and Robust Access Method for Points and Rectangles," Praktische informatik, Universitaet Brem en, D-28000 Bremen 33, West Germany.

[2]A. Guttman : "R-trees a dynamic index structure for spatial searching" ,*PROC. ACM SIGMODE Int. Conf. on management of data,* 45-47, 1984.

[3]T.Sellis, N.Roussopoulos, and C.Faloutsos : "The R+-Tree: A Dynamic Index for Multi-Dimensional Objects," *Proc. 16th Int. Conf. on VLDB,* 507-518, 1987.

[4]D.Greene : "An Implemetation and Performance Analysis of Spatial Data Access Methods," *Proc. 5th IEEE Int. Conf on Data Engineering,* 606-615. 1989.

[5]Hanan Samet : *The Design and Analysis of Spatial Data Structures,* ADDISON-WESLEY PUBLISHING COMPANY, INC.

[6]S.Berchtold, D.A.Keim and H.Kriegel : "The X-tree: An Indes Structure for High-Dimensional Data," *Proc. 22th Int Conf, on VLDB,* 28-39, 1996.

[7]B.Yu, R.Orlandic and M Evens :"Simple QSF-Tees : An Efficient and Scalable Spatial Access Method", *PROC. ACM SIGMODE Int. Conf.* 1999.

[8]D.Papadias, Y.Throforis, T.Sellis, M.egenhofer : "Topological Relation in the world of Minimal Bounding Rectangles: A Study with R-Tree," *PROC. ACM SIGMODE Int. Conf. on management of data,* 92-103, 1995.

[9]J.T.Robinson : "The K-D-B Tree : A Search Structure for Large Multidimensional Dynamic Indexes," *PROC. ACM SIGMODE Int. Conf. on management of data,* 10-18, 1981.