韓國 OA學會 論文誌
第4卷 第4號, 1999.12.

99-4-4-1-9

공간 데이타 큐브의 선택적 실체화에 관한 연구

이 기 영*

A Study on the Selective Materialization of Spatial Data Cube

Ki-Young Lee*

요 약

최근에 공간 데이타 웨어하우스에서 자주 사용되어지고 질의 응답 시간이 많이 걸리는 복잡한 공간 집계 질의들은 미리 그 결과를 계산하여 실체화시키는 방법들이 연구되고 있다. 본 논문에서는 기존의 선택적 실체화 알고리즘에 대한 개선 방안으로 공간 뷰의 공간 측정에 대한 공간 연산의 계산 시간과 접근 빈도를 고려하여 선택적 실체화에 대한 방법을 제시하고 개선된 선택적 실체화 알고리즘을 제안한다.

Abstract

Recently, it has been studied the methods to materialize and precompute the query results for complexed spatial aggregation queries with high response time and the popular use in spatial data warehouse. In this paper, we propose extended selective materialization algorithm and present the way to materialize selectively which is considered access frequency and computation time of spatial operation according to spatial measures of spatial views for improvement of existing selective materialization algorithms.

* 서울보건대학 사무자동화과 조교수
논문접수:1999.11.2. 심사완료:1999.12.10.

Ⅰ. 서론

최근 들어 의사 결정 과정을 지원하기 위해 데이타 웨어하우스에 관한 연구가 활발히 진행되고 있다[1, 2]. 그러나 데이타 웨어하우스에 구성된 데이타들의 유형을 살펴볼 때에 데이타의 80퍼센트 이상이 공간적인 요소를 내포하고 있음에도 불구하고 데이타 웨어하우스에 이러한 공간적인 요소들을 표현, 통합하여 관리할 수 있는 공간 데이타 웨어하우스에 관한 연구는 아직 미흡한 실정에 있다[3, 6].

공간 데이타 웨어하우스는 사용자의 빠른 정보 분석을 돕기 위한 공간 OLAP(On-Line Analytical Processing)에 사용되어진다. 그러나 OLAP환경에서의 공간 질의들은 공간 합병, 공간 Overlay, 공간 죠인, 공간 교차등 대부분 Aggregation 함수나 Group-by등이 포함된 복잡한 공간 질의 연산을 수행하여야 하므로 신속한 의사 결정 지원을 위해 사용자가 원하는 시간 내에 응답하기가 어렵다[4]. 그리하여 이와 같은 문제 해결을 위한 여러 가지 방법들이 연구되고 있다. 현재 일반적으로 사용되는 방법은 사용자가 미리 요구하는 공간 질의, 즉 자주 발생하는 공간 질의들을 미리 계산하여 결과를 실체화(Materialization)하는 것이다[5, 7, 8-11].

그러므로 어떤 공간 뷰를 실체화할 때 고려해야 할 사항은 실체 공간 뷰는 실제의 저장 공간에 계산 결과를 저장해 두기 때문에 사용자의 요구를 잘 만족시킬 수 있도록 제한된 저장 공간에 알맞은 공간 뷰를 선택하는 것이 중요한 문제로 연구되고 있다. 이때 중요한 고려 사항은 OLAP환경의 공간 질의처리 시간을 최소화하고 또한 유지 비용을 최소화하는 공간 뷰를 선택하는 것이다 12-14].

이러한 공간 뷰 선택 문제들을 해결하기 위한 기존의 공간 측정 실체화 알고리즘으로 Rough Measures, Spatial Greedy, Pointer Intersection, Object Connection 알고리즘들이 있다. 이 알고리즘들은 공간 질의 처리를 위한 공간 연산들의 계산 시간을 고려하지 않고 공간 질의 접근 빈도수를 가정하여 알고리즘을 제시하였다.

본 논문에서는 공간 질의를 효율적으로 처리하기 위하여 데이타 큐브를 비공간, 공간 데이타 큐브로 분류하였고 공간 연산들의 계산 시간도 고려하여 개선된 공간 뷰 선택 알고리즘을 제안한다.

본 논문의 구성은 다음과 같다. 제 2 장에서는 관련 연구로서 공간 뷰 선택 문제들을 해결하기 위하여 기존의 공간 측정 실체화 알고리즘에 대해 살펴보고, 제 3 장에서는 공간 질의를 효율적으로 처리하기 위하여 가상적 시나리오를 토대로 데이타 큐브를 비공간, 공간 데이타 큐브로 분류하고 공간 데이타 큐브에 대하여 고찰한다. 제 4 장에서는 효율적으로 공간 뷰 선택 문제를 해결하기 위해 개선된 공간 뷰 선택 알고리즘을 제안한다. 마지막으로 제 5 장에서는 결론과 앞으로의 연구 과제에 대해 언급한다.

Ⅱ. 관련연구

공간 데이타 웨어하우스에서 공간 측정의 선택적 실체화 방법의 기준은 공간 OLAP 질의에 대한 질의 응답 시간을 최소화하고자 하는 것이다. 이러한 공간 측정 실체화 방법에 대한 기존의 알고리즘은 여러 Heuristics요소들이 사용되고 있다. 본 장에서는 효율적인 공간 측정 선택적 실체화 알고리즘을 제시함에 있어 기존의 선택적 실체화 알고리즘에 대하여 살펴본다.

2.1 데이타 큐브

데이타 웨어하우스의 데이타를 표현하기 위해서 사실(Fact) 테이블과 차원(Dimension) 테이블로 구성되고 이러한 사실 테이블의 정보에는 측정값과 각 데이타를 구별하여 주는 차원 정보가 있다. 또한 차원 테이블의 데이타들 사이에는 계층 구조를 갖는다.

데이타 큐브는 사실 테이블로부터 유래될 수 있는 모든 다양한 집계 뷰(View)들에 대한 격자(Lattice) 구조로서 표현할 수 있다.또한 데이타 큐브의 각 셀들에 포함된 값은 데이타 측정값이다. 데이타 큐브는 다차원 데이타베이스의 다차원 배열에서 나온 개념이지만 단순한 다

차원 배열이 아니라 데이타 분석에 꼭 필요한 부가 정보들을 포함하고 있다. 그러므로 데이타 큐브는 다각도의 관점에서 측정할 수 있도록 하는 데이타 구조이다.

데이타 웨어하우스의 테이블은 상당히 큰 용량을 가지므로 상위 개념의 집계된 정보를 요구하는 사용자 질의에 대해서 그 결과를 미리 계산하여 두는 것이 효과적이다. 가장 기초적인 데이타에 대한 질의 만을 계산하여 결과를 실체화 뷰에 저장한 구조를 단순 격자라하고 실제 사용자 응답 시간을 줄이기위해 상위 계층의 집계된 정보를 계산하여 그 결과를 모두 데이타 큐브의 실체화 뷰에 저장한 구조를 복합 격자라 한다.

OLAP 환경에서 질의 응답 시간과 유지 비용을 최소화하기 위해 이러한 데이타 큐브에서 실체화할 뷰를 선택하는 것은 매우 중요한 문제이다.

2.2 기존 선택적 실체화 알고리즘

공간 데이타 웨어하우스에서 공간 OLAP의 공간 합병(Merge)에 대한 공간 측정을 위해 기존에 제안된 선택적 실체화 알고리즘으로는 Rough Measures, Spatial Greedy, Pointer Intersection, Object Connection 알고리즘들이 있다. 이러한 알고리즘에 대하여 살펴보면 다음과 같다.

(1) Rough Measures 알고리즘

공간 데이타 큐브 구조에서는 복잡한 공간 데이타를 그대로 저장하는 방법을 사용하지 않고 효율성를 높이기 위해 MBR(Minimun Bounding Rectangle), MMBR(Merged Minimun Bounding Rectangle)등과 같은 근사치를 사용한다. 선택적 실체화 공간 데이타 큐브를 생성하기 위해 잘 알려진 Cuboid-Selection 알고리즘(HRU)를 사용하고 생성된 공간 데이타 큐브에 대하여 다시 Area-Weight, Min-Weight를 이용하여 선택적 실체화를 수행한다.

이 알고리즘은 공간 측정의 질의 응답 시간이 빠르고 실체화 큐브를 저장하기 위한 작은 기억 공간을 요구하므로 현재 공간 응용분야에서 널리 사용된다. 그러나 공간 데이타에 대한 근사치를 사용하므로 정밀도가 떨어지는 단점이 있다.

(2) Spatial Greedy 알고리즘

이 알고리즘은 선택적 실체화 공간 데이타 큐브를 생성하기 위해 잘 알려진 Cuboid-Selection 알고리즘를 사용하고 생성된 공간 데이타 큐브에 대하여 HRU에서 사용하지 않은 공간 질의 접근 빈도수(Access Frequency)를 추정하여 적용한다. 또한 Direct, Indirect, Max Benefit를 이용하여 선택적 실체화 공간 데이타 큐브를 생성한다. 그러나 연결된 공간 영역들만 공간 합병되고 Indirect Benefit를 계산하는데 많은 시간이 걸리므로 알고리즘의 성능이 떨어진다.

(3) Pointer Intersection 알고리즘

이 알고리즘은 선택적 실체화 공간 데이타 큐브를 생성하기 위해 잘 알려진 Cuboid-Selection 알고리즘를 확장하여 사용한다는 가정을 전제로 이 HRU를 먼저 적용한다. 최종적인 선택적 실체화 공간 데이타 큐브를 생성하기 위해 공간 질의 접근빈도수,Get_Max_Intersection, Total_Access_Frequency,Min_Access_Frequency,Spatial_Connectivity를 테스트하여 선택적 실체화를 수행한다. 공간 측정에 대한 공간 객체들이 증가하거나 공간 측정에 대한 접근 빈도수가 증가하면 할수록 성능이 좋아진다. 그러나 기존 HRU 알고리즘의 확장을 전제로한 가정에는 모순이 있다.

(4) Object Connection 알고리즘

이 알고리즘도 Pointer Intersection 알고리즘과 마찬가지로 확장된 Cuboid-Selection 알고리즘를 사용한다는 가정을 전제로 한다. 최종적인 선택적 실체화 공간 데이타 큐브를 생성하기 위해 공간 질의 접근빈도수, Get_Max_Connected_Intersection,Total_Access_Frequency,Min_Access_Frequency_Threshold를 테스트하여 선택적 실체화를 수행한다. 공간 측정에 대한 공간 객체들이 증가하거나 공간 측정에 대한 접근 빈도수가 증가하면 할수록 성능이 Pointer Intersection 알고리즘보다 떨어진다.

본 논문에서는 공간 데이타 큐브의 효율적인 선택적 실체화 알고리즘을 제시함에 있어 기존의 선택적 실체화 알고리즘들을 토대로 확장하여 개선하였다.

Ⅲ. 공간 데이타 큐브

효율적인 선택적 실체화 알고리즘을 제시함에 있어서 공간 데이타 큐브에서 공간 측정 연산의 대상이 되는 뷰들 중에서 실체화할 뷰들을 선택하는 과정이다. 뷰를 실체화하는 것은 뷰의 크기 만큼의 저장 공간을 요구하므로 제한된 공간에서 알맞는 뷰를 선택하는 방법에 대한 연구는 공간 데이나 웨어하우스에서 상당히 중요한 문제이다. 뷰를 선택하는 기준은 공간 측정 질의 응답 시간을 줄이고 계산 결과를 유지하는 비용을 최소화하고자 하는 것이다.

이를 위해 먼저 데이타 큐브에서 각 뷰가 가지는 특성을 살펴보는 것이 바람직하다. 그래서 본 논문에서는 대상이 되는 뷰들의 특성에 따라 기존 데이타 큐브를 비공간, 공간 데이타 큐브로 분류하여 공간 측정 실체화를 효율적으로 수행하기 위하여 그림 1과 같이 분류하였고 이러한 공간 데이타 큐브는 공간 데이타 차원(Dimension)과 직접적으로 연관된 뷰 들로 구성된다. 공간 객체들은 공간 포인터나 공간 객체 식별자에 의해 표현되어 질 수 있다.

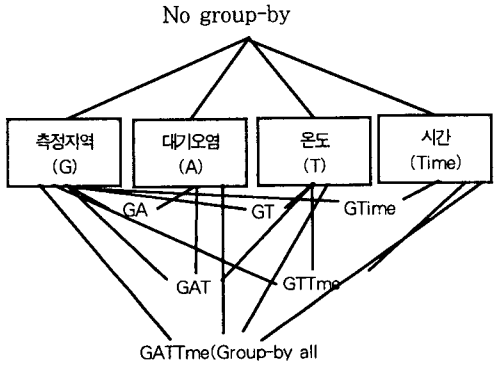

그림 1. 공간 데이타 큐브

그림 1에서 보는 바와같이 비공간 데이타 차원(대기오염, 온도, 시간)으로만 연관된 뷰들은 공간 데이타 큐브에 포함시키지 않고 공간 데이타 차원(측정지역)과 연관된 뷰(GA, GT, GTime, GAT, GTTime, GATTime)들만 공간 데이타 큐브를 따로 생성하여 관리함으로서 공간 측정의 선택적 실체화를 수행하는 방법을 단순화시킬 수 있다. 그러므로 공간 데이타 웨어하우스에서 공간 OLAP 환경의 공간 질의 응답 시간을 줄일 수 있다.

Ⅳ. 선택적 실체화 알고리즘 개선

본 장에서는 선택적 실체화 알고리즘을 개선하기 위해서 기존 알고리즘에서 고려하지 않은 요소를 고려하여 실체화 유무를 결정하여 개선된 선택적 실체화 알고리즘을 제시한다.

4.1 고려해야할 사항

기존의 선택적 실체화 알고리즘에서는 사용자의 공간 연산에 대한 공간 질의 접근 빈도수를 추정하여 알고리즘에 적용하였다. 그러므로 공간 질의 접근 빈도수가 크면 그에 관련된 뷰들은 자연히 데이타 큐브에서 실체화될 확률이 크다. 그러나 공간 질의 접근 빈도수가 크다 할지라도 관련된 공간 뷰들의 계산 시간이 많이 걸린다면 실체화하는 것은 고려해야할 중요한 문제이다.

접근 빈도수 / 계산시간	high	low
high	good	고려(1)
low	고려(2)	bad

그림 2. 고려해야할 사항

그래서 본 논문에서는 이러한 계산 시간을 고려하여 선택적 실체화 유무를 결정한다. 그림 2에서와 같이 공간 질의 접근 빈도수 가 높고 계산 시간이 많이 걸리면 그 공간 뷰들을 공간 데이타 큐브에서 실체화하는 것이 좋으며 반대로 공간 질의 접근 빈도수가 낮고 계산 시간이 적게 걸리면 실체화하는 것은 나쁘다. 이러한 경우들은 당연히 기존의 실체화 알고리즘에서 선택되어 진다.

그러나 공간 질의 접근 빈도수가 낮고 계산 시간이 많이 걸리면 그 공간 뷰들을 공간 데이타 큐브에서 기존의

실체화 알고리즘들은 공간 질의 접근 빈도수에 의해서만 비실체화하지만 이 경우는 계산 시간에 의해 고려해야만 한다. 또한 공간 질의 접근 빈도수가 높고 계산 시간이 적게 걸리면 기존의 실체화 알고리즘들은 공간 질의 접근 빈도수에 의해서만 실체화하지만 이 경우도 계산 시간에 의해 고려해야만 한다.

알고리즘 / 고려사항	pointer intersection	알고리즘 개선
good	O	O
고려(1)	X	O
고려(2)	O	X
bad	X	X

(O : 실체화, X : 비실체화)
그림 3. 선택적 실체화 유무 결정

그림 3은 앞에서 언급한 고려해야 할 사항(고려1, 2)에 대하여 기존의 선택적 알고리즘에서 성능이 비교적 좋은 Pointer Intersection 알고리즘과 이 논문에서 제시하고자 하는 알고리즘과의 공간 뷰들 중에서 실체화하는 경우를 나타낸다. Good, Bad 의 경우는 두 알고리즘 모두 공간 뷰들을 실체화하지 않으며 고려(1)의 경우는 Pointer Intersection 알고리즘에서는 비실체화하고 이 논문의 알고리즘에서는 실체화한다. 게다가 고려(2)의 경우에서는 Pointer Intersection 알고리즘은 실체화하는 반면에 이 논문의 알고리즘에서는 비실체화한다.

이러한 주된 이유는 공간 질의 빈도수가 낮으나 공간 측정의 뷰에 대한 계산 시간이 많이 걸린다면 실체화하는 것이 공간 질의 응답 시간에 대한 시스템의 성능을 향상시키는 결과가 되기 때문이다. 역으로 공간 질의 빈도수가 높으나 공간 측정의 뷰에 대한 계산 시간이 적게 걸린다면 비실체화하는 것이 저장 공간의 유지 비용을 최소화시킬 수 있다. 물론 공간 질의 접근 빈도수와 계산 시간의 효율적인 Threshold를 결정하는 것은 전체적인 응용 분야와 사용자 질의 요구 분석을 통하여 설정하는 것이 바람직하다.

4.2 개선된 선택적 실체화 알고리즘

본 절에서는 4.1절에서 언급한 내용을 토대로 공간 합병에 대한 공간 데이타 큐브의 효율적인 선택적 실체화 알고리즘에 대해 상세히 설명한다.

4.2.1 공간 데이타 큐브 생성

데이타 큐브에서 공간 차원과 연관된 공간 뷰를 추출하여 새로운 공간 데이타 큐브(S_D_cube)를 생성하기 위해 데이타 큐브(D_cube)에 속한 각 뷰(Vi)들을 조사한다. 이 뷰가 공간 뷰(S_view)에 속하면 S_D_cube에 첨가하고 모든 다른 뷰들에 대해서도 위와 같은 방법으로 수행하여 새로운 공간 데이타 큐브를 생성한다.

이러한 공간 데이타 큐브 생성 알고리즘은 그림 4와 같다.

```
For each view Vi ∈ D_cube Do
 if (Vi ∈ S_view) {
  Append a new S_view for Vi to view
    of a new S_D_cube
  Increase i of Vi
 }
```

그림 4. 공간 데이타 큐브 생성 알고리즘

4.2.2 실체화 후보 공간 뷰 생성

공간 데이타 큐브의 각 공간 뷰에서 공간 데이타 영역을 나타내는 셀(Cell)들이 얼마나 서로 공통으로 교차하고 있는지 조사한다. 셀의 공간 데이타 영역(Spatial_Portion)을 표현하는 구성 요소들이 교차된 영역(Intersected_Portion)이 있다면 실체화할 수 있는 실체화 후보 테이블(Candidate_Table)을 생성하여 저장한다. 즉 공간 데이타 큐브의 모든 공간 뷰(View_j)내에 존재하는 모든 셀과 셀(Cell_i)사이에 최대 교차 영역을 구하는 것이다. 이러한 실체화 후보 공간 뷰 생성 알고리즘은 그림 5와 같다.

```
Procedure
get_max_intersection(cell_i,view_j,candidate_tab
le) {
    Store first cell of View_j to Cell_j
    While (not empty cell_j) Do {
      Compare spatial_portion of cell_i to
              spatial_portion of cell_j
     If (exist intersected_portion) {
       Create candidate_table
       Store intersected_portion to
         candidate_table
     }
    }
}
```

그림 5. 실체화 후보 공간 뷰 생성 알고리즘

4.2.3 접근 빈도와 계산 시간에 의한 여과

4.2.2 절에서 생성된 실체화 후보 공간 뷰 중에서 실제로 실체화하기 위한 뷰를 설정하기 위해 공간 측정에 대한 접근 빈도와 계산 시간을 추정하고 여과(Filtering)하여 조건에 맞지 않으면 후보 테이블에서 교차 영역(P)를 제거한다. 다시 말해서 교차 영역들에 대해 접근 빈도의 합(Max_Frequency)이 최소 접근 빈도(Min_Frequency)보다 작고 그 교차 영역의 계산 시간(C_time)이 추정된 최소 계산 시간(Low_c_time)보다 작으면 실체화 후보 공간 뷰 테이블에서 제거한다. 또한 교차 영역들에 대해 접근 빈도의 합이 최소 접근 빈도보다 크고 그 교차 영역의 계산 시간이 추정된 최대 계산 시간(High_c_time)보다 크다면 실체화 후보 공간 뷰 테이블에서 제거한다. 이러한 여과 알고리즘은 그림 6과 같다.

```
Procedure
frequency_&_computation_time_computing_&_filtering(candidate_table) {
  For each entry P in candidate_table Do
    If (max_frequency(p) < min_frequency
       and c_time(p) < low_c_time) {
       Remove p from candidate_table }
    If (max_frequency(p) > min_frequency
       and c_time(p) > high_c_time) {
       Remove p from candidate_table }
}
```

그림 6. 접근빈도와 계산시간에 의한 여과 알고리즘

4.2.4 인접한 공간 객체 선택

4.2.3 절에서 접근 빈도와 계산 시간에 의해 여과되어 생성된 실체화 후보 공간 뷰들의 각각의 공간 측정 객체 영역들이 서로 인접(Connectivity)한 공간 객체(Object) 인지를 공간 객체 인접 테이블(Object_Neighbor_Table)을 참조하여 검사하고 인접한 공간 객체만을 추출한다. 추출된 공간 객체들은 인접한 공간 객체 테이블(Connected_object_table)에 저장된다.

이러한 인접한 공간 객체 선택 알고리즘은 그림 7과 같다.

```
Procedure
spatial_connectivity_testing(candidate_table, connected_object_table) {
  For each entry P in candidate_table Do
    Using object_neighbor_table
    Find the set of connected object(g)
    Store g to connected_object_table
    Remove g from candidate_table
}
```

그림 7. 인접한 공간 객체 선택 알고리즘

4.2.5 공간 객체 합병 공유

4.2.3 절에서 여과되어 생성된 실체화 후보 테이블의 모든 뷰의 그룹(G)에 대하여 4.2.4 절에서 생성된 인접한 공간 객체 테이블의 공간 객체 그룹들이 내포하고 있는지 비교한다. 그 결과 포함하고 있다면 합병된 공간 객체 테이블(Merged_Object_Table)을 생성하여 공간 객체 그룹을 저장하고 합병된 공간 객체 테이블의 식별자를 이용하여 실체화 후보 테이블의 뷰 그룹에 이를 대체하여 사용한다. 이것은 공간 데이타 큐브의 모든 셀과 연결되도록 포인터 식별자로 나타낼 수 있다. 이러한 공간 객체 합병 공유 알고리즘은 그림 8과 같다.

```
Procedure
shared_spatial_merging(connected_object_table, merged_object_table) {
  For G in candidated_object_table Do {
    If (not premerged G's subsets) {
       Premerge G's subsets to merged_
       object_table
    }
  }
  Populate spatial_data_cube
}
```

그림 8. 공간 객체 합병 공유 알고리즘

4.2.6 알고리즘 평가

공간 데이타 큐브에서 선택적 실체화 문제들을 해결하기 위해 기존의 알고리즘에서는 공간 질의 처리를 위한 공간 연산들의 계산 시간을 고려하지 않고 공간 질의 접근 빈도수를 가정하여 제시하였다.

본 논문에서는 공간 질의를 효율적으로 처리하기 위해 데이타 큐브를 비공간, 공간 데이타 큐브로 분류하여 선택적 실체화의 성능을 향상시켰고 또한 공간 연산들의 계산 시간도 고려하여 실체화함으로서 알고리즘의 효율성도

증가시켰다. 물론 공간 질의 접근 빈도수와 계산 시간의 효율적인 Threshold를 결정하는 것은 전체적인 응용 분야와 사용자 질의 요구 분석을 통하여 설정하는 것이 바람직하고 계산 시간은 뷰의 크기에 비례한다고할 수 있다.

V. 결론 및 향후 연구 과제

공간 데이타 웨어하우스의 OLAP환경에서 공간 질의들은 공간 합병, 공간 overlay, 공간 죠인, 공간 교차등 대부분 aggregation 함수등이 포함된 복잡한 공간 질의 연산을 수행하여야 하므로 신속한 의사 결정 지원을 위해 사용자가 원하는 시간 내에 응답하기가 어렵다. 그러므로 처리 시간을 최소화하고 또한 유지 비용을 최소화하기 위해 자주 발생하는 공간 질의들을 미리 계산하여 결과를 실체화하는 것이 가장 바람직하다

본 논문에서는 공간 질의를 효율적으로 처리하기 위하여 데이타 큐브를 비공간, 공간 데이타 큐브로 분류하였고 공간 연산들의 계산 시간도 고려하여 개선된 공간 뷰 선택 알고리즘을 제시하였다.

향후 연구 방향으로는 공간 데이타 큐브의 뷰에 대한 계산 시간은 뷰의 크기에 비례하므로 실체화 대상이 되는 뷰들의 크기를 효율적으로 추정하는 방법과 이런 방법의 연구가 선행되면 본 논문에서 제안한 알고리즘에 대한 시뮬레이션이 필요하다.

참고문헌

[1] Barguin, R.C. and Edelstein, H.A., "Planning and Designing the Data Warehouse", Prentice Hall, Inc., 1997.

[2] Butler Group, "Data Warehouse Issues", White Paper of Burler Group, 1997, available via http://www.butlergroup.co.uk.

[3] ESRI, "Spatial Data Warehouse", White Paper of ESRI, 1997, available via http://www.esri.com.

[4] Gupta, A., Harinarayan, V., and Quass, D., "Aggregate-Query Processing in Data Warehousing Environments", Proceedings of the 21st VLDB Conference Zurich, Swizerland, pp. 358-369, 1995.

[5] Gray, J., Bosworth, A., Layman, A., Pirahesh, H., "Data Cube: A Relational Aggregation Operator Generalizing Group-by, Cross-Tab, and Sub-Totals", Technical Report MSR-TR-95-22, 1995.

[6] Han, J., " Spatial Data Mining and Spatial Data Warehousing", SSD'97 Conference Tutorial, 1997, available via http://db.cs.sfu.ca.

[7] Harinarayan, A., Rajaraman, A., Ullman, J. D., "Implementing Data Cube Efficiently", Proceeedings of ACM SIGMOD, pp.311-322, 1995.

[8] Mumick, I.S., Quass, D., "Maintenance of Data Cubes and Summary Tables in a Warehouse", Proceeedings of ACM SIGMOD, May, 1997.

[9] Quass, D., "Materialized Views in Data Warehouse", Ph.D Thesis, Stanford University, Department of Computer Science, 1997.

[10] Stefanovic, N., "Design and Implementation of On-Line Analytical Processing (OLAP) of Spatial Data", M.S. Thesis, Simon Fraser University, Canada, Sept., 1997.

[11] Ullman, J. D., "Efficient Implementation of Data Cubes via Materialized Views", survey of the field for the 1996 KDD conference.

[12] 김수웅, 장지숙, 이전영, "데이타 웨어하우스 환경에서 계층 구조의 데이터 셋을 이용한 실체 뷰

의 크기 추정 방법", 한국 데이터베이스 학술대회 논문집, 제15권 1호, pp.9-15, 2월 1999.
[13] 이태희, 이익훈, 장재영, 이상구, "다차원 질의 모델에 기반한 OLAP 도구의 설계와 구현", 한국 데이터베이스 학술대회 논문집, 제15권 1호, pp.23-28, 2월 1999.
[14] 편성원, 장지숙, 이전영, "데이타 웨어하우스에서 대상 가지치기 기법을 이용한 뷰와 인덱스의 실체화", 한국 정보과학회 논문집, 제24권 2호, pp.69-72, 11월, 1997.

저 자 소 개

이 기 영
한국OA학회 논문지 제3권 제 4호 참조
현재 서울보건대학 사무자동화과 조교수