WWW 상의 도로 주행 안내 시스템 구현

권근주 심호현 차상균
서울대학교 전기공학부
{icdi, shimho, chask}@kdb.snu.ac.kr

Implementation of Car Navigation System on the WWW

Keun J. Kwon Ho H. Shim Sang K. Cha
School of Electrical Engineering, Seoul National University

요 약

인터넷의 발전으로 WWW 상에서 지도 정보를 서비스하는 사이트들이 늘어나고 있다. 본 논문은 이러한 WWW 상의 지리 정보 서비스를 확장한 도로 주행 안내 시스템의 구현에 관하여 기술한다. 도로 주행 안내 시스템은 도로, 건물 등의 지리 정보를 표시해주며 사용자의 최단 경로 질의를 받고 빠른 시간 내에 최단경로를 탐색할 수 있어야 한다. 본 연구에서는 이러한 요구를 수용할 수 있도록 OODBMS, CORBA, Java 를 사용하여 WWW 상의 도로 주행 안내 시스템을 설계 및 구현하였다. 이를 위해 이 논문에서는 도로 데이터 캐싱과 그래프 모델링, HEPV (Hierarchical Encoded Path View) 알고리즘 구현 등의 사항을 기술하였다.

1. 서론

WWW 의 보편화로 지도 정보를 제공하는 사이트들이 늘어나고 있으며 여러 검색 엔진의 부가 서비스로도 등장하고 있다. 그러나 이러한 시스템은 현재 지도 정보의 표시와 지형, 지물 검색만 지원한다. 도로가 복잡하고 교통체증이 심한 우리나라의 경우 도로 주행 안내 서비스는 유용한 지리 정보 서비스 중의 하나이지만 아직 WWW 을 통해 서비스되고 있지 않다.

기존의 도로 주행 경로 안내 프로그램은 CDROM 에 도로 데이터를 담아 독립적으로 실행되는 형태가 있었다. 이러한 방법은 변화하는 교통 상황을 최단 경로 탐색에 반영하기 어렵다. 현재의 WWW 상의 지도 서비스는 이미지를 사용하여 지도를 그리기 때문에 사용자 이동마다 데이터를 받아와야 한다. 도로 주행 안내 서비스는 지도 안내 서비스보다 스크롤 및 확대 축소가 훨씬 많기 때문에 이러한 방법을 적용할 경우 통신 비용이 문제가 된다. 최근에 개발된 버스 노선 안내 시스템은 변형된 A* 알고리즘을 사용하고 있다. [1] 그러나 모든 도로를 탐색하는 도로 주행 안내 서비스는 버스 노선 안내 서비스보다 노드의 수가 훨씬 많기 때문에 탐색이 느려질 수 있다.

본문에서는 WWW 상의 도로 주행 안내 시스템의 구현 경험에 관하여 기술한다. 도로 주행 안내 시스템은 사용자가 웹 브라우저 상에서 출발점과 목적지를 입력하면 가장 빠른 시간에 도착할 수 있는 주행 경로를 알려주는 시스템이다. 도로 주행 안내 시스템의 실행 환경을 나타내면 그림 1 과 같다. 클라이언트는 웹 브라우저 상에서 실행되는 Java 애플릿으로 구현하였다. 클라이언트에서 도로 데이터를 서버에 요청하면 서버는 DB 에 저장된 데이터를 접근하여 필요한 데이터를 클라이언트로 전송한다. 또한 최단 경로 탐색도 수행하여 결과를 넘겨준다. 서버와 클라이언트는 CORBA 를 사용하여 통신하였으며 도로 데이터는 OODBMS 를 사용하여 저장하였고 HEPV 알고리즘을 사용하여 최단 경로를 탐색하였다.

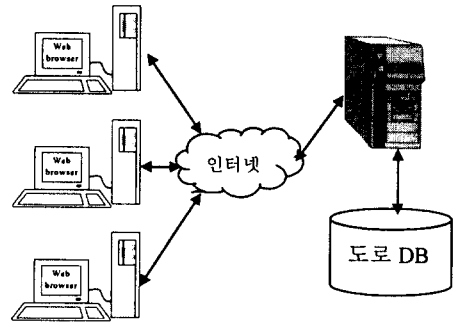

그림 1. 도로 주행 안내 시스템의 실행 환경

본 논문의 2 장에서는 도로 주행 안내 시스템의 요구사항에 대해서 살펴보고 3 장에서는 요구사항에 따라서 설계된 시스템의 구조와 구성 요소들에 대해서 설명한다. 4 장에서는 각각의 구현에 관하여 설명하고 5 장에서는 이를 마무리 한다.

2. 도로 주행 안내 시스템의 요구사항

도로 주행 안내 시스템은 첫째 다양한 사용자 인터페이스를 지원해야 한다. 사용자는 도로와 지형, 지물 등의 지리 데이터를 보고 출발지와 목적지를 선택할 수 있어야 한다. 이를 위해서는 지형, 지물 검색이나 공간 영역 질의, nearest neighbor 질의 등을 지원해야 하며, 또 탐색 된 경로를 따라서 스크롤 및 확대, 축소가 가능해야 한다.

둘째, 빠른 탐색 알고리즘이 필요하다. 최단 경로 탐색은 전체 도로 데이터를 필요로 하는 질의이고 데이터 양은 매우 많기 때문에 클라이언트는 모든 데이터를 갖질 수 없다. 따라서 최단 경로 탐색은 서버에서 수행하여야 하는데 사용자가 많은 경우 클라이언트의 최단 경로 탐색에 많은 부담이 걸리게 된다. 한편 도로의 교통 상황은 자주 변하기 때문에 빠른 탐색 알고리즘을 사용하더라도 주행 비용의 갱신에 대한 고려도 포함되어야 한다.

셋째, 복잡한 지리 데이터를 송신하고 다양한 질의를 받기 위해서는 유연성 있고 확장성 있는 통신 프로토콜이 필요하다. 질의 데이터의 내용이나 타입은 추가될 수 있고 사용자가 원하는 서비스의 방법도 변하기 때문에 쉽게 변경 및 추가가 가능해야 한다.

3. 시스템 구조

위와 같은 요구 사항을 해결하기 위해서 도로 주행 안내 시스템의 클라이언트는 Java 애플릿을 사용하여 웹 브라우저 상에서 실행될 수 있도록 하였으며 벡터 데이터를 그리도록 설계 하였다. Java 애플릿은 CGI 나 스크립트언어에 비해 사용자와의 상호 작용 기능을 강화할 수 있다. 또한 화면의 스크롤과 확대, 축소가 많은 경우 통신 비용이 높기 때문에 이를 줄이기 위한 캐싱 등의 알고리즘을 구현하기 용이하다. 캐싱을 하는 경우 래스터 이미지보다 벡터 데이터를 사용하는 것이 효율적이며 도로 선택 및 최단 경로 표시를 위해서도 벡터 데이터가 필요하다. 도로 주행 안내 시스템의 서버는 OODBMS 상에 도로 데이터를 저장하도록 하였다. 가변길이 데이터가 많은 공간 데이터와 참조가 많은 그래프 모델을 구현하기 위해서는 RDBMS 보다 OODBMS 가 적당하다. 또한 공간 질의를 효율적으로 수행하기 위해서 공간 인덱스로써 R*-tree[2]를 사용하였다.

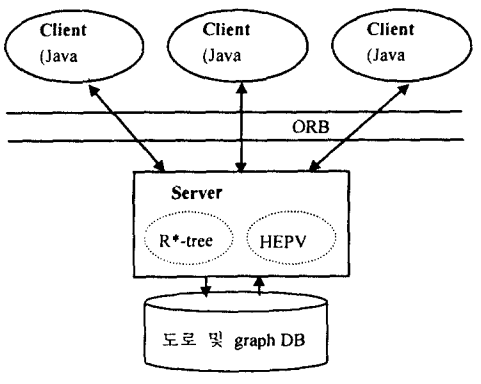

그림 2 도로 주행 안내 시스템 구조

최단 경로 탐색 알고리즘으로는 계층적 탐색 알고리즘 중 갱신이 용이한 HEPV (Hierarchical Encoded Path View) 알고리즘[3]을 사용하였다. 그래프를 계층적으로 나누고 최단 경로를 미리 계산하여 저장하는 계층적 탐색 알고리즘[4][5]은 큰 그래프에 대해서도 빠른 시간 내에 탐색 할 수 있으나 링크의 비용이 변할 경우 상위 그래프를 다시 생성해야 하기 때문에 갱신 비용이 크다. HEPV 는 계층 구조에 따라 생기는 그래프의 저장 비용은 상대적으로 크나 비용 갱신 시간은 빠르다.

OODBMS 를 접근하는 C++서버와 Java 애플릿을 연결하기 위해서 확장성 있는 객체 기반 통신 표준인 CORBA 를 사용하였다. CORBA 는 데이터 타입을 변경 및 추가 하기 쉽고, CORBA 와 OODBMS, Java 는 모두 객체 지향 모델이므로 연동하기 쉽다. 도로 주행 안내 시스템의 구조를 그림으로 나타내면 그림 2 와 같다.

4. 시스템 구현

4.1. 구현 환경

도로 주행 안내 시스템의 서버는 Sun Enterprise 3000 위에 구현하였고 컴파일러는 Sun SPARCcompiler C++4.0 을 사용하였다. OODBMS 는 Objectivity Inc.사의 Objectivity/DB 5.0 을 사용하였으며 ORB 로는 Iona 사의 Orbix2.3MT 와 OrbixWeb3.0 을 사용하고 클라이언트는 JDK 1.2 상에서 개발되었다. 데이터는 약 70,000 개의 교차로와 110,000 개의 도로로 이루어진 서울시 도로 데이터를 사용하였다.

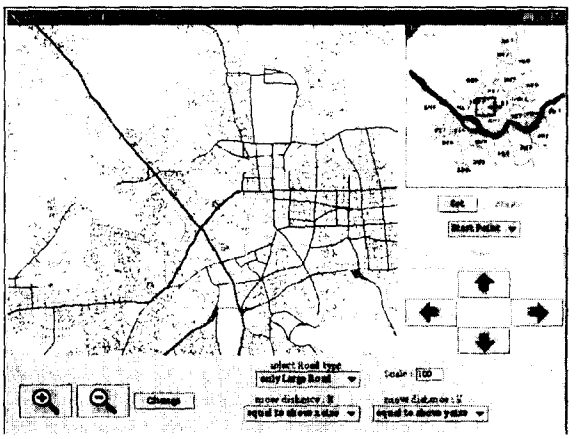

그림 3. 도로 주행 안내 시스템의 GUI

4.2. GUI 와 데이터 캐싱

도로 주행 안내 시스템의 GUI 를 살펴보면 그림 3 과 같다. 서울시 전체 중 선택한 영역의 도로 데이터를 보여주며 상하 좌우로 스크롤과 확대, 축소가 가능하다. 또한 출발점과

도착점을 선택하고 최단 경로 탐색 질의를 할 수 있도록 되어 있다.

GUI 에서 사용자가 스크롤이나 확대, 축소 명령을 내리면 화면을 다시 그려야 하고 그때마다 해당 영역의 도로 데이터가 필요하다. 이 때마다 데이터를 서버로부터 받아오면 통신 비용이 크므로 최근에 가져온 도로 데이터는 클라이언트에서 캐싱을 해야 한다. 캐싱을 위해서는 여러 객체를 하나의 단위로 묶어 주어야 하며 데이터 요청이 왔을 때 요청한 데이터가 캐시에 있는지 확인할 수 있어야 한다. 화면을 다시 그려주기 위해서는 영역 질의가 반복되므로 데이터 전체의 공간 영역을 격자로 나누고 격자 내에 속한 도로를 하나의 묶음으로 캐싱하였다. 영역 질의가 반복되었을 때 해당 질의 결과가 캐시에 있는지 확인하기 위해서 격자로부터 Quad tree 를 생성하여 유지한다.

4.3. 데이터 모델과 최단 경로 탐색 알고리즘

도로를 DB 에 저장하고 최단 경로 탐색할 수 있는 그래프로 만들기 위해서 도로 데이터를 roadsegment 와 segmentjoin, 두 가지 객체로 모델링 하였다. Roadsegment 는 도로를 나타내며 linestring 을 공간 속성으로 갖고 차선 수, 일방 통행 여부, 도로 종류 등의 속성도 갖는다. Segmentjoin 은 교차로를 나타내며 회전 금지 등의 속성을 갖으며 교차하는 roadsegment 들과 양방향 관계가 있다.

도로 데이터로부터 생성된 그래프는 도로의 일방통행, 회전 금지 정보 등을 포함해야 한다. 따라서 교차로를 교차하는 도로의 개수만큼 쪼개고 여기서 금지된 링크와 노드들을 삭제하여 그래프를 완성한다. 좌회전 금지가 있고 일방통행 도로가 있는 사거리를 그래프로 모델링 하면 나타내면 그림 4 와 같다.

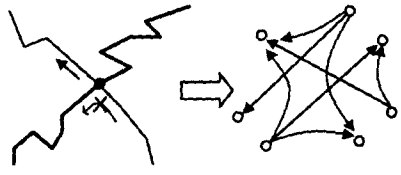

그림 4. 도로 데이터의 그래프 모델링

이 그래프를 HEPV 알고리즘에 사용하기 위해서는 여러 개의 그룹으로 나누고 이 그룹으로 계층 그래프를 생성해야 한다. 계층적 탐색 알고리즘에서는 그래프를 그룹으로 나누는 방법에 따라서 성능에 큰 영향을 미치게 된다. 그룹 수가 많으면 그룹내의 노드 숫자는 적으나 상위 그룹이 커지게 된다. 또한 그룹 내의 경계 노드 숫자가 많으면 그래프 생성에 많은 시간이 걸린다. 본 시스템에서는 그룹과 그룹사이의 링크가 최소가 되게 하는 Metis 프로그램[6]을 사용하여 그래프를 나누고 3 레벨이 되도록 계층 그래프를 생성하였다. 레벨 1 의 그룹은 350 개로 하였으며 레벨 2 의 그룹은 15 개로 하였다. 그룹 수에 따른 상위 노드 수와 그룹내의 노드, 경계 노드 수의 변화를 나타내면 표 1 과 같다.

그룹 수	상위 노드 수	그룹내의 노드	그룹 경계노드
10	3265	1548	391
15	**3560**	**1026**	**271**
20	5234	842	300

표 1. 레벨 2 에서의 그룹 수와 노드수의 관계

4.4. CORBA 인터페이스 정의

CORBA 인터페이스는 우선 DB 에 저장된 객체를 IDL(Interface Definition Language)로 정의하여야 한다. 그런데 DB 객체를 CORBA 객체로 정의 할 경우 CORBA 객체는 method shipping 만을 지원하기 때문에 클라이언트에서 객체 하나하나를 가져오는데 너무 많은 시간이 걸린다.[7] 따라서 DB 객체를 IDL 의 struct 로 정의하고 질의 결과를 한꺼번에 가져오도록 하였다. 여기에 DB 를 대표하는 객체를 정의하고 영역 질의, 검색 질의, 최단 경로 탐색 질의를 할 수 있는 함수를 추가하였다.

5. 결론

본 논문에서는 WWW 을 통한 도로 주행 안내 시스템을 설계하고 구현하였다. 다양한 사용자 인터페이스와 효율적인 이동 및 확대 축소를 위하여 Java 애플릿과 벡터 데이터를 사용하였으며 지리 데이터 저장 및 탐색 알고리즘 구현을 위하여 OODBMS 를 사용하였다. 그리고 Java 와 OODBMS 를 연동하기 위하여 CORBA 를 사용하였으며 빠른 경로 탐색을 위하여 HEPV 알고리즘을 구현하였다.

향후 연구 과제로는 사용자가 도로를 쉽게 찾기 위해 여러 가지 지형 지물 데이터의 추가와 검색을 지원해야 한다. 한편 벡터 데이터는 축소를 하면 많은 데이터를 그리기 때문에 스케일에 따라서 상세도를 조절할 수 있는 기능도 필요하다. 또한 계층 그래프를 최적화하는 방법을 체계화 하여 임의의 그래프 구조에 대해서도 계층 그래프를 생성하도록 확장하여야 한다.

참고 문헌

[1] 배수강, 이승룡, 최대순, 정태충, 승현우, "웹기반 대중교통 안내 시스템", 정보과학회지 논문지 VOL. 5, NO.4, 426-439, 1999

[2] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, Bernhard Seeger, "The R*-tree: An Efficient and Robust Access Method for Points and Rectangle", Proc. ACM SIGMOD Int'l Conf. on Management of Data, 322-331, 1990

[3] Ning Jing, Yun-Wu Huang, and Elke A. Rundensteiner, "Hierarchical Encoded Path Views for Path Query Processing: an Optimal Model and Its Performance Evaluation", IEEE Transactions on Knowledge and Data Engineering VOL.10, NO.3, 409-432, 1998

[4] Sungwon Jung and Sakti Pramanik. "HiTi Graph Model of Topographical Road Maps in Navigation Systems", Proc. of IEEE Twelfth Int'l Conf. on Data Engineering, 76-84, 1996

[5] Shashi Shekhar, Andrew Fetterer, and Bjajesh Goyal, "Materialization Trade-Offs in Hierarchical Shortest Path Algorithms", Int'l Symposium on Spatial Databases, 94-111, 1997

[6] METIS: Family of Mutilevel Partitioning Algorithms, http://www-users.cs.umn.edu/~karypis/metis/

[7] 황상용, "CORBA 를 통한 효율적인 Object Database 접근", 서울대학교 공학석사학위논문, 1999