상호운용을 지원하는 코바 기반 공간 데이타 제공자의 설계 및 구현†

Implementation of CORBA based Spatial Data Provider for Interoperability

김민석*, 안경환**, 홍봉희***
Min-Seok Kim, Kyoung-Hwan An, Bong-Hee Hong

요 약 CORBA와 같은 분산 컴퓨팅 환경에서 이종의 시스템 및 데이타베이스들의 통합을 위해 기존에는 포장자(wrapper)를 사용하였다. 이러한 여러 분야에 적용되는 포장자들 중에는 클라이언트에게 공통의 인터페이스를 제공하는 공간 데이타 제공자가 있다. 기존에는 공간 데이타 제공자를 데이타 소스별로 작성해야 하기 때문에 중복 구현과 구현 후의 변경 확장에 대한 문제가 있다. 이 논문에서는 공간 데이타 제공자의 재사용과 확장성을 위해 독립 포장자 객체와 종속 포장자 객체의 두개의 층으로 이루어진 새로운 구조를 제시한다. 종속 포장자 객체는 각 데이타소스에 대한 공간 데이타 작성시에 새로이 작성되야 하는 객체이고, 독립 포장자 객체는 재사용될 수 있는 객체를 말한다. 또한 이 논문에서는 미들웨어에서 질의 처리 결과 유지 방법에 따른 구현 방법론을 제시한다. 질의 결과 유지 방법에는 질의 결과를 데이타로 유지하는 방법과 CORBA 객체로 변환하여 유지하는 방법으로 나뉘어진다. 각 방법에 대한 성능 평가 결과 CORBA 객체를 생성하는 방법의 비용이 상당히 높은 것을 알 수 있다.

ABSTRACT In distributed computing platforms like CORBA, wrappers are used to integrate heterogeneous systems or databases. A spatial data provider is one of the wrappers because it provides clients with uniform access interfaces to diverse data sources. The individual implementation of spatial data providers for each of different data sources is not efficient because of redundant coding of the wrapper modules. This paper presents a new architecture of the spatial data provider which consists of two layered objects : independent wrapper components and dependent wrapper components. Independent wrapper components would be reused for implementing a new data provider for a new data source, which dependent wrapper components should be newly coded for every data source. This paper furthermore discussed the issues of implementing the representation of query results in the middleware. There are two methods of keeping query results in the middleware. One is to keep query results as non-CORBA objects and the other is to transform query results into CORBA objects. The evaluation of the above two methods shows that the cost of making CORBA objects is very expensive.

키워드 : 상호운용, 개방형, CORBA, OpenGIS, 공간데이타제공자

1. 서 론

오늘날 기업 및 지역 자치 단체에 구축된 공간 데이타베이스 시스템이나 정보 시스템들은 서로 다른 GIS 소프트웨어 사용과 구축방법의 상이함으로 이질성과 다양성을 띄고 있으며 새로운 공간 데이타베이스의 구축은 막대한 비용을 요구하고 있다. 따라서 기존에 구축된 데이타 소스들의 상호운용 지원이 필요하다. 상호운용은 표준화된 지리 데이타의 접근, 교환, 분산 지리정보처리를 수행함으로써 데이타의 공유 뿐만 아

† 본 연구는 정통부의 대학기초연구지원사업의 연구비 지원으로 이루어졌음
* 씨엔이 드림소프트 연구원
** 부산대학교 컴퓨터공학과
*** 부산대학교 컴퓨터공학과 교수 {mskm, khan, bhhong}@hyowon.cc.pusan.ac.kr

니라 표준 인터페이스를 통한 지리정보 처리 서비스의 공유까지 가능하게 해주는 것이다[1].

상호운용을 지원하기 위해서는 서로 다른 데이타 소스들의 지리 데이타 표현에 대한 데이타 접근 모델을 외부 응용 프로그램이 접근할 수 있도록 표준화된 데이타 접근 모델로 변환해야 한다. GIS분야에서 이러한 요구를 수용할 수 있는 표준 데이타 접근 모델로는 OGC의 OpenGIS Simple Feature 모델이 있다. 표준 데이타 접근 모델을 제공하기 위해 기존 데이타 접근 모델의 데이타와 인터페이스 변환이 필요한데 이를 위해 필요한 소프트웨어 모듈로 포장자가 있다[8]. 포장 기술의 사용분야로는 시스템간의 인터페이스의 전환, 데이타 구조의 전환, 그리고 데이타 중개자 등이 있다. 이러한 여러 분야에 적용되는 포장자 중에 데이타 중개자로써 데이타를 일관된 공통의 인터페이스로 제공하는 소프트웨어 모듈을 데이타 제공자로 정의한다. 공간 데이타 제공자는 데이타 제공자의 지리데이타를 제공하기위한 확장된 형태이며 포장자의 한 형태이다.

기존 방법의 공간 데이타 제공자는 상호운용을 위해 데이타 소스별로 작성되고 해당 데이타 소스에서만 운용되기 때문에 데이타 소스의 한계를 그대로 물려 받는다. 즉, GIS 데이타 소스가 교체 되거나 추가 될 때마다 데이타 제공자가 구축되어야 한다. 이로 인해 각 데이타 소스의 데이타 제공자의 개발은 중복된 비용과 개발 부담을 개발자들에게 주고 있으며 개발 후 변경 및 확장이 어려운 문제가 있다. 이러한 문제를 해결하기 위해 기존 방법의 구성요소를 재사용하는 방향으로 공간 데이타 제공자를 제시한다.

본 논문에서는 공간 데이타 제공자의 설계 방법으로 기존 공간 데이타 제공자의 구현 비용 감소를 위한 방법으로 접근한다. 공간 데이타 제공자는 기존 공간 데이타 제공자의 구성 요소에서 재사용이 가능한 객체들을 컴포넌트로 생성하고 질의처리 서비스를 제공하기 위한 데이타 소스에 종속된 객체들을 코바의 인터페이스로 정의하여 개별적으로 구현함으로써 데이타 소스의 추가를 용이하게 한다. 제시한 공간 데이타 제공자의 기본 설계를 바탕으로 구현방법에 따라 성능에 많은 영향을 주는 질의결과의 구성 및 접근 방법에 관한 구현 기술을 제시한다. 그리고 구현 방법에 따른 성능평가와 비교 분석을 실시하여 최적의 구현 기술을 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 이 논문의 관련 연구를 기술하고 3장에서는 공간 데이타 제공자 설계를 위한 기존 방법을 제시하여 비교분석하고 4장에서는 3장에서 비교 분석한 결과를 기반으로 공간 데이타 제공자를 포함한 전체구조와 구성요소의 세부 구조를 기술한다. 5장에서는 기본 설계에 따른 구현 방법들의 성능평가를 실시하여 최적의 구현기술을 제시하고 6장에서는 결론 및 향후 연구 방향에 대해서 기술한다.

2. 관련연구

2.1 객체 포장자(Object Wrapper)

포장자는 기존 시스템의 재사용을 위해서 또는 소프트웨어 모듈, 객체간의 인터페이스 불일치를 해소, 데이타 구조의 전환 등을 위해서 작성되는 소프트웨어 모듈이다. 포장자가 적용되는 분야 중에 미들웨어 기능을 위한 방법으로써 데이타 포맷의 변환이나 질의어 변환 등의 역할을 하는 데이타 중개자 구성 방법을 본 논문에서는 공간 데이타 제공자에 적용한다.

포장자의 특정 형태로서 데이타 구조 전환에 중점적으로 사용되는 객체 포장자가 있다. 객체 포장자는 포장자를 기반으로 하며 그림1과 같이 구조적 구현 방법을 적용한 시스템을 객체지향적 시스템으로 전환하거나 서로 다른 시스템간의 데이타 모델이나 기능을 객체지향 방법으로 전환하는데 사용되는 소프트웨어 공학 기술이다[17]. 그림 1은 포장자가 클라이언트와 서버의 서로 다른 데이타 접근 모델을 변환해서 클라이언트에게 서비스를 제공하는 것을 보여준다. 객체 포장자는 기존 시스템의 특성을 객체지향 기반으로 포장하여 객체의 맴버 함수와 데이타 같은 인터페이스를 외부에 제공하는 기술을 가지고 있으며 공간 데이타 제공자는 이러한 기술을 기반으로 하고 있다. 하지만 본 논문에서 제시할려는 공간 데이타 제공자와 다르게 객체 포장자는 해당 시스템에 맞게 개발되고 운용되기

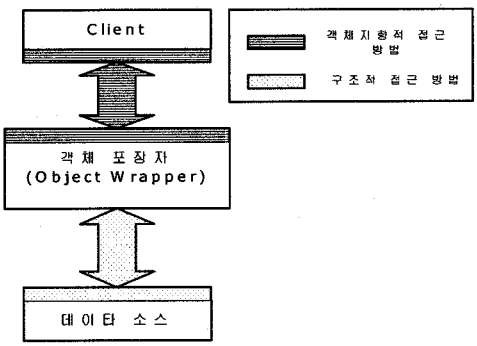

그림 1. 포장자의 사용 예[1]

때문에 다른 용도로 사용될 수 없으며 시스템간에 객체 포장자라는 레이어가 추가되므로 성능에도 영향을 줄 수가 있다.

2.2 공간 데이타 제공자

최근에는 코바를 이용하여 DBMS들을 포함한 여러 데이타 소스들을 통합 및 연동하려는 연구가 활발히 이루어지고 있다[4]. 그렇지만 기존의 연구들은 표준 인터페이스를 이용하지 않는 구조로 다양한 시스템간의 상호운용을 지원하지 못한다. 이러한 문제점을 극복하기 위해서는 상호운용 지리정보처리를 위한 개방형 표준으로 자리잡고 있는 OpenGIS(Open Geodata Interoperability Specification)의 이용이 필요하다.

OpenGIS는 공간 데이타와 공간연산에 대한 분산 접근을 위한 소프트웨어 기본설계의 포괄적 명세이며, OpenGIS에 부합하는 소프트웨어간에 상호운용이 가능하도록 해주는 공통된 표준 인터페이스 명세이다[1]. OpenGIS는 상호운용의 기본이 되는 공통적인 기본지리 정보타입 및 지리정보를 접근, 관리, 표현 공유할 수 있도록 서비스 명세를 정의하고 있으며 지리정보의 생산자(provider)와 소비자(consumer)간의 공간정보유통 방법을 제시하고 있다. 또한 OGC (OpenGIS Consortium)에서는 이를 구현하기 위한 기술로서 OLE/COM, CORBA, SQL(ODBC)을 이용한 구현 명세를 내놓고있다[2, 3]. 이들 구현 명세를 보면 OpenGIS 추상명세의 부분집합으로 이루어져 있으며, IDL(Interface Definition Language)과 같은 인터페이스 기술언어로서 표준 인터페이스를 정의하고 있다.

현재 OpenGIS의 코바 구현 명세를 따르는 공간 데이타 제공자에 관한 연구들로는 [18, 19, 20]이 있다. 그러나 이러한 논문들은 모두 다음과 같은 특징을 갖는다. 첫째 공간 데이타 제공자 계층에 해당하는 코바 객체들이 데이타 소스에 종속적이다. 만약 하부의 데이타 소스가 다른 데이타 소스로 교체될 때 공간 데이타 제공자를 완전히 새로 구현해야 하는 부담이 존재하며 기존의 모듈을 재사용하기가 힘들어진다. 두번째로 코바의 경우 객체의 생성 및 유지 방법에 따라 성능에 많은 영향을 주는데 이를 고려하지 않고 있다. 상호 운용을 지원하는 시스템에서의 가장 큰 문제점은 성능에 관한 것으로 이에 대한 분석 및 구현방법의 제시가 반드시 필요하다. 이 논문의 공간 데이타 제공자는 이러한 기존의 연구들의 문제점을 해결하기 위해 먼저 기존 연구들을 분석 한 다음 이를 바탕으로 새로운 구조를 제시한다.

3. 공간 데이타 제공자의 설계 접근 방법

이 논문에서 공간 데이타 제공자의 설계는 기존 방법의 구성요소를 재사용할 수 있는 방법으로 접근한다. 그래서 기존방법에서 사용된 객체들을 독립객체와 종속객체로 구분, 정의한다. 독립객체는 기존 공간 데이타 제공자에서 표준화된 데이타 접근 모델에 따라 작성되어 데이타 소스와는 독립적으로 수행되며 재사용이 가능한 객체이다. 종속객체는 데이타 소스의 종속적인 데이타 접근 모델에 따라 작성되는 객체로서 다른 데이타 소스에서 사용할 수 없는 객체이다. 그러나 이러한 종속객체들 중에는 종속정보들을 데이타 소스별로 매핑시킴으로써 독립화 시킬 수 있는 객체가 있다. 대표적인 예가 질의어 변환 객체이다. 본 논문에서는 기존방법의 구성 객체들의 종속과 독립여부를 구분하기위해 두 가지의 기존 공간 데이타 제공자를 제시하였으며 데이타 소스는 오라클과 고딕을 대상으로 한다[18, 19]. 두 데이타 소스를 대상으로 한 이유는 서로 비교할 수 있는 특성들을 갖추고 있기 때문이다. 오라클은 관계형 DBMS를 기반으로 하고 자체에서 지원하는 패키지인 Spatial Cartridge를 통해 검색, 조작할 수 있는 지리 데이타를 지원하며 질의처리는 질의어(SQL)를 통해 수행된다. 반면에 고딕은 객체지향 DBMS를 기반으로 하는 공간 데이타베이스 시스템이다. 고딕에서 질의어는 지원되지 않으며 고딕의API를 사용하여 질의를 수행한다. 공간 데이타 제공자가 클라이언트에게 제공하는 서비스는 질의처리를 대상으로 하며 OpenGIS의 코바 구현명세에서 정의한 피처 모델 인터페이스를 사용하여 클라이언트에게 제공한다[18].

3.1 오라클 공간 데이타 제공자

오라클 공간 데이타 제공자의 주요역할은 질의어를 통한 질의처리 서비스를 클라이언트에게 제공하는 것이다. 질의처리는 오라클에서 제공하는 Embedded SQL로 구성된 지역 DB인터페이스를 사용하며 영역질의를 대상으로 한다. 지역 DB 인터페이스는 DB에 종속된 데이타를 외부로 유출하기 위해 지원하는 API 집합이다. 공간 데이타 제공자의 구조를 그림 2에서 나타내었다. 그림 2에서 클라이언트의 질의실행 및 결과 데이타 접근은 OpenGIS의 코바 구현 명세에서 정의한 피처 인터페이스를 통해 이루어진다. 질의처리 서비스를 수행하기 위해 필요한 객체들로는 클라이언트가 질의를 하기 위한 스키마 처리 객체와 질의어를 변환하고 실행하는 객체 그리고 질의결과 접근 객체

등으로 구성된다. 스키마 질의 처리를 따로 분리한 것은 질의처리방법이 다르며 스키마의 저장 구조도 다르기 때문이다.

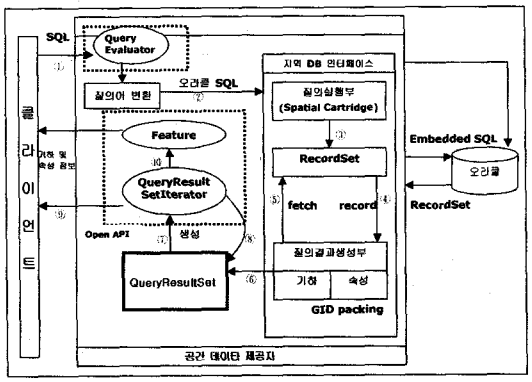

그림 2. 오라클 공간 데이타 제공자(질의처리) 구조

그림 2의 질의어 변환 객체는 클라이언트의 질의어를 오라클용 SQL로 변환하며 질의실행 객체는 변환된 질의어를 실행한다. 질의결과 생성객체는 질의결과인 레코드 집합을 공간 데이타 단위로 생성하며 클라이언트는 생성된 질의결과를 QueryResultSetIterator 표준 인터페이스를 통해 속성들을 얻는다. 질의 실행 객체와 질의결과 생성 객체는 지역 DB 인터페이스이며 데이타 제공자와 오라클을 연결한다. QueryResult SetIterator 표준 인터페이스는 데이타 소스에 종속된 질의결과를 표준 지리 데이타인 피처로 변환한다. 질의결과는 관계형 데이타베이스의 레코드 집합으로 생성되며 이것은 기하정보와 속성정보로 구분된다. 기하정보는 하나의 레이어내에서 유일 객체 식별자인 GID단위의 결과객체로 생성된다[13].

오라클에서 스키마 처리는 오라클에서 제공하는 자료사전을 사용하여 이루어진다. 스키마 정보는 레이어 정보와 레이어내의 속성 정의 정보로 구성된다. 공간 데이타 제공자는 오라클로부터 생성된 스키마 질의결과를 FeatureType 표준 인터페이스에서 사용할 표준 데이타 형으로 전환한다. 클라이언트는 Feature Collection 표준 인터페이스를 통해 스키마 질의결과에 접근한다. 구체적으로 클라이언트는 Feature Collection 내의 FeatureType 인터페이스를 사용하여 레이어 정보와 속성정의 정보를 얻는다.

3.2 고딕 공간 데이타 제공자

고딕에서 외부 모듈과의 연결은 소켓통신을 사용해야만 한다. 따라서 오라클과 비교하여 고딕 공간 데이타 제공자에서는 고딕간의 통신모듈이 추가된다. 그림 3의 공간 데이타 제공자는 질의를 처리하고 결과를 표준 데이타 형으로 전환하는 객체들로 구성된 Open GIS의 피처 인터페이스를 클라이언트에게 제공한다. 그리고 고딕은 질의 실행과 결과 생성을 위해 고딕의 종속된 데이타 접근 모델을 기반으로 한 자체 API를 제공한다. 고딕의 API들이 공간 데이타 제공자에게 지역 DB인터페이스를 제공하며 공간 데이타 제공자와 소켓통신을 한다. 공간 데이타 제공자는 질의를 수행할 때, 고딕이 질의어를 지원하고 있지 않기 때문에 클라이언트로부터 얻은 질의어를 질의인자로 분해한다. 여기서 질의인자는 질의를 실행하기 위해 질의어를 파싱하고 생성된 데이타 구조를 말한다. 질의인자가 고딕에 전해지면 지역 DB 인터페이스에 의해 수행된 데이타 접근 모델에 종속된 질의결과를 소켓통신으로 얻는다. 고딕의 질의 실행 후 결과 형태는 OID의 집합이다. OID는 고딕에서 정의한 공간 객체의 유일 식별자이다. OID를 통해서 지리 데이타의 기하와 속성을 구할 수 있다. 공간 데이타 제공자는 고딕 데이타 접근 모델에 종속되는 질의결과를 표준 데이타 접근 모델로 변환하여 QueryResultSetIterator를 통해 클라이언트에게 피처나 속성으로 제공한다.

스키마 처리는 질의처리와 같이 고딕의 API를 사용하여 스키마 결과를 얻는다. 고딕에서 처리된 스키마 결과는 데이타 접근모델에 종속된 형태로 소켓을 통해 공간 데이타 제공자에게 전달된다. 공간 데이타 제공자는 고딕으로부터 전달 받은 스키마 질의결과를 오라클에서와 같이 FeatureCollection 내의 Feature Type 표준 인터페이스를 사용하여 레이어 정보와 속성 정의정보를 클라이언트에게 제공한다.

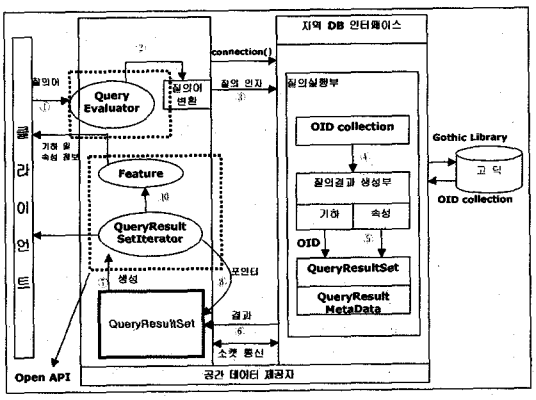

그림 3. 고딕 공간 데이타 제공자(질의처리) 구조

3.3 비교 분석

앞에서 제시한 두 공간 데이타 제공자의 특성과 구조를 비교해보면 차이점과 공통점이 존재함을 알 수가 있다. 공통점은 첫번째로 질의를 실행하기 전에 질의어를 지역 데이타 소스에서 수행 가능한 질의로 변환을 해야 한다. 개방된 표준 API를 제공하기 위해 공간 데이타 제공자는 클라이언트와 데이타 소스 사이에서 포장자 역할을 해야 하기 때문이다. 그러므로 모든 공간 데이타 제공자는 클라이언트의 데이타 접근 모델이 지역 데이타 소스의 데이타 접근 모델에 맞게 변환돼야 한다. 두 번째로 두 데이타 소스는 DBMS의 질의처리기를 통해 질의가 수행된다. 질의처리가 지원되지 않는 파일 시스템 같은 데이타 소스에 대해서는 질의처리기를 데이타 소스에서 구현하거나 데이타 제공자에서 구현하여야 한다. 데이타 제공자에서 질의처리기 구현은 오라클과 같은 데이타 소스를 대상으로 할 때는 적합하지 못하다. 세번째로 질의결과에 대한 접근은 클라이언트에게 상호운용을 지원해야 하므로 표준 데이타 접근모델로 이루어져야 한다. 표준 데이타 접근 모델은 OpenGIS에서 정하고 있는 피처모델을 사용한다[2, 3]. 클라이언트는 데이타 소스에 종속된 질의결과를 ResultSetIterator 표준 인터페이스를 통해 접근한다. 질의결과는 ResultSetIterator를 통해 필요한 경우에만 피처로 생성되어 클라이언트에게 제공되거나 직접 공간, 비공간 속성에 접근할 수도 있다.

두 데이타 제공자의 차이점은 첫째로 두 데이타 소스의 질의처리에 있어 질의어 사용 유무이다. 고딕은 질의어가 지원되지 않는 반면 오라클은 질의어가 지원된다. 고딕의 데이타 제공자에서는 클라이언트의 질의어에서 질의요소를 추출한 후에 자체 API를 사용하여 질의처리를 수행한다. 질의요소는 질의 인자와 같은 개념으로 질의어를 파싱한 결과에서 생성된 데이타 구조이다. 오라클 데이타 제공자는 오라클 질의어를 생성한 후에 DBMS에서 처리한다. 둘째로 고딕과 오라클의 데이타 모델이 각각 관계형 모델과 객체지향형 모델을 기반으로 하기 때문에 질의결과 생성방법이 서로 다르다. 고딕은 객체 단위로 질의결과를 만들면 되지만 오라클은 레코드 단위로 얻은 결과를 객체단위로 다시 전환해야 한다.

앞의 분석을 토대로 공간 데이타 제공자에서 공통으로 처리되면서 독립적으로 수행할 수 있는 객체는 표준 데이타 접근 모델관련 객체들과 질의어 변환 객체이다. 그리고 공통으로 수행되는 객체들 중에 데이타 소스의 데이타 접근 모델에 종속되는 객체가 있다. 그것은 스키마 질의와 일반 질의를 수행하는 객체, 처리결과를 변환하는 객체 등이 해당된다.

지금까지 살펴본 두 데이타 제공자를 통해서 독립으로 수행되는 객체와 종속 되는 객체를 구분, 정의하였다. 다음 장에서는 이장에서 데이타 제공자의 구성요소 중에 새로 구분, 정의한 객체를 중심으로 데이타 제공자 컴포넌트를 설계하고 구현방법을 제시한다.

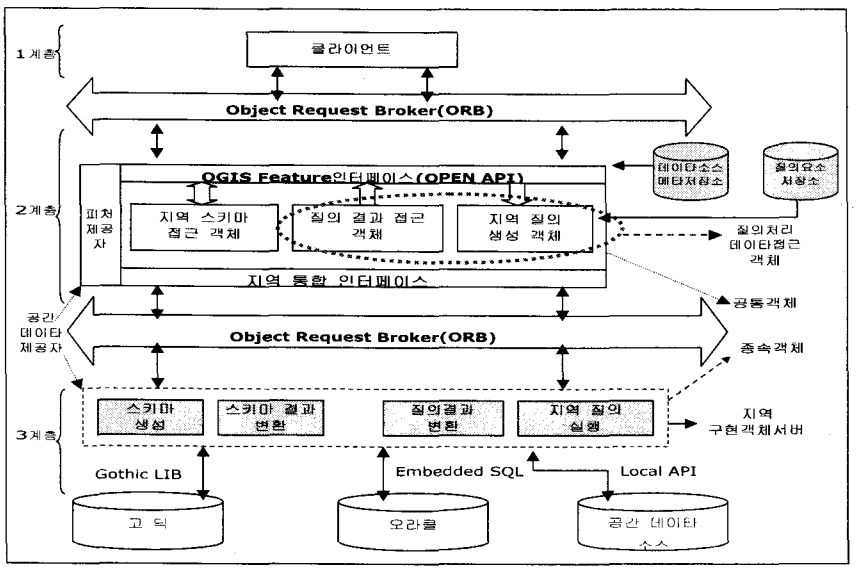

그림 4. 전체 시스템 구조

4. 전체 시스템 구조

전체 시스템의 구조는 아래의 그림 4와 같이 3개의 계층으로 구성된다. 첫번째가 공간 데이타 제공자를 통한 서비스를 제공받는 클라이언트 계층이고 두번째가 데이타 소스와는 독립운용, 수행되며 클라이언트에는 표준화된 데이타 접근 모델을 제공하는 공간 데이타 제공자 계층이다. 마지막 계층은 데이타를 소유하고 있으면서 종속된 데이타 접근 모델을 제공하는 데이타 소스 계층이다. 각 요소에 대한 구체적 기술은 다음 절에서 다룬다.

4.1 공간 데이타 제공자

공간 데이타 제공자는 클라이언트에게 이질적인 데이타 소스들의 종속 데이타 접근 모델을 표준화된 데이타 접근 모델로 전환하여 제공한다. 공간 데이타 제공자가 제공하는 서비스는 질의처리 서비스를 대상으로 한다. 공간 데이타 제공자의 구성은 그림4와 같이 질의처리 데이타 접근 객체와 지역 스키마 접근 객체, 지역 통합 인터페이스로 이루어진다.

클라이언트는 데이타 소스의 서비스를 제공 받기 위해 데이타 소스 메타 저장소가 필요하다. 데이타 소스 메타 저장소는 공간 데이타 제공자에서 관리, 운용되는 저장소이며 클라이언트가 질의를 처리하기위한 데이타 소스 선정을 위해 사용된다. 데이타 소스 메타 저장소는 서비스를 클라이언트에게 제공할 수 있는 데이타 소스들의 접속정보가 등록되어 있으며 내용은 데이타 소스명, 서버명 등이다. 데이타 소스명은 클라이언트에서 데이타 소스를 식별하기 위한 것이며 서버명은 공간 데이타 제공자가 클라이언트에 의해 선택된 데이타 소스의 코바 객체 참조를 얻기 위한 기초 정보이다. 데이타 소스가 추가될때 데이타 소스 메타 저장소에 새로운 메타 정보를 데이타 소스를 추가하는 관리자가 메타 정보 관리기를 통해 추가 함으로써 클라이언트는 추가된 데이타 소스 서비스를 공간 데이타 제공자에게 요청할 수 있다.

4.1.1 질의처리 데이타 접근 객체

질의처리 데이타 접근 객체는 질의어를 보낼 데이타 소스에서 처리되도록 데이타 소스 질의어로 변환하고 처리된 결과를 표준 데이타 접근 모델로 클라이언트에게 제공한다. 구성은 그림5와 같으며 지역 질의 생성 객체와 질의결과 접근 객체로 이루어진다. 그림 5에서 클라이언트로부터 얻은 질의어는 QueryEvaluator 객체내의 지역 질의생성 객체를 통해 해당 지역 데이타 소스에 맞는 질의로 변환된다.

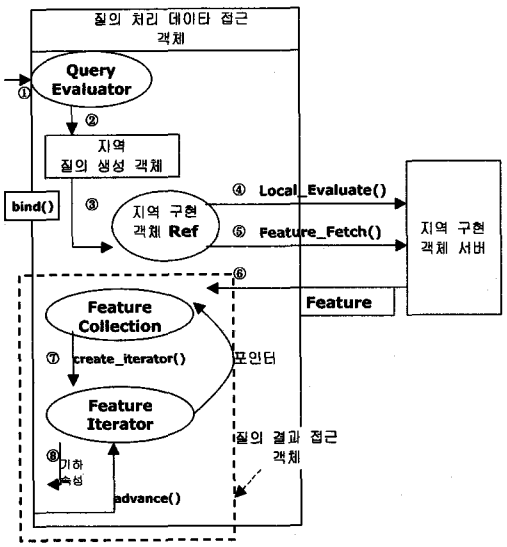

그림 5. 질의처리 데이타 접근 객체의 구조

지역질의생성 객체는 질의어 파서 객체, 질의요소매핑 객체, 질의생성 객체, 질의요소 저장소로 구성된다. 질의어 파서 객체는 질의어를 파싱하여 질의요소들을 생성하며 질의요소매핑 객체는 질의요소 저장소를 참조하여 해당 데이타 소스의 질의요소로 변환한다. 질의요소는 질의어를 파싱하여 생성된 질의어 구성요소들이며 질의요소 저장소는 질의요소를 데이타 소스에 따른 지역질의요소로 매핑하기 위한 저장소이다. 지역질의요소는 질의요소 저장소를 사용하여 매핑된 데이타 소스의 질의요소이다. 질의어 생성 객체는 질의요소 저장소를 통해 매핑된 질의요소를 해당 데이타 소스의 질의어로 생성하는 객체이다.

생성된 질의는 지역 구현 객체 참조를 통한 Local_Evaluate 메소드 호출로 지역 구현 객체 서버에 전달된다. 질의처리 데이타 접근 객체가 질의결과를 Feature_Fetch메소드를 사용하여 얻어서 Feature Collection 객체를 생성한다. 클라이언트는 질의결과 FeatureCollection을 그림5의 질의결과 접근 객체에서 생성되는 FeatureIterator 객체를 통해 접근한다.

4.2 지역 통합 인터페이스

지역 통합 인터페이스는 서로 다른 데이타 소스들의 종속객체들을 코바 객체로 작성하고 공간 데이타 제공자 컴포넌트가 여러 데이타 소스들에 대해서 독립된 소프트웨어 모듈로 수행할 수 있게 하는 인터페이스이다. 지역 통합 인터페이스의 IDL은 그림 6과 같다.

그림6에서 공용체 QueryStr은 질의어 지원 유무에 따른 질의를 수용하기위한 데이타 구조이다. 공용체 QueryStr은 질의어 지원시 문자열의 Query가 사용되고 질의어가 지원되지 않을 경우는 구조체 Elements를 사용한다. 그림 7에서 LocalServer Integration이 지역 통합 인터페이스이다. Local ServerIntegration 인터페이스는 질의를 수행하고 그 결과를 변환하여 표준 데이타 형으로 얻는 인터페이스 함수 Local_Evaluate, Feature_Fetch와 스키마 정보를 얻어 표준 데이타 형으로 변환하는 인터페이스 함수 Local_Schema, FeatureType_Fetch로 구성된다.

버이다. 지역 구현 객체 서버에서 작성되는 객체들은 각 데이타 소스의 특성에 따라 구현되는 객체이지만 지역 통합 인터페이스에 맞추어 작성된다. 그림 7은 지역 구현 객체 서버의 구조이며 질의처리를 수행하는 구조이다. 그림 7에서 공간 데이타 제공자에서 변환된 질의는 지역 통합 인터페이스에 정의된 Local_ Evaluate 메소드를 통해 지역 구현 객체 서버에 전달된다. 질의 수행 후 생성된 지역 질의결과는 데이타 소스의 데이타 접근 모델에 일치하는 공간 객체로 생성된다. 공간 객체 집합은 공간 데이타 제공자가 표준화된 질의결과를 얻기위해 사용하는 Feature_Fetch 메소드 호출로 인해 지역 구현 객체 서버에서 피처로 생

```
.....
struct GeomConstraint {
        spatial_op oper;
        geometry geom;
        short geo_type;
}
struct Elements {
            Select select_list;
            From layer_list;
            Geom_Constraint constraint;
};

union QueryStr (long){
            case 1 : Elements elements;
            case 2 : string Query;
};

struct UserInfo {
        string user;
        string passwd;
};
. . . . .
interface LocalServerIntegration
{
  Boolean Local_Evaluate (in QueryStr query, in UserInfo userinfo, out long
                          FeatureNumber);
  FeatureSeq  Feature_Fetch (in long pointer);
  FeatureDataSeq  FeatureData_Fetch (in long pointer);
  Boolean Local_Schema (in UserInfo userinfo);
  FeatureTypeSeq  FeatureType_Fetch (in long  pointer);
 }
```

그림 6. 지역 통합 인터페이스 IDL

4.3 지역 구현 객체 서버

지역 구현 객체 서버는 여러 데이타 소스에 대한 동일한 인터페이스를 지원하기위해 정의된 지역 통합 인터페이스의 구현 부분으로써 코바 객체를 제공하는 서

성되어 공간 데이타 제공자로 보내진다. 공간 데이타 제공자는 Feature_Fetch메소드의 인자 조정을 통해 피처 데이타의 양을 조정할 수 있다.

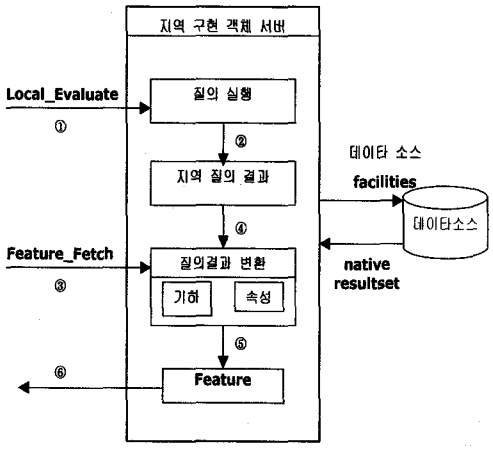

그림 7. 지역 구현 객체 서버(질의처리)의 구조

5. 구 현

5.1 구현방법의 분류

전체 시스템의 주요기능은 클라이언트가 표준화된 데이타 접근 모델을 제공하는 공간 데이타 제공자를 통해서 다양한 데이타 소스들로부터 질의처리 서비스를 제공 받는 것이다. 이를 위해 필요한 구성요소는 스키마 정보를 얻는 객체와 질의어를 변환하는 객체와 질의결과를 조작, 제어 할 수 있는 객체들이다. 여기서 시스템의 성능에 주요한 영향을 주는 것이 질의결과의 생성과 조작이다. 따라서 이 절에서는 질의결과의 생성과 접근을 대상으로 구현 방법을 제시한다.

구현방법은 공간 데이타 제공자에서 질의결과를 데이타로 유지하는 방법과 참조로 유지하는 방법으로 나누어지며 이것은 다시 각각 공간 데이타 제공자에서 생성되는 질의결과 구조인 FeatureCollection을 이루는 구성요소에 따라 4가지 방법으로 나누어진다. 표 1은 FeatureCollection의 구성 요소와 질의결과 유지 장소사이에서 구현이 가능한 경우를 O로 표시하고 그렇지 못한 경우는 X로 표시하였다. 표1의 가로요소가 표준 질의결과 집합 구조인 FeatureCollection을 구성하는 요소이고 세로요소는 질의결과 유지 장소를 나타낸다.

표 1에서 공간 데이타 제공자에서 질의결과를 데이타로 유지 할 때 가능한 FeatureCollection의 구성요소는 피처 데이타와 피처 객체이다. 피처 데이타는 데이타 소스에서 생성된 질의결과가 표준 지리 데이타 구조인 피처로 변환되어 공간 데이타 제공자로 전달된 데이타를 말하며 피처 객체는 데이타 소스로부터 전달된 피처 데이타를 코바 객체로 생성한 것을 말한다. 그리고 질의결과를 참조로 유지할 때 FeatureCollection은 질의결과 집합 참조와 피처객체 참조를 구성요소로 사용할 수 있다. 질의결과 집합 참조는 데이타 소스에서 생성된 질의결과 집합에 대한 코바 객체 참조이며 공간 데이타 제공자는 질의결과 집합 참조를 사용하여 질의결과에 접근한다. 피처 객체 참조는 데이타 소스에서 생성된 질의결과가 피처를 단위로 하는 코바객체 집합으로 생성된 데이타 구조를 말한다.

표 1. 질의결과 유지방법에 따른 구현 가능한 방법

	FeatureCollection 구성방법			
질의결과 유지 방법	피처 데이타	피처 객체	질의결과 집합 참조	피처 객체 참조
데이타 유지 방법	○	○	X	X
참조 유지 방법	X	X	○	○

5.1.1 구현 모델

설계를 기반으로 한 세부적인 구현방법은 구현모델을 통해서 명확하게 제시될 수 있다. 구현 모델은 공간 데이타 제공자의 기본 설계를 기반으로 작성되었고 UML의 클래스다이어그램을 통해 2가지 모델로 구분하여 나타내었다. 구현모델은 피처 구성 모델과 질의결과 구성 모델로 나누었다. 구분 기준은 질의결과 유지 방법을 위한 데이타 구성과 참조 구성에 따른 것이다. 구현 모델에서 다루는 주요 기능은 질의처리 서비스이며 이를 위해 필요한 클래스들의 정의와 클래스들간의 제어관계를 그림 8과 그림 9에서 나타내었다.

그림 8의 피처 구성 모델은 피처 단위를 FeatureCollection의 구성요소로 사용하는 피처 데이타, 피처객체, 피처객체 참조 방법들로 구체화되며 OpenGIS 코바 구현 명세의 Simple Feature 모델을 기반으로 한다. 공간 데이타 제공자의 QueryEvaluator객체는 LocalServerIntegration 객체를 생성하여 수행 결과에 의존하며 LocalServerIntegration객체는 LocalEvaluate객체와 LocalSchema객체의 결과에 의존한다. LocalEvaluate객체는 질의결과를 피처단위로 구성되는 FeatureCollection으로 변환한다. 그림 9의 질의 결과 구성 모델은 질의결과 집합 단위로 FeatureCollection을 구성하는 질의결과 집합 참조 방법에 대한 모델이고 피처 구성 모델과 공통으로 OpenGIS Simple Feature모델을 기반으로 한다. 피처 구성모델과 다르게 이 모델에서 LocalEvaluate 객체는 질의 결과에 대해서 QueryResultSet

Iterator 객체를 FeatureCollection으로 구성하여 클라이언트에게 제공한다. 따라서 전반적인 두 모델의 차이는 공간 데이타 제공자의 FeatureCollection 구성 방법의 차이와 공간 데이타 제공자가 지역 구현 객체 서버로부터 전달 받는 질의결과 구조의 차이이다. 즉, 피처 구성 모델은 클라이언트에게 피처 단위의 질의결과에 대해 공간 데이타 제공자를 통해 접근할 수 있게 하지만 질의결과 구성 모델은 질의결과에 대한 Iterator를 통해 클라이언트가 피처에 접근할 수 있는 방법을 제공한다.

5.1.2 피처 데이타에 의한 질의결과 유지방법

클라이언트에서 요청한 질의는 데이타 소스에서 수행되어 생성된 질의결과가 공간 데이타 제공자로 전달되어 관리되는 방법이다. 공간 데이타 제공자에서

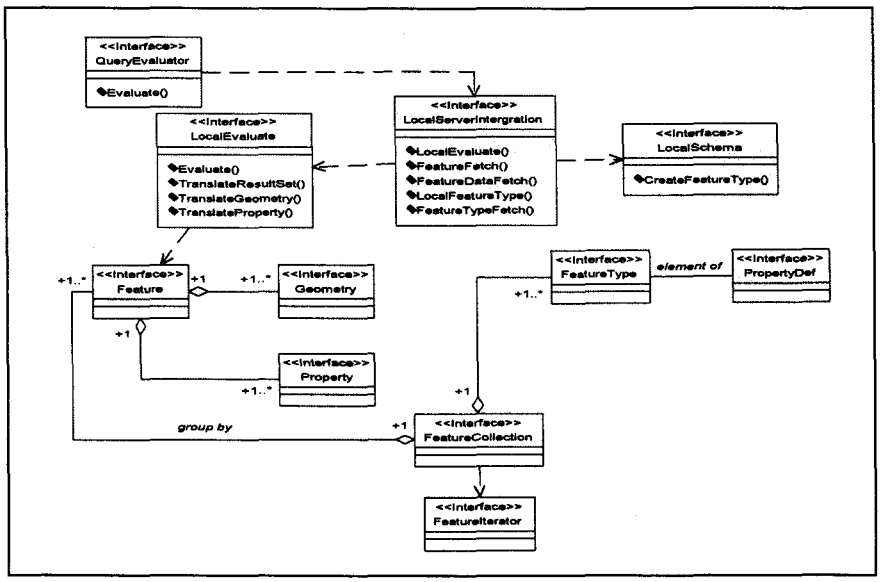

그림 8. 피처 구성 모델

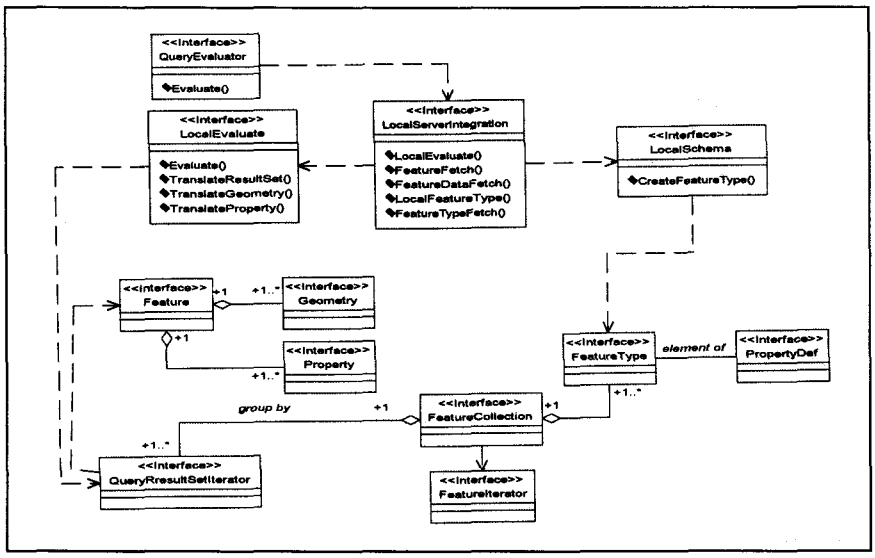

그림 9. 질의결과 구성 모델

FeatureCollection의 생성은 데이타 소스에서 피처 데이타로 전달받아 이루어진다. 전달된 피처 데이타는 두 가지 방법에 의해 FeatureCollection의 구성요소로 사용된다. 첫째가 피처 데이타를 이용한 Feature Collection 생성 방법이며 또 다른 하나는 피처 데이타를 피처단위의 코바 객체로 생성하여 Feature Collection을 구성하는 방법이다.

가. 피처 데이타 구성 방법

피처 데이타 구성 방법은 데이타 소스로부터 질의결과를 피처 데이타로 변환하여 공간 데이타 제공자에게 전달하고 이것을 변환없이 FeatureCollection으로 만든다. FeatureCollection 객체는 클라이언트에게 FeatureIterator 객체를 통해서 속성과 기하를 직접 제공하거나 피처 객체를 생성하여 제공할 수 도 있다.

이 방법의 장점은 필요할 때만 피처 객체를 생성하므로 모든 공간 데이타에 대한 피처 객체 생성 부하가 없으며 FeatureIterator를 사용하여 피처의 속성에 직접 접근 할 수 있다. 단점으로는 클라이언트가 많은 양의 피처를 단위로 데이타를 요구할 때 피처 생성비용의 증가 문제가 있다.

그림 10은 피처 데이타 구성방법을 사용한 데이타 제공자의 구조를 나타내고 있다. 질의수행 후 생성된 질의결과는 공간 데이타 제공자에서 FeatureFetch메소드를 통해 피처 데이타로 변환되어 제공된다. 피처 데이타로 생성된 FeatureCollection은 클라이언트에게 FeatureIterator를 통해서 기하 정보를 제공하고 피처 단위의 결과가 필요할 때는 피처객체를 생성하여 클라이언트에게 제공한다.

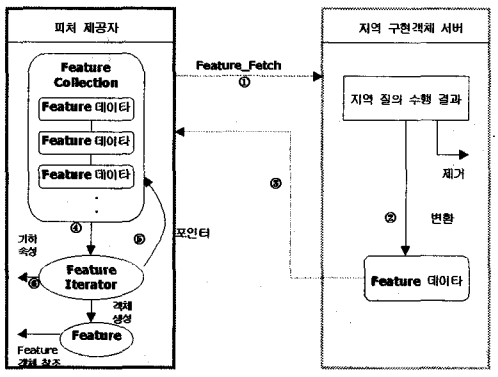

그림 10. 피처 데이타를 이용한 구성방법

나. 피처 객체 구성 방법

피처 객체 구성 방법은 데이타 소스에서 전달받은 피처 데이타를 피처 객체로 생성하여 피처 객체로 FeatureCollection을 구성하는 방법이다. 클라이언트는 피처의 속성에 바로 접근 할 수 있으며 질의결과가 피처 객체로 구성되어 있기 때문에 질의결과에 대한 피처단위의 접근이 용이하다. 단점으로는 모든 질의결과를 피처 단위의 객체로 생성해서 결과를 유지하기 때문에 피처객체 생성 부하로 인한 성능문제가 발생한다.

그림 11에서 데이타 소스가 공간 데이타 제공자에게 피처 데이타를 전달하는 과정은 피처 데이타 구성 방법과 같다. 그러나 공간 데이타 제공자는 Feature Collection을 구성하기 위해 피처 데이타를 피처 객체로 생성한다. 생성된 피처객체는 FeatureCollection의 구성요소로 추가되고 클라이언트는 Feature Iterator를 통해서 피처객체를 클라이언트에게 직접 제공한다.

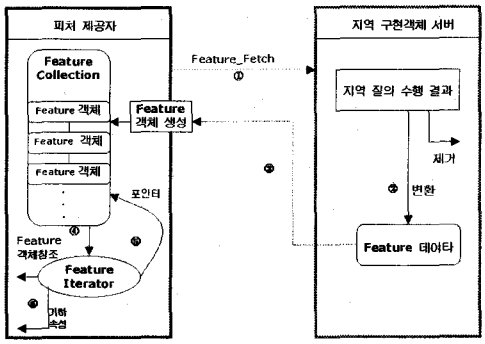

그림 11. 피처객체 구성 방법

5.1.3 참조에 의한 질의 결과 유지 방법

이 방법은 클라이언트에서 요구한 질의를 수행하여 생성한 결과를 데이타 소스에서 유지하는 방법이다. 공간 데이타 제공자는 질의결과를 참조로 유지함으로써 클라이언트는 공간 데이타 제공자에서 유지하는 질의결과의 참조나 피처단위의 코바 객체 참조를 통해 데이타를 얻는다.

장점으로는 공간 데이타 제공자에서 질의결과를 참조만으로 유지하기 때문에 부하가 적으며 여러 데이타 소스에서 생성된 질의결과를 동시에 유지 할 수 있다. 단점으로는 클라이언트가 질의결과를 얻기 위해서 데이타 제공자 참조를 통해 데이타 전달 받아야 하는 전송 부하가 있다.

가. 질의결과 집합 참조를 이용한 구성 방법

질의결과 집합 참조 구성 방법은 데이타 소스에서

생성된 질의 결과의 참조는 공간 데이타 제공자에게 전달하여 필요시 참조를 사용하여 클라이언트에게 데이타를 전달하는 방법이다.

이 방법의 특징은 질의결과를 집합단위의 참조로 유지하기 때문에 피처객체를 참조로 유지하는 방법보다 부하가 적다. 단점으로는 피처 단위로 관리되지 않기 때문에 피처를 생성해야 할 때 부하가 존재하며 참조를 통해서 질의결과가 유지되므로 데이타 전송에 대한 부하가 존재한다.

그림 12에서 공간 데이타 제공자는 수행된 질의결과를 FeatureFetch메소드를 사용하여 질의결과 참조를 얻는다. 공간 데이타 제공자는 참조를 통해 질의결과를 얻을 때 지역 구현 객체의 next와 get_property를 사용하여 전달 받은 피처를 클라이언트에게 제공한다. 변환된 피처의 형은 피처단위의 코바 객체나 구조체형의 데이타로 작성될 수 있다. 클라이언트는 공간 데이타 제공자로 전달된 피처 데이타를 피처 객체로 제공 받거나 FeatureIterator를 통해 속성으로 접근하여 사용한다.

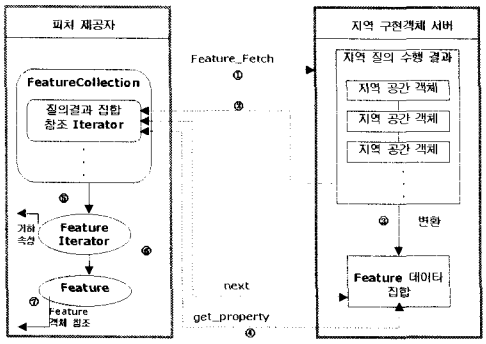

그림 12. 질의결과 집합참조를 이용한 구성방법

나. 피처 객체 참조를 이용한 구성 방법

질의결과는 데이타 소스에서 피처 객체 단위로 생성되어 유지되고 피처객체의 참조를 공간 데이타 제공자에게 제공하여 사용된다. 공간 데이타 제공자는 피처 객체의 참조만을 관리하기 때문에 실제 데이타를 관리하는 것보다 부하가 적다. 이 방법은 질의결과 집합 참조와 같이 피처 객체에 대한 참조만을 공간 데이타 제공자가 유지 하지만 데이타 소스에서 피처에 대한 코바 객체 생성 부하가 크며 공간 데이타 제공자가 참조를 통해서 질의결과를 데이타 소스로부터 전송받는 부하도 있기 때문에 상당한 성능저하 문제를 가지고 있다.

그림 13의 내용은 다음과 같다. 데이타 소스에서 생성된 질의결과는 피처단위의 코바 객체의 집합으로 변환된다. 공간 데이타 제공자는 FeatureFetch 메소드를 사용하여 변환된 피처 객체 참조를 구하고 이들로 구성된 FeatureCollection을 생성한다. 클라이언트는 다른 방법들 처럼 FeatureCollection에 접근 할 때 FeatureIterator를 사용하여 피처의 속성에 바로 접근 하거나 피처객체를 생성하여 접근 할 수 있다.

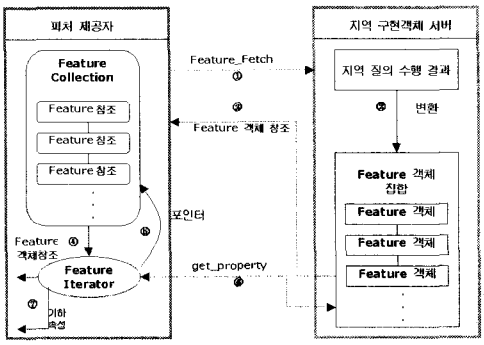

그림 13. 피처 객체 참조를 이용한 구성방법

5.1.4 성능평가 및 비교

앞 절에서 공간 데이타 제공자 설계를 기반으로 질의결과 유지방법에서 구현될 수 있는 4가지 방법을 제시하였다. 각 구현 방법의 성능 평가 요소는 구성요소에서 발생되는 성능비용 중에서 큰 비용을 차지하는 부분을 대상으로 해야 된다. 그리고 클라이언트가 공간 데이타 제공자의 질의결과에 접근하는 시간이나 데이타 소스에서의 질의 수행 시간 등은 각 구현 방법이 같으므로 성능평가 대상에서 제외하였다. 그래서 성능평가요소는 각 구현 방법에 따라 시스템의 동작에서 시간비용이나 수행 비율이 많이 소모되는 연산들을 선택하였다.

가. 성능 평가 환경

성능평가를 위한 환경은 다음과 같다. 데이타 소스는 상용GIS 소프트웨어인Gothic 3.0이고 데이타는 창원시 지리정보데이타가 공간 데이타베이스로 구축되어있으며 공간 데이타의 양은 약 60만 객체이다. 성능평가를 위해 사용된 장비로는 서버가 Digital DEC400-500이고 클라이언트는 CPU 166Mhz, 64M의 사양을 갖춘 PC를 사용하였다.

나. 성능 비교

그림 14는 앞에서 제시한 구현 방법들을 성능비교

하기 위해 전체 질의 처리비용 중에서 공간 데이타 제공자에서 소요되는 비용을 성능평가 요소로 정하였다. 그림 14에서와 같이 피처 데이타 구성 방법의 처리 성능비용이 가장 적으며 피처 객체 참조를 통한 구현 방법의 성능 비용이 가장 크게 나타나고 있다. 성능저하 원인은 피처 단위의 코바 객체 생성 부하와 공간 데이타 제공자가 참조를 통해서 데이타 소스에서 클라이언트로 질의결과를 전송하는 데 발생되는 부하에 의한 것이다. 그리고 질의결과 집합 참조를 사용하여 FeatureCollection을 구성하는 방법이 피처단위의 코바객체를 사용한 FeatureCollection 구성 방법보다 좋음을 알 수 있다. 이유는 피처 단위로의 코바 객체 생성 부하가 원격지 메소드 호출에 따른 데이타 전송 부하보다 더 크게 나타났기 때문이다. 따라서 질의결과에 대한 피처 객체를 필요에 따라 생성하는 것이 좋게 나타났다.

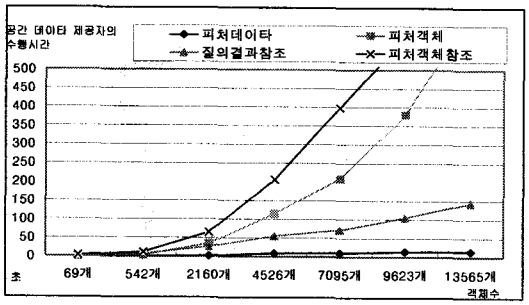

그림 14. 구현 방법 간의 성능비교

5.2 구현 결과

클라이언트는 원하는 데이타 소스로부터 질의응답을 받기 위해 공간 데이타 제공자를 통해서 레이어와 속성 메타 정보를 제공받는다. 그리고 클라이언트는 스키마 정보를 사용하여 데이타 소스에 질의할 질의어를 구성한다. 질의어는 표준 SQL를 따르며 영역 질의이다. 질의어가 데이타 제공자를 거쳐서 데이타 소스에서 처리된 후 생성된 질의결과는 클라이언트의 사용자 인터페이스에 출력된다. 공간 데이타 제공자는 단일 데이타 소스에 대한 질의 결과를 유지하며 새로운 질의가 발생할 때 삭제된다.

그림 15는 동일한 공간 데이타 제공자에서 제공한 인터페이스를 사용하여 서로 다른 두 데이타 소스의 레이어들 중 필지 레이어를 출력한 결과이다. 화면의 상단부분의 필지 레이어는 고딕 데이타 소스에서 가져온 것이며 아래의 필지 레이어는 오라클에서 가져온 것이다.

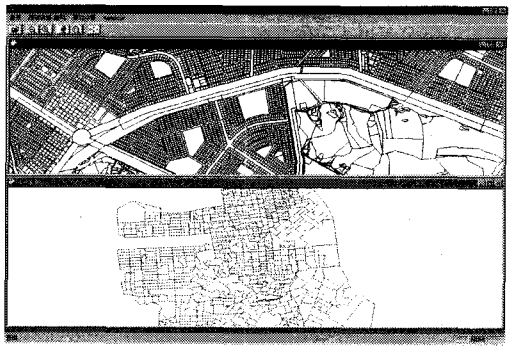

그림 15. 클라이언트의 지도출력 화면

시스템의 구현 환경에서 데이타 소스는 Spatial Cartridge를 이용한 오라클 8.05와 GIS 소프트웨어인 고딕 3.0을 사용하였다. 코바제품은 IONA사의 평가판인 Orbix 2.03c(C++매핑)을 사용하였다. 데이타 소스가 운영되는 머신은 SunOS 6.0이 설치되어 있는 Sun Sparc과 Digital UNIX V4.0B가 설치된 Digital DEC이다. 클라이언트 프로그램은 Visual C++을 사용하여 작성하였다.

5.3 구성 요소의 재사용

본 논문에서는 기존 공간 데이타 제공자를 구성하는 객체들 중에서 데이타 소스에 독립되어 수행되는 객체들을 정의하여 공간 데이타 제공자를 구성하였고 데이타 소스에 종속된 데이타 형을 다루는 종속 객체들은 지역 통합 인터페이스 규칙에 따라서 코바 객체로 작성하였다. 그리고 제시한 공간 데이타 제공자는 질의 변환을 위한 데이타 소스의 질의와 메타 정보를 매핑 저장소에 등록하여 이용하였다. 따라서 새로운 데이타 소스는 지역 통합 인터페이스를 따른 코바 객체만을 작성함으로써 공간 데이타 제공자와 실시간에 연결될 수 있으며 공간 데이타 제공자는 데이타 소스에 대해 독립된 컴포넌트로 동작한다. 따라서 데이타 소스는 기존 공간 데이타 제공자 구현시 작성해야 했던 클래스들을 공간 데이타 제공자의 컴포넌트화로 재사용이 가능하게 됐다.

표 2는 공간 데이타 제공자의 컴포넌트화로 데이타 소스가 재사용할 수 있는 클래스들과 구현해야 하는 클래스들을 구분하여 나타내었다. 데이타 소스가 재사용할 수 있는 객체들로는 공간 데이타 제공자를 구성하는 지역 질의어 생성, 질의결과 접근, 스키마 접근 객체등이 있다. 데이타 소스는 이들 객체들의 재사용으로 구현 부담이 감소되었다.

표 2. 재사용되는 객체와 구현해야 될 객체

	구성 객체		클래스 리스트
재사용되는 객체	질의처리 데이타 접근 객체	지역 질의어 생성 객체	Parser, Mapper, LocalQuery
		질의결과 접근객체	Feature, QueryEvaluator, Property, FeatureCollection, FeatureIterator
	지역 스키마 접근 객체		FeatureType, PropertyDef, FeatureTypeCollection, FeatureTypeIterator
구현해야할 객체	구현 객체 서버		Feature, LocalServerIntegration

6. 결론 및 향후 연구 방향

본 논문에서는 다양한 GIS의 데이타 소스들의 상호운용을 지원하기 위해 사용되는 기존 방법의 공간 데이타 제공자를 통한 재사용할 수 있는 방향으로 접근, 설계하였다. 공간 데이타 제공자는 종속객체로 구성된 데이타 소스측 모듈에 지역 통합 인터페이스를 제공하고 클라이언트에게 표준 지리 데이타 접근모델을 따르는 과 개방된 표준 API를 제공한다. 지역 통합 인터페이스는 이질적인 데이타 소스들의 특성을 코바 객체로 캡슐화하여 공간 데이타 제공자에게 제공한다. 그리고 지역 통합 인터페이스와 질의요소 저장소의 사용은 종속부분을 독립화 시킴으로써 클라이언트 프로그램이나 공간 데이타 제공자를 변경하지 않고서도 GIS 데이타 소스의 실시간 추가가 가능하며 구현비용도 감소될 수 있다. 구현에서는 설계를 기반으로 공간 데이타 제공자 구현방법을 질의결과 유지방법에 따라 분류하고 각 방법에 대한 성능평가를 실시하였다. 성능평가 결과에서 지도출력 같이 피처 데이타에 접근 하기 위한 기능들은 최소 비용을 소요하는 피처 데이타 구성 방법이 적합했다. 참조 유지 방법내의 질의결과 집합 참조 구성 방법은 여러 데이타 소스의 질의결과를 유지하는데 비용을 최소화 시킬 수 있는 방법으로 나타났다. 그러나 질의결과에 대해 피처단위로 코바 객체를 생성하는 데이타 유지 방법내의 피처 객체 구성 방법과 참조 유지 방법내의 피처 객체 참조 구성 방법은 비용이 크기 때문에 다량의 질의결과에 대한 접근 방법으로 적합하지 않다. 따라서 피처를 필요시에 생성하여 사용하도록 하는 것이 시간비용을 감소시킬 수 있는 방법이다.

향후연구로 코바 기반에서의 성능향상을 위한 지리 데이타의 객체단위나 영역단위의 캐쉬기법에 관한 연구나 질의결과의 재사용과 미들웨어내의 추가적 서비스를 위한 분산질의처리에 관한 연구의 진행이 필요하다.

참 고 문 헌

[1] OpenGIS Consortium, Inc., The OpenGIS Guide, 1998.

[2] OpenGIS Consortium, Inc., The OpenGIS Abstract Specification Model, version3, 1998.

[3] OpenGIS Consortium, Inc., OpenGIS Simple Features Specification for CORBA, Revision 1.0, 1998.

[4] Ebru Kil, Gokhan Ozhan, Cevdet Dengi, Nihan Kesim, etc, Experiences in Using CORBA for a Multidatabase Implementation, 6th International Workshop on Database and Expert Systems Applications, 1995.

[5] Yannis Papakonstantinou, Ashish Gupta, Hector Garcia-Molna, A Query Translation Schema for Rapid Implementation of Wrappers, In DOOD 95, 1995.

[6] Ling Lu, Ling Ling, and M.Tamer Ozsu, Interoperability in Large-scale Distributed Information Delivery Systems. In Advances in Workflow Systems and Interoperability, 1996.

[7] Asuman Dogac Cevdet Dengi M.Tamer Ozsu, Building Interoperable Databases on Distributed Object Management Platforms, Communications of the ACM, 1996.

[8] Yooshin Lee and Ling Liu, Calton Pu, Towards Interoperable Heterogeneous Information Systems: An Experiment Using the DIOM Approach, In the Proceesings of the 12th ACM Symposium on Applied Computing (ACM SAC'97) Special track on Database Technology, 1997.

[9] Yooshin Lee, Prototyping The DIOM

Interoperable System, 1997.
[10] Erich Gamma, Richard Heln, Ralph Johnson, John Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison Wesley Inc., 1995.
[11] Alan Pope, The CORBA Reference Guide, Addison Wesley, 1997.
[12] Sean Baker, CORBA Distributed Objects Using Orbix, acm press, 1997.
[13] Oracle Corporation, Oracle8 Spatial Cartridge Users Guide and Reference, 1997.
[14] Oracle Corporation, Programmer's Guide to the Pro*C/C++ Precompiler, 1997.
[15] IONA Technologies, Orbix Programmer's Guide, 1997.
[16] IONA Technologies, Orbix Programmer's Reference, 1997.
[17] 객체 랩퍼 클래스를 이용한 RPC프로그램의 재공학 기법, 정보과학회논문지, 1999.
[18] 안경환, 조대수, 홍봉희, 상호운용을 지원하는 CORBA 기반 개방형 지리정보시스템의 설계 및 구현, 추계정보과학회, 1998.
[19] 안경환, 조대수, 홍봉희, CORBA를 이용한 OpenGIS 기반 미들웨어 구현,개방형GIS연구회논문지, 1999.
[20] 장염승, 윤재관, 한기준, GEUS 기반 Open GIS 서버의 설계 및 구현, '99 개방형 지리 정보 시스템 학술회의 논문집, 1999.
[21] 김민석, 안경환, 홍봉희, 이질적인 GIS 데이타 소스의 상호운용을 지원하는 CORBA기반의 표준데이타 제공자 설계, '99 개방형 지리 정보 시스템 학술회의 논문집, 1999.

김민석
1998년 경성대학교 컴퓨터공학과 졸업(공학사)
2000년 부산대학교 대학원 GIS 학과 졸업(공학석사)
2000년~현재 씨제이 드림소프트 연구원
관심분야 : GIS, 공간데이타베이스, 분산 객체 기술, 개방형 지리정보시스템

안경환
1997년 부산대학교 컴퓨터공학과 졸업(공학사)
1999년 부산대학교 대학원 컴퓨터공학과 졸업(공학석사)
1999년~현재 부산대학교 대학원 컴퓨터공학과, 박사과정
관심분야: GIS, 분산객체기술, 개방형지리정보시스템

홍봉희
1982년 서울대학교 전자계산기공학과 졸업(공학사)
1984년 서울대학교 대학원 전자계산기공학과 졸업(공학석사)
1988년 서울대학교 대학원 전자계산기공학과 졸업(공학박사)
1987~현재 부산대학교 공과대학 컴퓨터공학과 교수
관심분야: 공간 데이타베이스, GIS표준화, 병렬 GIS, 개방형지리정보시스템