Mobile GIS DBMS에서 비용효율적인 대량삽입기법에 관한 연구

이경아° 이근주 진성일
충남대학교 컴퓨터과학과
{kalee°, eggnzoo, sijin}@cnu.ac.kr

A Cost-Effective Bulk Insertion Method in Mobile GIS DBMS Environment

Kyung ah Lee° K. J. Lee S.I. Jin
Department of Computer Science Chungnam National University

요 약

최근 무선 통신 구조가 확산대면서, 모바일 기기와 다양한 모바일 GIS 서비스가 일반화되었다. 그래서 모바일 기기에서 효과적으로 GIS 자료들을 검색할 수 있는 모바일용 GIS DBMS가 필요하다. 물론 모바일 GIS DBMS는 환경적 제약사항과 GIS 자료들의 특성들을 극복하기 위한 대량삽입 기법이 필요하며 이 기법은 모바일 GIS DBMS를 개발하는데 있어서 가장 중요하다. 이 제약사항은 낮은 대역폭, 작은 저장공간 그리고 비공간 데이터 보다 상대적으로 매우 큰 공간 데이터의 특징을 포함한다. OBO기법을 이용하는 GIS DBMS로의 데이터의 삽입은 차례대로 삽입된다. 그러나 대부분의 공간데이터들은 전송비용과 긴 로딩시간으로 인해 서비스를 지연시키기 때문에 공간 인덱스를 기반으로 하는 대량삽입을 선택한다. 본 논문에서는 높은 성능과 안전성을 가진 동적인 공간 인덱스 R*-tree을 이용하는 대량 삽입 기법을 제공한다.

1. 서 론

최근 들어 이동통신 기술 등의 발달로 인해 공간과 위치의 제약없이 사용할 수 있는 이동 컴퓨팅(Mobile Computing)이 주목을 받으면서, 사용자들은 휴대용 무선 단말기를 이용하여 수많은 인터넷 컨텐트 뿐만 아니라 다양한 서비스를 이용할 수 있게 되었다.

이렇듯 무선 인터넷 환경이 일반화되면서 대용량화 및 고성능화된 다양한 GIS 서비스가 요구되고 있다. Mobile GIS 서비스는 무선 인터넷 단말기를 가진 이동 사용자에게 자신이 원하는 다양한 GIS 부가정보(위치정보, 도로 · 버스안내, 이동단속업무, 위치기반 비교쇼핑등)를 제공해주고 있다.

현재 많은 GIS 분야의 서비스가 제공되고 있지만 DBMS를 사용하지 않고 파일 시스템을 사용하고 있다. 하지만 사용자의 요구를 만족시키기 위해 여러 종류의 application들과 이를 위한 많은 공간 데이터들을 관리하는 Mobile를 위한 GIS DBMS를 필요로 하게 된다.

그러나 Mobile GIS 서비스를 효율적으로 사용하기 위해서는, 이동 컴퓨팅 환경에서 존재하는 많은 제약사항들(대량의 공간 데이터를 낮은 대역폭, 작은 디스플레이 화면, 작은 저장공간)이 고려되면서도 신속한 서비스가 될 수 있어야 한다. 서비스 시간은 보통 검색 시간과 삽입 시간으로 이루어져 있으며, 주로 검색 작업이 더 많이 수행되지만, Mobile의 제약조건과 비용면에서 고려해 볼 때, 삽입으로 인한 오버헤드는 서비스 시간에 있어서 검색보다 훨씬 큰 영향을 주게 된다. 따라서 Mobile 데이터베이스에 대량의 공간 자료를 빠른 시간내에 삽입하는 것은 이동형 GIS DBMS 개발에 있어서 매우 중요한 초점이 된다.

기존의 공간 데이터를 삽입기법인 OBO(One-by-One) 방식에서는 데이터가 순차적으로 레코드 단위로 삽입되며 각각의 데이터가 삽입될 때마다 인덱스가 재구성되는데, 이런 비효율적인 연산으로 인하여 확장성이 떨어지게 된다. 그러므로 삽입 오버헤드와 비용을 줄이기 위해 다량의 큰 공간 데이터를 빠른 시간 내에 삽입 할 수 있는 대량 삽입(Bulk insertion)기법이 필요하다.

현재 나와 있는 공간 데이터에 대한 대량 삽입 기법들은 삽입 시간 단축만을 고려할 뿐, Mobile환경의 제약사항을 고려하지 않으며, 다른 노드들과 오버랩 또한 늘어나지 않는다고 확신할 수 없다.

따라서, 본 논문에서는 위와 같은 문제를 해결하기 위하여 가장 많은 비용 및 시간을 차지하는 트리 구성 시간의 단축을 제일 중요한 목표로 삼고, 그 밖의 다른 제약사항과 함께 해결하고자 한다. 이를 위해 안정성과 성능이 입증된 R*-tree와 GBI방법을 Mobile GIS DBMS 환경에 맞도록 변경·설계하고 구현함으로써, Mobile DBMS에서의 색인 구성 시간의 단축 및 질의 성능 향상을 기대할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 관련된 연구들을 기술하고, 3장에서는 분석 내용을 바탕으로 제안된 Mobile GIS DBMS의 구조와 공간 대량 삽입 기법에 대해 설계하고 구현한다. 4장에서는 가장 기본적인 삽입 방법인 OBO방법과 본 논문에서 구현된 대량 삽입 기법을 비교 실험하고 이에 대한 성능평가를 하며, 5장에서는 본 연구의 결론 및 향후 연구에 대해 기술한다.

2. 관련연구

2.1. Mobile DBMS

Mobile 컴퓨팅이란, 사용자가 이동 중일 때도 휴대용 컴퓨터와 무선 통신을 통해 네트워크에 접속할 수 있도록 해주는 컴퓨팅 패러다임으로 현재 많은 제품들이 출시되고 있다.

Mobile 환경에서는 장시간의 접속 단절, 유선에 비해 고가이면서 저속인 네트워크상태, 긴 대기시간, 간헐적인 연산, 낮은 통신 대역폭과 적은 저장공간등의 고려사항들 때문에, 이를 위한 Mobile DBMS 요구사항들이 존재하게 된다.

첫째, Mobile DBMS는 플랫폼에 독립적이어야 한다. 다양한 이동 통신 제품들이 출시되면서 이동 단말기의 운영체제도 다양하게 채택되고 있다. 따라서, 이러한 다양한 하드웨어와 소프트웨어를 지원하기 위해서는 플랫폼에 독립적이어야 한다.

둘째, 시스템에 최적화되어야 한다. 이동 단말기는 컴퓨팅 파워와 메모리의 용량이 적으며, 배터리 수명 또한 짧기 때문에 적은 자원을 이용해도 동작할 수 있는 최적화가 필요하다.

셋째, 데이터 일치성(data consistency)과 보안을 보장하여야 한다. 이동 단말기는 서버와의 연결이 단절된 상태에서 작업 처리가 빈번하므로 서버와 이동 단말간의 데이터 복제 및 데이터 일치성의 유지와 사용자 보안이 중요하다.

넷째, 데이터 충돌(conflicts)을 해결하여야 한다. 서버와 접속이 단절된 상태에서 다수의 작업자는 각자 자신의 트랜잭션이 유효하다고 믿고 있다. 그러나 동기화를 했을 때, 충돌 발생이 가능하므로 이에 대한 해결이 필요하다.

2.2. 공간색인연구

CAD나 GIS와 같은 공간데이터베이스(Spatial Database) 응용에서는 다차원 공간상의 객체들을 f차원 공간의 한 점으로 대응시켜, 그 점을 탐색하고 클러스터링 하기 위해 다중속성접근방법(multi-attribute access method)을 사용하는데, 이것이 공간 인덱싱이며 가장 많이 사용되는 기법이 계층화된 검색트리를 기반으로 하는 R-Tree와 R*-Tree이다.

R-tree는 최소 사각형(MBR)에 기반한 방법으로 현재 가장 널리 쓰이고 있고 B-tree와 같이 높이 균형 트리이며, 각 트리 노드는 디스크 페이지에 대응된다.

R*-tree는 R-tree의 랜덤한 삽입에 따른 열악한 공간 구성을 극복하기 위해 삽입 알고리즘을 개선한 R-Tree의 변형으로, 개선 목표는 노드 겹침, 노드 포함관계 그리고 노드 사각형의 둘레길이를 최적화하는 것이다. 그러나 이 세 개의 파라메터를 동시에 최적화하는 기술은 아직 알려지지 않았으므로 R-tree에 가장 중요한 개선을 가져오는 두 개의 다른 방법이 있다. 첫번째 방법은 분할 알고리즘을 개선시킨 것으로 처음엔 가장 멀리 떨어져 있는 두 개의 엔티티를 가진 두 그룹을 초기화하고 위에서 언급한 directory의 세 가지 파라메터를 차례대로 적용하여 남아있는 엔트리들을 할당한다. 두 번째 방법은 규정된 재 삽입 전략을 이용한 것으로, 주어진 레벨에서 첫 번째 오버플로우가 발생하게 되면 엔트리들을 재삽입함으로써 좋지 않은 경우를 피하는 것이다. 재삽입시에도 오버플로우가 발생하는 노드에 있는 객체들은 엔트리과 노드의 MBR 사이의 중심거리를 계산하여 내림차순으로 정렬하고, 최대 엔트리 개수의 30%를 추출하여 이웃하는 다른 노드에 재삽입한다. 또한 boundering rectangle을 재조정함으로써 겹침을 감소시키고 저장 활용성도 증가하게 되었다.

2.3 공간색인을 이용한 대량 삽입 기법 연구

기존의 대량 삽입 기법들에는 STLT, 합병기반기술(Merging-based techniques) 그리고 버퍼링기반기술(Buffering-based techniques) 등이 있다.

2.3.1 STLT(Small Tree Large Tree)

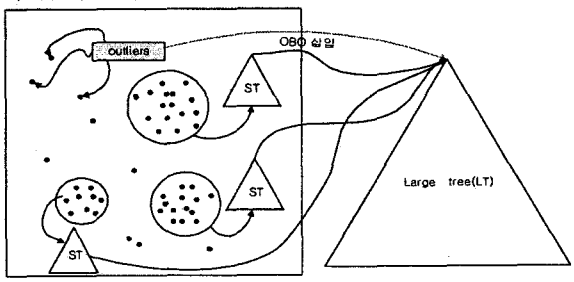
STLT기술은 기존에 이미 존재하는 R-tree에 다수의 skewed data을 삽입하기 위해 제안된 방식으로 새로 삽입할 데이터들은 일부 특정 공간에 군집되어 있는 데이터이다. 따라서 이러한 데이터들은 기존의 대량 로딩 알고리즘에 의해 R-트리에 이미 존재하는 데이터와 독립적으로 새로운 트리 생성하고 이를 R-트리에 삽입하게 된다.

그러나 Small-tree는 원래 트리에 비해 매우 Skewed되어 있다고 가정하였기 때문에, Small-tree를 재분배할 필요 없이 한 번에 원래 트리에 삽입을 수행한다는 장점을 가지고 있기는 하지만, 대량삽입시 데이터가 산재되어 있을 경우 순차적으로 삽입하는 것과 마찬가지이므로 효과적인 색인 구성이 불가능하다.

2.3.2 GBI(Generalized R-Tree Bulk-Insertion)

Choubey et al.에 의해 제안된 GBI 기법은 일반화된 R-tree 대량 삽입 기술이라고 불리며, 다양한 방법으로 향상되고 있다. 이 기법은 처음으로 주어진 데이터 집합에 대해 클러스터링을 수행하여 군집이 잘 되어 있는 데이터들에 대해서는 클러스터(Cluster)와 그렇지 않는 데이터들에 대해서는 outlier로 구분하는 전통적인 클러스터링 알고리즘을 사용했다. R-tree가 대량 로딩 (Bulk-loading)기법으로 구성되고 난 후, 각각의 클러스터 집합들은 STLT 알고리즘에 의해 Small-tree를 생성하여 대상 R-tree에 삽입되고 Outlier들은 OBO(one-by-one) 형태로 삽입된다.

다음 [그림 1]은 클러스터링 알고리즘에 생성된 클러스터와 Outlier를 삽입하는 과정을 그림으로 나타낸다.



[그림 1] 생성된 클러스터의 삽입

위와 같은 일반적인 대량 삽입 기법이 Mobile 환경의 환경적, 서비스상의 제약사항을 고려할 때 가장 적합하므로, 3장에서 제안하는 시스템에 맞도록 위 대량삽입기법을 변경·적용하였다.

2.3.3 Buffering-Based Techniques

R-tree에서 대량 삽입을 위해 구현된 또 다른 기법은 Arge et al.에 의해 제안된 것으로 내부 메모리를 사용하는 데이터 구조로 변환하여 효율적으로 I/O를 감소 시킨 것이다. 이 방식은 단말 노드를 레벨0으로 정의하고 각 레벨마다 일정크기 블록의 버퍼를 할당한 후, 이 노드들을 버퍼 노드라 칭한다.

영역을 삽입하기 위해서, 단말노드를 바로 검색하는 것이 아니라 대신에 삽입될 rectangle들을 대상 R-tree를 따라 같은 계통 경로를 공유하는 삽입 MBR 아이템들을 그룹화하고, 버퍼에 블록이 수집될 때까지 기다린 후, 루트의 버퍼에 있는 블록들을 디스크에 저장한다.

버퍼가 가득차게 되면, buffer emptying 처리를 호출하여 버퍼에 있는 각각의 rectangle들을 다음의 낮은 레벨에 있는 버퍼로 삽입한다. Rectangle이 도달되면 단말에 삽입이 이루어지고 필요하다면 인덱스는 split을 사용하여 재구성된다.

이러한 삽입과정 때문에, 몇몇 삽입은 전혀 I/O비용이 들지 않는다. 반면에 많은 buffer emptying 처리로 인하여 다른 삽입들은 매우 많은 비용을 소요할지도 모른다. 하지만 버퍼의 사용은 큰 블록과 내부 메모리 크기의 이점을 주고 우리가 이전에 사용했던 방법보다는 적은 I/O 비용으로 전체적으로 대량 삽입을 가능하게 해준다. 그러나 보조 구조 때문에 추가의 비용이 요구되므로 상용화된 데이터베이스 환경에서는 실행이 불가능하다.

3. Mobile GIS DBMS를 위한 대량 삽입 기법의 설계 및 구현

지속적으로 생겨나는 대량의 복잡한 공간 데이터들을 관리하는 MMDBMS에서는 OBO방식의 삽입으로 인한 비효율적인 비용과 긴 삽입시간으로 인해 떨어지는 확장성을 극복하기 위해 몇 가지 일반적인 대량 삽입 방법을 제시하고 있다.

그러나 Mobile GIS DBMS에서는 기존연구에서 언급된 대량 삽입 기법들이 적합하지 않다. 그러므로 이런 환경에서 공간 데이터를 빠른 시간내에 삽입하면서도 결과로 얻어지는 트리의 질을 향상시키기 위한 대량 삽입 기법이 필요하며, 여기에는 몇 가지 제기되는 사항들이 있다.

그래서 이런 사항들을 해결하기 위하여 이 장에서는 Mobile GIS DBMS에 적합한 대량 삽입 기법을 모델링하고 설계하여 이런 프로세스와 산출물을 이용하여 구현한 내용을 기술한다.

3.1. 문제정의 및 요구분석

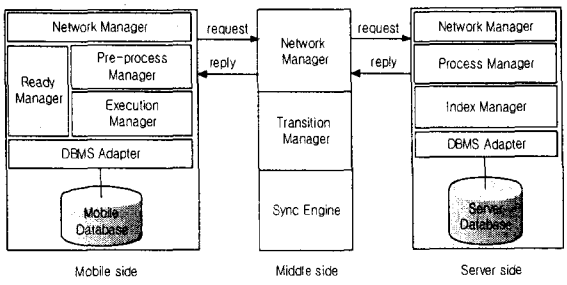
전체적인 시스템의 프로세스는 다음과 같다. 사용자가 지도 검색을 요청하거나 사용자가 이동하면서 다음의 지도가 요청하면, 요청된 공간데이터는 Client database에 그 내용의 존재유무가 검색이 되고, 있다면 사용자의 Mobile 화면에 디스플레이 되지만 요청된 부분이 전체 혹은 일부분이 없다면 데이터를 Middle side로 요청한다. Middle side에서는 요청된 데이터를 받아 서버에서 필요로 하는 형식에 맞게 변환되어 Server로 보낸 후, 서버에서는 이에 맞는 데이터를 검색하고 적합한 공간객체를 Middle side로 보내어 Client에 맞게 변환한 후 Client에서 이를 받아 기존 DB에 있던 내용과 병합하여 사용자에게 결과를 화면에 보여준다.

위와 같은 프로세서에서는 갖가지 제약사항 때문에 요구사항들을 가지게 되므로 이를 만족시키기 위해서 다음과 같은 가정을 갖는다.

1) Mobile 환경이기 때문에 Mobile과 Server 중간에 동기화를 해결해주는 Middle side를 두는 Mid-tier 구조를 갖는다.
2) Mobile side에 없는 공간 데이터를 가져오는 것이기 때문에 데이터 간의 충돌은 없다.
3) Mobile GIS DBMS는 소켓을 통한 TCP/IP 통신을 한다.
4) Mobile GIS DBMS는 충돌이 없기 때문에 Middle side 중심의 설계가 아닌 Mobile과 서버 중심 설계를 한다.
5) 서버로부터의 정보 전달은 사용자에 의해 혹은 Mobile에 의해 받은 요청에 대해서만 결과를 응답해 준다.

3.2. 전체시스템 구성도

GIS 데이터를 위한 Mobile DBMS 구조는 [그림 2]와 같이 Server side에는 Station Server, Middle side에는 Sync Server, Mobile side 즉, Client에는 Mobile 시스템으로 구성되어 있다.



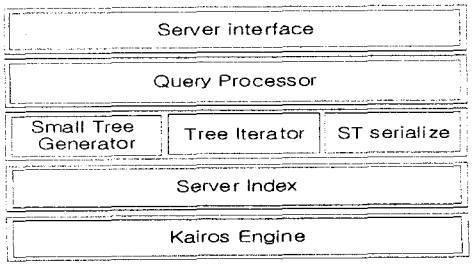
[그림 2]Mobile GIS DBMS 구조도

기존의 GBI에서는 서버에 필요한 데이터를 요청하여 Client에서 이 데이터를 받아서 클러스터링하고 이를 집합으로 만들어 Client의 DB에 병합하였는데 이 시스템에서는 특성상 Mobile분야에서 많은 연산을 하게 되면 많은 전력 소비와 늦은 검색 처리 속도를 가질 수 있기 때문에 이를 위해 세 부분 중 Server side에서 결과를 추출하여 미리 tree를 구축하여 전송하는 역할을 갖게 된다.

3.3. 시스템 설계

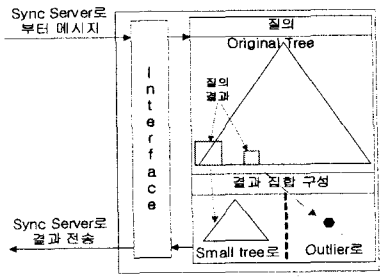
3.3.1 Server side 설계

서버의 구조를 모듈 별로 나누어 보면 [그림 3]과 같다. 모듈별로 살펴보면 Network와 관련되어 메시지를 송수신하는 역할을 하는 Server Interface가 제일 윗단에 위치하고 그 밑에는 쿼리를 처리하는 모듈인 Query Processor, 그 밑단에는 추출된 자료를 가공하는 모듈, Server의 인덱스 구조를 관리하는 Server Index, Database 접근 및 Data등을 관리하는 Kairos Engine들로 구성되어 있다.



[그림 3] sever side 모듈 구성

이러한 모듈들은 클래스로 구성되어 캡슐화 되여 메시지 기반의 구조로 연결되고 각 모듈이 하는 절차는 다음 [그림 4]와 같다. Middle side로부터의 메시지를 Socket을 통해 받는 과정이 Server interface module에서 이루어 진 후, 받은 메시지를 Query Processor에 의해 질의를 실행하게 되는 데 이때 기존 Server에 구축되어 있는 Server index 에 접근하여 그 객체가 속한 MBR들을 검색하여 데이터들을 추출하게 된다. 이때 검색된 결과는 Cluster set과 Outlier set으로 나뉜 후, Cluster set은 위해 SmallTree Generator에 의해 small tree를 생성하고 위에서 추출된 record들은 Tree Iterator에 의해 리스트 형태로 형성되며 구성된 small tree에 각 데이터를 삽입하고 ST serialize에서는 이 결과를 인덱스 그대로 middle side로 보낼 수 없기 때문에 flat화 시키는 작업을 하게 된다. Kairos engine에서는 인덱스와 관련된 데이터들을 저장하고 있는 DB를 연결하고 있다.

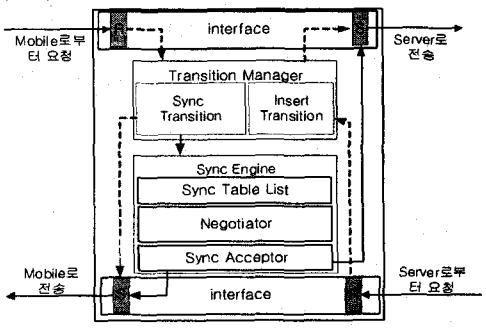


[그림 4] Server side 진행 절차 및 상세구조

3.3.2 Middle side 설계

다음[그림 5]는 Middle side의 상세 구조와 사건이 일어나

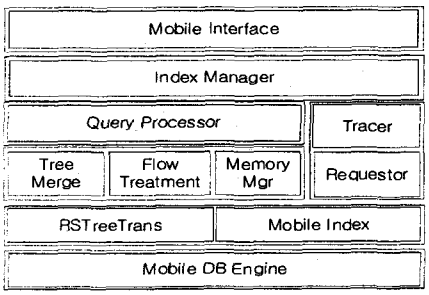
는 사건의 절차를 간단하게 도식화 한 것이다. Socket Syne를 통해 Mobile로부터의 요청 메시지를 Receiver에서 받고 일련의 과정을 거친 후 다시 Sender를 이용해 보내게 되는데 이러한 과정이 모두 Mobile Interface에서 이루어진다. 받은 요청 메시지가 Transition Manager를 통해 Server에 맞는 적절한 변환을 거친 후, 다시 이를 Network Module에 의해 목적지인 Server로 보낸다. 이와 반대로 요청된 MBR에 대한 결과가 Server로부터 전송되면 이 또한 Mobile Interface에서 받아서 이를 다시 Transition Manager에서 Mobile에 적절하게 변경을 가해준다. 변경된 메시지는 Mobile Interface를 통해 Mobile로 보내진다. 이러한 일련의 과정들은 점선으로 표시된 부분이며 나머지 부분은 동기화에 관련된 부분이므로 설명을 생략한다.



[그림 5] Middel side 진행 절차 및 상세 구조

3.3.3 Mobile side 설계

모듈별로 살펴보면 Middle side로부터 메시지를 받거나 혹은 Middle side로 메시지를 보내는 역할을 하는 Mobile interface, 시스템의 시작은 요청으로부터 시작되는데, 이런 요청은 두 가지 종류가 있고 그 중 하나는 사용자에 의한 요청을 받아들이는 Requestor이고, 나머지 하나는 사용자의 위치추적 중에 필요한 정보를 받을 수 있도록 하는 Mobile의 요청을 받아들이는 Tracer이다. 또한 위의 두 가지 모든 요청에 대한 쿼리를 실행하고 이 질의를 잠시 저장하는 Query Processor, 기존의 인덱스와 응답된 인덱스를 병합하는 Tree Merge, 병합하는 과정 중 오버플로우나 Underflow 처리를 위한 FlowTreatment, Server로부터 넘어온 데이터 공간을 낭비없이 Mobile의 공간과 연결시키는 Mempry Mgr, 메시지를 다시 복원 시키는 RSTreeTrans, Mobile의 인덱스를 관리하는 Mobile index, Mobile의 데이터들을 관리하고 있는 Mobile DB Engine 등을 구성 요소로 가진다. 다음 [그림 6]은 Mobile side를 모듈별로 나눈것이다.



[그림 6] Mobile side 모듈 구성

3.4. 시스템 구현

3.4.1 Server side 구현

1) RSTreeiterator class

입력된 영역 질의에 대해 Query Processor에 의해 Server의 Index을 통해 조건에 맞는 데이터들을 추출하기 위하여 Kairos Engine에 접속하여 MBR과 Record들을 R*-tree형태의 Linked List로 구성해 준다.

2) RSTree class

Server에 있는 전반적인 R*-Tree의 인덱스 구조에 관련된 전반적인 사항을 관리한다.

함수의 원형	설 명
RSTree()	Original Tree에 대한 ID, 인덱스 타입,키 정보등에 대한 내용을 초기화한다.
getSize()	Original tree의 크기를 반환한다.
isValid()	모든 값의 유효성을 체크한다.
getRootNode()	Tree의 루트노드를 반환한다.
SplitNode()	노드에 오버플로우 발생시 분열이 일어날 때 노드를 분할한다.
isExit()	정해진 레코드 크기만큼을 읽고 나온다.
readMBR()	현재 위치의 MBR에 대한 정보를 읽는다.
reinsertNode()	넘침등이 발생했을 때 일부를 분할하지 않고 재삽입한다.
compareLeaf()	정해진 위치로 가기 위해 자식 단말 노드끼리 비교한다.
compareNonLeaf()	객체를 삽입한다던지 검색하기 위해 자식 비단말 노드끼리 비교한다.

[표 1] RSTree class

3) SmallTreeGen class

RSTreeIterator로부터 추출된 결과를 Clustering 한 결과 중 Cluster set을 저장하기 위한 SmallTree를 구성하고 데이터와 MBR들을 삽입한다.

함수의 원형	설 명
BuildTree()	Small Tree를 구성하고 Small Tree의 Root를 초기화 한다.
getNodeMBR()	현재 위치한 노드 MBR을 가져온다.
AdjustMBR()	검색된 하위 트리에 MBR에 맞는지 체크하고 그 정보를 반환한다.
ChooseSubtree()	Small Tree의 적정 위치에 MBR과 데이터를 삽입하기 위해 하위트리를 검색한다.
insertMBR()	구성된 Small Tree에 MBR들을 삽입한다.

[표 2] SmallTreeGen class

4) STreeSender class

SmallTreeGen에서 구성된 SmallTree와 Outlier로 Clustering된 객체들을 Mobile side로 넘겨주기 위해 Flat시켜 주는 작업을 해준다.

3.4.2 Middle side 구현

1) Transitor class

ServerL나 Mobile로부터 SocketSync를 통해 ReceiveMgr에 들어온 메시지를 목적지인 Mobile과 Server에 맞게 질의문을 변경시키고 이 메시지가 동기화를 위한 것인지 대량 삽입을

위한 메시지인지 판단하는 역할을 한다.

함수의 원형	설 명
Check()	요청된 메시지가 동기화를 위한 것인지 대량 삽입을 위한 것인지 판단한다.
StoMtrans()	대량 삽입을 위한 메시지 중 Server로부터 들어와서 Mobile로 전달시켜주어야 하는 메시지들을 변환 시킨다.
MtoStrans()	대량 삽입을 위한 메시지 중 Mobile로부터 들어와서 Server로 전달시켜주어야 하는 메시지들을 변환 시킨다.
Synctrans()	동기화를 위한 메시지를 적용할 테이블에 맞게 변환 시킨다.
makeOp()	동기화를 위해 Operation만 분리한다.
~Transitor()	각 함수들에 의해 변환된 메시지들을 SendMgr로 보내고 프로세스를 중단한다.

[표 3] Transitor class

3.4.3 Mobile side 구현

1) Interface class

Mobile side에서 가장 상단에 위치하여 Network와 관련된 일을 수행한다. Socket을 생성하고 메시지를 주고 받는 역할을 하게 된다.

2) IndexMgr class

삽입과 관련된 프로세스인지 동기화와 관련된 프로세스를 확인하고 삽입과 관련된 요청이면 다시 사용자의 요청인지 혹은 Mobile 자체의 요청인지 구분한 후 쿼리를 할 수 있는 준비를 하고 Mobile side의 인덱스를 관리한다.

3) TreeMerge class

기존의 DB에 있는 인덱스와 레코드를 Server로부터 받은 새로운 인덱스와 레코드를 받아 질의를 가능케 할 수 있도록 빠르게 병합하는 역할을 한다.

함수의 원형	설 명
FindSTPos()	삽입할 곳의 위치를 기존 Mobile side에 구축되어 있는 Large Tree에서 찾는다.
insertSTMBR()	Small Treed의 Root Node의 MBR의 정해진 위치의 Node에 넣는다.
ChooseSubtree()	삽입될 위치를 찾았으면 그 위치를 찾아가기 위해 하위 트리를 검색한다.
CheckTree()	복원된 데이터가 Tree인지 객체인지 확인
insertSTintoLT()	Mobile side에 구축되어 있는 Large Tree에 전송된 Small Tree를 삽입한다.
ResizeMBR()	병합이 된 후 전체적인 MBR의 크기 재조정
insertOBO()	클러스터링 된 것 중 outlier를 순차적으로 삽입한다.

[표 4] TreeMerge class

4) OverflowTreatment class

병합하는 도중 엔트리가 슬롯보다 최대 엔트리수보다 많거나 최소 엔트리수보다 적을 때 흐름을 처리한다.

함수의 원형	설 명
SelectSibling()	병합하기 위해 형제 노드를 선택
MergeSibling()	형제 노드와 병합을 시도
SelectFuthestChild()	병합을 위해 가장 먼 형제노드 선택
ReInsert()	병합이 실패하거나 최대 엔트리 개수를 넘을 때 삽입 알고리즘을 통해 재삽입
SplitNode()	삽입할 엔트리 슬롯을 생성하기 위해 인덱스 노드를 분할한다.
OverflowTreatment()	삽입과정중에 엔트리가 넘치게 되면 호출되어 인덱스의 효율성의 위해 구조를 조정할 값들을 셋팅한다.
OverflowExecute()	직접 오버플로우를 실행

[표 5] OverflowTreatment class

5) RSTreeTrans class

Middle side로부터 받은 결과 질의의 내용을 트리 형태로 복원시켜주는 역할을 한다.

6) MemoryMgr class

Middle side로부터 온 Real Data를 저장하고 있는 heap공간을 재사용하기 위하여 링크를 걸어주고 그 밖의 변경사항등을 적용시키는 역할을 한다.

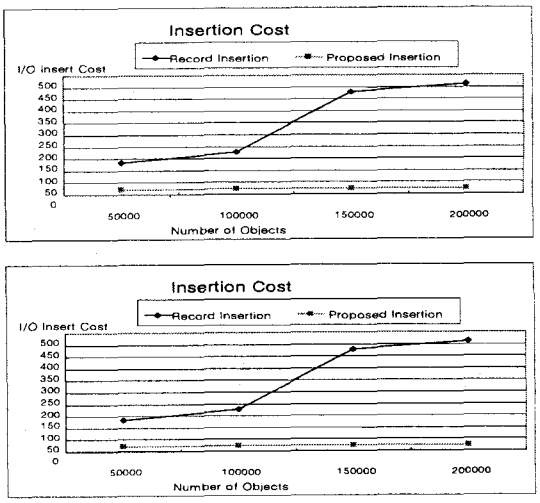
4. 성능분석 및 평가

4.1 실험 환경

이 테스트의 환경은 Server side와 Mobile side로 구성되어 있다. Server side는 UNIX 운영체제를 사용하며, Main Memory 상주형DBMS Kairos를 사용하였고, Mobile side는 Redhat Linux 7.1을 운영체제로 사용하며, Embedded Kairos를 Mobile 환경에 변형시킨 DBMS를 사용하였다.

4.2 실험결과

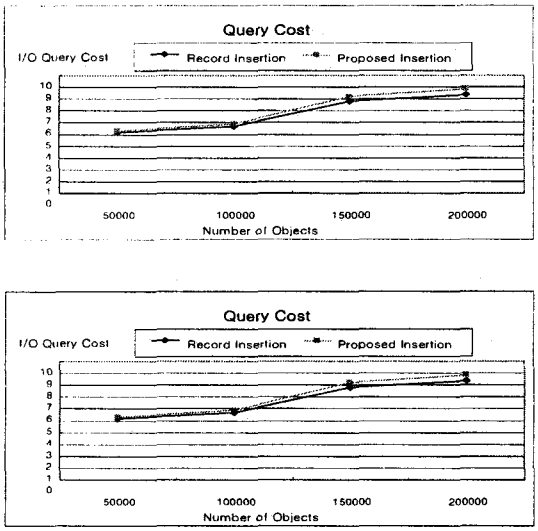
본 절에서는 제안한 대량 삽입기법과 순차로 레코드를 삽입하는 것을 비교 평가하였다.



[그림 7] Insert Cost 비교

위 [그림 7]은 삽입비용에 있어서 Record Insertion보다 제안된 Insertion이 훨씬 낫다는 것을 명확하게 보여준다. 데이터 집합의 크기가 증가할수록, 삽입 비용에 있어서 축척된 비용이 더 늘어남을 알 수 있다.

다음 [그림 8]은 생성된 결과 트리에 대한 질의 비용을 비교한 것인데, 결과의 질이 Record 단위보단 최상의 최적화가 이루어지지는 않았지만 수용할 만한 차이를 보여준다.



[그림 9] Query Cost 비교

5. 결론 및 향후연구 방향

기존의 주기억 장치 상주형 DBMS에서는 방대한 GIS data의 양과 크기를 효율적으로 제어하면서 사용자에게 서비스하기 위해 대량삽입기법을 도입하고 연구하고 있다. 그러나 아직 Mobile GIS 환경을 위한 DBMS 뿐만 아니라 이런 DBMS의 성능 향상을 위한 방법 또한 개발 혹은 상용화되지 않고 있다. 그러나 사용자의 서비스 요구 범위나 속도를 볼 때 현재의 파일 시스템을 기반으로 한 Mobile GIS 서비스로는 한계가 생길 것이다. 따라서 본 논문에서는 Mobile GIS환경에서의 서비스 시간의 향상과 Mobile 환경의 제약사항을 해결하기 위해 기존의 순차적 자료삽입 방식인 OBO 대신 Mobile GIS DBMS에 적합한 대량 삽입기법에 대해 연구하였다.

이를 위하여 Mobile DBMS를 Client로 간주하고 주기억 상주형 DBMS를 서버로 하며 Mobile 환경 특성상 동기화를 하는 엔진을 중간에 두는 3-tier 시스템 모델을 설정하고, 성능 및 안전성이 검증된 R*-tree 인덱스 구조를 이용하고, 주기억 상주형에서 일반적인 GBI 대량삽입 방법을 변경 및 적용하여 시스템을 설계 및 구현하였다. 이러한 변경된 대량삽입기법으로 로딩시간을 향상시켰고 안정된 인덱스 구조로 높이 re-balancing을 위한 고비용을 감소시켰다. 또한 성능평가를 위해 기존의 OBO방식과 비교하여 평가를 하였다.

앞으로 대량 삽입 이후 결과의 질을 향상시키기 위해 본 논문에서 기술된 간단한 클러스터링이 아닌 Mobile이 가진 트리까지 고려한 클러스터링 기법 연구가 필요하며, Mobile에서 영역 질의 요청시 중복에 대해 어떤 식으로 해결할 것인가가 연구되어야 할 것이다. 또한 서버와 Mobile 사이 통신을 하면서 좀 더 통신 비용을 낮추면서도 빠르게 flat화된 데이터를 전송하고 복원할 수 있는 방법이 연구되어야 할 것이다.

참고문헌

[1] A. Guttman, " R-trees: A dynamic index structure for spatial searching", Proceedings of International Conference on Management of Data, ACM SIGMOD, pp. 47-57, 1984

[2] N.Beckmann, H. P. Kroegel, "The R*-Tree: An Efficient and Robust Access Method for points and Rectangles", Proceedings of the 1990 ACM SIGMOD, pp. 332-331, 1990

[3] L. Arge. K.Hinrichs, et al, "Efficient Bulk Operations on Dynamic R-Trees", In Workshop on Algorithm Engineering and Experimentation(ALENEX), pp. 328-348, 1999.

[4] R. Choubey, L. Chen, E. Rundensteiner, "GBI : A Generalized R-Tree Bulk-Insertion Strategy", In Symposium on Large Spatial Database, pp. 91-108, 1999.

[5] N. Beckmann, H. Kriegel, R. Schneider, B. Seeger, "The R* tree: An efficient and rob-ust access method for points and rectangles", In Proc. ACM SIGMOD Int. Conf. on Mana-gement of Data, pp. 322-331, 1990.

[6] L. Chen, R. Choubey, E. Rundensteiner, "STLT : BulkInsertion into R-trees", In Proc-eedings of ACM International Workshop on Advances in Geographic Information Systems, pp. 161-162, 1998.

[7] I. Kamel, M. Khalil, V. Kouramajian, "Bulk Insertion In Dymanic R-trees", In Proceedi-ngs of 4th International Symposium on Spatial Data Handling, pp. 31-42, 1996.

[8] Ning An, Ravi Kanth, Kothuri, Siva Ravada, "Improving Performancewith Bulk-Inserti- on in Oracle R-Trees"

[9] Li Chen, Rupesh Choubey, Elke A. Rundensteiner, " Bulk insertions into R-Trees Using the Small-Tree-Large-Tree Approach,"ACM-GIS, pp.161-162, 1998.

[10] H Samet, "The Design and Analysis of Spatial Data Structures", Addison-Wesley, 1989

[11] 이석호, 문봉기, 이태원, "R-Tree에서 Seeded 클러스터링을 이용한 다량 삽입", 2004 정보과학회논문지, vol. 31, no. 1, pp.30-38, 2004

[12] 임덕성, 이재호, 홍봉희, "Mobile 지도 서비스를 위한 에이전트 기반의 공간 데이터 캐쉬의 설계 및 구현", vol. 10, no. 2, pp 175-186

[13] 김은영, 전봉기, 서영덕, 홍봉희, "PDA에서 공간 데이터를 저장하기 위한 혼성 색인 구조", 2001 한국정보과학지 논문집, vol. 28, no. 1, pp. 34-36, 2001

[14] 최우영, 이경아, 염태진, 진성일, "내장형 DBMS를 위한 동기화 서버 시스템", 2004 춘계학술대회, vol. 31, no. 1, pp127-129, 2004