2001년 한국정보처리학회 춘계 학술발표논문집 제8권 제1호

성능 향상을 위한 개방형 GIS의 CORBA 인터페이스 구현

강구*, 김상호*, 김승환*, 류근호*, 오병우**
*충북대학교 전자계산학과, **한국 전자 통신 연구원
e-mail: *{kangih, shkim, kshwan, khryu}@dblab.chungbuk.ac.kr
**bwoh@etri.re.kr

Implementation of a CORBA Interface on Open GIS for Improving performance

Koo Kang*, Sang-Ho Kim*, Sung-Hwan Kim*, Keun-Ho Ryu,
Byoung-Woo Oh**
*Dept. of Computer Science, Chungbuk University
**ETRI

요약

지리 정보 자원을 서로 공유하기 위한 표준으로 OGC의 OpenGIS simple feature specification이 있다. Feature의 집합을 다루는 FeatureIterator 인터페이스에서 기존의 인터페이스는 서버에 새로운 Feature나 Geometry 객체를 생성하고 클라이언트가 이를 다시 접근 해야하는 비효율성이 존재한다. 그래서 서버에 접근하는 비용을 줄이기 위해, 클라이언트가 서버의 Feature들을 접근하는 방법인, 새로운 get_WKSGeometries 인터페이스를 제안하고 구현한다. 이 인터페이스는 서버에 객체를 생성하는 것이 아니라, WKS 형태로 Feature의 데이터를 클라이언트에 직접 전달함으로써 클라이언트가 서버에 재접근을 요구하지 않는다. 따라서, 이 논문에서 제안한 get_WKSGeometries 인터페이스를 이용하면, 클라이언트가 서버에 접근하는 비용을 줄이고 성능을 향상시키도록 하였다. 아울러 구현 명세를 구현하고 새로운 인터페이스를 제안 검증하였다.

1. 서론

정보화 사회의 급속한 발전에 의해 지리정보의 사용이 점차 활성화되고, 이러한 지리정보 자원의 생성, 저장, 관리 및 응용을 위한 지리정보 시스템(GIS: Geographical Information System)의 구축과 응용개발이 보편화 단계에 이르고 있다. 지리정보 시스템을 이용하여 특정한 목적을 가지고 제작된 지형 공간 데이터를 다양한 분석에 이용할 수 있으며, 또한 지형 공간 데이터를 다른 정보와 결합시켜 함께 분석하여 원하는 정보를 추출할 수 있다[9][10]. 상호운용은 표준화된 지리 데이터의 접근, 교환, 분산 지리정보처리를 수행함으로써 데이터의 공유뿐만 아니라 표준 인터페이스를 통한 지리정보 처리 서비스의 공유까지 가능하게 해주는 것이다[1]. 상호운용을 지원하기 위해서는 서로 다른 데이터 소스들의 지리 데이터 표현에 대한 데이터 접근 모델을 외부 응용 프로그램이 접근할 수 있도록 표준화된 데이터 접근 모델로 변환해야 한다[11].

GIS분야에서는 이러한 공간 데이터 및 서비스에 대한 표준이 필요한데 OGC(OpenGIS Consortium)의 OpenGIS(Open Geodata Interoperability Specification) 서비스를 이용하여 상호 운용성을 제공해줄 수 있다[2][5][8]. OpenGIS에서는 모든 분산 컴퓨팅 환경에 독립적인 추상 명세(Abstract Specification)와 함께 각 정보기술별 구현 명세를 제시하고 있다. 현재에는 SQL, OLE/COM, CORBA의 세 가지 구현 명세가 나와 있다.

이 논문은 OpenGIS 추상 명세 중 CORBA를 위한 추상 명세의 FeatureIterator interface에서 advance() 와 current()를 이용해서 하나의 Feature 객체를 얻을 수 있고, 또는 next()나 next_n()을 이용하여 Feature 나 Geometry 객체(들)를 얻을 수 있다. CORBA의 특성상 클라이언트와 서버 사이의 통신비용이 많은 비중을 차지하기 때문에 각각의 데이터를 읽는 방법은 많은 비용상의 문제점을 가지고 있다. 따라서, 이 논문에서는 WKSGeometry들을 반환해주는 get_WKSGeometries()라는 새로운 인터페이스를 추가하여 CORBA의 단점인 서버 객체에 대한 접근 비용을 줄이면서, 현재의 모든 기하객체들을 한번의 접근으로 읽어오도록 하여 성능향상을 도모한다. 이 성능을 증명하기 위하여 구현 명세정의와 새로운 인터페이스를 제안 검증한다.

이 논문의 효과적인 전개를 위하여, 2장에서는 관련연구로 OGC의 CORBA 구현명세에 대하여 알아보고, 3장에서는

† 이 연구는 2000년도 한국 전자 통신 연구소의 "코바를 위한 개방형 GIS 인터페이스 표준화" 과제의 연구비 지원에 의해 수행되었음.

서버에 접근하는 두 가지 인터페이스 방식에 대해서 알아본다. 4장에서는 기존의 접근 방법과 개선된 접근 방법에 대해서 설명한다. 그리고 5장에서는 실험 및 결과 분석, 마지막 6장은 결론으로 이루어진다.

2. 관련연구

OpenGIS의 CORBA 구현 명세인 OpenGIS Simple Feature Specification은 SQL, OLE/COM, 그리고 CORBA를 위한 3가지 구현 명세가 있다.

SQL을 위한 구현 명세는 ODBC API를 통해서 simple feature들의 집합에 대한 저장, 검색, 질의, 갱신을 지원하는 표준 SQL 스키마를 정의한다. simple feature들은 공간과 비 공간 속성을 갖고, OpenGIS 추상 명세서에 의해 정의된다. 공간 속성들은 기하 값들을 갖고, simple feature들은 두 점간의 선형 보간법으로 2차원의 기하로 표현된다[3]. simple feature의 집합들은 개념적으로 관계형 데이터베이스에서 기하 값을 갖는 테이블로서 저장되고, 각각의 feature는 테이블의 행으로 저장된다. OpenGIS feature들을 표현하는 테이블의 행들은 feature 테이블로 참조되고, 한 개 이상의 기하 값을 갖는 컬럼들을 포함한다.

OLE/COM을 위한 구현 명세는 ODBC, DAO, RDO, ADO, OLE DB를 포함하는 OLE/COM 기술을 이용한다[4]. 특히 ADO는 데이터베이스에 대한 접근과 처리를 위한 OLE 자동화 객체 지향의 표준을 제공하고, 언어 독립적인 기술인 OLE 자동화는 응용의 변경과 통합을 위한 표준을 제공한다. 이러한 패러다임은 GIS 데이터 접근이라는 필요성에 매우 잘 부합한다. GIS는 측지 좌표 체계, 기하, 그래픽 디스플레이와 같은 추가적인 요구를 가진 데이터베이스 문제로 고려될 수 있는데, 이 명세는 마이크로소프트 데이터 접근 기술을 통해 사용 가능한 현재의 인터페이스 외에 GIS 인터페이스들을 다루고 있다[4].

CORBA를 위한 구현 명세는 GIS 소프트웨어 개발자들에게 OMG의 CORBA 기술을 이용하여 응용프로그램을 개발할 수 있는 인터페이스를 제공한다. 응용프로그램은 기하와 피처를 포함하는 지형공간(geospatial) 정보에 접근하고 관리하는 기능을 수행한다. CORBA(Common Object Request Broker Architecture)는 언어, 플랫폼, 운영체제 및 판매자 독립 방식의 객체 지향적인 분산시스템을 위한 상호 운용성을 제공한다[7]. OpenGIS 구조체를 표현하기 위한 CORBA-IDL(interface Definition Language)에서 정의된 인터페이스의 생성을 통해서 이루어진다. 인터페이스들은 Feature 모듈과 Geometry 모듈이라고 하는 두 개의 서브 모듈로 구성된다[5].

이 논문은 OpenGIS의 세 가지 명세 중에서 CORBA를 위한 구현 명세를 구현하였으며, 또 새로운 인터페이스를 제안하고 검증하였다.

3. 인터페이스 방식

OpenGIS simple feature for CORBA 스펙의 구현 명세에는 두 가지의 인터페이스 방식이 있다. 하나는 ContainerFeatureCollection 인터페이스를 이용하는 방식과 QueryableContainerFeatureCollection 인터페이스를 이용하는 방식, 두 가지 방식이 있다. CFC, QCFC는 이 두 가지 인터페이스의 약칭으로 기술한다. CFC은 Feature의 타입에 의해 해당하는 타입의 Feature들을 접근하는 방식이고, QCFC은 SQL 질의문에 의해 해당하는 Feature들을 접근하는 방식이다. 클라이언트는 미들웨어인 CORBA 서버를 통해서 간접적으로 데이터베이스 서버에 접근하게 된다. 초기에 바인딩을 통해서 생성되어있는 CORBA 객체를 식별할 수 있다. CORBA 서버는 TCP/IP를 통해서 데이터베이스 서버에 접근한다. 다음절은 두 가지 인터페이스 방식을 자세히 설명한다.

3.1 ContainerFeatureCollection 인터페이스

CFC는 그림 1과 같이 클라이언트와 서버간의 인터페이스를 정의한다. 클라이언트는 먼저 바인딩을 통해 CFC를 생성 시켜주는 Factory 객체인 CFCF (ContainerFeatureCollectionFactory)를 찾아 CFCF의 참조를 얻게 된다. 얻어진 참조를 통해 Feature의 타입과 속성에 대한 정보를 가지고 create를 호출하게 되면, CFCF는 CFC를 생성한 후 참조를 반환한다. 클라이언트는 얻어진 참조를 통해서 Feature를 추가, 삽입, 삭제, 변경의 작업을 수행할 수 있다.

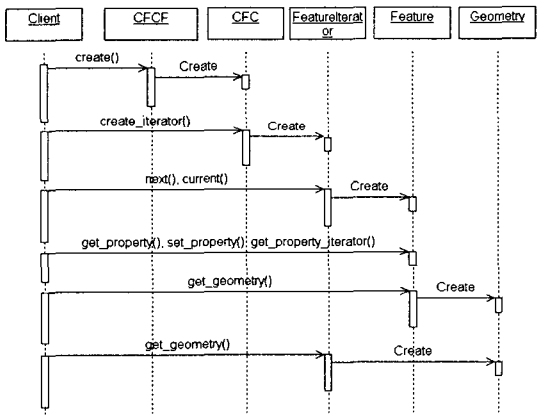

그림 1. ContainerFeatureCollection을 통한 접근 방법

그림 1은 FeatureIterator를 통해서 Feature에 접근하는 인터페이스를 보여주고 있다. CFC에서는 타입과 속성정보에 해당하는 Feature들을 식별하게 되고, 이 정보를 클라이언트의 요구에 의해 제공하게 된다. FeatureIterator에서는 식별된 Feature들의 세부 정보를 클라이언트에 제공하게 된다.

3.2 QueryableContainerFeatureCollection 인터페이스

QCFC는 그림 2와 같이 클라이언트와 서버간의 인터페이스를 정의한다. 클라이언트는 먼저 바인딩을 통해 QCFC를 생성 시켜주는 Factory 객체인 QCFCF (QueryableContainerFeatureCollectionFactory)를 찾아 CFCF의 참조를 얻게 된다. 얻어진 참조를 통해 Feature

의 타입과 속성에 대한 정보를 가지고 create를 호출하게 되면, QCFCF는 QCFC를 생성한 후 참조를 반환한다. 클라이언트는 얻어진 참조를 통해서 SQL 질의문을 가지고 evalute를 호출하게 되면, QCFC는 데이터 베이스를 참조하여 해당 질의의 결과로 QRSI(QueryResultSetIterator)를 생성하고, 참조를 반환한다. 클라이언트는 반환된 QRSI를 통해서 Feature를 접근할 수 있다. 위의 그림은 질의를 통해서 Feature에 접근하는 인터페이스를 보여주고 있다. QCFC는 질의 결과에 해당하는 Feature들을 식별하게 되고, 이 정보를 클라이언트의 요구에 의해 제공하게 된다. QRSI에서는 식별된 Feature들의 세부 정보를 클라이언트에 제공하게 된다.

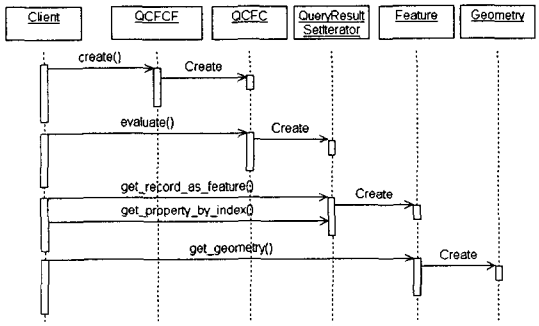

그림 2. QueryableContainerFeatureCollection을 통한 접근 방법

4. 구현

4.1 기존 접근 방법

OpenGIS simple feature specification for CORBA 인터페이스에는 FeatureIterator, FeaturePropertySetIterator, PropertyDefIterator, QueryResultSetIterator, 그리고 GeometryIterator, 총 5개의 Iterator 인터페이스가 존재한다. Iterator 인터페이스는 각각의 객체들이 여러 개가 존재하는 상황에서 각각의 객체들을 순차적으로나 비순차적으로 개개의 객체들에 접근하는 방법을 제공하는 인터페이스이다. 이 논문에서는 5개의 인터페이스 중에 FeatureIterator 인터페이스에 대해서만 다룬다.
이번 장에서는 두 가지 인터페이스 방식에서 사용한 Iterator 인터페이스와 기존 접근 방법을 개선한 방법을 설명한다.

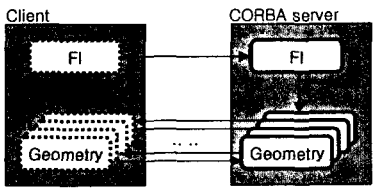

그림 3. 기존 접근 방법

그림 3은 기존 FeatureIterator 인터페이스를 통해서 각각의 Feature에 접근하는 방법을 보여주고 있다. FI 객체는 해당하는 Feature들의 정보를 가지고 있다. 클라이언트는 FI의 멤버 함수인 get_geometry()를 통해서 Geometry를 접근하는 인터페이스를 가지고 있다. get_geometry는 현재 포인터 위치의 Feature를 Geometry 타입으로 얻고 포인터를 증가시킨다. 그림 3과 같이 CORBA 서버에 Feature, Feature들의 집합, 또는 Geometry 객체를 생성하게 한다. 그래서 클라이언트를 통해서 접근한 Feature들에 대해서 다시 한번 추가의 접근을 요구하게 된다. 다음은 이 방법을 개선한 방법에 대해 자세히 설명한다.

4.2 개선된 접근 방법

OpenGIS simple feature specification for CORBA에는 WKS(Well-Known Structure) 타입이 존재한다. Feature나 Geometry 객체는 CORBA 서버에 객체 형태로 존재하지만, WKS 타입은 실질적인 값을 갖는 구조체이다.
표 1은 스펙의 IDL 파일의 일부이다. WKSPoint 타입은 x와 y의 좌표 값을 갖고, WKSLinearRing은 WKSPoint의 집합으로 이루어진다. WKSPolygon은 외부의 경계와 내부의 홀들로 구성된다. 또, WKSGeometry는 이러한 모든 구조체들을 갖는 타입으로, WKSGeometry 안에는 또 다른 WKSGeometry의 집합을 갖는 WKSCollectionType이 존재한다.

표 1. IDL 스펙의 일부

```
// Well-known Structures
struct WKSPoint {
   double x;
   double y;
};
typedef sequence<WKSPoint> WKSLinearRing;
typedef sequence<WKSLinearRing> WKSLinearRingSeq;
struct WKSLinearPolygon {
   WKSLinearRing externalBoundary;
   WKSLinearRingSeq internalBoundaries;
};
union WKSGeometry
switch(WKSType) {
   case WKSPointType :
      WKSPoint point;
   case WKSLinearRingType :
      WKSLinearRing linear_ring;
   case WKSLinearPolygonType :
      WKSLinearPolygon linear_polygon;
   case WKSCollectionType :
      sequence<WKSGeometry> collection;
};
```

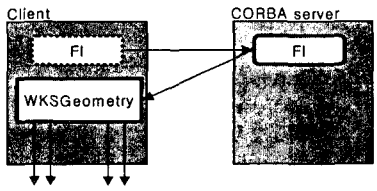

그림 4. 개선된 접근 방법

CORBA 서버가 Feature나 Geometry 객체를 생성하고, 이것을 클라이언트가 이용하기 위해서는 클라이언트는 네트웍을 통해서 다시 서버에 접속해야만 한다. 그림 4는 FI가 새로운 Feature 나 Geometry 객체를 생성하는 것이 아니고 WKSGeometry 타입으로 Feature의 값을 클라이언트에게 바로 전달함으로써 클라이언트가 다시 접근하게

하는 비용을 줄일 수 있다. WKSGeometry에는 WKSCollectionType의 형태로 또 다른 WKSGeometry들이 존재한다. 그래서 클라이언트는 FI에 한번의 접근으로 해당하는 Feature들을 모두 읽어서 처리 할 수 있게 된다.

5. 실험 및 결과 분석

이 논문에서 제안한 FeatureIterator 의 get_WKSGeometries는 CORBA 서버에 객체를 생성하지 않고 객체의 집합을 값으로써 바로 클라이언트에 전달하여, CORBA의 단점인, 객체에 접근하는 비용을 줄일 수 있다. 실험 데이터는 도시의 건물과 도로를 나타내는 Polygon들을 대상으로 했으며, 각각의 튜플은 ID, CODE, SHAPE으로 구성되었다. 표2는 레코드의 스키마이다.

표 2. 레코드 스키마

이름	널?	유형
ID	NOT NULL	NUMBER
CODE		VARCHAR(10)
SHAPE		MDSYS.SDO_GEOMETRY

표 3. 실험 결과

접근 방법 / 데이터의 양	기존 방법 get_geometry()	개선 방법 get_WKSGeometries()
200	08:14	00:19
400	15:54	00:20
600	20:48	00:27
800	27:24	00:30
1000	33:15	00:43

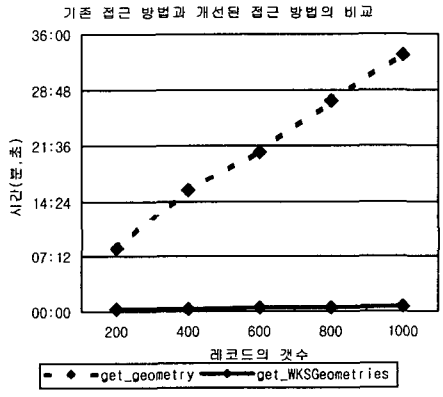

그림 5. 기존 방법과 개선된 접근 방법의 비교

구현 및 테스트 시스템은 오라클 서버로 Pentium 180, Ram 40MB, Alzza Linux 6.2에서 오라클 8i를 사용하였으며, CORBA 서버로는 Pentinum 400, Ram 128MB, Window 2000, Visibroker for cpp 3.3.3 그리고, 클라이언트로는 Pentium 400, Window 환경에서 실험을 하였다.

이 논문에서는 FeatureIterator 인터페이스의 get_geometry와 이 논문에서 제안한 get_WKSGeometries 두 개의 인터페이스만을 실험했다. 표 3은 레코드의 개수를 200개, 400개, 600개, 800개, 1000개를 대상으로 실험한 결과 값이다. 걸린 시간은 초 단위이다. get_geometry의 경우는 레코드의 양에 따라 걸린 시간이 증가했고, get_WKSGeometries의 경우는 레코드의 양에 따라 걸린 시간이 거의 변화가 없음을 알 수 있다. 그림 5는 두 가지 인터페이스 실험 결과에 대한 비교 도표이다.

6 결론

이 논문에서는 OpenGIS simple feature for CORBA의 구현 명세에서 FeatureIterator 인터페이스의 새로운 인터페이스를 제안하고 구현하였다. FeatureIterator는 current(), next(), next_n() 그리고 get_geometry()를 이용하여, Feature나 Geometry 객체를 생성하고 접근할 수 있게 한다. 기존의 get_geometry는 하나의 객체를 CORBA 서버에 생성시키고, 클라이언트가 다시 접근하기 때문에 접근 비용이 많이 요구된다. 이 논문에서 제안된 get_WKSGeometries 인터페이스를 이용하여, CORBA 서버에 객체를 생성하지 않고, WKS 형태의 값으로 클라이언트에게 Feature에 해당하는 값들을 전달할 수 있는 인터페이스를 제안하고 구현하였다. 그래서 get_WKSGeometries 인터페이스를 이용하여, CORBA의 단점인 접근 비용을 줄일 수 있고, 성능이 향상됨을 보였다.

참고 문헌

[1] Open GIS Consortium, Inc., The OpenGIS Guide, 1998.
[2] Open GIS Consortium, Inc., The OpenGIS Abstract Specification Model, Version 3, 1998.
[3] Open GIS Consortium, Inc., OpenGIS Simple Features Specification for SQL, Revision 1.0, 1998.
[4] Open GIS Consortium, Inc., OpenGIS Simple Features Specification for OLE/COM, Revision 1.0, 1998.
[5] Open GIS Consortium, Inc., OpenGIS Simple Features Specification for CORBA, Revision 1.0, 1998.
[6] Open GIS Consortium, Inc., OpenGIS Project Document Number 96-015R1, The OpenGIS Abstract Specification Revision 1, 1996.
[7] OMG, Common Object Request Broker Architecture and Specification, Revision 2.2, 1998
[8] Yooshin Lee and Ling Liu. Calton Pu. Towards Interoperable Heterogeneous Information Systems: An Experiment 사용자 매뉴얼 Release 1.0, 1998. Using the DIOM Approach. In the Processings of the 12th ACM Symposium on Applied Computing (ACM SAC'97) Special track on Database Technology. 1997.
[9] 한국 전산원, Internet GIS의 데이터 공유 표준연구, 1998.
[10] 한국통신, GEUS/X
[11] Asuman Dogac Cevdet Dengi M.Tamer Ozsu, Building Interoperable Databases on Distributed Object Management Platforms, Communications of the ACM, 1996