2000년 한국정보처리학회 추계 학술논문발표집 제7권 제2호

관계형 데이터베이스를 이용한 그래프 라이브러리 개발

추인경, 박휴찬
한국해양대학교 컴퓨터공학과
e-mail: myfall@ce.kmaritime.ac.kr, hcpark@hanara.kmaritime.ac.kr

Development of Graph Library on the Relational Database

In-Kyung Chu, Hyu-Chan Park
Dept. of Computer Engineering, Korea Maritime University

요 약

그래프는 실세계의 많은 문제를 푸는데 아주 강력한 방법을 제공한다. 이와 같은 그래프를 효율적으로 표현하기 위한 자료구조와 그래프 연산에 대한 알고리즘이 개발되어 왔다. 본 논문에서는 그래프를 관계형 테이블로 표현하고, 그래프에 대한 연산과 알고리즘을 라이브러리화 하는 방법을 제안한다. 제안한 방법은 관계형 데이터베이스를 이용하여 개발할 수 있으며, 개발된 라이브러리는 그래프로 모델링되는 실세계의 많은 문제를 푸는데 손쉽게 활용할 수 있을 것이다. 또한, 방대한 양의 그래프를 효율적으로 관리할 수 있으며 다수의 사용자가 공유할 수도 있을 것이다.

1. 서론

그래프는 수학의 한 분야로 network 설계, data structures, process scheduling, computations 등 시스템 영역뿐만 아니라 실세계의 많은 문제를 해결하는 아주 강력한 방법을 제공한다. 이러한 이유로 과거부터 현재까지 그래프를 효율적으로 표현하기 위한 자료구조와 그래프 연산에 대한 알고리즘이 개발되어 왔다. 하지만 이렇게 개발된 그래프 자료구조와 그래프 알고리즘들을 실세계에 적용하는 것은 간단한 작업이 아니다. 특히 그래프에서 사용되는 용어와 기호들이 다른 학문분야와 견주어 표준화가 미흡한 상태이기 때문에 그래프에 관해서 전문적인 지식을 소유한 개발자가 아닌 다른 분야의 전문가들이 실세계에서 발생하는 문제를 해결하기 위해 그래프를 직접 적용하는 것은 결코 간단한 작업이 아니다.

본 논문에서는 실세계에서 발생하는 문제에 대해 그래프를 효과적으로 적용하기 위해서 관계형 데이터베이스를 이용한 그래프 라이브러리(graph library)설계를 제안한다. 이러한 설계는 방대한 양의 그래프 데이터의 효율적인 관리, 데이터 중복(redundancy),데이터 비일관성(inconsistency), 데이터 무결성(integrity), 다수 사용자들의 효율적인 데이터 공유가 가능할 것이다.

그래프는 vertex와 edge의 구조를 가지므로 그래프 데이터베이스의 테이블은 그래프 정보 테이블(graph information table), vertex 테이블, edge 테이블로 구성되며 그래프 아이디(*g_id*), vertex 아이디(*v_id*), edge 아이디(*e_id*)가 각 테이블의 기본 키(primary key)가 된다. 그래프 라이브러리는 DBMS 접속/ 해제(connect/disconnect), 그래프 생성(create)/갱신(modification)/저장(save)/삭제(drop), vertex 와 edge에 대해 삽입(insert)/삭제(delete)/추가(addition)/갱신(update)/정보추출(get_data)/연산(operation), 경로탐색(search path)의 부분으로 구분해서 설계된다.

본 논문은 2장에서 그래프 라이브러리를 설계하는 데 필요한 기본적인 그래프이론과 표기법 및 응용한 사례를 살펴보고, 3장에서 그래프 관계형 데이터베이스 테이블의 구조, 4장에서 그래프 라이브러리 설계에 대해 논하고, 5장에서 결론 및 향후 과제를 제시한다.

2. 그래프이론과 응용사례

2.1 기본적인 그래프 이론과 표기법

이 장에서는 그래프 라이브러리(graph library)를 설계에 필요한 그래프 기본과 표기법을 다음과 같이 정리한다.

[정리 1] 그래프 *G=(V, E)*는 *vertices*라고 불리는 요소의 유한집합인 *V* 와 두 개의 *vertex* 쌍으로 구성되는 *edge*라고 불리는 요소의 집합 *E*로 구성되어 있다. *edge*가 방향을 나타내면 방향그래프(*digraph/ directed graph*)라하고 *edge*가 가중치를 가지면 가중치 그래프(*weighted graph*)라 한다.

[정리 2] *vertex u*와 *v*를 끝점으로 가지는 *edge e*는 *vertex u*와 *v*에 달려있다(*incident with*)라고 말하고, *u*와 *v*는 서로 인접(*adjacent*)인 *vertex*라고 한다

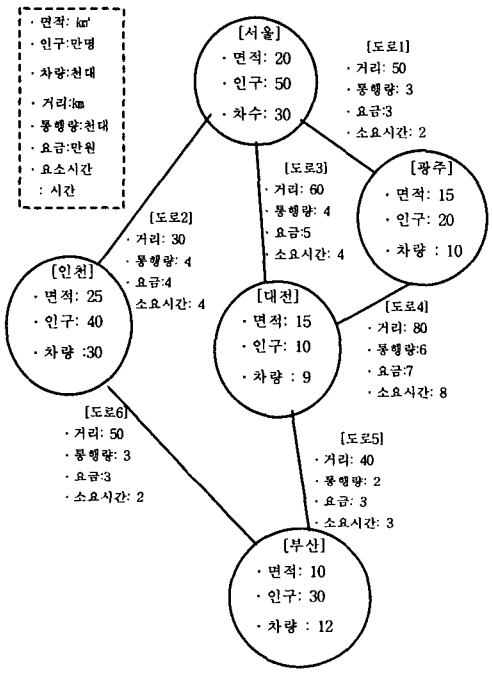
[정리 4] *vertex v*의 차수(*degree*)는 *v*에 달린 *edge e*의 수로서 *deg(v)* 라고 표시한다.

[정리 5] 그래프 $G=(V_G, E_G)$ 와 $H=(V_H, E_H)$에서 $V_H \subseteq V_G$ 이고 $E_H \subseteq E_G$ 이면 H를 G의 부그래프(*subgraph*)라고 한다. 만약, H가 G의 부그래프이고 $V(H)=V(G)$이면, H는 G의 스패닝그래프(*spanning subgraph*)라고 한다.

[정리 6] 그래프 *G*의 두 *vertex*를 *x, y*라 하면. *walk*는 *vertex x*에서 시작해서 *vertex y*에서 끝나는 유한한(finite) *vertex*와 *edge*의 순열(sequence)이며 *x-y walk*로 표시한다. *walk*의 길이(length)는 *walk* 내의 *edge*의 개수이다. 이때 순열에서 *edge*가 중복되지 않는 것을 *x-y trail, vertex*가 중복되지 않는 것을 *x-y path*라고 한다.

2.2 그래프의 응용의 사례

(그림 1)은 우리 나라 주요도시와 도로들에 대한 정보를 그래프로 표현했다. 도시는 vertex, 도로는 edge로 대응시켰다. 각각의 vertex와 edge들은 많은 data를 저장하고 있다.



(그림 1) 그래프 응용 사례

3. 그래프 관계형 데이터베이스 테이블의 구조

그래프 데이터베이스는 그래프에 대한 전체적인 데이터를 저장하는 그래프 정보 테이블(graph information table), vertex의 정보를 저장하는 vertex 테이블, edge의 정보를 저장하는 edge 테이블로 구성된다. 각 테이블의 구조를 이 장에서 살펴본다.

3.1 그래프 정보 테이블 (Graph Information Table)

이 테이블은 메타 데이터의 성격을 가지고 있으며 테이블 속성들은 고정되어 있다. 그래프 아이디

(*g_id*)를 기본 키로 하고, 그래프 이름(*g_name*), 그래프에 대한 키워드 (*key_word*), 그래프 작성 시작 날짜(*start_day*)와 최종 수정 날짜(*update-day*), 기타정보(*ect*)로 구성된다.

g_id	그래프 아이디, PK
g_name	그래프 이름
key_word	키워드
start_day	그래프 작성 시작일
update_day	그래프 최종 수정일
ect	기타정보

(그림 2) *GI_table* : 그래프 정보 테이블의 구조

3.2 Vertex 테이블 (Vertex Table)

vertex table은 vertex 데이터를 저장한다. 기본 키 *v_id*와 *v_att(1~n)*로 구성된다. vertex와 edge table은 설계 대상과 목적에 따라 데이터 종류와 개수가 유동적 때문에 vertex와 edge의 속성(*v_att* / *e_att*)은 1~ *n*개로 유동적으로 설계한다.

v_id	vertex 아이디, PK
v_att1	vertex 속성 1
v_att2	vertex 속성 2
. . .	. . .
v_att n	vertex 속성 n

(그림 3) *V_table* : vertex 테이블 구조

3.3 Edge 테이블 (Edge Table)

edge table은 edge 데이터를 저장한다. 기본 키 *e_id*와, *e_id*가 연결하는 두 개의 vertex *x*와 *y*의 id (*v_x_id, v_y_id*), 방향성 여부를 나타내는 *dir* 그리고 edge 속성 *e_att(1~n)*로 구성된다 .

e_id	edge 아이디, PK
ev_x_id	edge_id 가 연결하는
ev_y_id	두 *vertex x, y*의 *id*
dir	· *true/y* : x → y · *false/n* : x — y
e_att 1	edge 속성 1
. . .	. . .
e_att n	edge 속성 n

(그림 2) *E_table* : 테이블 구조

그래프 응용 사례 (그림 1)을 그래프 관계형 데이터베이스 테이블로 저장한 형태를 (그림 4)에서 보여준다.

· *GI_table*

g_id	*g_name*	*key_word*	*s_day*	*u_day*	*ect*
G1	수송 계획	수송	00-09-05	00-09-09	도시에 대한 물류계획

· *V_table*

v_id	*지명*	*면적*	*인구*	*차량*
V1	서울	20	50	30
V2	인천	25	40	30
V3	대전	15	10	9
V4	광주	15	20	10
V5	부산	10	30	12

· *E_table*

e_id	*ev_x_id*	*ev_y_id*	*dir*	*도로명*	*거리*	*통행량*	*요금*	*시요시간*
E1	V1	V4	n	도로1	50	3	3	2
E2	V1	V2	n	도로2	30	4	4	4
E3	V1	V3	n	도로3	60	4	5	4
...	...	...	...	...	...	...	...	...

(그림 4) (그림1)의 그래프 관계형 데이터베이스화

4. 그래프 라이브러리의 설계

4.1. 그래프 관계형 데이터베이스 연결/해제

관계형 데이터베이스로 저장되어 있는 그래프 테이블을 이용해야 하기 때문에 개발자의 DBMS 개정과 암호로 접속 및 해제하는 라이브러리가 필요하다.

· *Do_Connect(user_id, user_password)*
· *Do_Disconnect()*

4.2 그래프 정의 및 관리 라이브러리

그래프 생성(create)/ 갱신(modification)/ 저장(save)/ 열기(open)/ 삭제(drop)를 담당하는 라이브러리이다. 대표적인 라이브러리를 정리하면 다음과 같다.

- *Graph_Open/Save/Drop(g_id)*
- *Graph_Create(g_id,v_id,v_att(1~n),e_id,ev_x_id, ev_y_id,dir,e_att(1~n))*
- *Graph_Insert_Vertex(v_id,v_att(1~n))*
- *Graph_Insert_Edge(e_id,ev_x_id,ev_y_id,dir, e_att(1~n))*
- *Graph_Delete_Vertex(v_id)*
- *Graph_Delete_Edge(e_id)*

(예제 1) 그래프테이블 만들기
Graph_Create(G1,v_id,지명,면적,인구,차량,e_id,ev_x , ev_y,dir,도로명,거리,통행량,요금,소요시간);

4.3 Vertex / Edge 데이터 라이브러리

vertex 와 edge 데이터에 대해 삽입(insert)/삭제(delete)/추가(addition)/갱신(update)/정보추출(get)/연산(operation)을 담당하는 라이브러리이다. 대표적인 라이브러리를 정리하면 다음과 같다.

- *Graph_Get_Data_Vertex(v_id, v_att)*
- *Graph_Get_Max/Min/Sum/Avg_Vertex(v_id, v_att)*
- *Graph_Change_Vertex(v_id, v_att, new_value)*
- *Graph_Degree_Vertex(v_id)*

(예제 2) vertex 값 바꾸기
Graph_Change_Vertex(V1,인구, 60);

v_id	*지명*	*면적*	*인구*	*차량*
V1	서울	20	*60*	30

4.4 경로탐색 라이브러리

그래프 경로 탐색을 담당하는 라이브러리이다. 대표적인 라이브러리를 정리하면 다음과 같다.

- *Graph_Path(v1_id,v2_id)*
- *Graph_nReach_Path(v1_id,v2_id,count)*
- *Graph_Shortest_Path(v1_id, v2_id, v_att)*
- *Graph_Shortest_Trail(v1_id, v2_id, e_att)*

(예제 3) v1(서울)에서 v5(부산)까지 2번만에 도착하는 경로
Graph_nReach_Path(V1,V5,2);

V1	V2	V5
V1	V3	V5

5. 결론 및 향후 과제

그래프는 실세계에서 발생하는 문제들을 해결하는데 강력한 방법을 제공하며 과거부터 현재까지 효율적인 표현을 위한 데이터 구조와 알고리즘들이 연구되어 왔다.

본 논문에서는 그래프를 관계형 데이터베이스로 구축하고 이것을 이용하는 그래프 라이브러리를 제안함으로써 효과적으로 그래프를 실세계에 응용할 수 있는 방법을 제안했다.

향후 연구과제는 그래프 라이브러리를 좀 더 보강하고 객체 지향 데이터베이스(object-oriented database)를 이용한 그래프 라이브러리를 개발하여 성능비교를 할 것이다.

참 고 문 헌

[1] Tony T, Lee and Ming-Yee Lai, "A Relational Algebraic Approach to Protocol Verification," IEEE Transaction on Software Engineering, Vol.14, No.2, February 1998.

[2] Guting,R., "GraphDB: Modeling and Querying Graphs in Database," VLDB, pp297-308, 1994.

[3] Joachuim Biskup, Uwe Räsch and Holger Stiefeling, "An Extension of SQL for Querying Graph Relations," Computer Lang, Vol.5, No.2, pp.65-82, 1990.

[4] Robin J.Wilson and John J.Watkins, *Graphs an Introductory Application*, John Wiley & Sons.Inc, 1990.

[5] James A. Mchugh, *Algorithmic Graph Theory, Prentice-Hall*, 1990.