방향성 스키마 그래프 매핑 규칙을 이용한 GML 문서의 공간 데이터베이스 변환 기법

The Conversion Scheme of GML Document into Spatial Database using the Directed Schema Graph Mapping Rules

정원일*, 박순영**, 배해영***
Warn-ill Chung, Soon-Young Park, Hae-Young Bae

요 약 XML을 기반으로 지리 정보의 저장 및 전송을 위한 인코딩 표준으로 제안된 GML은 점차 그 활용도가 증가하고 있다. 이에 웹 환경에서 지리 정보의 상호 운용성을 제공하기 위해 다양한 모델링, 저장 및 질의에 관한 연구가 수행되어 왔으며, 특히 구조적인 특성을 갖는 GML 문서를 효율적으로 저장하는 연구는 필수적이다. 따라서, 본 논문에서는 GML 스키마를 기반으로 작성된 GML 문서를 공간 데이터베이스로 저장하기 위한 GML 문서 변환 기법을 제안한다. 제안 기법에서는 GML 스키마를 방향성을 갖는 그래프 구조로 변환하고 이를 기반으로 공간 스키마로의 매핑에 대해 기술한다. 그리고 GML 문서를 공간 데이터베이스로 변환할 때 발생하는 의미적 손실을 보상하기 위해 GML 스키마에 존재하는 제약 조건을 변환된 공간 스키마에서 유지하기 위한 규칙을 제안한다. 또한, 제안 기법은 공간 데이터베이스를 GML 문서의 저장소로 활용할 수 있도록 함으로써 이질적인 지리 정보의 상호 운용성 제공뿐 아니라 대량의 GML 문서에 대해 효과적인 저장과 관리가 가능하게 한다.

Abstract GML (Geography Markup Language) has become the widely adopted standard for transport and storage of geographic information. So, various researches such as modeling, storage, query, and etc have been studied to provide the interoperability of geographic information in web environments. Especially, there are increased needs to store semi-structured data such as GML documents efficiently. Therefore, in this paper, we design and implement a GML repository to store GML documents on the basis of GML schema using spatial database system. GML Schema is converted into directed GML schema graph and the schema mapping technique from directed schema graph to spatial schema is presented. Also, we define the conversion rules on spatial schema to preserve the constraints of GML schema. GML repository using spatial database system is useful to provide the interoperability of geographic information and to store and manage enormous GML documents.

주요어 : GML, GML 변환, 공간 데이터베이스
KeyWords : GML, GML 변환, Spatial Database

1. 서 론

XML(eXtensible Markup Language)[1]은 차세대 웹 문서의 표준으로 정착되어 다양한 분야의 문서들이 XML을 기반으로 작성되고 있다. 특히, 지리 정보의 교환 표준으로 XML을 이용하는 연구가 활발히 진

* 한국전자통신연구원 선임연구원 wnchung@etri.re.kr
** 인하대 컴퓨터정보공학과 박사과정 sunny@dblab.inha.ac.kr
*** 인하대 컴퓨터공학부 교수 hybae@dblab.inha.ac.kr

행되고 있다[2]. 이러한 연구의 결과로 OGC(OpenGIS Consortium)에서는 XML을 기반으로 지리 정보의 저장 및 전송을 위한 인코딩 표준으로 GML(Geography Markup Language)[3][4][5] 제안하였다. 이러한 GML은 각 지리정보시스템의 개별적인 지리 정보 구축과 지리 정보 변환 및 공유 분야에 활용될 수 있으며, 이에 따라 GML을 기반으로 웹 환경에서 지리 정보의 통합, 시각화, 질의 등을 수행하기 위한 연구들과 함께 대량의 GML 문서들을 효율적으로 저장하고 관리하는 GML 저장소에 대한 필요성이 증가하고 있다[6][7][8][9][10]

XML과 같이 반 구조적 데이터의 특성을 갖는 문서를 위한 저장소는 텍스트 파일 시스템을 이용하는 방법, 전용 시스템(Special Purpose System)을 구축하는 방법[11][12][13], 그리고 기존의 데이터베이스 시스템을 이용하는 방법[14][15][16][17] 등으로 구현될 수 있다. 그러나 아직까지 GML의 공간 데이터를 지원하는 전용 시스템은 보고되지 않고 있다. 또한, 기존의 데이터베이스 시스템을 이용한 경우에도 관계형 데이터베이스나 객체지향 데이터베이스를 이용하여 XML 문서를 관리하는 방법에 대한 연구는 활성화되어 있으나, 이러한 데이터베이스 관리 시스템들은 공간 데이터가 아닌 비공간 데이터 처리에 최적화되어 있으므로 GML 문서를 관리하기에 기존의 데이터베이스 시스템들은 비효율적이다. 따라서 공간 데이터에 대해 풍부한 공간 연산 및 최적화 기능 등을 지원하는 공간 데이터베이스 시스템을 이용한 GML 문서의 관리가 필요하다. 그러나 GML 문서를 데이터베이스에 저장하는 기존의 연구[22][23]에서는 GML 문서가 가지는 계층이나 순서 등의 구조적 특성을 저장시에 반영하지 못함으로써 GML 문서를 출판할 때 GML 문서의 구조적 특성을 상실하게 되는 문제점을 갖고 있다. [22]에서는 GML 변환 모듈을 이용하여 분석된 GML 문서의 기하 객체를 공간 데이터베이스 시스템인 ZEUS의 공간 데이터로 변환한다고 기술하고 있으나, 구체적인 방법론에 대한 설명을 명시하고 있지 않다. [23]에서는 DTD 또는 스키마로 규격화된 GML 문서를 공간 데이터베이스에 저장하기 위해 GML 문서를 공간 레이어에 매핑시키고 있다. 그러나 [23]에서는 GML 문서의 구조적 정보를 유지하지 않기 때문에 사용자 질의에 대한 GML 결과를 출력할 때 원 문서를 복원할 수 없다. 따라서 본 논문에서는 GML 문서의 구조 정보를 나타내는 스키마(Schema)를 기반으로 표현된 GML 문서를 공간 데이터베이스 시스템에 저장하는 방법을 제안한다.

제안 기법에서는 Beech[21]에서 제시한 XML 모델을 확장하여 GML[3][4][5] 기반의 모델링을 지원한다. 제안 모델링 기법에서 GML 스키마는 방향성을 갖는 그래프 구조로 변환하며, 이 그래프를 기반으로 GML 스키마를 공간 스키마로 매핑하는 규칙에 대해 설명하고, GML 문서를 이용하여 공간 릴레이션을 구성하기 위한 방법에 대해 설명한다. 그리고 GML 스키마에 존재하는 도메인과 키 등에 대한 다양한 제약 사항들을 유지하기 위해 공간 데이터베이스에서 제공하는 도메인 제약 조건과 무결성 제약 조건을 이용하는 방법에 대해 설명한다. GML 문서의 기하 객체 데이터는 OGC 기하 객체 모델[18]을 기준으로 공간 데이터베이스의 공간 객체로 변환된다. 이러한 기법을 적용한 구현된 GML 저장소는 지리 정보의 상호 운용성을 제공하며 대용량의 GML 문서를 저장하고 관리함에 있어 공간 데이터베이스 시스템을 이용함으로써 효과적으로 응용될 수 있다.

본 논문의 구성은 다음과 같다. 먼저 제2장에서는 관련 연구를 수행하고, 제3장에서는 방향성을 갖는 그래프 구조의 GML 문서에 대한 모델링에 대해 설명한다. 제4장에서는 GML 문서를 공간 데이터베이스로 변환하는 기법에 대해 기술하고, 제5장에서는 제안 기법을 이용한 GML 저장소를 설계하고 구현한다. 마지막으로 제6장에서는 결론에 대해 설명한다.

2. 관련 연구

본 절에서는 XML의 지리 정보 저장 및 전송을 위한 인코딩 표준인 GML을 위한 저장소를 생성하기 위한 관련 연구에 대해 설명한다.

2.1 XML 저장소

XML 문서를 저장하기 위한 시스템으로는 파일 시스템, 전용 저장시스템, 데이터베이스 관리시스템 등

이 있다. 또한 데이터베이스 시스템의 경우에는 관계형 데이터베이스관리시스템, 객체관계형 데이터베이스시스템 등이 있다[11][12][13][14][15][16][17].

파일 시스템을 이용한 XML 저장 시스템은 저장소의 구축은 간단하지만 질의 수행을 위해 해당 XML 문서를 매번 트리 구조로 파싱하여야 한다는 단점이 있다.

전용 저장 시스템[11][12]은 XML 구조에 기반한 데이터 모델을 제공함으로써 XML 문서가 가지는 구조적 특성을 잘 지원할 수 있으나, 대용량 데이터의 관리 및 다중 사용자 지원 등에 있어 아직 성숙 단계에 이르지 못한 단점이 있다[15][16].

관계형 데이터베이스 관리시스템[14][15][16][17]을 이용하여 XML 저장시스템을 구축하는 경우는 XML 문서의 구조를 관계형 데이터베이스의 이차원 테이블에 저장하는 방법으로, 기존의 관계형 데이터베이스의 장점을 수용할 수 있으나 XML 문서의 구조적 특성을 이차원의 평면 구조로의 매핑이 용이하지 않으며 질의 처리시에 다량의 조인 연산을 발생한다는 단점이 있다. 객체지향형 데이터베이스 관리시스템[13]을 이용하여 XML 문서를 저장할 경우에는 XML 문서의 그래프 구조를 객체지향 개념에 적용하여 표현이 가능하므로 관계형 데이터베이스 관리시스템보다 보다 효과적인 XML 문서의 관리가 가능하다. 그러나 Christophides[10]에서는 입력되는 XML 문서의 구조에 따른 저장과 검색의 효율이 떨어질 수 있는 단점을 지니고 있다.

본 논문에서는 이러한 XML 저장소의 구현을 위한 근간이 되는 XML 모델링 기법을 적용 하여 GML 문서의 기하 객체를 지원하기 위한 확장 GML 모델링 기법을 제안한다.

2.2 GML 저장소

공간 데이터베이스 관리시스템[19][22][23]은 점, 선, 면 등의 다양한 공간 데이터 타입 및 교차, 포함, 내포 등의 풍부한 공간 연산들을 지원하므로 공간 데이터베이스 시스템을 이용하여 지리정보를 포함하고 있는 GML을 저장하고 관리할 경우, 공간 데이터베이스 시스템에서 제공하는 다양한 공간 기능들을 활용할 수 있으므로, 관계형 데이터베이스 시스템이나 객체지향형 데이터베이스시스템에서는 용이하지 않은 공간 데이터에 대한 효율적인 처리가 가능하다.

[22]에서는 GML 문서들을 사용자들이 쉽게 분석, 생성, 출력, 그리고 갱신할 수 있는 기능을 제공하며, 또한 기존 공간 데이터베이스 시스템의 지리 정보와 GML 문서 상호간의 변환 기능을 제공한다. 이 기법은 GML 명세에 따라 GML 문서를 공간 데이터베이스에 저장하거나 공간 데이터베이스에 저장된 지리 정보를 검색하는 방법을 기술하여 이질적인 지리 정보간의 상호 운용성을 제공하고 있다. [23]에서는 공간 데이터베이스에 저장된 지리 정보를 GML 문서로 출력하고, 또한 DTD나 XML 스키마 기반의 저장 기법을 통해 GML 문서를 공간 데이터베이스에 저장하는 방법 등 공간 데이터베이스와 GML 문서상의 지리정보 변환에 대해 기술하고 있다. 이러한 기법들은 GML 문서의 저장을 위해 공간 데이터베이스 시스템을 이용함으로써 공간 데이터베이스 시스템의 장점을 활용할 수 있으나, GML 문서를 공간 데이터베이스로 변환할 때 발생할 수 있는 의미적 손실에 대해 설명하고 있지 않다. 따라서 본 논문에서는 방향성 구조를 갖는 GML 문서의 의미를 유지할 수 있는 기법을 제안한다.

3. 방향성 그래프 구조를 갖는 GML 문서의 모델링

본 장에서는 방향성 그래프 구조를 갖는 GML[3][4][5]의 모델링에 대해 설명한다. GML 모델링은 4장에서 기술할 GML 문서의 공간 데이터베이스 변환 기법의 기반이 된다.

GML 모델링은 Beech[21]의 XML 모델링 방법을 확장하여 정의하며, 스키마의 매핑에 앞서 GML 문서 구조에서 GML 스키마를 표현하기 위해 언어 독립적인 정형화를 수행한다. GML 문서는 방향성을 갖는 그래프(Directed Graph)의 구조로 표현되며, $G=(V,E,A,R,O)$로 표기한다.

V는 그래프에서 엘리먼트(Element)와 값의 정점(Vertex)들로 구성된 집합으로, V는

$V = (V_{element} \cup V_{geometry} \cup V_{MO} \cup V_{int} \cup V_{string} \cup \Lambda)$ 는 로 형식화된다. $V_{element}$ 는 엘리먼트를 나타내는 정점이고, $V_{geometry}$ 는 기하 객체 타입을 표현하기 위한 정점이며, V_{MO} 는 이동 객체를 표현하는 정점이다. V_{int} 와 V_{string} 은 데이터 값을 포함하는 정점들이다. 각 GML 엘리먼트는 엘리먼트 정점 $v \in V_{element}$ 로 표현되며, 엘리먼트 정점의 타입은 엘리먼트이다. v는 유일하고(unique), 논리적인(logical) 그리고 불변의(immutable) 추상 식별자(abstract identifier)를 갖는다. 각 데이터 값은 값의 정점인 $v \in V_{type(v)}$ 로 표현되고, 문자나 숫자 등에 해당하는 값과 타입 속성을 갖는다. 기하 객체 엘리먼트 정점은 $v \in V_{geometry}$ 로 표현되고, 기하 객체 엘리먼트 정점의 타입은 좌표(coordinate) 엘리먼트, 기본(primitive) 기하 객체 엘리먼트, 그리고 집단(aggregate) 기하 객체 엘리먼트 등으로 분류한다. GML에서 좌표 엘리먼트는 coord, coordinates, pos 엘리먼트 등이 있으며, 기본 기하 객체 엘리먼트로는 Point, LineString, Polygon 등이 있고, 집단 기하 객체 엘리먼트에는 MultiPoint, MultiLineString, MultiPolygon, GeomCollection 등이 있다. 이동객체[20]는 이동점과 이동영역으로 분류되고, 또한 각 이동점과 이동영역에 대한 집단화된 데이터 타입이 존재한다. 이에 V_{MO} 에서는 이동점 엘리먼트(MOPoint), 이동영역 엘리먼트(MORegion), 다중 이동점 엘리먼트(MOMultiPoint), 다중 이동영역 엘리먼트(MOMultiRegion) 등 이동객체 엘리먼트에 대한 4개의 엘리먼트 타입을 정의한다. 그리고 정의된 각 타입에서는 GML 스키마에서 제공하는 엘리먼트들을 포함하게 되는 데, 공간 정보를 표현하기 위한 점(Point), 영역(Polygon), 다중점(MultiPoint), 다중영역(MultiPolygon) 등의 기하 객체 엘리먼트들과 시간 정보를 나타내기 위한 타임스탬프(timeStamp) 엘리먼트들을 갖는다. 기타 GML에서 제공된 스키마에서 참조되는 엘리먼트에 대한 설명은 생략한다.

엘리먼트 간선(Element Edges: E)은 부모 엘리먼트와 자식 엘리먼트의 연관성을 나타내는 방향성 간선으로, 자식 엘리먼트는 데이터 값을 나타내는 데이터 정점이나 하위 엘리먼트 정점이 될 수 있다. 각 간선 $e \in E$ 는 ($name \in T_{name}$, $parent \in (V_{element}, V_{geometry}, V_{MO})$, $child \in V$)로 표기된다. *name*은 엘리먼트 간선의 이름을 나타내고, *parent*는 간선의 부모 정점을 나타낸다.

속성 간선(Attribute Edge: A)은 엘리먼트와 속성의 관계를 나타내는 방향성이 있는 간선으로, 속성 간선은 엘리먼트 간선과 같이 *parent, value, name, type* 같은 속성을 갖는다. *name*은 속성 이름을 나타내고 *parent*는 속성을 갖는 엘리먼트이고 *value*는 속성 값을 나타낸다. 속성 간선은 ($name \in T_{name}$, $parent \in (V_{element}, V_{geometry}, V_{MO})$, $child \in V_{type(v)}$)로 표시한다.

참조 간선(Reference Edge: R)은 엘리먼트 사이의 참조 관계를 나타내는 방향성 간선으로, 속성 중 *IDREF, IDREFS,* foreign key, Xlink, URI와 같은 참조 타입으로 식별되는 경우 그래프 모델에서 참조 간선 집합 R에 속한다. *refedges*는 참조 정보를 제공하는 속성 또는 엘리먼트 간선에 대한 정보를 나타낸다. 그리고, 타입 속성은 간선이 참조 간선임을 나타내고 Xlink 또는 URI와 같은 참조 종류를 나타낸다. 참조 간선은 ($parent \in V_{element}$, $refedges \in P(E \cup A)$, $child \in V$ „)로 표기된다.

O는 부모 엘리먼트를 자식 엘리먼트로 연결하는, E, A 또는 R에서 간선들 사이의 지역적 순서를 나타낸다. O는 모든 간선들이 같은 간선 집합으로 분류되고, 동일한 부모를 공유하는 경우에 한해서 간선들 사이의 순서를 정의한다. O는 아래와 같이 형식화될 수 있고, *succ*는 순서상 e의 후행자를 의미한다.

$$O \equiv \{(e \in E \cup A \cup R, succ \in E \cup A \cup R)$$
$$| parent(e) = parent(succ) \wedge$$
$$(e \in E \wedge succ \in E \vee e \in A \wedge succ \in A \vee e \in R \wedge succ \in R)\}$$

참조 간선들의 경우 다중치 참조(multi-valued references)들의 각 참조들 간 순서는 참조 규칙에 의해 정의되고, *refedges*간의 순서는 다른 참조들간의 순서를 결정한다. 즉, 동일한 타입의 *refedges*를 갖는 참조들 간의 총 순서는 오직 하나이고, 모든 참조 간선들 간의 순서는 부분적임을 의미한다. 순서에서 선행 간선은 $pred(x) \in E \cup A \cup R \equiv e(o) \mid \exists o \in O : succ(o) = x$ 형식에 의해 결정된다.

4. GML 문서의 공간 데이터베이스 변환 기법

본 장에서는 이동 객체 정보를 포함하는 GML 문서를 공간 데이터베이스에 매핑하는 방법에 대해 설명한다.

4.1 GML 스키마 분석

본 절에서는 3장에서 설명한 GML 모델링을 실제 GML 문서에 적용하여 공간 스키마를 생성하는 방법에 대해 설명한다.

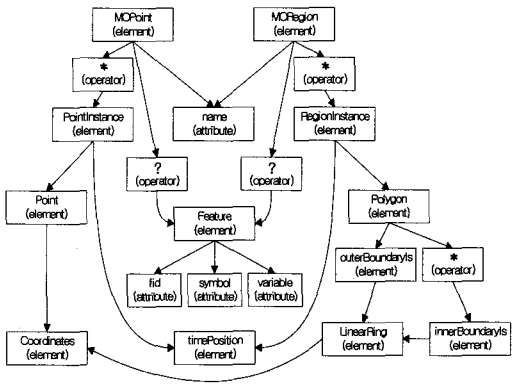
먼저 GML 스키마를 공간 스키마로 변환하기 위해 이동 객체 정보를 포함하는 GML 스키마의 예제를 <그림 1>과 같이 나타내고 있다.

```
01: <schema>
02:          <element          name="MovingObject"
type="MovingObjectType" />
03: <complexType name="MovingObjectType">
04:      <element ref="MOPointType" minOccurs="0"
maxOccurs="unbounded"/>
05:      <element ref="MORegionType" minOccurs="0"
maxOccurs="unbounded"/>
06: </complexType>
07: <element name="MOPoint" type="MOPointType" />
08: <complexType name="MOPointType">
09:                <attribute          name="Name"
type="string"use="required"/>
10:  <element ref="FeatureType"/>
11:   <element ref="PointInstanceType" minOccurs="1"
maxOccurs="unbounded"/>
12: </complexType>
13:          <element          name="PointInstance"
type="PointInstanceType" />
14: <complexType name="PointInstanceType">
15:      <element ref="gml:Point" minOccurs="1"
maxOccurs="1" />
16:    <element ref="gml:timePosition" minOccurs="1"
maxOccurs="1" />
17: </complexType>
18: <element name="MORegion" type="MORegionType"
/>
19: <complexType name="MORegionType">
20:      <attribute name="Name" type="string"
use="required"/>
21:  <element ref="FeatureType"/>
22:  <element ref="RegionInstanceType" minOccurs="1"
maxOccurs="unbounded"/>
23: </complexType>
24:          <element          name="RegionInstance"
type="MORegionInstanceType" />
25: <complexType name="RegionInstanceType">
26:      <element ref="gml:Polygon" minOccurs="1"
maxOccurs="1" />
27:      <element ref="gml:timePosition"minOccurs="1"
maxOccurs="1" />
28: </complexType>
29:   <element name="Feature" type="FeatureType"
substitutionGroup="gml:_Feature" />
30: <complexType name="FeatureType">
31:  <complexContent>
32:   <extension base="gml:AbstractFeatureType">
33:    <attribute name="FID" type="string"/>
34:    <attribute name="symbol" type="string"/>
35:    <attribute name="variable" type="NMTOKEN"
use="default" value="true">
36:     <simpleType>
37:      <restriction base="string">
38:       <enumeration value="true"/>
39:       <enumeration value="false"/>
40:      </restriction>
41:     </simpleType>
42:    </attribute>
43:   </extension>
44:  </complexContent>
45: </complexType>
46: </schema>
```

<그림 1> An Example of Moving Objects GML Schema

<그림 1>은 이동점과 이동영역으로 분류되는 이동 객체[20]에 대한 정보를 기반으로 작성된 GML 스키마이다. 이동객체를 나타내기 위한 MovingObject 엘리먼트(line 02)는 이동점과 이동영역을 표현하기 위한 MOPoint 엘리먼트(line 07)와 MORegion 엘리먼트(line 18)로 구성된다. MOPoint 엘리먼트는 이름 속성과 주제 속성을 위한 Feature 엘리먼트(lines 29-46), 그리고 이력 정보를 나타내기 위한 PointInstance 엘리먼트로 구성된다(lines 08-12). 또한 PointInstance 엘리먼트는 이동점의 이력 정보 표현을 위해 특정 시간과 그 시간에 대한 이동점의 좌표값을 나타내기 위한 timePosition 엘리먼트와 Point 엘리먼트로 구성된다(lines 14-17). MORegion 엘리먼트도 MOPoint 엘리먼트와 유사하게 이동 영역이 시간의 변화에 따른 그 영역값을 유지할 수 있는 구조를 취하고 있다(lines 19-23).

<그림 1>에서 표현된 GML 스키마를 이용하여 공간 스키마를 생성하기 위해서 <그림 2>에서는 이동점과 이동영역에 대해 방향성을 갖는 GML 스키마 그래프를 생성한다.



<그림 2> Directed GML Schema Graph

<그림 2>에서 각각의 노드는 GML 스키마에서 엘리먼트, 속성(attribute), 그리고 연산자(operator)를 나타낸다. "*" 연산자는 상위 엘리먼트에 대해 하위 엘리먼트가 0개 이상의 집합임을 표현하기 위해 사용된다. 또한, "?" 연산자는 하위 엘리먼트에 대한 표현 유무를 나타내는 연산자이다. Point 엘리먼트와 Polygon 엘리먼트는 GML에서 공간 데이터를 인코딩하기 위해 제공하는 기본 기하 객체 엘리먼트의 예로써, 추가로 LineString, LinearRing, Box 엘리먼트 등도 이에 속한다. 또한 이러한 기본 기하 객체 엘리먼트의 집단화된 형태로 GML에서는 집단(aggregate) 기하 객체 엘리먼트로 MultiPoint, MultiLineString, 그리고 MultiPolygon 엘리먼트에 대한 스키마를 제공하고 있다. 또한, coordinates 엘리먼트는 GML에서는 좌표를 나타내기 위해 사용되는 엘리먼트로, coord 엘리먼트와 pos 엘리먼트로도 표현이 가능하다[5]. <그림 2>의 방향성을 갖는 GML 스키마 그래프는 깊이 우선 탐색을 통해 순항되며, 공간 스키마를 생성하기 위한 규칙들은 다음과 같다.

GML 스키마 그래프로부터 공간 스키마를 생성하기 위해 먼저 최상위 엘리먼트들에 대한 릴레이션을 구성한다. GML 스키마는 DTD와 달리 모든 엘리먼트가 시작 노드가 될 수 있으나 진입 차수가 0인 엘리먼트에 대해서만 시작 노드로 선정하여 독립적인 릴레이션으로 구성한다. <그림 2>에서는 MOPoint 엘리먼트와 MORegion 엘리먼트가 시작 노드에 해당되어 독립된 릴레이션으로 구성된다. GML 스키마 그래프에서 진입 차수가 2 이상인 엘리먼트는 독립적인 릴레이션으로 구성한다. 단, 진입 차수가 1이거나 진출 차수가 0인 엘리먼트는 인라인시킨다. IDREF 속성은 정점 집합에 속하거나 NULL인 엘리먼트들중에서 하나의 엘리먼트를 참조하므로, IDREF 속성을 공간 릴레이션에 매핑하기 위해서는 인식자(identifier) 속성을 이용하여 참조하도록 정의한다. * 연산자 노드에 후행하는 자식 엘리먼트에 대해서는 새로운 릴레이션을 구성한다. 단, <그림 2>의 Polygon 엘리먼트와 같이 기하 객체 모델[18]에 정의된 기하 객체 유형에 후행하는 * 연산자 노드 아래의 엘리먼트들에 대해서는 상위 엘리먼트에 인라인시킨다. 또한 ? 연산자 노드를 후행하는 엘리먼트들은 선행 엘리먼트에 인라인할 때 NULL을 허용한다. <그림 1>의 name 속성(line 09)은 생략할 수 없는 제약을 가지는 속성으로 공간 릴레이션에 매핑할 때 NOT NULL 옵션을 이용하여 제약 조건을 매핑한다. 또한 <그림 1>의 variable 속성(lines

35-42)과 같이 도메인을 갖는 경우에는 공간 릴레이션을 정의할 때 CHECK 옵션을 이용하여 도메인 제약조건을 검사한다. 각 릴레이션은 인식자의 역할을 위해 ID 필드가 추가된다. 또한 상위 엘리먼트에 대해 독립적으로 유지되는 하위 엘리먼트 릴레이션은 부모 릴레이션과의 참조를 위해 상위 릴레이션의 ID 필드를 참조키로 정의한다. 상위 엘리먼트에 대한 하위 엘리먼트 간의 순서를 규정하기 위해서 하위 엘리먼트에 대한 순서(order) 필드를 상위 엘리먼트를 위해 생성된 공간 릴레이션에 추가한다. 이러한 순서 필드는 GML 문서의 재구성을 위해 사용된다.

4.2 스키마 매핑을 통한 공간 릴레이션 구성

본 절에서는 앞 절에서 GML 스키마 분석을 통해 기술한 내용을 실제 GML 문서에 적용하여 공간 릴레이션을 구성하는 방법에 대해 설명한다.

<그림 1>의 이동 객체 정보를 표현하기 위한 GML 스키마에 기반하여 생성된 GML 문서의 예는 다음과 같다.

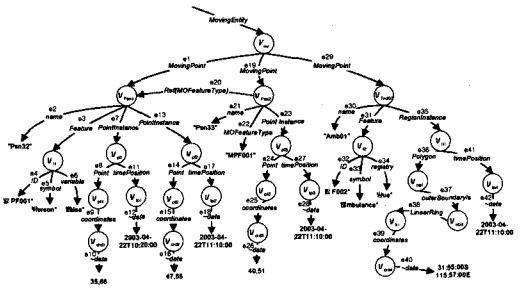
```
01: <MovingEntity>
02: <MovingPoint name="Psn01">
03:  <Feature FID="MPF001" symbol="person"
04: variable="false"/>
05:  <PointInstance>
06:  <Point>
07:   <coordinates>35,68</coordinates>
08:     </Point>
09:<timePosition>2003-04-22T10:20:00</timePosition>
10:    </PointInstance>
11:    <PointInstance>
12:      <Point>
13:        <coordinates>47,55</coordinates>
14:      </Point>
15: <timePosition>2003-04-22T11:10:00</timePosition>
16:    </PointInstance>
17:  </MovingPoint>
18:  <MovingPoint name="Psn02">
19:    <FeatureType::IDREF="MPF001"/>
20:    <PointInstance>
21:      <Point>
22:        <coordinates>40,51</coordinates>
23:      </Point>
24: <timePosition>2003-04-22T11:10:00</timePosition>
25:    </PointInstance>
26:  </MovingPoint>
27:  <MovingRegion name="Tnd01">
28:      <Feature FID="MRF001" symbol="tornado"
29:   variable="true"/>
30:    <RegionInstance>
31:      <Polygon>
32:        <outerBoundaryIs>
33:          <LinearRing>
34:            <coordinates>23,37    58,37    58,71
35:  23,71</coordinates>
36:          </LinearRing>
37:        </outerBoundaryIs>
38:      </Polygon>
39: <timePosition>2003-04-22T11:10:00</timePosition>
    </RegionInstance>
  </MovingRegion>
 <MovingEntity>
```

<그림 3> An Exmaple of Moving Objects GML Document

<그림 3>은 이동 객체들인 이동점(Moving Point)과 이동영역(Moving Region)을 실제 인스턴스들을 표현하기 위해, 이동점 객체로 두 명의 사람(person)과 이동영역 객체로 돌풍(tornado)을 예를 들고 있다. 이동점 객체인 MovingPoint는 객체의 주제 속성인 Feature 엘리먼트와 연속적으로 변화하는 시간에 따른 객체의 위치 정보를 나타내기 위한 PointInstance로 구성되어 있다. 이름이 'Psn01'인 이동점 객체는 Feature와 두 특정 시간에 객체가 위치했던 좌표 정보를 나타내고 있다(lines 02-16). 그리고 이름이 'Psn02'인 이동점 객체는 Feature 정보를 IDREF 속성을 통해 참조하고 있다(lines 17-25). 또한 이동 영역 객체인 MovingRegion은 Feature 엘리먼트와 특정 시간에 대한 이동 영역의 좌표 정보를 표현하기 위한 RegionInstance로 구성되어 있다(lines 26-38).

이동 객체에 대한 정보를 포함하는 GML 문서를 이용하여 공간 릴레이션을 구성하기에 앞서 GML 문서

를 방향성 그래프 형태의 GML 스키마 그래프를 생성하고, 3.2절의 릴레이션 생성 기준을 적용한다.



<그림 4> An Exmaple of Directed Graph generated by Moving Objects GML Document

<그림 4>는 <그림 3>에서 주어진 GML 문서를 방향성을 갖는 스키마 그래프로 변환한 그림이다. 위 예제의 구조에 대한 기본적인 설명은 3.1절의 GML 모델에서 언급하였고, 추가적으로~data, ~comment, ~PI 간선의 경우는 child가 데이터 정점이 되고, 그렇지 않으면 경우에 child는 간선의 하위 엘리먼트 정점이 된다. 엘리먼트 간선의 이름 중 ~data는 데이터를 포함하는 데이터 정점과 엘리먼트 정점을 연결하는 간선이고, ~comment, ~PI는 XML 문서에 대한 코멘트와 처리 명령들을 포함한 정점을 포함하는 간선이다. 3.2절의 스키마 분석을 적용하여 공간 릴레이션을 구성하는 예는 다음과 같다.

공간 릴레이션을 구성하기 위해 <그림 4>에서 가상의 루트인 MovingEntity 엘리먼트를 제외한 하위 엘리먼트는 MovingPoint와 MovingRegion 엘리먼트들로, 두 엘리먼트에 대한 릴레이션을 구성한다. GML에서의 기하 객체들은 공간 데이터베이스에서 하나의 타입으로 분류되며, [19]에서는 SPATIAL이라는 공간 객체 타입으로 변환된다. 이에 MovingPoint와 MovingRegion 엘리먼트에 대한 공간 릴레이션은 스키마 정의에서 동일한 형태를 취하게 된다. 이에 MovingPoint와 MovingRegion 엘리먼트에 대한 공간 릴레이션은 스키마 정의에서 동일한 형태를 취하지만 독립적인 릴레이션으로 구성된다. MovingPoint 엘리먼트에 대한 공간 스키마는 다음과 같다.

```
01:   TABLE MovingPoint =
02:     (  ID INT,
03:        order INT,
04:        name CHAR(20),
05:        FID INT,
06:        symbol INT,
07:        variable CHAR(10),
08:        object SPATIAL,
09:        timeposition DATE   )
```

<그림 5> Spatial Schema for Moving Points

MovingPoint 스키마는 <그림 4>에서 name 속성과 Feature 엘리먼트, 그리고 시간에 따라 위치가 변화되는 PointInstance 엘리먼트로 구성되어 있으며, 이를 공간 스키마로 변환할 때 <그림 5>와 같이 인식자를 위한 ID 필드(line 02), 엘리먼트 간의 순서 유지를 위한 order 필드(line 03)을 추가로 구성한다. SPATIAL 타입은 공간 객체 유형을 위한 데이터 타입으로 사용되며, DATE 타입은 GML의 timePosition 엘리먼트를 매핑하기 위해 사용된다. 그러나 <그림 4>의 Feature 엘리먼트는 진입 차수가 2이상인 엘리먼트로 독립적인 릴레이션으로 구성되어야 하며, 또한 <그림 4>의 PointInstance 엘리먼트의 인스턴스들에 대해 동일한 name 속성과 Feature 엘리먼트의 값들에 중복이 발생하므로 독립적인 릴레이션으로 구성한다. <그림 5>의 공간 스키마를 PointInstance 엘리먼트와 Feature 엘리먼트를 독립적인 릴레이션으로 구성한 예는 다음과 같다.

TABLE MovingPoint = (ID INT, order INT, name CHAR(20) NOT NULL, FID INT, primary key (ID))	TABLE PointInstance = (ID INT, PID INT, order INT, object SPATIAL, timeposition DATE, primary key (ID,PID), foreign key (PID) references MovingPoint)	TABLE Feature = (FID INT, symbol INT, variable CHAR(10), primary key(FID), check (variable in ('true','false')))
(a) MovingPoint Schema	(b) PointInstance Schema	(c) Feature Schema

<그림 6> Modified Spatial Schemas

<그림 6>의 수정된 공간 스키마는 <그림 5>의 공간 스키마에서 PointInstance 릴레이션과 Feature 릴레이션을 새로이 구성한 그림이다. <그림 6>(a)에서 ID 필드는 기본키(primary key)로 정의되고 있으며, name 필드는 <그림 1>의 GML 스키마(line 09)에서 'use="required"'라는 제약을 가지므로, 공간 스키마로 매핑할 때 NOT NULL을 설정한다. FID 필드는 Feature 정보를 참조하기 위해 정의한다. <그림 6>(b)에서는 ID 필드와 부모 릴레이션인 MovingPoint 릴레이션을 참조하기 위한 PID 필드를 이용하여 기본키를 설정한다. 또한 인스턴스 사이의 순서를 유지하기 위해 order 필드를 정의하고, Point와 timePosition 엘리먼트를 매핑하기 위한 SPATIAL, DATE를 사용하였다. <그림 6>(c)에서는 FID 필드가 기본키로 설정되고, 인스턴스의 분류를 위한 심볼값을 나타내는 symbol 필드, 그리고 symbol의 가변 여부를 표현하기 위한 variable 필드를 정의한다. 또한, variable 필드에 대해서는 그 속성값이 <그림 1>의 GML 스키마에서 정의한 도메인에 존재하는 지를 검사하기 위해 공간 SQL의 CHECK절을 사용한다. 또한 <그림 6>(b)의 기하 객체 유형인 object 필드에 대한 공간 데이터베이스에서의 필드 구조는 다음과 같다.

<그림 2>의 GML 스키마 그래프에서 x축과 y축에 대한 좌표 정보를 표현하는 Point 엘리먼트와 LinearRing을 이용하여 외곽 경계인 outerBoundaryIs와 외곽 경계 내부에 존재하는 2개의 innerBoundaryIs들을 표현하는 Polygon 엘리먼트는 지리정보를 표현하기 위해 사용된 공간 객체들이다. GML에서의 기하 객체들은 OGC 기하 객체 모델[18]을 지원하는 공간데이터베이스관리시스템인 GMS[19]에서 공간 데이터로 매핑된다. (그림 3)의 GML 문서에서 실체화된 Point 엘리먼트(lines 05-07, 11-13, 20-22)들과 Polygon 엘리먼트(lines 29-35)를 나타내고 있다.

<Point> <coordinates>35,68</coordinates> </Point>	'POINT (35 68)'
(a) Point Instance	(b) POINT Object

<그림 7> Conversion of Point Instance

<그림 7>(a)는 GML 문서에서의 Point 인스턴스를 나타내고, <그림 7>(b)에서 공간 데이터베이스의 공간 객체 모델로 변환한 구조를 표현하고 있다.

<Polygon> <outerBoundaryIs> <LinearRing> <coordinates> 23,37 58,37 58,71 23,71 </coordinates> </LinearRing> </outerBoundaryIs> </Polygon>	'POLYGON ((23 37, 58 37, 58 71, 23 71))'
(a) Polygon Instance	(b) POLYGON Object

<그림 8> Conversion of Polygon Instance

<그림 8>(a)는 GML 문서에서의 Polygon 인스턴스를 나타내고, <그림 8>(b)에서 공간 데이터베이스의 공간 객체 모델로 변환한 구조를 표현하고 있다. innerBoundaryIs 엘리먼트는 <그림 2>에서처럼 * 연산자 노드에 후행하고 있지만 독립적인 노드로 구성하지 않고 하나의 노드로 구성한다. 이는 Polygon 인스턴스가 다수의 innerBoundaryIs 엘리먼트로 내부 영역을 정의하더라도 기하 객체 모델에서는 하나의 Polygon 타입으로 매핑됨을 말한다.

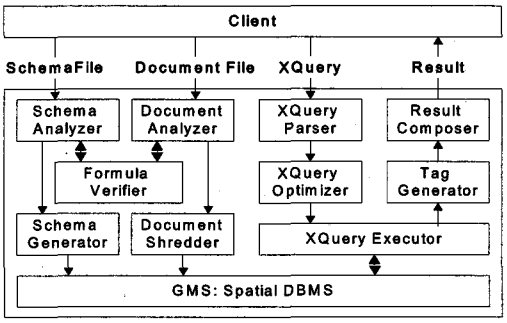
5. GML 저장소의 설계 및 구현

본 장에서는 공간 데이터베이스 시스템[19]을 이용하여 GML 문서 저장소를 설계하고 구현한다. 본 실험에서는 GML 문서를 저장하기 위한 알고리즘 및 시스템과 함께 GML 저장소에 대한 질의 처리 기능을 포함하고 있으나, 본 연구의 목적이 GML 문서의 저장에 있으므로 질의 처리를 위한 알고리즘은 설명에서 제외한다.

5.1 시스템 설계

본 연구에서 제안하는 시스템은 GML 스키마를 분석하여 공간 스키마를 생성하고, GML 스키마를 기반으

로 작성된 GML 문서를 이용하여 공간 릴레이션을 구성한다. 또한 공간 데이터베이스 시스템에 대한 질의 수행을 위해 XQuery 질의 처리가 가능하도록 하였다. <그림 9>는 공간 데이터베이스 시스템인 GMS[19]를 이용한 GML 저장소의 구조도를 나타내고 있다.



<그림 9> An Architecture of GML Repository

GML 스키마를 이용하여 공간 스키마를 생성하기 위해서 먼저 *Schema Analyzer*에서 입력된 GML 스키마를 분석하며, 이때 유효성 검사를 위해 *Formula Verifier*를 이용한다. 스키마 분석기의 결과를 바탕으로 *Schema Generator*에서 공간 스키마를 생성한다. 생성된 공간 스키마를 기반으로 공간 릴레이션을 구성하기 위해 입력된 GML 문서는 *Document Analyzer*에서 형식 검증기를 통해 유효성 검사를 수행하여 분석된 문서의 결과를 *Document Shredder*를 거침으로써 GML 문서의 공간 데이터가 공간 데이터베이스에 저장된다.

GML 저장소에 대한 질의는 XQuery를 기반으로 수행된다. XQuery 형태의 사용자 질의는 XQuery Parser에서 파싱되고, XQuery Optimizer에서 최적화 과정을 거쳐 생성된 실행 플랜을 XQuery Executor에게 전달된다. XQuery Executor는 질의 수행을 통한 결과를 반환하며, Tag Generator에서 입력 질의의 RETURN문에 명시된 질의 결과의 구조에 따라 엘리먼트를 생성하고 형식화한다. Result Composer는 XQuery Executor를 통해 유도된 질의 처리 결과와 Tag Generator의 결과를 이용하여 질의 결과를 완성한다.

5.2 GML 문서의 저장 알고리즘

본 절에서는 GML 문서를 공간 데이터베이스에 저장하기 위해 GML 스키마를 이용하여 공간 스키마를 구성하는 알고리즘에 대해 설명한다. 공간 스키마 생성을 위한 알고리즘은 아래 (알고리즘 1)과 같다.

(알고리즘 1: Spatial Schema Generation)

SchemaGeneration(G)
INPUT: Directed Schema Graph G
OUTPUT: Spatial Schema R

```
01:  for each vertex v
02:     initialize Tbl_Info and Attr_List;
03:     if (v.indegree = 0) or
04:        (v.indegree > 1 and v.outdegree<>0) then
05:        add v to Tbl_Info;
06:        add 'order' to Attr_List;
07:     end if
08:     if any child vertex of v is set with 'ID' then
09:        add 'ID' to Attr_List;
10:     else
11:        add new 'ID' to Attr_List;
12:     end if
13:     if v.type = geometry element then
14:        convert v to spatial data v'
15:        add v'to Attr_List;
16:     end if
17:     if v.indegree > 1 then
18:        add key of v.parent to Attr_List;
19:        add foreign key constraints to Tbl_Info;
20:     end if
21:     add attribute to Attr_List;
22:     if v.option = 'required' then
23:        add 'NOT NULL' constraints to Tbl_Info;
24:     end if
25:     if v.option = 'domain' then
26:        add 'CHECK' constraints to Tbl_Info;
27:     end if
28:     R ← SpatialContext(Tbl_Info, Attr_List);
29:     end for
30:     return R
```

알고리즘 SchemaGeneration은 방향성을 갖는 GML 스키마 그래프를 입력으로 공간 스키마를 반환한다. 각 정점에 대해 먼저 테이블 정보(Tbl_Info)와 속성 리스트(Attr_List)를 초기화한다(line 02). 진입 차수가 0인 최상위 엘리먼트와 진입 차수가 2 이상이면서 진출 차수가 0인 엘리먼트는 독립적인 테이블을 구성하기 위해 정점의 정보를 테이블 정보에 추가하고 또한 속성간의 순서 정보를 유지하기 위해 'order' 필드를 추가한다(lines 03-07). 인접한 자식 노드들중에서 'ID'로 설정된 정점이 있는 경우 속성 리스트에 추가하고, 그렇지 않은 경우에는 임의로 'id' 필드를 속성 리스트에 추가하여 인식자 필드로 사용한다(lines 08-12). 정점의 엘리먼트 타입이 기하 객체인 경우에는 후행하는 노드들을 이용하여 공간 데이터베이스에 적용하기 위한 기하 객체의 형태로 변환하여 속성 리스트에 추가한다(lines 13-16). 진입 차수가 1보다 클 경우 상위 테이블과의 관련성을 부여하기 위해 상위 테이블의 키를 속성 리스트에 추가하고 외래키 제약조건을 테이블 정보에 정의한다(lines 17-20). 일반 속성들은 속성 리스트에 그대로 추가된다(line 21). 정점의 속성이 'required' 옵션일 경우에는 널(NULL)을 허용하지 않으므로 'NOT NULL' 제약조건을 테이블 정보에 추가한다(lines 22-24). 정점의 속성이 'domain' 옵션일 경우 값의 범위를 지정하기 위해 'CHECK' 제약 조건을 테이블 정보에 추가한다(lines 25-27). 마지막으로 테이블 정보와 속성 리스트를 이용하여 공간 스키마를 생성하고(line 28), 이를 결과로 반환한다(line 30).

5.3 평가

본 절에서는 GML 문서를 공간 데이터베이스로 변환하는 기존의 연구들과 제안 기법에 대한 비교 평가 내용을 기술한다.

[22]에서는 GML 문서의 기하 객체 타입과 속성 타입을 ZEUS의 공간 및 비공간 데이터 타입으로 저장하여 GML 문서를 공간 데이터베이스에 저장하는 방법을 제공하지만, 이 기법에서는 GML 문서를 2차원의 공간 데이터베이스에 저장할 때 방향성 그래프 구조의 GML 문서가 갖는 계층 그리고 순서 따위의 구조적 특성을 고려하지 않음으로써 GML 문서의 의미가 손실될 수 있는 단점이 있다. 따라서 사용자 질의 등에 따라 GML 문서를 출판하려할 때 원문을 복원할 수 없는 문제가 도출된다.

[23]에서는 GML 문서의 구조 정보를 나타내는 DTD나 스키마를 이용하여 GML 문서의 기하 객체 타입을 공간 데이터베이스의 공간 객체 타입으로 저장하는 방법을 제공하고 있다. 이 기법은 GML 문서의 구조 정보에서 한 GML 문서에 하나의 레이어에 대한 정보만을 갖도록 규정하고 있다. 이는 GML 문서를 공간 데이터베이스로에 저장할 때 작업을 단순화시킬 수 있으나, GML 문서에 대한 구조는 생성되는 형태에 따라 매우 다양하게 표현될 수 있으므로 다양한 형태의 DTD나 스키마를 수용할 수 없다는 단점이 있다. 그리고, 이 기법 또한 GML 문서를 공간 데이터베이스에 저장할 때 제약 조건 및 순서 정보에 대한 정보를 유지하지 않으므로 GML 문서 복원이 불가능하다.

제안 기법은 방향성 그래프 구조를 갖는 GML 문서의 모델링 및 분석을 통해 GML 문서상에 나타나는 제약 조건 및 순서 정보를 유지할 수 있는 방법을 제공하였다. 제안 모델링 기법에서 GML 스키마는 방향성을 갖는 그래프 구조로 변환되며, 이 그래프를 기반으로 GML 스키마를 공간 스키마로 매핑하는 규칙과 GML 문서를 이용하여 공간 릴레이션을 구성하였다. 그리고 GML 스키마의 다양한 의미적 제약 조건들을 유지하기 위해 공간 데이터베이스에서 제공하는 도메인 제약 조건과 무결성 제약 조건 등을 이용하여 기술하였다.

6. 결론

본 논문에서는 방향성을 갖는 스키마 그래프에 대한 매핑 규칙을 정의하여 GML 문서를 공간 데이터베이스로 변환하는 방법을 제안하였다.

본 논문은 XML 기반의 GML 문서를 공간 데이터베이스에 변환하기 위해 먼저 GML 문서에 대한 모델링 과정을 통해 GML 문서의 구조 정보를 표현하는 GML 스키마를 방향성을 갖는 그래프의 형태로 정점과 간선 그리고 순서 속성을 통해 언어 독립적인 정형

화를 수행하였다. 이를 기반으로 GML 스키마를 이용하여 공간 스키마로의 매핑을 위한 스키마 분석을 수행하였으며, 그 결과로 GML 스키마로부터 공간 스키마로의 매핑을 위한 규칙을 정의하였다. 이러한 규칙은 이동객체를 포함하는 GML 문서의 예제에 적용되어 GML 문서가 공간 스키마로 매핑되고 공간 릴레이션이 구성되는 과정을 설명하였다. 여기에서는 기본적인 릴레이션 구성외에도 GML 문서에 존재하는 도메인 제약 사항과 참조 무결성 제약 사항들을 만족하기 위한 규칙들도 포함된다. 제안 기법을 통해 GML 문서를 공간 데이터베이스로 변환할 때 발생할 수 있는 GML 문서의 의미적 손실을 보상할 수 있으며, 공간 데이터베이스를 GML 저장소로 활용할 수 있는 방법을 제공하여 이질적인 지리 정보의 상호 운용성뿐만 아니라 대량의 GML 문서에 대해 효과적인 저장과 관리가 가능하게 되었다.

향후 연구로는 GML 문서의 스키마가 존재하지 않을 경우 이를 효과적으로 저장할 수 있는 방법에 대한 연구와 지리정보를 효과적으로 검색할 수 있는 질의 처리 방법에 대한 추가적인 연구가 필요하다.

참고문헌

[1] T. Bray and et al., "Extensible Markup Language (XML) 1.0 (Second Edition)," W3C Recommendation, http://www.w3.org/TR/REC-xml, 2000

[2] M. Arikawa and K. Kubota, "A Standard XML Based Protocol for Spatial Data Exchange-Its Capabilities and Real Applications," International Workshop on Emerging Trechnologies for geo-based applications, May 21-26, pp37-45, 2000.

[3] OGC, "Geography Markup Language (GML) Implementation Specification v1.0," Document Number: 00-029, http://www.opengis.net/gml/00-029/GML.html,2000

[4] OGC, "Geography Markup Language (GML) Implementation Specification v2.1.1," Document Number: 02-009, http://www.opengis.net/gml/02-009/GML2-11.html , 2002..

[5] OGC, "Geography Markup Language (GML) Implementation Specification v3.0," Reference number:?OGC 02023r4, 2003.

[6] OGC, "Web Map Server Draft Candidate Implementation Specification," Version 1.0.7, 2001.

[7] S. Shekhar and et al., "WMS and GML based Interoperable Web Mapping System," In Proceedings of the Ninth ACM International Symposium on Advances in Geographic Information Systems, ACMGIS. ACM Press, 2001.

[8] J.E. Corcoles and P. Gonzalez, "A Specification of a Spatial Query Language over GML," In Proceedings of the Ninth ACM International Symposium on Advances in Geographic Information Systems, ACMGIS. ACM Press, 2001.

[9] R. R. Vatsavai, "GML-QL: A Spatial Query Language Specification for GML," http://www.cobblestoneconcepts.com/ucgis2summer2002/vatsavai/vatsavai.htm, 2002.

[10] A. Zipf and S. Kr?ger, "TGML: extending GML by temporal constructs - a proposal for a spatiotemporal framework in XML," Proceedings of the ninth ACM international symposium on Advances in geographic information systems, November 09-10, 2001.

[11] eXcelon Corporation, "eXtensible Information Server," http://www.exln.com

[12] J. McHugh, S. Abiteboul, R. Goldman, D. Quass and J. Widom, "Lore: A Database Management System for Semi-Structured

Data," ACM SIGMOD Record 26(3), 1997.

[13] V. Christophides and et al., "From Structured Documents to Novel Query Facilities," In Proc. Int'l Conf. on Management of Data, ACM SIGMOD, pp.313-324, 1994.

[14] D. Florescu and D. Kossman, "A Performance Evaluation of Alternative Mapping Schemes for Storing XML Data in a Relational Database," Rapport de Recherche, No.3680, INRIA, 1999.

[15] J. Shanmugasundaram, K. Tufte, C. Zhang, G. He, D. J. DeWitt and J. F. Naughton, "Relational Databases for Querying XML Documents: Limitations and Opportunities," In Proc. of VLDB Conference, 1999.

[16] D. Florescu and D. Kossman, "Storing and Querying XML Data using an RDBMS," IEEE Data Engineering Bulletin, Vol. 22, No. 3, pp. 27-34, 1999.

[17] J. Cheng and J. Xu, "XML and DB2," In Proc. the 16th International Conference on Data Engineering, pp. 569-573, San Diego, CA, USA, 2000.

[18] OGC, "OpenGIS simple features specification for sql," http://www.opengis.org/techno/specs/99-049.pdf, 1999.

[19] S. K. Park, S. Y. Park, W. Chung, M. K. Kim and H. Y. Bae, "GMS: Spatial Database Management System," Proceedings of Open GIS Conference, 2003.

[20] R. H. Guting and et al, "A Foundation for Representing and Querying Moving Objects," ACM Transactions on Database Systems, Vol.25, No.1, pp.1-42, 2000.

[21] David Beech, Ashok Malhotra and Michael Rys, "A Formal Data Model and Algebra for XML," W3C XML Query working group note, September 1999.

[22] Dong-O Kim, Jae-Kwan Yun and Ki-Joon Han, "Design and Implementation of a GML Document Management System," Proceedings of The KISS Fall Conference, Vol. 28, No. 2, pp. 85-87, Oct. 2001.

[23] Young-Soo Ahn, Soon-Young Park, Warnill Chung and Hae-Young Bae, "A Method of converting geographic information between spatial database and GML," Proceedings of The KIPS Fall Conference, Vol. 9, No. 2, pp.1745-1748, 2002.

정원일

1998년 인하대학교 전자계산공학과 졸업(공학사)

2004년 인하대학교 대학원 컴퓨터정보공학과 졸업(공학박사)

2004년 ~ 현재 한국전자통신연구원 디지털홈연구단 인터넷서버그룹 선임연구원

관심분야 : 데이터스트림, 센서 네트워크, 이동 객체 데이터베이스, LBS 등

박순영

1989년 인하대학교 전자계산학과 졸업(이학사)

1991년 인하대학교 전자계산공학과 졸업(공학석사)

1991년 ~ 현재 한국전자통신연구원 컴퓨터시스템연구부 선임연구원

2002년 ~ 현재 인하대학교 대학원 전자계산공학과 박사과정

관심분야 : 저장 시스템, XML, GML, 분산 데이터베이스 시스템

배해영

1974년 인하대학교 응용물리학과 졸업(공학사)

1978년 연세대학교 대학원 전자계산학과 졸업(공학석사)

1989년 숭실대학교 대학원 전자계산학과 졸업(공학박사)

1985년 Univ. of Houston 객원 교수

1992년 ~ 1994년 인하대학교 전자계산소 소장

1982년 ~ 현재 인하대학교 전자계산공학과 교수

1999년 ~ 현재 지능형 GIS 연구센터 소장

2000년 ~ 현재 중국 중경우전대학교 대학원 명예교수

관심분야 : 분산데이터베이스, 공간데이터베이스, 지리정보시스템, 멀티미디어 데이터베이스 등