

工學碩士 學位論文

S-57 전자해도의 GML 변환 및
데이터베이스 관리

GML Conversion and Database Management of S-57 Electronic
Navigational Charts

指導教授 金 載 熏

2007年 2月

韓國海洋大學校 大學院

컴퓨터工學科

田 成 煥

本 論文을 田成煥의 工學碩士 學位論文으로 認准함

委員長 工學博士 柳 吉 洙 印

委 員 工學博士 辛 沃 根 印

委 員 工學博士 金 載 熏 印

2006年 12月

韓國海洋大學校 大學院

컴퓨터工學科 田 成 煥

목 차

그림 목차.....	ii
표 목차.....	iii
Abstract.....	iv
제 1 장 서론.....	1
제 2 장 관련 연구.....	3
2.1 S-57 전자해도	3
2.2 GML S-57 응용 스키마	9
2.3 XML 데이터베이스	13
2.4 XML 질의어	15
제 3 장 전자해도 변환 및 데이터베이스 설계.....	19
3.1 S-57 전자해도의 GML 변환.....	21
3.2 GML 전자해도의 데이터베이스 저장.....	25
3.3 GML 전자해도의 사용자 인터페이스.....	27
제 4 장 시스템의 구현.....	34
4.1 구현 환경	34
4.2 구현된 시스템	35
제 5 장 결론.....	41
참고 문헌.....	42

그림 목차

그림 2.1 전자해도의 예.....	4
그림 2.2 S-57 객체 클래스의 구조.....	5
그림 2.3 공간 객체를 위한 벡터 모델.....	6
그림 2.4 기본 스키마.....	9
그림 2.5 S-57 응용 스키마 구조.....	11
그림 2.6 GML 스키마 “Object.xsd-FRID”.....	12
그림 3.1 제안된 시스템 구조.....	20
그림 3.2 S-57의 GML 변환과정.....	21
그림 3.3 GML 변환 알고리즘.....	22
그림 3.4 최종 GML 문서.....	24
그림 3.5 GML 저장 구조.....	25
그림 3.6 XML 데이터베이스 저장 구조.....	26
그림 3.7 XML 데이터베이스 저장 모듈 순서도.....	27
그림 3.8 XML 데이터베이스 연결.....	28
그림 3.9 GML 질의 처리.....	30
그림 3.10 GML 코드 표시.....	32
그림 4.1 구현된 시스템 구조도.....	35
그림 4.2 S-57, 텍스트, GML 문서.....	36
그림 4.3 XML 데이터베이스의 구조.....	37
그림 4.4 질의 화면.....	39
그림 4.5 GML 결과 화면.....	40

표 목차

표 2.1 S-57 객체 클래스의 구조	5
표 2.2 식별자의 종류와 의미	7
표 2.3 종류에 따른 어트리뷰트 분류.....	8
표 2.4 사용빈도에 따른 어트리뷰트 분류.....	8
표 4.1 구현 환경.....	34
표 4.2 데이터의 크기	38

GML Conversion and Database Management of S-57 Electronic Navigational Charts

Sung-Hwan Jeon

Department of Computer Engineering,
Graduate School, Korea Maritime University, Busan, Korea

Advisor:Jae-Hoon Kim

Abstract

Electronic Navigational Charts (ENC's) are digital charts encoded in S-57 format, which contain navigational informations such as coastlines, depth areas, and nautical marks. Although they have been successfully used for safe navigation of ships, they have limited usages and applications because of their specialized data format and access systems. To cope with such drawbacks, S-57 ENC's need to be transformed into more generalized format such as Geography Markup Language (GML). The transformed GML ENC's can be kept in a database for efficiency, and can be accessed through Internet for usability. This thesis proposes a new method for transforming the S-57 ENC's into GML for and managing the XML database on GML. S-57 ENC's are first translated into GML data, and then stored in a XML database. On the database, users can query for their

needs. To validate the feasibility of the proposed method, a prototype system is developed, and then several test runs are conducted. The system can provide users with easy access to marine informations contained in ENC's. It also provides accessibility and efficiency, by virtue of GML and database, respectively.

제 1 장 서 론

전자해도(ENC:Electronic Navigational Chart)는 해안선, 수심, 항로표지(등대, 부이), 위험물, 항로 등과 같은 다양한 정보가 국제수로기구(IHO:International Hydrographic Organization)의 S-57에 따라 제작된 디지털 지도이다. 이러한 전자해도는 해양 지리정보로서의 유용성에도 불구하고 전자해도 표시시스템, 항해용 전자참고도, 혹은 어선 조업용 장치 등과 같이 특수목적의 장비에서 주로 사용되고 있다. 하지만 최근 해양에 대한 관심이 높아지면서 언제 어디서나 바다에 대한 정보를 이용하려는 사용자들의 요구가 증대되고 있다. 이러한 요구에 부응하기 위하여 전자해도가 활용될 수 있음에도 불구하고 고가의 전용 장비나 브라우저를 구비해야 하는 문제 때문에 그 활용이 쉽지 않다. 또한, 전자해도는 특수한 형태의 S-57로 기술되어 있기 때문에 다른 시스템과의 호환성이 떨어지고 다양한 활용에 많은 어려움이 있다.

이전의 연구를 살펴보면 S-57을 XML로 변환하여 데이터 교환을 효율적으로 사용하려는 시도가 있었다[1]. 그러나 복잡한 형태의 지리정보를 표현하기가 어려웠다. 최근에는 OGC(Open Geospatial Consortium)에서 지리정보를 더 효율적으로 저장, 활용할 수 있는 GML(Geography Markup Language)을 표준안으로 제안하였고, XML(eXtensible Markup Language)의 구조를 따르기 때문에 다른 시스템과의 호환성이 뛰어나고 사용이 간편하며, 재사용성 및 확장성이 뛰어나다는 장점으로 많은 분야에 적용할 수 있다. 그리고 변환된 GML을 SVG로 변환해서 사용자들이 손쉽게 전자해도를 웹 상에서 볼 수 있도록 전자해도 시스템에 관련된 논문이 있었다[2]. 그러나 이전의 논문에서는 관계형 데이터베이스를 사용하고 있기 때문에 속도가 느리고 GML을 데이터베이스로 저장하는데 많은 어려움이 있었다.

그 외 GaldosSystem사에서 S-57을 GML로 변환하여 여러 응용 분야에 적용시키려는 연구가 진행되고 있고[3], ESRI사의 ArcGIS 제품에서는 S-57을 XML로 변환해서 관계형 데이터베이스에 저장 및 표현하는 시스템이 개발되었다[4]. 그리고 CNAVI사에서는 『cel』 포맷을 개발하여 S-57을 『cel』 포맷으로 변환하고 PC 또는 PDA에서 전용의 브라우저를 사용해서 쉽게 전자해도를 보여주는 시스템을 개발하였다[5].

본 논문에서는 우선 S-57로 표현된 전자해도를 GML로 변환한다. 그리고 변환된 GML 전자해도를 효율적으로 관리하기 위하여 기존의 관계형 데이터베이스가 아닌 XML 데이터베이스에 저장한다. 저장된 XML 데이터베이스에 대하여 사용자는 자신이 원하는 정보를 손쉽게 검색할 수 있도록 XQuery와 XPath를 이용한다. 그 결과, GML에 의한 데이터 교환과 웹 접근이 용이해질 뿐만 아니라 데이터베이스에 의한 관리가 용이해지고 사용자의 사용 목적에 따라 다양한 활용이 가능하게 된다.

논문의 2장에서는 관련 연구로서 S-57 전자해도, GML S-57 응용 스키마, XML 데이터베이스, XML 질의어에 대하여 살펴본다. 그리고 3장에서는 전자해도의 변환, 전자해도를 XML 데이터베이스에 저장하고 데이터베이스 활용을 위한 사용자 인터페이스를 설계하고, 4장에서는 시스템의 구현을 기술한다. 마지막으로, 5장에서 결론을 맺고 향후 연구과제를 논한다.

제 2 장 관련 연구

본 장에서는 S-57문서를 GML형식의 문서로 변환하기 위해서 S-57 전자해도 구조와 지리 정보 표준으로 제안된 OGC의 GML에 대한 구조를 살펴보고, GML문서를 저장하기 위한 XML 데이터베이스의 특징과 저장된 GML의 질의를 위한 XQuery와 XPath에 대해서 알아본다.

2.1 S-57 전자해도

S-57은 전자해도 등 해양 데이터의 표현 및 교환을 위한 표준으로 국제수로 기구가 2000년 11월에 버전 3.1을 공표하였다. S-57은 지도상의 등대, 항구 등과 같은 특징(feature) 객체 및 위치 표현을 위한 공간(spatial) 객체로 구성되며, 각 객체는 객체 식별자(identifier)와 어트리뷰트(attribute)로 구성된다 [6].

S-57 전자해도 데이터를 표현하거나 이용하기 위해서는 전용의 장비나 브라우저가 필요로 하며, 그림 2.1은 S-57 전자해도 데이터를 사용하여 전자해도를 표시하는 전용 시스템의 예를 보인다.

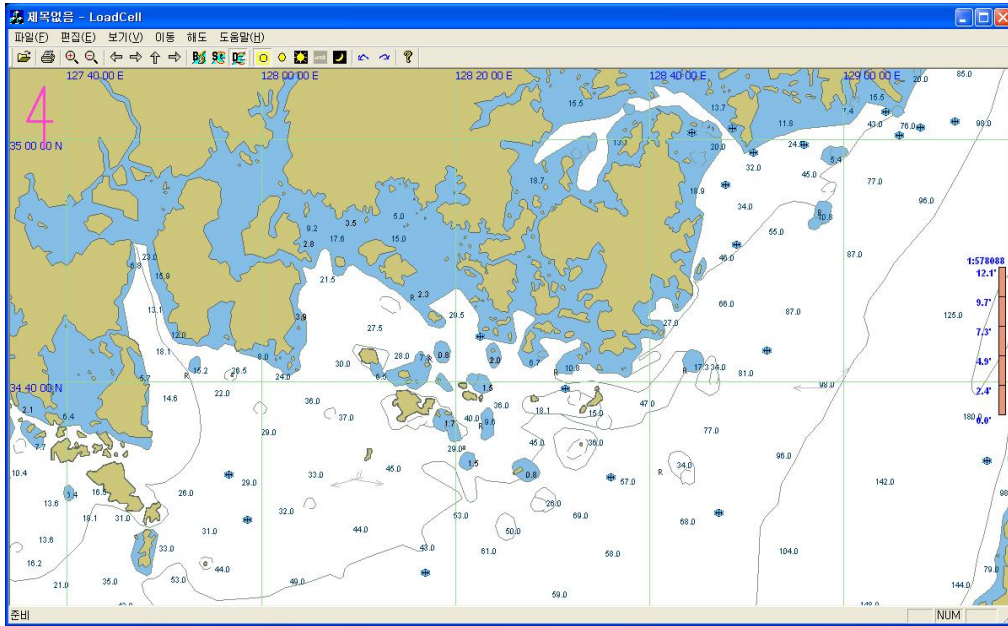


그림 2.1 전자해도의 예
 Fig. 2.1 An example of an electronic navigational chart

전자해도를 위한 데이터 표준 형식인 S-57은 현실세계의 실체, 즉 객체 클래스(object class)들의 집합으로 구성된다. 객체 클래스는 특징 객체와 공간 객체로 구성된다.

그림 2.2에서는 등대, 항구, 부이 등과 같은 특징 객체와 위치를 표현하는 공간 객체의 클래스 구조를 보여주고 있다[6].

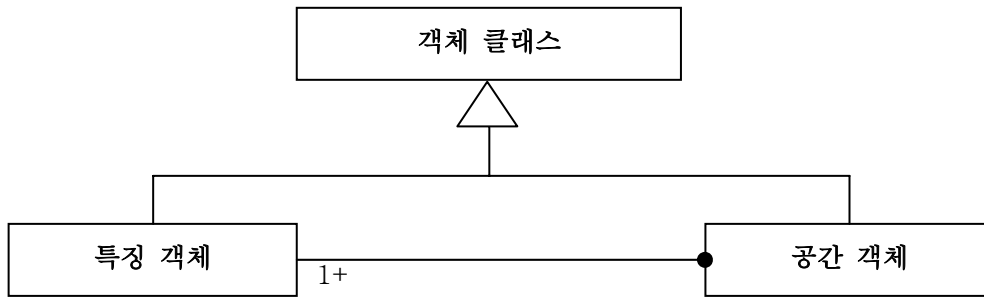


그림 2.2 S-57 객체 클래스의 구조
 Fig. 2.2 The structure of S-57 object classes

(1) 특징 객체

특징 객체는 데이터의 효율적인 교환을 위해서 네 가지 범주로 구성되며, 특징 객체들의 종류와 의미는 표 2.1과 같다. 먼저 Meta 특징 객체들은 데이터의 정확도, 편집 범위, 수행 신빙성 등의 정보를 표현한다. Cartographic 특징 객체들은 라인, 심벌, 나침반, 문자열 등의 해도 정보를 표현한다. Geo 특징 객체들은 등대, 항구, 부이 등의 현실세계 엔티티들의 정보를 표현한다. Collection 특징 객체들은 데이터의 집합이나 연관관계 등의 정보를 표현한다 [6].

표 2.1 S-57 객체 클래스의 종류
 Table 2.1 The types of S-57 object classes

특징 객체의 종류	특징 객체의 의미	예
Meta	다른 객체들에 대한 정보를 포함	데이터 정확도:M_ACCY
Cartographic	텍스트를 포함한 현실세계 엔티티의 지도 제작 표현에 대한 정보를 포함	라인:\$LINES
Geo	현실세계 엔티티들의 상세한 특징들을 포함	등대:LIGHT
Collection	다른 객체간의 관계 기술	연관:C_ASSO

(2) 공간 객체

특정 객체는 하나, 혹은 그 이상의 공간 객체와 결합된다. 공간 객체란 위도와 경도 좌표의 쌍, 또는 위도, 경도 좌표 및 깊이의 쌍으로 표현되는 위치 정보이다. 공간 객체를 표현하기 위한 방법으로 S-57 전자해도 데이터에서는 벡터(Vector), 래스터(Raster), 매트릭스(Matrix)의 세 가지 방법을 제시하며 공간 객체의 벡터 모델을 UML(Unified Modeling Language) 다이어그램으로 표현하면 그림 2.3과 같다[6].

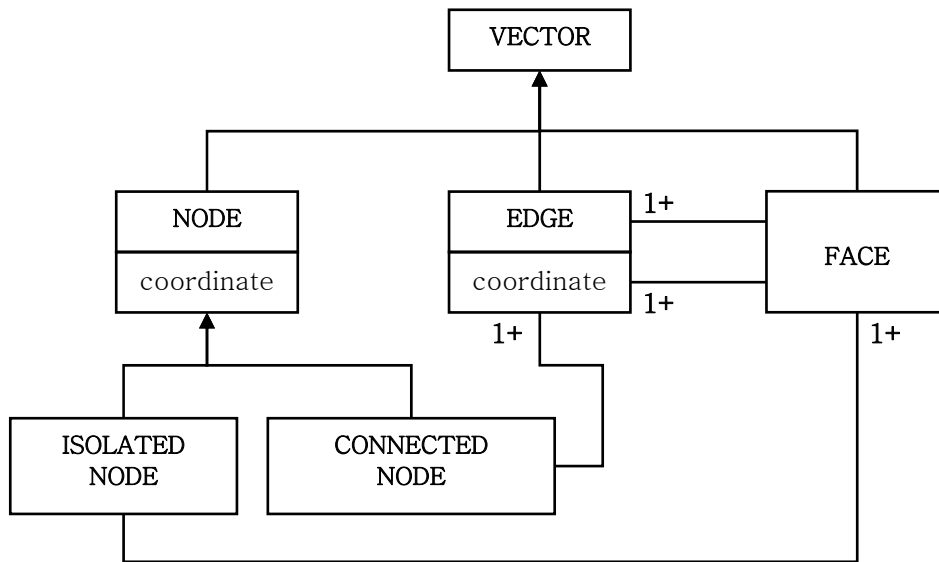


그림 2.3 공간 객체를 위한 벡터 모델
Fig. 2.3 The vector model for spatial objects

이 그림에서처럼 공간 객체에서 벡터로 표현되는 정보의 형태는 NODE, EDGE, FACE의 세 가지이며, 지도에서 표현되는 데이터는 Point(점), Line(선), Area(면)으로 대응한다. Point는 위도와 경도 좌표를 가지는 한 지점의 위치 정보이며, Line은 Point와 Point의 연결, Area는 Point들의 연결로 구성된 다각

형 공간으로 표현된다[6].

(3) 객체 식별자와 어트리뷰트

S-57 전자해도 표준 형식에는 각 객체 클래스들을 위하여 식별자와 여러 가지 어트리뷰트들을 제공한다. 객체 식별자는 해당 객체 클래스가 어떤 객체인지에 관한 정보를 가지고 있으며, 어트리뷰트들은 해당 객체가 어떤 정보를 가질 수 있는지 결정한다. 객체 식별자의 종류와 의미는 표 2.2와 같다[6].

표 2.2 식별자의 종류와 의미

Table 2.2 Types and meanings of identifiers

식별자 종류	식별자 의미	'LIGHTS'객체 클래스의 예
GRUP	그룹	GRUP= "2"
OBJL	객체 클래스의 산술 코드/라벨	OBJL= "75"
RVER	기록 버전	RVER="1"
AGEN	발행 기관	AGEN="550"
FIDN	특징 객체 식별번호	FIDN="234260030"
FIDS	특징 객체 식별 상세번호	FIDS="1315"

S-57 전자해도 표준 형식의 어트리뷰트들은 종류와 사용빈도에 따라서 세 가지로 구분된다. 먼저 종류에 따라서 구분하면 특징 객체 어트리뷰트, 국가 언어 어트리뷰트, 공간과 메타 객체 어트리뷰트로 구분된다. 자세한 내용은 표 2.3과 같다.

표 2.3 종류에 따른 어트리뷰트 분류

Table 2.3 Attribute classification according to their types

어트리뷰트집합의 종류	어트리뷰트집합의 의미	예
특징 객체 어트리뷰트	특징 객체에 관련된 어트리뷰트	색깔:COLOUR
국가 언어 어트리뷰트	국가 언어와 관련된 어트리뷰트	국가언어정보:NINFOM
공간과 메타 객체 어트리뷰트	공간과 메타 객체에 관련된 어트리뷰트	위치정확도:POSACC

어트리뷰트를 사용빈도에 따라 구분하면 SubSet Attribute_A, B, C 같이 세 가지 부분 집합으로 구분되며, 표 2.4에서 보여준다[6].

표 2.4 사용빈도에 따른 어트리뷰트 분류

Table 2.4 Attribute classification according to the frequency of use

어트리뷰트집합의 종류	어트리뷰트집합의 의미	예
SubSet Attribute_A	객체의 개별적 특징 정의	색깔:COLOUR
SubSet Attribute_B	데이터의 사용에 관련된 정보	최대해상도:SCAMAX
SubSet Attribute_C	객체와 데이터에 대한 관리 정보	기록 날짜:RECDAT

S-57은 기본적으로 2진(binary) 형식으로 표현되기 때문에, 사용하는 목적에 따라 적절한 형태의 포맷으로 변환하여야 한다. 본 논문에서는 S-57 데이터를 GML 문서로 변환한 후 XML 데이터베이스에 저장하는 방식을 채택하였다.

2.2 GML S-57 응용 스키마

GML은 지리공간 정보의 저장 및 전송을 위해 데이터를 구조화된 문서인 XML로 표현하며, OGC에서는 2000년 5월에 버전 1.0이 발표된 이후, 2001년에 XML 스키마에 기반을 둔 GML 2.0을 제시하였으며, 2003년에 26개의 코어 스키마를 가진 GML 3.0을 채택하였다[7].

GML 3.0은 특징(feature) 스키마와 지오메트리(geometry) 스키마를 기본 스키마로 사용하고, 이 외의 부가적인 기능을 하는 스키마들이 참조된다. 26개의 코어(core) 스키마는 어플리케이션에서 바로 참조할 수 없다. 따라서 코어 스키마의 구조를 어플리케이션에 적합하도록 재정의한 것이 응용 스키마이다. 응용 스키마의 관계를 표현하면 그림 2.4과 같다. 특징 스키마는 지오메트리 스키마를 참조하고 지오메트리 스키마는 XLinks 스키마를 포함한다[7].

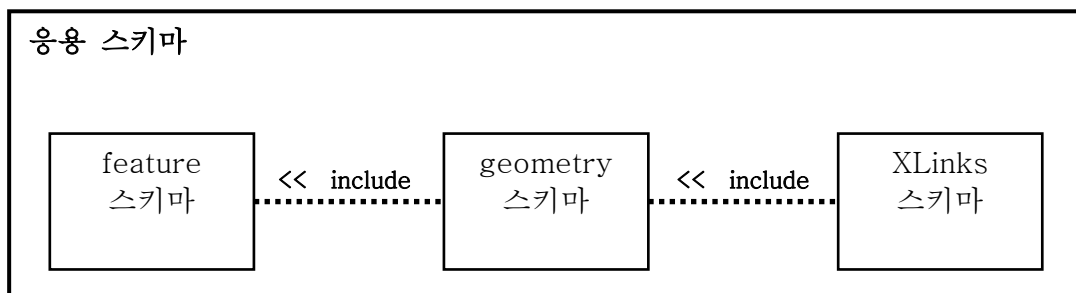


그림 2.4 기본 스키마
Fig. 2.4 Basic schemas

특징 타입, 특징 식별자(fid), 이름, 설명과 같은 공통적인 특징 어트리뷰트를 정의한다. 특징 스키마는 지오메트리 스키마를 참조하며, 이를 위해 include 엘리먼트를 사용한다[7]. 그리고 XLinks 스키마는 Linking 기능을 지원하기 위한 Xlinks 어트리뷰트를 제공한다.

(1) 지오메트리 스키마

지오메트리 스키마에서는 지오메트리 엘리먼트를 정의한다. 이를 위해 OGC 심플 특징 사양에 따라 Curve, Surface, MultiSurface, MultiCurve를 제외한 Point, LineString, LinearRing, Polygon, MultiPoint, MultiLineString, MultiPolygon, 이들에 대한 Geometry Collection 및 Box를 제공한다[7].

GML에서는 지오메트리 정보를 표현하기 위해서 실수(real number) 타입의 x, y, z 좌표를 가지는 Point, LineString, Box, LinearRing, Polygon과 여기서 확장된 MultiPoint, MultiLineString, MultiPolygon등의 지오메트리 타입을 제공한다. GML 지오메트리 스키마에서 제공하는 지오메트리 스키마의 구조는 본 논문에서의 공간 객체 엘리먼트의 스키마 구조와 매우 유사하다. 따라서 본 논문에서는 공간 객체 엘리먼트를 표현하기 위해, 필요한 공간 객체 엘리먼트의 데이터구조인 Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon을 GML의 지오메트리 스키마 구조를 수용하여 사용한다.

(2) GML 응용 스키마

GML의 기본 스키마만으로는 실제 사용할 데이터를 표현하기 위한 적합한 스키마를 제공할 수 없다. 즉 기본 스키마가 지리 정보를 표현하기 위한 기본적인 형태를 제공한다면, 응용 스키마는 개발자가 GML 표준 컴포넌트를 사용해 실제 사용할 특징 타입이나 프로퍼티(property) 타입들을 직접 정의해 놓은 스키마이다. 즉 응용 스키마 개발자는 기본 스키마를 이용함으로써 OGC표준에 대한 깊은 이해 없이도 표준에 맞는 응용을 보다 쉽게 개발할 수 있다. 본 논문에서는 GaldosSystem사에서 개발한 S-57 전자해도용 응용 스키마를 채택하였다. 그림 2.5에서는 GaldosSystem사의 스키마 구조를 보여주고 있으며, 각 스키마의 구성은 다음과 같다[3,8].

- Objects.xsd:S-57 전자해도의 Object에 대한 정의
- Attributes.xsd:Attributes_A, B, C에 대한 정의
- Support.xsd:기본적인 타입
- AbstractAndSuperTypes.xsd:추상 타입, 메타데이터

위의 네 가지 응용 스키마와 코어 스키마에서 필요한 요소들만 모아서 gml4s57.xsd라는 스키마를 만든다. GML 3.0.1 Core 스키마에서 필요한 요소들만 추출해서 gml4s57 스키마와 같이 사용하고 이것을 xlink.sxs에서 참조한다. 최종적으로는 그림 2.5와 같이 6개의 스키마를 사용하게 된다.

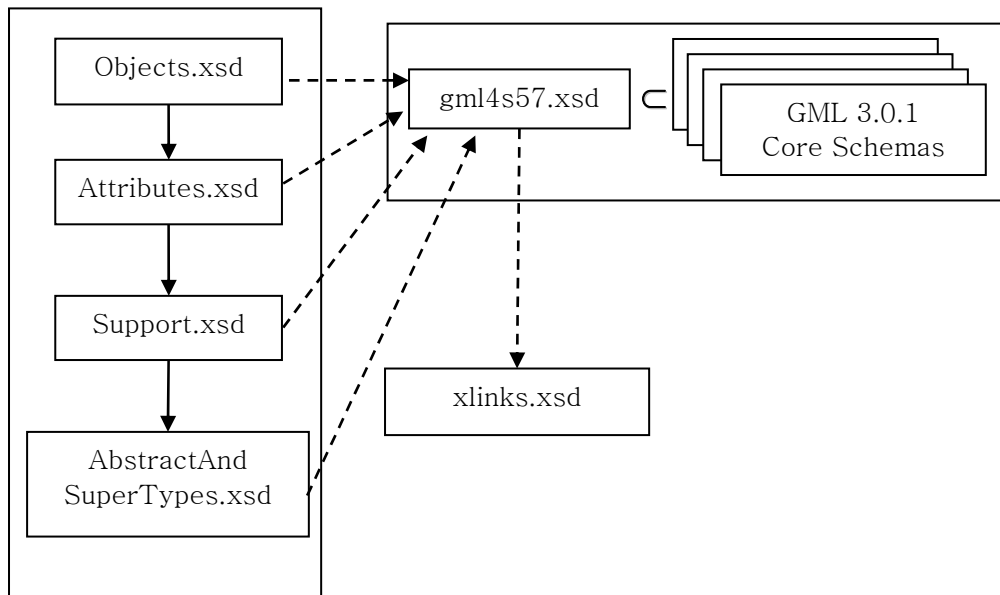


그림 2.5 S-57 응용 스키마 구조
Fig. 2.5 The structure of S-57 application schema

스키마 중에서 Objects.xsd 파일의 FRID(FeatureRecordIdentifier)부분의

스키마를 살펴보면 그림 2.6과 같다.

```
<!--===== element =====>
<element name="FeatureRecordIdentifier"
type="s57:FeatureRecordIdentifierType"
substitutionGroup="gml:_MetaData">
  <annotation>
    <documentation>Feature metadata representing S-57
      Feature Record Identifier (FRID)</documentation>
  </annotation>
</element>
<!--===== complexType =====>
<complexType name="FeatureRecordIdentifierType" mixed="true">
  <complexContent mixed="true">
    <extension base="gml:AbstractMetaDataType">
      <sequence>
        <element name="recordName" type="s57:String3"/>
        <element name="recordIdentificationNumber" type="s57:Integer-
          4294967294"/>
        <element name="group" type="s57:Integer255"/>
        <element name="objectLevel" type="s57:Integer504"/>
        <element name="recordVersion" type="s57:Integer999"/>
        <element name="recordUpdateInstruction" type="s57:String1"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

그림 2.6 GML 스키마의 예:“Objects.xsd-FRID”
Fig. 2.6 An example GML of schema “Objects.xsd-FRID”

FRID 오브젝트의 명칭은 FeatureRecordIdentifierType이고, recordName, recordIdentificationNumber, group, objectLevel, recordVersion, recordUpdateInstruction의 6개의 엘리먼트를 정의해서 사용하는 것을 알 수 있다. 이렇게 총 177개의 특징 객체 오브젝트와 197개의 엘리먼트 스키마를 본 논문에서는 GaldosSystem사에서 제공하는 전자해도용 응용 스키마를 채택하여 사용한다[3].

2.3 XML 데이터베이스

확장형 표시언어(XML)는 1998년 W3C에서 정식표준으로 제정되었고, 사용이 간편하고 재사용성 및 확장성이 뛰어나다[9,16]. 이러한 장점으로 인해 XML은 전자거래, 전자민원서비스 등 많은 분야에서 활용되면서 웹 상에서 유통되는 수많은 정보자원들과 메시지들을 XML 전자문서로 생성하게 되었다. 따라서 업체 및 조직에서는 이런 XML 전자문서들을 효과적으로 저장, 관리할 수 있는 데이터베이스가 필요하다고 인식하게 되어 XML 데이터베이스가 개발되었다.

XML 문서는 크게 데이터중심의 XML 문서와 문서중심의 XML 문서로 나눌 수 있다. 데이터 중심의 XML 문서는 데이터 전송을 위해 XML 문서를 사용하는 경우를 말하며 전자거래를 위한 구매주문서, 온라인상에서의 민원처리를 위한 신청서, 부처간 데이터교환을 위한 메시지 등과 같은 것을 포함한다. 문서중심의 XML 문서는 주로 학술분야의 논문 및 연구보고서, 사용자 메뉴얼, 마케팅 브로셔 등과 같이 웹을 통해서 최종사용자에게 보여주기 위해 작성된 XML 문서를 말한다.

XML 문서를 관리하는 방법으로는 파일 시스템 기반방식, 통합 RDBMS 기반방식, 분할 RDBMS 기반방식이 사용되고 있다. 파일시스템 기반방식은 개별 XML 문서를 파일의 형태로 특정 폴더에 저장하는 것으로, 문서 전체에 대한 추가 작업이 파일단위로 이뤄져 상대적으로 저장 및 추출 속도가 빠르나, XML 문서를 추출 할 때마다 XML 파싱(parsing)이 필요하다는 단점이 있다. 통합 RDBMS 기반방식은 XML 문서를 RDBMS의 BLOB(Binary Large Object)나 CLOB(Character Large Object)의 형태로 저장하는 것으로 문서의 저장 및 추출 속도가 빠르나, 문서의 부분 수정이나 검색에 많은 시간이 필요하다는 단점이 있다. 분할 RDBMS 기반방식은 XML 문서를 구성하는 각 요소 단위로 분

할하여 RDBMS에서 테이블의 필드에 저장하는 것으로, 문서의 부분 수정이나 검색을 빠르게 수행할 수 있으나, 문서를 추출 할 때마다 여러 테이블에 대한 조인과정을 거치므로 속도가 매우 느리다는 단점이 있다.

데이터 중심의 XML 데이터를 저장하고 가져오기 위해서는 관계형 또는 객체지향 데이터베이스 등과 같은 데이터 저장소를 위해 튜닝된 XML 가능 데이터베이스(XML enabled database)와 몇 가지 데이터 전송 소프트웨어가 필요하다. 이러한 전송소프트웨어는 XML 가능 데이터베이스에 구축되거나 서드파티 미들웨어가 될 수 있다.

문서중심의 XML 문서를 저장하거나 가져오기 위해서는 XML 데이터베이스(XML database) 또는 콘텐츠 관리 시스템(contents management system)이 필요하다. XML 데이터베이스 제품은 많은 업체에서 다양하게 제공하고 있는데 XML 데이터베이스의 특징에 따라서 XML 가능 데이터베이스와 XML 데이터베이스로 구분할 수 있다.

- XML 가능 데이터베이스:XML 문서와 문서구조간의 데이터를 전송하기 위해 확장을 제공하는 데이터베이스로서, 주로 데이터 중심의 애플리케이션 또는 데이터 중심의 문서용으로 사용할 수 있다. 대표적인 솔루션으로는 IBM의 DB2 XML Extender, Informix의 Web DataBlade, Microsoft의 XML for SQL Server 2005, Oracle의 Oracle 10g 등이 있다.
- XML 데이터베이스:XML 문서에 대한 (논리적)모델을 정의하고, 그 모델에 따라 XML 문서를 저장하고 추출할 수 있어야 한다. 저장하는 기본 단위로는 특정한 물리적인 저장 모델을 요구하지 않지만, 논리적인 저장단위로 XML 문서이어야 한다. 대표적인 솔루션으로는 Software AG의 Tamino, X-Hive사의 X-Hive/DB, eXcelon사의 eXtensible Information Server(XIS), Ipedo사의 Ipedo 등이 있다.

본 논문에서는 eXcelon 사의 eXtensible Information Server(XIS)를 이용하여 XML 데이터베이스를 구축하였다.

2.4 XML 질의어

(1) XQuery

XQuery는 현재 W3C에서 표준화된 XML 질의 언어이다. XQuery 질의를 구성하는 기본 단위는 표현식(expression)이며, XQuery 질의 언어가 제공하는 기본 표현식으로는 경로 표현식(path expression), FLWR 표현식(FLWR expression), 요소 생성자(element constructor), 비교 표현식(comparison expression) 등이 있다. 그리고 각 표현식은 다른 표현식 안에 중첩될 수 있는 특성이 있다[10,12].

먼저 경로 표현식은 XPath의 형식을 따르는 표현식으로, 경로 표현식에서 대괄호 안에 나타나는 표현식은 술어(predicate)를 의미한다.

질의 1. 경로 표현식을 이용하여 높이가 17미터인 등대를 찾아라.

```
document("KP314200.xml")xlnxql:query=  
"/DataSet/LIGHTS/HEIGHT/Height[code="95" and value="17.00"]"
```

질의 1의 경로 표현식은 XML 문서 "KP314200.xml"에서 루트 요소(root element)인 LIGHTS의 자식 요소(child element) 중에서 HEIGHT를 검색하고, 다시 자식 요소 중에서 Height를 검색하여 Height의 자식 요소인 code의 값이 "95"이며, value의 값이 "17.00"인 것의 검색결과로 Height 반환한다.

질의 1을 수행하면 XQuery 질의의 정규화 결과 1을 얻을 수 있다.

결과 1

```
<Height>
  <code>95</code>
  <value>17.00</value>
</Height>
```

다음으로 FLWR 표현식은 XQuery 질의를 구성하는 주요 표현식으로서, for 절, let 절, where 절, 그리고 return 절로 구성된다. for 절, where 절, return 절은 각각 SQL의 from 절, where 절, select 절과 유사한 의미를 가지며, let 절은 표현식을 하나의 변수로 치환하는 의미를 가진다. 질의 2는 질의 1 경로 표현식을 다음의 FLWR 표현식으로 바꾼 것이다.

질의 2. FLWR 표현식을 이용하여 높이가 17미터인 등대를 찾아라.

```
for $x in
document("KP314200.xml")/DataSet/LIGHTS/HEIGHT/Height
where $x/code="95" and value="17.00"
return $x/Height
```

질의 2을 수행하면 XQuery 질의의 정규화 결과 2를 얻을 수 있다.

결과 2

```
<Height>
    <code>95</code>
    <value>17.00</value>
</Height>
```

(2) XPath

XPath의 주요 목적은 XML 문서에서 특정 부분들을 나타내는 것이다. XPath 표현식(XPE)은 XML 문서 트리의 노드들에 부합되는 구조적인 패턴을 나타낸다[11].

XPE는 엘리먼트와 엘리먼트들 사이의 관계를 기술하는 연산자들을 나타내는 로케이션 스텝(location step)들의 배열(sequence)이다. XPath의 로케이션 스텝에 사용되는 연산자들로는 ‘부모-자식’ 관계를 나타내는 ‘/’, ‘조상-후손’ 관계를 나타내는 ‘//’과 임의의 엘리먼트를 나타내는 ‘*’ 등이 있다. 또한 각 로케이션 스텝은 선택되는 노드 집합을 세밀하게 정의하는 술어(predicate)를 포함할 수도 있다(술어는 ‘[’와 ‘]’ 사이에 기술된다). 술어 사용으로 형제(sibling) 노드들 사이의 순서 및 위치를 기술할 수 있으며 W3C 표준으로서 다음의 기능을 제공한다[13].

- 문자열, 숫자, Boolean 값을 다루기 위한 기본적인 기능을 제공한다.
- URI(Uniform Resource Identifier) 디렉토리를 찾아가는 구문과 속성값을 다루기 위한 다른 XML 구문을 제공한다.
- 표면적인 구문보다는 XML의 논리적인 구조로 추상화되어 작동한다.
- XML의 계층적인 구조를 조사하기 위해서 경로 표기법으로부터 이름을

가져온다.

- XML 문서의 섹션에 주소를 지정하여 필요로 하는 정보의 정확한 조각을 얻을 수 있도록 해준다.

예를 들어, 다음과 같은 XML이 있을 때

```
<nodes>
    <node>a</node>
    <node>b</node>
    <node>c</node>
</nodes>
```

두 번째 <node> 엘리먼트만 가져오고 싶을 때의 XPath는 다음과 같다.

```
<xsl:template match="/nodes/node[position()=2]">
```

제 3 장 전자해도 변환 및 데이터베이스 설계

전자해도는 S-57 포맷의 데이터를 정보교환을 위한 스펙인 IHO/IEC 8211에 따라 코딩된 이진 파일이다. S-57 전자해도를 보다 손쉽게 이용할 수 있도록 전자해도용 GML 응용 스키마를 따르는 GML 문서로 변환하는 프로그램이 필요하다. 본 장에서는 변환 프로그램을 설계하고, 변환된 GML 문서를 저장하고 관리하는 XML 데이터베이스의 구조를 제안한다. 그리고 변환된 문서를 인터넷을 통해서 보다 손쉽게 접근할 수 있는 사용자 인터페이스를 설계한다. 그림 3.1은 본 논문에서 제안하는 S-57 전자해도를 GML로 변환하고, XML데이터베이스에 저장하는 시스템 구조와 사용자 인터페이스를 표현한다.

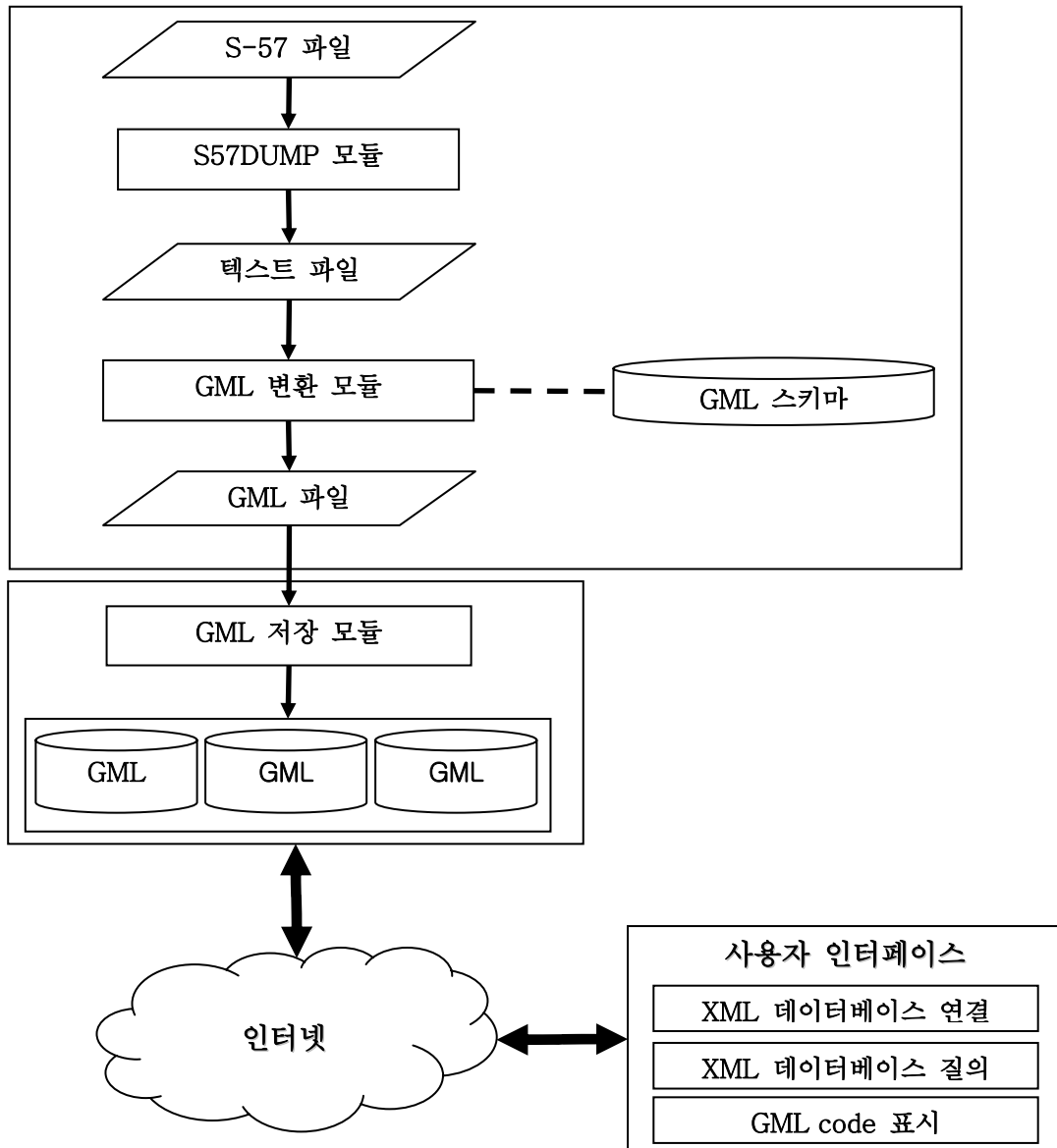


그림 3.1 제안된 시스템 구조
 Fig. 3.1 The structure of the proposed system

3.1 S-57 전자해도의 GML 변환

전자해도를 위한 데이터 표준인 S-57은 객체 클래스의 집합으로 구성된다. 객체 클래스는 지도상의 등대, 부이 등과 같은 지리적 특징 객체와 위치 정보를 나타내기 위한 공간 객체, 객체 식별자와 어트리뷰트로 구성된다. 이러한 S-57 전자해도를 GML로 변환하는 프로그램의 구성은 그림 3.2와 같다.

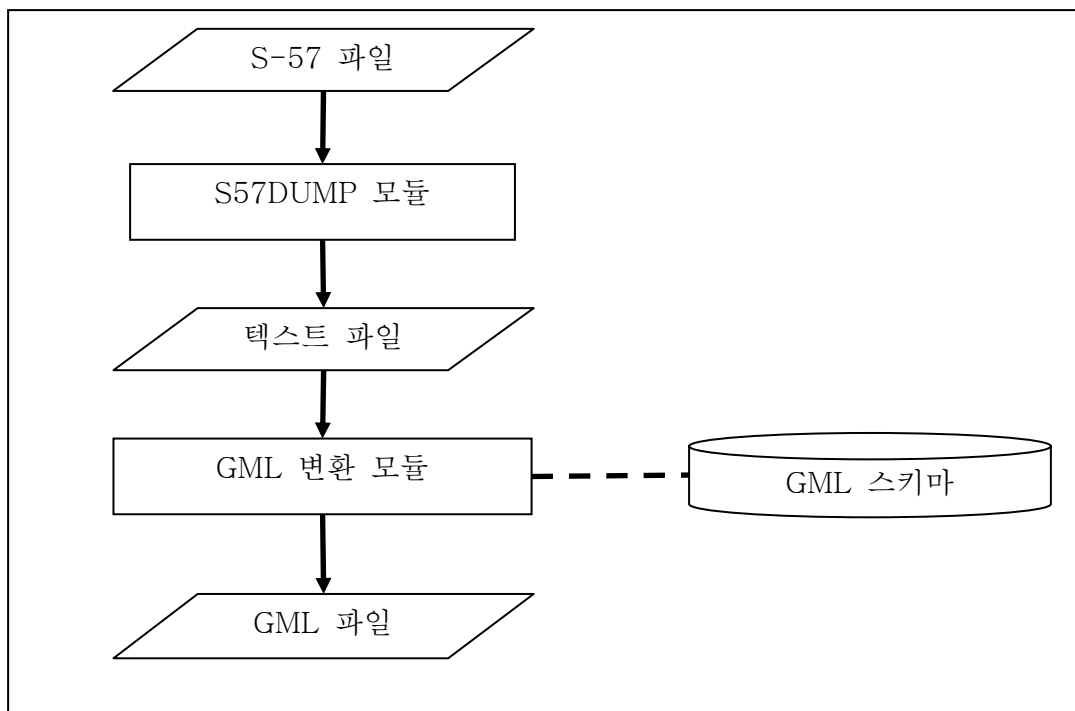


그림 3.2 S-57의 GML 변환과정
Fig. 3.2 The Converting S-57 ENC's into GML

S-57 전자해도는 이진 형식의 파일로 만들어 졌으며 객체 클래스 단위의 순차적인 구조이다. 본 논문에서는 S-57 전자해도 파일을 입력 데이터로 받는다. 먼저, 입력 받은 S-57 전자해도를 객체 클래스 단위로 분리하여 gdal-1.1.5

프로그램 사용해서 변환한다. 텍스트 파일을 GML로 변환하는 과정을 알고리즘으로 표현 것이 그림 3.3이다.

```
Algorithm S-57 Into GML(T)

//입력:S-57파일을 S57DUMP를 이용하여 변환된 텍스트 파일을 'T'라고 사용한다.
//출력:GML 파일을 'GMLfile'이라고 사용한다.

begin
  while (T == EOF) {
    //텍스트 파일을 1줄씩 읽어들인다.
    read_line = get_line() ;
    //1라인을 *단위로 자른다.
    token[] = Tokenizer(read_line, '*') ;
    for (i = 0 ; i < token.size() ; i++) {
      if (isobject (token[i])) {
        if (isfeature_object(token[i])) {
          //텍스트 토큰을 특징GML토큰으로 변환
          GML_token = feature_pattern_matching(token[i]) ;
          //출력에 GML_token을 추가
          GMLfile.write (GML_token) ;
        } else {
          //텍스트 토큰을 공간GML토큰으로 변환
          GML_token = space_pattern_matching(token[i]) ;
          //출력에 GML_token을 추가
          GMLfile.write (GML_token) ;
        }
      }
    }
  }
}
End
```

그림 3.3 GML 변환 알고리즘
Fig. 3.3 An algorithm for converting S-57 into GML

알고리즘을 설명하면 S-57파일을 S57DUMP 프로그램을 사용하여 텍스트 파일로 변환된 전자해도 데이터를 'T'라고 정의하고 입력으로 사용하게 된다.

GML 변환 과정에서 사용되는 함수의 기능을 살펴보면 다음과 같다.

- `get_line()`: T에서 1줄씩 내용을 읽어오며 그 내용을 `read_line`로 넣는다.
- `Tokenizer():read_line` 1줄을 '*' 단위로 자르고 token배열에 저장한다.
- `isobject()`: token이 오브젝트인지 아닌지를 구별한다.
- `isfeature_object()`: 특징 객체를 구별한다.
- `feature_pattern_matching()`: 오브젝트라고 구별된 token을 특징 객체의 스키마를 참조해서 GML로 만들고 `GML_token`에 값을 넘겨준다.
- `space_pattern_matching()`: 오브젝트라고 구별된 token을 공간 객체의 스키마를 참조해서 GML로 만들고 `GML_token`에 값을 넘겨준다.
- `GMLfile.write()`: `GML_token`을 최종 GML파일에 내용을 삽입하는 역할을 한다.

`feature_pattern_matching()`, `space_pattern_matching()` 함수에서 사용되는 스키마는 전자해도용 GML 응용 스키마로 `Objects.xsd`, `Attributes.xsd`, `Support.xsd`, `AbstractAndSuperTypes.xsd`, `gml4s57.xsd`, `xlinks.xsd` 적용하여 적합한 GML 문서로 변환한다. 6개의 스키마는 2.2절에 GML S-57 응용 스키마에서 설명하였다.

그림 3.4는 이러한 스키마를 적용하는 과정을 통하여 변환된 GML 전자해도 문서의 'RIVERS' 특징 객체와 공간 객체를 보여준다. 이 특징 객체는 어트리뷰트와 공간 객체로 구성되어 있으며 각각의 어트리뷰트는 S-57 전자해도가 가지고 있던 데이터 값을 가지고 있다. 변환 과정에서 데이터 값이 없는 빈 어트리뷰트는 삭제된다.

```

<RIVERS>
  <gml:metaDataProperty>
    <FeatureRecordIdentifier>
      <group>2</group>
      <objectLevel>114</objectLevel>
      <recordVersion>2</recordVersion>
    </FeatureRecordIdentifier>
  </gml:metaDataProperty>
  <gml:metaDataProperty>
    <FeatureObjectIdentifier>
      <agency>280</agency>
      <featureIdentificationNumber>792874</featureIden
      <featureIdentificationSubdivision>1</featureIdentifi
    </FeatureObjectIdentifier>
  </gml:metaDataProperty>
  <STATUS>
    <Status>
      <code>149</code>
      <idList>1</idList>
      <value>permanent</value>
    </Status>
  </STATUS>
  <SCAMIN>
    <ScaleMinimum>
      <code>133</code>
      <value>50000</value>
    </ScaleMinimum>
  </SCAMIN>
  <gml:element>
    <gml:Polygon>
      <gml:pos>'129.20164000';'35.44349000'</gml:pos>
      <gml:pos>'129.20119000';'35.44310000'</gml:pos>
      <gml:pos>'129.20164000';'35.44349000'</gml:pos>
    </gml:Polygon>
  </gml:element>
</extent/>
</RIVERS>

```

그림 3.4 최종 GML 문서
 Fig. 3.4 The final GML document

3.2 GML 전자해도의 데이터베이스 저장

본 논문에서는 GML 전자해도의 저장 및 관리, 사용자 질의를 효율적으로 처리할 수 있는 XML 데이터베이스를 설계한다. 기존의 관계형 데이터베이스(RDBMS)는 XML과 기본 데이터 모델이 상이하기 때문에 XML 형식의 데이터를 저장하는데 많은 어려움이 있다[14]. 이러한 어려움을 극복하기 위하여, 본 논문에서는 XML 데이터베이스인 Sonicsoftware사의 (XIS:eXtensible Information Server)를 이용하여 GML 전자해도 데이터베이스를 구축하였다. 이러한 XML 데이터베이스는 GML 전자해도를 체계적으로 저장할 수 있을 뿐만 아니라 사용자의 검색 질의를 효율적으로 처리할 수 있으며 XIS를 이용하여 설계한 GML 저장 구조를 그림 3.5에서 보여준다.

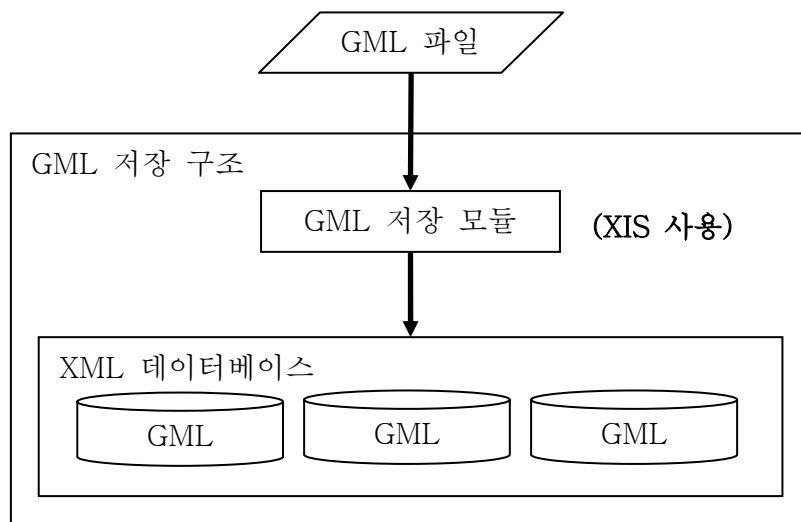


그림 3.5 GML 저장 구조
 Fig. 3.5 The GML storage structure

그림 3.6은 이러한 XML 데이터베이스의 저장 구조를 나타내고 있다. 기본적으로 Partition, GML Store, Directory 형태의 계층적 구조로 데이터가 저장되

며, 각각을 설명하면 다음과 같다.

- Partition: 물리적인 저장 위치와 운영체제의 저장 위치(directory)가 연결되어 있는 것으로 GML Store 그룹이 저장되는 장소이다. 주로 RAID Disk Array의 Mount Directory가 사용된다. Partition의 이름은 서버상에서 중복될 수 없다.
- GML Store: 계층적 구조로 저장된 데이터의 최상위 Root 역할을 수행한다. GML Store의 이름은 XIS 서버 상에서 중복될 수 없다. 생성될 GML Store의 개수는 응용프로그램이 처리해야 하는 업무와 데이터 양과 그 설계에 의존적이다.
- Directory: Directory에는 Child Directory 또는 파일이 저장된다.

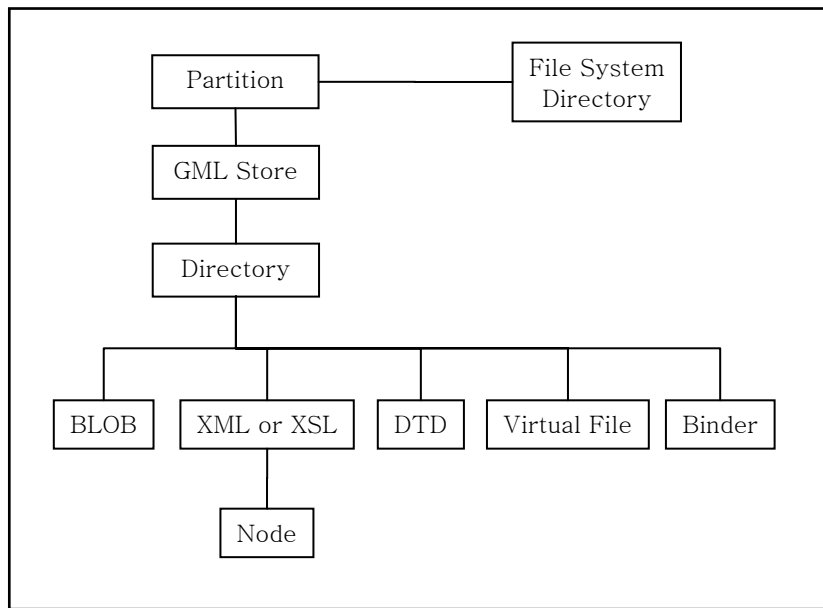


그림 3.6 XML 데이터베이스 저장 구조
Fig. 3.6 The structure of the XML database storage

GML 저장 모듈에서는 DTD나 스키마를 GML 데이터와 비교하여 문법적으로나 스키마 구조적으로 오류가 있는지는 확인하는 기능을 한다. 그림 3.7은 저장모듈을 순서도로 나타낸 것이다.

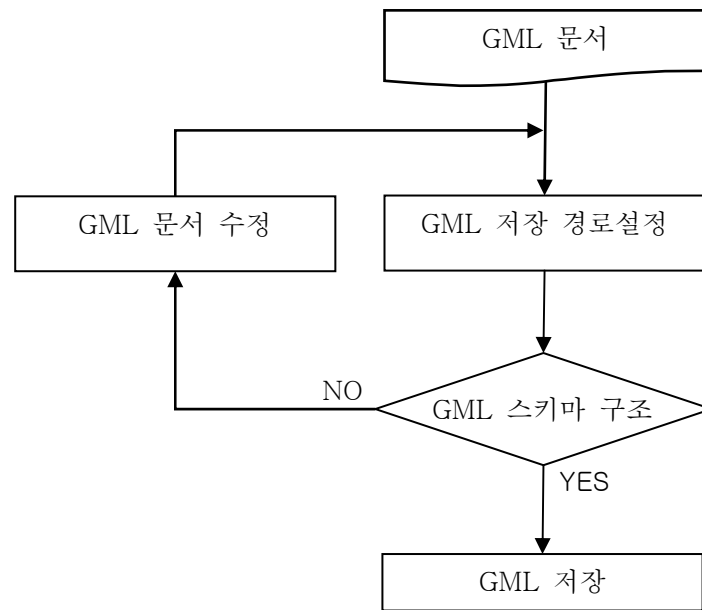


그림 3.7 XML 데이터베이스 저장 모듈 순서도
 Fig. 3.7 The flowchart of the XML database storage module

3.3 GML 전자해도의 사용자 인터페이스

XML 데이터베이스와 사용자 인터페이스를 연결하는 과정을 그림으로 살펴보면 그림 3.8과 같다.

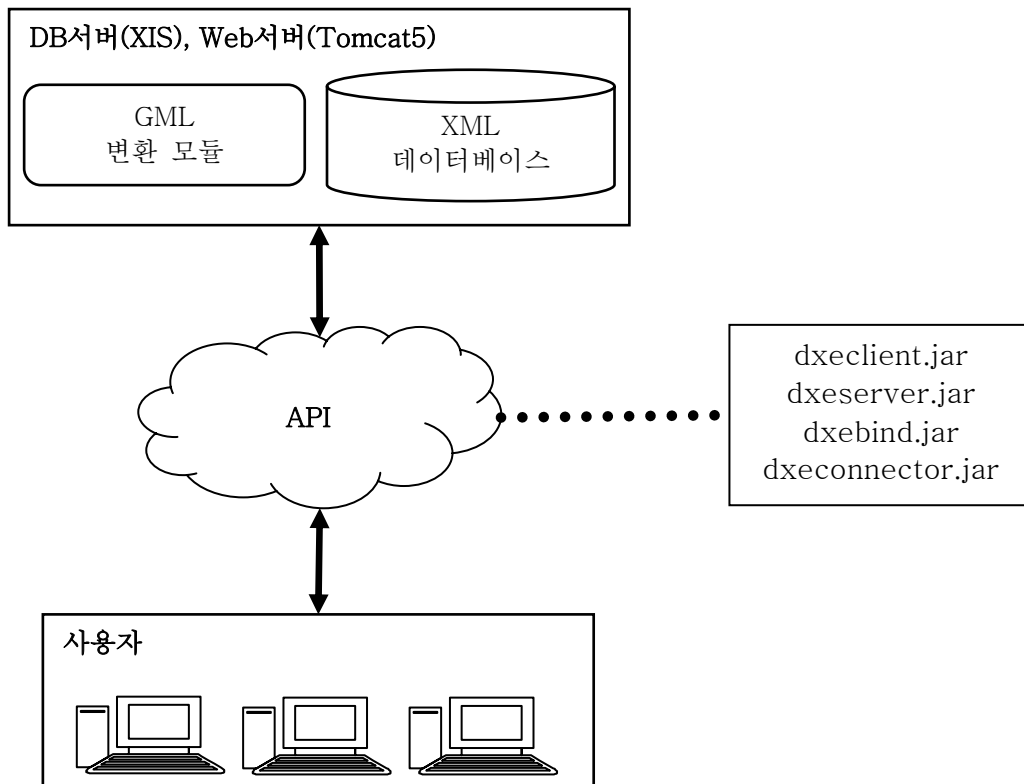


그림 3.8 XML 데이터베이스 연결
Fig. 3.8 Connection of the XML database

사용자는 인터넷을 이용해서 웹으로 서버에 접근하게 된다. 서버에 접근하는 과정에서 XML 데이터베이스를 응용프로그램과 연결시켜 사용하기 위해서는 다음과 같은 4개의 파일을 사용한다.

- Client Session:dxeclient.jar
- Local Session:dxeserver.jar
- Java Binding:dxebind.jar
- J2EE connector:dxeconnector.jar

네 가지 파일 중에서 dxeclient.jar와 dxeserver.jar 2개의 파일은 XML 데이

터베이스와 사용자를 연결해주는 역할을 한다. dxeclient.jar 파일에는 XML 데이터베이스 사용자 API가 포함되어 있다. 사용자 모듈은 쿼리, 업데이트 등의 내용을 서버에 송신하고 그 결과를 수신하는 JAVA 모듈로 구성되어 있다. dxeserver.jar 파일은 서버 API가 포함되어 있고, 트랜잭션 관리 및 사용자의 송신 데이터를 처리하는 작업을 수행한다.

다음으로 XML 데이터베이스 쿼리 처리과정을 살펴본다. 정보를 검색하는 방법에는 크게 구조기반 검색과 내용기반 검색이 있다[14]. 먼저, 사용자 인터페이스를 통하여 전자해도에 대한 검색 질의가 입력되면, 이를 웹 서버에 전달한다. 웹 서버는 전달된 질의를 XQuery와 XPath를 이용하여 XML 데이터베이스에 요청하여 처리하는 과정을 그림 3.9는 보여주고 있다[15].

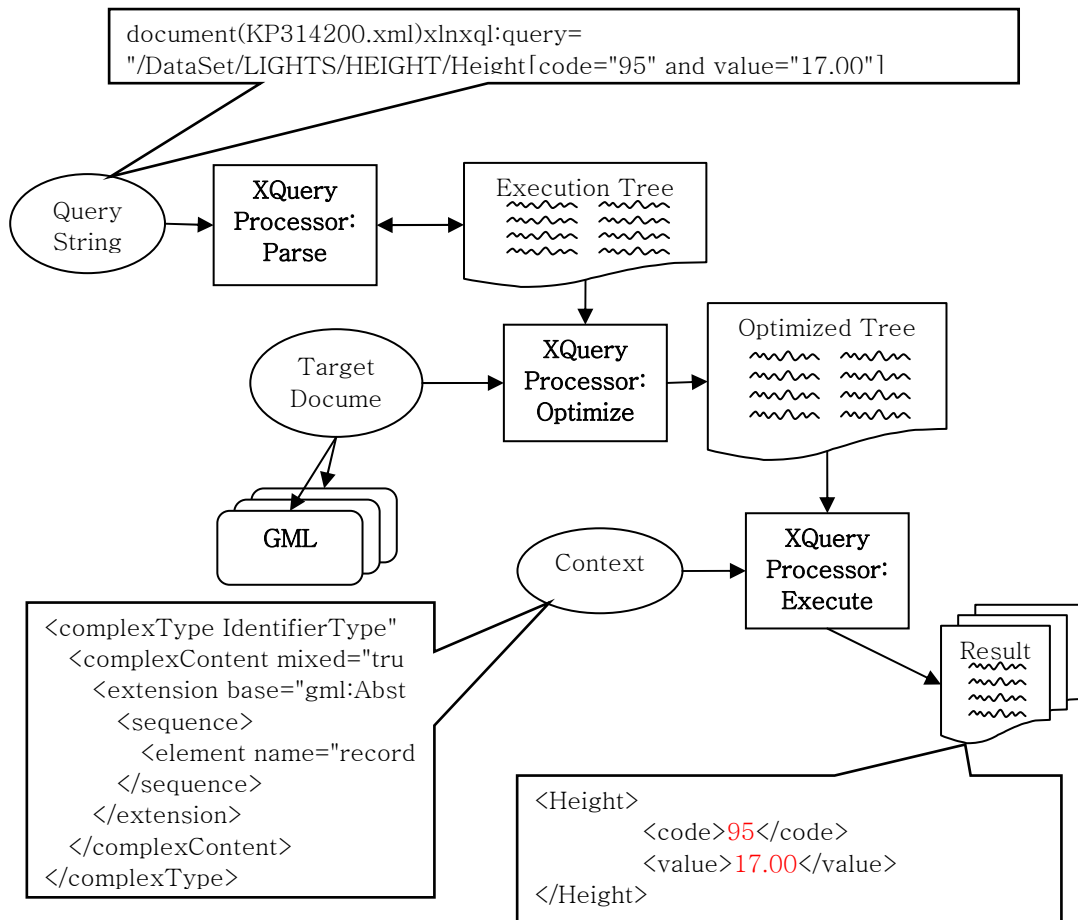


그림 3.9 GML 질의 처리
Fig. 3.9 GML query processing

처음 쿼리가 들어오면 실행이 가능한 트리형태로 문장을 변환하고, 변환된 문장은 찾고자 하는 문서에 맞게 최적화시켜서 트리형태로 만든다. 마지막으로 최적화된 트리를 실행하고 결과를 XML 구조에 맞게 출력한다.

그림 3.10은 웹에서 GML 코드를 보여주는 처리과정을 그림으로 표현한 것이며 사용자가 XML 데이터베이스에 연결하고 질의를 해서 원하는 결과를 가

저오는 과정을 살펴보면 다음 순서와 같다.

- ① 사용자는 웹에서 오브젝트와 엘리먼트를 선택한다.
- ② Query1.jsp는 오브젝트와 엘리먼트의 값을 DBelement.java에 넘겨준다.
- ③ DBelement.java는 넘겨받은 값을 XML 질의어(XPath)로 변환하여 XML 데이터베이스에 질의를 한다.
- ④ DBCgml.java는 검색된 결과를 데이터베이스에서 받아서 GML형식으로 변환한다.
- ⑤ gmlview.jsp는 GML형식의 검색 결과를 웹에서 볼 수 있도록 변환한다.
- ⑥ 웹 상에서 GML 코드 표시한다.

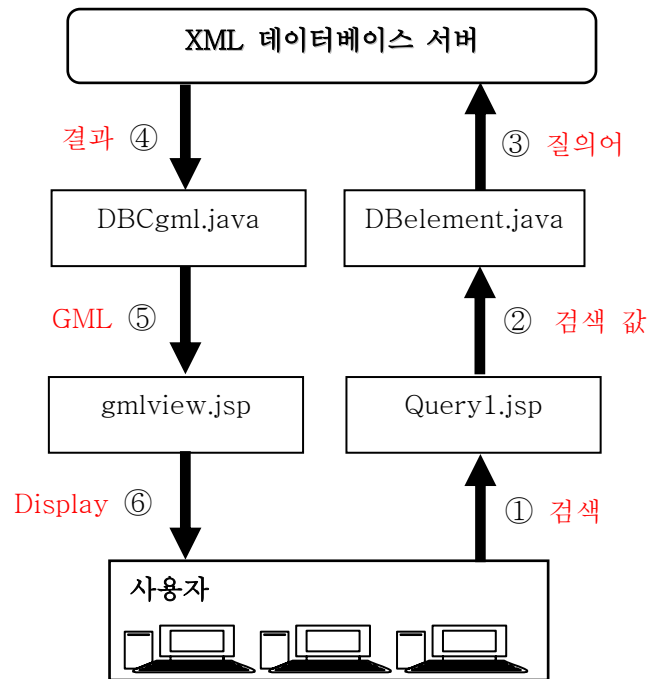


그림 3.10 GML 코드 처리
Fig. 3.10 GML code porcess

위의 순서로 진행되는 과정에서 사용되는 파일의 기능을 설명하면 다음과 같다.

- Query1.jsp: 웹에서 사용자가 직접 오브젝트와 엘리먼트를 선택할 수 있도록 SELECT BOX를 제공하며 선택된 값을 DBelement.java로 넘겨주는 기능을 한다.
- DBelement.java: Query1.jsp에서 선택 받은 값을 7개의 클래스파일 (com.exln.dxe, com.exln.dxe.client, com.exln.dxe.engine, com.exln.dxe.servext, com.exln.dxe.servlet, com.exln.dxe.filesystem, com.exln.dxe.dom)을 사용해서 XML 질의어로 변환하고 XML 데이터베

이스 서버에 질의하는 기능을 한다.

- DBCgml.java:XML 데이터베이스 서버에서 결과값을 넘겨받는 기능을 하며 결과값을 이용해서 최종 GML구조로 재구성하여서 gmlview.jsp로 넘겨준다.
- gmlview.jsp:DBCgml.java에서 넘겨받은 GML을 사용자가 볼 수 있는 웹 형태로 변환해서 보여준다.

제 4 장 시스템의 구현

본 논문에서 제안하는 전자해도 S-57를 GML로 변환하는 시스템과 변환된 GML 파일을 XML 데이터베이스에 저장하는 시스템, 그리고 저장된 데이터를 XQuery와 XPath를 이용해서 사용자가 원하는 GML 엘리먼트를 검색하는 시스템을 설명한다.

4.1 구현 환경

S-57을 GML로 변환하는 시스템과 전자해도 데이터를 저장하고 저장된 데이터를 사용자가 검색하는 과정에서의 구현 환경을 다음 표 4.1에서 표현하고 있다.

표 4.1 구현 환경

Table 4.1 Implementation environments

구 분	구 성	
사용자	웹 브라우저	Internet Explorer 6.0
	운영체제	Microsoft Windows XP
서 버	CPU	Intel PentiumIV 1.7 GHz
	Memory	768MB
	운영체제	Microsoft Windows XP Professional SP 2
	웹 서버	Tomcat5
	XML 데이터베이스 서버	eXcelon (XIS) Version 3.1 SP3
	웹 개발 언어	JSP
	GML 변환	JAVA NetBeans 5.0

4.2 구현된 시스템

본 절에서는 구현된 시스템의 전체 구성 및 세부적으로 구현된 기능을 설명한다. 그림 4.1은 전체 시스템 구성도를 보여주고 있다. 사용자 질의의 검색결과는 인터넷을 통해서 XML 데이터베이스에 접근하여 검색된 결과를 사용자에게 보여준다.

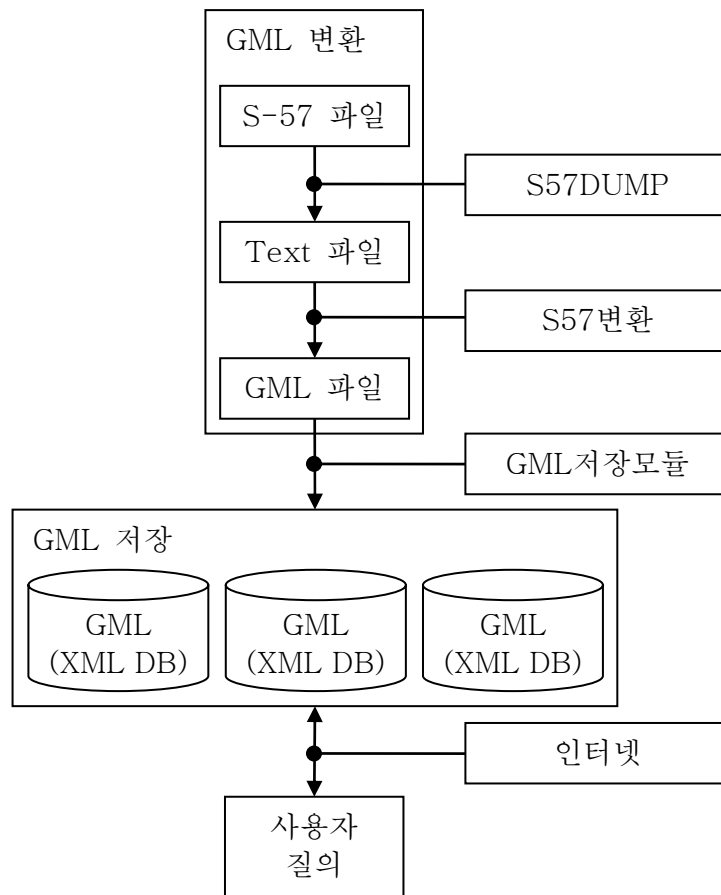


그림 4.1 구현된 시스템 흐름도

Fig. 4.1 The flowchart of the implemented system

(2) 전자해도 데이터베이스

GML 파일을 XML 데이터베이스에 저장하고, 저장된 데이터에 대하여 XPath를 이용하여 질의한다. 저장하는 과정과 저장 모듈의 역할은 앞의 3.2절에서 설명하였다.

그림 4.3에서는 XML 데이터베이스의 구조를 보여준다. 3.2절에서 설명한 Partition, GML Store, Directory 형태의 계층적 구조를 보여준다.

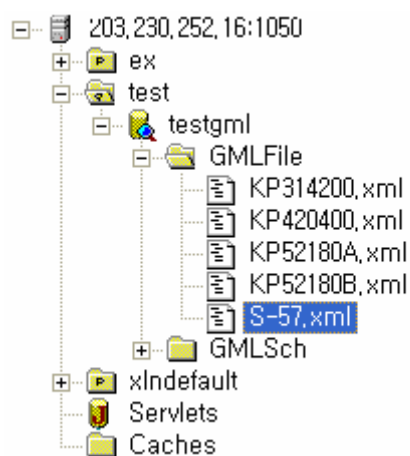


그림 4.3 XML 데이터베이스의 구조
Fig. 4.3 The Structure of the XML database

(3) 사용자 질의

XML 데이터베이스 저장된 데이터를 사용자 요구에 따라서 원하는 데이터를 보여주는 인터페이스를 구현하였다.

본 논문에서 예제로 사용한 S-57 실험 데이터는 대한민국 주변 환경을 표현한 전자해도 중에서 KP314200.000, KP420400.000, KP52180A.000, KP52180B.000 4개의 파일을 이용한다. 각각의 데이터를 변환과정에서 데이터

의 크기를 살펴보면 다음 표 4.2와 같다.

표 4.2 데이터의 크기
Table 4.2 The size of data

데이터 이름	S-57 데이터 크기	텍스트 데이터 크기	GML 데이터 크기
KP314200.000	696KB	2,010KB	5,094KB
KP420400.000	1,448KB	3,902KB	9,827KB
KP52180A.000	135KB	436KB	1,037KB
KP52180B.000	79KB	265KB	640KB

표를 보면 데이터의 크기가 점점 커지는 것을 볼 수 있는데, S-57의 2진 데이터에서 텍스트형태로 바꾸면 크기가 조금 증가하지만 다시 텍스트를 GML형태로 변환하는 과정에서 각각의 엘리먼트의 데이터 이름을 우리가 쉽게 볼 수 있는 형태로 변환하면서 데이터의 크기가 커지는 것이다. 예를 들면 FRID는 FeatureRecordIdentifier 형식의 이름을 가지고 있다.

GML 데이터를 XML 데이터베이스에 저장한 후에 사용자가 원하는 객체를 선택하고 객체 정보를 GML 문서로 생성한다. 그림 4.4는 검색하는 화면이다.

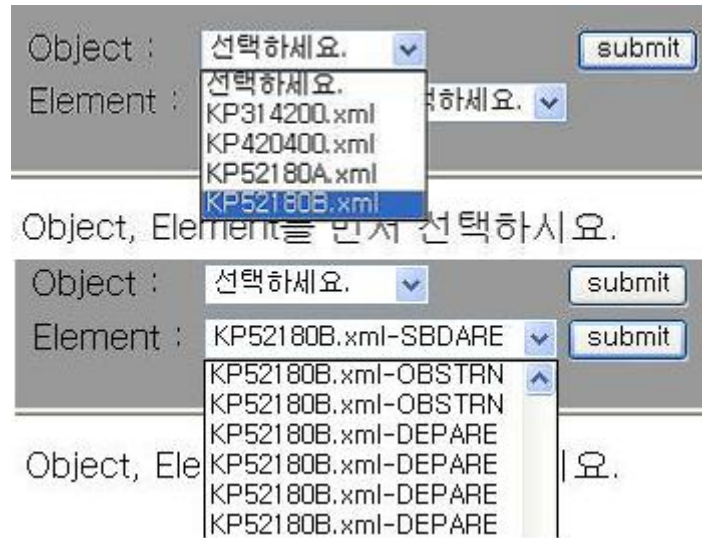
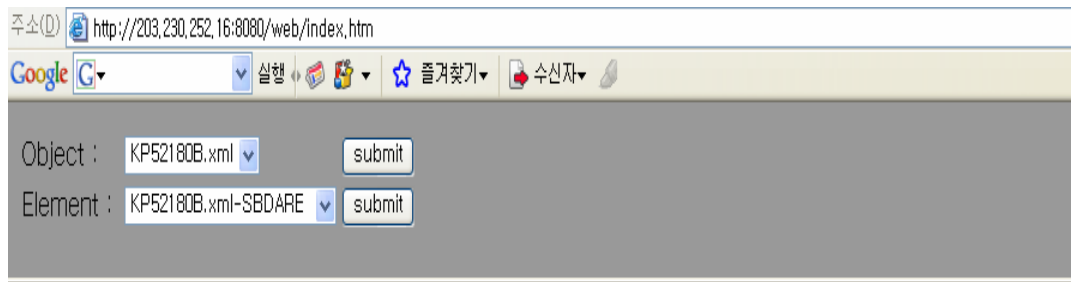


그림 4.4 질의 화면
Fig. 4.4 Query screen

KP52180B.XML-SBDARE 를 선택할 경우 XPath 표현식은 다음과 같다.

`xmlns:query="/DataSet/SBDARE"`

표현식을 이용해서 GML 결과를 그림 4.5 에서 보여준다.



```

<?xml version="1.0" encoding="UTF-8" ?>
- <xmlns:query xmlns:query="/DataSet/SBDARE" xmlns:count-requested="46" xmlns:count="46" xmlns:type="node_list"
  xmlns:xlnxql="http://www.exceloncorp.com/DXE/namespaces/query" xmlns="http://www.ukho.gov.uk/schema/S57"
  xmlns:gml="http://www.opengis.net/gml" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/
  instance">
- <SBDARE>
- <gml:metaDataProperty>
  - <FeatureRecordIdentifier>
    <group>2</group>
    <objectLevel>121</objectLevel>
    <recordVersion>1</recordVersion>
  </FeatureRecordIdentifier>
  </gml:metaDataProperty>
- <gml:metaDataProperty>
  - <FeatureObjectIdentifier>
    <agency>280</agency>
    <featureIdentificationNumber>58804</featureIdentificationNumber>
  </FeatureObjectIdentifier>
  </gml:metaDataProperty>
  </SBDARE>
  </query>
  </node_list>
  </instance>
  </xml>

```

그림 4.5 GML 결과 화면
 Fig. 4.5 GML result screen

제 5 장 결 론

기존의 S-57 형식의 전자해도를 더 효율적으로 활용하기 위해서 OGC에서 표준으로 제안하는 지리정보를 저장하고 활용할 수 있는 GML사용하였다. GML은 XML의 구조를 따르기 때문에 사용이 간편하고 재사용성 및 확장성이 뛰어나다는 장점이 있어 많은 분야에 적용할 수 있다.

본 논문에서는 S-57 전자해도를 특징 객체와 공간 객체로 분류하여서 JAVA언어 그리고 GaldosSystem사의 6개의 스키마를 적용하여 GML로 변환하는 시스템을 구현하였다. 그러나 GML은 기본 데이터 형식이 트리 형태이다. 기존에 사용하던 관계형 데이터베이스는 그래프 형태를 가지고 있으므로 사용할 경우 속도나 GML을 다른 형태로 변형하여 데이터베이스에 저장하는 어려움이 있었다. 그 문제점을 해결하고자 XML 데이터베이스를 설계하고 인터넷을 통하여 데이터베이스에 손쉽게 접근할 수 있을 뿐만 아니라 다른 정보시스템과의 정보교환도 용이하게 할 수 있는 사용자 인터페이스를 구축하였다. 사용자 인터페이스는 JSP언어를 이용하여서 개발하였고, 웹 서버는 Tomcat5를 사용하였다.

향후에는, 효과적이고 기능이 추가된 사용자 인터페이스를 개발하고, 데이터의 보안성을 향상 시킬 수 있는 방안과 다른 지리정보시스템과의 연계방안을 연구할 필요가 있다.

참고문헌

- [1] 이성대, 강형석, 박휴찬, “전자 해도용 XML 스키마의 정의 및 변환”, 한국해양정보통신학회논문지, 제8권, 제1호, pp. 200-212, 2004.
- [2] 감승철, 이성대, 곽용원, 박휴찬 “GML과 SVG를 사용한 웹 기반 전자해도 시스템의 개발”, 한국해양대학교 부설 산업기술연구소 연구논문집, 제23집, pp. 83-88, 2006.
- [3] Galdos Incorporated, S-57 Schema and Related Tools Manual, S-57/GML Project, 2004.
- [4] ESRI Incorporated, ArcGIS S-57 Converter 9.0 beta Documentation, 2004.
- [5] CNAVI Incorporated, CnENC Ver. 1.0 Document, 2004.
- [6] IHO, IHO Transfer Standard for Digital Hydrographic Data Version 3.1, Special Publication No. 57 (S-57), <http://www.iho.shom.fr>, 2000.
- [7] OpenGIS, Geography Markup Language (GML) Version 3.0, <http://www.opengeospatial.org>, 2003.
- [8] D. S. Burggraf, S-57 Schema and Related Tools, http://www.ukho.gov.uk/b2b_gml_home.asp, 2004.
- [9] W3C, Extensible Markup Language (XML) 1.1, <http://www.w3.org/TR/xml11>, 2006.
- [10] W3C, XQuery: A Query Language for XML. W3C Working Draft, May <http://www.w3.org/TR/2003/WD-xquery-20030502/>, 2003.
- [11] W3C, XML Path Language(Xpath) Version 1.0,

<http://www.w3.org/TR/xpath>, 1999.

- [12] 김서영, 이기훈, 황규영, “효율적인 XML 질의 처리를 위한 XQuery 질의의 정규화”, 정보과학회논문지:컴퓨팅의 실제, 제 10 권, 제 5 호 (2004.10)
- [13] 전재명, 정연돈, 김명호, 이윤준, “XPath 표현식의 필터링을 통한 XML 접근 제어 기법”, 정보과학회논문지:데이터베이스, 제 32 권, 제 2 호 (2005.4)
- [14] 권훈, 김정희, 곽호영, “저장 공간과 검색 효율을 위한 XML 문서의 RDB 스키마 모델”, 한국콘텐츠학회논문지, 제6권, 제4호, pp. 19-28, 2006. 4.
- [15] eXcelon Incorporated, eXtensible Information Server Developer Guide Documentation, 2003.
- [16] 이성대, 곽용원, 박휴찬, “객체관계형 데이터베이스에 기반한 XML 문서 저장 및 검색 시스템의 설계 및 구현”, 한국해양정보통신학회논문지, 제 7권, 제2호, pp.183-193, 2003.