공간 데이터 웨어하우스에서 부분 색인을 이용한 효율적인 색인 재구축 기법†

Efficient Index Reconstruction Methods using a Partial Index in a Spatial Data Warehouse

곽동욱*, 정영철**, 유병섭***, 김재홍****, 배해영*****

Dong-Uk Kwak, Young-Cheol Jeong, Byeong-Seob You, Jae-Hong Kim, Hae-Young Bae

요 약 공간 데이터 웨어하우스는 공간정보를 주제 중심적이고 통합적이며 시간성을 가지는 비 휘발성 자료로 저장하여 의사결정을 효율적으로 지원하는 시스템이다. 이 시스템은 구축기와 공간 데이터 웨어하우스 서버로 구성되어 있다. 공간 데이터 웨어하우스 서버는 구축기에서 전송된 데이터를 적재하기 위해 사용자 서비스를 정지하고, 사용자의 빠른 응답시간을 위해 적재된 데이터로 색인을 구축한다.

색인 구축을 위한 기존 기법에는 벌크 삽입 기법과 색인 전송 기법이 있다. 벌크 삽입 기법은 색인을 구축하기 위한 클러스터링 비용이 크며 검색 성능도 떨어진다. 색인 전송 기법은 주기적인 소스 데이터의 변경을 지원하지 않는 문제점이 있다.

본 논문에서는 공간 데이터 웨어하우스에서의 부분 색인을 이용한 효율적인 색인 재구축 기법을 제안한다. 제안 기법은 부분 색인을 직접 전송, 기록하며 물리적 위치 정보를 예상하여 기록할 수 있는 효율적인 색인 재구축 기법이다. 구축기에서 추출된 데이터를 공간의 근접도가 아닌 색인의 구조에 맞게 클러스터링하며, 생성된 각 클러스터를 부분 색인으로 구성하여 페이지 단위로 전송한다. 공간 데이터 웨어하우스 서버에서는 전송된 부분 색인을 저장하기 위해 물리적으로 연속된 공간을 예약하고 예약된 공간에 부분 색인을 기록한다. 기록된 부분 색인을 공간 데이터 웨어하우스 서버의 기 구축된 색인에 삽입함으로써 색인 재구축을 위한 검색, 분할, 재조정 비용이 최소화 된다.

Abstract A spatial data warehouse is a system that stores geographical information as a subject oriented, integrated, time-variant, non-volatile collection for efficiently supporting decision. This system consists of a builder and a spatial data warehouse server. A spatial data warehouse server suspends user services, stores transferred data in the data repository and constructs index using stored data for short response time.

Existing methods that construct index are bulk-insertion and index transfer methods. The Bulk-insertion method has high clustering cost for constructing index and searching cost. The Index transfer method has improper for the index reconstruction method of a spatial data warehouse where periodic source data are inserted.

In this paper, the efficient index reconstruction method using a partial index in a spatial data warehouse is proposed. This method is an efficient reconstruction method that transfers a partial index and stores a partial index with expecting physical location. This method clusters a spatial data making it suitable to construct index and change created clusters to a partial index and transfers pages that store a partial index. A spatial data warehouse server reserves sequent physical space of a disk and stores a partial index in the reserved space. Through inserting a partial index into constructed index in a spatial data warehouse server, searching, splitting, remodifing costs are reduced to the minimum.

주요어 : 공간 데이터 웨어하우스, 색인 재구축, 공간 색인

KeyWords : Spatial Data Warehouse, Index Reconstruction, Spatial Index

† 본 연구는 대학 IT연구센터 육성·지원사업의 연구결과로 수행되었음

* 인하대 컴퓨터·정보공학과 석사과정 kwakdonguk@dblab.inha.ac.kr

** LG CNS 금융솔루션 개발팀 aslongas3@dblab.inha.ac.kr

*** 인하대 컴퓨터·정보공학과 박사과정 subi@dblab.inha.ac.kr

**** 영동대학교 컴퓨터공학과 교수 leciel@youngdong.ac.kr

***** 인하대 컴퓨터공학부 교수 hybae@inha.ac.kr

1. 서 론

최근 급변하는 시장 변화에 대응하기 위해 정확하고 신속한 의사 결정이 기업의 중요 요소가 되었다. 따라서 기업은 입지선정, 물류이동경로선택 등의 의사결정을 하기 위해 경영에 필요한 정보와 공간 정보를 통합하고 있으며, 이로 인해 공간 의사결정 지원 시스템인 공간 데이터 웨어하우스에 관심이 집중되고 있다[15].

공간 데이터 웨어하우스는 공간 정보를 주제 중심적이고 통합적이며 시간성을 가지는 비 휘발성 자료로 저장하여 효율적인 의사결정을 지원하는 시스템이다[8][9][13][16]. 이 시스템은 구축기와 공간 데이터 웨어하우스 서버로 구성된다[20]. 구축기는 소스 데이터베이스의 변경된 데이터를 주기적으로 추출하여 구축기 내부의 임시 저장소에 적재하며, 적재된 데이터를 사용자가 지정한 규칙에 따라 변형한다. 또한 구축기는 적재 및 변형된 데이터를 일정 주기마다 일괄처리의 형태로 공간 데이터 웨어하우스 서버에 적재한다[4][12]. 공간 데이터 웨어하우스 서버는 변경된 데이터 적재 시 효율적인 적재를 위해 사용자 서비스를 정지한다[4]. 또한 공간 데이터 웨어하우스 서버는 사용자의 질의에 빠른 응답을 하기 위해 적재된 데이터로 색인을 구축한다.

색인 구축을 위한 기존 기법에는 벌크 삽입 기법과 색인 전송을 통한 색인 구축 기법이 있다. 벌크 삽입 기법은 삽입 데이터를 공간상에서 근접한 데이터끼리 클러스터링하여 여러 개의 클러스터를 생성한다[6]. 이 기법은 데이터를 클러스터링 하는 비용이 크며, 기존 R-Tree 노드들과 새로 삽입되는 Small Tree 노드들 간의 겹침이 크다[5][7]. 따라서 이 기법은 삽입과 검색 성능이 떨어지는 단점이 있다. 색인 전송을 통한 색인 구성 기법은 기 구축된 색인의 재활용을 위해 색인 구조를 직접 전송하여 색인을 구축한다. 이 기법은 색인 전송을 통해 색인 구성을 위한 검색, 분할, MBR 재구축 비용을 제거하여 색인 구성 비용이 감소하지만 특정 테이블의 전체 색인 전송만 가능하기 때문에 주기적인 소스 데이터의 변경이 발생하는 공간 데이터 웨어하우스의 색인 재구축 기법으로는 적절하지 않다[18].

따라서, 본 논문에서는 공간 데이터 웨어하우스에서 부분 색인을 이용한 효율적인 색인 재구축 기법을 제안한다. 제안 기법은 대량의 공간 데이터에 대한 색인 재구축을 위해 부분 색인을 구성하여 전송 및 기록함으로써 재구축 비용을 최소로 하는 기법이다. 구축기는 소스 데이터베이스로부터 추출한 데이터를 공간의 근접도가 아닌 기 구축된 색인의 구조에 맞게 클러스터링한다[1][14]. 기 구축된 색인의 구조에 맞게 생성된 각 클러스터는 부분 색인으로 구성되어 공간 데이터 웨어하우스 서버에 노드 단위로 전송된다. 공간 데이터 웨어하우스 서버는 전송된 부분 색인을 저장하기 위해 물리적으로 연속된 공간을 예약하고 예약된 공간에 부분 색인을 기록한다. 그 후, 공간 데이터 웨어하우스 서버에 기록된 부분 색인을 기 구축된 색인에 삽입함으로써 색인 재구축을 위한 검색, 분할, 재조정 비용이 감소한다. 따라서, 공간 또는 비공간 데이터의 색인 재구축 비용 절감을 위한 다수의 트리 계열 색인 구조에 적용할 수 있으며, 특히 공간 객체 삽입 시 부모 노드 엔트리의 재구축 비용이 높은 R-Tree 계열 색인의 재구축 비용 절감에 효율적이다[7].

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로 다차원 색인 구조의 벌크 로딩 기법, 벌크 삽입 기법 및 색인 전송을 통한 색인 구성 기법에 대해 설명하고, 3장에서는 본 논문에서 제안하는 부분 색인을 이용한 색인 재구축 기법을 설명한다. 4장에서는 제안 기법의 성능평가를 한다. 마지막으로, 5장에서는 결론 및 향후 연구를 기술한다.

2. 관련 연구

본 장에서는 색인 구축을 위한 기존 기법인 다차원 색인 구조의 벌크 로딩 기법, 벌크 삽입 기법, 색인 전송을 통한 색인 구성 기법에 대한 관련 연구를 설명한다.

2.1 다차원 색인 구조의 벌크 로딩 기법

다차원 객체에 대하여 검색 성능을 높이기 위한 Hilbert 정렬 알고리즘은 다차원 데이터에 대한 질의의 대부분이 영역 질의(range query) 임에 착안하여 널리 분포되어 있는 객체들을 영역에 대한 정렬 과정을 통해 데이터를 packing함으로써 검색 성능을 개선하였다[2][3][10][11]. Hilbert 정렬 알고리즘은 차원분열도형을 이용한 Z-order, bit-interleaving order, Hilbert curve 등 다양한 space filling curve 중 클러

스터링 효과가 가장 뛰어난 Hilbert curve를 이용하여 Hilbert 값에 따라 주어진 객체의 MBR을 정렬한 후 R-tree를 bottom-up 방식으로 구성하게 된다[2][11].

Hilbert 정렬 알고리즘은 다차원 색인 중 영역 질의에 보다 효율적인 R-Tree를 구성하기 위해 Hilbert 값에 따라 단말 노드를 먼저 정렬한 후 보다 상위 레벨로 정렬 과정을 연속해서 수행하게 된다.

이러한 packing은 공간 객체의 특성을 고려하여 객체의 디클러스터링(declustering) 효과를 높임으로써 R-tree의 공간 이용률을 높이며, 이로 인해 사용자 질의에 대한 보다 빠른 응답시간을 제공한다[11]. 그러나 이와 같은 정렬 알고리즘을 이용한 벌크 로딩 기법은 데이터를 정렬하기 위해 시간을 너무 많이 소요하며 특정 차원에 대한 정렬 기법만을 고려했기에 다차원 데이터에 대한 정렬의 경우 겹침 영역이 증가하여 검색에 대한 성능이 현저히 떨어지는 문제점을 지니고 있다.

2.2 벌크 삽입 기법

벌크 삽입 기법은 R-Tree 기반에서 대량의 공간 데이터 삽입 시 색인 재구축 비용을 줄이기 위한 기법이다. 삽입된 대량의 공간 데이터는 공간상에서 근접한 데이터끼리 클러스터링된다[1][6].

벌크 삽입 기법은 클러스터를 생성하기 위해 삽입되는 공간 데이터의 특성에 따라 적절한 파라미터인 클러스터의 최대 개수(K), 클러스터 간의 최소 거리(C), 클러스터와 공간 데이터 간의 최대 범위(R), 클러스터의 최대(fmax), 최소(fmin) 공간 데이터 개수를 설정한다. 처음 들어온 K개의 데이터는 클러스터가 가질 수 있는 최대 개수인 K개의 클러스터를 생성한다. 생성된 각 클러스터들 사이의 거리가 클러스터 간의 최소 거리인 C보다 작은 클러스터들은 하나의 클러스터로 합친다. 다음 공간 데이터 입력 시 클러스터의 중심과 공간 데이터의 중심을 계산하여 공간 데이터의 중심에서 가장 가까운 클러스터를 선택한다. 선택된 클러스터의 중심과 삽입된 공간 데이터의 중심이 클러스터와 공간 데이터 간의 최대 범위인 R보다 작으면 공간 데이터를 선택된 클러스터에 삽입한다. 클러스터의 중심을 재계산하고 클러스터 간의 거리가 C보다 작은 것은 하나의 클러스터로 합친다. 삽입된 모든 공간 데이터를 클러스터링한 후 클러스터에 포함되지 못한 공간 데이터와 공간 데이터의 개수가 fmin보다 작은 클러스터의 공간 데이터는 열외자 리스트에 삽입한다.

공간 데이터의 클러스터링을 통해 생성된 각 클러스터는 R-Tree의 일반적인 삽입 알고리즘으로 Small Tree로 구성된다[5][7]. 색인의 특성상 모든 단말 노드에서 높이가 같아야 하기 때문에 Small Tree의 삽입 가능한 높이가 계산되며, 계산된 높이에 있는 노드 중 Small Tree를 삽입 하기 위한 적절한 노드는 R-Tree의 일반적인 검색 알고리즘으로 검색한다[7]. 선택된 노드에 빈 엔트리가 있다면 부분 색인은 빈 엔트리에 삽입된다. 하지만 선택된 노드에 빈 엔트리가 없다면 경험적인(heuristic) 방법을 사용하여 현재 노드에 빈 엔트리를 만들거나 Small Tree에 있는 공간 데이터를 R-Tree의 일반적인 삽입 알고리즘으로 기 구축된 색인에 삽입한다[5]. 또한 열외자 리스트에 있던 공간 데이터는 R-Tree의 일반적인 삽입 알고리즘에 의해 하나씩 삽입된다[6].

이 기법은 데이터를 클러스터링 하는 비용이 많이 들며, 기존 R-Tree 노드들과 새로 삽입되는 Small Tree 노드들 간의 겹침이 크다. 따라서 벌크 삽입 기법은 삽입 성능 및 검색 성능이 떨어지는 단점이 있다.

2.3 색인 전송을 통한 색인 구성 기법

색인 전송을 통한 색인 구성 기법은 분산 공간 데이터베이스 시스템에서 기 구축된 색인의 재활용을 위해 색인 구조를 직접 전송하여 색인을 구성한다[18]. 분산 공간 데이터베이스 시스템은 특정 노드로 집중되는 부하의 분산이나 가용성 및 안정성 제공을 위해 데이터 복제 기법을 사용한다. 근원 사이트에 특정 데이터와 데이터에 대한 색인이 구축되어 있으며 특정 데이터에 대한 질의가 급증하고 있다면 분산 공간 데이터베이스의 코디네이터는 근원 사이트에 특정 데이터를 목적 사이트로 전송 요청을 한다. 근원 사이트는 특정 공간 데이터에 대한 데이터를 전송하고 근원 사이트에 있는 기 구축된 색인을 재활용 하기 위해 색인을 목적 사이트로 전송한다.

색인 전송을 통한 색인 구축 기법은 노드를 페이지 단위로 전송, 기록함으로써 색인 구축 시 발생하는 검색, 분할 및 MBR 재조정 비용을 삭제하였다. 그리고, 근원 사이트에 있는 색인을 목적 사이트에 전송함으로써 생기는 물리적인 사상문제는 목적 사이트에 색인을 저장할 연속적인 저장공간을 예약함으로써 해결한다. 색인 전송을 통한 색인 구성 기법은 색인 구성

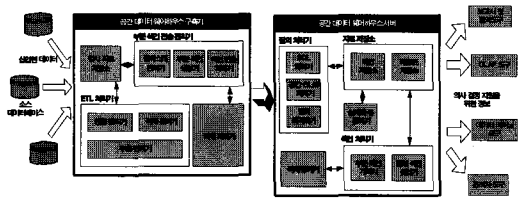
을 위한 검색, 분할, MBR 재조정 비용을 제거하여 색인 구성 비용을 줄이지만 주기적인 소스 데이터의 삽입에 대한 색인 재구축이 불가능하다. 따라서 이 기법은 공간 데이터 웨어하우스의 색인 재구축 기법으로는 적절하지 않다.

3. 부분 색인을 이용한 색인 재구축 기법

본 장에서는 부분 색인을 이용한 색인 재구축 기법을 제안한다. 첫째로 본 제안 기법의 환경이 되는 공간 데이터 웨어하우스 시스템에 대해서 설명한 후 이를 바탕으로 구축기에서 부분 색인 생성 기법과 구축기에서 공간 데이터 웨어하우스 서버로 부분 색인 전송 기법 및 공간 데이터 웨어하우스 서버에서 부분 색인 삽입 기법을 설명한다[19].

3.1 공간 데이터 웨어하우스 시스템 및 자료 저장소 구조

공간 데이터 웨어하우스 시스템은 소스 데이터베이스 및 구축기와 서버로 구성된다. 본 논문의 환경이 되는 공간 데이터 웨어하우스 시스템 구조는 <그림 1>과 같다.



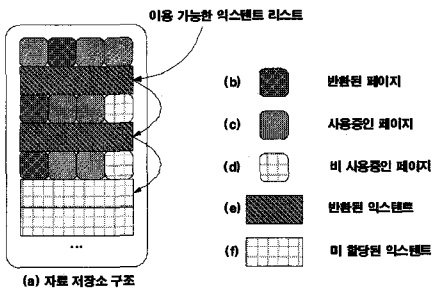
<그림 1> 공간 데이터 웨어하우스 시스템 구조

소스 데이터베이스는 지역 자치성을 가지며 이기종의 시스템이다. 또한 소스 데이터베이스의 데이터는 시간성을 가지기 때문에 데이터의 삽입 연산이 대부분이며, 갱신 및 삭제 연산은 자주 일어나지 않는다. 그로 인해 공간 데이터 웨어하우스 시스템은 갱신 및 삭제 연산의 효율적인 처리보다 시간성이 있는 데이터에 대한 삽입 연산을 효율적으로 처리 하는 기법이 필요하다.

구축기는 ETL 처리기의 추출 관리기를 통해 다수의 소스 데이터베이스의 삽입된 소스 데이터를 추출한다. 추출된 소스 데이터는 구축기의 변형 관리기에서 사용자가 지정한 규칙에 의해 변형 및 정제되며, 변형 및 정제된 데이터들은 임시 자료 저장소에 임시 적재된다. 또한 부분 색인 전송 관리기는 본 논문이 제안하는 부분 색인을 이용한 효율적인 색인 재구축 기법을 사용하여 임시 적재된 데이터로 부분 색인을 생성한다. 생성된 부분 색인은 임시 자료 저장소에 저장되어 있는 데이터와 함께 적재 처리기를 통해 공간 데이터 웨어하우스 서버로 전송된다.

공간 데이터 웨어하우스 서버는 적재 중계기를 통해 삽입된 소스 데이터 및 부분 색인을 전송 받는다. 색인 처리기의 부분 색인 기록기는 전송된 부분 색인을 공간 데이터 웨어하우스 서버의 자료 저장소에 기록한다. 부분 색인의 기록시 부분 색인 기록기는 부분 색인의 전송으로 인한 물리적인 사상문제를 해결한다. 색인 처리기의 전체 색인 삽입기는 자료 저장소에 저장되어 있는 부분 색인을 공간 데이터 웨어하우스 서버에 기 구축되어 있는 색인에 삽입을 담당한다. 자료 저장소의 레코드 저장소에는 부분 색인과 함께 전송된 레코드를 저장한다. 공간 데이터 웨어하우스 서버는 부분 색인의 삽입으로 인한 재구축된 색인을 이용하여 SQL 처리 및 공간 OLAP 연산을 지원한다.

제안 기법의 환경이 되는 구축기의 임시 자료 저장소와 공간 데이터 웨어하우스 서버의 자료 저장소 구조는 <그림 2>와 같으며 서로 같은 구조로 되어 있다.

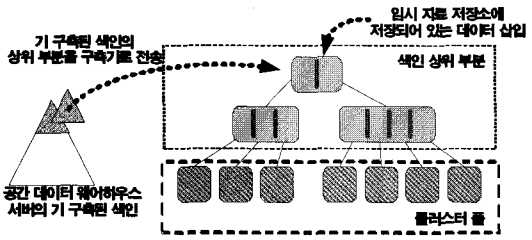


<그림 2> 자료 저장소 구조

<그림 2>의 (a)는 파일 입력과 출력의 기본 단위로 페이지를 사용하며, 4개의 페이지는 공간할당의 기본 단위인 익스텐트를 구성한다. 페이지 ID는 자료 저장소 ID와 페이지 오프셋으로 구성되며 페이지 오프셋은 자료 저장소 내 페이지의 위치정보를 나타내는 것으로 0부터 1씩 순차적으로 증가하는 값을 지니고 페이지 오프셋과 페이지 크기를 이용하여 실제 데이터에 접근하게 된다.

3.2 부분 색인 생성 기법

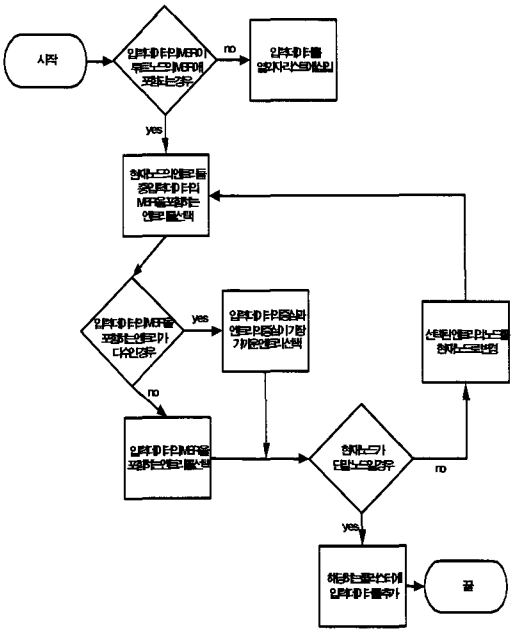
본 논문의 제안 기법은 소스 데이터베이스에 대량의 데이터가 삽입되었을 시 재구축을 위한 기법이며 공간 데이터 웨어하우스에서 부분 색인을 기 구축되어 있는 색인에 삽입하는 기법이다. 따라서 제안 기법은 공간 데이터 웨어하우스 서버에 기 구축되어 있는 색인이 없거나 기 구축되어 있는 색인의 높이가 3레벨 이상이 아니면 사용하지 않는다. 기 구축되어 있는 색인의 높이가 높지 않으면 대량의 데이터를 부분 색인으로 구성하였을 시 부분 색인이 기 구축된 색인의 높이와 같거나 더 높을 가능성이 높으며, 따라서 모든 단말 노드에서 루트 노드까지의 높이가 같아야 한다는 특성을 맞출 수 없다[7]. 기 구축되어 있는 색인이 없거나 높이가 3레벨 이상이 아니라면 색인의 일반적인 삽입 알고리즘으로 색인을 구축 및 재구축 한다. 이후에 기 구축된 색인의 높이가 3레벨 이상이 되었을 경우에 제안 기법을 사용하도록 한다.



<그림 3> 데이터 클러스터링 과정

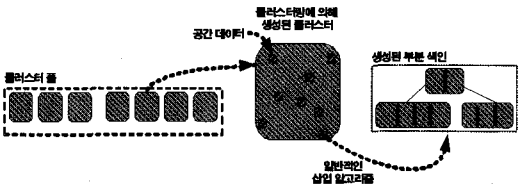
소스 데이터베이스에서 삽입된 데이터는 구축기에 의해 추출되어 임시 저장소에 적재된다. 구축기는 임시 적재된 데이터를 부분 색인으로 생성하기 위해 공간 데이터 웨어하우스 서버로부터 전송된 기 구축된 색인의 상위 부분의 유무를 확인한다. 기 구축된 색인의 상위 부분이 없다면, 구축기는 공간 데이터 웨어하우스 서버에게 기 구축된 색인의 상위 부분 전송 요구 메시지를 보낸다. 공간 데이터 웨어하우스 서버는 색인의 상위 부분 전송 요구 시, 주기적인 삽입 데이터의 개수와 기 구축된 색인의 높이를 통해 색인 상위 부분의 레벨을 결정한다. 색인 상위 부분의 레벨은 구축기에서 질의 응답에 필요한 모든 데이터 개수 중 10%를 가질 수 있는 레벨로 결정한다. 여기서 데이터 자체를 포함하는 비율이 아니라 데이터 개수만 고려한 비율이다. 그리고 데이터 개수를 포함하는 최적의 비율 결정은 4.2.1절의 실험평가에 나타나 있다. 결정된 색인의 상위 부분은 구축기로 전송되며 구축기는 전송된 색인의 상위 부분을 이용하여 임시 자료 저장소에 저장되어 있는 데이터를 <그림3>과 같이 클러스터링한다.

기 구축된 색인 구조를 고려한 클러스터링 기법은 기존의 공간 근접도에 따른 클러스터링 방법이 아닌 이미 구축되어 있는 색인의 구조에 맞추어 클러스터링 하는 기법이다. 공간 근접도에 따른 클러스터링 기법은 기 구축된 색인의 구조를 고려하지 않아 색인의 영역이 확장될 가능성이 크며, 과정이 복잡하여 클러스터링 비용이 많이 든다. 하지만 기 구축된 색인의 구조를 고려한 클러스터링 기법은 색인의 상위부분에 데이터를 삽입하여 클러스터링하기 때문에 기존의 클러스터링 기법보다 노드간의 겹침이 줄어들며 클러스터링 비용을 줄일 수 있다. 기 구축된 색인의 구조를 고려한 클러스터링 과정은 <그림 4>와 같다.



<그림 4> 기 구축된 색인의 구조를 고려한 클러스터링 과정

기 구축된 색인 구조를 고려하여 클러스터링한 공간 데이터는 클러스터 풀에 있는 해당 클러스터에 저장된다. 클러스터 풀에 저장되어 있던 각 클러스터는 <그림 5>와 같이 부분 색인으로 생성된다.



<그림 5> 부분 색인 생성 과정

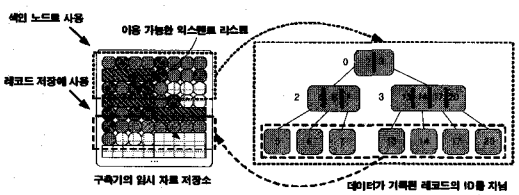
클러스터 풀에 저장되어 있던 각 클러스터는 클러스터가 포함하는 공간 데이터의 개수에 의해 부분 색인으로 구성되어야 할지 결정된다. 클러스터가 포함하는 공간 데이터의 개수가 적다면 클러스터를 부분 색인으로 생성, 전송하여 재구축 비용을 절감하는 것은 효율적이지 않기 때문에 클러스터에 있는 모든 공간 데이터를 열외자 리스트에 삽입한다. 클러스터가 포함하는 공간 데이터의 개수가 많은 클러스터는 일반적인 색인의 삽입 알고리즘으로 부분 색인을 생성한다.

3.3 부분 색인 전송 기법 및 부분 색인 기록 기법

부분 색인 전송 기법은 구축기에서 생성된 부분 색인을 전송하며, 전송된 부분 색인의 물리적인 사상문제를 해결하기 위하여 공간 데이터 웨어하우스 서버에서 물리적으로 연속된 저장 공간을 예약한다. 공간 데이터 웨어하우스 서버에서 연속적으로 예약된 저장 공간은 노드들의 ID인 페이지 오프셋 값을 예측 가능하게 하고, 중간노드 내의 엔트리들이 가지고 있는 자식노드 ID값을 결정함으로써 부모 노드 기록과 동시에 자식 노드 위치 정보를 기록하여 자식 노드의 위치 정보를 기록하기 위한 차후 부모 노드로의 접근 비용을 제거하였다. 본 제안기법은 색인 재구축 시 발생하는 검색 비용과 분할 비용 및 루트 노드로의 MBR 재조정 비용을 제거한다.

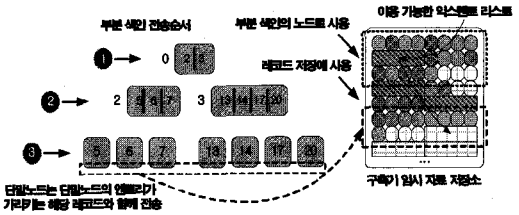
3.3.1 부분 색인 전송 기법

구축기에서 생성되어 공간 데이터 웨어하우스 서버에 전송될 부분 색인의 구조는 <그림 6>과 같다.



<그림 6> 구축기에서 생성된 부분 색인의 구조

구축기에서 10개의 노드로 구성된 색인 구조와 레코드가 <그림 6>과 같이 노드 ID의 연속성을 보장하지 않는 여러 개의 익스텐트 구조에 저장되어 있는 경우 구축기는 먼저 공간 데이터 웨어하우스 서버로 전송 할 노드 개수를 저장 공간 예약 메시지와 함께 전송한 후 구축기는 예약 완료 메시지를 수신하게 된다. 이후 부분 색인 전송 그림은 <그림 7>과 같다.



<그림 7> 구축기에서 색인 구조 전송순서

<그림 7>은 예약 완료 메시지를 수신하게 된 구축기에서 부분 색인 구조를 전송하는 그림이다. 부분 색인 구조는 ①~③의 순서로 전송하며 동일 레벨에서는 좌에서 우방향 순서로 전송한다. 이때 단말 노드 전송 시 구축기에서 사용하는 레코드 ID는 물리적 특성에 의해 공간 데이터 웨어하우스 서버에서 사용할 수 없기 때문에 노드 전송 시 단말 노드의 엔트리가 가리키는 해당 레코드와 단말 노드를 동시에 전송한다. 위 과정들을 알고리즘으로 표현하면 (알고리즘 1)과 같다.

```
Input
Node          : 전송해야 할 부분 색인의 루트 노드
Variables
RecordID      : 레코드의 물리적인 저장 위치

Algorithm SendPartialIndex( Node )
Begin
01 :  while( Type of Node is Internal )
02 :     if( Node is Root of partial index )
03 :        SendNode( Node )
04 :     Node = GetNextLevelNode( Node )
05 :     while( Node is not NULL )
06 :        SendNode( Node )
07 :        Node = GetNextRightNode( Node )
08 :     end while
09 :  end while
10 :  while( Node is not NULL )
11 :     SendNode( Node )
12 :     for( i = first recordID of the node to last recordID )
13 :        SendRecord( recordID )
14 :     end for
15 :     Node = GetNextRightNode( Node )
16 :  end while
End
```

(알고리즘 1) 구축기에서의 부분 색인 전송 알고리즘

(알고리즘 1)은 다음과 같이 수행된다. 1번째 줄에서 노드가 단말 노드가 아닌 루트 노드 및 중간 노드일 경우 반복한다. 2~3번째 줄에서는 루트 노드를 공간 데이터 웨어하우스로 전송하며, 4~8번째 줄에서는

현재 레벨에 있는 모든 노드를 전송한다. 10~16번째 줄에서는 모든 단말 노드를 순회하며 현재 노드 전송과 동시에 현재 노드의 엔트리가 가리키고 있는 레코드를 전송한다.

3.3.2 부분 색인 기록 기법

구축기에서 생성된 부분 색인은 부분 색인 전송 기법을 통해 공간 데이터 웨어하우스 서버에 전송된 후 부분 색인 기록 기법을 통해 서버에 기록된다. 서버에서 구축기로부터 전송된 부분 색인 기록 과정은 (알고리즘 2)와 같다.

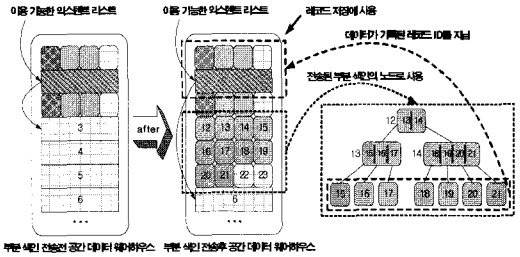
```
Input
NodeNum              : 전송되어진 노드의 개수
Node                 : 전송되어진 노드들의 루트 노드
Variables
FirstReservedNode    : 예약된 처음 노드
NodeIDtoWrite        : 예약된 저장 공간의 오프셋

AlgorithmWritePartialIndex( NodeNum, Node )
Begin
01 :   FirstReservedNode = ReserveStorageForPartialIndex( NodeNum )
02 :   NodeIDtoWrite = FirstReservedNode
03 :   while( Tpye of Node is Internal )
04 :      for( i = fisrt item# of the node To end item# )
05 :         ModifyChildID( Node.item[i] )
06 :      end for
07 :      WriteNode( Node, NodeIDtoWrite )
08 :      NodeIDtoWrite++
09 :      Node = NextNode( Node )
10 :   end while
11 :   while( Node is not NULL )
12 :      i = 0
13 :      while( Node.item[i] is not NULL )
14 :         ReceiveRecord( record )
15 :         recordID = WriteRcord( record )
16 :         Node.item[i].recordID = recordID
17 :         i++
18 :      end while
19 :      WriteNode( Node, NodeIDtoWrite )
20 :      NodeIDtoWrite++
21 :   end while
End
```

(알고리즘 2) 공간 데이터 웨어하우스 서버에서의 부분 색인 기록 알고리즘

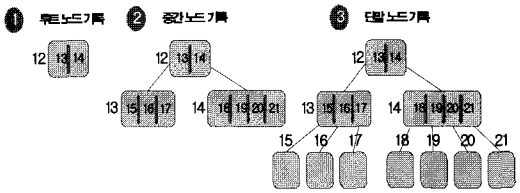
(알고리즘 2)는 다음과 같이 수행된다. 1번째 줄에서 전송 될 부분 색인의 노드 만큼 연속적인 공간을 예약한다. 2번째 줄에서 예약된 저장 공간 중에서 첫 번째 노드가 저장 될 주소를 지정한다. 3번째 줄에서 전송 된 노드가 단말 노드가 아닌 루트 노드이거나 중간 노드이면 반복한다. 4~6번째 줄에서 자식 노드들의 ID를 예상하여 현재 노드의 엔트리를 수정한다. 7번째 줄에서 수정 된 노드를 저장 공간에 기록한다. 11번째 줄에서 노드가 NULL이 아닐 경우 반복한다. 13번째 줄에는 노드의 엔트리가 NULL이 아닐 경우 반복한다. 14~16번째 줄은 단말 노드와 함께 전송 된 레코드를 받아서 저장 공간에 기록한 후 저장 공간에 기록된 주소를 노드의 엔트리가 기록한다. 19번째 줄에서 수정된 단말 노드를 기록한다.

공간 데이터 웨어하우스 서버는 구축기에서 전송된 부분 색인을 기록한다. 하지만 전송 이전의 중간 노드 내 기록된 자식 노드 ID는 물리적 저장구조의 특성에 의해 전송 후 공간 데이터 웨어하우스 서버의 자료 저장소에서는 사용할 수 없다. 물리적 사상 문제를 해결하기 위해 공간 데이터 웨어하우스 서버는 미 할당된 연속적인 익스텐트를 필요한 색인 노드 개수만큼 예약한다.



<그림 8> 전송되어 기록된 부분 색인의 구성

<그림 8>과 같이 연속적으로 예약된 익스텐트는 노드들의 ID인 페이지 오프셋 값을 예측 가능하게 하고, 중간노드 내의 엔트리들이 지니고 있는 자식노드 ID 값을 결정한다. 따라서 자식노드 ID 기록을 위한 부모노드로의 연속적인 차후 접근 비용은 제거된다. 구축기에서 공간 데이터 웨어하우스 서버로의 부분 색인 전송 과정 중 자식 노드 ID 기록 과정은 <그림 9>와 같다.



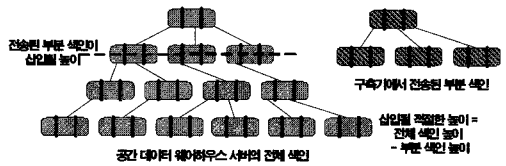
<그림9> 자식 노드 ID 기록 과정

자식 노드 ID의 기록은 <그림9> ①~③의 순서로 이루어진다. 즉 루트 노드에서 단말 노드 방향으로 기록하며 동일 레벨에서는 왼쪽에서 오른쪽 방향으로 기록한다. 전송된 노드의 엔트리들이 지니는 자식 노드 ID는 예약된 페이지 오프셋 값 중 제일 작은 값부터 큰 값 순으로 기록되기 때문에 각 노드 내 엔트리

의 기록 과정 중 자식 노드에 대한 ID인 페이지 오프셋 값을 예상할 수 있다. 이로 인해 자식 노드 기록을 위한 부모 노드로의 차후 접근 비용은 제거된다.

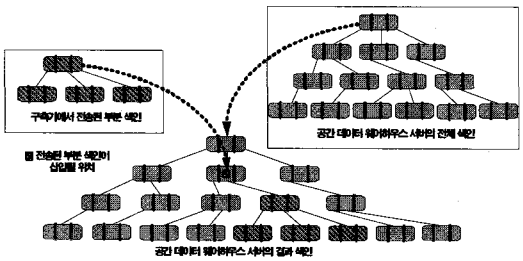
3.4 부분 색인 삽입 기법

공간 데이터 웨어하우스 서버는 구축기에서 전송된 부분 색인을 기 구축되어 있던 전체 색인에 삽입 한다. 부분 색인을 삽입하기 위해 적절한 노드를 찾는 방법은 일반적인 색인의 검색 방법과 동일하다. 하지만 색인의 특성상 모든 단말 노드에서 높이가 같아야 하기 때문에 <그림 10>과 같이 부분 색인의 삽입 가능한 높이를 계산한다.



<그림 10> 부분 색인 삽입 가능 높이 계산

구축기에서 전송된 부분 색인을 삽입하기 위해 전체 색인 높이와 부분 색인 높이의 차이를 계산하여 삽입될 적절한 높이를 결정한다. 결정된 높이에 있는 노드 중 부분 색인이 삽입될 적절한 노드는 일반적인 검색 방법을 통해 선택되며 부분 색인은 <그림 11>과 같이 선택된 노드의 엔트리에 삽입된다.



<그림 11> 공간 데이터 웨어하우스 서버에서
부분 색인 삽입 과정

선택 된 노드에 빈 엔트리가 있는 경우에는 부분 색인은 빈 엔트리에 삽입된다. 하지만 선택 된 노드에 빈 엔트리가 없다면 부분 색인은 경험적인 방법을 사용하여 삽입된다[7]. 위 과정들을 알고리즘으로 표현하면 (알고리즘 3)과 같다.

```
Input
IndexNode            : 기 구축되어 있는 색인의 루트 노드
PartialIndexRoot     : 부분 색인의 루트 노드
DiffHeight           : 기 구축되어 있는 색인의 높이와 부분 색인의 높이 차이
Variables
RootofExistingIndex  : 기 구축되어 있는 색인의 루트 노드
numChildren          : 해당 노드가 포함하는 엔트리 개수
S1, S2               : 노드가 가지는 엔트리 중 가장 근접한 형제 노드
Child                : 현재 노드의 엔트리 중 노드의 중심과 가장 먼 엔트리의 노드
FANOUT               : 노드가 포함할 수 있는 전체 엔트리 개수

Algorithm InsertPartialIndex( IndexNode, PartialIndexRoot, DiffHeight )
Begin
01 :  if( DiffHeight > 1 )
02 :     DiffHeight--
03 :     IndexNode = SelectNextLevelNode( RootofExistingIndex, IndexNode, PartialIndexRoot )
04 :     InsertPartialIndex( IndexNode, PartialIndexRoot, DiffHeight )
05 :  else
06 :     if( NumChildren( IndexNode ) < FANOUT )
07 :        IndexNode.child[numChildren++] = PartialIndexRoot
08 :     else
09 :        ( S1, S2 ) = SelectSibling( IndexNode )
10 :        if( not MergeSibling( S1, S2 ) )
11 :           Child = SelectFurthestChild( IndexNode )
12 :           if( not CompareSelectCost( Child, PartialIndexRoot ) )
13 :              if( not ReInsertOBO( Child ) )
14 :                 SplitNodeQuadratic( PartialIndexRoot, IndexNode )
15 :           else
16 :              InsertOBO( PartialIndexRoot )
17 :        else
18 :           InsertPartialIndex( PartialIndexRoot )
19 :  ResizeMBR( IndexNode )
End
```

(알고리즘 3) 부분 색인 삽입 알고리즘

(알고리즘 3)은 다음과 같이 수행된다. 1~4번째 줄에서 부분 색인이 삽입될 적절한 높이(기 구축된 색인의 높이-부분 색인의 높이)에 있는 노드 중 부분 색인이 삽입될 적절한 노드를 검색한다. 6~7번째 줄에서는 부분 색인이 삽입될 노드에 빈 엔트리가 있다면 부분 색인을 삽입한다. 8~18번째 줄에서는 부분 색인이 삽입될 노드에 빈 엔트가 없는 경우 경험적인 방법을 사용하여 부분 색인을 삽입한다. 9~10번째 줄에서는 선택된 노드의 엔트리 노드 중 가장 근접한 두 개의 형제 노드를 선택하며 선택된 두 개의 형제 노드를 합친다. 11~12번째 줄에서는 현재 노드의 엔트리의 노드 중 현재 노드의 중심에서 가장 먼 노드를 선택하여 선택된 노드가 부분 색인과 비교하여 검색 성능이 좋지 않다면 선택된 엔트리의 노드를 OBO(OneByOne) 방식으로 재 삽입한다[7]. 14번째 줄에서는 일반적인 분할 알고리즘으로 노드를 분할한 후 부분 색인을 삽입한다. 16번째 줄에서는 부분 색인의 검색 성능이 좋지 않기 때문에 부분 색인을 OBO 방식으로 삽입한다.

4. 성능평가

본 장에서는 대량의 공간 데이터에 대한 일반적인 삽입 방식인 OBO와 대량의 공간 데이터에 대한 효율적인 색인 재구축 기법인 벌크 삽입 기법 및 본 논문에서 제안하는 부분 색인을 이용한 색인 재구축 기법에 대한 성능평가를 하였다[4][7].

4.1 실험 환경

본 평가에서 비교할 부분 색인을 이용한 색인 재구축 기법과 OBO 기법 및 벌크 삽입 기법은 데이터베이스 연구실에서 개발한 공간 데이터 웨어하우스 시스템에서 구현되었다. 그리고, 공간 데이터 웨어하우스 내부의 시스템에 사용된 환경은 <표 1>과 같다.

<표 1> 시스템 환경

기종	IBM PC 호환
CPU	Pentium IV 3GHz
메인 메모리	1GB
하드 디스크	120GB, 7200RPM 버퍼8M, ATA방식
운영 체제	Windows XP Professional
개발 언어	Visual C++ 6.0
네트워크 환경	Category 5 cable

실험에 사용된 공간 데이터는 TIGER/Line 개체 데이터 중 약 200만개를 추출하여 사용하였다. TIGER/Line 개체 데이터는 공간 데이터베이스 분야에서 널리 사용되는 표준 벤치마크 데이터이다[17].

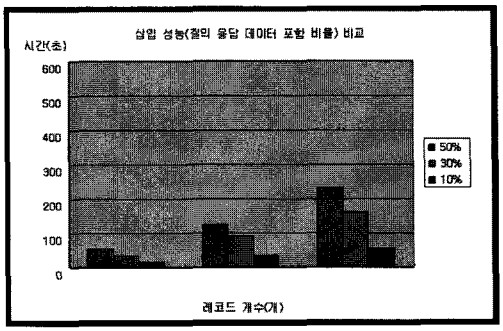
4.2 실험평가

본 실험평가에서 클러스터를 만들기 위한 기 구축된 색인의 상위 부분은 질의 응답에 필요한 데이터 개수를 포함하는 비율에 따른 삽입 성능 비교를 제외하고 삽입 성능 평가를 위한 디스크 I/O와 소요 시간을 실험할 때 상위 2레벨을 이용하였다.

4.2.1 삽입 성능평가

본 절에서는 삽입 성능을 평가하기에 앞서 구축기에서 색인의 상위 부분 요청 시 공간 데이터 웨어하우스 서버에서 전송하는 최적의 레벨을 결정하기 위해 질의 응답에 필요한 데이터 개수를 포함하는 최적의 비율을 결정하는 실험평가를 하였고, 삽입 성능을 평가하기 위해 일반적인 삽입 방식인 OBO 방식과 벌크 삽입 기법과 제안 기법의 실험평가를 하였다. 실험평가의 성능은 구축기에 전송되는 색인의 상위부분에서 질의 응답에 필요한 데이터 개수를 가지는 비율에 따른 색인의 재구축 시간과 소스 데이터의 삽입에 대한 색인 재구축 시 디스크 I/O와 소요되는 시간으로 평가하였다.

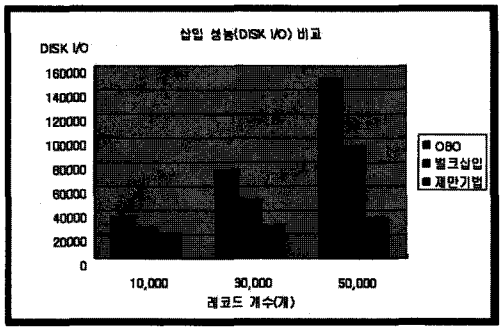
실험은 공간 데이터 웨어하우스 서버에서 100만개의 데이터를 갖는 색인을 OBO 방식을 통해 구성한 후 소스 데이터베이스에 다양한 개수의 공간 데이터를 삽입하였다. 삽입 된 공간 데이터는 구축기에 의해 추출되어 구축기의 임시 자료 저장소에 임시 저장된다. OBO 방식은 임시 저장된 데이터를 공간 데이터 웨어하우스 서버에 전송하여 구축기에서 전송된 데이터를 일반적인 삽입 알고리즘으로 삽입하여 실험하였다[7]. 벌크 삽입 기법은 구축기에 임시 저장된 데이터를 공간 데이터 웨어하우스 서버에 전송하며, 공간 데이터 웨어하우스 서버에서는 전송된 데이터를 공간상의 근접도로 클러스터링 하여 Small Tree를 생성 및 삽입하여 실험하였다[1][5][6]. 제안 기법은 임시 저장된 데이터를 구축기에서 공간상의 근접도가 아닌 기 구축된 색인의 구조에 맞게 클러스터링하며, 생성된 각 클러스터를 부분 색인으로 생성 및 전송한다. 공간 데이터 웨어하우스 서버에서는 전송된 부분 색인을 기 구축된 색인에 삽입하여 실험하였다. 이때 구축기에서 색인의 상위 부분 요청 시 공간 데이터 웨어하우스 서버에서 기 구축된 색인의 상위 부분을 구축기로 한번만 전송한다. 따라서 공간 데이터 웨어하우스 서버에서 색인의 상위 부분을 구축기로 전송하기 위한 삽입 비용과 시간 비용은 본 실험평가에 크게 영향을 미치지 않는다.



<그림 12> 삽입 성능(질의 응답 데이터 포함 비율) 비교

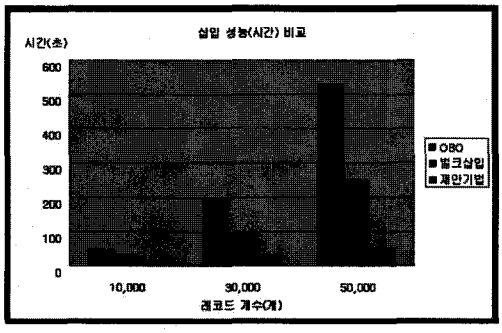
<그림 12>는 공간 데이터 웨어하우스 서버에서 구축기로 색인의 상위 부분 전송 시 최적의 레벨을 결정하기 위해 구축기에서 질의 응답에 필요한 데이터 개수를 가지는 비율에 따라 데이터 전송과 색인 재구축 시간을 평가한 결과이다. <그림 12>와 같이 질의 응답에 필요한 데이터 개수를 10% 가지는 레벨을 전송할 때 성능이 좋은 것을 볼 수 있다. 그 이유는 질의 응답에 필요한 데이터 개수를 포함하는 비율이 커지면 구

축기에서 재구축 된 부분 색인의 높이가 공간 데이터 웨어하우스 서버에서 기 구축되어 있는 색인의 높이와 같거나 더 높을 가능성이 크기 때문에 모든 단말 노드에서 루트 노드까지의 높이가 같아야 한다는 특성을 맞출 수 없고 색인 전송 비용도 커지기 때문이다.



<그림 13> 삽입 성능(DISK I/O) 비교

<그림 13>은 OBO, 벌크 삽입 기법 및 제안 기법의 삽입 실험을 DISK I/O로 평가한 결과이다. <그림 13>과 같이 제안기법은 OBO와 벌크 삽입 기법보다 성능이 월등한 것을 볼 수 있다. 또한 소스 데이터베이스에 삽입되는 데이터의 양이 많아지면 더욱 성능이 향상되는 것을 볼 수 있으며, 데이터 50000개 삽입 시 제안기법은 OBO 보다 80% 삽입 성능이 좋아지며, 벌크 삽입 기법 보다 45% 삽입 성능이 좋아진다.



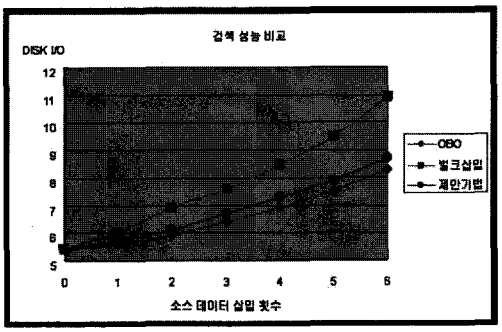
<그림 14> 삽입 성능(시간) 비교

<그림 14>는 데이터 전송과 색인 재구축 시간을 OBO, 벌크 삽입 기법 및 제안 기법으로 평가한 결과이다. <그림 14>와 같이 제안기법의 삽입 시간이 OBO와 벌크 삽입 기법의 삽입 시간보다 월등히 작은 것을 알 수 있다. 기존 기법은 구축기 내부에서 색인을 재구축하지 못하고 소스데이터 모두를 일일이 공간 데이터 웨어하우스 서버에 전송하여 색인에 삽입하기 때문에 소요 시간이 많지만 제안 기법은 구축기 내부에서 클러스터링 및 부분 색인을 생성하기 때문에 부분 색인 전송 시간과 공간 데이터 웨어하우스 서버에서 전송된 부분 색인 기록 시간 및 부분 색인을 기 구축된 색인에 삽입하는 시간 비용만 필요하므로 DISK I/O 평가보다 소요 시간 평가가 월등히 뛰어나다. 따라서 제안 기법은 레코드의 개수가 많아질수록 성능이 좋아지며, 50000개의 데이터 삽입 시, OBO 방식보다는 90%, 벌크 삽입 기법보다는 80% 시간 절약 효과를 보인다.

4.2.2 검색 성능평가

본 절에서는 제안 기법이 주기적으로 공간 데이터의 삽입 이후에도 효과적인 질의 처리 성능을 보이는지 검증하기 위한 실험을 수행하였다. 색인은 삽입을 빠르게 하는 것도 중요하지만 질의 처리를 효율적으로 하기 위한 것이다. 특히 공간 데이터 웨어하우스처럼 주기적으로 공간 데이터의 삽입이 일어나는 환경에서는 반복적인 삽입 이후의 검색 성능이 중요하다.

본 실험에서는 공간 데이터 웨어하우스 서버에서 100만개의 데이터를 갖는 색인을 OBO 방식을 통해 구성한 후 소스 데이터베이스에 10만개의 공간 데이터를 총 6번 삽입하여 삽입 후의 질의 처리 비용을 측정하였다. 질의 처리 비용은 색인의 검색 성능이 디스크 I/O에 가장 많은 영향을 받기 때문에 5000번의 임의 포인트 질의를 수행한 후 평균 디스크 I/O를 측정하였다.



<그림 15> 검색 성능 비교

<그림 15>와 같이 제안 기법은 OBO 방식과는 비슷한 검색 성능을 보이며, 벌크 삽입 기법보다는 향상된 검색 성능을 보인다. 벌크 삽입 기법은 공간상의 근접도에 따라 클러스터링을 하기 때문에 기 구축된

색인과 생성된 Small Tree와의 겹침이 크다. 따라서 포인트 질의 시 벌크 삽입 기법은 다른 여러 노드를 순회할 가능성이 크다. 하지만 제안 기법은 공간상의 근접도가 아닌 기 구축된 색인의 구조에 맞게 클러스터링하여 겹침이 크지 않기 때문에 OBO 방식과 비슷한 검색 성능을 보인다.

5. 결론 및 향후 연구

본 논문은 구축기에서 추출된 데이터를 색인의 구조에 맞게 클러스터링하며, 생성된 각 클러스터로 부분 색인을 생성 및 전송하는 기법을 제안하였다. 공간 데이터 웨어하우스 서버에서는 부모 노드와 자식 노드와의 물리적 사상 문제를 해결하기 위해 물리적으로 연속된 공간을 예약하고 예약된 공간에 부분 색인을 기록한다. 기록된 부분 색인은 기 구축된 색인에 일반적인 삽입 알고리즘으로 삽입된다. 따라서 제안 기법은 색인 재구축 시 발생하는 데이터 분할 비용, 특히 공간 데이터의 경우 각 데이터의 삽입과정 중 발생하는 단말 노드로부터 루트 노드까지의 MBR 재조정 비용을 최소화한다. 또한 제안 기법은 클러스터링 과정을 구축기에서 처리함으로써 공간 데이터 웨어하우스 서버의 부하를 최소로 줄인다. 따라서 제안 기법은 공간 데이터 웨어하우스에서 소스 데이터베이스의 주기적인 대량의 데이터의 삽입에 대해 효율적인 삽입 성능을 보이며, 검색 성능은 OBO 방식과 비슷한 성능을 보임을 알 수 있다.

제안 기법은 공간 또는 비공간 데이터의 색인 재구축 비용 절감을 위해 다수의 트리 계열 색인 구조에 적용할 수 있으며 특히, 공간 객체 삽입 시 부모 노드 엔트리의 재구축 비용이 높은 R-Tree 계열 색인의 재구축 비용 절감에 보다 효율적이다.

향후 연구로서, 기 구축된 색인의 상위 부분 레벨 계산 방법에 관한 연구와 검색 성능을 높이기 위해 재구축된 색인의 노드들 간의 겹침을 줄이는 색인 재 포장 기법에 관한 연구가 있다. 또한 공간 데이터 웨어하우스 시스템에서 시간성을 가진 대량의 데이터가 주기적으로 삽입되지만 저장 공간의 한계로 인해 삽입된 데이터 중 오래된 데이터는 백업하며 기 구축된 색인에서 대량의 삭제 과정이 필요하다. 따라서 색인에서 대량의 데이터에 대한 삽입과 삭제를 효율적으로 할 수 있는 연구가 필요하다.

참고문헌

[1] M.R. Anderberg, Probability and Mathematical Statistics, Academic Press, New York, San Francisco, London, 1973, pp. 132-147.

[2] T. Bially, "Space-filling curves: Their generation and their application to bandwidth reduction," IEEE Trans. On Information Theory, IT-15(6), 1969, pp. 658-664.

[3] Christian B?hm, Hans-Peter Kriegel, "Efficient Bulk Loading of Large High-Dimensional Indexes," Proc. Int. Conf. on Data Warehousing and Knowledge Discovery (DaWaK'99), Florence, Italy, 1999.

[4] S. Chaudhuri, U. Dayal "An Overview of Data Warehousing and OLAP Technology," SIGMOD Record, Vol. 26, No. 1, 1997.

[5] L. Chen, R. Choubey, and E. A. Rundensteiner, "Bulk-insertions into R-trees using the small-tree-large-tree approach," ACM GIS, 1998, pp. 161-162.

[6] R. Choubey, L. Chen, and E. A. Rundersteiner, "GBI: A Generalized R-tree Bulk-Insertion Strategy," Advances in Spatial Databases, 1997, pp. 91-108.

[7] A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," Proc. ACM SIGMOD, 1984, pp. 47-57.

[8] W.H. Inmon, What is a Data Warehouse?, Prism Solutions Tech Topics, Vol. 1, No. 1, 1995, pp. 32-37.

[9] W. H. Inmon, Building the Data Warehouse, 2nd Ed. John Wiley & Sons. Inc, 1996, pp. 66-67.

[10] I. Kamel and C. Faloutsos, "Hilbert R-tree: An Improved R-tree Using Fractals," Proc. 20th VLDB Conf., Santiago, Chile, 1994, pp. 500-509.

[11] Ibrahim Kamel, Christos Faloutsos, "On Packing R-tree," Proceedings of International Conference on Information and Knowledge Management, 1993, pp. 490-499.

[12] R. Kimball, L. Reeves, M. Ross, and W. Thornthwaite, The Data Warehouse Lifecycle Toolkit, John Wiley, 1998, pp. 18-21.

[13] Lafond, "Designing and Building the Distributed Geospatial Data Warehouse Architecture," In Proc. Twelfth Annual Symposium, Toronto, 1998.

[14] Taewon Lee, Bongki Moon, and Sukho Lee, "Bulk Insertion for R-tree by Seeded Clustering," DEXA2003, 2003, pp. 129-138.

[15] Eric Sperley, The Enterprise Data Warehouse: Planning, Building, and Implementation, Prentice Hall PTR, 1999, pp. 8-15.

[16] P. Vassiliadis, A. Simitsis, and S. Skiadopoulos, "Conceptual Modeling for ETL processes," In Proc. DOLAP, 2002, pp. 14-21.

[17] TIGER/Line Files, 2000 Technical Documentation, U.S. Bureau of Census, Washington DC, accessible via URL http://www.census.gov/geo/www/tiger/tigerua/ua_tgr2k.html.

[18] 박상근, 김호석, 이재동, 배해영, "분산 데이터베이스 시스템에서의 색인 구성비용 절감을 위한 효율적인 색인 전송 기법," 한국정보과학회 추계학술대회 논문집, 제30권 제2호, 2003, pp. 223-225.

[19] 정영철, 유병섭, 박순영, 이재동, 배해영, "공간 데이터 웨어하우스에서 부분 색인 전송을 이용한 효율적인 색인 재구성 기법," 한국정보처리학회 춘계학술대회 논문집, 제12권 제1호, 2005, pp. 39-42.

[20] 최유신, 유병섭, 박순영, 배해영, "공간 데이터 웨어하우스 구축기에서 사실 테이블 사전 계산 기법," 한국정보처리학회 추계학술대회 논문집, 제11권 제2호, 2004, pp. 165-168.

곽동욱
2004년 인하대학교 졸업(공학사)
2004년~현재 인하대학교 대학원 컴퓨터 · 정보공학과 (석사과정)
관심분야 : 공간 데이터 웨어하우스, 공간 데이터베이스, 센서 네트워크, RFID 미들웨어

정영철
2003년 인하대학교 컴퓨터공학부 (공학사)
2005년 인하대학교 컴퓨터 · 정보공학과 (공학석사)
2005년~현재 LG CNS 금융솔루션 개발팀
관심분야 : 공간 데이터 웨어하우스, LBS, 분산 데이터베이스

유병섭
2002년 인하대학교 컴퓨터공학부 (공학사)
2004년 인하대학교 컴퓨터 · 정보공학과 (공학석사)
2004년~현재 인하대학교 대학원 컴퓨터 · 정보공학과 (박사과정)
관심분야 : 공간 데이터 웨어하우스, XML, GML, 클러스터 시스템

김재홍
1985년 인하대학교 전자계산학과 (공학사)
1990년 인하대학교 전자계산학과 (공학석사)
1994년 인하대학교 전자계산학과(공학박사)
1995년~현재 영동대학교 컴퓨터공학과 교수
관심분야 : 멀티미디어 데이터베이스, 이동 객체 데이터베이스, 지리정보시스템

배해영
1974년 인하대학교 응용물리학과 (공학사)
1978년 연세대학교 대학원 전자계산학과 (공학석사)
1989년 숭실대학교 대학원 전자계산학과(공학박사)
1985년 Univ. of Houston 객원교수
1992년~1994년 인하대학교 전자계산소 소장
1982년~현재 인하대학교 컴퓨터공학과 교수
1999년~현재 지능형GIS연구센터 센터장
2000년~현재 중국 중경우전대학교 대학원 명예교수
2004년~현재 인하대학교 정보통신대학원 원장
관심분야 : 분산 데이터베이스, 공간 데이터베이스, 지리정보 시스템, 멀티미디어 데이터베이스 등