공간 데이터 웨어하우스에서 공간 분석을 위한 공간 집계연산†

Spatial Aggregations for Spatial Analysis in a Spatial Data Warehouse

유병섭* / ByeongSeob You, **김경배**** / GyoungBae Kim
이순조*** / SoonJo Lee, **배해영****** / HaeYoung Bae

요 약

공간 데이터 웨어하우스는 공간 의사결정을 지원하는 시스템으로 공간 데이터 큐브를 이용한다. 공간 데이터 큐브에는 분석의 기준이 되는 공간 차원테이블과 분석의 대상이 되는 공간 사실테이블들로 구성되는데 의사결정 지원을 위해서는 공간 차원테이블의 개념계층 지원과 공간 사실테이블의 요약정보 제공이 필요하다. 그러나 기존의 연구들은 공간 개념계층에 대해서만 연구하였을 뿐 공간 요약정보에 대한 연구가 미비하였다. 따라서 본 논문에서는 공간 데이터 웨어하우스에서 공간 공간 요약정보를 위한 공간 집계연산에 대하여 제안한다. 본 논문에서는 공간 집계연산을 숫자화 집계연산과 객체화 집계연산으로 나누어 제안한다. 숫자화 집계연산은 공간 분석의 결과로 숫자 형태의 데이터를 반환하며, 객체화 집계연산은 공간 객체 형태로 결과를 반환한다. 본 논문에서는 확장된 공간 데이터 자료구조를 제공하여 공간 집계연산의 효율성을 높인다.

Abstract

A spatial data warehouse is a system to support decision making using a spatial data cube. A spatial data cube is composed of a dimension table and a fact table. For decision support using this spatial data cube, the concept hierarchy of spatial dimension and the summarized information of spatial fact should be provided. In the previous researches, however, spatial summarized information is deficient. In this paper, the spatial aggregation for spatial summarized information in a spatial data warehouse is proposed. The proposed spatial aggregation is separated of both the numerical aggregation and the object aggregation. The numerical aggregation is the operation to return a numerical data as a result of spatial analysis and the object aggregation returns the result represented to object. We provide the extended struct of spatial data for spatial aggregation and so our proposed method is efficient.

† 본 연구는 건설교통부 첨단도시기술개발사업 – 지능형국토정보기술혁신사업 과제의 연구비지원(과제번호: 07국토정보CO5)에 의해 수행되었습니다.

■ 논문접수 : 2007.7.16 ■ 심사완료 : 2007.10.2

* 교신저자 인하대학교 대학원 정보공학과 박사과정(bsyou@dblab.inha.ac.kr)
** 서원대학교 컴퓨터교육과 조교수(gbkim@seowon.ac.kr)
*** 서원대학교 컴퓨터정보통신공학부 부교수(sjlee@seowon.ac.kr)
**** 인하대학교 대학원 원장(hybae@inha.ac.kr)

주요어 : 공간 데이터 웨어하우스, 공간 의사결정, 공간 집계연산
Keyword : Spatial Data Warehouse, Spatial Decision Support, Spatial Aggregation

1. 서 론

최근 빠르게 변화하는 시장에 효율적으로 대응하기 위하여 과거 축적된 데이터를 기반으로 신속한 의사결정을 지원하는 것이 중요하게 되었다[1, 2, 3]. 특히 GPS와 모바일 기기 등의 발달로 인한 위치데이터는 지리정보와 결합하여 의사결정의 중요한 요소로 자리 잡고 있다[4]. 따라서 기업에서는 기존의 경영에 필요한 정보에 공간 정보를 통합하여 입지선정 등의 공간 의사결정을 요구하고 있으며, 이를 해결하기 위하여 공간 데이터 웨어하우스에 관심이 집중되고 있다. 공간 데이터 웨어하우스는 기존의 데이터 웨어하우스에 공간 정보의 분석을 할 수 있도록 확장한 시스템으로 공간 데이터 큐브를 이용한다[5, 6]. 공간 데이터 큐브는 공간 및 비공간 데이터를 각각 하나의 차원 축으로 구성하여 각 차원 축의 특정 값이 교차하는 셀에 분석결과로 활용되는 사실 데이터를 관리하는 구조를 갖는다. 이때 셀에 존재하는 데이터는 그 특성상 반드시 집계정보로 표현된다.

공간 차원테이블에 대해서는 Hybrid Index[7], aR-tree[8], OLAP Favored Search[9] 등과 같이 인덱스를 이용한 공간 계층구조를 지원하는 연구들이 진행되었다. 이 연구들에서는 공간 개념계층에 대해 사전집계정보를 제공하였으나 단지 비공간 데이터에 대한 집계정보만을 제공하였다.

공간 데이터 큐브의 사실테이블은 사실 데이터가 공간 데이터 일 경우 공간 데이터의 요약정보를 제공하여야 하는데, 아직까지 이를 공간 집계연산에 대한 연구가 미비하였다.

기존의 공간 집계연산에 대한 연구는 [10]과 [11]에서 온도를 계산하기 위해 사용되었다. [10]에서는 공간 집계연산을 위해 최종 공간 데이터를 여러 개의 공간 데이터로 분리하여 집계하고 이들을 합산하는 연산에 대해서 계산비용 감소 및 정확성 향상을 위한 방법을 논의하였다. 그러나 단지 분산 처리일 뿐 공간 의사결정 지원을 위해서는 적합하지 않았다. [11]에서는 공간 객체들에 대하여 온도 및 강우량에 대한 집계정보를 유지하는 기법에 대하여 연구하였다. 하지만 이는 단지 특정 영역에서 가장 높은 온도 또는 가장 낮은 온도 등에 대한 처리만을 제공하였기 때문에 공간 데이테웨어하우스의 공간 요약정보를 위해서는 부족하다.

따라서 본 논문에서는 공간 데이터 웨어하우스에서 공간 분석을 지원하기 위한 공간 집계연산에 대하여 제안한다. 제안기법을 위해 본 논문에서는 확장 공간 데이터 자료구조를 이용한다. 이 구조에서는 공간 객체의 숫자화 집계연산을 효율적으로 처리할 수 있도록 객체의 면적과 개수에 대한 사전 집계정보를 갖는다. 따라서 숫자형태의 공간 집계연산을 처리에 있어 성능향상을 가져온다.

제안기법에서는 공간 집계연산을 결과의 형태에 따라 숫자화 공간 집계연산과 객체화 공간 집계연산으로 나누는데, 결과가 면적과 같이 숫자 형태로 반환되는 숫자화 연산과 공간 객체의 형태로 반환되는 객체화 연산으로 나뉜다. 숫자화 집계연산은 공간 객체에 대한 면적이나 객체의 개수를 처리하는것으로 sum, count, avg, max, min 등이 있는데 이는 용도별 토지면적의 변화 등에 대한 응용에 활용할 수 있다. 객체화 집계연산의 경우 그룹에 속한 공간 객체들을 하나의 공간 객체로 반환하는 것으로 boundary, merge 등이 있으며 이는 공간 데이터의 분포 형태 등을 제공하여 고객의 주거지 분포 등을 지원할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 설명하고, 3장에서는 공간 분석을 위한 공간 집계연산에 대하여 설명한다. 4장에서 성능평가를 하며, 5장에서 결론을 맺는다.

2. 관련연구

2.1 공간 데이터 웨어하우스

공간 데이터 웨어하우스는 기존의 비공간 데이터 웨어하우스에 공간 개념을 추가한 것으로 공간 및 비공간 데이터를 주제 중심적(subject-oriented), 통합적(integrated), 영속적(time-variant), 비휘발적(nonvolatile)으로 수집하여 의사결정을 지원하는 시스템이다[1, 3]. 공간 데이터 웨어하우스에서는 공간 의사결정 지원을 위하여 SOLAP(Spatial On-Line Analytical Processing)을 이용하는데, SOLAP이란 기존의 온라인 분석 처리인 OLAP에 공간 분석을 지원하는 기법으로 공간 및 비공간 데이터의 효율적인 분석을 지원한다[12, 13]. SOLAP은 분석 데이터를 일반화 시키는 Rollup 연산, 상세화 시키는 Drilldown 연산, 분석의 기준 차원을 바꾸는 pivot 연산, 특정 차원의 일정 범위 데이터를 기준으로 데이터를 분석하는 Slice 연산, 여러 차원의 일정 범위 데이터를 기준으로 데이터를 분석하는 dice 연산을 이용하여 다각화된 분석을 지원한다. 이러한 분석을 효율적으로 지원하기 위해서는 주제 중심적인 데이터 관리가 필요한데, 공간 데이터 웨어하우스는 공간 데이터 큐브를 이용한다.

2.2 공간 데이터 큐브

공간 데이터 큐브는 기존의 데이터 큐브에 공간 개념을 추가한 것으로 공간 의사결정을 위하여 사용한다[14]. 데이터 큐브는 차원 테이블과 사실 테이블로 구성되는데, 차원 테이블이란 고객 차원, 상품 차원, 시간 차원 등 분석을 위한 기준이 되는 데이터가 저장된 테이블이다. 차원 테이블들은 각각 독립적으로 Rollup, Drilldown 등의 OLAP 연산들을 지원하여 다각적인 분석을 제공한다. 사실 테이블이란 각 차원 테이블을 연결하고 이 연결된 기준에 대한 분석 정보를 관리하는 테이블이다. 사실 테이블에서 관리되는 분석 정보는 같은 기준에 속하는 데이터들의 집계 값을 갖게 된다. 따라서 사실 테이블의 분석 정보는 모두 집계연산으로 구성된다.

데이터 큐브에 공간 개념을 추가한 공간 데이터 큐브는 차원 테이블로 공간 개념을 추가하는 방법과 사실 테이블로 공간 개념을 추가하는 방법이 있다. 전자는 차원 테이블 중 하나가 공간 데이터로 이루어진 형태로 공간 의사결정을 위한 분석에 있어 하나의 기준으로 사용된다. 공간 차원 테이블을 이용한 기법은 여러 기법들이 연구되었는데, 대부분 공간 인덱스를 사용하여 공간 개념 계층을 지원하는 연구를 진행하였다 [8, 9]. 후자는 사실 테이블의 분석 정보 중 하나가 공간 데이터로 구성된 형태로 공간 의사결정을 위한 분석 정보로 사용된다. 이때 사실 테이블의 공간 데이터는 공간 집계연산을 통한 집계정보가 요구되는데 현재까지 공간 분석을 위한 공간 집계연산에 관한 연구는 미비하였다.

2.3 공간 집계연산

기존의 공간 집계연산에 대한 연구는 [10]과 [11]이 있었다.

[10]에서는 공간 데이터의 집계정보 계산에 대한 연구로 집계정보를 요청하는 공간객체를 먼저 여러 개의 작은 공간 객체들의 집합으로 나눈다. 이렇게 나누어진 공간 객체들에 대하여 각각 집계정보를 계산하며, 나누어진 공간 객체들에 대한 집계정보를 합하여 최종 집계정보를 계산한다. 이 기법에서는 공간 집계연산을 간략하게 정의하였으며, 이를 효율적으로 계산하기 위해 계산비용의 감소 및 정확성 향상을 위한 방법을 제안하였다. 그러나 이는 단지 공간 객체의 온도 등과 같은 비공간 정보의 집계연산만을 제공하여 공간 분석을 위해 요구되는 공간 집계정보를 제공하지 못한다.

[11]에서는 공간 객체들에 대하여 온도 및 강우량에 대한 max, min 집계연산에 대하여 논의하였다. 이 기법에서는 각 공간 객체들에 대한 온도를 사전에 집계하여 저장하며, 특정 지역의 가장 높은 온도를 갖는 공간 객체를 반환하기 위하여 공간인덱스인 R*-tree를 이용하였다. 하지만 이 기법도

특정 분야에만 적용이 가능하며, 공간 데이터 웨어하우스에서 요구하는 공간 집계연산으로 적합하지 않았다.

3. 공간 분석을 위한 공간 집계연산

본 장에서는 공간 의사결정 지원을 위한 공간 집계연산을 제안한다. 제안 기법을 위하여 먼저 공간 데이터웨어하우스의 공간 집계연산을 정의한다. 이 정의를 통해 요구되는 공간 집계연산을 효율적으로 지원하기 위한 공간 데이터 관리 구조를 제안하며, 이를 이용한 공간 집계연산을 제안한다.

3.1 공간 집계연산

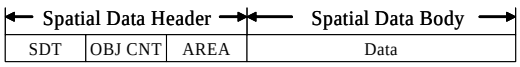
공간 데이터웨어하우스에서 공간 집계연산이 요구되는 경우는 사실테이블에 공간 데이터가 위치하는 경우로 분석의 대상이 공간 데이터가 된다. 공간 데이터웨어하우스는 과거부터 현재에 이르기까지 데이터를 시간별 변화를 주로 처리하기 때문에 시간에 따른 공간 데이터의 집계정보 변화를 요구한다. 따라서 공간 데이터 웨어하우스는 시간의 변화에 따른 공간 데이터의 집계연산이 요구된다.

공간 집계연산은 그 결과에 따라 숫자화 공간 집계연산과 객체화 공간 집계연산으로 나눌 수 있는데 이는 각각 연산의 결과가 비공간 데이터인 경우와 공간 데이터인 경우이다. 전자는 “인천시에 대해 2000년부터 2007년까지 연도별 주거면적의 변화를 조사하라.” 또는 “인천시에 대해 2000년부터 2007년까지 각 연도에 따른 공업지구의 사업체별 평균면적의 변화를 조사하라.”와 같이 공간 객체에 대한 면적을 다루는 연산이다. 대부분의 공간 분석에서는 이와 같이 공간 데이터의 면적에 의한 집계연산을 요구한다. 후자는 “인천시에 위치한 A마트에 대하여 2000년부터 2007년까지 연도별 이용고객의 분포 변화를 조사하라.” 또는 “인천시에서 2000년부터 2007년까지 연도별 수해지역 범위의 변화를 조사하라.”와 같이 조건을 만족하는 공간 객체들의 분포나 공간 객체들을 포함하는 영역을 요구한다.

따라서 공간 데이터 웨어하우스에서 공간분석을 위해 공간 객체에 따른 면적에 대한 공간 집계연산과 공간 객체들의 분포에 대한 공간 집계연산들의 연구가 필요하다.

3.2 공간 집계연산을 위한 데이터 관리

공간 데이터 웨어하우스의 숫자화 공간 집계연산과 객체화 공간 집계연산은 각각 연산하는 방식이 다르다. 숫자화 공간 집계연산의 경우 공간 데이터의 면적 및 객체의 수를 이용한 연산이 요구되며, 객체화 공간 집계연산의 경우 공간 객체들간의 합병이 요구된다. 따라서 본 논문에서는 공간 집계연산의 효율성을 위한 데이터 관리 방법을 제공한다.



<그림 1> 공간 데이터 관리 구조

<그림 1>은 제안기법을 위해 본 논문에서 사용하는 공간 데이터의 관리 구조를 나타낸다. 본 논문에서는 공간 데이터를 관리하기 위하여 크게 공간 데이터 헤더(Spatial Data Header)와 공간 데이터 몸체(Spatial Data Body)로 나눈다. 공간 데이터 헤더에서는 해당 공간 객체의 정보를 가지며, 공간 데이터 몸체에서 실제 공간 객체의 좌표값들에 대하여 관리한다. 공간 데이터 헤더에는 4개의 객체 정보를 갖는데, 각각은 다음과 같다. SDT(Spatial Data Type)는 공간 객체의 타입에 대한 정보를 갖는다. 객체의 타입이란 point, polyline, polygon 등 공간 객체를 표현하는 형태를 말한다. OBJ_CNT(Object Count)는 공간 객체의 개수를 갖는다. 공간 객체의 경우 단일 공간 객체는 하나의 공간 객체로 구성되지만 multi-point, multi-polyline, multi-polygon 등과 같은 다수의 공간 객체들로 구성되는 경우도 있다. 이때 OBJ_CNT는 다수의 공간 객체 각각을 계산한 공간 객체의 수를 저장한다. AREA는 공간 객체에 대한 면적의

값을 가지며, 공간 객체의 형태가 point이거나 polyline일 경우 AREA 값을 가지지 않는다.

제안기법에서 공간 객체의 정보를 헤더로 가짐으로써 공간 집계연산에서의 면적 계산이나 공간 객체간의 합집합에 대해 효율적으로 연산할 수 있다. 면적 계산의 경우 매 집계연산 호출시마다 각각의 공간 객체들에 대한 면적을 구성 좌표값들을 이용하여 구하는 것이 아니라 헤더에 갖고 있는 AREA 값을 이용하여 빠르게 구할 수 있다. 이는 공간 객체를 구성하는 점의 수가 많을수록 더 효율적인 면적 계산을 가능하게 한다.

3.3 숫자화 공간 집계연산

숫자화 집계연산에는 sum, avg, count, max, min의 다섯 가지 연산이 있으며, 본 절에서는 이 연산들에 대하여 공간 데이터를 적용한 숫자화 공간 집계연산을 설명한다.

3.1.1 sum

sum 연산은 숫자 데이터에 대하여 그 값들을 합한 값을 결과로 갖는다. 즉, sum(A)와 같이 기술하면, 속성 A에 속하는 값들의 합을 결과로 반환한다. 그러나 공간 데이터의 경우 숫자가 아닌 좌표값들로 이루어져 있으므로 이에 대한 다른 처리방법이 요구된다. 공간 데이터 웨어하우스에서는 주거면적과 같이 공간 객체들의 면적에 대한 집계정보를 요구하므로 sum의 연산은 공간 객체의 면적에 대한 연산으로 표현된다. 따라서 공간 데이터에 대한 sum 연산은 다음과 같이 정의할 수 있다.

(정의 1) 임의의 테이블 T의 공간 데이터 필드 S를 T_S라고 하자. T_S에 속하는 공간 데이터들이 면적을 계산할 수 있는 폴리곤(polygon) 타입이라고 할 때, 임의의 조건을 만족하는 그룹 G의 집계함수 sum(T_S)는 그룹 G에 속하는 공간 데이터들에 대한 면적의 합이다.

$$sum(T_S) = \{x \mid x = \sum area(y);$$
$$\exists y \leftarrow (y \in G \wedge y \subset (polygon \vee multi-polygon))\}$$

(정의 1)에서 sum 집계연산은 공간 데이터의 면적들에 대한 합을 계산한다. 이때 공간 데이터의 면적을 계산하기 위해서는 공간 객체들이 닫혀있어야 한다. 따라서 폴리곤 타입의 공간 데이터만이 sum 집계연산을 지원한다. 다음 <그림 2>는 sum 집계연산 알고리즘에 대하여 나타낸다.

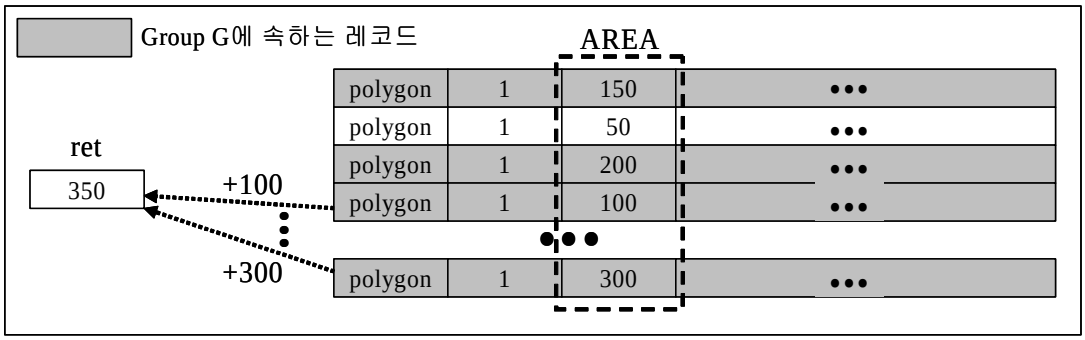
```
01: sum ( TS )
02: {
03:     ret : 반환값
04:
05:     ret = 0
06:     if the type of TS is polygon or multi polygon
            then
07:         for each record R contained group G in a
                table
08:           add to ret the area of spatial data TS in R
09:       end for
10:   else
11:       ret = 1; // 닫혀있지 않는 공간 객체일 경우 에러
12:   end if
13:
14:   return ret
15: }
```

<그림 2> sum 연산 알고리즘

<그림 2>에서 보면 6번째 줄에서 공간 객체 T_S가 닫혀있는 객체인 polygon 또는 multi-polygon인지를 판별한다. 이후 테이블에서 그룹 G에 속하는 모든 레코드 *R*에 대하여 8번째 줄에서 공간 객체의 넓이를 *ret*에 더한다.

<그림 3>은 sum 연산의 실제 수행과정을 나타낸다.

<그림 3>에서 보면 진한 레코드가 그룹에 속하는 레코드이고 현재 3번째 레코드까지 연산을 수행한 상태이다. 여기서 그룹에 속하는 1, 3번 레코드는 그 면적을 반환하는데 이때 공간 데이터 헤더에 미리 계산된 AREA 값을 이용한다. 그 값이 각각 150과 200이므로 현재 ret의 값은 350이다. 다음



<그림 3> sum 연산 수행과정

레코드인 4번째 레코드도 그룹에 속하며 이 레코드의 AREA 값은 100 이므로 이 레코드의 집계연산 과정이 수행되면 ret 값은 100을 더한 값인 450으로 변한다. 이와 같이 모든 레코드를 모두 연산하게 되면 sum 연산의 결과인 면적의 합을 반환하게 된다.

공간 데이터 헤더에서 AREA 정보를 가짐으로써 sum 집계연산을 호출할 때마다 공간 데이터의 면적을 계산하지 않아도 되므로 연산비용을 감소시킨다.

3.1.2 count

count 연산은 어떠한 그룹에 속하는 레코드의 수를 반환하는 연산이다. 그러나 이는 비공간 데이터에 대한 연산에서의 정의로 공간 데이터의 경우는 multipolygon과 같이 하나의 레코드에도 여러 개의 공간 객체들이 존재하게 된다. 따라서 공간 데이터의 타입 중 multipoint, multipolyline, multipolygon, geometrycollection 등은 그 객체에 포함된 공간 객체들의 수까지 계산하여야 한다. 예를 들어 어떤 지역의 총 건물의 수를 구할 때 만약 아파트가 multipolygon으로 표현되어 있다면 이는 multipolygon에 속한 각각의 polygon 객체들을 하나로 계산하여야 한다. 따라서 공간 데이터의 count 연산은 다음과 같이 정의한다.

(정의 2) 임의의 테이블 T의 공간 데이터 필드 S를 T_S라고 하자. 임의의 조건을 만족하는 그룹 G의 집계함수 count(T_S)는 그룹 G에 속하는 공간 데이터들에 대한 객체의 수이다. 이때, multipoint, multipolyline, multipolygon, geometrycollection 등은 그 안에 포함된 객체들의 수를 집계한다.

$$count(T_S) = \{x \mid x = \sum count\_obj(y), \exists y \leftarrow (y \in G)\}$$

(정의 2)에서 count_obj(y)는 객체 y에 대해서 그 안에 속하는 모든 객체의 수를 반환하는 함수를 의미한다. 즉, 정의대로 모든 객체들의 개수를 반환한다. <그림 4>는 공간 객체의 count 집계연산에 대하여 나타낸다.

<그림 4>에서 보면 각 레코드에 대해서 단일 객체로 이루어진 point, polyline, polygon과 다중 객체로 이루어진 multi-point, multi-polyline, multi-polygon으로 나누어 계산한다. 전자의 경우 객체의 개수가 하나이므로 11번째 줄과 같이 ret 값에 1을 더하며, 후자의 경우 객체의 수가 여러 개이므로 16~18번째 줄과 같이 해당 다중객체에 포함되어 있는 객체들의 수를 더한다. 만일 객체의 타입이 geometrycollection일 경우 해당 객체에 다양한 형태의 공간 객체들이 포함되어 있으므로 21~29번째 줄과 같이 각각에 대하여 공간 객체의 수를 계산한다.

<그림 5>는 count 연산의 실제 수행과정을 나타낸다.

```
01: count(TS)
02: {
03:     ret : 반환값
04:
05:     ret = 0
06:     for each record R contained group G in a table
07:         switch( the type of TS )
08:         case point:
09:         case polyline:
10:         case polygon:
11:             ret = ret + 1
12:             break
13:         case multi  point:
14:         case multi  polyline:
15:         case multi  polygon:
16:             for each object contained in multi object of
                    TS
17:                 ret = ret + 1
18:             end for
19:             break
20:         case geometrycollection:
21:             for each object O in TS
22:                 if type of object O is point, polyline or
                        polygon then
23:                     ret = ret + 1
24:                 else
25:                     for each object contained in multi
                            object O
26:                         ret = ret + 1
27:                     end for
28:                 end if
29:             end for
30:             break
31:         end switch
32:     end for
33: }
```

<그림 4> count 연산 알고리즘

<그림 5>에서 보면 현재 공간 객체들이 모두 multi-polygon 객체이며 3번째 레코드까지 연산을 수행하였다. 이때 ret 값은 7로 첫번째 레코드와 세 번째 레코드의 OBJ CNT값을 이용하여 계산되었다. 4번째 레코드의 경우도 마찬가지로 OBJ_CNT에 미리 계산하여 둔 공간 객체의 수 3을 ret에 더하게 된다. 마지막 레코드까지 이 과정을 반복하고 더 이상 레코드가 없으면 ret의 값을 반환하게 된다.

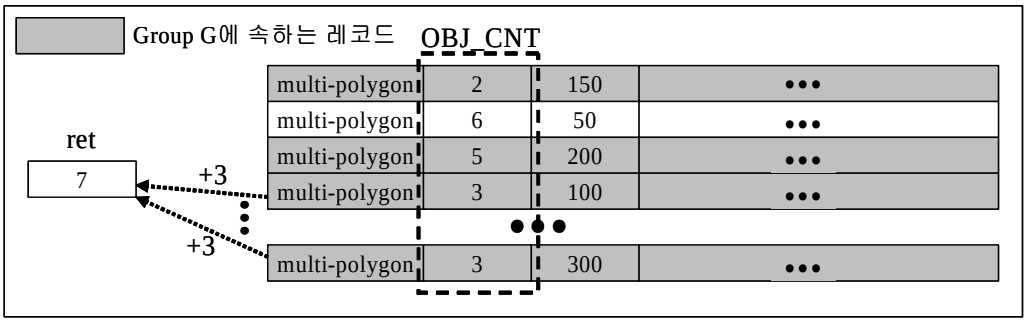
그림에서와 같이 OBJ_CNT를 공간 데이터 헤더에 가짐으로써 count 집계연산 수행시 매번 공간 데이터에 포함된 모든 객체들의 수를 찾기 위한 비용이 감소하며, 이는 전체적인 연산 비용의 감소를 가져온다.

3.1.3 max

max 연산은 그룹내 데이터에 대해 가장 큰 값을 반환하는 연산이다. 공간 데이터 웨어하우스에서의 max 연산은 각 객체들의 면적의 크기가 가장 큰 값을 반환하는 연산으로 다음과 같이 정의된다.

(정의 3) 임의의 테이블 T의 공간 데이터 필드 S를 T_S라고 하자. 임의의 조건을 만족하는 그룹 G의 집계함수 $\max(T_S)$는 그룹 G에 속하는 닫힌 공간 데이터들에 대해서 가장 큰 객체의 면적을 반환하는 연산이다.

$$\max(T_S) = \{x \mid x = \max(area(y)),$$
$$\exists y \leftarrow (y \in G \wedge y \subset (polygon \vee multi - polygon))\}$$



[그림 5] count 연산의 실제 수행과정

max 연산도 다른 연산과 비슷하게 공간 객체의 면적을 연산을 하고, 이에 따른 면적이 가장 큰 결과를 가져오며, <그림 6>의 알고리즘과 같이 동작한다.

```
01: max(TS)
02: {
03:     ret : 반환값
04:
05:     ret = 0
06:     if the type of TS is polygon or multi polygon
          then
07:       for each record R contained group G in a table
08:         if the area A of spatial data in R is higher
              than ret then
09:           ret = A
10:         end if
11:       end for
12:     else
13:       ret = 1; // 닫혀있지 않는 공간 객체일 경우 에러
14:     end if
15:
16:     return ret
17: }
```

<그림 6> max 연산 알고리즘

<그림 6>에서 보면 6번째 줄에서 공간 객체 T_S가 polygon 또는 multi-polygon인지를 판별한다. 7~11번째 줄에서 테이블에 속하는 레코드 중 그룹 G에 포함되는 레코드들에 대해서 현재 ret에 가지고 있는 값보다 큰 값의 면적이 있으면 ret에 그 값을 넣는다.

<그림 7>은 max 연산이 실제 수행되는 과정을 나타낸다.

<그림 7>에서 현재 3개의 레코드를 읽었고 그중 첫번째와 세번째 레코드가 그룹에 속하는 레코드이다. 이때 각 레코드의 면적 값은 각각 150, 200 이므로 큰 값인 200이 ret에 저장되어 있다. 이후 4번째 레코드를 읽으면 면적 값이 100 이므로 현재 ret 값보다 작아 버려진다.

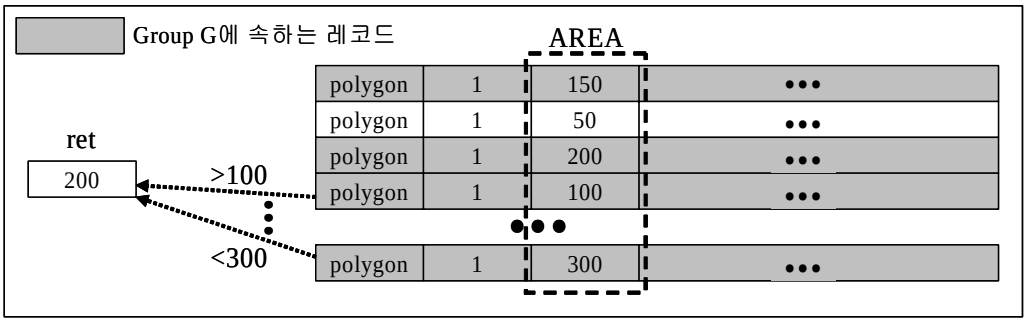
공간 데이터의 헤더에 AREA 정보를 가짐으로써 max 연산 수행시 각각의 공간 데이터에 대한 면적을 구하는 시간을 감소시켜 전체적인 연산비용의 감소를 가져온다.

3.1.4 min

min 연산은 공간 데이터의 면적 중에서 가장 작은 면적을 가지는 객체의 면적을 반환하는 연산으로 max 연산과 거의 유사하다. 이를 정의하면 다음 (정의 4)와 같다.

(정의 4) 임의의 테이블 T의 공간 데이터 필드 S를 T_S라고 하자. 임의의 조건을 만족하는 그룹 G의 집계함수 $\min(T_S)$는 그룹 G에 속하는 닫힌 공간 데이터들에 대해서 가장 작은 객체의 면적을 반환하는 연산이다.

$$\min(T_S) = \{x \mid x = \min(area(y)),$$
$$\exists y \leftarrow (y \in G \wedge y \subset (polygon \vee multi - polygon))\}$$



<그림 7> max 연산의 실제 수행과정

(정의 4)를 보면 min 연산이 max 연산과 거의 유사한 것을 알 수 있다. 즉, 이 두 연산은 결과로 가져오는 면적의 크기만 다를 뿐이다. 따라서 min 연산에 대한 알고리즘은 위 [그림 6]에서 8번째 줄을 ret 보다 작은 면적을 갖는 조건으로 바꾸면 되므로 여기서 min 연산의 알고리즘은 생략하도록 한다.

3.1.5 avg

avg는 평균값을 구하는 연산으로 비공간 데이터에 대하여 sum 연산의 결과 값을 count 연산의 결과 값으로 나눈 값으로 표현될 수 있다. 그러나 공간 데이터의 경우 sum 연산은 polygon 또는 multipolygon 등 닫혀있는 연산에 대해서만 적용이 가능하며, count 연산은 모든 공간 데이터에 가능하나 다중 공간 객체의 경우 각각의 객체를 하나로 하기 때문에 기존의 연산 방법과 다르다. 따라서 avg 연산은 다음과 같이 정의할 수 있다.

(정의 5) 임의의 테이블 T의 공간 데이터 필드 S를 T_S라고 하자. 임의의 조건을 만족하는 그룹 G의 집계함수 avg(T_S)는 그룹 G에 속하는 닫힌 공간 데이터들의 평균 면적을 반환하는 연산이다.

$$avg(T_S) = \{x \mid x = \sum area(y) \div \sum count\_obj(y),$$
$$\exists y \leftarrow (y \in G \wedge y \subset (polygon \vee multi - polygon))\}$$

(정의 5)에서 보면 그룹 G에 속하는 모든 객체들의 면적의 합을 구하고, 모든 객체들의 수를 구하여

```
01: avg(TS)
02: {
03:     sum : 면적 합계값
04:     count : 객체의 수
05:     ret : 반환값
06:
07:     if the type of TS is polygon or multi  polygon then
08:         for each record R contained group G in a table
09:             add to sum the area of spatial data in R
10:
11:             switch( the type of TS )
12:             case polygon:
13:                 count = count + 1
14:                 break
15:             case multi  polygon:
16:                 for each object contained in multi object TS
17:                     count = count + 1
18:                 end for
19:                 break
20:             end switch
21:         end for
22:         ret = sum / count
23:     else
24:         ret =   1; // 닫혀있지 않는 공간 객체일 경우 에러
25:     end if
26:
27:     return ret
28: }
```

<그림 8> avg 연산 알고리즘

전자를 후자로 나눈 값을 평균면적으로 반환한다. 이를 알고리즘으로 표현하면 <그림 8>과 같다.

<그림 8>에서 보면 7번째 줄에서 닫힌 객체인 polygon과 multi-polygon에 대해서만 연산을 수행하며, 8~21번째 줄에서 테이블의 레코드들 중 그룹 G에 속하는 레코드들에 대한 연산을 수행한다. 9번째 줄에서 속성 TS에 대한 공간 객체의 면적을 sum에 더하고, 11~20번째 줄에서 속성 TS에 속하는 공간 객체들에 대해서 단일 객체일 경우와 다중 객체일 경우에 따라 공간 객체들의 수를 더한다. 이렇게 계산된 값들은 22번째 줄에서 sum을 count로 나눔으로써 결과인 평균값을 구한다.

3.4 객체화 공간 집계연산

객체화 공간 집계연산은 공간 데이터를 하나의 다중 공간 객체로 표현되는 merge 연산과 여러 공간 데이터를 포함하는 볼록 다각형으로 표현되는 boundary 두 가지로 나누어진다. 본 절에서는 이 두 연산자에 대해 각각 설명한다.

3.2.1 merge

시간에 따른 구매자의 분포 현황 분석이나 용도별 건물의 분포 분석 등을 위해서는 각 조건에 따른 공간 데이터들을 모두 표현하여야 한다. 그러나 기존의 연구들에서 이를 제공하기 위해서는 모든 데이터를 읽어 클라이언트에서 그룹별 공간 데이터들을 판별하고 이를 표현하였다. 따라서 클라이언트에 따라 성능이 크게 저하되었고, 계층별 분석을 위해서는 매번 다시 분석해야 하는 비용이 발생하였다. 따라서 본 절에서는 공간 데이터의 분포 분석을 효율적으로 지원할 수 있는 merge 연산을 제안한다. merge 연산이란 어떤 그룹에 속하는 공간 데이터들을 모두 가지는 공간 데이터 집합체를 만드는 연산이다. 즉, 그룹에 속하는 공간 데이터들을 표현하는 하나의 공간 데이터로 합하는 연산이다. 따라서 merge 연산을 다음과 같이 정의한다.

(정의 6) 임의의 테이블 T의 공간 데이터 필드 S를 T_S라고 하자. 임의의 조건을 만족하는 그룹 G의 집계함수 merge(TS)는 그룹 G에 속하는 공간 데이터들의 집합이다.

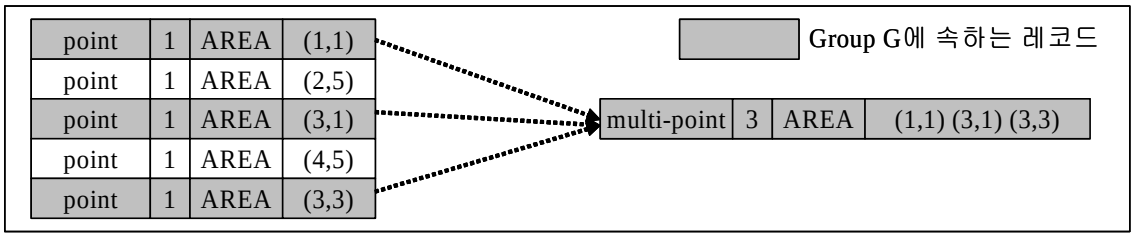
$$merge(T_S) = \{x \mid x = set(y), \exists y \leftarrow (y \in G)\}$$

공간 데이터들의 집합은 여러 개의 공간데이터들을 하나의 공간 데이터로 표현하는 것을 의미한다. 따라서, (정의 6)에 의하여 merge연산은 그 결과로 하나의 공간 데이터로 표현되는 공간 데이터들의 집합을 가져와야한다. 이는 merge연산이 결과로 다중 공간 데이터 타입을 갖는 것을 의미한다. 예를 들어 어떤 그룹에 속하는 공간 데이터가 모두 point 타입이었다면, merge연산의 결과는 multi-point 타입의 하나의 객체로 반환된다. 즉, 그룹내 모든 point 데이터들을 multi-point로 묶는 것이다.

merge 연산의 알고리즘은 다음 <그림 9>과 같다.

```
01: merge(TS)
02: {
03:     obj : 반환값
04:
05:     if the type of TS is point or multi  point then
06:         obj.SDT is multi  point
07:       else if the type of TS is polyline or multi
              polyline then
08:         obj.SDT is multi  polyline
09:       else if the type of TS is polygon or multi
              polygon then
10:         obj.SDT is multi  polygon
11:     else
12:         obj.SDT is geometry
13:     end if
14:
15:     for each record R contained group G in a table
16:         add object O of R to obj.objList
17:         obj.OBJ_CNT += O.OBJ_CNT
18:         if obj.SDT is not multi  point then
19:             obj.AREA += O.AREA
20:         end if
21:     end for
22:
23:     return obj
24: }
```

<그림 9> merge 연산 알고리즘



<그림 10> merge 연산의 수행과정

<그림 9>에서 보면 5~13번째 줄에서 merge 연산의 결과 데이터에 대한 형식을 결정하고 있으며, 15~21번째 줄에서 그룹 G에 속하는 레코드들의 T_S 속성의 공간 객체들에 대한 연산을 수행한다. 16번째 줄에서는 결과 공간객체의 데이터에 새로운 공간 객체를 추가하는 부분으로 최종 공간 객체의 좌표 리스트를 갖는다. 17번째 줄에서는 공간 데이터의 헤더에 결과 객체가 포함하는 총 객체의 수에 대한 연산을 수행한다.

<그림 10>는 merge 연산의 수행과정을 보여준다.

<그림 10>에서 보면 5개의 레코드들 중 그룹 G에 속하는 레코드는 1, 3, 5번째 레코드이다. 따라서 merge 연산을 수행하면 이 세 레코드들의 좌표들을 하나의 다중 좌표로 바꾸어 (1,1) (3,1) (3,3)으로 하나의 공간 객체로 표현된다. 이때 3개의 point 객체를 가지므로 merge 결과의 객체 타입 SDT는 multi-point가 되며, 총 3개의 포인트를 포함하고 있으므로 OBJ_CNT는 3으로 값을 갖는다.

<그림 10>에서와 같이 merge연산은 그 결과가 다중 공간 데이터 타입으로 나타나는데, 이를 정리하면 <표 1>과 같다.

<표 1> merge연산 전과 후의 공간 데이터 타입

merge 연산 전 데이터 타입	merge 연산 후 공간 데이터 타입
point multi-point	multi-point
polyline multi-polyline	multi-polyline
polygon multi-polygon	multi-polygon
geometry	geometry

<표 1>은 merge 연산을 하기 전의 공간 데이터 타입과 merge 연산 이후 반환되는 공간 데이터 타입을 보여준다. 여기서 point와 polyline, polygon은 단일 공간 데이터 타입이고, multi-point와 multi-polyline, multi-polygon, geometry는 다중 공간 데이터 타입이다. Merge연산은 그 결과로 공간 데이터의 집합을 반환하기 때문에 원래의 공간 데이터 타입과 관계없이 다중 공간 데이터 타입으로 결과가 표현된다. 따라서 단일 공간 데이터 타입인 point와 다중 공간 데이터 타입인 multi-point는 merge연산의 결과가 다중 공간 데이터 타입인 multi-point로 표현된다.

3.2.2 boundary

boundary 연산은 특정 그룹에 속하는 공간 객체들을 포함하는 최소 볼록 다각형이다. 이는 자동차를 소유하지 않은 백화점 고객의 분포 등의 분석을 할 때 사용되어 향후 백화점 셔틀버스의 정류장 등을 결정할 수 있도록 도와주게 된다. 이 이외에도 시간별로 백화점을 찾는 고객의 거주지 분포영역 분석 등 특정 그룹에 대한 분포 영역 분석을 위한 응용에 사용될 수 있다. 이를 위해서는 조건에 맞는 그룹을 분류하고 이 그룹에 속하는 공간객체들의 데이터를 포함하는 하나의 다각형을 만들어야 한다. 이때 모든 공간 객체를 포함하는 최소의 다각형이 가장 이상적이지만 계산 비용이 높고, 분포영역 자체가 근사값을 제공하면 되므로 최소 볼록 다각형을 결과로 가져오도록 한다. 따라서 boundary 연산을 다음과 같이 정의한다.

(정의 7) 임의의 테이블 T의 공간 데이터 필드 S를 T_S라고 하자. 임의의 조건을 만족하는 그룹 G의 집계함수 boundary(T_S)는 그룹 G에 속하는 공간 데이터들을 모두 포함하는 최소 볼록 다각형이다. 단, 그룹에 속하는 공간객체는 3개 이상이다.

(정의 7)에서 모든 그룹들은 그 결과로 최소 볼록 다각형을 반환하며, 이를 위해 그룹에는 반드시 3개 이상의 공간객체가 존재해야 한다. 또한 Boundary 연산은 영역을 반환하게 되므로 그 값은 항상 닫힌 다각형이 된다. 따라서 boundary 함수의 결과는 다음의 두가지 특징을 갖는다. 첫째로 다각형의 각 점은 그룹 내 하나의 객체와 일치한다. 최소 다각형을 구하기 위해서는 각 점을 기준으로 각도와 함께 계산하게 되므로 그 점은 모두 각각 그룹 내 하나의 객체와 대응하게 된다. 둘째로 연산의 기반 데이터의 형태와 상관없이 하나의 폴리곤 형식의 결과를 반환한다. 이는 boundary 연산의 기반 데이터가 point 형식이든 polygon 형식이든 상관없이 모든 결과는 최소 볼록 다각형을 반환해야 하므로 polygon 객체 하나로 반환되는 것을 의미한다. <그림 11>은 boundary 연산의 알고리즘을 나타낸다.

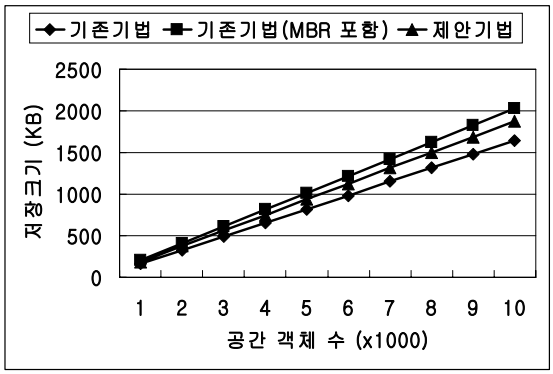
<그림 11>에서 보면 6~8번째 줄에서 현재 포함되는 모든 객체들의 좌표값들을 ptList에 저장한다. 이렇게 저장된 점들을 가지고 10번째 줄에서 가장 왼쪽에 위치한 좌표 하나를 처음 좌표로 선택한다. 이때 가장 왼쪽에 위치한 좌표가 둘 이상일 경우 그 중 가장 아래에 위치한 좌표를 선택한다. 선택한 좌표를 ret에 넣은 후 12~18번째 줄에서 최소볼록 다각형을 위한 연산을 수행한다. 13번째 줄에서는 현재의 좌표와 연결하였을 때 이 선분을 기준으로 나머지 모든 좌표들이 왼쪽에 위치하는 좌표를 구한다. 14번째 줄에서 이를 ret에 저장하며, 15~17번째 줄에서 현재 구한 좌표 NP가 처음 구한 좌표 FP와 일치를 하면 연산을 종료한다.

4. 성능평가

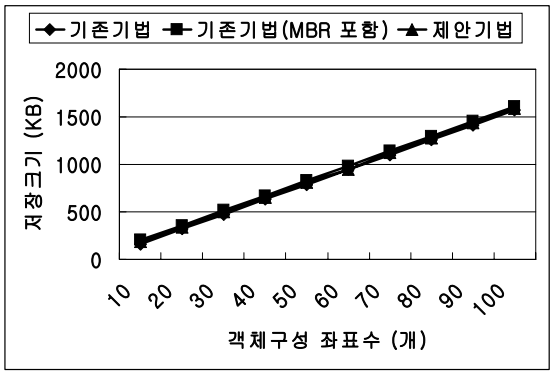
본 장에서는 제안한 공간 집계연산에 대한 성능을 평가한다. 본 성능평가에서는 본 논문에서 제안

```
01: boundary(TS)
02: {
03:     ret : 반환값
04:     ptList : 포인트 리스트
05:
06:     for each record R contained group G in a table
07:         add points to ptList
08:     end for
09:
10:     find the first point FP with minimum x coordinate in ptList
11:     add FP to ret
12:     while
13:         find next point NP where all points lie to the left
                                        of the line connecting with current point
14:         add NP to ret
15:         if NP equals FP then
16:             break
17:         end if
18:     end while
19:
20:     return ret
21: }
```

<그림 11> boundary 연산 알고리즘



a) 공간 객체 수에 따른 저장크기 비교



b) 객체구성 좌표수에 따른 저장크기 비교

<그림 12> 공간 데이터 저장 크기 비교

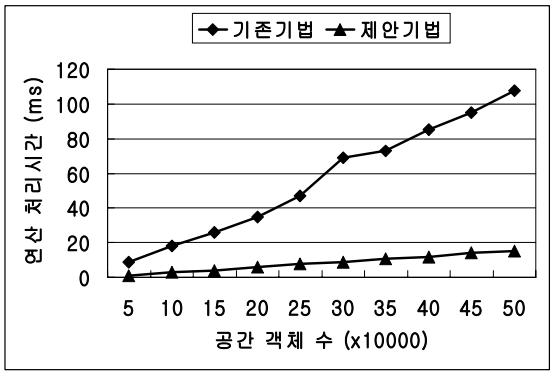
하는 공간 데이터 관리구조를 이용하였을 경우에 대한 비용평가를 한다. 성능평가를 위한 환경은 팬티엄4 3.0GHz CPU와 메모리 1.0GB인 환경에서 시뮬레이션을 제작하여 평가하였다.

먼저 공간 객체의 수가 증가함에 따라 공간 객체를 유지하기 위한 저장비용의 평가를 하였다. 평가를 위해 10개의 좌표를 갖는 공간 객체의 수를 1000개부터 10000개까지 늘려가며 저장 공간의 크기를 비교하였으며, 공간 객체의 수를 1000개로 고정시키고 각 객체가 갖는 좌표의 수를 10부터 100까지 늘려가며 저장 공간의 크기를 비교하였다.

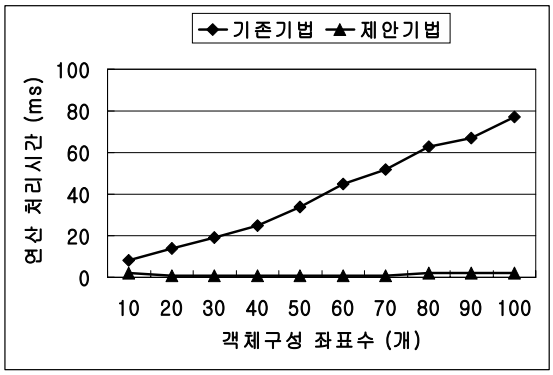
<그림 12>에서 보면 기존기법과 제안기법 모두 공간 객체가 늘어남에 따라 저장크기가 선형적으로 증가함을 볼 수 있다. 먼저 기존기법의 경우 공간 객체의 관리 방법에 따라 두 경우로 나눌 수 있는데 첫째는 공간데이터의 타입 정보만을 헤더로 가지는 경우이며, 둘째는 공간 객체의 필터링을 위해 각 객체마다 MBR 값을 가지고 공간 객체의 타입 정보를 가지는 경우이다. 전자의 경우 제안기법보다 저장공간의 크기를 적게 사용하는 것을 볼 수 있으며, 후자의 경우 MBR 정보를 유지하기 위한 크기가 커 제안기법보다 많은 저장공간을 사용하는 것을 볼 수 있다. <그림 12>의 a)에서 보면 공간 객체 수에 따라 저장공간의 크기가 약간의 차이를 가지는 것을 알 수 있으며, b)의 경우 세 기법 모두 저장크기의 차이가 거의 없음을 볼 수 있다. 따라서 공간 객체를 구성하는 좌표의 수는 기법에 따른 저장크기에 큰 연관성이 없음을 알 수 있다.

다음으로 sum, count, max, min, avg 각각에 대하여 연산 처리비용의 비교를 수행하였다. 기존기법의 경우 MBR은 집계연산에 아무 영향을 미치지 않으므로 MBR을 사용하지 않는 기존기법을 비교대상으로 하였다.

<그림 13>의 a)에서 보면 공간 객체의 수가 많아짐에 따라 sum 집계연산의 처리시간이 증가함을 볼 수 있다. 기존기법의 경우 모든 객체에 대하여 면적을 구해야 하기 때문에 객체의 수에 비례하여 처리시간이 크게 증가하였다. 제안기법의 경우 각 객체의 헤더 정보에 면적 값을 가지고 있어 면적 계산비용이 감소하였고, 따라서 객체 수에 따른 연산 처리시간 증가폭이 작았다. <그림 13>의 b)에서 보면 기존기법은 모든 객체의 면적을 계산하여야 하기 때문에 객체의 수가 고정되더라도 좌표의 수가 증가함에 따라 연산 처리시간이 큰 폭으로 증가함을 보인다. 그러나 제안기법의 경우 모든 객체가 자신의 면적 정보를 헤더에 갖고 있으므로 좌표수와는 무관하게 일정한 연산 처리시간을 보였다. <그림 13>을 통해서 기존기법은 객체의 수와 객체가 갖는 좌표의 수 모두에 영향을 받는 것을 보였으며, 제안기법의 경우 객체의 수에는 적은 영향을 받지만 객체가 갖는 좌표의 수에는 영향을 받지 않는 것을 보였다.



a) 공간 객체 수에 따른 처리시간 비교



b) 객체구성 좌표수에 따른 처리시간 비교

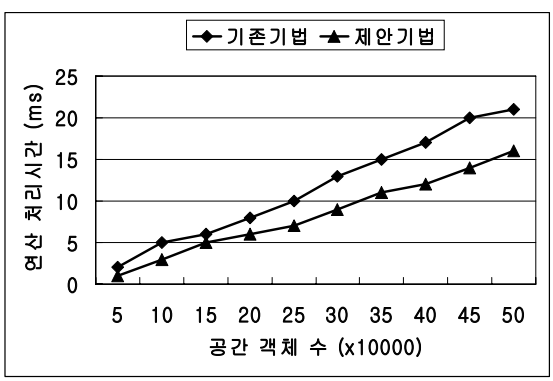
<그림 13> sum 집계연산의 처리시간 비교

<그림 14>의 a)에서 보면 두 기법 모두 공간 객체의 수에 따라 연산 처리시간이 증가하는 것을 볼 수 있다. 이는 기존기법의 경우 실제 공간객체를 유지하는 자료구조에 공간 객체의 수를 포함하고 있어 빠른 검색이 가능하기 때문이며, 제안기법의 경우 헤더 정보에 객체의 수를 포함하고 있기 때문이다. 제안기법의 경우 헤더에 정보를 가지고 있어 기존기법보다 성능이 향상됨을 볼 수 있다. <그림 14>의 b)에서 보면 count 집계연산을 처리하는데 있어 객체가 갖는 좌표수와는 무관함을 볼 수 있다.

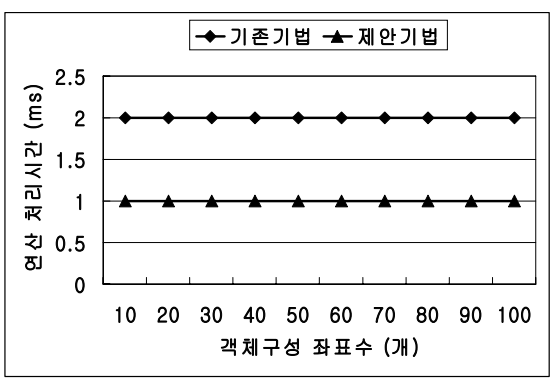
max, min, avg 연산의 경우 모두 면적을 계산한 결과를 이용하여 연산을 처리하기 때문에 그 성능평가에서 <그림 13>의 sum 집계연산의 성능평가 결과와 거의 유사하게 나타났다.

5. 결론 및 향후연구

본 논문에서는 공간 의사결정 지원을 위한 공간 집계연산을 제안하였다. 제안기법에서는 공간 집계연산의 효율성을 위하여 공간 객체의 개수와 면적 등의 헤더 정보를 갖는 공간 데이터의 구조를 제안하였다. 이는 공간 객체의 개수 및 면적을 필요로하는 공간 집계연산의 계산비용을 줄였다. 제안기법에서 공간 집계연산을 크게 숫자화 공간 집계연산



a) 공간 객체 수에 따른 처리시간 비교



b) 객체구성 좌표수에 따른 처리시간 비교

<그림 14> count 집계연산의 처리시간 비교

과 객체화 공간 집계연산으로 나누어 설명하였다. 전자는 대표적인 집계연산자인 sum, count, avg, max, min 등 5개의 숫자화 집계연산자에 대하여 공간 데이터에 대한 연산을 지원할 수 있도록 확장하였으며, 후자는 공간 객체의 분포를 집계할 수 있는 merge와 boundary 연산을 제안하였다. 제안기법은 공간 집계연산을 지원하여 공간 데이터 웨어하우스에서 의사결정 지원을 위한 공간 분석을 가능하게 한다. 성능평가를 통해 확장된 공간 데이터 자료구조를 이용하는 제안기법이 기존기법에 비해 저장 공간 비용이 약간 증가하였지만 공간 집계연산에 있어 기존기법보다 많이 향상된 것을 보였다. 향후 연구로는 홍수 피해 예상 지역에 대해 미리 전기, 가스 등의 위험시설 차단이 가능한 의사결정을 지원하기 위해 레이어로 구성된 공간 정보들의 분석을 지원하는 연구를 할 것이다.

참고문헌

1. Savary, L. and Zeitouni, K., "Spatial Data Warehouse - A Prototype," EGOV 2003, LNCS 2739, 2003, pp. 335-340.
2. Nehme, C. C. and Simões, M., "Spatial Decision Support System for Land Assessment," ACM GIS ' 99, 1999, pp. 85-90.
3. Park, J. M. and Hwang, C. S., "A Design and Practical Use of Spatial Data Warehouse," IEEE IGARSS, 2005, pp.726 -729.
4. Stefanovic, N., Han, J., and Koperski, K., "Object-Based Selective Materialization for Efficient Implementation of Spatial Data Cubes," IEEE, 2000, pp. 938-958.
5. Zhang, Z., Kunqing, X., Xiujun, M., Dan, X., Cuo, C., and Shiwei, T., "Spatial Data Cube: Provides Better Support for Spatial Data Mining," IEEE IGARSS, 2005, pp. 795-798.
6. Wang, Z. C., Chi, Z. X., and Chen, X. Q., "RESEARCH OF SDSS BASED ON SPATIAL DW AND COMGIS," IEEE MLC, 2006, pp. 1528-1531.
7. You, B. S., Lee, D. W., Eo, S. H., Lee, J. D., Bae, H. Y., "Hybrid Index for Spatio-Temporal OLAP Operations," ADVIS 2006, LNCS 4243, 2006, pp. 110 - 118.
8. Papadias, D., Kalnis, P., Zhang, J., and Tao, Y., "Efficient OLAP Operations in Spatial Data Warehouses," Technical Report: HKUST-CS01-01, University of Science & Technology, Hon Kong, 2001.
9. Rao, F., Zhang, L., Yu, X. L., Li, Y., and Chen, Y., "Spatial Hierarchy and OLAP-Favored Search in Spatial Data Warehouse," DOLAP, 2003, pp. 48-55.
10. Indulska, M. and Orlowska, M. E., "On Aggregation Issues in Spatial Data Management," Thirteenth Australasian Database Conference, 2002, pp. 75-84.
11. Zhang, D. and Tsotras, V. J., "Improving Min/Max Aggregation over Spatial Objects," GIS01, 2001, pp.88-93.
12. Matias, R. and Pires, J. M., "Spatial On-Line Analytical Processing (SOLAP): A Tool the to Analyze the Emission of Pollutants in Industrial Installations," IEEE AI, 2005, pp.214-217.
13. Pestana, G., Silva, M. M., and Bcdard, Y., "Spatial OLAP Modeling: An Overview Base on Spatial Objects Changing over Time," IEEE ICCC, 2005, pp. 149-154.
14. Prasher, S. and Zhou, X., "Multi-resolution Amalgamation: Dynamic Spatial Data Cube Generation," ADC, 2004, pp. 103-111.

유병섭 (ByeongSeob You)
2002년 인하대학교 전자·전기·컴퓨터공학부-컴퓨터공학(공학사)
2004년 인하대학교 대학원 전자계산공학과(공학석사)
2004년~현재 인하대학교 대학원 정보공학과(박사과정)
관심분야 : 공간 데이터베이스, 공간 데이터 웨어하우스, 센서 네트워크

김경배 (GyoungBae Kim)
1992년 인하대학교 전자계산공학과(공학사)
1994년 인하대학교 대학원 전자계산공학과(공학석사)
2000년 인하대학교 대학원 전자계산공학과(공학박사)
2000년~2004년 한국전자통신연구원 선임연구원
2004년~현재 서원대학교 컴퓨터교육과 조교수
관심분야 : 이동실시간데이터베이스, 스토리지시스템, GIS, VOD

이순조 (SoonJo Lee)
1985년 인하대학교 전자계산학과(이학사)
1987년 인하대학교 대학원 전자계산학과(이학석사)
1995년 인하대학교 대학원 전자계산공학과(공학박사)
1997년~현재 서원대학교 컴퓨터정보통신공학부 부교수
관심분야 : 데이터베이스 시스템, 정보보안, GIS

배해영 (HaeYoung Bae)
1974년 인하대학교 응용물리학과(공학사)
1978년 연세대학교 전자계산공학과(공학석사)
1990년 숭실대학교 전자계산공학과(공학박사)
1992~1994년 인하대학교 전자계산소 소장.
1982년~현재 인하대학교 컴퓨터공학부 교수.
1999년~2007년 지능형GIS연구센터 센터장.
2000년~현재 중국 중경우전대학교 대학원 명예교수.
2004년~2006년 인하대학교 정보통신대학원 원장.
2006년~현재 인하대학교 대학원 원장.
관심분야는 분산 데이타베이스, 공간 데이타베이스, 지리정보 시스템, 멀티미디어 데이타베이스