Journal of Korea Multimedia Society Vol. 12, No. 10, October 2009(pp. 1386-1396)

공간 데이터웨어하우스에서 통합된 다차원 개념 계층 지원을 위한 데이터 큐브 색인

이동욱[†], 백성하[††], 김경배[†††], 배해영[††††]

요 약

공간 데이터 웨어하우스에서 의사 결정 지원을 위한 공간 데이터 큐브는 크기가 방대하기 때문에 이를 효율적으로 관리하고 질의 처리의 수행 속도를 높이기 위한 공간 데이터 큐브 색인 기법이 요구된다. 제안된 데이터 큐브 색인 기법들 중 Hierarchical Dwarf는 사실 테이블의 튜플 필드 값의 중복을 이용하여 큐브를 압축하여 저장 비용과 질의응답 속도 면에서는 우수하지만 공간 차원을 지원하지 않으며, OLAP-favored Search 기법은 R-tree기반으로 공간 차원에 대한 계층적 집계 값을 제공하고 공간 OLAP 연산을 지원하지만 공간 및 비공간 차원들을 통합한 의사결정을 지원하지 못한다. 본 논문에서는 통합된 다차원 개념 계층 지원을 위한 데이터 큐브 색인을 제안한다. 이는 개념 계층에 대한 정보와 사실 테이블에 저장된 튜플들을 참조하여 각각의 차원에 대해 생성된 개념 계층 트리들이 연결되어 통합된 색인이다. 이 때, 중복되는 개념 계층 트리가 존재할 경우 이를 공유함으로써 저장 비용을 줄인다. 특히 제안 기법은 공간 및 비공간 차원이 통합된 개념 계층 트리들을 사용하므로, 공간 및 비공간 차원에 대한 OLAP 연산 비용이 감소한다.

Data Cude Index to Support Integrated Multi-dimensional Concept Hierarchies in Spatial Data Warehouse

Dong Wook Lee[†], Sung Ha Baek[††], Gyoung Bae Kim[†††], Hae Young Bae[††††]

ABSTRACT

Most decision support functions of spatial data warehouse rely on the OLAP operations upon a spatial cube. Meanwhile, higher performance is always guaranteed by indexing the cube, which stores huge amount of pre-aggregated information. Hierarchical Dwarf was proposed as a solution, which can be taken as an extension of the Dwarf, a compressed index for cube structures. However, it does not consider the spatial dimension and even aggregates incorrectly if there are redundant values at the lower levels. OLAP-favored Searching was proposed as a spatial hierarchy based OLAP operation, which employs the advantages of R-tree. Although it supports aggregating functions well against specified areas, it ignores the operations on the aspatial dimensions. In this paper, an indexing approach, which aims at utilizing the concept hierarchy of the spatial cube for decision support, is proposed. The index consists of concept hierarchy trees of all dimensions, which are linked according to the tuples stored in the fact table. It saves storage cost by preventing identical trees from being created redundantly. Also, it reduces the OLAP operation cost by integrating the spatial and aspatial dimensions in the virtual concept hierarchy

Key words: Multi-dimensional Concept Hierarchy(다차원 개념 계층), Satial Data Warehouse(공간데이터웨어하우스), Data Cube, Index(데이터 큐브 색인)

※ 교신저자(Corresponding Author) : 김경배, 주소 : 충청북도 청주시 흥덕구 무심서로 241(361-742), 전화 : 043)299-8431, FAX : 043)299-8430, E-mail : gbkim@seowon.ac.kr
접수일 : 2009년 3월 16일, 수정일 : 2009년 7월 2일
완료일 : 2009년 7월 10일
[†] 준회원, 인하대학교 정보공학과 박사과정
(E-mail : dwlee@dblab.inha.ac.kr)
[††] 준회원, 인하대학교 정보공학과 박사과정
(E-mail : shbaek@dblab.inha.ac.kr)
[†††] 정회원, 서원대학교 컴퓨터교육과 조교수
[††††] 정회원, 인하대학교 정보공학부 교수
(E-mail : hybae@inha.ac.kr)
※ 본 연구는 건설교통부 첨단도시기술개발사업- 지능형 국토정보기술혁신 사업과제의 연구비지원(07국토정보C05)에 의해 수행되었음

1. 서 론

기업들의 정확하고 효율적인 의사결정을 위하여 급변하는 상황에 대한 데이터를 효율적으로 분석할 수 있는 정보 관리 시스템에 대한 수요가 증가하고 있다. 공급망 관리나 입지 선정과 같은 의사결정을 위해서는 지도 데이터와 같은 공간 정보를 포함해야 한다. 따라서 과거부터 현재까지의 공간 및 비공간 데이터를 통합 저장하고 이에 대한 공간 OLAP (On-Line Analytical Processing) 연산을 제공하는 공간 데이터 웨어하우스의 중요성이 대두되고 있다[1]. 공간 데이터 웨어하우스는 이질적인 데이터 소스로부터 추출된 공간/비공간 데이터를 하나의 저장소에 통합하고, 데이터 분석가에게 공간 OLAP 연산을 제공하여 의사결정을 지원한다[2,3]. 공간 데이터 웨어하우스에 통합 및 저장되어 있는 다차원 데이터는 효율적인 OLAP 연산을 위해 큐브의 형태로 관리된다[4]. 이는 다차원 데이터인 사실 테이블에 존재하는 수치 데이터에 대하여 다차원 집계 연산을 수행한 결과 값들을 저장하고 있는 일종의 실체화 뷰이다[5]. 이러한 공간 데이터 큐브에 개념 계층을 갖는 차원이 존재할 경우 사용자는 특정 차원에 대한 롤업 및 드릴다운 연산을 통해 원하는 수치에 해당하는 집계한 값을 요청한다[6,7]. 특히, 공간 데이터 웨어하우스에서는 공간 이러한 공의 공간 및 비공간 차원의 개념 계층을 통합하여 효율적으로 지원하는 색인이 필요하다[8].

본 논문은 공간 데이터웨어하우스에서 통합된 다차원 개념 계층 지원을 위한 데이터 큐브 색인 기법을 제안한다. 이는 공간 및 비공간 차원의 개념 계층 구조를 트리 형태로 표현한 개념 계층 트리로 구성되고, 각 차원의 개념 계층 트리는 사실 테이블에 저장된 튜플들에 따라 연결되어있어 모든 차원의 개념 계층을 내부에 포함하는 통합 색인이다. 개념 계층 트리는 상위 개념과 하위 개념의 종속관계를 트리 구조를 사용하여 직관적으로 표현한 것으로 각 노드는 동일한 상위 개념에 속하는 하위 개념 그룹의 값들을 각각 담고 있는 여러 개의 셀들로 구성된다. 색인 구축 시, 의미론적으로 동일한 개념 계층 트리에 연결되는 셀이 존재할 경우 이를 하나만 생성하여 공유함으로써 색인을 압축저장 한다. 색인이 구축된 후, 사실 테이블에 새로운 데이터가 적재되면 이에 대한 전처리 과정을 통해 임시 색인을 구축하여 이를 일괄적으로 갱신한다[9].

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로 기존의 데이터 큐브 색인 기법들에 대해 설명한다. 3장에서는 본 논문에서 제안하는 다차원 개념 계층을 위한 색인의 구조와 생성 및 검색 방법을 기술한다. 4장에서 기존의 데이터 큐브 색인과의 비교 성능평가 결과를 보이고, 5장에서 결론을 맺고 향후 연구에 대해 언급한다.

2. 관련연구

본 장에서는 기존의 데이터 큐브 색인 중 데이터 큐브의 셀들을 의미론적으로 분류하여 압축 저장하는 QC-tree와 비공간 차원의 개념 계층을 이용한 집계 연산을 지원하는 Hierarchical Dwarf 색인, 공간 차원의 개념 계층을 이용하여 시공간 데이터의 집계 값을 검색하는 OLAP-favroed Search에 대해 기술한다[10-12].

데이터 큐브를 압축하기 위해 동일한 의미를 같은 셀들의 집합인 여러 클래스로 나누어 이에 대한 정보만을 저장하는 Quotient Cube가 제안되었다[10]. 이는 같은 집계 값을 갖고 큐브 내에서 부모-자식 관계를 갖는 셀들을 분류한 것으로 의미론적 OLAP 연산을 제공하는 밀집된 큐브 구조이다. QC-tree는 이를 실제로 구현하기 위해 각 클래스의 상위경계 간에 포함관계와 공통이 되는 부분을 이용한 트리 색인이다. QC-tree는 분류된 클래스의 정보만을 저장하는 방식으로 데이터 압축을 통해 데이터 큐브의 저장 비용을 절감하였고, 질의 처리 속도, 갱신 속도 면에서도 비교적 좋은 성능을 보였다. 하지만 각 차원에 존재하는 개념 계층에 대해 고려되지 않아 Rollup/Drilldown과 같은 중요한 OLAP 연산 시에는 구축된 QC-tree를 사용할 수 없다는 문제점이 있다.

Hierarchical Dwarf는 이전에 제안된 Dwarf라는 큐브 색인에 개념 계층이 포함되도록 확장한 것이다[12]. Dwarf는 데이터 큐브의 중복되는 집계 값을 효율적으로 저장하기 위해 색인이 생성될 때 중복되는 중간 노드 및 집계 값을 하나만 생성하고 이를 공유하도록 하여 색인을 구성하는 기법이다[13]. 이러한 Dwarf의 구조와 장점을 유지하면서 개념 계층을 이용한 질의 처리를 가능하게 하기 위해 해

당 노드의 전체 셀에 대한 집계 값을 의미하는 'ALL' 셀에 상위 계층의 값들이 저장된 노드를 연결시킨 구조이다.

Hierarchical Dwarf는 모든 차원의 개념 계층 이용한 집계 연산을 지원하고 동일한 노드의 중복 생성을 방지하여 저장 공간의 낭비를 줄일 수 있는 장점이 있다. 그러나 공간 차원을 지원하지 못하고, 만약 질의에서 특정 노드의 셀을 찾는 경우 이를 위해 순차 검색 방식을 사용하기 때문에 집계 연산 속도가 떨어진다는 단점이 있다. 또한, 시간 차원에서 2006년 1월과 2007년 1월이라는 값이 존재하듯이 서로 다른 상위 개념에 속하는 하위 개념들 중에서 서로 같은 값이 존재할 경우 발생할 수 있는 문제에 대한 언급이 되어있지 않다.

공간 차원이 포함된 다차원 데이터에 대한 검색 기법으로 OLAP-favored Search가 제안되었다[11]. 이는 공간 차원에 대해 R-tree 기반으로 질의 영역에 해당하는 집계 값을 계산하기 위한 기법으로 과거부터 현재까지의 총 집계 값을 담고 있는 요약 테이블을 함께 사용한다. 요약 테이블은 다차원 데이터의 스키마에 따라 차원 속성과 집계 값 속성으로 구성된다. 공간 차원의 속성에는 R-tree에 존재하는 모든 노드의 식별자가 저장된다. OLAP-favored Search 기법은 R-tree의 비단말 노드의 MBR에 해당하는 집계 값을 요약 테이블에 관리함으로써 색인 검색 속도를 향상시켰다. 이는 하나의 집계 값에 대한 자세한 정보보다는 여러 개의 집계 값을 요약한 정보가 더 중요시되는 OLAP 연산의 특징에 보다 적합한 검색 기법이다. 그러나 OLAP-favored Search 기법은 과거로부터 현재까지의 총 집계 값만을 지원하며, 시간 차원이나 이외의 차원에 대한 OLAP 연산 성능을 향상시키지 못했다.

3. 통합된 다차원 개념 계층 지원을 위한 데이터 큐브 색인 기법

본 장에서는 공간 및 비공간의 통합된 다차원 개념 계층 지원을 위한 데이터 큐브 색인 기법을 제안한다. 먼저 제안 기법의 전체적인 구조에 대해 설명하고 이를 생성 및 갱신하는 알고리즘에 대해 기술한다. 그리고 제안된 색인을 검색하여 개념 계층 질의가 처리되는 과정을 예를 들어 설명한다.

3.1 통합된 다차원 개념 계층 지원을 위한 데이터 큐브 색인 구조

제안 기법은 각 차원에 대한 여러 개의 개념 계층 트리로 구성된다. 이들은 차원의 순서대로 사실 테이블의 레코드에 저장된 내용에 따라 연결되어 있다. 예를 들어, 그림 1에서와 같이 상점, 시간, 제품 3개의 차원이 존재하면, 먼저 공간 차원인 상점 차원에 대해 개념 계층 트리를 생성한다. 그리고 각 상점에서 물건이 팔린 시간 목록을 사실 테이블에서 얻어와 이에 대한 개념 계층 트리를 생성하여 상점 차원과 연결한다. 마지막으로, 다음 차원인 제품 차원에 대해서도 같은 방법으로 개념 계층 트리를 생성하여 연결하고 단말 노드의 child에 집계 값을 계산하여 저장한다.

그림 2에 표현된 각각의 삼각형은 하나의 개념 계층 트리를 나타낸다. (1)번과 (2)번, 그리고 (4)번과 (5)번이 각각 동일한 개념 계층 트리이므로 이를 하나만 생성하여 각각 (3)번, (6)번과 같이 하나의 개념 계층 트리만 생성하여 이를 공유한다. 의미론적으로 동일한 개념 계층 트리가 중복되어 생성되는 것을

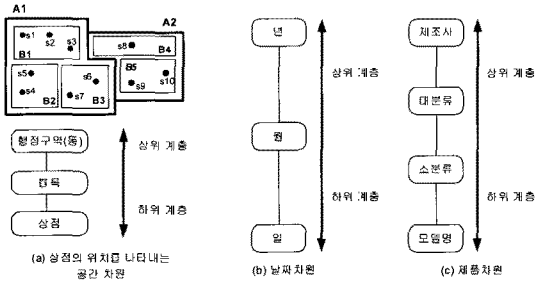


그림 1. 공간 및 비공간 차원의 개념 계층 스키마

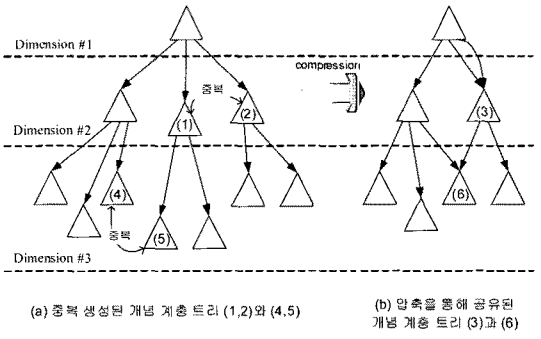


그림 2. 통합된 다차원 개념 계층 지원을 위한 데이터 큐브 색인 구조

방지하고 하나만 생성하여 공유하므로 개념 계층 트리는 그림 2와 같이 전체적으로 DAG(Directed Acyclic Graph) 구조로 연결된다.

앞서 설명한 그림 2에서 삼각형으로 표시된 개념 계층 트리의 자료구조를 자세히 나타내면 그림 3과 같다. 이는 제안된 색인의 효율적인 검색과 갱신을 위해 개념 계층 트리를 확장한 것으로 refCount, updated, root, copy와 같은 자료가 추가되었다. 이는 각 개념 계층 트리마다 추가되는 정보로 다음 절에서 설명할 생성 및 갱신 알고리즘에 사용된다.

그림 3에서 refCount는 개념 계층 트리를 공유하고 있는 노드의 개수를 나타내며 이는 색인 생성 시 계산하여 저장한다. updated는 색인 갱신 과정 중에 다른 경로를 통해 이미 갱신되었을 경우 사용하는 플래그이다. 또한 root는 개념 계층 트리의 최상위 노드를 가리키며 copy는 갱신 과정에서 갱신이상 현상을 방지하기 위해 생성되는 개념 계층 트리 복사본에 대한 포인터이다. 개념 계층 트리는 각 계층별 여러 개의 노드들로 이루어진다. 각 노드는 여러 개의 셀로 구성되며 각 셀은 차원 속성 값을 나타내는 label, 이웃 셀을 가리키는 sib, 하위 개념들이 저장된 노드를 가리키는 child로 구성된다. 또한 개념 계층 트리의 모든 노드마다 'ALL' 이라는 값을 갖는 셀이 추가되었다. 이는 해당 노드 내의 모든 셀에 대한 집계 값을 구할 때 사용된다. 개념 계층 트리의 단말 노드에 속하는 셀들은 다음 차원의 개념 계층 트리를 child로 가리키며, 마지막 차원의 경우에는 단말노드의 child에 집계 값이 저장된다.

이러한 개념 계층 트리는 해당하는 차원에 대한 개별 색인과 같이 사용되어 효율적인 검색을 지원할

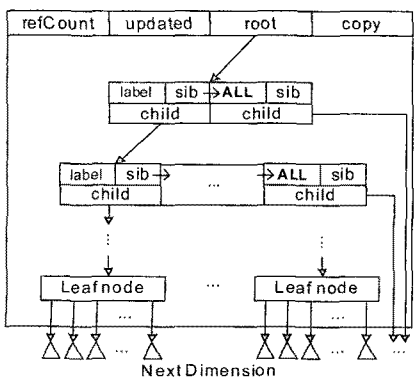


그림 3. 개념 계층 트리의 자료구조

수 있으며 서로 다른 개념에 속하는 하위 계층에서 나타나는 중복된 값들을 구분하여 처리 할 수 있다. 공간 차원 개념 계층 트리의 경우 label에 속성 값 대신 공간 데이터의 MBR이 저장되어 효율적인 공간 검색을 지원한다.

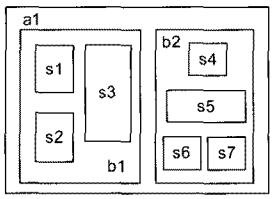
3.2 다차원 개념 계층에 대한 통합된 큐브 색인 생성

제안 기법의 환경이 되는 공간 데이터 웨어하우스 시스템은 소스 데이터를 추출 및 변환하여 임시로 저장하는 ODS(Operational Data Store)와 다차원 데이터를 저장 및 관리하는 공간 데이터 웨어하우스 서버를 구성요소로 갖는다. 제안하는 개념 계층 트리 통합 색인은 ODS에서 추출 및 통합된 후 공간 데이터 웨어하우스 서버에 적재된 다차원 데이터와 메타데이터로 관리되는 각 차원의 개념 계층 스키마에 기반 하여 생성된다. 통합 색인의 생성과정의 예시가 그림 4와 그림 5에 표현되어 있다. 우선 그림 4의 (a)에 나타난 개념 계층 트리 통합 색인의 기반이 되는 사실 테이블 상점 차원은 상점의 위치를 나타내는 공간 차원으로 그림 4의 (b)와 같고 이에 대한 개념 계층 스키마는 (c)와 같다. 날짜는 시간 차원으로 '년-월-일'과 같은 개념 계층을 갖고 있고 판매 물건 차원은 편의상 개념 계층이 존재하지 않는다고 가정한다.

차원의 순서에 따라 상점 차원의 개념 계층 트리를 생성한다. 사실 테이블에 저장된 튜플들에 존재하는 상점은 {s1, s2, s5, s7} 이므로 이들을 최하위 계층으로 갖는 개념 계층 트리를 생성하며 이는 그림 5에서 가장 위에 그려진 개념 계층 트리와 같다. 상점 차원은 공간 차원이므로 그림에 나타난 'a1', 'b1' 등과 같은 속성 값 대신 그림 4의 (b)에 나타난 공간 데이터에서 각각에 해당되는 공간 객체의 MBR이 저장된다. 이렇게 생성된 개념 계층 트리의 단말 노드와 'ALL' 셀들에 대해 다음 차원인 날짜 차원의 개념 계층 트리를 생성하여 이를 child로 연결한다. 즉, (1)번 노드의 's1' 셀의 경우 's1' 상점에서 제품이 팔린 날짜를 사실 테이블에서 얻어와 {06-01-01}에 대한 개념 계층 트리를 생성하여 's1'의 child로 연결한다. 이렇게 생성된 날짜 차원의 개념 계층 트리에 대해서도 마찬가지로 다음 차원인 판매물건 차원의 개념 계층 트리를 생성하여 연결하고 판매물건 차원의 개념 계층 트리에 각각의 집계 값을 계산하여 저

상점	날짜	판매 물건	판매 수량
s1	06-01-01	p2	15
s2	06-01-01	p1	10
s2	06-01-02	p2	20
s5	06-01-01	p2	30
s7	06-01-02	p3	25
s7	06-01-03	p1	15

(a) 사실 테이블 SALES



(b) 상점 차원의 공간 데이터

개념 계층 스키마	계층 인스턴스
행정구역(동)	b1 → a1
	b2 → a1
	s1 → b1
블록	⋮
상점	s6 → b2
	s7 → b2

(c) 개념 계층 스키마

그림 4. 개념 계층 트리 통합 색인을 위한 다차원 데이터의 예

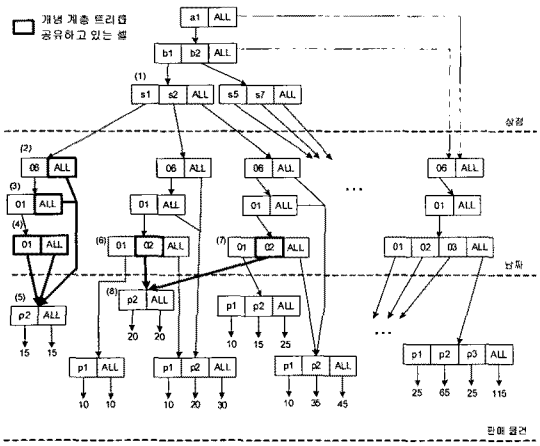


그림 5. 제안 기법으로 생성된 개념 계층 트리 통합 색인의 예

장하여 색인을 완성한다. (2), (3), (4)번 노드로 구성되는 날짜 차원의 개념 계층 트리에서 다음 차원과 연결되는 4개의 셀이 모두 (5)번 노드를 공유하는 것은 's1' 상점에서 전체 날짜에 대해 팔린 물건과 06년에 팔린 물건, 06년 01월에 팔린 물건, 06년 01월 01일에 팔린 물건에 대한 집계 값이 모두 동일하기 때문이다. 이와 같이 완성된 개념 계층 트리 통합 색인은 그림 5와 같다. 각 차원은 점선으로 구분되어 있으며, 위에서부터 '상점', '날짜', '판매 물건' 그리고 집계 값인 '판매 수량'의 순서로 나타난다. 논문의 공간상 제약으로 색인의 구조를 이해하기 위한 최소한의 그래프만 표현하였다.

그림 5에서 (6)번 노드의 '02' 셀과 (7)번 노드의 '02' 셀이 판매물건 차원에 대해서 같은 개념 계층 트리를 공유하고 있다. 이는 's2' 상점에서 06년 01월 02일 에 판매한 제품에 대한 집계 값과 'b1' 블록에 속하는 모든 상점에서 06년 01월 02일 에 판매한 제품에 대한 집계 값이 의미론적으로 동일하기 때문이다.

개념 계층 트리 통합 색인을 생성하려는 사실 테이블과 차원의 개수, 그리고 이에 대한 개념 계층 스키마가 주어졌을 때 알고리즘 1과 같은 과정을 거쳐 제안하는 통합 색인이 생성된다.

알고리즘 1. 통합된 개념 계층 색인 생성 알고리즘

Algorithm CreateDII(basetable, nDim, CHSchema)
Input
basetable: 기반 사실 테이블
nDim : 사실 테이블의 차원 개수
CHSchema: 개념 계층 스키마
Output
DII: 생성이 완료된 개념 계층 트리 통합 색인

Begin
01: DII := CreateMidHTree(*NULL*, basetable, CHSchema, nDim, 1)
02: ComputeRefCount(*DII*)
03: **return** DII
End

먼저 01번 라인에서 재귀함수인 CreateMidHTree를 호출하여 완성된 통합 색인을 얻은 다. 그리고 향후 갱신에 사용될 refCount 변수들을 02번 라인에서 각 개념 계층 트리마다 계산하여 채워 넣는다. refCount는 앞서 설명한 바와 같이 자신을 참조하는 셀의 개수를 계산한 것이다.

각 차원의 개념 계층 트리를 재귀적으로 반복하여 생성함으로써 통합 색인을 생성하는 CreateMidHTree 함수는 알고리즘 2와 같다.

먼저 01번 라인에서 주어진 parentCell에 연결될 개념 계층 트리를 그림 2의 자료구조를 사용하여 생성한다. 이들 중 단말 셀들과 'ALL' 셀들을 하나씩 탐색하면서 13번 라인에서 이 함수를 재귀 호출하여 각 셀에 연결될 다음 차원의 개념 계층 트리를 생성한다. 마지막 차원의 트리를 생성 중인 경우 05번 라인에서 각 셀의 집계 값을 계산하여 child에 저장한다. 특히, 10~11번 라인에서와 같이 'ALL' 셀의 경우 child에 연결될 트리를 생성하기 위해서 특수한

알고리즘 2. 개념 계층 트리 생성 알고리즘

```
Algorithm CreateMidHTree(parentCell, basetable,
                         nDim, CHSchema, D)
Input
    parentCell: 생성하려는 개념 계층 트리를 child로 갖
                는 셀
    basetable: 기반 사실 테이블
    nDim     : 사실 테이블의 차원 개수
    CHSchema: 개념 계층 스키마
    D: 생성하려는 개념 계층 트리의 차원 번호
Output
    HTree: parentCell에 연결될 개념 계층 트리
Variables
    currCell: HTree의 셀들을 재귀적으로 탐색

Begin
01: Construct HTree //parentCell에 해당하는 D차원
                    의 필드 값 목록과
      //CHSchema를 참조하여 개념 계층 트리 생성
02: currCell := HTree.next
03: if D = nDim then
04:     while currCell != NULL
05:         ComputeAgg(currCell)
06:         currCell := HTree.next
07:     end while
08: else
09:     while currCell != NULL
10:         if currCell.label = ALL then
11:             ProcessALL(currCell, D+1)
12:         else
13:          currCell.child := CreateMidHTree(currCell,
             basetable, nDim, CHSchema, D+1)
14:         end if
15:         currCell := HTree.next
16:     end while
17: end if
18: return HTree
End
```

처리과정이 필요하며 이는 알고리즘 3에 기술되어 있다.

먼저 01~02번 라인에서 노드에'ALL' 셀을 포함하여 두 개의 셀만이 존재할 경우 두 셀은 동일한 child를 갖게 되므로 이를 공유시킨다. 그렇지 않을 경우 06번 라인에서 노드내에 'ALL' 셀을 제외한 나머지 셀의 child를 병합시켜 'ALL' 셀의 하위 개념 계층 트리를 생성하는 MergeHTree 함수를 호출한다. 이는 알고리즘 4와 같다.

입력으로 주어지는 개념 계층 트리 dest에 src를 병합하는 연산을 수행하며, 하위에 존재하는 트리 또

알고리즘 3. 'ALL' 셀의 child 생성 알고리즘

```
Algorithm ProcessALL(allCell, D)
Input
    allCell: 기반 사실 테이블
    D: allCell의 child에 연결될 개념 계층 트리의 차
    원 번호
Variables
    sibCell: allCell내의 형제 셀들을 순차적으로 탐색
    tempHTree: allCell의 자식 트리에 대한 임시 포인터

Begin
01: If allCell.sib.sib = allCell then
                    //sib은 형제 셀을 가리키는 포인터
02:   allCell.child := allCell.sib.child
03: else
04:   sibCell := allCell.sib
05:    while sibCell != allCell
06:       MergeHTree(tempHTree, sibCell.child, D)
07:       sibCell := sibCell.sib
08:    end while
09:   allCell.child := tempHTree
10: end if
End
```

한 모두 병합해야 하기 때문에 20, 23번 라인에서 MergeHTree 함수를 재귀 호출하게 된다. 또한, 마지막 차원의 개념 계층 트리를 병합하는 경우 07번 라인에서 단말 노드의 셀들과 'ALL' 셀에 child로 저장된 집계 값들을 합치기 위해 AGG_FUNCTION을 호출한다. 이는 집계 값을 계산할 때 사용하는 집계함수를 표현한 것이며 그 예로는 MIN(), MAX, SUM(), COUNT(), AVG() 등이 있다.

이러한 과정을 거쳐 생성된 통합 색인은 공간 차원과 비공간 차원을 동시에 지원할 수 있으며 각 차원에 대한 개념 계층을 포함하여 개념 계층을 사용한 질의를 효율적으로 처리할 수 있다. 특히, 공간 차원의 개념 계층 트리에는 공간 데이터의 MBR이 저장되어 있어 공간 영역 검색을 수반하는 OLAP 연산 비용이 감소한다.

3.3 다차원 개념 계층에 대한 통합된 큐브 색인 검색

이 절에서는 제안된 색인의 검색 알고리즘을 이용하여 공간 개념 계층 질의가 처리되는 과정을 예를 들어 설명한다. 검색 알고리즘은 알고리즘 5와 같다. 이는 재귀적으로 통합 색인을 검색하는 알고리즘으로 시스템에 의해 최초 호출될 때에는 HTree가

알고리즘 4. 개념 계층 트리 병합 알고리즘

```
Algorithm MergeHTree(dest, src, nDim, D)
Input
    dest: 병합된 트리가 저장될 변수
    src: 병합하려는 대상이 되는 개념 계층 트리
    nDim: 사실 테이블의 차원 개수
    D: dest와 src의 차원 번호
Variables
    currSrcCell: src의 셀들을 하나씩 탐색

Begin
01: if dest is empty then
02:   copy all of src to dest
        //모든 'ALL' 셀의 child 값을 NULL로 세팅
03: else if D = nDim then
04:   currSrcCell := src.next
05:   while currSrcCell != NULL
06:     if currSrcCell is in dest then
07:       dest.rddt.child := AGG_FUNCTION
          (dest.rddt.child, currSrcCell.child)
08:     else
09:       dest.insert(currSrcCell)
10:     end if
11:     currSrcCell := src.next
12:   end while
13: else
14:   currSrcCell := src.next
15:   while currSrcCell != NULL
16:     if currSrcCell.label = ALL then
17:       ProcessALL(dest.currALL, D+1)
18:     else if currSrcCell is in dest then
19:       if dest.currALL.child = NULL then
          //첫 번째 병합이므로 새로운
          //개념 계층 트리 변수 할당
20:         MergeHTree(tempHTree, dest.rddt.child,
                       nDim, D+1)
21:         dest.rddt.child :=tempHTree
22:       end if
23:       MergeHTree(dest.rddt.child, currSrcCell.
                     child, nDim, D+1)
24:     else
25:       dest.insert(currSrcCell)
26:     end if
27:     currSrcCell := src.next
28:   end while
29: end if
End
```

첫 번째 차원의 개념 계층 트리가 되고, 차원의 순번을 나타내는 D는 1이 된다.

공간 질의의 형태는 공간 차원에 대한 점 질의와 영역 질의로 나누어지며, 사용자는 각 차원에 대해

알고리즘 5. 통합된 개념 계층 색인의 검색 알고리즘

```
Algorithm RtrvHTree(HTree, query, D, nDim)
Input
    HTree: 검색 대상인 개념 계층 트리
    query: 각 차원에 대한 계층과 차원 속성 값 또는
           질의 영역 포함
    D: HTree에 해당하는 차원의 순번
    nDim : 검색하려는 통합 색인의 전체 차원의 개수
Output
    answer: query에 대한 결과 집계 값

Begin
01: idCell := (query에 대해 검색된 HTree내 하나의 셀)
02: if D = nDim then
03:   if idCell is in leaf node then answer :=
      idCell.child
04:   else answer := idCell.child.ALL.child
05:   end if
06:   return answer
07: else
08:   if idCell is in leaf node then
09:     return RtrvHTree(idCell.child, query, D+1,
                         nDim)
10:   else return RtrvHTree(idCell.child.ALL.child,
                            query, D+1, nDim)
11:   end if
12: end if
End
```

원하는 계층을 설정한 후 질의한다. 알고리즘 5는 점 질의에 대한 검색 알고리즘이며, 그림 6의 통합 색인에 대한 점 질의의 예를 들면, "s2상점에서 2006년 1월에 판매한 모든 물건의 수량"을 구하는 질의는 다음과 같이 나타낼 수 있다. QPoint는 공간상에서 사용자가 질의하고자 하는 점을 나타내며 이는 's2' 상점의 영역에 포함된다고 가정한다.

"QPoint (상점), 06-01(월), ALL" (질의 1)

질의가 처리되는 과정을 알고리즘 5와 그림 6을 이용하여 설명하면, 알고리즘의 01번 라인에서 첫 번째 차원인 '상점' 차원에 대해 상점 계층까지 R-tree와 같은 알고리즘으로 탐색하여 QPoint가 위치하는 's2'를 찾아서 child에 연결되어 있는 노드로 이동한다. 's2'가 속한 노드는 단말 노드이므로 08번 라인에서와 같이 child 노드를 매개변수로 하여 재귀 호출한다. 시간 차원에 대해서는 2006년 1월을 찾아야 하므로 '06'에 연결되어있는 노드에서 '01' 셀을 찾고 이는 단말노드가 아니므로 09번 라인에서와 같이

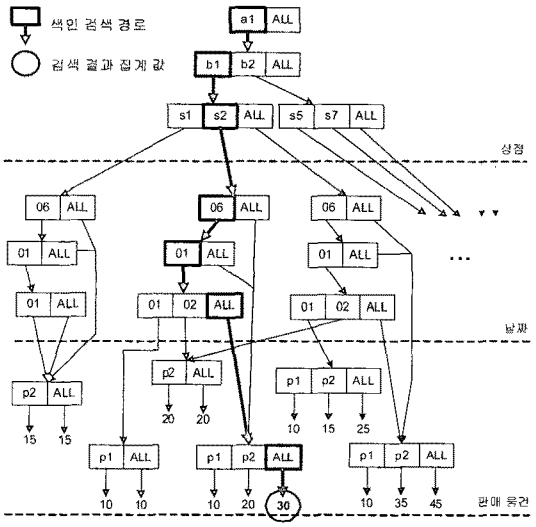


그림 6. (질의 1)에 대한 통합 색인 검색 경로

'01' 셀의 child 노드의 'ALL'에 연결된 노드를 매개변수로 하여 재귀 호출한다. 모든 판매 물건에 대한 질의이므로 03번 라인에서와 같이 'ALL'에 연결된 '30'이 질의 1의 결과로 출력된다.

영역 질의의 경우 점 질에서 사용된 QPoint 대신 사용자가 지정한 질의 영역을 이용한다. 예를 들어, 사용자가 상점 차원의 개념 계층을 '블록'으로 설정하고, 지정된 질의 영역이 'b1'과 'b2'에 겹칠 경우, 루트 노드로부터 블록 계층까지 영역 검색을 하여 'b1'과 'b2'를 찾아내고 이 영역 질의는 'b1'과 'b2'에 대한 두 개의 점 질의로 나뉘어 처리된다.

제안된 검색 알고리즘을 사용하면 모든 노드에 추가된 'ALL'셀을 통해 중간 계층의 집계 값을 하위 노드에 대한 탐색 비용 없이 직접 얻을 수 있다. 또한 영역 질의 처리 시 질의에 설정된 계층보다 상위의 계층에서 질의 영역에 완전히 포함되는 중간 계층의 노드가 검색될 경우 더 이상 검색하지 않고 해당 노드의 'ALL' 셀을 통해 집계 값을 얻을 수 있다.

4. 성능평가

본 장에서는 제안 기법인 통합된 개념 계층 색인과 기존의 기법들에 대한 저장 비용, 갱신 성능 그리고 질의 처리를 위한 색인 검색 성능들을 비교 평가한다.

본 실험에 사용된 다차원 데이터는 인자 2를 사용한 Zipf 분포를 갖는 합성 데이터를 담고 있는 사실 테이블을 사용하였다. Zipf의 법칙은 실제 데이터에 근접한 임의의 데이터를 생성하는 무작위화 알고리즘으로 데이터에서 나타나는 문자 및 숫자의 발생 빈도수를 실제 데이터와 유사한 패턴으로 합성한다[14]. 합성 데이터를 갖는 테이블은 공간 차원으로 주소 개념 계층을 갖는 서울시 강남구에 대한 공간 데이터를 저장하고 있다. 이외에 개념 계층에 대해 균일한 기수(cardinality)를 갖도록 Zipf(2) 분포를 나타내는 데이터를 담고 있는 9개의 비 공간 차원을 포함하고 있다. 집계 값 또한 같은 방법으로 합성된 수치 데이터이다.

색인의 확장성을 분석하기 위해 10개의 차원과 100,000개의 레코드를 갖는 합성 데이터에 대해 개념 계층을 갖는 차원의 개수를 1개부터 10개까지 변화시키면서 4가지 기법의 색인을 생성하여 저장 비용을 측정하였다. 공간 차원으로 서울특별시 강남구 지번도 데이터를 사용하였으며 주소에 대한 개념 계층을 적용하였다. Hierarchical Dwarf는 공간 데이터를 지원하지 못하므로 비공간 속성인 주소를 링크로 사용해 공간 데이터에 접근하는 방식을 이용하였고 QC-tree는 개념 계층을 지원하지 못하여 각 계층을 사실 테이블의 속성으로 추가하여 색인을 구축하였다.

실험 결과는 그림 7과 같다. OLAP-favored Search 기법을 이용한 색인은 데이터에 대한 압축을 지원하지 않기 때문에 개념 계층의 개수가 늘어남에 때라 저장용량이 급격히 증가하였다. 반면, 제안 기법에 의해 생성된 개념 계층 트리 통합 색인은 효율적인 무손실 압축을 지원하는 Hierarchical Dwarf에 비해 다소 많은 저장 공간을 차지하지만, 차원의 개수가 늘어나

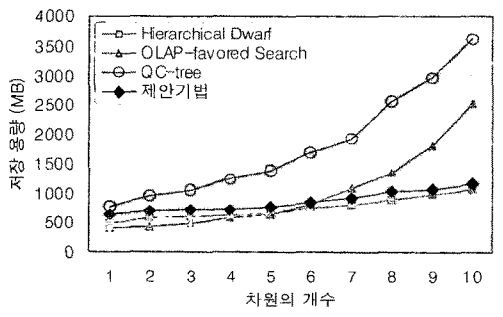


그림 7. 개념 계층이 존재하는 차원의 개수에 따른 색인의 저장 용량

도 차이가 더 이상 벌어지지는 않았다. 이는 제안 기법이 구조상 Hierarchical Dwarf에 비해 다소 많은 포인터가 존재하고 공간 객체의 문자 속성 대신 MBR이 저장되기 때문이다. QC-tree는 의미론적으로 동일한 셀들을 클래스로 묶는 압축 기법을 사용하여 개념 계층을 갖는 차원의 수가 적을 때에는 다른 기법들과 비슷한 비용을 요구하지만 개념 계층을 지원하지 못해 개념 계층 차원의 개수가 점차적으로 늘어난다.

본 실험에서는 개념 계층 트리의 공유를 통해 색인을 압축하는 제안 기법이 Hierarchical Dwarf를 제외한 다른 기법을 통해 생성된 색인보다 적은 용량을 차지하였으며 개념 계층을 갖는 차원의 개수가 증가함에 따른 저장 용량 증가율도 가장 낮게 측정된다.

다음은 검색 성능평가로 1개의 공간 차원과 5개의 비공간 차원에 대해 전체 150,000개의 레코드를 담고 있는 사실 테이블을 사용하여 공간 및 비공간 차원에 대한 질의 응답시간을 측정하였다. 먼저 공간 차원에 대한 검색 성능을 평가하기 위해 제안 기법과 OLAP-favored Search 기법을 이용해 공간 질의 영역의 크기를 1~80%로 변화시키면서 질의 응답시간을 측정하였다. 전체 공간영역 중 80% 이상의 영역을 검색하는 것은 거의 모든 데이터가 검색되어 공간 차원을 제외한 검색과 결과가 유사하여 의미가 없으므로 실험에서 제외 되었다. 사용된 질의는 공간 차원의 최하위 계층에 대한 질의이며 비공간 차원에 대한 검색조건을 "2006년 1월에 판매된 제품 p3의 개수"와 같이 고정시킨 채로 공간상의 질의 영역 크기를 변화시키면서 생성하였다. 실험 결과는 그림 8과 같다. QC-tree와 Hierarchical Dwarf는 공간 차원을 지원하지 못하므로 이들을 이용해 비공간 차원에 대한 질의 응답시간을 측정하여 제안 기법과 검색 성능을 비교하는 실험을 따로 진행한다.

공간 차원에 대한 검색 성능은 질의 영역이 전체의 5% 이하일 경우 제안 기법과 OLAP-favored Search 기법은 거의 비슷한 성능을 보인다. 그러나 질의 영역의 크기가 10% 이상일 경우 집계 값을 계산하기 위해 검색해야 하는 비공간 데이터가 점차 증가한다. OLAP-favored Search의 경우 비공간 데이터에 대한 색인 기법은 제공하지 않고 데이터베이스에 저장된 데이터를 순차적으로 검색하기 때문에 접근되는 비공간 데이터가 많아지는 10% 이후로는 오히려 검색 성능이 저하되는 것을 볼 수 있다. 반면 제안 기법은 공간 및 비공간 차원 모두에 대해 통합된 색인을 지원하기 때문에 10% 이후부터는 OLAP-favored Search 기법보다 향상된 성능을 보인다. 제안 기법에서 질의 영역의 크기가 커짐에 따라 응답시간이 짧아지는 것은 질의 영역의 크기가 클수록 중간 계층의 MBR이 질의 영역에 속해 하위 노드의 탐색이 생략되는 빈도가 높아지기 때문이다.

비공간 차원에 대한 검색 성능을 비교하기 위해 공간 차원을 제외시킨 사실 테이블을 이용해 비공간 차원 질의의 레코드 선택률을 변화시키면서 응답 속도를 측정하였다. 레코드 선택률은 질의 조건을 만족하는 레코드의 비율이며 이는 질의 결과의 크기를 나타낸다. 이 실험에서는 시간의 범위와 제품의 범위를 이용해 질의 범위를 조절하였다. 비공간 차원에 대한 색인을 제공하지 않는 OLAP-favored Search 기법을 제외한 QC-tree, Hierarchical Dwarf, 그리고 제안 기법을 이용하여 질의응답 시간을 측정한 결과는 그림 8과 같다.

QC-tree는 다차원 데이터를 클래스로 나누어 압축한 후 각 클래스에 대한 빠른 검색을 지원하지만 개념 계층을 고려하지 않아 선택률이 높아질수록 노드 접근 수가 많아져 응답 시간이 점점 길어진다. 반면 Hierarchical Dwarf와 제안기법은 질의 범위의 크기가 커질수록 상위 계층의 집계 값을 사용하여 하위 계층의 노드에 대한 접근을 생략하기 때문에 레코드 선택률이 높아짐에 따라 응답 시간이 짧아진다. 또한 질의 범위가 5% 이상인 경우 제안 기법이 Hierarchical Dwarf 보다 평균 15%가량 빠른 검색 성능을 보인다. 이는Hierarchical Dwarf는 bottom-up 방식으로 개념 계층을 검색하지만 제안 기법은 top-down 방식

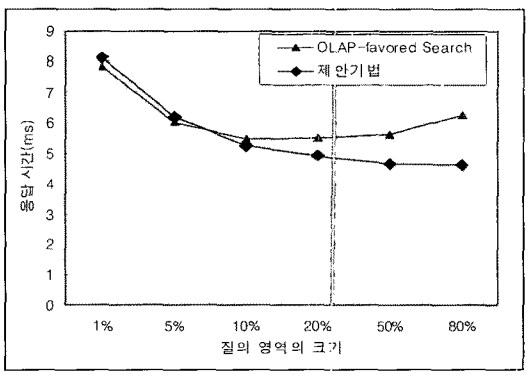


그림 8. 질의 영역의 크기에 따른 공간 질의응답 시간

을 사용하기 때문이다. 또한 질의 범위가 5% 이하로 작은 경우에는 항상 최하위 계층에서 결과를 찾기 때문에 bottom-up 방식이 더 효율적임을 볼 수 있다.

5. 결론 및 향후연구

본 논문에서는 통합된 개념 계층 지원을 위한 데이터 큐브 색인 기법을 제안하였다. 이는 기존 색인 기법들의 문제점들을 개선하고 공간 및 비공간 차원에 대한 개념 계층을 지원하는 것으로서, 보다 유용한 공간OLAP 연산을 제공한다. 각각의 개념 계층 트리는 해당 차원에 대해 개별적인 색인으로 작용하여 검색 성능을 제공시킨다. 특히 공간 차원의 개념 계층 트리에는 공간 객체 트리에는 공저장되어 효율적인 공간 체 검색을 지원한다. 제안 기법을 통해 계층별 체 집계 연산을 효율적으로 제공할 수 있으며 동일한 개념 계층 트리를 중복 생성하는 것을 방지하여 데이터 큐브를 원시적인 다차원 배열에 저장하는 기법에 비해 저장 공간 비용을 크게 줄일 수 있다.

성능평가에서 Hierarchical Dwarf와 비교하였을 때 제안 기법의 저장 비용이 15% 가량 더 크지만, 비공간 차원에 대한 검색 성능이 질의 범위 5% 이상인 경우 약 17% 정도 향상되었다. 또한 공간 차원에 대해서는 OLAP-favored Search 기법과 달리 질의 영역이 커지더라도 검색 성능이 저하되지 않았다. 제안 기법의 갱신 속도 면에서도 QC-tree보다 약 50% 정도 향상된 속도를 보였으며, Hierarchical Dwarf보다 갱신 소요시간의 증가율이 약 40% 정도 낮았다. 또한, 재구축 기법과 거의 동일한 저장 비용을 보여 제안된 갱신 알고리즘이 기존 개념 계층 트리 통합 색인의 압축 성능을 유지함을 보였다.

향후연구로는 개념 계층 스키마에 변화가 생겼을 경우 색인을 재구축하지 않고 갱신하는 기법에 대한 연구가 필요하다.

참 고 문 헌

[1] E. F. Codd, S. B. Codd, and C. T. Salley, "Providing OLAP to user-analysts: An IT mandate," Technical Report San Jose IBM, 1993.

[2] S. Bimonte, A. Tchounikine, M. Miquel, "Towards a Spatial Multidimensional Model," *ACM DOLAP*, pp. 39-46, 2005.

[3] S. Chaudhuri, U. Dayal, "An Overview of Data Warehousing and OLAP Technology," *ACM SIGMOD*, Vol.26, No.1, pp. 65-74, 1997

[4] J. Gray, A. Bosworth, A. Layman, H. Pirahesh, "Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals," *IEEE ICDE*, pp. 152-159, 1996.

[5] A. Gupta and I. S. Mumick. "Maintenance of materialized views: Problems, techniques, and applications," *IEEE Data Engineering Bulletin*, Vol.18, No.2, pp. 3-18, 1995.

[6] M. Ester, J. Kohlhammer, H.-P. Kriegel, "DC-tree: A Fully Dynamic Index Structure for Data Warehouses," *IEEE ICDE*, pp. 379-388, 2000.

[7] K. Hu, L. Chen, S. Jie, Q. Gu, X. Tang, "A Highly Performance Dimension Hierarchy Aggregate Cube and Its Incremental Update Algorithm," Proc. of Int. Conf. on Machine Learning and Cybernetics, Vol.4, No.1, pp. 2195-2199, 2005.

[8] 옥근형, 이동욱, 유병섭, 배해영, "공간 데이터웨어하우스에서 개념 계층을 지원하는 공간 데이터 큐브," 한국정보처리학회 춘계학술대회, Vol.13, No.1, pp. 35-38, 2006.

[9] 옥근형, 이동욱, 유병섭, 이재동, 배해영, "다차원 개념 계층을 지원하는 공간 데이터 큐브의 점진적 일괄 갱신 기법," 한국멀티미디어학회 논문지, Vol.9, No.11, pp. 1395-1409, 2006.

[10] L. Lakshmanan, J. Pei, Y. Zhao, "QC-Trees: An Efficient Summary Structure for Semantic OLAP," *ACM SIGMOD*, pp. 64-75, 2003.

[11] F. Rao, L. Zhang, X.L. Yu, Y. Li, Y. Chen, "Spatial Hierarchy and OLAP-favored Search in Spatial Data Warehouse," *ACM DOLAP*, pp. 48-55, 2003.

[12] Y. Sismanis, A. Deligiannakis, Y. Kotidis, N. Roussopoulos, "Hierarchical Dwarfs for the Rollup Cube," *ACM DOLAP*, pp. 17-24, 2003.

[13] Y. Sismanis, A. Deligiannakis, N. Roussopoulos, Y. Kotidis, "Dwarf: Shrinking the PetaCube," *ACM SIGMOD*, pp. 464-475, 2002.

[14] Y. M. Ioannides, H. G. Overman, "Zipf's law for cities: an empirical examination," Regional Science and Urban Economics, Vol.33, No.2, pp. 127-137, 2003.

김 경 배

1992년 인하대학교 전자계산공학과 공학사
1994년 인하대학교 전자계산공학과 공학석사
2000년 인하대학교 전자계산공학과 공학박사
2004년~현재 서원대학교 컴퓨터교육과 조교수

관심분야 : 이동실시간 데이터베이스, 스토리지 시스템

이 동 욱

2003년 상지대학교 전자계산공학과 이학사
2005년 인하대학교 정보공학과 공학석사
2005년~현재 인하대학교 정보공학과 박사과정

관심분야 : 유비쿼터스 환경에서의 SDBMS, SDW 및 DSMS

배 해 영

1974년 인하대학교 응용물리학과 공학사
1978년 연세대학교 전자계산공학과 공학석사
1989년 숭실대학교 전자계산학과 공학박사
1982년~현재 인하대학교 정보공학부 교수

관심분야 : 공간 데이터베이스, 멀티미디어 데이터베이스 등

백 성 하

2005년 인하대학교 수학통계학부 이학사
2007년 인하대학교 정보공학과 공학석사
2007년~현재 인하대학교 정보공학과 박사과정

관심분야 : DSMS, 데이터베이스 클러스터