GOVP1199800998

최종보고서

GIS 자료 편집 및 통합 관리 기술 개발

Development of GIS Data Editing and Integrated Management Technologies

연 구 기 관

시 스 템 공 학 연 구 소

과 학 기 술 처

배 포 선

사본 번호	부수	배 포 처
1/100	1	시스템공학연구소 영구보존용
2/100 ~ 3/100	2	시스템공학연구소 도서과 보관용
4/100 ~ 5/100	2	시스템공학연구소 연구관리과 보관용
6/100 ~ 10/100	5	시스템공학연구소 지리정보시스템연구실 보관용
11/100 ~ 92/100	90	기타 배포기관

제 출 문

과학기술처 장관 귀하

본 보고서를 "GIS 자료 편집 및 통합 관리 기술 개발"의 최종보고서로 제출합니다.

1997. 11.

연 구 기 관 명 : 시스템공학연구소

연구책임자 : 책임연구원 이 종 훈

책임연구원 양 영 규

선임연구원 이 기 원

하 영 렬

연 구 원 이 호 근

김 경 호

김 광 수

김 민 수

여 백

요 약 문

I. 제 목

GIS 자료 편집 및 통합 관리 기술 개발

II. 연구의 목적 및 중요성

본 연구 과제의 최종 목적은 일반적인 GIS 의 자료 편집 기능이 아니라 네트워크를 통하여 서버에 저장되어 있는 GIS의 자료를 편집하고 관리할 수 있는 기술을 개발하는데 있다.

현재 국내외적으로 네트워크를 기반으로한 GIS S/W의 개발 및 관련 연구는 어느 정도 시작단계에 머물러 있는 수준이나 실제 GIS는 그 다양한 활용범위 만큼이나 실제 이용자에 의하여 제기되는 기술 개발 및 가시적 성과 물에 대한 요구가 급증하고 있다. 그러나 대부분의 GIS이용자들은 GIS 자료구축과정에서 실제 상당한 인적/물적 노력을 투입해야 하는 실정이며 이는 GIS의 다양하고 성공적인 운영에 커다란 장애가 되고 있다. 따라서, Internet과 같은 네트위크상에서 일반 자료나 정보를 GIS자료화하여 공급관리시스템이 갖는 경제적 가치는 이루 말할 수 없다.

GIS자료 입출력, 편집, DB관리용 시스템은 GIS자체가 자연 환경과 사회 문화적인 실제 양상을 전산화하는 주요한 수단으로서의 충분한 기능을 담당하고 있으므로 어떠한 측면에서는 본 연구에서 개발되는 네트워크기반 시스템은 이러한 기본 기능에 가장 충실한 역할을 담당하며 이는 다양한 자료의 효용성을 극대화시키는 데 크게 이바지하여 궁극적으로는 사회적 안정이나 도농간의 문화적

격차를 줄이는 데 가장 크게 기여할 수 있는 소프트웨어로 볼 수 있다.

III. 연구의 내용, 범위 및 성과

첫 째, 네트워크를 통하여 서버내의 GIS 자료를 입출력 할 수 있는 기능을 구현하였다. 따라서, JAVA를 이용하여 GIS 자료의 내부 포맷을 정의 및 원하는 형태로의 자료 변환 그리고 다양한 형태의 출력이 가능하게 되었다.
둘 째, 서버내의 GIS 자료를 편집할 수 있는 기능을 구현함으로써 GIS 자료의 오류를 수정하고 자료의 생성, 삭제, 변경 기능이 개발되었다. 또한, 3차원 객체의 위상 관계를 형성할 수 있게 되었다.
셋 째, 공간 데이터베이스의 설계 및 관리 기능과 관련하여 3차원 데이터에 관한 공간 데이터베이스 스키마를 정의하였으며, 외부 데이터베이스와의 인터페이스를 구현하였다.

각 부문 별 연구 결과는 다음과 같다.

1) 네트워크 상의 GIS 자료 입출력 기능

가) 네트워크 GIS를 위한 GIS 자료 내부 포맷 정의

- o SVF(SERI Vector Format) 정의
- o VPF(Vector Product Format)의 분석

나) GIS의 자료 변환 기능

- o 자료 변환
- o DXF 형식
- o SVF와 DXF 간의 자료 변환

2) 네트워크를 통한 GIS 자료의 편집 기술 개발

가) 입력 자료 오류 검색 및 수정 기능

o 댕글 길이 o 위드 오차

o 노드 매치 오차 o 서로 교차하는 체인

나) 공간 객체 식별 기능

다) 공간 객체 편집 기능

o 생성 o 삭제 o 이동 o 복사

라) 위상 관계 형성 기능

o 위상 관계의 종류

o R*-tree 공간 색인을 이용한 위상 관계 형성

o Winged-edge 위상 관계

3) 네트워크 GIS 자료를 위한 공간 데이터베이스 설계 및 관리 기능 개발

가) 3-Tier 공간 데이터베이스의 설계 및 구현

o 3-Tier 공간 데이터베이스

o JDBC를 이용한 데이터베이스 연결

o DBMS 인터페이스

나) 서버 데이터베이스의 구성

o 데이터베이스 관리기

o 공간 색인 관리기

4) AWT(Abstract Window Toolkits)를 이용한 사용자 작업 환경 구현

가) 사용자 인터페이스 환경 구축

o 자바 프로그래밍 언어 설치하기

o 자바 응용프로그래밍 인터페이스(Application Programming

o 좌표변환

나) 사용자 인터페이스 설계 및 구현

o 사용자 인터페이스의 설계 (GUI Design)

o Zone 사용자 인터페이스의 구성 요소

5) GIS 응용 사례

가) 도시 재난 관제 시스템

o Dispatch System

o UDMS(Urban Disaster Management System)

나) 네트워크 상의 공간 분석 및 통계처리

다) 입력 자료 모델

Summary

The purpose of this research is to develop practical technologies for editing. compling, and integrated managing of large volume of spatial data in network server. Current domestic research level related to Web-based GIS technologies is just beginning stage, but this unmatured stage is normally unusual case, even to developed other countries such as US and Canada, due to emerging computer science and software development of technologies themselves. However, requirement at user-level towards GIS fields is also varied and advanced, with the recongition of linkage of Internet/Intranet environment with advantage spatial database engine and interface over custodial database management.

In this research, development of core engines among the whole Web-based GIS software system is the main target: GIS data input/output modules in internet netwotk environment, internal GIS format definition in consideration of spatial data standard, spatial data transformation modules based on Java, GIS data editing and management functionalities, interface modules with external database system, and so forth.

As results, the following achievements are obtained and can be itemized,

1) GIS data input/output modules in internet netwotk environment

o Internel GIS data format definition

- SVF(SERI Vector Format) Definition
- VPF(Vector Product Format) Analysis

o GIS Data Transformation

- Format conversion between DXF and SVF
- Map Coordinate Transformation or Cartographic Transformation

2) GIS data editing technology through network

○ Error rectification

- dangling arc
- Weird arc
- Node unmatching error
- intersect arc
- undershoot or overshoot arc

○ Spatial Object Indentification

○ Spatial Object Editing

- Creating
- Deleting
- Moving
- Copying

○ Topology Building

- Types of Topology
- Automated topology building based on R*-tree
- Winged-edge topology

3) Database Design and Implementation of Management Technology

○ 3-tier Database Design and implementation

- 3-tier Database Architecture
- Linkage of Spatial Database using JDBC
- DBMS Interface

○ Server-side Database

- Database Manager
- Spatial Indexing Manager

4) Graphical User Interface using AWT(Abstract Window Toolkits)

○ User Interface

- Java programming environment
- API(Application Programming Interface)
- GUI Design
- Components in Zone interface

5) GIS Application

○ UDMS(Urban Disaster Management System)

- Dispatch System
- UDMS application

o Spatial Analysis and Statistics on the Network

o Description of GIS Data for UDMS

여 백

CONTENTS

여 백

목 차

여 백

그림 목차

여 백

표 목 차

여 백

제 1 장 서 론

제 1 절 연구 일반

1. 연구의 필요성

현재 벡터형 GIS S/W의 가장 큰 핵심 요소는 자료 관리시스템과 분석 시스템이라 해도 과언이 아니다. 후자의 경우에는 전 세계적으로 그간 상당한 기술력이 축적되어 있으며 현재 국내에서 이용되고 있는 수입 GIS S/W는 이러한 종합적인 연구 개발의 산물이다.

물론 GIS내의 자료관리기능에 관한 연구도 많이 이루어지고 중요한 과제이나 현재의 기술수준은 인터넷과 같은 세계적인 네트워크를 효율적으로 이용하지 못하고 있으며, 이를 위해서는 기존의 GIS S/W구현 방법과 기본 틀을 달리해야 할 필요가 있음은 주지의 사실이다. 이러한 목적을 위하여 JAVA와 같은 인터넷 응용소프트웨어 개발 언어가 효과적으로 이용될 수 있다. 그러나 Internet 기반 GIS S/W자체가 현재 전세계적으로 핫 이슈이며 개발 언어가 다른 연구과제이기 때문에 본 연구에서 구현되는 결과물은 전산 분야 전반에 여러모로 시사점을 줄 것으로 생각되며 NGIS S/W 개발 과제의 DB 관련과제, Mapping 관련 과제에도 중요한 기술적 증진을 도모하는 촉진제 역할을 할 것으로 판단된다.

현재 국내외적으로 네트워크를 기반으로한 GIS S/W의 개발 및 관련 연구는 어느 정도 시작 단계에 머물러 있는 수준이나 실제 GIS는 그 다양한 활용 범위 만큼이나 실제 이용자에 의하여 제기되는 기술 개발 및 가시적 성과물에

대한 요구가 급증하고 있다. 그러나 대부분의 GIS이용자들은 GIS 자료 구축 과정에서 실제 상당한 인적/물적 노력을 투입해야 하는 실정이며 이는 GIS의 다양하고 성공적인 운영에 커다란 장애가 되고 있다. 따라서 Internet과 같은 네트워크상에서 일반 자료나 정보를 GIS 자료화하여 공급 관리시스템이 갖는 경제적 가치는 이루 말할 수 없다.

GIS자료 입출력, 편집, DB관리용 시스템은 GIS자체가 자연 환경과 사회 문화적인 실제 양상을 전산화하는 주요한 수단으로서의 충분한 기능을 담당하고 있으므로 어떠한 측면에서는 본 연구에서 개발되는 네트워크기반 시스템은 이러한 기본 기능에 가장 충실한 역할을 담당하며 이는 다양한 자료의 효용성을 극대화시키는 데 크게 이바지하여 궁극적으로는 사회적 안정이나 도농간의 문화적 격차를 줄이는 데 가장 크게 기여할 수 있는 소프트웨어로 볼 수 있다.

2. 연구의 목적 및 내용

가. 연구의 목적

GIS 자료 편집 및 통합 관리 기술 개발 과제에서는 일반적인 GIS 의 자료 편집 기능이 아니라 네트워크를 통하여 서버에 저장되어 있는 GIS의 자료를 편집하고 관리할 수 있는 기술의 개발을 최종 목표로 한다.

본 과제에서 개발되는 주요 기술들은 다음과 같다.

o 네트워크를 통하여 서버내의 GIS 자료를 입출력 할 수 있는 기능을 구현한다.

- JAVA를 이용하여 GIS 자료의 내부 포맷을 정의 및 원하는 형태로의 자료 변환 그리고 다양한 형태의 출력을 구현한다.

o 서버내의 GIS 자료를 편집할 수 있는 기능을 구현한다.

- GIS 자료의 오류 수정 및 생성, 삭제, 변경 기능을 구현하고 또한 3차원 객체의 위상 관계를 형성한다.

o 공간 데이터베이스의 설계 및 관리 기능을 구현한다.

- 3차원 데이터에 관한 공간 데이터베이스 스키마를 정의하고 외부 데이터베이스와의 인터페이스를 구현한다.

나. 연구의 내용

o 네트워크 상에서의 GIS 자료의 입출력 기능 개발

- 네트워크 GIS를 위한 GIS 자료의 내부 포맷의 정의
- AWT (Abstract Window Toolkit)를 이용한 사용자 작업 환경의 구현
- GIS의 자료 변환 기능의 구현
- 다양한 출력 형태의 지원
- 입력 자료의 오류 검색 및 수정 기능의 구현

o 네트워크를 통한 GIS 자료의 편집 기능의 개발

- 공간 객체의 생성, 삭제, 변경 기능의 구현
- 공간 객체의 접합, 분할 기능의 구현
- 위상 관계 형성 기능의 개발

- 공간 데이터베이스를 위한 스키마의 설계
- 공간 데이터베이스의 입·출력 및 갱신 기능의 개발

o 네트워크 GIS 자료를 위한 공간 데이터베이스 설계 및 관리 기능의 개발

- JDBC(JAVA DB Compiler)를 이용한 DB 관리 기능 개발
- 외부 DBMS와의 인터페이스 구현

o 네트워크 GIS를 제공하는 Web Site 구축

제 2 절 Internet GIS

1. 개 요

현재 미국과 같은 GIS 선진국에서는 국가 GIS 사업을 통하여 구축된 지형공간정보의 유통을 위한 Internet의 활용과 연계된 연구과제가 활발히 진행중에 있다. 미국의 공간 정보 유통기구(Geospatial Data Clearinghouse)와 관련된 표준, 실험연구 Project, 응용 Project들을 연결시켜보면 미국에서는 공간정보의 유통은 네트워크를 이용하는 것이 일반적임을 알 수 있다.

특히 인터넷을 이용하는 것이 큰 흐름으로 포착된다. 최근 개발된 웹(WWW) 브라우저의 등장으로 일반인들의 인터넷 이용은 급증하였고 이에 따른 기술의 개발 역시 그 속도가 상당히 빠르다. 그래서 외국에 이러한 웹 브라우저를 이용한 공간정보의 검색 및 구득을 위한 연구가 활발히 이루어지고 있다. 그러나, 웹 브라우저를 이용한 공간정보의 검색에는 현실적으로 상당한 어려움이

있어 이를 보완하기 위한 연구가 활발히 이루어지고 있다. 미국의 WWW와 Z39.50의 결합 연구가 그 좋은 예이다. 기술적으로 웹 서버는 실질적인 수치지리 정보를 사용자가 질의하는 다양한 내용에 대해 제공하는 것이 어렵다. 반면 실질적인 공간정보 검색 질의는 Z39.50 규약을 사용하는 것이 거의 일반적이다.

현재 Internet이 주로 Image의 데이터를 주로 다루고 있기 때문에 Vector를 비롯한 여러 형태의 공간정보를 공간적 인덱스를 통해 받으려면 새로운 Program의 개발이 있어야 할 것이다. 따라서 최근의 Java 프로그래밍 언어를 이용해 웹 클라이언트와 Z39.50 서버 간의 질의와 응답을 가능케 하는 연구들이 이루어져 미국의 경우는 어느 정도 표준 소프트웨어가 나올 정도로 그 진척이 이루어졌다. 그리고 Alexandria 전자도서관 프로젝트와 같이 Internet을 통하여 공간 정보 유통기구와 결합하는 형식을 지향하고 있다.

WWW를 이용한 GIS 정보유통의 요건은 다음과 같다.

- o 인터넷 상에서 "검색"과 "브라우저" 양자 모두 지원
- o Z39.50 통신규약 : 네트워크 정보 검색을 위한 ANSI/ISO 표준
- o 다중 접속 경로 전략
- o 예측 가능한 형태의 검색 결과
- o 소프트웨어 개발 및 업데이트에 대한 투자 최소화
- o 클라이언트 프로세서에서 다수의 서버로 분산 검색 제공
- o 지원 자료형태 및 추출(공간 사상으로 수집수준 하향) 수준의 확장

가. Internet GIS 관련 표준 및 규약

Internet GIS 구축과 관계있는 표준 및 규약으로는 OGIS, CORBA,HTTP, Z39.50, JAVA, Metadata Content Standard 등이 있다. 이들 표준 및 규약의 목적은 공간 정보의 Interoperability (상호 가동성)를 구현하고 이를 Internet을 통한 검색 및 유통을 가능하게 하기 위함이다.

Z39.50은 Internet을 통한 정보 Retrieval Service와 Protocol의 ANSI / NISO 표준이다. Z39.50은 1988년에 처음 정해졌으나 제조업체들에 의해서 광범위하게 사용되지는 않았다. 이 Z39.50 표준에 기반을 둔 주목할 만한 사실은 WAIS (Wide Area Information Services)의 개발이다. 1988년의 이 표준은 ANSI/NISO Z39.50-1992 (Version 2)로 Update되면서 ISO의 Search and Retrieval (SR) Service와 Protocol 정의 ISO 10162/10163와 제휴하게 된다. 다음 Version인 (Version 3)은 1994년 ANSI에 의해 공식적으로 받아들여 졌다.

나. OGIS(Open Geodata Interoperability Specification)

OGIS는 최근에 만들어진 여타 분산 객체지향 Software와 비교하면 객체 기술(Object technology)의 적용에 따른 기본적인 구조나 혜택은 비슷하다. 그러나, OGIS는 전 지구적인 수준 또는 국가 정보의 Infrastructure 수준에서의 지리 정보 체계에 대하여 거대 규모로 이루어지는 최초의 객체 기술 적용이다.

OGIS 연구 과제의 목표는 이질적인 지리 공간자료의 분산 처리에 대한 상호 가동성 표준의 개발을 촉진하는 것으로서 다음과 같은 세 가지의 특성을 만족시키는 표준을 마련하는 것이다.

o 현존하거나 잠재적인 Spatio-temporal data 및 Process model의 표편화와 단일화

o OGIS data model을 구축하는 주요 DB 언어 각각의 사양

o OGIS process model을 구축하는 주요 분산 처리 환경 각각의 사양

1) OGIS 프로젝트의 기술적 목적

OGIS 프로젝트의 목적은 어플리케이션 개발자로 하여금 단일 환경과 단일의 작업 흐름(work flow) 내에서 네트워크 상에서 활용 가능한 모든 형태의 지리 자료나 지리 공간적 기능 또는 과정을 사용할 수 있게 하는 기술의 파악에 있다.

전통적인 지리 정보처리 어플리케이션을 사용하는 조직들과 기타의 전통적인 정보 기술은 몇가지 독특한 독자적 어플리케이션을 가지고 있는데, 이들은 보통 플랫폼 의존형이고, 전산 처리와 데이터 자원을 공유에 있어서는 매우 제한적인 능력을 가진다. 종종 어플리케이션 사이에는 기능과 데이터베이스에 있어서 중복이 발생하였으며, 사용자 인터페이스의 다양성으로 인해 과다한 교육 훈련이 필요하였다. 또한, 이러한 어플리케이션들은 필요에 따라 새로운 데이터 유형과 방법을 손쉽게 수용할 수 있는 기능이 부족하였다. 이러한 단점은 지리정보처리 기술의 잠재력을 크게 제한하였다.

전통적인 지리 정보처리 기술과는 대조적으로, OGIS에서 정의된 개방형 지리 정보처리 기술은 소프트웨어 개발업자들이 다음과 같은 특징을 가진 지리 정보처리 어플리케이션과 소프트웨어 컴포넌트들을 구축할 수 있는 공통의 기술적 기반을 확립한다.

2) 기술적 목표의 달성 방법

OGIS 프로젝트의 기술위원회 과정은 OGIS의 목적과 목표에 관한 지리정보처리 커뮤니티의 다양한 시각을 통일하기 위해 노력하고 있다. 기술위원회 산하의 그룹들이 각각 다음의 주제를 다루고 있다.

- o 지리자료 모델의 통일.
- o 지리정보처리 서비스의 통일.
- o 커뮤니티 사이의 데이터 및 처리 자원의 공유.
- o 공통의 전반적인 지리정보처리의 틀 안에서 GIS, 원격탐사 및 기타의 지리정보처리 분야를 통합할 수 있는 통일된 접근 방법의 개발.

기술위원회 참여자는 사양 결정을 위한 회합과 저술 활동을 할 뿐 아니라 OGIS를 포괄하는 소프트웨어 개발 활동에 참여하고 있다. 그들의 다양한 경험은 OGIS 사양 개발에 크게 도움이 될 것이다. 기술위원회와 관리위원회 멤버들은 또한 국가적 또는 세계적 표준화 움직임에 관련되어 있어서 OGIS는 다른 표준화 활동을 반영함과 동시에 이에 영향을 미치고 있다.

3) OGIS의 범주

OGIS는 다양한 자료원으로부터의 지리 자료에 접근하고 사용함에 있어서 발생하는 문제들의 세 가지 측면 중 두 가지를 다룬다.

가) 연결 확보(Getting connected)

OGIS는 분산 컴퓨팅 플랫폼(DCP)의 영역--상이한 컴퓨터 사이의 상호

작용을 가능하게 하는 문제--은 다루지 않는다. DCP는 네트워킹, 상이한 컴퓨터 시스템 사이의 통신, 시스템 보안, 분산 자료저장, 기타 클라이언트/서버 플랫폼에 관한 문제를 취급한다. OGIS 사양은 이 분야에서 이루어지고 있는 어떠한 활동과도 중복되지 않는다. 이러한 과제들은 다른 기술 공급자들에 의해 계속 발전될 것이며 OGIS는 그 과정을 따라 이루어지는 기술기반에 의존하게 될 것이다. OGIS는 특정 DCP에 국한되지 않는다.

나) 서비스의 취득(Getting service)

이는 OGIS의 영역으로, 어플리케이션으로 하여금 지리자료를 운영, 제공, 처리하는 다른 어플리케이션들과 상호작용하게 하는 일이다. 이 측면은 서비스를 어떻게 제공할 것인지, 서비스를 어떻게 요구하도록 할 것인지, 특정 요구가 데이터 또는 데이터에 대한 작업을 요구하는지 혹은 두 가지 모두인지를 판단하는 방법을 다룬다. 이 범주는 표준 데이터 유형의 세트와 이러한 데이터 유형을 이용한 작업을 정의하고, 이에 따라서 지리자료의 공급자와 소비자들간의 상호 가동성을 위한 공동의 틀을 제공하게 된다. 또한 이는 서로 다른 정보 사회(여러 의미에서의 지리자료와 지리정보처리를 위한 사용자 모임)들 사이의 데이터 교환을 촉진시키는 서비스를 제공한다.

이러한 능력은 지리 자료 변환을 위한 공통의 데이터 모델의 정의와 이 정보를 이용한 작동 유형의 정의에 의존한다. 기존의 지리 자료와 GIS에 대한 막대한 투자의 효용성을 살리고, 지리자료 운영과 작동을 위한 새로운 방법 도입의 가능성을 열어두기 위해, OGIS는 데이터 저장 방식과 처리 방식에 대한 정의만을 다룬다.

다) 결과에 대한 이해(Understanding the result)

이 범주는 데이터 일치에 대해 공동의 이해가 걸려 있는 개인이나 그룹에 관계되는 부분이다. 그들은 데이터를 해석하는 틀--의도된 의미, 정확도, 보증의 수준 등--을 제공한다. 이들 데이터 내용의 정의는 OGIS 사양의 범주에서 벗어나는 것이기는 하지만, OGIS 사양은 공급자와 소비자들 사이에서 공유될 수 있는 이러한 정보의 틀을 제공할 필요가 있다.

4) 지리자료, 서비스 및 정보 공유를 위한 개방형 모델

OGIS의 틀은 다음과 같은 세 가지의 주된 부분으로 구성된다.

가) 개방형 지리자료 모델(The Open Geodata Model)

OGIS 사양 모델의 이 부분은 객체 기반(object-based) 및/또는 일반적 프로그래밍 기법을 사용하여, 보다 특수한 어플리케이션 영역에서의 지리자료 요구를 모델화하는데 활용될 수 있는 기본 지리자료 유형에 대한 일반적이고 공통적인 세트를 정의한다.

나) OGIS 서비스 모델(The OGIS Services Model)

이 부분은 다음의 작업을 수행하는데 필요한 서비스들을 정의한다.

- o 개방형 지리자료 모델 내에서 정의되는 지리자료 유형의 접근과 처리.
- o 공통의 지리특성 정의를 사용하는 사용자 커뮤니티 내에서 지리자료를 공유할 수 있는 기능과 상이한 지리특성 정의를 사용하는 사용자 커뮤니티 사이의 변환 기능.

다) 정보 커뮤니티 모델(The Information Communities Model)

이 모델은 다음의 항목들을 확립하기 위한 계획에서 개방형 지리자료 모델과 OGIS 서비스 모델을 채용한다. 이미 공통의 지리 특성 정의를 가지고 있는 지리 자료 생산자와 사용자의 커뮤니티를 위한 것으로, 지리특성 정의를 효율적, 효과적으로 유지 관리하고 이 정의에 맞게 구축된 데이터 셋의 목록 작성과 공유를 위한 방법을 확립한다.

상이한 지리 자료 사용자 커뮤니티들을 위한 것으로, 지리특성 정의의 차이를 극복하고 지리 자료를 공유하게 만들 수 있는 효율적이고 정확도가 확보되는 방법을 확립한다. 예를 들면, 도시 공학자, 지질학자, 농업 경영학자들은 각자의 전문 목적에 따라 각각 다르게 분류된 토양 유형을 사용한 토양 데이타를 공유할 수 있는 방법을 찾으려할 수도 있으며, 정보 커뮤니티 모델은 이러한 경우에 적용될 수 있는 상이한 지리 특성 목록 사이의 자동 변환을 위한 계획을 정의한다.

다. CORBA

CORBA(Common Object Request Broker Architecture)는 오늘날 급격하게 증가하고 있는 Hardware와 Software 사이에서 요 구되고 있는 Interoperability(상호 가동성)에 대한 OMG(Object Management Group)의 사양(Specification)이다. CORBA는 객체 지향 표준과 상호 가동성으로 가는 출발점에 있다. CORBA를 사용하여 User들은 얻고자 하는 정보가 Network상의 어디에 존재하며 정보가 있는 Platform의 Hardware/Software를 알 필요가 없이 자료에 접근할 수 있다. 간단히 말해서, CORBA는 Application이 어디에 위치해 있으며 누가 만들었는 지에 상관없이 서로간의 통신을 가능하게 한다.

1) CORBA 1.1와 CORBA 2.0

CORBA 1.1은 1991년에 Object Management Group (OMG)에 의해서 도입되어 Object Request Broker (ORB)의 구체적인 구축 범위 내에서 Client/Server 객체 상호 작용을 가능하게 하는 Interface Definition Language (IDL)와 Application Programming Interfaces (API)가 정의되었다.

1994년에 채택된 CORBA 2.0은 어떻게 서로 다른 제조업체의 ORB가 Interoperable할 수 있는가를 명시함으로써 진정한 상호 가동성을 정의하였다. CORBA 2.0의 사양에 대한 Full-text는 "http://www.omg.org/corbask.htm"에서 구할 수 있다.

2) ORB

ORB는 객체간에 client-server relationship을 성립시키는 middleware이다. ORB를 이용하여 Client는 System 본체내에 있을 수도 있고 Network 상에 있을 수도 있는 Server object에 대해 Method를 명쾌하게 요청할 수 있다. Client는 객체가 어디에 위치 해 있는지, Programming 언어가 무엇인지, Operating system이 무엇인지 등의 객체 Interface의 부분이 아닌 System의 모습에 대해 알 필요가 없다.

따라서 ORB는 이질적인 분산 환경에 있는 서로 다른 기종의 Application 사이에 Interoperability를 제공하고 다중 객체 System 사이를 매끄럽게 연결시킨다.

ORB에서 Protocol은 IDL을 통해 정의된다. IDL은 language-independent specification이다. IDL은 Programmer로 하여금 가장 적절

한 운영 체제, 실행 환경, 심지어 개발 언어까지도 선택할 수 있도록 도와준다. 더욱 중요한 점은 IDL이 기존 Components의 통합을 가능하게 한다는 점이다. ORB 기반에서는 Application 개발자들은 똑같은 IDL을 사용하여 Legacy components들을 간단히 모형화하고 Object를 감싸듯이 Coding하면 된다.

라. JAVA

자바(Java)는 간단하고 객체 지향적이며 Platform-independent, multi-threaded, dynamic한 특징을 갖고 있는 Programming 언어이다. 특히 자바의 Platform-independent한 특성때문에 Internet, intranet, 또는 다른 복잡한 분산 Network 환경하에서 Application을 손쉽게 작성할 수 있는 도구로 부각되고 있다.

Hotjava는 SUN Microsystems사에서 개발한 Browser로서 Java applet이나 Script를 인식하여 보여줄 수 있다. 최근에는 Netscape나 MS-Internet Explorer로도 Java applet을 실행할 수 있다.

2. Internet GIS 개발 현황(State of the Arts)

가. 비 상업용 Internet GIS

1) TAGIS(Technical Application & GIS) : Java Map Interface

o URL http://poca.osmre.gov

o 기 능 : query tool(cgi 이용), zoom box, measure line(on-screen measurement, select box, select circle, pan, zoom in/out 부분 강조

o Data Layer : 도로(1:100,000 scale Tiger data), 강, Watershed, 24K Quad, 고도(USGS-1 degree DEM), 행정 경계
(data layer 추가/삭제 가능)

o 좌표 체계 : UTM17

o Dataset : ASCII File

o Frame : Upper Frame/Lower Frame(검색 용)

o 화면 배경 : gif

o 검색 도구 : CGI & HTML

2) IDGIS(Interactive Distributed GIS)

Java Applet을 이용한 Web Browser용 분산 GIS로서 검색과 공간 분석 기능을 가지고 있음

o URL http://starr-www.tamu.edu/choo/idgis/intro.html

o 기 능 : 지도 선택, 확대/축소, 선분 측정, Tin/VRML(개발 중)

o Data Layer : 도로, Watershed, 강, 토양, Trail, 수계
(추가/삭제 가능)

o 좌표 체계 : Map X,Y

o Frame : Multi-frame(Java Applet)

o DBMS 지원 : Oracle 7.x dbserver(ArcView와 Arc/Info server 지원)

o 지원 환경 : SunSparc IPX, Win95/NT with JDK, Java Script, JDBC-ODBC, MS-Active X

3) IRIS : Java Applet을 이용한 수치 통계 분석 및 도표 처리

o URL http://allanon.gmd.de/and/and.html

o 기 능 :

- 지도 확대 기능 : 배율 선택, 별도 Frame 사용, 특정 layer 강조
- Map Layer 도시 : 도시 순서 변경 기능
- 자료 검색, 계산 및 통계 처리

o 응용 분야 : data visualization

- 인구 동태 분석
- 유럽 국가별 인구 및 경제
- 산업 성장률 분석

나. 상업 용 Internet GIS

1) GEO/WEB

Java 언어로 개발되어 인터넷과 인트라넷 기반의 네트워크 컴퓨팅 환경 하에서 운영되는 클라이언트-서버 지리정보 시스템

o URL http://www.kgi.co.kr

o 기능 : 출력, 색인, 레이어 처리, 테이블 처리, 질의, 좌표계 출력

o Client/Server 구조

2) AtiveMaps

Java를 이용한 online 지도 도시 및 검색,GIS 시스템 개발 도구

o URL http://internetGIS.com

o 기 능 : zoom in/out, zoom area. unzoom, pan, identify, query, clear, label, hlink, help

o Dataset : ArcView shapefile, MapInfo file, DBF file, ASCII file, DXF file (사용자 정의 속성 data 추가 가능)

o Client/Server 구조

o 환 경 : Win95/NT, Solaris 용 Web-browser

o 응용 분야 :

- 미국 주별 조세 부담 현황, 범죄발생 현황, 현금 자동지급기 현황
- 주별 미국 대통령 선거 결과(1996)
- Internet T3 Line Backbone

o 좌표 체계 : Map X,Y

2) MapQuest : Java를 이용한 관광 안내용 지도 검색

o URL http://www.mapquest.com

o 기 능 : 확대/축소(단계 : 국가 > 지역 > 시 > 주소), 위치 및 정보 검색, 길 안내, 사용자 용 지도 제작 기능

o Frame : Single

4) CARIS Internet Server : Graphic manipulation & querying

o URL http://caris0.univerdal.ca/demo/login/index.html

o 기 능 : 확대/축소, 부분강조, 검색(form-based/feature-based), 검색 수
수료 부과

o Dataset : CARIS file

o Frame : Single

o 화면 배경 : Raster Image

o Data Layer : 거리, Property lines & PID, Raster Backdrop

o 좌표 체계 : 중심 N/W만 표시

o 응용 분야 :

- New Brunswick 관광 안내도

- New Brunswick 자원 분포

- New Brunswick Fredericton Property Maps

- Canadian Hydrographic Service

o DBMS : CARIS data manager

o Web-browser : Netscape, Explorer, Mosaic

5) ESRI GIS Application : Internet Mapping Solution

o URL http://maps.esri.com

o 기 능 : 확대/축소, 부분 강조, 길 안내,
검색(주소, 좌표, 우편 번호에 의한 대상 검색)

o 도 구 : ESRI 제품 군(MapObjects, ArcView)

o 응용 분야 :

- 샌프란시스코 관광 안내
- 미국 주별 주제도 작성
- 우편 번호에 의한 EPA의 독성 물질 배출 현황
- 세계지도 검색

비 상업용		TAGIS	IDGIS	IRIS
개발 국가		미국	미국	러시아
URL		http://poca.osmre.gov	http://starr-www.tamu.edu	http://allanon.gmd.de
시스템 개요		Java Map Interface	Java Applet을 이용한 분산 GIS	Java Applet을 이용한 수치 통계분석 및 도표 처리
주요 기능	확대/축소	O	O	O
	부분 강조	O	O	X
	검 색	O	O	O
	공간분석	O(선분 측정)	O(선분 측정)	X
	길 안내	X	X	X
	기 타		Tin/VRML	특정Layer강조 기능, data visualizion
Data Layer		도로, 강, 24K Quad, 고도, 행정경계 (Layer추가/삭제가능)	도로, Watershed, 강 토양, 수계, Trail	
좌표 체계		UTM17	Map X, Y	
Frame		upper/lower frame	Multi-frame	Multi-frame
화면 배경		gif	X	X
DBMS 지원		X	O(Oracle)	X
응용 분야		GIS 일반	GIS 일반	자료 도시화 -인구 동태 분석 -유럽국가 별 인구 및 경제 현황
성능 비교				

상업 용		ActiveMaps	CARIS	ESRI GIS
개발 국가		미국	캐나다	미국
URL		http://internetGIS.com	http://caris0.universal.ca	http://maps.esri.com
시스템 개요		Java를 이용한 online지도 도시 및, 검색, GIS시스템 개발 도구	지도 조작/표현 및 검색 시스템	Internet mapping solution
주요 기능	확대/축소	O	O	O
	부분 강조	O	O	X
	검 색	O	O (form/feature based)	O (주소,좌표,우편번호 등에 의한 검색)
	공간분석	X	X	X
	길 안내	X	X	O
	기 타	Hyper Link	검색수수료 부과기능	ESRI 제품 이용
Data Layer		도로,강,24K Quad, 고도,행정경계 (Layer추가/삭제가능)	CARIS file	방대한 지도 자료 이용 가능
좌표 체계		Map X,Y	중심좌표(N/W)	
Frame		Single	Single	Multi-frame
화면 배경		X	Raster image	X
DBMS 지원		X	O(Oracle)	X
응용 분야		GIS 일반	GIS 일반 -관광 안내 -토지 이용 안내	GIS 일반 -관광 안내 -주별 주제도 작성
성능 비교				

상업 용		MapQuest	GEO/WEB
개발 국가		미국	한국
URL		http://www.mapquest.com	http://starr-www.tamu.edu
시스템 개요		Java를 이용한 관광 안내용 지도 검색 및 제작	Internet환경하에서의 지리정보 검색
주요 기능	확대/축소	0 (국가>지역>시>번지의 단계별 선택기능)	
	부분 강조	x	
	검 색	0 (위치/정보 검색)	
	공간분석	X	
	길 안내	0	
	기 타	사용자용 지도 제작 가능	
Data Layer			
좌표 체계			
Frame		Single	
화면 배경			
DBMS 지원		X	
응용 분야		GIS 일반 -관광 안내 전용	GIS 일반
성능 비교			

여 백

제 2 장 네트워크상의 GIS 자료 입출력 기능

GIS가 가지고 있어야할 기본적인 5가지 기능이 있다. 이것들은 자료 획득 기능, 획득된 자료에 대한 오류 수정 기능이 포함된 전처리 기능, 자료 관리 기능, 자료 처리 및 분석 기능, 자료 출력 기능이다. 본 장에서는 이러한 기본적인 5가지 기능들에 공통적으로 사용되는 벡터 자료의 내부 포맷에 관하여 설명한다.

제 1 절 네트워크 GIS를 위한 GIS 자료 내부 포맷 정의

1. SVF (SERI Vector Format) 정의

정보 처리 시스템마다 자료를 처리하기 위한 서로 다른 내부 자료 형식을 가지고 있다. GIS에서 사용하는 자료 형식은 크게 래스터(Raster) 형식과 벡터(Vector) 형식으로 나누어지고, 래스터 형식의 GIS와 벡터 형식의 GIS는 각각 고유한 영역의 응용 분야에서 발전을 거듭하고 있다. 지리 정보 시스템을 구축할 때 가장 중요한 부분이 자료 구조의 결정이다. 따라서, 구축하고자 하는 공간 자료의 특성과 응용 분야의 특성을 면밀히 분석하여 그에 맞는 적절한 자료 구조를 선정해야 한다.

그러나, 현재 제품으로 시판되고 있는 GIS 패키지들의 자료 구조는 공개된 것이 없으므로 본 연구에서는 지리 정보 처리를 위한 별도의 자료 구조를 정의하여 시스템을 구현하였다. 본 연구는 인터넷 상에서 운영되는 특성이 있다. 따라서, 이러한 특성을 고려하여 인터넷을 통해 전송되는 자료의 부하를 줄이면서 다양한 사용자의 요구를 충족시킬 수 있도록 공간 정보와 비공간 속성 정보를 처리하기 위한 적절한 내부 자료 구조를 객체 지향 방법을 이용하여 자체 자

료 구조인 SVF(SERI Vector Format)를 정의 하였으며, 모든 지리 객체는 SVF를 이용한 피쳐(Feature)로 표시하였다. 피쳐는 공간 정보를 나타내는 프리미티브와 비공간 속성 정보로 구성되었다.

가. 지리 객체의 종류

지도면상에 표현 가능한 지리 객체들을 노드 객체, 체인 객체, 폴리곤 객체의 세 가지로 분류된다.

1) 점 성분의 지리 객체

점 성분으로 표시할 수 있는 지리 객체는 x, y 또는 x, y, z 위치로 나타내질 수 있는 맨홀, 전주, 건물, 유전 등의 시설물들이나 범죄 발생지점, 주소에 의하여 공사지점을 표시할 때 등과 같이 지리적으로 표현할 수 있는 사건이 발생한 지점 등이 있다. 이러한 점 성분의 지리 객체는 컴퓨터 내부에서 처리 될 때에 노드(Node)로 표현된다.

2) 선 성분의 지리 객체

선 성분으로 표시할 수 있는 지리 객체는 도로 중심선이나 상수도 배수관, 하수관, 등고선 등을 표시하는데 사용된다. 이러한 선 성분의 지리 객체는 컴퓨터 내부에서 처리 될 때에 체인(Chain)으로 표현되고, 체인을 구성하는 일련의 점들의 나열로 구성된다. 체인의 시작점과 끝점은 시작 노드와 끝 노드로 분류하여 별도로 관리한다.

3) 면 성분의 지리 객체

면 성분은 다각형 성분이라고도 불리며, 이것은 2차원 면적으로 정의되고 경계선을 갖는 지역을 표시한다. 면 성분의 지리 객체는 주로 지적선이나 인구 조사 표준지역 및 유지 보수 구역, 토지, 토양 분포, 토지 이용도 등을 표시하는데 많이 이용된다. 이러한 면 성분의 지리 객체는 컴퓨터 내부에서 처리 될 때에 다각형(Polygon)으로 표현하였다.

나. 피쳐 정의

1) 피쳐 형식의 정의

지리 객체를 모델링한 피쳐를 표현하기 위한 포맷을 정의하기 위해서는 다음 몇 가지를 생각해 볼 수 있다.

- 어떤 비공간적 속성들을 정의할 것인가?
- 어떻게 공간 정보를 표현할 것인가?

이상의 몇 가지 물음에 대한 고려를 바탕으로 피쳐에 대한 내부 포맷을 정의할 수 있다.

본 연구에서는 지도면상에 표현 가능한 지리 객체들을 노드 객체, 체인 객체, 폴리곤 객체의 세 가지로 분류하였고, 이 세 가지의 객체들을 객체 지향 기법을 이용하여 노드 피쳐, 체인 피쳐, 폴리곤 피쳐로 표현하였으며 각각의 피쳐 포맷을 정의하였다. 각 피쳐의 구조는 피쳐의 속성을 나타내는 비공간 속정 정보와 피쳐의 공간적 위치와 모양을 표현하는 공간 정보를 가지고 있는 프리미티브로 구성되어 있으며, [그림 2-1]에 피쳐의 구조를 나타내었다.

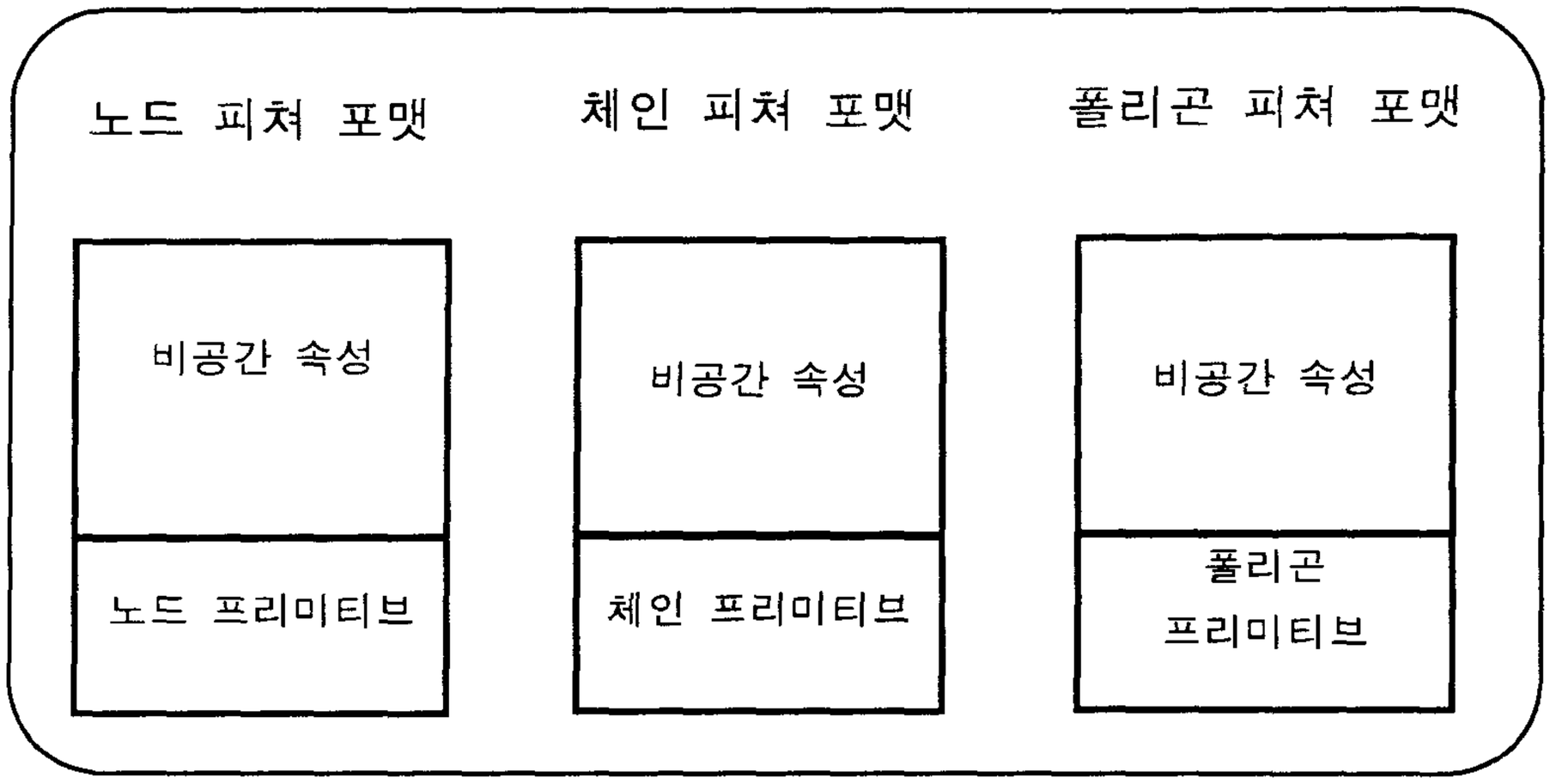


그림 2-1. 피쳐 구조

각 지형 지물을 하나의 피쳐로 정의 하지 않고 이렇게 세 개의 피쳐로 분류하여 포맷을 정의 한데에는 몇 가지 이유가 있다.

첫째, 지구상에 존재한다고 알려진 수백 개의 피쳐(U.S. Geological Survey에 의하면 약 350개의 피쳐가 존재함)를 각각의 피쳐 포맷으로 정의하면 프로그램 상에서 각 피쳐를 대상으로 어떠한 작업을 수행할 때, 일일이 피쳐의 클래스 타입을 instanceOf 등의 연산자로 체크하여야 하는 번거로움이 있다.

둘째, 피쳐의 종류를 세 가지로 구분함으로써 피쳐를 구성하는 프리미티브와의 관계를 명확히 파악할 수가 있고 피쳐간 중복되는 속성만큼의 공간을 절약할 수가 있다.

2) 프리미티브 (Primitive) 정의

지리 객체는 지구 표면 또는 그 부근에 위치하고 있는 사물을 지칭하는 것이다. 지리 객체는 자연물, 인공물, 눈에 보이지 않는 토지의 분할 경계 등 지

구 표면에 위치할 수 있는 모든 것이 그 대상이 될 수 있다. 그러나, 수많은 지리 객체가 존재할 수 있지만 이러한 지리 객체들은 크게 세 가지 형식으로 표현할 수 있다. 즉, 점 성분의 지리 객체, 선 성분의 지리 객체, 면 성분의 지리 객체로 구분할 수 있다. 이러한 지리 객체들이 공간상에 표현되는 형태를 프리미티브라고 하였다.

가) 프리미티브 계층 구조

지리 정보 시스템은 지리 객체를 표현하기 위하여 점 성분 객체, 선 성분 객체, 면 성분 객체를 기본 구조로 가지고 있다. 이러한 3가지 구조는 전통적인 지리 정보 시스템에서 사용하는 것으로 객체 지향 개념을 적용하였을 때에도 이 3가지 객체를 기본 객체로 사용하였다. 그러나, 1992년 Tang이 객체 지향 개념을 도입한 피쳐(Feature)에 기반한 지리 정보 시스템을 구현하였을 때 그는 벡터 자료의 기본형을 6가지로 표현하였다. 이 6가지 기본 구조의 계층도는 [그림 2-2]에 나타내었다. [그림 2-2]에서 사용한 지리 객체의 프리미티브는 점(Point), 노드(Node), 선(Line), 체인(Chain), 링(Ring), 다각형(Polygon) 이다. 이 계층구조의 특징은 각각의 프리미티브가 서로의 Id를 매개체로 이용하여 연결되어 있고, 실제 지리 좌표는 점 성분만이 가지고 있는 것이다. 이러한 구조는 중복된 저장 없이 저장 장치를 이용할 수 있어서 매우 효율적이다. 그러나, 자료를 처리하는 과정에서는 효율성이 떨어진다. 예를 들어 Polygon 객체를 화면에 그리려고 할 때의 순서는 다음과 같은 절차가 필요하다.

1. 그리려고 하는 다각형을 찾기 위하여 다각형 테이블을 검색한다.
2. 검색된 다각형에서 다각형을 구성하고 있는 링의 Id를 찾는다.
3. 선택된 링의 Id를 이용하여 링 테이블을 검색한다.

4. 링 테이블에서 링을 구성하는 체인들을 찾는다.

5. 검색된 체인의 Id를 이용하여 체인 테이블을 검색한다.

6. 체인 테이블에서 체인을 구성하는 선의 Id를 찾는다.

7. 검색된 선의 Id를 이용하여 선 테이블을 검색한다.

8. 선 테이블에서 선을 구성하는 점의 Id를 찾는다.

9. 검색된 점의 Id를 이용하여 점 테이블을 검색한다.

10. 검색된 점의 좌표를 이용하여 화면에 그림을 그린다.

이상과 같이 단순히 다각형을 그리는 과정에도 많은 테이블 검색을 하여야 하므로 이러한 구조는 시스템의 처리 속도를 저하시키는 매우 중요한 요소로 작용하게 된다. 더욱이, 인터넷을 이용하는 본 시스템은 통신상의 부하로 인하여 생기는 처리 속도의 저하를 극복하여야 하는 상황에서 Tang이 제시한 구조는 너무 많은 단점들을 가지고 있는 구조인 것이다. 따라서, 본 시스템의 설계 단계에서는 Tang의 구조를 개선하여 처리 속도를 향상시키는 데에 중점을 두고 프리미티브 구조를 정의하였다. 본 시스템에서 설계된 프리미티브 계층도는 [그림 2-3]에 표시하였다. 이 계층도의 특징은 지리 객체들을 표현하는 기본 객체인 노드 객체, 체인 객체, 그리고 다각형 객체를 기본 객체로 사용하였으며 다각형 객체 내부에 다각형 객체가 포함되어 있는 객체(호수내부에 섬이 있을 때 호수를 표시하는 객체나 타이어와 같은 형식의 지리 객체)를 표현하기 쉽게 하기 위하여 링 객체로 설계되었다. 또한, 각 객체의 검색 속도를 향상시키기 위하여 노드 객체와 체인 객체가 자신을 구성하는 지리 좌표를 포함하고 있다는 것이다. 다각형 객체는 지리 좌표를 포함하고 있지 않은데 그 이유는 다각형 객체는 위상 관계 형성 과정을 통하여 노드 객체와 체인 객체를 기반으로 새롭게 만들어지는 객체이기 때문이다. 따라서, 본 시스템에서 다각형을 그리기 위해서는 다음과 같은 단계를 거친다.

1. 그리려고 하는 다각형을 찾기 위하여 다각형 테이블을 검색한다.
2. 검색된 다각형에서 다각형을 구성하고 있는 링을 찾는다.
3. 선택된 링을 구성하고 있는 체인들을 찾는다.
4. 검색된 체인을 구성하는 점들의 리스트를 이용하여 그림을 화면에 그린다.

이상과 같이 본 시스템에서 설계한 프리미티브 구조를 이용하면 4단계만 거치면 다각형을 화면에 그릴 수 있다. 따라서 Tang의 설계에 비해서 검색 속도가 최소한 2.5배 이상 빠르다. 이것은 검색 과정의 단계를 비교한 것이지만 실제적으로는 2.5배 보다 훨씬 빠를 것이다. Tang의 설계에서는 기본적으로 객체의 Id를 가지고 저장 장소를 찾아가서 원하는 객체를 검색하지만, 본 시스템의 설계에서는 각 객체들이 하위 객체에 대한 포인터를 이용하고 있으므로 단순한 포인터 연산을 통하여 하부 객체를 찾아갈 수 있다.

지리 정보 처리 소프트웨어에서 검색은 모든 분석 기능의 기본적인 요소이므로 공간 자료에 대한 검색이 빠르면 전체 시스템의 속도를 향상시킬 수 있는 중요한 분야이다. 본 시스템에서는 공간 자료에 대한 검색의 속도 향상을 위하여 R*-tree 공간 색인 기법을 이용한 공간 검색 시스템을 개발하여 사용하였다.

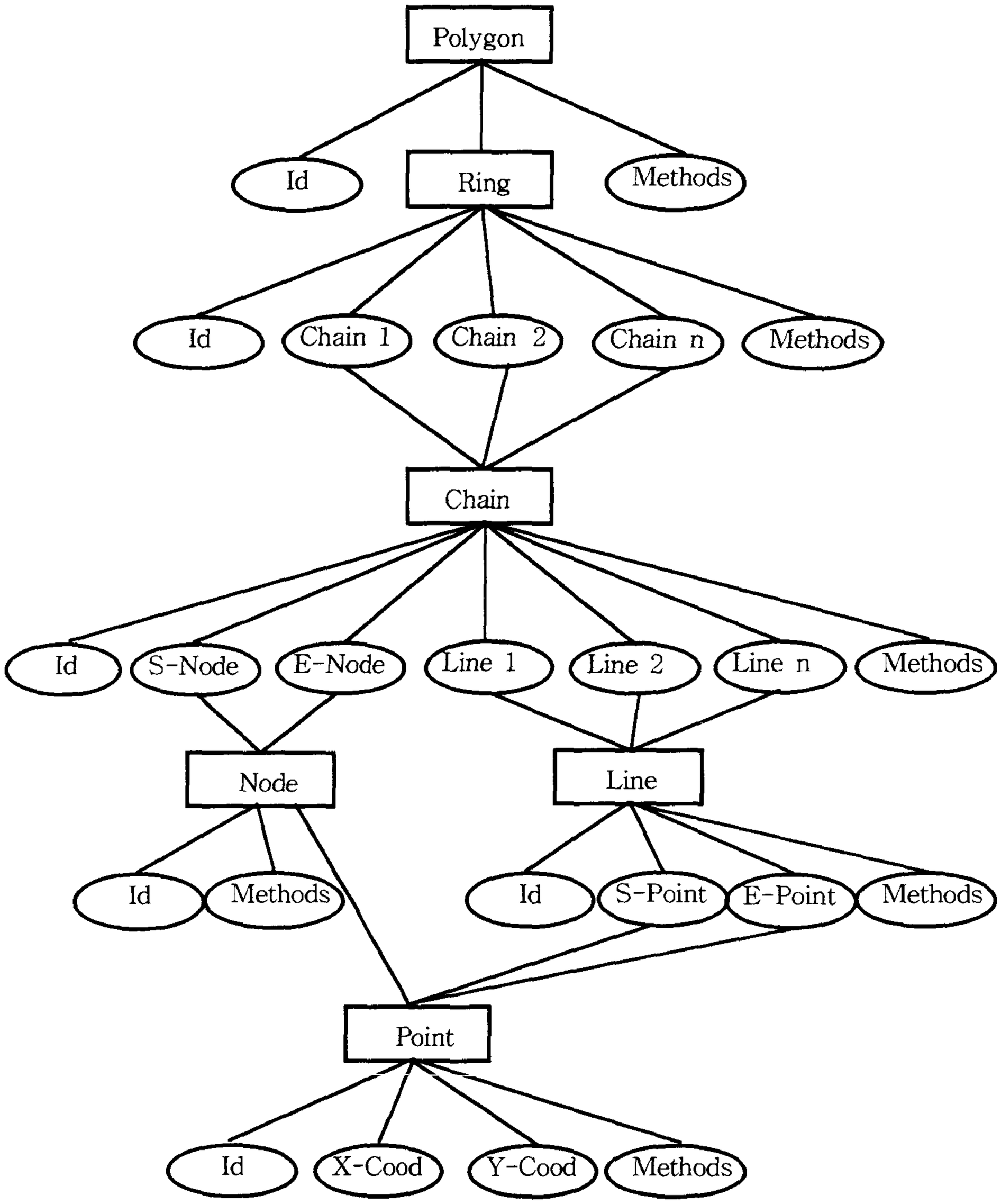


그림 2-2. Tang(1992)의 프리미티브 계층도

그림 2-3. 본 시스템의 프리미티브 계층도

3) 비공간 속성(Aspatial attributes) 정보

피쳐들의 비공간 속성들을 각각 노드 피쳐, 체인 피쳐, 폴리곤 피쳐의 비공간 속성으로 분류하였다.

가) 노드 피쳐 속성

노드 피쳐의 속성은 다음과 같이 정의하였다.

```
int      id              // 피쳐의 식별 번호
String   className       // 클래스 명
String   name            // 피쳐 이름
int      type            // 피쳐의 종류.
                         // NODE, CHAIN, POLYGON 중의 하나
int      exs             // 건물의 완공 여부(Existence Category)
int      birth           // 준공 연도(Birth Year)
int      aoo             // 건물의 방향각(Angle of Orientation(degree))
int      bfc             // 건물의 용도(Building Function Category)
int      zv2             // 건물의 최고 높이(Highest Z-value)
String   phoneNum        // 전화 번호(Representative Phone Number)
String   addr            // 주소(Address)
int facility             // 주유소가 보유하고 있는 부대시설
                         // 예를 들어 세차 시설, 정비 시설 등
int numWard              // 병원이 보유하고 있는 입원실의 수
                         // (Number of Ward
int treatment            // 병원의 종류(Treatment)
```

노드 피쳐의 id, className, name, type은 노드 피쳐의 객체들 모두의 공통적인 속성이며, 이 4가지 속성 이외의 다른 속성들은 VPF(Vector Product

Format)의 피쳐 테이블을 참조하였다.

위의 비공간 속성에서 정수 값을 지니는 변수는 그 정수 값의 의미를 알기 위해서 디코딩 루틴을 거쳐야 한다. 노드 피쳐의 정수형 비공간 속성 값의 의미는 다음과 같다.

type	-1	null 값
	1	NODE 타입
	2	CHAIN 타입
	3	POLYGON 타입
exs	-1	null 값
	0	Unknown
	10	Under Construction
	20	Operational
bfc	-1	null 값
	0	Unknown
	1	Government Building
	2	Hospital
	3	Museum
	4	Police Station
	5	School
	6	House
	7	Post Office
	8	Fire Station
	9	Bank
	10	Hotel

		11	Gas Station
		12	Restaurant
		13	Church
		14	Company
		15	Bus Terminal
		16	Store
		17	Other
	facility	-1	null 값
		0	Unknown
		10	Washing Machine
		20	Light Fixing
		30	Washing and Fixing
		40	Other
	treatment	-1	null 값
		0	Unknown
		10	Internal
		20	Surgery
		30	Pediatry
		40	Orthopedy
		50	Psychiatry
		60	Obstetrics and Gynecology
		70	Dentistry
		80	General

나) 체인 피쳐 속성

체인 피쳐의 속성은 다음과 같이 정의하였다.

```
int     id              // 피쳐의 식별 번호
String  className       // 클래스 명
String  name            // 피쳐 이름
int     type            // 피쳐의 종류.
int     exs             // Existence Category
int     loc             // Location Category
int     ltn             // Lane/Track Number
int     rst             // Road/Runway Surface Type
int     rtt             // Route Intended Use
int     med             // Median Category
int     use             // Usage
double  length          // 하천의 길이. 단위는 meter
double  width           // 하천의 폭. 단위는 meter
```

체인 피쳐의 id, className, name, type은 체인 피쳐의 객체들 모두의 공통적인 속성이며, 이 4가지 속성 이외의 다른 속성들은 VPF(Vector Product Format)의 피쳐 테이블을 참조하였다.

위의 비공간 속성에서 정수 값을 지니는 변수는 그 정수 값의 의미를 알기 위해서 디코딩 루틴을 거쳐야 한다. 노드 피쳐의 정수형 비공간 속성 값의 의미는 다음과 같다.

```
// type, exs 필드는 노드 피쳐의 경우와 동일
loc             -1      null 값
                 0      Unknown
                10      On Ground Surface
```

	20	Suspended or Elevated
rst	-1	null 값
	0	Unknown
	10	Paved
	20	Unpaved
rtt	-1	null 값
	0	Unknown
	10	Primary Route
	20	Secondary Route
	30	Limited Access
med	-1	null 값
	0	Unknown
	10	With Median
	20	Without Median
use	-1	null 값
	0	Unknown
	10	National
	20	City
	30	Private
	40	International
	50	Other

다) 폴리곤 피쳐 속성

폴리곤 피쳐의 속성은 다음과 같이 정의하였다.

```
int     id              // 피쳐의 식별 번호
String  className       // 클래스 명
String  name            // 피쳐 이름
int     type            // 피쳐의 종류.
int     exs             // Existence Category
int     numBld          // Number of Buildinds(Dong)
int     birth           // Birth Year
int     numDoumi        // Number of Doumi
int     degree          // Elementary, Middle, High, Univ.
int     numStu          // Number of Students
int     bfc             // Building Function Category
int     zv2             // Highest Z-value
int     story           // Story of Building
int     Pyoung          // Area measured as "Pyoung"
int     numMan          // Number of man
int     product         // Main Product
String  phoneNum        // Phone number
String  faxNum          // Fax number
String  addr            // Address
double  area            // 구 면적
int     numHouse        // number of houses
int     pop             // population
int     projection      // Projection Type
double  fromLati        // From Latitude
double  toLati          // To Latitude
double  fromLongi       // From Longitude
double  toLongi         // To Longitude
```

폴리곤 피쳐 역시 id, className, name, type은 폴리곤 피쳐의 객체들 모두의 공통적인 속성이며, 이 4가지 속성 이외의 다른 속성들은 VPF(Vector

Product Format)의 피쳐 테이블을 참조하였다.

위의 비공간 속성에서 정수 값을 지니는 변수는 그 정수 값의 의미를 알기 위해서 디코딩 루틴을 거쳐야 한다. 폴리곤 피쳐의 정수형 비공간 속성 값의 의미는 다음과 같다.

```
// type, exs, bfc 필드는 노드 피쳐의 경우와 동일
degree          -1      null 값
                10      Kindergarten
                20      Elementary
                30      Middle
                40      High
                50      Colleage
                60      Universe
product         -1      null 값
                 0      Unknown
                10      Chemical
                20      Metal
                30      Software
                40      Not Applicable
                50      Other
projection      -1      null 값
                 0      Unknown
                10      UTM
                20      BESSEL
                30      GEO
```

40 Other

4) 피쳐 종류

본 연구에서는 다음과 같은 피쳐를 정의하여 사용하였다.

노드 피쳐	EtcBuildingN	기타 빌딩
	GasStationN	주유소
	HospitalN	개인 및 종합병원
	PublicBuildingN	공공기관(경찰서, 관공서 등)
체인 피쳐	Road#1C	8~6차선 도로
	Road#1CentC	8~6차선 도로의 중앙선
	Road#2C	4~2차선 도로
	Road#2CentC	4~2차선 도로의 중앙선
	Road#3C	이면 도로
	RoadExpressC	고속도로
	RoadExpCentC	고속도로의 중앙선
	StreamC	하천
폴리곤 피쳐	APTComplexP	아파트 단지 경계
	DongBoundaryP	동 경계
	ExpoP	엑스포 구역 경계
	GuBoundaryP	구 경계
	InnerBuildingP	아파트단지 등의 내부 건물들

InstituteP	연구단지 내 연구소들
MapSheetP	도엽 경계(6개의 구역)
SchoolP	초, 중, 고, 대학교

2. VPF(Vector Product Format)의 분석

가. 개 요

지형 정보(geographic information)의 수집은 이를 담당하고있는 여러 국가기관에서는 모두 그들 나름대로의 목적을 위해서 이루어져왔다. 이러한 정보들은 계획(planning), 자원관리, 과학조사, 그리고 여러 종류의 데이터에 관한 공간 베이스로 이용된다. 이러한 정보들은 그 발생(producing) 기관에서 사용하기 유용한 형태로 제작된 후 일반에 공개되는 것이 일반적이다. 하지만 미국의 Defense Mapping Agency(이하 DMA) 에서는 군대 및 해양관련 고객들이 바로 사용할 수 있는 형태로 mapping, charting, geodetic, 그리고 지리정보 등을 수집, 배포하고있다. 이러한 데이터 집합 또는 생산품(products)들은 사용하기 쉽게 디자인 되어야하고 고객의 응용분야에 대한 충분한 지원을 보장해야한다. 이러한 이유들로 인해 DMA는 Vector Product Format(이하 VPF)을 군사 표준으로 채택하였다. DMA는 많은 양의 데이터를 VPF로 제작하고 있으며 이는 갈수록 널리 사용될 것이다.

DMA는 수치화된(digital) 지리공간 생산품(geospatial product)들을 Vector Product Format(VPF), Raster Product Format(RPF), 그리고 Text Product Format(TPF)의 세 가지 형태로 공급하고 있다. 이 중에서도 VPF는 디지털 지도 데이터에 대한 수요의 급속한 증가로 인해 중요한 역할을 지니게되었다. 따라서 DMA는 광대한 범위의 생산품(product)들을 VPF 표준에 맞게 제

작하고 있으며 지형공간 정보의 배포 및 응용분야에의 직접사용 등에 있어서 계속 그 이용 범위를 확대해 나가고 있다.

나. 배 경

많은 정부 기관들이 단지 기관 내에서의 사용을 위해 지리정보를 수집하고 있는 것과는 달리, DMA는 고개의 요구를 충족시키기 위해 지리정보, 지도 데이터, 측지정보, 주행정보 등을 수집 및 공급하고 있다. 미연방, 주(state), 그리고 각 지역 기관에서는 공공사업, 도시계획, 인구성향조사 뿐만 아니라 삼림, 토지, 수자원, 동력자원관리 등을 위해 지형공간 정보를 수집하고 이용한다. 일반적으로 이러한 정보들은 성장, 발전 및 재창조를 도모하기 위해 공중 또는 일반에게 저가의 부산물(by-product) 형태로 공급된다. 미국 국방성은 mapping, charting, 그리고 측지학과 관련된 지형공간 정보들을 전세계의 해양항해 및 미육, 해, 공군과 정보기관 등에서 사용할 수 있도록 하기 위해서 DMA를 설립했다. 이러한 생산품(product)들은 실로 그 응용 범위가 방대한데, 예를 들어 전략 및 전술의 수립, 임무계획 및 예행연습, 무기시스템 개발에 대한 모델링 및 성능시험, 비행기 조종석에서의 이동 맵 디스플레이, 주행시스템 등이 있다

Spatial Data Transfer Standard(SDTS) 나, DMA의 내부 포맷인 Mapping Charting Feature Data Exchange Standard(MC&GFDES) 등의 지리정보 표준은 단지 정보의 교환을 위해 고안된 견실한 표준의 예이다. MC&GFDES 는 초대형 집적 생산 시스템 내에서의 데이터 교환을 위해 개발되었다. SDTS는 많은 경우, 상품화된 지리정보시스템을 이용하는 각 조직간의 임의형태의 공간정보의 교환을 위해서 개발되었다. 지리데이타의 형(type)과 그 내용은 너무나 다양하므로 모든 SDTS의 다양성을 처리할 수 있는 범용의 번역기

(translator)를 만드는 것은 불가능하다. 데이터 교환 효율의 극대화를 위해 SDTS는 데이터의 양을 줄이지 않았으며 데이터로의 빠른 접근을 위한 인덱스도 제공하지 않는다. 또한 데이터로의 접근 및 사용의 편리를 위한 부가정보나 선행처리된(pre-computed) 데이터도 사용하지 않는다.

즉, SDTS는 생산품의 분배를 위한 목적으로는 고안되지 않았다는 것이다. FIPS 173의 8번째 문단에는, FIPS SDTS는 특정한 데이터 구조나 컴퓨터 시스템, 배포 매체에 의해 특징 지워진 응용 소프트웨어를 이용하여 바로 쓸 수 있도록 고안된 형태의 공간 데이터를 수용하기 위해 개발된 것은 아니다. 라고 적혀있다. 그러나 VPF와 같은 생산품 규격(product standard)은 응용 소프트웨어가 다양한 내용의 정보에 쉽게 접근할 수 있도록 지원해야한다. 또한 생산품 규격은 데이터를 분배하기 위해 필요한 테이프나 디스크의 수를 줄이기 위해 정보를 효율적으로 저장, 관리해야만 한다. 그리고 인덱스를 제공함으로써 사용자가 효율적으로 정보를 검색할 수 있게 해야한다.

다. VPF 표준

VPF표준은 DMA의 모든 벡터 생산품들의 근간을 이루는 개념적 그리고 실제적인(physical) 데이터 모델이다. 이것은 5개의 계층적 단계로 구성된 지리적 관계 모델(geo-relational model)을 이용하는데 여기에는 데이터베이스 단계, 라이브러리 단계, 커버리지 단계, 지형요소(feature) 단계, 그리고 그래픽 기본요소(primitive) 단계가 있다. 지리 공간적인 지형요소는 VPF 데이터 모델의 중심적인 개념이다. 실세계의 지형요소는 그것의 위치와 속성을 기술함으로써 모델링 된다.

그래픽 기본요소 단계는 점(node), 선(edge), 면(face), 그리고 문자(text)의 네 기지 기본요소를 이용하여 위치정보를 처리한다. 이는 다섯 가지의 기본

요소 테이블에 의해 구현되는데, 이 테이블에는 점 테이블, 선 테이블, 면 테이블, 문자 테이블, 그리고 면을 구성하는 선 요소에 대한 정보를 저장하는 링 테이블(ring table)이 있다. 이러한 테이블들은 위상학적인 관계정보와 좌표 데이터를 함께 포함하고있다. 지형요소 테이블은 지리 공간적인 지형요소들의 특징을 기술하는데 이용된다. 지형요소 테이블의 각 레코드는 지형요소와 그것의 속성 및 그래픽 기본요소 등으로 구성된다. 지형요소와 구성 그래픽 기본 요소사이의 관계는 일대일(one-to-one), 다대일(many-to-one), 일대다(one-to-many), 그리고 다대다(many-to-many)의 네 가지 관계가 가능하다. 지형요소와 그래픽 요소들이 모여서 커버리지를 구성하고, 커버리지들이 모여서 라이브러리를 구성하고 또 이 라이브러리들이 모여서 데이터베이스를 구성한다.

VPF 의 지형요소간에는 위상관계가 존재하는데 이는 그 정도에 따라0(위상관계가 없음)에서 3단계로 구분된다. 데이터의 종류간에 여하한 연관관계가 없을 때에는 이들은 새로운 커버리지를 구성하게된다. 지형요소간의 완전한 위상관계(full topology)가 필요할 때는 이들을 하나의 커버리지로 결합시켜야한다. 여러 개의 지형요소를 결합하여 단일의 지형요소처럼 사용할 수도 있는데 이를 복합지형요소(complex feature)라 한다.

컴퓨터 메모리와 주변저장장치의 용량이 갖는 한계로 인해 대용량의 지형공간 데이타베이스를 다루기 쉬운 크기의 단위, 즉 타일(tile)로 나눌 필요가 생긴다. VPF는 그래픽기본요소들을 지리학적 단위로 조직하여 관리하는 이른바 타일링(tiling)을 제공하며 타일의 경계에 위치하는 지형요소를 마치 하나의 연결된 요소처럼 관리하는 타일간 위상관계(inter-tile topology)도 제공한다.

지리정보의 보다 효율적인 이용을 가능하게 하는 VPF의 특징들에는 다음과 같은 것들이 있다.

1) 알기 쉬운 형식(Self Describing Format)

VPF의 각 단계(데이터베이스, 라이브러리, 지형요소, 그래픽 기본요소)에는 그 단계에 포함된 각종 정보를 기술하는 헤더 테이블이 있다. 그리고 각 VPF 테이블에도 그 테이블을 기술하는 헤더가 붙어있다. 따라서 개발자들은 생산품 자체의 세부 디자인에 무관하게 VPF의 데이터베이스에 적합한 응용 소프트웨어를 개발할 수 있게된다.

2) 온라인 데이터 사전(On-line Data Dictionary)

VPF의 데이터 사전은 지형요소와 속성(attribute) 정의가 생산품에 명시됨으로써 사용자가 쉽게 이해할 수 있게 한다. VPF는 값을 해설하는 테이블(Value Description Table)을 이용하여 각 커버리지내의 지형요소의 속성 값에 대한 정보를 나타낸다. 따라서 사용자들은 그들 자신이 정의한 데이터를 데이터베이스에 추가시킬 때 이를 이용함으로써 그들의 목적에 가장 적합한 지형요소와 그 속성 값들을 기술할 수 있게된다.

3) 데이터의 품질(Data Quality)

VPF는 라이브러리, 커버리지, 그리고 지형요소의 각 단계에 대한 데이터 품질 정보를 제공한다. 이 정보는 사용자가 지리학적 분석 작업을 수행할 때 도움을 준다. 즉, 분석 작업시 사용자가 생산품의 정확성, 유통성, 완성도 등을 평가할 수 있게된다.

VPF 표준은 공간지리학적인 지형요소의 구조, 위상관계, 그리고 속성정보

를 어떻게(how) 모델링 하느냐에 대한 규칙과 일반적인 데이터베이스 구조를 정의한다. 그것은 무슨(what) 지형요소, 위상관계, 그리고 속성이 사용되어야 하는지를 정의하는 것은 아니다. 생산품 명세서(Product Specification)는 VPF표준에 정의된 규칙에 따라서 무엇이 데이터 집합에 포함되어야 하는가를 결정할 때 도움이 된다. 즉, 어떤 종류의 기하학이(평면상의 직교좌표계 또는 타원체의 경위도좌표계, 2차원 또는 3차원 좌표계), 어느 정도의 위상관계가, 그리고 어떠한 지형요소와 속성 값들이 생산품 규격에 포함되어야 하는가를 결정하는데 도움이 된다.

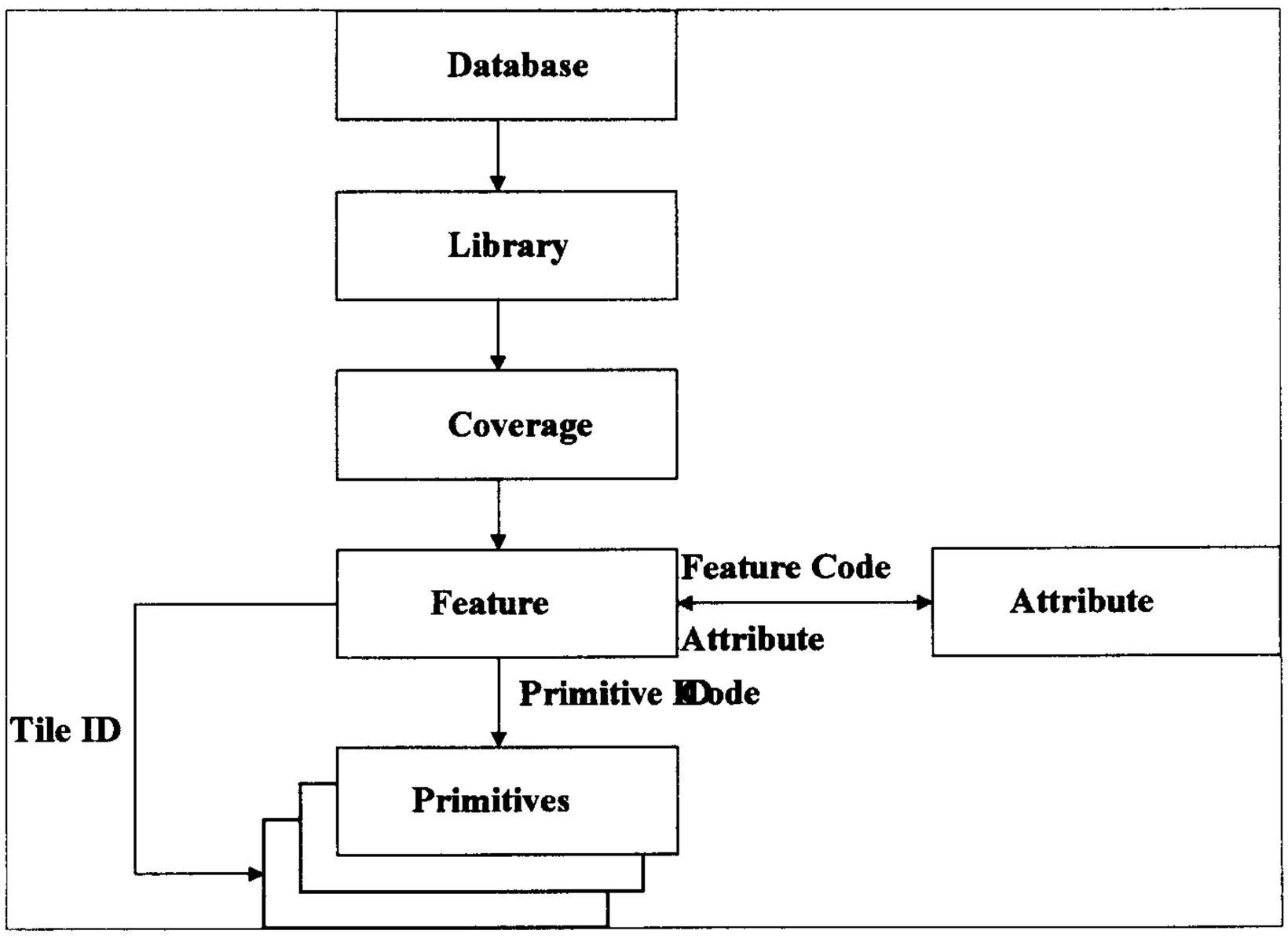


그림 2-4. VPF의 데이터 모델

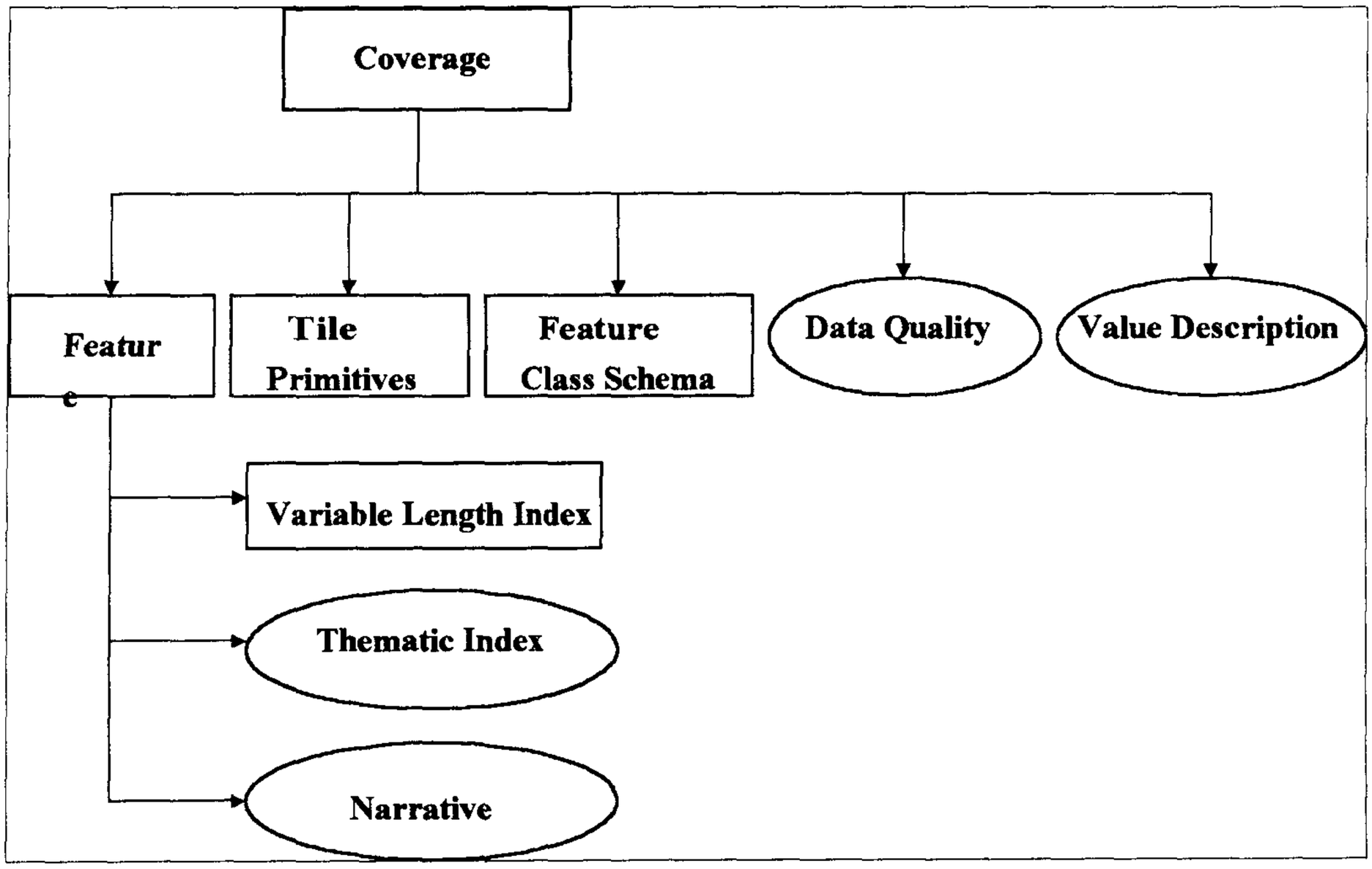


그림 2-5. 주제별 계층(thematic coverage) 구조

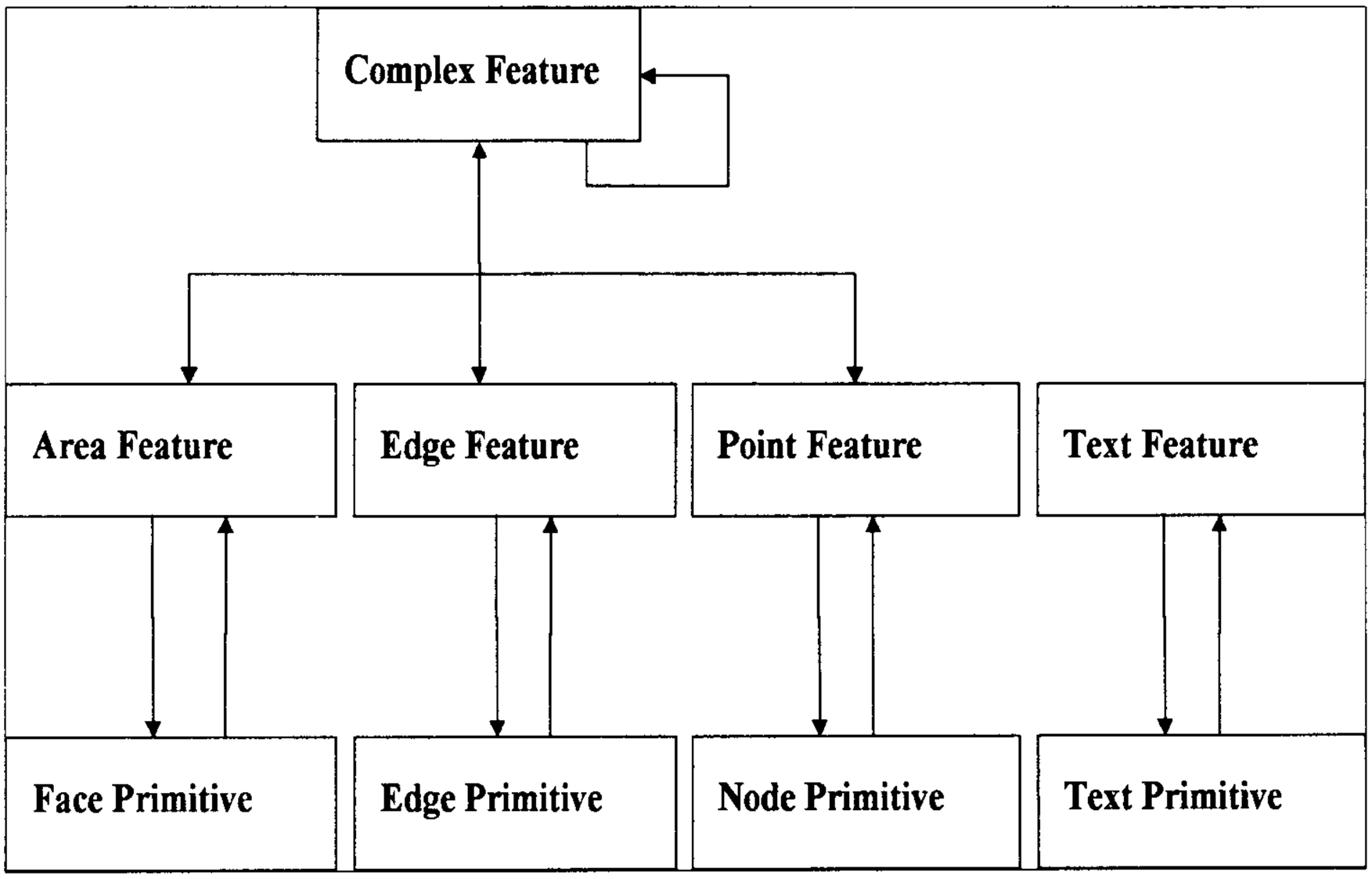


그림 2-6. 지형요소(feature) 구성

제 3 절 GIS의 자료 변환 기능

1. 자료 변환의 개요

현재, 각 지방 자치 단체를 중심으로 지리 정보 시스템 구축이 활발히 진행되고 있으며, 각 지방 자치 단체들은 지하의 전신망, 수도망, 하수도망, 상수도망 등의 시설물과 지상의 도로, 건물, 부존 자원 등의 관리 및 보수 등 지리 정보 시스템의 활용 범위는 날로 증가하고 있다. 따라서, 이들 단체들이 구축한 지리 정보 시스템은 고유한 목적에 맞는 자료 구조를 가지고 각각의 필요에 따라 지도를 수치화한 지리 자료를 이미 구축하여 사용하고 있고, 국립 지리원에서는 전 국토를 대상으로 수치 지도를 만들고 있다. 그러므로, 이미 구축된 지리 정보 시스템들과 지리 자료의 호환성을 유지하는 것은 지리 자료 구축에 드는 비용을 줄임으로써 전체 지리 정보 시스템을 구축하는 비용과 시간을 줄이기 위한 매우 중요한 요소이다.

따라서, 본 연구에서는 다른 지리 정보 시스템에서 구축된 자료를 이용할 수 있는 자료 변환 기능을 제공한다. 이 기능은 다른 시스템에서도 많이 사용되고 있는 AutoCAD의 DXF(Drawing Interchange File Format) 형식을 이용하였다. 이 기능은 DXF 형식의 자료를 읽어들이는 기능과 DXF 형식의 파일로 저장하는 기능으로 구분된다.

2. DXF 형식

DXF 형식은 이진 자료 형식과 아스키 자료 형식으로 저장되며

AutoCAD drawing을 다른 CAD system에 의해서 해독되고 사용될 수 있는 형식으로 변환할 수 있도록 지원해주는 형식이다. DXF는 사실상 거의 모든 컴퓨터의 CAD, GIS 시스템의 표준 저장파일이다. 현재 DXF의 표준은 보편화되어 있으나, 하나의 CAD 시스템에서 다른 CAD 시스템으로 변환하는 것은 100% 호환성을 가지고 있지 못하다. 또한, DXF는 공간 정보의 자료 교환 및 저장에만 사용될 뿐 비공간 속성 정보는 처리하지 못하는 한계를 가지고 있다.

DXF 파일의 구조는 크게 4부분으로 나누어진다. 첫 번째 ENDSEC까지의 처음 부분은 Header Section이며 DXF 파일이 작성될 때 존재한 AutoCAD drawing의 환경이 기술된다. 다음 부분은 Tables Section이라고 불리며, drawing에서 정의한 Linetype(LTYPE), LAYER, Textstyle(STYLE), VIEW 등에 관한 정보가 포함되어 있다. 세 번째 부분은 Blocks Section이며 drawing의 각 block에 대한 entity의 기술이 포함되어 있다. 마지막으로 네 번째 부분은 Entities Section으로 지리 객체의 공간 좌표와 같은 실제 drawing 정보가 포함되어 있다. 이러한 기본 구조의 DXF 파일의 각 섹션은 한 라인에 한 항목씩 나열되는 것이 특징이며, 첫 라인에 그룹 코드(group code)가 들어가고, 두 번째 라인에는 그 그룹에 해당하는 실제 값이 들어간다.

DXF 파일의 기본 구조는 아래에 나타내었으며, [표 2-1]에 그룹 코드와 그 의미에 대하여 나타내었다.

표 2-1. DXF 그룹 코드

그룹 코드	의미
0 - 9	문자열
10 - 59	실수
60 - 79	정수
210 - 239	실수
999	설명문(문자열)

0
SECTION
2
HEADER
<< 헤더 정보 >>
0
ENDSEC
0
SECTION
2
TABLES
<< 테이블 정보 >>
0
ENDSEC
0
SECTION
2
BLOCKS
<< 블럭 정보 >>
0
ENDSEC
0
SECTION
2
ENTITIES
<< 지리 객체 정보 >>
0
ENDSEC
0

EOF

3. SVF와 DXF 사이의 자료 변환

SVF는 자료 구조에 공간 정보과 비공간 속성 정보가 포함되어 있는 반면, DXF는 자료 구조에 공간 정보만이 포함되어 있다. 따라서, SVF와 DXF 사이의 자료 변환은 공간 정보만을 대상으로 하였다.

DXF의 공간 정보는 Point, Line, Polyline으로 구성되어 있고, SVF는 Node, Chain, Polygon으로 구성되어 있다. 따라서, DXF의 Polyline은 처음 시작점과 끝점이 일치하는 가를 검사하여 일치하면 SVF의 Polygon으로 변환하였고, 그렇지 않은 경우에는 Chain으로 변환하는 방식을 취하였다.

DXF와 SVF 사이의 상호 자료 변환 관계는 [표 2-2]에 나타내었다.

표 2-2. DXF와 SVF사이의 관계

DXF	SVF
Point ⟷	Node
Line →	Chain
Polyline ⟷	Polygon
Text ⟷	Text

여 백

제 3 장 네트워크를 통한 GIS 자료의 편집 기술 개발

제 1 절 입력 자료 오류 검색 및 수정 기능

지리 정보 시스템에서 자료의 입력은 종이 지도, 인공 위성 사진, 항공 사진 등의 형태로 존재하는 실세계의 지리 자료를 컴퓨터에서 처리할 수 있는 자료 형태로 변환하는 과정으로, 지리 정보 시스템을 구축하는데 있어서 가장 많은 시간과 비용이 소요되는 부분이다. 또한, 구축된 지리 자료는 지리 정보 시스템이 제공하는 기능의 질을 결정하는 매우 중요한 역할도 담당한다. 따라서, 지리 정보 시스템의 올바른 구축을 위해서는 데이터의 올바른 구축이 필수적이다. 한편, 지리 정보 시스템의 자료 구축이 전체 지리 정보 시스템 구축 과정의 70% 이상이라는 면에서 본다면 오류 수정 과정은 지리 정보 시스템 프로젝트에서 중요한 단계의 하나로 간주된다.

본 연구에서 제공하는 벡터형 지리 자료의 위상학적 오류에는 열려진 다각형, 서로 교차하는 체인, 다른 체인에 연결되어 있지 않은 체인(undershoot), 다른 체인을 지나쳐서 그려진 체인(overshoot), 너무 조밀하게 디지타이징된 점들의 삭제 등이 있다.

또한 이러한 오류의 종류에 따라 오류를 수정하는 과정에서 필요로 하는 허용오차(Tolerance) 값을 입력하여야 한다. 적절하지 않은 허용 오차의 값을 설정할 때에는 원하지 않는 결과를 얻을 수도 있기 때문에 허용 오차를 설정할 때에는 많은 주의를 필요로 한다. 허용 오차 값에는 댕글 길이 (Dangle Length), 노드 매치 오차(Node Match Tolerance), 위드 오차(Weed Tolerance) 등이 있다. 오차의 종류를 기준으로 하여 오류 검색 및 수정 기능을 설명하였다.

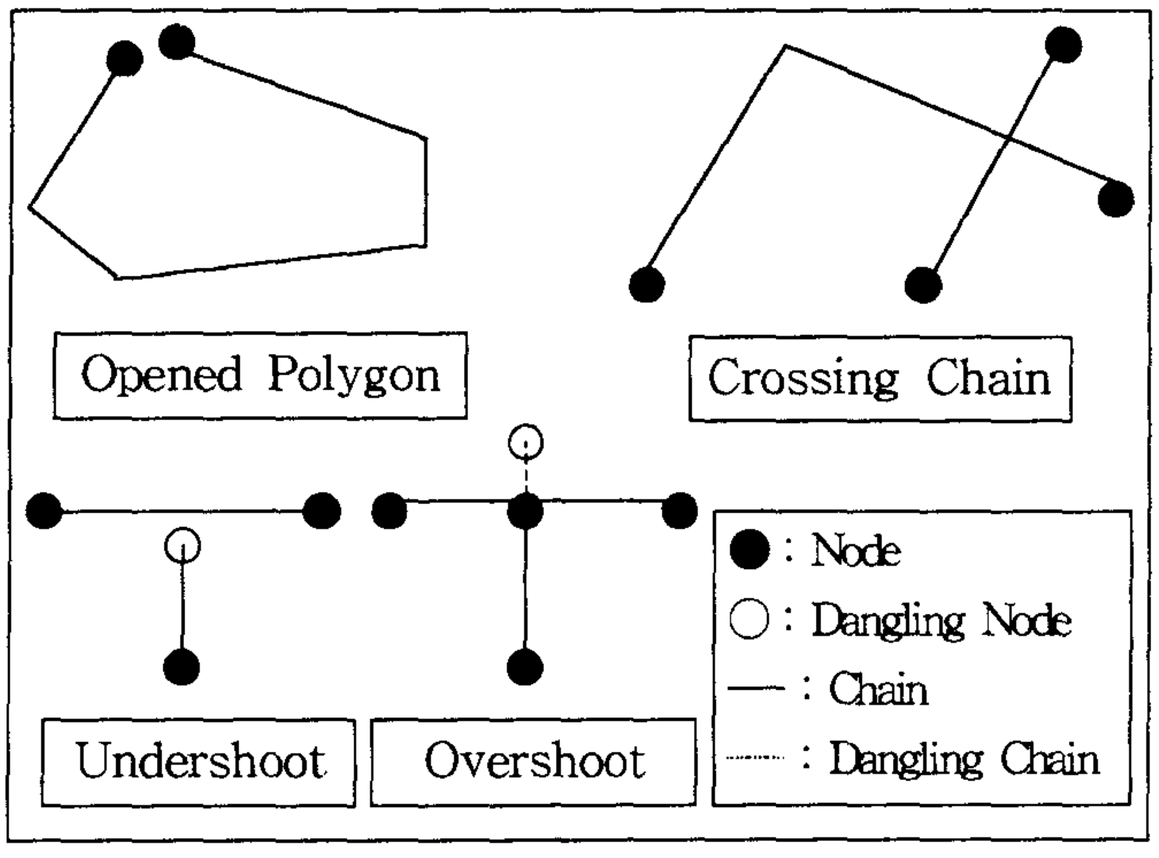


그림 3-1. 위상학적 오류의 예

1. 댕글 길이

벡터형 지리 정보 시스템에서 하나의 체인은 다른 체인과 연결되어 있어야 한다. 댕글 길이라는 것은 다른 체인에 연결되어 있지 않은 체인을 판단하는 기준이 되는 것이며 다른 체인에 연결되어 있지 않은 체인은 댕글링 체인(Dangling Chain)이라고 한다. 또한, 댕글링 체인의 노드 중에서 다른 체인에 연결되지 않은 노드를 댕글링 노드라고 한다. 댕글링 체인에는 2가지가 있다. 하나는 다른 체인을 지나쳐서 그려진 체인으로 overshoot라고 하고, 다른 하나는 undershoot라고 한다. 따라서, 댕글 길이는 overshoot과 undershoot을 결정하는 기준이 된다.

이 댕글 길이를 너무 크게 잡으면 필요한 체인 삭제되거나 댕글링 체인으로 남아 있어야할 체인이 다른 체인에 연결되는 경우가 생기게 된다. 그러나, 반대로 너무 작게 잡으면 불필요한 체인을 제거할 수 없게 된다. 따라서, 입력

된 지리 자료의 특성에 맞게 적절한 값을 설정하여야 한다.

가. Overshoot 삭제

Overshoot된 체인은 체인의 길이를 계산하여 입력된 댕글 길이 값 이하이면 삭제하는 과정을 통하여 오류 수정을 할 수 있다. Overshoot 수정 예를 [그림 3-2]에 표시하였고, 수정 흐름도를 [그림 3-3]에 나타내었다.

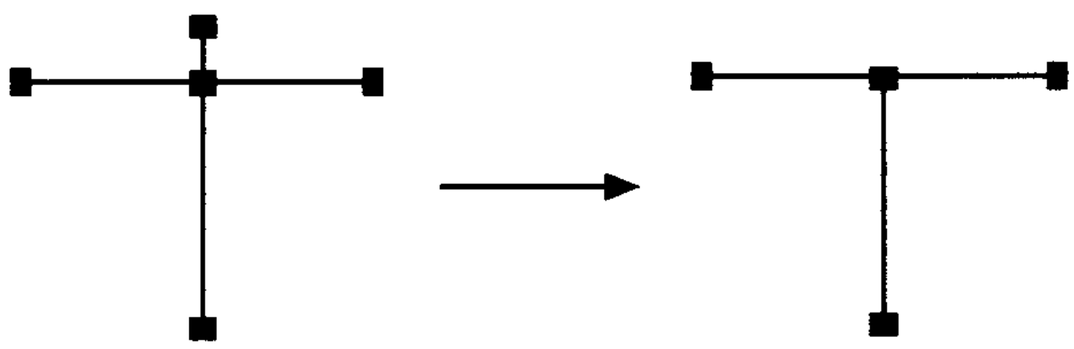


그림 3-2. Overshoot 수정

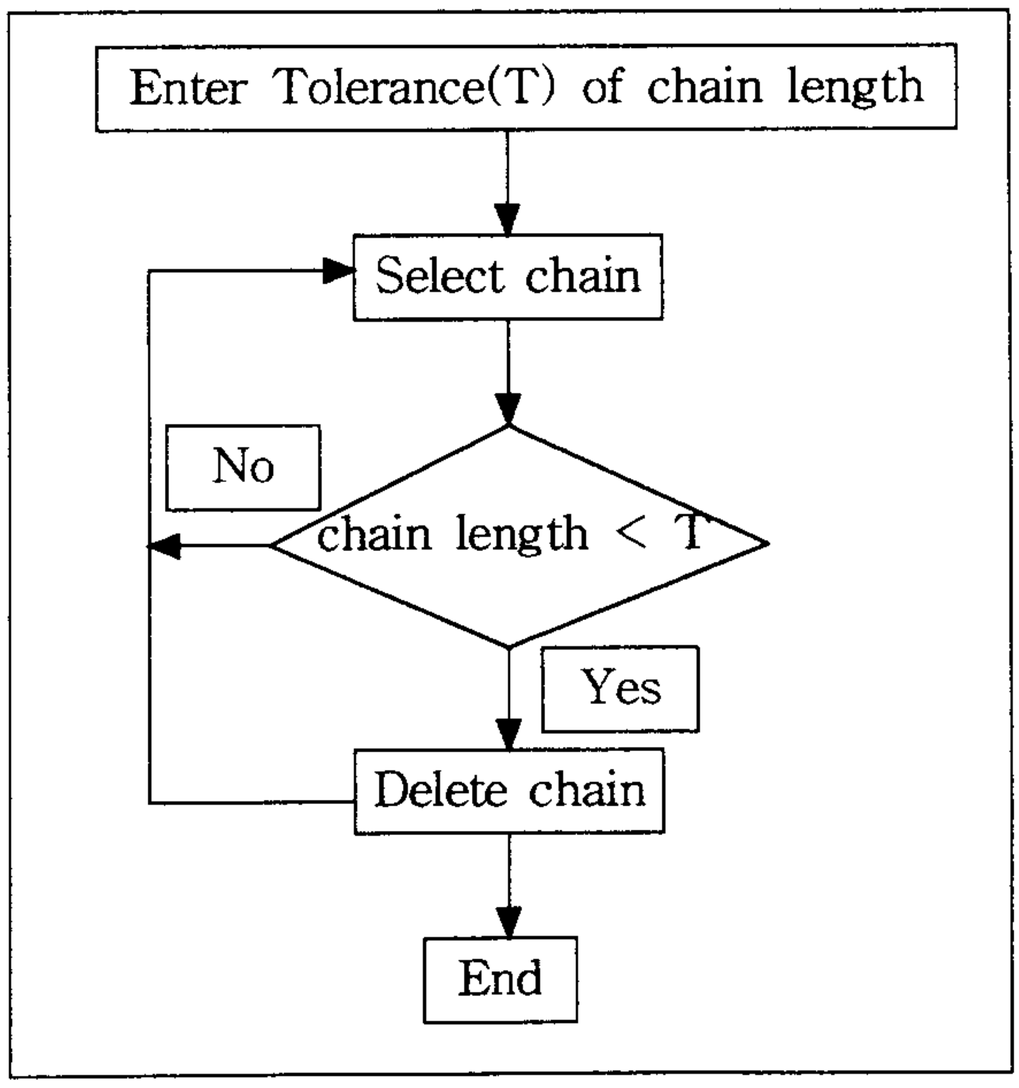


그림 3-3. Overshoot 수정 흐름도

나. Undershoot 삭제

Undershoot된 체인의 삭제는 먼저 각 체인이 댕글링 체인인가를 먼저 검사한다. 그리고 댕글링 체인으로 판정된 경우 이 체인의 댕글링 노드와 다른 체인 사이의 거리를 계산하여 이 거리가 주어진 댕글 길이보다 작으면 두 체인의 교차점을 찾아서 댕글링 체인을 연장하여 다른 체인에 붙인다. 이때 붙임을 당한 체인은 계산된 교차점을 중심으로 두 개로 분할된다. [그림 3-4]와 [그림 3-5]은 각각 Undershoot 수정 흐름도와 Unsershoot를 수정한 예를 나타낸다.

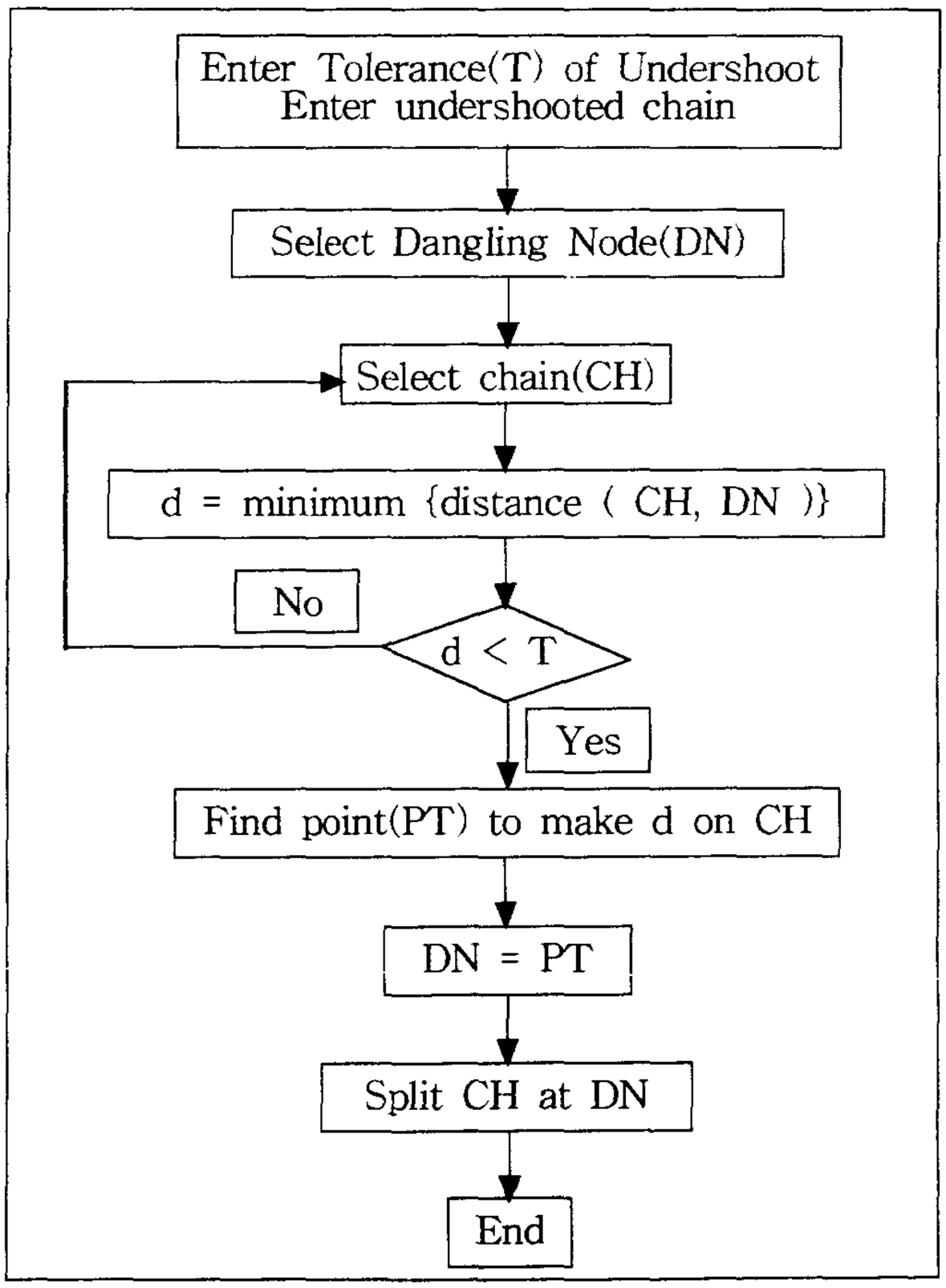


그림 3-4. Undershoot 수정 흐름도

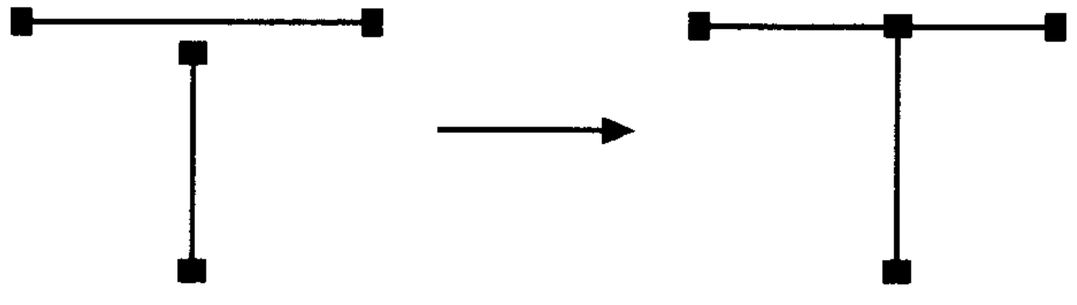


그림 3-5. Undershoot 수정 예

2. 위드 오차

위드 오차란 체인을 구성하는 점들을 대상으로 한다. 점과 점 사이가 너무 조밀하면 많은 저장 공간을 필요로 하므로, 이러한 점들을 대표하는 몇 개의 점만 남겨 두고 나머지는 모두 제거한다. 이 값을 너무 크게 주면 저장 공간은 절약할 수 있으나 정보의 손실로 인하여 체인의 모양이 변한다. 따라서 체인 모양의 변경이 최소가 되도록 하는 범위 안에서 필요한 최소한의 점들만 남겨 두고 불필요한 점들은 모두 제거할 수 있도록 위드 오차를 알맞게 설정하여야 한다.

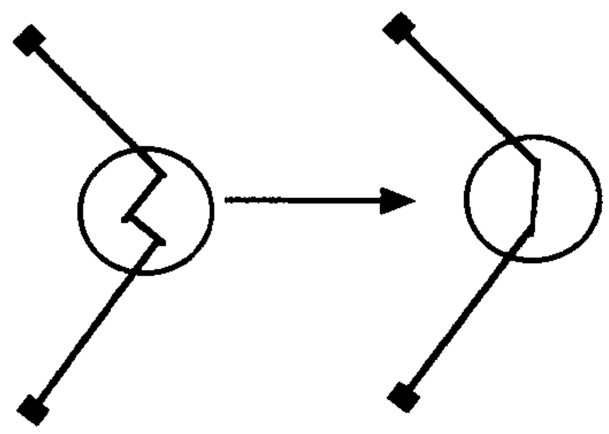


그림 3-6. 위드 오차를 이용한 오류 수정

3. 노드 매치 오차

노드 매치 오차는 각 체인을 구성하는 노드들 사이의 거리를 말한다. 노드와 노드 사이의 거리가 노드 매치 오차 이내이면 두 노드를 결합하여 하나의 노드로 만든다. 이 오차는 주로 다각형을 그릴 때 많이 이용된다. 다각형을 디지타이징하는 경우에 시작점과 끝점은 항상 일치하여야 한다. 그러나 디지타이징 환경이 정밀하지 못하기 때문에 시작점과 끝점을 동일한 곳에 찍는다는 것은 거의 불가능한 일이다. 따라서 동일한 점으로 찍어야 하는데 실수로 서로 다른 점으로 찍혀진 두 점을 하나로 만드는 것이다. 즉, 다각형의 시작점과 끝점 사이의 거리를 계산하여 이 거리가 설정된 노드 매치 오차 이내이면 다각형의 시작점과 끝점을 하나로 결합한다. 다각형의 시작점과 끝점 이외에도 도로의 교차로와 같은 지리 자료는 하나의 노드가 두 개 이상의 체인과 연결되어 있는 경우가 있다. 이 경우에도 노드와 노드 사이의 거리를 계산하여 이 거리가 설정된 오차의 범위에 포함되면 두 개 이상의 노드를 결합하여 하나의 노드로 만든다.

노드 매치 오차 값이 너무 작으면 노드 스냅의 효과가 적어지는 반면, 이 오차의 값이 너무 크면 이상한 모양의 지리 자료가 만들어 질 수도 있다. 따라서, 지리 자료의 성질에 따라 적절한 노드 매치 오차를 설정하여야 한다.

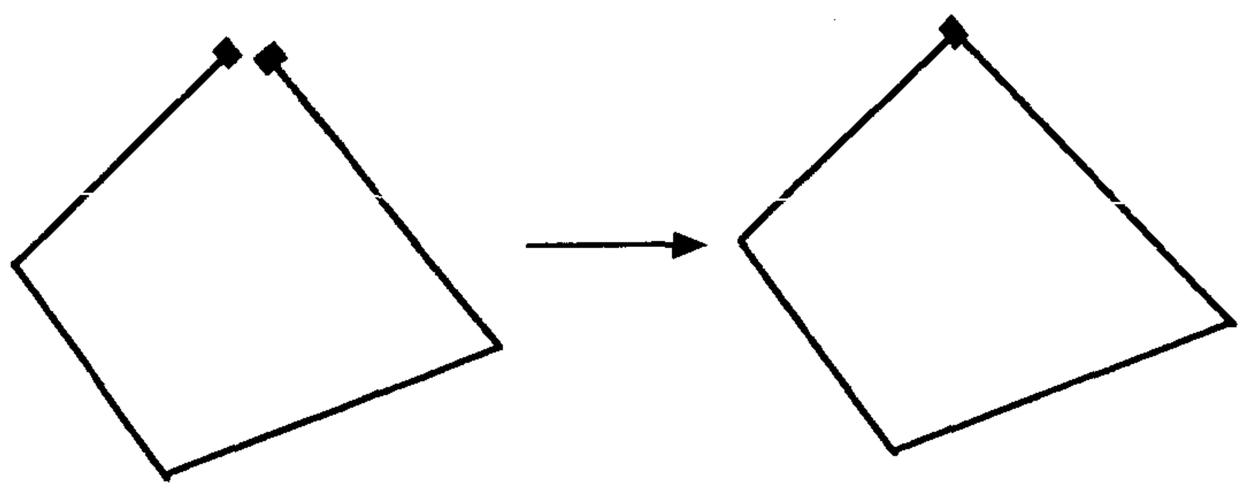


그림 3-7. 노드 매치 오차를 이용한 열려진 다각형의 수정 예

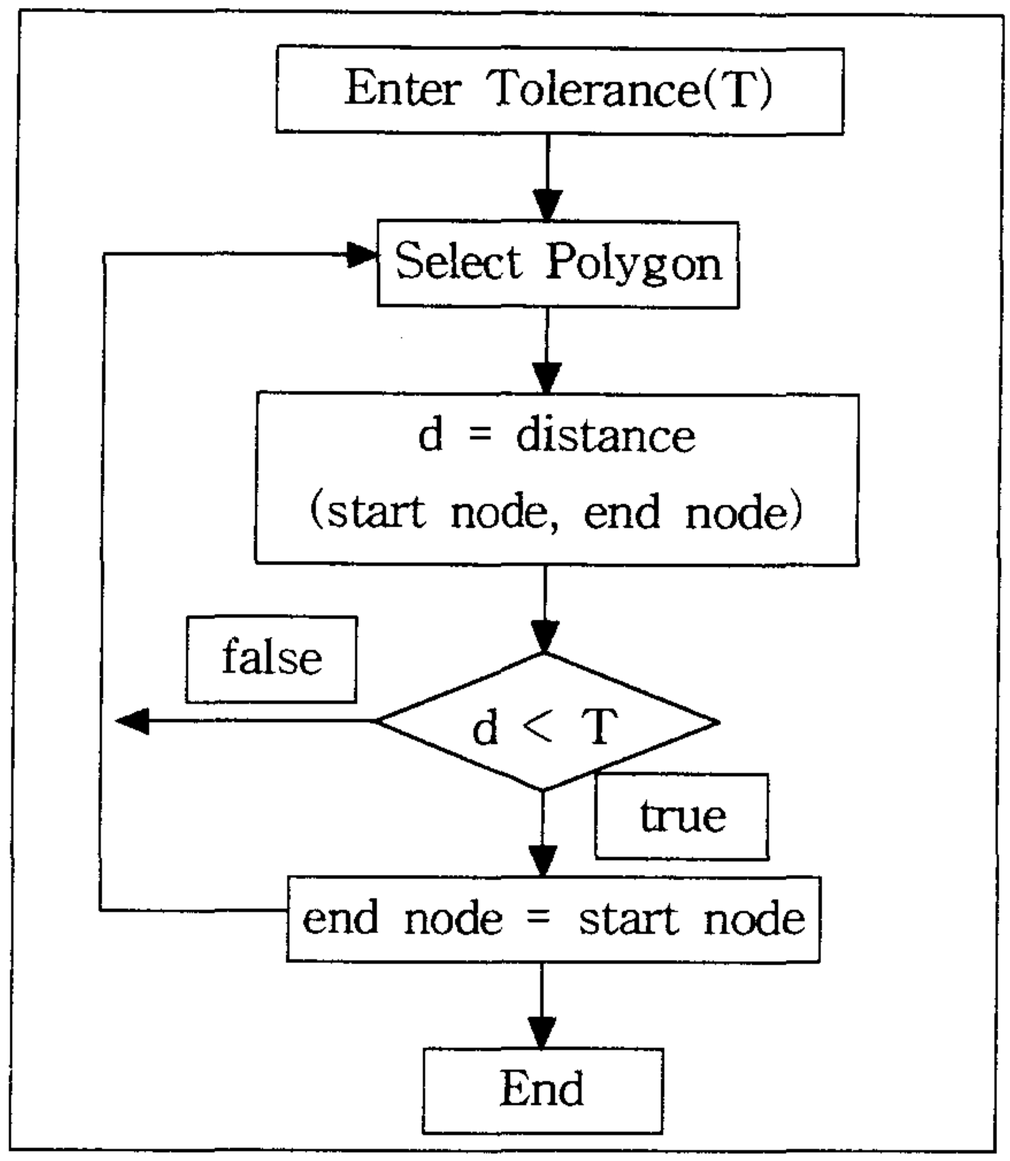


그림 3-8. 열려진 다각형 오류 수정 흐름도

4. 서로 교차하는 체인

서로 교차하는 체인은 3차원을 처리하는 시스템에서는 오류가 되지 않을 수도 있다. 그러나, 2차원 지리 자료를 처리하는 본 연구에서는 교차하는 체인을 오류로 간주한다. 따라서, 교차하는 체인을 찾아내어 교차점을 중심으로 체인들을 분할하고 교차점은 새로운 노드로 등록한다. 서로 교차하는 체인을 분할하지 않으면 하나의 체인이 두 개 이상의 다각형과 부분적으로 관계를 맺게 되기 때문에 체인-다각형 위상 관계를 만드는 것이 어렵다. 고가도로나 입체 교차로와 같은 지리 객체는 3차원 공간상에서는 교차되는 것이 아니지만 2차원 평면상에서는 교차되는 것으로 간주되는 어려움이 있다.

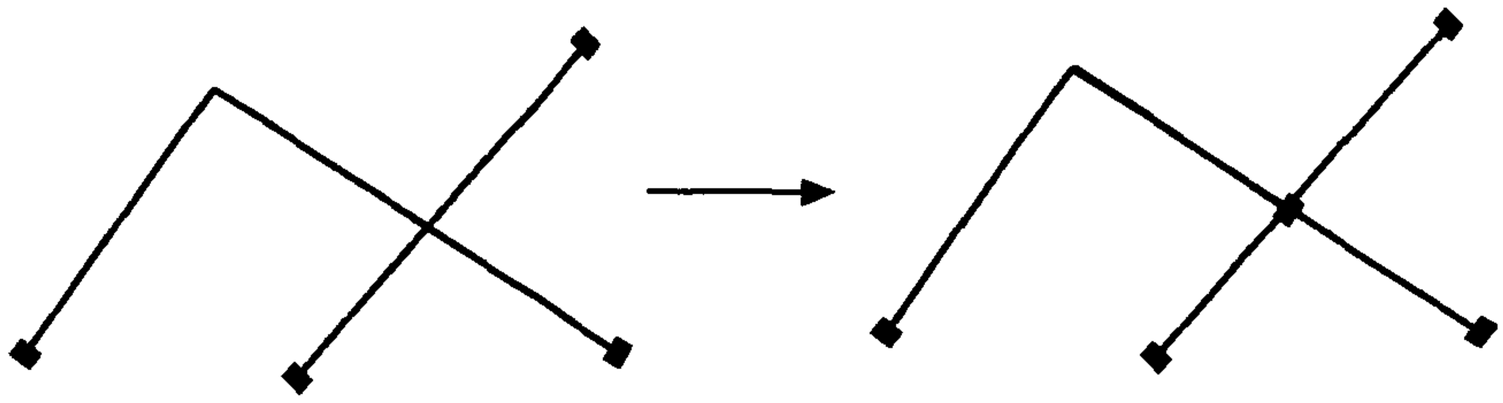


그림 3-9. 서로 교차하는 체인의 수정 예

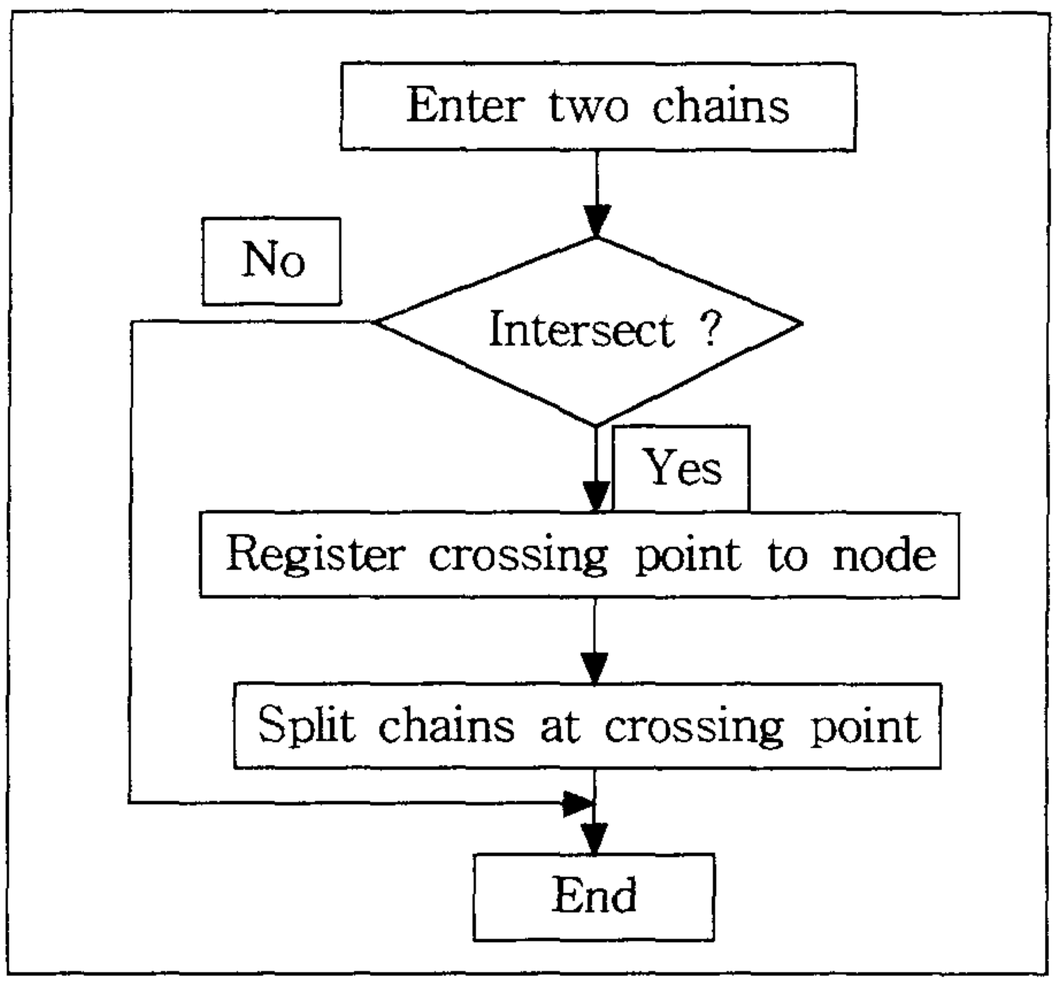


그림 3-10. 교차하는 체인의 수정 흐름도

제 2 절 공간 객체 식별 기능

본 절에서는 본 연구에서 사용한 공간 객체의 식별 기능에 대하여 설명한다. 공간 객체는 크게 노드 객체, 체인 객체, 그리고 다각형 객체로 구별되고, 각각의 비공간 속성 정보는 서로 다르게 표현된다. 그리고, 각 객체별로 객체를 선택하는 방법이 다르므로 공간 객체의 식별은 세 가지로 분리된 기능을 제공한다.

객체의 선택은 마우스를 이용하여 구현되었고, 선택된 객체는 다른 객체들과의 구별을 위하여 객체의 색을 반전시켰으며 선택된 객체와 연관된 속성 정보를 속성 정보 창을 이용하여 사용자에게 보여준다. 객체를 선택할 때에는 각각 별도의 객체 탐색 방법을 사용하였다. 노드 객체를 탐색할 때에는 현 마우스의 위치로부터 각 노드 객체까지의 거리를 계산하여 마우스의 위치에서 가장 가까운 거리에 있는 객체를 선택하였다. 체인 객체도 현 마우스의 위치로부터 각 체인 객체까지의 거리를 계산하여 마우스의 위치에서 가장 가까운 거리에 있는 객체를 선택하였다. 그러나, 다각형을 선택할 때에는 노드 객체나 체인 객체와는 다른 방법을 사용하였다. 다각형 객체는 거리 계산을 하지 않았으며 현 마우스의 위치가 다각형의 내부에 위치하는지 혹은 외부에 위치하는지를 판정하는 Point-in-Polygon 알고리즘을 이용하여 마우스의 현 위치를 포함하는 다각형 객체를 선택하였다. 이러한, 객체 식별 흐름도는 [그림 3-11]에 나타내었고, 객체 식별 결과는 [그림 3-12], [그림 3-13], [그림 3-14]에 나타내었다.

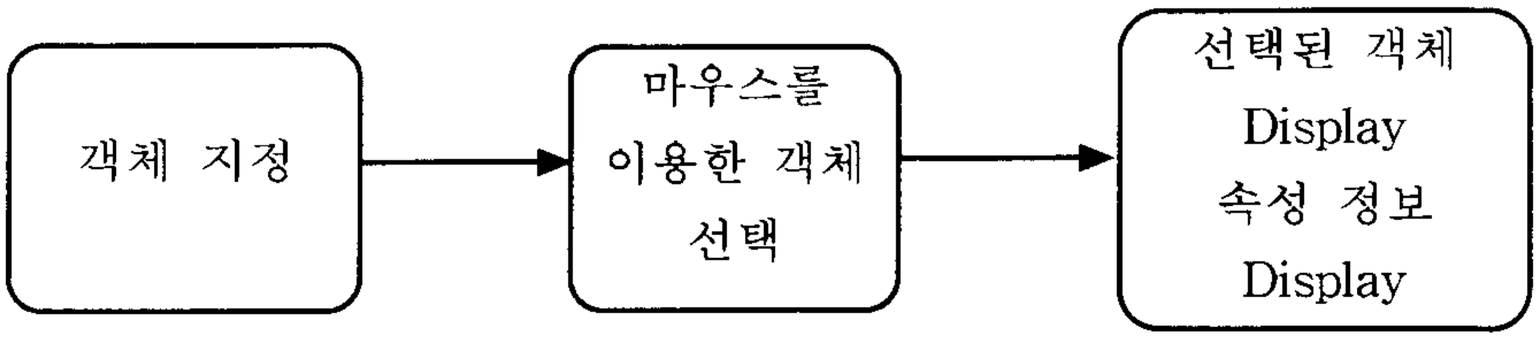


그림 3-11. 객체 식별 흐름도

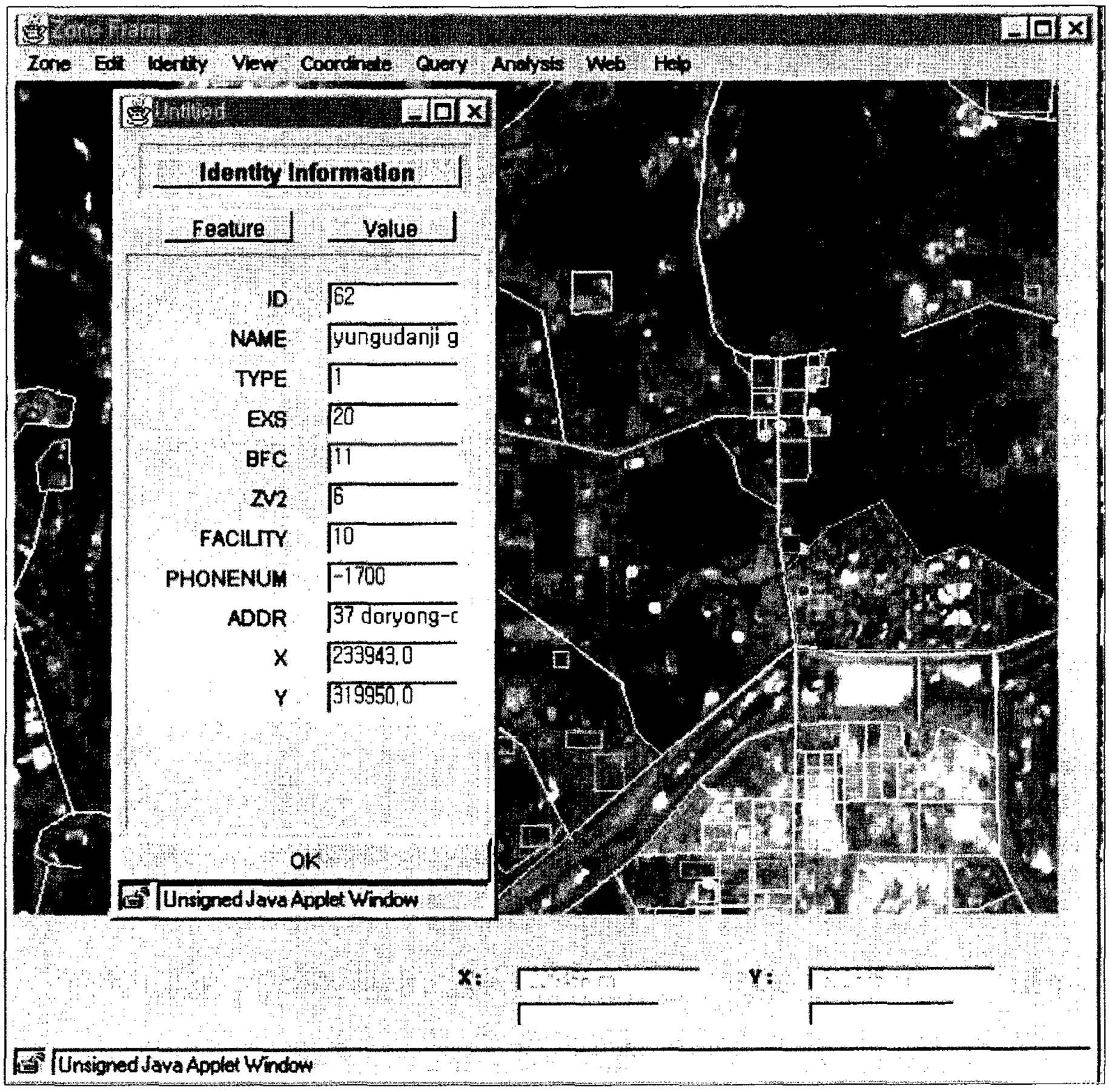


그림 3-12. 노드 객체 식별 결과 예

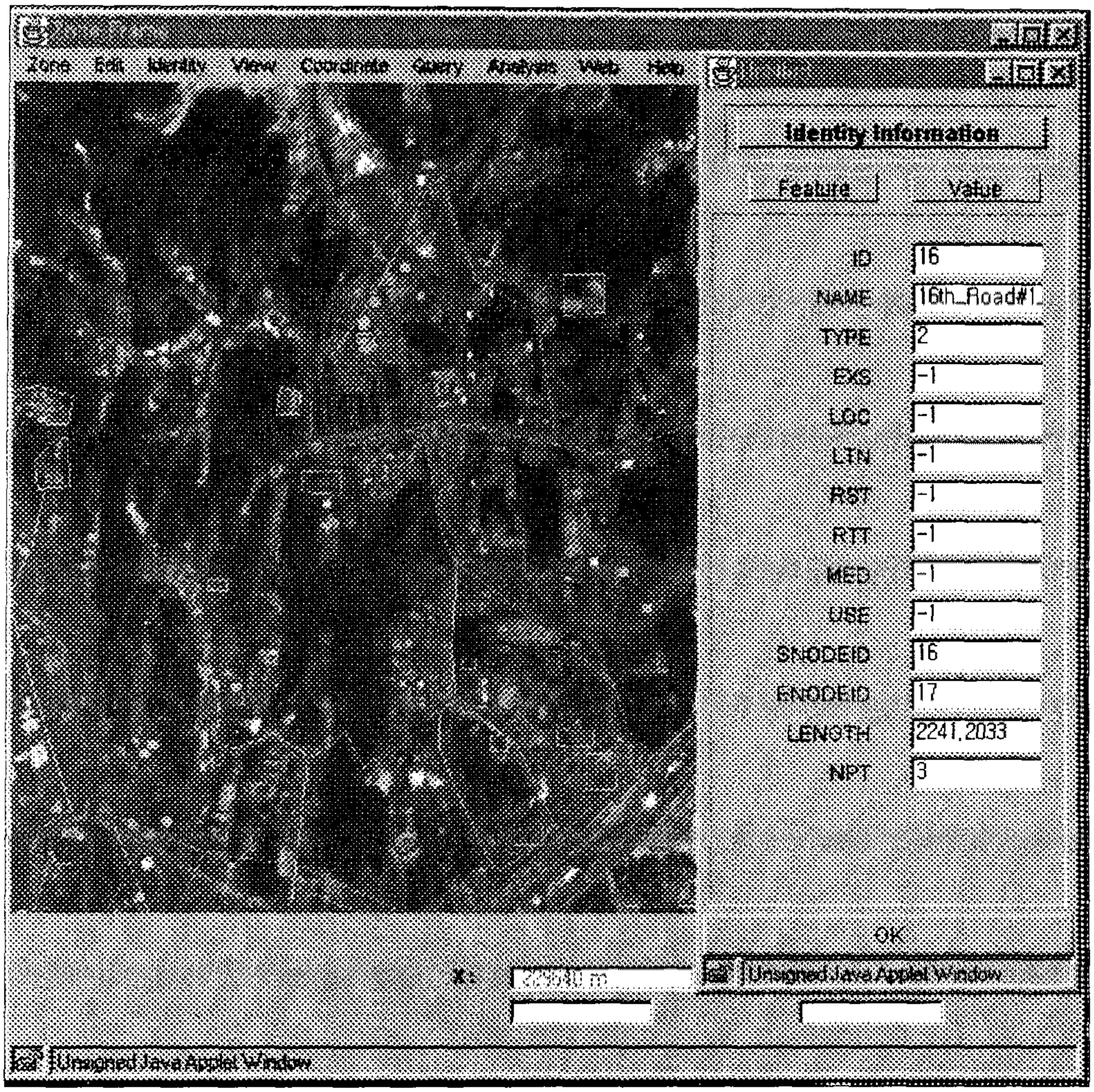


그림 3-13. 체인 객체 식별 결과 예

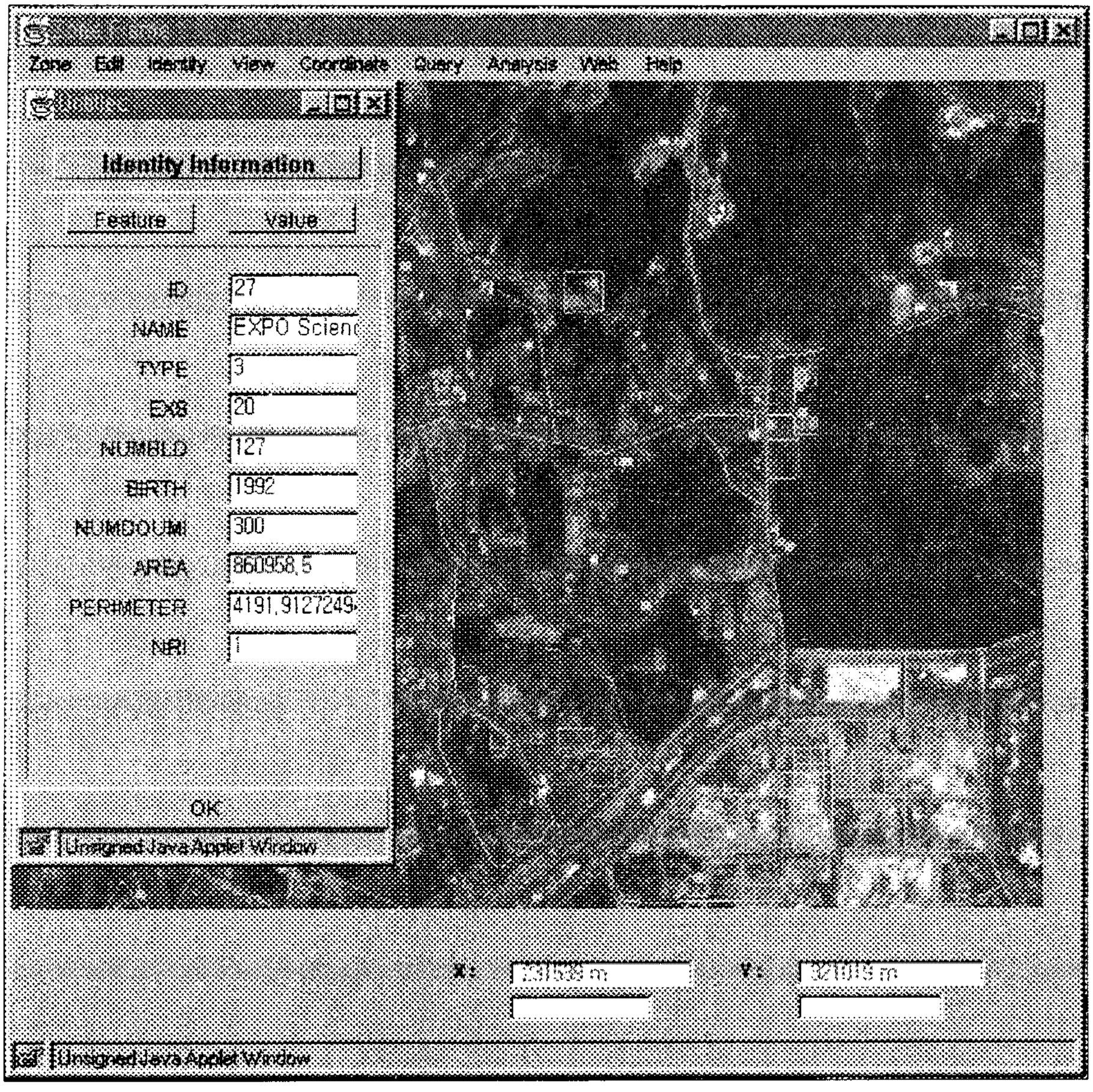


그림 3-14. 다각형 객체 식별 결과 예

제 3 절 공간 객체 편집 기능

본 절에서는 본 연구에서 개발한 공간 객체 편집 기능에 대하여 설명한다. 본 연구에서의 편집 기능은 객체가 지니는 공간 좌표와 비공간 속성 정보를 대상으로 구현되어 있다. 편집 기능은 공간 객체의 생성, 삭제, 이동, 복사, 갱신 등 5가지 기능을 제공한다. 위 5가지 편집 기능들은 공간 좌표의 값과 비공간 속성 정보의 값을 함께 변화시키지만, 이동과 복사 기능을 수행할 때에는 비공간 속성 정보의 값은 변화되지 않는다.

1. 생성

공간 객체를 생성하는 기능이다. 생성되는 객체는 노드 객체, 체인 객체, 다각형 객체 중 하나가 되며, 각 객체의 속성 정보는 사용자가 키보드를 통하여 입력하여야 한다. 편집의 객체 생성을 선택하면 현재 선택되어 화면에 그려진 객체들을 표시하는 [그림 3-15]와 같은 객체 리스트가 표시되고, 이 객체 리스트에 포함된 객체 중 하나를 생성 객체로 선택한다. 이 객체 리스트에 포함되지 않은 객체는 생성시킬 수 없다.

객체 생성은 마우스를 이용하여 수행된다. 원하는 위치로 마우스를 이동한 후 마우스의 왼쪽 버튼을 누르면 현 위치의 좌표 값을 읽어 들여 만들려고 하는 객체의 공간 좌표로 이용한다. 공간 좌표의 입력이 완성되어 하나의 객체가 생성되면 이 객체의 속성 정보를 입력할 수 있는 [그림 3-16]와 같은 속정 정보 입력 창이 뜬다. 생성된 객체의 종류에 따라 속성 정보가 다르기 때문에 속성 정보 입력 창도 객체에 따라 변한다. 또한 속성 정보 입력 창의 입력 정보 중에서 시스템에 의해 자동적으로 입력되는 부분과 변경되어서는 안될 부분은 색이 반전되어 나타나며 이 부분은 사용자가 속성을 입력하거나 속성의 내용을 변경

할 수 없는 부분이다. [그림 3-16]은 다각형 객체인 연구소 객체를 생성시킨 예이다.

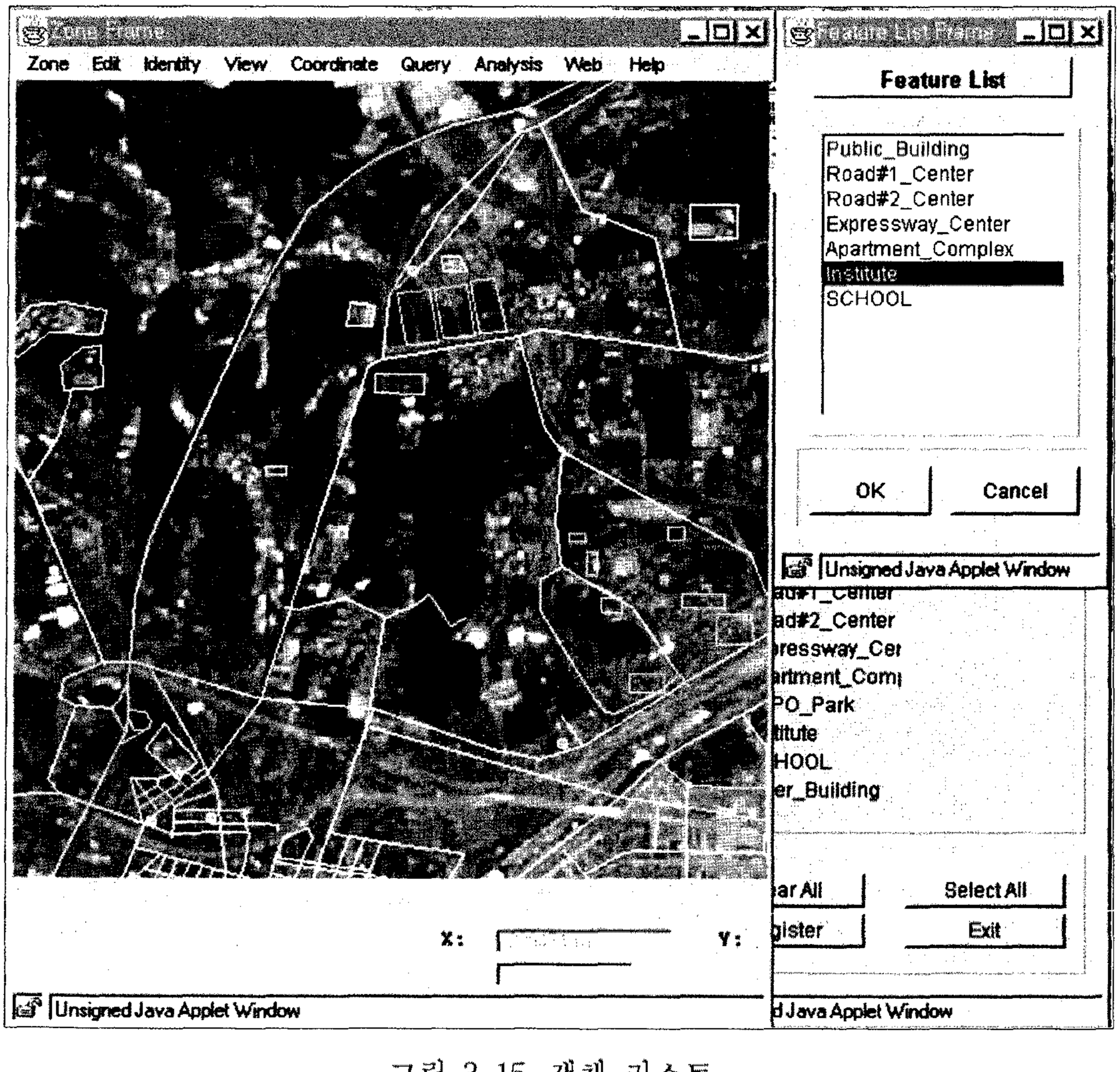


그림 3-15. 객체 리스트

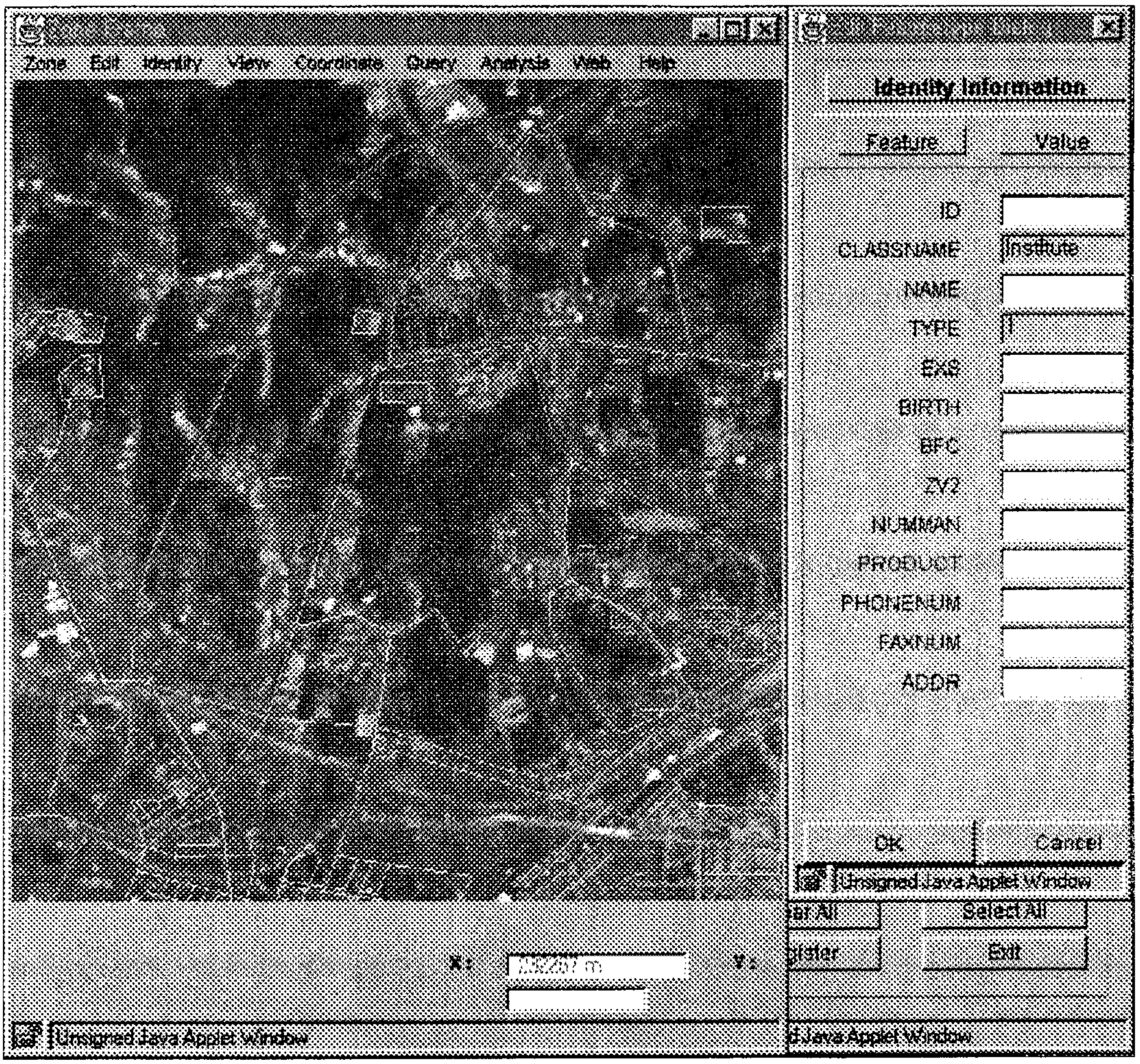


그림 3-16. 연구소 객체 생성의 예

2 삭제

현재 선택된 공간 객체를 삭제하는 기능이다. 삭제되는 객체는 현재 선택된 객체들 중 하나가 된다. 삭제 대상이 되는 객체는 마우스를 이용하여 선택한다. 마우스를 이동하여 삭제하고 싶은 객체를 선택할 수도 있고, 원하는 위치로 마우스를 이동하여 마우스의 왼쪽 버튼을 누르면 현재 마우스가 위치한 곳으로부터 일정한 거리 이내에 위치한 객체들이 3가지 이하로 객체 선택 창에 [그림 3-17]의 오른쪽에 보이는 것과 같이 리스트 형식으로 나타나다. 이 객체들은 노드 객체, 체인 객체, 다각형 객체들이 각각 하나씩이다. 만일 현재 마우스의 위치로부터 일정한 거리 이내에 있는 객체가 하나도 없으면 선택되어지는 객체는 없다. 사용자는 객체 선택 창의 리스트에 나타난 객체들 중 삭제를 원하는 객체를 선택하면 선택된 객체는 삭제된다. [그림 3-17]은 마우스의 왼쪽 버튼을 이용하여 삭제 대상이 될 객체를 선택하였을 때, 체인 객체인 도로 중심선 객체와 다각형 객체인 내부 건물 객체가 선택된 것을 표시하고 있다. 사용자는 객체 선택 창의 리스트에 있는 이들 선택된 두 개의 객체 중에서 하나를 선택할 수 있다.

[그림 3-17]에서 선택된 객체는 내부 건물 객체이다. 이러한 과정을 거쳐서 선택된 객체는 객체 선택 창의 중간 부분의 Feature란에 객체 이름이 표시되고, Type란에 객체의 종류가 표시된다. 원하는 객체를 객체 선택 창에서 선택한 후에 O.K. 버튼을 누르면 선택된 객체는 삭제되고 화면에서 제거된다. 객체 삭제의 결과는 [그림 3-18]에 표시하였다.

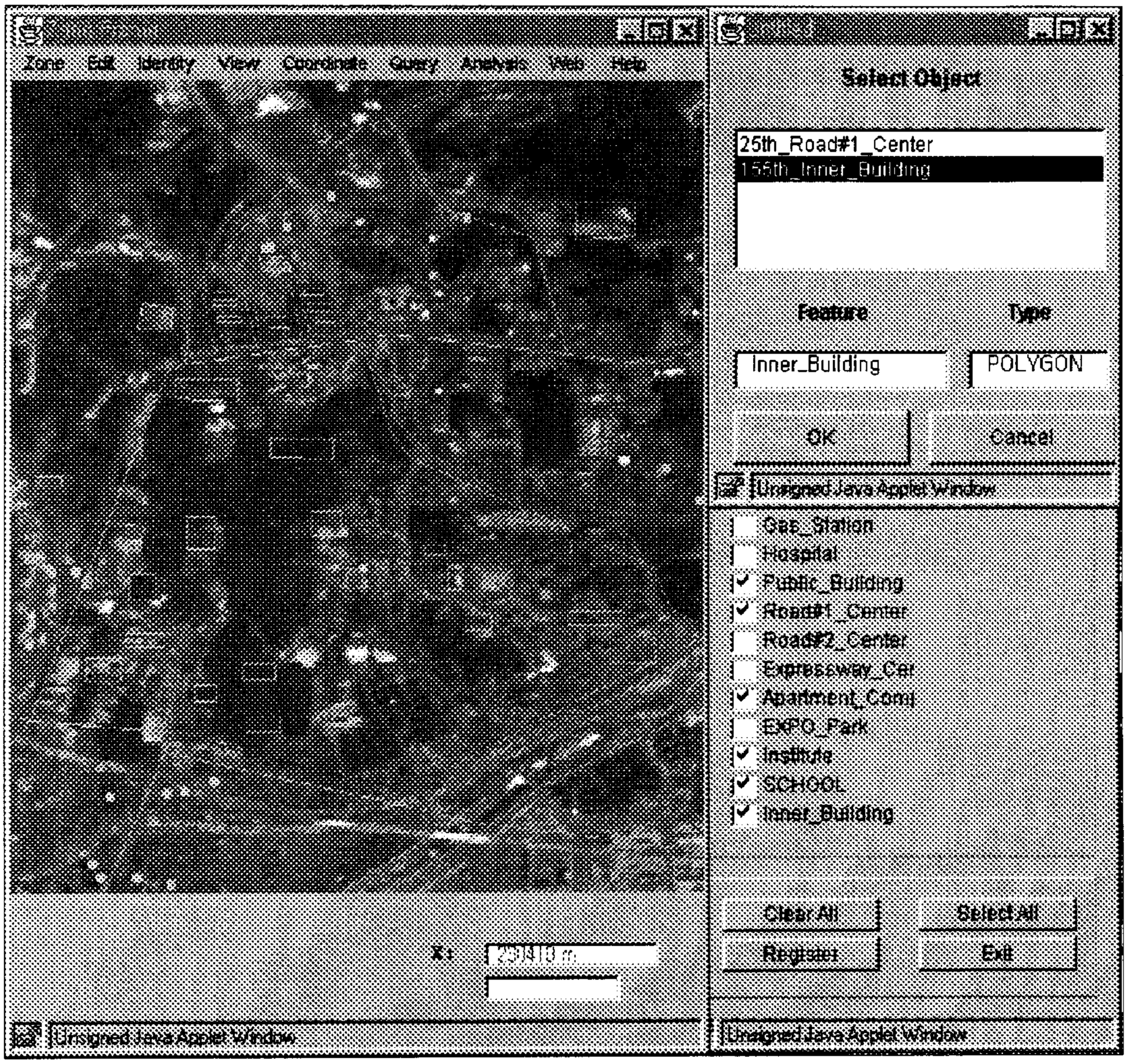


그림 3-17. 삭제될 객체 선택

그림 3-18. 객체 삭제 결과

3. 이동

현재 선택된 공간 객체를 새로운 위치로 이동하는 기능이다. 이동되는 객체는 현재 선택된 객체들 중 하나가 된다. 이동 대상이 되는 객체는 마우스를 이용하여 선택한다. 마우스를 이동하여 이동하고 싶은 객체를 선택할 수도 있고, 원하는 위치로 마우스를 이동하여 마우스의 왼쪽 버튼을 누르면 현재 마우스가 위치한 곳으로부터 일정한 거리 이내에 위치한 객체들이 3가지 이하로 객체 선택 창에 [그림 3-19]의 오른쪽에 보이는 것과 같이 리스트 형식으로 나타나다. 이 객체들은 노드 객체, 체인 객체, 다각형 객체들이 각각 하나씩이다. 만일 현재 마우스의 위치로부터 일정한 거리 이내에 있는 객체가 하나도 없으면 선택되어지는 객체는 없다. 사용자는 객체 선택 창의 리스트에 나타난 객체들 중 이동을 원하는 객체를 선택하고 마우스를 이 객체가 새롭게 위치될 위치로 이동하여 마우스의 왼쪽 버튼을 누르면 선택된 객체는 새로운 위치로 이동된다.

[그림 3-19]는 마우스의 왼쪽 버튼을 이용하여 이동 대상이 될 객체를 선택하였을 때, 체인 객체인 도로 중심선 객체와 다각형 객체인 연구소 객체가 선택된 것을 표시하고 있다. 사용자는 객체 선택 창의 리스트에 있는 이들 선택된 2개의 객체 중에서 하나를 선택할 수 있다. [그림 3-19]에서 선택된 객체는 체인 객체인 도로 중심선 객체이다. 이러한 과정을 거쳐서 선택된 객체는 객체 선택 창의 중간 부분의 Feature란에 객체 이름이 표시되고, Type란에 객체의 종류가 표시된다. 원하는 객체를 객체 선택 창에서 선택한 후에 O.K. 버튼을 누르면 선택된 객체는 이전의 위치에서 사라지고, 새로운 위치로 이동되어 표시된다. 객체 이동의 결과는 [그림 3-20]에 표시하였다.

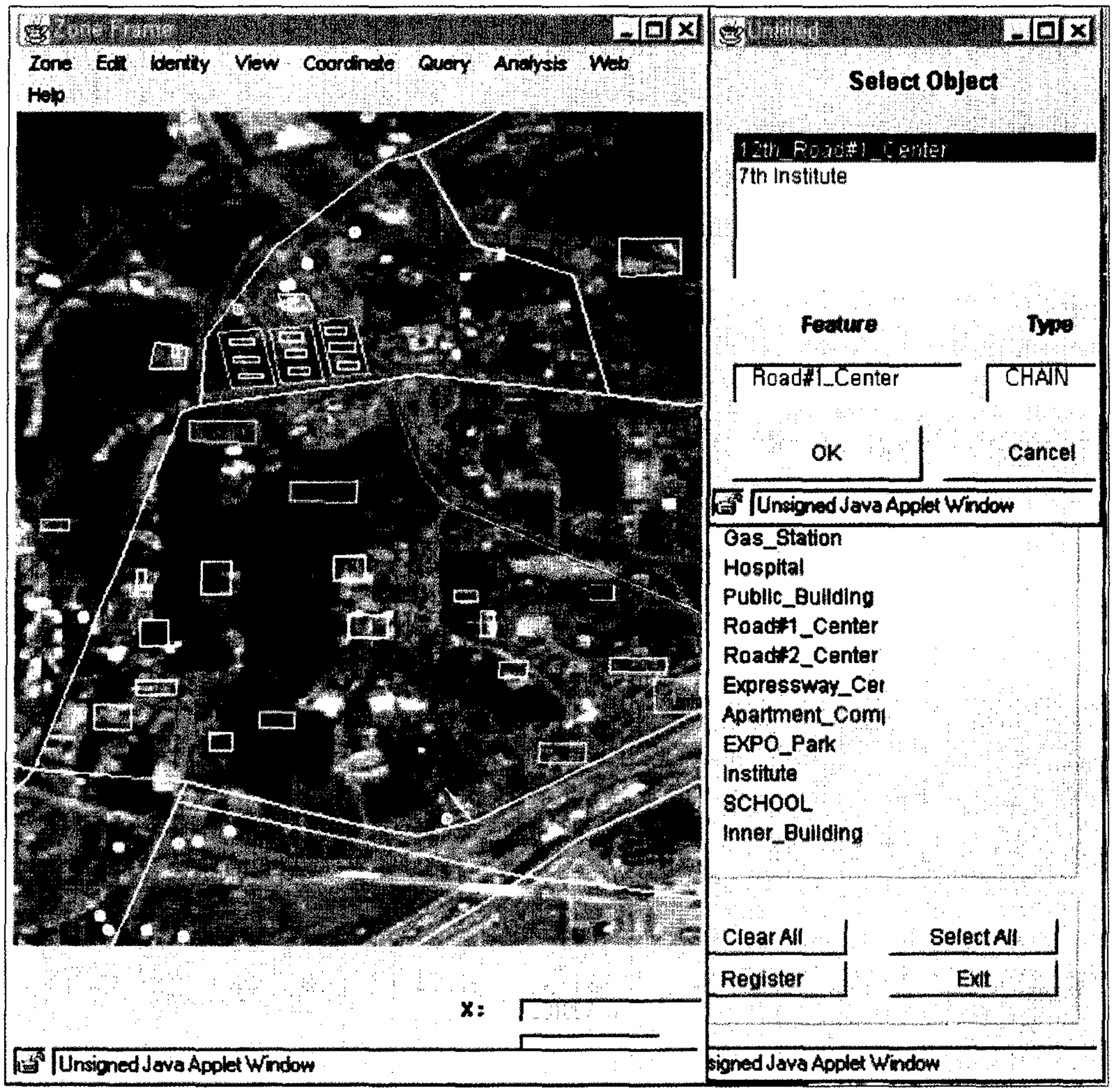


그림 3-19. 이동할 객체 선택

그림 3-20. 객체 이동 결과

4. 복사

현재 선택된 공간 객체와 동일한 모양과 특성을 갖는 새로운 객체를 생성하여 원하는 장소에 위치시키는 기능이다. 복사되는 객체는 현재 선택되어 화면에 표시되어 있는 객체들 중 하나가 된다. 복사 대상이 되는 객체는 마우스를 이용하여 선택한다. 마우스를 이용하여 복사의 대상이 되는 객체를 선택하기 위해서는, 원하는 위치로 마우스를 이동한 후 마우스의 왼쪽 버튼을 누르면 현재 마우스가 위치한 곳으로부터 일정한 거리 이내에 위치한 객체들이 3가지 이하로 객체 선택 창에 [그림 3-21]의 오른쪽에 보이는 것과 같이 리스트 형식으로 나타난다. 이 객체들은 노드 객체, 체인 객체, 다각형 객체들이 각각 하나씩이다. 만일 현재 마우스의 위치로부터 일정한 거리 이내에 있는 객체가 하나도 없으면 객체 선택 창에 표시되는 객체는 없다. 사용자는 객체 선택 창의 리스트에 나타난 객체들 중 복사하기를 원하는 객체를 선택하고 마우스를 이 객체가 새롭게 위치될 위치로 이동하여 마우스의 왼쪽 버튼을 누르면 선택된 객체와 동일한 모양과 특성을 지니는 새로운 객체가 생성된다.

[그림 3-21]은 마우스의 왼쪽 버튼을 이용하여 복사 대상이 되는 객체를 선택하였을 때, 체인 객체인 도로 중심선 객체와 다각형 객체인 연구소 객체가 선택된 것을 표시하고 있다. 사용자는 객체 선택 창의 리스트에 있는 이들 선택된 2개의 객체 중에서 하나를 선택할 수 있다. [그림 3-21]에서 선택된 객체는 다각형 객체인 연구소 객체이다. 이러한 과정을 거쳐서 선택된 객체는 객체 선택 창의 중간 부분의 Feature난에 객체 이름이 표시되고, Type란에 객체의 종류가 표시된다. 원하는 객체를 객체 선택 창에서 선택한 후에 O.K. 버튼을 누르면 선택된 객체는 [그림 3-22]와 같이 붉은 색으로 반전되고, 원하는 위치로 마우스를 이동하여 마우스의 왼쪽 버튼을 누르면 선택된 객체와 동일한 모양과 동일한 속성을 갖는 객체가 새로운 위치에 생성된다. 객체 복사의 결과는 [그림

3-23]에 표시하였다.

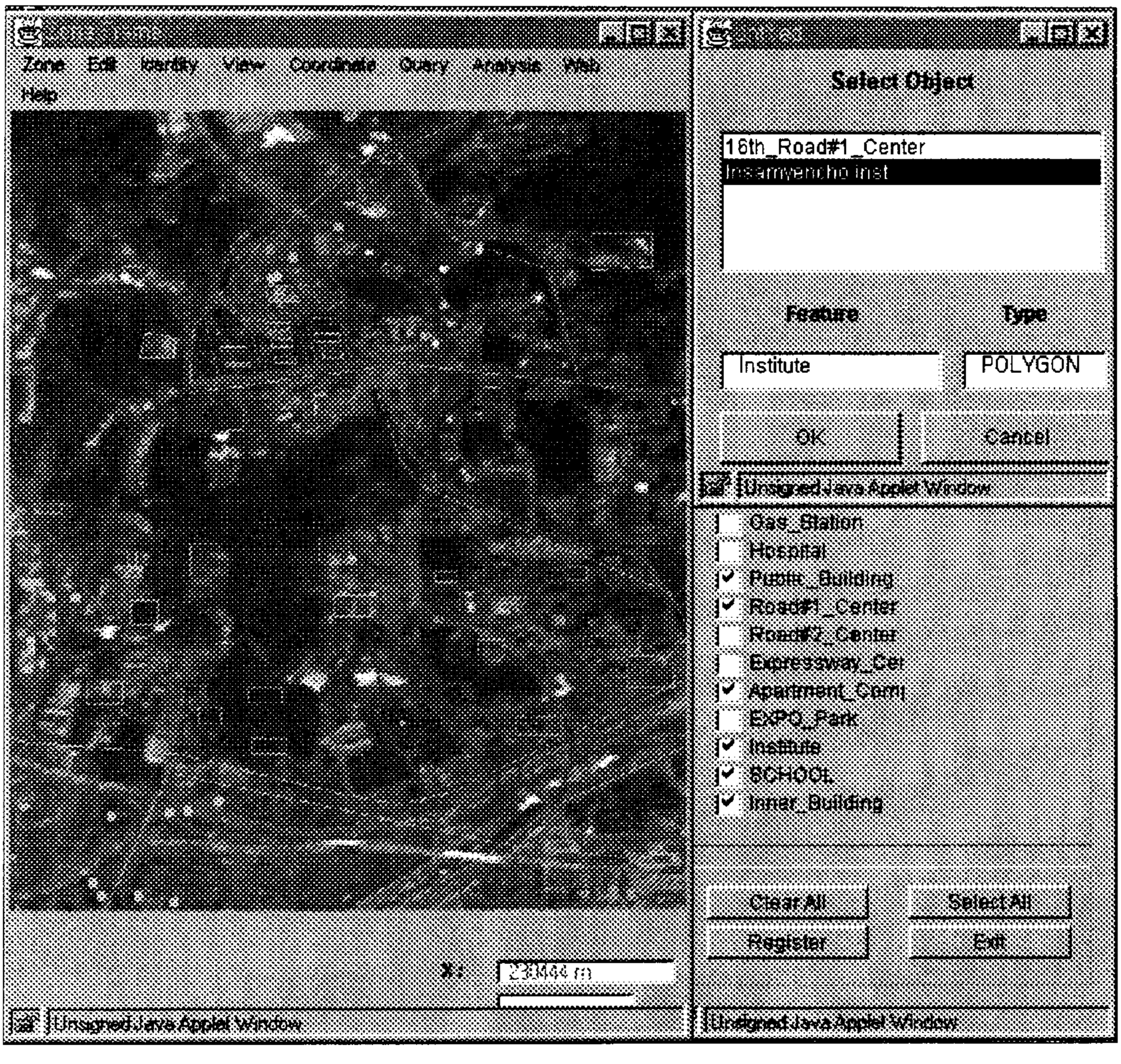


그림 3-21. 객체 선택

그림 3-22. 복사할 객체 선택

그림 3-23. 객체 복사 결과

제 4 절 위상 관계 형성 기능

지리 정보 시스템의 입력 과정을 거쳐 획득된 지리 자료는 단순히 점 성분, 선 성분, 다각형 성분을 갖는 자료들로 분류될 뿐, 지리 객체들이 실세계에서 가지고 있었던 지리 객체들 사이의 상호 관련성을 잃어버리게 된다. 따라서, 지리 객체들이 실세계에서 가지고 있었던 지리 객체들 사이의 공간적인 상호 관련성을 형성시켜 주는 위상 관계 형성(Topology Building) 과정이 필요하다.

벡터형 지리 정보 시스템에서 처리하는 지리 객체들 사이의 공간적인 관계를 표현하는 위상 관계는 노드-체인 위상 관계, 체인-다각형 위상 관계, 좌-우 위상 관계로 분류된다. 공간 객체들 사이에 위상 관계를 가지는 것은 분석 과정에서 매우 중요한 역할을 할뿐만 아니라, 지리 정보 시스템을 컴퓨터 그래픽, CAD, 데이터베이스 시스템 등과 구분을 짓는 여러 요소들중 하나가 된다.

위상 관계 형성 과정은 모든 공간 객체들을 대상으로 이루어지므로 데이터베이스나 파일에 저장된 모든 지리 객체들을 주 기억 장치에 저장하고 위상 관계를 형성하는 것이 일반적으로 많이 사용하는 방법이다. 그러나, 이러한 방법은 처리 대상이 대용량일 때에는 모든 지리 자료들을 주 기억 장치에 읽어들일 수 없는 경우가 생기기 때문에 적용할 수 없는 단점이 있다. 따라서, 이러한 어려움을 극복할 수 있는 방법은 처리 대상을 전체로 하지 않고 전체 자료를 여러 개의 소영역으로 나눈 후에 나누어진 소영역을 대상으로 위상 관계 형성 과정을 거쳐 전체를 통합하는 것이다.

본 연구에서는 입력된 공간 객체들을 저장하고, 저장된 자료들에 대하여 R^*-tree를 이용하여 공간 색인을 구현하였으며, R^*-tree의 색인을 이용한 자료 검색은 Page 단위로 이루어지는 특성을 이용하여 Page 단위의 위상 관계를 형성하므로써 빠르고 정확하게 지리 객체들 사이의 위상관계를 형성하였다. [그림 3-24]에 전체 시스템의 개략도를 나타내었다.

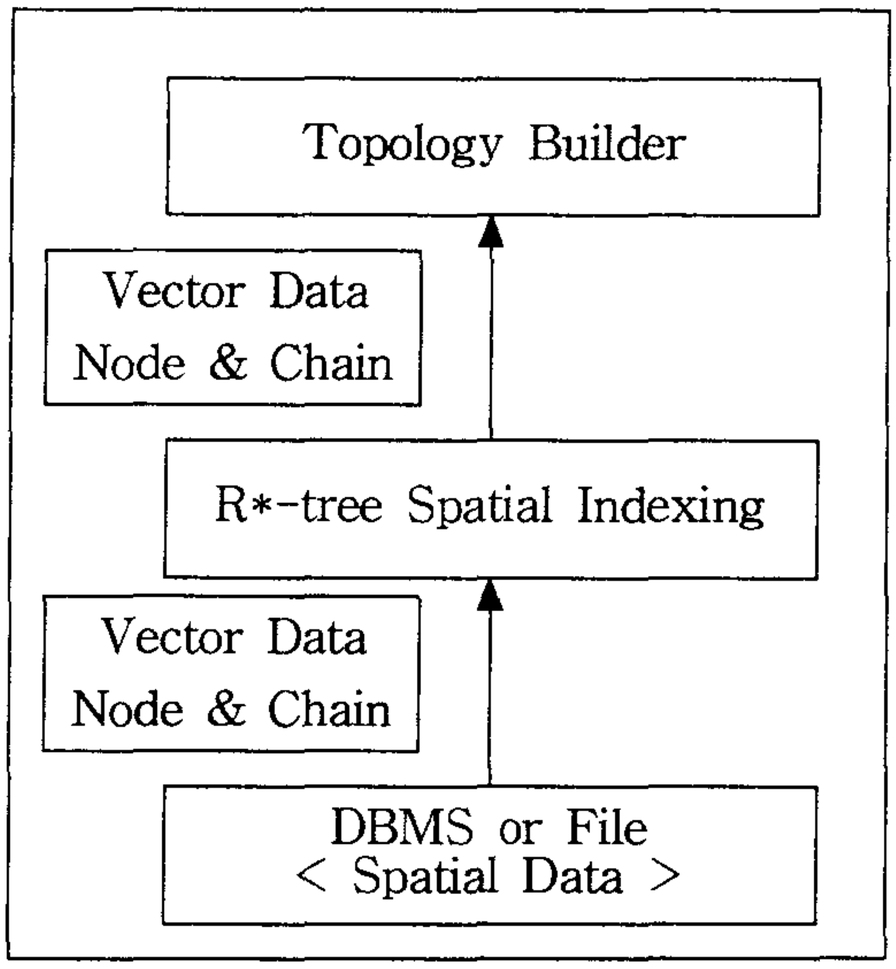


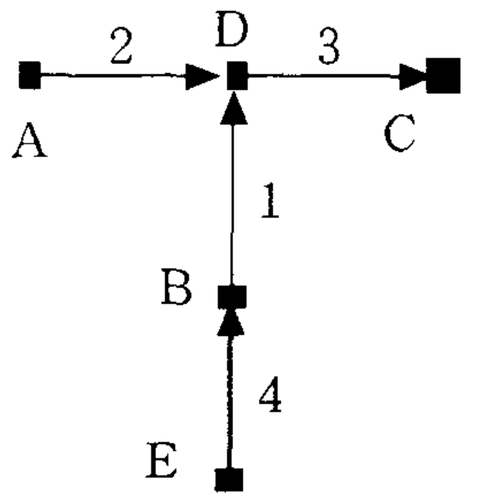
그림 3-24. 전체 시스템 구성도

1. 위상 관계의 종류

가. 노드-체인 위상 관계

노드-체인 위상 관계는 체인과 노드 사이의 관계를 맺어 주는 것으로 [그림 3-25]에 개략적인 예를 나타내었다. 이 위상 관계는 지리 자료를 읽는 과정에서 형성시켜 준다. 체인을 구성하는 점들을 읽어 들 인후에 체인의 시작점과 끝점을 각각 시작 노드(From-Node)와 끝 노드(To-Node)로 등록한다. 벡터형 지리 정보 시스템의 체인은 방향성을 가지고 있기 때문에 노드-체인 위상 관계를 맺어줄 수 있다. 따라서, [그림 3-26]에 나타난 것과 같이 노드 D가 공유된다는 것으로 아크 1, 2, 3이 서로 연결됨을 알 수 있다. 따라서, 컴퓨터는 체인 1에서 체인 3으로는 직접 연결될 수 있지만 체인 4에서 체인 3으로는 직접 연결

될 수 없음을 파악할 수 있다.

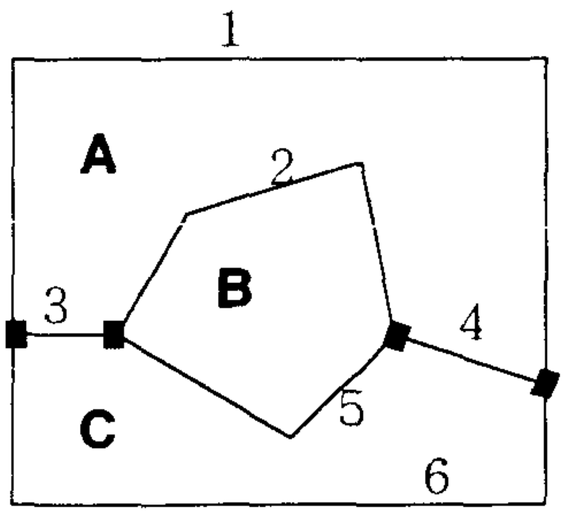


Chain	From-Node	To-Node
1	B	D
2	A	D
3	D	C
4	E	B

그림 3-25. 개략적인 노드-체인 위상 관계의 예

나. 체인-다각형 위상 관계

체인-다각형 위상 관계는 다각형을 구성하는 체인들을 찾아내는 것이다. 이것은 지리 자료를 읽는 과정에서는 형성 시켜줄 수 없고, 위상 관계 형성 과정에서 형성된다. 다각형은 닫혀져 있으므로 하나의 체인이 다각형을 형성하는 경우도 있으며 여러 개의 체인들이 하나의 다각형을 형성할 수도 있다. [그림 3-26]은 개략적인 체인-다각형 위상 관계를 나타내었다. [그림 3-26]에서 다각형 C는 체인1, 4, 2, 그리고 3으로 구성된다.

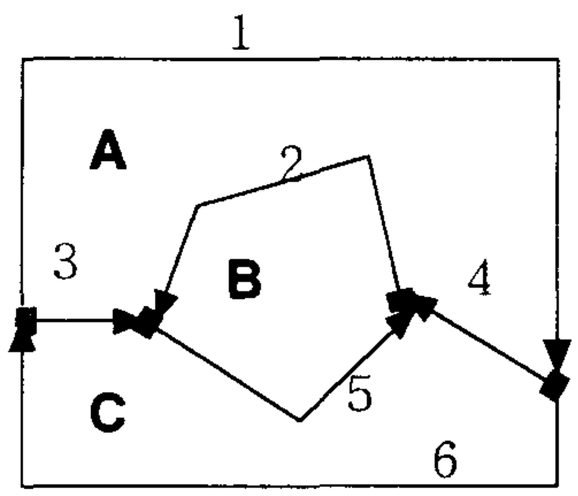


Polygon	Chain-List
A	1 4 2 3
B	2 5
C	4 6 3 5

그림 3-26. 개략적인 체인-다각형 위상 관계의 예

다. 좌-우 위상 관계

좌-우 위상 관계는 체인과 다각형 사이의 위상 관계로써 체인이 접하고 있는 좌측과 우측의 다각형을 찾아내는 것이다. 이것은 체인이 방향성을 가지고 있기 때문에 가능한 것으로 체인의 진행방향을 기준으로 좌-우가 결정된다. 좌-우 위상 관계도 체인-다각형 위상 관계처럼 위상 관계 형성 과정을 통하여 만들어진다. 개략적인 예는 [그림 3-27]에 나타내었다. [그림 3-27]에서 체인 5의 좌측 다각형은 B이고 우측 다각형은 C가 된다.



Chain	Left-Polygon	Right-Polygon
1	0	A
2	B	A
3	A	C
4	C	A
5	B	C
6	0	C

그림 3-27. 개략적인 좌-우 위상 관계의 예

2. R*-tree 공간 색인을 이용한 위상 관계 형성

가. R*-tree 공간 색인

지리 정보 시스템에서 처리하는 지리 자료는 기본적으로 대용량이다. 따라서, 대용량의 자료를 빠르게 접근하는 검색 방법이 필요하다. R*-tree는 공간 검색 기법 중에서 비교적 우수한 성능을 내는 것으로 알려져 있다. 따라서, 본 연구에서는 지리 자료들이 데이터베이스나 파일에 존재할 때 빠른 지리 자료 검색을 위하여 R*-tree를 이용하여 지리 자료들에 대하여 공간 색인을 만들었다.

[그림 3-28]는 R*-tree 공간 색인의 예제로 한 페이지에 들어갈 수 있는 자료의 수는 최대 4개로 한정되어 있다. 실제 자료는 영문 소문자로 되어있는 사각형 영역에 포함되어 있고 R*-tree의 leaf 노드에 해당된다. 영문 대문자는 R*-tree의 중간 노드에 해당된다.

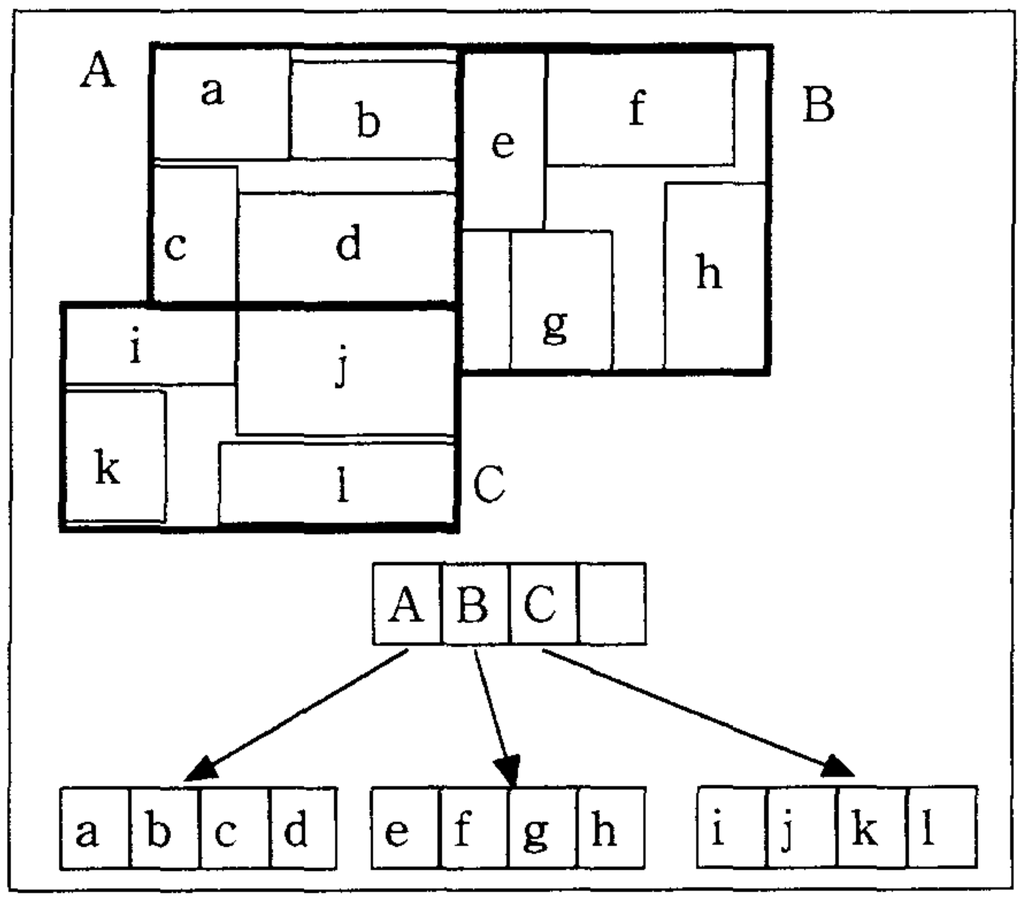


그림 3-28. R*-tree 의 예

나. 위상 관계 형성 방법

본 논문에서 제안된 위상 관계 형성 알고리즘은 체인을 기준으로 다각형을 찾아가는 과정은 winged-edge 방법과 유사하지만 다음과 같은 차이가 있다.

winged-edge 방법에서는 체인 구조에 체인의 시작 노드에 연결된 left edge와 끝 노드에 연결된 right edge를 갖는다. 따라서, winged-edge 방법은 체인의 left edge나 right edge를 찾아가면서 체인-다각형 위상 관계와 좌-우 위상 관계를 형성한다.

그러나, 본 논문에서 구현한 알고리즘은 각 노드를 기준으로 이 노드와

연결된 체인을 연결해 주는 노드-체인 테이블을 위상 관계 형성을 위한 전처리 과정에서 만든다. 이 노드-체인 테이블은 위상 관계를 형성하는 과정에서 다각형을 찾는데 에 사용된다. 따라서, 본 논문에서 구현한 알고리즘은 노드 테이블, 체인 테이블, 노드-체인 테이블 세 가지를 이용하여 체인-다각형 위상 관계와 좌-우 위상 관계를 다음의 세 가지 규칙을 이용하여 형성한다.

규칙 1 : 다각형 검색은 체인 테이블에 있는 임의의 체인에서 시작한다.

규칙 2 : 다각형 검색 과정에서 시작한 체인과 동일한 체인을 재 검색하거나 시작한 노드와 동일한 노드를 재 검색하면, 체인-다각형 위상 관계 (새로운 다각형으로 등록)를 형성한 후 좌-우 위상 관계 (각 체인의 왼쪽 혹은 오른쪽 다각형을 설정)를 형성한다.

규칙 3 : 모든 체인에 대하여 좌-우 위상 관계가 형성되면 위상 관계 형성 알고리즘을 끝낸다.

다. 공간 색인을 이용한 위상 관계 형성

R*-tree 공간 색인은 그 특성상 각 노드에 해당하는 페이지(Page)에 지리 자료들을 가지고 있다. 따라서, leaf node에 해당하는 자료들을 각 leaf node 별로 위상 관계를 형성한 후 다른 노드(node)에 해당하는 자료들과 공간적인 관련성이 있는 후보 자료들을 추출한 후 추출된 후보 자료들을 대상으로 한 level 위에서 다시 위상 관계를 형성하는 과정을 반복하는 방법으로 본 연구에서는 R^*-tree 공간 색인을 이용한 지리 자료들 사이의 위상 관계를 형성하였다. 공간 색인을 이용한 위상 관계를 형성하는 처리 과정은 [그림 3-29]에 나타내었다.

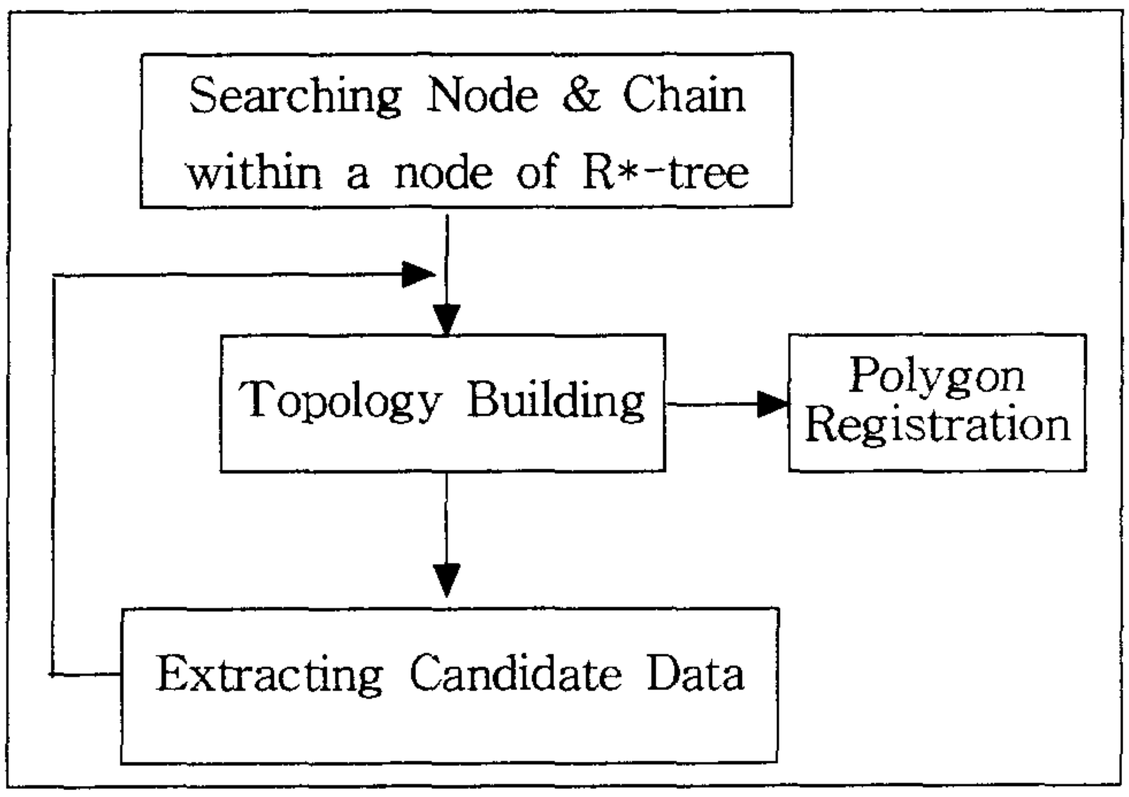


그림 3-29. 공간 색인을 이용한 위상 관계 형성 흐름도

1) 후보 추출 방법

위상 관계를 형성한 후에 다른 노드와 관련성이 있는 후보가 될 수 있는 자료는 반드시 각 노드의 MBR(Minimum Bounding Rectangle)에 연결되어 있어야 한다. 따라서, 후보 자료는 두 가지 과정을 거쳐서 추출하였다.

Step 1 :

위상 관계가 형성된 노드의 모든 체인의 left-polygon과 right-polygon이 설정 되었는 지를 검사한다. 만일 두 가지 모두가 설정된 체인은 후보 자료에서 제외한다.

Step 2 :

노드의 MBR과 연결된 모든 체인들과 이 체인들과 연결된 모든 체인들을 후보 자료로 등록한다.

3. Winged-edge 위상 관계

가. 개요

Winged-edge 위상 관계는 VPF(Vector Product Format) 자료 구조의 기본적인 구조이다. 이 구조는 선형 분석의 일종인 네트워크 분석과 면 위상 관계와 연관된 기능들을 제공한다. 또한, VPF에서 타일(tile) 구조로 저장되어 있는 벡터 자료들의 상호 연결성을 보장하는 중요한 기능도 담당하고 있다. 이 구조의 핵심 요소는 크게 3가지이다. 첫째는 노드 정보이고, 둘째는 에지(Edge) 정보, 셋째는 면(Face) 정보이다.

노드 정보는 각 에지들이 포함하고 있는 시작 노드(start node)와 끝 노드(end node)에 대한 정보로 이 노드 정보는 에지들의 진행 방향에 근거하여 정의가 된다.

에지는 서로 연결되어 있는 이웃한 에지와 오른쪽 에지와 왼쪽 에지에 대한 정보를 포함하고 있다. 오른쪽 에지는 자신의 끝 노드에 연결된 에지들 중에서 자신을 기준으로 반 시계 방향으로 원을 그리며 진행하였을 때에 첫 번째 만나는 에지의 정보이고, 왼쪽 에지는 자신의 시작 노드에 연결된 에지들 중에서 자신을 기준으로 반 시계 방향으로 원을 그리며 진행하였을 때에 첫 번째 만나는 에지의 정보이다.

면 정보는 에지들의 집합으로 이루어진 벡터 자료가 있을 때에 다각형 성분이 만들어지는 경우 에지의 왼쪽 면과 오른쪽 면에 대한 정보를 포함한다. 왼쪽 면과 오른쪽 면은 에지의 진행 방향에 의해 결정된다. 에지의 이 면 정보는 서로 이웃하는 면들에 대한 정보로도 이용된다. 면 정보는 다각형에 대한 정보와 동일하다.

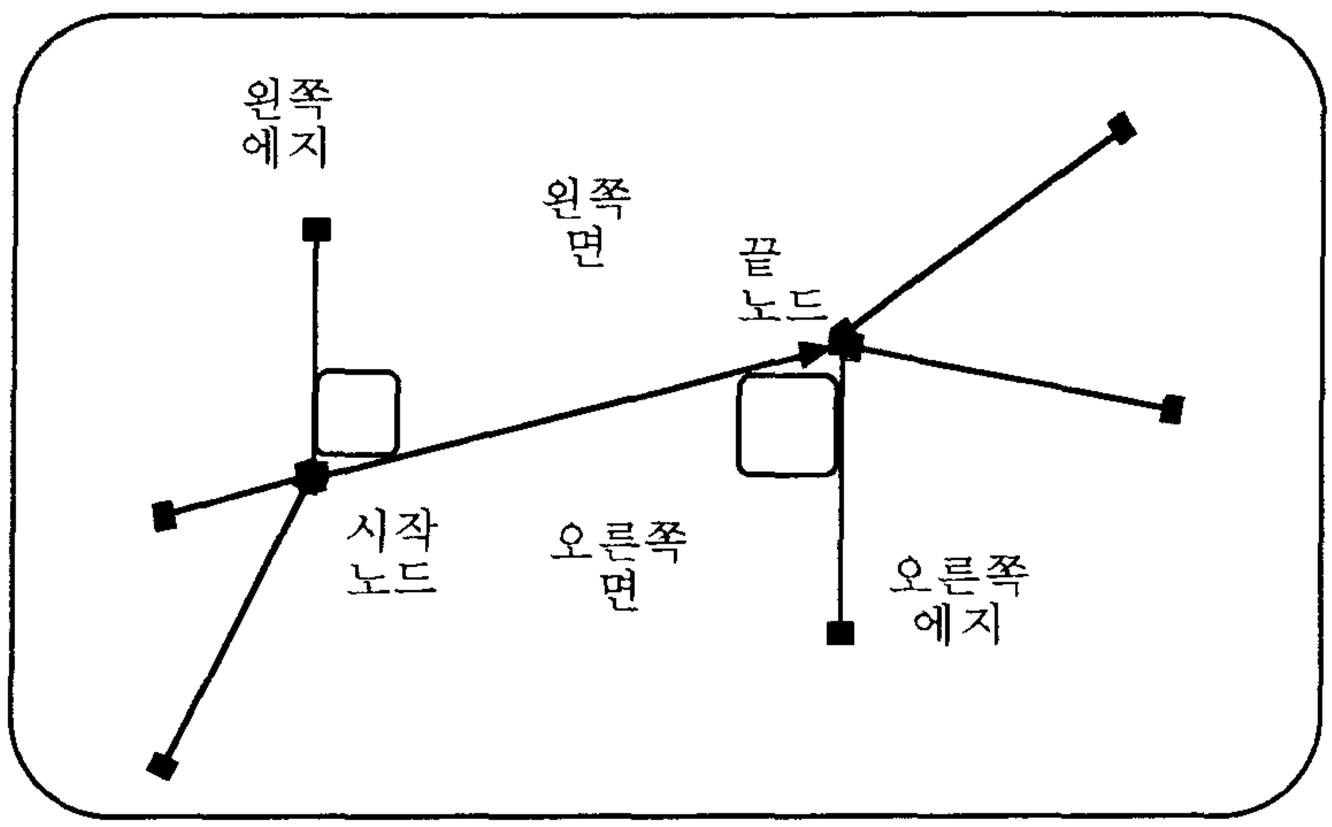


그림 3-30. Winged-edge 구성 요소

나. 위상 관계 모델

VPF에서 제공하는 winged-edge 위상 관계는 4가지가 있으며, 응용분야에 따라 적절하게 선택할 수 있다. 이 위상 관계의 종류는 [표 3-1]에 나타내었다.

Level 0 위상 관계는 아무런 위상 관계가 없는 상태를 나타낸다. 이 모델에서 spaghetti란 벡터 자료 구조에서 공간 정보를 저장하기 위한 모델로 점, 선, 다각형 등의 지리 객체가 특별한 정보 없이 단순한 점들의 나열로 저장되어 있는 상태이다. 즉, 지도나 항공 사진 등으로부터 벡터 자료를 디지타이징한 상태를 의미한다.

Level 1 위상 관계는 Level 0 위상 관계의 정보를 가공하여 노드 정보와 에지 정보를 생성하여 네트워크 분석을 지원하는 모델이다. 이 모델에서는 라인과 라인의 교차를 허용하는 모델로 다각형 성분은 생성되지 않는다.

Level 2 위상 관계는 Level 1 위상 관계와 같이 네트워크 분석을 지원하

는 모델이다. 그러나, 이 모델에서는 선과 선의 교차를 허용하지 않는 특성이 있다. 선 성분들 사이의 연결성 분석이 가능하고, 시작점과 종점이 주어졌을 때 시작점과 종점 사이의 최단 거리를 검색할 수 있다.

Level 3 위상 관계는 가장 고수준의 위상 관계로 선 성분 사이의 교차를 허용하지 않으며, 네트워크 분석뿐만 아니라 공간 분석을 지원하는 모델로 다각형 성분이 형성된다. 따라서, 객체들 사이의 포함선 관계 분석, 객체들 사이의 연결성 분석 등이 가능하다.

표 3-1. Winged-edge 위상 관계의 종류

위상 관계 종류	특 성
Level 0	No Topology (Spaghetti Model)
Level 1	Node and Edge Topology (Network Representation)
Level 2	Level 1 + Planar Graph (Network Representation)
Level 3	Level 2 + Face Topology (Spatial Analysis)

다. 알고리즘

Winged-edge 위상 관계 형성 알고리즘을 구현하기 위해서는 미리 노드 정보와 에지 정보를 만들어야하고 이들 정보는 위상 관계가 필요 없는 경우에도 계속 유지되므로 저장 장치의 낭비를 가져올 수 있다.

Winged-edge 위상 관계 구현 알고리즘은 다음과 같은 규칙이 있다.

1. 체인에서 시작한다.
2. 왼쪽 에지와 오른쪽 에지를 선택하였는가를 구분하기 위하여 왼쪽 에지를 선택하면 에지 앞에 "-"부호를 붙인다.

3. 시작된 체인과 마지막 체인을 비교하여 이 둘이 동일하면 면 성분 검색을 중단하고 면 성분에 대한 정보를 입력한다.
4. 면 성분이 존재하지 않으면 위상 관계 형성 과정을 끝낸다.

본 절에서는 Winged-edge 위상 관계 구현 알고리즘을 이용하여 간단한 예제를 제시하였다. 예제 벡터 자료는 [그림 3-31]에 나타내었으며 그 처리 과정은 다음과 같이 단계별로 나타내었다.

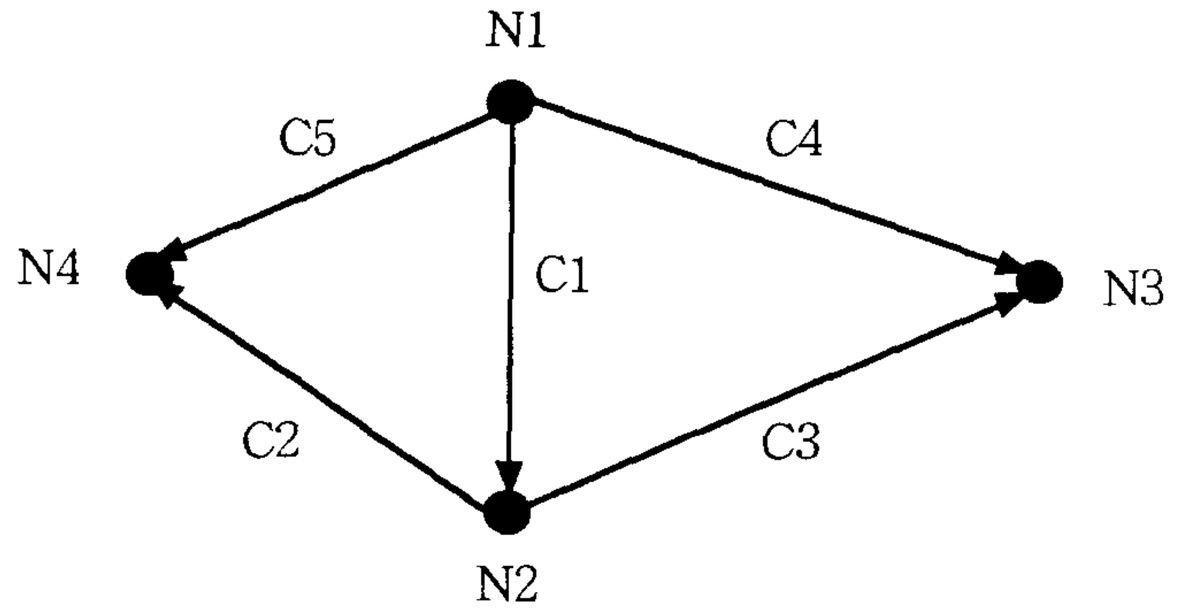


그림 3-31. 예제 벡터 자료

제 1 단계 :

노드 정보와 에지 정보를 포함하고 있는 에지 테이블을 만든다.

표 3-2. 에지 테이블

	Start Node	End Node	Left Edge	Right Edge	Left Face	Right Face
C1	N1	N2	C4	C2		
C2	N2	N4	C3	C5		
C3	N2	N3	C1	C4		
C4	N1	N3	C5	C3		
C5	N1	N4	C1	C2		

제 2 단계 :

면 정보가 없는 체인들 중에서 시작 체인을 선택한다.

본 예제에서는 C1 선택.

제 3 단계 :

노드 정보와 에지 정보를 이용하여 면 성분이 형성되는 체인들의 집합을 검색한다.

C1 -> N2 -> C2 -> N4 -> C5 -> N1 -> -C1

제 4 단계 :

검색된 체인의 집합인 { C1, C2, C5, -C1 }을 이용하여 면 성분을 입력한다.

표 3-3. 면 성분 입력 결과

	Start Node	End Node	Left Edge	Right Edge	Left Face	Right Face
C1	N1	N2	C4	C2		F1
C2	N2	N4	C3	C5		F1
C3	N2	N3	C1	C4		
C4	N1	N3	C5	C3		
C5	N1	N4	C1	C2	F1	

모든 체인의 왼쪽 면 성분과 오른쪽 면 성분이 설정될 때까지 위의 제 2 단계부터 제 4 단계까지를 반복 수행한다. 위상 관계 형성 과정이 끝난 결과는 [표 3-4]과 [그림 3-32]에 나타내었다. [표 3-4]의 Ext 표시는 외부를 표현한 것이다.

표 3-4. 위상 관계 형성 결과 테이블

	Start Node	End Node	Left Edge	Right Edge	Left Face	Right Face
C1	N1	N2	C4	C2	F2	F1
C2	N2	N4	C3	C5	Ext	F1
C3	N2	N3	C1	C4	Ext	F2
C4	N1	N3	C5	C3	F2	Ext
C5	N1	N4	C1	C2	F1	Ext

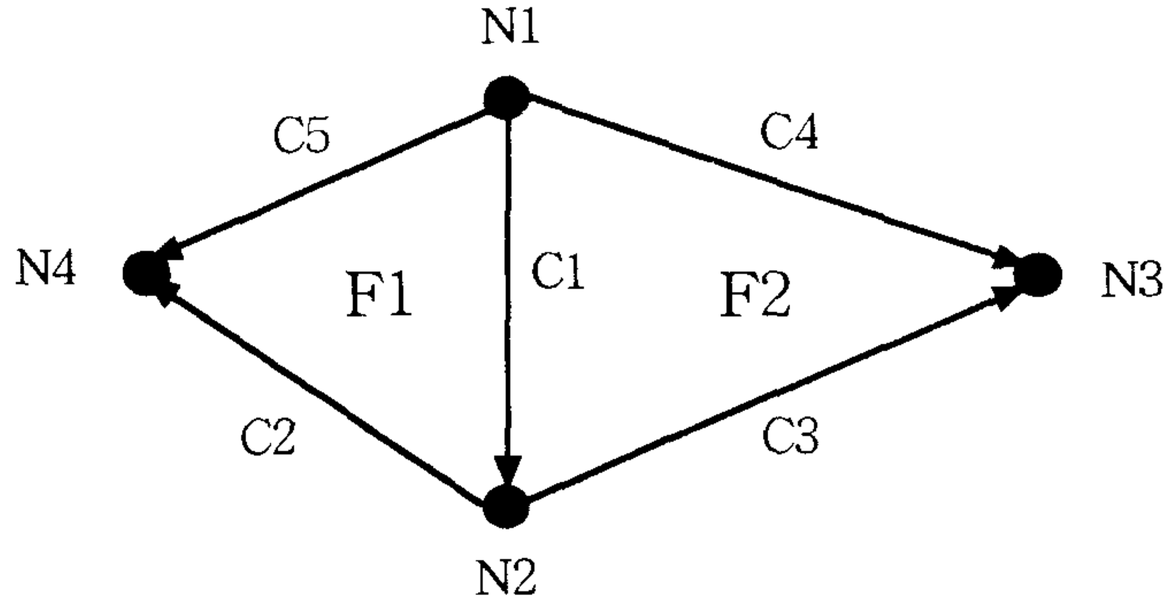


그림 3-32. 위상 관계 형성 결과

제 4 장 네트워크 GIS 자료를 위한 공간 데이터베이스 설계 및 관리 기능 개발

제 1 절 3-Tier 공간 데이터베이스의 설계 및 구현

1. 3-Tier 공간 데이터베이스

본 시스템은 3-Tier 데이터 베이스 시스템을 구성하고 있다. 사용자 인터페이스를 제공하고 응용 프로그램이 수행되는 자바 클라이언트 시스템, JDBC를 이용하여 오라클 데이터베이스를 접속하는 서버 시스템, 그리고 클라이언트 시스템과 서버 시스템을 독립성을 가지고 연결을 설정하여 시스템을 완성하는 클라이언트-서버 브로커 시스템이 있다. 이들 각 시스템들은 모두 독립적으로 구성되어 있으므로 한 부분만을 따로이 구성할 수 있는 장점이 있다. 다시 말해서, 현재는 네스케잎 네비게이터에서 수행되는 자바 애플릿 클라이언트 시스템은 네스케잎 없이 수행될 수 있는 어플리케이션 프로그램으로 쉽게 대치할 수 있다. 또한 특정한 어플리케이션을 위하여 클라이언트 시스템을 따로이 쉽게 구성할 수 있는 장점이 있다. 브로커 시스템 역시 독립적이므로 쉽게 변화 시킬 수 있다. 현재 브로커 시스템은 오라클 데이터베이스와의 연결을 설정해 주고 있으며 하나의 브로커 시스템은 멀티 스레드를 이용하여 최대 다섯 개의 클라이언트가 접속을 설정할 수 있다. 그러나 이러한 브로커 시스템을 수정함으로써 보다 많은 클라이언트와의 접속을 수행할 수도 있으며 오라클 데이터베이스가 아닌 이기종 데이터베이스와의 접속도 쉽게 구현할 수 있다. 끝으로 서버 시스템은 JDBC와 표준 질의어(SQL)를 이용하여 오라클 데이터베이스와의 연결을 구현하였으므로 거의 수정을 하지 않고 인포믹스, 사이베이스등 다른 데이터베이스와의 연결을

구현할 수 있는 장점이 있다. [그림 4-1]은 클라이언트, 브로커, 서버 시스템간의 연관 및 간략한 구성을 보여준다.

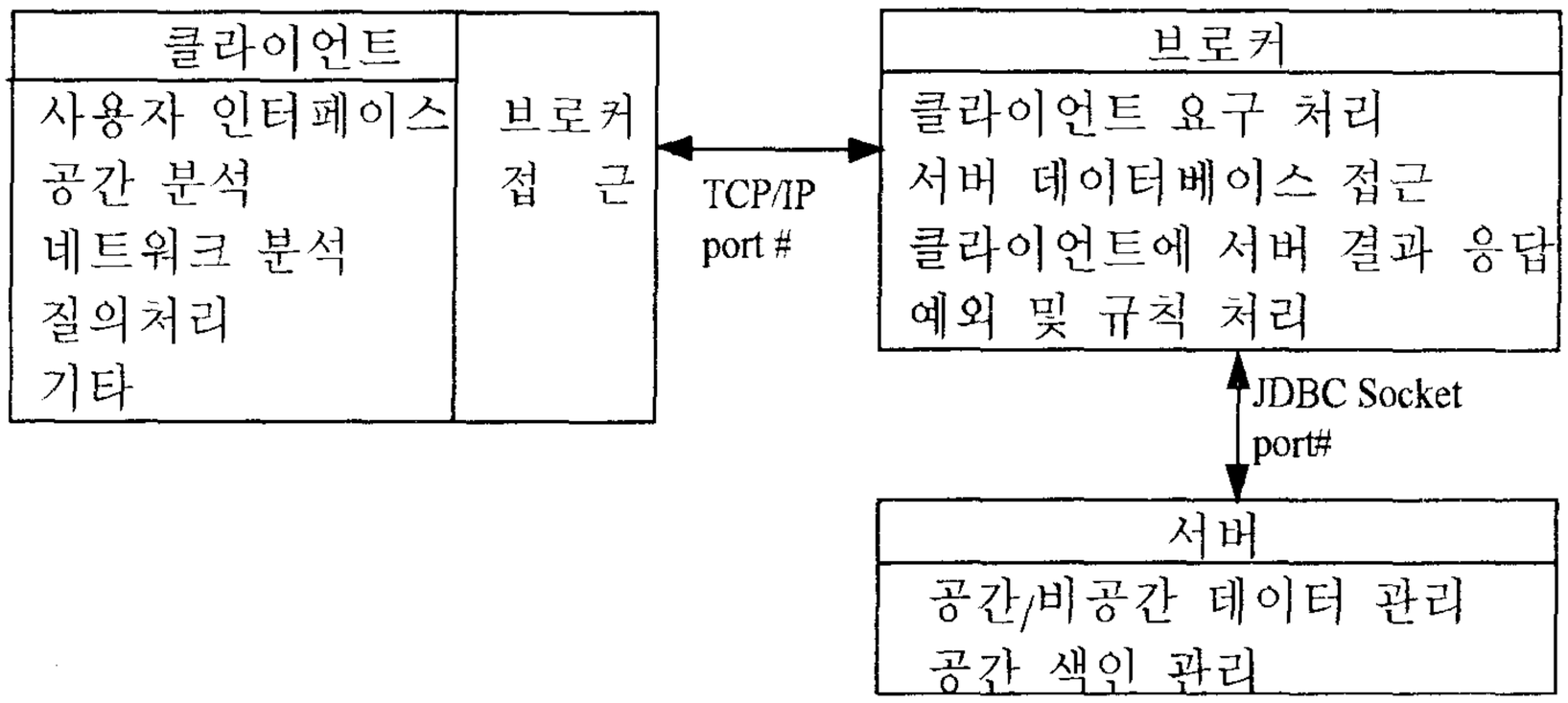


그림 4-1. 클라이언트, 브로커, 서버 시스템간의 간략한 구성도

가. 클라이언트 시스템

현재 클라이언트 시스템은 자바 애플릿으로 구현되었으며, 사용자 인터페이스 프로그램, 어플리케이션 프로그램, 그리고 브로커 시스템과의 접속 프로그램으로 구성되어 있다.

사용자 인터페이스 프로그램에서 구현된 기능은 확대, 축소, 다중 레이어 기능, 영역 선택 기능, 연속된 맵 기능, 레이블링 기능등이 있으며, 다음 장에서 자세히 알아 볼 것이다. 어플리케이션 프로그램으로는 질의 처리 기능, 통계 처리 기능, 공간 분석 기능들이 있으며 다음 장에서 자세히 알아볼 것이다.

그림 4-2. 클라이언트 시스템

나. 클라이언트-서버 브로커 시스템

클라이언트와 서버의 연계를 위한 브로커 시스템은 자바 어플리케이션 프로그램으로 구성되어 있다. 그리고 클라이언트와는 TCP/IP의 소켓으로 연결되었으며, 서버와는 JDBC(Java Database Connector)로 연결되어 있으며 실제로 JDBC역시 내부적으로는 TCP/IP를 구현하고 있다. 이러한 브로커 시스템은 독립적으로 서버 시스템에서 데몬(Daemon) 프로그램으로 구동되어 있어야만 한다. 현재 시스템에서는 데이터베이스가 존재하는 서버 시스템내에서 브로커 시스템도 같이 구동되고 있지만 서버 시스템이 아닌 제 3의 시스템에서도 구동될 수

있다. 브로커 시스템이 하는 주요 기능은 다음과 같이 요약될 수 있다.

1) 클라이언트 시스템 접속 기능

브로커는 TCP/IP의 소켓을 이용하여 클라이언트 시스템과 접속을 수행한다. 이를 위해 클라이언트측에는 브로커 접근 클래스가 브로커측에는 클라이언트 접근 클래스가 존재한다. 다음 [그림 4-3]은 클라이언트에서 브로커에게 접근하기 위해서 제공되는 클래스 집합들이다.

```
//---------------------- 브로커 접근 클래스 메소드 리스트 -----------------------

//  public BrokerAccess (String server)
//  public void close ()
//  public int getResponse ()
//  public void sendCommand (int command)
//  public void sendCommand (int command, String arg1)
//  public void sendCommand (int command, String arg1, int arg2, int arg3, int arg4, arg5)
//  public void sendCommand (int command, String arg1, String arg2)
//  public void sendCommand (int command, int arg1, int arg2)
//  public void sendCommand (int command, String arg1, String arg2, int arg3, int arg4)
//  public void sendCommand (int command, NodeFeature arg)
//  public void sendCommand (int command, ChainFeature arg)
//  public void sendCommand (int command, PolygonFeature arg)
//  public int FindFeatureType(String fName)
//  public NodeFeatureList GetNodeFeatureListWithMBR(String fName, int minx,
//                                        int miny, int maxx, int maxy, int searchType)
//  public ChainFeatureList GetChainFeatureListWithMBR(String fName, int minx,
//                                        int miny, int maxx, int maxy, int searchType)
//  public PolygonFeatureList GetPolygonFeatureListWithMBR(String fName, int minx,
//                                        int miny, int maxx, int maxy, int searchType)
//  public boolean AddNodeFeature(NodeFeature nodeF)
//  public boolean AddChainFeature(ChainFeature chainF)
//  public boolean AddPolygonFeature(PolygonFeature polyF)
```

```
//  public NodeFeatureList GetNodeFeatureListWithCondition(String fName, String condition)
//  public ChainFeatureList GetChainFeatureListWithCondition(String fName, String condition)
//  public PolygonFeatureList GetPolygonFeatureListWithCondition(String fName, String
//                                                                condition)
//  public NodeFeatureList MouseSelectNodeFeature(int x, int y)
//  public ChainFeatureList MouseSelectChainFeature(int x, int y)
//  public PolygonFeatureList MouseSelectPolygonFeature(int x, int y)
//  public double GetNodeStatistics(String fName, String what, int minx, int miny,
//                                                              int maxx, int maxy)
//  public double GetChainStatistics(String fName, String what, int minx, int miny,
//                                                              int maxx, int maxy)
//  public double GetPolygonStatistics(String fName, String wha, int minx, int miny,
//                                                              int maxx, int maxy)
//  public NodeFeatureList GetNodeListWithMBR(String fName, int minx, int miny,
//                                                  int maxx, int maxy, int searchType)
//  public ChainFeatureList GetChainListWithMBR(String fName, int minx, int miny,
//                                                  int maxx, int maxy, int searchType)
//  public PolygonFeatureList GetPolygonListWithMBR(String fName, int minx, int miny,
//                                                  int maxx, int maxy, int searchType)
//  public NodeFeatureList ReceiveNodeFeatureList(int response)
//  public ChainFeatureList ReceiveChainFeatureList(int response)
//  public PolygonFeatureList ReceivePolygonFeatureList(int response)
//  public NodeFeatureList ReceiveNodeList(int response)
//  public ChainFeatureList ReceiveChainList(int response)
//  public PolygonFeatureList ReceivePolygonList(int response)
//  public GetAllFdt()

//-------------------------------------------------------------------------
```

그림 4-3. 클라이언트-브로커 접속 메소드 리스트

가) BrokerAccess 메소드

BrokeAccess 클래스의 생성자로서 입력으로 주어지는 서버 주소에 있는 브로커에 대한 연결을 설정한다.

나) close 메소드

클라이언트와 브로커 사이에 소켓 연결을 종료한다. 클라이언트 시스템의 종료시에 자동적으로 수행된다.

다) getResponse 메소드

브로커로부터 수행된 메소드의 결과 상태에 대한 값을 받기 위해 사용된다. '-1'이 값이 수행되면 반환되는 결과가 없음을 의미한다.

라) sendCommand 메소드

브로커에서 수행되는 메소드를 위한 명령 코드와 그 메소드에 필요한 인자값들을 전송하는 역할을 수행한다. 인자값들에 따라서 여러 개의 메소드가 존재한다.

마) FindFeatureType 메소드

피쳐 이름이 입력으로 주어졌을 때 그 피쳐에 대한 공간 타입(노드, 체인, 폴리곤)을 반환한다.

바) GetNodeFeatureListWithMBR, GetChainFeatureListWithMBR,

GetPolygonFeatureListWithMBR 메소드

피쳐 이름과 최소 경계 사각형 그리고 검색 유형(포함, 피포함, 일치, 교차)에 따른 피쳐 집합을 검색하여 반환한다.

사) AddNodeFeature, AddChainFeature, AddPolygonFeature 메소드

주어진 피쳐를 데이터베이스내의 해당되는 피쳐 테이블에 삽입하고, 그 결과를 반환한다. 실패하면 '-1'을 반환한다.

아) GetNodeFeatureListWithCondition,
GetChainFeatureListWithCondition,
GetPolygonFeatureListWithCondition 메소드

피쳐 이름과 질의어의 조건에 따른 피쳐 집합을 검색하여 반환한다.

자) MouseSelectNodeFeature, MouseSelectChainFeature,
MouseSelectPolygonFeature 메소드

마우스로 선택되 노드, 체인 또는 폴리곤 피쳐를 반환한다. 내부적으로 마우스로 선택된 점을 중심으로 임의의 최소 경계 사각형을 만들고, 이를 이용하여 공간 검색 수행 후에 결과를 반환한다.

차) GetNodeStatistics, GetChainStatistics, GetPolygonStatistics 메소드

입력으로 들어오는 피쳐 이름과 최소 경계 사각형을 이용하여 최소 경계 사각형 내의 피쳐들을 검색한 후에 평균, 분산, 합계, 최대, 최소 값등의 통계 값들을 계산하여 반환한다.

카) GetNodeListWithMBR, GetChainListWithMBR,
GetPolygonListWithMBR 메소드

피쳐이름과 최소 경계 사각형 그리고 검색 유형을 입력으로 하여 입력 조건을 만족하는 공간 데이터들을 반환한다. 바)의 경우는 공간 데이터와 비공간 데이터 모두를 반환하는 메소드인데 비하여, 이 메소드는 공간 데이터만을 반환하는 것이다.

타) ReceiveNodeFeatureList, ReceiveChainFeatureList,
ReceivePolygonFeatureList 메소드

브로커로부터 전송되는 피쳐 집합들을 관리한다.

파) ReceiveNodeList, ReceiveChainList, ReceivePolygonList 메소드

브로커로부터 전송되는 공간 데이터 집합들을 관리한다.

하) GetAllFdt 메소드

공간 데이터베이스 내에 저장되는 피쳐 정의 테이블의 내용을 읽어와서 클라이언트 시스템내에 상주시킨다.

2) 서버 접속 기능

브로커는 클라이언트와의 접속외에 서버와의 접속을 위해 JDBC를 이용하여 오라클 데이터베이스에 접속을 하고 있다. JDBC를 이용한 공간 데이터베이스와의 접속은 추후에 자세히 설명될 것이다.

3) 멀티 프로세스 기능

클라이언트들의 요청에 따라서 브로커는 멀티 프로세싱을 가능하게 해준다. 다시 말해서 브로커는 새로운 클라이언트가 추가로 데이터베이스에 접속을 요청할 때 브로커는 먼저 새로운 스레드(Thread)를 생성시키고 이렇게 새로이 생성된 스레드에 하나의 클라이언트를 할당하여 데이터베이스에 접속을 시키는 다중 처리 기능을 수행할 수 있다.[그림 4-4]

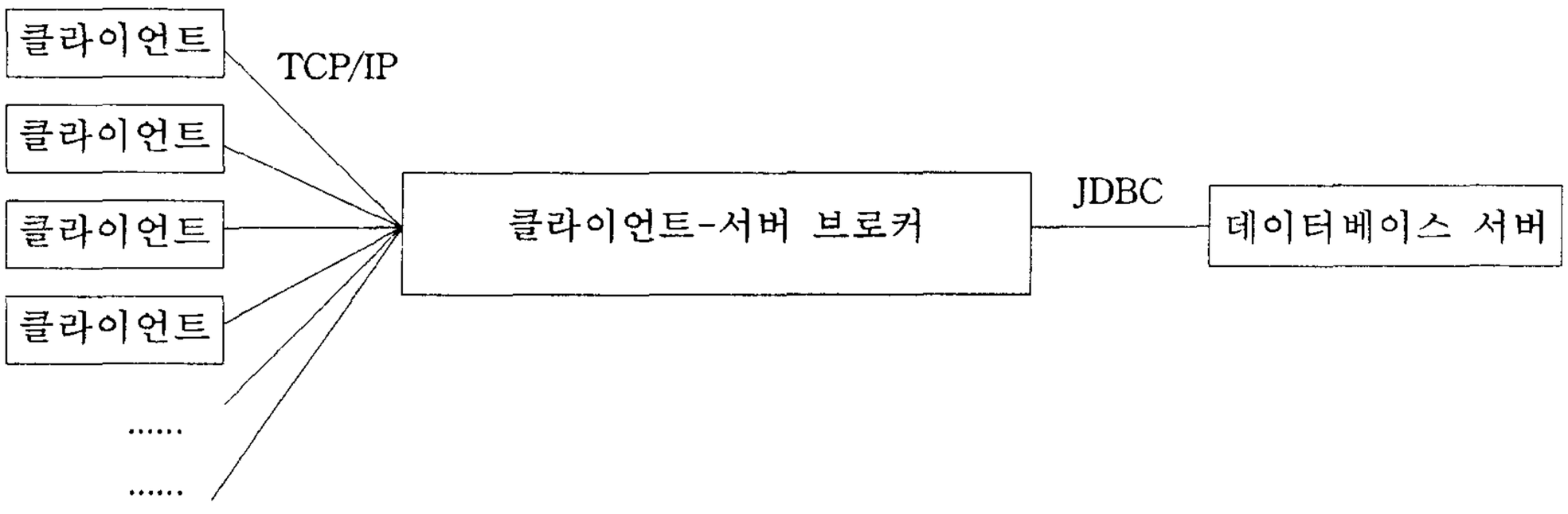


그림 4-4. 클라이언트-서버 브로커의 멀티프로세스 기능

클라이언트에 대하여 서버의 새로운 스레드를 생성시키는 코드는 다음 [그림 4-5]와 같다.

```
public class DBProtocolHandler implements Runnable {
        int             listenPort = 5500;          // TCP/IP 포트 #
        ServerSocket    myServer = null;
        RulesThread     checkIt;                    // 스레드
        Socket          answerSocket;
        Database        myDB;                       // 서버 데이터베이스

        // 브로커 초기화 메소드
        public void init(String dbServer, String projectName) {
                // 데이터베이스와의 연결 (입력 : 데이터베이스 서버, 프로젝트 이름)
                try {
                        myDB = new Database (dbServer, projectName);
                } catch (Exception e) {
                        System.out.println("데이터베이스 접속 에러 " +dbServer);
                }

                // 데이터베이스와의 연결, 포트 5500이 사용중일 때 다른 포트 사용
                try {
                        myServer = new ServerSocket(listenPort, 5);
                } catch (IOException e) {
                        System.out.println(e.toString());
                        listenPort = alternatePort++;
                        System.out.println("대용 포트 접속 에러 " +alternatePort);
                }
        }
```

```
// 새로운 스레드 생성 메소드
public void run() {
        // 클라이언트로부터 연결이 설정될때까지 대기상태
        while (true) {
                try {
                        answerSocket = myServer.accept();   // 연결 설정
                } catch (IOException e) {
                        return;
                }
                // 새로운 스레드를 생성하여 클라이언트에게 할당하고 다시 대기 상
                // 태로 돌아간다.
                System.out.println("클라이언트로부터 연결 설정");
                // 새로운 스레드 생성
                checkIt = new RulesThread(answerSocket, this);
                Thread.yield();
        }       // end of while
}
```

그림 4-5. 멀티 프로세스 생성을 위한 코드

본 시스템에서는 서버의 성능을 고려하여 현재는 5개의 클라이언트까지 데이터베이스에 접속이 가능하도록 설계되어 있다.

4) 기타 기능

브로커는 클라이언트, 서버 접속 기능, 멀티프로세스 기능외에도 공간 분석시에 필요한 공간 색인을 이용한 여과(Filtering) 단계 수행 및 에러 처리등을 수행하고 있으며, 차후에 보안 문제도 쉽게 처리할 수 있다.

다. 서버 시스템

3 Tier 공간 데이터베이스 시스템에서 마지막 시스템은 서버 시스템으로 클라이언트-서버 브로커 시스템과 JDBC로 연결되어 있다. 서버의 다음과 같은 기능들을 수행한다.

1) 공간 테이블 관리

노드, 체인, 폴리곤 테이블에 대한 입.출력 및 삭제, 갱신 작업등의 관리를 수행한다.

2) 비공간 테이블 관리

피쳐의 유형별로 비공간 테이블로 구성하고, 각 비공간 테이블에 대한 입.출력 및 삭제, 갱신 작업등의 관리를 수행한다.

3) 피쳐 정의 테이블 관리

새로운 피쳐 생성시 또는 피쳐 삭제시에 필요한 피쳐 정의 테이블 관련 작업을 수행한다.

4) 색인 정보 테이블 관리

노드, 체인, 폴리곤 데이터에 대한 입력 및 삭제, 그리고 갱신 작업이 일어났을 때 각 공간 데이터 타입에 해당되는 공간 색인 테이블을 수정하게 되는

데, 서버 시스템이 이러한 관리를 수행한다.

5) 테이블간 상호 연관 관계 관리

공간 테이블과 비공간 테이블간의 연관관계, 공간 테이블과 색인 정보 테이블간의 연관관게, 그리고 비공간 테이블과 피쳐 정의 테이블간의 연관관계를 처리하는 역할을 수행한다.

위와 같은 기능들을 수행하기 위해서 서버 시스템은 웹 서버와 데이터베이스 드라이버의 두 서브 모듈로 구성되어 있다. 웹 서버는 클라이언트로부터 서버에 접속을 허용하고 접속된 클라이언트에서 응용 프로그램을 실행시키기 위해서 필요한 자바 클래스 파일들을 전송하는 역할을 한다. 데이터베이스 드라이버는 웹 서버가 자바 응용 클래스를 모두 전송한 다음, 클라이언트가 응용 프로그램을 수행하면서 데이터를 필요로 할 때 데이터를 전송하는 창구로서의 역할을 한다. 결론적으로 서버는 자바 응용 프로그램과 데이터 모두를 보유하고 있으며, 클라이언트에 전송하기 위한 창구로는 웹 서버와 데이터베이스 드라이버 두 가지 시스템을 유지하고 있다.

라. 클라이언트, 서버, 브로커간의 연결

지금까지 클라이언트, 브로커, 그리고 서버 시스템의 각각에 대하여 알아 보았는데, 각 시스템들은 다음 [그림 4-6]과 같이 구성되어진다.

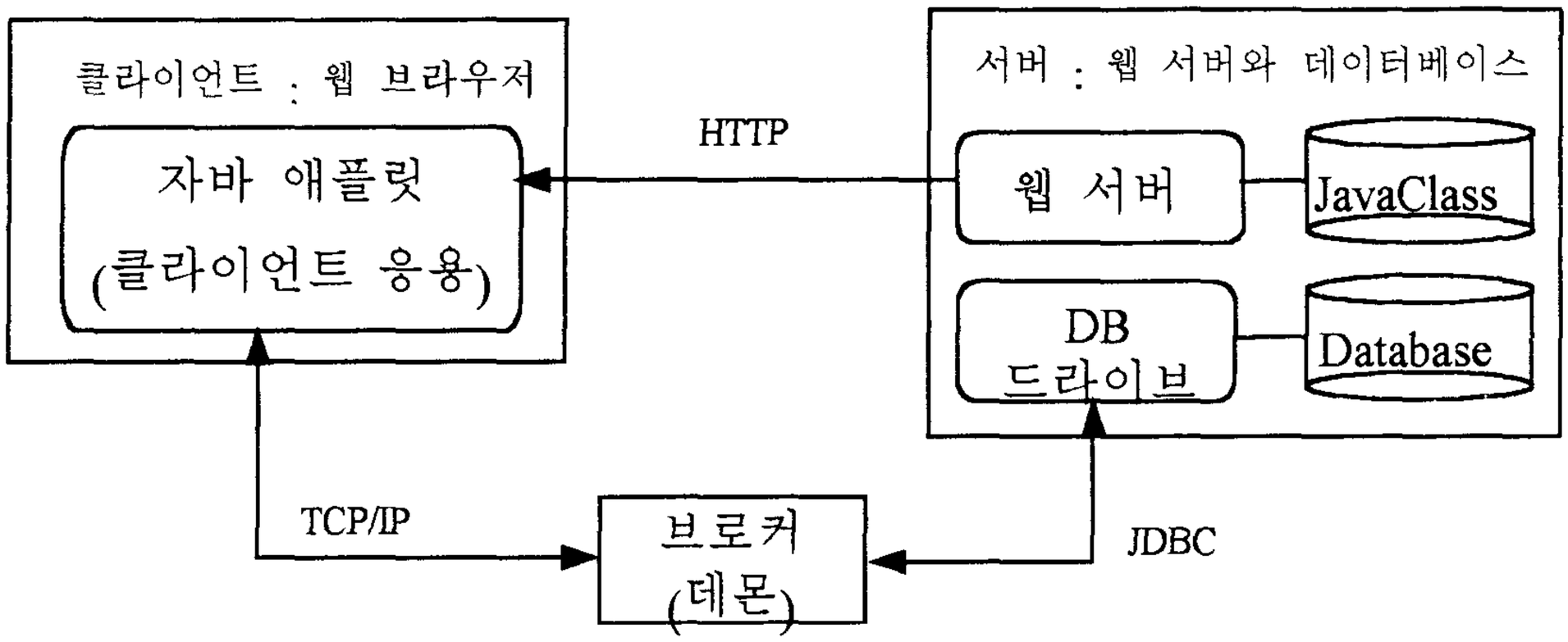


그림 4-6. 클라이언트, 브로커, 서버 시스템간의 연관 관계

클라이언트, 서버, 그리고 브로커 시스템의 구성은 [그림 4-6]과 같이 이루어져 있으며 이들 시스템간의 동작 과정은 [그림 4-7]에 설명되어 있다.

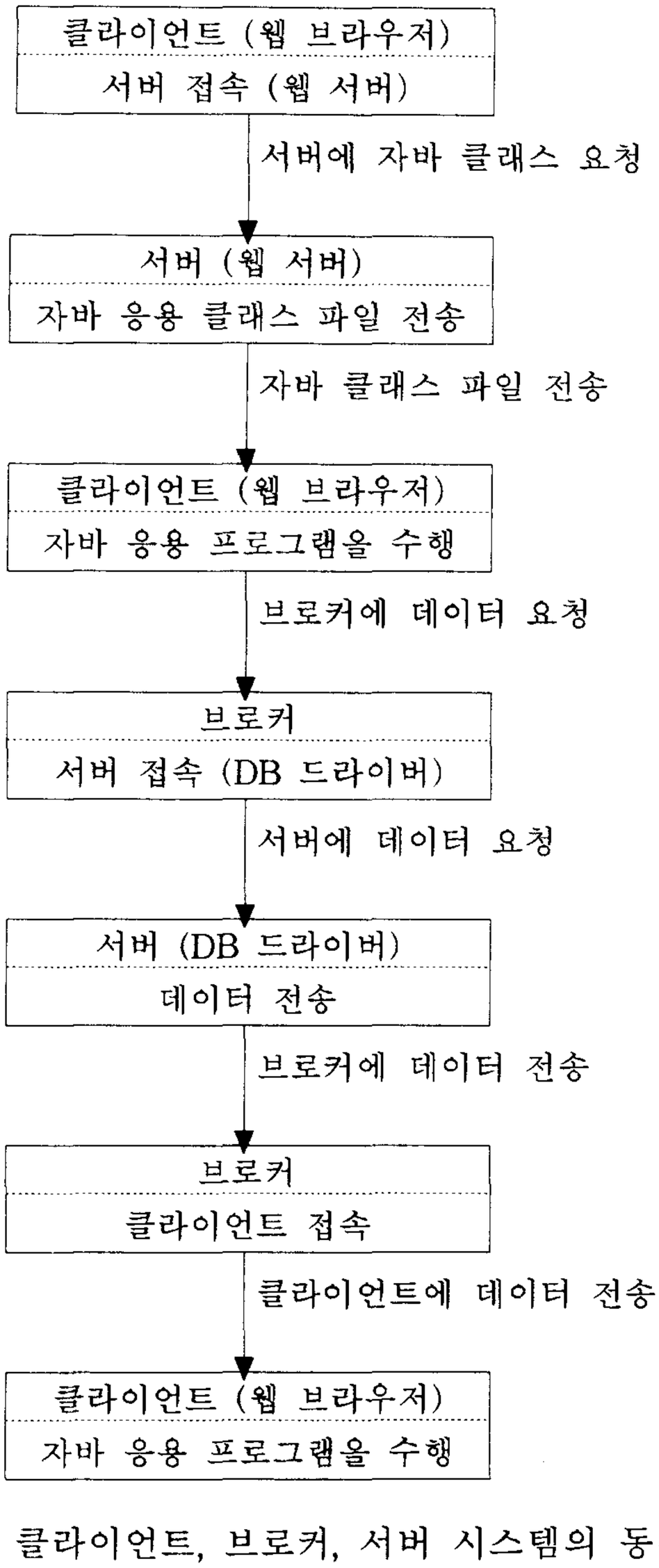


그림 4-7. 클라이언트, 브로커, 서버 시스템의 동작과정

위의 [그림 4-7]과 같이 자바 응용 프로그램을 위해서는 클라이언트와 서버가 직접적으로 접속을 하고 있으며, 응용 프로그램에서 필요로 하는 데이터

의 전송을 위해서는 클라이언트가 브로커와 접속을 하고 브로커가 서버와 접속을 하고 있다. 다시 말하면 자바 프로그램들은 웹 서버의 HTTP 프로토콜을 이용하여 전송되지만, 데이터들은 데몬 프로그램으로 구성되어 있는 브로커의 TCP/IP를 이용하여 데이터를 전송하고 있다는 것이다. 이렇게 TCP/IP를 이용하여 데이터를 전송하게 되면 HTTP 프로토콜을 이용하는 것보다 다음과 같은 특징이 있다.

- TCP/IP는 연결 위주(Connection Oriented) 프로토콜이므로 연결(Connectless) 프로토콜인 HTTP에 비해 데이터 전송때마다 연결을 새로이 설정할 필요가 없는 장점이 있다.

- TCP/IP 프로토콜은 연결이 설정되고 나면 전용선이 확보되므로 HTTP 프로토콜보다 빠른 시간내에 데이터를 전송할 수 있는 장점이 있다.

- 웹 서버에서 제공하는 프로토콜이 아닌 다른 프로토콜을 이용하여 데이터를 전송하므로 이를 위한 데몬 프로그램 작성이 필요한 단점이 있다.

2. JDBC를 이용한 데이터베이스 연결

본 시스템은 오라클 데이터베이스와의 연결을 위해서 JDBC를 이용하였다. JDBC와 더불어 오라클 데이터베이스의 sqlnet을 구동시키기 위해서 openlink의 데이터베이스 구동기(Database Driver)를 이용하였다. 이러한 JDBC와 openlink는 표준 SQL을 사용하므로 오라클 데이터베이스가 아닌 다른 데이터베

내용누락

내용누락

름과 암호가 필요한데, 위의 코드에서 보면 모두 "mskim"으로 입력되어 있다. 그리고 데이터베이스를 종료하고자 할 때는 연결이 이미 종료되었는지를 조사한다음에 연결이 종료되지 않았을 경우에만 종료를 시키도록 구성되어 있다.

나. JDBC를 이용한 데이터 입력

본 시스템에서 데이터의 입력은 공간 데이터, 비공간 데이터 그리고 색인 데이터별로 각각의 입력 메소드를 가지고 있다. 각 메소드는 모두 JDBC를 이용하므로 유사하게 구성되어진다. 다음 [그림 4-9]는 일반적인 입력 메소드에 대한 의사 코드이다.

```
// 데이터 입력
// 반환 값 : 입력 결과
// 입력 : 데이터베이스 연결 변수(conn), 전체 데이터 집합 이름(projectName)
public int DBInsert(Connection conn, String projectName) throws Exception {
        Statement       stmt = null;        // 질의문을 위한 변수
        ResultSet       result = null;      // 질의 결과
        int             ret = -1;
        String          tableName = projectName + "테이블 이름";

        // SQL문을 위한 변수를 생성한다.
        stmt = conn.createStatement();

        // 입력하고자 하는 테이블이 존재하는지 조사한다.
        result = stmt.executeQuery("select count(object_name) from user_objects
                                    where object_name = '" +tableName+ "'");
        if (result.next()) {
                // 테이블이 없으면, 입력을 하지 않고 수행을 종료한다.
                // 테이블이 존재하면, 데이터 입력을 수행한다.
        }
```

```
// 데이터 테이블 내의 가장 큰 ID값을 얻어온다.
maxId = GetMaxId(conn, projectName);

// 데이터 입력
ret = stmt.executeUpdate("insert into " +tableName+ " values
                         (" +value1+ "," +value2+ "," +value3");
// SQL문 변수를 해제한다.
stmt.close();

return ret;
```

그림 4-9. 데이터 입력 코드

위의 [그림 4-9]에서 보듯이 데이터를 입력하기 위해서 취하는 과정은 첫째 데이터가 입력되는 테이블이 존재하는지를 미리 조사한다. 둘째 테이블이 존재하면 그 테이블을 이용하여 새로이 입력될 데이터의 ID값을 새로이 설정하여 준다. 테이블 조사와 ID값이 설정되면 마지막으로 새로운 데이터가 테이블에 입력되는 형식을 취하고 있다. JDBC에 제공하는 메소드들로는 "createStatement()", "executeQuery()", "executeUpdate()", 그리고 "close()"등의 메소들이 각각의 사용법은 다음과 같다.

- createStatement() : SQL을 수행하기 위한 statement를 생성한다.
- close() : 생성된 statement를 해제시킨다.
- executeQuery() : 결과 값을 반환하는 SQL문을 수행한다. 데이터 추출의 경우에 사용된다.
- executeUpdate() : 결과 값을 반환하지 않는 SQL문을 수행한다. 데이터 입력, 갱신, 삭제의 경우에 사용된다.

다. JDBC를 이용한 데이터 삭제

데이터의 삭제 메소드 역시 공간, 비공간, 색인 데이터별로 구현되어 있으며, 일반적인 삭제 메소드는 [그림 4-10]과 같다.

```
// 데이터 삭제
// 반환 값 : 삭제 결과
// 입력 : 데이터 연결 변수(conn), 전체 데이터 집합이름(projectName),
        삭제 데이터 ID(id)
public int DBDelete(Connection conn, String projectName, int id) throws Exception {
        Statement       stmt = null;        // 질의문을 위한 변수
        ResultSet       result = null;      // 질의 결과

        int             ret = -1;
        String          tableName = projectName + "테이블 이름";

        // SQL Statement 생성
        stmt = conn.createStatement();

        // 삭제하고자 하는 테이블의 존재 조사
        result = stmt.executeQuery("select count(object_name) from user_objects
                                    where object_name = '" +tableName+ "'");
        if (result.next()) {
                if (result.getInt(1) < 1) {
                        // 테이블이 존재하지 않으면, 이 메소드를 종료한다.
                        // 테이블이 존재하면 삭제할 데이터를 삭제한다.
                }
        }
```

```
// 하나의 데이터 삭제
ret = stmt.executeUpdate("delete from " +tableName+ " where id = " + id);

// SQL Statement 해제
stmt.close();

return ret;
```

그림 4-10. 데이터 삭제 코드

데이터의 삭제 메소드 역시 데이터 입력 메소드에서 사용된 JDBC 라이브러리 메소드를 같이 사용하고 있으며, 단지 입력으로 들어가는 SQL문의 내용에 차이가 있을 뿐이다. 본 시스템에서는 데이터를 삭제하기 위해서 위의 그림과 같이 하나의 데이터를 삭제하는 메소드와 데이터 집합을 삭제하기 위한 메소드를 구현하고 있다.

라. JDBC를 이용한 데이터 갱신

기존의 데이터를 갱신하는 메소드는 데이터의 삭제 메소드와 거의 유사하게 구현되었으며, 역시 데이터 집합을 갱신하는 메소드가 구현되어 있다. [그림 4-11]은 데이터 갱신을 위해 구현된 코드의 일부분이다.

```
// 데이터 갱신
// 반환 값 : 갱신 결과
// 입력 : 데이터 연결 변수(conn), 전체 데이터 집합이름(projectName),
// 갱신 데이터 ID(id)
public int DBUpdate(Connection conn, String projectName, int id) throws Exception {
        Statement       stmt = null;        // 질의문을 위한 변수
        ResultSet       result = null;      // 질의 결과

        int             ret = -1;
        String          tableName = projectName + "테이블 이름";

        // SQL Statement 생성
        stmt = conn.createStatement();

        // 갱신 테이블 조사
        result = stmt.executeQuery("select count(object_name) from user_objects
                                    where object_name = '" +tableName+ "'");
        if (result.next()) {
        // 갱신 테이블이 존재하지 않으면 수행을 종료하고, 존재하면 갱신을 수행
        }

        // 데이터 갱신 수행
        ret = stmt.executeUpdate("update " +tableName+ " set value1=" +v1+ ",
                                 value2=" +v2+ ", value3=" +v3+ " where id=" +nid);

        // SQL Statement 해제
        stmt.close();

        return ret;
}
```

그림 4-11. 데이터 갱신 코드

위의 그림을 보면 데이터의 갱신 역시 "executeUpdate()" 메소드를 이용하여 구현되었음을 알 수 있으며, 그 외의 부분은 데이터의 삭제와 큰 차이점이 없음을 알 수 있다.

마. JDBC를 이용한 데이터 추출

데이터의 추출은 앞에서 설명한 입력, 삭제 그리고 갱신과는 다른 면이 있다. 이는 앞서와 마찬가지로 데이터의 추출이 일반적으로 데이터의 ID를 가지고 데이터를 추출하는 경우도 있지만, 공간 색인을 이용한 다음 데이터 테이블의 rowid를 이용하여 데이터를 추출하는 경우도 있기 때문이다.

먼저, 일반적으로 데이터의 ID를 가지고 데이터를 추출하는 예는 [그림 4-12]와 같이 구현되어져 있다.

```
if (result.next()) {
        // 테이블이 존재하지 않으면 수행을 종료하고, 테이블이 존재하면
        // 추출을 계속해서 수행한다.
}
// 하나의 데이터 추출
result=stmt.executeQuery("select * from " +tableName+ " where id ="+ id);

// 추출된 데이터를 읽어 들인다.
if(result.next()) {
        // Get value1 (int : NUMBER(16))
        Data.setValue1(result.getInt(1));
        // Get value2 (int : VARCHAR2(20))
        Data.setValue2(result.getString(2));
        // Get value3 (int : NUMBER(16, 4))
        Data.setValue3(result.getDouble(3));
}
else {
        System.out.println("추출된 데이터 없음.");
        ret = -1;
}

        // SQL statement 해제
        stmt.close();

        return ret;
}
```

```
// 데이터 추출
// 반환 값 : 추출 결과
// 입력 : 연결 변수(conn), 전체 데이터집합 이름(projectName), 데이터 ID (id)
public int DBSelect(Connection conn, String projectName, int id) throws Exception {
        Statement       stmt = null;        // 질의문 변수
        ResultSet       result = null;      // 질의 결과
        int             ret = -1;
        String          tableName = projectName + "테이블 이름";

        // SQL Statemnet 생성
        stmt = conn.createStatement();

        // 테이블이 존재하는지 조사
        result = stmt.executeQuery("select count(object_name) from user_objects
                                    where object_name = '" +tableName+ "'");
```

그림 4-12. 데이터 추출 코드

데이터 추출은 JDBC의 "executeQuery()" 메소드를 이용하고 있으며, 데이터베이스의 테이블로부터 추출된 데이터 값을 얻기 위해서 "next()" 와 "get*Something*()" 메소드를 이용하고 있다. 이들 메소드의 사용법은 다음과 같다.

- next() : 데이터베이스의 테이블에서 데이터 집합이 추출되었을 때, 다음 행(row)을 나타내기 위해서 사용된다.
- get*Something*() : 추출된 데이터 집합에서 데이터의 타입에 따라 값을 읽어 올 때 사용되는 메소드이다. *Something*에 해당되는 데이터

타입으로는 Short, Integer, Long, Float, Double, String, Byte, Boolean등이 있다.

둘째로 공간 색인을 이용하여 데이터 테이블의 rowid를 추출하고, 이 rowid를 이용하여 데이터를 추출하는 예는 위의 [그림 4-11]에서 데이터 id 부분을 문자열의 rowid 타입으로 대치하면 되며, 그 이외의 부분은 동일하게 구현되어 있다.

3. DBMS 인터페이스

본 시스템은 데이터베이스를 효율적으로 관리하기 위하여 크게 공간 데이터 테이블, 비공간 데이터 테이블, 그리고 색인 테이블로 테이블을 분류하였다. 그리고 이러한 테이블들을 효율적으로 관리하기 위하여 많은 인터페이스 메소드들을 제공하고 있는데, 이러한 인터페이스 메소드들 역시 테이블별로 나뉘어서 구현되어 있다.

제공되는 인터페이스 메소드들은 주로 데이터의 입력, 삭제, 갱신, 추출, 검색등의 테이블 관리 및 테이블간의 연관 관계 관리를 위해 구현되었다. 그리고 이러한 메소들이 현재는 오라클 데이터베이스상에서 만들어졌으나, 다른 데이터베이스와의 연결을 위해서 새로운 인터페이스를 구현하는데 있어서 큰 수정없이 구현할 수 있는 장점을 가지고 있다. 각 테이블에 관한 인터페이스 메소드들에 대하여 자세히 살펴보면 다음과 같다.

가. 공간 데이터 테이블 인터페이스

공간 데이터 테이블로는 노드, 체인, 폴리곤 테이블이 있으며 각 테이블에 대하여 인터페이스 메소드들이 구현되어 있다. 각각의 테이블의 인터페이스

메소드에 대한 구체적인 설명은 아래와 같다.

1) 노드 테이블 인터페이스 메소드

노드 테이블 인터페이스 메소드 리스트는 [그림 4-13]와 같으며 자세한 설명은 다음과 같다.

```
//------------------ 노드 데이터 테이블 인터페이스 메소드 리스트 -------------------

//  public NodeDB()
//  public void NodeDBConnect(Connection conn, String server)
//  public void NodeBDisConnect(Connection conn)
//  public int NodeDBCreate(Connection conn, String projectName)
//  public int NodeDBDrop(Connection conn, String projectName)
//  public int NodeDBInsert(Connection conn, String projectName)
//  public int NodeListDBInsert(Connection conn, String projectName)
//  public int NodeDBSelect(Connection conn, String projectName, int nid)
//  public int NodeListDBSelect(Connection conn, String projectName, int[] nid)
//  public int NodeAllDBSelect(Connection conn, String projectName)
//  public int NodeDBDelete(Connection conn, String projectName, int nid)
//  public int NodeListDBDelete(Connection conn, String projectName, int[] nid)
//  public int NodeAllDBDelete(Connection conn, String projectName)
//  public int NodeDBUpdate(Connection conn, String projectName, int nid)
//  public int NodeListDBUpdate(Connection conn, String projectName, int[] nid)
//  public boolean NodeDBExist(Connection conn, String projectName)
//  public int NodeCreateIndex(Connection conn, String projectName, CSis nodeSis)
//  public int NodeInsertIndex(Connection conn, String projectName, int nodeId, CSis nodeSis)
//  public int NodeDBSelectWithIndex(Connection conn, String projectName, StringBuffer[]
//                                                                    rowid)
//  public void SetNode(SERINode node)
//  public void SetNodeList(SERINodeList list)
//  public SERINode GetNode()
//  public SERINodeList GetNodeList()
//  public int GetMaxNodeId(Connection conn, String projectName)
//-------------------------------------------------------------------------------
```

그림 4-13. 노드 테이블 인터페이스 메소드

가) NodeDB() 메소드

노드 데이터 테이블을 접근 하기 위해서 필요한 노드 데이터베이스 객체를 생성하는 생성자이다.

나) NodeDBConnect(), NodeDBDisConnect() 메소드

노드 데이터 테이블에 접근하기 위해서는 미리 데이터베이스에 연결이 설정되어 있어야 한다. 만약, 데이터베이스에 연결이 설정되어 있지 않다면 NodeDBConnect() 메소드를 이용하여 연결을 설정할 수 있으며, NodeDBDisConnect() 메소드를 이용하여 연결을 종료 시킬 수 있다. 연결 설정을 위해서는 연결 환경을 설정하기 위한 연결 변수(Connection)와 서버의 IP 주소를 입력으로 필요로 하며, 연결 해지를 위해서는 연결이 이루어져 있는 연결 변수(Connection)만 있으면 된다.

다) NodeDBCreate(), NodeDBDrop() 메소드

NodeDBCreate()는 데이터베이스를 구축하고자 할 때 데이터베이스내에 노드 데이터 테이블을 생성하기 위한 메소드이며, NodeDBDrop()는 노드 데이터 테이블을 새로이 구성하고자 할 때 테이블을 삭제시키기 위한 메소드이다. 노드 테이블의 생성 및 삭제를 위해서는 연결 변수(Connection)와 전체 데이터 집합 이름이 필요한데, 여기서 전체 데이터 집합 이름은 노드 테이블 이름의 일부가 된다. 예를 들어 전체 데이터 집합 이름이 “TAEJON”이면 노드 테이블의 이름은

"TAEJON" + "NODE" 로서 "TAEJONNODE"가 된다. 이 메소드들은 실패하면 '-1'을 반환한다.

라) NodeDBInsert(), NodeListDBInsert() 메소드

노드 테이블에 데이터를 입력시키기 위한 메소드들로서 전자는 하나의 노드 데이터를 후자는 노드 데이터 리스트를 입력하기 위한 메소드이다. 연결 변수와 전체 데이터 집합 이름이 입력 인자로서 필요하며, 실지로 입력되는 데이터는 NodeDB 클래스에 있다. 이 메소드들은 입력이 실패하면 '-1'을 반환한다.

마) NodeDBSelect(), NodeListDBSelect(), NodeAllDBSelect() 메소드

노드 데이터를 추출(Retrieve)하기 위해서 필요한 메소드 리스트들로서, NodeDBSelect()는 입력 인자로서 연결 변수와 전체 데이터 집합 이름외에 추출하고자 하는 노드 데이터 ID를 가지고 있다. 그리고 NodeListDBSelect()는 노드 데이터 ID 리스트를 가지고 원하는 노드 데이터 리스트를 추출하는 메소드이며, 끝으로 NodeAllDBSelect()는 특정 노드 ID가 주어지지 않고 모든 노드를 추출하는 메소드이다. 추출된 데이터들은 NodeDB 클래스내의 노드 데이터 집합에 저장되어진다. 이 메소드들은 추출이 실패하면 '-1'을 반환한다.

바) NodeDBDelete(), NodeListDBDelete(), NodeAllDBDelete 메소드

노드 테이블내에서 삭제하고자 원하는 데이터 또는 데이터 리스트를 삭제하는 메소드로서, NodeDBDelte()는 노드 ID를 NodeListDBDelte()는 노드 ID 리스트를 인자로 가지고 있다. 이 메소드들은 삭제가 실패하면 '-1'을 반환한다.

사) NodeDBUpdate(), NodeListDBUpdate() 메소드

노드 테이블내에서 원하는 데이터 또는 데이터 리스트를 갱신하는 기능을 수행하며, 새로이 갱신되는 데이터는 NodeDB 클래스내에 있는 데이터를 이용한다. 이 메소드들은 갱신이 실패하면 '-1'을 반환한다.

아) NodeDBExist() 메소드

노드 테이블내에 입력으로 주어지는 노드가 존재하는지를 조사한다. 입력으로 주어지는 메소드는 NodeDB 클래스내에 있으며, 노드 데이터가 존재하면 'true'를 존재하지 않으면 'false'를 반환한다.

자) NodeCreateIndex() 메소드

노드 테이블에 구축된 모든 데이터에 대하여 처음으로 공간 색인 정보를 구축하고 이렇게 구축된 색인 정보를 색인 정보 테이블에 입력하는 일을 수행한다. 입력 인자로는 연결 변수, 전체 데이터 집합 이름과 더불어 공간 색인을 위한 변수가 이용된다. 이 메소드는 색인 생성이 실패하면 '-1'을 반환한다.

차) NodeInsertIndex() 메소드

노드 테이블에 새로운 노드 데이터를 입력함으로써 기 구축된 색인 정보를 수정하여 색인 정보 테이블을 수정하여 재저장하는 일을 수행한다. 입력 인자로는 연결 변수, 전체 데이터 집합 이름, 공간 색인을 위한 변수와 새로이 입

력되는 노드 데이터의 ID가 사용된다. 이 메소드는 색인 정보에 새로운 데이터의 입력이 실패하면 '-1'이 반환된다.

카) NodeDBSelectWithIndex() 메소드

이 메소드는 공간 검색을 수행할 때 사용되는 메소드로서, 색인 정보 테이블의 노드 색인 정보를 이용하여 공간 색인을 수행하고 나면 그 결과로 노드 테이블에 대한 rowid 리스트를 반환한다. 그리고 이 반환된 rowid 리스트를 이용하여 노드 데이터를 추출하는 방법이 NodeDBSelectWithIndex() 메소드이다. 이 메소드의 입력으로는 연결 변수, 전체 데이터 집합의 이름, 그리고 노드 테이블에 대한 rowid 리스트가 사용되며, 데이터 추출이 실패되면 '-1'을 반환한다.

타) SetNode(), SetNodeList() 메소드

노드 데이터의 입력 및 갱신을 위해서 필요한 데이터 집합을 NodeDB 클래스내에 set시키기 위한 메소드들이다. 입력 인자로는 set 시킬 노드 데이터 또는 노드 데이터 집합이 사용된다.

파) GetNode(), GetNodeList() 메소드

노드 데이터 추출 및 색인을 이용한 노드 데이터 추출을 수행하였을 때, 추출된 데이터를 읽어오기 위한 메소드들이다. 노드 데이터 또는 노드 데이터 집합을 반환한다.

하) GetMaxNodeId() 메소드

노드 데이터를 입력할 때 노드 테이블의 가장 마지막 ID를 이용하여 새로운 ID를 생성하는데 있어서 이 메소드를 사용한다.

2) 체인 테이블 인터페이스 메소드

체인 테이블 인터페이스 메소드 리스트는 [그림 4-14]와 같다. 그리고 대부분의 메소드들의 기능들이 메소드 이름만 다를 뿐 노드 인터페이스 메소드 리스트와 같으며 한가지 차이점만 있다. 노드 테이블이 한 쌍의 좌표값을 가지는데 비해 체인 테이블은 여러 쌍의 좌표값들을 가지게 된다. 이를 데이터베이스 테이블내에 저장하기 위해서 본 시스템에서는 체인의 좌표값들을 저장하기 위한 포인트 테이블을 사용하고 있다. 그러므로 체인 테이블은 데이터의 입력, 삭제, 갱신, 그리고 추출 작업을 수행할 때 부가적으로 포인트 테이블에 대한 입력, 삭제, 갱신, 그리고 추출 작업을 동시에 수행하고 있다. 포인트 테이블에 대한 작업을 제외하고는 아래 그림의 메소드들은 노드 테이블에서 설명한 것과 같은 방식으로 수행된다.

```
//------------------- 체인 데이터 테이블 인터페이스 메소드 리스트 -------------------
//
//  public ChainDB()
//  public void ChainDBConnect(Connection conn, String server)
//  public void ChainBDisConnect(Connection conn)
//  public int ChainDBCreate(Connection conn, String projectName)
//  public int ChainDBDrop(Connection conn, String projectName)
//  public int ChainDBInsert(Connection conn, String projectName)
//  public int ChainListDBInsert(Connection conn, String projectName)
//  public int ChainDBSelect(Connection conn, String projectName, int cid)
//  public int ChainListDBSelect(Connection conn, String projectName, int[] cid)
//  public int ChainAllDBSelect(Connection conn, String projectName)
//  public int ChainDBDelete(Connection conn, String projectName, int cid)
//  public int ChainListDBDelete(Connection conn, String projectName, int[] cid)
//  public int ChainAllDBDelete(Connection conn, String projectName)
//  public int ChainDBUpdate(Connection conn, String projectName, int cid)
//  public int ChainListDBUpdate(Connection conn, String projectName, int[] cid)
//  public boolean ChainDBExist(Connection conn, String projectName)
//  public int ChainInsertIndex(Connection conn, String projectName, int chainId, CSis
//                                  chainSis)
//  public int ChainCreateIndex(Connection conn, String projectName, CSis chainSis)
//  public int ChainDBSelectWithIndex(Connection conn, String projectName, StringBuffer[]
//                                  rowid)
//  public void SetChain(SERIChain chain)
//  public void SetChainList(SERIChainList list)
//  public SERIChain GetChain()
//  public SERIChainList GetChainList()
//  public int GetMaxChainId(Connection conn, String projectName)
//-----------------------------------------------------------------------------
```

그림 4-14. 체인 테이블 인터페이스 메소드

가) ChainDB() 메소드

데이터베이스내의 체인 테이블을 접근하기 위해 필요한 체인 테이터베이스 객체를 생성한다.

나) ChainDBConnect(), ChainDBDisConnect() 메소드

ChainDBConnect 메소드는 입력으로 주어지는 서버의 주소를 가지고 데이터베이스와의 연결을 설정하며, ChainDBDisConnect 메소드는 시스템 종료시 서버 데이터베이스와의 연결을 해지시킨다.

다) ChainDBCreate(), ChainDBDrop() 메소드

ChainDBCreate 메소드는 체인 데이터를 위한 체인 테이블의 생성을 수행하며, ChainDBDrop 메소드는 체인 테이블의 삭제를 수행한다. 테이블의 이름은 노드 테이블 생성시와 마찬가지로 전체 데이터 집합 이름(TAEJON)에 "CHAIN"을 더하면 된다. 그리고 노드 테이블과는 달리 테이블의 생성시와 삭제시에 체인을 형성하는 포인트 테이블을 추가적으로 생성 또는 삭제한다.

라) ChainDBInsert(), ChainListDBInsert() 메소드

체인 데이터 또는 체인 데이터 리스트를 테이블에 입력한다. 입력되는 데이터는 ChainDB 클래스내에 정의되어 있으며, 입력이 실패하면 '-1'을 반환한다.

마) ChainDBSelect(), ChainListDBSelect(), ChainAllDBSelect() 메소드

순서대로 하나의 체인, 체인 집합, 모든 체인을 추출하기 위한 메소드로서, 체인 테이블과 포인터 테이블을 접근하여 데이터를 읽어온다. 추출된 체인 데이터들은 ChainDB 클래스내에 저장되며, 원하는 데이터의 추출이 실패하면 '-1'을 반환한다.

바) ChainDBDelete(), ChainListDBDelete(), ChainAllDBDelete() 메소드

입력으로 주어지는 체인의 ID 집합을 삭제하는 메소드로서 체인 테이블과 포인트 테이블의 데이터를 삭제한다.

사) ChainDBUpdate(), ChainListDBUpdate() 메소드

체인 테이블과 포인트 테이블의 내용을 갱신한다. 테이블이 존재하지 않거나 갱신하기 위한 데이터가 존재하지 않으면 '-1'을 반환한다.

아) ChainDBExist() 메소드

체인 테이블에 찾고자 하는 데이터가 있는지를 검색하여, 검색 결과에 따라서 true/false를 반환하다.

자) ChainCreateIndex(), ChainInsertIndex(), ChainDBSelectWithIndex() 메소드

체인 데이터에 대한 공간 색인 관련 메소드들로서 ChainCreateIndex 메소드는 체인 데이터에 대하여 처음으로 색인 정보를 구축하고자 할 때 사용되며,

ChainInsertIndex 메소드는 생성된 색인 정보에 새로운 체인을 삽입하고자 할 때 사용된다. 그리고 ChainDBSelectWithIndex 메소드는 색인 관리기에서 제공하는 공간 검색을 수행할 때 사용되는 메소드이다.

차) SetChain(), SetChainList() 메소드

체인 데이터의 입력 및 갱신을 위해서 필요한 데이터 집합을 ChainDB 클래스내에 set시키기 위한 메소드들이다.

카) GetChain(), GetChainList() 메소드

체인 데이터의 추출 메소드를 수행하였을 때, 추출된 데이터를 접근하기 위한 메소드들이다.

타) GetMaxChainId() 메소드

체인 테이블에 새로운 데이터를 입력할 때 새로운 ID를 생성시킨다.

3) 폴리곤 테이블 인터페이스 메소드

먼저, 폴리곤 테이블은 실제로 폴리곤 정보를 표현하기 위하여 링 테이블을 이용하고 있다. 폴리곤 테이블은 하나의 폴리곤 ID에 대하여 임의 개수의 링 테이블 ID를 가지는데, 이들 각 링들은 폴리곤의 형태를 가지고 있다. 그러므로 하나의 폴리곤 피쳐는 여러개의 폴리곤을 유지할 수 있는데, 이는 하나의 폴리곤 피쳐가 여러개의 폴리곤을 내부에 포함할 수 있음을 의미한다. 링 테이블은

폴리곤을 형성하기 위해서 체인 테이블의 ID 집합으로 이루어져 있다.

그러므로 이러한 폴리곤 테이블을 위한 인터페이스 메소드들은 앞서 설명한 체인 테이블 인터페이스 메소드를 내부적으로 사용하고 있으며, 링 테이블을 관리하기 위한 메소드들이 포함되어 있다. [그림 4-15]

```
//------------------ 폴리곤 데이터 테이블 인터페이스 메소드 리스트 ------------------
//
//  public PolygonDB()
//  public void PolygonDBConnect(Connection conn, String server)
//  public void PolygonBDisConnect(Connection conn)
//  public int PolygonDBCreate(Connection conn, String projectName)
//  public int PolygonDBDrop(Connection conn, String tableName)
//  public int PolygonDBInsert(Connection conn, String projectName)
//  public int PolygonListDBInsert(Connection conn, String projectName)
//  public int PolygonDBSelect(Connection conn, String projectName, int pid)
//  public int PolygonListDBSelect(Connection conn, String projectName, int[] pid)
//  public int PolygonAllDBSelect(Connection conn, String projectName)
//  public int PolygonDBDelete(Connection conn, String projectName, int pid)
//  public int PolygonListDBDelete(Connection conn, String projectName, int[] pid)
//  public int PolygonAllDBDelete(Connection conn, String projectName)
//  public int PolygonDBUpdate(Connection conn, String projectName, int pid)
//  public int PolygonListDBUpdate(Connection conn, String projectName, int[] pid)
//  public boolean PolygonDBExist(Connection conn, String projectName)
//  public int PolygonInsertIndex(Conneciton conn,String projectName,int polyId, CSis, polySis)
//  public int PolygonCreateIndex(Connection conn, String projectName, CSis polySis)
//  public int PolygonDBSelectWithIndex(Connection conn, String projectName, StringBuffer[]
//                                                                    rowid)
//  public void SetPolygonList(SERIPolygonList list)
//  public void SetPolygon(SERIPolygon poly)
//  public SERIPolygonList GetPolygonList()
//  publis SERIPolygon GetPolygon()
//  public int GetMaxPolygonId(Connection conn, String projectName)
//  public int GetMaxRingId(Connection conn, String projectName)
//-------------------------------------------------------------------------------
```

그림 4-15. 폴리곤 테이블 인터페이스 메소드

가) PolygonDB() 메소드

폴리곤 테이블을 접근하기 위한 폴리곤 데이터베이스 테이블 객체를 생성한다.

나) PolygonDBConnect(), PolygonDBDisConnect() 메소드

서버 데이터베이스와의 연결 및 해지를 위해 사용되는 메소드이다. 일반적으로 시스템의 시작 시에 미리 데이터베이스와 연결이 설정되고 시스템 종료 시 연결이 해지되도록 구성되어 있으므로 특별한 일이 없는 한 불리어지지 않는 메소드들이다.

다) PolygonDBCreate(), PolygonDBDrop() 메소드

폴리곤 테이블의 생성 및 삭제를 위한 메소드로서 테이블의 이름은 전체 데이터 집합 이름(TAEJON)에 "POLYGON"을 더하면 된다. 그리고 폴리곤 테이블은 링 테이블을 가지므로 폴리곤 테이블 생성 및 삭제 시 링 테이블도 같이 생성되고 삭제된다.

라) PolygonDBInsert(), PolygonListDBInsert() 메소드

폴리곤 데이터 또는 폴리곤 데이터 리스트를 입력하는 메소드로서 입력되는 데이터는 PolygonDB 클래스 내에 정의되어 있다.

마) PolygonDBSelect(), PolygonListDBSelect(), PolygonAllDBSelect() 메소드

폴리곤 데이터를 추출하기 위한 메소드들로서 추출된 폴리곤들은 PolygonDB 클래스 내에 저장된다.

바) PolygonDBDelete(), PolygonListDBDelete(), PolygonAllDBDelete() 메소드

폴리곤 데이터를 삭제하기 위한 메소드들로서 순서대로 하나의 데이터, 폴리곤 리스트, 전체 폴리곤 데이터를 삭제하기 위한 메소드이다. 그리고 폴리곤 테이블에서 삭제된 폴리곤 데이터에 따라서 링 테이블의 데이터 역시 삭제된다.

사) PolygonDBUpdate(), PolygonListDBUpdate() 메소드

폴리곤 데이터를 갱신하기 위한 메소드들로서 갱신을 위한 데이터는 PolygonDB 클래스 내에 저장된다. 그리고 갱신된 폴리곤 데이터에 해당되는 링 데이터도 같이 갱신된다.

아) PolygonDBExist() 메소드

폴리곤 테이블에 찾고자 하는 데이터가 있는지를 검색하여, 검색 결과에 따라서 true/false를 반환하다.

자) PolygonCreateIndex(), PolygonInsertIndex(),

PolygonDBSelectWithIndex() 메소드

폴리곤 테이블의 공간 색인과 관련된 메소드들로서, 공간 색인의 생성, 공간 색인의 수정, 공간 검색에 필요한 기능들을 수행한다.

차) SetPolygon(), SetPolygonList() 메소드

폴리곤 데이터의 입력 및 갱신을 위해 필요한 폴리곤 데이터들을 set 시키기 위한 메소드들이다.

카) GetPolygon(), GetPolygonList() 메소드

폴리곤 추출로 인하여 저장된 폴리곤 데이터 집합들을 접근하기 위한 메소드들이다.

타) GetMaxPolygonId(), GetMaxRingId() 메소드

새로운 폴리곤 데이터를 입력하고자 할 때 새로운 폴리곤 그리고 링 ID를 생성하는 메소드들이다.

나. 비공간 데이터 테이블 인터페이스

현재 시스템에서 비공간 데이터들은 노드 피쳐 테이블, 체인 피쳐 테이블, 그리고 폴리곤 피쳐 테이블에 유형별로 저장되며, 이러한 테이블들을 관리하기 위한 인터페이스 메소드 역시 각각 존재한다.

1) 노드 피쳐 테이블 인터페이스 메소드

노드 피쳐 테이블을 관리하기 위해서 NodeFeatureDB 클래스를 구현하였으며, 노드 피쳐 테이블에 관한 모든 인터페이스 메소드들은 이 클래스내에 정의되어 있다. 먼저 NodeFeatureDB 클래스를 살펴보면 [그림 4-16]과 같다.

```
//------------------------- NodeFeatureDB 클래스 정의 -------------------------
import openlink.sql.Connection;
import openlink.sql.ResultSet;
import openlink.sql.Statement;
import openlink.sql.PreparedStatement;
import openlink.sql.DriverManager;

public class NodeFeatureDB {
        // 공용 속성 정보
        private int             id;             // 객체 ID
        private String          className;      // 클래스(피쳐) 이름
        private String          name;           // 객체 이름
        private int             type;           // 노드 / 체인 / 폴리곤 유형
        // 기타 건물 속성 정보
        private int             exs;            // 분류
        private int             birth;          // 설립 년도
        private int             aoo;            // 방향
        private int             bfc;            // 건물 기능 분류
        private int             zv2;            // Z 값
        // 공공 건물 속성 정보
        //private int           exs;            // 분류
        //private int           birth;          // 설립 년도
        //private int           bfc;            // 건물 기능 분류
        //private int           zv2;            // Z 값
        private String          phoneNum;       // 전화번호
        private String          addr;           // 주소
```

```
// 주유소 속성 정보
//private int           exs;          // 분류
//private int           bfc;          // 건물 기능 분류
//private int           zv2;          // Z 값
//private String        phoneNum;     // 전화번호
//private String        addr;         // 주소
private   int           facility;     // 처리 용량
// 병원 속성 정보
//private int           exs;          // 분류
//private int           bfc;          // 건물 기능 분류
//private int           zv2;          // Z 값
//private String        phoneNum;     // 전화번호
//private String        addr;         // 주소
private int             numWard;      // 병동수
private int             treatment;    // 치료법

// 노드 피쳐 데이터 집합 (입력, 갱신, 추출시에 사용)
private NodeFeature         oneNodeFeature = null;     // 노드 피쳐 데이터
private NodeFeatureList     nodeFeatureList = null;    // 노드 피터 데이터 리스트
}
```

그림 4-16. 노드 피쳐 클래스 데이터 정의

위의 [그림 4-16]에서 보듯이 노드 피쳐 클래스의 데이터는 네 가지 유형의 비 공간 데이터들에 대한 정보와 데이터 입력, 출력시에 사용되는 노드 피쳐 정보를 저장하기 위한 데이터로서 구성되어 있다. 비 공간 데이터에 따라서 각각 피쳐 클래스를 생성하지 않고 노드 피쳐 클래스 내에 모든 비 공간 속성 정보를 구현함으로써 테이블 관리 및 개발에 있어서 용이한 면이 있다. 그러나 이러한 방법들은 메모리 관리면이나 확장성에서 문제가 된다. 그러므로 이러한 확장성 부분을 고려하여 새로이 비공간 테이블 관리기가 설계 되어 있다. 그리고 노드 피쳐 테이블 인터페이스 메소드들은 테이블의 속성값을 제외하고 공간 데

이터 테이블에서 사용되는 인터페이스 메소드들과 비교해서 제공되는 기능이 매우 비슷하다. [그림 4-17]은 노드 피쳐 테이블에 대한 인터페이스 메소드 리스트이다.

```
//---------------- 노드 피쳐 데이터 테이블 인터페이스 메소드 리스트 ----------------
//
//  public NodeFeatureDB()
//  public void NodeFeatureDBConnect(Connection conn, String server)
//  public void NodeFeatureBDisConnect(Connection conn)
//  public int NodeFeatureDBCreate(Connection conn, String projectName)
//  public int NodeFeatureDBDrop(Connection conn, String projectName)
//  public int NodeFeatureDBInsert(Connection conn, String projectName)
//  public int NodeFeatureListDBInsert(Connection conn, String projectName)
//  public int NodeFeatureDBSelect(Connection conn, String projectName, int nfid)
//  public int NodeFeatureListDBSelect(Connection conn, String projectName, int[] nfid)
//  public int NodeFeatureAllDBSelect(Connection conn, String projectName)
//   public int NodeFeatureDBSelectWithPrimitiveIds(Connection conn, String projectName, int
//                                                  nid[], String fName)
//  public int NodeFeatureDBDelete(Connection conn, String projectName, int nfid)
//  public int NodeFeatureListDBDelete(Connection conn, String projectName, int[] nfid)
//  public int NodeFeatureAllDBDelete(Connection conn, String projectName)
//   public int NodeFeatureDBDeleteWithPrimitiveIds(Connection conn, String projectName, int
//                                                  nid[])
//  public int NodeFeatureDBUpdate(Connection conn, String projectName, int nfid)
//  public int NodeFeatureListDBUpdate(Connection conn, String projectName, int[] nfid)
//  public boolean NodeFeatureDBExist(Connection conn, String projectName)
//  public void SetNodeFeature(NodeFeature nodeFeature)
//  public void SetNodeFeatureList(NodeFeatureList list)
//  public NodeFeature GetNodeFeature()
//  public NodeFeatureList GetNodeFeatureList()
//  public int GetMaxNodeFeatureId(Connection conn, String projectName)
//-----------------------------------------------------------------------------
```

그림 4-17. 노드 피쳐 테이블 인터페이스 메소드

가) NodeFeatureDB() 메소드

노드 피쳐 테이블을 관리하기 위한 노드 피쳐 데이터베이스 객체를 생성한다.

나) NodeFeatureDBConnect(), NodeFeatureDBDisConnect() 메소드

데이터베이스 서버와의 연결 및 해지를 수행한다. 일반적으로 이 두 메소드는 시스템 시작시와 종료시에 자동으로 수행되도록 구현되어 있다.

다) NodeFeatureDBCreate(), NodeFeatureDBDrop() 메소드

데이터베이스내에서 노드 피쳐 테이블을 생성하고 삭제하기 위한 메소드들로서, 수행이 실패하면 '-1'을 반환한다.

라) NodeFeatureDBInsert(), NodeFeatureListDBInsert() 메소드

NodeFeautre 클래스에 정의되어 있는 노드 피쳐 데이터 집합을 데이터베이스에 저장하는 역할을 수행한다.

마) NodeFeatureDBSelect(), NodeFeatureListDBSelect(), NodeFeatureAllDBSelect() 메소드

노드 피쳐 ID를 이용하여 조건을 만족하는 노드 피쳐 데이터 집합을 추출한다.

바) NodeFeatureDBSelectWithPrimitiveIds() 메소드

노드 ID와 피쳐 이름을 입력으로 받아서 노드 테이블과 조인 테이블을 이용하여 노드 피쳐 데이터 집합을 추출한다.

사) NodeFeatureDBDelete(), NodeFeatureListDBDelete(),
NodeFeatureAllDBDelete() 메소드

노드 피쳐 ID를 입력으로 조건을 만족하는 노드 피쳐 데이터를 삭제한다.

아) NodeFeatureDBDeleteWithPrimitiveIds() 메소드

노드 테이블과 조인 테이블을 이용하여 노드 피쳐 데이터를 검색하여 검색된 데이터를 삭제한다.

자) NodeFeatureDBUpdate(), NodeFeatureListDBUpdate() 메소드

입력으로 주어진 노드 피쳐 데이터에 대하여 갱신 작업을 수행한다.

차) NodeFeatureDBExist() 메소드

입력으로 주어진 노드 피쳐 데이터가 테이블에 존재하는지 조사하고 결과에 따라 true/false를 반환한다.

카) SetNodeFeautre(), SetNodeFeatureList(), GetNodeFeautre(), GetNodeFeatureList() 메소드

노드 피쳐 데이터의 입력, 삭제, 갱신, 추출에 사용되는 데이터들을 읽고 저장하기 위한 메소드들이다.

타) GetMaxNodeFeautreId() 메소드

새로운 노드 피쳐 데이터를 입력할 때 새로운 피쳐 ID를 생성한다.

2) 체인 피쳐 테이블 인터페이스 메소드

체인 피쳐 테이블을 관리하고 체인 유형의 비공간 정보를 관리하기 위한 ChainFeatureDB 클래스 정의는 [그림 4-18]와 같다.

```
//------------------------- ChainFeatureDB 클래스 정의 -------------------------
import openlink.sql.Connection;
import openlink.sql.ResultSet;
import openlink.sql.Statement;
import openlink.sql.PreparedStatement;
import openlink.sql.DriverManager;

public class ChainFeatureDB {
        // 공용 속성 정보
        private int             id;             // 객체 ID
        private String          className;      // 클래스(피쳐) 이름
        private String          name;           // 객체 이름
        private int             type;           // 노드 / 체인 / 폴리곤 유형
```

```
// 도로 정보
private int          exs;          // 분류
private int          loc;          // 위치
private int          ltn;          // 도로 번호
private int          rst;          // 노면 유형
private int          rtt;          // 사용 정도
private int          med;          // 평균 사용
private int          use;          // 사용 용도

// 강 정보
//private int        exs;          // 분류
private double       length;       // 길이
private double       width;        // 폭

// 체인 피쳐 데이터 집합 (입력, 삭제, 추출, 갱신 시에 사용)
private ChainFeature      oneChainFeature = null;
private ChainFeatureList  chainFeatureList = null;
}
```

그림 4-18. 체인 피쳐 클래스 데이터 정의

본 시스템에서 체인 유형의 데이터로는 도로와 강의 두가지 정보가 정의 되어 있으며 위의 [그림 4-18]은 이를 위한 ChainFeatureDB 클래스를 보여주고 있다. 또한 JDBC를 이용하기 위해서 openlink관련 클래스들을 import하고 있으며, 테이블과 관련된 작업을 처리하기 위하여 oneChainFeature, chainFeatureList의 체인 피쳐 데이터들을 내장하고 있다. 이와같이 정의된 체인 피쳐 데이터들에 대한 인터페이스 메소드 리스트들은 다음 [그림 4-19]에 잘 나타나 있다.

```
//---------------- 체인 피쳐 데이터 테이블 인터페이스 메소드 리스트 ----------------
//
//  public ChainFeatureDB()
//  public void ChainFeatureDBConnect(Connection conn, String server)
//  public void ChainFeatureBDisConnect(Connection conn)
//  public int ChainFeatureDBCreate(Connection conn, String projectName)
//  public int ChainFeatureDBDrop(Connection conn, String projectName)
//  public int ChainFeatureDBInsert(Connection conn, String projectName)
//  public int ChainFeatureListDBInsert(Connection conn, String projectName)
//  public int ChainFeatureDBSelect(Connection conn, String projectName, int cfid)
//  public int ChainFeatureListDBSelect(Connection conn, String projectName, int[] cfid)
//  public int ChainFeatureAllDBSelect(Connection conn, String projectName)
//  public int ChainFeatureDBSelectWithPrimitiveIds(Connection conn, String projectName, int
//                                                  cid[])
//  public int ChainFeatureDBDelete(Connection conn, String projectName, int cfid)
//  public int ChainFeatureListDBDelete(Connection conn, String projectName, int[] cfid)
//  public int ChainFeatureAllDBDelete(Connection conn, String projectName)
//  public int ChainFeatureDBDeleteWithPrimitiveIds(Connection conn, String projectName, int
//                                                  cid[], String fName)
//  public int ChainFeatureDBUpdate(Connection conn, String projectName, int cfid)
//  public int ChainFeatureListDBUpdate(Connection conn, String projectName, int[] cfid)
//  public boolean ChainFeatureDBExist(Connection conn, String projectName)
//  public void SetChainFeature(ChainFeature ChainFeature)
//  public void SetChainFeatureList(ChainFeatureList list)
//  public ChainFeature GetChainFeature()
//  public ChainFeatureList GetChainFeatureList()
//  public int GetMaxChainFeatureId(Connection conn, String projectName)
//-----------------------------------------------------------------------------
```

그림 4-19. 체인 피쳐 테이블 인터페이스 메소드

가) ChainFeatureDB() 메소드

체인 피쳐 테이블을 관리하기 위한 체인 피쳐 데이터베이스 객체를 생성한다.

나) ChainFeatureDBConnect(), ChainFeatureDBDisConnect() 메소드

본 시스템의 시작시와 종료시에 자동적으로 수행되는 메소드로서 데이터베이스와의 연결 및 해지를 수행한다.

다) ChainFeatureDBCreate(), ChainFeatureDBDrop() 메소드

체인 피쳐 데이터를 저장하기 위한 체인 피쳐 테이블을 생성하고 삭제한다.

라) ChainFeatureDBInsert(), ChainFeatureListDBInsert() 메소드

체인 피쳐 데이터 또는 체인 피쳐 데이터 집합을 데이터베이스에 입력한다.

마) ChainFeatureDBSelect(), ChainFeatureListDBSelect(),
ChainFeatureAllDBSelect() 메소드

체인 피쳐 데이터를 추출하기 위한 메소드들로서 하나의 체인, 체인 집합, 모든 체인을 추출할 수 있다. 추출하기 위한 조건으로는 체인 ID가 입력으로 주어진다.

바) ChainFeatureDBSelectWithPrimitiveIds() 메소드

체인 테이블 ID와 피쳐 이름을 입력으로하여 체인 테이블과 체인-피쳐 조인 테이블을 이용하여 체인 피쳐 데이터들을 추출한다.

사) ChainFeatureDBDelete(), ChainFeatureListDBDelete(),
ChainFeatureAllDBDelete() 메소드

입력으로 주어지는 체인 피쳐 ID에 해당되는 체인 피쳐 데이터를 삭제한다.

아) ChainFeatureDBDeleteWithPrimitiveIds() 메소드

입력으로 주어지는 체인 ID들을 이용하여 체인 테이블과 체인-피쳐 조인 테이블을 참조하여 체인 피쳐 데이터를 검색하고 검색된 결과를 삭제한다.

자) ChainFeatureDBUpdate(), ChainFeatureListDBUpdate() 메소드

체인 피쳐 테이블에 대하여 갱신 작업을 수행한다.

차) ChainFeatureDBExist() 메소드

체인 피쳐 테이블내에 입력으로 주어지는 체인 피쳐 데이터가 존재하는지 조사하고 존재하면 true를 존재하지 않으면 false를 반환한다.

카) SetChainFeautre(), SetChainFeatureList(), GetChainFeautre(), GetChainFeatureList() 메소드

체인 피쳐 데이터의 입력, 삭제, 갱신, 추출에 사용되는 데이터들을 읽고 저장하는 역할을 한다.

타) GetMaxChainFeautreId() 메소드

새로운 체인 피쳐 데이터를 입력하고자 할 때 이 새로운 데이터를 위한 새로운 피쳐 ID를 생성한다.

3) 폴리곤 피쳐 테이블 인터페이스 메소드

노드나 체인 피쳐 테이블과 마찬가지로 폴리곤 피쳐 테이블을 관리하기 위해서 PolygonFeatureDB 클래스가 정의되어 있으며, 이 클래스내에 폴리곤 피쳐 테이블을 관리하기 위한 인터페이스 메소드들이 구현되어 있다. [그림 4-20]은 PolygonFeatureDB 클래스의 데이터 정의를 보여준다.

```
//------------------------- PolygonFeatureDB 클래스 정의 -------------------------

import openlink.sql.Connection;
import openlink.sql.ResultSet;
import openlink.sql.Statement;
import openlink.sql.PreparedStatement;
import openlink.sql.DriverManager;

public class PolygonFeatureDB {
        // 공용 속성 정보
        private int             id;             // 폴리곤 객체 ID
        private String          className;      // 클래스(피쳐) 이름
        private String          name;           // 객체 이름
        private int             type;           // 노드 / 체인 / 폴리곤 유형

        // 학교 정보
        //private int           exs;            // 분류
        //private int           birth;          // 설립년도
        private int             degree;         // 학교 유형(초등/중/고/대학교)
        private int             numStu;         // 학생 수

        // 기타 건물(내부) 정보
        //private int           exs;            // 분류
        //private int           birth;          // 설립 년도
        private int             bfc;            // 건물 기능 분류
        private int             zv2;            // Z 값(고도값)
        private int             story;          // 건물 내력
        private int             pyoung;         // 면적 (평)

        // 연구소 정보
        //private int           exs;            // 분류
        //private int           birth;          // 설립 년도
        //private int           bfc;            // 건물 기능 분류
        //private int           zv2;            // Z 값(고도값)
```

```
    private int         numMan;         // 사람 수
    private int         product;        // 주요 산물
    private String      phoneNum;       // 전화 번호
    private String      faxNum;         // 팩스 번호
    private String      addr;           // 주소

    // 구 경계 정보
    private double      area;           // 면적
    private int         numHouse;       // 주거 수
    private int         pop;            // 인구

    // 동 경계 정보
    //private double    area;           // 면적
    //private int       numHouse;       // 주거 수
    //private int       pop;            // 인구

    // 폴리곤 피쳐 데이터 집합 (입력, 삭제, 추출, 갱신 시에 사용)
    private PolygonFeature      onePolygonFeature = null;
    private PolygonFeatureList  polygonFeatureList = null;

    // 아파트 단지 정보
    private int         exs;            // 분류
    private int         numBld;         // 건물 수
    private int         birth;          // 설립년도

    // 공원 정보
    //private int       exs;            // 분류
    //private int       numBld;         // 건물 수
    //private int       birth;          // 설립 년도
    private int         numDoumi;       // 직원 수

}
```

그림 4-20. 폴리곤 피쳐 클래스 데이터 정의

현재 본 시스템에서는 폴리곤 유형의 피쳐로 여섯가지의 비공간 정보를 정의하고 있다. 그리고 이외에 openlink 관련 클래스 import나 데이터의 입출력을 위해서 onePolygonFeature, 또는 polygonFeatureList를 사용하는 것은 노드나 체인 피쳐 클래스와 같다. 이러한 폴리곤 피쳐 클래스의 인터페이스 메소드는 다음 그림과 같다.

```
//--------------- 폴리곤 피쳐 데이터 테이블 인터페이스 메소드 리스트 ---------------
//
//  public PolygonFeatureDB()
//  public void PolygonFeatureDBConnect(Connection conn, String server)
//  public void PolygonFeatureBDisConnect(Connection conn)
//  public int PolygonFeatureDBCreate(Connection conn, String projectName)
//  public int PolygonFeatureDBDrop(Connection conn, String projectName)
//  public int PolygonFeatureDBInsert(Connection conn, String projectName)
//  public int PolygonFeatureListDBInsert(Connection conn, String projectName)
//  public int PolygonFeatureDBSelect(Connection conn, String projectName, int pfid)
//  public int PolygonFeatureListDBSelect(Connection conn, String projectName, int[] pfid)
//  public int PolygonFeatureAllDBSelect(Connection conn, String projectName)
//   public int PolygonFeatureDBSelectWithPrimitiveIds(Connection conn, String projectName,
//                                int pid[], String fName)
//  public int PolygonFeatureDBDelete(Connection conn, String projectName, int pfid)
//  public int PolygonFeatureListDBDelete(Connection conn, String projectName, int[] pfid)
//  public int PolygonFeatureAllDBDelete(Connection conn, String projectName)
//   public int PolygonFeatureDBDeleteWithPrimitiveIds(Connection conn, String projectName,
//                                int pid[])
//  public int PolygonFeatureDBUpdate(Connection conn, String projectName, int pfid)
//  public int PolygonFeatureListDBUpdate(Connection conn, String projectName, int[] pfid)
//  public boolean PolygonFeatureDBExist(Connection conn, String projectName)
//  public void SetPolygonFeature(PolygonFeature PolygonFeature)
//  public void SetPolygonFeatureList(PolygonFeatureList list)
//  public PolygonFeature GetPolygonFeature()
//  public PolygonFeatureList GetPolygonFeatureList()
//  public int GetMaxPolygonFeatureId(Connection conn, String projectName)
//-------------------------------------------------------------------------
```

그림 4-21. 폴리곤 피쳐 테이블 인터페이스 메소드

가) PolygonFeatureDB() 메소드

폴리곤 피쳐 테이블을 접근하기 위해서 필요한 폴리곤 피쳐 데이터베이스 객체를 생성한다.

나) PolygonFeatureDBConnect(), PolygonFeatureDBDisConnect() 메소드

노드, 체인 피쳐와 마찬가지로 데이터베이스에 연결 및 해지를 위해 실행된다.

다) PolygonFeatureDBCreate(), PolygonFeatureDBDrop() 메소드

폴리곤 피쳐 데이터 테이블을 생성하고 삭제한다.

라) PolygonFeatureDBInsert(), PolygonFeatureListDBInsert() 메소드

폴리곤 피쳐 데이터를 입력하기 위한 메소드들이다.

마) PolygonFeatureDBSelect(), PolygonFeatureListDBSelect(), PolygonFeatureAllDBSelect() 메소드

입력으로 주어지는 폴리곤 피쳐 ID 집합을 이용하여 폴리곤 피쳐 집합을 추출한다.

바) PolygonFeatureDBSelectWithPrimitiveIds() 메소드

공간 색인 시스템을 이용하여 폴리곤 피쳐를 추출하고자 할 때 사용되는 메소드로서, 폴리곤 테이블 ID와 피쳐 이름을 입력으로하여 폴리곤 테이블과 폴리곤-피쳐 조인 테이블을 이용하여 폴리곤 피쳐 데이터들을 추출한다.

사) PolygonFeatureDBDelete(), PolygonFeatureListDBDelete(),
PolygonFeatureAllDBDelete() 메소드

해당되는 폴리곤 피쳐 데이터를 삭제한다.

아) PolygonFeatureDBDeleteWithPrimitiveIds() 메소드

공간 색인 관리기에서 호출하는 메소드로서, 폴리곤 테이블과 폴리곤-피쳐 조인 테이블을 참조하여 검색된 폴리곤 피쳐 데이터를 삭제한다.

자) PolygonFeatureDBUpdate(), PolygonFeatureListDBUpdate() 메소드

폴리곤 피쳐 테이블에 대하여 갱신 작업을 수행한다.

차) PolygonFeatureDBExist() 메소드

폴리곤 피쳐 테이블내에 원하는 체인 피쳐 데이터가 존재하는지를 검색하고, 결과로는 Boolean값을 반환한다.

카) SetPolygonFeautre(), SetPolygonFeatureList(), GetPolygonFeautre(), GetPolygonFeatureList() 메소드

폴리곤 피쳐 데이터의 입력, 삭제, 갱신, 추출에 사용되는 데이터들을 읽고 저장하는 역할을 한다.

타) GetMaxPolygonFeautreId() 메소드

새로운 폴리곤 피쳐 데이터를 입력하고자 할 때 이 새로운 데이터를 위한 새로운 피쳐 ID를 생성한다.

다. 기타 테이블 인터페이스

공간 테이블과 비공간 테이블 이외에 데이터베이스내에 유지되고 있는 테이블은 색인 정보 테이블과 피쳐 정의 테이블이 있다. 색인 정보 테이블은 공간 색인 정보를 저장하고 있으며 피쳐 정의 테이블은 공간 피쳐들에 대한 부가 정보를 유지하고 있다. 이들 각 테이블에 대한 자세한 설명은 다음과 같다.

1) 색인 정보 테이블 인터페이스 (Spatial Indexing Table Interface)

본 시스템의 색인 관리기는 R*-tree의 공간 색인 정보를 저장하기 위해서 데이터베이스내에 색인 정보 테이블을 생성하고 관리하고 있다. 색인 정보 데이터베이스 클래스의 주요 역할은 데이터베이스내에서 색인 테이블의 생성 및 삭제, 색인 정보 저장, 색인 정보 추출, 색인 정보 삭제, 그리고 색인 정보 갱신

등의 역할을 수행한다. 이러한 색인 정보 데이터베이스 클래스는 다음 [그림 4-22]과 같다.

```
//------------------ 색인 정보 데이터베이스 인터페이스 메소드 리스트 ------------------
//
//  public CSisDB()
//  public void SISDBConnect(Connection conn, String server)
//  public void SISDBDisConnect(Connection conn)
//  public int SISDBCreate(Connection conn, String projectName)
//  public int SISDBDrop(Connection conn, String projectName)
//  public int SISDBSelect(Connection conn, String projectName, int tid)
//  public int SISDBInsert(Connection conn, String projectName)
//  public int SISDBDelete(Connection conn, String projectName, int tid)
//  public int SISDBUpdate(Connection conn, String projectName)
//  public boolean SISDBExist(Connection conn, String projectName, int tid)
//-------------------------------------------------------------------------
```

그림 4-22. 색인 정보 데이터베이스 클래스 정의

색인 정보 데이터베이스 클래스(CSisDB)는 데이터베이스내의 테이블 스키마와 같은 데이터들을 정의하고 있다. 공간 색인 ID, 공간 색인 이름, 공간 색인 크기, 그리고 공간 색인의 정보를 유지하고 있다. 공간 색인 클래스(CSis)로부터 상속을 받음으로써 공간 색인 클래스의 모든 메소드들을 이용할 수 있다. CSis에서 제공하는 메소드들은 1절의 'R*-tree 구현'에서 알아 보았듯이 색인의 생성, 삭제 및 검색작업등을 수행한다. 결론적으로 색인 정보 데이터베이스는 공간 색인 클래스의 메소드를 이용하여 공간 색인을 관리하고 또한 이러한 색인 정보를 데이터베이스와 연결하여 데이터베이스에 저장, 삭제, 갱신, 그리고 추출 등의 작업을 수행한다. [그림 4-23]는 이러한 색인 정보 데이터베이스 클래스의 인터페이스 메소드를 보여준다.

```
//------------------------------ SISDB 클래스 정의 ------------------------------

import openlink.sql.Connection;
import openlink.sql.ResultSet;
import openlink.sql.Statement;
import openlink.sql.DriverManager;

public class CSisDB extends CSis {
        int             tid;            // R*-tree ID
        String          tname;          // R*-tree 이름
        int             tsize;          // R*-tree 크기
        byte[]          tdata;          // R*-tree 색인 정보
}
```

그림 4-23. 색인 정보 데이터베이스 인터페이스 메소드

가) CSisDB() 메소드

데이터베이스내의 공간 색인 정보 테이블을 접근하기 위해서 필요한 색인 정보 데이터베이스 객체를 생성한다. 본 시스템에서는 노드, 체인, 그리고 폴리곤 유형에 따라서 공간 색인 정보가 저장되어 있으므로 세 개의 색인 정보 데이터베이스 객체가 생성되어 진다.

나) SISDBConnect(), SISDBDisConnect() 메소드

공간 색인 정보를 저장하고 추출하기 위해서는 데이터베이스와 미리 연결이 서정되어 있어야 된다. 이를 위하여 SISDBConnect() 메소드가 이용되고,

시스템을 종료할 때 연결을 해지해야 하는데 이를 위하여는 SISDBDisConnect() 메소드를 이용한다.

다) SISDBCreate(), SISDBDrop() 메소드

공간 색인 정보 테이블을 생성하고 삭제할 때 사용된다. 색인 정보 테이블의 생성은 공간 데이터가 데이터베이스에 구축될 때 동시에 생성되고 색인 정보가 입력된다.

라) SISDBSelect() 메소드

공간 색인 정보를 추출하기 위한 메소드로서 현재는 색인 ID를 입력으로 하여 원하는 색인 정보를 추출한다. 본 시스템에서는 현재 노드, 체인, 폴리곤별로 색인 정보가 저장되며 타입은 이진형태의 정보로서 구성되어 있다.

마) SISDBInsert() 메소드

새로이 생성되거나 갱신 되어진 색인 정보를 색인 정보 테이블에 입력하기 위한 메소드이다. 입력되는 값으로는 색인 ID, 색인 이름, 색인 정보의 크기, 그리고 색인 정보가 주어진다.

바) SISDBDelete() 메소드

공간 색인 ID를 입력으로 받아서 공간 색인 정보를 삭제한다.

사) SISDBUpdate() 메소드

선택되어진 공간 색인 정보를 갱신 시키는 역할을 수행한다.

아) SISDBExist() 메소드

공간 색인 정보가 색인 정보 테이블내에 존재하는지 검사하고 결과로는 true/false를 반환한다.

2) 피쳐 정의 테이블 인터페이스 (Feature Definition Table Interface)

피쳐 정의 테이블은 피쳐에 대한 요약 정보(피쳐 이름, 공간 유형, 디스플레이 레벨)를 유지하는 테이블로서 임의의 피쳐가 주어졌을 때 쉽게 피쳐에 대한 정보를 얻을 수 있는 장점이 있다. 예를 들어 피쳐 이름만 주어지고 모든 피쳐 정보를 추출하고자 할 경우 피쳐 정의 테이블의 정보를 이용하여 공간 데이터의 유형을 알 수가 있다. 이러한 피쳐 정의 테이블은 시스템이 동작을 시작할 때 클라이언트 시스템으로 캐슁되어 상주하게 된다. 이러한 피쳐 정의 테이블의 클래스 정의는 [그림 4-24]와 같다.

```
//---------------------------- FeatureDB 클래스 정의 ----------------------------

import openlink.sql.Connection;
import openlink.sql.Statement;
import openlink.sql.PreparedStatement;
import openlink.sql.ResultSet;
import openlink.sql.DriverManager;

public class FdtDB {
        private int         id;             // 피쳐 정의 테이블 ID
        private int         type;           // 공간 유형
        private String      fName;          // 피쳐 이름
        private int         vlevel;         // 디스플레이 레벨
        private double      fValue;         // Reserved

        private String      fNameList[];    // 피쳐 이름 리스트
        private FDT         oneFdt;         // 피쳐 정의
        private FDTList     fdtList;        // 피쳐 정의 리스트
}
```

그림 4-24. 피쳐 정의 테이블 클래스 정의

피쳐 정의 테이블 클래스는 실제로 피쳐 정의 테이블 ID, 공간 유형, 피쳐 이름, 그리고 디스플레이 레벨과 같은 정보들을 유지하고 있다. 또한 피쳐 정의 테이블에 피쳐 요약 정보를 저장하고 추출하기 위해 oneFdt와 fdtList 정보를 유지하고 있다. 이러한 피쳐 정의 테이블 클래스가 제공하는 메소드 리스트는 다음 [그림 4-25]와 같다.

```
//-------------------- 피쳐 정의 테이블 인터페이스 메소드 리스트 --------------------
//
//  public FdtDB()
//  public void FdtDBConnect(Connection conn, String server)
//  public void FdtDBDisConnect(Connection conn)
//  public int FdtDBCreate(Connection conn, String projectName)
//  public int FdtDBDrop(Connection conn, String projectName)
//  public int FdtDBInsert(Connection conn, String projectName)
//  public int FdtAllSelect(Connection conn, String projectName)
//  public int FdtTypeSelectWithFname(Connection conn, String projectName, String fName)
//  public int FdtVLevelSelectWithFname(Connection conn, String projectName, String fName)
//  public int FdtFvalueSelectWithFname(Connection conn, String projectName, String fName)
//  public int FdtAllFnameSelect(Connection conn, String projectName)
//  public int FdtDBDeleteWithFname(Connection conn, String projectName, String fName)
//  public boolean FdtDBExist(Connection conn, String projectName)
//  public void SetFdt(int id, int type, String fName, int vlevel, double fValue)
//  public int GetFdtId()
//  public int GetFdtType()
//  public String GetFdtFname()
//  public String[] GetFdtFnameList()
//  public int GetFdtVLevel()
//  public double GetFdtFvalue()
//  public int GetMaxFdtId(Connection conn, String projectName)
//-----------------------------------------------------------------------------
```

그림 4-25. 피쳐 정의 테이블 인터페이스 메소드

가) FdtDB() 메소드

피쳐 정의 테이블을 관리하기 위한 피쳐 정의 테이블 객체를 생성한다.

나) FdtDBConnect(), FdtDBDisConnect() 메소드

데이터베이스와의 연결 설정 및 해지를 수행한다. 시스템의 시작시와 종료시에 수행된다.

다) FdtDBCreate(), FdtDBDrop() 메소드

피쳐 정의 테이블을 생성하고 삭제하는 역할을 수행한다. 피쳐 정의 테이블은 데이터베이스가 구축될 때 동시에 생성된다.

라) FdtDBInsert() 메소드

피쳐 정의 테이블에 새로운 피쳐에 대한 요약 정보를 입력한다.

마) FdtAllSelect() 메소드

피쳐 정의 테이블에 들어 있는 모든 정보를 추출한다. 이는 시스템의 구동시에 수행되는데, 피쳐 요약 정보를 클라이언트 시스템에 상주시키기 위해서 모든 데이터를 읽어 온다.

바) FdtTypeSelectWithFName(), FdtVLevelSelectWithFName(), FdtFValueSelectWithFName() 메소드

피쳐 이름을 입력으로 하여 필요한 피쳐 정보를 추출한다. 예를 들어 공간 유형, 디스플레이 레벨등의 정보를 얻어온다.

사) FdtAllFNameSelect() 메소드

피쳐 정의 테이블에 정의되어 있는 모든 피쳐들의 이름을 추출한다. 이 메소드는 현 시스템내에 정의되어 있는 모든 피쳐들을 파악할 때 유용하게 사용된다.

아) FdtDBDeleteWithFName() 메소드

입력으로 주어진 피쳐 이름과 일치하는 피쳐 정의 테이블내의 데이터를 삭제한다.

자) FdtDBExist() 메소드

피쳐 정의 테이블을 검색하여 일치하는 데이터가 있는지 검색한다. true/false를 반환한다.

차) SetFdt(), GetFdtId(), GetFdtType(), GetFdtFname(), GetFdtFnameList(), GetFdtVLevel(), GetFdtFValue() 메소드

피쳐 정의 테이블의 데이터 집합을 저장하거나 읽는 일을 수행한다.

카) GetMaxFdtId() 메소드

새로운 피쳐 정보를 입력하고자 할 때 새로운 ID를 부여한다.

제 2 절 서버 데이터베이스의 구성

본 시스템에서 서버내의 공간 데이터베이스는 공간 데이터와 비공간 데이터의 효율적인 관리를 위하여 데이터베이스 관리기와 공간 색인 관리기의 두 관리기를 유지하고 있다. 데이터베이스 관리기는 공간 데이터, 비공간 데이터 그리고 색인 데이터에 대한 생성 및 관리를 수행한다. 색인 관리기는 다양한 공간 검색을 위한 R*-tree를 생성하고 관리하는 역할을 수행한다. 다음 [그림 4-26]은 서버 시스템내의 공간 데이터베이스 시스템의 구성을 보여준다.

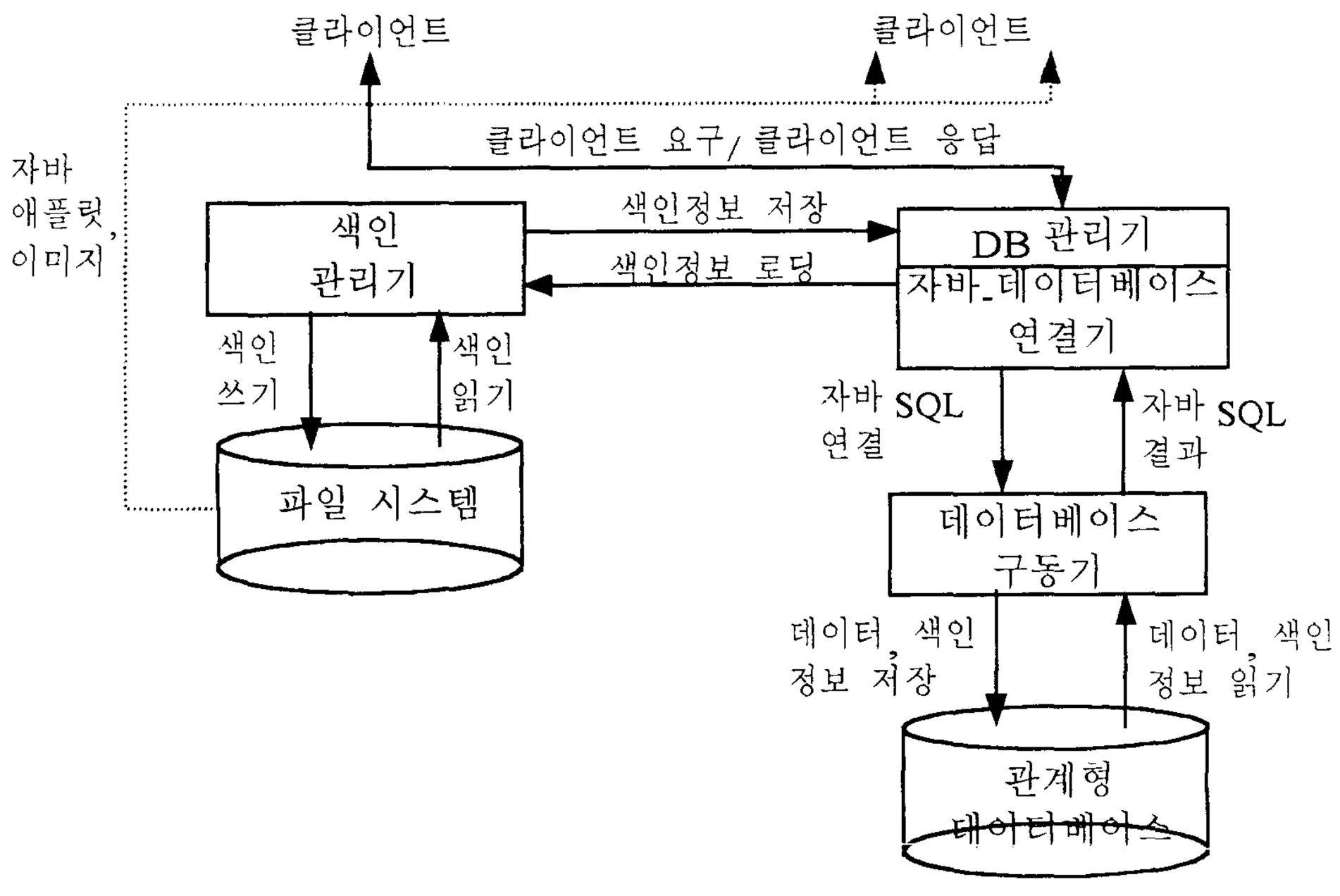


그림 4-26. 공간 데이터베이스 시스템의 구성

[그림 4-26]에서 보듯이 공간 데이터베이스 시스템은 주로 데이터베이스

관리기와 색인 관리기가 주 역할을 수행하고 있으며, 그 외에 데이터베이스와의 직접적인 접속을 위하여 데이터베이스 구동기를 필요로 한다. 현재 본 시스템에서는 오라클(Oracle) 7.3.3을 이용하고 있으며, 이를 위하여 오픈링크(Openlink)사의 JDBC-ODBC(Java Database Connector-Object Database Connector) 구동기를 사용하고 있다.

1. 데이터베이스 관리기

데이터베이스 관리기는 첫째로 데이터의 입력, 삭제, 갱신, 검색 작업을 수행할 뿐만 아니라 공간 분석을 수행하고자 할 때 필요한 모든 작업을 수행한다. 둘째로 데이터베이스 관리기는 공간 데이터 테이블, 비공간 데이터 테이블, 공간 테이블과 비공간 테이블을 연결시키는 조인 테이블, 그리고 공간 데이터에 대한 색인 정보 테이블들을 일관성을 유지하며 관리하는 역할을 수행한다.

가. 공간 데이터 테이블

먼저 공간 데이터 테이블의 관리에 대하여3 살펴보면 다음과 같다. 공간 데이터 테이블은 구체적으로 공간 테이블로는 노드, 체인, 링, 폴리곤의 테이블이 존재하는데 다음과 같은 정보를 유지 하고 있다[그림 4-27].

[노드 테이블]

노드 ID	X좌표	Y좌표	연결된 체인의 갯수

[체인 테이블]

체인 ID	시작노드 ID	끝노드 ID	체인 길이	최소 X	최소 Y	최대 X

최대 Y	왼쪽 폴리곤 ID	오른쪽 폴리곤 ID	점의 갯수	점 리스트

[링 테이블]

링 ID	체인의 갯수	체인 리스트

[폴리곤 테이블]

폴리곤 ID	면적 둘레	최소 X	최소 Y	최대 X	최대 Y	링 갯수	링 리스트

그림 4-27. 공간 데이터 테이블의 구조

공간 데이터 테이블에서 노드 테이블의 경우를 보면 연결된 체인의 개수를 가지는데, 이는 노드의 위상(Topology) 정보를 나타내기 위해서 유지하고 있다. 그리고 노드 테이블은 한 쌍의 X좌표와 Y좌표만을 유지하므로 (X좌표 , Y좌표 , X좌표, Y좌표)를 이용하여 최소 경계 사각형(MBR : Minimum Bounding Rectangle)의 정보를 유지하고 있다. 체인 테이블의 경우는 위상 정보를 표현하기 위하여 왼쪽 폴리곤 ID와 오른쪽 폴리곤 ID, 그리고 시작 노드 ID, 끝 노드 ID의 정보를 유지하고 있으며, 부가 정보로는 체인 길이와 최소 경계 사각형의 정보를 유지 하고 있다. 그리고 체인을 구성하는 (Xi, Yi)쌍의 좌표값 리스트를 위하여 체인내의 좌표 쌍의 개수 정보와 좌표 리스트에 대한 정보를 유지하고 있다. 링 테이블은 체인의 집합으로 다각형을 구성하는데 체인의 개수와 체인 리스트의 정보를 유지하고 있다. 폴리곤 테이블은 링의 집합으로 구성되며 하나의 폴리곤은 내부 다각형을 포함하여 다수 개의 다각형을 포함할 수 있도록

구성되어 있다. 이를 위하여 링 개수 정보와 링 리스트 정보를 유지하고 있다. 또한 면적, 둘레등의 부가 정보와 최소 경계 사각형의 정보를 유지하고 있다. 끝으로 각 테이블에서 유지하고 있는 최소 경계 사각형의 정보는 R*-tree의 공간 색인 정보 형성에 사용된다.

나. 비공간 데이터 테이블

비공간 데이터는 공간 데이터와 연결되는 속성 정보로서 이를 관리하기 위한 테이블로는 유형별로 나누어진 속성 테이블들이 있으며, 속성 테이블들에 대한 정의를 유지하는 속성 정의 테이블, 그리고 비공간 데이터와 공간 데이터와의 연결을 위한 속성-공간 조인 테이블이 있다[그림 4-28].

[피쳐(Feature) 정의 테이블]

피쳐 정의 테이블 ID	공간 타입	피쳐 테이블 이름	디스플레이 레벨

[피쳐 테이블 (노드, 체인, 폴리곤 별로 속성 저장)]

피쳐 테이블 ID	테이블에 따른 속성 정보들

[피쳐-공간 조인 테이블]

피쳐-공간 조인 테이블 ID	피쳐 테이블 ID	공간 테이블 ID

그림 4-28. 비공간 데이터 테이블의 구조

피쳐(Feature) 정의 테이블은 공간 타입, 비쳐 테이블 이름과 디스플레이 레벨로 구성되어 있으며 각 정보의 사용 용도는 다음과 같다. 공간 타입과 피쳐 테이블 이름은 주어진 피쳐의 공간 타입에 대한 정보와 피쳐 테이블에 대한 정보를 제공함으로써 공간 정보와의 연결을 손쉽게 할 수 있도록 도와 준다. 디스플레이 레벨은 현재의 화면에서 피쳐가 보일지 안보일지를 결정하는 기준으로

서 사용된다. 이러한 피쳐(Feature) 정의 테이블은 시스템이 시작되고 피쳐 데이터 집합들을 읽어들일 때 미리 클라이언트 시스템내에 상주하도록 구성되어 있으므로, 클라이언트는 빠른 시간내에 위의 정보들을 액세스 할 수 있다. 현재 시스템에서 피쳐(Feature) 테이블은 노드, 체인, 폴리곤에 대하여 각 하나씩 존재하며, 12개 피쳐들의 속성 정보들을 유지하고 있다. [그림 4-29]는 공간 타입별로 묶인 12개 피쳐들을 보여준다.

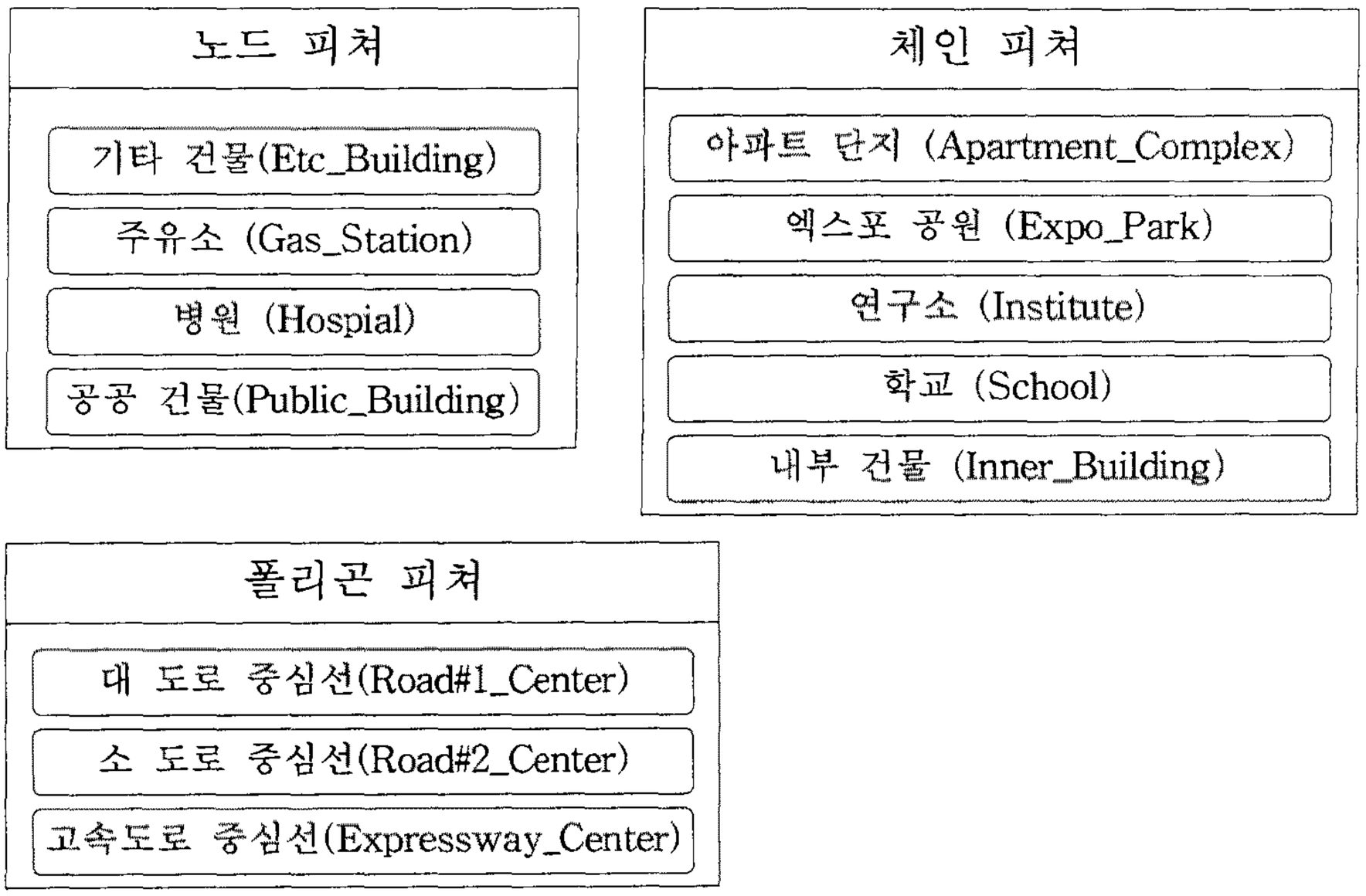


그림 4-29. 현 시스템에서 구성된 피쳐

피쳐-공간 조인 테이블은 피쳐 테이블 ID와 공간 테이블 ID를 이용하여 공간 정보와 비공간 정보와의 연결을 구성해 주는 테이블이다. 피쳐 테이블과 공간 테이블간의 관계는 하나의 피쳐 ID에 여러개의 공간 ID가 대응될 수 있고, 또한 하나의 공간 ID에 여러개의 피쳐 ID가 대응될 수 있다. 다시 말하면, 하나의 같은 위치의 공간 객체에 대하여 여러개의 속성 정보를 가질 수 있으며, 여러

개의 공간 객체가 모여서 하나의 속성 정보를 유지할 수 있음을 말한다. 이를 종합해 보면 피쳐 테이블과 공간 테이블간의 관계는 임의의 N : M의 관계가 설정될 수 있음을 알 수 있다.

다. 색인 데이터 테이블

데이터베이스 관리기에서 끝으로 관리하고 있는 테이블은 색인 정보 테이블이다. 색인 정보 테이블은 공간 데이터에 대한 신속한 검색을 위해 사용되는 색인 정보를 유지하고 있다. 이러한 색인 테이블은 피쳐 테이블과는 관계없이 모든 공간 테이블, 노드, 체인, 폴리곤 테이블에 대하여 각각 색인 정보를 생성하여 저장하고 있다. 실제 색인 테이블의 항목을 보면 [그림 4-30]과 같다.

[색인 정보 테이블]

색인트리 ID	색인트리 이름	색인 트리 크기	색인 정보

그림 4-30. 색인 데이터 테이블의 구조

라. 공간 테이블, 비공간 테이블, 색인 정보 테이블간의 관계

먼저 공간 테이블과 비공간 테이블은 피쳐-공간 조인 테이블을 이용하여 연결이 되며, 색인 정보 테이블은 공간 테이블의 최소 경계 사각형 정보를 이용하여 구성된다. 위의 세 테이블간의 관계는 [그림 4-31]과 같이 나타내어진다.

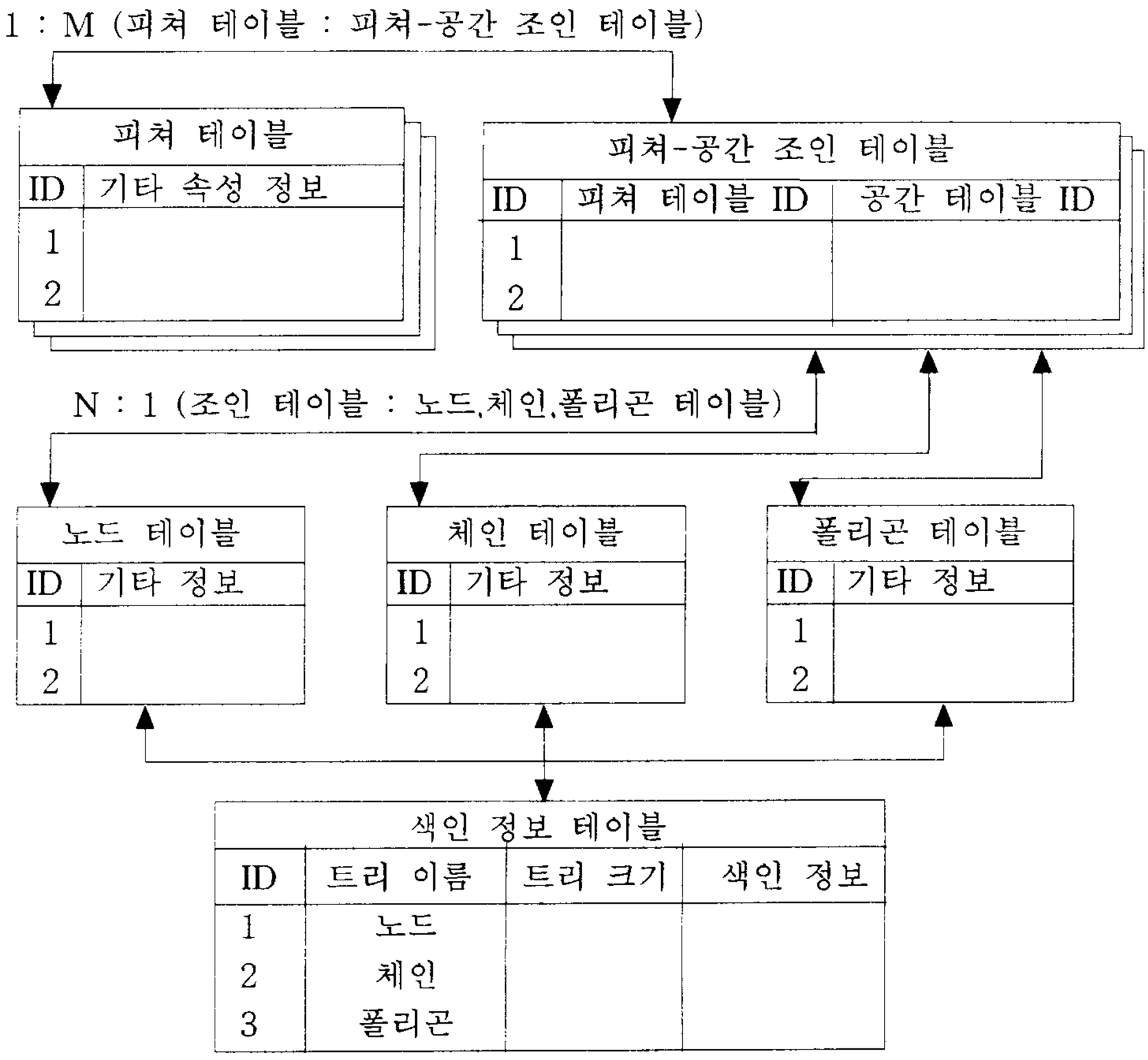


그림 4-31. 공간, 비공간, 색인 정보 테이블간의 관계

위의 그림에서 보듯이 먼저 피쳐 테이블은 피쳐-공간 조인 테이블과 피쳐 ID를 이용하여 1 : M의 관계로 조인되어지고, 피쳐-공간 조인 테이블은 각각의 노드, 체인, 폴리곤 테이블의 ID와 N : 1의 관계로 연결이 된다. 결론적으로 피쳐 테이블과 공간 테이블 사이의 관계는 M : N이 된다. 또한 공간 테이블은 색인 정보 테이블과 연결되는데, 각 공간 테이블에 대하여 하나의 색인 정보가 생성된다.

2. 공간 색인 관리기

공간 색인 관리기의 주요 기능은 노드, 체인, 폴리곤 테이블에 대한 색인 정보의 생성과 생성된 색인 정보를 이용하여 신속한 데이터베이스 검색 기능을 제공하는 것이다. 또한 데이터베이스와 파일 시스템 사이에 색인 정보가 자유로이 저장될 수 있는 기능을 제공한다. 본 시스템에서 사용된 공간 색인 방법은 최소 경계 사각형을 이용하는 R-tree 계열의 방법중에서 R*-tree의 색인 방법을 사용하고 있다.

가. R*-tree의 공간 색인 방법

1) R*-tree의 구성

R*-tree 는 공간 데이터를 색인하기 위해서 최소 경계 사각형이라는 데이터 구조를 이용하는데, 데이터들의 최소 경계 사각형을 계산하고 이를 통하여 공간 데이터에 대하여 접근 방법을 만드는 것이다. 다음 [그림 4-32]는 레벨 2까지의 R*-tree 구성을 보여준다.

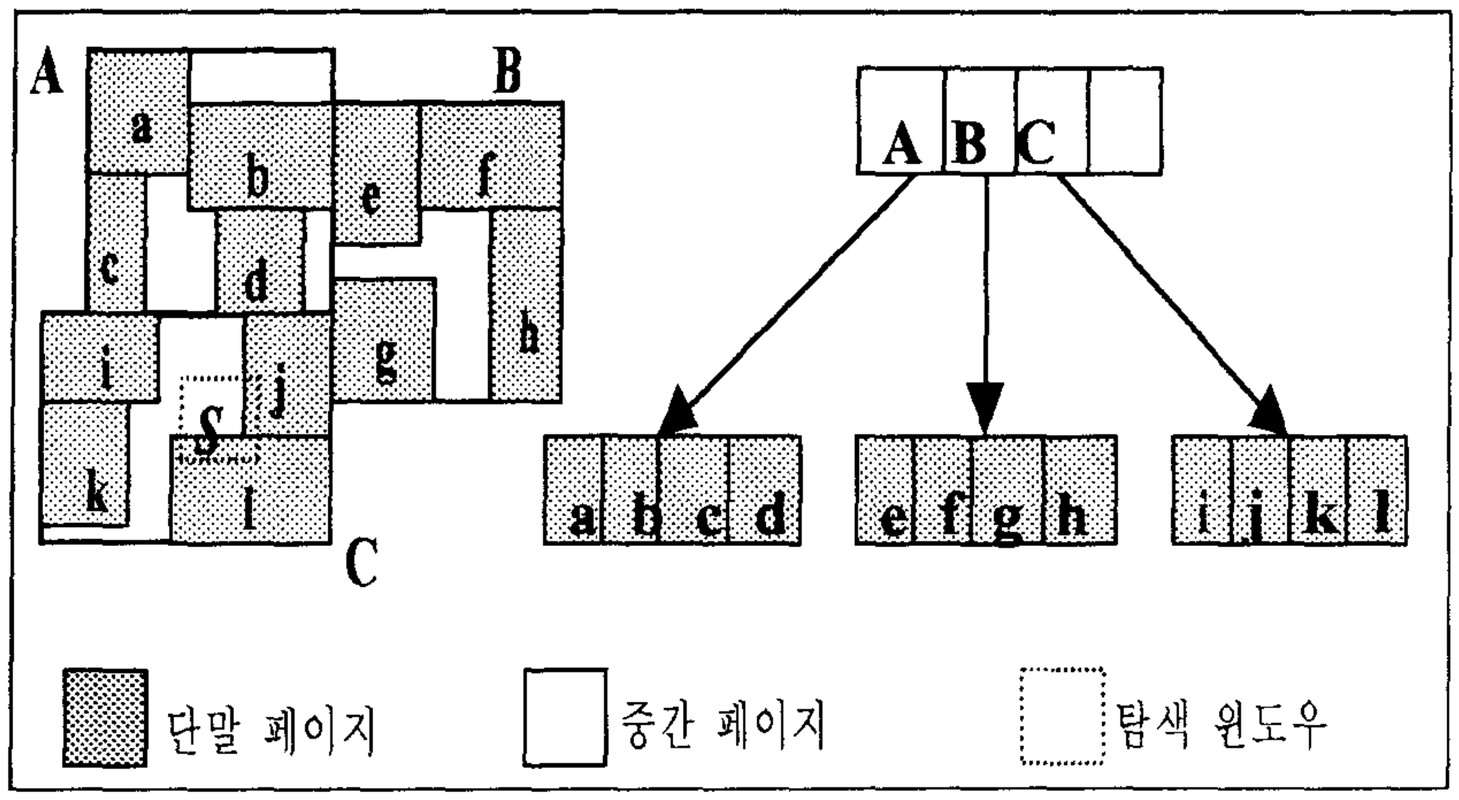


그림 4-32. 한 페이지내 데이터의 수가 최대 4이고 레벨이 2인 R*-tree 예

위의 그림 3-32에서 보듯이 R-tree계열의 방식들은 인접한 데이터들(a, b, c, d, e, f, g, h, i, j, k, l)을 다음 레벨의 최소 경계사각형(A, B, C)로 묶어서 같은 페이지내에 저장을 시키는 방식을 순환적으로 반복하면서 트리를 구성해 나간다. 또한 위의 경우 데이터가 '4' 보다 크거나 데이터의 삭제로 페이지가 사라지는 경우는 B+-tree와 같은 방식으로 분할 또는 접합을 수행하여 균형 트리(Balanced Tree)를 구성하도록 한다. 인접한 데이터들을 묶는 방식에 따라서 R-tree 계열이 나누어 지는데 먼저 데이터가 들어오는 순서대로 임의로 묶어주는 방식을 사용하는 R-tree가 있다. 둘째로 필요한 경우에는 데이터 자체를 분할하여 양쪽의 부모 페이지에 저장시키도록 하는 R+-tree 방법이 있으며, 끝으로 R-tree와 유사하게 입력 순서대로 묶은 다음, 부모 페이지를 조사하여 잘 못 묶인 경우는 다시 입력을 수행하도록 하는 방법인 R*-tree 방법이 있다. 본 시스템에서는 성능면에서와 안정성면에서 뛰어난 R*-tree의 방법을 채택하여 구현하였다.

2) R*-tree의 구현

R*-tree의 색인 방법은 다음 [그림 4-33]과 같은 클래스 집합들로 구성되어 있다.

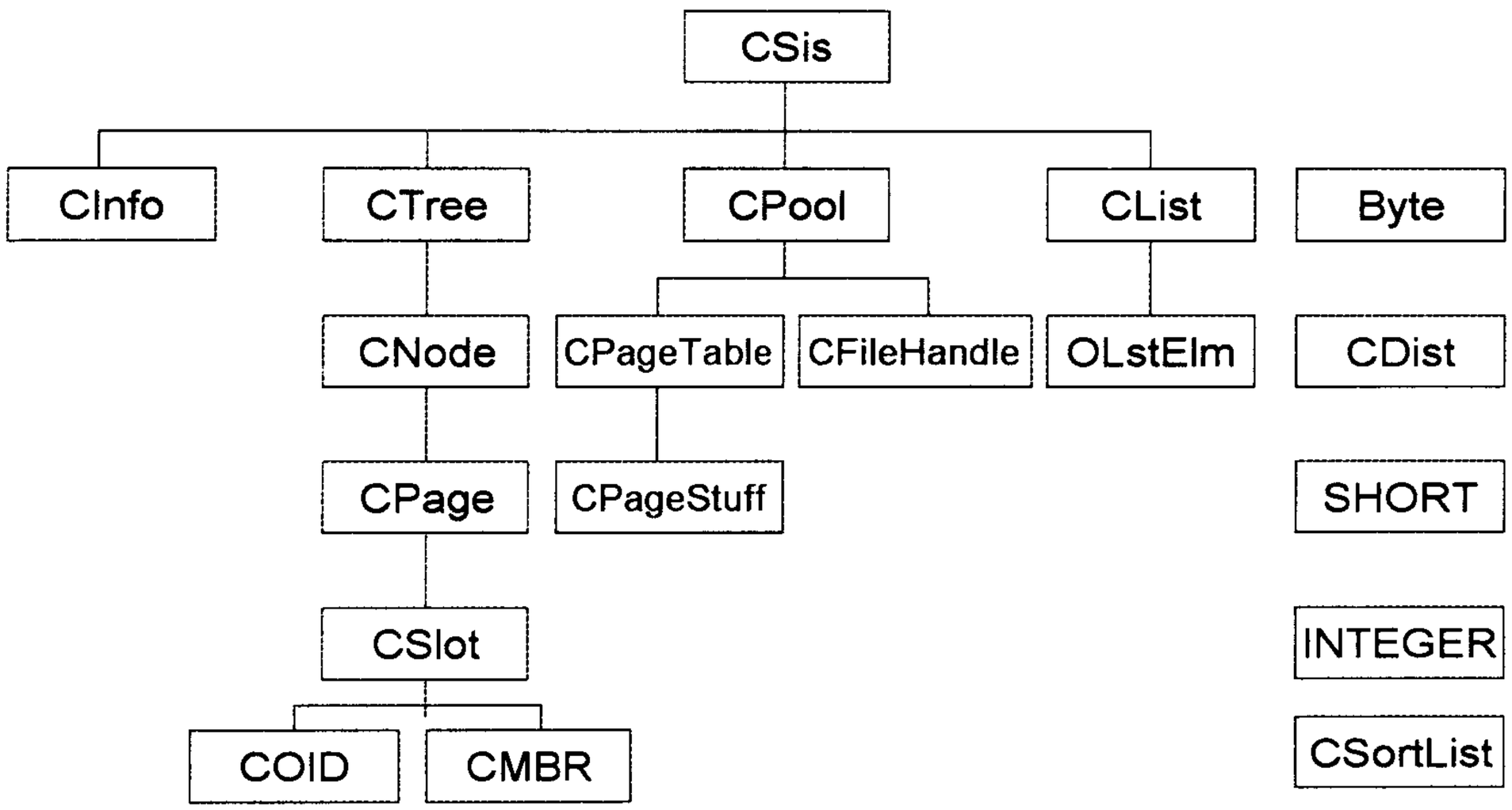


그림 4-33. R*-tree 클래스 구성도

가) 공간 색인 시스템 클래스 (CSis)

R*-tree 시스템의 최상위 클래스로서 R*-tree를 사용하기 위해 필요한 인터페이스 메소드들 유지하고 있다.

나) 색인 정보 클래스 (CInfo)

R*-tree를 형성하고 유지하기 위한 기본 정보를 유지하고 관리하는 클

래스이다. R*-tree를 유지하고 있는 정보로는 색인 트리 이름, 색인 트리의 크기, 색인 페이지의 크기, 전체 색인 페이지의 수, 색인 페이지내의 최대 데이터의 수, 색인 페이지내의 최소 데이터의 수, 그리고 페이지내의 데이터가 오버플로우 할 때 페이지 분할 방법의 수가 있다.

다) 색인 트리 클래스 (CTree)

R*-tree의 생성 및 탐색을 위한 핵심 모듈이다. R*-tree를 형성하기 위한 데이터의 입력, 삭제를 처리하고, 데이터의 언더플로우 또는 오버플로우 발생시에 페이지의 분할이나 접합을 처리한다. 또한 공간 탐색을 위한 기본 메소드가 구현 되어 있으며, 데이터베이스나 파일에 색인 정보를 저장하거나 로딩하기 위한 기본 메소드들이 구현 되어 있다.

라) 색인 풀 클래스 (CPool)

R*-tree의 완전한 색인을 형성하기 위해서 미리 색인 정보를 저장하는 색인 풀을 관리하는 메소드들을 가지고 있다. 색인 풀의 생성 및 삭제, 색인 풀에 만들어진 색인 결과를 파일이나 데이터베이스에 저장하거나 또는 색인 정보 수정시 읽어 들이는 역할을 수행한다.

마) 기타 클래스 (CList, OLstElm)

그 외의 클래스로는 색인 정보를 저장하기 위한 클래스와 정렬 메소드 외에 R*-tree 유지에 필요한 클래스들을 가지고 있다.

3) R*-tree 인터페이스 메소드

R*-tree를 손쉽게 접근하기 위해서 생성 및 삭제 그리고 갱신, 검색을 위한 인터페이스 메소드들이 다음과 같이 정의 되어 있다.

가) 색인 풀 생성 메소드

- boolean CreateIndexPool(색인 풀 이름, 색인 페이지 크기, 색인 풀 크기)
- 색인 풀은 색인이 생성되고 있는 임시 파일을 말한다.
- 현재 시스템에서 색인 페이지의 크기는 4K이다. 색인 페이지의 크기가 증가 하면 한 페이지내의 최대 데이터의 수가 증가하나 일반적으로 디스크에 읽고 쓰는 단위인 4K로 페이지 크기를 정한다.
- 색인 풀의 크기는 색인이 완성되었을 때의 전체 색인 페이지들의 개수의 추정 값이다. 파일 시스템에 색인 풀이 생성되므로 넉넉하게 잡아도 색인 완성후 다시 반환되므로 큰 문제가 되지 않는다. 본 시스템에서는 1,000개의 색인 페이지를 추정치로 사용하고 있으며, 이는 약 100,000개 가량의 데이터를 수용할 수 있다.

나) 색인 트리 생성 메소드

- boolean CreateIndexTree(색인 트리 이름, 색인 트리 ID, 색인 페이지 크기)
- 생성되는 색인 트리의 이름은 색인 정보가 파일에 저장될 때 파일 이름으로 사용되며, 색인 트리 ID는 데이터베이스에 저장될 때의 ID로

사용된다. 색인 페이지의 크기는 색인 풀을 생성할 때 주어진 값과 같은 것이다.

다) 색인 데이터 입력 메소드

- int RSTInsert (최소 X 좌표, 최소 Y 좌표, 최대 X 좌표, 최대 Y 좌표, 데이터의 ID)
- 최소 경계 사각형과 ID를 이용하여 데이터를 입력한다. 입력이 실패하면 '-1'을 반환한다.

라) 색인 데이터 삭제 메소드

- int RSTDelete(최소 X 좌표, 최소 Y 좌표, 최대 X 좌표, 최대 Y 좌표)
- 주어진 최소 경계 사각형과 일치하는 데이터를 삭제한다. 일치하는 데이터가 없으면 '-1'을 반환한다.

마) 색인 정보 출력 메소드

- void RSTReport(정보 타입), void RSTGather(색인 트리 레벨)
- 주어지는 정보의 타입에 따라서 다른 정보를 출력하는데, '1'인 경우는 실제 색인 내의 데이터 정보(CTree)를 출력하고 '2'인 경우는 앞에서 설명한 색인 정보 클래스(CInfo)의 내용을 출력한다. 그리고 '3'인 경우는 색인 풀(CPool)의 내용을 출력한다.
- 색인 트리 레벨이 주어지는 경우는 색인 트리에서 원하는 레벨에 있

는 데이터들만을 따로이 출력할 수 있다. '-1'인 경우는 모든 레벨에 있는 데이터들을 출력한다.

바) 색인 검색 메소드

- int Search(최소 X 좌표, 최소 Y 좌표, 최대 X 좌표, 최대 Y 좌표, 검색 타입, 결과 포트)
- 주어진 최소 경계 사각형의 좌표는 검색을 위한 윈도우이다.
- 검색 타입으로는 네가지가 있는데, '1'의 경우 겹침(Overlap) 검색, '2'는 일치(Equal) 검색, '3'은 포함(Contained) 검색, 그리고 '4'는 피포함(Containing) 검색 방법을 말한다.
- 결과 포트는 '1'과 '2'의 두 개의 포트가 있으며 어느 것을 이용해도 상관은 없다.

사) 색인 트리 삭제 메소드

- void RemoveIndexTree()
- 색인 트리를 삭제한다.

아) 색인 풀 삭제 메소드

- void RemoveIndexPool()
- 색인 풀을 삭제한다. 색인 형성이 완료되면 불리어 진다.

자) 색인 데이터 파일 접근 메소드

- int ImportIndexFile (색인 트리 이름), int ExportIndexFile (색인 트리 이름)
- 색인 정보를 파일에 저장하거나 파일로부터 읽어오는 메소드이다.

4) R*-tree의 동작 순서

R*-tree의 동작 순서는 먼저 색인 풀 생성 그리고 R*-tree를 새로이 생성하거나 또는 R*-tree를 파일이나 데이터베이스에서 읽어온다. R*-tree의 색인 정보가 준비 되었으면 입력, 검색, 삭제 작업을 수행하고 다시 R*-tree를 저장하고 색인 풀을 삭제하는 것으로 동작 순서를 마친다고 볼 수 있다. [그림 4-34]는 이러한 동작 순서를 보여준다.

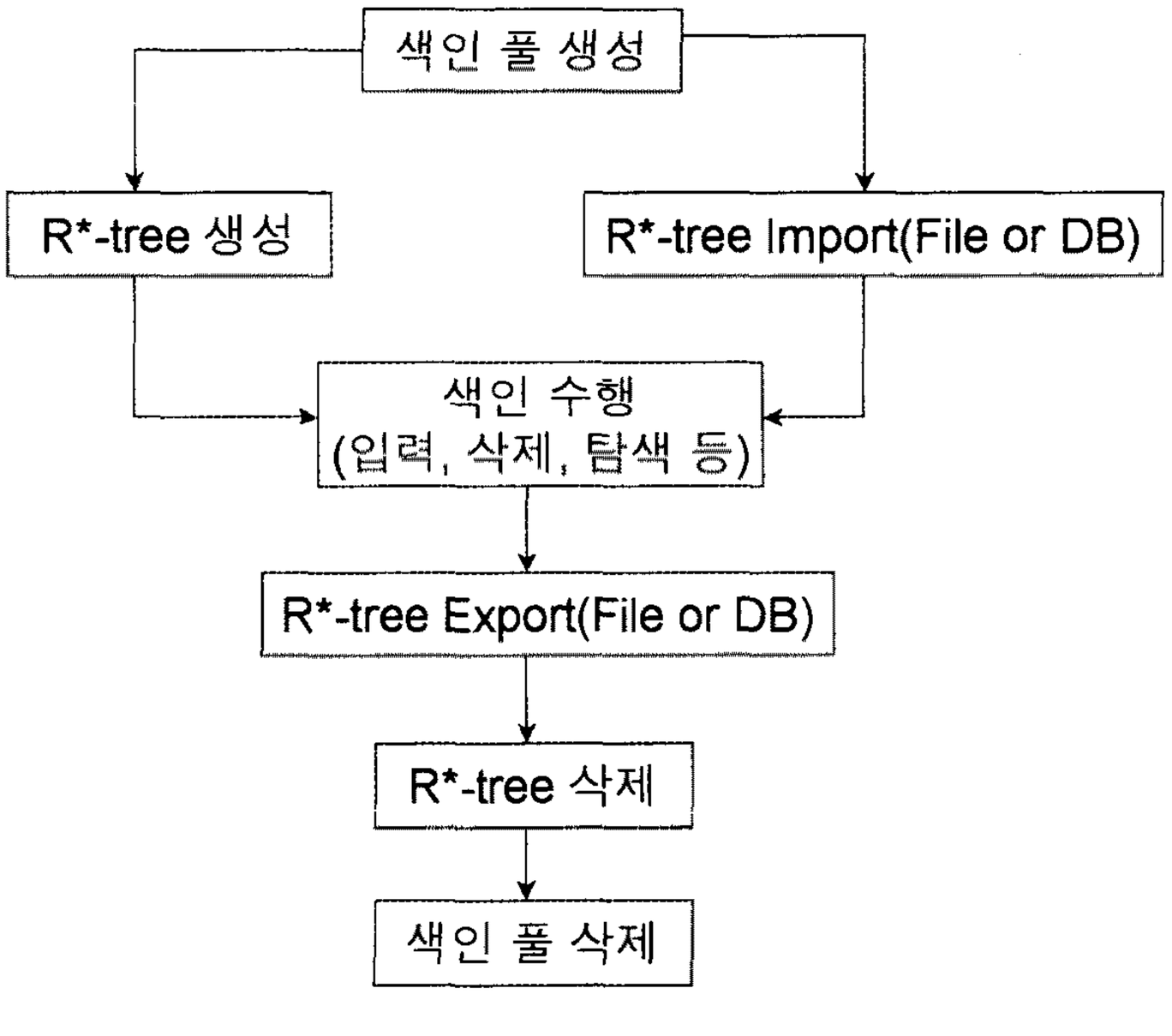


그림 4-34. R*-tree의 일반적인 동작 순서

나. 공간 데이터의 색인

본 시스템은 노드, 체인, 그리고 폴리곤의 공간 데이터에 대하여 공간 색인을 구축하는데, [그림 4-35]는 공간 색인 생성을 위한 의사 코드를 보여준다.

```
public void CreateSpatialIndex(DataFileName) {
        // 데이터 파일 읽기
        Data = ReadDataFromFile(DataFileName);

        // 데이터 입력
        while (Data.next) {
                // 공간 데이터 입력
                SpatialDBInsert(SpatialData);

                // 비공간 데이터 입력
                AspatialDBInsert(AspatialData);
}

// 공간 색인 생성
// 색인 풀 생성
if (CreateIndexPool("INDEXPOOL", 4096, 1000))
        System.out.println("색인 풀 생성 : Success");
else
        System.out.println("색인 풀 생성 : Failure");

// 색인 트리 생성
if (CreateIndexTree(TreeName, 1, 4096))
        System.out.println("색인 트리 생성 : Success");
else
        System.out.println("색인 트리 생성 : Failure")
```

```
            // 색인 생성
            ret = CreateIndex(SpatialData);

            // 색인 정보 저장
            ret = SISDBInsert(IndexingInformation);

            // 색인 트리 삭제
            RemoveIndexTree();

            // 색인 풀 삭제
            RemoveIndexPool();
        }
```

그림 4-35. 임의의 공간 데이터에 대하여 공간 색인을 생성하는 의사코드

1) 공간 데이터 구축

먼저 노드, 체인, 폴리곤에 대한 공간 데이터를 데이터베이스에 구축한다. 본 시스템은 파일 시스템내의 데이터를 이용하지 않고 데이터베이스내의 데이터를 이용하므로, 색인 정보를 생성하기 위해서는 먼저 공간 데이터들이 데이터베이스에 구축됨을 필요로 하는 것이다.

2) 공간 색인 생성을 위한 정보 추출

공간 색인을 생성하기 위해서는 각 공간 데이터에 대하여 최소 경계 사각형과 ID의 정보를 필요로 하게 된다. 본 시스템에서는 공간 데이터 테이블내에 저장되어 있는 최소 경계 사각형의 정보를 이용하며, ID로는 공간 테이블내의 각 데이터에 대한 rowid(오라클에서 제공하며, 데이터에 대하여 직접 액세스 가능한

ID)를 이용한다. 끝으로 {최소 경계 사각형, rowid}의 쌍이 공간 색인을 형성하기 위한 기본 정보로서 이용된다.

만약, 데이터베이스가 아닌 파일 시스템내에 존재하는 공간 데이터들에 대한 색인을 생성하고자 한다면, 색인을 위한 정보로서 오라클의 rowid가 아닌 파일시스템내의 물리적 주소를 이용하여 색인을 생성하면 된다. 물론 데이터베이스내에서도 rowid가 아닌 테이블에 주어진 ID를 이용해도 된다.

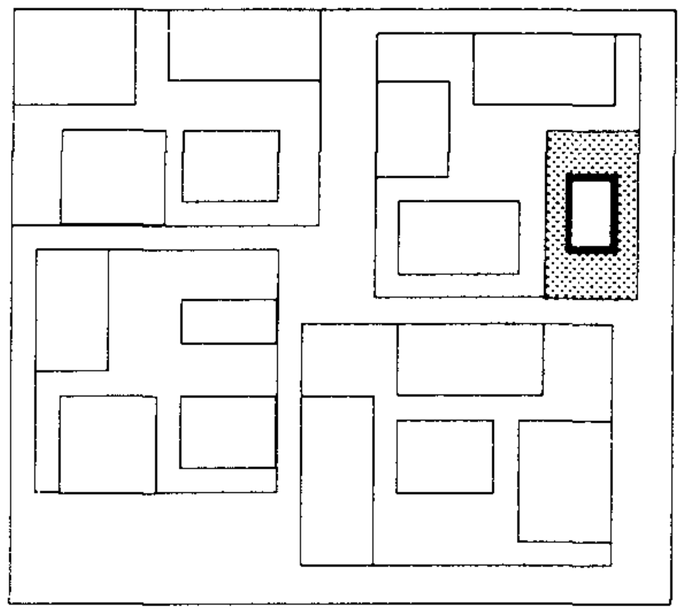
3) 공간 색인 생성

공간 색인은 한 작업 영역에서 노드, 체인, 폴리곤의 공간 데이터 테이블에 대하여 각 하나씩 생성되므로 공간 색인 정보 테이블내에는 노드, 체인, 폴리곤의 색인 정보들이 생성되게 된다. 그리고 공간 색인 정보는 이진 형태로 저장되며, 색인 정보의 크기는 4K 바이트 단위로 성장하게 된다.

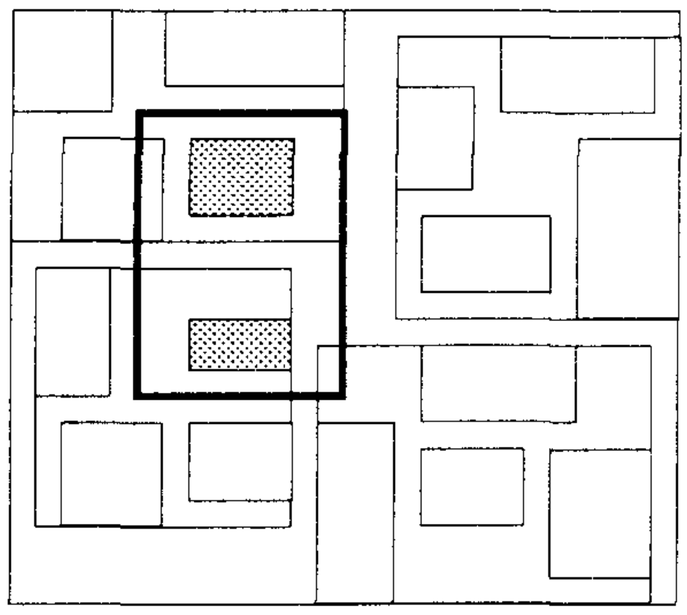
다. 공간 색인을 이용한 검색 방법

1) 공간 검색 방법의 유형

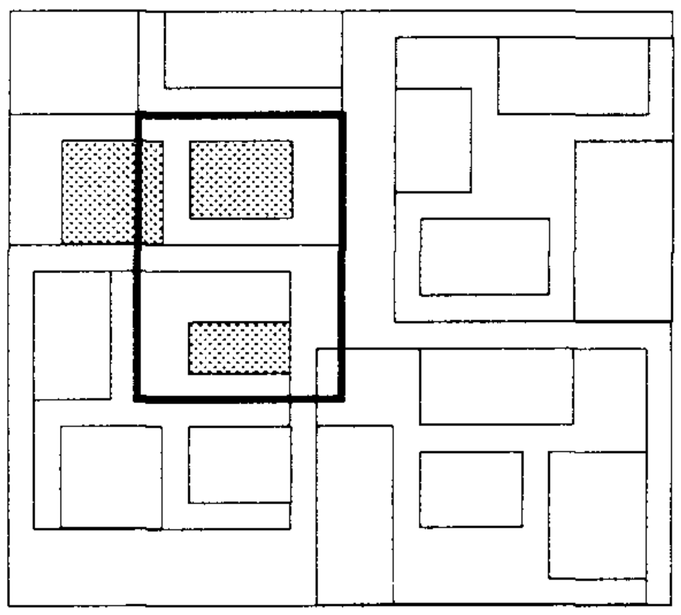
공간 색인의 구축이 완료되면 색인 정보를 이용하여 공간 검색을 수행할 수 있는데, 본 시스템에서 제공하는 공간 검색 방법은 [그림 4-36]과 같이 네 가지 방법을 제공하고 있다.



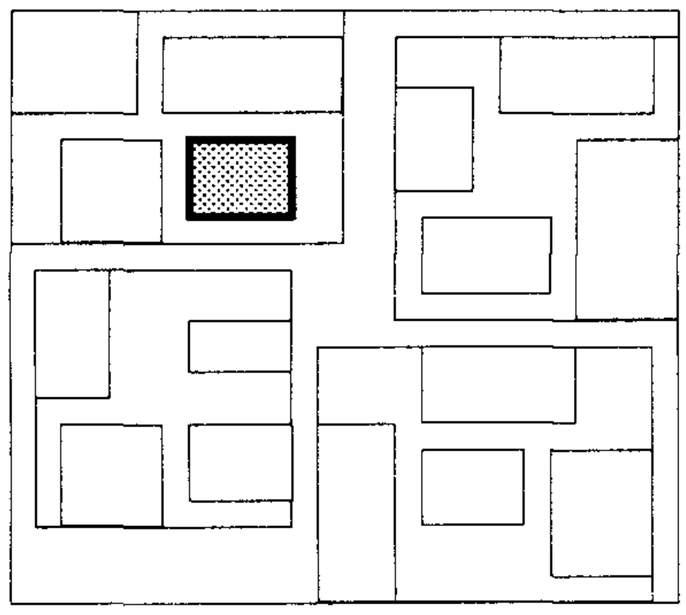
A. 피포함(Containing) 검색



B. 포함(Contained) 검색



C. 겹침(Intersect) 검색



D. 일치(Exact) 검색

그림 4-36. 본 시스템에서 제공하는 공간 검색 유형

위의 그림에서 보듯이 A의 피포함 검색방법은 검색 윈도우가 주어졌을 때 검색 윈도우를 포함하는 폴리곤 데이터를 탐색하는 방법으로, 임의의 지역이 어느 구역에 속하는지를 알고자 하는 질의에 쉽게 응용 될 수 있다. B의 포함 검색은 임의의 영역을 설정한 다음 그 영역내에 있는 데이터들을 검색하는 응용에 주로 사용이 된다. 예를 들면, "유성구에 포함되어 있는 모든 병원을 찾아라." 라는 질의에 쉽게 이용이 될 수 있다. C의 겹침 검색은 포함 검색과 유사한 응용에 사용되는데, 겹치는 것을 포함해서 검색을 수행하므로 B보다 넓은 범위의

검색을 수행하는 응용에 사용된다. 끝으로 D의 일치검색은 임의의 공간 데이터를 찾는데 사용되는 검색 방법이다.

2) 공간 검색의 동작 과정

공간 검색을 이용하여 공간 데이터와 비공간 데이터를 찾고 원하는 질의를 수행하는 일반적인 과정은 [그림 4-37]와 같이 일어난다.

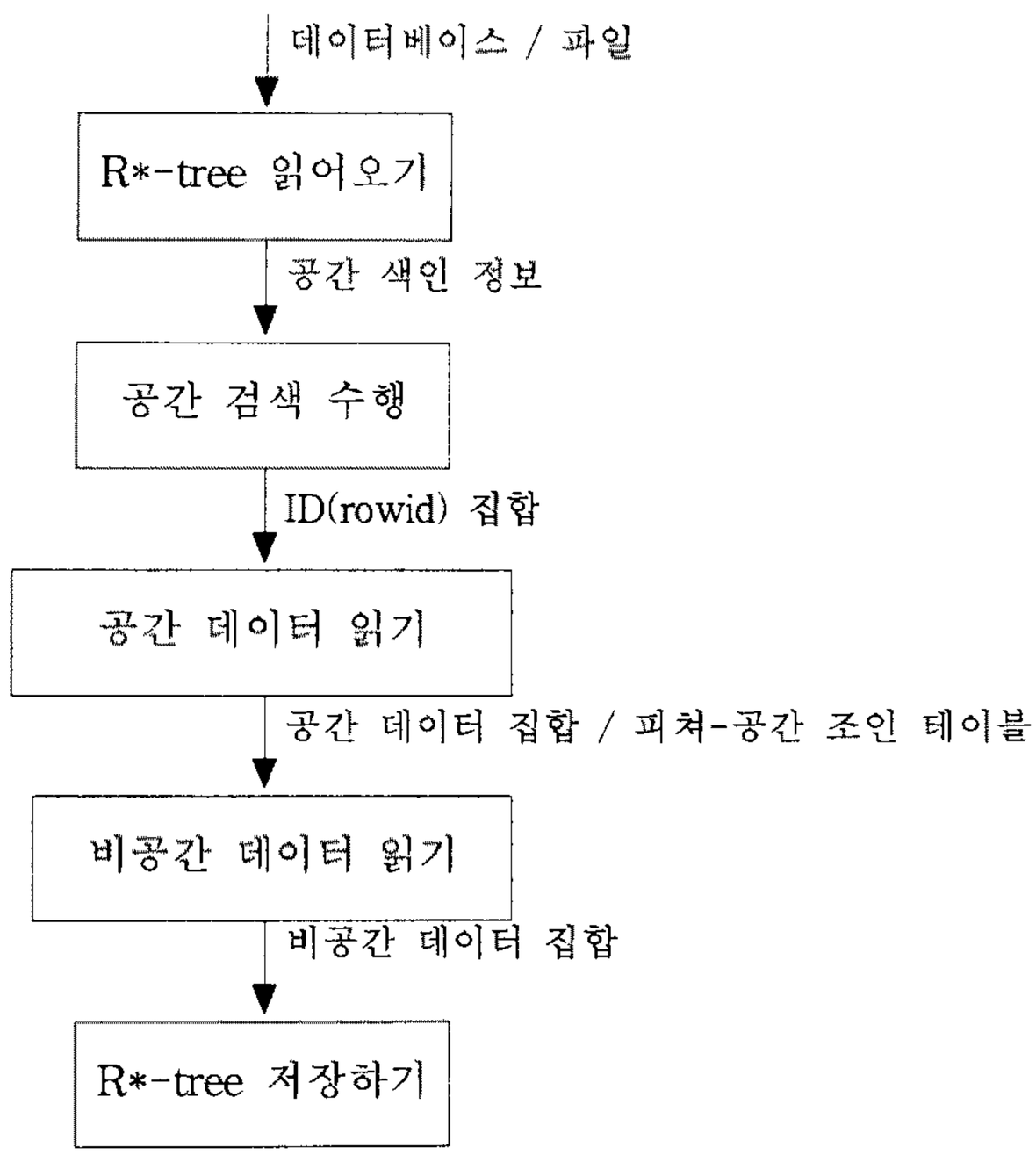


그림 4-37. 공간 색인 관리기의 공간 검색 수행 과정

공간 검색을 수행하기 위해서는 공간 색인 정보가 필요하므로, 위의 그림에서 보듯이 먼저 R*-tree의 색인 정보를 데이터베이스 또는 파일 시스템으로부터 읽어 온다. 공간 색인 정보가 로딩 되면 검색 윈도우와 유형에 따라 공간

검색을 수행한다. 공간 검색 수행의 결과는 검색 조건을 만족시키는 공간 데이터의 ID(rowid) 집합이 된다. 이후에는 공간 데이터의 ID 집합을 가지고 공간 데이터 테이블을 액세스하여 공간 데이터를 읽어 들이고 피쳐-공간 조인 테이블을 이용하여 비 공간 데이터를 또한 읽어 들이면 원하는 데이터 집합을 찾게 된다. 그리고 끝으로 더 이상 공간 색인 정보를 사용하지 않는다면 R*-tree 색인 정보를 다시 저장하고 검색 작업을 종료한다.

제 5 장 AWT(Abstract Window Toolkits)를 이용한 사용자 작업 환경 구현

제 1 절 사용자 인터페이스 환경 구축

전 세계적으로 정보를 하나로 묶어주는 인터넷에 대한 관심과 사용이 증대되고 있다. 또한 인터넷에서 동적인 프로그램이 가능하게 하는 자바에 대한 관심은 가히 폭발적이다. 자바 프로그래밍 언어는 월드 와이드 웹(World Wide Web) 사용자들이 인터넷상에서 플렛폼에 구애받지 않고 응용프로그램을 작성, 사용할 수 있기 때문이다. 자바는 WWW에서 애플렛(applet)을 사용하여 웹 페이지 안에서 작동하는 작은 크기의 프로그램으로써 단 한번 다운로드 되면 웹 서버(Web Server)의 자원들을 사용하는 것이 아니라 웹 페이지를 검색하고 있는 사용자들과 상호 작용을 하면서 작업을 수행한다. 이런 이유 때문에 웹에서 작동 가능한 지리정보시스템(GIS)을 개발하는데 자바는 아주 유용하다. 현재 일반적으로 사용되고 있는 GIS는 데스크탑 형태로써 작업을 수행하지만 웹에서 수행되는 것은 네트워크를 통하여 클라이언트/서버 구조를 가지면서 서버에 있는 자료를 네트워크를 통하여 지리정보를 공유, 분석할 수 있다. 자바에서 제공되는 AWT(Abstract Windows Toolkits)가 계속적으로 보완되면서 버전이 올라가고 있어 GUI를 구현하기 위한 환경이 계속 좋아지고 있다.

자바 언어는 또 다른 특징은 플렛폼 독립적이지만 운영체제에 따라 윈도우즈 형태가 다르기 때문에 GUI는 운영체제에 따라 다르게 나타난다. 현재까지 사용된 버전은 자바 1.1(JDK1.1) 이고 이를 이용하여 응용프로그램에 사용되는 세부 클래스들을 디자인하여 사용하였다. 여기서는 자바의 설치 및 사용 방법에 대하여 상세히 설명할 것이다. 자바 프로그래밍을 하기 위해서는 자바의 설치와

환경설정이 무엇보도도 중요하다. 아래는 자바를 입수 및 설치하는 과정을 나타내는 것이다.

1. 자바 프로그래밍 언어 설치하기

자바 프로그래밍 언어의 설치는 프로그램 구입, 운영체제에 따른 디렉토리 경로설정하고 자바를 컴파일 하면 된다. 그런데 컴파일 시 JDK는 DOS 모드에서 운영된다는 점을 잊어서는 안 된다. 프로그램 구입은 인터넷의 http://java.sun.com에 가서 다운로드 받아 사용하거나 CD-ROM을 가지고 설치한다.

가. 윈도우즈 95 및 윈도우즈 엔티

윈도우즈 95 및 윈도우즈 엔티에서 최상의 디렉토리로 이동한 후 다운로드된 파일을 풀거나 아래와 같이 CD-ROM을 통하여 설치한다.

```
C:\WINDOWS> CD\ <ENTER>
C:\D:\JDK101 <ENTER>
```

나. 유닉스(SUN Solaris2.5)

JDK를 설치(/usr/local/jdk1.1.4)한 후에는 path를 설정하여야 한다. CLASSPATH는 ~/.cshrc에 set path=($path /usr/local/jdk1.1.4/bin) 있나를 확인한 후 source 시킨다.

```
% source ~/.cshrc
```

(상세한 것은 http://java.sun.com에서 JDK에 관한 README 설명서를 참조)

다. 디렉토리 구성

환경변수를 설정하기 위하여 PATH 및 HOME를 수정해야 한다. PATH는 PATH 명령을 사용해서 C:\JAVA\BIN 디렉토리 명을 추가하여야 하고, HOME = SET 명령을 사용해서 HOME 엔트리를 JAVA 디렉토리를 포함한 드라이브와 디렉토리에 할당한다. AUTOEXEC.BAT 파일을 편집하고 PATH 명령을 갱신해서 C:\JAVA\BIN 디렉토리를 포함한다. 그 다음에 자바파일의 디렉토리 경로에 HOME 변수를 할당하는 SET 명령을 추가시킨다.

서브디렉토리	내용
Java	주 디렉토리
java\lib	자바 클래스 라이브러리를 포함할 때 기본적으로 편집 중에 자바 컴파일러가 classes.zip에서 클래스의 압축을 푼다
java\bin	컴파일러, 디버거, 애플렛 뷰어 등과 같은 실행 가능한 자바프로그램을 포함한다.
java\include	다양한 자바 클래스에 해당하는 c++ 헤더파일을 포함한다.

라. Visual programming 인 visual cafe1.1설치하기

본 프로그램을 구입(다운로드 또는 CD-ROM)하여 설치한다. 본 프로그램은 symentec에서 개발한 것으로 여기서는 1.1d 버전을 사용하여 개발하였다. 그러나 기본적으로 JDK만 설치되어 있는 사용자들의 편의를 위하여 본 프로그램에서만 제공되는 모듈은 거의 사용하지 않고 개발하고자 노력하였다.

1) 응용프로그램 컴파일하기

간단한 소스프로그램(HelloWorldApp.java)을 만들었을 때 아래와 같이

컴파일하면된다.

c:\>javac HelloWorldApp.java

만일 컴파일러가 에러 메시지가 없으면, 새로운 파일 HelloWorldApp.class가 동일한 디렉토리에 저장된다.

2) 응용프로그램 작동하기

응용프로그램인 HelloWorldApp을 작동하기 위해서는 자바 해석기를 사용하고 아래와 같이 실행하면 된다.

c:>java HelloWorldApp

3) 자바 애플렛 테스트하기

HelloWorldApp 소스 코드 파일을 응용프로그램 컴파일하기에서와 같은 방법을 사용하면 성공적으로 컴파일 될 것이다. 그러면 생성된 HellowWorldApp.class은 HelloWorldApp 애플릿을 나타내는 바이트 코드이다. 이 애플릿을 빠르고 쉽게 테스트하는 방법은 웹을 이용하는 방법과 애플릿 뷰어를 이용하는 2가지 방법이 있다. 본 과제에서는 두 가지 방법을 모두 사용하여 시험하면서 프로그램을 완성하였다.

첫째 : 웹에서 HTML을 만들어 이용하는 방법

애플릿을 테스트하기 위하여 HTML 파일 내에 애플릿을 위치시키는데 사용되는 <applet> 태그를 포함하는 간단한 HTML 파일은 만들 필요가 있다.

<applet> 태그는 자바 소스코드 파일처럼 이 파일은 text-only 형식으로 파일을 저장할 수 있는 워드프로세서나 텍스트 에디터를 사용하여 만들 수 있다. 그리고 적절한 파일이름을 줄 수 있다. 즉 index.html은 사람들이 이러한 파일을 위하여 사용하는 이름이다.

빈 윈도우에 다음을 입력하면 된다.

```
<title> HelloWorldApp</title>
<hr>
<applet code="HelloWorldApp" width=200 height=40>
</applet>
<hr>
```

아래 [그림 5-1]은 window의 에디터인 notepad 에디터로써 사용된 일례를 보여주는 것이다.

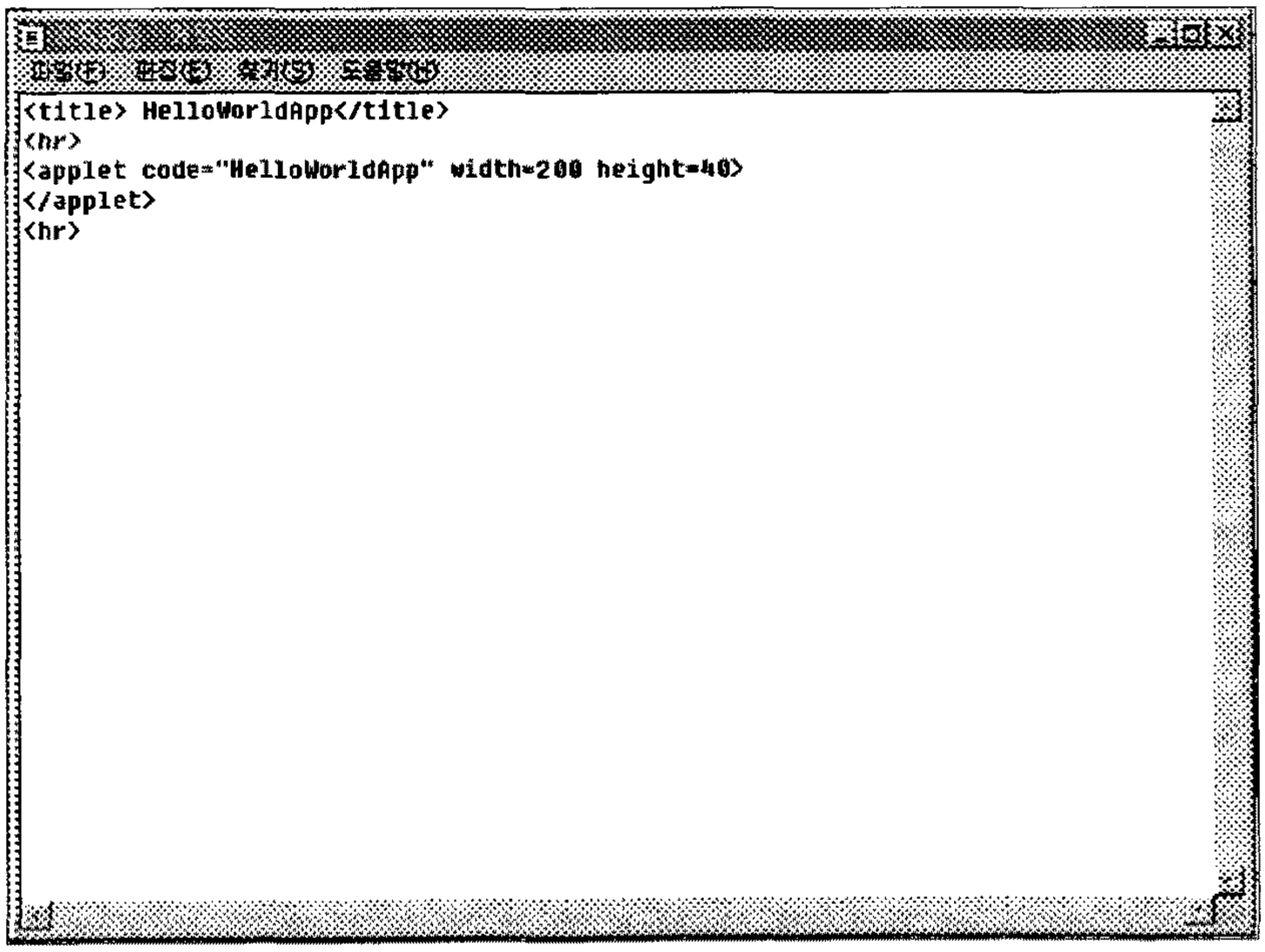


그림 5-1. index.html을 만들기 위해 사용한 윈도우 에디터인 노트패드

둘째 애플릿 뷰어를 이용하는 방법

- 에플렛 뷰어는 웹브라우저 없이 HTML 파일을 실행할 수 있게 해주는 작은 어플리케이션이다. 에플렛 뷰어가 자바 호환 어플리케이션이기 때문에 그것은 애플릿의 바이트 코드를 실행할 수 있다. 우선 애플릿을 가지는 폴더로 이동하여 도스 프롬프트에서 애플릿 뷰어를 실행할 수 있다. 그리고 appletviewer index.html을 입력하고 애플릿으로 참조를 가지는 HTML 파일의 이름 다음에 애플릿 뷰어 실행파일 이름이다. 아래 [그림 5-2]는 애플릿 뷰어가 나타내는 것을 보여주는 것이다.

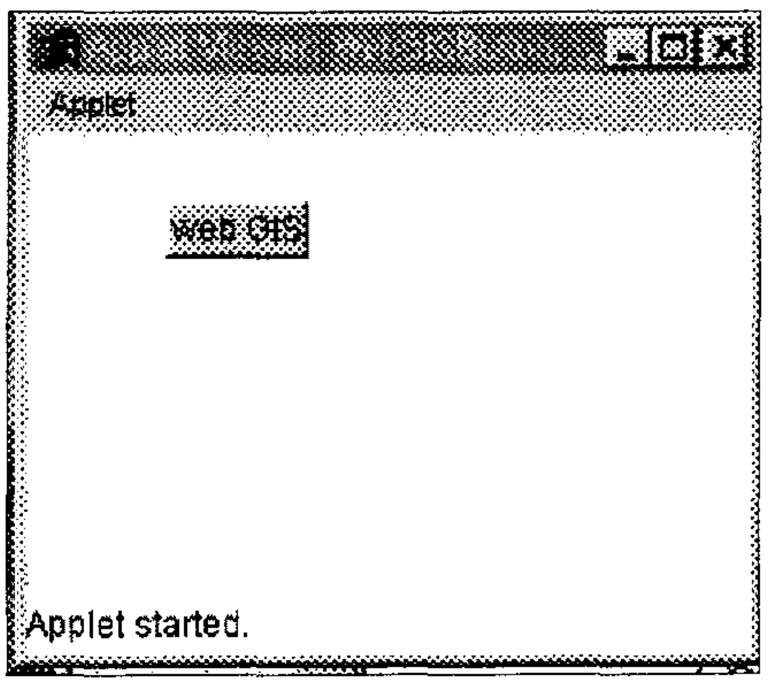


그림 5-2. 에플렛 뷰어에서 사용되는 WebGISButton.

2. 자바 응용프로그래밍 인터페이스(Application Programming Interface)

응용프로그래밍 인터페이스(Application Programming Interface) 또는 API는 일반적으로 오퍼레이팅 시스템(OS) 또는 프로그래밍 환경을 수반하는 기능(function)들의 큰 세트이다. 썬 사에서 정의된 클래스들은 자바 응용프로그래

밍 인터페이스, 또는 API를 구성한다. 자바 API를 구성하는 6개의 자바 패키지가 간단한 설명과 함께 아래 표 1 에 나타내었다. 그리고 이중에서 본 과제의 GUI(Graphical user Interface)를 수행하기 위하여 중점적으로 사용한 것은 java.awt이다(Sydow, 1997).

표 5-1. 자바 API 패키지.

패키지	목적
java.applet	애플릿의 도구를 위한 클래스
java.awt	그래픽과 GUI의 도구를 위한 클래스
java.io	입력과 출력의 조작을 위한 클래스
java.lang	자바 언어 자체를 정의한 클래스
java.net	네트워크 기능의 도구를 위한 클래스
java.util	각종 유틸리티 조작을 위한 클래스

가. Java.awt 클래스

자바 Abstract Window Toolkit(AWT)은 앞에서도 언급했듯이 그래픽에 의한 사용자인터페이스(GUI)를 역동적으로 구성하기 위하여 사용하는 것으로 버튼, 팝업메뉴, 체크박스 및 입력상자 등과 같은 것을 포함하는 오브젝트로써 애플릿을 사용자가 사용하기 쉽게 만들 수 있게 해 준다. [그림 5-3]은 AWT에 속한 여러 클래스의 계층을 보여주는 것이다(Sydow, 1997).

1) 콘테이너 클래스

콘테이너 클래스는 AWT 에 속한 것으로 콤포넌트 클래스의 여러 서브 클래스 중의 하나이다. 본 클래스는 요약 클래스이고 자기 소유의 메소드들은 콤포넌트 클래스로부터 상속받고 이러한 모든 메소드들을 자신 소유의 두 서브클래스인 판넬 클래스와 윈도우 클래스로 전달하는 큰 몸체의 메소드에 추가하는 것이다.

2) 판넬 클래스

판넬 오브젝트는 사용자 인터페이스 성분들을 배열하고 간격을 놓는 것을 쉽게 하는 성질을 가지고 있다.

3) 애플릿 클래스

애플릿 클래스는 [그림 5-3]에서 알 수 있듯이 패널클래스의 서브클래스로써 다른 것들을 패널에 포함되어 있는 것을 포함하는 것이라는 것을 의미한다. 또한 사용자 인터페이스 성분은 이 클래스에 포함된다.

4) 애플릿에 사용자인터페이스 성분 더하기

사용자인터페이스(UI) 성분은 [그림 5-3]에서 알 수 있듯이 항상 콤포넌트 클래스의 서브 클래스이다. 버튼은 Button 클래스의 오브젝트이며, 체크박스는 CheckBox 클래스의 오브젝트 등이다. UI 성분은 생성하기 위해 사용자들은 이러한 클래스 중 한 클래스의 오브젝트를 생성해야 한다. UI 성분은 추가하려면 다음과 같은 3 단계를 따라야 한다.

① 콤포넌트 클래스의 적절한 서브 클래스 변수를 선언한다.

② new를 이용하여 UI 성분의 예를 생성한다.

③ add() 메소드를 이용하여 UI 성분을 애플릿에 추가한다.

애플릿에 "WebGIS" 이라고 표시된 버튼을 추가하기 위해 사용된 예는 다음과 같다.

```
Button WebGISButton;
WebGISButton = new Button("WebGIS");
add(WebGISButton);
```

생성된 후 UI 성분은 항상 콘테이너에 추가되어야 한다. [그림 5-3]에서 보여 주듯이 애플릿 클래스는 패널 클래스의 서브클래스이다. 또한 그것은 콘테이너 클래스의 서브클래스이며 콤포넌트 클래스의 서브클래스이다.

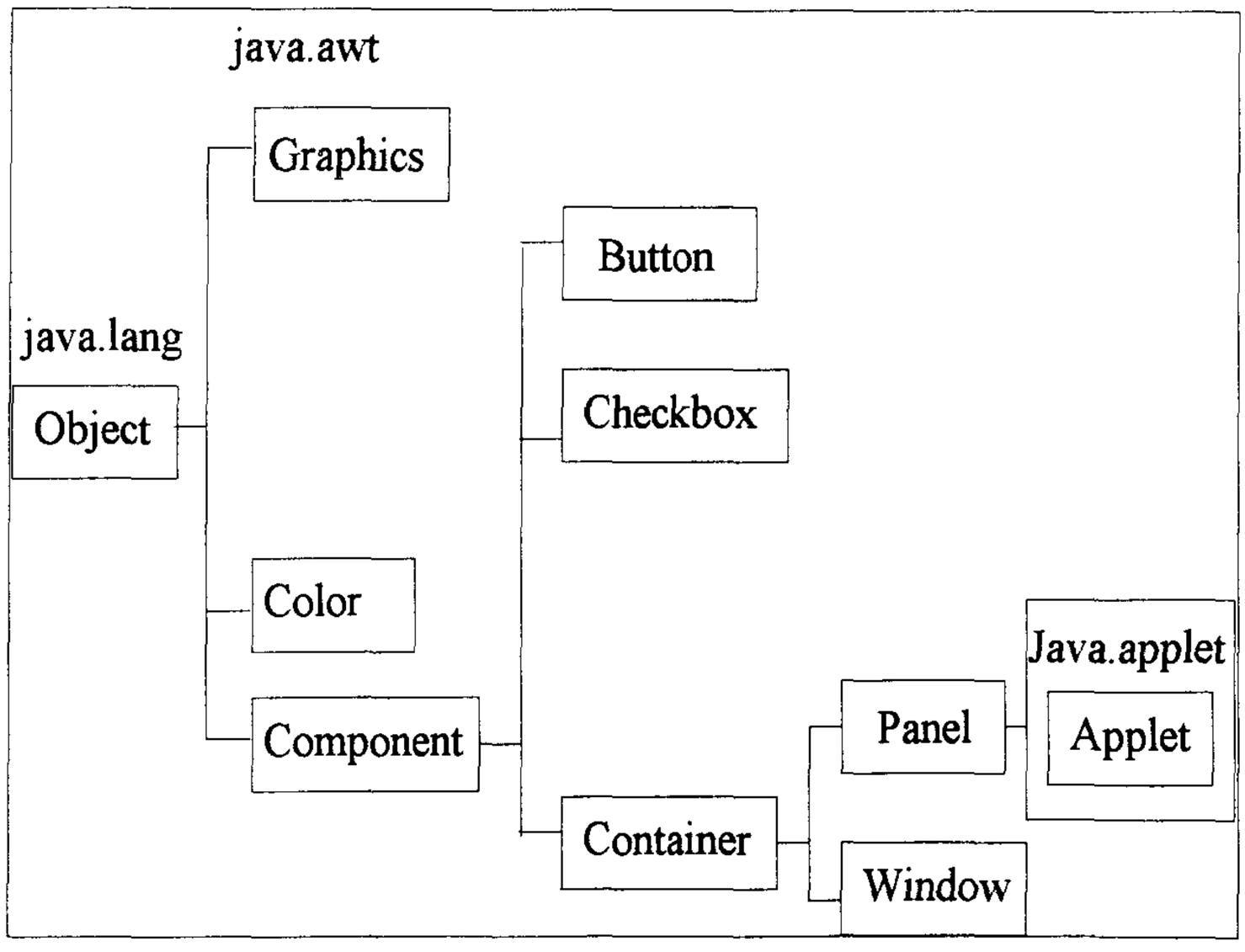


그림 5-3. Java.awt의 부분적인 클래스 계통도

3. 좌표변환

지구상의 위치를 표시하기 위한 좌표계에는 평면좌표계(평면직교좌표계, 평면극좌표계, UTM 좌표계), 곡면좌표(경위도 좌표, 구면극좌표), 3차원 좌표 등이 사용된다. 그러나 본 연구에서는 우리나라가 TM 좌표계를 사용하고 있고 또한 일반인들은 경위도좌표계를 사용하므로 이에 대한 것은 제공할 수 있게 하였고 또한 전세계 좌표에서 널리 사용하고 있는 UTM 좌표계를 나타내었다. 본 시스템에서 구현된 것에 대한 자세한 설명은 다음과 같다.

가. 경위도 좌표계(GEO)

지구상 절대적 위치를 표시하는데 일반적으로 가장 널리 쓰이는 좌표계이다. 3차원 구면좌표계에서는 구의 반경 ρ와 두 개의 편각 θ, π 로 구성되는 세 개의 실수 (ρ, θ, π)가 대응하여야 하지만 통상 지구좌표계에서는 경도 λ와 위도 Ψ에 의한 좌표(λ, Ψ)로 수평위치를 나타낸다. 따라서 3차원 취치를 표시하려면 타원체면으로부터 높이, 또는 표고를 도입할 필요가 있다.
지구표면상 자오선(경선)은 북극과 남극을 지나는 대원의 두극에서 끝나는 반원을 가리킨다. 기준반원인 본초자오선은 영국 그리니치 천문대를 지나는 자오선이며, 본초자오선과 적도의 교점을 원점으로 삼는다(경도, 0, 위도 0). 어느 지점의 경도는 본초자오선으로부터 적도를 따라 그 지점의 자오선까지 잰 최소각 거리로서 동, 서쪽으로 0도에서 180도까지 잰다(시스템공학연구소, 1996).

실제 시스템에서 구현된 위경도 좌표는 [그림 5-4]에 나타내었고 좌표변환 모드에 의해 위경도 좌표로 변화될 수 있다. 위경도 좌표는 GUI의 하단에 표시되어 있다.

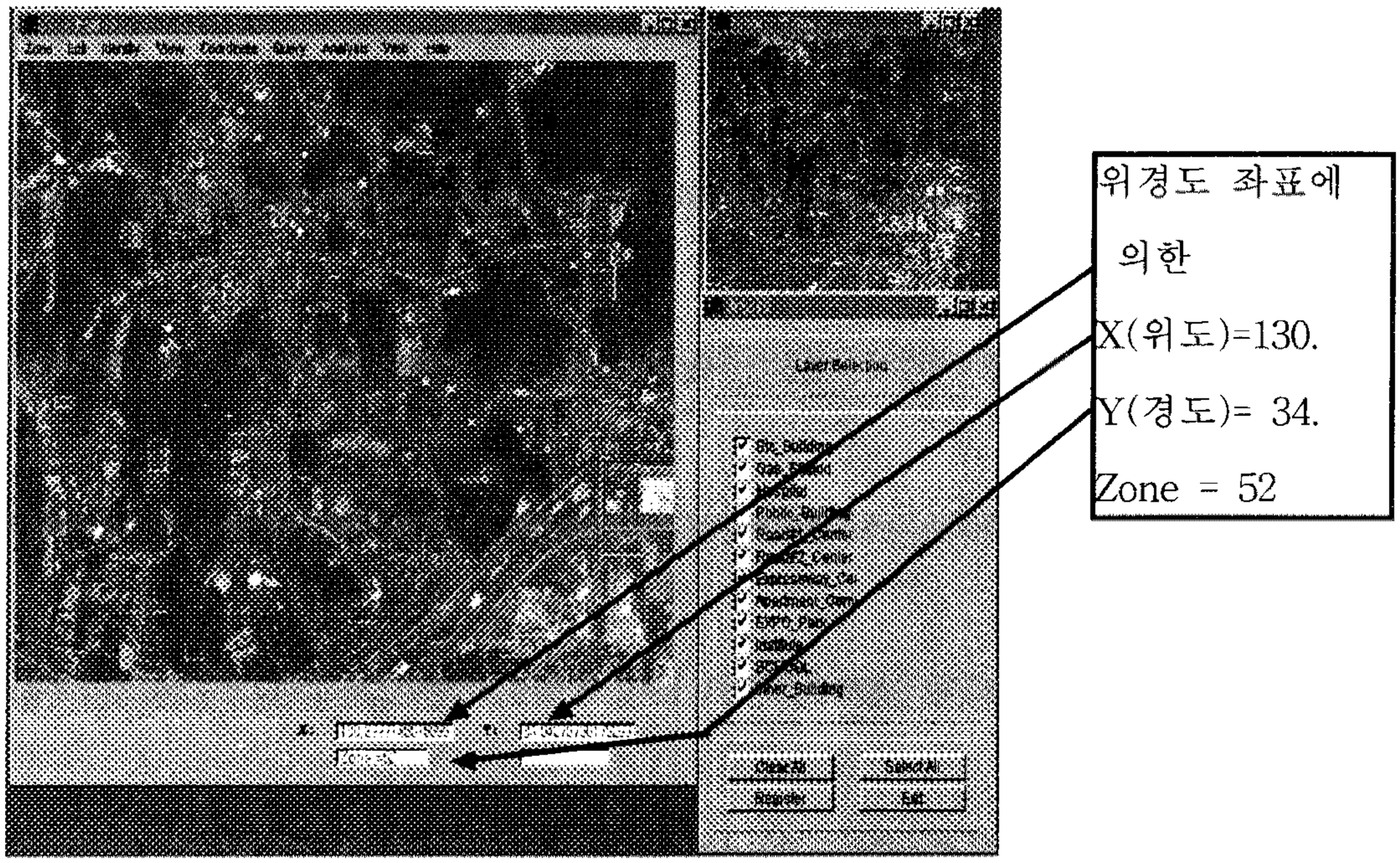


그림 5-4. GUI 상에 위경도 좌표값 표현

나. TM(Transverse Mercator Coordinate)

평면 직교 좌표계로서 측량범위가 넓지 않은 일반측량을 위하여 주로 사용되는 좌표계이다. 평면직각 좌표계에서는 측량지역에 1점을 택하여 좌표원점으로 정하고 그 평면상에서 원점을 지나는 경선을 X 축(북을 + 로 표시함), 동서방향을 Y축(동을 + 로 표시함)으로 하여 각 지점의 위치를 직각 좌표값 x, y로 표시한다. 이때 P1, P2의 좌표 값은 다음 식으로 표시된다.

$$x1 = S1 \cos T1 \qquad y1 = S1 \sin T1$$

$$x2 = x1 + S2 \cos T2 \qquad y2 = y1 + S2 \sin T2$$

여기서 S1, S2는 측선의 길이이고 T1, T2는 X축 방향(X′) 으로부터 측선 까지 오른쪽으로 관측한 수평각으로서 이것을 방향각(direction angle) 이라 한다. 다각측량에서는 x를 위거(attitude), y를 경거(departure)라 부르며 한점의 좌표는 위거와 경거의 합으로 구한다.

삼각측량과 같은 넓은 지역에서 위치를 정확하게 표시하려면 x축을 진북방향으로 위해야 하지만 다각측량에서는 임의의 방향을 x축으로 잡을 수 있다. 우리나라 전국에 대한 평면 직각좌표계에서는 서부원점(125도 E, 38도 N), 중부원점(127도 E, 38도 N), 동부원점(129도 E, 38도 N)의 3개의 원점을 사용한다. 측량원점에서는 경선(자오선)과 원점을 지나는 X 축 방향이 일치하지만 원점에서 동서로 멀어질수록 X축과 평행한 X′ 방향은 서로 일치하지 않고 차이가 생기게 된다. 즉, 진북 방향 N을 기준으로 한 진북방위각(α)와 방향각 T 사이에서는 진북방향각(∨) 만큼의 차이가 생긴다(시스템공학연구소, 1996).

본 시스템에 사용된 좌표변환은 [그림 5-5]과 같이 하단에 표시되게 된다. 초기 값으로 유입된 값이 TM으로 설정되었고 이들 값이 그림에서와 같이 숫자로 나타나게 된다.

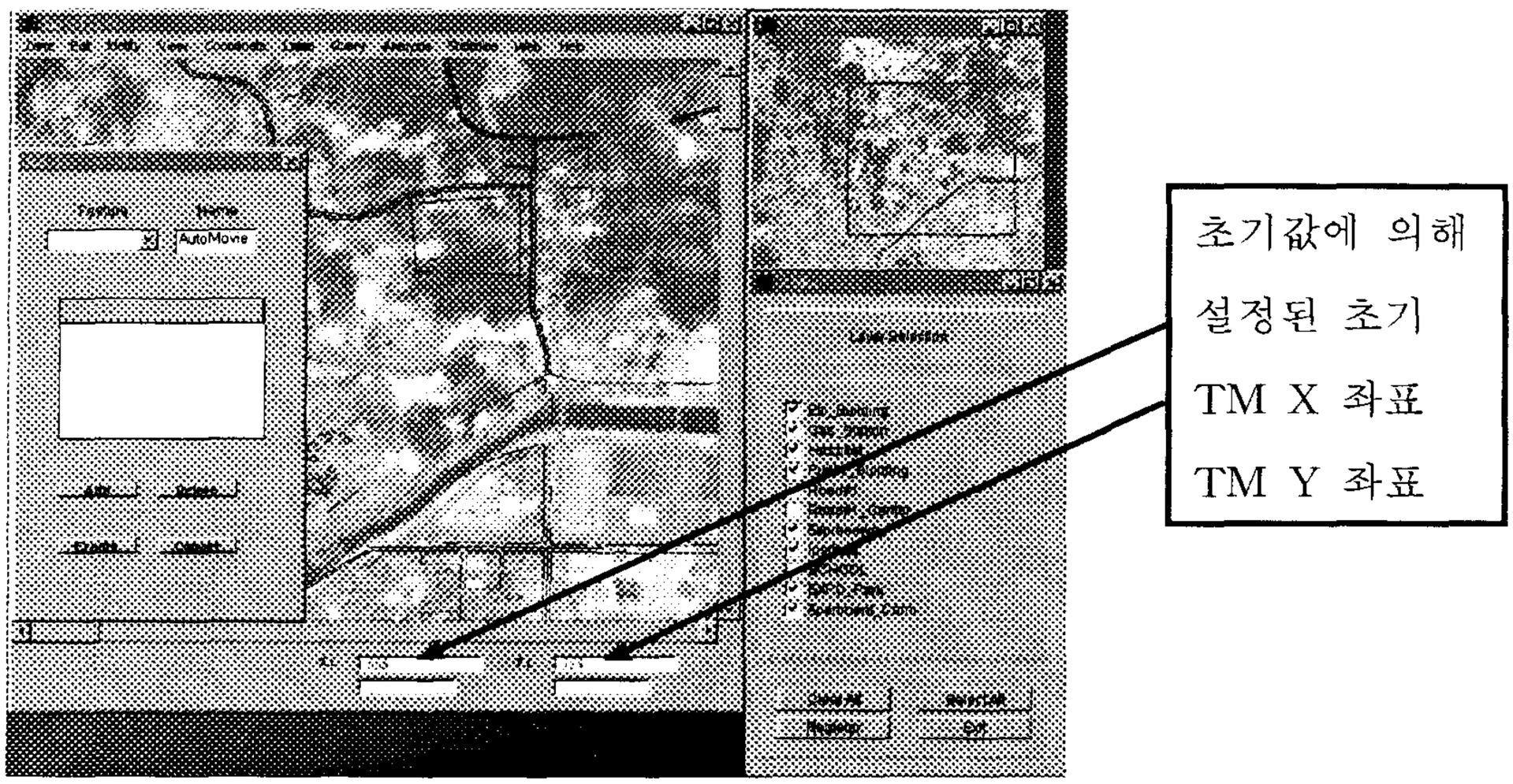


그림 5-5. GUI상에서 TM 좌표값의 표현

다. UTM(Universal Transverse Mercator Coordinate)

UTM(국제횡 Mercator) 투영법에 의하여 표현되는 좌표계로서 적도를 횡축, 자오선을 종축으로 한다. 이 방법은 지구를 회전타원체로 보고 지구전체를 경도 6도씩 60개 구역(종대, column)으로 나누고 그 각 종대의 중앙자오선과 적도의 교점으로 하여 원통도법인 횡 Mercator(TM) 투영법으로 등각투영 한다. 각 종대는 180도W 자오선에서 동쪽으로 6도 간격으로 1부터 60까지 번호를 붙인다. 종대에서 위도는 남.북에 80까지만 포함시키며 8도 간격으로 20구역(raw)으로 나누어 C(80도S ~ 72도S)에서 X(72도N ~ 80도N)까지(단, I와 O는 제외) 알파벳 문자로 표시한다. 따라서 종대 및 횡대는 결국 경도 6도 x 위도8도의 직사각형 구역으로 구분된다(시스템공학연구소, 1996).

[그림 5-6]은 UTM 변환하여 GUI상에서 숫자로 나타낸 것이다. 여기서

존(Zone=52)와 중심선(Center=129)을 나타내게 되어 있다.

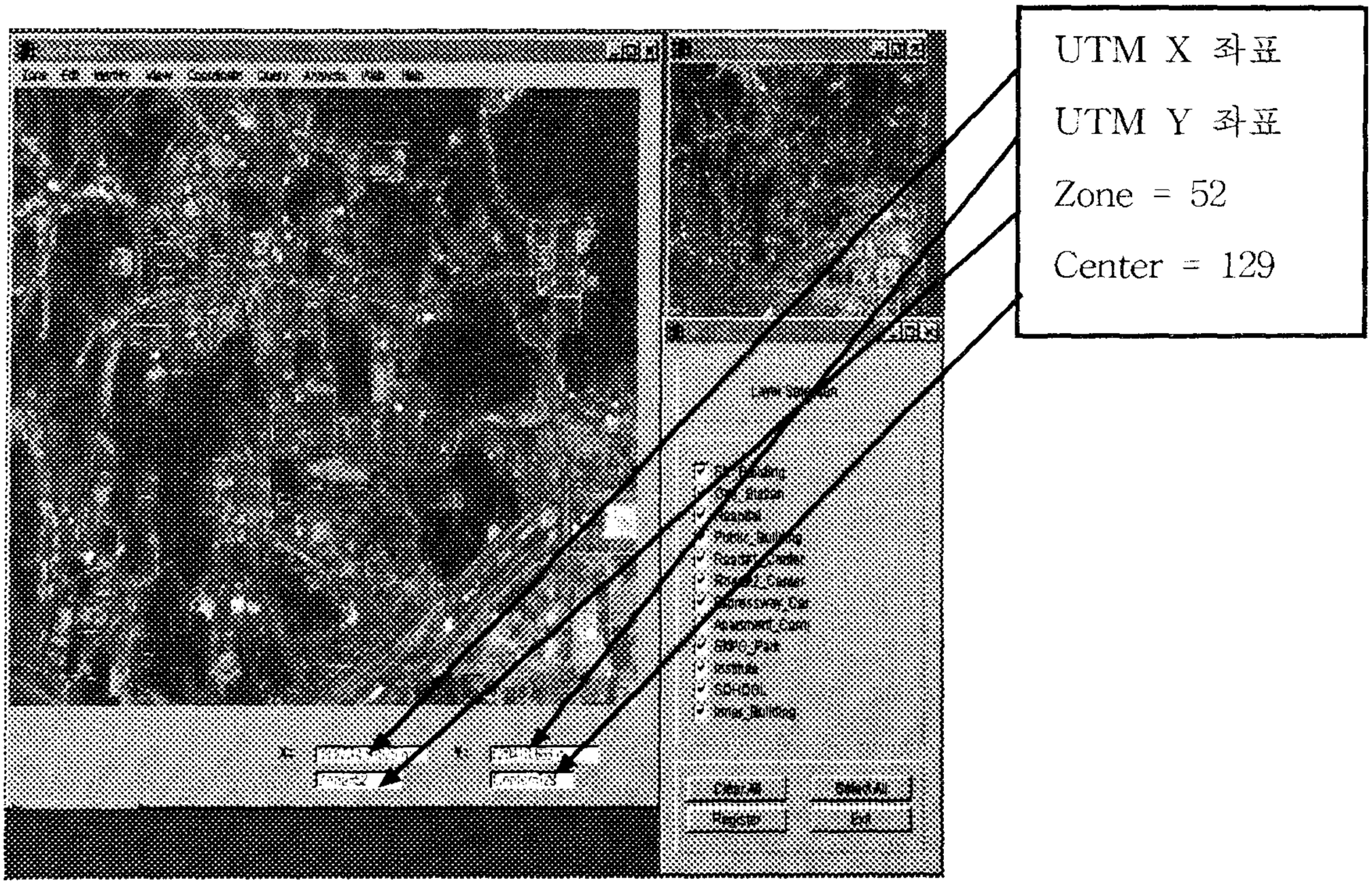


그림 5-6. GUI 상에 UTM 좌표값의 표현

4. 홈페이지 구축

본 시스템은 최종적으로 [그림 5-7]과 같이 홈페이지에 구축되어 인터넷을 통하여 사용자들이 원격지에서도 지리정보를 쉽고 빠르게 받아 볼 수 있게 하기 위함이다. 그러기 위해서 본 연구에서는 본 연구실의 시스템에 한시적으로 서버를 구축하였다. 구축된 내용은 [표 5-2]에 나타내었다.

표 5-2. 홈페이지에 구축된 내용

문서	내용
head.html	2 및 3 차원 지리정보시스템 페이지
3DGIS.html	3차원 지리정보시스템의 소개에 관한 문서
wgis.html	웹 지리정보시스템에 대한 소개 문서
paper.html	제출된 논문 요약
event.html	행사안내 문서
qanda.html	질문 및 답변에 관한 문서
TwoDGIS.html	2차원 지리정보시스템 페이지
ThreeDGIS.html	3차원 지리정보시스템 페이지

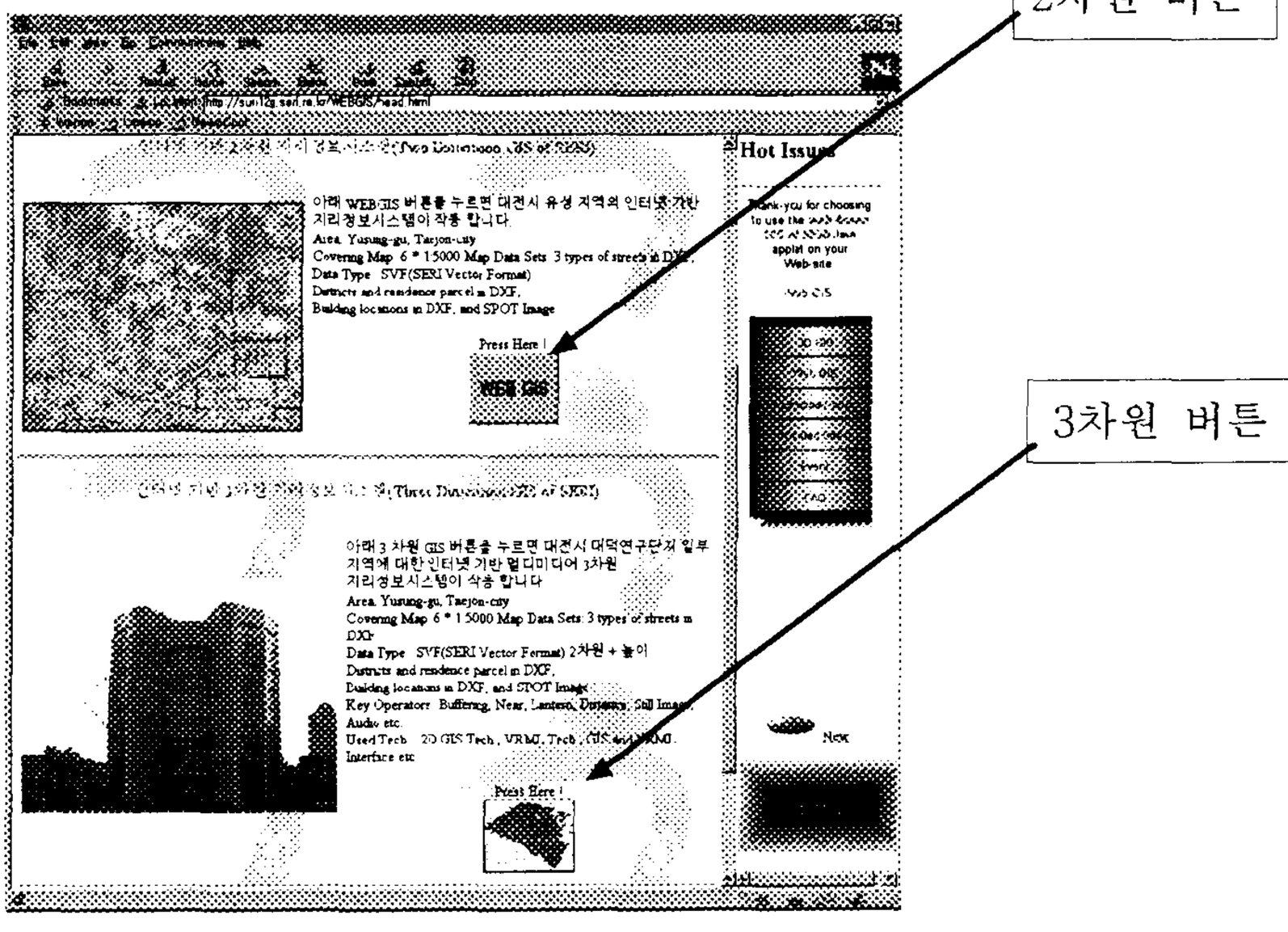


그림 5-7. 2차원 및 3차원 지리정보 메인 홈페이지(head.html)

가. 3차원 GIS 페이지(3DGIS.html)에 소개되는 내용

본 페이지에 포함된 내용은 3차원에 관한 공간분석, 색인 방법론, 연산자 및 디스플레이 등의 내용 등이 포함되어야 한다[그림 5-8]. 구체적인 내용은 웹 상에서 3차원 공간분석(3D Spatial Analysis on Web), 2. 3차원 공간 색인 방법론(3D Spatial Indexing Methodology), 3차원 연산자(3D Spatial Operators), 자

료 질의(Data Query), VRML(Virtual Reality Modeling Language)을 이용한 3차원 디스플레이(3D Display using VRML, 네트워크 상에서 통계 기능 및 벡터 오버레이 기능(Statistical Function and Vector Overlay Functions on Network) 및 객체지향(Object Orient) 등이다.

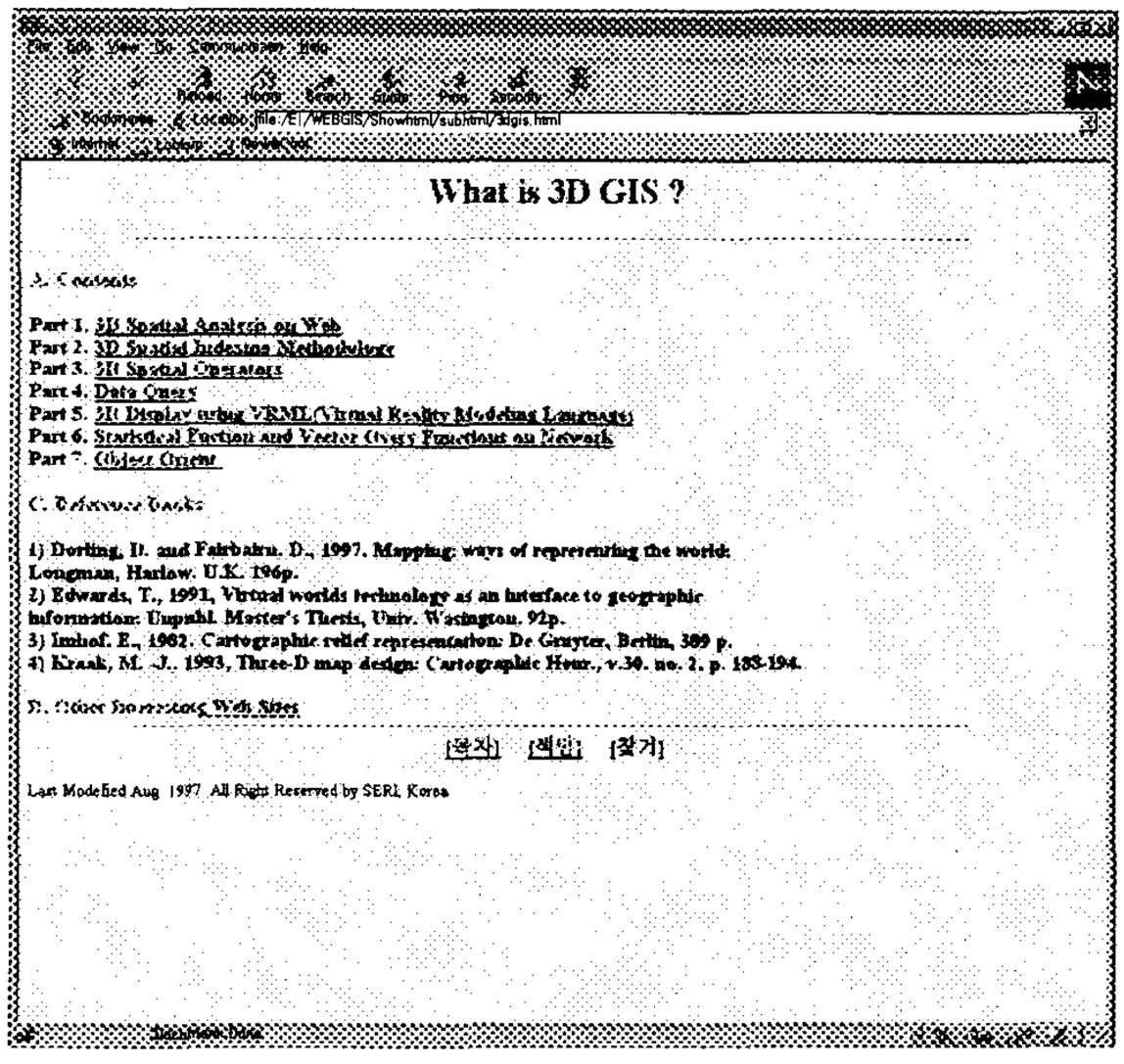


그림 5-8. 3차원 GIS 에 관련된 문서에 대한 페이지

나. 웹 지리정보시스템 페이지(wgis.html)에서 소개되는 내용

본 페이지에 포함된 내용은 웹 지리정보시스템에 대한 기술(Description of the Web GIS)이다. 구체적으로 지리정보시스템이란 무엇인가와 웹기반 지리정보시스템의 GUI에 대한 소개로써 이들에 대한 정보를 소개하는 페이지로써 이들에 대한 간단한 기술이 있다[그림 5-9].

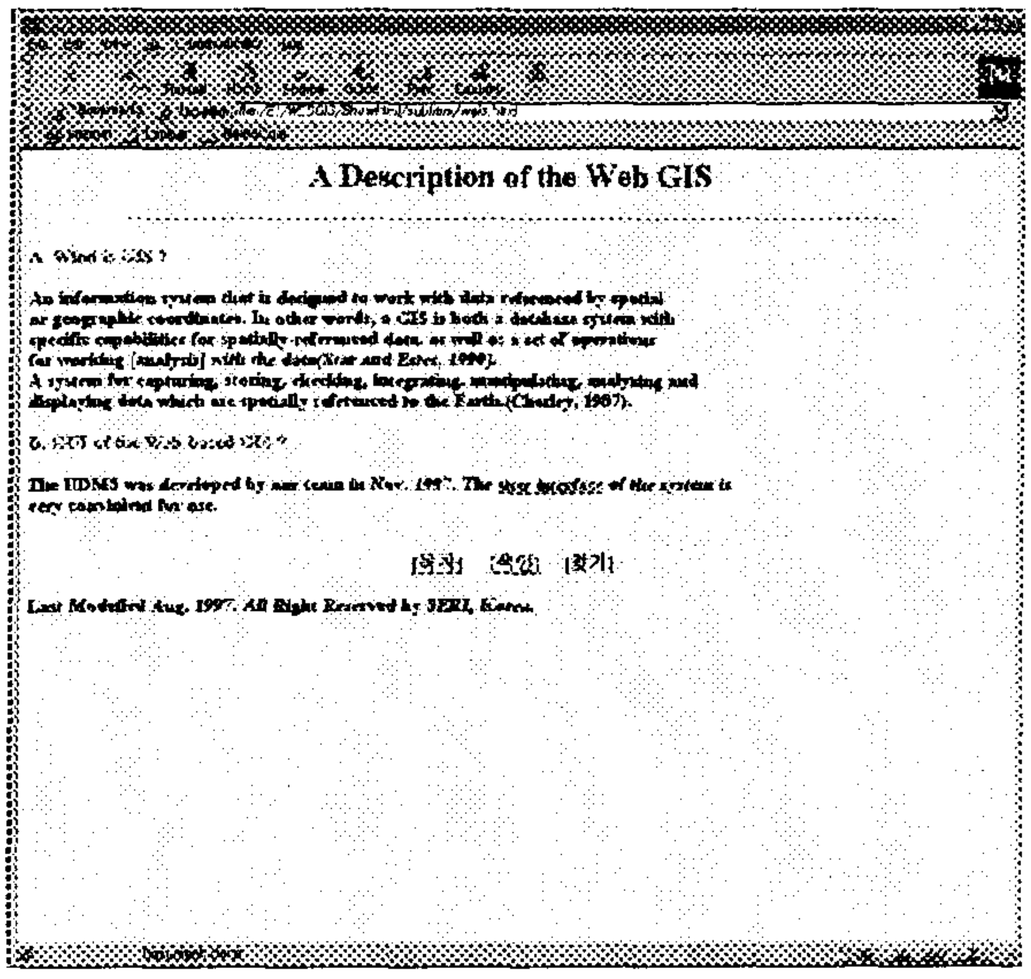


그림 5-9. Web GIS 에 대한 서술 내용이 있는 페이지(wgis.html)

다. 논문 페이지(paper.html)에서 소개되는 내용

본 페이지에 포함된 내용은 본 연구과제에서 발표된 논문을 정리하여 초록만을 나타낸 것으로 그리고 앞으로 개최될 학회에 관한 논문을 소개하는 페이지이다. 구체적으로 정보과학회, 정보처리학회, 대한원격탐사학회 및 GIS/LIS'97 프로시딩에 제출된 논문을 기술하여 놓았다[그림 5-10].

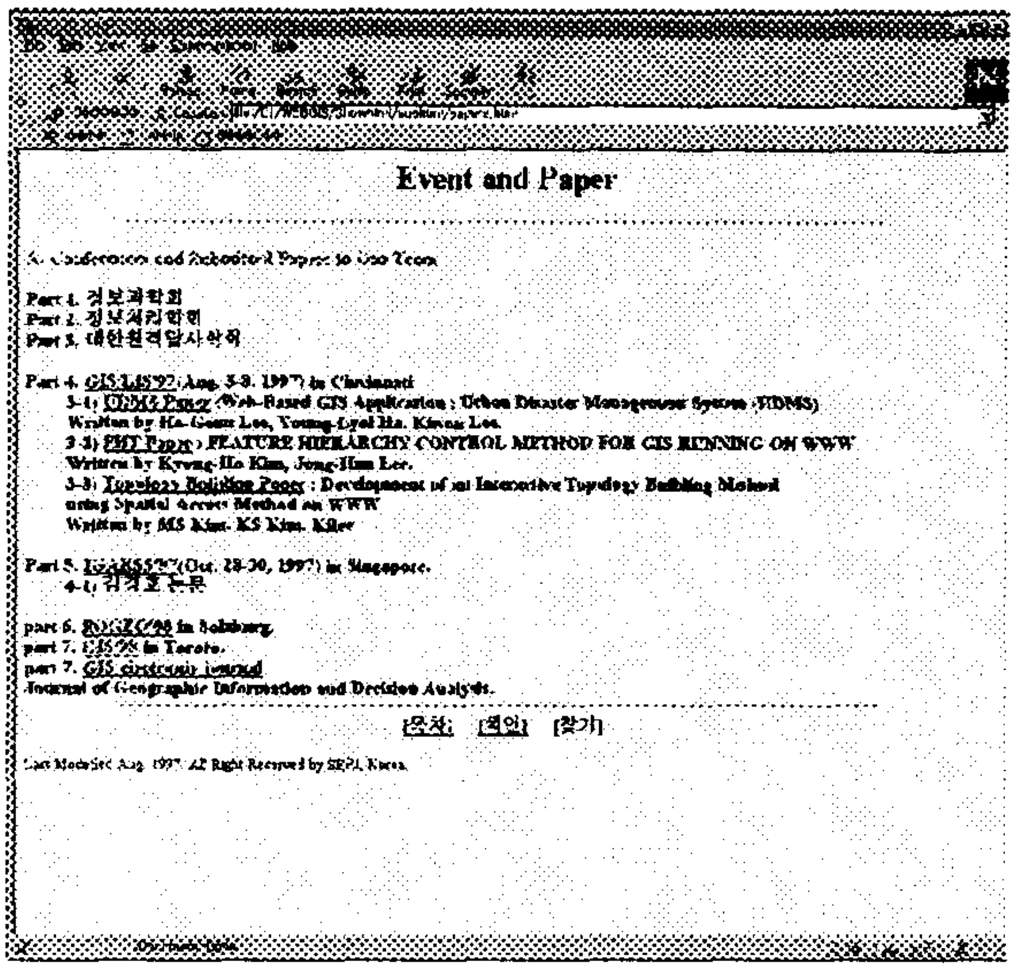


그림 5-10. 학회 행사 및 관련된 학회에서 발표된 논문 페이지

라. 행사 페이지(event.html)에서 소개되는 내용

본 페이지에서는 웹 지리정보시스템과 관련된 각종행사 및 학회에 대한 정보를 알려주는 것으로 현재까지는 정보과학회, 정보처리학회, 대한원격탐사학회, GIS/LIS'97, IGARSS'97, EOGEO'98, GIS electronic 저널, Journal of Geographic Information and Decision Anaysis 가 포함되어 있다[그림 5-11].

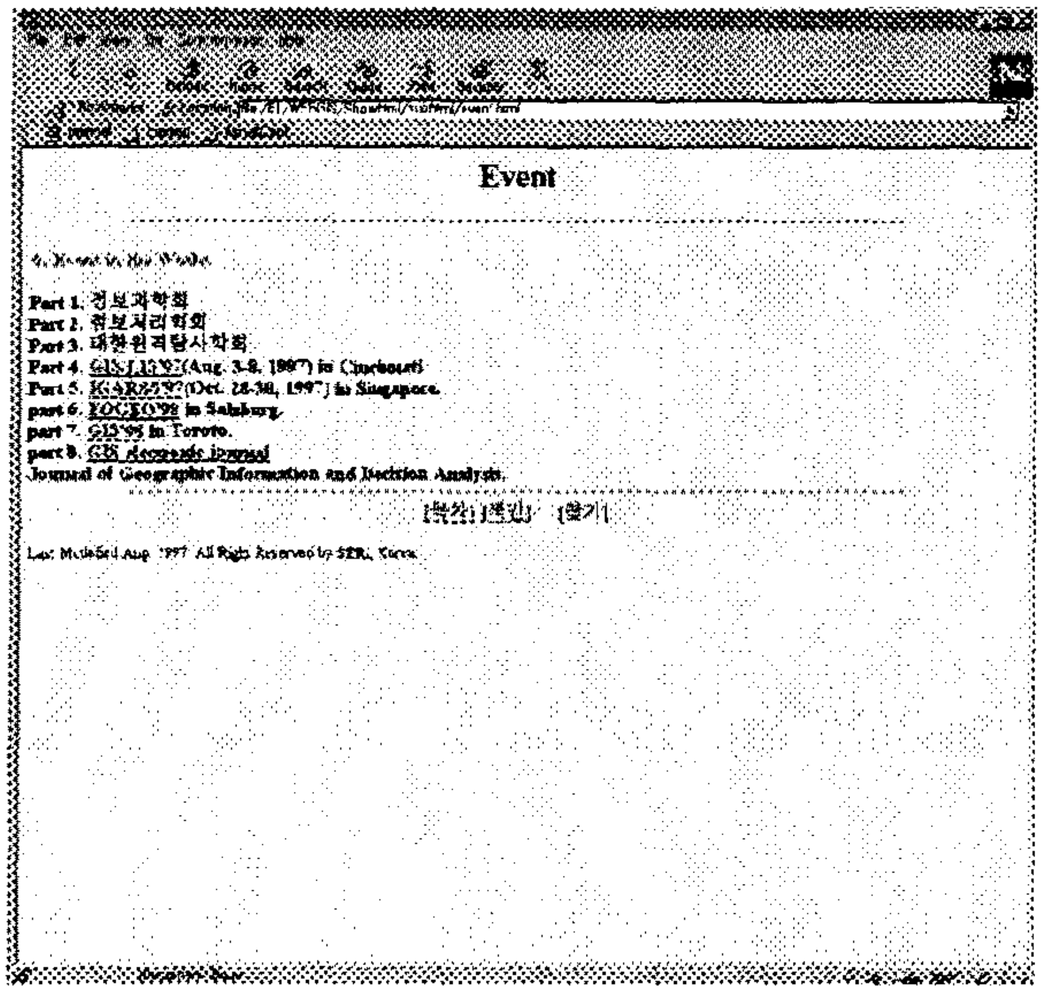


그림 5-11. 행사에 관련된 내용만을 모아둔 페이지(envent.html)

제 2 절 AWT(Abstract Window Toolkit)를 이용한 사용자 작업 환경 구현

1. 사용자 인터페이스의 설계 (GUI Design)

본 장에서는 클라이언트 시스템에서 사용자 작업 환경의 설계 및 구성 그리고 사용자 인터페이스간의 연관 관계에 대해서 알아본다. 현재 본 시스템의 사용자 인터페이스는 JDK 1.14 버전으로 구현되었으며, Semantec사의 Visual Cafe 1.1을 이용하여 만들어 졌다.

먼저 본 시스템은 Web상에서 동작하므로 자바 애플릿으로 구성된 Web 상에서의 인터페이스에 대하여 알아본 다음에, 전체적인 사용자 인터페이스의 구성에 대하여 먼저 알아보고 사용자 인터페이스의 핵심을 이루는 Project, Zone, 그리고 Layer 개념 및 구현에 대해서 알아본다.

가. Web 인터페이스

기본적으로 웹 브라우저사에서 자바 프로그램을 수행시키기 위해서는 자바 애플릿(Applet) 프로그램을 이용해야 한다. 그러므로 본 시스템은 클라이언트 시스템의 사용자 인터페이스를 애플릿을 기본으로 하여 구성하였는데, 웹 브라우저상에서의 화면은 다음 [그림 5-12]과 같다.

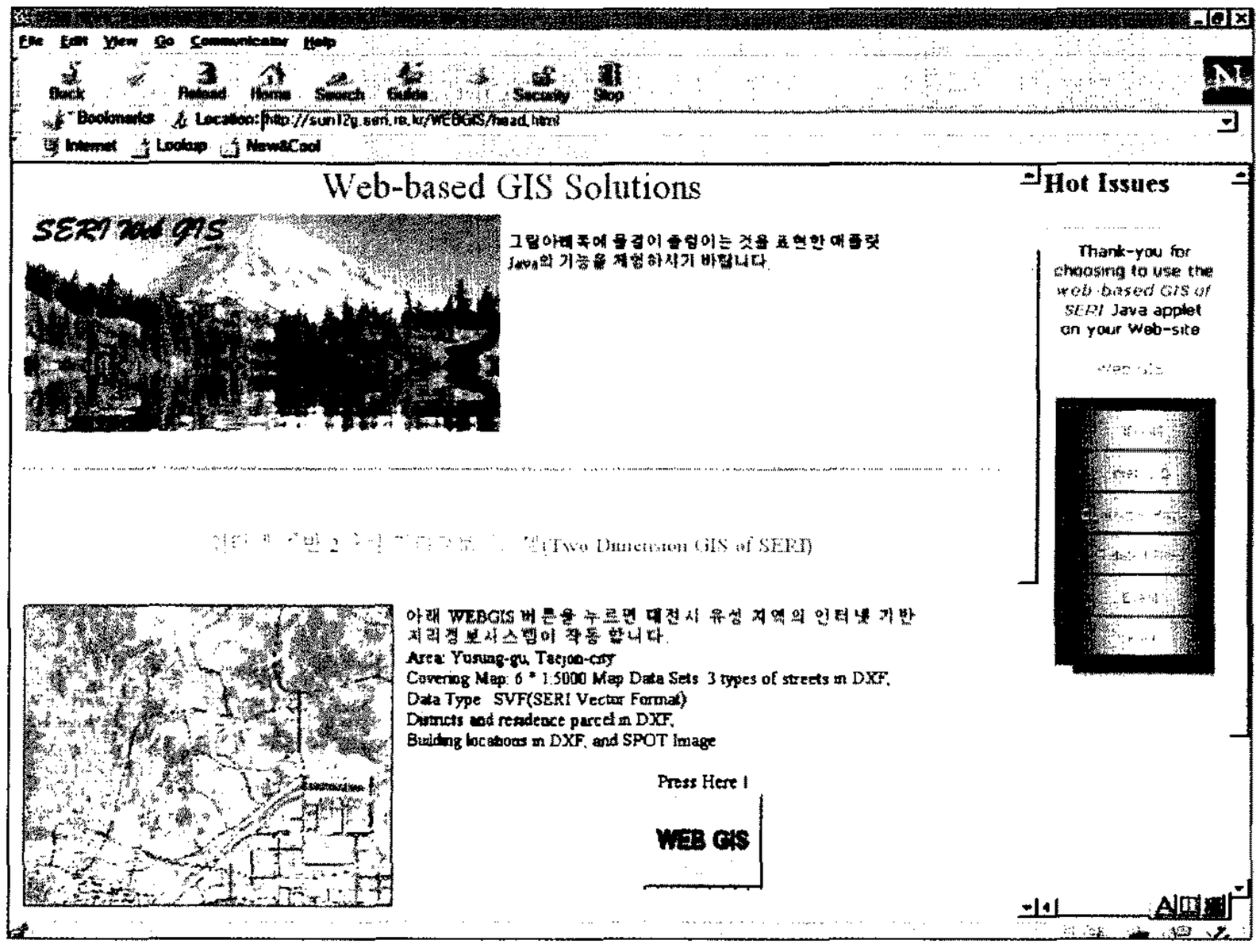


그림 5-12. 웹 브라우저상에서의 사용자 인터페이스

위의 [그림 5-12]에서 보면 'WEB GIS'라는 자바 애플릿으로 만들어진 버튼이 있는데, 이 버튼을 클릭하게 되면 본 시스템의 메인 메뉴가 로딩되도록 구성되어 있다. 본 시스템의 메인 메뉴는 프로젝트(Project) 메뉴로서 풀 다운(Pull Down)형식의 메뉴인데, 이를 위해서 이 메뉴는 자바에서 제공하는 프레임(Frame) 클래스를 이용하여 구현되었다. 다음 [그림 5-13]의 소스 코드는 'WEG GIS'의 자바 애플릭 버튼을 클릭하였을 때 수행되는 과정을 보여주고 있다.

```
// WebGISButton 클래스

import java.awt.*;
import java.applet.*;
import java.net.*;

public class WebGISButton extends Applet { // 애플릿 프로그램
    java.awt.Frame AttributeFrame;
    public String  server = null;                  // 서버 이름
    public String  projectName = null;             // 프로젝트 이름

        // WEB GIS 버튼 클릭 이벤트 처리 메소드
        void GISButton_Clicked(java.awt.event.ActionEvent event) throws Exception {
            URL url = getCodeBase();
            // 메인 메뉴 생성
            ProjectAreaFrame pjFrame = new ProjectAreaFrame(this);
            pjFrame.show();
        }

        java.awt.Button GISButton;
        // WEG GIS 버튼의 이벤트 발생 여부 조사
        class Action implements java.awt.event.ActionListener {
                public voi└d actionPerformed(java.awt.event.ActionEvent event) {
                        Object object = event.getSource();
                        if (object == GISButton){
                            try {
                                GISButton_Clicked(event);
                            } catch(Exception e) {
                            }
                        }
                }
        }
}
```

그림 5-13. WEB GIS 버튼의 이벤트 수행과정

위의 그림에서 보듯이 'WEB GIS' 버튼의 클릭 이벤트 발생시 pjFrame이라는 ProjectAreaFrame을 수행시키는데, 이는 클라이언트에서 웹 GIS를 실행시키기 위한 메인 메뉴를 동작 시키는 것이다.

나. 전체적인 사용자 인터페이스의 구성

본 시스템의 주요 사용자 인터페이스는 웹 브라우저의 'WEB GIS' 버튼의 실행 이후에 구성되어지는데 먼저 전체적인 사용자 인터페이스의 구성도를 보면 다음 [그림 5-14]과 같다.

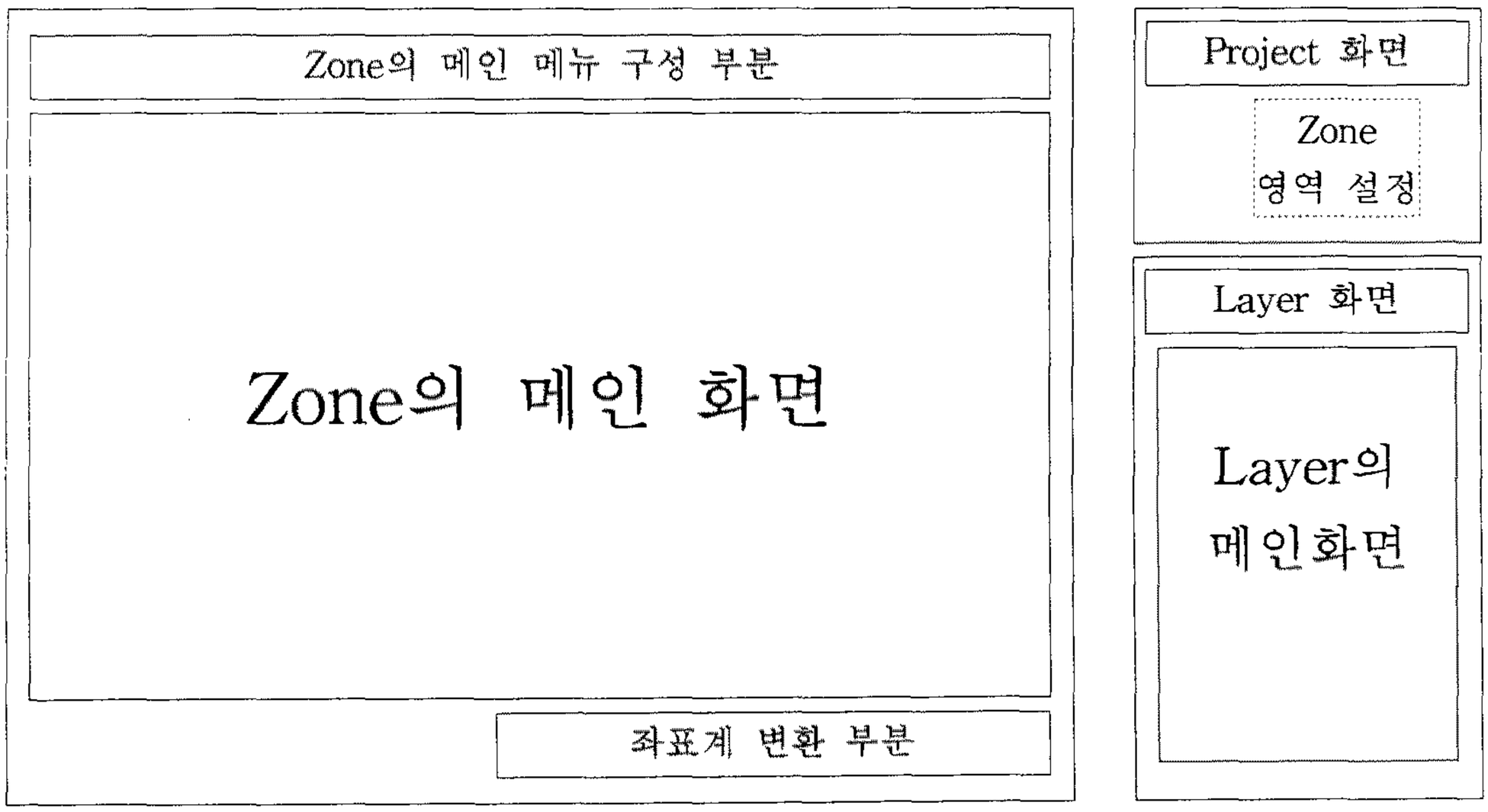


그림 5-14. 전체적인 사용자 인터페이스 구성도

위의 [그림 5-14]의 전체 인터페이스 구성을 크게 나누어 보면 다음과 같은 세 가지의 인터페이스로 나누어짐을 알 수 있다.

- 프로젝트(Project) 인터페이스

. 데이터베이스내에 구축된 데이터의 전체 영역을 나타내는데, 다시 말하면 일반적인 Map Extent의 영역을 포함하는 영역이다.

. Project 영역에서 임의의 작업하고자 원하는 영역을 선택함으로써 Zone 영역이 생성된다.

. Zone 영역이 생성되고 나면, Project 영역은 화면이 축소되고 인덱스 맵(Index Map)으로 변화한다.

. 인덱스 맵에서는 Zone 영역을 임의로 이동할 수 있다.

- 존(Zone) 인터페이스

. Project 영역에서 임의로 작업하고자 원하는 영역을 선택하였을 때 생성된다.

. 공간 분석 및 공간 검색, 그리고 질의 처리등의 모든 작업이 이루어진다.

- 레이어(Layer) 인터페이스

. Zone 영역에서 작업하고자 원하는 피쳐(Feature) 집합들을 선택할 수 있다.

. 원하는 피쳐만을 로딩시키므로 시스템의 전체 속도면에서 효율적이다.

위에서 간단히 설명하였듯이 이들 Project, Zone, Layer 인터페이스는 본 시스템의 사용자 작업 환경을 구현하는데 있어서 가장 기본이 되는 것으로, 상호간의 관계를 그림으로 나타내면 다음 [그림 5-15]와 같다.

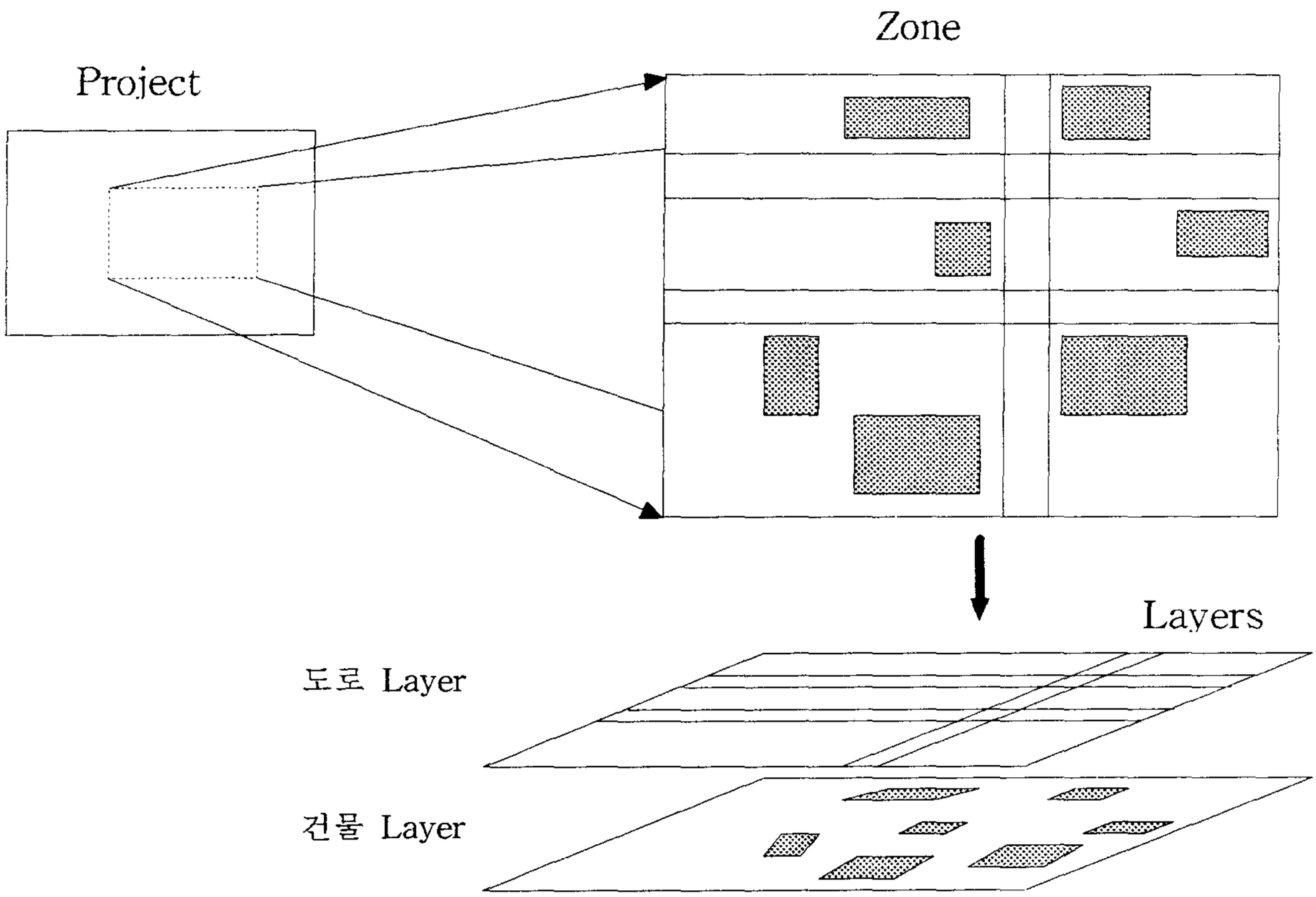


그림 5-15. Project, Zone, Layer간의 상호 관계

위의 [그림 5-15]에서 보면 Project 인터페이스에서는 점선으로 구성된 임의의 영역이 설정되어 있는데, 이 영역이 바로 Zone 영역을 구성한다. 그리고 Zone 내부에는 도로와 건물의 두 Layer가 보이고 있다. 클라이언트 시스템에서 Project와 Zone, 그리고 Layer가 실제로 구현된 화면은 다음 [그림 5-16]과 같다.

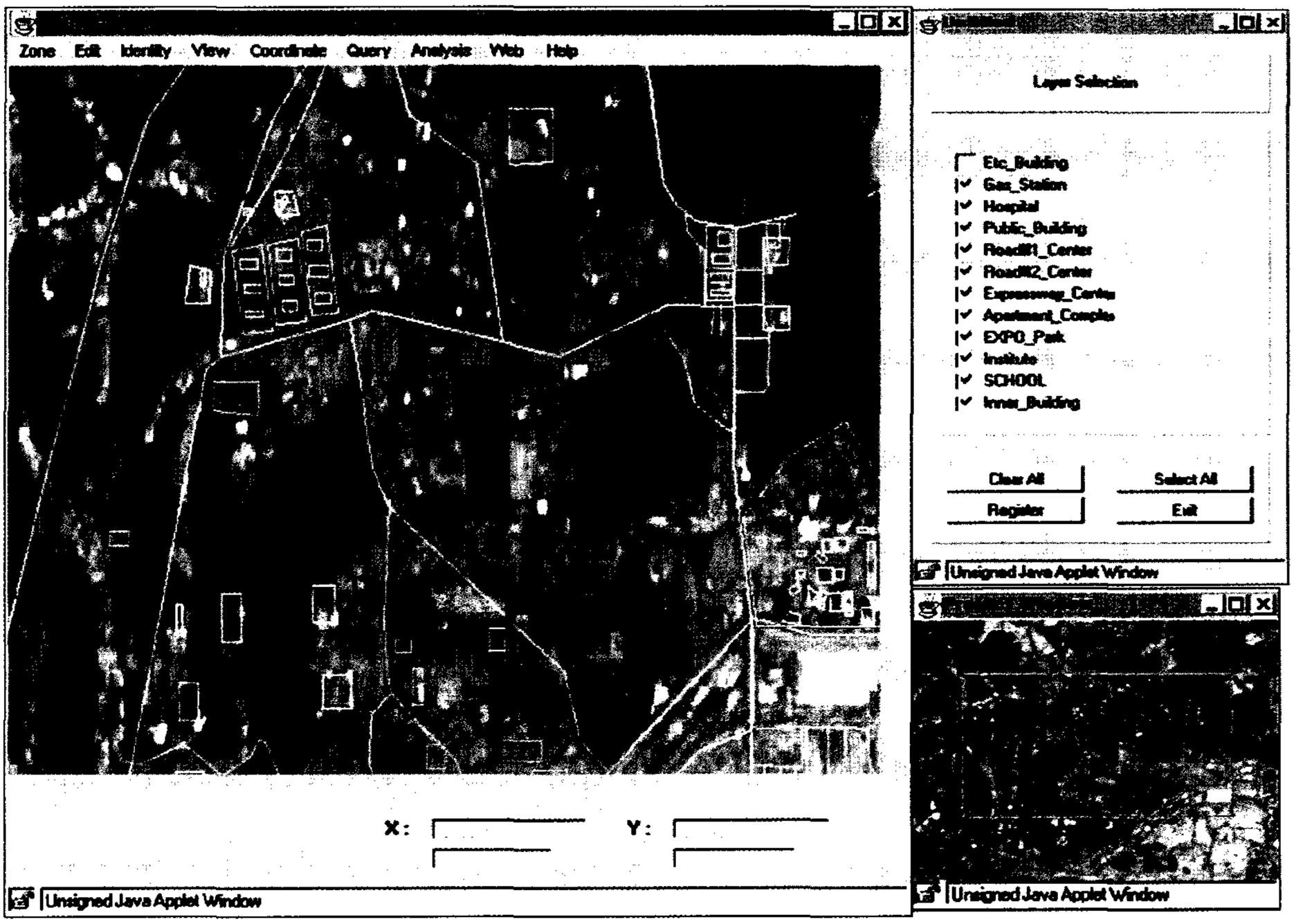


그림 5-16. 클라이언트 시스템에서의 Project, Zone, Layer 구현 예

지금까지 웹 브라우저에서 실제로 본 시스템을 동작시켰을 때 시작 단계에서 이용되어지는 전체적인 사용자 작업 환경에 대하여 알아보았다. 이를 요약해 보면 다음과 같이 정리될 수 있다.

- 웹 브라우저상에서 동작하도록 자바 애플릿으로 구성되어 있다.
- 풀 다운 형식의 메뉴를 지원하기 위해서 프레임을 이용한다.
- Project, Zone, Layer의 인터페이스가 상호 연관 관계를 가지고 동작되도록 구성되어 있다.

다. 프로젝트(Project) 인터페이스

Project 인터페이스는 크게 두 개의 클래스로 구성되어 있는데, Project 클래스와 GISFrame 클래스이다. 먼저 Project 클래스는 데이터베이스와의 연결을 설정하여 데이터를 주고 받는 역할이나 공간 분석등의 필요한 연산을 처리하는 부분이다. 이에 비해 GISFrame 클래스는 사용자 인터페이스 부분으로서 Project 클래스에서 유지하고 있는 데이터들을 출력하는 역할을 수행한다.

1) Project 클래스

이 클래스는 데이터베이스로부터 불러오는 정보들을 유지하고 있으며 사용자 작업환경을 구성하는데 있어서 필요한 부가적인 정보들을 유지하고 있다. Project 클래스는 다음과 같은 정보들을 유지하고 있다.[그림 5-17]

```
//------------------------------ Project 클래스 정의 ------------------------------

public class Project {
        // 디스플레이 정보
        private int                 minX, minY, maxX, maxY; //  Project 전체 영역 설정
        private int                 displayX, displayY;          // 실제 화면 크기
        private int                 diffX, diffY;                // 원점으로 평행이동시 이용
        private int                 zminX, zminY, zmaxX, zmaxY; // Zone 전체 영역 설정
        private String              projectName;                 // Project 이름

        // 데이터 관련 정보
        private NodeFeatureList     nodeFList;                   // 노드 피쳐 데이터
        private ChainFeatureList    chainFList;                  // 체인 피쳐 데이터
        private PolygonFeatureList  polyFList;                   // 폴리곤 피쳐 데이터
        private SERINodeList        nodeList;                    // 노드 데이터
```

```
    private SERIChainList       chainList;              // 체인 데이터
    private SERIPolygonList     polyList;               // 폴리곤 데이터
    private FDTList             fDefTable;              // 피쳐 정의 테이블 유지
    private String[]            fNames;                 // 선택된 피쳐 이름 리스트

    // 데이터베이스 접속을 위한 변수
    private DBProtocol          dbchannel;              // 브로커와의 연결 변수
    private String              serverName;             // 데이터베이스 서버 이름
}
```

그림 5-17. Project 클래스 정의

위의 그림을 보면 Project의 클래스는 크게 디스플레이 관련 정보, 데이터 관련 정보, 그리고 데이터베이스 접속을 위한 변수로 나누어져 있다. 먼저 디스플레이 관련 정보를 보면 다음과 같다.

- minX, minY, maxX, maxY : 데이터베이스내에 존재하는 모든 데이터 집합을 포함하는 Project의 영역을 설정한다.
- displayX, displayY : 이 값들은 실제 화면상에서 X축과 Y축의 크기를 말한다. 실제 좌표에서 화면상의 좌표로 변환할 때 사용된다.
- diffX, diffY : 실제 데이터 집합의 좌표는 원점에서 시작되지 않으므로 실제 데이터 집합들을 원점으로 평행이동 시키기 위해서 사용되는 값이다.
- zminX, zminY, zmaxX, zmaxY : Project에서 임의로 선택되는 Zone의 영역을 설정한다.
- projectName : 현재 수행되고 있는 Project의 이름 정보를 유지하고 있다.

데이터 관련 정보는 다음과 같다.

- nodeFList, ChainFList, PolyFList : 데이터베이스에서 불러오는 노드, 체인, 폴리곤 피쳐 정보를 저장한다.
- nodeList, chainList, polyList : 데이터베이스내의 노드, 체인, 폴리곤의 속성 정보를 제외한 공간 정보만을 저장한다.
- fDefTable : 데이터베이스내의 피쳐 정의 테이블의 내용을 저장하기 위해서 사용된다.
- fNames : 현재 Project에서 선택되어진 피쳐 이름들을 저장하기 위해서 사용된다.

끝으로 브로커와 데이터베이스와의 연결 통로를 위하여 브로커와의 연결을 위한 변수인 dbchannel과 데이터베이스 서버이름을 유지하고 있다.

위와 같이 정의된 Project 클래스는 디스플레이 및 데이터베이스 접속을 위하여 많은 메소드 리스트를 제공하는데 다음 [그림 5-18]에 나타나 있다.

```
//---------------------- Project 클래스 인터페이스 메소드-----------------------
//
//  public Project (String server)
//  public void SetMbr(int x1, int y1, int x2, int y2)
//  public void SetZoneMbr(int x1, int y1, int x2, int y2)
//  public void SetProjectName(String project)
//  public void ServerName(String server)
//  public void SetNodeFeatureList(NodeFeatureList list)
//  public void SetChainFeatureList(ChainFeatureList list)
//  public void SetPolygonFeatureList(PolygonFeatureList list)
//  public void SetFDTList(FDTList table)
//  public void SetDBProtocol(DBProtocol db)
//  public void SetFNames(String[] name)
//  public int GetMinX(), public int GetMinY(), public int GetMaxX(), public int GetMaxY()
//  public int GetZMinX(),public int GetZMinY(),public int GetZMaxX(), public int GetZMaxY()
//  public String GetProjectName()
//  public String GetServerName()
//  public NodeFeatureList GetNodeFeatureList()
//  public ChainFeatureList GetChainFeatureList()
//  public PolygonFeatureList GetPolygonFeatureList()
//  public FDTList GetFDTList()
//  public DBProtocol GetDBProtocol()
//  public String GetFName(int i)
//  public int GetFeatureType(String fName)
//  public NodeFeatureList LoadNodeFeature(String fName)
//  public ChainFeatureList LoadChainFeature(String fName)
//  public PolygonFeatureList LoadPolygonFeature(String fName)
//  public NodeFeatureList LoadNodeFeature(String fName, int x1, int y1, int x2, int y2)
//  public ChainFeatureList LoadChainFeature(String fName, int x1, int y1, int x2, int y2)
//  public PolygonFeatureList LoadPolygonFeature(String fName, int x1, int y1, int x2, int y2)
//  public String MakeClassNameCondition(String fName)
//  public String[] CopySelectedFNames()
//  public void SetDisplay(int x, int y)
//  public int GetDisplayX(), public int GetDisplayY()
//  public void SetDiff(int x, int y)
```

```
//  public viod GetDiffX(), public void GetDiffY()
//  public int RealToDisplayX(int realCord), public int RealToDisplayY(int realCord)
//  public int DisplayToRealX(int displayCord), public int DisplayToRealY(int displayCord)
//  public SERINodeList GetNodeList()
//  public SERIChainList GetChainList(),
//  public SERIPolygonList GetPolygonList()
//  public void MakeNodeList(NodeFeatureList nfList)
//  public void MakeChainList(ChainFeatureList cfList)
//  public void MakePolygonList (PolygonFeatureList pfList)
//  public void InitialNodeList(), public void InitialChainList(), public void InitialPolygonList()
//-----------------------------------------------------------------
```

그림 5-18. Project 클래스 인터페이스 메소드 리스트

- Project(String server) 메소드 : 디스플레이 및 데이터베이스와의 연결을 위한 Project 객체를 생성한다.

- SetMbr(int x1, int y1, int x2, int y2), SetZoneMbr(int x1, int y1, int x2, int y2) : Project 영역과 Zone 영역의 범위를 설정한다. 실제 좌표 값을 이용하여 설정한다.

- SetProjectName(String project) : Project의 이름을 설정한다. 이 Project의 이름은 데이터베이스내의 테이블 생성시에 이용된다.

- ServerName(String server) : 데이터베이스 서버의 이름을 설정한다. 디폴트로 주어진 서버 주소에 데이터베이스 서버가 없는 경우 새로이 서버 이름을 설정한다.

- SetNodeFeatureList(NodeFeatureList list),
  SetChainFeatureList(ChainFeatureList list),
  SetPolygonFeatureList(PolygonFeatureList list) : 노드, 체인, 폴리곤 피쳐 데이터 리스트를 설정한다. 데이터베이스로부터 추출된 데이터 리스트를 Project 클래스내에 유지할 수 있다.

- SetFDTList(FDTList table) : 데이터베이스로부터 읽어오는 피쳐 정의 테이블의 내용을 Project 클래스 내에 유지 시킨다.

- SetDBProtocol(DBProtocol db) : 브로커 시스템과 연결하기 위한 변수를 설정한다.

- SetFNames(String[] name) : Project 클래스내에 선택된 피쳐 이름들을 설정한다.

- GetMinX(), GetMinY(), GetMaxX(), GetMaxY(), GetZMinX(), GetZMinY(), GetZMaxX(), GetZMaxY() : Project 와 Zone의 영역 값을 반환한다.

- GetProjectName(), GetServerName() : Project와 Server 이름을 반환한다.

- GetNodeFeatureList(), GetChainFeatureList(),
  GetPolygonFeatureList() : Project 클래스내에 저장되어 있는 노드, 체인, 폴리곤 피쳐 데이터 리스트를 반환한다.

- GetFDTList() : Project 클래스내의 피쳐 정의 테이블을 반환하는데, 이 피쳐 정의 테이블은 데이터베이스로부터 읽어오는 것이다.

- GetFeatureType(String fName) : 입력으로 주어진 피쳐 이름에 대하여 그 피쳐의 타입(노드, 체인, 폴리곤)을 반환한다. 이 메소드는 피쳐 이름만 주어지고 데이터베이스 테이블을 접근하고자 할 때 테이블 이름을 알기 위해서 많이 사용된다.

- LoadNodeFeature(String fName), LoadChainFeature(String fName), LoadPoygonFeature(String fName) : 입력으로 주어진 피쳐 이름을 가지고 그 피쳐에 해당하는 노드, 체인, 폴리곤 피쳐 데이터를 데이터베이스로부터 읽어온다. 데이터 로딩시에 주로 사용되는 메소드이다.

- LoadNodeFeature(String fName, int x1, int y1, int x2, int y2), LoadChainFeature(String fName, int x1, int y1, int x2, int y2), LoadPolygonFeature(String fName, int x1, int y1, int x2, int y2) : 위의 메소드와 같은 기능을 수행하는데 한 가지 다른 점은 입력 인자로 피쳐 이름과 영역이 같이 주어짐으로써 영역내에 존재하는 피쳐 데이터를 읽어오는 역할을 수행한다. 데이터베이스내에서 공간 검색이 같이 수행된다.

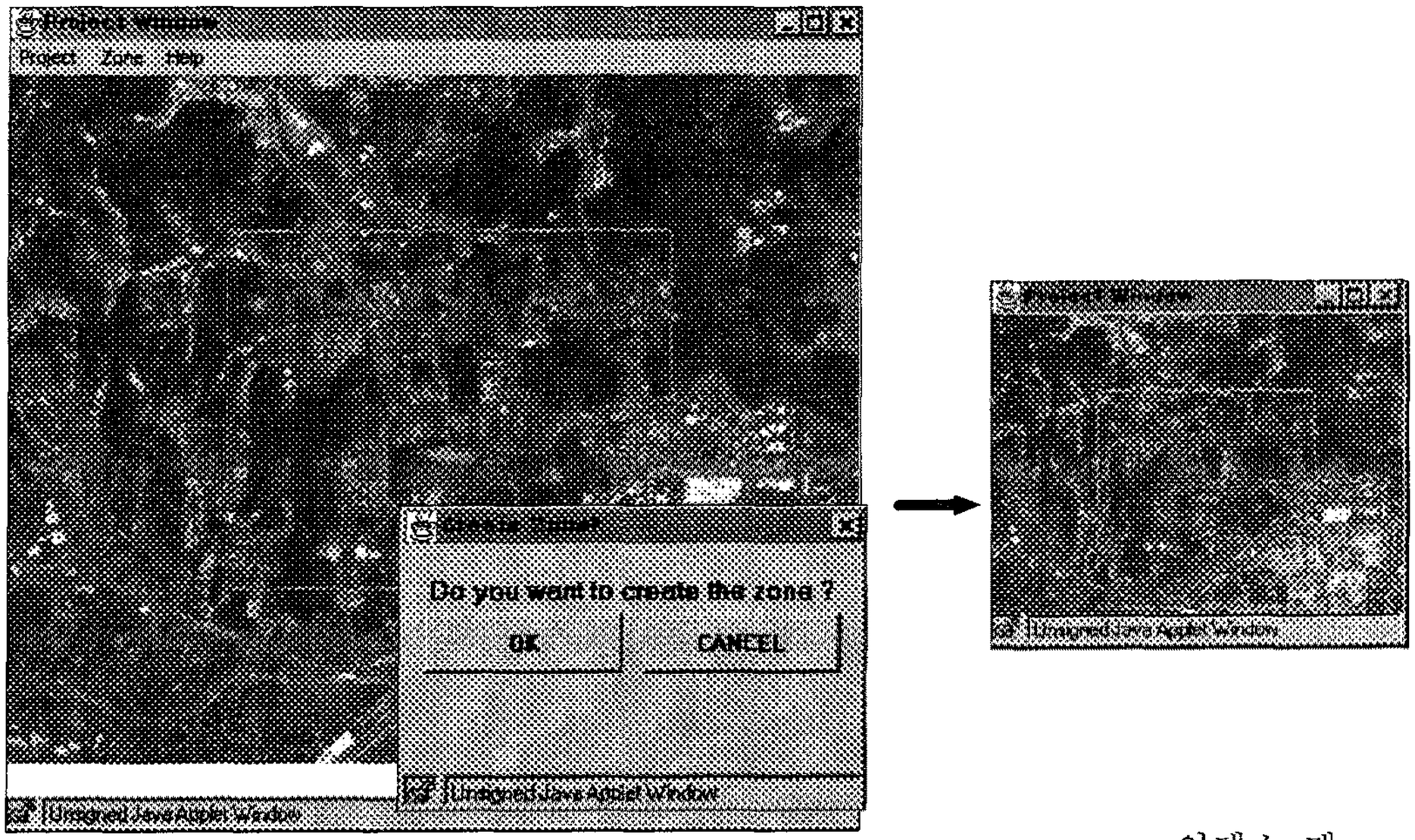
- RealToDisplayX(int realCord), RealToDisplayY(int realCord) : 실제 좌표 값을 디스플레이 화면의 스케일에 맞도록 변환시킨다.

- DisplayToRealX(int displayCord), DisplayToRealY(int displayCord) : 디스플레이 화면상의 좌표 값을 실제 좌표로 변환시킨다. 공간 검색 및 실제 위치 값을 얻고자 할 때 사용되며, X축과 Y축의 스케일이 다르므로 두 개의 메소드가 따로 존재하고 있다.

- MakeNodeList(NodeFeatureList nfList),
  MakeChainList(ChainFeatureList),
  MakePolygonList(PolygonFeatureList pfList) : 노드, 체인, 폴리곤 피쳐 데이터 리스트를 속성 정보를 제외하고 공간 정보만 가지는 노드, 체인, 폴리곤 리스트로 변환시킨다. 공간 정보를 이용하여 디스플레이를 시키고자 할 때 주로 이용된다.

2) GISFrame 클래스

GISFrame 클래스는 Project 인터페이스의 작업 환경을 구축하고 데이터를 디스플레이 하기 위해 이용되는 클래스로서, 후에 Zone 영역이 설정되고 나면 이 클래스는 인덱스 맵을 지원하는 클래스로서의 역할을 수행하게 된다. 다음 [그림 5-19]는 Project 인터페이스와 Zone 생성후의 인덱스 맵을 보여준다.



Project 인터페이스 인덱스 맵

그림 5-19. Project 인터페이스와 인덱스 맵

Project 인터페이스에서 제공하는 메뉴로는 크게 Project, Zone, 그리고 Help를 제공하고 있는데, 다음과 같은 기능을 제공하고 있다.

- Project 메뉴 : Load, Layer Selection, Exit 메뉴로 구성되어 있으며, Load는 전체 화면의 백그라운드 이미지 또는 공간 데이터를 로딩시킨다. Layer Selection 메뉴는 원하는 피쳐를 선택할 수 있도록 팝업 메뉴를 실행시킨다. Exit 메뉴는 전체 시스템을 종료 시킨다.

- Zone 메뉴 : 이 메뉴는 Create Working Area 메뉴로 구성되어 있는데, 전체 화면상에서 임의의 지역을 선정함으로써 Zone 영역을 설정하는 역할을 수행한다.

- Help 메뉴 : Project 인터페이스에 대한 도움말 기능을 제공한다.

데이터의 로딩, Zone의 생성, 그리고 인덱스 맵의 생성등의 기능들을 제공하는 GISFrame 클래스의 주요 제공 메소드 리스트는 다음 [그림 5-20]과 같다.

```
//---------------------- GisFrame 클래스 인터페이스 메소드----------------------
//
//  public GISFrame(String server, String projectName)
//  public void drawFeatureList()
//  public void drawImage(String projectName)
//  public void removeMenu()
//  public double SetZoneScaleX( int width )
//  public double SetZoneScaleY( int height)
//  public void createZone()
//  public void createZoneOk()
//  public void createZoneCancel()
//  public void moveZone( )
//  public void moveZoneOk()
//  public void moveZoneCancel()
//  private void LoadMenuItem_Action(java.awt.event.ActionEvent event)
//  private void LayerMenuItem_Action(java.awt.event.ActionEvent event)
//  private void CreateMenuItem_Action(java.awt.event.ActionEvent event)
//  private void ExitMenuItem_Action(java.awt.event.ActionEvent event)
//-------------------------------------------------------------------------
```

그림 5-20. GISFrame 클래스 인터페이스 메소드

- GISFrame(String server, String projectName) : Project 인터페이스의 작업환경을 구성한다. 메뉴 구성, 디스플레이 크기 구성등의 작업을 수행한다.

- drawFeatureList() : Project 메뉴의 Load 메뉴를 수행될 때 불리어지는 메소드로서 현재 Project에서 선택되어진 피쳐들의 집합을 디스플레이 시키는 일을 수행한다.

- drawImage(String projectName) : 입력으로 주어진 전체 데이터 집합의 이름에 해당하는 배경 이미지를 디스플레이 시킨다.

- removeMenu() : Zone 영역을 설정된 후 Project 윈도우를 인덱스 맵으로 변화시킬 때 Project 윈도우에서 메뉴를 삭제시키는 메소드이다.

- SetZoneScaleX(int width), SetZoneScaleY(int height) : Zone을 생성하고자 할 때, Project에서 제공되는 디스플레이 관련 스케일 정보를 바탕으로 Zone 윈도우내에서의 스케일 값을 계산한다.

- CreateZone(), CreateZoneOk(), CreateZoneCancel() : Project 윈도우에서 임의의 영여게 대하여 Zone 영역을 생성하는데, 생성이 성공하게 되면 Project 윈도우는 인덱스 맵으로 이동하게 된다.

- MoveZone(), MoveZoneOk(), MoveZoneCancel() : 이는 Project 윈도우가 아닌 인덱스 맵상에서 Zone영역을 이동시키기 위해서 사용되는 메소드들이다.

- LoadMenuItem_Action(java.awt.event.ActionEvent event) : Project 메뉴의 Load 메뉴를 실행시키는 것으로, 배경 이미지와 피쳐 데이터

집합들을 디스플레이 시킨다.

- LayerMenuItem_Action(java.awt.event.ActionEvent event) : 현재 Project에서 디스플레이하고자 하는 피쳐들을 선택할 수 있는 기능을 제공한다.

- CreatemenuItem_Action(java.aset.event.ActionEvent event) : Zone 영역을 설정하는 기능을 수행한다.

- ExitMenuItem_Action(java.aet.event.ActionEvent event) : Project에서 시스템을 종료시킨다.

라. 존(Zone) 인터페이스

Zone 인터페이스는 Project 영역에서 임의의 작업 영역을 설정하였을 때 생성되는데, 공간 분석을 비롯한 대부분의 작업이 여기서 수행된다. Zone 인터페이스에서 수행되는 작업을 보면 다음과 같은 것들이 있다.

- 이미지와 피쳐 데이터 집합 로딩 기능
- 작업하고자 하는 피쳐 선택(Layer Select) 기능
- 수정, 삭제, 추가된 데이터에 대하여 서버 데이터베이스로 저장기능
- Copy, Cut, Move, Create, Update의 공간 객체 편집 기능
- Zoom In, Zoom Out, UnZoom, User Defined Zoom, Zoom Area등의 View 기능
- 피쳐 레이블링(Labeling) 기능

- 피쳐 Legend 기능
- TM, UTM, GEO 좌표등의 좌표계(Coordinate System) 변환 기능
- SQL 질의 처리 기능
- 특정 통계 처리 기능
  - Near, Containment, Adjacency, Connectivity, Buffer Zone, Shortest Path등의 공간 분석 기능
- Web을 통한 외부 홈 페이지 링크 기능
- 도움말 기능

이러한 대부분의 기능을 수행하는 Zone 인터페이스도 Project 인터페이스와 마찬가지로 크게 두 개의 클래스로 나뉘어서 구성되어 있다. 먼저 Zone 클래스는 데이터베이스와 연결 설정뿐만 아니라 공간 분석 클래스와의 연결, 질의 처리기능 클래스와의 연결, 통계 처리기능 클래스와의 연결, 좌표계 처리 기능 클래스와의 연결등 Zone 인터페이스에서 수행되는 대부분의 기능에 대한 클래스와의 연결을 구성하고 있다. 그리고 사용자의 작업 환경 구성과 View 기능, Labeling 기능등을 제공하기 위해서 ZoneFrame 클래스가 있다.

1) Zone 클래스

Zone 클래스는 데이터베이스와의 연결 및 공간 분석 처리, 질의 처리, 통계 처리, 좌표계 변환 처리등 대부분의 기능을 수행하기 위해서 많은 정보들을 유지하고 있다. 또한 데이터베이스로부터 읽어온 피쳐 데이터 집합을 유지하기 위해 캐쉬 메모리를 유지하고 있다. 다음 [그림 5-21]은 Zone 클래스내에 저장되고 유지되는 정보들의 집합을 보여 준다.

```
//------------------------------ Zone 클래스 정의 ------------------------------

public class Zone {
        private static double       zoomFactor = 1.25;          // Zoom 비율
        private double              zoomValue;                  // 현재 Zoom 값
        private int                 pminX, pminY, pmaxX, pmaxY;       // Project MBR
        private int                 minX, minY, maxX, maxY;           // Zoomed MBR
        private int                 displayX, displayY;         // Zone의 출력 윈도우 크기
        private int                 diffX, diffY;       // X축과 Y축 방향으로의 차이

        String                      projectName;                // Project 이름
        private FDTList             fDefTable;                  // 피쳐 정의 테이블
        private DBProtocol          dbchannel       // 데이터베이스 접근 프로토콜
        private String[]            fNames;         // 현재 선택된 피쳐 이름 집합
        private boolean             changeFlag;

        // 캐쉬 메모리
        private int                 cacheSize;                  // 캐쉬 크기
        private int                 ncLoc;          // 현재 노드 캐쉬 메모리의 위치
        private int                 ccLoc;          // 현재 체인 캐쉬 메모리의 위치
        private int                 pcLoc;          // 현재 폴리곤 캐쉬 메모리의 위치
        private NodeFeatureList[]   nodeCache;              // 노드 캐쉬
        private ChainFeatureList[]  chainCache;             // 체인 캐쉬
        private PolygonFeatureList[]    polyCache;          // 폴리곤 캐쉬

        private NodeFeatureList     insNodeFList;   // 추가되는 노드 피쳐 리스트
        private ChainFeatureList    insChainFList;  // 추가되는 체인 피쳐 리스트
        private PolygonFeatureList  insPolyFList;   // 추가되는 폴리곤 피쳐 리스트

        private NodeFeatureList     delNodeFList;   // 삭제되는 노드 피쳐 리스트
        private ChainFeatureList    delChainFList;  // 삭제되는 체인 피쳐 리스트
        private PolygonFeatureList  delPolyFList;   // 삭제되는 폴리곤 피쳐 리스트
}
```

그림 5-21. Zone 클래스 정의

- ZoomFactor : Zoom In 이나 Zoom Out 기능을 수행할 때 기본적으로 확대/축소 되는 값이다.

- ZoomValue : 현재 화면의 Zoom 값을 유지하고 있다.

- pminX, pminY, pmaxX, pmaxY : Project 클래스의 최소 경계 사각형 정보를 유지한다.

- minX, minY, maxX, maxY : Zone 클래스의 최소 경계 사각형 정보를 유지

- displayX, displayY, diffX, diffY : 피쳐 데이터들의 실제 좌표값들을 Zone 화면상의 값으로 변환시키기 위한 정보들을 유지하고 있다.

- fDefTable : 데이터베이스내에 정의되어 있는 피쳐 정의 테이블을 유지한다.

- dbchannel : 브로커 시스템과의 접속을 위한 채널 변수이다.

- fNames : 현재 Zone 상에서 선택되어진 피쳐 이름 집합들을 유지한다.

- cacheSize : 현재 Zone상의 캐쉬 메모리의 크기 정보를 유지한다. 캐쉬의 크기는 상황에 따라 변할 수 있다.

- ncLoc, ccLoc, pcLoc : 각 노드, 체인, 폴리곤 피쳐 캐쉬 메모리의 위치를 나타내는데, 캐쉬 메모리의 내용을 바꾸기 위해서 사용된다.

- nodeCache, chainCache, polyCache : 노드, 체인, 폴리곤 피쳐에 대한 캐쉬 메모리를 할당하고 있다.

- insNodeFList, insChainFList, insPolyFList : Zone 상에서 피쳐 데이터에 편집 작업을 수행하였을 때, 새로이 입력될 데이터 집합들을 유지하고 있다.

- delNodeFList, delChainFList, delPolyFList : Zone 상에서 피쳐 데이터에 대한 편집 작업 후, 삭제될 데이터 집합들을 유지하고 있다.

위와 같이 정의된 Zone 클래스는 데이터베이스 접속 처리 및 공간 분석 처리등의 많은 메소드 리스트를 제공하는데, 중요한 메소드를 정리하면 다음 [그림 5-22]와 같다.

```
//------------------------- Zone 클래스 인터페이스 메소드-------------------------
//
//  public Zone (Project project)
//  public void SetInsNode{Chain/Polygon}FeatureList(Node{Chain/Polygon}FeatureList list)
//  public void SetDelNode{Chain/Polygon}FeatureList(Node{Chain/Polygon}FeatureList list)
//  public void SetFDTList(FDTList table)
//  public void SetZoomIn (double value), public void SetZoomOut (double value)
//  public void ReSetZoomValue ()
//  public String[] GetAllFNames()
//  public Node{Chain/Polygon}FeatureList LoadNode{Chain/Polygon}Feature(String condition,
//                  int x1, int y1, int x2, int y2)
//  public Node{Chain/Polygon}FeatureList GetNode{Chain/Polygon}FeatureWithCond(String
//                  fName, String condition, int x1, int y1, int x2, int y2)
//  public double GetNode{Chain/Polygon}Statistics(String fName, String condition)
//  public void AppendInsNode{Chain/Polygon}FeatureList(Node{Chain/Polygon}FeatureList
//                  insList)
//  public void AppendDelNode{Chain/Polygon}FeatureList(Node{Chain/Polygon}FeatureList
//                  delList)
//  public Node{Chain/Polygon}FeatureList GetNode{Chain/Polygon}Candidates(String fName,
//                  double x, double y)
//  public Node{Chain/Polygon}Feature SelectNode{Chain/Polygon}Feature
//                  (Node{Chain/Polygon}FeatureList candidate, double x, double y)
//  public void CopyFeature(Node{Chain/Polygon}FeatureList list, double x1, double y1, double
//                  x2, double y2)
//  public void CutFeature(Node{Chain/Polygon}FeatureList list, double x, double y)
//  public void MoveFeature(Node{Chain/Polygon}FeatureList list, double x1, double y1, double
//                  x2, double y2)
//  public void AddFeatureList(Node{Chain/Polygon}FeatureList list)
//  public void DeleteFeatureList(Node{Chain/Polygon}FeatureList list)
//  public int RealToDisplayX(int realCord), public int RealToDisplayY(int realCord)
//  public int DisplayToRealX(int displayCord), public int DisplayToRealY(int displayCord)
//  public int GetFeatureType(String fName)
//  public int IsNode{Chain/Polygon}CacheExist(String fName)
//  public Node{Chain/Polygon}FeatureList GetNode{Chain/Polygon}Cache(String fName)
//  public void SetNode{Chain/Polygon}Cache(Node{Chain/Polygon}FeatureList nodeFList)
```

```
//   public void ClearCache()
//   public Color GetColor(String fName)
//   public String[] GetSchema(String fName)
//-----------------------------------------------------------------------
```

그림 5-22. Zone 클래스 인터페이스 메소드 리스트

- Zone(Project project) 메소드 : Zone 객체를 생성하며, 데이터베이스와의 연결 설정, 캐쉬 메모리의 초기화, Project로부터 Zone 영역 정보 추출, 편집을 위한 데이터 정보 초기화등의 작업을 수행한다.

- SetInsNodeFeatureList(NodeFeatureList list),
  SetInsChainFeatureList(ChainFeatureList list),
  SetInsPolygonFeatureList(PolygonFeatureList list) : 편집 작업이 끝난후에 데이터베이스에 입력하고자 하는 피쳐 리스트를 세팅한다.

- SetDelNodeFeatureList(NodeFeatureList list),
  SetDelChainFeatureList(ChainFeatureList list),
  SetDelPolygonFeatureList(PolygonFeatureList list) : 편집 작업후에 데이터베이스에서 삭제하여야 할 피쳐 리스트를 세팅한다.

- SetFDTList(FDTList table) : 데이터베이스내에 저장되어 있는 피쳐 정의 테이블을 로딩하여 Zone 클래스내에 유지시킨다.

- SetZoomIn(double value), SetZoomOut(double value) : 주어진 값을

이용하여 확대, 축소의 값을 설정한다.

- ResetZoomValue() : View 상에서 UnZoom 메뉴를 수행시켰을 때 사용되는 메소드로서 원래의 화면상태로 복구 시키는 일을 수행한다.

- GetAllFNames() : 피쳐 정의 테이블에 정의되어 있는 모든 피쳐 이름들을 반환한다.

- LoadNodeFeature(String condition, int x1, int y1, int x2, int y2), LoadChainFeature(String condition, int x1, int y1, int x2, int y2), LoadPolygonFeature(String condition, int x1, int y1, int x2, int y2) : 입력으로 주어진 조건과 최소 경계 사각형을 이용하여 데이터베이스내에 있는 모든 노드, 체인, 폴리곤 피쳐 데이터를 로딩한다.

- GetNode{Chain/Polygon}FeatureWithCond(String fName, String cond, int x1, int y1, int x2, int y2) : 입력으로 피쳐 이름, 조건, 그리고 최소 경계 사각형이 주어지는데 이 세가지를 모두 만족하는 피쳐 데이터 리스트를 반환한다.

- GetNode{Chain/Polygon}Statistics(String fName, String condition) : 입력으로 피쳐 이름과 조건이 주어지는데, 데이터베이스내에서 조건을 만족하는 통계 값을 계산하여 반환한다.

- AppendInsNode{Chain/Polygon}FeatureList (Node{Chain/Polygon}FeatureList insList) : 편집 작업시에 입력이나

갱신 메뉴가 수행될 때 사용된다.

- AppendDelNode{Chain/Polygon}FeatureList
  (Node{Chain/Polygon}FeatureList insList) : 편집 작업의 삭제나 갱신 메뉴 수행시에 사용된다.

- RealToDisplayX(int realCord), RealToDisplayY(int realCord) : 데이터베이스내의 피쳐 데이터의 실제 좌표값을 Zone 윈도우의 화면상의 좌표로 변환 시킨다.

- DisplayToRealX(int displayCord), DisplayToRealY(int displayCord) : Zone 윈도우의 화면상의 좌표를 실제 좌표값으로 변환시킨다. 공간 분석의 공간 검색을 위해 주로 사용된다.

- IsNode{Chain/Polygon}CacheExist(String fName) : 입력으로 주어진 피쳐 이름에 대하여 현재 Zone 상의 캐쉬내에 피쳐 데이터가 존재하는지 조사한다. 결과로는 true/false를 반환한다.

- GetNode{Chain/Polygon}Cache(String fName) : 입력으로 주어진 피쳐 이름에 해당하는 피쳐 데이터 리스트를 캐쉬 메모리로부터 읽어온다.

- SetNode{Chain/Polygon}Cache(Node{Chain/Polygon}FeatureList list)
  : 데이터베이스로부터 읽은 노드, 체인, 폴리곤 피쳐 데이터 리스트를 Zone 인터페이스의 캐쉬 메모리에 저장하는 역할을 수행한다. 캐쉬내

의 피쳐 대치 기법은 FIFO(First In, First Out) 기법을 이용하고 있다.

- ClearCache() : 캐쉬내의 모든 피쳐 데이터 리스트를 모두 삭제 시킨다.

- GetColor(String fName) : 주어진 피쳐 이름에 대한 Color를 반환하며 이 메소드는 출력시에 이용한다.

- GetSchema(String fName) : 주어진 피쳐 이름에 대하여 데이터베이스내의 스키마를 읽어온다. 공간 분석시, 질의 처리시, 그리고 통계 처리시등에 주로 이용된다.

2) ZoneFrame 클래스

ZoneFrame 클래스는 실제로 Zone의 작업환경을 구축하고, 공간 분석, 질의 처리, 통계 처리등의 모든 작업을 수행하기 위한 그래픽 인터페이스를 제공하고 있다. 다음 [그림 5-23]은 편집 기능을 수행했을 때의 Zone 사용자 인터페이스 환경을 보여준다.

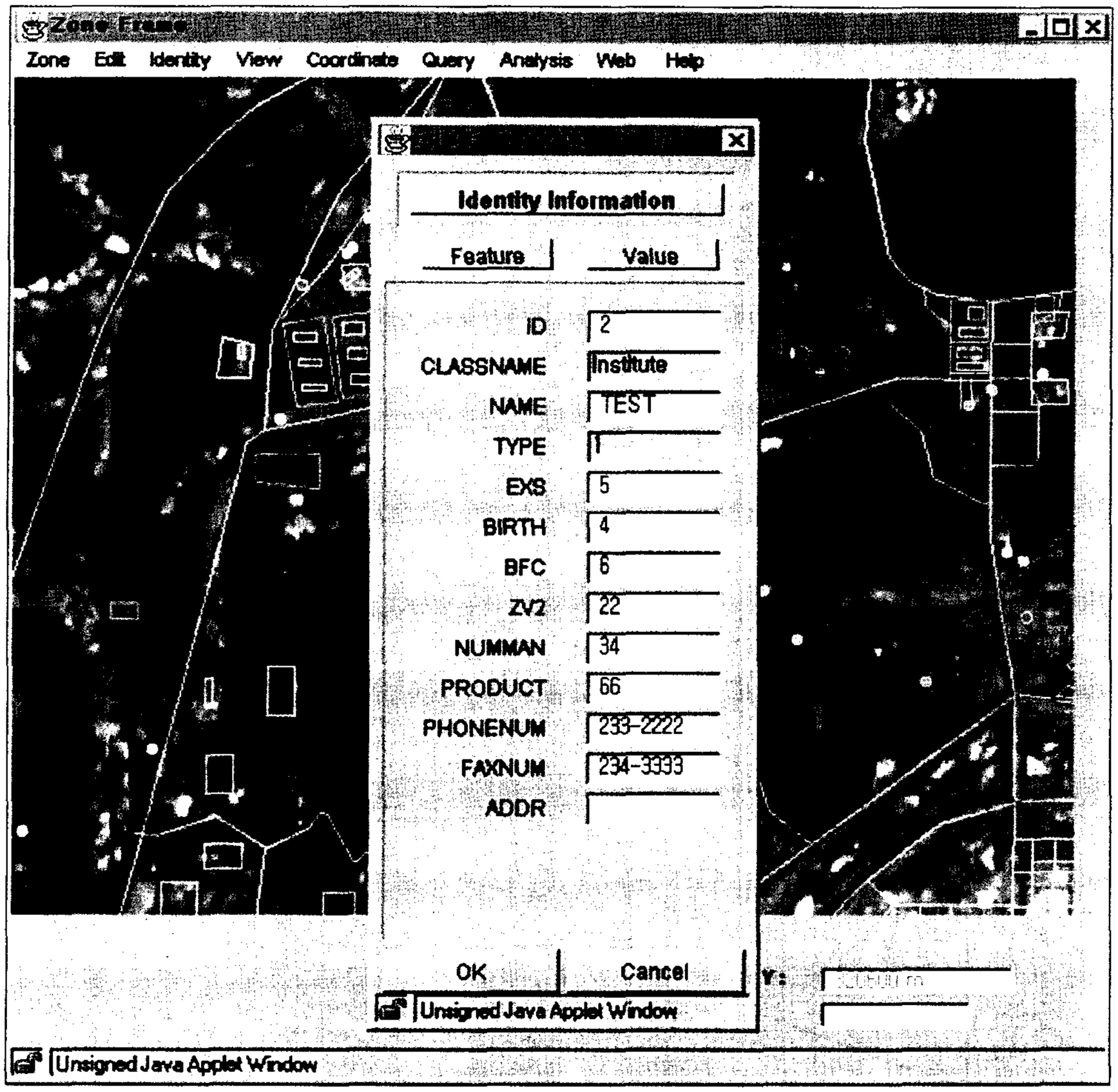


그림 5-23. 편집기능을 수행하였을 때의 Zone 사용자 작업 환경

Zone에서 제공하는 메뉴로는 Zone, Edit, Identity, View, Coordinate, Query, Analysis, Web Link, Help 메뉴들이 있으며, 이들의 자세한 설명은 다음 장에서 설명하도록 하고, 먼저 이러한 메뉴 및 메뉴의 처리 기능 그리고 디스플레이 기능들을 제공하는 ZoneFream 메소드들을 살펴보면 다음 [그림 5-24]와 같다.

```
//---------------------- ZoneFrame 클래스 인터페이스 메소드----------------------
//
//  public ZoneFrame(Project project)
//  public void ReloadFeature(int x1, int y1, int x2, int y2)
//  public void drawFeatureList()
//  public String MakeCondition(String[] fnames, int count)
//  void menuItemZoneload_Action(java.awt.event.ActionEvent event)
//  void menuItemVIS_Action(java.awt.event.ActionEvent event)
//  void ExitMenuItem_Action(java.awt.event.ActionEvent event)
//  void Node{Chain/Polygon}IdentityMenuItem_Action(java.awt.event.ActionEvent event)
//  void ZoomInMenuItem_Action(java.awt.event.ActionEvent event)
//  void ZoomOutMenuItem_Action(java.awt.event.ActionEvent event)
//  void UnzoomMenuItem_Action(java.awt.event.ActionEvent event)
//  void UserzoomMenuItem_Action(java.awt.event.ActionEvent event)
//  void LabelMenuItem_Action(java.awt.event.ActionEvent event)
//  void LegendMenuItem_Action(java.awt.event.ActionEvent event)
//  void menuItemSQL_Action(java.awt.event.ActionEvent event)
//  void menuItemStatistics_Action(java.awt.event.ActionEvent event)
//  void menuItemNear_Action(java.awt.event.ActionEvent event)
//  void menuItemContain_Action(java.awt.event.ActionEvent event)
//  void menuItemAdj_Action(java.awt.event.ActionEvent event)
//  void menuItemConnect_Action(java.awt.event.ActionEvent event)
//  void menuItemBuffer_Action(java.awt.event.ActionEvent event)
//  void menuItemSPath_Action(java.awt.event.ActionEvent event)
//  void copyMenuItem_Action(), void cutMenuItem_Action(), void moveMenuItem_Action()
//  void createMenuItem_Action(), void updateMenuItem_Action()
//  void CoordinateHelp_Action(java.awt.event.ActionEvent event)
//  void UdmsHelp_Action(java.awt.event.ActionEvent event)
//  void xyMenuItem_Action(java.awt.event.ActionEvent event)
//  void LatMenuItem_Action(java.awt.event.ActionEvent event)
//  void UtmMenuItem_Action(java.awt.event.ActionEvent event)
//-------------------------------------------------------------------------
```

그림 5-24. ZoneFrame 클래스 인터페이스 메소드

- ZoneFrame(Project project) : Zone, Edit, View, Identity, Analysis, Query등의 기본적인 작업 환경을 구현하기 위한 객체를 생성한다.

- ReloadFeature(int x1, int y1, int x2, int y2) : 입력으로 주어진 최소 경계 사각형 내의 모든 피쳐들을 데이터베이스로부터 읽어온다. 단, 기존의 캐쉬내의 데이터를 모두 삭제하고 새로 불러온 데이터로 캐쉬 내용을 갱신 시킨다.

- drawFeatureList() : Zone상에서 유지되는 캐쉬내의 피쳐 데이터 리스트를 디스플레이 시킨다.

- MakeCondition(String[] fnames, int count) : 질의 처리나 통계 처리를 위해서 사용되는 메소드로 입력으로 주어진 피쳐 이름들을 가지고 오라클 SQL 형식의 조건문을 생성한다.

- MenuItemZoneload_Action(ActionEvent event),
  menuItemVIS_Action(ActionEvent event),
  ExitMenuItem_Action(ActionEvent event) : Zone 메뉴 관련 메소드.

- Node{Chain/Polygon}IdentityMenuItem_Action(ActionEvent event) :
  Identity 메뉴 관련 메소드.

- ZoomInMenuItem_Action(ActionEvent event),
  ZoomOutMenuItem_Action(ActionEvent event),
  UnzoomMenuItem_Action(ActionEvent event),

UserzoomMenuItem_Action(ActionEvent event),

LabelMenuItem_Action(ActionEvent event),

LegendMenuItem_Action(ActionEvent event) : View 메뉴 관련 메소드.

- menuItemSQL_Action(ActionEvent event),

  menuItemStatistics_Action(ActionEvent event) : Query 메뉴 관련 메소드.

- menuItemNear_Action(ActionEvent event),

  menuItemContain_Action(ActionEvent event),

  menuItemAdj_Action(ActionEvent event),

  menuItemConnect_Action(ActionEvent event),

  menuItemBuffer_Action(ActionEvent event),

  menuItemSPath_Action(ActionEvent event) : Analysis 메뉴 관련 메소드.

- copyMenuItem_Action(ActionEvent event),

  cutMenuItem_Action(ActionEvent event),

  moveMenuItem_Action(ActionEvent event),

  createMenuItem_Action(ActionEvent event),

  updateMenuItem_Action(ActionEvent event) : Edit 메뉴 관련 메소드.

- CoordinateHelp_Action(ActionEvent event),

UdmsHelp_Action(ActionEvent event) : Help 메뉴 관련 메소드.

- xyMenuItem_Action(ActionEvent event),
  LatMenuItem_Action(ActionEvent event),
  UtmMenuItem_Action(ActionEvent event) : Coordinate 메뉴 관련 메소드.

마. 레이어(Layer) 인터페이스

Layer 인터페이스는 Zone 메뉴에서 Layer Selection 메뉴를 실행시켰을 때 생성되는 인터페이스로서, 데이터베이스내에 존재하는 모든 피쳐 리스트에 대하여 사용자가 작업하기를 원하는 피쳐를 선택할 수 있는 기능을 제공한다. [그림 5-25]는 Layer Selection의 수행 예를 보여준다.

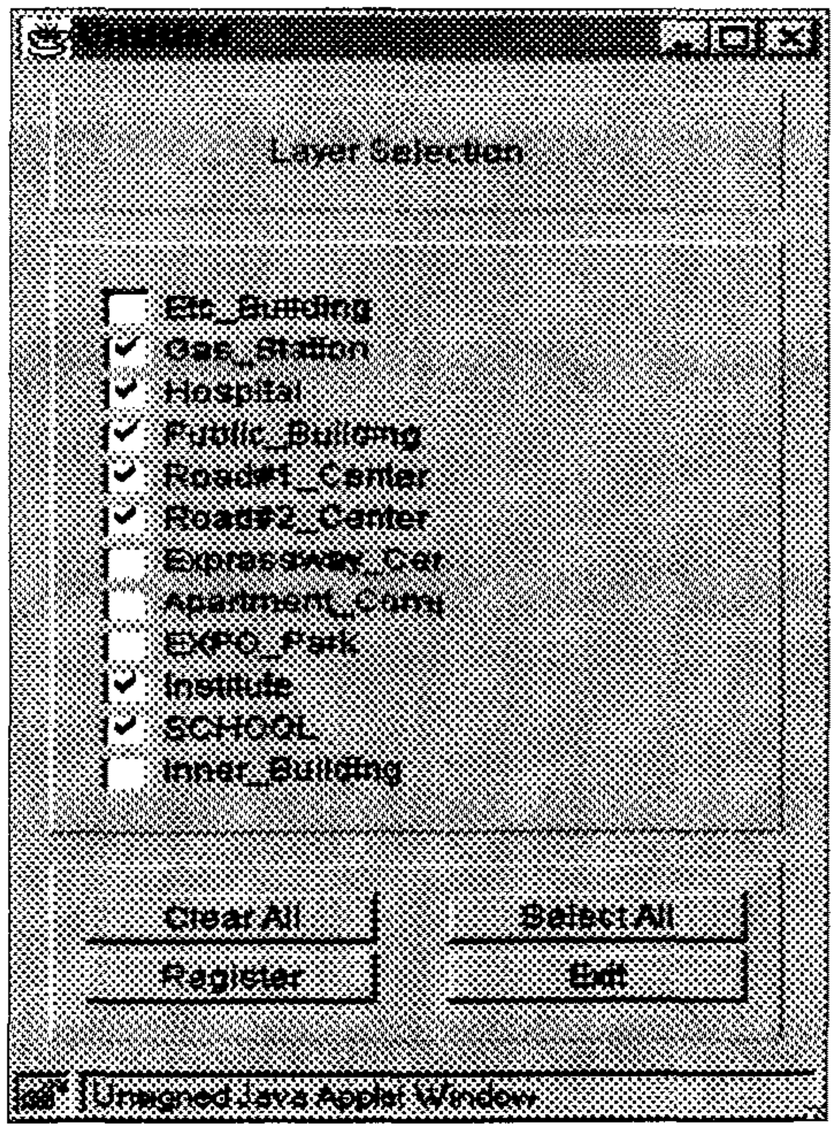


그림 5-25. Layer 인터페이스

위의 그림에서 보면 현재 데이터베이스내에 존재하는 전체 피쳐는 Etc_Building, Gas_Station, Hospital, Public_Building, Road#1_Center, Road#2_Center, Expressway_Center, Apartment_Complex, EXPO_Park, Institute, SCHOOL, Inner_Building의 12개이고, 현재 선택된 피쳐는 Gas_Station, Hospital, Public_Building, Road#1_Center, Road#2_Center, Institute, SCHOOL의 7개이다. 그리고 이러한 피쳐 선택 및 삭제 그리고 등록을 위하여 네 개의 버튼이 있는데 각 기능은 다음과 같다.

- Clear All 버튼 : 현재 시스템에 선택되어 있는 모든 피쳐를 삭제시킨다.
- Select All 버튼 : 현재 데이터베이스내에 존재하는 모든 피쳐를 선택한다.
- Register 버튼 : 선택되어진 피쳐 이름들을 Zone이나 Project 인터페이스에 등록한다.
- Exit 버튼 : Layer Selection을 종료한다.

2. Zone 사용자 인터페이스의 구성 요소

앞에서 살펴 보았듯이 공간 분석, 질의 처리, 통계 처리, 매핑등의 모든 작업이 Zone 인터페이스상에서 수행되고 있다. 그러므로 이번 장에서는 Zone의 사용자 작업 환경의 구성 요소에 대해서 자세히 알아보고자 한다. 현재 Zone의 구성요소를 메뉴별로 나누어 보면, Zone, Edit, Identity, View, Coordinate, Query, Analysis, Web Link, Help의 9가지가 있으며, 내부적으로 세부적인 구성요소를 가지고 있다. 이러한 세부적인 구성요소에 대한 구체적인 설명은 다음과

같다.

가. Zone 메뉴

Zone 메뉴는 데이터의 로드, 피쳐 선택, 또는 작업 데이터를 저장하는 역할을 수행하며, Load, Layer Select, Save, Exit의 4가지 서브 메뉴로 구성되어 있다.

1) Load 메뉴

이 메뉴는 배경의 이미지 데이터와 데이터베이스내의 데이터를 로드하기 위해 사용되는데, 선택되어진 피쳐들만 불러온다. 데이터의 화면 디스플레이나 Reload하기 위해 주로 사용된다.

2) Layer Select 메뉴

이 메뉴는 데이터베이스내에 존재하는 모든 피쳐들중에서 작업하고자 원하는 피쳐를 선택하기 위한 기능을 제공하는데, 이를 이용하여 Map Overlay 기능을 구현할 수 있다.

3) Save 메뉴

Zone 영역에서 수행된 작업 데이터중에서 새로이 입력하거나 삭제되어야 할 피쳐 데이터를 데이터베이스에 저장하는 역할을 수행한다.

4) Exit 메뉴

Zone 영역에서의 작업을 종료한다.

나. Edit 메뉴

편집 기능들을 수행하기 위한 메뉴로서, Copy, Cut, Move, Create, Update의 서브 메뉴들로 구성되어 있다.

1) Copy 메뉴

Zone 영역내의 임의의 객체를 선택하고, 이 객체를 복사하여 다른 위치에 이동시킬 수 있는 기능을 수행한다. 다음 [그림 5-26]은 이 메뉴를 수행할 때 생성되는 사용자 인터페이스이다.

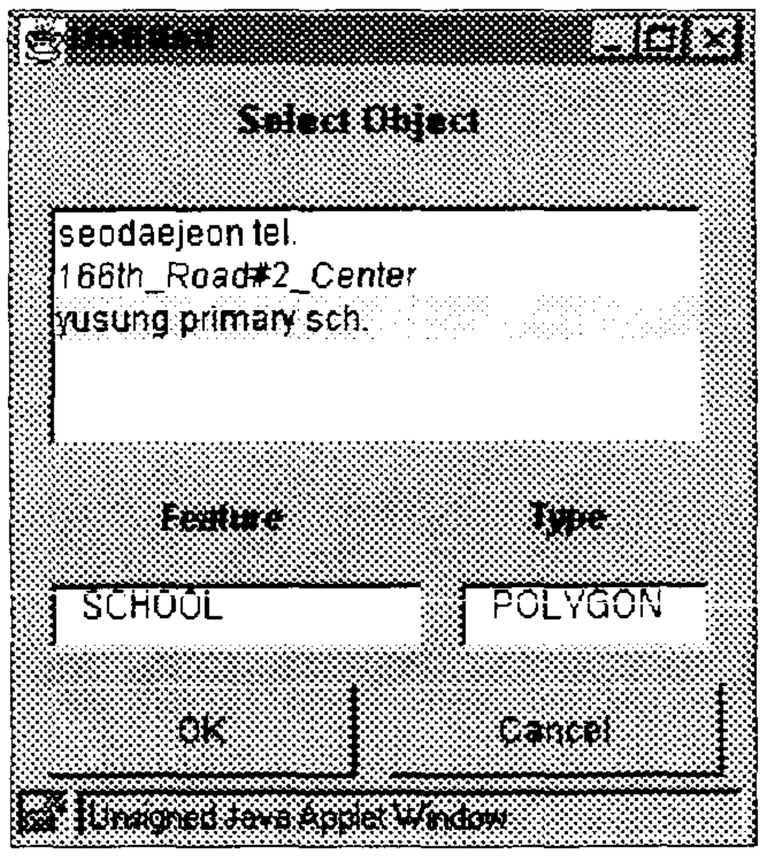


그림 5-26. Copy 인터페이스

위의 그림에서 보면 임의의 객체를 선택할 때 3개의 객체가 리스트 되어 있는데, 이는 사용자가 임의의 위치에 있는 객체를 선택하였을 때 노드, 체인, 폴리곤 피쳐 간의 간격이 너무 좁아 사용자가 원하지 않는 객체를 선택할 위험이 있다. 이를 위하여 다시 한번 사용자가 객체 이름과 타입을 보면서 선택하기 위한 인터페이스를 제공하고 있는 것이다.

2) Cut 메뉴

Zone 영역에서 선택된 임의의 객체를 삭제시키는 역할을 수행하며, 객체 선정은 Copy 메뉴와 동일하게 화면상에서 임의의 객체를 클릭한 다음에 다시 사용자 인터페이스를 보면서 세부 선정을 한다.

3) Move 메뉴

Zone 영역에서 임의의 객체를 선택하고, 이 선택된 객체를 다른 위치로 이동시키는 역할을 수행한다. 객체의 선택과정은 Copy 메뉴에서와 같다.

4) Create 메뉴

Create 메뉴는 새로운 객체를 생성하며, 생성 순서는 다음과 같다.

- 새로이 생성할 객체의 피쳐를 먼저 결정한다. 피쳐의 결정은 현재 Zone 영역에 존재하는 피쳐들중에서 선택한다.
- 새로운 객체를 화면상에 원하는 위치에 그린다.
- 새로운 객체의 속성 정보를 완성한다.

[그림 5-27]은 위의 새로운 피쳐 생성과정을 보여준다.

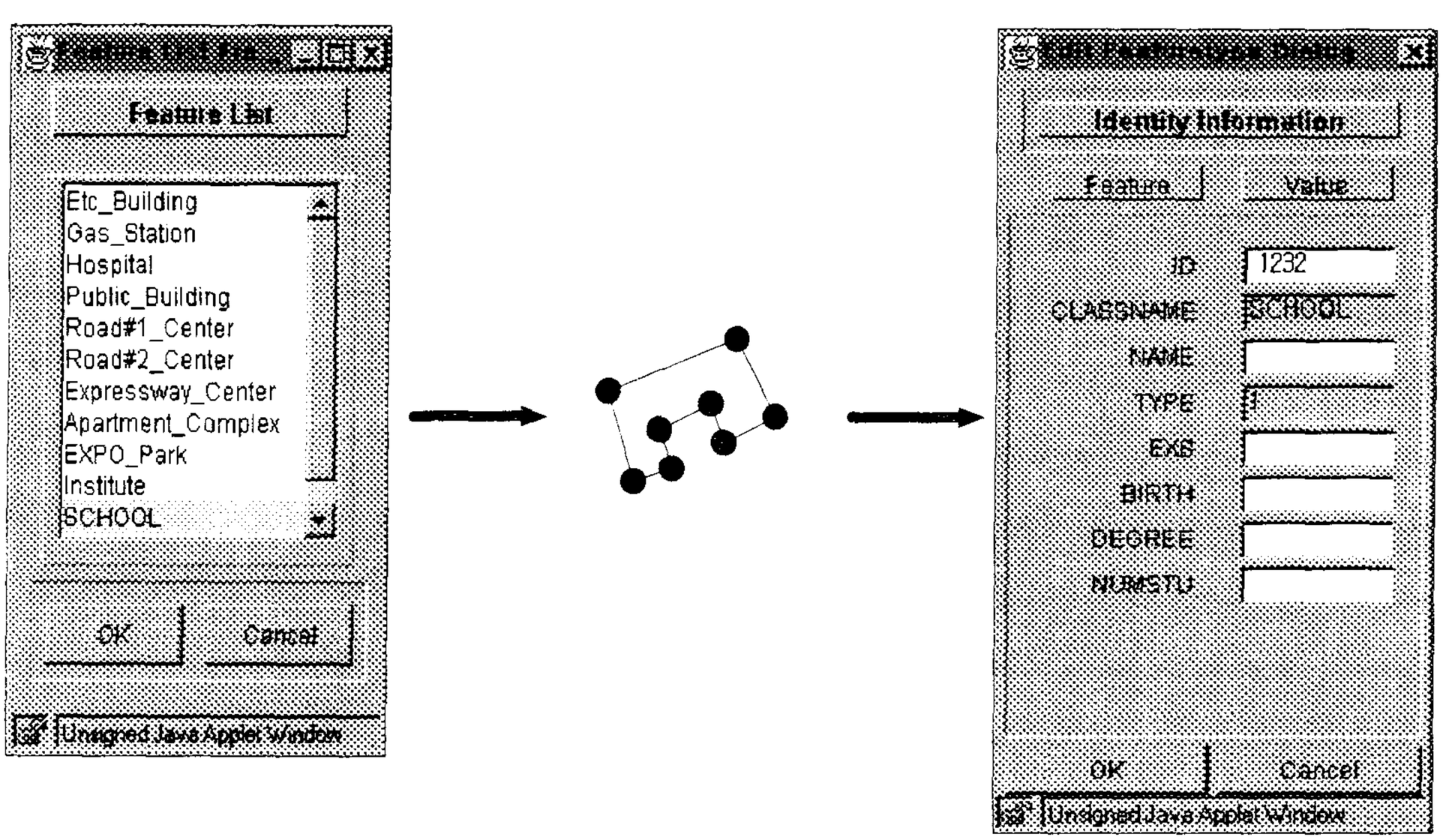


그림 5-27. Create 인터페이스 과정

5) Update 메뉴

갱신 메뉴는 임의의 객체를 선택하고 선택된 객체의 속성 정보를 수정하는 역할을 수행한다.

다. Identity 메뉴

객체의 정보를 추출하기 위한 메뉴로서 노드, 체인, 폴리곤의 서브 메뉴

로 구성되어 있다.

1) Node Identity 메뉴

노드 피쳐들에 대한 속성 정보를 추출하기 위한 메뉴로서, 임의의 지역을 선택하였을 때 그 지역으로부터 가장 가까이 존재하는 노드 피쳐의 객체를 찾고 속성 정보를 출력한다. 노드 속성 정보 추출의 예는 [그림 5-28]과 같다.

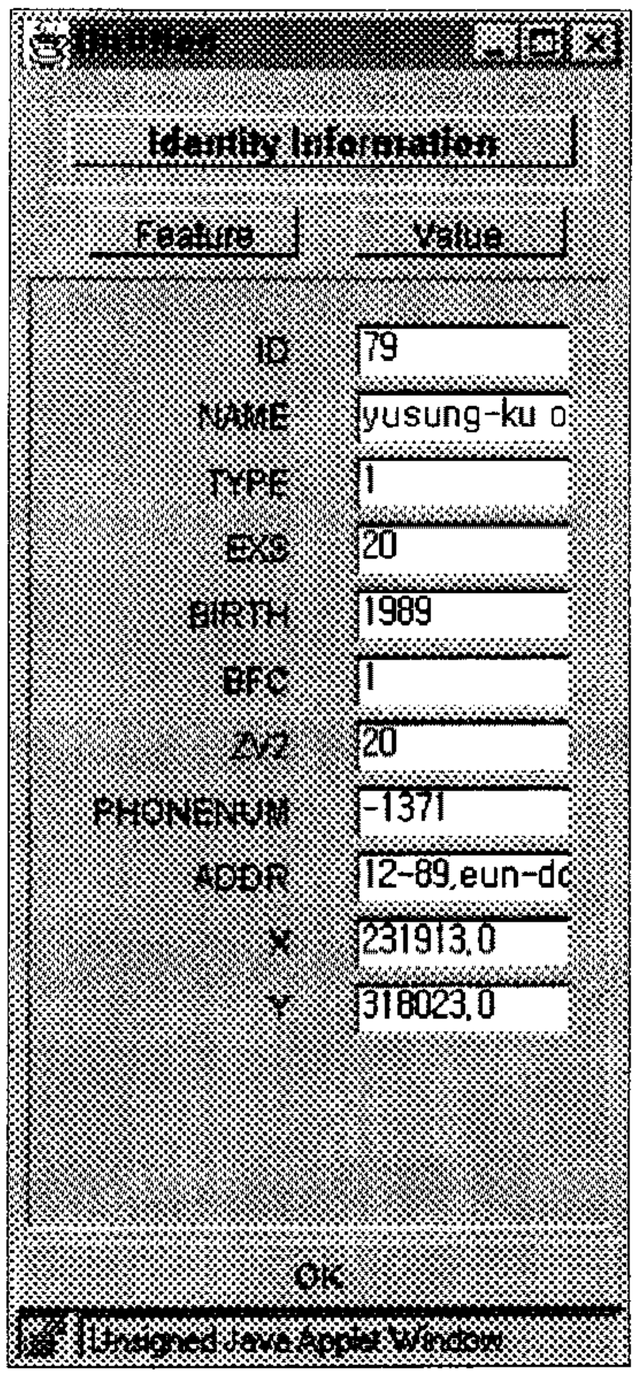


그림 5-28. Node Identity 인터페이스

2) Chain Identity 메뉴

체인 피쳐에 대하여 Node Identity와 같은 기능을 수행한다.

3) Polygon Identity 메뉴

폴리곤 피쳐에 대하여 수행되며, 기능은 Node Identity와 같다.

라. View 메뉴

Zoom In, Zoom Out, UnZoom, User Defined Zoom 그리고 Labeling등의 서브 메뉴로 구성되어 있으며, 화면 출력과 관련있는 일들을 수행한다.

1) Zoom In 메뉴

화면을 일정한 비율로 확대 시킨다.

2) Zoom Out 메뉴

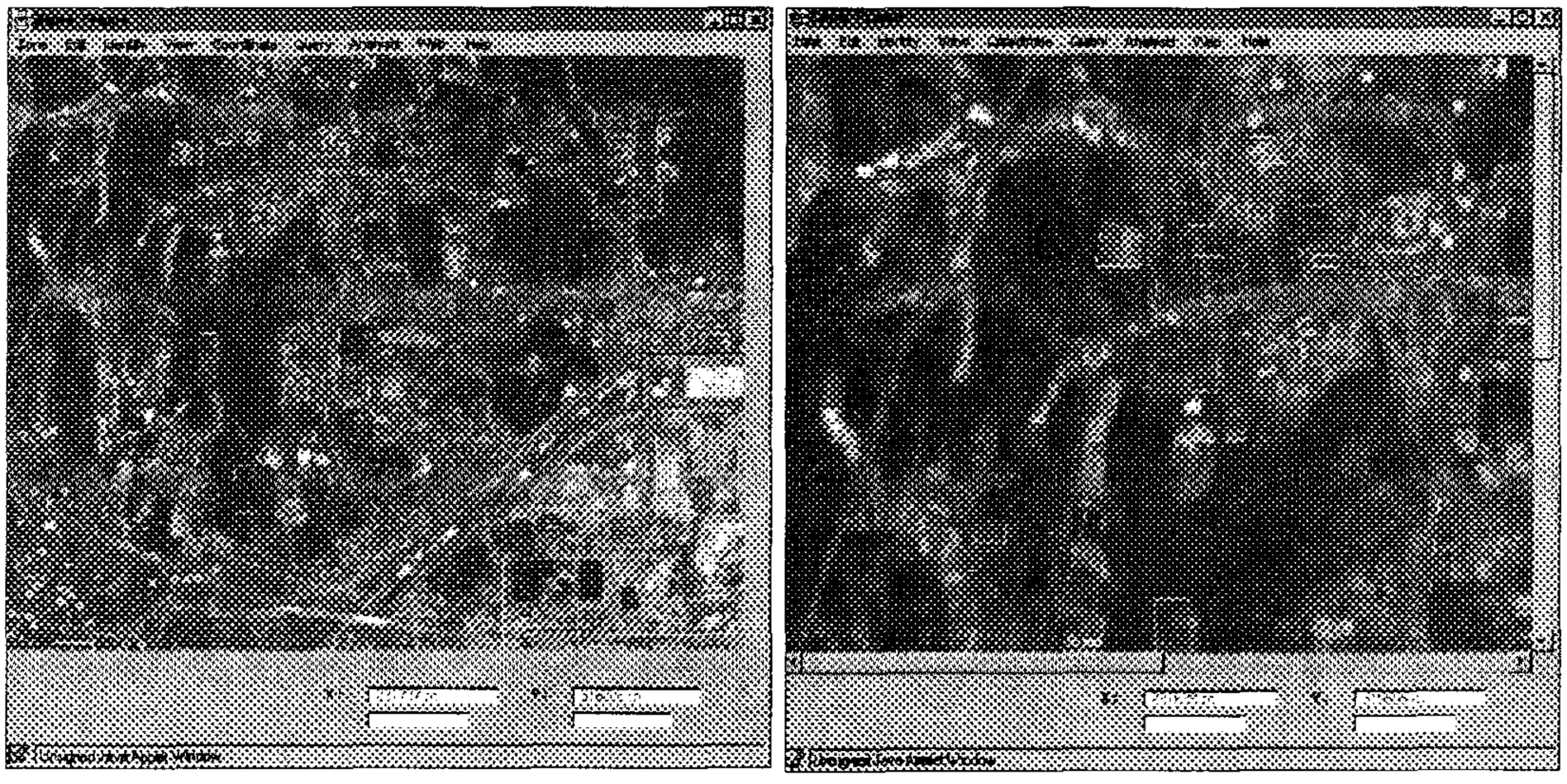
화면을 일정한 비율로 축소 시킨다.

3) UnZoom 메뉴

화면을 처음에 데이터를 로드하였을 때의 크기로 돌려준다.

4) User Defined Zoom 메뉴

사용자 정의에 따라서 화면을 확대 시키거나 축소시키는 역할을 수행한다. 다음 [그림 5-29]는 화면 확대의 예를 보여준다.



확대 이전　　확대 이후

그림 5-29. Zoom In 예

5) Label 메뉴

Zone 영역에서 임의의 선택된 피쳐들에 대하여 레이블을 출력할 수 있는데, 레이블을 위한 피쳐 선택 인터페이스는 다음 [그림 5-30]과 같다.

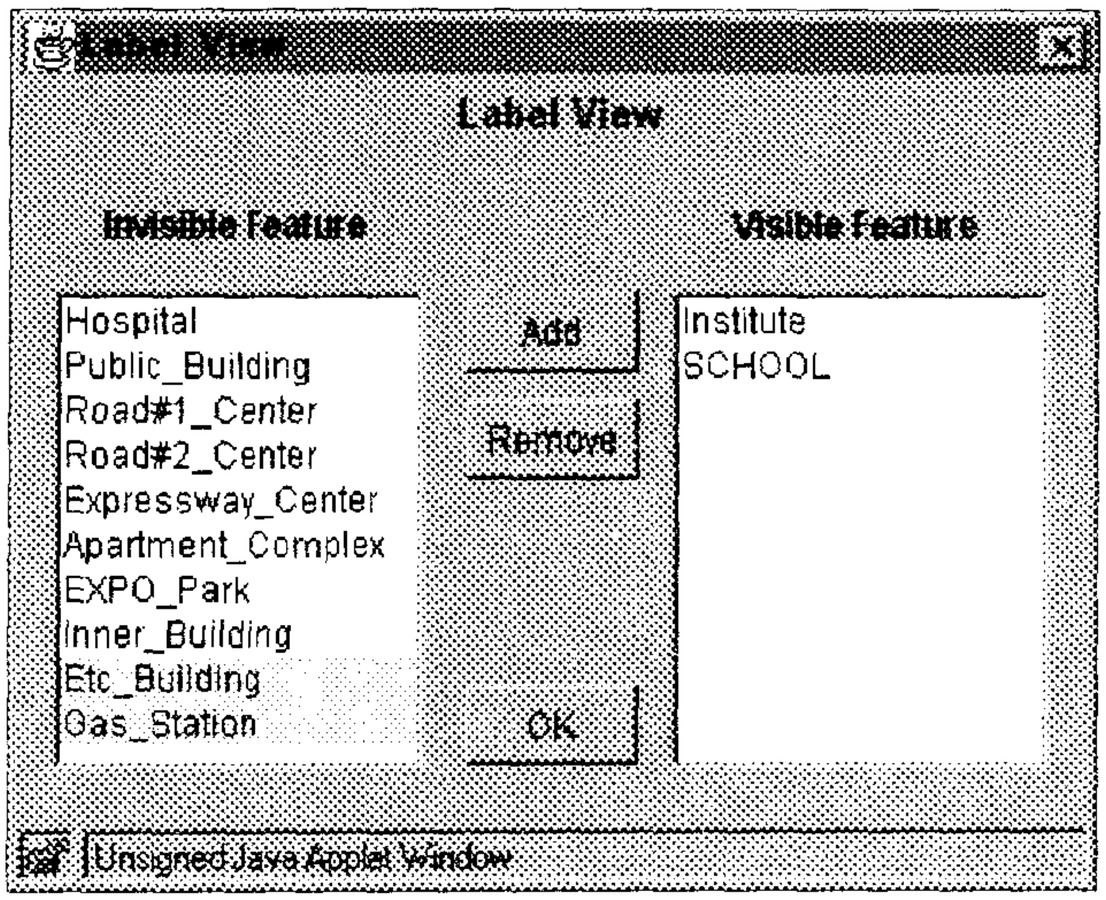


그림 5-30. Label 인터페이스

위의 인터페이스에서 현재 레이블이 출력되는 것은 Institute와 SCHOOL 피쳐이며 Etc_Building와 Gas_Station이 추가되고 있으므로, 위의 인터페이스 결과는 다음 [그림 5-31]과 같은 화면 출력을 보여준다.

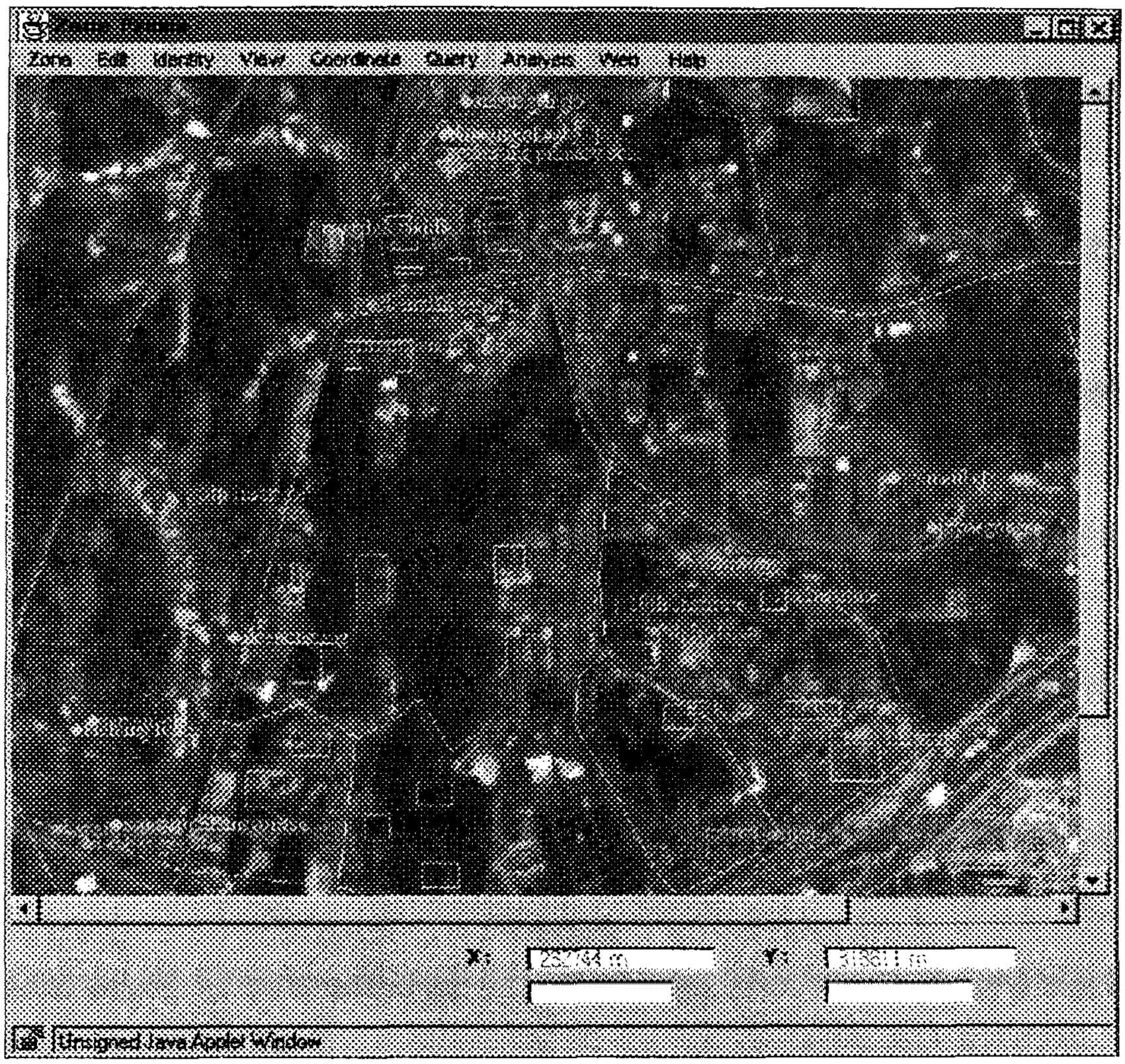


그림 5-31. 레이블링된 화면 출력

마. Coordinate 메뉴

현재 시스템에서는 TM, UTM, LAT/LON의 좌표계를 지원하고 다. 현재 좌표 값의 출력은 Zone 화면 우하단 부분에 위치하며 Coordinate 메뉴의 서브 메뉴를 선택함으로써 원하는 좌표계의 값을 얻을 수 있다.

바. Query 메뉴

Query 메뉴를 위해서는 오라클의 SQL을 직접이용하는 SQL Query 서브 메뉴와 SQL을 이용하여 통계 값을 계산하는 SQL Statistics 서브 메뉴의 두 가지가 존재한다.

1) SQL Query 메뉴

오라클 데이터베이스에 직접 접속하여 질의 처리를 수행한다. 기본적으로 피쳐 이름, 질의 처리를 하고자 하는 피쳐의 속성, 그리고 질의를 위한 정보를 입력으로 수행된다. 다음 [그림 5-32]는 이러한 질의 처리를 하기 위한 사용자 인터페이스를 보여준다.

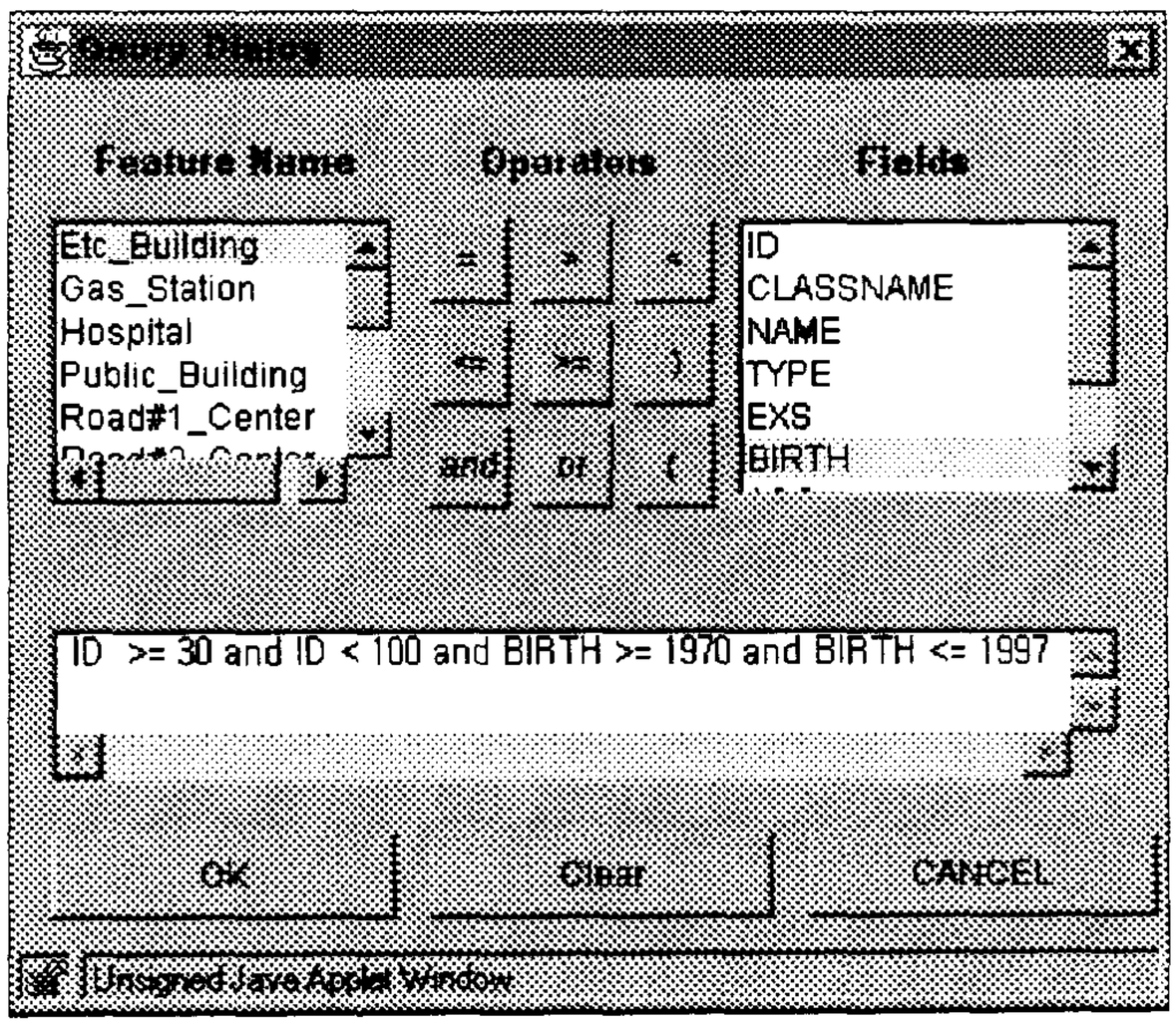


그림 5-32. SQL Query 사용자 인터페이스

위의 예제를 보면 피쳐 이름이 Etc_Builiding이고, 속성 정보로는 ID와

BIRTH가 사용된 질의 처리를 수행하고 있다.

2) SQL Statistics 메뉴

오라클 데이터베이스에 직접 접속하여 SQL을 이용하여 간단한 통계 처리 기능을 수행하는 메뉴로서 SQL Query와 비슷한 인터페이스를 가지고 있다 [그림 5-33]

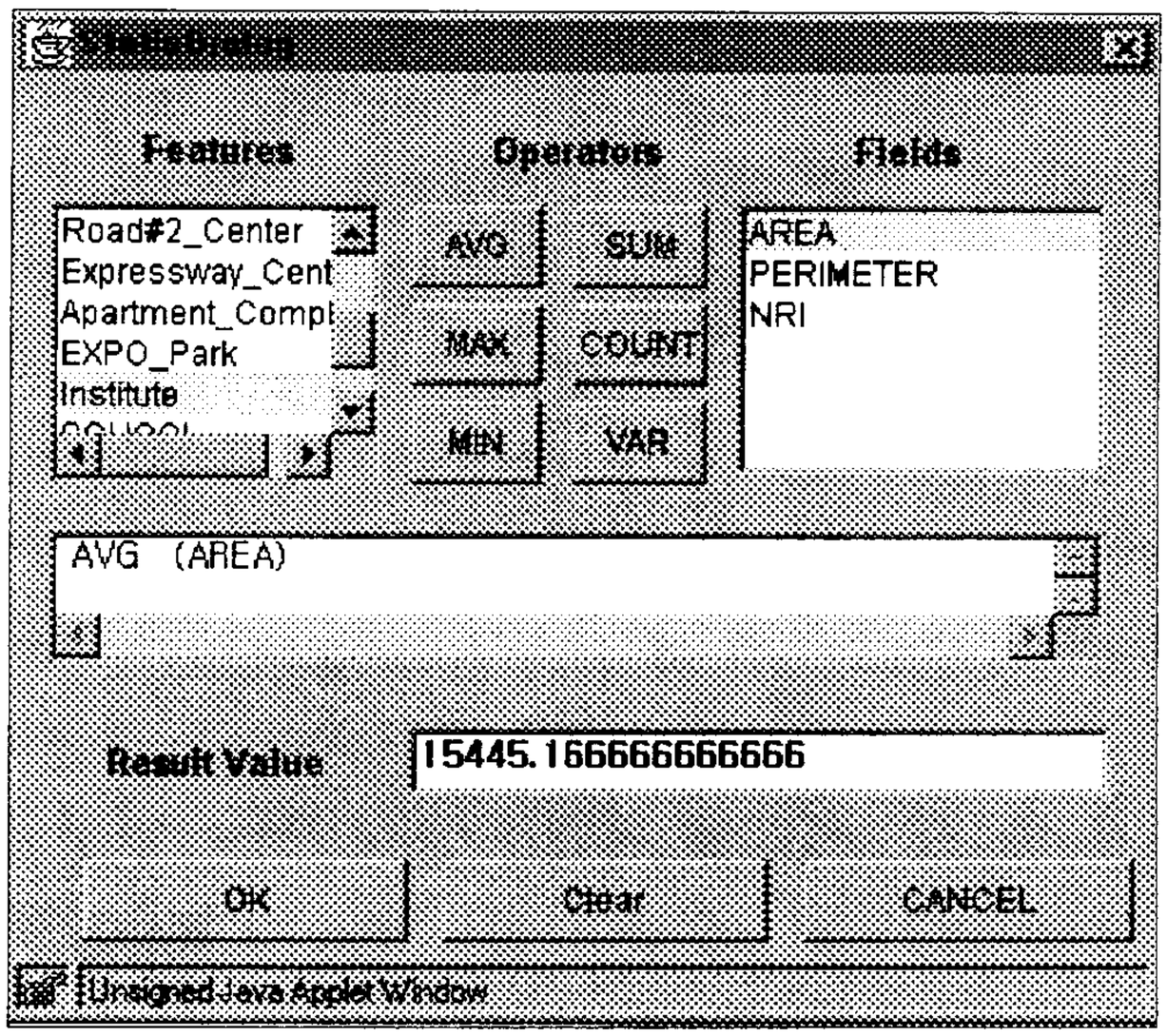


그림 5-33. SQL Statistics 사용자 인터페이스

위의 그림에서 보듯이 현재 제공되는 통계 처리 기능은 평균, 분산이며 그 이외에 합계, 최대치, 최소치, 전체 개수등의 기능을 제공한다. 위의 예에서는 Institute 피쳐에 대하여 평균 면적을 계산하는 예를 보여주고 있다.

사. Analysis 메뉴

공간 분석 메뉴에서는 Near, Containment, Adjacency, Connectivity, Buffering과 Shortest Path의 공간 분석기능들을 제공하고 있다.

1) Near 메뉴

임의의 공간 객체로부터 주어진 거리내에 존재하는 객체를 탐색하는 방법으로, 입력으로는 객체와 거리가 주어지며 출력으로는 주어진 거리내의 존재하는 객체들을 반환한다. 다음 [그림 5-34]는 Near 연산을 위한 사용자 인터페이스를 보여준다.

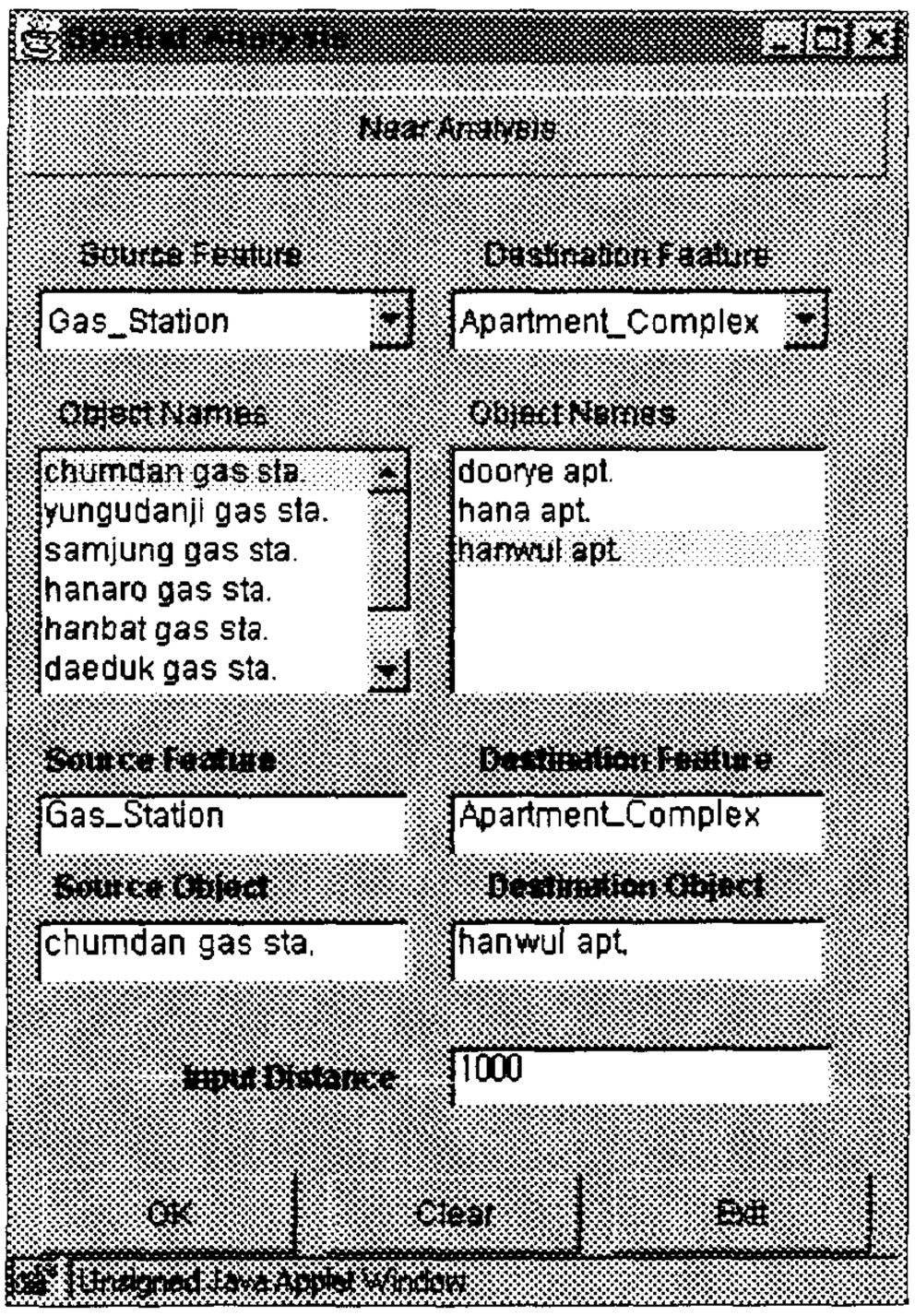


그림 5-34. Near 공간분석 인터페이스

위의 그림에서 입력 피쳐로는 노드, 체인, 폴리곤 유형의 어떤 피쳐도 올 수 있으며, 결과 피쳐도 마찬가지로 모든 피쳐를 받을 수 있다. 위의 예제의 결과는 다음과 같다.

- 입력 피쳐 : 주유소(Gas_Station)
- 입력 객체 : chumdan gas sta
- 입력 거리 : 1000 *m*
- 결과 피쳐 : 아파트 단지(Apartment_Complex)
- 현재 선택되어진 결과 객체 : hanwul apt.

2) Containment 메뉴

Containment는 임의의 객체를 선택하고, 그 객체내의 포함되어 있는 객체를 탐색한다. 다음 [그림 5-35]는 Containment의 인터페이스 예를 보여준다.

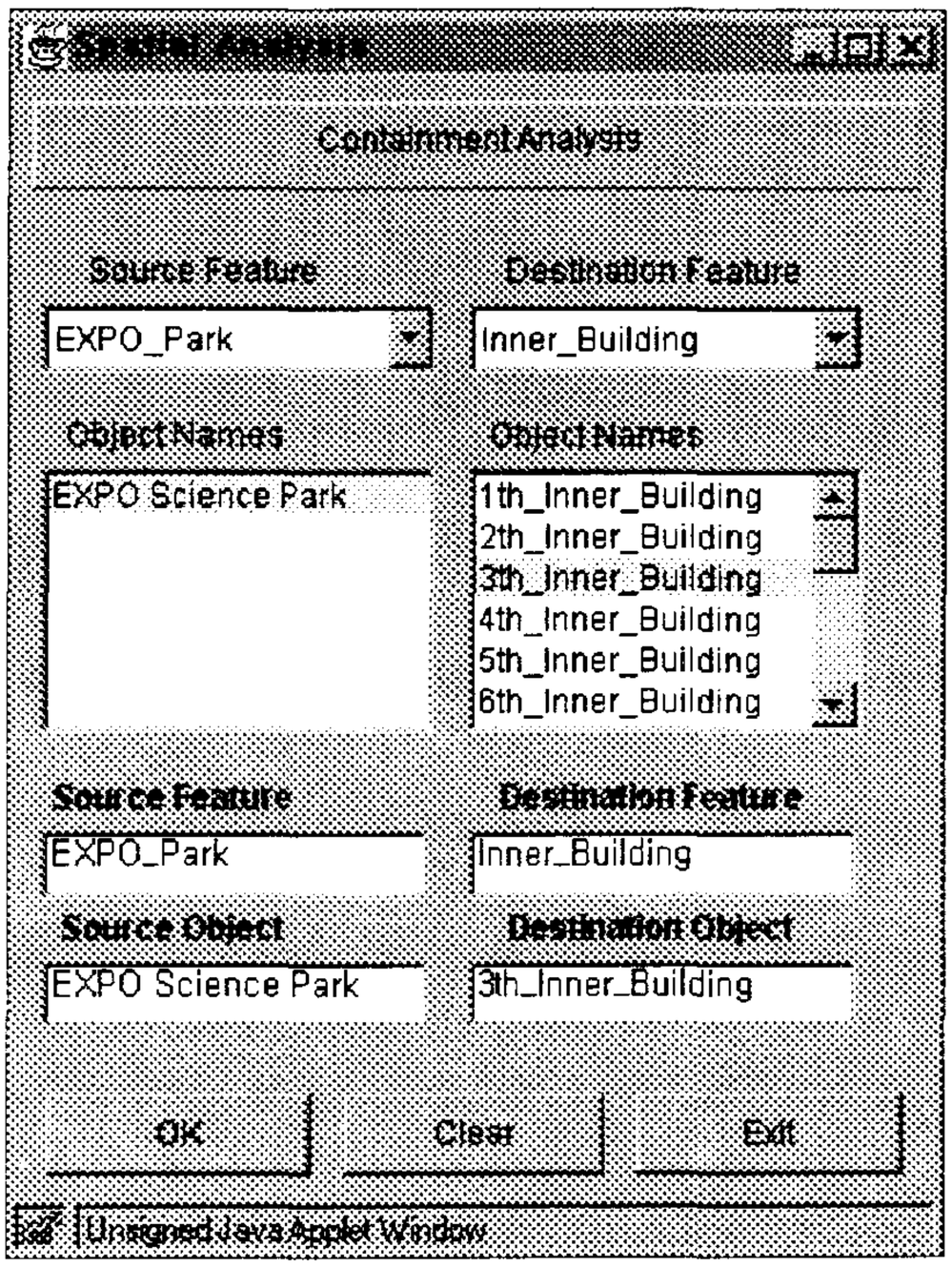


그림 5-35. Containment 공간분석 인터페이스

Containment는 포함 관계를 나타내므로 입력 피쳐로는 폴리곤 유형의 피쳐만이 사용될 수 있으며, 결과 피쳐로는 모든 피쳐의 이용이 가능하다. 위의 예의 결과는 다음과 같다.

- 입력 피쳐 : EXPO_Park
- 입력 객체 : EXPO Science Park
- 결과 피쳐 : Inner_Building
- 현재 선택되어진 결과 객체 : 3th_Inner_Building

3) Adjacency 메뉴

Adjacency는 폴리곤 유형의 피쳐를 입력으로 하여 인접한 폴리곤 피쳐들을 탐색하는 방법으로 폴리곤-폴리곤 피쳐의 공간 분석 방법이다.

4) Connectivity 메뉴

Connectivity는 체인-체인 피쳐의 공간 분석 방법으로 임의의 체인 객체에 연결되어 있는 체인들을 검색하는 방법이다. [그림 5-36]은 Connectivity 메뉴에 대한 인터페이스를 보여준다.

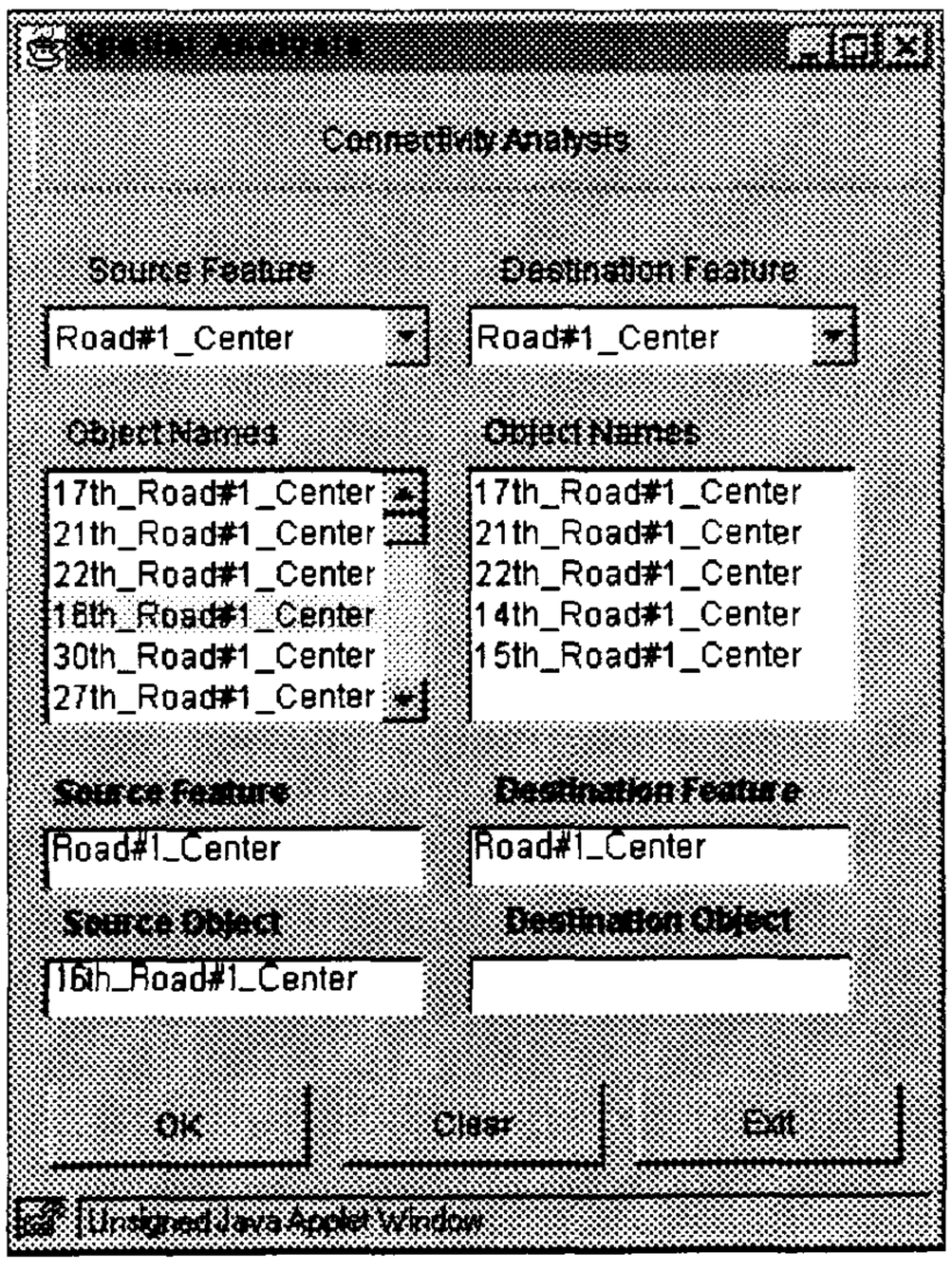


그림 5-36. Connectivity 공간분석 인터페이스

- 입력 피쳐 : Road#1_Center
- 입력 객체 : 16th_Road#1_Center
- 결과 피쳐 : Road#1_Center
- 결과 객체 : 5개

5) Buffering 메뉴

Buffering 메뉴는 임의의 선택된 객체에 대하여 주어진 길이 값을 인자로 하여 Buffer Zone을 생성한다. 입력으로는 노드, 체인, 폴리곤의 모든 피쳐들이 해당될 수 있고 결과로는 Buffer Zone의 폴리곤을 반환한다. [그림 5-37]은 Buffer Zone을 생성하기 위한 사용자 인터페이스 환경을 보여준다.

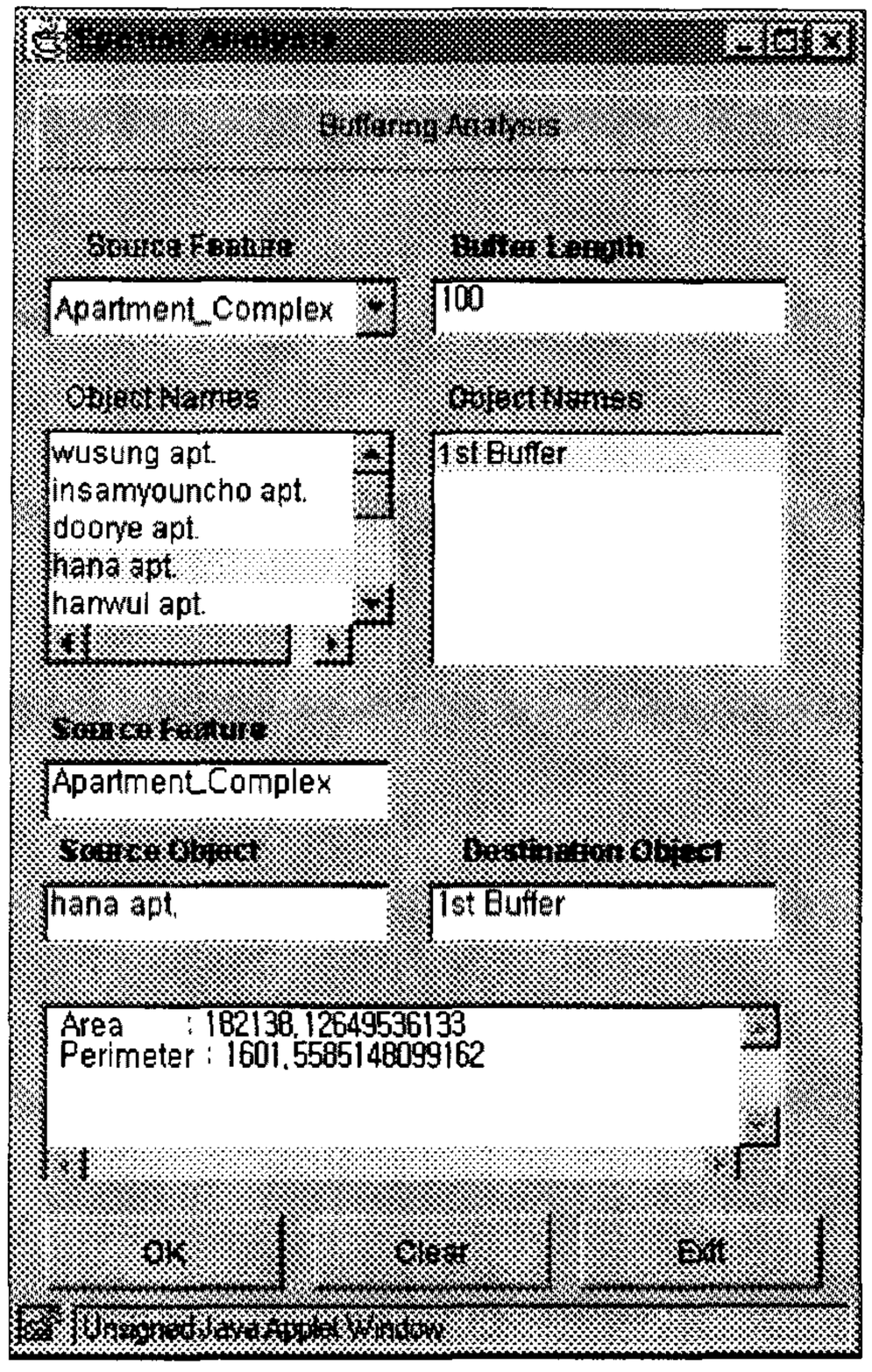


그림 5-37. Buffering 공간분석 인터페이스

- 입력 피쳐 : Apartment_Complex
- 입력 객체 : hana apt.
- 버퍼 길이 : 100 m
- 결과 : 면적 182138.1265 m^2, 둘레 1601.5585 m

6) Shortest Path 메뉴

Shortest Path의 수행순서는 다음과 같이 진행된다.

- 먼저 Shortest Path를 계산하고자 하는 피쳐를 선택한다.
- 피쳐 선택후 피쳐 근처의 임의의 두 지점을 선택한다.
- 임의의 두 지점으로부터 선택된 피쳐의 가장 가까운 각 객체들을 발견한다.
- 두 객체 사이의 Shortest Path를 계산한다.

다음 [그림 5-38]은 Shortest Path를 계산하기 위한 사용자 인터페이스를 보여준다.

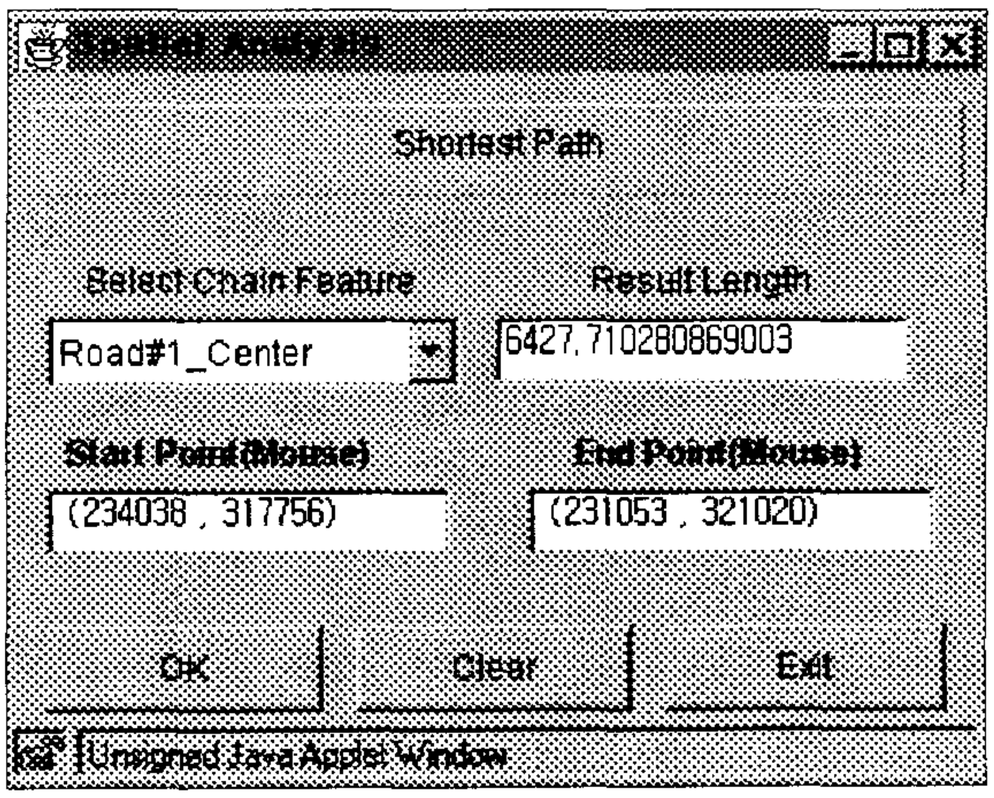


그림 5-38. Shortest Path 공간분석 인터페이스

아. Web Link 메뉴

다른 웹과의 링크를 제공한다.

자. Help 메뉴

현재는 본 시스템에 관한 간략한 도움말을 제공하고 있으나, 추후 확장이나 개선이 용이하게 구성되어 있다.

여 백

제 6 장 GIS 응용 사례

제 1 절 도시 재난 관제 시스템

1. Dispatch System

GIS는 도시에 재난이 발생하였을 때 효과적인 관제 시스템으로 활용되고 있다. 그러기 위해서는 그 시스템이 일상적인 상태에서도 재난 시와 마찬가지로 운용되고 있어야 하며, 공간 정보와 주민의 주거 관련 정보의 관리를 바탕으로 하여 재난에 관한 정보를 처리할 수 있는 통합된 시스템의 체계를 갖추고 있어야 한다. 특히, GIS는 3 차원의 공간 데이터와 시간에 따라 변화된 정보를 체계적으로 관리할 수 있는 장점이 있다.

GIS가 재난에 대응할 수 있는 dispatch system 역할을 수행하기 위해서는 다음과 같은 사항을 만족시켜야 한다.

- o 정보 시스템이 이동 가능 할 것
- o 비전문가도 사용할 수 있을 것
- o 평상시에도 활용되고 있을 것
- o 다른 정보 시스템 특히, 데이터베이스와 연계되어 있을 것

2. UDMS(Urban Disaster Management System)의 개요

◇ Find the parcels of the gas station reported

≫ Display the map showing the gas station area.

≫≫ Select Query > Query Condition menu for the map covering the area

[GUI] : Query Condition : district, scale >> [DATA LIST]

▶ 지역 / 축척 선택

≫≫≫ Display the selected map

[MAP DISPLAY]

▶ 지도 Display : - 행정 구역(구/동 경계) Layer

도로(고속도로, 도로#1,도로#2,도로#3) Layer

하천 Layer

≫≫ Select and show the gas station on the map

≫≫≫ Select Query > Query Condition menu item from the DB Manager

[GUI] : Query Condition : property name, area

▶ 주유소/ 지역 선택

≫≫≫ Retrieve the properties of the gas station

[DATA LIST] : Pop-up for Data List

▶ 주유소 List Pop-up

▶ 지도에 주유소 Display

≫≫≫ Highlight the gas station in the map

[MAP DISPLAY]

▶ 해당 주유소 Pan / Zoom

▶ 해당 주유소 Attribute Data 검색

: 주유소명, 소유자, 주소, 주유소 개요(주유기,저장용량)

위험반경(M), PID

◇ Create a buffer zone around the gas station and overlay

≫≫ Create a buffer zone

[GUI] : Select the Utilities > Zone > Create > By Corridor menu

: object, buffer width, zone name

≫≫ Show the zone and the gas station in the map

[MAP DISPLAY]

▶ 지도에 위험 지역 표시

≫≫ Overlay the zone on the polygons

≫≫≫ Select the MapQuery > By Zone menu

[GUI] : Map Query by Zone : zone name, feature type

≫≫≫ Display the map with the parcels shaded

[MAP DISPLAY]

▶ 위험 지역 내 주요 건물 Display

≫≫≫ Retrieve the parcels(residential district) to be affected

≫≫≫≫ Select Query > Query Condition menu (nearest polygon searching)

[GUI] : Query Condition : parcel name, conditions(residential)

▶ 집단 주민 거주 건물(아파트 등) 검색 :

아파트 명, 동 별 거주 인구, PID

≫≫≫≫ Compute the population of the district

[DATA LIST] :

▶ 거주 인구 파악

≫≫≫ Search the safe places to accommodate the people

≫≫≫≫ Select Query > Query Condition menu

[GUI] : Query Condition : nearest & wide, outside the dangerous zone

>> [DATA LIST]

▶ 대피 가능 장소(학교) 검색

: 학교 명, 운동장 넓이(Area Measurement 대체 가능), 급수 시설, PID

≫≫≫≫ Select the suitable polygon

▶ 대피 장소 선정

≫≫≫≫ Highlight the alternate locations in the map

▶ 대피 장소 Pan / Zoom

◇ Find the suitable hospitals to locate the patient

≫≫ Select Query > Query Condition menu

[GUI] : Query Condition : hospital >>

[DATA LIST]

▶ 병원 검색 :

병원 명, 주소, 병원 구분, Bed 수, PID

▶ 병원 Layer Display

≫≫ Check the capacities of the hospital

≫≫ Select the suitable hospitals

▶ 환자 이송 병원 선정

≫≫ Find the shortest path along arc or chain as network analysis

[GUI] : Network Analysis

▶ 병원까지의 거리, 최단 Path설정, 후송 시간 예측

제 2 절 네트워크상의 공간 분석 및 통계 처리

초기 단계의 지리 정보 시스템은 의사 결정을 지원하기 위한 지리 자료의 획득, 저장, 처리, 그리고 표현을 중심으로 개발되었다. 그러나, 이들 기능만으로는 지리 정보 시스템에 대한 사용자의 요구를 충족시켜 줄 수 없을 뿐만 아니라 이들 기능들은 데스크탑용 매핑 소프트웨어가 모두 가지고 있는 기능들이다. 지리 정보 시스템을 개발하는 전세계 회사 중 가장 많은 영향력을 가지고 있는 미국의 ESRI사의 정의에 의하면 완전한 지리 정보 시스템이 되기 위해서는 위 4가지 기능 외에 반드시 공간 분석 기능을 제공하여야 한다. 특히, 공간 분석 기능은 매핑 소프트웨어, 데이터베이스 시스템, CAD 시스템과 지리 정보 시스템을 구별해주는 중요한 요소 중 하나이다.

공간 분석은 공간 데이터를 다른 형태로 처리 또는 조작하여, 결과적으로 또 다른 의미를 이끌어 내는 과정이다. 파생된 데이터를 개발하기 위한 분석 방법으로 중첩 분석 (Overlay Analysis), 근접성 분석(Near Analysis), 포함성 분석(Containment Analysis), 인접성 분석(Adjacency Analysis), 연결성 분석(Connectivity Analysis), 영향권 분석(Buffering Analysis), 네트워크 분석(Network Analysis) 등이 있다.

지리 정보 시스템의 공간 분석은 3가지 형식의 기능이 있다. 이 기능들은 특성에 대한 질의(비 공간 속성 정보에 대한 질의), 공간 자료에 대한 질의, 기존의 자료를 이용한 새로운 자료들의 집합을 만드는 기능이다. 공간 분석 기능은 공간 자료들에 대한 단순한 질의에서부터 복잡한 속성이나 공간 자료의 결합, 새로운 자료의 생성 등에 이르기까지 많은 영역에 걸쳐 발생한다.

공간 분석에 있어서 속성에 대한 질의와 공간 자료에 대한 질의는 비슷한 분야이다. 속성에 대한 질의는 공간 자료는 처리하지 않고 속성 정보에 대한 처리를 수행하는 것이다. 예를 들어, 어떤 도시의 토지 구획도가 데이터베이스

에 저장되어 있고, 모든 토지 구획은 토지 사용과 관련된 부호를 가지고 있다고 가정하면, 단순한 속성에 대한 질의는 어떤 특정한 토지 사용에 대하여 일어날 수 있다. 이러한 질의에 대하여 시스템은 단순히 토지 구획도가 저장된 테이블을 검색하면서 각 토지 구획의 토지 사용 부호를 검사하여 질의된 특정한 토지 사용과 일치하는 토지 구획을 추출하면 된다. 이러한 과정에서 공간 자료에 대한 검색은 필요없다. 따라서, 이 속성 질의는 주로 지리 객체의 식별에 많이 사용된다.

공간 자료 질의는 공간 정보에 대한 처리가 발생한다. 예를 들어, 간선 도로로부터 1Km 이내에 속한 토지에 대한 질의가 발생할 수 있다. 이러한 질의에 답하기 위하여 시스템은 간선 도로의 공간 정보를 이용하여 간선 도로의 위치를 파악하고 각 토지의 공간 정보를 이용하여 간선 도로와 거리를 계산하여 이 거리가 주어진 1Km이내에 존재하는 토지 구획들만을 추출하여야 한다.

새로운 자료 집합을 만드는 기능은 영향권 설정 기능이 대표적이다. 예를 들어, 원자력 발전소에서 핵폭발이 발생하였을 때 핵폭발의 영향이 미칠 수 있는 영향권의 범위를 계산하여 주민들을 이 영향권 밖으로 대피시켜야 한다. 이 영향권은 기존의 속성 정보나 공간 정보만을 가지고는 알 수 없으며, 기존의 공간 정보를 이용하여 새롭게 계산되는 정보이다.

1. 공간 분석 기능

본 연구에서 지원하는 공간 분석 기능은 근접성 분석(Near Analysis), 포함성 분석(Containment Analysis), 인접성 분석(Adjacency Analysis), 연결성 분석(Connectivity Analysis), 영향권 분석(Buffering Analysis), 네트워크 분석(Network Analysis) 등 6가지이다. 또한 본 연구는 객체 지향 기법을 이용한 피처를 기반으로 하는 시스템이다. 따라서, 모든 공간 분석 기능은 피처를 기

준으로 구현되었으며, 다음과 같은 피처를 사용하였다.

노드 피처	EtcBuildingN	기타 빌딩
	GasStationN	주유소
	HospitalN	개인 및 종합병원
	PublicBuildingN	공공기관(경찰서, 관공서 등)
체인 피처	Road#1CentC	8~6차선 도로의 중앙선
	Road#2CentC	4~2차선 도로의 중앙선
	RoadExpCentC	고속도로의 중앙선
폴리곤 피처	APTComplexP	아파트 단지 경계
	ExpoP	엑스포 구역 경계
	InnerBuildingP	아파트단지 등의 내부 건물들
	InstituteP	연구단지 내 연구소들
	SchoolP	초,중,고,대학교

가. 근접성 분석

근접성 분석은 총 9개의 기능으로 세분화된다. 이 기능에서 기준(Source) 객체와 대상(Destination) 객체를 설정하여야 한다. 기준 객체가 될 수 있는 객체는 노드 객체, 체인 객체, 다각형 객체가 될 수 있고, 대상 객체도 노드 객체, 체인 객체, 다각형 객체가 될 수 있기 때문에 총 9개의 기능이 제공된다. 붉은 색은 기준 객체를 나타내고, 노란 색은 대상 객체들 중에서 공간 분

석의 결과가 되는 객체를 나타내고, 분홍색은 공간 분석의 결과 중에서 선택한 객체를 나타낸다. 입력된 거리의 단위는 m이다. 즉, 기준객체를 중심으로 반경 4000m 이내에 있는 대상 객체를 찾는 것이다.

1) 노드 객체와 노드 객체사이의 근접성 분석

기준 객체와 대상 객체가 모두 노드 객체이다.

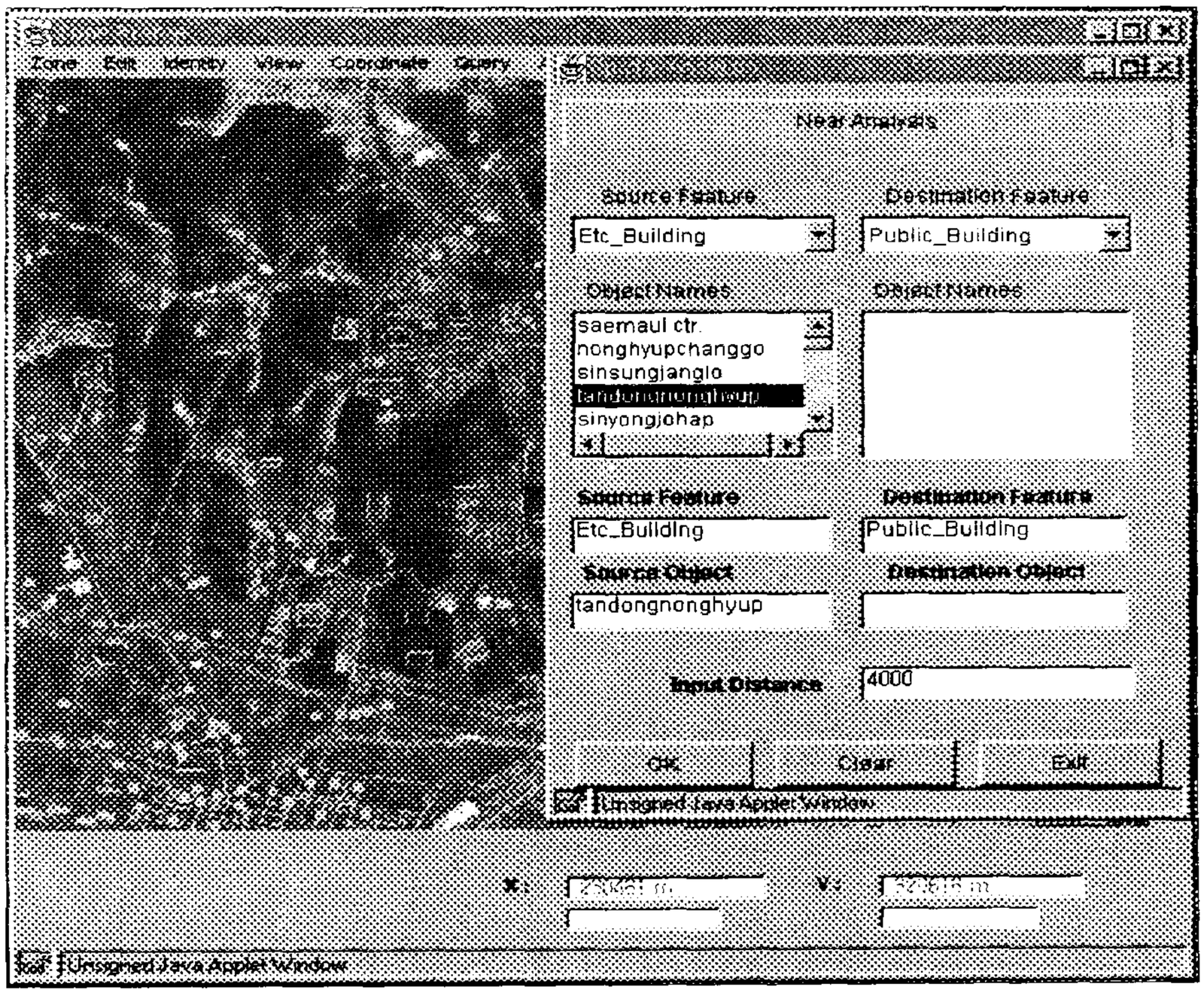


그림 6-1. 노드 객체와 노드 객체 사이의 근접성 분석

2) 노드 객체와 체인 객체 사이의 근접성 분석

기준 객체는 노드 객체이고 대상 객체는 체인 객체이다. 본 예제에서는 기준 객체는 노드 객체인 주유소이고 대상 객체는 도로의 중심선이다.

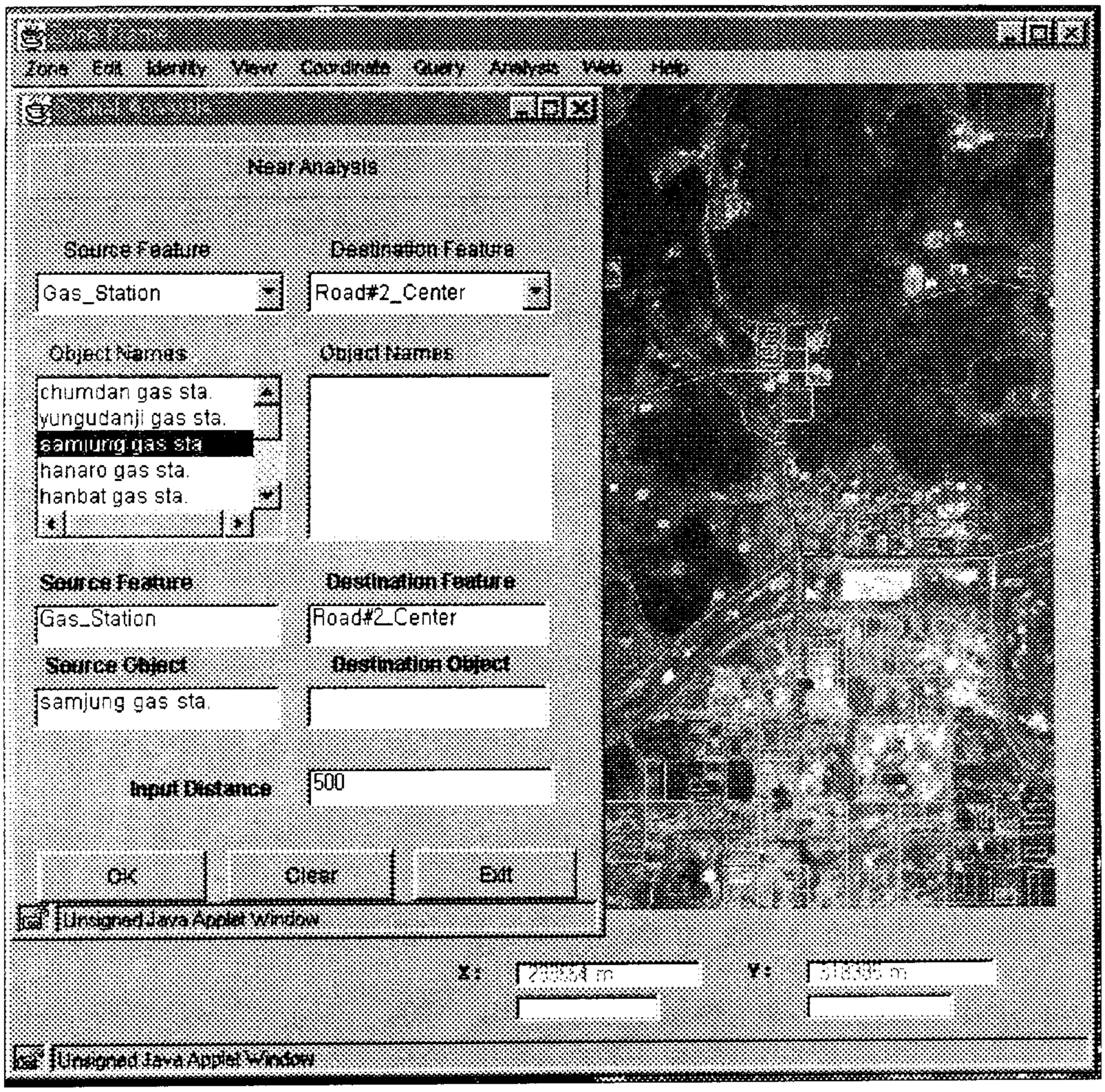


그림 6-2. 노드 객체와 체인 객체 사이의 근접성 분석

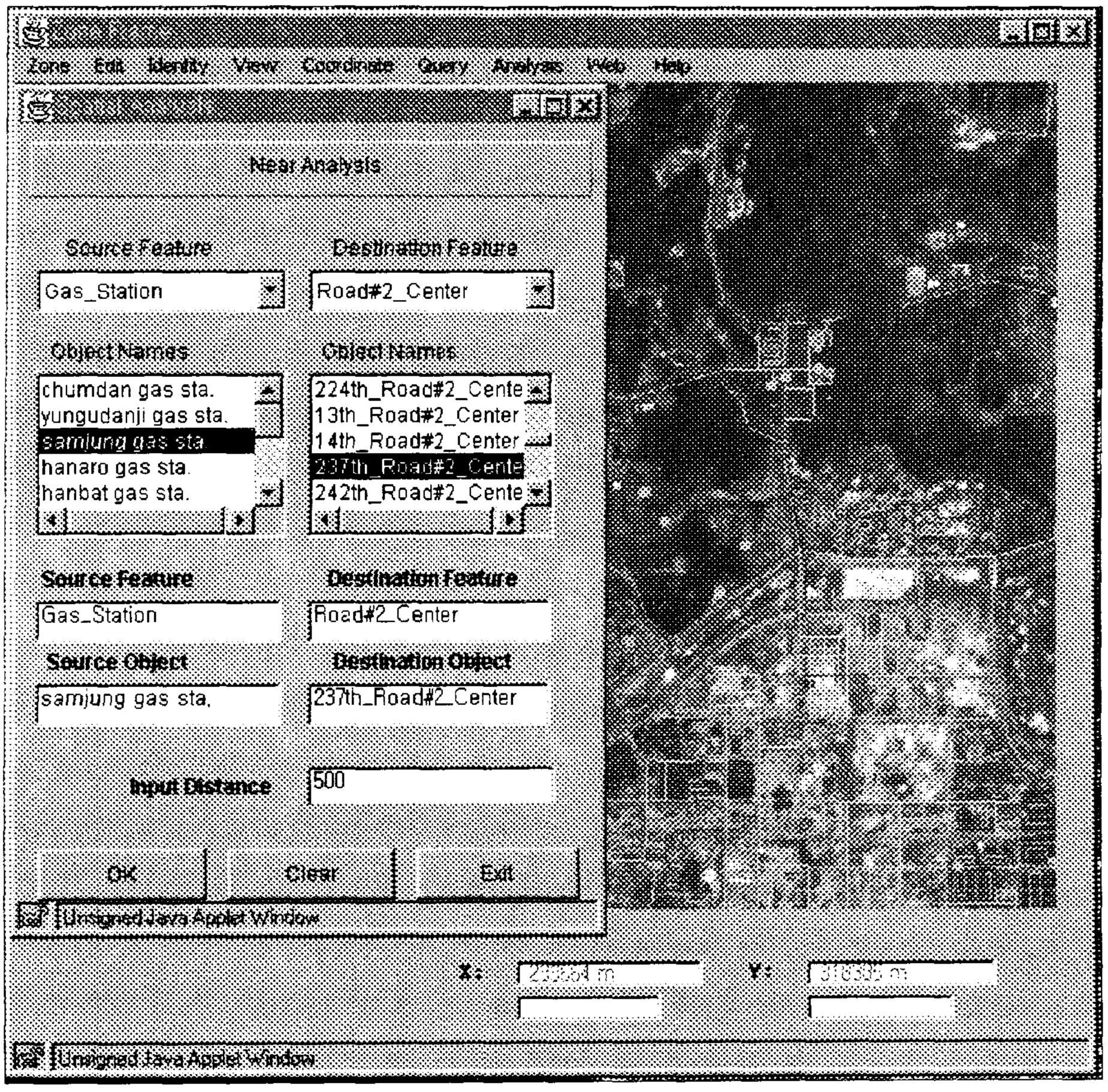


그림 6-3. 노드 객체와 체인 객체 사이의 근접성 분석 결과

3) 노드 객체와 다각형 객체 사이의 근접성 분석

기준 객체는 노드 객체이고 대상 객체는 다각형 객체이다. 본 예제에서는 기준 객체는 노드 객체인 공공 건물이고 대상 객체는 다각형 객체인 내부 건물이다. 입력 반경은 500m로 월평 2동사무소를 기준으로 이 반경 이내에 있는 내부 빌딩을 찾는 것이다.

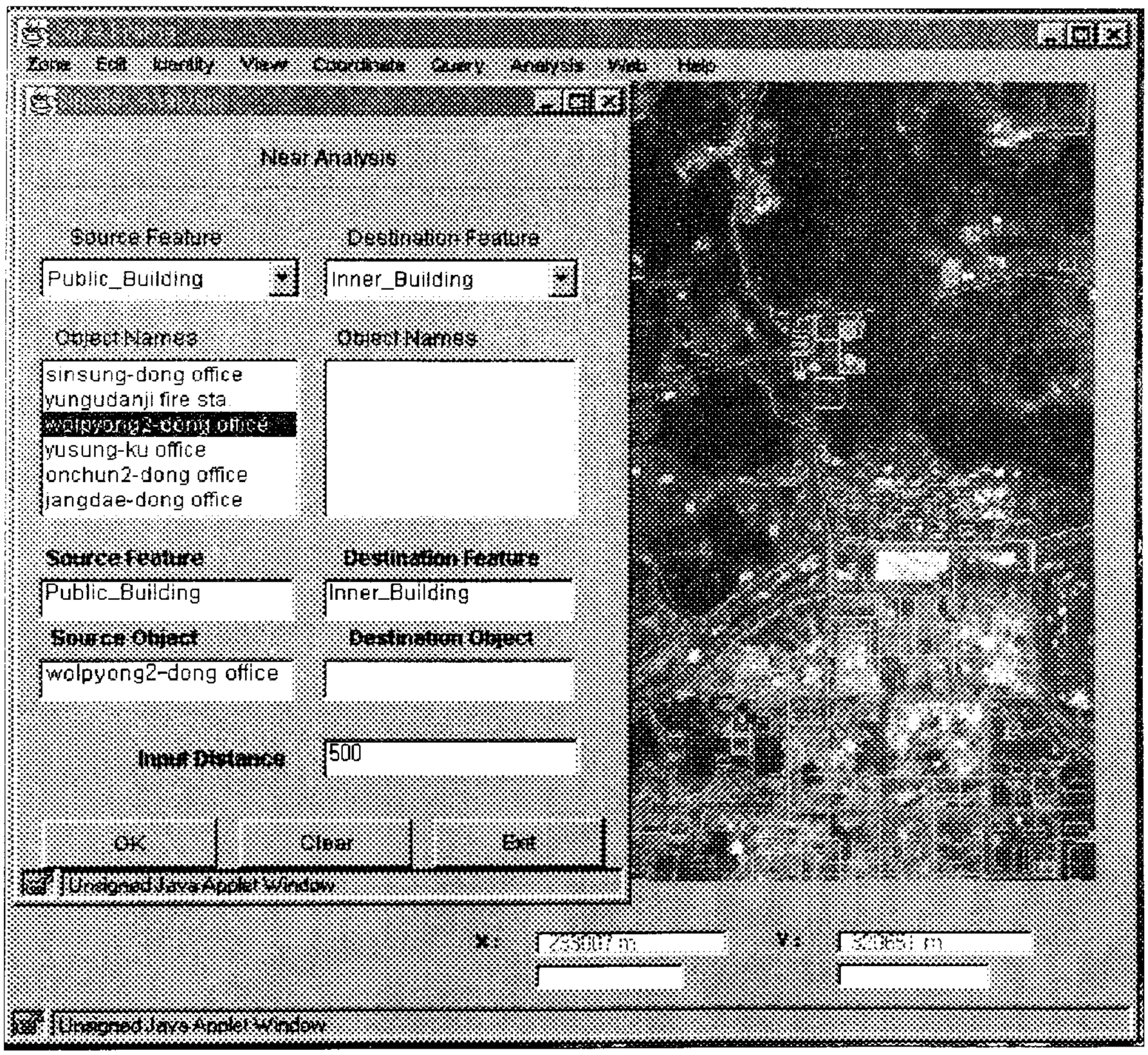


그림 6-4. 노드 객체와 다각형 객체 사이의 근접성 분석

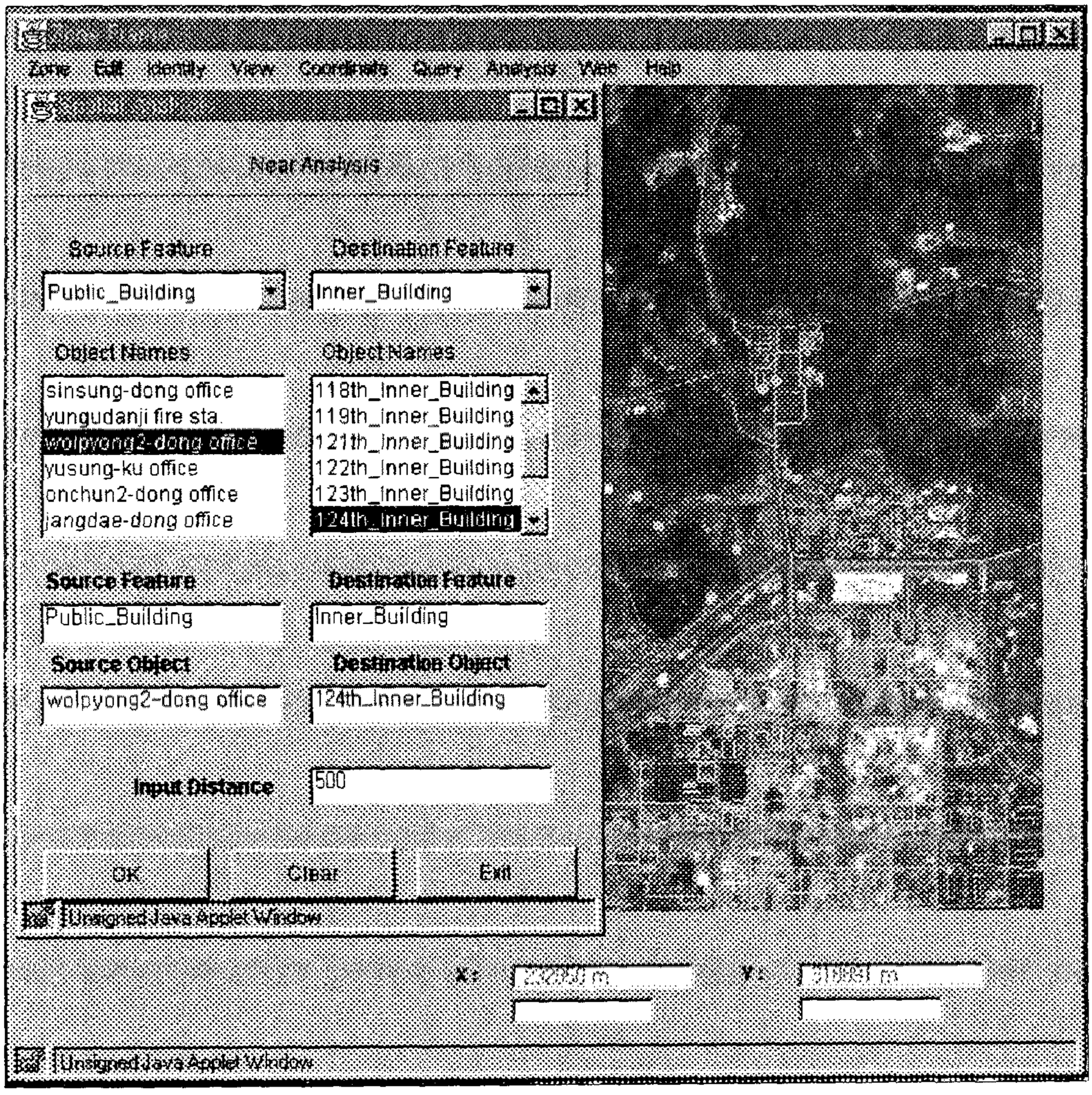


그림 6-5. 노드 객체와 다각형 객체 사이의 근접성 분석 결과

4) 체인 객체와 노드 객체 사이의 근접성 분석

기준 객체는 체인 객체이고 대상 객체는 노드 객체이다. 본 예제에서는 기준 객체는 체인 객체인 도로 중심선이고 대상 객체는 노드 객체인 기타 건물이다. 입력 반경은 300m로 9번 도로 중심선을 중심으로 이 반경 이내에 있는 기타 빌딩을 찾는 것이다.

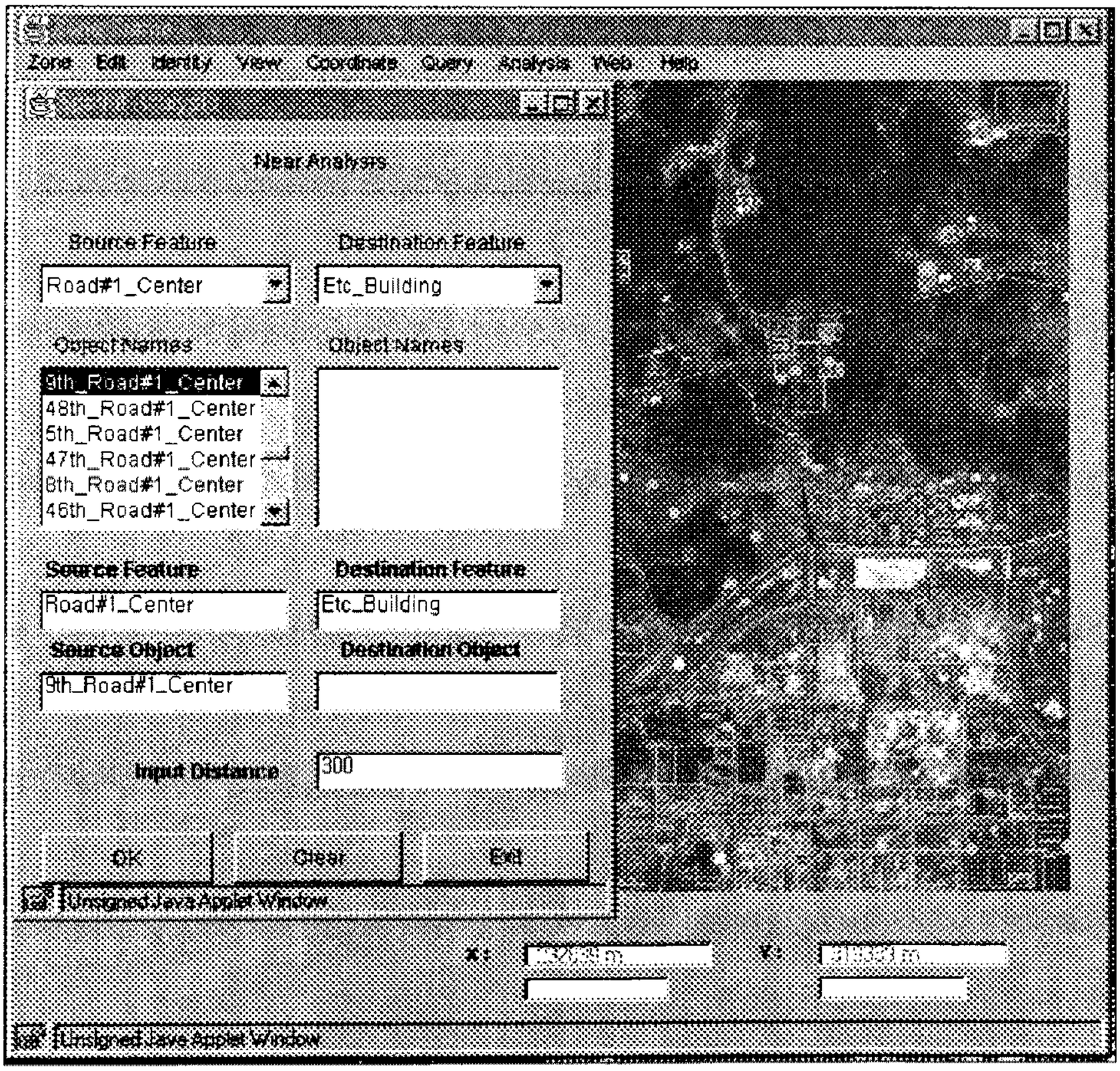


그림 6-6. 체인 객체와 노드 객체 사이의 근접성 분석

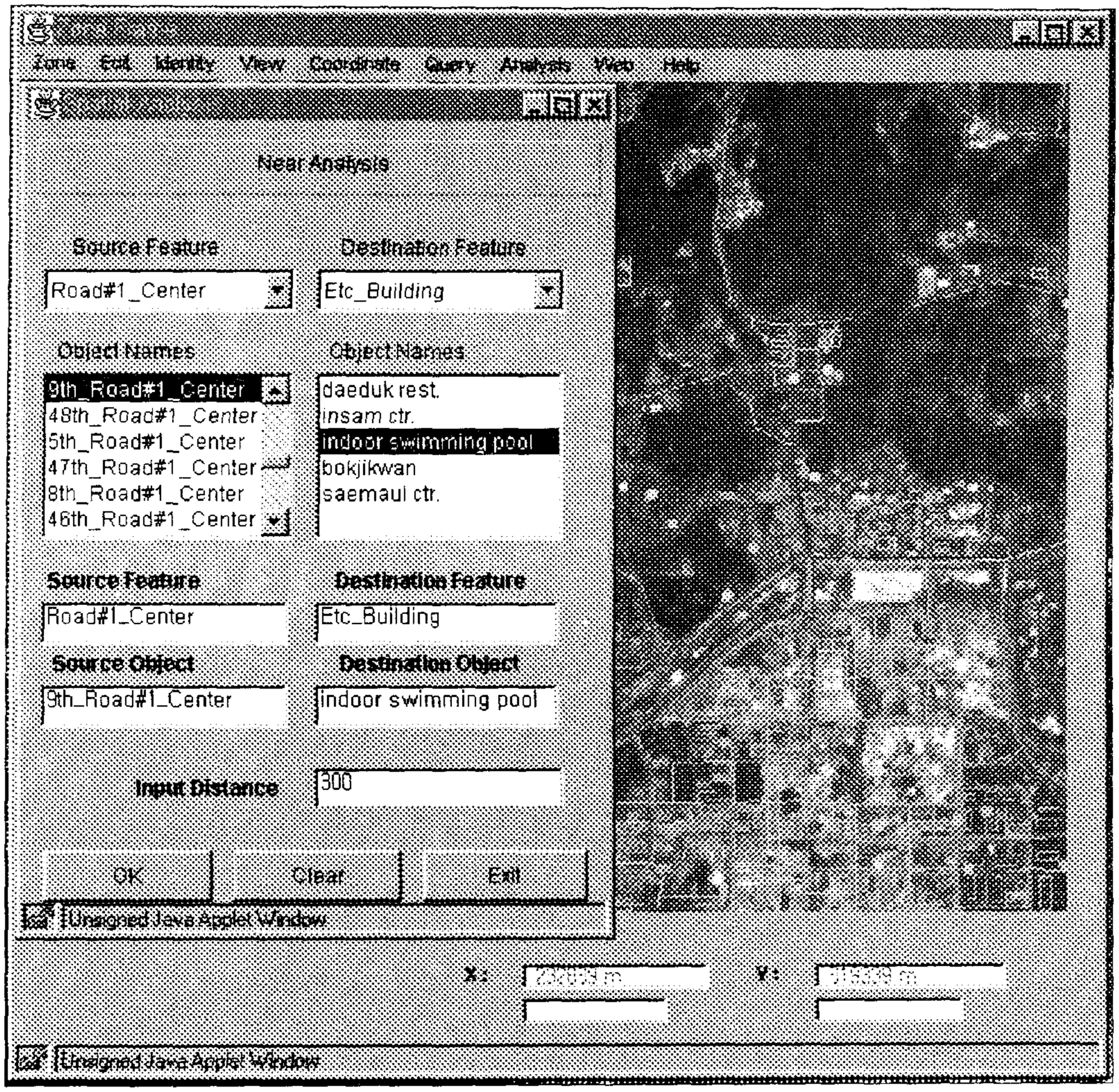


그림 6-7. 체인과 노드 사이의 근접성 분석 결과

5) 체인 객체와 체인 객체 사이의 근접성 분석

기준 객체는 체인 객체이고 대상 객체는 체인 객체이다. 본 예제에서는 기준 객체는 체인 객체인 도로 중심선이고 대상 객체는 체인 객체인 도로 중심선이다. 입력 반경은 120m로 49번 도로 중심선을 중심으로 이 반경 이내에 있는 도로 중심선을 찾는 것이다.

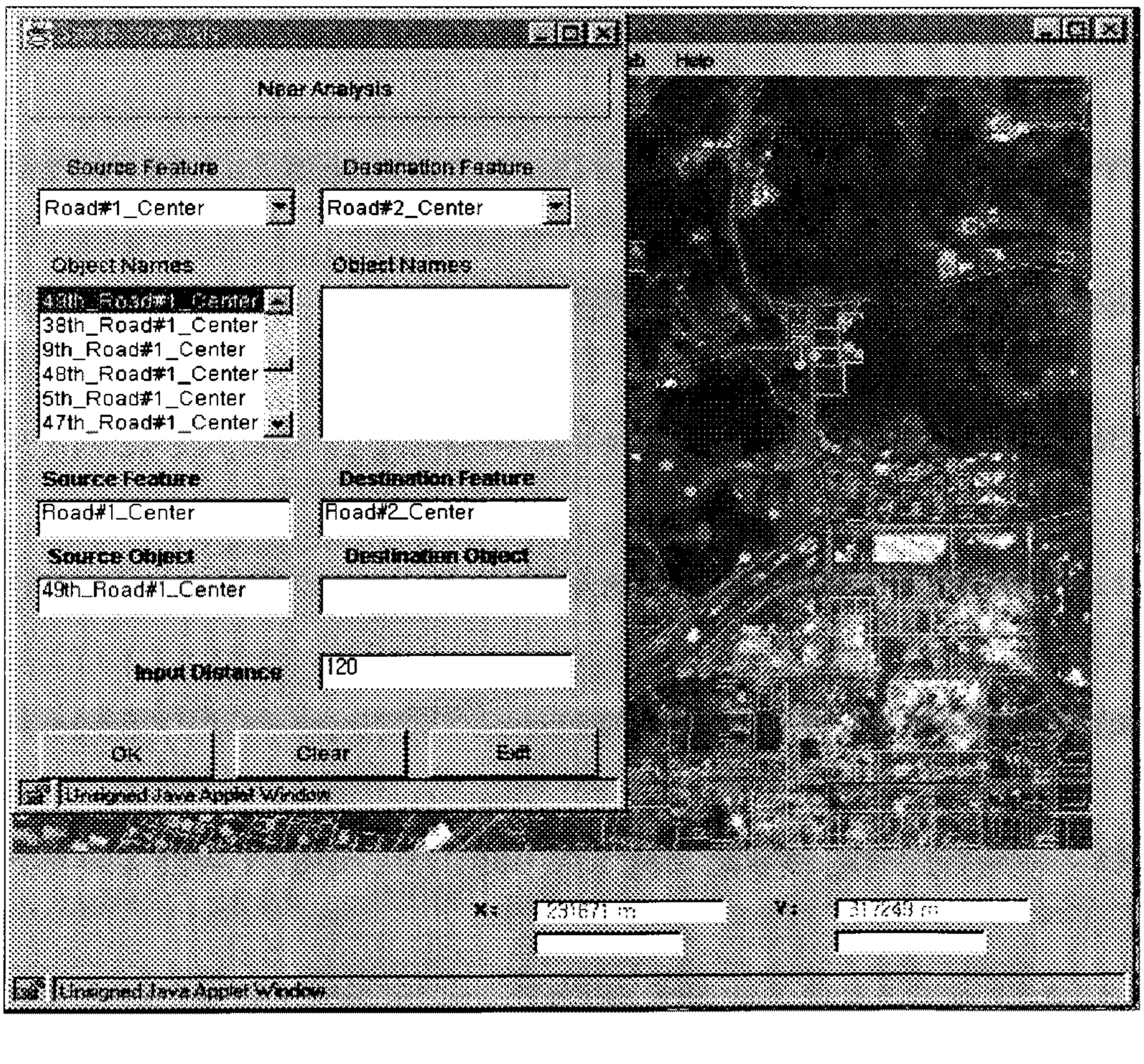


그림 6-8. 체인과 체인 사이의 근접성 분석

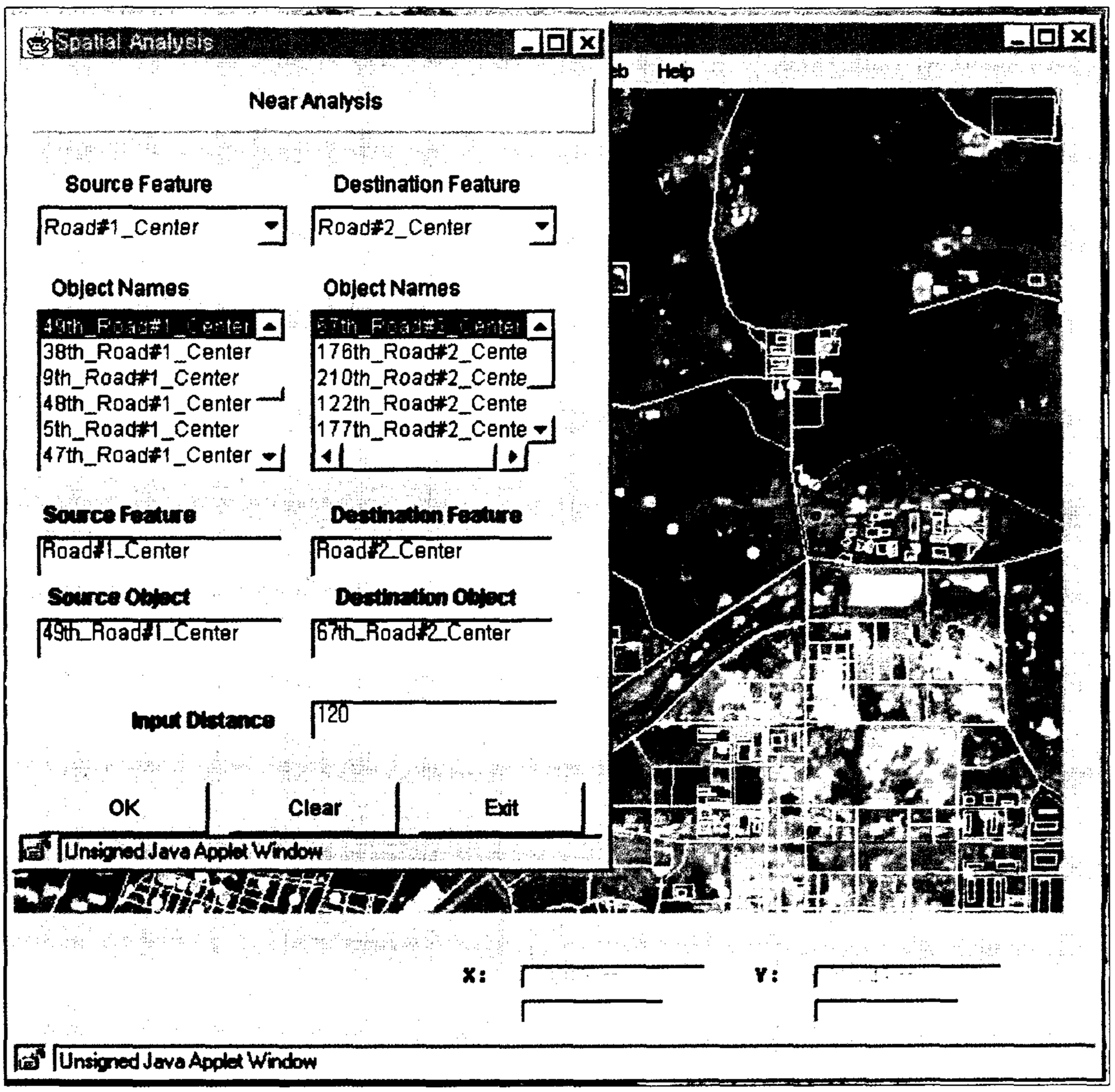


그림 6-9. 체인과 체인 사이의 근접성 분석 결과

6) 체인 객체와 다각형 객체 사이의 근접성 분석

기준 객체는 체인 객체이고 대상 객체는 다각형 객체이다. 본 예제에서는 기준 객체는 체인 객체인 도로 중심선이고 대상 객체는 노드 객체인 내부 건물이다. 입력 반경은 500m로 16번 도로 중심선을 중심으로 이 반경 이내에 있는 내부 빌딩을 찾는 것이다.

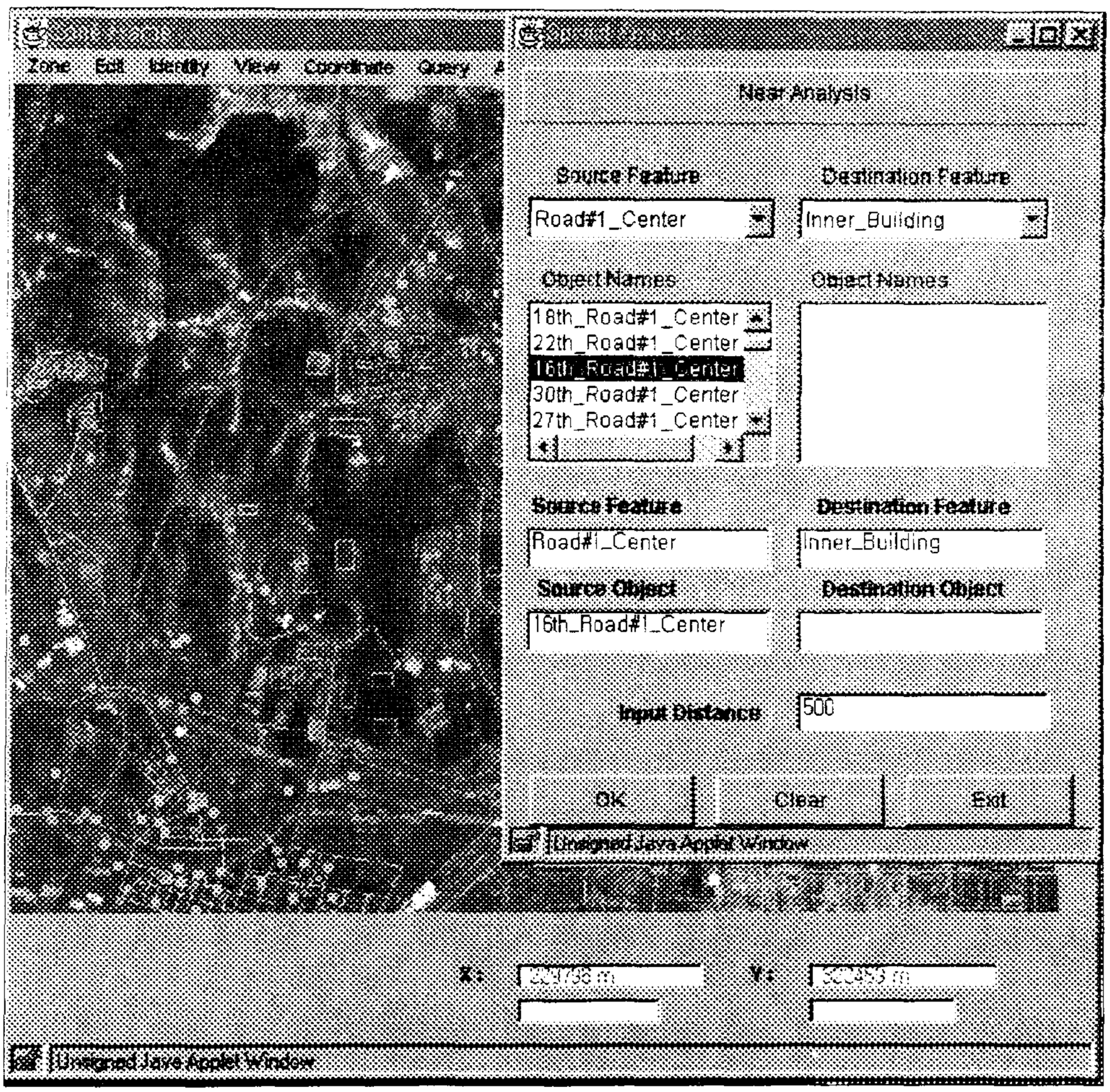


그림 6-10. 체인과 다각형 사이의 근접성 분석

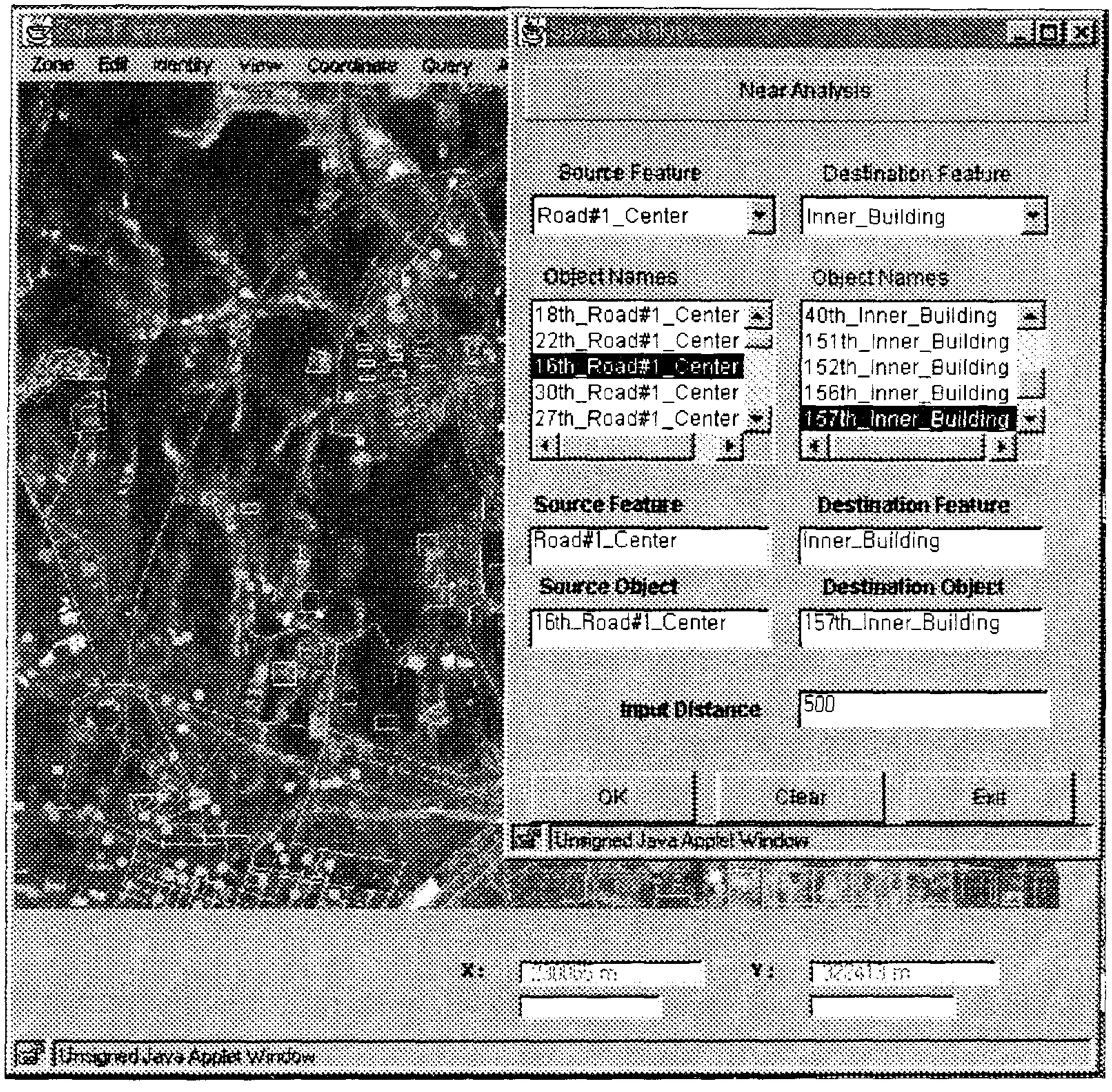


그림 6-11. 체인과 다각형 사이의 근접성 분석 결과

7) 다각형 객체와 노드 객체 사이의 근접성 분석

기준 객체는 다각형 객체이고 대상 객체는 노드 객체이다. 본 예제에서는 기준 객체는 다각형 객체인 아파트 단지이고 대상 객체는 노드 객체인 기타 건물이다. 입력 반경은 600m로 한울 아파트를 중심으로 이 반경 이내에 있는 기타 빌딩을 찾는 것이다.

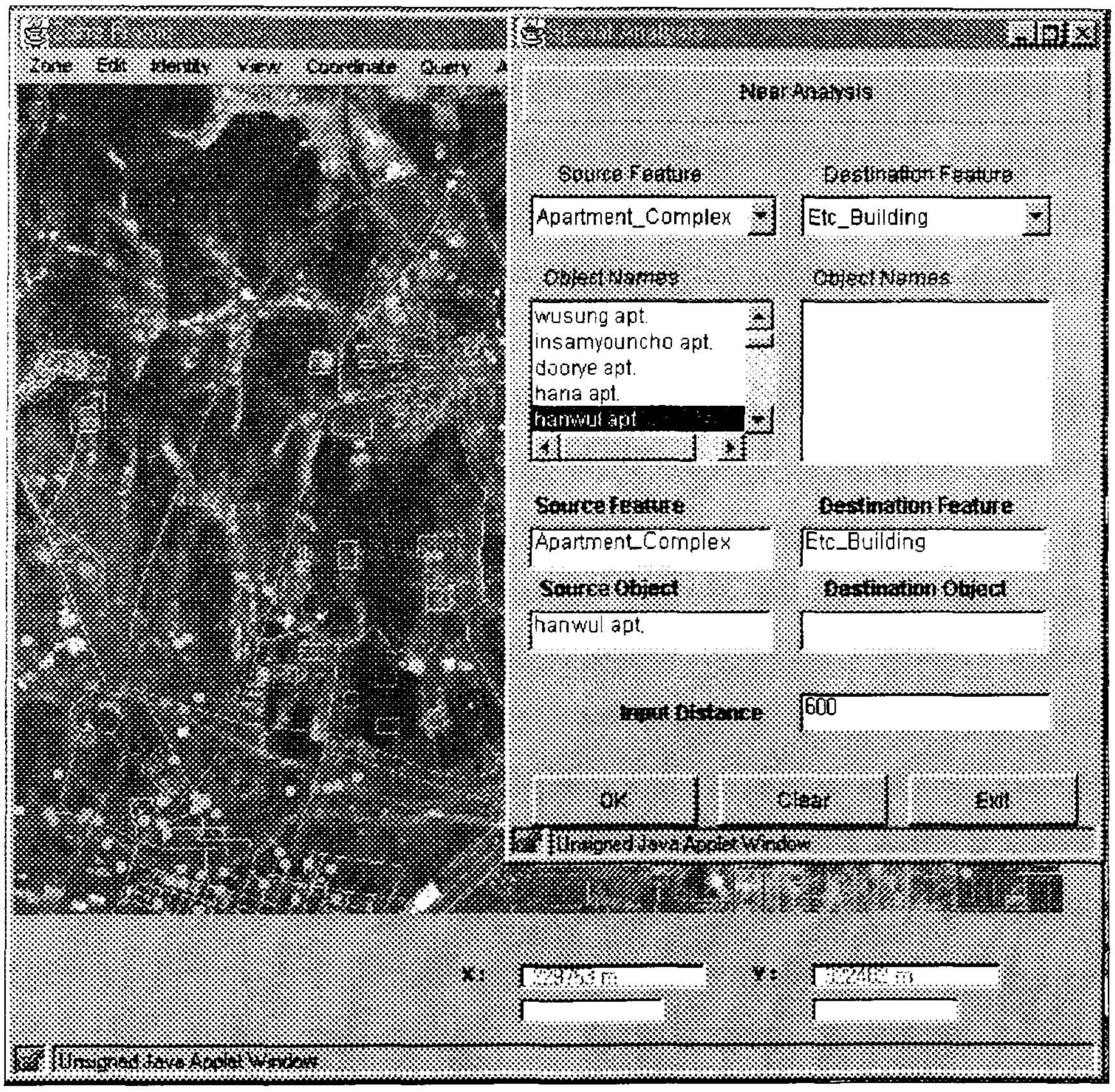


그림 6-12. 다각형 객체와 노드 객체 사이의 근접성 분석

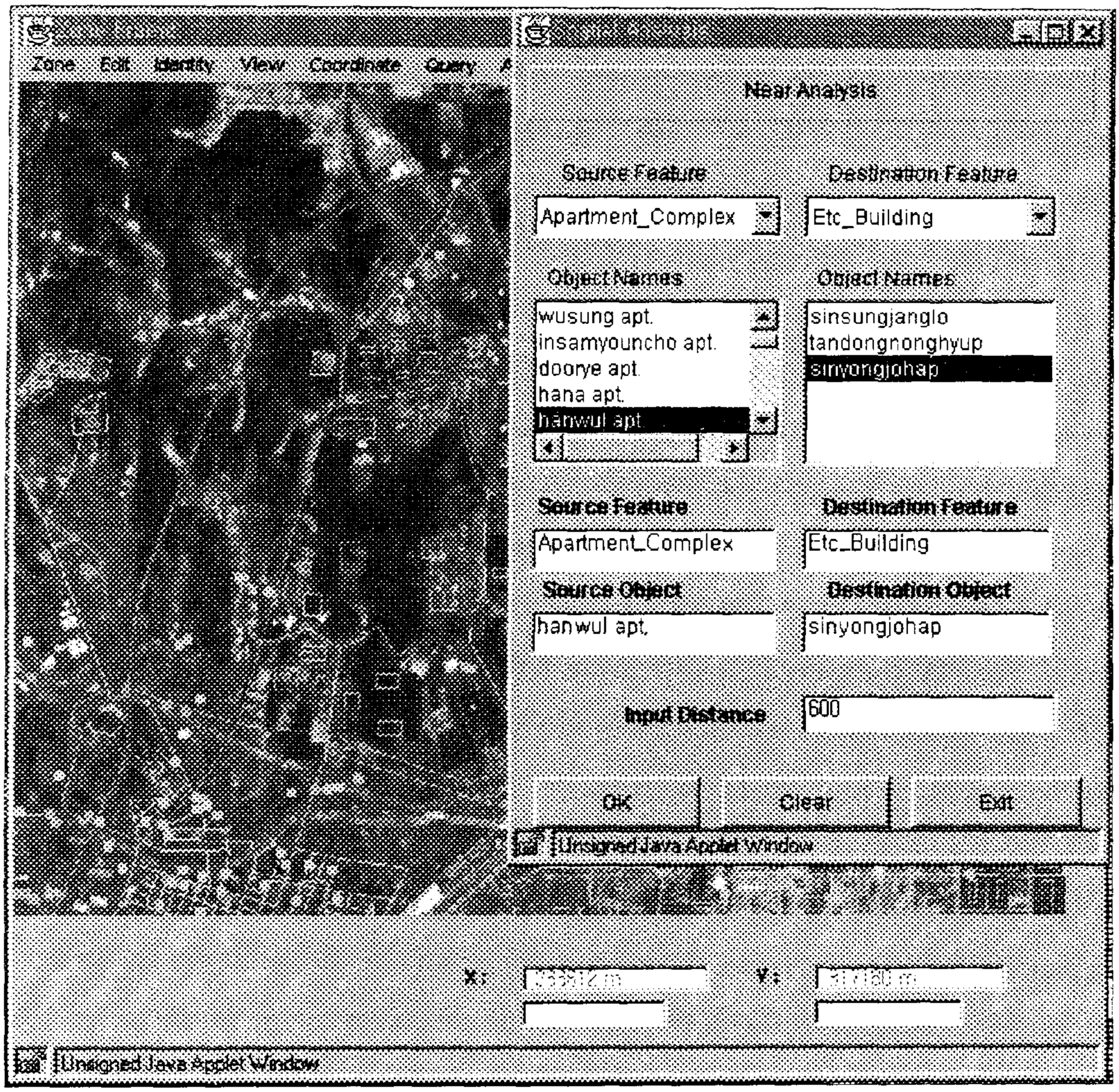


그림 6-13. 다각형 객체와 노드 객체 사이의 근접성 분석 결과

8) 다각형 객체와 체인 객체 사이의 근접성 분석

기준 객체는 다각형 객체이고 대상 객체는 체인 객체이다. 본 예제에서는 기준 객체는 다각형 객체인 아파트 단지이고 대상 객체는 체인 객체인 도로 중심선이다. 입력 반경은 350m로 시스템공학연소를 중심으로 이 반경 이내에 있는 도로 중심선을 찾는 것이다.

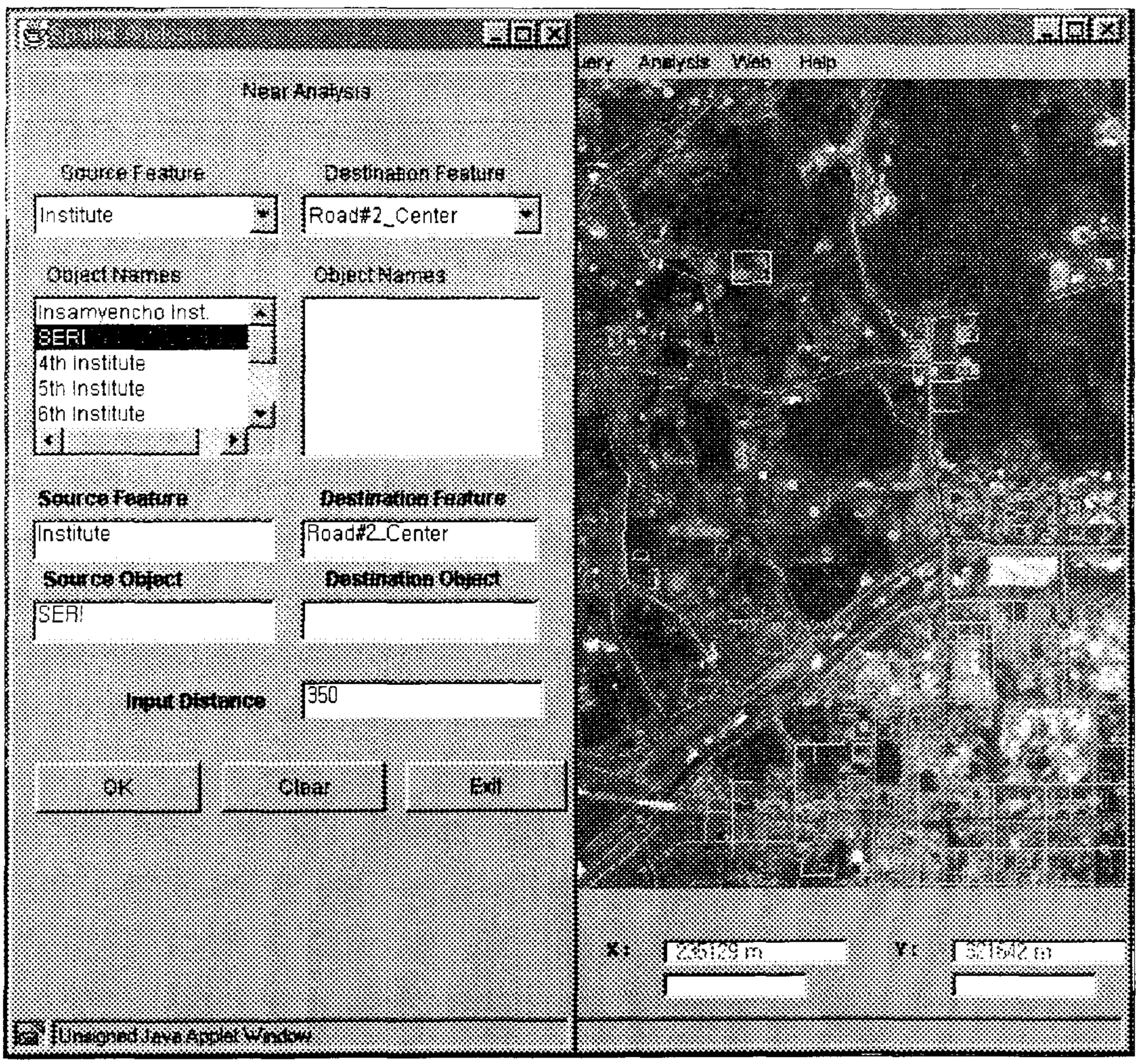


그림 6-14. 다각형 객체와 체인 객체 사이의 근접성 분석

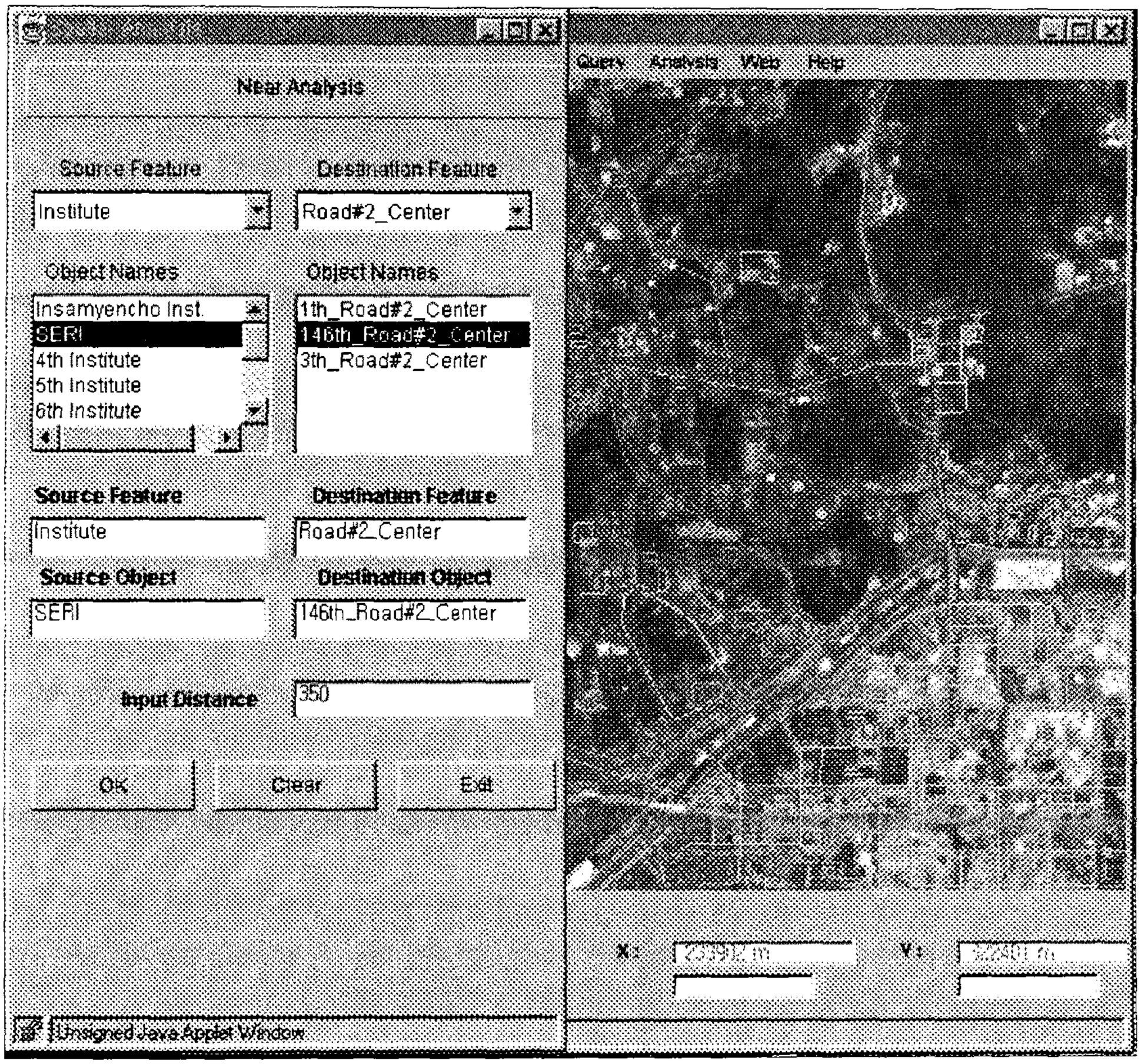


그림 6-15. 다각형 객체와 체인 객체 사이의 근접성 분석 결과

9) 다각형 객체와 다각형 객체 사이의 근접성 분석

기준 객체는 다각형 객체이고 대상 객체는 다각형 객체이다. 본 예제에서는 기준 객체는 다각형 객체인 아파트 단지이고 대상 객체는 다각형 객체인 아파트 단지이다. 입력 반경은 280m로 하나 아파트를 중심으로 이 반경 이내에 있는 아파트 단지를 찾는 것이다.

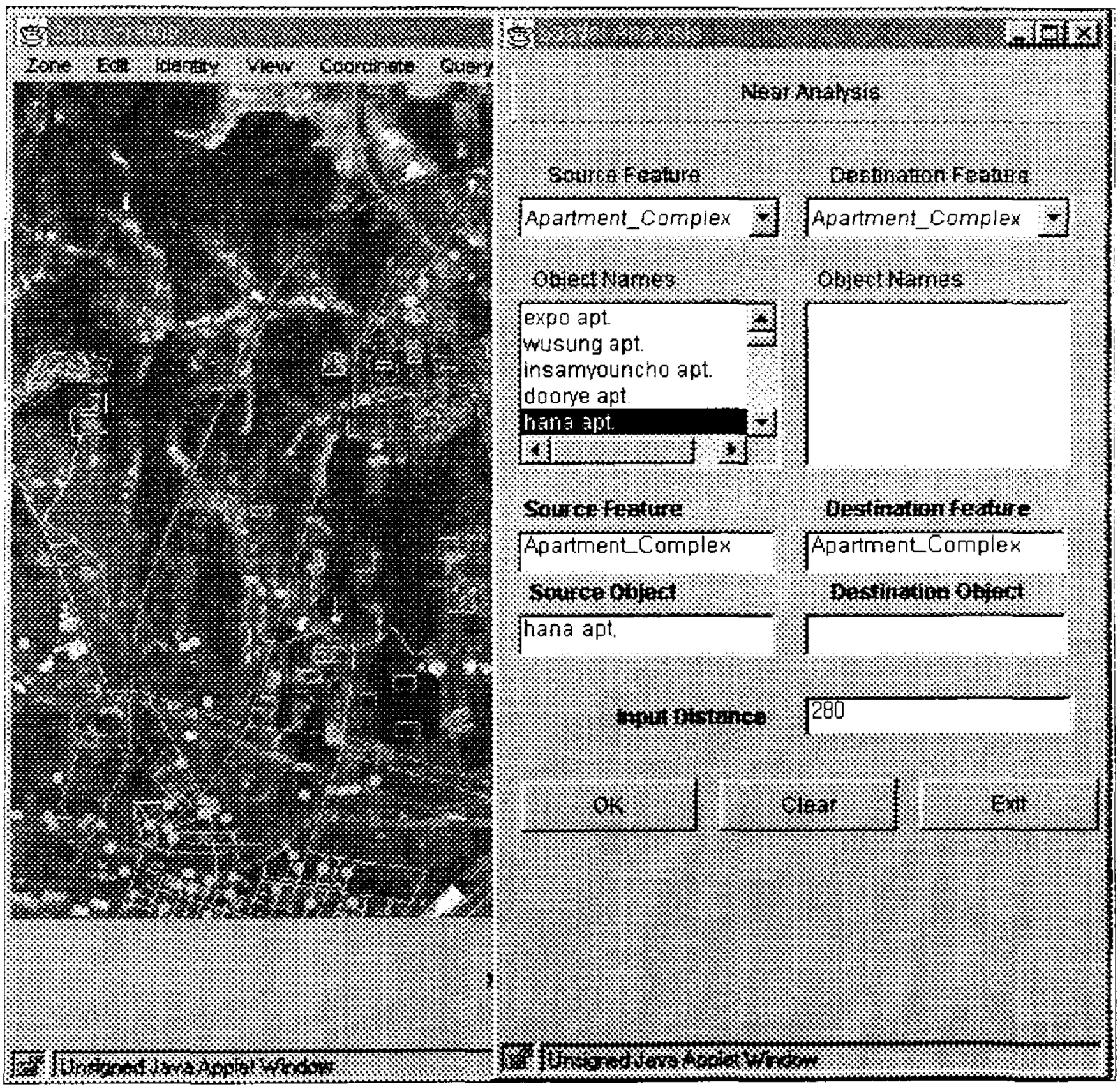


그림 6-16. 다각형 객체와 다각형 객체 사이의 근접성 분석

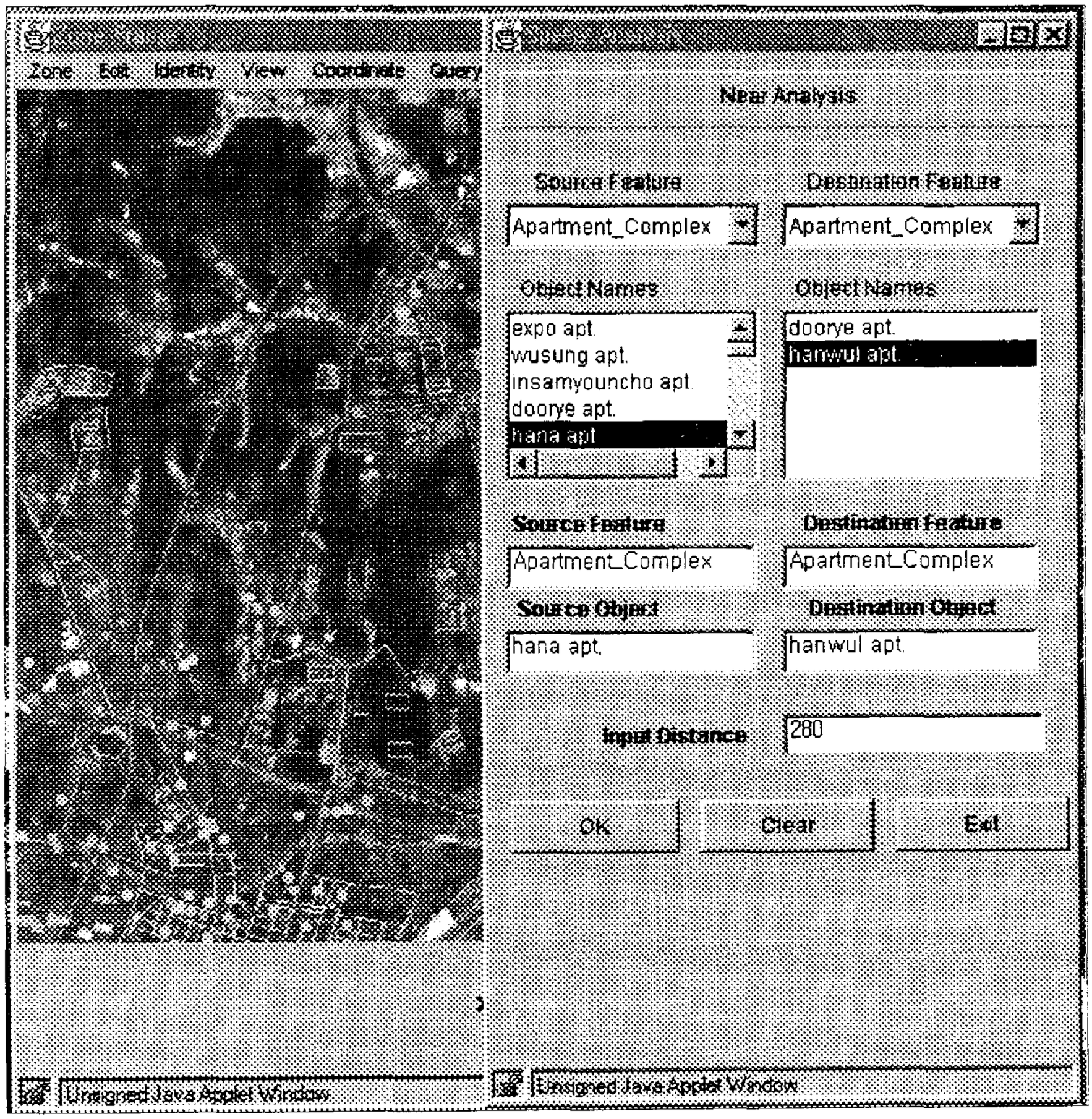


그림 6-17. 다각형 객체와 다각형 객체 사이의 근접성 분석 결과

나. 포함성 분석

포함성 분석에서 기준(Source) 객체와 대상(Destination) 객체를 설정하여야 한다. 기준 객체가 될 수 있는 객체는 다각형 객체만 될 수 있고, 대상 객체는 노드 객체, 체인 객체, 다각형 객체가 될 수 있기 때문에 총 3개의 기능이 제공된다. 붉은 색은 기준 객체를 나타내고, 노란 색은 대상 객체들 중에서 공간 분석의 결과가 되는 객체를 나타내고, 분홍색은 공간 분석의 결과 중에서 선택한 객체를 나타낸다.

1) 다각형 객체와 노드 객체 사이의 포함성 분석

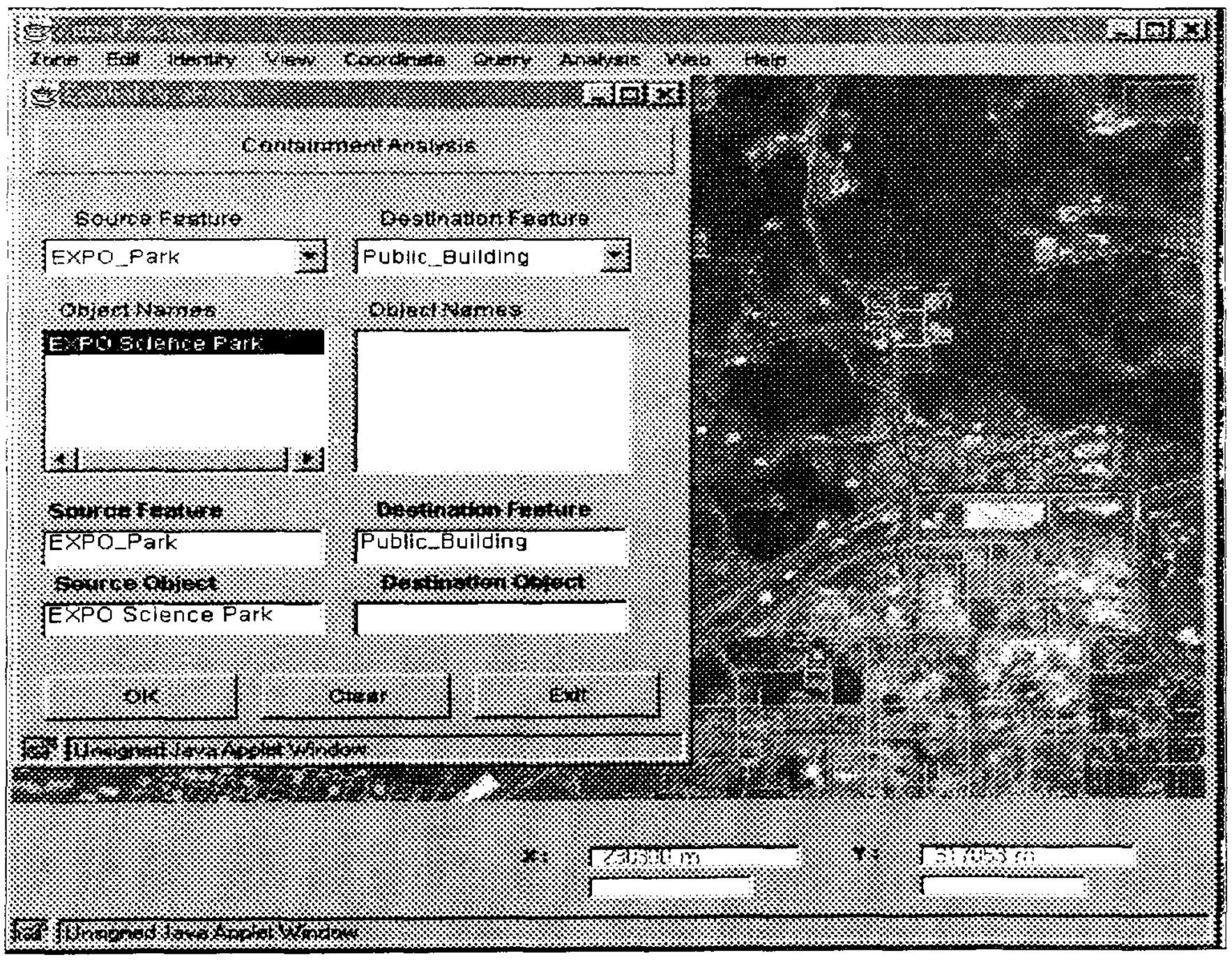


그림 6-18. 다각형 객체와 노드 객체 사이의 포함성 분석

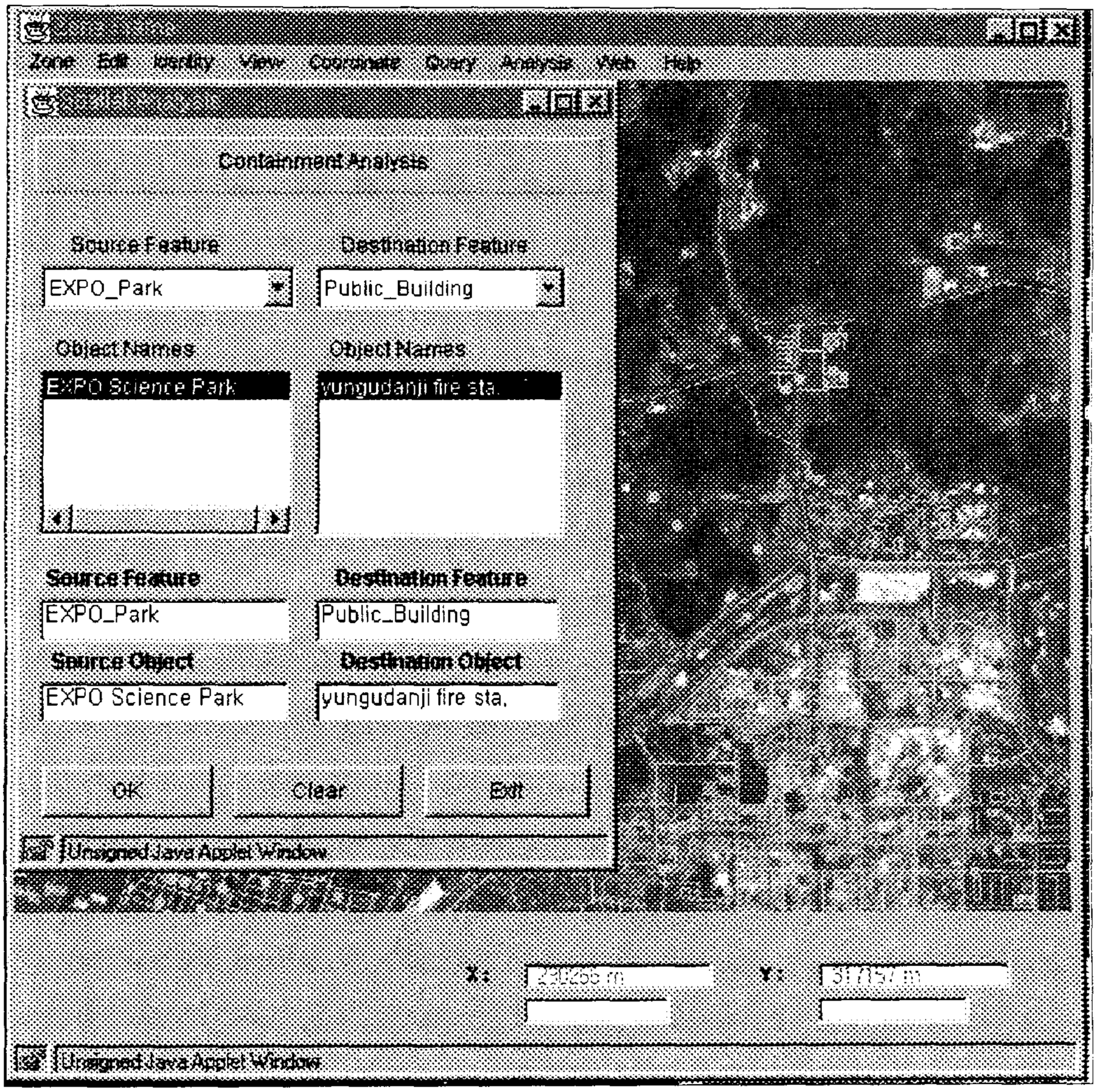


그림 6-19. 다각형 객체와 노드 객체 사이의 포함성 분석 결과

2) 다각형 객체와 다각형 객체 사이의 포함성 분석

본 예제에서는 기준 객체는 다각형 객체인 엑스포 과학 단지이고 대상 객체는 다각형 객체인 내부 빌딩이다. 즉, 엑스포 과학 단지 내부에 있는 빌딩들을 찾는 것이다.

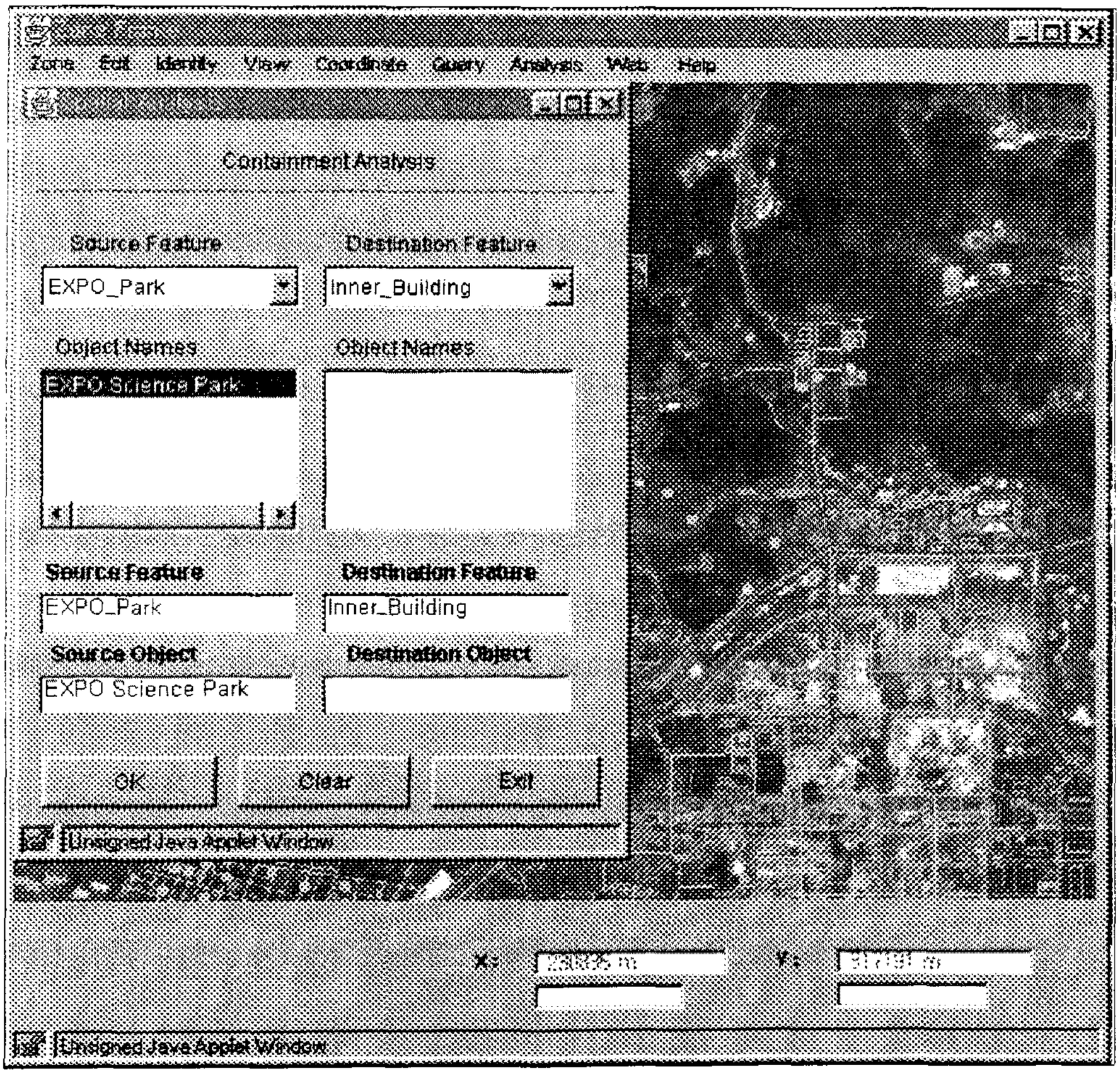


그림 6-20. 다각형 객체와 다각형 객체 사이의 포함성 분석 결과

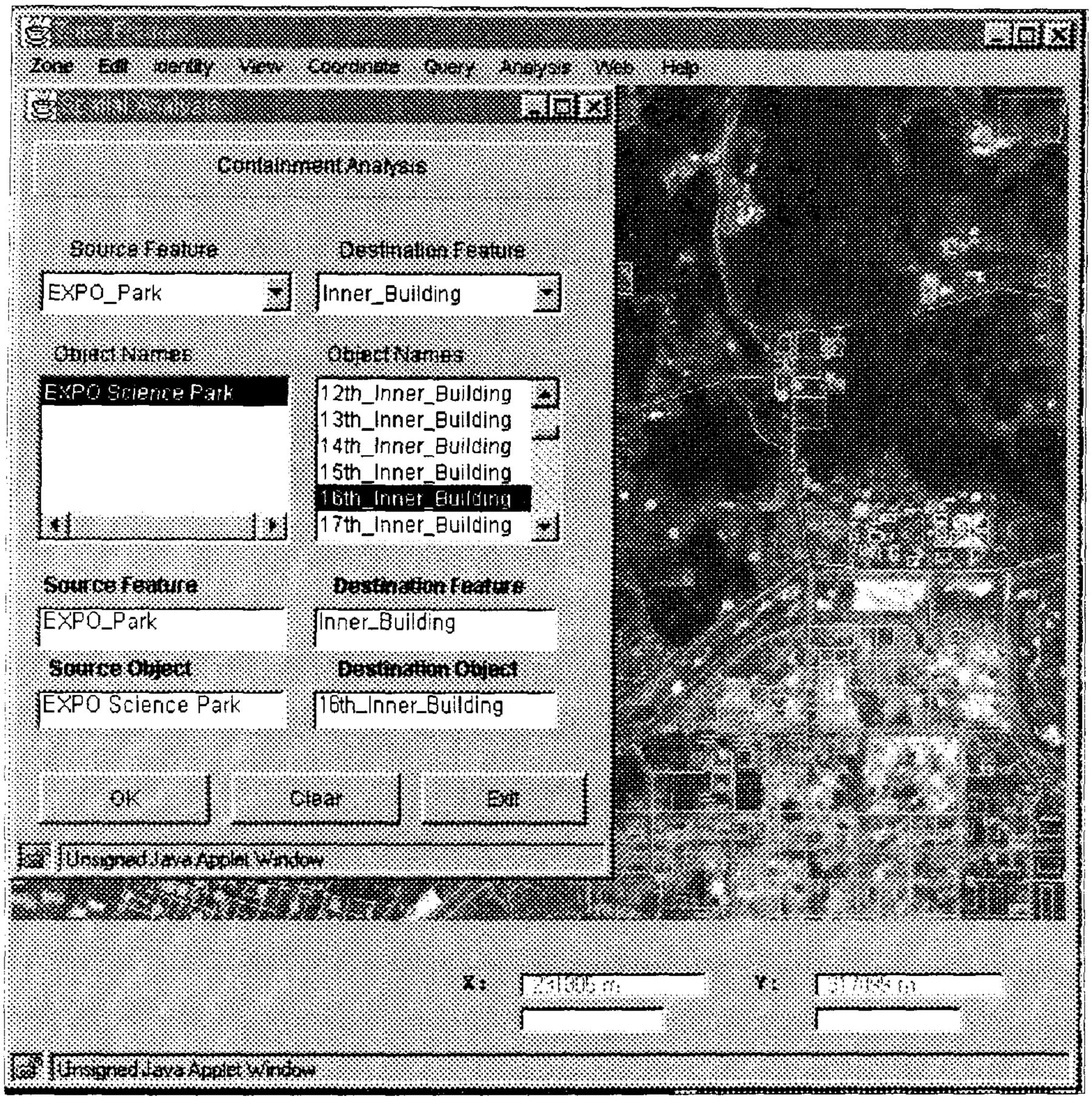


그림 6-21. 다각형 객체와 다각형 객체 사이의 포함성 분석 결과

다. 연결성 분석

이 기능은 기준(Source) 객체와 대상(Destination) 객체를 설정하여야 하는데, 기준 객체와 대상 객체는 체인 객체만 될 수 있다. 붉은 색은 기준 객체를 나타내고, 노란 색은 대상 객체들 중에서 공간 분석의 결과가 되는 객체를 나타내고, 분홍색은 공간 분석의 결과 중에서 선택한 객체를 나타낸다.

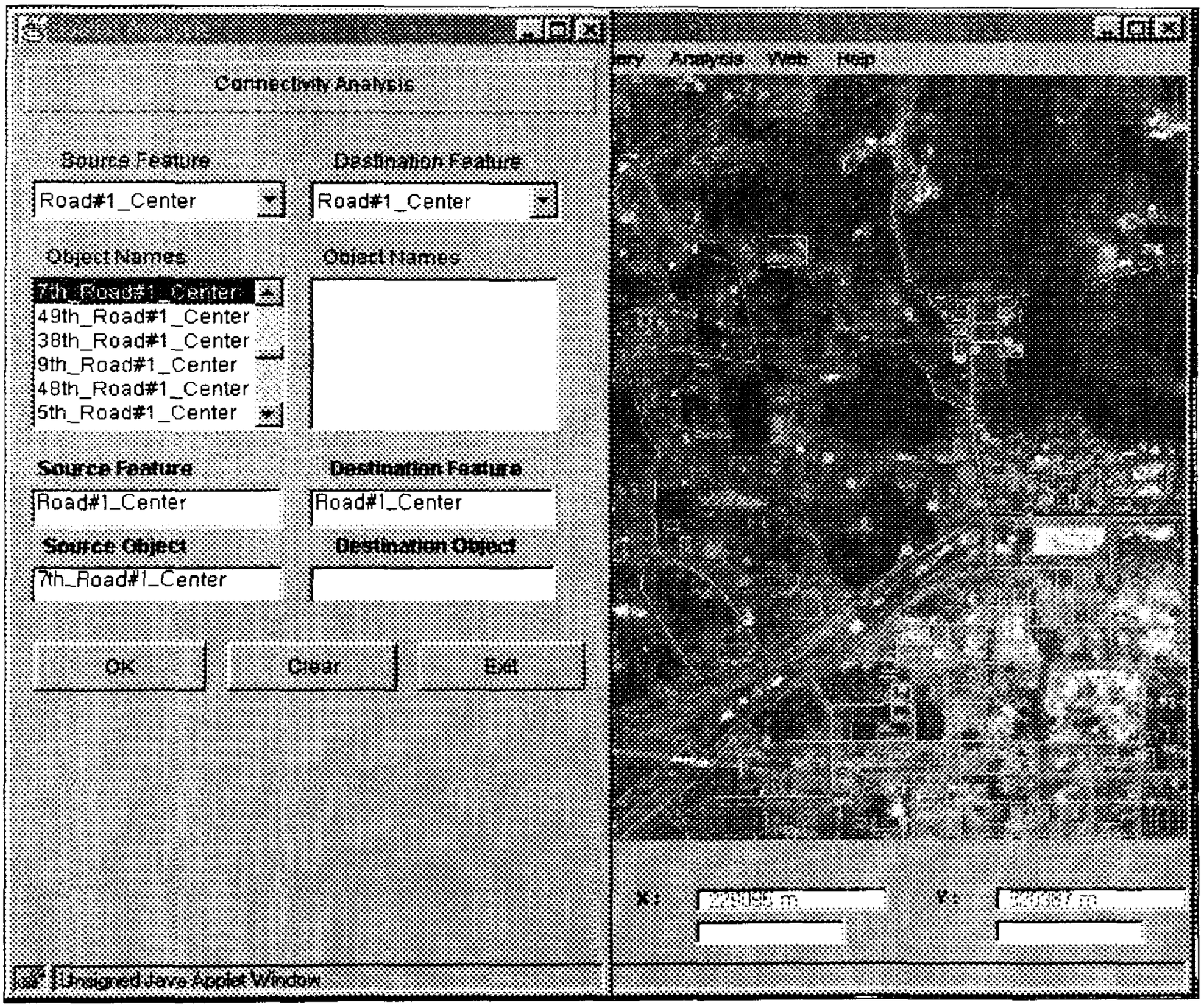


그림 6-22. 체인 객체 사이의 연결성 분석

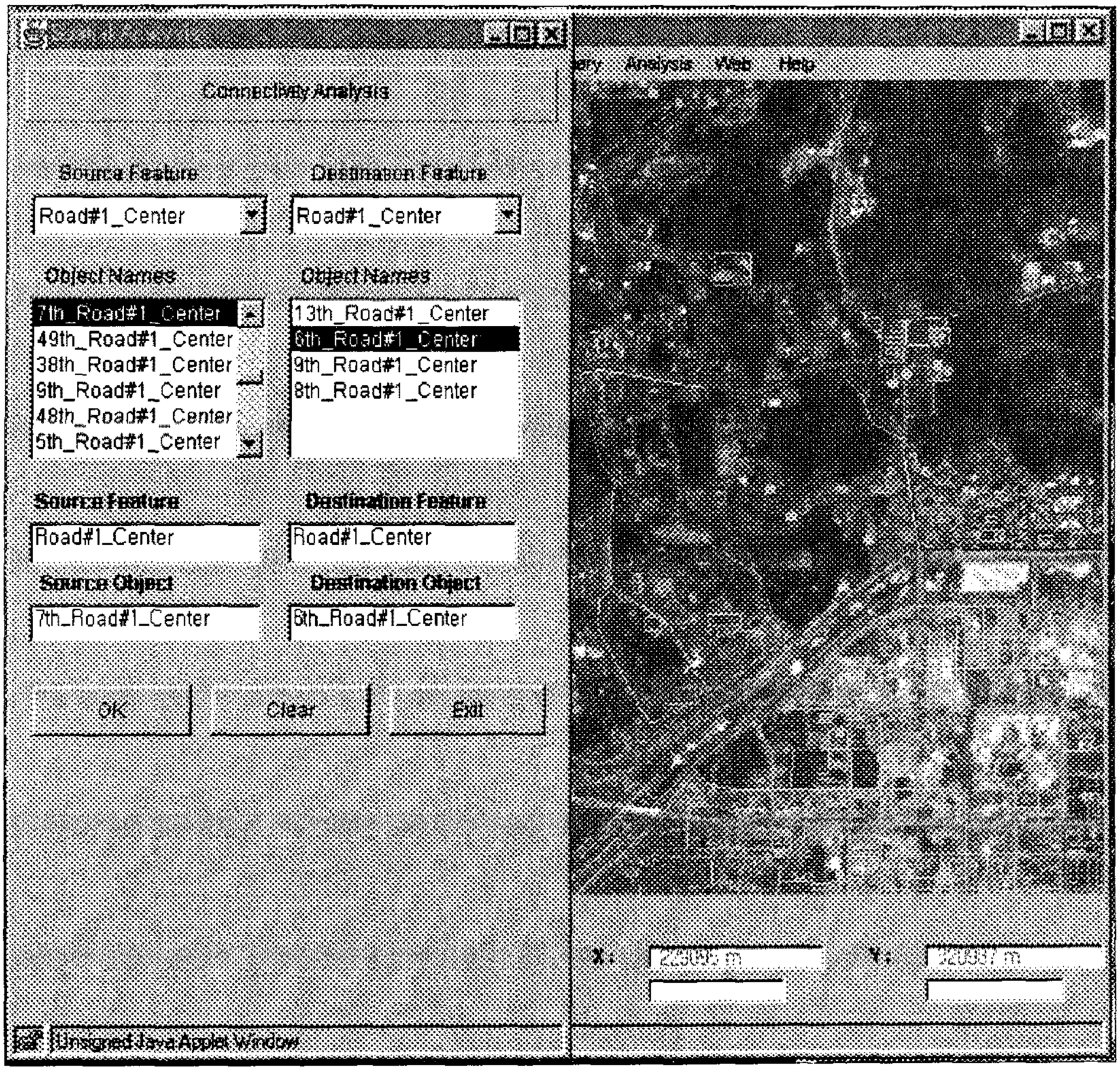


그림 6-23. 체인 객체 사이의 연결성 분석 결과

라. 영향권 분석

영향권 분석은 총 3개의 기능으로 세분화된다. 이 기능에서는 기준(Source) 객체만 있다. 기준 객체가 될 수 있는 객체는 노드 객체, 체인 객체, 다각형 객체가 될 수 있기 때문에 총 3개의 기능이 제공된다. 붉은 색은 기준 객체를 나타내고, 노란 색은 대상 객체들 중에서 공간 분석의 결과가 되는 객체를 나타내고, 분홍색은 공간 분석의 결과 중에서 선택한 객체를 나타낸다. 입력된 반경은 m단위이다. 즉, 기준 객체를 중심으로 입력된 반경만큼의 영향권 범위를 만들어내는 것이다.

1) 노드 영향권 설정

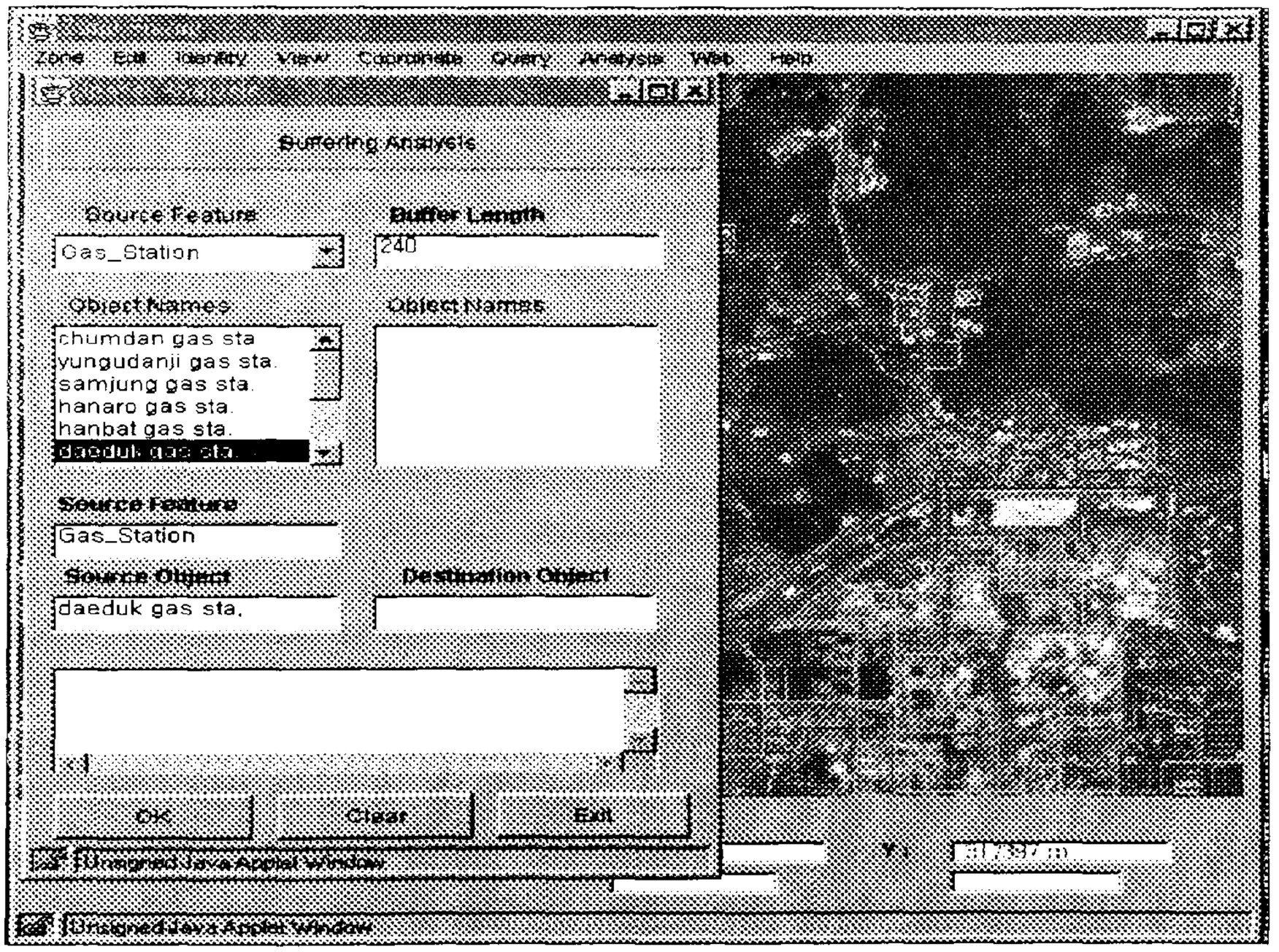


그림 6-24. 노드 객체 주위에 영향권 설정

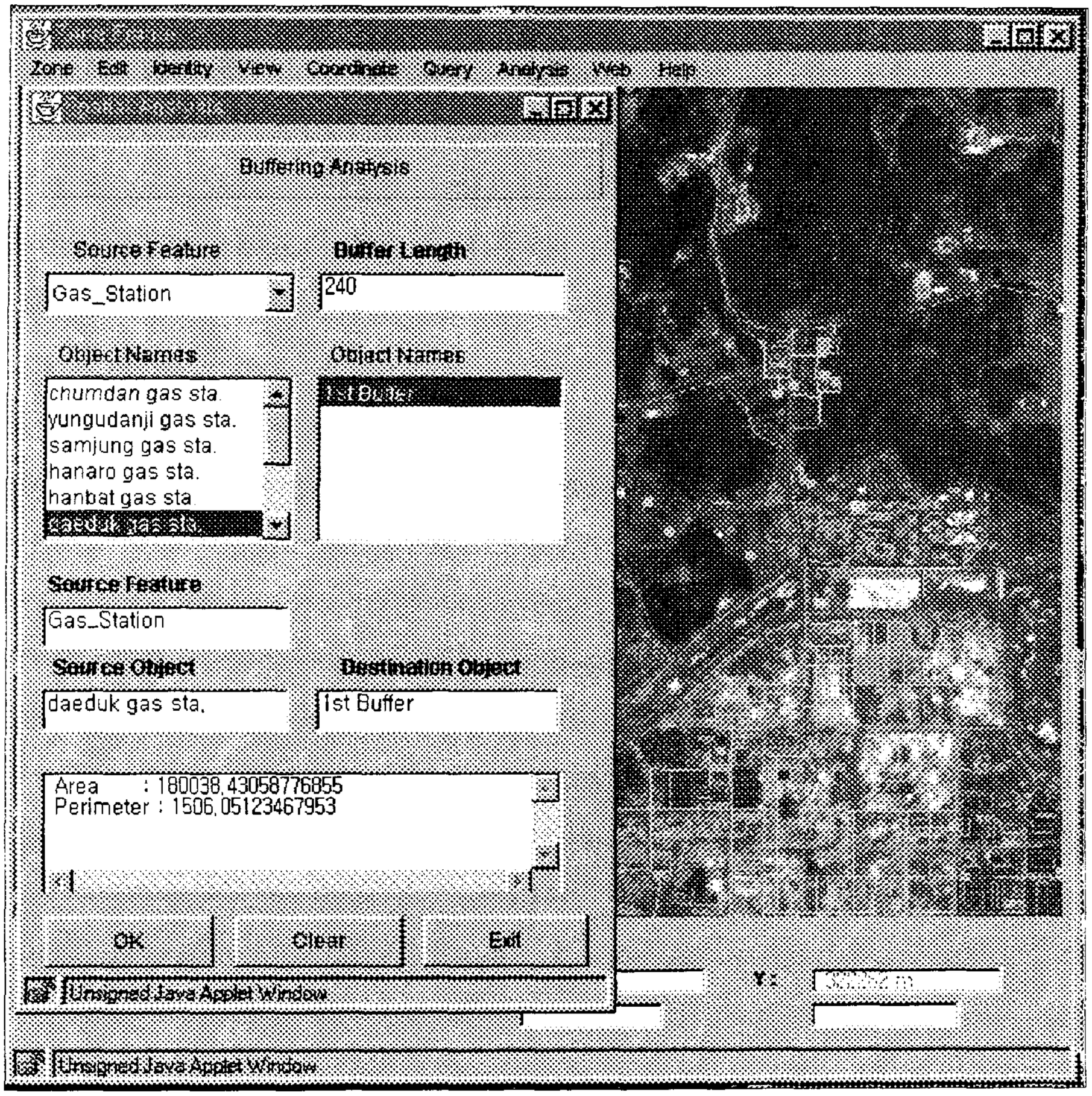


그림 6-25. 노드 객체 주의에 영향권 설정 결과

2) 체인 영향권 설정

기준 객체는 체인 객체인 도로 중심선이다. 영향권의 반경은 180m로 17번 도로 중심선 주위에 180m 범위의 영향권을 형성하는 것이다. 생성된 영향권 범위는 다각형 형식으로 주위 둘레와 면적이 만들어진다.

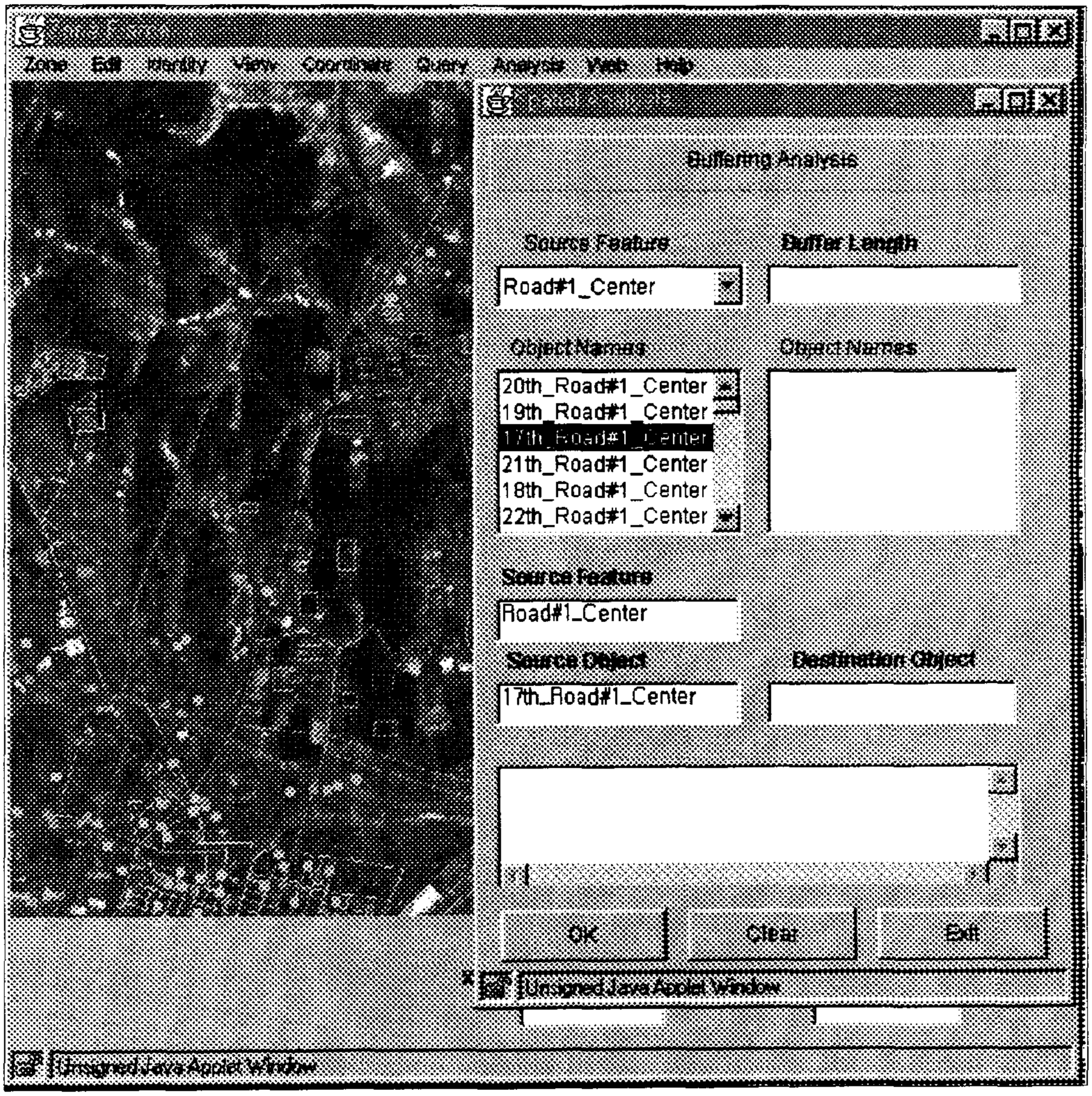


그림 6-26. 체인 객체 주변의 영향권 설정

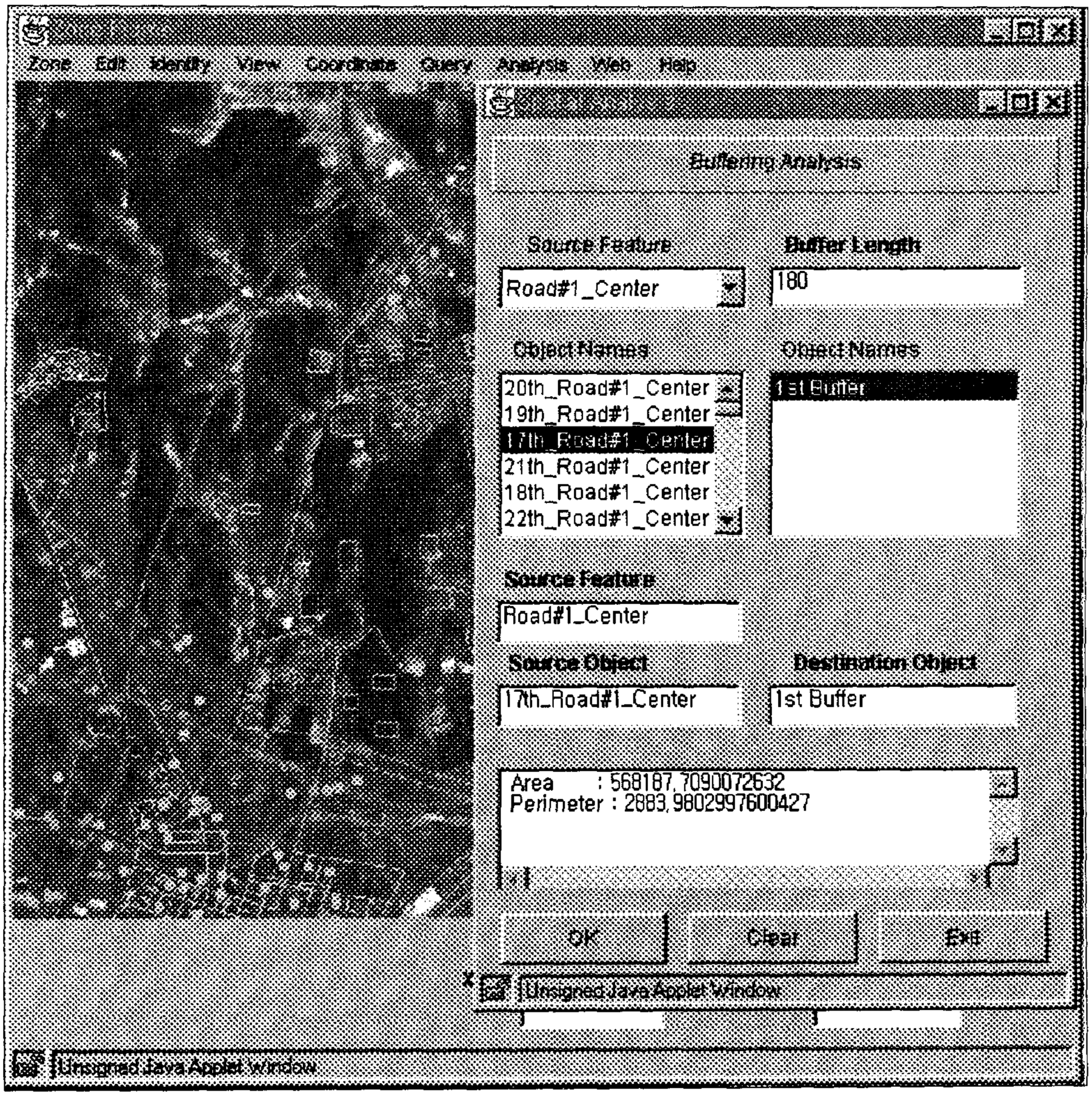


그림 6-27. 체인 객체 주위의 영향권 설정 결과

3) 다각형 영향권 설정

기준 객체는 다각형 객체인 엑스포 과학 공원이다. 영향권의 반경은 80m로 엑스포 과학 공원 내부와 외부에 80m 범위의 영향권을 2개 형성하는 것이다. 생성된 영향권 범위는 다각형 형식으로 주위 둘레와 면적이 만들어진다.

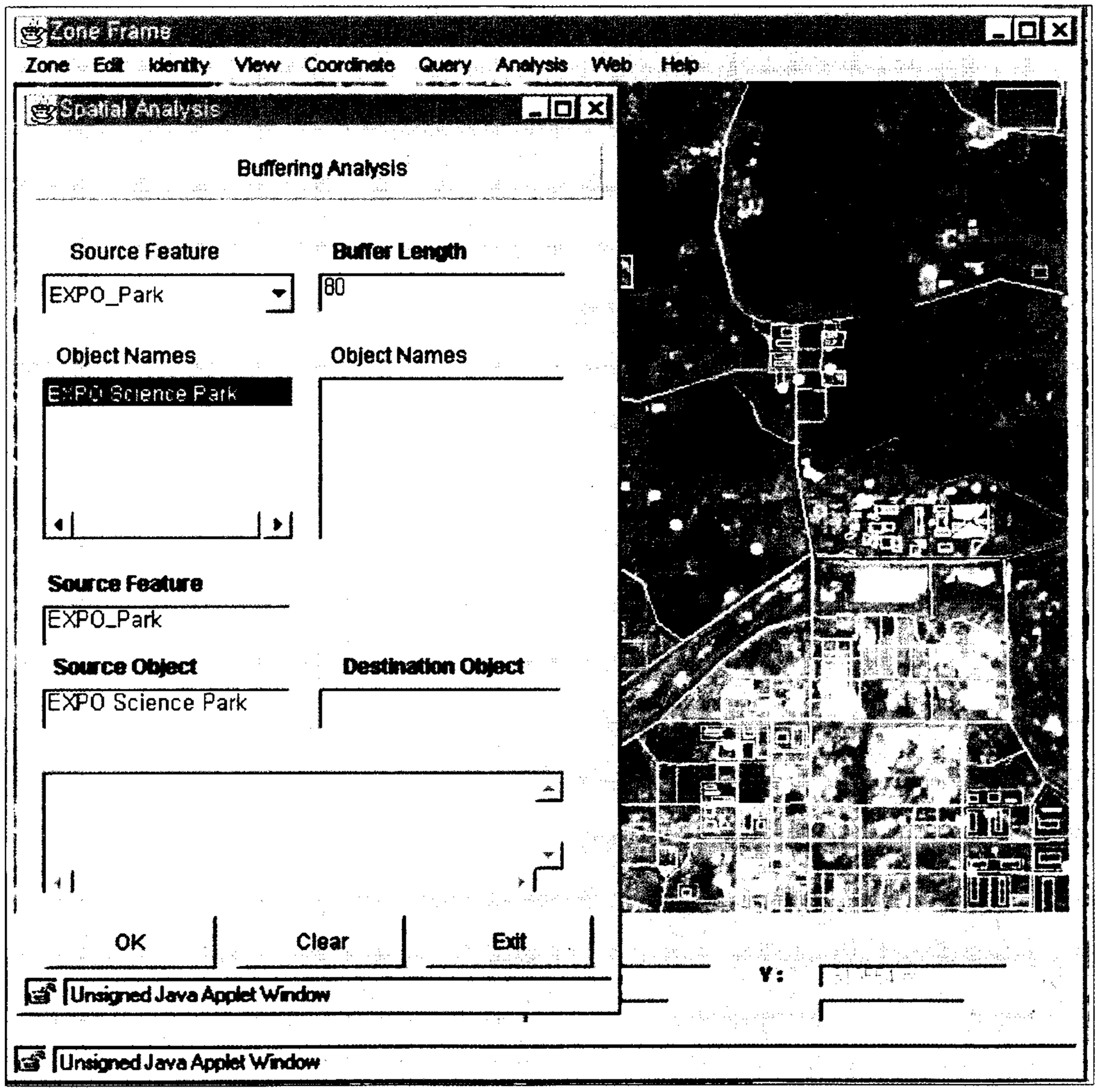


그림 6-28. 다각형 객체 주위의 영향권 설정

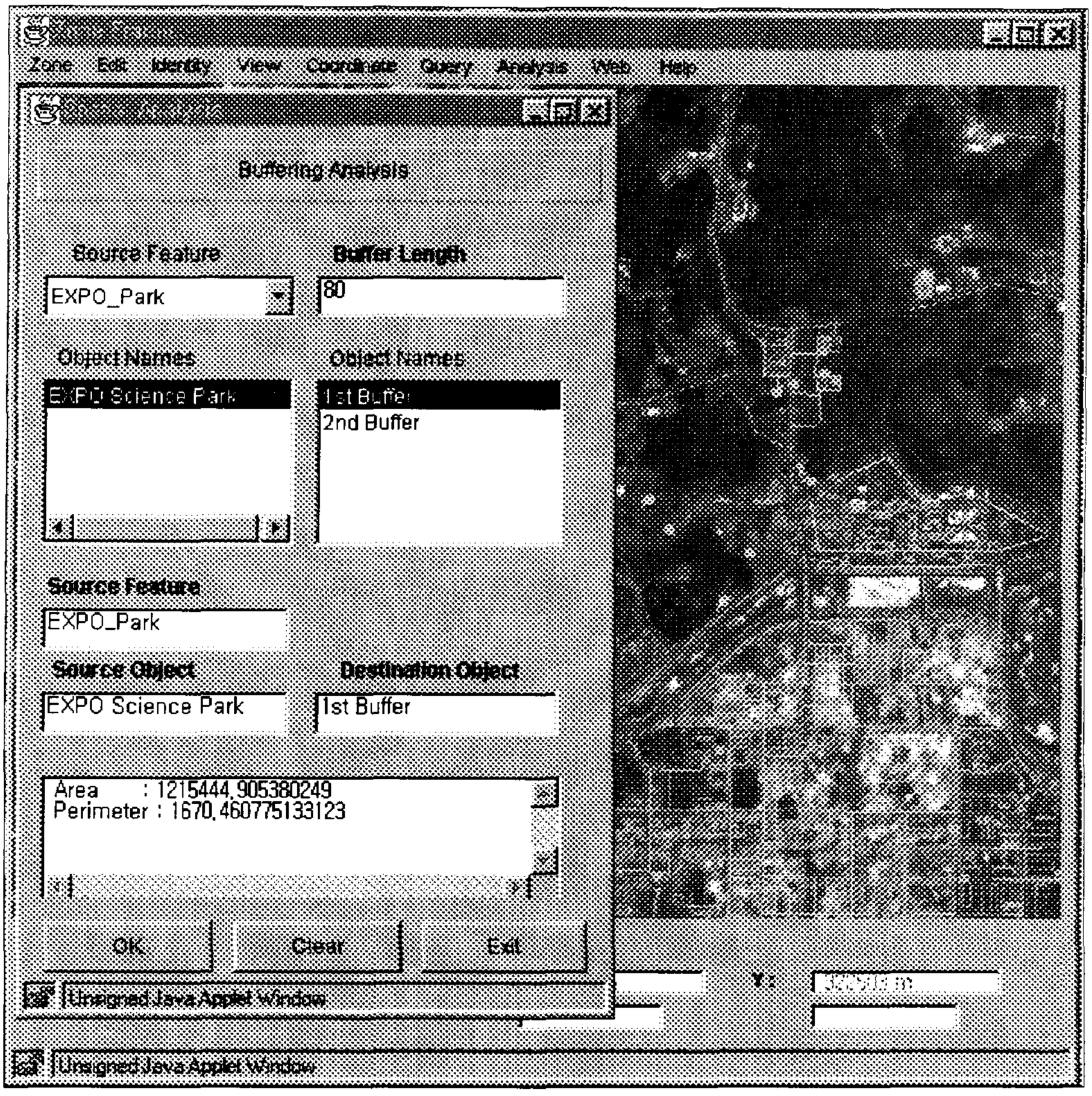


그림 6-29. 다각형 주위의 영향권 설정 결과

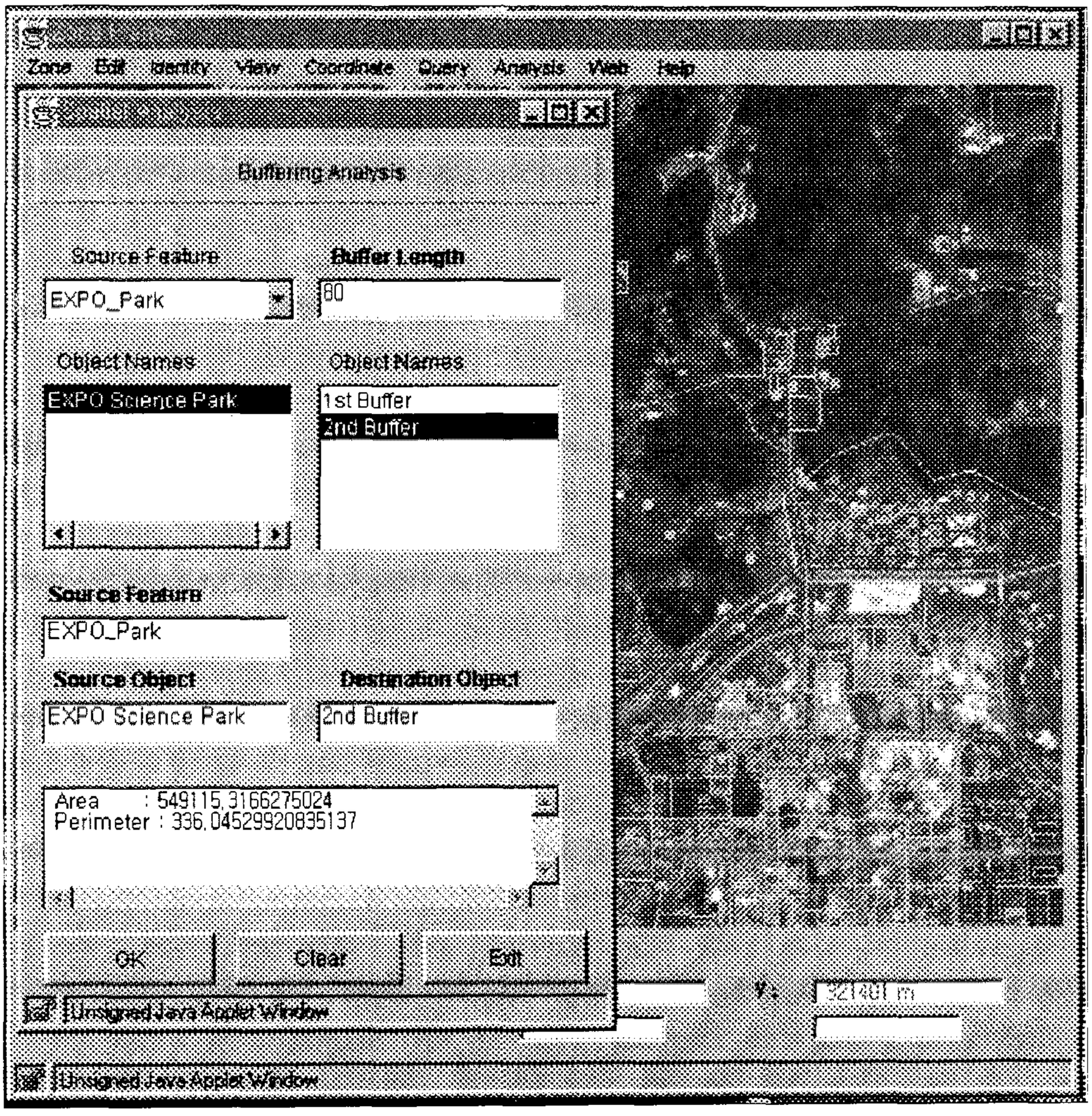


그림 6-30. 다각형 주의의 영향권 설정 결과

마. 네트워크 분석

네트워크 분석에서 제공하는 기능은 시작점과 종점이 주어졌을 때에 이 두점 사이의 최단 경로를 체인 객체의 집합으로 구하는 것이다. 마우스를 이용하여 시작점과 종점을 선택하면 시작점에서 출발하여 종점에 이르는 최단 경로를 검색하고 거리를 계산하여 사용자에게 제공한다. 노란 점은 시작점과 종점이며 노란 선이 최단 경로이다.

1) 최단 경로 탐색 구현 방법

표 3-1과 표 3-2는 각각 최단 경로 테이블과 탐색 테이블로 최단 경로 테이블은 탐색되어진 노드에서 가장 가까운 이전 노드와 탐색 비용(누적된 시간)을 기록한 테이블로 이 테이블이 완성되면 출발 노드로부터 목표 노드까지의 최단 경로 및 거리를 알 수 있게 된다.

표 6-1. 최단 경로 테이블

노 드	이전 노드	탐색 비용 (누적 거리)
a		
b		
c		
d		
e		
f		

o 노드 : 탐색 테이블로부터 선택된 노드

o 이전 노드 : 현 노드 이전에 탐색된 최단 경로의 노드

o 탐색 비용 : 출발 노드로부터 현 노드까지의 누적된 거리

탐색 테이블은 최단 경로 테이블을 완성하기 위한 중간 단계의 테이블로서 현재 노드에서 탐색 가능한 노드들을 모두 기록하고 각각의 이전 노드와 탐색 비용을 기록한다. 이때 이미 최단 경로 테이블에 기록된 노드들은 제외한다. 다음으로 탐색 테이블에서 탐색 비용이 가장 적은 노드를 최단 경로 테이블에 기록한다. 이와 같은 과정을 반복하여 완성된 최단 경로 테이블에서 목표 노드로부터 이전 노드를 거슬러 올라가면서 만들어지는 경로가 본 알고리즘에서 만들어지는 최단 경로가 된다.

표 6-2. 탐색 테이블

탐색 노드	이전 노드	탐색 비용 (누적 거리)

o 노드 : 탐색 테이블로부터 선택된 노드

o 이전 노드 : 현 노드 이전에 탐색된 최단 경로의 노드

o 탐색 비용 : 출발 노드로부터 현 노드까지의 누적된 거리

[그림 6-31]은 최단 경로 추출 알고리즘 설명을 위해 사용된 도로망에 대한 그래프 표현으로 최단 주행 경로 추출을 위한 과정은 다음과 같다. 이 때 출발 보드는 a이고 목표 노드는 f이다.

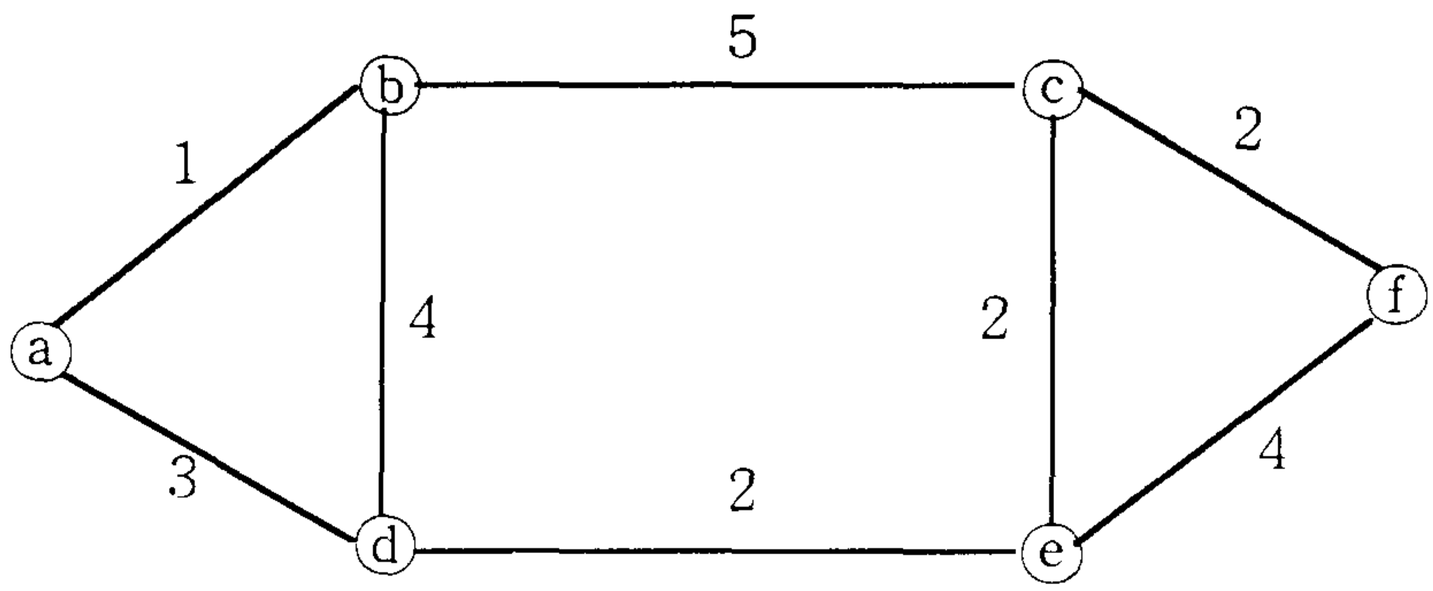


그림 6-31. 도로망의 그래프 표현 예

단계 1. 출발 노드 a를 탐색 테이블에 기록하며 이때 탐색 비용은 0이다. 탐색 테이블에서 기록된 노드는 a노드 하나이므로 아래의 도표와 같이 이 노드를 선택하여 최단 경로 테이블에 기록한다. 이때 a노드가 최단 경로 테이블에 기록되었으므로 a노드는 탐색 테이블로부터 삭제한다.

단계 2. 앞 단계에서 최단 경로 테이블에 기록된 a노드로부터 탐색 가능한 노드는 b와 d노드이며 각각의 이전 노드, 누적된 탐색 비용을 탐색 테이블에 기록한다. 다름으로 탐색 테이블에서 탐색 비용이 가장 적은 b노드를 선택하여 최단 경로 테이블에 기록한다. 따라서 b노드는 다음 단계에서 탐색 테이블로부터 삭제된다.

탐색 노드	이전 노드	탐색 비용 (누적 거리)
a		0

노 드	이전 노드	탐색 비용 (누적 거리)
a		0

탐색 노드	이전 노드	탐색 비용 (누적 거리)
b	a	1
d	a	3

노 드	이전 노드	탐색 비용 (누적 거리)
a		0
b	a	1

단계 3. 앞 단계에서 선택된 b노드에서 탐색 가능한 c및 d노드를 탐색 테이블에 기록한다. 따라서 탐색 비용이 가장 적은 d노드로 선택되어 최단 경로 테이블에 기록된다. d노드로 탐색되는 과정이 두 가지가 존재하므로 그 중 탐색 비용이 적은 경로를 선택하여 최단 경로 테이블에 기록하고 탐색 테이블에서는 d노드로 향하는 모든 경로도 함께 삭제된다.

탐색 노드	이전 노드	탐색 비용 (누적 거리)
d	a	3
d	b	5
c	b	6

노 드	이전 노드	탐색 비용 (누적 거리)
a		0
b	a	1
d	a	3

단계 4. 앞 단계에서 선택된 d노드에서 탐색 가능한 노드는 b와 e노드이지만 b노드는 최단 경로 테이블에 기록되었으므로 탐색 테이블에 기록되지 않는다. 따라서 e노드만 탐색 테이블에 기록된다. 탐색 테이블에 기록된 노드 중에서 최소 비용을 갖는 e노드가 선택되어 최단 경로 테이블에 기록되고 다음 단계에서 탐색 테이블로부터 삭제된다.

탐색 노드	이전 노드	탐색 비용 (누적 거리)
c	b	6
e	d	5

노 드	이전 노드	탐색 비용 (누적 거리)
a		0
b	a	1
d	a	3
e	d	5

단계 5. 앞 단계에서 선택된 e노드에서 탐색 가능한 노드는 c와 f노드이며 각각을 탐색 테이블에 기록한다. 이때 최소 비용을 갖는 c노드가 선택되어 최단 경로 테이블에 기록되고 탐색 테이블에서는 삭제된다. 여기서 목표 노드 f가 탐색되었지만 최단 경로 테이블이 미완성이며, 이것은 모든 탐색 가능한 경로가 탐색되지 않았음을 의미한다. 따라서 최단 경로 테이블이 완성될 때까지 계속하여 탐색 가능한 경로를 탐색하게 된다.

탐색 노드	이전 노드	탐색 비용 (누적 거리)
c	b	6
c	e	7
f	e	9

노 드	이전 노드	탐색 비용 (누적 거리)
a		0
b	a	1
d	a	3
e	d	5
c	b	6

단계 6. 앞 단계에서 선택된 c노드에서 탐색 가능한 노드는 f노드 하나이므로 f노드를 선택하여 탐색 테이블에 기록한다. 여기서 탐색 테이블에는 목표 노드만 남게 되고 그 중 탐색 비용이 가장 적은 노드를 선택하여 최단 경로 테이블에 기록함으로써 모든 과정이 종료되며 동시에 최단 경로 테이블이 완성되게 된다.

탐색 노드	이전 노드	탐색 비용 (누적 거리)
f	c	8
f	e	9

노 드	이전 노드	탐색 비용 (누적 거리)
a		0
b	a	1
d	a	3
e	d	5
c	b	6
f	c	8

단계 6에서 완성된 최단 경로 테이블에서 목표 노드로부터 이전 노드를 거슬러 올라가면서 만들어지는 경로가 본 알고리즘에서 추출한 최단 경로가 된다. 따라서, [그림 6-31]과 같은 도로망에서 추출한 최단 경로는 f -> c -> b -> a가 되며 그에 따른 탐색 비용은 8이 된다.

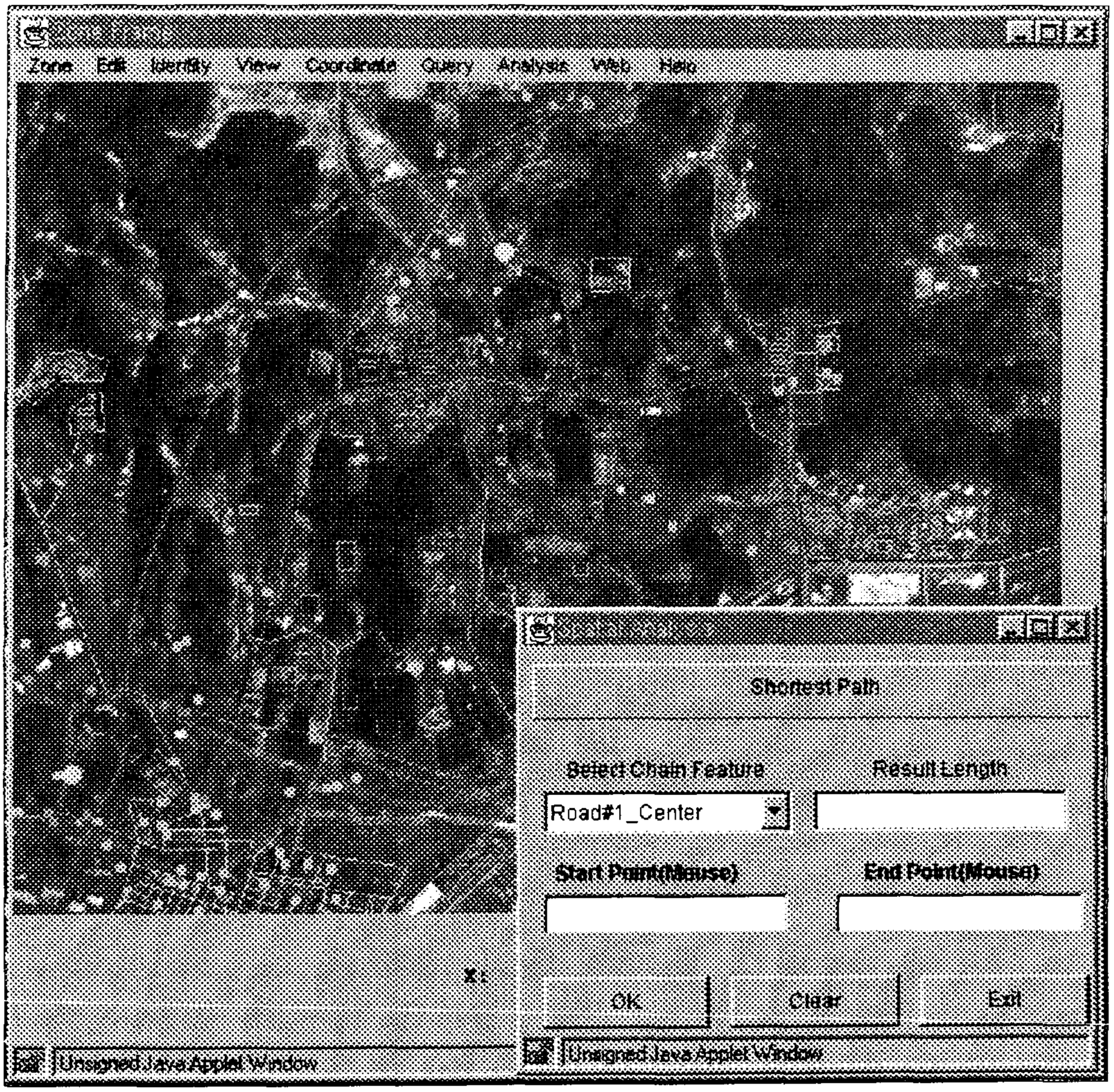


그림 6-32. 최단 거리 검색 화면

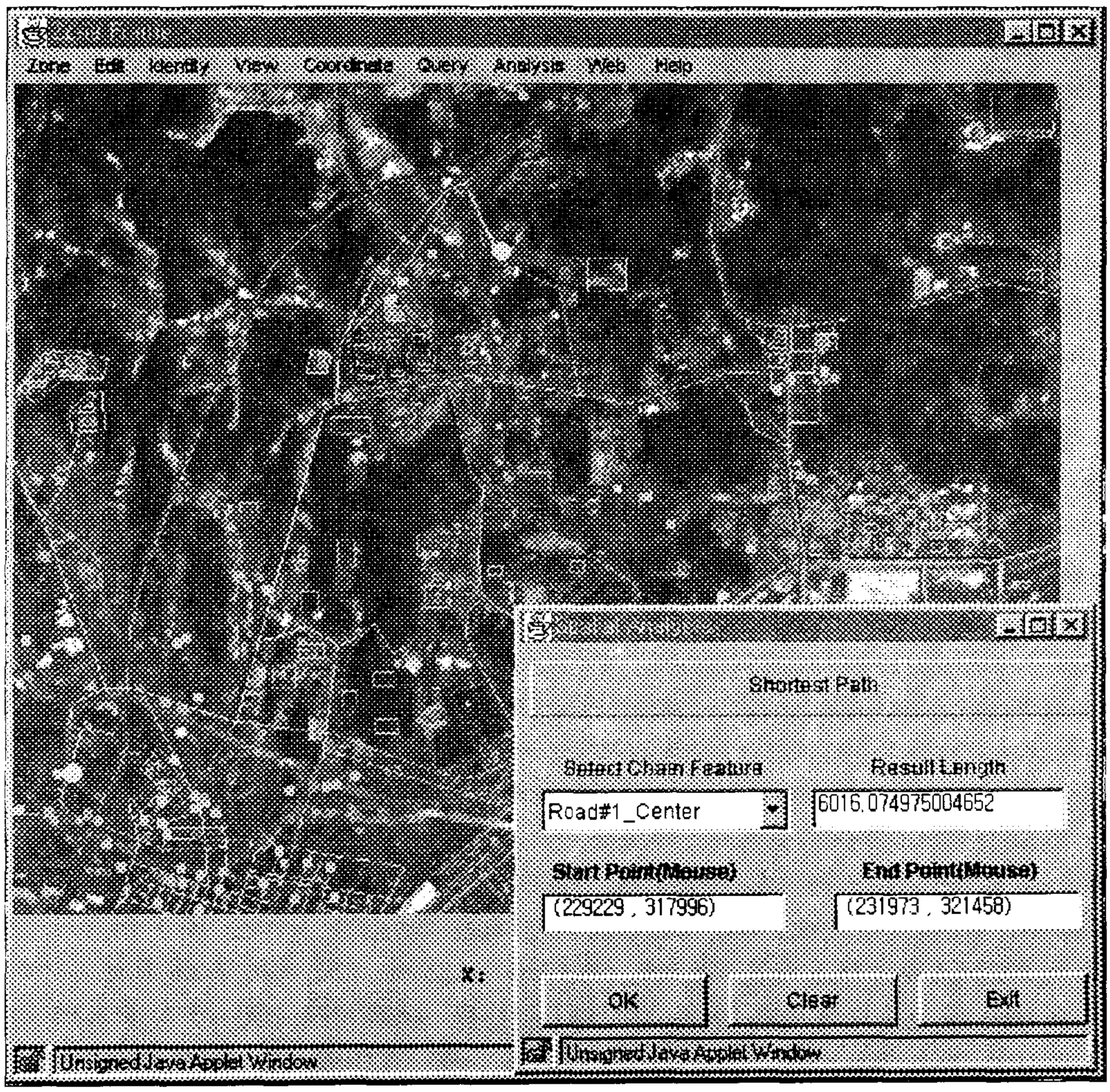


그림 6-33. 최단 거리 검색 결과 화면

바. 인접성 분석

이 기능은 기준(Source) 객체와 대상(Destination) 객체를 설정하여야 한다. 기준 객체와 대상 객체가 될 수 있는 객체는 다각형 객체만 가능하다. 즉, 다각형 객체를 기준으로 이 다각형과 인접해 있는 다각형을 찾아내는 기능이다. 이 분석은 인접한 정보에 기초한 두 다각형 객체 사이의 공간적인 관계를 결정하는데 필요하다. 예를 들면, 공원의 바로 옆에 위치한 토지 구획들 중에서 평균 토지 가격 보다 높은 토지 가격의 토지 구획을 찾는 질의 등에 이용된다.

2. 통계 처리

본 시스템은 인터넷상에서 특정 통계처리 기능을 제공한다. 이러한 통계처리 기능들은 각 피쳐(Feature) 데이터들의 속성정보를 이용하여 처리되며, 통계치가 가능한 모든 속성에 대하여 통계 값을 계산할 수 있다. 피쳐에 대하여 계산되어질 수 있는 통계 값은 현재 합계, 평균, 분산을 계산할 수 있고 이외에 부가적으로 최대값, 최소값, 그리고 전체 개수 등을 계산할 수 있다. 통계처리 시스템의 구성 및 구현 그리고 수행과정에 대해서 부가적으로 살펴보면 다음과 같다.

가. 통계처리 시스템의 구성

본 시스템에서 통계 처리는 임의의 주어진 지역(Zone)내부에 존재하는 노드, 체인, 폴리곤 유형의 피쳐(Feature)중에서 임의의 피쳐에 대해서 수행될 수 있으며, 선택 지역을 설정하기 위해서는 최소 경계 사각형(MBR : Minimum Bounding Rectangle)을 이용하고 있다. 또한 본 통계처리 시스템은 기본적으로 3 Tier 데이터베이스 모델에서 수행되고 있으며, 기본적인 구성은 다음 [그림

6-34]와 같다.

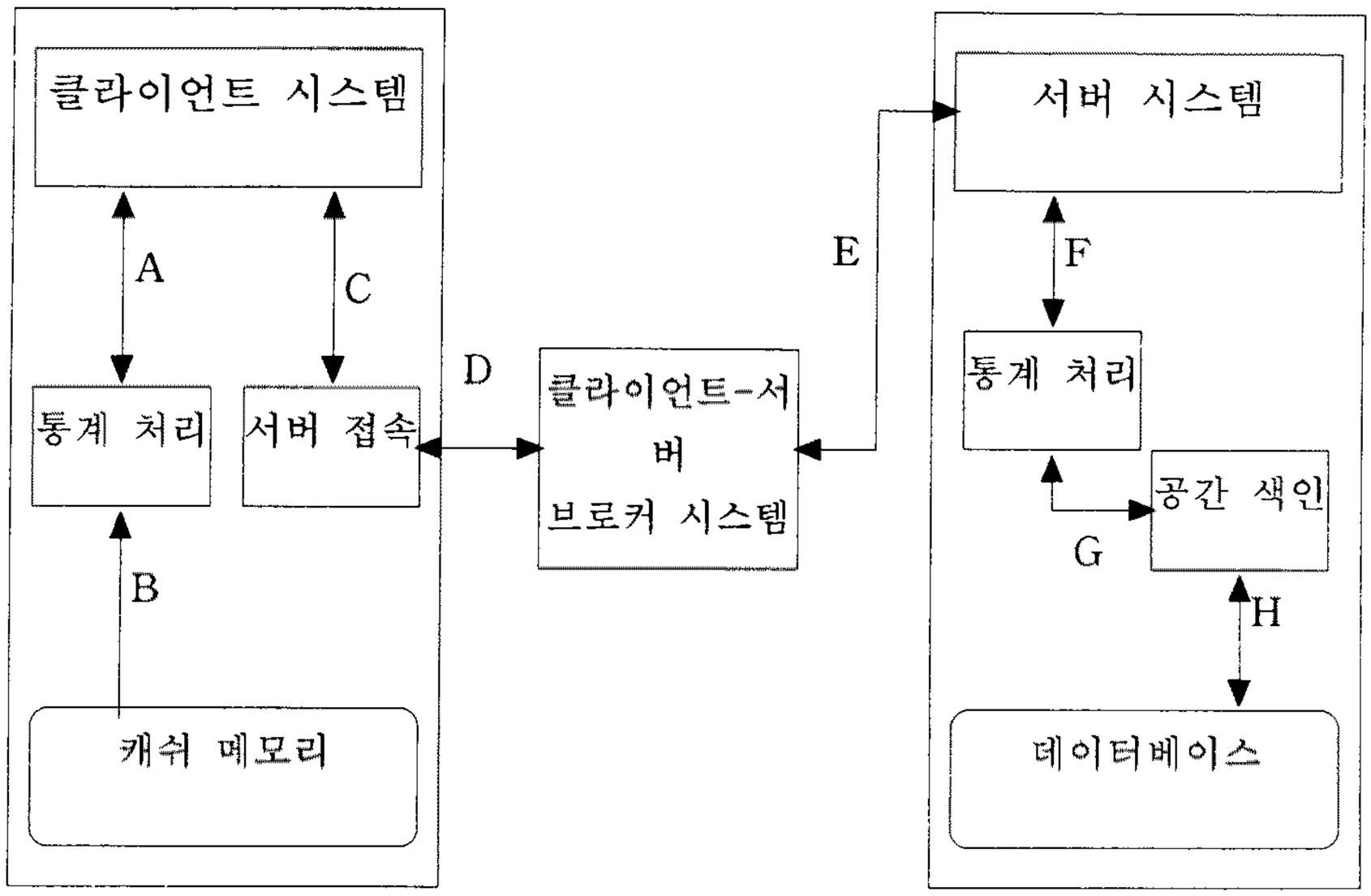


그림 6-34. 통계 처리 시스템의 구성

위의 그림에서 보듯이 클라이언트 시스템이 통계 처리를 수행하고자 하면 먼저 클라이언트 시스템내의 통계 처리 메소드들을 직접 이용함으로써 결과를 얻도록 구성되어 있다. 그러므로 클라이언트 시스템은 A의 통로를 이용하여 통계 처리 메소드들에게 선택 영역(Zone)과 선택된 피쳐(Feature)들을 입력으로 전달하고, 통계 처리 메소드들은 계산된 결과 값을 클라이언트 시스템에 반환한다. 통계 처리 메소드들은 통계치를 계산하기 위해서 필요한 피쳐 데이터를 위해 우선적으로 캐쉬 메모리의 검색을 요청하고, 피쳐 데이터가 존재하면 캐쉬 메모리는 이 데이터를 통계 처리 메소드들에게 전송한다.(B) 만약, 캐쉬 메모리 내에 검색 하고자 하는 데이터가 존재하지 않게 되면 캐쉬 메모리는 null값을 반환하게 되고, 클라이언트 시스템 내에서의 통계 처리는 실패하게 되므로 클라이언트 시스템은 서버에 접속을 요청하게 된다.(C) 서버 접속 인터페이스는 서버에서의

통계처리를 위하여 클라이언트-서버 브로커 시스템에게 선택 영역(Zone), 피쳐 이름, 통계 처리 메소드 이름을 전송하고, 결과로서 서버 시스템에서의 통계 처리 종료 후 결과를 반환 받도록 구성되어 있다.(D) 클라이언트-서버 브로커 시스템은 클라이언트 시스템의 요구사항을 올바른 서버 시스템에 전송하고(E), 수행결과가 반환되기를 대기하게 된다. 브로커를 통하여 클라이언트의 요구를 받은 서버 시스템은 서버 시스템내의 통계 처리 메소드를 수행시키고, 결과를 반환 받는다.(F) 서버 시스템 내에서의 통계 처리에 대하여 좀더 자세히 살펴보면, 먼저 통계 처리 메소드들은 클라이언트에서 전송된 최소 경계 사각형 형태의 선택영역과 피쳐 이름을 이용하여 먼저 공간 검색을 수행하며(G), 통계 처리를 수행한다. 현재 시스템은 서버에서의 통계 처리를 위한 메소드를 새로이 구성하지 않았으며, 데이터베이스에 접근하기 위한 SQL문을 이용하고 있으므로 실제 공간 검색과 통계 처리는 동시에 수행되고 있다.(H)

나. 통계처리 기능의 구현

통계 처리 기능으로 제공되는 기능은 현재 평균, 분산이 있으며 부가적으로 합계, 최대값, 최소값, 전체개수를 계산하는 기능을 제공하고 있다. 그리고 통계 처리 기능은 클라이언트 시스템과 서버 시스템에 존재하고 있는데 클라이언트에서는 메소드로 구현되어 있고 서버에서는 SQL문을 이용하여 구현되어 있다. 이들 기능들의 구현에 대한 자세한 사항은 다음과 같다.

1) 클라이언트 시스템

클라이언트 시스템에서의 통계 처리는 캐쉬 메모리의 데이터를 이용하여 일부가 메소드로 구성되어 있으며 빠른 시간 내에 통계 처리값을 계산해준다. 다음 [그림 6-35]는 클라이언트 시스템에서의 통계 처리 의사 코드를 보여준다.

```
public double GetStatistics(String featureName, String field, int option) {
    // 캐쉬 메모리로부터 피쳐 데이터를 읽어 온다.
    FeatureList FList = zf.zone.GetCache(featureName);

    // 통계 처리 : 평균, 합계 계산
    if (option == 평균 || option == 합계) {
        sum = Calculate_Summation_and_Aveage (FList);
        if (option == 평균)
            // 평균값 반환
            return ( sum/FList.size() );
        else if (option == 합계)
            // 합계 반환
            return ( sum );
    }

    // 통계 처리 : 최소값 계산
    else if (option == 최소값) {
        // 최소값 반환
        min = Cacluate_Minimum(FList);
        return ( min );
    }

    // 통계 처리 : 최대값 계산
    else if (option == 최대값) {
        // 최대값 반환
        max = Cacluate_Maximum(FList);
        return ( max );
    }

    // 통계 처리 : 전체 개수
    else if (option == 전체개수) {
        // 전체 개수 반환
        return ( FList.size() );
    }
```

그림 6-35. 클라이언트 시스템에서의 통계 처리 의사 코드

위의 그림에서 보듯이 클라이언트 시스템에서 통계 처리 메소드는 먼저 캐쉬 메모리로부터 피쳐 데이터를 읽어들인 다음, 이 피쳐 데이터를 통계 처리 옵션 코드에 따라서 처리하도록 구성되어 있다. 그리고 만약 통계 처리 메소드가 '-1'을 반환하면 클라이언트 시스템은 서버 접속 인터페이스를 이용하여 서버 시스템으로부터 통계 처리 결과를 반환 받도록 구성되어 있는데, 서버를 거치는 경우는 데이터베이스 접속 및 네트워크 부하로 인하여 속도가 다소 떨어지는 면이 있다.

2) 서버 시스템

서버 시스템에서의 통계 처리는 JDBC를 이용하여 오라클 데이터베이스의 SQL문을 사용하여 구현하고 있는데 다음과 같은 세 가지의 입력 인자가 필요하다.

- 속성(Column) : 데이터베이스 테이블내의 한 열(Column)로서 테이블 스키마 중의 어느 속성에 대하여 통계치를 구할 것인가에 대한 정보
- 테이블(Table) : 통계 처리를 수행할 때 데이터베이스내의 어떤 테이블들을 이용할 것인가에 대한 정보
- 조건(Condition) : 어떤 조건하에서 통계 처리를 수행할 것인가에 대한 정보를 유지한다. 복합 조건이 사용될 수 있다.

속성, 테이블, 그리고 조건의 세 인자를 입력으로 하였을 때 통계 값을 계산하는 SQL문의 형식은 다음과 같다.

- 평균 계산 : SELECT AVG (속성) FROM 테이블 WHERE 조건
- 분산 계산 : SELECT VARIANCE (속성) FROM 테이블 WHERE 조건
- 합계 계산 : SELECT SUM (속성) FROM 테이블 WHERE 조건
- 최대값 계산 : SELECT MAX (속성) FROM 테이블 WHERE 조건
- 최소값 계산 : SELECT MIN (속성) FROM 테이블 WHERE 조건
- 전체 개수 계산 : SELECT COUNT (속성) FROM 테이블 WHERE 조건

본 시스템에서는 위의 경우처럼 단순한 경우의 현재 단순한 통계 처리값만을 처리할 수 있지만, 클라이언트 시스템내의 통계 처리 메소드를 조금 수정하면 쉽게 확장이 가능하도록 구성되어 있다.

다. 통계처리 기능의 수행과정

클라이언트 및 서버내에서 수행되는 통계 처리 시스템의 구성도를 중심으로 보았을 때 동작과정을 그림으로 나타내면 다음 [그림 6-36]과 같다.

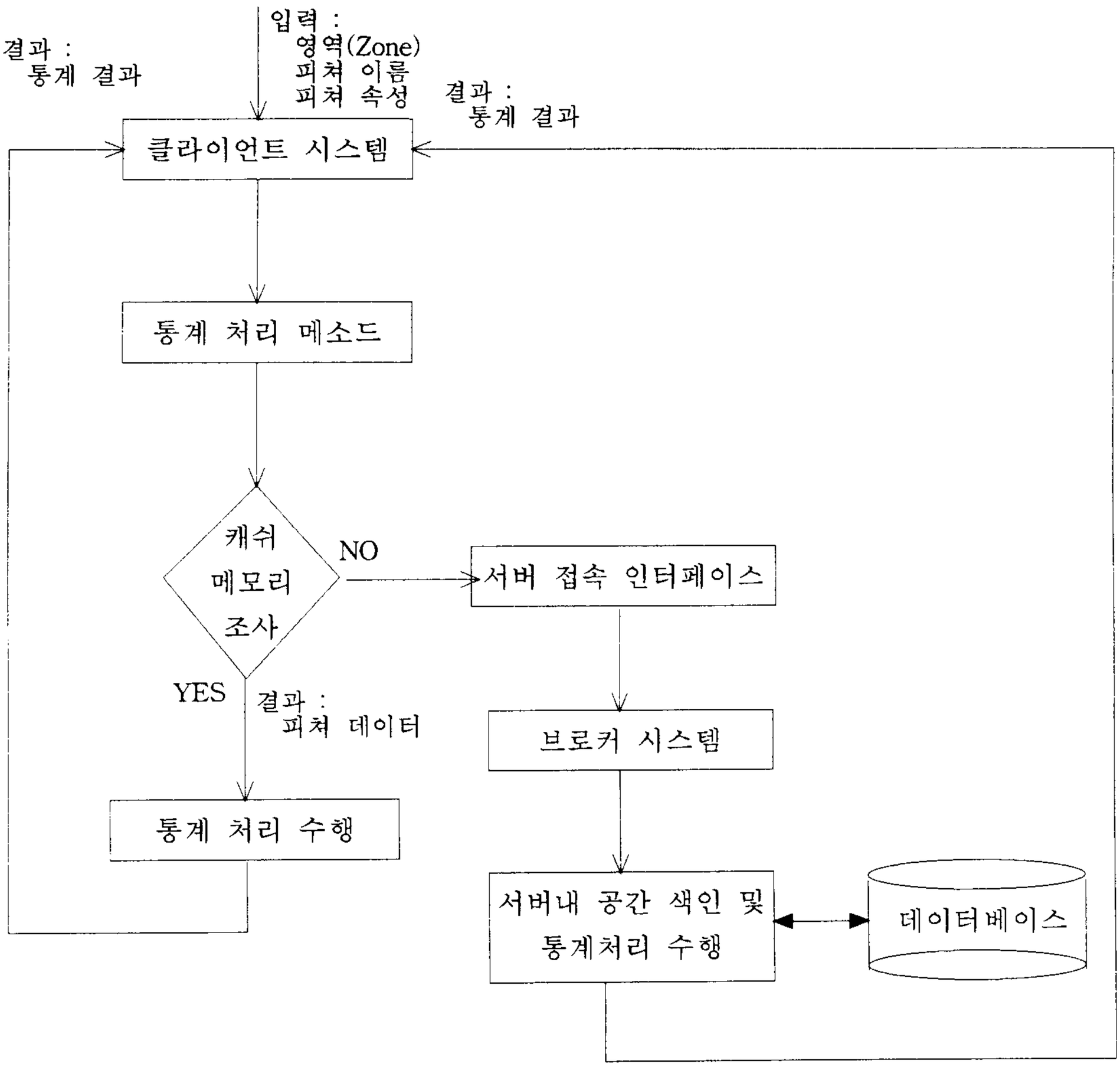


그림 6-36. 통계 처리 시스템의 동작 과정

위의 그림과 같은 통계 처리 시스템의 동작과정을 구체적으로 살펴보면 다음과 같다.

- 클라이언트 시스템에서 입력인자로 '영역', '피쳐 이름' 그리고 '피쳐 속성'을 가지고 원하는 통계 처리 메소드를 부른다.
- 통계 처리 메소드는 먼저 클라이언트 시스템내의 캐쉬 메모리를 조사하여 주어진 '피쳐 이름'에 해당하는 피쳐 데이터가 존재하는지 조사한다.
- 캐쉬 메모리에 피쳐 데이터가 존재하면 그 데이터를 이용하여 통계 처리 메소드를 완료시키고 결과를 반환하고 수행을 종료한다.
- 캐쉬 메모리에 원하는 피쳐 데이터가 존재하지 않는 경우는 브로커 시스템을 통하여 서버에 접속한다.
- 서버에 접속 후, 통계 처리를 위한 SQL문을 생성하고 이를 이용하여 데이터베이스에 접속한다.
- 데이터베이스에 접속하여 공간 색인 및 통계 처리를 수행하고 결과를 다시 브로커를 통하여 클라이언트에 반환하고 수행을 종료한다.

라. 통계 처리 기능의 실행

통계 처리 기능의 실행에서는 실제 개발된 클라이언트 시스템에서의 수행된 결과를 합계, 평균, 분산, 최대값, 최소값, 전체 개수의 순으로 보여준다. 그리고 실제로 여기서 보여지는 이러한 예제들은 쉽게 응용 시스템에 사용될 수 있는 것들임을 알 수 있다.

1) 합계 (SUM) 실행 예

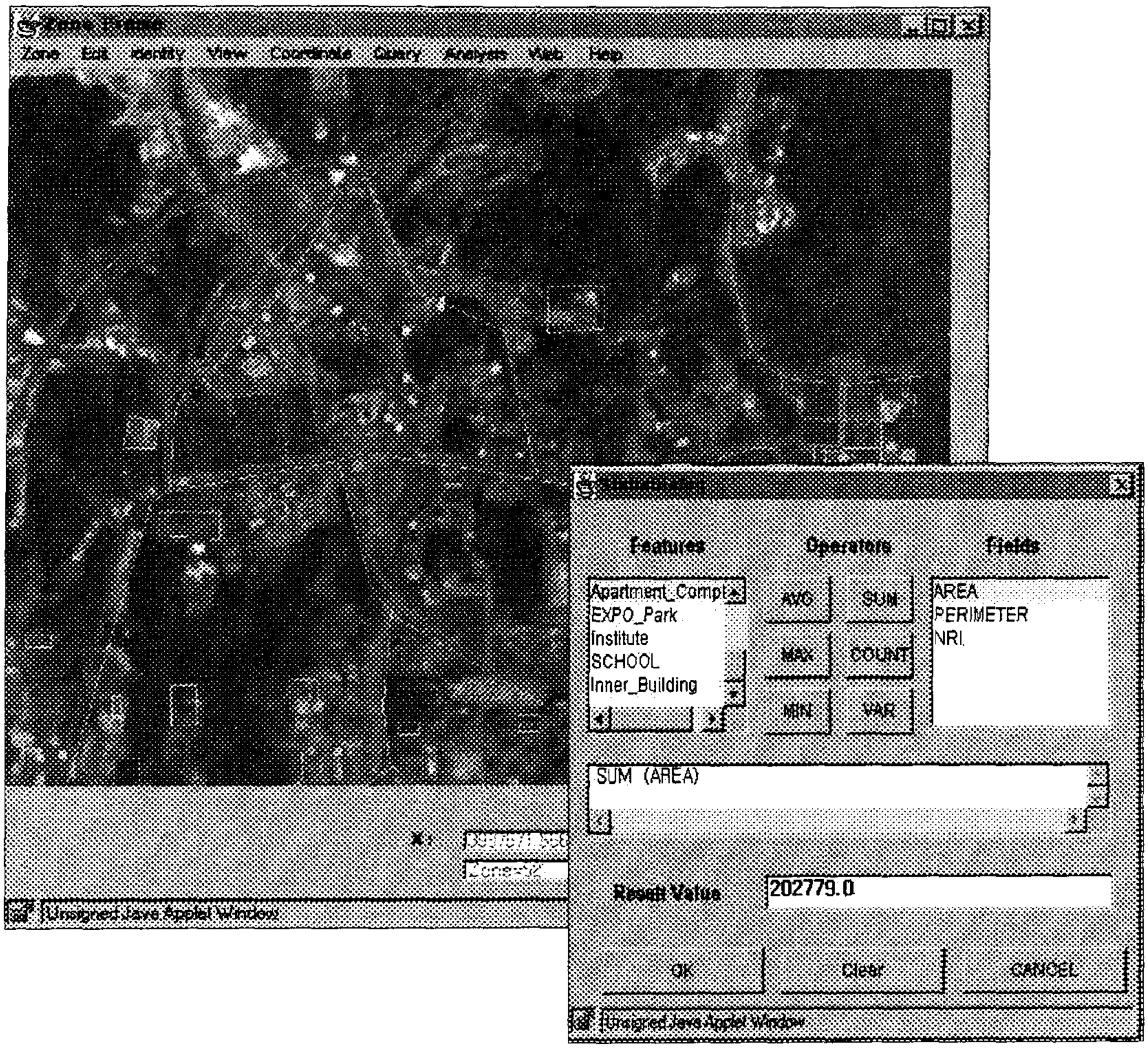


그림 6-37. 통계 처리 시스템에서 합계의 실행 예

위의 [그림 6-37]의 메인 화면을 보면 빨간색으로 채워진 다섯 개의 폴리곤은 메뉴화면의 피쳐(Feature) 리스트에서 선택된 아파트 단지(Apartment_Complex) 객체를 말한다. 그리고 선택된 아파트 단지 피쳐의 속성(Fields)으로는 면적(Area)을 선택하였으며, 실제 수행되는 메소드는 합계를 계산

하는 것이다. 이를 요약하여 보면 다음과 같다.

- 선택된 피쳐 : 아파트 단지(Apartment_Complex)
- 선택된 피쳐 속성 : 면적 (Area)
- 수행된 메소드 : 합계 (Sum)
- 결과 (아파트 단지 면적 합계) : 202779 m^2

2) 평균 (AVERAGE) 실행 예

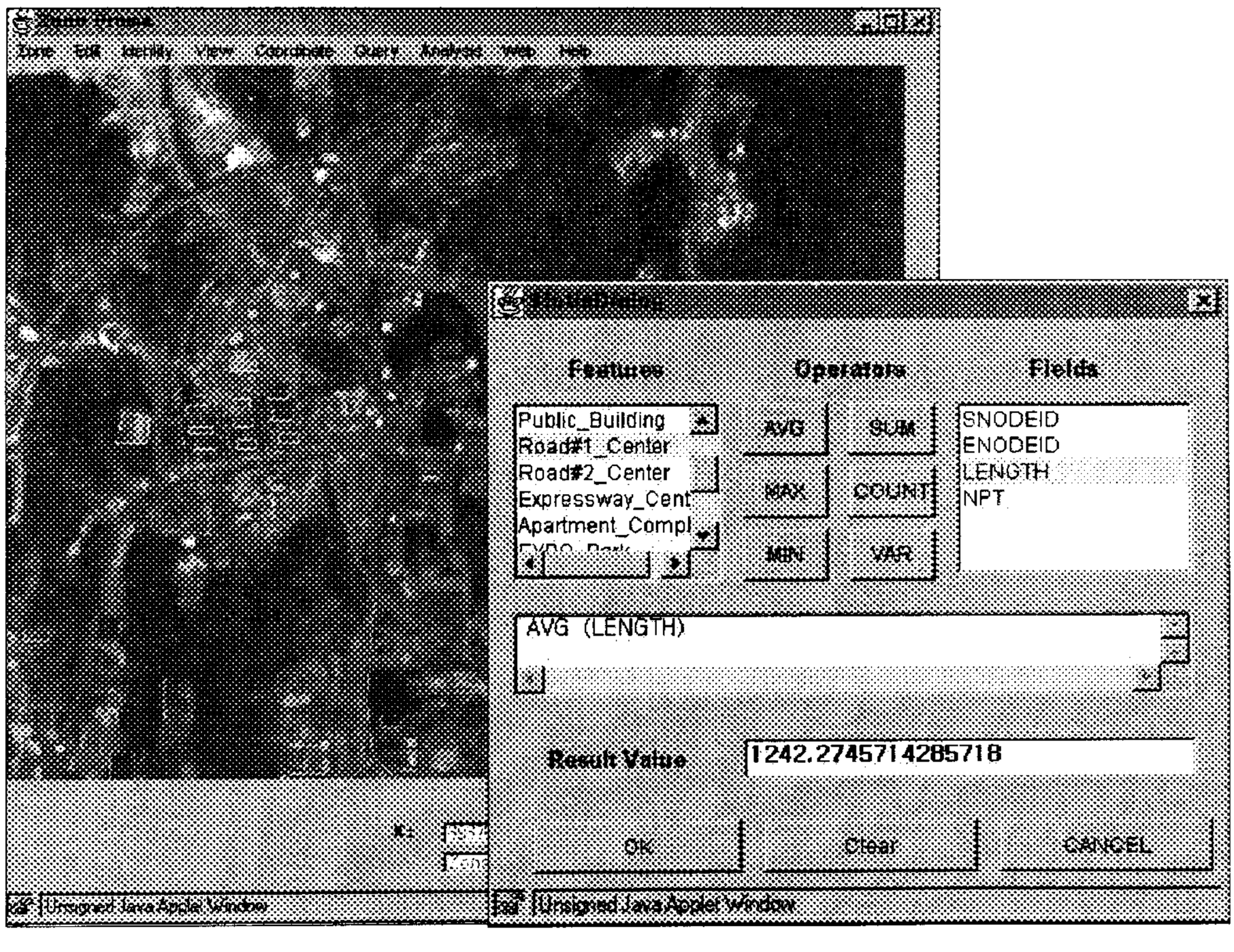


그림 6-38. 통계 처리 시스템에서 평균의 실행 예

위의 메인 화면에서 빨간색 선으로 나타난 부분은 메뉴화면에서 선택되어진 대도로 중심선(Road#1_Center) 객체들을 나타낸다. 그리고 선택된 대 도로 중심선의 길이(Length)를 속성으로 하여 평균값을 계산하였는데 결과는 다음과 같다.

- 선택된 피쳐 : 대 도로 중심선(Road#1_Center)
- 선택된 피쳐 속성 : 길이 (Length)
- 수행된 메소드 : 평균 (Average)
- 결과(대 도로 평균 길이) : 1242.2745714285718 m

3) 분산 (VARIANCE) 실행 예

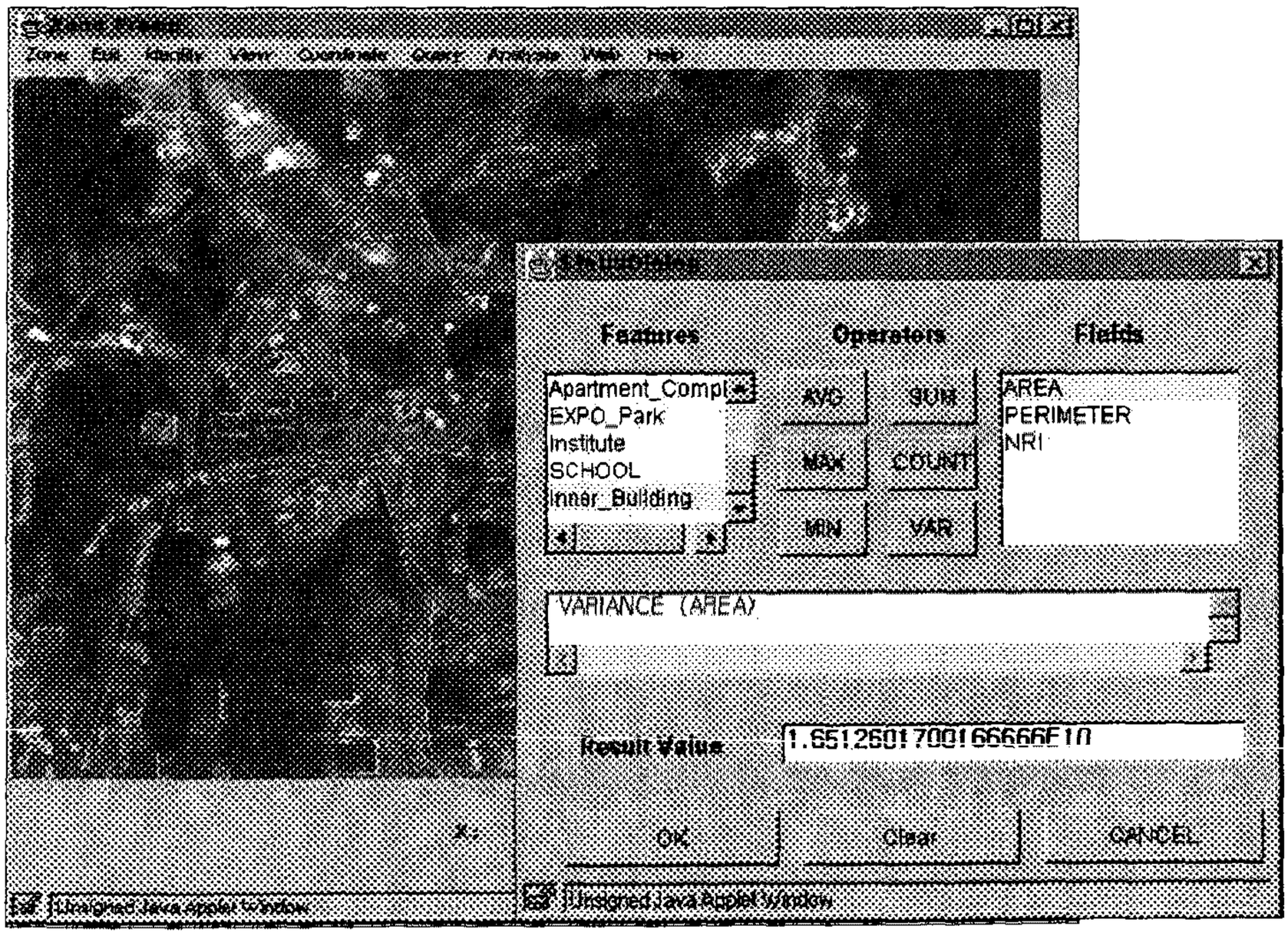


그림 6-39. 통계 처리 시스템에서 분산의 실행 예

위의 메인 화면에서 선택된 폴리곤들은 메뉴의 내부 건물(Inner_Building)피쳐에 대한 객체들이고, 속성으로는 내부 건물의 면적(Area)을 선택하였다. 통계 처리를 위해 수행된 메소드는 분산이며, 결과 및 자세한 사항은 다음과 같다.

- 선택된 피쳐 : 내부 건물(Inner_Building)
- 선택된 피쳐 속성 : 면적 (Area)
- 수행된 메소드 : 분산 (Variance)
- 결과(내부 건물 면적의 분산) : 1.65126 m^2

4) 최대값 (MAXIMUM VALUE) 실행 예

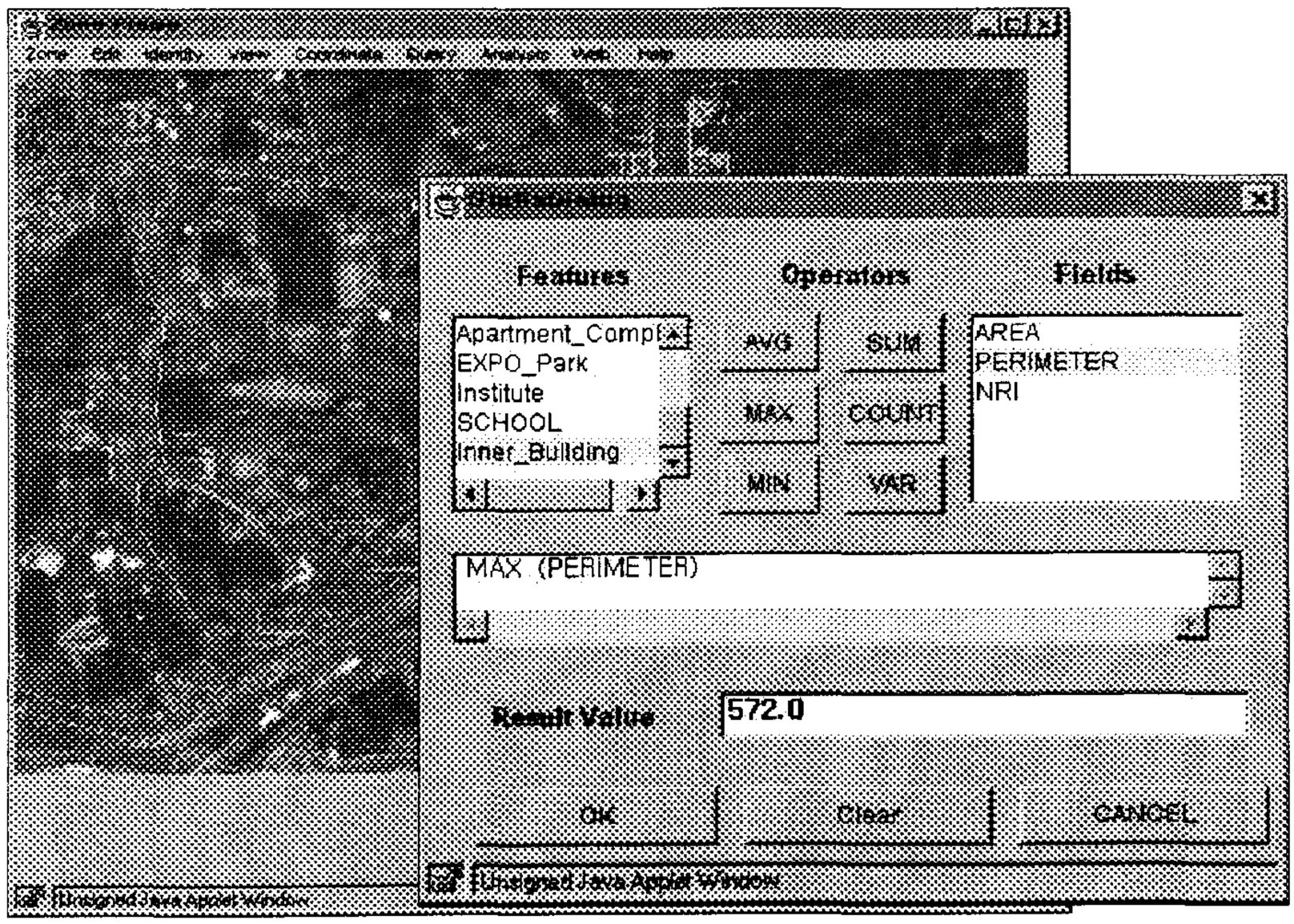


그림 6-40. 통계 처리 시스템에서 최대값의 실행 예

위의 메인 화면에서 선택된 객체들은 내부 건물(Inner_Building)의 피쳐이고, 속성으로는 둘레(Perimeter)를 이용하고 있다. 그리고 최대값을 계산하였는데 결과는 다음과 같다.

- 선택된 피쳐 : 내부 건물(Inner_Building)
- 선택된 피쳐 속성 : 둘레 (Perimeter)
- 수행된 메소드 : 최대값 (Maximum Value)
- 결과(내부 건물중 최대 둘레 값) : 572.0 m

5) 최소값 (MINIMUM VALUE) 실행 예

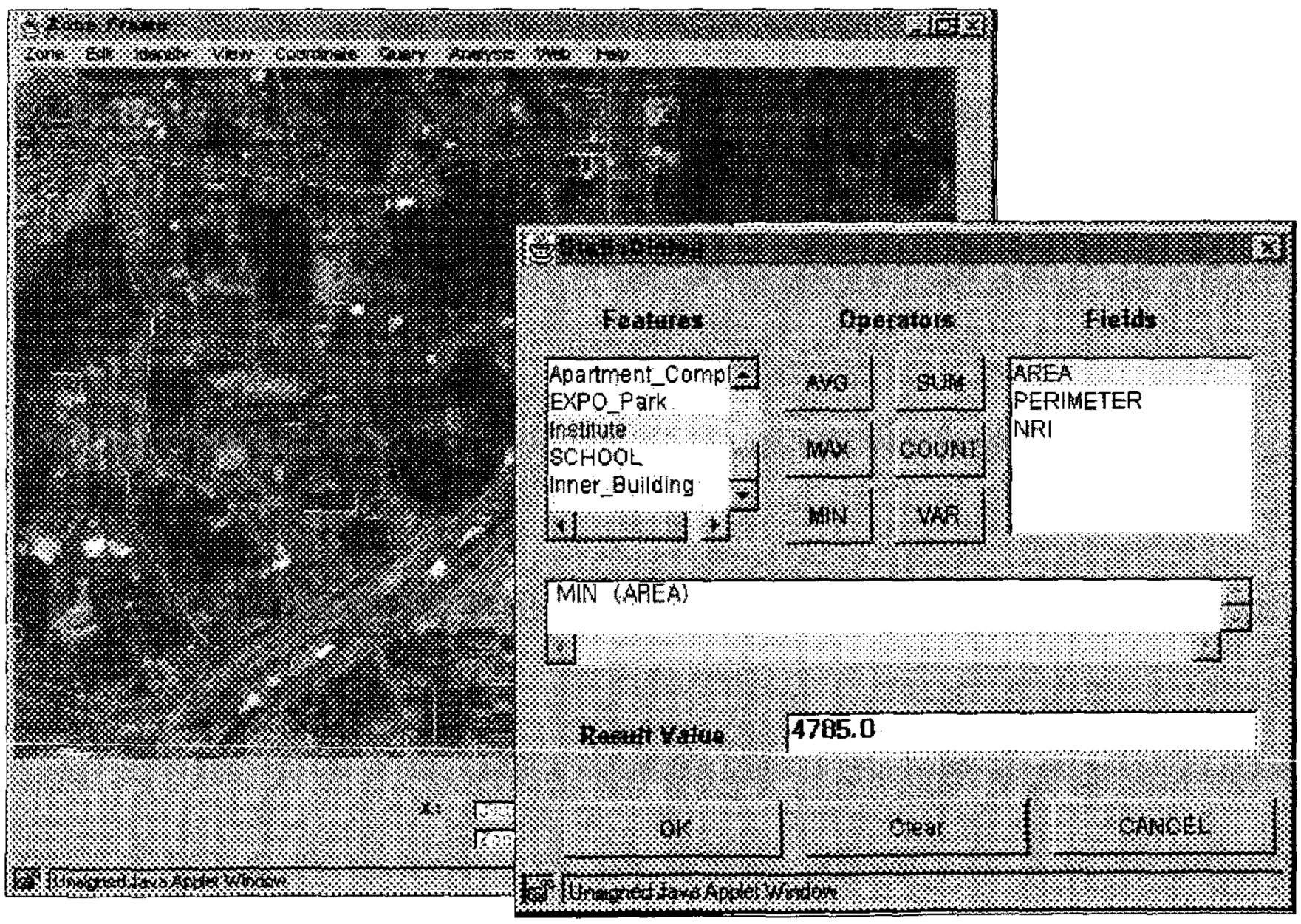


그림 6-41. 통계 처리 시스템에서 최소값의 실행 예

위의 그림은 최소값을 계산한 결과로서 선택된 객체들은 연구소 (Institute) 피쳐들이고, 속성 정보로는 면적(Area)을 이용하였다.

- 선택된 피쳐 : 연구소 건물(Institute)
- 선택된 피쳐 속성 : 면적 (Area)
- 수행된 메소드 : 최소값 (Minimum Value)
- 결과(연구소 건물중 최소 면적) : 4785.0 m^2

5) 전체 개수 (TOTAL COUNT) 실행 예

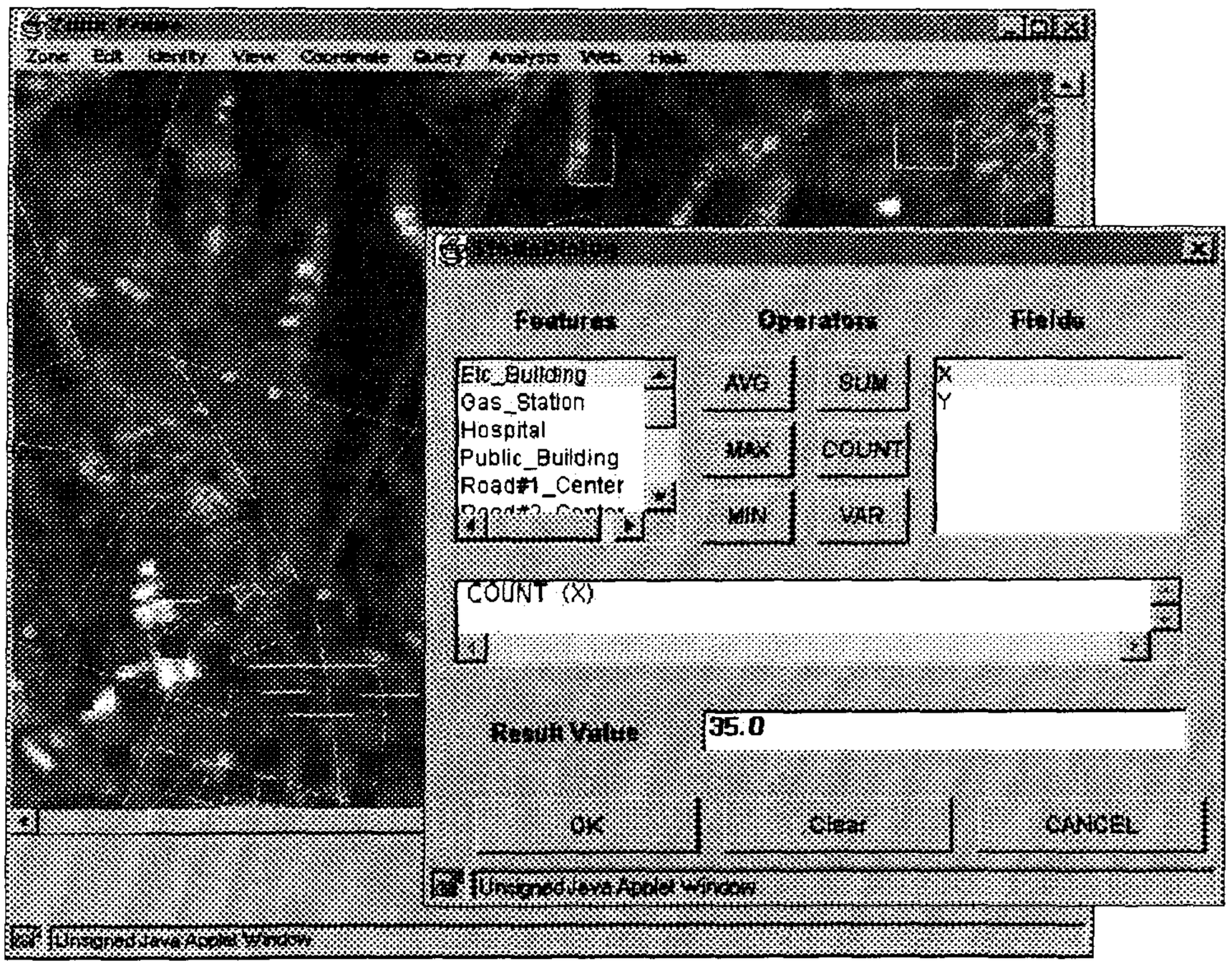


그림 6-42. 통계 처리 시스템에서 전체 개수의 실행 예

위의 그림에서 선택된 객체들은 노드 유형의 피쳐로서 기타 건물(Etc_Building) 피쳐이고, 속성 정보로는 단지 x 좌표를 선택하였다. 그리고 실행된 메소드는 전체 개수를 파악하는 것으로서 결과는 다음과 같다.

- 선택된 피쳐 : 기타 건물(Etc_Building)
- 선택된 피쳐 속성 : x 좌표
- 수행된 메소드 : 전체 개수 (Total Count)
- 결과(기타 건물의 수) : 35 개

제 3 절 입력 자료 모델

1. 입력 자료 및 구분

o 행정 구역 > 2 Layers
  - 구 경계
  - 동 경계

o 도로 : 도로와 중앙선을 별도의 Layer로 표시(도로#3 제외) > 4 Layers
  - 고속 도로
  - 도로#1
  - 도로#2
  - 도로#3

o 건물 > 5 Layers
  - Boundary : 아파트 단지, 학교 등
  - Boundary 내의 건물 표시
  - 나머지 독립 건물: 이름이 있는 건물
  - 관공서(구청, 소방서, 동사무소)

- 기타(연구소, 주유소, 병원)

o 하천 > 1 Layer

o Source Map

1:5000 유성/대전 지형도(6매)

제작 연도:1991-1996

(참조: 대전 행정지도(1:60,000), 유성 행정지도(1:41,000)

대전 044	대전 045	대전 046
대전 054	대전 055	대전 056

2. 지도 데이터 항목 및 내용

표 6-3 지도 데이터 항목 및 내용

항 목	구 분	coverage	table	code	입력 내용	비 고
행정구역	구 경계	Ad_gu	PAT AAT	1	polygon, 건물 명	
	동 경계	Ad_dong	PAT AAT	2	polygon, 동이름	유성구
도곽경계	도엽별 경계	Box	AAT	3	line	
도 로	고속도로	Road	AAT	11(41)	line, 중심선	()안의 숫자:중심선 코드
	도로#1			12(42)	line, 중심선	
	도로#2			13(43)	line, 중심선	
	도로#3			14	line	
건 물	boundary(아파트단지, 학교, 엑스포 공원)	Bd_bound	PAT AAT	21	polygon, 건물 명	
	boundary 내 건물	Bd1	AAT AAT	22	polygon	
	기타 건물	bd2	PAT AAT	23	polygon, 건물 명	
	관공서(구청, 동사무소, 소방서)	Bd3	PAT AAT	24	polygon, 건물 명	
	기타(연구소, 주유소, 병원)	Bd4	PAT AAT	25	polygon, 건물 명	
하 천	하천	Stream	AAT	31	line	

참고:

o dxf file명은 coverage명과 동일함

o dxf file의 layer 구분은 coverage의 code와 동일함

o coverage table의 item 구성

item name	length	type	n.dec	내 용
code	3	numeric	0	layer 구분
name	40	character	-	이름(행정명, 건물명)

o 추가 작업 내역

표 6-4 지도 데이터 수정 항목

작업coverage	dxf name	별도 분리
Ad_gu	GuBound	
Ad_dong	DngBound	
Box	MapSheet	
Road	RoadExp	고속도로
	RoadExpc	고속도로 중앙선
	Road#1	도로#1
	Road#1C	도로#1 중앙선
	Road#2	도로#2
	Road#2C	도로#2 중앙선
	Road#3	도로#3
Stream	Stream	
Bd_bound	APTComp	아파트 단지
	EXPO	엑스포 공원
	School	학교
Bd1	InnerBld	
Bd2	ETCBld	
Bd3	PubBld	
Bd4	Inst	연구소
	GasStn	주유소
	Hospital	병원

Polygon : 학교, 아파트 단지, 엑스포 공원, 연구소

Node : 공공기관, 병원, 주유소, 건물

Chain : 도로, 행정경계, 하천

3. 시범 지역 자료

o Polygon

- apartcomp.pft:

>id,class-name,name,type,exs,numbld,birth

+ imagename(3),color(green)

- innerbld.pft:

>id,class-name,name,type

+ imagename(9),height,color(cyan)

- school.pft:2

>id,class-name,type,exs,birth,degree,student

+ imagename(2),color(yellow)

o Node

- etcbld.nft:4

>id,class-name,name,type,exs,birth,aoo,bfc,zv2

+ imagename(4),height,color(magenta)

o Chain

- road#1.cft

>id,class-name,name,type,exs,loc,rst,use

+ imagename(3),color(red)

- road#1c.cft

>id,class-name

+ color(black)

- road#2.cft

>id,class-name

+ color(blue)

- road#2c.cft

>id,class-name

+ color(white)

- road#3.cft

o image-file

o sound_file

Polygon:School

id 03 대덕고등학교

Audioclip: daedukhs.au

id 06 대덕초등학교

Audioclip: daedukps.au

Node:etcbld

id 06 우래옥

Audioclip: woorae.au

id 07 마노

Audioclip: mano.au

id 08 대덕 실내수영장

Audioclip: indoor.au

Chain:road_1

id 115 도룡로

Audioclip: doryong.au

여 백

제 7 장 결 론

현재 미국을 비롯한 GIS 선진국에서는 국가 GIS 사업을 통하여 구축된 지형공간정보의 유통을 위한 Internet의 활용과 연계된 연구과제가 활발히 진행 중에 있다. 이러한 선진국의 여러 사례를 분석하여 보면 공간정보의 유통은 네트워크를 이용하는 것이 일반적임을 알 수 있다. 특히 현재 제반 전산 및 소프트웨어개발분야에 대한 인터넷 응용기술의 개발동향은 GIS 분야에서도 점차 비중이 커가는 상황에 있다고 볼 수 있다. 부연하여 최근 개발된 웹(WWW) 브라우즈의 등장으로 일반인들의 인터넷 이용은 급증하였고 이에 따른 기술의 개발 역시 그 속도가 상당히 빠르다. 그래서 외국에서는 이러한 웹 브라우즈를 이용한 공간정보의 관리와 검색을 위한 연구가 활발히 이루어지고 있다.

본 연구와 관계된 관련연구의 기술발전동향을 고려할 때 일종의 차세대 GIS를 위한 네트워크기반 GIS 자료입출력, 편집, 관리기능시스템으로 각광받을 수 있는 인터넷 기반 GIS S/W로서 세계적으로 GIS관련 업계나 학계 등에서도 현재 연구가 활발히 진행 중에 있으며 시기적으로도 본 연구의 수행시점이 적절하기 때문에 연구발전동향에 보조를 맞추는데 문제가 없으며 연구개발에 따른 시제품은 전반적으로 기존의 GIS 응용 분야뿐만 아니라 새로운 응용 및 활용분야 창출에 크게 기여할 것으로 생각된다.

한편, 본 과제의 주요 연구성과를 대비한 기대 효과는 다음과 같이 설명될 수 있다.

o 기술적 측면

Internet 응용개발언어(예: JAVA)를 이용한 소프트웨어 개발은 현재 전

세계적으로 여러분야에서 진행중인 하나의 중요한 흐름이므로 Web을 기반으로 한 GIS S/W 개발 역시 많은 기술적인 노하우를 축적할 수 있다.

- JAVA를 이용한 벡터형 자료입출력기능 구현
- JAVA를 이용한 공간객체 편집기능 구현
- Internet상에서의 GIS-DBMS 접속기술
- 다양한 Web Browser지원 및 응용기술

o 경제·산업적 측면

본 과제의 주요 연구주제인 네트워크 상에서의 공간자료 입출력기능, 자료편집기능, 공간 DB로의 자료 관리 기능 등은 이러한 기술적 요구사항에 시기적으로 적절한 주제가 된다. 현재 인터넷을 기반으로 한 범세계적인 네트워크는 공공자료로의 이용을 용이하게 하는데 대부분의 공공자료는 내재적으로 그래픽 자료로서 또는 속성자료로서 GIS 자료로서의 전환 및 이용이 충분히 가능한 자료들이 대부분이다. 공간자료 입출력기능을 통하여 수요자가 공공자료를 취득, 등록하여 효과적인 편집기능과 공간 DB로의 자료관리기능을 통하여 실제 GIS에서 가장 많은 비용과 인력이 요구되는 GIS-DB를 수요자의 업무특성에 적절하도록 구축하는 것은 경제적인 측면에서 시스템운용에 있어서의 절대적인 비용 절감 효과를 가져올 수 있다.

본 연구 성과물의 예상되는 활용 분야는 다음과 같은 분야를 들 수 있다.

- 수요자의 요구사항에 맞는 Web Browser를 통한 Geo-data/information의 공급시스템 구축
- 저렴한 비용으로 가능한 Information Archive/Host Site관리/User Web

Site 관리시스템 개발

- Internet기반 GIS S/W또는 개별분석모듈과의 유기적인 통합/운영시스템 개발
- Internet Access가 가능한 Site에서의 원격 GIS 교육프로그램

우리나라의 GIS S/W 개발은 Internet과 같은 Network 기술과 연계되어 앞으로도 지속적으로 추진되어 할 것으로 판단된다. 결론적으로 최근 국제 상업시장을 중심으로 한 GIS 소프트웨어개발 연구동향은 다음과 같은 몇 가지의 중요한 발전양상을 띄고 발전하고 있으며 본 연구성과와의 직접적인 연계성이 가능하므로 향후 국내 GIS 연구동향이나 기술발전방향정립에도 본 연구 결과가 많은 기여를 할 것으로 생각된다.

○ 인터넷 기반 GIS: 1997년 현재 인터넷 환경을 이용하여 공간검색과 기본질의 기능을 수행하는 일종의 공공정보 서비스 지원 시스템으로 인식되는 GIS이나 본 연구 개발환경이 자체 기술에 의한 순수 자바 기술로 이루어졌기 때문에 본 연구성과자체가 인터넷 GIS의 구성요소를 모두 충족하고 있다.

○ 엔터프라이스 GIS: GIS운영 조직과 관계된 통합환경 GIS로 정의되며 GIS 사업수행과 관련된 하드웨어, 소프트웨어 및 자원관리 요소까지 고려하여 통합적 시스템운영과 관련되어 있으나 본 연구개발의 주요 성과중 하나인 객체지향 개념에서 3-tier 모델은 이러한 시스템 조직화에 필요한 기본 조건을 이미 갖추고 있으므로 현재 인트라넷 기반의 엔터프라이즈 GIS로의 확장성을 보장한다.

○ 컴포넌트 GIS: GIS기능 중 부분적인 모듈을 시스템개발을 위한 독립적인 소프트웨어로 개발하여 전문적인 특화기능의 GIS구현이 가능하며 상업용 엔터프라이스 GIS 구현을 위한 조건을 만족하고 있다.

○ 데스크 탑 GIS: 기존의 고가형 워크스테이션 GIS의 기능중 일반적인 사용도가 높은 기능을 중심으로 테스크 탑 PC환경에서 운영할 수 있도록 한 저가형 소규모 프로젝트수행을 위한 GIS로 기본 개념이 인식되고 있으며 인터넷환경에서 운용 가능한 공간정보 편집관리기술 및 데이터 베이스처리기술은 별도의 시스템 없이 인터넷에 접속하여 사용이 가능하기 때문에 일반 사용자와 전문 사용자가 데스트 탑 GIS의 기능을 본 연구 성과를 통하여 충분히 활용할 수 있을 것으로 생각된다.

참고 문헌

[1] N.B.Kriegel and B.Seeger, The R*-tree : An Efficient and Robust Access Method for Points and Rectangles, Proc. ACM SIGMOD Conf. Management of Data, 1990, pp.322-331

[2] N. Beckmann, et. al., The R*-tree : An Efficient and Robust Access Methods for Point and Rectangles, Proc. of ACM SIGMOD, pp.323-331, 1990.

[3] Michael Girdley, Kathryn A. Jones, et al, Web Programming with Java, Sams Net, 1996.

[4] JDBC : A Java SQL API, Sun Microsystems, http://java.sun.com

[5] The JDBC API Version 1.20, Sun Microsystem., http://splash.javasoft.com/jdbc

[6] MIL-STD 2407, Vector Product Format, Feature Class Relations, Appendix C, pp.114-139

[7] M.S.Kim, Y.S.Shin, M.J.Cho, K.J.Li, A Comparative Study of Spatial Access Methods, Proc. ACM GIS, 1995, pp.29-36

[8] R. Laurini and D.Thompson, Fundamental of Spatial Information Systems,

Academic Press, 1991, pp.175-216

[9] M.S.Kim, K.S.Kim, K.W.Lee, Development of an Adaptive Topology Building Methodology Using Spatial Access Method on WWW

[10] Agatha Y. Tang, Teresa M. Adams, and E. Lynn Usery, A spatial data model design for feature-based geographical information systems, International Journal of GIS, Vol.10, No.5, pp.643-659, 1996.

[11] E. Lynn Usery, A Feature-Based Geographic Information System Model, Photogrammetric Enginnering & Remote Sensing, Vol.62, No.7, pp.833-838, 1996.

[12] Hanan Samet, The Design and Analysis of Spatial Data Structures, Addison-Wesley, 1994.

[13] Hanan Samet, Applications of Spatial Data Structures, Addison-Wesley, 1995.

[14] Department of Defense USA, Vector Product Format, 1995.

[15] Joseph O'Rourke, Computational Geometry in C, Cambridge Universty, 1994.

[16] Robert Sedgewick, Algorithms in C, Addison-Wesley, 1990.

[17] Robert Laurini and Derek Thompson, Fundamentals of Spatial Information Systems, Academic Press, 1994.

[18] Robert G. Cromley, Digital Cartography, Prentice Hall, 1992.

[19] Franco P. Preparata and Michael Lan Shamos, Computational Geometry An Introcuction, Springer-Verlag, 1985.

[20] Michael F. Worboys, GIS:A Computing Perspective, Taylor & Francis, 1995.

[21] Jeffrey Star and John estes, Geographic Information Systems An Introduction, Prentice Hall, 1990.

[22] Graeme F. Bonham-Carter, Geographic Information Systems for Geoscientists Modelling with GIS, Pergamon, 1996.

[23] Patrick Chan and Rosanna Lee, The Java Class Libraries : An Annotated Reference, Addison-Wesley, 1997.

[24] Ken Arnold and James Gosling, The Java Programming Language, Addison-Wesley, 1996.

[25] H.M. Deitel and P.J. Deitel, Java How to Program, Prentice Hall, 1997.

[26] Sydow, Dan Parks, "Jumping to Java", 1997, 삼양출발사.

[27] Sun Microsystem, "Introduction to Java Programming, 1996, Sun Microsystem, Inc.

[28] MapGuide, ESRI, http://www.esri.com, http://maps.esri.com/srmap/srmap.html

[29] ActiveMaps, InternetGIS Com, http://www.internetGIS.com

[30] 김민수, 김광수, 이기원, WWW 기반 GIS에서 R*-tree를 방법을 이용한 공간 데이터베이스 설계 및 구현, 한국정보처리학회 1997 추계 학술 발표 논문집, Vol.1., pp.405-410

[31] 이 경호, 조 성배, 최 윤철, 지리정보시스템을 위한 지식 기반 자동 벡터라이징 시스템, 정보과학회 가을학술발표논문집(A), 제23권 2호, pp.295-298, 1996.

[32] 시스템공학연구소, "GIS 관리자 과정" 교재, 1996, 시스템공학연구소.

[33] 홍봉희, 문상호, 성원모, GIS와 Internet의 통합 기술, 한국정보과학회 데이터베이스 연구회지, 1996, 8, pp.97-115

[34] 김평철, 웹을 위한 데이터베이스 통로의 분류체계, 데이터베이스 1997 춘계

튜토리얼, 웹과 데이터베이스, pp.49-70

[35] 박재범, Java와 데이터베이스, 데이터베이스 1997 춘계 튜토리얼, 웹과 데이터베이스, pp.167-182

[36] 류희상, 김경배, 조영섭, 이영걸, 배해영, WWW 환경에서 지리정보시스템의 설계 및 구현 : GEO/WEB, 한국정보과학회 1997 춘계 학술 발표 논문집, Vol.2., pp.171-174

[37] 캐드랜드, 지리정보시스템의 이해, 1992.