The Journal of GIS Association of Korea, Vol. 13, No. 2, pp. 119-128, July 2005

3차원 기하모델에 대한 공간 관계 연산 설계

이동헌* · 홍성언** · 박수홍***

Design of Spatial Relationship for 3D Geometry Model

Dong-Heon Yi* · Sung-Eon Hong** · Soo-Hong Park***

요 약

GIS 분야에서 다루는 공간 데이터는 대부분 2차원의 데이터이다. 현실 공간에 존재하는 3차원 객체의 2차원 정보만을 취하거나 혹은 2차원 공간으로 투영하는 등의 방법으로 데이터를 저장한다. 이러한 방법은 정보의 손실로 인한 데이터 활용범위가 축소되고, 현실 공간을 정확하게 반영하지 못하는 문제가 있다. 최근 3차원 공간 데이터를 저장, 관리 가능한 DBMS가 개발되고, 3차원 데이터에 대한 관심과 요구가 높아지고 있다. 하지만 이들은 단순히 3차원 공간의 데이터를 저장만 가능할 뿐 공간 데이터베이스 관리 시스템의 핵심이라 할 수 있는 공간 질의가 불가능하다. 또한 이에 대한 연구가 미흡한 실정이다.

본 연구에서는 3차원 공간 모델을 이용하여 공간 데이터베이스 표준에서 정의하고 있는 공간 관계 연산을 설계하였다. 공간 데이터 모델로는 OGC에서 제시한 GML3에서 정의하는 모델을 사용하였고, 공간 관계 연산에 대한 설계 도구로는 공간 관계를 연산하는데 가장 좋은 방법으로 알려진 Point-Set Topology 기반의 DE-9IM을 이용하였다.

주요어 : 공간 데이터베이스, 공간 관계 연산, 3차원, DE-9IM(Dimensionally Extended 9 Intersection Method)7

ABSTRACT : Most spatial data handled in GIS is two-dimensional. These two-dimensional data is established by selecting 2D aspects from 3D, or by projecting 3D onto 2D space. During this conversion, without user's intention, data are abstracted and omitted. This unwanted

*인하대학교 지리정보공학과 석사과정
**인하대학교 지리정보공학과 박사과정
***인하대학교 지리정보공학과 조교수

data loss causes disadvantages such as restrictingof the range of data application and describing inaccurate real world. Recently, three dimensional data is getting wide interests and demands. One of the examplesis Database Management System which can store and manage three dimensional spatial data. However, this DBMS does not support spatial query which is the essence of the database management system. So, various studies are needed in this field.

This research designs spatial relationship that is defined in space database standard using the three-dimension space model. The spatial data model, which is used in this research, is the one defined in OGC for GMS3, and designing tool is DE-9IM based on Point-Set Topology know as the best method for topological operation.

Keywords : Spatial Database, Spatial Relationship, 3Dimension, DE-9IM (Dimensionally Extended 9 Intersection Method)

1. 서 론

1.1 연구배경과 목적

공간 데이터베이스 기술은 방대한 양의 지리정보 데이터의 효율적인 저장 관리 처리를 위하여 개발 발전되고 있다. 최근까지의 공간 데이터베이스 관리 시스템에서는 실세계의 공간 객체를 저장할 때 2차원의 공간정보만을 저장하는 방법을 사용하고 있다. 하지만 실세계에 존재하는 객체는 높이 값을 갖는 3차원 객체이다. 따라서 공간 데이터베이스 관리 시스템에 객체를 저장 할 때 정보의 손실이 불가피해지고 높이 값을 이용한 처리가 불가능해 진다. 최근 GIS 분야에서도 3차원 정보를 이용하려는 연구가 진행되고 이에 따라 3차원에 대한 관심이 높아지고 있다. 하지만 3차원 공간 모델에 관한 연구와 이에 적합한 공간 연산자에 관한 연구가 미비한 실정이다. 따라서 본 연구에서는 공간데이터베이스의 핵심적인 부분이라 할 수 있는 공간 연산자 중 객체 사이의 관계를 판단할 수 있는 공간 관계 연산자를 3차원 기하 모델에 적용될 수 있도록 설계 제안 하였다. 설계는 다음과 같은 세부 목적을 만족 시키고자 한다.

기존의 공간 데이터베이스 표준(David Beddoe · Paul Cotton, 1999)의 2차원 데이터 모델에서도 적용 가능한 설계가 되어야 한다. 이는 3차원 공간상에 존재 할 수 있는 2차원 이하의 객체에도 적용 가능하도록 하기 위함 이다. 또한 OpenGIS Consortium (이하 OGC)표준을 준수하는 3차원 데이터 모델을 기반으로 설계 되어야 한다. 이는 산업 표준으로 인정되는 OGC의 표준을 준수함으로서 상호 운용성을 보장하기 위함 이다. 그리고 기존의 공간데이터베이스 표준(David Beddoe · Paul Cotton, 1999)에서 제시된 공간 관계 연산의 정의를 만족하는 설계가 필요하다. 마지막으로 사용자 관점에서 최소한의 연산만으로 결과를 나타낼 수 있도록 설계 되어야 한다.

1.2 관련 연구

현재까지 3차원 공간 관계 연산자에 관한 연구는 다양하지 못하다. 대표적인 연구로 3차원 공간 위상 관계 연산자의 설계(김상호 · 강구, 2003)가 있는데 이 연구에서는 공간 데이터베이스를 위한 자체적인 3차원 공간 데이터 모델을 제안 하였고, 그에 맞는 3차원 공간 연산자를 설계하였다. 하지만 이 연구에서는 3차원 공간 위상 연산자 설계에 있어서 3차원 공간상에 존재하는 2차원 이하 객체들의 위상 관계를 표현하는데 문제가 있다. 또한 산업 표준인 OGC표준(Simon Cox · Paul Daisey, 2003)에서 제시한 기하 모델과는 상이한 데이터 모델을 사용하여 연산자를 설계하였다. 상용 DBMS 제품 중 최근에 출시된 버전에서는 대부분 3차원 정보를 저장할 수 있도록 확장되어있다. 그러나 이들은 3차원 데이터 모델이 아닌 2차원 모델에 높이 값만을 추가한 정도이고 공간 위상 관계 역시 3차원에 대해서는 지원하지 못 하는 단순히 공간 객체를 저장하는 수준에 머무른다.

2.3차원 데이터 모델의 선정

실세계의 다양한 공간 객체를 처리하기 위해서는 OGC에서 1999년에 발표된 공간 데이터베이스의 표준인 Simple Features Specification For SQL1.1(David Beddoe · Paul Cotton,1999)에서 제시하는 2차원의 공간 데이터 모델만으로는 부족하다. 하지만 OGC에서는 더 이상의 공간 데이터베이스 표준이 제시 되지 않았다. 따라서 본 연구에서는 GIS 데이터 분야의 표준으로 제시된 GML3(Simon Cox · Paul Daisey · Ron, 2003)의 여러 데이터 모델 중 3차원 공간객체에 대한 정의를 갖는 공간 기하 모델을 사용하여 공간 연산자를 설계하였다. 최근에는 XML을 기반으로 하는 GML을 공간 데이터베이스에 저장하려는 연구가 활발히 진행 중 이다(정호영 · 이민우, 2003). 본 연구에서는 GML을 저장할 수 있는 공간 DBMS에 적용 가능하도록, 즉 GML의 데이터 모델을 이용하여 공간 관계 연산을 설계하였다.

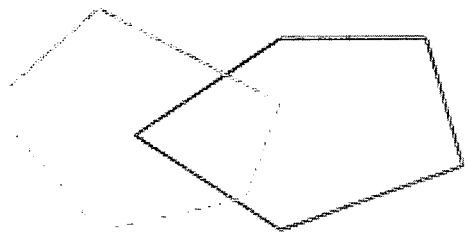
3. 공간 관계의 표현 기법

공간 데이터베이스가 지원하는 공간 연산자로 Set-oriented, Topological, Directional, Metric 과 같은 네 가지 종류의 연산자를 정의하고 있다(Shashi Shekhar · Sanjay Chawla, 2002). Set-oriented 연산자는 집합관계에 기반을 둔 연산으로 Union, Intersection, Difference 등이 있다. Directional연산자는 geometry의 방향성에 기반을 둔 연산자 이다. 방향 관계를 나타내는 데에는 기준이 필요한데 절대적 기준, 특정 객체 기준, 혹은 사용자관점을 기준으로 하여 방향성을 결정하는 연산자이다. Metric 연산자는 Euclidean 공간 내에서 실수로 표현된 좌표를 이용한 연산을 의미한다. 마지막으로 Topological 연산자는 위상 공간에 존재하는 두 개의 기하 객체 사이에 존재하는 관계를 판단하는 연산자이다. 공간 위상 관계를 판단하는 데에는 몇 가지 다른 방법들이 제시 되어 왔다. 가장 널리 사용되는 방법으로는 Point-set

topology(Max Egenhofer · Robert Franzosa, 1991) 방식이 있다. Point-set topology 방식에는 공간 객체의 내부와 경계의 만나는지 여부를 이용한 4IM(Max Egenhofer · Robert Franzosa, 1991; Max Egenhofer · Robert Franzosa, 1995), 내부와 경계에 외부 개념을 도입한 9IM(Max Egenhofer, 1994), 여기에 만나는 공간의 차원수를 이용한 DEM(Eliseo Clementini · Paolino Di Felice, 1993), DE-9IM(Eliseo Clementini · Paolino Di Felice, 1994) 등이 있다. 이와 같은 방법들은 각각 표현할 수 있는 공간 위상 관계의 범위가 다르고 표현 방법도 다르다. 이들 중 객체사이의 위상 관계를 가장 다양하게 표현할 수 있는 방법은 DEM과 9IM을 결합하여 만든 DE-9IM이다(Eliseo Clementini · Paolino Di Felice, 1994). DE-9IM은 두 객체의 내부 경계 외부가 각각 만나는 영역의 차원수를 아래와 같은 행렬식으로 작성하는 것으로써 두 객체의 위상 관계를 표현하는 방법이다.

<표 1> DE-9IM

	Interior	Boundary	Exterior
Interior	*dim(I(a) ∩I(b))*	*dim(I(a) ∩I(b))*	*dim(I(a) ∩I(b))*
Boundary	*dim(I(a) ∩I(b))*	*dim(I(a) ∩I(b))*	*dim(I(a) ∩I(b))*
Exterior	*dim(I(a) ∩I(b))*	*dim(I(a) ∩I(b))*	*dim(I(a) ∩I(b))*



[그림 1] 공간 관계 : 예) Area/Area

<표 2> DE-9IM : 예) Area/Area

	Interior	Boundary	Exterior
Interior	2	1	2
Boundary	1	0	1
Exterior	2	1	2

OGC의 공간 데이터베이스 표준(David Beddoe · Paul Cotton, 1999)에서는 두 기하 객체의 공간관계를 위의 DE-9IM을 이용하여 표현 하도록 하고 이렇게 표현된 값과 정의된 patternMatrix와의 비교를 통하여 공간 관계를 판단한다. 따라서 본 연구에서도 가장 다양한 표현이 가능하고, OGC의 표준 위산관계 표현기법으로 사용되는 DE-9IM을 이용하여 3차원 공간상에 존재 하는 3차원 이하 객체에 대하여 공간 관계를 설계하고자 한다.

4.3차원 공간 관계 연산의 설계

4.1 연구 대상 공간

연구 대상은 3차원 공간으로 하였다. 공간 Feature type으로는 위치를 표현할 수 있는 P(point), 길이를 갖는 닫혀있고 연속적인 1차원의 점 집합인 L(line), 넓이를 갖는 닫혀있고 연속적인 2차원의 점 집합 A(area)를 이용한다(Eliseo Clementini · Paolino Di Felice, 1994). 3차원에 대한 Feature type으로는 부피를 가지며 닫혀있고 연속인 3차원의 점 집합인 S(solid)를 이용한다. 이들은 GML 데이터 중 기하 모델을 공간 데이터베이스와의 매핑 규칙에 맞게 변환된 Point, LineString, Polygon, Solid 를 의미한다. 또한 GML 기하 모델에서 수용하고

있는 complex 한 기하 객체의 경우를 제외 하였다. 기하 객체가 complex한 경우, 즉 Self-intersect가 존재하다면 객체간의 공간 관계를 DE-9IM으로 표현하는 데에 중요한 경계와 내부와의 구분이 모호해지기 때문이다. GML 기하 모델에서는 _GeometricAggregate 로 표현되는 기존 공간 데이터베이스 표준에서는 Multi type 역시 수용하고 있다. 같은 속성을 갖는 다수의 공간객체들 사이에서도 공간 관계가 정의 되어야 한다. 따라서 연구에서는 MP, ML, MA, MS에서도 연산이 가능 하도록 각각 P, L, A, S와 동일한 Feature로 사용 하여 연산자를 설계하였다. 즉 연구에 사용될 Feature type은 complex를 제외한 기하 객체의 최대 차원 수에 따라 정의 된 P, L, A, S를 사용한다.

4.2 Notation 정의

<표 3>은 본 연구의 전반에 사용될 기호들을 요약 정의한 것이다. ∂와 I()는 객체의 내부, ∘와 B()는 객체의 경계, -와 E()는 각각 객체의 외부를 의미한다. dim()은 객체의 차원수를 반환하는 함수로서, 인자로는 객체 자신이나 객체들의 만나는 영역으로 이루어진 새로운 객체가 올 수 있다. ∩는 두 객체가 만나는, 즉 교집합이 되는 영역을 반환한다. p는 두 객체의 내부, 경계, 외부를 이용하여 만나는 영역을 행렬로 작성하였을 때 각각의 행렬 값을 나타낸다. 마지막으로 relate()는 두 객체의 Intersection Matrix와 주어진 pattern Matrix가 일치하는가를 판단하여 반환하는 함수이다.

<표 3> Notation 정의

Symbol	Definition
∂ I()	Interior
∘, B()	Boundary
-, E()	Exterior
dim()	Dimension of Object
∩	Intersection`
p	pattern value
max()	maximum value
relate()	Return T if the spatial relationship specified by the patternMatrix holds

4.3 Spatial Relationship 정의

OGC의 표준(David Beddoe · Paul Cotton, 1999)에서 정의하고 있는 공간 관계 연산으로는 <표 4>와 같은 여덟 가지가 있다. 이들 중 equals를 제외한 나머지 일곱 가지 연산은 위상 관계 연산이다. equals 연산의 위상 관계연산은 아니지만 DE-9IM을 이용하여 표현이 가능하다.

<표 4> 공간 관계 연산 정의

Operation	Definition
disjoint	$a \cap b = \emptyset$
touches	$(I(a) \cap I(b) = \emptyset) \wedge (a \cap b \neq \emptyset)$
within	$(a \cap b = a) \wedge (I(a) \cap I(b) \neq \emptyset)$
contains	$(a \cap b = b) \wedge (I(a) \cap I(b) \neq \emptyset)$
overlaps	$(\dim(I(a)) = \dim(I(b)) = \dim(I(a) \cap I(b))) \wedge (a \cap b \neq a) \wedge (a \cap b \neq b)$
crosses	$(\dim(I(a) \cap I(b)) < \max(\dim(I(a)), \dim(I(b))) \wedge (a \cap b \neq a) \wedge (a \cap b \neq b)$
intersects	$a \cap b \neq \emptyset$
Equals	spatially equals

4.4 3차원 공간 관계 연산 설계

각 feature type 별로 내부와 경계 그리고 외부를 정의하면 다음과 같이 경계를 정의하고 각 feature에서 경계를 제외한 부분을 내부, 전체 공간 영역에서 feature를 제외한 부분을 외부로 정의한다<표 5>. 연구에서 사용할 S에 대한 세 가지 공간 영역역시 같은 방법으로 정의한다.

<표 5> Feature의 I, B, E 정의
(Eliseo Clementini · Paolino Di Felice, 1994)

	Boundary	Interior	Exterior
P	EMPTY	$P-\partial P$	R^3-P
L	end points of L	$L-\partial L$	R^3-L
A	closed curve	$A-\partial A$	R^3-A

경계는 $\partial S = closed\,surface$로, 내부는 $S^{\circ} = S-\partial S$, 그리고 외부는 R^3-S 로 정의 하였다. 두 feature 사이에서 나타날 수 있는 행렬값은 9개의 pattern value로 구성되는데 3차원 공간에서 나타날 수 있는 pattern value p는 {T, F, *, 0, 1, 2, 3}이 된다. 각각의 값이 의미하는 바는 아래와 같다(김상호 · 강구, 2003).

p = T → dim(x) ∈ {0, 1, 2, 3} i.e. $x \neq \emptyset$
p = F → dim(x) ∈ -1 i.e. $x = \emptyset$
p = * → dim(x) ∈ {T, F, 0, 1, 2, 3}
i.e. Don't Care
p = 0 → dim(x) ∈ 0
p = 1 → dim(x) ∈ 1
p = 2 → dim(x) ∈ 2
p = 3 → dim(x) ∈ 3

각 pattern value p가 가질 수 있는 값의 범위는 위와 같이 7가지 경우이다. 하지만 실제 feature들 사이에서 나타날 수 있는 공통 영역의 차원 수는 만나지 않는 경우를 포함한 0, 1, 2, 3차원의 5가지가 존재한다. 위의 정의들을 이용하여 3차원 feature에 대한 공간 관계 연산자를 DE-9IM 기반의 pattern matrix로 설계하였다. 먼저 disjoint연산의 경우 주어진 두 객체 a,b에 대하여 정의 $a.disjoint(b) \Leftrightarrow a \cap b = \emptyset$를 DE-9IM 기반의 Matrix 형태로 표현하면 다음과 같은 결과를 얻는다.

$$a.disjoint(b) \Leftrightarrow (I(a)\cap I(b)=\emptyset)\wedge (I(a)\cap B(b)=\emptyset)\wedge (B(a)\cap I(b)=\emptyset)\wedge (B(a)\cap B(b)=\emptyset)$$
$$\Leftrightarrow \text{a.Relate(b,'FF*FF****')}$$

즉 a와 b의 두 객체가 disjoint 인지 판단하는 것은 두 객체의 내부, 경계, 외부 사이 만나는 영역이 몇 차원인지 연산하여 matrix 형태로 표현 되었을 때 위에서 제시한 pattern과 일치하게 되면 두 객체의 공간 관계는 disjoint 하다고 말한다. 나머지 일곱 가지 공간 연산들도 이와 같은 방법으로 설계가 가능하다. disjoint 연산의 경우에는 2차원에서의 pattern과 같은 결과를 보여주며 feature type P에 대해서는 경계부분의 연산을 제외하면 가능하다. intersect연산에서는 disjoint연산과 정의가 상반되고 있다. 따라서 모든 경우에 연산이 가능하며 다음과 같은 관계가 성립한다.

$$\text{a.intersects(b)} \Leftrightarrow !\, a.disjoint(b)$$

touches 연산에서는 같은 방법으로 정의 a.touches(b) $\Leftrightarrow (I(a)\cap I(b)=\emptyset)\wedge(a\cap b\neq\emptyset)$ 를 DE-9IM 기반의 pattern matrix 를 도출해 내면 P/P 그룹을 제외하고 나머지의 경우에서 다음과 같이 가능하다.

$$
\begin{aligned}
a.touches(\mathbf{b}) \quad & \Leftrightarrow (I(a)\cap I(b)=\emptyset)\wedge \\
& ((B(a)\cap I(b)\neq\emptyset)\vee \\
& ((I(a)\cap B(b)\neq\emptyset)\vee \\
& ((B(a)\cap B(b)\neq\emptyset)) \\
& \Leftrightarrow \text{a.Relate(b,'FT*******')}\vee \\
& \quad \text{a.Relate(b,'F**T*****')}\vee \\
& \quad \text{a.Relate(b,'F***T****')}
\end{aligned}
$$

touches 연산은 두 개의 객체 중 하나 이상이 경계가 존재하는 type의 feature이어야 한다. 따라서 P/P 그룹을 제외하면 다른 모든 경우에서 2차원에서와 같은 pattern을 보여준다. within 연산은 contains 연산과 함께 두 객체의 포함 관계에 대한 연산 이다. 따라서 두 연산 모두에서 포함 되는 feature type의 차원수가 포함하는 feature type의 차원수보다 같거나 작아야 한다. within 연산에서도 정의 마찬가지로 contains 연산은 다음과 같은 pattern을 갖는다.

$a.within(b) \Leftrightarrow (a\cap b=a)\wedge(I(a)\cap I(b)\neq\emptyset)$ 를 DE-9IM의 pattern으로 표현하면 다음과 같다.

$$
\begin{aligned}
a.within(b) & \Leftrightarrow (I(a)\cap I(b)\neq\emptyset)\wedge \\
& \quad (I(a)\cap E(b)=\emptyset)\wedge \\
& \quad (B(a)\cap E(b)=\emptyset)) \\
& \Leftrightarrow \text{a.Relate(b,'T*F**F***')}
\end{aligned}
$$

$$
\begin{aligned}
a.contains(b) & \Leftrightarrow b.within(a) \\
& \Leftrightarrow (a\cap b=b)\wedge(I(a)\cap I(b)\neq\emptyset) \\
& \Leftrightarrow \text{a.Relate(b,'T*****FF*')}
\end{aligned}
$$

within연산과 contains연산에서도 2차원에서와 같은 pattern을 보인다. overlaps 연산에서는 정의에서 단순히 만나는지 만을 판단하는 것이 아닌 만나는 영역의 차원까지 고려하고 있다. 따라서 2차원 공간에서의 pattern 과 다른 결과를 보이게 된다.

$a.overlaps(b)\Leftrightarrow(\dim(I(a))=\dim(I(b))=\dim(I(a)\cap(I(b)))\wedge(a\cap b\neq a)\wedge(a\cap b\neq b)$ 우선 두 객체의 내부에 대한 차원이 같아야 한다는 것은 두 객체가 같은 feature type 을 가져야 한다는 의미이다. 따라서 P/P, L/L, A/A, S/S 그룹 사이에서만 정의가 가능하다. 또한 두 객체 내부가 만나는 영역의 차원역시 객체 내부의 차원과 같아야 한다. 마지막으로 어느 하나의 객체에 다른 쪽이 포함되면 안 된다. 이를 만족하는 pattern을 표현하면 다음과 같다.

$$
\begin{aligned}
a.overlaps(b) \quad & \Leftrightarrow (I(a)\cap I(b)\neq\emptyset)\wedge \\
& \quad (I(a)\cap E(b)\neq\emptyset)\wedge \\
& \quad (E(a)\cap I(b)\neq\emptyset) \\
& \Leftrightarrow \text{a.Relate(b,'T*T***T**')} \\
& \quad \text{if P/P, S/S case} \\
& \Leftrightarrow (I(a)\cap I(b)\neq 1)\wedge \\
& \quad (I(a)\cap E(b)\neq\emptyset)\wedge \\
& \quad (E(a)\cap I(b)\neq\emptyset) \\
& \Leftrightarrow \text{a.Relate(b,'1*T***T**')} \\
& \quad \text{if L/L case} \\
& \Leftrightarrow (I(a)\cap I(b)\neq 2)\wedge \\
& \quad (I(a)\cap E(b)\neq\emptyset)\wedge \\
& \quad (E(a)\cap I(b)\neq\emptyset) \\
& \Leftrightarrow \text{a.Relate(b,'2*T***T**')} \\
& \quad \text{if A/A case}
\end{aligned}
$$

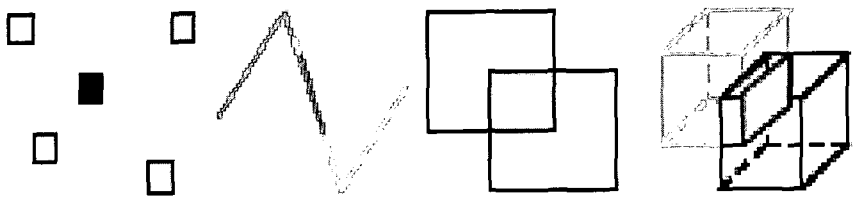
각각 feature type별로 다른 결과를 보인다. P/P 그룹의 경우에는 두 객체 모두 single point일 경우에는 내부끼리 만나면서 포함되지 않아야 한다는 정의에 의해

서 overlaps관계가 성립되지 않는다. 모두 multipoint 일 경우에만 해당이 된다. S/S 그룹에서는 feature type의 차원수가 대상 공간의 차수와 같은 경우이므로 내부 사이에 만나는 영역의 pattern value가 'T'이면 항상 차수로는 3이 된다. L/L, A/A 그룹에서는 feature type의 차원수보다 높은 공간 영역에 존재하므로 내부 간에 만나는 영역의 pattern value가 'T' 일 경우에도 다른 결과가 나타날 수 있다. 따라서 내부 사이의 만나는 영역에 대한 pattern value를 명확하게 명시 해 주어야 정확한 결과를 얻을 수 있다. [그림 2]에서는 overlaps연산이 참의 값을 갖게 되는 경우를 나타내었다. overlaps연산과 마찬가지로 crosses연산에서도 2차원과는 다른 pattern을 갖는다. crosses연산의 정의 에맞게 DE-9IM 기반의 pattern matrix를 설계하였다. *a.crosses*(b)⇔ $(\dim(I(a)\cap(I(b))<\max(\dim(I(a)),\dim(I(b)))\wedge(a\cap b\neq a)\wedge)(a\cap b\neq b)$

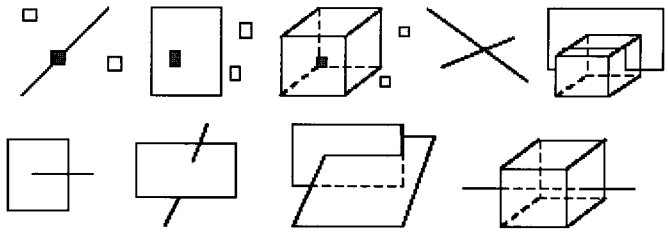
crosses 연산은 P/P, S/S 그룹에서는 나타나지 않는다. P/P 그룹에서는 두 객체 내부 사이의 만나는 영역이 0차원 미만이 되어야 하는데 0차원 미만은 불가능하기 때문에 나타나지 않고, S/S그룹에서는 내부사이의 만나는 영역이 3차원 이하가 되어야 하는데 pattern value 가 'T'일 경우에는 항상 3이 되므로 나타나지 않는다.

$$
\begin{aligned}
a.crosses(b) &\Leftrightarrow \dim(I(a)\cap I(b))=0 \\
&\quad a.Relate(b,'0********') \\
&\quad \text{if L/L case} \\
&\Leftrightarrow \dim(I(a)\cap I(b))=1 \\
&\quad a.Relate(b,'1********') \\
&\quad \text{if A/A case} \\
&\Leftrightarrow(\dim(I(a)\cap I(b))=1)\wedge(I(a)\cap E(b)\neq\emptyset) \\
&\quad a.Relate(b,'1*T******') \\
&\quad \text{if A/L case} \\
&\Leftrightarrow \dim(I(a)\cap I(b))=0 \\
&\quad a.Relate(b,'0********') \\
&\quad \text{if A/L case} \\
&\Leftrightarrow (I(a)\cap I(b)\neq\emptyset)\wedge(I(a)\cap E(b)\neq\emptyset) \\
&\quad a.Relate(b,'T*T******') \quad \text{else}
\end{aligned}
$$

P/L, P/A, P/S 그룹에서의 P는 overlaps 연산에서와 같은 이유로 인하여 multipoint 일 경우에만 해당한다. L/L 그룹에서는 2차원에서의 정의와 같지만 A/A 그룹에서는 2차원에서 존재하지 않았던 crosses 관계가 발생한다. 내부 사이에서 만나는 영역의 차원이 1과 2를 가질 수 있는데, 정의에 의해서 2는 탈락하게 되므로 pattern matrix 에서는 1로 명시해 주어야 한다. A/L 그룹에서는 두 가지의 pattern이 존재하는데, 이는 a.Relate(b,'T*T******')으로 표현할 수 있지만 불필요한 연산을 줄이고자하여 분리하였다. [그림 3]은 crosses 관



[그림 2] overlaps 관계



[그림 3] crosses 관계

계가 성립하는 경우를 나타내었다.

equals 연산은 위상 관계연산이라고 볼 수 없다. 위상적으로 동일한지를 판단 하는것이 아닌 공간적으로 동일한지를 판단하는 연산이다. 하지만 DE-9IM을 이용한 연산이 가능하다. equals 연산은 contains 연산과 within 연산을 이용하여 정의 할 수 있다. 두 객체의 관계가 contains관계인 동시에 within관계가 성립한다면 공간적으로 동일하다고 볼 수 있다. 따라서 equals 연산에 대한 pattern을 다음과 같이 설계하였다. 모든 경우에 대하여 feature type이 같다는 즉 dim(a)=dim(b) 전제가 포함된다.

$$a.equals(b) \Leftrightarrow (\dim(a)=\dim(b)) \wedge (I(a)\cap I(b)\neq\varnothing) \wedge (I(a)\cap E(b)=\varnothing) \wedge (E(a)\cap I(b)=\varnothing)$$

$\Leftrightarrow$ a.Relate(b,'T*F***F**')

if P/P case

$$\Leftrightarrow \dim(a)=\dim(b) \wedge (I(a)\cap I(b)\neq\varnothing) \wedge (I(a)\cap E(b)=\varnothing) \wedge \Leftrightarrow (B(a)\cap E(b)=\varnothing) \wedge (E(a)\cap I(b)=\varnothing) \wedge (E(a)\cap B(b)=\varnothing)$$

$\Leftrightarrow$ a.Relate(b,'T*F**FFF*') else

P/P 그룹에서는 경계가 되는 영역이 없다. 그리고 multi point의 경우를 고려하여 외부까지를 포함한 pattern을 설계하였다. 나머지 그룹에 대해서는 within과 contains를 동시에 만족하는 'T*F**FFF*'가 되어야 한다.

위의 내용을 요약 하면 <표 6>과 같다.

<표 6> 공간 관계 연산의 설계

Operation	patternMatrix
disjoint	$\Leftrightarrow$ a.Relate(b,'FF*FF****')
intersects	$\Leftrightarrow$!a.disjoint(b)
touches	$\Leftrightarrow$ a.Relate(b,'FT*******') $\vee$ a.Relate(b,'F**T*****') $\vee$ a.Relate(b,'F***T****')
within	$\Leftrightarrow$ a.Relate(b,'T*F**F***')
contains	$\Leftrightarrow$ a.Relate(b,'T*****FF*')
overlaps	$\Leftrightarrow$ a.Relate(b,'T*T***T**') if P/P, S/S case a.Relate(b,'1*T***T**') if L/L case a.Relate(b,'2*T***T**') if A/A case
crosses	$\Leftrightarrow$ a.Relate(b,'0********') if L/L case a.Relate(b,'1********') if A/A case a.Relate(b,'0********') if A/L case a.Relate(b,'T*T******') else
equals	$\Leftrightarrow$ a.Relate(b,'T*F***F**') if P/P case a.Relate(b,'T*F**FFF*') else

5. 결론 및 향후 연구

본 연구에서는 공간 데이터베이스 분야에서 핵심이라 말할 수 있는 공간 관계

연산을 3차원 공간상에서 0부터 3차원을 갖는 객체들에 대하여 설계하였다. OGC에서 정의하는 표준을 따르는 8가지 공간 관계 연산을 3차원의 기하 모델에 적용이 가능하도록 설계하였다. 표준이 되는 GML3의 데이터 모델을 이용하였으며, 위상 관계연산을 표현하는 방법인 Point-set Topology 중 DE-9IM을 사용하여 설계를 하였다. 세부적인 설계의 목표로는 기존의 공간 데이터베이스 표준에서 제시하는 2차원 데이터 모델에서도 적용가능하고, 동인한 속성을 갖는 Multi model까지 적용가능도록 설계하였다. 사용자의 관점에서 불필요한 연산을 제거하여 연산 비용을 줄였다.

본 연구에서 설계된 연산은 데이터 모델과 연산 정의에 있어서 표준으로 자리잡은 명세서를 준수 함 으로서 객관성을 확보하였고 3차원의 공간객체를 저장할 수 있는 공간 데이터베이스 관리 시스템의 연산자 구현에 이용될 수 있을 것으로 기대된다. 하지만 향후 연구과제로서 연구에서 설계한 연산들에 대한 타당성을 검증하는 연구가 요구된다. 이와 함께 차원수의 증가에 따라 객체들 사이에 더욱 다양한 공간 관계가 나타나는 것을 확인할 수 있다. 따라서 현재까지의 대상 공간이었던 2차원공간에서 나타나는 관계 이외에 나타날 수 있는 3차원 공간 관계에 대한 연구가 요구된다.

참고문헌

David Beddoe · Paul Cotton · Robert Uleman · Sandra Johnson · Dr. John R. · Herring, 1999, "OpenGIS Simple Features Specification for SQL Revision 1.1", OpenGIS Consortium.

Eliseo Clementini · Paolino Di Felice, 1994, "A Comparision of Methods for Representing Topological Relationships", Information Sciences 80, pp.1-34

Eliseo Clementini · Paolino Di Felice · Peter van Oosterom, 1993, "A Small Set of Formal Topological Relationships Suitable for End-User Interaction", International Symposium on Advances in spatial databases, pp.277-295

Max Egenhofer · Robert Franzosa, 1991, "Point-Set Topological spatial relations", International Journal of Geographical Information Systems 5 (2), pp.161-174

Max Egenhofer · Robert Franzosa, 1995, "On the Equivalence of Topological Relations", International Journal of Geographical Information Systems 9 (2), pp.133-152

Max Egenhofer, 1994, "Deriving the Composition of Binary Topological Relations", Journal of Visual Language and Computing 5 (2), pp.133-149

Max Egenhofer, 1994, "Spatial SQL: A Query and Presentation Language.", IEEE TKDE, 6 (1), pp.86-95

Simon Cox · Paul Daisey · Ron Lake · Clemens Portele · Arliss Whiteside, 2003, "Geography Markup Language (GML) Implementation Specification v3.0", OpenGIS Consortium.

Shashi Shekhar · Sanjay Chawla, 2002, "Spatial Databases A TOUR", Prentice Hall, pp.27-31

김상호 · 강구 · 류근호, 2003, "3차원 공간 위상관계 연산자의 설계", 한국정보처리학회 정보처리논문지D, 제10권 제2호, pp.211-220.

정호영 · 이민우 · 전우제 · 박수홍, 2003, "GML 응용스키마를 이용한 공간데이터베이스 스키마 모델링", 한국GIS학회 추계학술대회