공간 객체 관리 시스템에서 래스터 데이터를 위한 지리정보 시스템의 설계 및 구현†

Implementation of a Geographic Information System for the Raster Data in Spatial Object Management Systems

박재진(朴宰振)*, 김 광(金 光)*, 허 신(許 信)***
Jae Jin Park, Kwang Kim, Shin Heu

요 약 지리 정보에 관련된 많은 시스템이 벡터 데이터만을 다루고 있으나 정보의 종류에 따라 벡터 데이터 뿐 아니라 래스터 데이터를 다루는 것이 필요하다. 본 논문에서는 OMEGA라는 공간 객체 관리 시스템에서 래스터 형태의 공간 정보를 처리하는 래스터 처리기를 설계하고 구현한다. 먼저 공간 객체 관리 시스템의 저장 시스템 안에 저장될 래스터 정보의 데이터 구조를 정의하고, 이를 클래스 라이브러리의 형태로 구현하여 응용 애플리케이션의 작성 시 사용할 수 있도록 하였다. 그리고, 래스터 정보를 추출하기 위한 질의문 작성에 포함되는 질의 연산 및 관련 모듈을 구현하고 이를 공간 객체 관리 시스템의 질의 수행기에 삽입하여 래스터 정보에 대한 질의를 처리할 수 있도록 하였다. 마지막으로, 실험에 대한 결과도 제시한다.

ABSTRACT Currently, many geographic information systems deal with only vector data. Therefore, handling raster data for various types of information is greatly needed. In this paper, we design and implement a raster processor which handles spatial information in a spatial object management system called Object Management system for Geospatial Application(OMEGA). We define the data structure of the raster information to be stored in the spatial object management system and implement it to a class library in order for use in making an application program. Furthermore, a query operation and related module is implemented to extract raster information. They are then imbedded in the query executer to process a query of the raster information. Simulation results are given.

키워드 : 지리정보 시스템, 래스터, 데이터베이스

1. 서 론

최근 지리정보 시스템에 대한 요구가 늘어남에 따라 다양한 상용 또는 비상용 지리정보 시스템들이 등장하고 있다. 게다가 인터넷의 빠른 확산에 따라 웹의 장점들을 수용하여 웹 상에서 지리정보 시스템을 서비스 해주는 곳도 증가하고 있다.

그러나 이들 시스템들의 대부분은 벡터 데이터를 기반으로 하고 있다. 래스터 데이터를 사용하는 시스템이라 할 지라도 대부분의 데이터는 벡터 방식의 점, 사각형, 다각형으로 표시하고, 래스터 데이터는 단지 지형의 모양을 사용자에게 보여주기 위해 사용하고 있는 것들이 대부분이다.

본 연구에서는 래스터 데이터를 지형의 모양을 보기 위한 것이 아니라 각 셀에 의미를 두어 다양한 목적에 맞게 사용할 수 있도록 하였다.

지형의 모양이나 위상 구조를 나타나기에 적절한 것이 벡터 데이터라면, 사용자가 각 지역에 대한 속성

† 본 논문은 과학기술부의 핵심 S/W 기술개발사업에서 지원 받았음.
* 한양대학교 전자계산학과 응용연구실
** 한양대학교 전자계산학과 교수
{jjpark, kkim, shinheu}@cse.hanyang.ac.kr

값을 쉽게 알 수 있고 다른 데이터와 비교해 볼 수 있는 장점을 지닌 것이 래스터 데이터라 할 수 있다.

게다가 순수 국내 기술로 개발된 공간 OODBMS는 현재 없는 실정이며, 단순 OODBMS로는 과학기술원에서 개발한 ODYSSEUS, 서울대에서 개발한 SOP 및 O-Base 등이 있다. 또한 Multimedia 데이터를 처리하기 위해 서울대에서 개발한 ALPHA와 RDBMS상에서 공간 데이터를 저장관리 할 수 있게 한 인하대의 KORED/GEO가 개발된 바 있다. 또한, 국내에서 현재 사용되고 있는 일반 상용 지리 정보 시스템에서는 대부분 공간 데이터의 일부로서 래스터 데이터를 취급하고 있으나, 본 공간 객체 관리 시스템과 같이 객체 지향적으로 설계되고 구현된 바는 아직까지 개발된 사례가 없다.

또한, 이러한 연구 결과를 바탕으로 앞으로 광범위하게 사용될 객체 데이터베이스를 기반으로 하여 래스터 지도 데이터에 대한 일반 지리 정보 시스템의 각 기능들을 효과적으로 설계, 구현하는데 이용될 수 있을 것이다.

본 논문의 배경이 되는 공간 객체 관리 시스템은 ODMG 2.0에 따른 데이터 구조와 메쏘드를 정의하고 있으며[3], 객체 데이터베이스 시스템으로 공간 객체를 관리하고 있다. 본 논문은 이상의 공간 객체 관리 시스템의 일부로 동작하는 래스터 처리기를 설계하고 구현하였다.

본 논문의 구성은 다음과 같다. 2장에서는 공간 객체 관리 시스템에 대해 설명하며, 3장과 4장에서는 본 논문에서 설계 및 구현한 래스터 처리기의 구조와 구현된 질의 연산 등을 설명한다. 그리고, 5장에서는 구현 결과를 고찰하며 6장에서 결론을 맺는다.

2. 국내외 기술개발 현황

2.1 확장된 관계형 DBMS에서의 공간 객체 관리

가. SIRO/DBMS : 프로토타입 형태의 지리 데이타베이스 tool-kit. 관계형 DBMS의 커널에 공간 데이타 타입을 추가한 형태임

나. Starbust : 질의어 처리기와 데이타 관리자로 구성된 시스템을 확장하여 GIS시스템의 DBMS로 사용하고자 하는 연구가 있었다.

다. POSTGRES : POSTGRES에 그래픽 처리 모듈을 추가한 형태임

2.2 객체 지향 DBMS에서의 공간 객체 관리

가. GODOT : 'Object-Store'라는 객체 지향 DBMS을 기반으로 벡터 데이터를 위한 지리 객체 관리 시스템

나. GeO2 : 객체 지향 DBMS인 O2를 사용하여 지도 제작 및 환경 관리를 위한 응용 시스템에 사용함

다. Gothic : OODB 개념을 도입하였으나 질의 기능이 없고, 동시성제어를 버전으로 해결하는 등 OODB의 여러 가지 장점을 수용하지 못하고 있다.

2.3 상용 GIS S/W 에서의 공간 객체 관리

가. Arc/Info 등 수많은 소프트웨어에서 공간데이터를 관리하나 이들 대부분 file 이나 RDB등을 이용하였고, 대부분의 공간 데이터와 비 공간 데이터를 분리하여 관리하고 있기 때문에 데이터 일관성에 문제가 발생할 수 있게 된다.

3. 공간 객체 관리 시스템(OMEGA)

본 논문의 배경이 되는 공간 객체 관리 시스템 Object Management system for Geospatial Application(OMEGA)는 DBMS를 확장하여 공간객체를 관리하는 기능을 추가했다. 공간 정보를 표현할 수 있는 데이터 모델과 스키마 정보를 정의하여 이들을 객체 데이터베이스를 이용하여 관리하고 있다.

OMEGA는 ODMG 2.0의 객체 지향 DBMS 표준안에 따라 설계했으며 Object Definition Language (ODL), Object Query Language (OQL), Object Manipulation Language(OML) 및 C++ OML 바인딩을 제공한다. 자체 개발한 객체 관리 언어 인터페이스를 통해 지리 정보 시스템의 애플리케이션을 제작할 수 있도록 지원하며 객체 질의 처리를 위하여 질의 언어를 입력받아 실행 계획을 세우고 필요한 연산을 수행시켜 결과를 도출하는 질의 처리 및 수행기가 포함된다[4]. 현재 위의 공간 객체 관리 시스템에서는 SHORE를 저장 시스템으로 사용하고 있으며, 모든공간, 비공간 객체 정보는 이 곳에 저장된다.

그림 1은 OMEGA의 전체 구조를 나타낸다.

본 논문의 목적은 공간 객체 관리 시스템인 OMEGA 내에서 래스터 형식의 공간 정보를 관리하기 위한 데이터 구조를 정의하고, 질의 처리를 위한 연산자와 관련 모듈을 설계하여 이를 구현하는 것이다.

4. 래스터 처리기의 설계와 구조

4.1. 래스터 데이터 구조

래스터 데이터의 클래스는 공간 객체 관리 시스템에

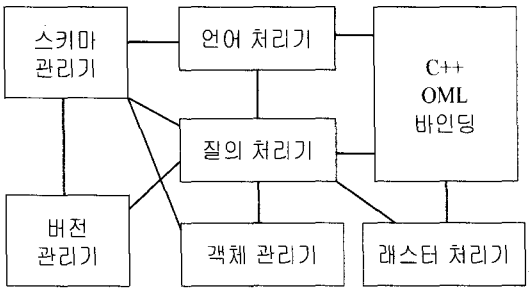

그림 1. OMEGA의 전체 구조

서 모든 클래스들의 수퍼 클래스인 Spatial 클래스의 하위 클래스로 정의된다. 그림 2는 UML notation으로 표현한 래스터 공간 객체들의 클래스 구조이다.

24bit 데이터의 래스터 이미지를 저장하며 다룰 수 있다.

실제 래스터 이미지는 크기가 매우 크기 때문에 S_Raster 클래스에서 실제 래스터 이미지를 지니고 있으면 시스템에서 래스터 데이터를 매번 읽어야 하기 때문에 오버헤드가 너무 크다. 그러므로 실제 래스터 이미지는 S_Bitmap 이라는 클래스에 저장하여 래스터 데이터에 대한 일반적인 정보는 S_Raster 클래스를 참조하고, 실제 래스터 이미지는 S_Bitmap 클래스를 읽어와서 처리하도록 하였다.

지리정보 시스템의 애플리케이션 프로그래머에게 위에서 정의된 래스터 클래스를 비롯하여 공간 객체 관리 시스템에서 제공하는 여러 클래스를 라이브러리로 사용할 수 있도록 지원된다.

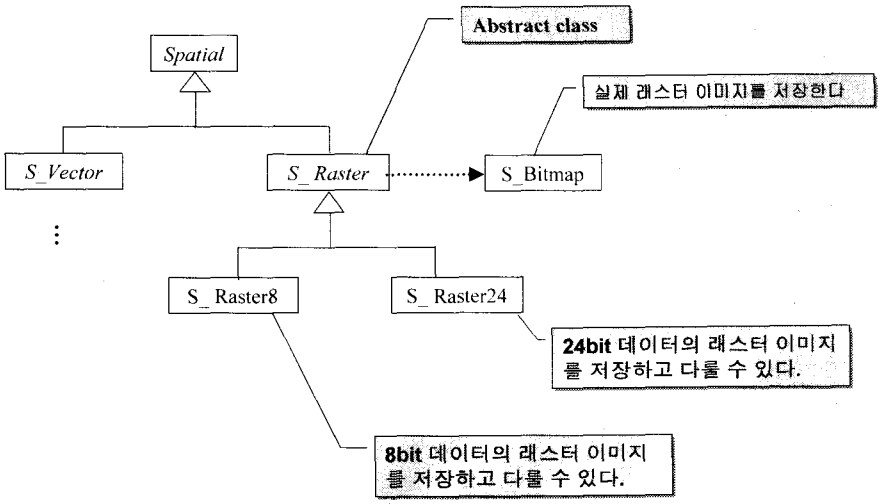

그림 2. 래스터 클래스의 구조도

Spatial 클래스는 모든 공간 클래스의 루트로서 S_Vector 클래스와 S_Raster 클래스의 두 하위 클래스를 가진다. S_Vector 클래스는 지리 정보를 벡터로 다루기 위한 클래스들의 최상위 클래스이고, S_Raster는 모든 래스터 클래스들의 수퍼 클래스이다. S_Raster8, S_Raster24 두 클래스는 모두 Abstract Base Class 인 S_Raster 클래스에서 상속을 받으며 뒤에 붙은 숫자로 저장하고 있는 래스터 데이터의 cell value depth를 구별하도록 하였다. 즉, S_Raster8 클래스는 8bit 데이터의 래스터 이미지를 저장하며 다룰 수 있고, S_Raster24 클래스는

4.2. 래스터 질의 처리 과정

래스터 데이터에 대한 질의 처리는 공간 객체 관리 시스템의 질의 처리기의 일부로 동작되며, 처리되는 과정은 이미 구현되어 있는 공간 객체 관리 시스템의 질의 수행 과정을 따르고 있다[4][5]. 이를 간단히 설명하면 다음과 같다.

질의 처리기에서 수행하고자 하는 질의문은 OQL로 작성되어 공간 객체 스키마 정보를 바탕으로 OQL 파서를 거쳐 질의 최적화 과정으로 들어간다. 질의 최적화 과정에서는 파서의 출력 결과에 대하여 최적의 질의 수행 계획을 수립하여, 이를 실행 트리 구조로 만

들어 질의 수행기로 보낸다. 질의 수행기에서는 입력된 질의 수행 트리의 각 터미널 노드로부터 관련된 질의 연산 루틴을 반복 수행함으로 최종 질의 결과를 도출해낸다[6]. 그림 3은 질의처리기의 수행과정을 나타낸다.

공간 객체 관리 시스템의 객체 관리자를 이용하면 벡터와 래스터 등의 공간 데이터와 비공간 데이터를 모두 포함하는 지리 정보 시스템의 애플리케이션을 작성할 수 있다[7].

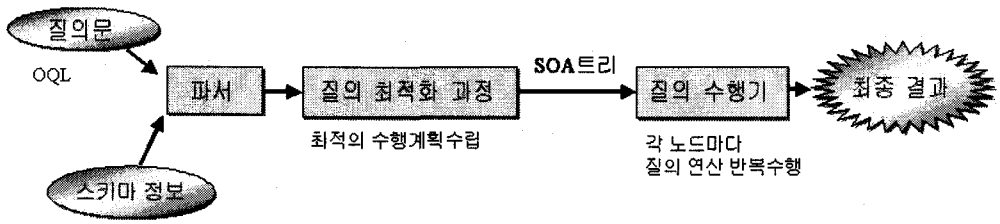

그림 3. 질의 처리기의 수행과정

5. 래스터 처리기의 구현

5.1. 래스터 클래스 라이브러리

3장에서 정의된 래스터 클래스의 구조와 각 메쏘드를 객체 관리 언어인 ODMG OML C++을 사용하여 라이브러리로 구현하였다. 이 라이브러리가 포함되는

```
module Raster{
    interface S_Bitmap{
        attribute long x;
        attribute long y;
        attribute short color_depth;
        attribute List<char> cell;

    };
    interface S_Raster:Spatial{
        attribute S_Bitmap raster;
        attribute long width;
        attribute long height;
        attribute long x_base;
        attribute long y_base;
        attribute long r_width;
        attribute long r_height;
        attribute short color;
    };
    interface S_Raster8:S_Raster{
    };
    interface S_Raster24:S_Raster{
    };
}
```

래스터 클래스의 정의된 내용을 ODMG 2.0에서 정하고 있는 ODL로 표현한 것은 다음과 같다.

ODL 선언 시에는 메쏘드가 표시되지 않지만 C++ 바인딩의 OML 라이브러리를 이용해 래스터 클래스를 작성하게 되면 각각에 클래스의 멤버 함수를 정의하게 된다. 다음과 같은 함수들을 정의했다.

- long getWidth() :
래스터 데이터의 가로 길이를 돌려준다.
- long getHeight():
래스터 데이터의 세로 길이를 돌려준다.
- S_Bitmap getImage() :
래스터 클래스가 저장하는 S_Bitmap 객체를 돌려준다.
- float average() :
래스터 데이터의 각 Pixel의 평균치를 계산한다.
- bool overlap(S_G_Polygon &pg) :
래스터 데이터와 폴리곤 객체의 중첩 여부를 알려준다.
- bool overlap(S_Line &line) :
래스터 데이터와 선 객체의 중첩 여부를 알려준다.
- bool overlap(S_Point &pod_Short):
래스터 데이터와 점 객체의 중첩 여부를 알려준다.
- float proportion(int c) :
래스터 데이터 중 특정 값에 해당하는 셀 들의 점유율을 계산한다.

여기서 average, proportion, overlap, clip 등의 멤버 함수는 질의함수와 중복되는 내용이므로 각 멤버

```
typedef struct {
        d_Long  x;                      // 가로 크기
        d_Long  y;                      // 세로 크기
        d_Long  x_base;                 //  가로 기준점 (왼쪽상단)
        d_Long   y_base;                //  세로 기준점 (왼쪽상단)
        int color_depth;                // color depth (8/24)
        char*  cell;                    // 실제 래스터 데이터를 가리키는 포인터
} Raster_Result;
```

함수에서 내부적으로 질의 함수를 호출하는 구조로 되어있다. 질의함수의 경우 OML 라이브러리를 거치지 않으므로 S_Bitmap 객체를 그대로 파라미터로 넣어 줄 수가 없다. 그러므로 질의 함수에서 인식할 수 있는 타입으로 변형하여 넘겨주어야 한다. 본 논문에서는 Raster_Result라는 struct 타입으로 변환하도록 하였으며, 이는 다음과 같은 구조로 되어 있다.

5.2. 래스터 질의 처리 모듈

공간 객체 시스템에서 래스터 데이터에 대한 질의가 포함된 질의 문을 처리하기 위하여 다음과 같은 질의 연산을 구현하였다. 그리고, 질의 수행 계획 중에 정제 과정(filtering)을 위한 R_Filter()를 구현하였다.

- float z_average()

래스터 데이터의 각 Pixel의 평균치를 계산한다.

- float z_proportion()

래스터 데이터 중 특정 값에 해당하는 픽셀의 점유율을 계산한다.

- Bool zr_overlap(),zl_overlap(),zp_overlap()

래스터가 특정 영역, 라인, 또는 점과 중첩되는지 검사.

6. 프로그램 수행

6.1 스키마 정의

테스트용으로 2 종류의 지도를 준비했다. 하나는 이미지 데이터의 각 셀의 값이 특정 정보를 나타내는 주제도로서 토지이용도로서 총 8장의 지도로 이루어져 있다. 각 cell value에 따른 의미는 표 1과 같다. 다른 하나는 모든 셀의 값이 단일 척도로 처리되는 고도 지도로서 단위는 m이다. 총 4장의 지도이며 하나의 셀이 3 바이트로 구성된다.

각각을 USAGE, ALTITUDE 라는 두개의 스키마로 만들어 DB에 저장했다. 각각의 스키마에는 지도 이름이나 지도의 설명 등 여러 가지 attribute를 지니

표 1. 토지이용도에서의 각 셀의 의미

cell value	의 미
0	forest (산림)
1	grass (초지)
2	bare soil (나대지)
3	rice field (논)
4	tidal flat (조간대, 간석지)
5	urban area (주거지/공장)
6	water (호수/바다)

고 이중 USAGE.DATA는 S_Raster8, ALTITUDE.DATA는 S_Raster24를 나타낸다. 다음은 이 두 지도에 대한 스키마 정의를 ODL로 표현한 것이다.

```
module Raster_test{
   interface ALTITUDE : d_Object
   //(extent ____extent_altitude)
   {
         attribute short          AREACODE;
         attribute char           AREANAME[30];
         attribute char           UNIT[10];
         attribute S_Raster24     DATA;
   };

   interface USAGE : d_Object
   //(extent ____extent_usage)
   {
         attribute short          AREACODE;
         attribute char           AREANAME[30];
         attribute char           UNIT[10];
         attribute short          X_UNIT;
         attribute short          Y_UNIT;
         attribute S_Raster8      DATA;
   };
}
```

여기서 AREACODE는 지도의 일련번호이며 AREANAME은 지도의 이름, UNIT은 지도의 단위를 표시하는 문자열, 그리고 X_UNIT과 Y_UNIT은 각각 지상 해상도 가로 크기와 세로 크기를 의미한다.

6.2 입력을 위한 래스터 파일 포맷

스키마 정보를 입력한 후에 실제 지도 데이터를 DB에 입력해야 한다. 래스터 비트맵은 크게 헤더와 바디로 구성이 되어 있는데, 헤더에는 비트맵 파일에 대한 정보를, 바디에는 실제 비트맵의 내용이 들어있다. 비트맵 파일은 cell value에 따라 8bit 비트맵과 24bit 두 종류로 나뉜다.

6.2.1 비트맵 헤더

비트맵 헤더는 다음과 같이 구성되어 있다. 이는 S_Raster8 과 S_Raster24에 공통으로 사용된다. 처음에는 이 파일이 래스터 비트맵인지를 표시하기 위한 헤더정보 필드로서 입력프로그램에서 이를 보고 파일을 구별하여 래스터 비트맵일 경우에만 DB에 저장하게 된다. 그 다음엔 cell value depth 정보가 들어간다. 현재 S_Raster8 과 S_Raster24 두 종류의 파일을 지원하므로 값은 8이나 24를 갖게 된다. 그 다음으로는 이미지의 가로 세로 크기를 나타내는 필드가 각각 들어가게 되는데 입력 프로그램에서 이를 바탕으로 래스터의 크기 등을 계산하게 된다. 이를 정리한 것이 표 2이다.

표 2. 입력을 위한 래스터 비트맵 헤더

필드설명	크기 (byte)	내 용
헤더정보	24	"NGIS Raster Bitmap File"
Cell value depth	4	8 or 24
Width	4	d_Long 범위의 수
Height	4	d_Long 범위의 수

6.2.2 비트맵 바디

비트맵 바디는 위에서 언급했듯이 비트맵의 종류에 따라 달라지게 된다. 입력프로그램은 헤더 정보를 바탕으로 비트맵의 종류를 판별하고, 각각의 종류에 맞게 DB에 저장하게 된다. (여기서 N은 Width* Height)

● 8 bit 비트맵 : 한 바이트 당 하나의 픽셀이 1:1 대응된다.

Pixel 1	Pixel 2	...	Pixel N

● 24 bit 비트맵 : 세 바이트 당 하나의 픽셀이 1:1 대응된다. 8 bit 비트맵에 비해 더 많은 정보를 저장할 수 있다.

Pixel 1-1	Pixel 1-2	Pixel 1-3	...	Pixel N-1	Pixel N-2	Pixel N-3

6.3 래스터 질의 및 수행 결과

다음과 같은 질의의 예에 대하여 각각의 수행 결과를 그림으로 나타내었다.

● 질의 1 : 토지이용도(USAGE)에서 임의 영역(163,470,830,1263)을 포함하는 지역에서의 주거지(#5)의 점유율을 구하라.

```
select z_proportion(a.DATA,5), a.DATA
from a in USAGE
where a.DATA z_overlap s_rectangle(163,470,830,1263);
```

그림 4는 질의에서 입력한 사각형 영역을 토지 이용도에 표시한 것이고, 그림 5는 질의 입력화면에 대한 그림이다.

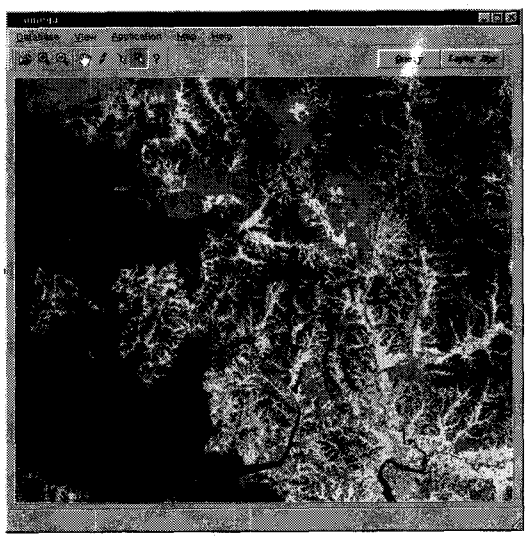

그림 4. 토지 이용도

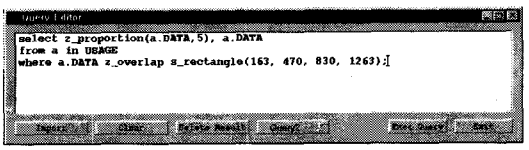

그림 5. 질의 입력 창

이를 수행한 결과는 다음과 같다. 그림 6은 질의에 해당하는 영역이 속한 곳의 지도를 외곽선으로 표시한 것이다. 그림 7은 각 영역의 점유율을 수치로 보여주는데 지도의 왼쪽 상단에서 오른쪽 아래 방향으로 지도 번호가 증가하므로 결과는 이에 따른 순서대로 출력된다.

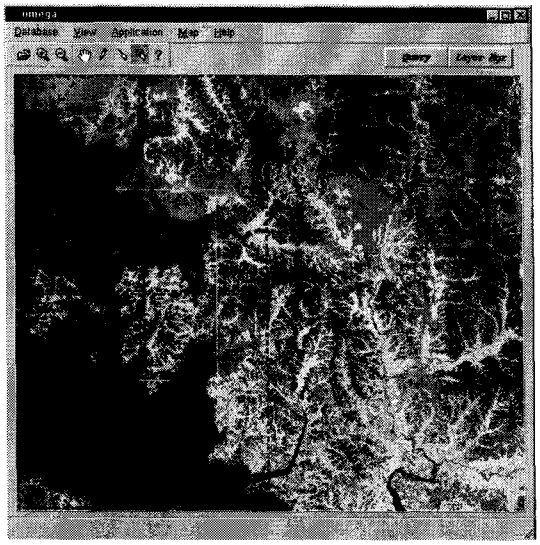

그림 6. 질의 수행 결과 화면

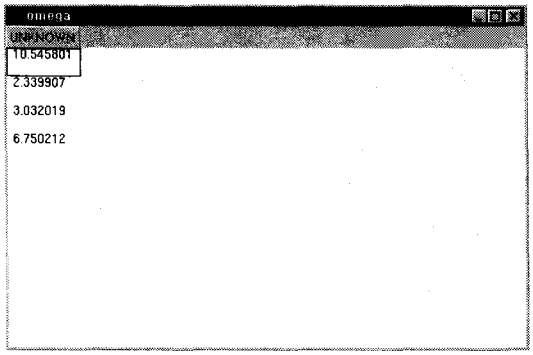

그림 7. 질의 수행 결과 창

• 질의 2 : 고도지도(ALTITUDE)에서 임의 영역(100,100,500,500)을 포함하는 지역의 평균 고도를 구하라

```
select a.DATA, z_average( a.DATA )
from a in ALTITUDE
where a.DATA z_overlap s_rectangle( 100,100,500,500 );
```

그림 8은 질의에서 입력한 사각형 영역을 고도 지도에 표시한 것이고, 그림 9는 질의 입력화면에 대한 그림이다.

그림 8. 고도 지도

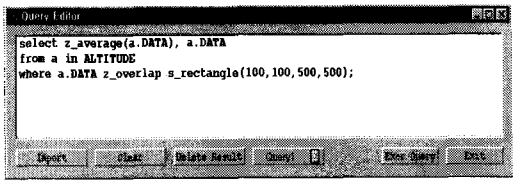

그림 9. 질의 입력 창

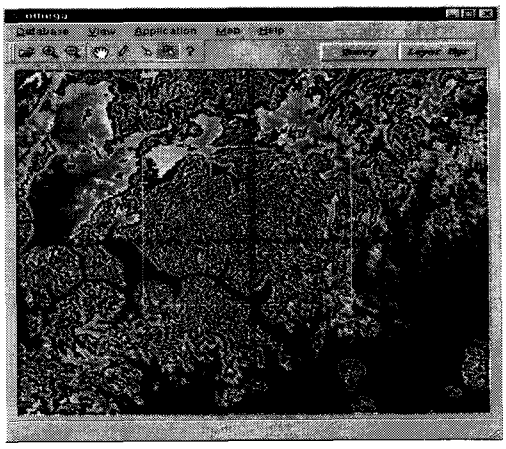

그림 10. 질의 수행 결과 화면

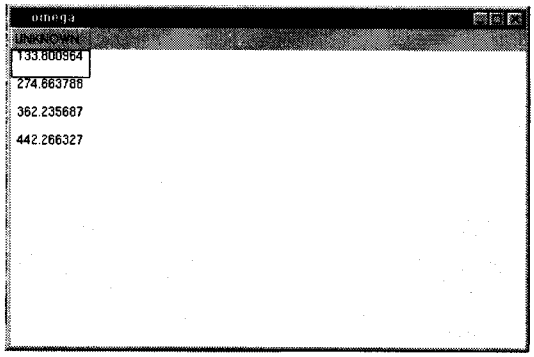

그림 11. 질의 수행 결과 창

이를 수행한 결과는 다음과 같다. 그림 10은 질의에 해당하는 영역이 속한 곳의 지도를 외곽선으로 표시한 것이다. 그림 11은 해당 지역의 평균 고도를 수치로 보여주는데 이 역시 지도의 번호에 따른 순서대로 출력된다.

7. 결 론

본 논문에서는 공간 객체 관리시스템의 일부로 동작하는 래스터 처리기를 설계하고 구현하였다. OMEGA는 ODMG 2.0에 따른 데이터 구조와 메쏘드를 정의하고 있으며, 객체 데이터베이스 시스템으로 공간 객체를 관리하고 있다. 이를 위해 래스터 데이터를 정의하여 클래스 구조로 표현한 후, 클래스 라이브러리의 형태로 구현하였다. 그래고, 래스터 데이터에 대한 질의 처리를 위하여 공간 객체 관리 시스템의 질의 처리를 따르도록 질의 연산을 정의하여 이를 구현하였다.

위에서 구현한 래스터 처리기를 바탕으로 시험 시스템으로 웹기반 래스터 지리정보 시스템을 제작했다. 래스터 데이터 전송을 위한 데이터 포맷과 전송 프로토콜을 설계하고 클라이언트-서버 구조로 구현했다. 클라이언트는 자바 애플릿으로 작성하여 웹 브라우저만 사용하면 누구나 쉽게 시스템을 이용할 수 있으며 결과를 바로 확인할 수 있다.

대부분의 경우 래스터 데이터의 크기가 매우 크기 때문에 래스터 데이터에 대한 압축 과정이 없이 그냥 전송하게 되면 전송 시간이 상당히 지연된다. 이는 프로그램 수행의 속도를 놓고 볼 때 매우 큰 비중을 차지하는 부분이다.

래스터 데이터는 데이터의 특성상 크기가 크기 때문에 지리정보 시스템에서 다량의 지도를 처리하기에는 무리가 따른다. 그러므로 데이터를 압축해야 할 필요가 있는데, 현재는 래스터 데이터에 대한 압축 과정이 없으므로 향후 과제로 래스터 데이터에 대한 압축에 대한 연구가 있어야 하겠다.

참 고 문 헌

[1] Shore Project Homepage, http://www.cs.wisc.edu/shore, University of Wisconsin - Madison, Computer Sciences Department.

[2] D. DeWitt, et al., "Client-Server Paradise," Proc. of 20th VLDB Conference, 1994.

[3] R. Cattell, et al., "The Object Database Standard: ODMG 2.0," 1997.

[4] 이찬근, "지리정보 시스템을 위한 질의 수행기의 설계 및 구현," KAIST 석사학위 논문, 1997.

[5] 박호현, "A Spatial Object Algebra in NGIS DB Tool," KAIST Tech. Report, 1997.

[6] 박호현 외, "OMEGA 질의 처리분야 상세 설계서," 1997

[7] 박재진 외, "공간 객체 관리 시스템을 위한 래스터 처리기의 설계와 구현," 정보과학회 '98 가을 학술발표논문집, 제 25권 pp. 147-149, 1998.

허 신

1969~1973년 서울대학교 전기공학 졸업(공학사)

1977~1979년 Univ. of Southern California. 전산학과 졸업 (공학석사)

1979~1986년 Univ. of South Florida 전산학과 졸업(공학박사)

1977~1979년 Univ. of Southern California 연구조교

1979~1986년 Univ. of South Florida 연구원보

1986~1988년 The Catholic Univ. of America 조교수

1986~1998년 한양대학교 부교수.

1998~현재 한양대학교 교수.

관심분야 : 결함허용 시스템, 실시간 시스템, 분산처리 시스템, 지리정보 시스템

박재진

1993~1998년 한양대학교 전자계산학과 졸업 (학사)
1998~2000년 한양대학교 전자계산학과 졸업 (석사)
2000~현재 (주)옴니텔 연구원
관심분야 : 정보검색, 데이터베이스

김 광

1990~1994년 한양대학교 전자계산학과 공학사
1994~1996년 한양대학교 전자계산학과 공학석사
1996~1999년 한양대학교 전자계산학과 박사과정 수료
1999~현재 삼일데이타시스템(주) 연구원
관심분야 : 운영체제, 지리정보시스템, 실시간 시스템