GOTHIC을 위한 SDTS 데이타 변환 시스템의 개발

Development of a SDTS Data Conversion System for GOTHIC

장염승*, 김준종**, 한기준***

Yan-Sheng Zhang, Jae-Kwan Yun, Ki-Joon Han

요 약 지리 정보 시스템(GIS: Geographic Information System)은 일반적으로 각각의 독립적인 저장 구조를 가지고 많은 양의 지리 정보 데이타를 다루고 있다. 이런 이유로 상호 호환적이지 못한 지리 데이타를 저장하고 있는 지리 정보 시스템들간의 지리 데이타 교환은 일반적으로 어려운 상황이다. 게다가, 지리 데이타를 저장하는 것은 많은 저장 공간을 필요로 하고, 그들을 입력하는데도 많은 비용이 필요하다.

본 논문에서는 현존하는 지리 정보 시스템인 GOTHIC을 위한 SDTS 데이타 변환 시스템을 설계하고 구현하였다. 데이타의 손실없는 변환을 위해서 먼저 SDTS 데이타와 GOTHIC 데이타간의 매핑을 설계하였다. 특히, SDTS 데이타가 ISO8211 형태이므로 엑세스를 위해 FIPS123 라이브러리를 이용하였고, 또한 GOTHIC의 내부적인 포맷은 공개되어 있지 않기 때문에 GOTHIC 데이타를 엑세스하기 위해 GOTHIC 라이브러리를 이용하였다. 본 논문에서 개발한 SDTS 데이타 변환 시스템은 효과적인 지리 정보 데이타의 변환을 위해 중간 화일을 이용하였고, 그래픽 사용자 인터페이스를 구현하기 위해 UIL을 이용하였다.

ABSTRACT A geographic information system (GIS) generally has a great deal of geographic data and has a singular storage structure individually. It is very hard to exchange geographic data between geographic information systems which store their geographic data with incompatible formats. Moreover, since it needs large amount of storage space to store geographic data and expensive cost to input them.

In this paper, we designed and implemented a SDTS (Spatial Data Transfer Standard) Data Conversion System for Gothic which is an existing geographic information system. In order to convert geographic data without loss of information, we first carefully define a mapping between SDTS data and Gothic data. Especially, since SDTS data are in the format of ISO8211, the FIPS123 library is used to access them. Because the internal data format of Gothic is not open to the public, we also use the Gothic library to access Gothic data. The SDTS data conversion system developed in this paper uses an intermediate file to convert geographic data efficiently. In addition, we use UIL (User Interface Language) to implement a graphic user interface (GUI) of our system.

키워드 : 지리 정보 시스템, 데이타 변환 시스템, 공간 데이타베이스, 공간 데이타 교환 표준, 객체-지향 데이타베이스

1. 서 론

지리 정보 시스템(GIS: Geographic Information System)에서 사용되는 지리 데이타는 기존의 데이타와는 달리 지리 정보를 위한 공간, 비공간 데이타들로 구성되어 복잡한 구조를 가지며, 그 크기 면에서도 위성, 지도, 기후, 토양 데이타 등을 포함하기 때문에 일반적으로 대용량이다. 또한, 위성 자료를 관리하는

* 건국대학교 컴퓨터공학과 박사과정

** 핸디소프트(주) 개발1팀

*** 건국대학교 컴퓨터공학과 교수

{yszhang, jjkim, kjhan}@db.konkuk.ac.kr

위성 전문가 시스템, 항해/항법 지도 정보를 관리하는 항해/항법 시스템, 강수량과 토양 등의 데이타를 가지고 홍수를 예방하는 홍수 재해 방지 시스템 등과 같이 많은 지리 정보 시스템이 다양한 범주에서 개발되고 있는데, 이들은 각기 다른 운영체제하에서 다른 응용 소프트웨어와 하드웨어 상에서 구축되고 있다[4,8].

서로 다른 지리 정보 시스템들의 데이타들이 서로 긴밀하게 요구될 필요가 발생하지만 이들 시스템들은 각자 자신들의 데이타 포맷을 가지고 있기 때문에 상호 데이타 교환에 큰 어려움이 있다. 일반적인 경우에 지리 정보 시스템들은 상호 데이타 교환의 어려움으로 인하여 공통 지리 데이타들의 중복 수집과 관리로 많은 기억 공간의 낭비와 경제적인 손실을 안고 있다. 이러한 문제점을 해결하기 위해 지리 정보 시스템들의 공간 데이타들을 다른 시스템과 서로 교환을 위한 공통 데이타 교환 포맷 표준을 채택하고, 이러한 표준 데이타 교환 포맷으로의 변환기 개발 작업 또한 활발히 이루어지고 있다[3,11,15].

국가 지리 정보 시스템(NGIS) 사업의 일환으로 정보통신부(한국전산원)에서는 표준화분과 위원회를 구성하여 한국 표준을 제정하였다. 특히, 국가 기본도 포맷과 공통 데이타 교환 포맷의 표준 제정을 위한 소위원회를 구성하여 표준화 작업을 진행한 결과 공통 데이타 교환 포맷으로 SDTS(Spatial Data Transfer Standard)를 채택하였다[14,16]. 미국내의 연구소, 산업체, 학계 전반의 노력으로 10여 년간의 개발 및 검증과정을 거쳐 1992년 미국 연방표준(FIPS 173)으로 인증을 받은 SDTS는 다양한 공간 데이타의 교환, 공유를 가능하게 해 준다[9]. SDTS는 다양한 지리 정보 시스템 사용자들을 위해 시스템 구축의 중복성을 피하며 효율적으로 지리 정보 데이타를 공유하게 하기 위해 사용될 수 있다. 현재 USGS(U.S. Geological Survey)에서 미국 내에서의 지도 제작 및 판매를 SDTS 포맷으로 사용 중이며, 오스트레일리아와 뉴질랜드 등도 국가의 지리 정보 시스템 데이타 교환 표준으로 SDTS를 채택하고 있다[7,10].

영국 Laser-Scan에서 개발된 지리 정보 시스템인 GOTHIC은 다양한 기능과 응용 프로그램 개발 환경을 제공하지만 현재 우리나라에서 지정한 공간 데이타 교환 표준 포맷인 SDTS의 데이타를 지원하는 데이타 변환 기능을 제공하지 않고 있다[2]. 그러므로, GOTHIC에서 구축된 지리 데이타를 다른 지리 정보 시스템에서 이용하기 위해 SDTS 데이타로 변환할 수 없으며, 마찬가지로 SDTS 형식으로 미리 구축된 지리 데이타를 GOTHIC에서 이용하기가 어려운 현실이다. 본 논문은 여러 지리 정보 시스템에서 데이타 교환에 이용하고 있는 SDTS 데이타를 GOTHIC에 맞는 데이타 포맷으로 변환하여 이용할 수 있도록 하고, 나아가서 GOTHIC 내에서 구축되어 있는 지리 데이타를 SDTS 데이타로 변환함으로써 여러 많은 지리 정보 시스템에서 이용 가능하도록 하기 위해 GOTHIC을 위한 SDTS 데이타 변환 시스템을 개발하였다.

본 논문의 구성은 다음과 같다. 제 2장의 관련연구에서 데이타 변환 시스템의 목적 데이타가 되는 SDTS와 GOTHIC을 분석하고, 제 3장에서는 SDTS와 GOTHIC의 데이타들이 어떻게 매핑되는 지에 관해 언급한다. 제 4장은 GOTHIC을 위한 SDTS 데이타 변환 시스템의 전체적인 구조의 설계, 변환 흐름, 인코딩, 디코딩 과정에 따른 상세적인 설계를 제시한다. 제 5장에서는 SDTS 데이타 변환 시스템 설계에 맞추어 GOTHIC을 위한 SDTS 데이타 변환 시스템의 구현에 대한 사항을 언급하며, SDTS 데이타 변환 시스템의 동작 방법을 보인다. 마지막으로 제 6장에서는 본 논문의 연구에 대해 결론을 맺는다.

2. 관련 연구

데이타 변환 시스템의 개발에 있어서 가장 중요하게 생각되어지는 것은 상호 변환되는 시스템의 데이타 포맷이다. 본 장에서는 GOTHIC을 위한 SDTS 데이타 변환 시스템을 설계 및 구현하기 위해 먼저 알아야 할 SDTS와 GOTHIC의 전체적인 구성과 특징 및 데이타 포맷들에 대해 분석한다.

2.1 SDTS

SDTS는 모든 종류의 공간 데이타(지리 정보, 지도)들을 서로 변환 가능하게 해주는 표준으로 상이한 하드웨어, 소프트웨어, 운영체제 사이에서 데이타 교환을 가능하게 한다. SDTS는 많은 다른 형태, 모델, 구조들로 서로 연관된 속성 데이타들을 FIPS 173을 사용하여 다른 시스템간에 교환될 수 있도록 정의하고 있으며, 이는 크게 3개의 부분인 Part 1, Part 2, Part 3으로 구성된다[9]. 여기에 Part 1, 2, 3에 의해 특정 데이타 유형을 위한 구체적인 변환 형식으로서 Part 4는 벡터 데이타를 위한 TVP(Topological Vector Profile)이다 [6].

2.1.1 Part 1

SDTS Part 1은 공간 데이타 표준의 일반적인 정의 및 규정으로 구성되어 있으며, 5개의 Section으로

분류된다. Section 1은 개론, Section 2는 공간 데이타의 개념적 모델, Section 3은 공간 데이타 질, 그리고 Section 4, 5는 변환과 관련된 레코드와 필드 수준의 내용을 나타낸다.

2.1.2 Part 2

데이타 변환 과정 중에서 공간 형상의 공통적인 정의가 필요하며, 특히 공간 데이타 사회(Spatial Data Community)에서의 데이타 공유의 필요성은 더욱 강조되고 있다. SDTS의 Part 2는 표준화된 개체 리스트의 개발을 위한 표준화된 시도이다. Part 2의 리스트들은 서로 다른 기관에서 저마다 다른 요구에 의해 약 2600개의 지리 형상 정의를 모두 고려할 수 있도록 수년간에 걸친 노력으로 이루어졌다. 현재 리스트들은 200개의 개체 형태 정의, 244개의 속성 정의, 1200개의 포함 용어(Included Term)들로 구성되어 있다.

2.1.3 Part 3

ISO 8211 화일은 Data Description File(DDF)로 불리며 두 가지 형태의 레코드인 DDR(Data Descriptive Record)과 DR(Data Record)을 가지고 있다. DDR은 데이타 구조와 데이타 구조 설명을 가지는 헤더 부분이며, DR은 실질적인 데이타를 가지는 데이타 부분이다. DDF의 레코드는 한 개 이상의 필드로 구성되며, 각각의 필드는 서브 필드들로 이루어지며, 서브 필드들은 데이타의 기본적인 요소(element)들을 가진다.

2.1.4 Part 4

SDTS Part 4는 Part 1, 2, 3에서 정의되고 규정된 SDTS의 개념적, 논리적 부분을 물리적 수준에서 구체적인 변환 지침을 제시하며, 특히 벡터 데이타의 규정 및 변환을 위한 내용이다. TVP는 지리 벡터 데이타를 플래너 그래프 위상과 함께 가지며 변환의 적합성, 인코더 및 디코더의 적합성 등의 규정을 갖고 있고, 또한 공간 객체로 노드, 플래너, 완전무결 체인, GT-다각형, 합성들이 필수적으로 필요하다. SDTS 모듈은 실제 변환시 모듈의 역할, 포함되는 정보 등에 대한 설명, 모듈의 한계와 요구되는 내용, 그리고 구체적으로 ISO 8211 방식으로 변환될 때 Part 1에서 규정된 내용이 어떻게 변환되는지에 대한 내용 등을 포함한다.

2.1.5 SDTS 모듈 분석

SDTS에서는 총 34개의 모듈을 정의하고 있는데, 공간 데이타 변환시 해당 모듈에 각 정보가 포함된다. 이 34개의 모듈은 크게 5가지 카테고리로 범주화될 수 있다. 실제로 데이타를 전환할 때는 34개 모듈 전부가 생성되는 것이 아니라 필요한 모듈만 생성된다[1,9,12].

Global 모듈은 전환의 전반적인 정보를 가지며, Identification, Catalog, Security, Data Dictionary, Transfer Statistics 모듈들로 구성된다. Identification 모듈은 전환의 헤더 기능을 하며, 변환에 사용된 프로화일의 버전, 날짜, 좌표체계명 등의 정보를 가진다. Catalog/Directory 모듈은 변환시 생성되는 모듈의 정보를 가진다. Catalog/Spatial Domain 모듈은 공간 영역, 지도명, 주제 등의 정보를 포함한다. 공간참조 모듈은 공간 데이타의 내부 저장주소방식 정보, 사용하는 좌표체계에 대한 규정, 공간 데이타가 존재하는 공간 영역을 도시한다. 데이타 사전 모듈은 각 속성이 가질 수 있는 값의 유형과 범주에 대한 규약과 속성값 코드의 정의와 속성 레코드 구조를 규정한다. Security 모듈은 각 모듈 레코드의 보안에 대한 정보를 제공하는데, 필수 모듈은 생성된 모듈내 레코드의 수와 공간주소의 수의 리스트 정보를 포함하고 있는 Transfer Statistics 모듈이다.

Data Quality 모듈은 Lineage 모듈, Positional Accuracy 모듈, Attribute Accuracy 모듈, Logical Consistency 모듈, Completeness 모듈로 구성된다. 이들 각각은 변환 데이타의 처리 이력에 대한 정보, 변환 데이타의 위치 정확성에 대한 정보, 변환 데이타의 에러 검증에 대한 정보, 변환 데이타의 그래픽과 위상구조적 무결성 정보, 변환관련 지도작업 규정의 준수 여부 정보를 가지고 있다.

Attribute 모듈은 Attribute Primary 모듈과 Attribute Secondary 모듈로 구분되며, 이 중에서 Attribute Primary 모듈은 필수 모듈로써 공간 객체와 직접 연결되어 공간 객체에 속성을 제공한다. Attribute Secondary 모듈은 Attribute Primary 모듈이나 다른 Attribute Secondary 모듈과 간접적인 연관성을 내부 정보로 가지며, 공간 객체뿐만 아니라 데이타 질 정보나 좌표체계와 관련된 정보를 속성 모듈 형태로 제공하는 모듈이다.

Spatial Object 모듈은 다양한 공간 데이타의 정의를 가지고 있다. 전체적인 구성은 Point, Line, Polygon 공간 객체로 나뉘어서 구성되며, 각각의 공간 데이타는 그 특성상 여러 개의 세분화된 형태의 공간 객체를 정의한다. 그 구분은 위상구조를 포함하였는지의 여부, Attribute 모듈을 참조하고 있는지의

여부 등이며, 그에 따라 각각의 공간 객체 모듈들은 서로 다른 포맷을 가지게 된다.

Graphic Representation 모듈은 공간 객체를 어떻게 효과적으로 표현하느냐를 규약하며 공간 객체와 관계 정보를 가지고 있다. 이 모듈은 칼라, 폰트, Area의 내부가 채워져 있는지의 여부, 텍스트, 라인 심볼의 표현 방법 등에 관한 정보가 저장되는 모듈이다.

2.2 GOTHIC

GOTHIC은 영국 Laser-Scan사에서 개발한 객체 지향 GIS 개발 도구로써 개발환경 내에 자체 객체 지향 데이타베이스를 두고 모든 공간 데이타와 비공간 데이타를 저장한다[5,13].

2.2.1 GOTHIC 데이타베이스

GOTHIC에서는 실세계를 구성하는 개체들을 유일하면서 서로간에 구분될 수 있는 공간 객체들로 생각하고, 이들 중 유사성을 가진 공간 객체들을 그룹화하여 클래스(class)로 정의한다. 즉, 하나의 공간 객체는 정의된 클래스에 대한 하나의 인스턴스(instance)이고, 그러기 위해서 하나의 공간 객체는 반드시 하나의 클래스에 소속되어 있어야 한다. 공간 객체들은 속성이라 불리는 데이타와 메소드를 캡슐화하고, 캡슐화된 항목들을 다른 공간 객체에서는 보이지 않게 하여, 오직 메소드에 의해서만 호출된다.

GOTHIC 데이타베이스에서는 논리적으로 관련있는 데이타들을 규정된 방법에 따라 여러 데이타셋(dataset)으로 구성하여 데이타베이스의 작업단위를 세분화하고 있다. 그리고, 데이타셋들을 계층화하여 데이타셋 단위로 데이타베이스 내의 데이타의 생성, 갱신, 삭제와 같은 일련의 트랜잭션을 수행하도록 하고 있다.

만약 데이타가 갱신될 때는 갱신된 내용들은 해당 데이타셋에서 바로 갱신되어 저장되지 않고 새로운 데이타셋을 자동으로 생성하여 저장되며, 기존의 데이타셋과 새로 생성된 데이타셋은 부모-자식(parent-child) 관계가 설정된다. 이 때 자식 데이타셋은 부모 데이타셋의 모든 데이타베이스 스키마를 상속하고, 새로 변경된 부분만을 저장하게 된다. 이러한 GOTHIC의 버전 메커니즘은 자식 데이타셋의 생성으로 생기는 부하(overload)를 줄여주고, 또한 장기 트랜잭션(long transaction)을 지원해 준다.

GOTHIC에서 각각의 사용자는 자신만의 개인 가지 데이타셋(branch dataset)을 지정하여 일련의 트랜잭션을 수행한다. 다수 사용자가 동시 접근할 수 있는 기능은 각각의 사용자에게 지정된 개인 버젼에만 갱신 엑세스 권한을 실질적으로 부여함으로써 가능하다. 같은 데이타셋으로부터 출발한 버젼은 논리적으로 같은 내용에서 출발하기 때문에 다수 사용자가 갱신 및 삭제를 수행할 수 있다.

기본적인 GOTHIC Toolkit Executable은 ANSI C로 작성된 것이다. 응용 프로그램 개발자들은 이 위에 LULL이라는 GOTHIC 자체 API 언어로 쉽고 빠르게 원하는 기능을 구현할 수 있다. LULL로 표현된 여러 함수들은 GOTHIC 라이브러리에 나타나 있으며, 응용 프로그램 개발자들은 자신이 필요한 기능에 맞는 함수를 이용하여 GOTHIC 데이타베이스를 엑세스하는 등의 여러 작업을 수행할 수 있다.

2.2.2 GOTHIC 클래스

GOTHIC에서 클래스는 공간 객체의 표현 단위이다. 모든 공간 객체들은 그들이 표현되어야 할 데이타 포맷에 따라 다음의 여러 클래스 중의 하나에 속해야 한다. GOTHIC에는 최상위의 4개의 클래스인 object, row, geometry, spatial 클래스가 존재하며, 하위의 모든 클래스는 이들 클래스로부터 상속된다. 이 4개의 최상위 클래스들은 모든 공간 객체들이 포함하는 일반적인 값을 정의한다. 즉, 각 객체에 대한 설명, 장기 트랜잭션이 끝났을 때 통합(merging)을 위한 메소드, 클래스들의 기하(geometry) 정보를 얻고 저장하는 메소드 등을 정의한다. 하위의 클래스들은 점, 선, 영역들에 관한 실제 공간 객체가 인스턴스로 설정되는 클래스들이다.

graphic 클래스는 위상 구조를 고려하지 않는 점, 선, 면의 표현을 위한 타입을 정의한다. 점, 선, 면의 표현에 각각 graphic_point 클래스, graphic_line 클래스, graphic_area 클래스가 상속되며, 이 클래스들에 해당하는 공간 객체들은 메모리와 저장공간을 적게 요구한다. simple 클래스는 위상 구조의 기하 정보를 갖는 공간 객체를 구성하기 위해 사용되는 클래스이다. 이 클래스는 _goth _pre_simple 클래스의 값을 상속받으며, 그 하위에 점, 선, 면에 대한 공간 객체를 표현하는 simple_point 클래스, simple_line 클래스, simple_area 클래스가 상속된다.

_goth_topology_primitive 클래스는 공간 객체를 표현함에 있어 각각의 위상 관계를 체계적으로 표현하는데 이용되는 클래스이다. goth_node 클래스는 위상 구조를 갖는 점을 표현할 수 있는데, 이 위상 규칙에 적용되어 선들이 교차하는 부분 등에 점이 새로 생성되기도 한다. goth_link 클래스는 두개의 노드를 연

결하여 선을 표현하기 위한 클래스이며, goth_face 클래스는 선이나 영역으로 구분되는 면을 표현하기 위한 클래스이다.

_goth_simple_geometry 클래스는 _goth_pre_simple 클래스에서 상속되며, 위상 구조의 공간 객체에 대한 기하정보 값을 가지고 있다. 그 하위에 각각 점, 선, 영역, 링을 표현하기 위한 goth_point 클래스, goth_line 클래스, goth_area 클래스, goth_ring 클래스가 상속된다. 여기서 점은 하나의 노드 객체로 표현되고, 선은 두 개 이상의 노드와 노드의 수보다 하나 적은 링크로 이루어진다. 영역은 링크와 노드로 구성된 닫힌 루프로 표현된 내부를 나타낸다. 그리고, 링크와 노드로 구성된 경계선을 링이라 한다.

3. 데이타 변환 시스템에서의 데이타 매핑

본 장에서는 SDTS-to-GOTHIC 데이타 변환과 GOTHIC-to-SDTS 데이타 변환을 구분하여 Source 데이타의 Point, Line, Polygon 공간 객체가 Target 데이타로 어떻게 매핑되는 가에 관해 언급한다.

3.1 SDTS-to-GOTHIC 변환의 데이타 매핑

SDTS에서는 모든 데이타가 각각 모듈 단위의 화일 형태로 존재하며, 그 데이타들은 화일의 이름으로 어떤 모듈인가 알 수 있도록 되어 있다. 공간 객체에 대한 변환을 하기 위해서 필요한 데이타들은 Global 모듈, 공간 객체 모듈, Attribute 모듈이며, 그 중에서 특정 모듈이 어떤 화일에 속해 있는가의 정보, 위도, 경도, 스케일, 좌표, 해상도 등에 관한 정보를 얻기 위해 Global 모듈을 참조한다.

변환할 SDTS 모듈이 선택되면 먼저 그 공간 객체의 타입 정보를 추출해 내고, 그와 관련된 Attribute 모듈도 파악해야 한다. SDTS와 GOTHIC은 Point, Line, Polygon 공간 객체를 가지고 있는데, 그 공간 객체가 표현하는 객체의 종류에 따라 SDTS 모듈에서 GOTHIC 클래스로의 데이타 변환을 수행한다.

3.1.1 Point 공간 객체

SDTS 모듈에는 여러 종류의 Point를 나타내는 공간 객체 모듈이 존재한다. 그 종류는 크게 3가지로 나눌 수 있는데, 속성이 없는 Point(NO), 속성 정보가 같이 존재하는 Point(NP), Polygon의 Label을 나타내는 Point(NA)가 그 것이다. NA 모듈에 관한 매핑은 Polygon 공간 객체의 매핑에서 언급한다.

NO 모듈의 공간 객체들은 각각 다른 레코드 ID를 가짐으로써 구분된다. NO 모듈은 참조하는 다른 SDTS 모듈이 없기 때문에 그 자신만을 분석하여 추출되는 정보를 중간 화일로 만든다. 분석의 첫 단계로 각각의 레코드별로 그 공간 객체가 가지는 Geometry 값을 추출하여 Attribute 값이 없는 Geometry 값만의 중간 화일을 만든다. GOTHIC 클래스 생성기에서는 Simple_point 클래스의 상속을 받는 공간 객체 클래스를 생성하여 GOTHIC 데이타베이스에 저장한다.

NP 모듈은 NO 모듈과 달리 공간 객체 모듈에 Geometry 값을 가지고 있으면서 동시에 참조 Attribute 모듈 정보를 가지고 있다. NP 모듈을 분석하여 얻은 Geometry 값을 이용하여 NP 모듈의 Geometry 임시 화일을 만들고, 이와 함께 참조해야 하는 Attribute 모듈에 대한 정보를 얻는다. 또한, 여기서 얻어진 Attribute 모듈도 분석하여 Attribute 임시 화일을 만든다. 두 개의 SDTS 모듈을 분석한 정보는 중간 화일로 통합되어 GOTHIC Simple_point 클래스의 상속을 받는 임의의 공간 객체 클래스로 매핑된다.

3.1.2 Line 공간 객체

SDTS의 여러 Line에 관한 모듈 중에서 가장 복잡한 형태의 포맷을 가지고 있는 것은 LE 모듈로써 공간 객체들은 완전한 위상 구조를 포함하고 있다. 즉, 모듈 내의 위상 정보로써 Start Node, End Node, Left Polygon, Right Polygon의 정보를 포함하고 있다. LE 모듈에서 Line 공간 객체 하나의 레코드 구조를 살펴보면 표 1과 같다.

LE 모듈의 레코드 구조에서 참조 Attribute 정보를 나타내는 필드 ATID는 해당 공간 객체가 특정 Attribute 정보가 없을 경우 생략되기도 한다. PIDL과 PIDR은 현재 레코드에서 표시하는 Line의 좌,우 Polygon 정보를 표현하며, SNID와 ENID는 현재

〈표 1〉 LE 모듈의 레코드 구조

필드명	LINE	ATID	PIDL	PIDR	SNID	ENID	SADR
하위 필드명	MODN RCID OBRP	MODN RCID	MODN RCID	MODN RCID	MODN RCID	MODN RCID	X Y .....

Line의 시작 노드와 끝 노드 정보를 나타낸다. SADR은 현재 Line을 구성하는 Geometry 정보를 위의 시작 노드의 정보를 시작으로 끝 노드의 정보까지 X좌표값과 Y좌표값의 쌍으로 표현한다.

LE 모듈의 변환 매핑 과정에서 공간 객체에 따라 참조 Attribute 모듈이 있다면 생성되는 중간 화일에 해당 Attribute 값이 함께 포함되며, 참조 Attribute 모듈이 없다면 Attribute가 없는 형태의 중간 화일이 생성된다. 중간 화일이 완성되면 그 정보는 GOTHIC Simple_line 클래스의 상속을 받는 새로운 공간 객체 클래스로 변환하게 된다. 클래스 생성시에 Attribute Schema는 중간 화일 내에 Attribute 정보의 유무에 따라 결정되게 된다.

3.1.3 Polygon 공간 객체

SDTS 데이타의 여러 모듈 중에서 Polygon에 관련된 모듈은 PC 모듈, LE 모듈, NA 모듈이 있다. 각각은 Point, Line, Polygon 정보를 나타내는 모듈이지만 실제 Polygon에 대한 변환이 수행될 때 꼭 참조될 필요가 있다.

SDTS에서 Polygon 모듈 정보는 PC 모듈에 포함되어 있다. PC 모듈은 공간 객체의 Geometry 정보를 포함하고 있지는 않으며, 관련 Attribute 모듈 이름과 그 모듈 내의 레코드 번호만 포함하고 있다. 공간 객체의 Geometry 정보를 알 수 있는 것은 LE 모듈이며, 그 Polygon에 대한 레이블 포인트의 정보를 가지고 있는 것은 NA 모듈이다. 그러므로, PC 모듈의 변환이 선택되면 먼저 LE 모듈과 NA 모듈을 분석하여서 그 정보를 알고 있어야 한다. 그리고, 두 개의 SDTS 모듈에 대한 분석과 더불어 PC 모듈 내에 존재하는 Attribute 모듈의 정보를 분석해야 한다.

PC 모듈의 변환 매핑에서 Geometry 값에 대한 정보는 LE 모듈을 분석한 임시 화일을 이용하는데, 이는 LE 모듈을 분석시에 각각 Line의 시작 노드와 끝 노드 정보를 이용하여 실제 Polygon을 형성하는 LE 모듈의 공간 객체 레코드를 찾을 수 있기 때문이다. NA 모듈에서 분석된 X좌표값, Y좌표값과 참조 Attribute 모듈에서 분석된 Attribute 값은 모두 Attribute 값의 형태로 중간 화일로 저장되어 전달된다. 이 때 NA 모듈에서 분석된 X좌표값, Y좌표값은 각각 LABELX, LABELY라는 Attribute 이름으로 설정되며, 이것은 현재 변환된 데이타를 역변환시에 다시 NA 모듈로 변환하기 위해 사용된다. 이렇게 생성된 중간 화일의 데이타는 Simple_area 클래스의 상속을 받아 GOTHIC 공간 객체 클래스로 저장된다.

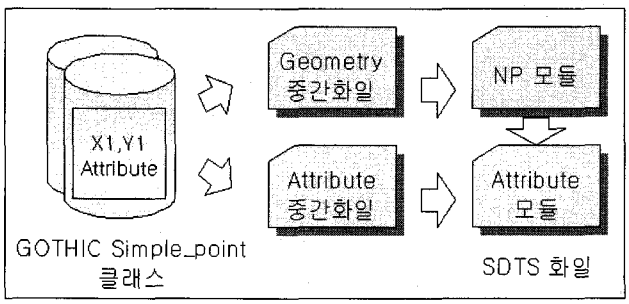
NA 모듈은 Polygon 모듈에 대한 레이블 포인터이다. 본 변환 시스템에서는 NA 모듈에 대한 변환이 요구되면 그 변환은 관련된 PC 모듈의 변환을 동시에 수행한다. NA 모듈이 선택되면 SDTS 모듈 분석기는 NA 모듈에 관련된 PC 모듈의 정보를 추출해 내고, 그 정보에 따라 PC 모듈의 분석을 시작한다.

3.2 GOTHIC-to-SDTS 변환의 데이타 매핑

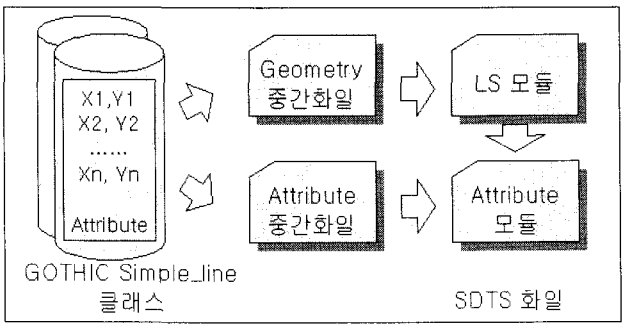
GOTHIC의 공간 객체 데이타는 하나의 클래스에서 상속을 받는 인스턴스들을 일컫는다. GOTHIC-to-SDTS 변환은 현재 설정되어 있는 데이타셋에 있는 모든 공간 객체의 변환을 목적으로 한다. 본 절에서는 GOTHIC에서 위상을 갖는 클래스인 Simple_point, Simple_line, Simple_area 클래스의 상속을 받는 공간 클래스를 SDTS 모듈로의 변환에 대해 언급한다.

3.2.1 Point 공간 객체

GOTHIC에서 Point를 나타내는 공간 객체 클래스들이 그 부모 클래스로써 상속받는 것은 Simple_point 클래스이다. 실제 Point 공간 객체들은 Simple_point 클래스에서 상속받는 임의의 공간 객체 클래스의 인스턴스로써 GOTHIC 데이타베이스에 저장되어 있다. 그림 1은 Simple_point 클래스에서 상속받는 공간 객체들이



〈그림 1〉 Simple_point 클래스의 변환 매핑도



〈그림 2〉 Simple_line 클래스의 변환 매핑도

SDTS의 NP 모듈로 변환되는 과정을 보여준다. 이때 생성되는 Attribute 모듈은 NP 모듈에 해당하는 공간 객체들의 Attribute로써 NP 모듈에서 참조하는 이름과 레코드 ID를 갖게 된다.

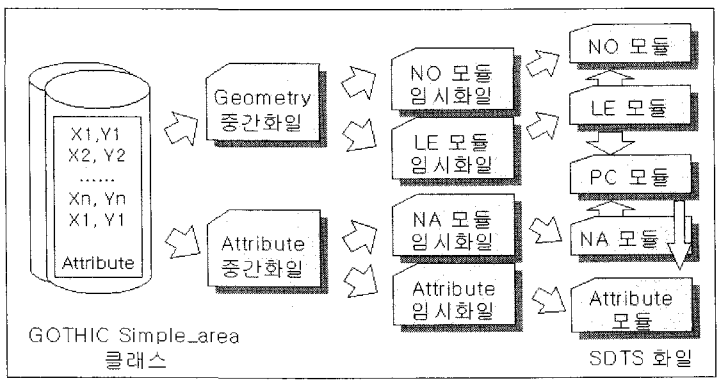
3.2.2 Line 공간 객체

SDTS의 Line을 나타내는 LE 모듈은 그 참조 정보로써 좌우에 위치한 Polygon 정보를 포함하고 있으나, GOTHIC의 Line을 나타내는 Simple_line 클래스는 그에 관한 정보를 가지고 있지 않다. 따라서, 좌우 Polygon 정보를 가지고 있지 않으면서 Line을 나타내는 모듈인 LS 모듈로의 변환을 수행한다. 그림 2는 Simple_line 클래스가 SDTS의 LS 모듈로 변환하는 과정을 보여준다. 전체적인 변환 과정은 Point 공간 객체의 변환과 크게 차이가 없으며, 다른 점은 Geometry 값이 X좌표값, Y좌표값 한 쌍이 아니라 Line을 구성하는 X, Y 좌표값 쌍들이라는 점이다.

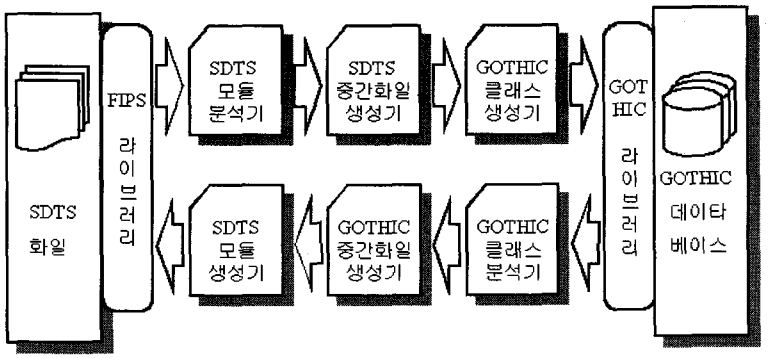
3.2.3 Polygon 공간 객체

SDTS에서 Polygon을 나타내는데 관계되는 모듈은 PC 모듈, LE 모듈, NA 모듈이다. 여기에서 LE 모듈은 Line의 시작 노드와 끝 노드 정보를 가지는 NO 모듈을 참조하고 있어야 하고, 또한 Polygon 객체에 대한 Attribute 정보를 포함하고 있는 Attribute 모듈이 있을 수 있다. 물론, 해당 Polygon 모듈이 Attribute를 가지고 있지 않다면, 해당 PC 모듈에 대한 Attribute 모듈은 없는 경우도 있다.

그림 3은 GOTHIC Simple_area 클래스가 SDTS의 여러 모듈로 변환되는 과정을 보여 준다. 여기에서 Geometry 중간 화일은 GOTHIC 공간 객체의 Geometry 데이타를 전달하는 중간 화일이며, Attribute 중간 화일은 각각의 공간 객체의 Attribute 값을 전달하는 중간 화일이다. Geometry 중간 화일은 각 공간 객체마다 다른 식별자를 가짐으로써 구분되는데, 공간 객체 Geometry 정보로써 LE 모듈에 포함될 X좌표값, Y좌표값의 쌍을 만들게 되고, 공간 객체의 시작 노드와 끝 노드의 X좌표값, Y좌표값쌍의 정보로써 NO 모듈 임시 화일을 만들게 된다. 이때, LE 모듈로



〈그림 3〉 Simple_area 클래스의 변환 매핑도



〈그림 4〉 SDTS 데이타 변환 시스템의 전체적인 구성도

써 참조되는 PC 모듈도 동시 에 생성되게 된다.

Attribute 중간 화일은 NA 모듈과 PC 모듈의 Attribute 모듈을 생성하는데 사용된다. 여기서 NA 모듈은 XLABEL, YLABEL이란 Attribute 이름이 존재할 경우만 생성하게 되는데, 이는 SDTS-to-GOTHIC 데이타 변환기를 이용하여 변환된 데이타일 경우 적용되며, 데이타 손실을 최소한으로 줄이기 위한 방안으로 제시한 방법이다. XLABEL, YLABEL이란 이름을 가진 Attribute 값을 제외한 나머지 Attribute는 실제 PC 모듈의 Attribute 모듈로 생성되게 된다.

4. 데이타 변환 시스템의 설계

본 장에서는 먼저 SDTS와 GOTHIC간에 어떻게 데이타가 변환되는지에 중점을 둔 전체적인 데이타 변환 시스템의 설계에 관해 언급하고, SDTS에서 GOTHIC으로의 데이타 변환에 대한 설계와 GOTHIC에서 SDTS로의 데이타 변환에 대한 설계에 대해 설명한다.

4.1 데이타 변환 시스템의 전체적인 설계

데이타 변환 시스템의 전체적인 설계에서 가장 먼저 고려하여야 할 것은 변환하고자 하는 데이타의 흐름이다. SDTS는 ISO8211 형식의 화일이어서 그 데이타를 분석하기 위해 USGS에서 제공하는 FIPS123 라이브러리를 이용하였으며, GOTHIC은 자체적으로 객체지향 데이타베이스 시스템을 제공하여 데이타를 저장하기 때문에 데이타베이스를 엑세스하기 위해 GOTHIC 개발사인 Laser Scan에서 제공하는 GOTHIC 라이브러리를 이용하였다.

그러나, 문제가 되었던 사항은 FIPS123 라이브러리와 GOTHIC 라이브러리가 각각 다른 언어를 사용한다는 점이다. FIPS123 라이브러리는 C 언어로 구현되어 있으며, GOTHIC 라이브러리는 Laser Scan에서 자체 개발한 LULL 언어로 구현되어 있다. 이와 같은 이유로 SDTS 데이타를 분석하고 저장하는 부분의 모듈은 C언어로 프로그래밍되며, GOTHIC 클래스를 분석하고 저장하는 부분의 모듈은 LULL 언어로 프로그램된다. 또한, 데이타를 변환하기 위해서는 중간 매개체가 필요하였는데, 이를 위해 본 시스템에서는 중간 화일을 이용하였다. 본 논문에서 설계한 GOTHIC을 위한 SDTS 데이타 변환 시스템의 전체적인 구성은 그림 4와 같다.

그림 4의 오른쪽으로 향하는 화살표는 FIPS123 라이브러리의 함수를 이용하여 SDTS 모듈들을 분석하고 분석된 내용으로 SDTS 중간 화일을 생성하고, GOTHIC 라이브러리의 함수들을 이용하여 GOTHIC 데이타베이스에 저장을 나타내다. 또한, 왼쪽으로 향하는 화살표는 GOTHIC 라이브러리를 이용해 GOTHIC의 Geometry 데이타 및 속성 데이타를 얻고, 데이타를 GOTHIC 중간 화일 형태로 변환한 이후에 FIPS123 라이브러리의 함수들을 이용하여 SDTS 화일 형식으로 변환되는 과정이다.

4.2 SDTS-to-GOTHIC 데이타 변환기의 설계

SDTS-to-GOTHIC 데이타 변환기는 그림 4에서 보였던 SDTS 데이타 변환 시스템의 전체적인 구성도의 위쪽 모듈들인 SDTS 모듈 분석기, SDTS 중간 화일 생성기, GOTHIC 클래스 생성기를 총칭한 것이다.

SDTS 모듈 분석기는 Source 데이타에 해당하는 SDTS 모듈을 FIPS123 라이브러리를 이용하여 일반적인 텍스트 형식의 데이타로 분석하는 모듈이며, SDTS 중간 화일 생성기는 SDTS 모듈 분석기에서 분석된 데이타를 GOTHIC 클래스 생성기로 전달하기

위한 중간 화일 포맷을 설정하고, 데이타를 그 포맷에 맞게 저장하는 모듈이다. 마지막으로, LULL 언어로 프로그래밍되는 GOTHIC 클래스 생성기는 중간 화일에 저장된 데이타를 읽어들여 GOTHIC 클래스를 설정하고, 필요한 클래스를 생성하는 작업을 수행한다.

4.2.1 SDTS 모듈 분석기

SDTS 모듈 분석기는 사용자가 변환하고자 하는 SDTS 모듈을 선택할 때 그 선택된 SDTS 모듈과 관계있는 참조 모듈을 설정하고, 참조 모듈이 결정되면 선택된 공간 데이타를 분석하며, 또한 참조 설정된 속성 데이타를 분석하는 작업을 수행한다.

SDTS 모듈 참조 설정은 Global 모듈, Data Quality 모듈을 분석하여 현재 변환하고자 하는 지리 데이타의 포괄적인 정보를 분석하고, 변환하고자 하는 공간 객체 모듈이 어떤 Attribute 모듈을 참조하고 있는가에 대한 정보를 찾아낸다. 이러한 정보는 참조 속성 데이타 분석에 영향을 주게 된다. 이렇게 분석된 정보는 공간 데이타를 내용으로 하는 공간 데이타 중간 화일을 생성하는데 사용된다.

공간 데이타 분석에서는 사용자가 변환을 요구하는 공간 데이타를 분석하는 일을 수행한다. 분석되는 SDTS 데이타는 Point를 저장하고 있는 NP, NE, NO, NA 모듈, Line 데이타를 저장하고 있는 LE 모듈, Polygon 데이타를 저장하고 있는 PC 모듈이다. 이들 SDTS 공간 객체 모듈은 그 특성에 따라 자신 이외에 다른 공간 데이타 모듈을 같이 분석해야 하는 경우도 있다.

Point 모듈이 분석될 때는 선택된 모듈 종류에 따라 NP, NE 모듈에 대한 독자적인 분석만으로 데이타 추출이 가능하다. 그러나, Line 모듈과 Polygon 모듈이 분석될 때는 다른 공간 객체 모듈이 같이 분석된다. LE 모듈을 분석할 때는 Line의 시작 노드와 끝 노드를 나타내는 NO 모듈, Line의 좌우에 위치한 Polygon 정보를 나타내는 PC 모듈이 함께 분석된다. PC 모듈은 자체적으로 지리 데이타를 가지고 있지 않기 때문에 LE 모듈을 먼저 분석하여 LE 모듈 내에 존재하는 좌우 Polygon 정보를 이용하여 Polygon을 생성한다. 먼저 시작 노드와 끝 노드 정보의 상호 일치 여부를 이용하여 Polygon을 형성하는 LE 모듈의 조합을 밝혀내고, 그 LE 모듈내에서 해당 레코드에 포함되어 있는 지리 데이타를 이용하여 하나의 닫혀있는 Polygon 정보를 만든다. 이렇게 분석되어진 각각의 Point, Line, Polygon 데이타는 참조 속성 데이타 분석에서 얻어진 데이타와 함께 중간 화일을 만드는 데 사용된다.

4.2.2 SDTS 중간 화일 생성기

SDTS 중간 화일 생성기에서의 주된 역할은 SDTS 모듈 분석기에서 분석되어진 데이타를 중간 화일로 작성하여 GOTHIC 클래스 생성기에 전달하는 것이다. 중간 화일은 SDTS 전체적인 정보를 전달하는 공간 정보 중간 화일과 실제 공간 객체 데이타 정보를 전달하는 데이타 중간 화일로 구성된다. 또한, 사용자 인터페이스에서 입력되는 여러 요구사항들을 체크하여 GOTHIC 클래스 생성기에서 반영하도록 한다.

본 시스템을 이용하는 사용자는 자신이 변환하고자 하는 SDTS 공간 데이타 모듈을 선택하여야 한다. SDTS 모듈 선택에서는 이를 위한 인터페이스를 제공하고, 선택된 모듈을 SDTS 모듈 생성기에 전달하는 역할을 한다. 이 단계에서 선택된 SDTS 공간 데이타 모듈 정보는 SDTS 모듈 분석기에 보내져서 여러 SDTS 모듈 중에서 실제로 분석되어야 하는 공간 데이타 모듈, 속성 데이타 모듈을 결정하고 분석을 시작하게 된다.

SDTS-to-GOTHIC 데이타 변환기를 이용하는 사용자는 현재 변환하고자 하는 공간 객체에 자신이 원하는 클래스 이름을 부여할 수 있다. 그 클래스 이름이 결정되면 모든 공간 객체들은 그 클래스에서 상속을 받는 하나의 인스턴스로써 GOTHIC 데이타베이스에 저장되게 된다. GOTHIC의 모든 클래스들은 하나의 데이타셋에 속해 있어야 하는데, 이런 이유로 현재 설정된 데이타셋의 이름을 찾는 과정이 필요하다. 그러므로, 데이타셋과 클래스 설정에서는 사용자가 요구하는 이름의 클래스를 받아 들여, GOTHIC 클래스 생성기에서 알맞는 클래스로부터 상속되도록 전달하는 작업이 수행된다.

4.2.3 GOTHIC 클래스 생성기

GOTHIC 클래스 생성기는 SDTS 중간 화일 생성기에서 전달된 여러 정보 및 중간 화일의 데이타를 GOTHIC 데이타베이스에 저장하는 역할을 수행한다.

사용자는 현재의 공간 객체에 대한 클래스 이름을 스스로 결정할 수 있고, 그 설정된 클래스는 알맞는 기존의 GOTHIC 클래스에서 상속받아야 한다. GOTHIC 생성 클래스 결정에서는 현재 변환하고자 하는 공간 객체가 Point 모듈이라면 Simple_point 클래스, Line 모듈이라면 Simple_line 클래스, Polygon 모듈이라면 Simple_area 클래스의 상속을 받는 공간 객체 클래스를 생성하게 된다.

공간 데이타 입력은 중간 화일을 읽어들여 공간 데

이타에 해당하는 X좌표값, Y좌표값을 GOTHIC 클래스의 인스턴스에 해당하는 공간 객체에 저장하는 작업을 수행한다. 이때 현재 변환 중인 공간 객체가 Point라면 X좌표값, Y좌표값은 한 쌍이 될 것이고, Line이라면 여러 개의 쌍을 이루며, Polygon이라면 여러 개의 X좌표값, Y좌표값 쌍을 갖는 것은 Line과 동일하지만 처음과 끝이 같은 닫혀 있는 좌표값의 쌍을 갖는다. 또한, 속성 데이타도 공간 데이타 입력과 비슷하게 GOTHIC 데이타베이스에 저장된다.

4.3 GOTHIC-to-SDTS 데이타 변환기의 설계

GOTHIC-to-SDTS 데이타 변환기는 그림 4에서 보였던 SDTS 데이타 변환 시스템의 전체적인 구성도의 아래쪽 모듈들인 GOTHIC 클래스 분석기, GOTHIC 중간 화일 생성기, SDTS 모듈 생성기를 총칭한 것이다.

GOTHIC 클래스 분석기는 GOTHIC 라이브러리를 이용하여 GOTHIC 클래스의 데이타를 추출하는 역할을 하며, GOTHIC 중간 화일 생성기는 SDTS 모듈 생성기로 데이타를 전달하기 위한 중간 화일을 생성하고, 그 외에 SDTS 모듈이 생성될 디렉토리 관리 및 전체적인 모듈 생성을 관리하는 역할을 한다. SDTS 모듈 생성기는 중간 화일 형태로 전달되어지는 데이타를 그에 맞는 SDTS 모듈로 생성하는 작업을 수행한다.

4.3.1 GOTHIC 클래스 분석기

GOTHIC 클래스 분석기는 GOTHIC 라이브러리의 여러 함수를 이용하여 현재 데이타셋에 설정되어 있는 여러 공간 객체 클래스를 찾아내고, 찾아낸 클래스에 인스턴스로 저장되어 있는 실제 GOTHIC의 Geometry 데이타와 속성 데이타를 추출해 내는 작업을 수행한다.

GOTHIC 클래스 분석기에서 수행해야 하는 최초의 작업은 현재 데이타셋에 어떤 클래스들이 존재하는가를 알아내는 일이라 할 수 있다. GOTHIC 클래스 파악에서는 데이타 변환에서 실질적으로 추출되어야 하는 공간 객체 클래스의 클래스 ID 리스트를 찾아내어 공간 데이타 분석, 속성 데이타 분석 단계로 전달하는 작업을 수행한다.

공간 데이타 분석에서는 GOTHIC 클래스 파악에서 전달되는 공간 객체 클래스 ID 리스트의 정보를 이용하여 실질적인 Geometry 데이타를 추출하는 작업을 한다. 이 경우 여러 개의 공간 객체 클래스가 존재하면 ID가 빠른 순서대로 분석을 시작하며, 이 때 분석되는 Geometry 데이타는 중간 화일 생성기에서 SDTS 모듈을 생성하기 용이한 포맷으로 저장되어 전달된다.

GOTHIC 데이타베이스에 저장되어 있는 공간 객체들은 그 Schema에 속성 데이타의 이름과 그 데이타형이 이미 설정되어 있다. 따라서, 하나의 클래스에 인스턴스로 속하는 여러 공간 객체들은 모두 동일한 속성 데이타 이름과 데이타형을 가지고 있다. 속성 데이타 분석에서는 클래스의 Schema에 설정되어 있는 속성 데이타 이름과 그 데이타형을 분석하고, 실제 공간 객체 인스턴스에 저장되어 있는 속성 데이타값을 추출하는 작업을 한다.

4.3.2 GOTHIC 중간 화일 생성기

GOTHIC 중간 화일 생성기는 GOTHIC 클래스 분석기에서 분석된 공간 객체들의 Geometry 데이타와 Attribute 데이타를 중간 화일로 변환하고, 여러 SDTS 모듈의 생성에 관련된 디렉토리 구조를 설정하는 역할을 수행한다.

GOTHIC-to-SDTS 데이타 변환기를 사용하기 위해서 사용자는 먼저 특정 데이타셋을 선택한다. 변환 데이타셋 정보분석에서는 이미 선택되어진 현재의 데이타셋에 대한 정보를 분석한다. 공간 객체 클래스가 파악되면 그 공간 객체 클래스가 어떤 클래스로부터 상속받아서 정의되었는지 분석하는 단계가 필요하다. 왜냐하면 그 클래스가 SDTS의 어느 모듈로 변환 가능한가에 대한 정보가 필요하기 때문이다.

GOTHIC-to-SDTS 데이타 변환기에서 생성하는 중간 화일은 두가지 종류로 나눌 수 있는데, SDTS 모듈중에서 Global 모듈, Data Quality 모듈을 생성하는데 필요한 정보를 가지는 참조모듈 중간 화일과 Spatial Object 모듈, Attribute 모듈을 생성하는데 필요한 데이타 중간 화일이 그것이다. 또한, 데이타 중간 화일은 SDTS 모듈을 생성에 용이하도록 두 종류의 중간 화일로 나누어서 생성된다. 그 하나는 Spatial Object 모듈을 생성하기 위해서 GOTHIC 데이타베이스의 정보 중에서 Geometry 데이타를 저장하는 Geometry 중간 화일이며, 다른 하나는 Attribute 모듈을 생성하기 위한 정보를 전달하기 위해 속성 데이타값을 저장하는 Attribute 중간 화일이다.

SDTS 모듈은 공간 객체 모듈, 속성 데이타 모듈과 더불어서 9개의 Global 모듈과 5개의 Data Quality 모듈이 함께 생성된다. 여러 개의 SDTS 모듈이 생성되기 때문에 본 시스템은 사용자로부터 특정 디렉토리 이름을 입력받고, 입력받은 디렉토리를 별도로 생성하여 새롭게 생성되는 모든 SDTS 모듈을 저장한다. 디

렉토리 관리에서는 이에 대한 모든 작업을 수행하고, 또한 중간 화일을 생성하는 중간 과정에서 만들어지는 임시 화일들을 생성 및 삭제하는 작업도 수행한다.

4.3.3 SDTS 모듈 생성기

SDTS 모듈 생성기는 GOTHIC 중간 화일 생성기를 통해 생성된 참조모듈 중간 화일과 데이타 중간 화일의 정보를 받아서 9개의 Global 모듈, 5개의 Data Quality 모듈, 공간 객체 모듈, 속성 데이타 모듈을 생성하는 역할을 수행한다.

본 시스템에서 생성되는 SDTS 공간 객체 모듈은 현재의 데이타셋에 존재하는 GOTHIC 클래스의 종류에 따라 결정된다. SDTS 생성 모듈 결정에서 수행하는 작업은 GOTHIC 중간 화일 생성기에서 생성된 중간 화일들을 분석하여 특정 중간 화일의 존재 여부를 파악하고, 그 중간 화일이 존재할 경우 그에 맞는 모듈의 생성을 시작하는 것이다. 이 때 생성되는 모듈은 SDTS Global 모듈, Data Quality 모듈이며, 각각의 모듈에 저장될 데이타는 참조모듈 중간 화일을 통해 전달된 정보를 이용한다.

SDTS 공간 객체 모듈 생성 단계에서는 두 종류의 데이타 중간 화일 중에서 Geometry 중간 화일의 정보를 읽어들여 그 중간 화일이 나타내는 해당 공간 객체 모듈을 생성한다. SDTS 공간 객체 모듈 중에는 독자적으로 하나의 공간 객체 클래스를 나타내는 모듈도 존재하지만, 상호 참조를 통해 하나의 공간 객체를 나타내는 모듈도 존재한다. 상호 참조를 통해 하나의 공간 객체를 나타내는 모듈의 대표적인 예로 LE 모듈을 들 수 있는데, GOTHIC의 위상이 있는 Simple_line 클래스를 변환시에는 LE와 함께 LE 모듈에서 참조하고 있는 PC, NO 모듈이 함께 생성되어야 한다.

GOTHIC의 공간 객체 클래스는 그 공간 객체의 Geometry 정보뿐만 아니라 Attribute Schema 및 실제 속성값을 포함하고 있다. SDTS 속성 모듈 생성 단계에서는 이러한 Attribute Schema 및 속성값을 전달하여 주는 Attribute 중간 화일에 포함된 정보를 이용하여 공간 객체에 알맞는 속성 모듈을 생성하는 작업을 수행한다. Attribute 중간 화일은 관련된 Geometry 중간 화일을 표현하기 위해 각각 다른 이름으로 생성되는데, 예를 들어 Simple_area 모듈에 해당하는 공간 객체의 Attribute 중간 화일은 appc.tmp란 이름으로 Attribute 중간 화일을 생성하여 해당하는 속성값을 저장하고 있다.

5. 데이타 변환 시스템의 구현

본 장에서는 먼저 데이타 변환 시스템이 구현된 시스템 환경을 살펴보고, 다음으로 SDTS-to-GOTHIC 데이타 변환기의 구현과 GOTHIC-to-SDTS 데이타 변환기의 구현에 대해 언급하고, 마지막으로 SDTS 데이타 변환 시스템의 사용자 인터페이스 구현 상황을 살펴본다.

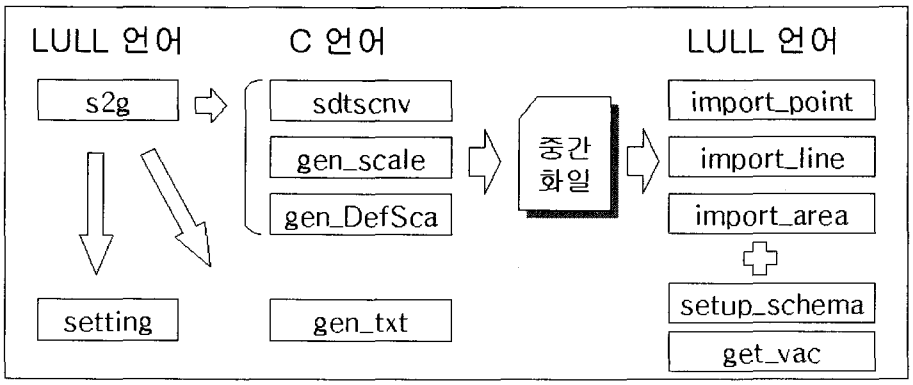
5.1 구현 환경

데이타 변환에서 하나의 목적 데이타가 되는 GOTHIC 데이타베이스가 UNIX 머신에서 운용되기 때문에 GOTHIC을 위한 SDTS 데이타 변환 시스템이 개발된 환경으로 하드웨어는 SunSparc Server 1000이 사용되고, 운영체제는 Solaris가 사용되었다. 본 시스템이 구현되는 GOTHIC ADE는 GOTHIC 3.0 Developer 에서 제공되며, GOTHIC 데이타베이스를 엑세스하기 위해서는 LULL 언어를 사용하였다. 그리고, SDTS 모듈의 정보를 추출하기 위해서 FIPS123 라이브러리를 사용하였고, 그런 이유로 C 언어가 사용되었다. C 언어 관련 프로그램은 GNU gcc를 이용하여 컴파일되었고, 또한 그래픽 사용자 인터페이스를 제공하기 위해 UIL(User Interface Language)이 사용되었다.

5.2 SDTS-to-GOTHIC 데이타 변환기의 구현

SDTS-to-GOTHIC 데이타 변환기는 SDTS 모듈을 분석하여 그에 알맞는 GOTHIC 클래스로 변환하는 역할을 수행하는데, s2g라는 LULL 프로그램에서 전체적인 프로그램의 흐름을 제어한다. s2g는 SDTS-to-GOTHIC 데이타 변환기의 메인 인터페이스를 관리하는 역할을 하며, 메인 인터페이스는 각각의 실행 버튼의 작동에 따라 다른 LULL 프로그램을 호출하고 서브 인터페이스로 이동을 수행한다. 그림 5는 각각의 실행 프로그램들의 실제 수행 과정을 보여준다.

그림 5에서 보는 바와 같이 전체적인 SDTS-to-GOTHIC 데이타 변환기의 데이타 변환 과정을 조정하는 것은 LULL로 구현된 s2g이다. 그러나, SDTS 모듈을 분석하는 프로그램은 모두 C 언어로 구현되어 있다. C 프로그램인 "sdtscnv", "gen_scale", "gen_DefSca"를 통해 SDTS 모듈이 분석되며, 분석된 정보는 이미 정해진 포맷의 중간 화일로 전달된다. 중간 화일에는 현재 변환이 요구되는 객체의 종류가 식별자를 통해 포함되어 있으며, 객체 종류에 따라 해당되는 LULL 프로그램을 호출하게 된다. 호출된 각각의 import 관련 프로그램은 Schema 설정 화일과 함께 공간 객체 데이타를



〈그림 5〉 SDTS-to-GOTHIC 데이타 변환기의 변환 수행 과정

GOTHIC 데이타베이스에 저장하게 된다.

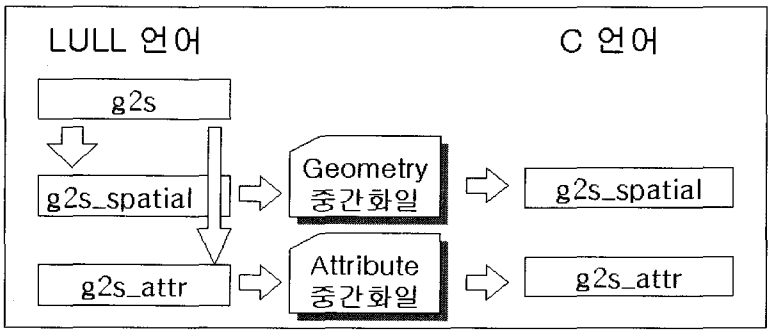
5.3 GOTHIC-to-SDTS 데이타 변환기의 구현

GOTHIC-to-SDTS 데이타 변환기는 전체적인 GOTHIC 데이타셋 정보 및 GOTHIC 공간 객체 클래스를 분석하여 미리 설정된 중간 화일로써 저장하여 전달하고, 각각의 공간 객체 타입에 맞는 SDTS 모듈을 생성하는 역할을 수행한다. GOTHIC-to-SDTS 데이타 변환기는 g2s라는 LULL 프로그램에서 전체적인 프로그램의 흐름을 제어한다. g2s에서는 GOTHIC-to-SDTS 데이타 변환기의 메인 인터페이스를 관리하는 역할을 수행하며, 다른 LULL 프로그램 호출, 중간 화일 생성 등의 전체적인 프로그램 흐름을 총괄한다. 그림 6은 GOTHIC-to-SDTS 데이타 변환기의 변환과정 중에서 공간 객체의 변환 과정의 흐름을 보여준다.

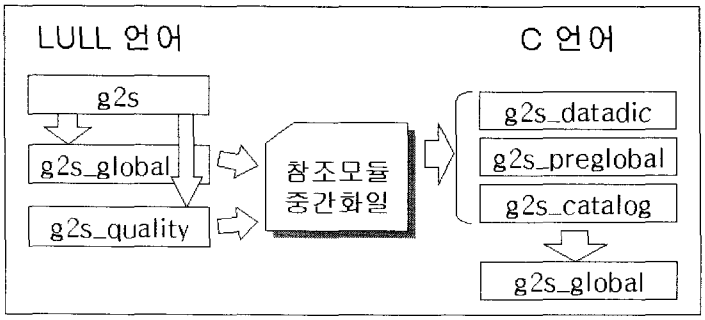
그림 6에서 보는 바와 같이 중간 화일의 좌우에 동일한 이름의 실행 프로그램이 있지만, 각각은 LULL 언어와 C 언어로 구현된 다른 프로그램이다. LULL 언어로 구현된 g2s_spatial은 현재 GOTHIC 데이타베이스에 포함된 공간 객체의 Geometry 값을 분석하고, LULL 언어로 구현된 g2s_attr은 각각 공간 객체가 포함하고 있는 Attribute 값을 분석한다. 그 분석된 값은 각각의 중간 화일로 만들어서 SDTS의 Spatial Object 모듈, Attribute 모듈을 생성하는데 사용된다.

SDTS에는 Spatial Object 모듈, Attribute 모듈 외에 Global 모듈과 Data Quality 모듈이 필수 모듈로 존재하는데, 그에 관련된 정보를 전달하는 것은 참조모듈 중간 화일이다. 그림 7은 GOTHIC-to-SDTS 데이타 변환기의 변환 과정 중에서 참조모듈의 변환 과정을 보여준다.

참조모듈 중간 화일을 생성하는 과정도 s2g에 의해 전체적인 흐름이 제어된다. Global 모듈에 포함될 정보는 LULL 언어로 구현된 g2s_global에서 분석되며, Data Quality 모듈에 포함될 정보는 LULL 언어로 구현된 g2s_quality에서 분석된다. 분석된 각각의 정보는 참조모듈 중간 화일에 저장되어 SDTS 모



〈그림 6〉 GOTHIC-to-SDTS 공간 객체의 변환 수행 과정

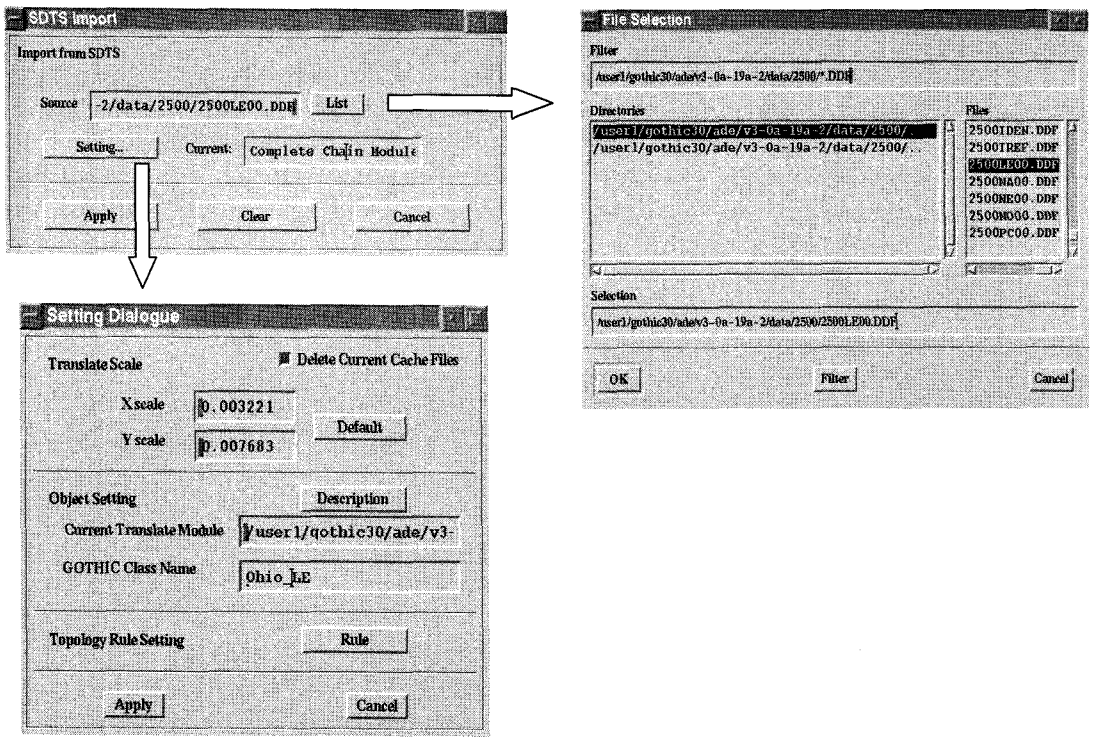


〈그림 7〉 GOTHIC-to-SDTS 참조모듈의 변환 수행 과정

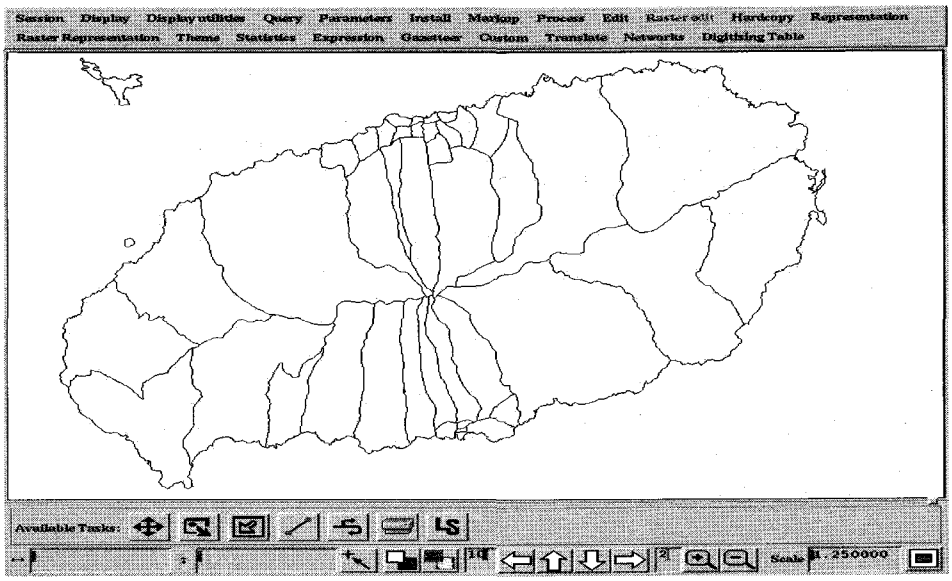
듈 생성기로 전달된다. 여기에서 g2s_global과 g2s_quality는 각각의 정보를 입력하기 위한 독자적인 사용자 인터페이스를 관리하는 역할도 함께 수행한다. 이러한 정보들은 SDTS 모듈에 저장될 수 있도록 그 형식에 조금의 변형을 주는 과정을 거쳐 최종적으로 g2s_global에서 SDTS 모듈을 생성하는데 사용된다.

5.4 데이타 변환 시스템 인터페이스의 구현

GOTHIC을 위한 SDTS 데이타 변환 시스템에서 사용자와 시스템간의 편리한 인터페이스를 위해 Graphic User Interface가 제공된다. 본 논문에서는 사용자 인터페이스를 구현하기 위해서 UIL이 사용되었으며, 사용자 인터페이스와 전체 프로그램간의 상호 정보 교환은 각각의 인터페이스를 호출하는 LULL 프로그램 내의 특정 변수에 값을 설정하고 전달하는 형식을 이용하였다.



〈그림 8〉 SDTS-to-GOTHIC 데이타 변환기의 인터페이스 흐름도



〈그림 9〉 SDTS-to-GOTHIC 데이타 변환의 예

5.4.1 SDTS-to-GOTHIC 데이타 변환기의 사용자 인터페이스

SDTS-to-GOTHIC 데이타 변환기는 GOTHIC ADE 상에 구현되며 사용자는 "SDTS Import"란 메뉴를 통해 메인 인터페이스를 호출할 수 있다. 그림 8은 SDTS-to-GOTHIC 데이타 변환기의 인터페이스 흐름도이다. 메인 인터페이스는 "SDTS Import"이며, 모든 서브 인터페이스로의 이동은 특정 버튼이 클릭되었을 때 이루어진다.

"List" 버튼이 클릭되면 호출되는 인터페이스는 "File Selection" 인터페이스로 사용자는 자신이 변환을 원하는 SDTS 모듈을 선택한다. "Setting..." 버튼은 모듈이 선택된 후에 사용자가 공간 객체의 스케일, 공간 객체들 사이에 Topology Rule 등을 설정할 수 있도록 해 준다. 모든 설정이 완료되면 "Apply" 버튼을 통해 SDTS 모듈에서 GOTHIC 클래스로 변환을 시작하게 된다. 그림 9는 제주도 경계면을 나타내는 SDTS 화일을 GOTHIC 클래스로 변환하여 실제 GOTHIC ADE 화면상에 디스플레이한 내용을 보여준다.

5.4.2 GOTHIC-to-SDTS 데이타 변환기의 사용자 인터페이스

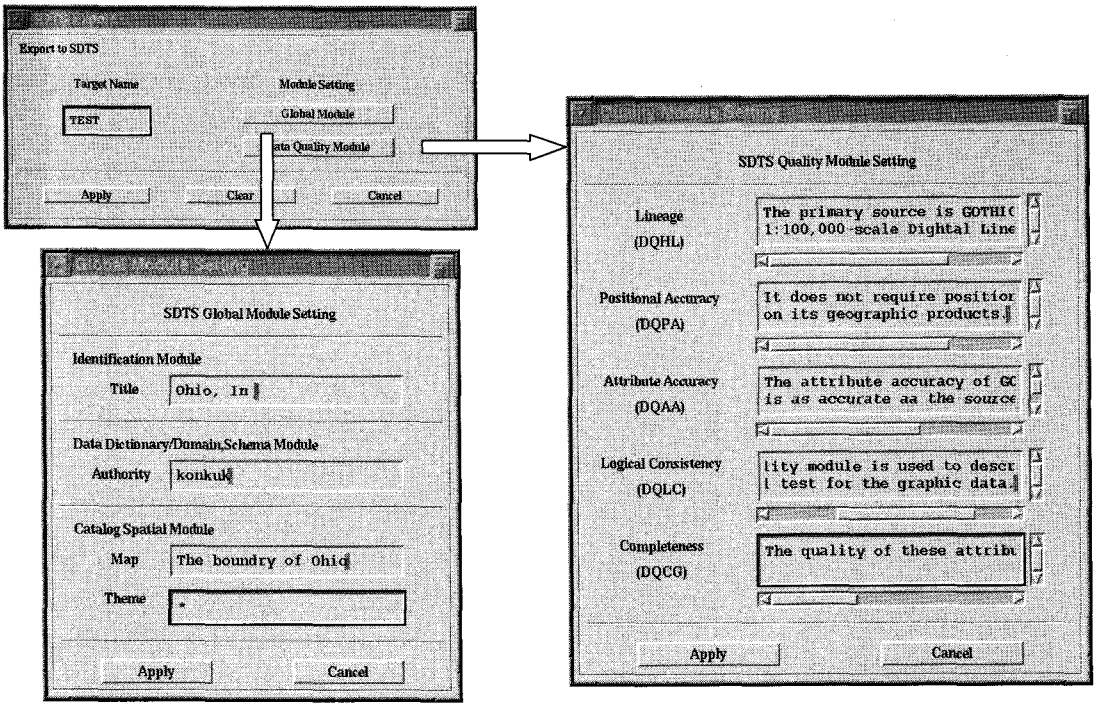
GOTHIC-to-SDTS 데이타 변환기는 GOTHIC 데이타베이스 내에 클래스 형태로 존재하는 공간 객체 데이타를 SDTS 화일로 변환하기 위한 사용자 인터페이스를 제공한다. 변환을 시작하려는 사용자는 GOTHIC ADE 상에서 "SDTS Export"란 메뉴를 선택함으로써 GOTHIC-to-SDTS 데이타 변환기의 메인 인터페이스를 호출할 수 있다. 그림 10은 GOTHIC-to-SDTS 데이타 변환기에서 사용자가 이용하는 인터페이스의 흐름을 보여준다.

처음으로 호출되는 "SDTS Export" 인터페이스에서 Global 모듈에 저장될 정보를 위해 "Global Module"이란 버튼이 제공되며, Data Quality 모듈에서 사용될 정보를 위해 "Data Quality Module"이란 버튼이 제공된다. 모든 설정이 완료되면 "SDTS Export" 인터페이스의 "Apply" 버튼을 이용하여 GOTHIC 클래스 분석 및 SDTS 모듈 생성을 시작하게 된다.

그림 11은 서울에 존재하는 동들의 경계면을 나타내는 GOTHIC 데이타를 SDTS 화일로 변환한 후 디스플레이한 내용을 보여준다.

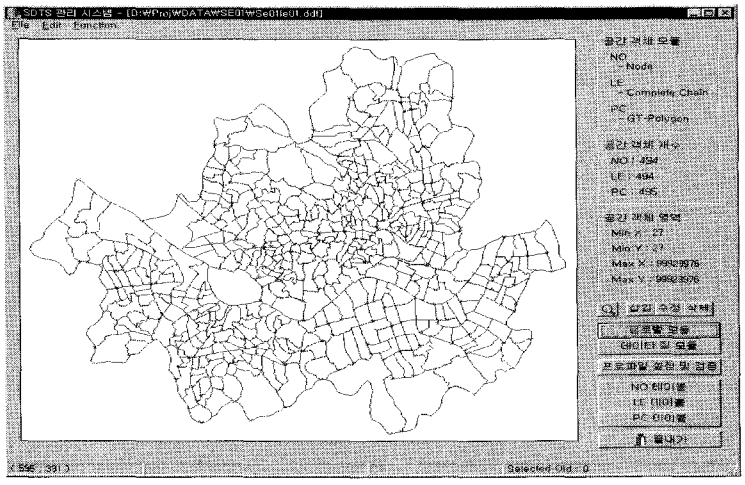
6. 결 론

현재 많은 공공기관이나 기업에서는 각각의 특성에 맞는 여러 종류의 지리 정보 시스템을 사용하고 있다. 지리 정보 시스템의 사용이 발달한 선진국에서는 정부 차원에서의 수치지도 구축이 이루어져 있고, 프로그램



〈그림 10〉 GOTHIC-to-SDTS 데이타 변환기의 인터페이스 흐름도

공급사에서는 각 프로그램에 맞는 데이타로 변환하여 각 사용자에게 제공하고 있다. 그러나, 우리나라의 경우 사용자들이 자신의 시스템에 맞는 데이타를 초기 취득하는데 대부분의 시간을 소비하고 있으며, 제각기 필요한 데이타를 자신의 포맷으로 각자 구축함으로써 지리 데이타가 중복 구축되는 것은 물론이고, 그에 따른 시간과 비용이 낭비되고 있는 실정이다. 또한, 이러한 데이타베이스의 구축과 유지에 따른 비용이 전체 지리 정보 시스템 구축 예산에서 대부분을 차지하고 있으므로 데이타의 공유는 지리 정보 시스템 활용에



〈그림 11〉 GOTHIC-to-SDTS 데이타 변환의 예

있어서 필수적인 사항이라 하겠다. 이러한 데이타의 공유를 위해 우리나라에서는 공간 데이타의 교환 표준 포맷으로 SDTS가 채택되어 있는 상태이기 때문에 SDTS를 중심으로 한 여러 지리 정보 시스템간의 데이타 변환은 필수적이라 할 수 있다.

본 논문에서는 국가 공통 데이타 교환 포맷인 SDTS와 기존의 지리 정보 시스템인 GOTHIC간의 데이타를 상호 교환하는 데이타 변환 시스템을 설계 및 구현하였다. 이를 위해 본 논문은 상호 교환 대상이 되는 SDTS와 GOTHIC을 분석하였고, 또한 이를 바탕으로 최소한의 데이타 손실을 야기하는 SDTS와 GOTHIC 공간 객체들간의 효율적인 매핑을 설계하였다. 그리고, 전체적인 데이타 변환 시스템은 SDTS 화일에서 GOTHIC 데이타로 변환과 GOTHIC 데이타에서 SDTS 화일로의 변환을 구분하여 설계 및 구현하였다.

SDTS-to-GOTHIC 데이타 변환기와 GOTHIC-to-SDTS 데이타 변환기의 구현을 세부적으로 살펴보면 변환하고자 하는 데이타의 분석, 중간 화일 생성, 변환된 데이타의 저장 단계를 거치게 된다. SDTS 모듈 분석 및 저장은 C 프로그램으로 구현되고, GOTHIC 데이타베이스에 대한 엑세스는 LULL 프로그램으로 구현되었다. 각각의 프로그램에서 분석된 SDTS 데이타와 GOTHIC 데이타는 중간 화일에 저장되어 각각을 저장하는 프로그램으로 전달되며, 이 때 생성되는 중간 화일의 구조는 저장되는 시스템의 데이타 형식에 맞게 설계되었다. 본 논문에서는 이러한 여러 고려 사항을 토대로 GOTHIC 라이브러리와 FIPS123 라이브러리를 이용하여 전체 SDTS 데이타 변환 시스템을 개발하였다.

이후에 추가적으로 연구되어야 할 분야가 있다면 GOTHIC을 위한 SDTS 데이타 변환 시스템을 통해 변환된 데이타의 정확성 및 정밀성 체크를 위한 검증 방법을 연구하고, 이를 활용하여 본 논문에서 개발한 SDTS 데이타 변환 시스템을 보완하는 일이다.

참고 문헌

[1] Altheide, P., "An Implementation Strategy for SDTS Encoding," Cartography and Geographic Information System, Vol.19, No.5, Dec. 1992, pp. 306-310.

[2] Billing, J., Gothic KGIS OODB Consultation, Aug. 1996.

[3] Federal Geographic Data Committee, Content Standards for Digital Spatial Metadata, June 1994.

[4] Gunther, G., and Buchemann, A., "Research Issues in Spatial Databases," ACM SIGMOD Record, Vol. 19, No. 4, Dec. 1988, pp. 61-68.

[5] Laser-Scan, GOTHIC Module Reference Manual, Ver2.0, 1994

[6] Lazar, R., "The SDTS Topological Vector Profile," Cartography and Geographic Information System, Vol.19, No.5, Dec. 1992, pp. 296-299.

[7] Lazar, R., "Conformance Testing for the Spatial Data Transfer Standard," Cartography and Geographic Information System, Vol.21, No.3, July 1994, pp. 159-161.

[8] Medeiros, C.B., and Pires, F., "Databases for GIS," Proc. of ACM SIGMOD Int. Conf., Mar. 1994, pp. 107-115.

[9] National Institute of Standards and Technology, The Spatial Data Transfer Standard, Federal Information Processing Standard Publication 173, U.S. Department of Commerce, 1992.

[10] PlanGraphics, SDTS Handbook for Technical Staff, Dec. 1997.

[11] 김준종, 설영민, 이강준, 한기준, "SDTS와 GOTHIC간의 데이타 변환 시스템의 개발," 한국정보과학회 학술발표논문집, 제 26권 1호, Oct. 1998, pp. 170-172.

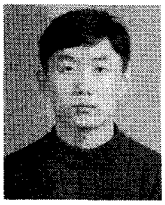
[12] 설영민, 이강준, 한기준, "SDTS Wizard의 설계 및 구현," 한국정보과학회 학술발표논문집, 제 26권 1호, Apr. 1999, pp. 176-178.

[13] 쌍용정보통신, GOTHIC 소개, Apr. 1998.

[14] 오병우, 한기준, "지리 정보 시스템을 위한 표준화," 한국정보과학회 정보과학회지, 제 13권 10호, 1995, pp. 46-55.

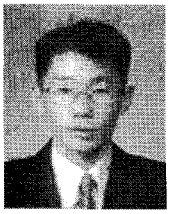
[15] 정선영, 최신영, 서재화, 이기준, "공간 객체 저장 시스템 GeoStore/ Shore와 SDTS 간의 공간 데이타 변환방법에 관한 연구," 한국정보과학회 논문지, 제 24권 2호, Oct. 1997.

[16] 한국통신 선로기술연구소, GIS DB용 데이타 포맷 변환 도구 개발, 최종보고서, Dec. 1996.



장염승

1993년 (中國)大連理工大學 고분자화공학과 졸업(공학사).
1993년~1997년 中國石油化工總公司 撫順石油化工研究院.
1999년 건국대학교 대학원 컴퓨터공학과 졸업(공학석사).
1999년~현재 건국대학교 대학원 컴퓨터공학과 박사과정.
관심분야: 지리 정보 시스템, 객체 관계형 데이타베이스, 컨포넌트GIS, GIS ASP.



김준종

1998년 건국대학교 컴퓨터공학과 졸업(공학사).
2000년 건국대학교 대학원 컴퓨터공학과 졸업(공학석사).
2000년~현재 (주) 핸디소프트 eCW팀 책임연구위원
관심분야: 데이타 마이닝, 관계형 데이타베이스, 객체지향 데이터베이스

한기준

1979년 서울대학교 수학교육학과 졸업(이학사).
1981년 한국과학기술원 전산학과 졸업(공학석사).
1985년 한국과학기술원 전산학과 졸업(공학박사).
1990년 Stanford 대학 전산학과 visiting scholar.
1985년~현재 건국대학교 컴퓨터공학과 교수.
관심분야: 지리 정보 시스템, 객체지향 테이타베이스, 공간 데이타 마이닝, 주기억-상주 데이타베이스.