GML 문서 저장을 위한 저장 스키마 및 하부 저장 시스템의 설계

김영국° 장재우
전북대학교 컴퓨터공학과
{ygkim°, jwchang}@dblab.chonbuk.ac.kr

Design of Storage Schema and Low-level Storage System for GML Documents

Youngguk Kim° Jaewoo Chang
Dept of Computer Engineering Chonbuk National University

요 약

현재 활발하게 진행 중인 공간 네트워크 데이터베이스 연구는 지리 정보의 교환 표준으로 제시된 GML(Geographic Markup Language)을 지원하는 연구가 거의 없는 실정이다. 하지만 LBS 또는 텔레매틱스 응용에서, 시간에 따라 위치가 변화하는 이동 객체를 표현하는 데 있어 GML의 활용도가 크게 증가하고 있다. 또한, 공간 네트워크 데이터베이스 연구 대부분이 인코딩 표준인 GML 로 작성된 데이터들을 다룰 수 없어, 각 연구에서 사용하는 저장 시스템은 범용화되기 어렵다. 이에 따라 범용성을 지닌 GML 지원 저장 시스템 개발이 필수적이다. 본 논문에서는 범용성을 지닌 GML 문서 저장을 위한 저장 스키마 및 하부 저장 시스템을 설계한다.

1. 서 론

현재 활발하게 진행 중인 공간 네트워크 데이터베이스 연구는 지리 정보의 교환 표준으로 제시된 GML(Geographic Markup Language)을 지원하는 연구가 거의 없는 실정이다. GML은 OGC(OpenGIS Consortium)에서 지리 정보의 저장 및 전송을 위한 인코딩 표준으로 제안한 마크업 언어이다. 일반적인 공간 데이터베이스에서 GML 지원을 위한 연구는 크게 3가지 연구로 요약된다. GML 문서의 파싱(parsing), GML 문서의 저장, GML 질의어 등이다.

본 논문에서는 위의 3가지 주제 가운데 GML 문서의 저장을 위한 연구를 다룬다. 먼저 기존 XML 저장스키마는 공간지리정보 표현에 적합하지 않기 때문에, 공간 지리정보를 포함한 GML을 저장하기 위한 저장스키마를 제시하고, 아울러 이러한 저장스키마를 바탕으로 범용성을 지닌 GML 지원 하부 저장 시스템을 설계한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구를 소개하고, 3장에서는 GML 문서를 저장하기 위한 저장스키마를 제시한다. 4장에서는 GML지원 하부 저장 시스템의 구조를 제시한다. 마지막으로 5장에서는 결론 및 향후연구를 제시한다.

2. 관련연구

일반적으로 관계형 데이터베이스에 XML/GML 문서를 저장하기 위한 연구는 모델 사상(Model Mapping)접근과 구조 사상(Structure Mapping)접근으로 분류할 수 있다[1][2][3].

모델 사상기법은 정해진 데이터베이스 스키마를 이용하여 DTD나 스키마 문서 없이 XML/GML문서를 저장하는 기법이다. 즉, 모델 사상기법은 XML문서를 데이터 모델로 변환하는 과정에서 XML의 구조적 특징과 내용을 문서 독립적으로 모델링하여 저장하지 않으므로 XML문서 구조의 갱신이 발생할 경우, 이에 따른 데이터베이스 구조정보가 수정되지 않아도 되는 기법이다. 대표적인 시스템으로 간선(Edge)기반의 Monet[4]과 정점(Node) 기반의 XParent[5], XRel[6] 등이 있다.

관계형 데이터베이스 기술을 기반으로 XML문서를 저장하는 또 다른 방법은 XML문서의 구조 정보를 표현하는 DTD(Document Type Definition)또는 XML 스키마를 기반으로 관계형 데이터베이스의 스키마를 정의하고 정의된 테이블에 XML문서를 저장하는 XML스키마 종속적인 구조 사상(Structure Mapping)방법이 있다. 구조 사상 방식은 XML 스키마를 기반으로 생성된 관계형 데이터베이스(Relational database)의 스키마(Schema)에 따라 저장이 이루어지기 때문에, 이 방법은 스키마들을 만족하는 XML문서들이 많이 존재하거나 스키마 생성에 사용된 스키마의 수정이 빈번하게 발생하지 않는 경우에 적합하다. 대표적인 연구로는 Shared Inlining / Hybrid Inlining 기법[7]과 비용 기반 접근 방식인 LegoDB[8]가 있다.

3. GML 문서를 위한 저장 스키마의 설계

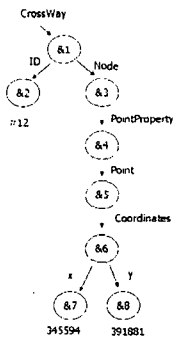
GML 문서를 저장하기 위해서는 GML 문서가 포함하

고 있는 공간지리정보를 저장해야 하기 때문에, 구조사상기법은 적합하지 않다. 그 이유는 구조사상기법은 범용적인 GML의 스키마를 표현하기에 한계성을 가지고 있기 때문이다. 또한, XML/GML 문서 고유의 Path 정보를 손실하기 때문이다.

본 연구에서는 모델사상기법의 가장 발전된 형태인 XParent를 변형하여 공간지리정보를 포함하는 GML 문서를 위한 저장스키마를 제안한다. 그림1-(a)와 같은 GML 문서가 있을 때, 그림1-(b)는 그 문서를 Data Graph로서 표현한 것이다.

```
<CrossWay>
  <ID>#12</ID>
  <gml:Node
xmlns gml="http://www.open
gis.net/gml">
    <gml:pointProperty>
      <gml:Point>
        <gml:coordinates>
          <x>345594</x>
          <y>391881</y>
        </gml:coordinates>
      </gml:Point>
    </gml:pointProperty>
  </gml:Node>
</CrossWay>
```

(a). GML 문서의 예시



(b). (a)의 Data Graph

그림1. GML 문서 및 문서의 Data Graph

그림2는 그림1의 Data Graph를 XParent의 5개의 테이블로 표현한 것이다. XParent의 5개의 테이블은 다음과 같다.

LabelPath(PathID, Length, Path)
Data(PathID, DataID, Value)
Element(PathID, Ordinal, DataID)
DataPath(Parent Data ID, Child Data ID)
Ancestor(DataID, Ancestor Data ID, Level)

PID	Len	Path
1	1	/CrossWay
2	2	/CrossWay /ID
3	2	/CrossWay /Node
4	3	/CrossWay /Node /PointProperty
5	4	/CrossWay /Node /PointProperty /Point
6	5	/CrossWay /Node /PointProperty /Point /Coordinates
7	6	/CrossWay /Node /PointProperty /Point /Coordinates /x
8	6	/CrossWay /Node /PointProperty /Point /Coordinates /y

(a). Label Path Table

PID	DID	Value
2	&2	#12
7	&7	345594
8	&8	391881

(b). Data Table

PID	Ordinal	Did
1	1	&1
2	1	&2
3	1	&3
4	1	&4
5	1	&5
6	1	&6
7	1	&7
8	1	&8

(c). Element Table

PID	CID
&1	&2
&1	&3
&3	&4
&4	&5
&5	&6
&6	&7
&6	&8

(d). Data Path Table

DID	Ancestor	Level
&2	&1	1
&3	&1	1
&4	&1	2
&5	&1	3
&6	&1	4
&7	&1	5
&8	&1	5
&4	&3	1
&5	&3	2
&6	&3	3
&7	&3	4
&8	&3	5
&5	&4	1
&6	&4	2
&7	&4	3
&8	&4	3
&6	&5	1
&7	&5	2
&8	&5	2
&7	&6	1
&8	&6	1

(e). Ancestor Table

그림2. XParent의 5개의 Table

그림2-(b)의 Data Table을 살펴보면, point 형태가 x,y로 나뉘어져서 point 자체의 공간지리정보의 의미를 잃게 된다. 따라서 본 논문에서 제시하는 기법은 그림3에서와 같이 현재의 Data Table 밑에 지리정보를 포함한 자료와 그렇지 않은 자료로 나누어 Spatial Data Table과 Non Spatial Data Table로 표현하는 기법이다. 따라서 Spatial Index를 사용하여 Spatial Data를 별도로 관리할 수 있으며, 그 외의 Non Spatial Data는 기존의 색인 시스템을 사용하여 관리할 수 있다.

PID	DID	type	Real ID
2	&2	NS	1
6	&6	S	1

(a). Data Reference Table

RID	type	value
1	string	#12

(b). Non Spatial Data Table

RID	type	value
1	point	345594,391881

(c). Spatial Data Table

그림3. Spatial Data을 지원하는 확장된 Table

따라서 본 논문에서 제안하는 저장스키마의 테이블은 다음과 같다.

LabelPath(PathID, Length, Path)
Element(PathID, Ordinal, DataID)
DataPath(Parent Data ID, Child Data ID)
Ancestor(DataID, Ancestor Data ID, Level)
Data Reference(PathID, DataID, type, RealID)
Non Spatial Data(RID, type, value)
Spatial Data(RID, type, value)

Data Reference Table의 type은 Non Spatial Data 인지 Spatial Data 인지 구분하는 값이고, Real ID는 하부 Data Table의 ID값을 가리킨다. Data Table 의 RID는 각각을 구분할 수 있는 ID값이고, type은 string, integer, real, point(s), polyline(s), polygon(s)을 구분하는 값이며, value는 실제 값이 들어가게 된다. 그 외 다른 Table의 내용은 XParent의 내용과 동일하다.

4. GML 지원 하부 저장 시스템의 설계

그림4는 GML 문서 저장을 위한 하부 저장 시스템의 구조를 보여주고 있다. Storage Manager, Spatial Data Type, Spatial Index, Non-Spatial Index, User Interface의 총 5개의 모듈로 구성되어 있으며 그 세부 내용은 아래와 같다.

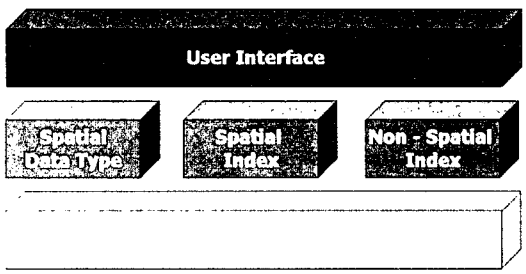


그림4. 하부저장 시스템의 구조

① Storage Manager : 하부저장구조에서 최하위단으로서 앞장에서 설명한 스키마를 저장 관리하기 위해 Berkeley DB[9]를 이용한다.

② Spatial Data Type : Spatial Index의 값을 저장하기 위해서 기본적인 Integer, char, real type을 확장해서 Spatial Data Type인 point(s), polyline(s), polygon(s) 등을 지원하기 위해 구성된 모듈이다.

③ Spatial Index : Spatial Data Type의 색인을 구성하기 위한 R*-Tree 모듈이다. 예를 들면 표8의 Spatial Data Table 의 Value의 색인을 구성한다.

④ Non-Spatial Index : Spatial Data Type을 제외한 Non-Spatial Type의 색인을 구성하기 위한 B*-Tree 모듈이다. Spatial Index가 쓰이지 않는 Table들의 Key 값들의 색인을 구성한다.

⑤ User Interface : 이 모듈을 통해서 사용자가 원하는 질의를 Spatial Index(R*-Tree) 또는 Non Spatial Index(B+-Tree)를 이용하여 처리하고, 그 결과값을 반환하는 모듈이다.

Storage Manager를 Berkeley DB로 사용하는 이유는 오픈소스 임베디드 데이터베이스 라이브러리이기 때문이다. 또한, 데이터베이스로서 갖추어야 할 동시성제어, Transaction(트랜잭션), Recovery(리커버리), Lock(락), Logging(로깅)을 제공한다. Berkeley DB는 DBMS가 아닌 DataBase Library 이기 때문에 본 논문에서 설계한 시스템에 적당한 라이브러리라 할 수 있다.

5. 결론

본 논문에서는 GML 문서 저장을 위해서 XParent를 변형하여, Spatial Data를 지원할 수 있는 저장 스키마를 제시하고, 이러한 저장 스키마에 맞는 GML 지원 하부 저장 시스템을 설계하였다.

향후 연구로 설계한 하부 저장 시스템을 구현하고, 제안한 저장 스키마의 효율성을 기존 스키마들과 비교하는 것이다.

Acknowledgment

본 결과물은 교육인적자원부, 산업자원부, 노동부의 출연금 및 보조금으로 수행한 최우수실험실지원사업의 연구결과입니다.

참고문헌

[1] F. Tian et al., "The Design and Performance Evaluation of Alternative XML Storage Strategies", SIGMOD record, vol 31, No 1, 2002.

[2] 민준기 외 3명, "다양한 저장소에서의 효율적인 XML 저장 기법에 대한 연구", 데이터베이스연구, 제19권 1호 2003.

[3] J. Corcoles et al., "Analysis of Different Approaches for Storing GML Documents", Proceedings of the tenth ACM international symposium on Advances in geographic information systems 2002.

[4] A. Schmidt et al., "Efficient Relational Storage and Retrieval of XML Documents" In Proceedings of WEBDB 2000.

[5] Haifeng Jiang et al., "Path Materialization Revisited: An Efficient Storage Model for XML Data", the 2nd Australian Institute of Computer Ethics Conference 2000.

[6] M. Yoshikawa et al., "Xrel: A path-based approach to storage and retrieval of XML Documents using Relational Databases", ACM Transactions on Internet Technology, Vol. 1, No. 1, 2001.

[7] Jayavel Shanmugasundaram, H. Gang, Kristin Tufte, Chun Zhang, David J. DeWitt, and Jeffrey F. Naughton. "Relational databases for querying XML documents: Limitations and opportunities." In VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, Edinburgh, Scotland, pages 302-304, 1999.

[8] P. Bohannon et al., "LegoDB - From XML scheme to relations : a cost-based approach to XML storage", In Proceeding of International Conference on Data Engineering 2002.

[9] Berkeley DB "http://www.sleepycat.com"