iSTORM에서의 공간 객체-관계 데이터 모델

박경현[U] 남광우 박성희 류근호
충 북 대 학 교 데이터베이스 연구실
{khpark, kwnam, shpark, khryu}@dblab.chungbuk.ac.kr

Spatial Object-Relational Data Model in iSTORM

Kyoung Hyun Park[U] Kwang Woo Nam Sung Hee Park Keun Ho Ryu
Dept. of Computer Science, Chungbuk National University

요 약

공간 데이터는 복합적인 속성들의 조합으로 이루어지며 연산 또한 복합적이라는 점에서 일반 데이터와 다른 특성을 갖는다. 따라서 공간 데이터는 일반 속성 데이터와 구별되는 파일이나 별도의 저장 구조를 사용하여 관리되어야 한다. 이것은 비공간 데이터와 공간 데이터간의 상호 동기화 문제와 트랜잭션의 처리 등에서 많은 문제점을 발생시키며 이를 해결하기 위해서는 공간 데이터와 비공간 데이터를 단일 데이터베이스화하여야 한다. 이 논문에서는 이러한 단일화된 데이터베이스 시스템을 지원하기 위한 공간 객체-관계 데이터 모델을 정의하고 이 모델을 구현하기 위한 타입 저장 방법들에 대해 기술한다.

1. 서론

주변 공간에는 많은 지형객체들이 산재되어 있고 서로간의 인접, 중첩, 분리등의 공간적 위치 관계를 유지하고 있으며, 데이터베이스에서 공간적인 크기, 넓이, 위치, 방향등의 정보에 의해 기술된다. 이러한 공간 정보의 저장은 기존의 공간 데이터베이스와 지리정보 시스템 분야에서 연구되어 왔으며 현재 공간 데이터를 이용한 많은 응용 소프트웨어들이 제공되고 있다.

그러나 이러한 소프트웨어들이 단순한 형태의 질의만을 제공하고 있기 때문에 보다 복잡한 질의나 공간, 비공간 질의가 혼합된 형태의 질의 기능을 제공하지 못한다. 따라서 이와 같은 질의 기능을 사용자에게 제공하기 위해서는 공간 질의 처리 시스템이 요구된다.

이 논문에서는 이러한 공간 질의 처리 시스템을 지원하기 위해 공간 데이터 타입을 추상 데이터 타입으로 설계하여 객체-관계형 데이터베이스에 공간 데이터를 효율적으로 저장하기 위한 공간 객체-관계 데이터 모델을 정의한다.

논문의 구성은 다음과 같다. 먼저 2장에서는 객체-관계 데이터 모델을 설명하고 3장에서는 공간 데이터 모델을 설명한다. 4장에서는 데이터 모델의 설계 및 구현에 대해서 설명하고 5장에서는 공간 데이터 연산에 대하여 설명한다. 마지막으로 6장에서 결론을 맺는다.

2. 객체-관계 데이터 모델

2.1. 값타입

INTEGER, DECIMAL, NUMBER, CHAR와 같은 기본 데이터 타입 T1,..,Tn 있을 때 이 기본 타입들은 기본 타입 BT(Basic Type)라하고, 타입의 객체 식별자 noid를 갖는 타입을 객체 타입 OT(Object Type)라고 하자. 이 때 값타입VT(Value Type)는 BT와 OT의 집합이다.

타입 식별자들의 집합을 TypeID라고 하고, 타입 이름들의 집합을 TN이라고 할 때 값 타입은 다음과 같다.

typeid∈TypeID, tn∈TN, super-typeid∈TypeID 일 때,

VT = (typeid, tn, super-typeid)

예를 들면, CHAR, DECIMAL, INTEGER 등은 값 타입이며 이 모델에서 다음과 같다.

tid1, tid2, tid3 ∈ TypeID

CHAR = (tid1, "char", null)
DECIMAL = (tid2, "decimal", null)
INTEGER = (tid3, "integer", null)

값 타입이 위와 같이 정의될 때 값 타입들의 도메인 VTD(Value Type Domain)는 타입을 구별하기 위한 타입 식별자와 이 타입에 대한 제한자들의 리스트로 이루어 진다. 여기서 제한자는 제한어 k와 제한 특성 a로 구성되는 투플의 집합으로 구성된다.

typeid ∈ TypeID 일 때

VTD = (typeid, {(k1:a1), (k2:a2), ⋯ , (kn:an)})

예를 들면, NUMBER(10,2)와 CHAR(10), INTEGER의 도메인은 다음과 같은 형태를 갖는다.

CHAR(10) = (tid1, [size:10])
DECIMAL(10, 2) = (tid2, {(size:10), (precision:2)})
INTEGER = (tid3, {})
FLOAT = (tid4, {})
LISTOF(FLOAT) = (tid5, {(obj:tid4)})

위의 예에서 CHAR(10)은 크기 10의 제한특성을 갖는 CHAR 타입을 의미하며, DECIMAL(10, 2)는 크기 10에 소수점 이하 2자리를 갖는 DECIMAL 타입이다. INTEGER는 별도의 제한자를 갖지 않으므로 빈 제한자 리스트를 갖게 된다.

2.2. 추상 데이터 타입

여기서, 만약 D1,⋯ ,Dn이 값타입 도메인들이고, A1,⋯ An이 속성이름들이며, TypeID가 타입식별자들의 집합, TN이 타입이름들의 집합,

이 연구는 한국과학재단의 99년도 특정 연구과제의 지원으로 수행되었음.

Methods가 메소드들의 집합이라고 하면 추상 데이터 타입(Abstract Data Type) ADT는 타입 식별자, 타입 이름, 상위 타입의 타입 식별자, 타입의 속성 리스트 [A1:D1, - ,An:Dn], 타입의 메소드 집합으로 이루어진다.

tid∈TypeID, tn∈TN, super-typeid∈TypeID, methods∈Methods 일 때

ADT = (tid, tn, super-ntypeid, val, methods)

val = [A1:D1, - ,An:Dn]

즉, 모든 ADT타입들은 타입 ID와 타입이름, 슈퍼 타입의 ID, 속성들의 리스트, 그리고, 메소드들의 리스트를 갖는다.

행정기관에 대한 ADT을 선언하는 예를 들어보자. 행정기관들이 공통적인 이름, 기관장, 유형의 속성을 갖는다고 했을 때 행정기관 ADT는 다음과 같이 선언될 수 있다.

```
CREATE TYPE person (              CREATE TYPE 행정기관_t (
  이름       CHAR(10),               이름       CHAR(30),
  주소       CHAR(50),               주소       CHAR(50),
  근무년수   INTEGER                  기관장     Person
);                                );
```

위의 타입 선언문으로 생성된 행정기관_t은 다음과 같은 내부의미를 갖는다.

tid6∈TypeID 일 때,

(tid6, 'person', null, val1, null)

val1=[이름:(tid1,{(size:10)}),주소: (tid1,{(size:10)}), 근무년수: (tid3,{})]

(tid7, '행정기관_t', null, val2, null)

val2 =[이름: (tid1,{(size:30)}), 주소: (tid1,{(size:10)}), 기관장: (tid6,{cid:null})]

2차원 공간 데이터 타입을 ADT의 형태로 지원하며 다음과 같이 구성된다.

tid7, tid8 ∈ NTypeID

NSpatialType = (tid7, "spatial", null, null, null)

NPolygonType = (tid8, "polygon", tid7, val3, null)

val3 = [geo: (tid5, {(obj:tid4)})]

위의 예에서 공간 데이터 타입 Polygon은 타입 ID가 3이며 "polygon"이라는 타입 이름을 갖는다. 또한, 슈퍼 타입으로 타입 ID가 16인 NSpatialType을 가지며 별도의 속성과 메소드들은 갖지않는다. 앞의 행정기관_t 타입이 실제 예에서는 Polygon타입 속성 geo를 가지며 다음과 같이 정의된다.

```
CREATE TYPE 행정기관_t (
                    이름            CHAR(30),
                    주소            CHAR(50),
                    유형            INTEGER,
                    geo             Polygon
        );
```

이 때, 내부의미는 다음과 같다.

(tid8, '행정기관_t', null, val2, null)

val4=[이름:(tid1,{(size:30)}),주소:(tid1,{(size:10)}),유형:(tid3,{}),geo:(tid7,{cid:null})]

2.3. 클래스

클래스는 타입의 실제화된 모습이며, 클래스 이름의 집합을 CN이라고 하고, 클래스 식별자의 집합을 ClassID, 타입식별자의 집합을 TypeID, 제약사항들의 집합을 Const라고 할때 클래스는 다음과 같은 튜플로 정의된다.

클래스가 C이고, cid, super_cid∈ClassID, cn∈CN, tid∈TypeID, const∈Const일 때,

C = (cid, cn, tid, super_cid, val, const)

여기서, cid는 이 클래스의 식별자이며, cn은 클래스이름 집합 CN의 원소이고, val은 [A1:D1, - ,An:Dn]로 이루어진 클래스 구조이며, tid는 타입id이고, super_cid는 이 클래스의 슈퍼 클래스의 식별자, const는 클래스의 인스턴스들에 대한 제약에 대한 정보들(constraint_name, constraint)의 집합이다.

예를들면 앞에서 선언된 행정기관_t를 사용하여 대전행정기관 클래스를 생성한다면 다음과 같다.

CREATE TABLE 대전행정기관 OF 행정기관;

이 때, 대전행정기관의 내부 의미는 다음과 같이 생성된다.

cid1, cid2∈ClassID, tid6∈TypeID 일때,

(cid1, '대전행정기관', tid8, null, val2, null)

val5=[이름:(tid1,{(size:30)}),주소:(tid1,{(size:10)}),유형:(tid5,{}), geo:(tid7,{cid:cid2})]

(cid2, '대전행정기관#GEO', tid7, val6, null, null)

val6 = [geo: (tid5, {(obj:tid4)})]

2.4. 객체

이 모델의 객체 식별자들의 집합 NOID는 클래스 이름 식별자 cn과 클래스내의 식별자 id로 구성되며 데이터베이스 DB내에서 유일하다. 이 때 객체들의 집합 NObject는 객체 식별자 noid와 값val을 가진다.

nobj∈NObject, noid∈NOID일 때,

noid = (cn, id)

nobj = (noid, val)

여기서, cn은 클래스의 이름이며,id는 클래스에서 객체의 식별자이고,

val은 attribute의 이름 an과 값타입 VT의 실제 값 value의연속값 즉, [an1:value1, an2:value2, - , ann:valuen]이다.

예) 대전행정기관 클래스에서 '동남세무소' 객체는 다음과 같이 정의 된다.

noid1,noid2∈NOID, nobj1, nobj2∈NObject 일때,

noid1=('대전행정기관',4)이고,

nobj1=(noid1, [이름:'동남세무소', 주소:'대전', 유형:1, geo:noid2])

nobj2=(noid2,[geo:(238787.02,317515.27,- 238787.02,317515.27)])

가 된다.

3. 공간 데이터 모델과 연산

객체-관계형 데이터베이스에서 구현된 공간 데이터 모델은 공간 데이터 타입들의 집합과 그 타입들에 대한 연산들을 포함하고 있다. 지난 십 여년간 공간 추상화 데이터 타입(spatial abstract data type: ADT)에 대한 설계와 그 데이터 타입들을 포함하는 질의어에 연구가 상당수 진행되어 왔으며, 현재 이러한 데이터 타입에 대한 표준화 노력들이 이루어지는 중이다.

이 논문에서 지원하는 공간 데이터 타입은 Spatial, Point, LineString, Polygon등의 단일 데이터 타입이며 다중 데이터 타입인 Geometry-Collection들은 단일 데이터 타입의 set-of, list-of를 사용한다.

공간 연산자는 크게 기하 연산자(geometric operator)와 위상 관계 연산자(topological operator), 그리고 공간 집계 함수(spatial aggregate function)등으로 구분된다. 기하 연산자는 하나의 공간 객체만을 이용하여 객체의 기하학적인 값을 얻어내거나 또는 두 공간 객체간의 거리와 같은 값을 얻어내는데 사용되는 연산자로 산술적인 수치를 반환한다. 위상 관계 연산자는 위상 데이터, 즉 공간 객체들간 관계의 참, 거짓 여부를 판단하는 연산자들로 연산을 만족하면 true를 만족하지 않으면 false를 반환한다. 제안하는 공간 SQL질의에서 연산자는 OpenGIS 연산자를 확장하여 구현하고 연산자중 기하 연산자는 *function*으로 맵핑되며, 위상관계연산자는 WHERE절에서 속성들간의 이항연산자로서 사용된다. 아래의 표 1.과 표 2.는 각각 기하 연산자와 위상관계연산자의 매핑을 보여준다.

표 1. 기하 연산자의 공간 SQL 매핑

OepnGIS	Spatial SQL
buffer	buffer function
difference	
union	
intersection	
symmetric difference	
-	shortestpath function
convexhull	

표 2. 위상관계 연산자의 공간 SQL 매핑

OpenGIS	Spatial SQL
equal	equal
disjoint	disjoint
touch	touch
within	within
overlap	overlap
cross	cross
intersects	intersects
contains	contains
	connect

4. 타입 및 클래스의 설계 구현

4.1. 타입의 설계 구현

그림 1.은 공간 데이터 타입의 구현을 위한 상속 구조를 보여주고 있다. NUDT타입은 사용자 정의 데이터 타입(User Defined Type)을 의미하며 메소드를 지원한다. RowtableType은 일반적인 단일 테이블에 연산을 수행하기 위한 데이터 타입이다. 2차원 공간 데이터 타입 PointType, LineStringType, PolygonType과 3차원 공간 데이터 타입 SRoadfaceType, SBuildBoxType, SBuildExtType, SBuildCylType, SPipeExtType, SBuildConType은 RowtableType을 상속하여 구현된다. 즉, 모든 공간 데이터 타입 속성은 단일 테이블에 저장관리된다.

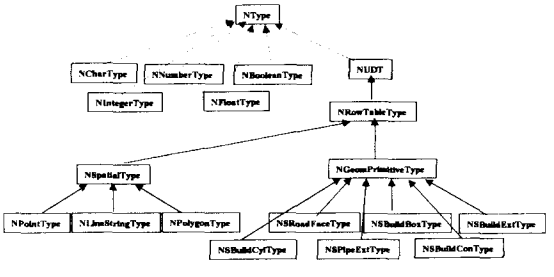

그림 2. 타입 구현의 상속 구조

4.2. 클래스의 설계 구현

클래스는 타입의 실제화된 모습이며 그림 2.는 앞 절의 공간 데이터 모델 예제로 사용되었던 대전행정기관 클래스의 실제 구현 자료구조의 모습이다. 예에서 geo속성은 Polygon타입으로 선언되었다. Polygon 데이터 타입은 Rowtable 타입을 상속한 서브 타입으로 별도의 클래스를 갖는다. 이 데이터 타입의 constraint는 실제 Polygon 타입을 구현한 클래스의 cid값을 갖는다. 클래스 관리자는 클래스 속성에 대한 참조를 요구할 경우, 각 속성의 타입에 따라 클래스의 정보를 찾게 된다.

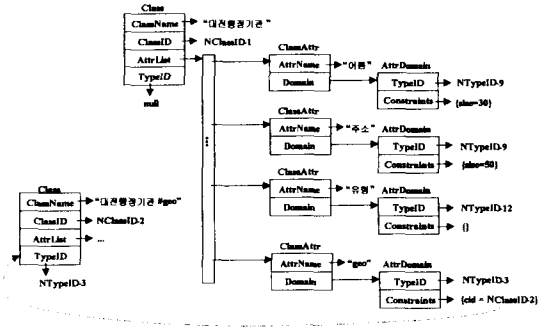

그림 3. 클래스 구성의 예

5. 결론

이 논문에서는 공간 사용자 질의에 유연성을 제공하고 공간 정보를 검색할 수 있는 기능을 제공해 줄 수 있는 공간 객체-관계 모델을 정의하였으며 이 모델을 구현하기 위한 타입저장 방법들에 대하여 기술하였다. 또한, 바탕이 되는 공간 데이터 타입 모델과 공간 연산자에 대하여 서술하였다.

이 논문에서 정의한 공간 객체-관계 데이터 모델은 테이블 식별자와 튜플 식별자의 쌍을 객체 식별을 위한 oid로 사용함으로써 관계형 튜플을 객체형으로 쉽게 매핑할 수 있도록 하였다. 또한 SQL3에서 지원하는 ARRAY와 같은 컬렉션 타입과 Distinct Type, Abstract Data Type, 상속성들을 관계형 테이블상에서 저장하기 위한 구현 모델을 제시하였다.

6. 참고문헌

[Falo89] C. Faloutsos, W. Rego, "Tri-Cell: A Data Structure for Spatial Objects," Information systems, Vo. 14, No. 2, pp. 131-139, 1989

[Guti94] R.H.Guting, "An Introduction to Spatial Database Systems". VLDB Journal, Vol. 3, No. 4, pp. 357-399, 1994

[Kim95] Won Kim, Modern Database Systems: The Object Model, Interoperability and Beyound, Addition-Wesley Publising Company, pp.338-360, 1995.

[Laru93] T. Larue, D. Pastre, Y. Viemont, "Strong Integration of Spatial Domains and Operators in a Relational Database System," SSD, pp. 53-72, 1993.

[임헌기00] 임헌기, 남광우, 김영삼, 김록원, 류근호, "iSTORM : 인터넷 지향적 공간 객체-관계 데이터베이스 시스템", 춘계 학술발표논문집, 한국정보과학회, 제27권, 2000.

[김영삼00] 김영삼, 임헌기, 남광우, 류근호, "iSTORM에서의 2D/3D 공간 질의어 및 질의 최적화 기법", 춘계 학술발표논문집, 한국정보과학회, 제 27권, 2000