모바일 환경을 위한 GML 기반 시공간 질의 처리 시스템

Spatio-Temporal Query Processing System based on GML for The Mobile Environment

김 정 준[*] 신 인 수[**] 원 승 호[***] 이 기 영[****] 한 기 준[*****]
Joung-Joon Kim In-Su Shin Seung-Ho Won Ki-Young Lee Ki-Joon Han

요 약 최근 무선 액세스 망의 범위가 증가하고 발전함에 따라 다양한 분야에서 u-GIS 서비스가 지원되고 있으며, 특히 모바일 환경에서의 u-GIS 서비스를 위해 시공간 데이터가 널리 활용되고 있다. 그러나 모바일 환경에서 활용되는 시공간 데이터에 대한 표준이 없으므로 서로 다른 시공간 데이터를 사용하는 모바일 u-GIS 서비스 간의 상호운용성을 위한 효율적인 시공간 데이터 처리 기술이 필요하다. 또한 모바일 장치의 저용량과 낮은 성능을 고려한 시공간 데이터의 수집, 저장, 관리 시스템이 필수적이다. 따라서 본 논문에서는 모바일 환경에서 시공간 데이터의 효율적인 관리를 위해 GML 기반의 질의 처리 시스템을 설계 및 구현하였다. GML 기반 시공간 질의 처리 시스템은 GML 문서의 특성인 상호운용성을 유지하고 저장 효율성을 높이기 위해 GML 스키마와 저장 테이블을 매핑하는 구조형 저장 방식과 Fast Infoset 기법을 이용한 바이너리 XML 저장 방식을 제공한다. 그리고 저장된 GML 문서의 시공간 데이터에 대한 신속한 질의 처리를 위하여 시공간 연산자를 제공한다. 마지막으로 본 논문에서 개발한 시스템을 가상 시나리오에 적용하여 본 시스템이 u-GIS 서비스를 위한 시스템으로 활용될 수 있음을 확인하였다.

키워드 : GML, u-GIS, 시공간 데이터, 시공간 연산자, 시공간 질의 처리, 모바일 환경

Abstract Recently, with increase and development of the wireless access network area, u-GIS Service is supported in various fields. Especially, spatio-temporal data is used in the mobile environment for the u-GIS service. However, there is no standard for the spatio-temporal data used in different spaces, spatio-temporal data processing technology is necessary to makes interoperability among mobile u-GIS services. Furthermore, it is also necessary to develop the system of gathering, storing, and managing the spatio-temporal data in consideration of small capacity and low performance of mobile devices. Therefore, in this paper, we designed and implemented a spatio-temporal query processing system based on GML to manage spatio-temporal data efficiently in the mobile environment. The spatio-temporal query processing system based on GML can offer a structured storage method which maps a GML schema to a storage table and a binary XML storage method which uses the Fast Infoset technique, so as to support interoperability that is an important feature of GML and increase storage efficiency. we can also provide spatio-temporal operators for rapid query processing of spatio-temporal data of GML documents. In addition, we proved that this system can be utilized for the u-GIS service to implement a virtual scenario.

Keywords : GML, u-GIS, Spatio-temporal Data, Spatio-temporal Operator, Spatio-temporal Query Processing, Mobile Environment

† 본 연구는 국토해양부 첨단도시기술개발사업 - 지능형국토정보기술혁신 사업과제의 연구비지원(10국토정보J71)에 의해 수행되었습니다.

[*] 건국대학교 컴퓨터공학부 강의교수 jjkim9@db.konkuk.ac.kr
[**] 건국대학교 컴퓨터공학부 박사과정 isshin@db.konkuk.ac.kr
[***] 동양증권 그룹서비스본부 금융서비스팀 사원 seungho.won@tongyang.co.kr
[****] 을지대학교 의료IT마케팅학과 교수 kylee@eulji.ac.kr
[*****] 건국대학교 컴퓨터공학부 교수 kjhan@db.konkuk.ac.kr (교신저자)

1. 서 론

최근 WiFi, 3G 등과 같은 무선 액세스 망의 범위가 증가하고 발전함에 따라 교통(u-Transport), 복지(u-Care), 문화(u-Culture), 환경(u-Green), 산업(u-Business), 도시(u-City), 행정(u-Government) 등의 유비쿼터스 환경이 구현되고 있다. 이러한 유비쿼터스 환경을 위한 핵심 기술인 u-GIS는 PDA, 넷북, 스마트폰과 같은 모바일 디바이스의 급속한 보급으로 인하여 모바일 환경에서의 활용이 점차 증대되고 있다. 특히 모바일 환경에서는 공간 데이터에 시간 데이터가 포함된 시공간 데이터가 많이 활용되고 있으며, 이에 대한 수집, 저장, 관리 시스템이 필수적이다[12, 13].

또한 u-GIS 서비스에서 제공되는 공간 데이터의 상호운용성을 위해 OGC가 제시한 국제 표준인 GML (Geography Markup Language)이 개발되어 널리 활용되고 있으며[4], 최근에는 국내 표준화 기구인 TTA에서 모바일 서비스용 GML 프로파일을 제시하였다[18]. OGC에서 제시한 GML은 모바일 서비스에서 그대로 사용하기에는 너무 많은 스키마들을 포함하고 있어 용량이 크기 때문에 모바일 상에서 사용하기 어렵다. 그리고 TTA에서 제시한 모바일 서비스용 GML 프로파일은 기존 GML에서 모바일 서비스에 반드시 필요한 스키마들만을 도출하여 정의하였기 때문에 모바일 상에서 활용할 수는 있지만 공간 데이터만을 고려하였으므로 시공간 데이터를 표현하기에는 어려움이 있다.

이처럼 현재 모바일 환경에서 활용되는 시공간 데이터에 대한 표준은 정해진 바가 없기 때문에 서비스에 따라 시공간 데이터를 제공하는 형태가 다르다. 그러므로 서로 다른 시공간 데이터를 사용하는 u-GIS 서비스간의 데이터 교환에 어려움이 많으므로 효율적인 시공간 데이터 처리가 필요하다. 그리고 모바일 환경에서 u-GIS 서비스를 제공하기 위해 시공간 데이터의 교환 시 상호운용성의 보장 또한 필요하다.

한편, 점차 증가하고 있는 모바일 환경에서의 다양한 u-GIS 서비스들에 비례하여 시공간 데이터도 기하급수적으로 늘어나고 있다. 상대적으로 성능이 낮은 모바일 디바이스에서의 파일 시스템만으로 점차 대용량화되어 가는 시공간 데이터를 저장하고 이에 대한 질의를 처리하기에는 무리가 따른다. 즉, 모바일 장치의 작은 저장 공간과 낮은 프로세서 성능을 이용하여 대용량의 시공간 데이터를 처리하기 위해서는 시공간 데이터의 용량을 줄이고 보다 효율적인 시공간 연산 기능이 요구된다. 따라서 모바일 환경에서 상호운용성을 위해 u-GIS에 기반하여 시공간 데이터를 효율적으로 처리할 수 있는 시스템이 절실히 필요하다[1, 11, 14].

본 논문에서는 모바일 환경을 위한 GML 기반 시공간 질의 처리 시스템을 설계 및 구현하였으며, 특히 모바일 환경을 위해 GML 3.0을 프로파일링하여 국내표준으로 제정한 모바일 서비스용 GML 프로파일[17, 18]을 자체적으로 확장하여 사용하였다. 본 논문에서 개발한 GML 기반 시공간 질의 처리 시스템은 모바일 장치용 공간 DBMS인 FUNs[1]를 기반으로 시공간 데이터 타입과 시간 및 시공간 연산자를 지원하도록 확장하였다. 그리고 GML을 이용한 대용량의 시공간 데이터를 효율적으로 저장 및 교환하기 위하여 GML 스키마와 저장 테이블을 매핑하는 구조형 저장 방식과 Fast Infoset을 이용한 바이너리 XML 저장 방식을 지원한다[2, 6]. 마지막으로 GML 기반 시공간 질의 처리 시스템을 시설물 관리 서비스에 적용하여 본 시스템의 효용성을 검증하였다.

본 논문의 구성은 다음과 같다. 제1장의 서론에 이어 제2장에서는 관련연구로 모바일 서비스용 GML 프로파일을 분석하고, 바이너리 XML에 대해서 설명하고, 마지막으로 본 논문에서 구현 시 사용한 FUNs에 대해서 기술한다. 제3장에서는 모바일 환경을 위한 GML 기반 시공간 질의 처리 시스템의 구조와 각 관리자들에 대하여 설명한다. 제4장에서는 시스템의 구현 환경과 GML 기반 시공간 질의 처리 시스템을 가상 시나리오에 적용하여 효용성을 검증한다. 마지막으로 제5장에서는 결론에 대하여 언급한다.

2. 관련연구

본 장에서는 모바일 환경을 고려한 모바일 서비스용 GML 프로파일에 대하여 분석하고, XML 문서의 성능을 개선하기 위한 바이너리 XML에 대하여 설명한다. 마지막으로 본 논문에서 구현 시 사용한 FUNs에 대하여 기술한다.

2.1 모바일 서비스용 GML 프로파일

모바일 u-GIS 서비스는 무선 통신 및 모바일 장치 등 환경적인 제약사항으로 인해 공간 데이터 교환을 GML을 통해 실시간으로 처리하기에는 무리가 따르며 데이터 압축 등 별도의 방법을 추가적으로 사용해야 한다[17]. 그러므로, TTA는 모바일 u-GIS 서비스에 필요한 기존 GML의 부분집합을 도출하여 정보통신단체표준으로 모바일 서비스용 GML 프로파일을 제시하였다[18].

그림 1은 모바일 서비스용 GML 프로파일의 전체 스키마 간의 관계를 보여준다.

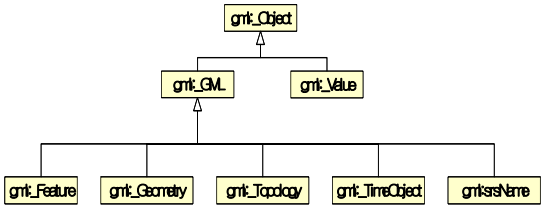


그림 1 전체 스키마 간의 관계

그림 1에서 보는 바와 같이 추상 클래스인 gml:_GML과 gml:_Value클래스는 최상위 클래스인 gml:_Object 클래스를 상속받으며, gml:_Feature, gml:_Geometry, gml:Topology, gml:TimeObject, gml:srsName 클래스들은 gml:_GML을 상속받는다.

그러나 TTA에서 제안한 모바일 서비스용 GML 프로파일은 GML 3.0을 기반으로 하고 있어서 객체들은 식별자를 필수적으로 가지지 않기 때문에 외부에서 정의한 스키마의 경우 객체들 간의 식별이 모호할 수 있다. 또한, u-GIS 서비스에서 활용되는 시공간 데이터를 GML 3.0 기반으로 인코딩하면 시간 데이터와 공간 데이터 간의 매핑 작업이 추가로 필요하므로 비효율적이다.

2.2 바이너리 XML

XML[10]은 데이터를 텍스트로 저장하기 때문에 문서의 크기가 상대적으로 큰 단점을 갖고 있다. 이로 인해 네트워크를 통해 XML 문서를 주고받는 것은 다량의 트래픽을 유발시키고 상대적으로 많은 대역폭을 요구한다. 이와 같은 XML의 성능 문제를 개선시켜 줄 대안으로 XML 문서를 특정 변환 규칙에 따라 바이너리 XML로 변환하는 방법이 제안되었다[8]. 바이너리 XML은 제한된 환경에서 XML 문서에 대해 의미 손실 없이 크기를 줄여 보다 효과적인 전송을 가능하게 하고, 문서의 스캔 비용을 줄일 수 있다. 특히, OGC에서 2003년에 바이너리 XML 인코딩 명세 0.0.8[3]을 발표하였다.

바이너리 XML은 세 가지 특징을 가지고 있다. 첫 번째로 텍스트로 표현된 기존의 XML 문서에 비해 같은 내용을 담고 있지만 그 크기가 작다. 두 번째로 텍스트로 작성된 XML 문서에 대해 기본적으로 수행하는 XML 문법 검사 및 유효성 검사 과정이 불필요하기 때문에 XML 처리 속도가 빠르다. 마지막으로 데이터의 표현 방식으로 바이너리를 사용함으로써 압축 및 암호화 기능을 갖는다.

바이너리 XML 등장 이후 W3C는 바이너리 XML의 표준화를 위하여 워킹그룹인 XML Binary Characterization Working Group(이하 XBC-WG)을 구성하였다. XBC-WG에서는 XML의 단점을 분석하여 바이너리 XML의 표준화를 통해 얻을 수 있는 효과를 위해 다양한 분야의 쓰임새를 재정립하였다. 그리고 각 쓰임새의 요구 사항을 분석하고 여러 쓰임새에 걸쳐 비교함으로써 각종 속성을 도출하였다[8, 9].

바이너리 XML의 표준 중에서 가장 널리 사용되고 있는 기술은 썬 마이크로시스템즈의 Fast Infoset 기법이다[2, 7, 9, 15]. Fast Infoset 기법은 문서의 크기를 줄이고자 테이블과 인덱스 기법을 사용하고 있다. 처음 나타나는 문자열을 테이블에 저장해 놓고 같은 문자열이 나타나면 인덱스로 대체하여 XML 문서의 크기를 줄인다. 또한 종료 태그(</>)가 없어서 불필요한 XML 데이터 부분을 줄일 수 있고, 바이너리 콘텐츠를 포함하고 있기 때문에 Base64 문자 표현 방식으로 변환될 필요가 없다.

2.3 FUNs

FUNs는 속성 데이터, 공간 데이터, 네트워크 데이터를 저장하고 질의할 수 있는 모바일 장치용 공간 DBMS이다[1]. FUNs는 일반적인 RDBMS와 유사한 기능을 제공하며, DBMS와 응용프로그램 간의 인터페이스로는 SQL이 아닌 API의 형태를 지원한다. 그리고 데이터를 테이블 스페이스 단위의 파일로 관리하고, 테이블 스페이스에는 다수의 테이블이 포함된다.

FUNs에서는 공간 데이터 타입으로 Point,

LineString, Polygon이 지원되며 속성 데이터 검색의 경우는 일반 DBMS에서 제공하는 검색 기능 이외에 한글 초성검색 기능이 지원된다. 공간 데이터 검색으로는 Windows 검색, KNN 검색, 공간 위상 관계 연산 기능을 제공한다. 그리고 빠른 경로 검색을 위해 물리적 레코드 ID 검색을 제공하며 트랜잭션 처리를 위해 logging 기법을 이용한 회복 기능도 제공한다. 특히 신속한 데이터 접근을 위한 인덱스로는 속성 검색을 위한 B+-Tree와 공간 검색을 위한 GRID, R*-Tree를 지원한다.

그림 2는 FUNs의 전체 구조도를 보여준다.

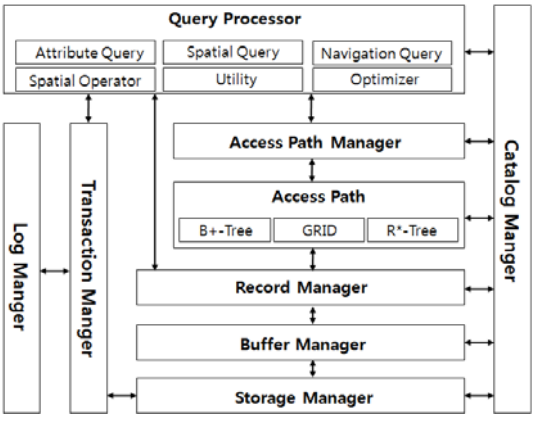


그림 2. FUNs 전체 구조도

그림 2에서 보여주는 FUNs의 주용 구성요소인 Manager와 Processor에 대한 설명은 다음과 같다.

Storage Manager는 데이터베이스를 생성 및 삭제하고 테이블 공간과 테이블을 페이지 단위로 관리하는 모듈이고, Buffer Manager는 캐시 데이터를 페이지 단위로 메모리에 적재하여 저장 공간의 I/O 속도를 향상시키는 모듈이다. Record Manager는 페이지 내의 레코드를 관리하는 모듈이고, Access Path Manager는 일반 속성 데이터를 빠르게 검색할 수 있는 B+-Tree 인덱스와 2차원 공간 데이터 검색을 위한 GRID 인덱스와 R*-Tree 인덱스를 제공하는 모듈이다. Catalog Manager는 데이터베이스의 테이블 공간과 테이블, 인덱스 등의 메타데이터를 관리하는 모듈이고, Transaction Manger와 Log Manager는 트랜잭션에 관한 장애와 복구를 관리하는 모듈이다. 마지막으로 Query Processor는 데이터베이스의 생성 및 삭제, 테이블 공간 할당, 기록, 검색, 삭제 등의 질의 처리를 위한 API를 제공하는 모듈이다.

3. 시스템 설계

본 장에서는 모바일 환경을 위한 GML 기반 시공간 질의 처리 시스템의 전체 구조를 설명하고, 시스템을 구성하는 GML 인터페이스 관리자, GML 질의 처리 관리자, GML 저장 관리자에 대해 각각의 모듈별로 설명한다.

3.1 시스템 전체 구조

모바일 환경을 위한 GML 기반 시공간 질의 처리 시스템은 모바일 서비스용 GML 프로파일을 따르는 GML 문서의 시공간 데이터에 대한 저장/검색/갱신/삭제 기능을 제공한다. 그림 3은 시스템의 전체 구조를 보여준다.

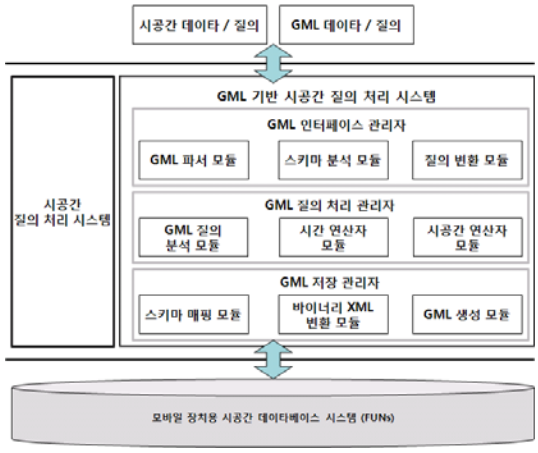


그림 3. GML 기반 시공간 질의 처리 시스템 구조

그림 3에서 보는 바와 같이 모바일 환경을 위한 GML 기반 시공간 질의 처리 시스템은 크게 GML 인터페이스 관리자, GML 질의 처리 관리자, GML 저장 관리자로 구성되며, 공간 질의만 가능한 기존의 FUNs[1]상에서 개발되었다.

GML 인터페이스 관리자는 사용자로부터 GML 파일을 입력받아 속성 또는 엘리먼트들의 파싱을 수행하는 GML 파서 모듈, 입력된 XSD와 파싱된 GML 문서의 스키마를 분석하여 테이블을 생성하는 스키마 분석 모듈, 사용자로부터 입력받은 질의를 GML 데이터 질의를 위한 형태로 변환하는 질의 변환 모듈로 구성된다.

GML 질의 처리 관리자는 사용자로부터 입력된 질의의 분석을 통해 오류를 검사하고 실행가능 여부를 검사하는 GML 질의 분석 모듈, GML 데이터에 대한 검색/갱신/삭제 수행을 위한 시간 연산자 모듈과 시공간 연산자 모듈로 구성된다.

GML 저장 관리자는 스키마 분석 모듈에 의해 생성된 테이블의 컬럼에 GML 데이터를 저장하는 스키마 매핑 모듈, GML 문서를 바이너리 형태로 저장하기 위해 인코딩을 수행하는 바이너리 XML 변환 모듈, 사용자 질의의 결과를 GML 형태로 생성하는 GML 생성 모듈로 구성된다.

3.2 GML 질의 처리 관리자

본 절에서는 GML 질의 처리 관리자를 구성하는 GML 질의 분석 모듈, 시간 연산자 모듈, 시공간 연산자 모듈에 대해 설명한다. 이러한 GML 질의 처리 관리자의 주요 모듈은 국내 표준화 기구인 TTA에서 제시한 모바일 GML 프로바일에 시공간 데이터를 확장하였고 국제 표준화 기구인 OGC에서 제시한 공간 연산자 관련 표준을 기반으로 시공간 연산자를 확장하여 적용하였기 때문에 상호운용성을 제공할 수 있고 추후 모바일상에서 시공간 데이터 및 시공간 연산자 사용시 가이드라인을 제공할 수 있다.

3.2.1. GML 질의 분석 모듈

GML 질의 분석 모듈은 GML 인터페이스 관리자의 질의 변환 모듈로부터 변환된 질의의 구문 분석을 통해 오류를 검사하고 실행가능 여부를 판단한다. GML 질의 분석 모듈이 변환된 질의가 올바른지 판단하기 위하여 해당 질의에 대해서 파싱을 수행하여 파스트리를 생성하고, 생성된 파스트리가 시간 및 시공간 SQL에 정의된 구문에 대해 적합하면 질의를 수행하고, 적합하지 않으면 질의를 수행하지 않는다.

3.2.2 시간 연산자 모듈

시간 연산자는 시간 특성에 맞게 타임스탬프(Timestamp) 연산자와 인터벌(Interval) 연산자로 구분된다. 또한, 시간 연산자는 연산에 대한 결과값으로 True 혹은 False를 반환하는 시간 관계 연산자와 인터벌을 반환하는 시간 분석 연산자로 구분된다[12]. 표 1은 시간 연산자 모듈이 지원하는 타임스탬프 및 인터벌 질의를 위한 시간 관계 연산자를 보여준다.

표 1. 타임스탬프 및 인터벌 질의를 위한 시간 관계 연산자

타임스탬프 시간 관계 연산자	설 명
T_Before (ST_Geometry, time)	시간 time이 ST_Geometry의 time에 선행하는지 여부 반환
T_After (ST_Geometry, time)	시간 time이 ST_Geometry의 time에 후행하는지 여부 반환
T_Equals (ST_Geometry, time)	시간 time과 ST_Geometry의 time이 같은지 여부 반환
인터벌 시간 관계 연산자	설 명
T_Contains (ST_Geometry, stTime, edTime)	인터벌 stTime와 edTime 사이에 ST_Geometry의 time을 포함하는지 여부 반환
T_Disjoint (ST_Geometry, stTime, edTime)	인터벌 stTime와 edTime 사이에 ST_Geometry의 time과 포함되지 않는지 여부 반환

표 1에서 보는 바와 같이 타임스탬프 및 인터벌 시간 관계 연산자는 입력값으로 ST_Geometry, time 혹은 ST_Geometry, stTime, edTime을 입력받고, 특정 시간 time 혹은 특정 인터벌 [stTime, edTime]을 기준으로 시간 연산을 수행한다. 또한 각 연산자에 대한 결과값으로는 True 혹은 False를 반환한다. 표 2는 인터벌 질의를 위한 시간 분석 연산자를 보여준다.

표 2. 인터벌 질의를 위한 시간 분석 연산자

인터벌 시간 분석 연산자	설 명
T_Union (ST_Geometry, stTime, edTime)	인터벌 stTime와 edTime 사이에 ST_Geometry의 time과의 합집합을 반환
T_Intersection (ST_Geometry, stTime, edTime)	인터벌 stTime와 edTime 사이에 ST_Geometry의 time과 교집합을 반환
T_Difference (ST_Geometry, stTime, edTime)	인터벌 stTime와 edTime 사이에 ST_Geometry의 time과 차집합을 반환

표 2에서 보는 바와 같이 인터벌 질의를 위한 시간 분석 연산자는 입력값으로 ST_Geometry, stTime, edTime을 입력받고 결과값으로 시간값을 반환한다.

3.2.3 시공간 연산자 모듈

시공간 연산자 모듈은 OGC에서 제안한 “Simple Features Specification for SQL”[5]에서 명시하는 공간 연산자를 기반으로 확장하여 시공간 연산자를 제공한다. 시공간 연산자는 시간 특성에 맞게 타임스탬프 연산자와 인터벌 연산자로 구분된다. 또한, 시공간 연산자는 결과값으로 True 혹은 False를 반환하는 시공간 관계 연산자와 시공간 객체를 반환하는 시공간 분석 연산자로 구분된다.

모바일 환경을 위한 GML 기반 시공간 질의 처리 시스템은 시공간 데이터에 대한 삽입/삭제/갱신/검색 연산의 수행을 위해 다양한 시공간 연산자를 제공한다. 표 3은 타임스탬프 및 인터벌 질의를 위한 시공간 관계 연산자를 보여준다.

표 3. 타임스탬프 및 인터벌 질의를 지원하는 시공간 관계 연산자

타임스탬프 시공간 관계 연산자	설 명
ST_Equals (ST_Geometry1, ST_Geometry2, time)	시간 time에 ST_Geometry1과 ST_Geometry2가 같은지 여부 반환
ST_Disjoint (ST_Geometry1, ST_Geometry2, time)	시간 time에 ST_Geometry1과 ST_Geometry2가 만나지 않는지 여부 반환
ST_Touches (ST_Geometry1, ST_Geometry2, time)	시간 time에 ST_Geometry1과 ST_Geometry2의 경계가 만나는지 여부 반환
ST_Overlaps (ST_Geometry1, ST_Geometry2, time)	시간 time에 ST_Geometry1과 ST_Geometry2가 겹치는지 여부 반환
ST_Crosses (ST_Geometry1, ST_Geometry2, time)	시간 time에 ST_Geometry1과 ST_Geometry2가 교차하는지 여부 반환
ST_Contains (ST_Geometry1, ST_Geometry2, time)	시간 time에 ST_Geometry1이 ST_Geometry2를 포함하는지 여부 반환
인터벌 시공간 관계 연산자	설 명
ST_Disjoint (ST_Geometry1, ST_Geometry2, stTime, edTime)	인터벌 stTime와 edTime 사이에 ST_Geometry1과 ST_Geometry2가 만나지 않는지 여부 반환
ST_Touches (ST_Geometry1, ST_Geometry2, stTime, edTime)	인터벌 stTime와 edTime 사이에 ST_Geometry1과 ST_Geometry2의 경계가 만나는지 여부 반환
ST_Overlaps (ST_Geometry1, ST_Geometry2, stTime, edTime)	인터벌 stTime와 edTime 사이에 ST_Geometry1과 ST_Geometry2가 겹치는지 여부 반환
ST_Crosses (ST_Geometry1, ST_Geometry2, stTime, edTime)	인터벌 stTime와 edTime 사이에 ST_Geometry1과 ST_Geometry2가 교차하는지 여부 반환
ST_Contains (ST_Geometry1, ST_Geometry2, stTime, edTime)	인터벌 stTime와 edTime 사이에 ST_Geometry1이 ST_Geometry2를 포함하는지 여부 반환

표 3에서 보는 바와 같이 타임스탬프 및 인터벌 질의를 위한 시공간 관계 연산자는 입력값으로 두 개의 시공간 객체 ST_Geometry1, ST_Geometry2를 입력받고, 특정 시간 time 혹은 특정 인터벌 [stTime, edTime]을 기준으로 시공간 연산을 수행한다. 또한 각 연산에 대한 결과값으로는 True 혹은 False를 반환한다. 표 4는 타임스탬프 질의를 위한 시공간 분석 연산자를 보여준다.

표 4. 타임스탬프 질의를 위한 시공간 분석 연산자

타임스탬프 시공간 분석 연산자	설 명
ST_Union (ST_Geometry1, ST_Geometry1, time)	시간 time에 ST_Geometry1과 ST_Geometry2의 합집합을 반환
ST_Difference (ST_Geometry1, ST_Geometry1, time)	시간 time에 ST_Geometry1에서 ST_Geometry2의 차집합을 반환
ST_Intersection (ST_Geometry1, ST_Geometry1, time)	시간 time에 ST_Geometry1과 ST_Geometry2의 교집합을 반환
ST_Distance (ST_Geometry1, ST_Geometry1, time)	시간 time에 ST_Geometry1과 ST_Geometry2 사이의 거리를 반환

표 4에서 보는 바와 같이 타임스탬프 질의를 위한 시공간 분석 연산자는 입력값으로 두 개의 시공간 객체 ST_Geometry1, ST_Geometry2를 입력받고, 특정 시간 time을 기준으로 시공간 연산을 수행한다. ST_Union, ST_Difference, ST_Intersection는 결과값으로 시공간 Geometry 객체를 반환하고, ST_Distance는 결과값으로 가장 짧은 거리를 반환한다.

3.3 GML 인터페이스 관리자

본 절에서는 GML 인터페이스 관리자를 구성하는 GML 파서 모듈, 스키마 분석 모듈, 질의 변환 모듈에 대해서 설명한다. 이러한 GML 인터페이스 관리자의 주요 모듈은 대부분의 상용 DBMS들이 사용하는 SQL 표준을 확장하여 시공간 질의 처리를 제공하기 때문에 상호운용성을 보장할 수 있다.

3.3.1 GML 파서 모듈

GML 파서 모듈은 사용자로부터 입력된 GML 문서의 속성 또는 엘리먼트의 파싱을 수행하고, 파싱된 결과를 스키마 분석 모듈과 GML 저장 관리자의 스키마 매핑 모듈로 전달한다. 상대적으로 낮은 성능을 갖고 있는 모바일 환경에서 모든 GML 문서의 유효성 검사를 수행하는 것은 비효율적이므로 GML 파서 모듈은 사용자가 입력하는 GML 문서에 대해 모두 유효성을 만족하는 것으로 가정하여 파싱 과정을 단순화한다.

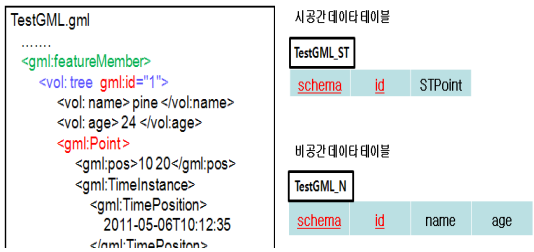
또한 본 논문에서는 XML 파서 모델 중 Pull 모델을 사용하여 GML 문서의 모든 스키마들을 파싱하지 않고 필요한 스키마만 파싱하여 GML 데이터 처리 효율을 높였다[16]. 즉, 응용 프로그램에서 파싱 요청이 발생하면 GML 파서 모듈은 XML 문서를 읽고 파싱 요청된 스키마를 토큰으로 추출하여 응용 프로그램으로 반환한다.

3.3.2 스키마 분석 모듈

스키마 분석 모듈은 GML 파서 모듈에 의해 파싱된 결과를 바탕으로 해당 GML 문서의 공간 또는 시공간 데이터를 갖고 있는 <featureMember>하위 스키마 구조를 분석하여 FUNs에서 제공하는 시공간 데이터 타입의 컬럼으로 구성된 테이블 스키마를 생성한다. GML 3.1.1 이후부터 GML 객체들은 객체들 간의 식별이 가능해야 한다. 그러므로 객체들 간의 식별을 위하여 모바일 서비스용 GML 프로파일에서 gml:_GML의 타입을 정의해 놓은 AbstractGMLType 스키마를 변경하였다. 즉, 표준 속성인 gml:id의 사용을 “optional” 에서 “required”로 변경하여 GML 객체를 생성할 때 gml:id를 반드시 사용하도록 하였다.

기존 모바일 서비스용 GML 프로파일을 이용하여 시공간 데이터의 인코딩을 하기 위해서는 공간 객체와 시간 객체를 각각 생성하고, 두 객체의 식별자를 맞춰줘야 하는 불편함이 있다. 그러므로 공간 데이터인 Point 엘리먼트를 정의해 놓은 gml:PointType 스키마를 변경하였다. 즉, gml:PointType의 엘리먼트 시퀀스 리스트에 gml:pos 외에 시간 데이터 엘리먼트인 gml:TimeInstance와 gml:TimePeriod의 사용을 “optional” 로 설정하였다.

그림 4는 <featureMember> 하위 스키마 구조를 분석하여 테이블 스키마를 생성하는 예를 보여준다.



(a) <featureMember> 하위 스키마 (b) 테이블 스키마

그림 4. 스키마 분석 예

그림 4에서 보는 바와 같이 그림 4(a)의 GML 문서의 <featureMember> 하위 스키마를 바탕으로 컬럼을 정의하여 그림 4(b)의 테이블 스키마를 생성한다. 하위 스키마를 분석할 때 공간 또는 시공간 데이터 엘리먼트 외에 다른 엘리먼트가 존재할 경우 시공간 데이터 테이블 스키마와 비공간 데이터 테이블 스키마를 각각 따로 생성한다. 테이블 스키마를 생성할 때 표준 속성인 gml:id를 갖는 “<vol:tree gml:id=“1”>” 태그의 “vol:tree”를 저장하기 위한 컬럼과 표준 속성 gml:id의 값을 저장하기 위한 컬럼을 생성하고, 또한 이 두 컬럼을 테이블 스키마의 기본키로 하여 GML 객체들의 식별이 가능하도록 한다.

3.3.3. 질의 변환 모듈

질의 변환 모듈은 사용자가 입력한 질의를 FUNs에서 실행 가능한 형태로 변환하는 기능을 제공한다. 즉, 사용자로부터 XSQL 형식으로 질의를 입력받으면, FUNs에서 사용될 수 있는 시간 및 시공간 SQL로 변환해 준다. 표 5는 GML 기반 시공간 질의 처리 시스템이 지원하는 시공간 SQL 예를 보여준다[11].

표 5에서 보는 바와 같이 시공간 SQL 예에서 검색은 시간 ‘2011/05/08/12:00:00’에 ST_Polygon(‘2011/05/07/15:30’, 10, 10, 20, 10, 20, 20, 10, 20, 10, 10)과 경계가 만나는 건물의 building_id, building_frame을 검색하는 예이며, 삽입은 building_id, building_name, building_frame이 각각 ‘9001201’, ‘새천년관’, ST_Polygon(‘2011/05/07/15:30’,10,10,20,10,20,20,10,20,10,10)인 데이터를 삽입하는 예이다. 그리고 삭제는 시간 ‘2011/09/09/12:00’에 ST_Polygon(‘2011/05/07/15:30’,10,10,20,10,20,20,

10,20,10,10)에 포함되는 건물을 삭제하는 예이며, 갱신은 building_name이 '새천년관'인 건물의 building_frame을 ST_Polygon('2011/ 05/07/15:50', 10, 10, 20, 10, 20, 20, 10, 20, 10, 10)으로 갱신하는 예이다.

표 5. 시공간 SQL 예

검색	SELECT building_id, building_frame FROM building WHERE ST_Touches(ST_Polygon('2011/05/07/15:30', 10, 10, 20, 10, 20, 20, 10, 20, 10, 10), building_frame, '2011/05/08/12:00:00');
삽입	INSERT INTO building(building_id, building_name, building_frame) VALUES(9001201, '새천년관', ST_Polygon('2011/05/07/15:30', 10, 10, 20, 10, 20, 20, 10, 20, 10, 10));
삭제	DELETE FROM building WHERE ST_Contains(ST_Polygon('2011/05/07/15:30', 10, 10, 20, 10, 20, 20, 10, 20, 10, 10), building_frame, '2011/09/09/12:00');
갱신	UPDATE building SET building_frame = ST_Polygon('2011/05/07/15:50', 10, 10, 20, 10, 20, 20, 10, 20, 10, 10) WHERE building_name = '새천년관';

표 6은 GML 기반 시공간 질의 처리 시스템이 지원하는 시간 SQL 예를 보여준다[11].

표 6. 시간 SQL 예

검색	SELECT building_name, building_frame FROM building WHERE T_Contains(building_frame, '2001/01/01/00:00:00', '2011/05/08/12:00:00');
삽입	INSERT INTO building(building_frame, building_name) VALUES(ST_Polygon('2011/05/08/12:00:00',10,10,20,10,20,20,10,20,10,10), '본관');
삭제	DELETE FROM building WHERE T_Equals(building_frame,' 2011/05/08/12:00:00');
갱신	UPDATE building SET building_frame = ST_Polygon('2009/05/07/15:50:08',10,10,20,10,20,20,10,20,10,10) WHERE T_Disjoint(building_frame, '2008/01/01/00:00:00','2008/12/31/24:00:00');

표 6에서 보는 바와 같이 시간 SQL 예에서 검색은 인터벌 '2001/01/01/00:00:00'에서 '2011/05/08/12:00:00' 사이에 building_frame의 시간이 포함되는 건물의 building_name, building_frame을 검색하는 예이며, 삽입은 building_frame이 ST_Polygon ('2011/05/08/12:00:00',10,10,20,10,20,20,10,20,10,10)이고, building_name이 '본관'인 데이터를 삽입하는 예이다. 그리고 삭제는 시간 '2011/05/08/12:00:00'과 building_frame의 시간이 일치하는 건물을 삭제하는 예이며, 갱신은 인터벌 '2008/01/01/00:00: 00'과 '2008/12/31/24:00:00' 사이에 building_frame의 시간이 포함되는 않는 건물의 building_frame을 ST_Polygon('2009/05/07/15:50:08',10,10,20,10,20,20,10,20,10,10)으로 갱신하는 예이다.

3.4 GML 저장 관리자

본 절에서는 GML 저장 관리자를 구성하는 스키마 매핑 모듈, 바이너리 XML 변환 모듈, GML 생성 모듈에 대해 설명한다. 이러한 GML 저장 관리자의 주요 모듈은 모바일 장치의 낮은 저장 공간을 고려하여 대용량의 시공간 데이터를 바이너리 형태로 저장 및 관리하기 때문에 질의 처리시 높은 성능 향상을 가져올 수 있다.

3.4.1 스키마 매핑 모듈

스키마 매핑 모듈은 GML 인터페이스 관리자의 스키마 분석 모듈에서 생성된 테이블에 GML 문서의 아이템들을 저장하는 기능을 제공한다. 그림 5는 그림 4(b)의 테이블 스키마에 맞게 GML 문서의 데이터가 저장된 예를 보여준다.

시공간 데이타 테이블

TestGML_ST

schema	id	STPoint
vol:tree	1	Stpoint('2011-05-06T10:12:35',10,20)

비공간 데이타 테이블

TestGML_N

schema	id	Name	age
vol:tree	1	pine	24

그림 5. GML 데이터 저장 예

그림 5에서 보는 바와 같이 GML의 각 요소의 내용들은 그림 4(b)의 테이블 스키마의 해당 컬럼에 맞춰 저장된다. 특히 시공간 데이터의 경우 FUNs에서 제공하는 데이터 타입으로 변환되어 저장된다.

시공간 데이터 테이블의 경우 스키마 정의 파일인 XSD 단위로 저장되므로, GML 문서들을 하나의 테이블에서 관리가 가능하다. 그러므로 시공간 데이터 테이블에서는 여러 GML 문서들을 대상으로 질의가 가능하다. 이와 다르게, 비공간 데이터 테이블은 GML 문서 단위로 저장되므로, 하나의 테이블에 한 GML 문서의 데이터들만이 저장된다. 또한 질의에 따라 시공간 데이터 테이블과 비공간 데이터 테이블의 조인을 사용하여 결과값을 얻을 수 있다.

3.4.2 바이너리 XML 변환 모듈

바이너리 XML 변환 모듈은 GML 문서를 XML 바이너리 형태로 저장하기 위해 인코딩을 수행하는 기능을 제공한다[8]. 본 논문에서는 XML 바이너리 형태로 변환하기 위해 바이너리 XML 표준으로 채택된 Fast Infoset[2, 7, 9]을 사용하여 인코딩을 한다.

그림 6은 XML 문서와 변환된 Fast Infoset 문서의 예를 보여준다.

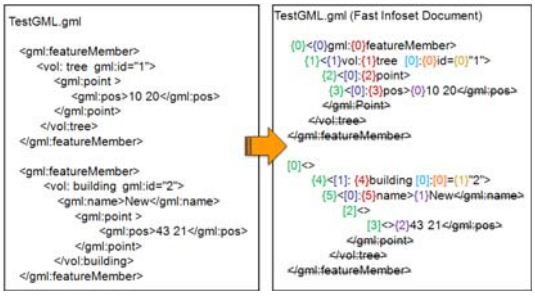


그림 6. XML 문서와 Fast Infoset 문서 예

그림 6에서 보는 바와 같이 Fast Infoset 문서는 XML 문서에서 종료 태그(</>)를 없애고, 문자열의 인덱스를 통해 반복되는 공통 문자열의 치환을 통해 문서의 크기를 감소시킨다. 그리고 XML 문서의 네임스페이스, 접두사, 지역이름, 속성이름, 속성값, 일반문자는 각각 인덱스된다. 또한 엘리먼트나 속성 앞에 '{}'을 사용하여 문자열을 인덱스하여 문자열 테이블에 등록하고, '[]'을 사용하여 반복되는 문자열을 치환한다.

3.4.3 GML 생성 모듈

GML 생성 모듈은 사용자 질의의 결과를 GML 문서로 변환하거나 FUNs에 저장된 데이터를 GML 문서로 변환하는 기능을 제공한다. GML 생성 모듈은 사용자로부터 입력된 SELECT 질의의 결과로 반환되는 ResultSet 인스턴스를 GML 형태로 변환하여 반환한다. 그림 7은 ResultSet의 클래스 정의를 보여준다.

```
class CResultSet
{
public:
    CResultSet() {}
    virtual ~CResultSet() {}

public:


    virtual int getCount() = 0;
    virtual bool next() = 0;
    virtual bool previous() = 0;
    virtual bool first() = 0;
    virtual bool last() = 0;
    virtual bool beforeFirst() = 0;
    virtual bool afterLast() = 0;
    virtual bool relative(int rows) = 0;
    virtual bool absolute(int rows) = 0;
    virtual CResultSetMetaData* getMetaData() = 0;


    */
    virtual char* getWKBPoint(STRING columnName, char* buffer, int &bufsize) = 0;
    virtual char* getWKBPoint(STRING columnName, char* buffer, int &bufsize, bool& isNULL) = 0;

    //! WKBPoint형 컬럼 값 획득
    /*!
        주어진 컬럼 Index에 해당하는 WKBPoint 값을 획득한다.

        \param  [in]columnIndex 획득할 컬럼의 Index
        \param  [in]buffer      값을 저장할 buffer (NULL일 경우 내부에서 메모리가 할당되어 리턴됨)
        \param  [out]bufSize    버퍼에 저장된 크기
        \param  [out]isNULL     값이 널인지?
        \return char*           획득된 WKBPoint byte stream
    */
    virtual char* getWKBPoint(int columnIndex, char* buffer, int &bufsize) = 0;
    virtual char* getWKBPoint(int columnIndex, char* buffer, int &bufsize, bool& isNULL) = 0;
```

그림 7. ResultSet 클래스 정의

그림 7에서 보는 바와 같이 ResultSet 멤버 함수들을 통해 결과로 반환된 ResultSet 인스턴스의 데이터를 추출할 수 있다. 또한 ResultSet 인스턴스의 멤버 함수인 getWKBPoint() 등을 이용하여 질의의 결과로 반환된 시공간 데이터를 추출하여 GML 문서 형태로 변환할 수 있다.

4. 시스템 구현

본 장에서는 시스템의 구현 환경을 살펴보고, 모바일 환경을 위한 GML 기반 시공간 질의 처리 시스템의 구현 내용과 주요 기능에 대하여 설명한다. 또한, GML 기반 시공간 질의 처리 시스템의 효용성을 검증하기 위하여 시설물 관리 서비스에 가상 시나리오를 적용한 내용에 대해서 기술한다.

4.1 구현 환경

본 논문에서는 GML 기반 시공간 질의 처리 시스템을 구현하기 위해 운영체제는 Microsoft Windows XP Professional ServicePack3를 사용하였고, 개발 도구는 Microsoft Visual Studio 2005를 사용하였으며, 언어는 C++를 사용하였다. 또한 SDK로는 Window Mobile 5.0 Pocket PC SDK를 사용하였다. 그리고 가상 시나리오의 적용을 위해 Intel PXA 270 520MHz, RAM 128MB, Microsoft

Windows Mobile 5.0을 운영체제로 하는 블루버드 503 PDA를 사용하였다.

4.2 가상 시나리오

본 논문에서 구현한 GML 기반 시공간 질의 처리 시스템의 효용성을 검증하기 위해 다양한 모바일 u-GIS 서비스 중 시설물 관리 서비스를 가상 시나리오로서 적용해 보았다. 시설물 관리 서비스에서는 사용자가 건물을 비롯한 다양한 시설물들의 위치 정보, 상태 정보 등을 검색하여 효율적으로 시설물을 관리할 수 있도록 도와주는 서비스다.

4.2.1 시설물 데이터 삽입

시설물 데이터를 삽입하기 위하여 각각의 GML 문서를 선택하여 삽입할 수 있다. 그림 8은 시설물 데이터를 삽입하는 예를 보여준다.

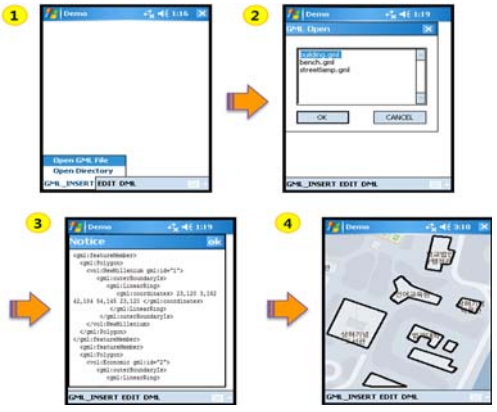


그림 8. 시설물 데이터 삽입

그림 8에서 보는 바와 같이 시설물 별로 작성된 GML 문서를 삽입하는 과정을 보여준다. ①에서는 Open GML File 메뉴를 이용하여 단일 파일을 선택할 수 있다. ②에서는 Open GML File 메뉴를 이용하여 여러 시설물 중 building.gml 문서를 선택하여 삽입한다. ③에서는 삽입될 building.gml 문서의 내용을 보여준다. ④에서는 building.gml 문서를 통해 삽입된 건물 데이터의 위치 정보를 보여준다.

4.2.2 시설물 데이터 검색

시설물 데이터를 검색하기 위하여 검색하고자 하는 시공간 영역을 가지고 있는 GML 문서를 입력받아 시설물의 데이터를 검색할 수 있다. 그림 9는 시설물 데이터를 검색하는 예를 보여준다.

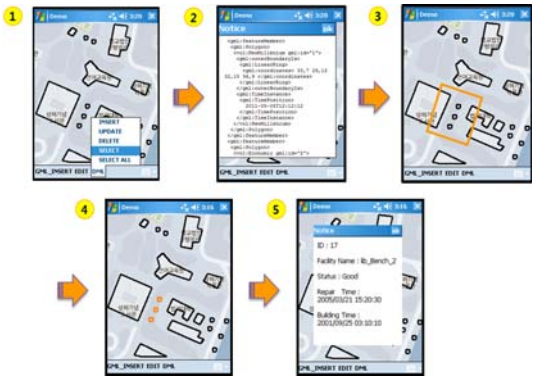


그림 9. 시설물 데이터 검색

그림 9에서 보는 바와 같이 도서관 근처의 벤치 데이터를 검색하는 과정을 보여준다. ①에서는 SELECT 메뉴를 이용하여 검색하고자 하는 GML 문서를 선택한다. ②에서는 검색하고자 하는 시설물의 범위, 시간 등이 저장된 GML 문서의 내용을 보여준다. ③에서는 GML 문서에서 읽어온 검색 질의 범위를 화면에 표시해 준다. ④에서는 검색된 시설물들의 위치 정보를 보여주고, ⑤에서는 검색된 시설물에 대한 데이터를 보여준다.

4.2.3 시설물 데이터 갱신

시설물의 위치 데이터를 갱신하기 위하여 갱신할 데이터를 가지고 있는 GML 문서를 입력받아 해당되는 시설물의 위치 데이터를 갱신할 수 있다. 그림 10은 시설물의 위치 데이터를 갱신하는 예를 보여준다.

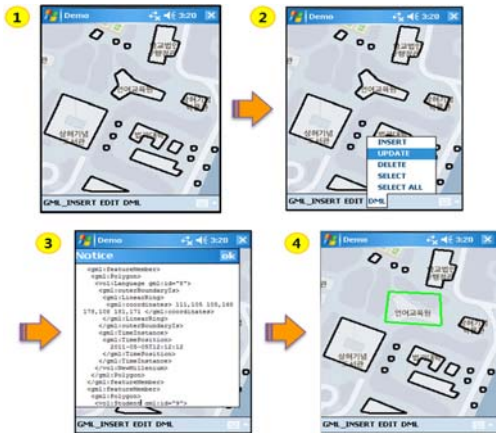


그림 10. 시설물 데이터 갱신

그림 10에서 보는 바와 같이 시설물의 형태가 변경되어 시설물의 위치, 시간 등의 데이터가 갱신되는 과정을 보여준다. ①에서는 갱신전의 시설물 데이터를 보여주고, ②에서는 UPDATE 메뉴를 이용하여 갱신하고자 하는 GML 문서를 선택한다. ③에서는 갱신하고자 하는 시설물의 범위, 시간 등이 저장된 GML 문서의 내용을 보여준다. ④에서는 위치 데이터의 갱신이 완료된 시설물을 보여준다.

4.2.4 시설물 데이터 삭제

시설물 데이터를 삭제하기 위하여 삭제할 시설물의 데이터를 가지고 있는 GML 문서를 입력받아 시설물 데이터를 삭제할 수 있다. 그림 11은 시설물 데이터를 삭제하는 예를 보여준다.

그림 11. 시설물 데이터 삭제

그림 11에서 보는 바와 같이 시설물 데이터를 삭제하는 과정을 보여준다. ①에서는 삭제전의 데이터를 보여주고 DELETE 메뉴를 이용하여 삭제하고자 하는 GML 문서를 선택한다. ②에서는 삭제하고자 하는 시설물의 범위, 시간 등이 저장된 GML 문서의 내용을 보여준다. ③에서는 해당 시설물 데이터가 삭제된 화면을 보여준다.

5. 결 론

본 논문에서는 모바일 환경에서 대용량의 시공간 데이터를 효율적으로 처리할 수 있는 GML 기반 시공간 질의 처리 시스템을 설계 및 구현하였다. 모바일 환경을 위한 GML 기반 시공간 질의 처리 시스템은 시공간 데이터에 대한 신속한 질의 처리를 위하여 시간 및 시공간 연산자를 지원하고, 대용량의 시공간 데이터를 효율적으로 저장하기 위하여 GML 문서를 바이너리 XML 형태로 변환하여 모바일 DBMS에 저장하는 기능을 제공한다. 또한 본 논문에서 설계 및 구현한 모바일 환경을 위한 GML 기반 시공간 질의 처리 시스템을 시설물 관리 서비스에 대한 가상 시나리오를 적용함으로써 본 시스템이 u-GIS 서비스에서 활용될 수 있음을 입증하였다.

참 고 문 헌

[1] ETRI, 2008, FUNs Specification (Core) - (Flash-aware Ubiquitous Navigation system), Version: 1.0.

[2] ITU-T, 2005, Rec. X891 | ISO/IEC, 24824-1, Fast Infoset.

[3] OpenGIS Consortium, Inc, 2003, Binary XML(BXML) Encoding Specification 0.0.8.

[4] Open Geospatial Consortium, Inc, 2007, Geography Markup Language (GML) Encoding Standard Version 3.2.1.

[5] Open Geospatial Consortium, Inc, 2010, OpenGIS Implementation Specification for Geographic Information-Simple Feature Access-Part 2:SQL Option, Version 1.2.1.

[6] Oracle Corporation, 2010, Oracle XML DB Developer's Guide 11g Release2 (11.2).

[7] Sun Developer Network(SDN), Fast Infoset, http://java.sun.com/developer/technical-Articles/xml/fastinfoset.

[8] Wireless Application Protocol Forum, Ltd, 2001, Binary XML Content Format Specification 1.3.

[9] W3Consortium, 2004, XML Information Set (Second Edition).

[10] W3Consortium, 2008, Extensible Markup Language (XML) 1.0 (Fifth Edition).

[11] 김정준, 양형식, 신인수, 한기준, 2011, "모바일 장치용 시공간 질의 처리 시스템의 설계 및 구현," 한국공간정보학회 춘계학술대회 논문집, pp. 189-191.

[12] 김정준, 정연종, 김동오, 한기준, 2009, "유비쿼터스 환경을 위한 시공간 미들웨어의 설계 및 구현," 한국공간정보시스템학회 논문지, 제11권, 제1호, pp.44-55.

[13] 박기호, 이양원, 안재성, 2005, "시간지리학 응용을 위한 시공간데이터베이스 기반의 GIS 컴퓨팅 연구," 한국GIS학회 논문지, 제13권, 제3호, pp

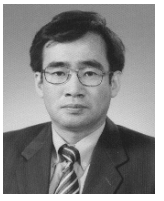
221-237.
[14] 시종익, 2003, “기존 GIS DB를 활용한 모바일 서비스용 GIS DB 구축 지침 연구,” 한국전산원, pp.27-31.
[15] 유성재, 최일선, 윤화묵, 안병호, 정회경, 2006, “Fast infoset을 이용한 Binary XML Encoder의 설계 및 구현,” 한국해양정보통신학회 종합학술대회 논문집, pp.943-946.
[16] 장주현, 노희영, 2004, “모바일 장치를 위한 XML 파서의 설계,” 한국정보과학회 춘계학술대회논문집, pp.826-828.
[17] 한국전산원, 2004, 모바일 서비스용 GML 엔코딩 및 응용스키마 표준 개발.
[18] 한국정보통신기술협회, 2005, 모바일 서비스용 GML 프로파일.

논문접수 : 2011.11.17
수 정 일 : 1차 2011.03.24 / 2차 2011.05.27
심사완료 : 2012.06.25

김 정 준
2003년 건국대학교 컴퓨터공학 공학사
2005년 건국대학교 대학원 공학석사
2010년 건국대학교 대학원 공학박사
2010년~현재 건국대학교 컴퓨터공학부 강의교수
관심분야는 공간 데이타베이스, 시공간 데이타베이스, GIS, LBS, 텔레매틱스, USN, Semantic Web

신 인 수
2006년 건국대학교 컴퓨터공학 공학사
2008년 건국대학교 대학원 공학석사
2008년~현재 건국대학교 컴퓨터공학 박사과정
관심분야는 시공간 데이타베이스, 모바일 데이타베이스, Geo Semantic Web

원 승 호
2009년 건국대학교 컴퓨터공학 공학사
2011년 건국대학교 대학원 공학석사
2011년~현재 동양증권 그룹서비스본부 금융서비스팀 사원
관심분야는 시공간 데이타베이스, 모바일 데이타베이스, GIS



이 기 영
1984년 숭실대학교 전자계산학과(공학사)
1988년 건국대학교 대학원 컴퓨터공학과(공학석사)
2005년 건국대학교 대학원 컴퓨터공학과(공학박사)
1996년~1998년 한국컴퓨터정보학회 이사 및 서울동부지회장
1991년~현재 을지대학교 의료IT마케팅학과 교수
관심분야는 GIS, LBS, USN, 텔레매틱스

한 기 준
1979년 서울대학교 수학교육학 이학사
1981년 한국과학기술원(KAIST) 전산학과 공학석사
1985년 한국과학기술원(KAIST) 전산학과 공학박사
1985년~현재 건국대학교 컴퓨터공학부 교수
1990년 Stanford 대학 전산학과 Visiting Scholar
2000년~2002년 한국정보과학회 데이타베이스 연구회 운영위원장
2004년~2006년 한국공간정보시스템학회 회장
2004년~2008년 한국정보시스템감리사협회 회장
관심분야는 공간 데이타베이스, GIS, LBS, 텔레매틱스, 정보시스템감리