GML 응용스키마를 이용한 공간데이터베이스 스키마 모델링

정호영*, 이민우*, 전우제*, 박수홍**
* 인하대학교 공과대학 지리정보공학과 대학원
** 인하대학교 공과대학 지리정보공학과 조교수

Jeung Ho-Young, Lee Min-Woo, Chun Woo-je, Park Soo-Hong

요 약

GML 데이터는 공간 및 비공간 정보를 동시에 갖는 GIS 데이터의 특징과 구조적(structured)인 XML 데이터의 성격을 함께 가지고 있어 일반적인 DBMS에 저장되기 힘들다. XML 저장이 가능한 데이터베이스는 공간데이터 처리 능력이 부족하고, 공간데이터베이스는 XML 데이터를 저장하기 어렵다.

본 연구는 GML 데이터가 공간데이터베이스에 저장될 수 있도록 GML 응용스키마로부터 공간데이터베이스 스키마를 모델링하는 방법을 제안한다. 이를 위하여 객체관계형 데이터베이스의 특징인 복합 애트리뷰트(composite attribute)와 추상데이터타입(ADT)을 이용한 GML 스키마의 맵핑 규칙을 정하였다. 맵핑 규칙은 OGC SQL 스키마에 적합하도록 GML 데이터의 공간정보와 비공간 정보를 분리하여 저장시킨다. 따라서 저장된 데이터는 공간데이터베이스가 제공하는 공간 연산자/함수 및 인덱스를 통하여 다양한 공간/비공간 질의가 빠르게 수행될 수 있다.

I. 서 론

1.1 연구 배경 및 목적

GML(Geography Markup Language)[OGC02a, OGC03]은 XML의 구조적인 성격 때문에 일반적인 DBMS에 저장되기 힘들다. 현재 XML을 기존 데이터베이스에 저장하려는 연구가 활발히 진행되고 있지만 이러한 연구들은 공간데이터에 대한 고려가 되어있지 않다. GML 데이터를 공간데이터베이스에 저장시키면 DBMS의 관리 기능과 공간 인덱스, 공간연산자 등의 다양한 공간데이터 처리 기능을 제공받을 수 있다. 하지만 지금까지는 XML(GML) 데이터를 공간데이터베이스에 저장시키려는 연구가 부족하였다.

본 연구에서는 GML 데이터를 공간데이터베이스에 저장하기 위한 스키마 모델링 기법을 제안하며 제안된 방법은 다음과 같은 세부 목적을 만족시키고자 한다.

- GML 2(version 2.1.2)와 GML 3(version 3.0.1) 스키마를 지원
- 공간데이터베이스 표준[OGC99]을 준수하여 특정 제품에 독립적인 모델링 기법을 제시
- 공간 정보의 빠른 검색과 업데이트를 위한 공간 인덱스 생성 가능

• XML 질의 언어가 아닌 공간데이터베이스에서 사용하는 질의 언어를 이용한 질의 수행

1.2 관련 연구

GML 데이터를 DBMS에 저장하고 질의를 수행하려는 연구로는 [CG02]가 있다. [CG02]에서는 3가지 XML 저장 엔진들(LegoDB, Monet, XParent)을 선정하고 GML 데이터를 저장하였다. 그리고 자신들의 선행연구[CG01]에서 제안한 GML 질의 언어로 구성된 질의문을 수행하여 각각에 대한 성능 비교를 하였다. 하지만 이 연구는 XML 저장 엔진이 있어야만 GML을 DBMS에 저장시킬 수 있고, 보편적이지 않은 특정 질의 언어를 사용해야 하는 문제가 있다.

[김동오+02] 연구는 GML 데이터를 객체-관계형 공간데이터베이스인 ZEUS에 저장시키기 위하여 변환기/관리기를 설계하고 구현하였다. GML 데이터를 공간데이터베이스에 저장시키려는 선구적인 연구였지만 ZEUS는 실질적인 산업 표준인 OGC의 표준[OGC99]과 다른 저장 스키마와 기하 데이터 모델을 가지고 있어서 특정 제품에 종속적인 변환기의 역할에 머물게 되었다.

상용 DBMS 제품들은 공간데이터 확장 기능과 XML 확장 기능을 사용하여 GML 데이터를 저장하고 공간연산을 수행할 수 있다. 하지만 각 제품에 따라 저장 방법이 다르고, XML 질의 언어도 상이하며 공간 질의 수행을 위해서는 복잡한 추가 작업을 필요로 한다.

II. GML에 적합한 XML 저장 모델

데이터베이스에 XML 문서를 저장할 때는 다음과 같은 접근 방법에 따라서 그 저장 방식이 크게 달라지며 시스템의 성능에 영향을 주게 된다.

- 문서 중심(Document-centric)의 XML과 데이터 중심(Data-centric)의 XML
- 구조-맵핑 접근(structure-mapping approach)과 모델-맵핑 접근(model-mapping approach)
- 데이터베이스 모델(관계형, 객체지향형, 객체관계형)의 유형

문서 중심 XML은 정보의 순서가 중요하며 마크업을 제거하면 기존의 텍스트 정보와 비슷한 형태를 갖게 되는 특징이 있고, 데이터 중심 XML은 정보의 순서가 중요치 않으며 비교적 정형적인 구조를 지니고 있다. GML은 전형적인 데이터중심의 XML로서 형제 노드의 정보와 같은 논리적인 XML 문서 구조의 정보보다는 XML 엘리먼트 각각의 값들을 저장, 검색, 갱신하는 작업이 중요한 의미를 갖는다.

[XREL01]에서는 XML 저장 방식을 구조-맵핑 접근과 모델-맵핑 접근법으로 나누었다. 구조-맵핑 접근 방법은 XML을 구조화 된 데이터(structured data)로 보고 DTD나 XML Schema로부터 데이터베이스 스키마를 결정하여 저장하는 방식이다[KM99, KM00, BFRS02]. 이러한 방식은 저장 공간이 줄어들고 처리 성능이 뛰어난 반면에[TDCZ99, BFRS02], DTD 요소의 열거자('|') 형식이나 카디날러티('?', '+', '*')가 관계/객체관계형으로 맵핑하기 힘든 문제가 있다. 모델-맵핑 접근 방법은 XML을 고정적인 데이터베이스 스키마를 이용하여 저장한

다[FK99, SYU99, XREL01, 박성희+00]. 이 접근 방법은 아무리 복잡한 구조의 XML 문서라도 저장시킬 수 있는 반면에 다양한 형태의 XML 데이터를 한가지 방식으로 저장하여 효율적이지 못하고 검색 성능이 구조-맵핑에 비하여 떨어진다. GML을 사용하는 응용 분야에서는 GML 응용스키마(GML application schema)를 구성하고 이에 맞는 인스턴스 데이터를 처리한다. 따라서 각 응용 분야에 적합한 데이터베이스 스키마를 결정하기 위해서는 구조-맵핑 접근 방법이 GML에 적합하다.

XML을 관계형 모델로 맵핑하면 XML 트리가 복잡할 경우 릴레이션의 수가 많아져 조인이 많이 발생하고 중복 저장되는 데이터의 양이 많아진다[SAZ94, SGT+99]. 객체지향 모델은 OGC 표준[OGC99]에 이 모델에 해당되는 스키마 설계가 존재하지 않아서 본 연구에 적합하지 못하다. 객체관계형 모델은 다중값 애트리뷰트(set-valued attribute)를 이용하여 XML(DTD)의 '*', '+'등과 같은 수량지시자를 하나의 애트리뷰트에 효과적으로 저장할 수 있다[KM99]. 따라서 XML 문서가 데이터베이스상에서 자연스럽게 표현되며 조인 연산이 줄어든다.

정리하면, (1) GML을 데이터 중심의 XML로 보고 (2) GML 응용스키마로부터 구조 정보를 분석하여 (3) 객체관계형 데이터베이스 스키마를 결정한다. 이렇게 결정된 데이터베이스 스키마에 따라 GML 인스턴스 데이터를 저장하는 것이 GML에 가장 적합한 저장 모델이다.

III. 스키마 모델링

3.1 개념 스키마 모델링(conceptual schema modeling)

E-R 데이터 모델에서 엔터티(Entity)란 실세계에 독립적으로 존재하는 실체로서 $E_1, E_2 ..., E_n$가 엔터티의 집합일때 관계 집합 $R = \{(e_1, e_2 ..., e_n) \mid e_1 \in E_1, e_2 \in E_2 ..., e_n \in E_n\}$의 부분집합이다. 여기서 $e_1, e_2 ..., e_n$는 관계 인스턴스를 말한다. 마찬가지로 $O_1, O_2 ..., O_n$이 GML에서 모델링하는 오브젝트(Object) 집합일때 GML 오브젝트 집합 $G = \{(o_1, o_2 ..., o_n) \mid o_1 \in O_1, o_2 \in O_2 ..., o_n \in O_n\}$의 부분 집합으로 표현된다. $o_1, o_2 ..., o_n$은 GML 오브젝트의 인스턴스이다.

GML에서 오브젝트는 엔터티와 동일한 개념이나 주로 지리현상과 관련된 세계를 모델링하는 엔터티이다. 따라서 다음과 같은 수학적 명제와 그에 따른 추론이 성립한다.

- $G \subseteq R$: GML 오브젝트 집합은 관계 집합의 부분 집합이다.
- $g \in G$ 이면 $g \in R$: g가 GML 오브젝트 집합의 원소이면 관계 집합의 원소이다.

따라서 GML 오브젝트의 인스턴스 o_i는 맵핑 규칙(mapping rules) 함수 μ를 적용하여 관계 인스턴스 $r_i = (e_1, e_2 ..., e_n)$에 참여하는 각 엔터티 $e_1, e_2 ..., e_n$로 표현될 수 있다.

- $\mu(o_i \ (1 \le i \le n)) \rightarrow e_j \ (1 \le j \le n)$ .....①

관계형 데이터베이스 모델에서 엔터티는 릴레이션의 튜플로 모델링되고 엔터티의 속성은 애트리뷰트[3](attribute)를 통해서 기술된다. GML 오브젝트는 XML 요소(element)로 모델링되고 GML 오브젝트의 특성을 나타내는 GML 프로퍼티(property)는 GML 오브젝트 요소의 자식 요

소로 인스턴스된다. 따라서 다음이 유도된다.

• e_i $(1 \le i \le n) \rightarrow \tau(a_1, a_2, ..., a_n)$: 엔터티 인스턴스 e_i는 n개로 이루어진 애트리뷰트 집합의 튜플 변수 τ로 표현된다.

• o_j $(1 \le j \le n) \rightarrow \ell\ (p_1, p_2, ..., p_n)$: GML 오브젝트 인스턴스 o_j는 n개로 이루어진 GML 프로퍼티 집합의 요소 변수 ℓ 로 표현된다.

• $G \subseteq R$ 이므로 $\ell\ (p_1, p_2, ..., p_n) \rightarrow \tau(a_1, a_2, ..., a_n)$ .....②

• ①, ②부터 $\mu(p_i)$ $(1 \le i \le n) \rightarrow a_j$ $(1 \le j \le n)$ .....③

그러나 ③이 항상 성립되기는 어렵다. 그 이유는 p_i가 갖는 값이 집합으로 표현되는 값이거나 또 다른 자식 요소 프로퍼티, 혹은 속성(XML attribute) 등으로 다양하게 나타날 수 있기 때문이다. 이러한 경우는 복합 애트리뷰트(complex attribute)과 다중값 애트리뷰트(set-valued attribute) 특성을 이용하여 제 1 정규형이 아닌 릴레이션을 허용하여 복합 타입의 프로퍼티를 맵핑할 수 있다.

• $\mu(p_i(p_{i1}, p_{i2}, ..., p_{in}))$ $(1 \le i \le n) \rightarrow a_j\{a_{j1}, a_{j2}, ..., a_{jn}\}$ $(1 \le j \le n)$ .....④

④는 $(p_{i1}, p_{i2}, ..., p_{in})$의 자식 노드를 갖는 GML 프로퍼티 p_i가 애트리뷰트 집합 $\{a_{j1}, a_{j2}, ..., a_{jn}\}$을 내포하는 복합 애트리뷰트 a_j로 맵핑되는 것을 보이고 있다.

3.2 논리 스키마 모델링(logical schema modeling)

1) 기본적인 스키마 맵핑 규칙

[표1]은 논리 스키마 모델링 전반에 사용되는 맵핑 규칙을 나타낸다. τ, ϵ, α는 GML 스키마에 정의된 타입, 요소, 속성을 나타내며 $\mu()$, $\mu\epsilon()$, μ_α는 각각에 해당하는 맵핑 함수를 말한다. '?', '*', '+', '|' 기호들은 DTD에서 정의하는 뜻과 같은 것으로 (0 또는 1), (0 이상), (1이상), (OR)를 말하고, <>기호는 데이터베이스에서 새로운 릴레이션 혹은 복합 애트리뷰트 생성을 표현한다. 가령 $\tau = (t_1, ..., t_n)$: $\mu(\tau) \rightarrow \langle \mu(t_1), ..., \mu(t_n) \rangle$는 타입 t_1, ..., 타입 t_n 으로 구성된 타입에 대한 변환 함수 $\mu(\tau)$를 정의한다. 변환 이후에는 각각의 변환함수를 거친 값으로 구성되는 스키마가 생성된다.

GML Schema	Database Schema
$\tau = (\tau_1, ..., \tau_n)$	$\mu(\tau) \rightarrow \langle \mu(\tau_1), ..., \mu(\tau_n) \rangle$
$\tau = (\tau_1?)$	$\mu(\tau) \rightarrow \mu(\tau_1)$
$\tau = (\tau_1+)$	$\mu(\tau) \rightarrow \{\mu(\tau_1)\}$
$\tau = (\tau_1*)$	$\mu(\tau) \rightarrow \{\mu(\tau_1)\}$
$\tau = (\tau_1 / ... / \tau_n)$	$\mu(\tau) \rightarrow \mu(\tau_1) \cup ... \cup \mu(\tau_n)$
$\tau = (\tau_1, \epsilon)$	$\mu(\tau) \rightarrow \langle \mu(\tau_1), \mu\epsilon(\epsilon) \rangle$
$\tau = (\tau_1, \alpha)$	$\mu(\tau) \rightarrow \mu(\tau_1) \cup \mu_\alpha(\alpha)$

표 1 : 기본 맵핑 규칙

3) 본 논문에서는 E-R 모델에서 사용되는 attribute는 '애트리뷰트'로, XML의 attribute는 '속성' 이라 호칭한다.

2) 단순 내용 모델 맵핑(simple content model mapping)

$\mu(p_i)$ $(1 \le i \le n) \rightarrow a_j$ $(1 \le j \le n)$ 이 성립하는 경우이다. 이 경우에 GML 프로퍼티는 속성이나 자식 요소를 내용 모델로 갖지 않아 애트리뷰트에 1:1로 맵핑될 수 있다. 하지만, 단순 타입의 경우에도 경우에 따라서는 배열과 같은 값을 가질 수 있기 때문에 애트리뷰트의 값도 다중 값을 허용할 수 있어야 한다. [표2]는 GML 오브젝트가 사용하는 단순 타입과 데이터베이스 스키마와의 맵핑 규칙을 보여준다. NullEnumeration, NullType은 GML 인스턴스 문서에서는 string 값이 오게 되지만 이것은 처리 프로세서를 위한 것이며, 실제 데이터베이스에서는 NULL 값이 저장되어야 한다.

GML Simple Types	Database Attribute Domain
τ = NullEnumeration	$\mu(\tau) \rightarrow$ NULL
τ = NullType	$\mu(\tau) \rightarrow$ NULL
τ = booleanOrNull	$\mu(\tau) \rightarrow$ BOOL
τ = booleanOrNullList	$\mu(\tau) \rightarrow$ {BOOL}
τ = booleanList	$\mu(\tau) \rightarrow$ {BOOL NOT NULL}
τ = stringOrNull	$\mu(\tau) \rightarrow$ STRING
τ = NameOrNull	$\mu(\tau) \rightarrow$ STRING
τ = NameOrNullList	$\mu(\tau) \rightarrow$ {STRING}
τ = NameList	$\mu(\tau) \rightarrow$ {STRING NOT NULL}
τ = doubleOrNull	$\mu(\tau) \rightarrow$ DOUBLE
τ = doubleOrNullList	$\mu(\tau) \rightarrow$ {DOUBLE}
τ = doubleList	$\mu(\tau) \rightarrow$ {DOUBLE NOT NULL}
τ = integerOrNull	$\mu(\tau) \rightarrow$ INTEGER
τ = integerOrNullList	$\mu(\tau) \rightarrow$ {INTEGER}
τ = integerList	$\mu(\tau) \rightarrow$ {INTEGER NOT NULL}
τ = CodeType	$\mu(\tau) \rightarrow$ STRING
τ = CodeListType	$\mu(\tau) \rightarrow$ {STRING NOT NULL}
τ = CodeOrNullListType	$\mu(\tau) \rightarrow$ {STRING}
τ = MeasureType	$\mu(\tau) \rightarrow$ DOUBLE
τ = MeasureListType	$\mu(\tau) \rightarrow$ {DOUBLE}
τ = MeasureOrNullListType	$\mu(\tau) \rightarrow$ {DOUBLE}
τ = CoordinatesType	$\mu(\tau) \rightarrow$ GEOMETRY
τ = SignType	$\mu(\tau) \rightarrow$ CHAR(1)

표 2 : GML 단순 타입의 맵핑

3) 복합 내용 모델 맵핑(complex content model mapping)

$\mu(p_i = \mu(p_{i1}, p_{i2}, \ldots, p_{in}))$ $(1 \le i \le n) \rightarrow a_j\{a_{j1}, a_{j2}, \ldots, a_{jn}\}$ $(1 \le j \le n)$는 복합 내용 모델을 갖는 GML 프로퍼티가 복합 애트리뷰트를 허용하는 비-제1정규형으로 맵핑되는 경우이다. [표3]은 이러한 경우의 맵핑 규칙을 나타내고 있다. ArrayType의 경우 <μ(AbstractGMLType), $\mu\epsilon$(members?)> 스키마가 결정되지만 μ(AbstractGMLType)은 또다시 <μ_α(id?), $\mu\epsilon$(metaDataProperty*), $\mu\epsilon$(description?) , $\mu\epsilon$(name*)>로 분리될 수 있다. 이럴 경우에는 복합 애트리뷰트를 이용하여 <<μ_α(id?), $\mu\epsilon$(metaDataProperty*), $\mu\epsilon$(description?), $\mu\epsilon$(name*)>,

$\mu\epsilon$(members?)> 스키마가 가능하다.

동일한 방법으로 맵핑 규칙을 세부적으로 적용하다 보면 GML 단순 타입을 만나게 되고, 2)절의 맵핑 규칙을 또 다시 적용하면 데이터베이스의 도메인 결정이 가능하다.

GML Basic Types	Database Schema
τ = AbstractGMLType	$\mu(\tau) \rightarrow <\mu_\alpha$(id?), $\mu\epsilon$(metaDataProperty*), $\mu\epsilon$(description?), $\mu\epsilon$(name*)>
τ = BagType	$\mu(\tau) \rightarrow <\mu$(AbstractGMLType), $\mu\epsilon$(member*), $\mu\epsilon$(members?)>
τ = ArrayType	$\mu(\tau) \rightarrow <\mu$(AbstractGMLType), $\mu\epsilon$(members?)>
τ = AbstractMetaDataType	$\mu(\tau) \rightarrow \mu_\alpha$(id?)
τ = GenericMetaDataType	$\mu(\tau) \rightarrow <\mu$(AbstractMetaDataType), Ψ*>
τ = AssociationType	$\mu(\tau) \rightarrow \mu\epsilon$(_Object?) $\cup$ μ_α(AssociationAttributeGroup)
τ = ReferenceType	$\mu(\tau) \rightarrow \mu_\alpha$(AssociationAttributeGroup)
τ = ArrayAssociationType	$\mu(\tau) \rightarrow \mu\epsilon$(_Object*)
τ = MetaDataPropertyType	$\mu(\tau) \rightarrow \mu\epsilon$(_MetaData?) $\cup$ μ_α(AssociationAttributeGroup) $\cup$ Ψ?
τ = AssociationAttributeGroup	$\mu(\tau) \rightarrow \mu_\alpha$(xlink:simpleLink) $\cup$ μ_α(remoteSchema?)
τ = StringOrRefType	$\mu(\tau) \rightarrow$ string $\cup$ μ_α(AssociationAttributeGroup)

표 3 : GML 복합 타입의 맵핑

'Ψ' 기호는 값을 참조하는 경우에 사용되며 프로세싱 타임에 처리되어야 할 사항을 말한다.

3.3 GML 스키마와 GML 응용스키마

GML 응용스키마는 애플리케이션 도메인에 맞도록 GML 스키마의 타입을 확장하거나 제한하여 기술된다. 따라서 두 스키마 사이에는 다음과 같은 관계가 존재한다.

- GML 스키마에 정의된 요소나 속성이 응용스키마의 내용 모델을 정의하는데 수정 없이 사용되거나(ex, gml:name, gml:id), 대체 그룹의 헤드로 사용된다(ex,gml:_GML)
- GML 스키마에 정의된 타입이 응용스키마에서 내용 모델을 기술하는데 그대로 사용되거나(ex, gml:EnvelopeType), 타입 유도의 기본 타입이 된다.(ex, gml: Abstract-GeometryType)

따라서 응용스키마에 정의된 구조 정보는 기본 타입(base type)이나 대체 그룹에 나타나는 헤드 요소를 추적해 나가면 GML 스키마의 정보를 만나게 된다. 이때 3.2절에서 정의해 둔 GML 스키마의 데이터베이스 맵핑 규칙을 적용하면 GML 응용스키마의 맵핑 규칙이 정해진다.

IV. 공간데이터베이스 스키마 결정

4.1 OGC 스키마

[OGC99]는 공간/비공간데이터의 저장을 위한 렐레이션 스키마, 기하 정보를 위한 추상데이터타입과 공간 함수 및 연산자들을 정의한다. 저장 스키마는 관계형 데이터베이스 모델에 해당하는 2가지의 저장 방식과 1가지의 객체관계형 저장 방식이 있다. 객체관계형 저장 방식은 메타데이타를 포함하는 GEOMETRY_COLUMNS TABLE, 좌표계 정보를 갖는 SPATIAL_REF_SYS TABLE, 실제 데이터를 저장하는 FEATURE TABLE을 두고 키를 이용하여 연결되어 있다[그림 1].

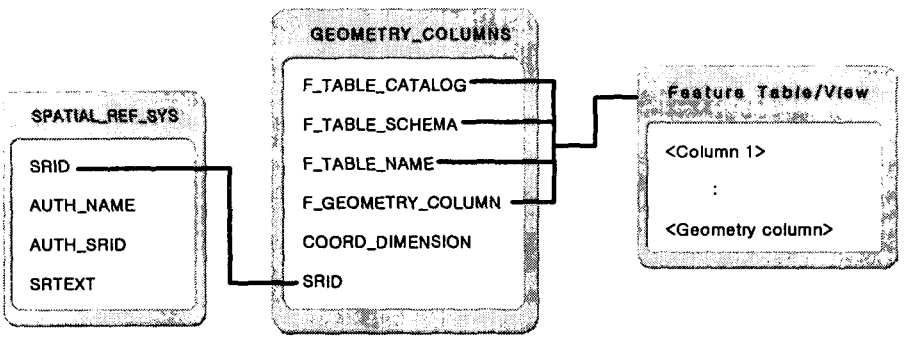

그림 1 : SQL 92 with Geometry Type

GML 데이터를 공간데이터베이스에 저장할 때, 도형 정보는 FEATURE TABLE의 Geometry 컬럼으로, 속성 정보는 다른 컬럼으로 분리하여 저장해야 한다. 그 이유는 Geometry 컬럼에 저장된 기하 정보는 공간인덱스를 쉽게 생성하고, 다양한 공간 연산자/함수 사용이 가능하기 때문이다.

4.2 GML Geometry 맵핑

공간데이터베이스의 기하 모델은 Simple Features를 따르고 있기 때문에 기하 정보를 7가지 인스턴스형 타입으로 표현하고 있다. GML2[OGC02]에서는 동일한 기하 모델을 사용하므로 맵핑에 문제가 없으나 GML3[OGC03]에서는 기하 모델이 다양해졌기 때문에 문제가 발생한다.

본 연구에서는 이러한 문제를 해결하기 위하여 NonSimpleGeometry 라는 추상데이터 타입을 기존의 Geometry 타입에 추가할 것을 제안한다.

GML3 Geometry		Spatial Database Geometry
Point	→	Point
LineString	→	LineString
Curve	→	NonSimpleGeometry 또는 *LineString*
OrientableCurve	→	NonSimpleGeometry
CompositeCurve	→	NonSimpleGeometry
Polygon	→	Polygon
Surface	→	NonSimpleGeometry 또는 *Polygon*
OrientableSurface	→	NonSimpleGeometry
CompositeSurface	→	NonSimpleGeometry
Solid	→	NonSimpleGeometry
CompositeSolid	→	NonSimpleGeometry
GeometricComplex	→	NonSimpleGeometry
MultiPoint	→	MultiPoint
MultiCurve	→	*MultiLineString* 또는 NonSimpleGeometry
MultiSurface	→	*MultiPolygon* 또는 NonSimpleGeometry
MultiSolid	→	NonSimpleGeometry
MultiGeometry	→	*GeometryCollection* 또는 NonSimpleGeometry

표 4 : GML3 기하 모델 맵핑

[표4]는 GML3와 공간데이터베이스 기하 모델의 맵핑 규칙을 나타낸다. 이탤릭체는 일부가 허용될 수 있음을 뜻한다.

4.3 GML Feature 맵핑

다음은 GML 피쳐 오브젝트 타입으로 구성된 응용스키마의 인스턴스 데이터가 공간데이터베이스로 저장되는 과정이다. 이 과정에서 앞서 설명된 모델링 방법과 세부 맵핑 규칙이 적용된다.

- 피쳐 컬렉션은 FEATURE TABLE로 맵핑된다. 피쳐 컬렉션이 포함하는 경계(boundedBy) 정보는 GEOMETRY_COLUMNS TABLE에 새로운 필드를 생성하여 저장한다. 피쳐 컬렉션의 이름은 FEATURE TABLE의 이름이 된다.
- 피쳐 멤버(feature member)는 FEATURE TABLE의 레코드로 맵핑한다.
- 피쳐 멤버의 자식 요소로 표현되는 피쳐 프로퍼티는 FEATURE TABLE의 컬럼으로 맵핑한다.
- 피쳐 프로퍼티의 이름은 FEATURE TABLE에서 맵핑되는 컬럼의 이름이 된다.
- 피쳐 프로퍼티가 기하 정보라면 FEATURE TABLE의 Geometry 컬럼에 맵핑한다.
- 주어진 응용스키마의 구조 정보를 분석하여 초기 스키마를 결정한다.
- 초기 스키마로에 [표5]의 맵핑 규칙을 적용하여 중간 스키마를 결정한다.
- 중간 스키마에 맵핑 규칙을 계속적으로 적용하여 더 이상 분해 될 수 없을 때까지 반복한다.
- 이제 최종스키마가 정해졌으면 데이터베이스에 적용하고 GML 인스턴스 데이터를 저장한다.

GML Feature Structures	Spatial Database Schema
τ = AbstractFeatureType	$\mu(\tau) \rightarrow$ <μ(AbstractGMLType), $\mu\epsilon$(boundedBy?), $\mu\epsilon$(location?), $\cup$ STRING>
τ = FeaturePropertyType	$\mu(\tau) \rightarrow \mu\epsilon$(_Feature?) $\cup$ μ_α(AssociationAttributeGroup)
τ = FeatureArrayPropertyType	$\mu(\tau) \rightarrow \mu\epsilon$(_Feature*)
τ = BoundedFeatureType	$\mu(\tau) \rightarrow$ <$\mu\epsilon$(metaDataProperty*), $\mu\epsilon$(discription?), $\mu\epsilon$(name*), $\mu\epsilon$(boundedBy?), $\mu\epsilon$(location?)>
τ = AbstractFeatureCollectionType	$\mu(\tau) \rightarrow$ <μ(BoundedFeatureType), $\mu\epsilon$(featureMember*), $\mu\epsilon$(featureMembers?)>
τ = FeatureCollectionType	$\mu(\tau) \rightarrow \mu$(AbstractFeatureCollectionType)
τ = boundingShape	$\mu(\tau) \rightarrow \mu\epsilon$(Envelope) $\cup$ $\mu\epsilon$(Null)
τ = BoundingShapeType	$\mu(\tau) \rightarrow \mu$(boundingShape)
τ = EnvelopeWithTimePeriodType	$\mu(\tau) \rightarrow$ <μ(EnvelopeType, <$\mu\epsilon$(TimePosition)> $\cup$ Ψ?>
τ = LocationPropertyType	$\mu(\tau) \rightarrow \mu$(locator?) $\cup$ μ_α(AssociationAttributeGroup)
τ = locator	$\mu(\tau) \rightarrow$ <$\mu\epsilon$(_Geometry) $\cup$ $\mu\epsilon$(LocationKeyWord) $\cup$ $\mu\epsilon$(LocationString)>
τ = PriorityLocationPropertyType	$\mu(\tau) \rightarrow$ <μ(LocationPropertyType), STRING?>

표 5 : GML Feature 맵핑

VI. 결론 및 향후 연구

본 논문은 GML 응용스키마 분석을 통해 공간데이터베이스 스키마를 결정하는 모델링 방법을 제안하였다. 이 과정에서 GML 데이터에 적합한 XML 저장 모델을 제시하였으며, GML 스키마와 공간데이터베이스 스키마 사이의 개념적, 논리적 맵핑 규칙을 유도하였다. 맵핑 규칙은 저장된 GML 데이터에 대한 질의 수행 능력과 비용을 고려하였고 OGC 표준을 준수한다.

제안된 기법은 구현 과정에 독립적이어서 공간데이터베이스의 엔진 레벨에서도 구현이 가능하고, Web Feature Server에도 이용될 수 있다. 빠른 처리 성능을 요구하는 분야가 아니라면 XSL/XSLT 기술을 이용한 손쉬운 응용도 가능하다.

차후 연구는 제안된 모델링 기법을 기반으로 구현 과정을 거친 이후에, coverage나 observation, topology와 같은 다양한 데이터 모델을 저장하기 위한 방법을 연구하고자 한다.

참고문헌

[OGC99]David Beddoe, Paul Cotton, Robert Uleman, Sandra Johnson, Dr. John R. and Herring, "OpenGIS Simple Features Specification for SQL Revision 1.1", OpenGIS Consortium, 1999.

[OGC02]Simon Cox , Adrian Cuthbert, Ron Lake, and Richard Martell, "Geography Markup Language(GML) Implementation Specification, version 2.1.2",

OpenGIS Consortium, 2002.

[OGC03]Simon Cox , Paul Daisey, Ron Lake, Clemens Portele, and Arliss Whiteside, "Geography Markup Language (GML) Implementation Specification v3.0", OpenGIS Consortium, 2003.

[CG01]J. E. Córcoles , P. González, "A specification of a spatial query language over GML", Proceedings of the ninth ACM international symposium on Advances in geographic information systems, November 09-10, 2001

[CG02]J.E. Corcoles and P. Gonzlez, "Analysis of Different Approaches for Storing GML Documents", GIS'02, November 8-9, 2002.

[SAZ94] R. Sacks-Davis, T. Arnold-Moore, and J. Zobel, "Database Systems for Structured Documents" In International Symposium on Advanced Database Technologies and Their Integration, pp. 272-283, Naran Japan, 1994

[SGT+99]Jayavel Shanmugasundaram, H. Gang, Kristin Tufte, Chen Zang, David DeWitt, and Jeffrey F. aughton, "Realtional Databases for Querying XML Documents: Limitaions and Opportunities", In Proceedings of the Conference on Very Large Databases, 1999.

[BFRS02]P. Bohannon, J.Freire, P. Roy and J. Simeon, "From XML Schema to Relations: A Cost-Based Apporach to XML Storage". 18th International Conference on Data engineering(ICDE2002), 2002.

[FK99]Daniela Florescu and Donald Kossmann, "A Performance Evaluation of Alternative Mapping Schemas for Storing XML Data in a Relational Database", Technical Report 3680, INRIA, 1999.

[KM99]M. Klettke and H. Meyer, "Managing XML documents in object-relational databases" Rostocker Informatik Fachberichte, 1999.

[KM00] Kanne, C.-C. and Moerkotte, G. "Efficient storage of XML data", In Proceedings of the International Conference on Data Engineering, 2000.

[SYU99]T. Shimura, M. Yoshikawa, and S. Uemura, "Storage and retrieval of XML documents using object-relational databases", In Proceedings of DEXA, 1999.

[XREL01]M. Yoshikawa, T. Amagasa, T. Shimura & S Uemura, "XREL: A Path-Based Approach to Storage and Retrieval of XML documents using Relational Databases," Proc. ACM Transactions on Internet Technology, Volume 5, 2001.

[TDCZ99]F. Tian, D. DeWitt, J. Chen, and C. zhung, "The design and performance evaluation of various XML storage strategies", 1999.

[김동오+02]이혜진, 김동오, 손훈수, 한기준, " GML 3.0 기반 엔코딩 서비스의 설계 및 구현", 개방형 지리정보시스템 학회 추계학술대회, 2002

[박성희+00]박성희, 박경현, 김록원, 남광우, 류근호, "ORDBMS를 이용한 XML 문서의 저장 및 질의", 한국정보과학회 춘계 학술발표논문집, 제27권 제1호, 2000