2차원 지리 객체를 위한 시공간 객체 모델 설계

이 현 아[†]·남 광 우[†]·류 근 호[††]

요 약

지금까지 시공간 객체를 다양한 관점에서 접근하여 표현하려는 많은 연구가 이루어졌다. 이 대부분의 연구는 GIS적 관점과 시간 데이터베이스 관점, 그리고 객체 지향적 관점 및 데이터타입 접근 방법으로 시공간 객체를 표현하고 있다. 시공간 객체는 공간 속성이 불연속적으로 변하는 객체, 위치 정보가 연속적으로 변하는 객체, 그리고 면적과 위치가 연속적으로 변하는 객체로 분류된다. 그러나 기존의 시공간 모델들은 한 종류의 객체에만 초점을 맞추고 있다. 따라서 이 논문에서는 유클리드 평면상에서 세가지 형태의 객체를 모두 표현할 수 있는 시공간 객체 모델을 제안한다. 이를 위해 유효시간 개념을 확장한 시간 모델과 함께 객체의 연속된 두 버전 간의 관계성을 정의하여 이동 객체와 이력 객체 모두를 표현하는 방식을 사용한다. 여기서 제안하는 2차원 시공간 객체 모델은 개방형 GIS 명세서에서 제시하고 있는 2차원 공간 객체 모델을 따름으로써 표준 공간 모델과의 호환성을 보장한다.

Design of a spatiotemporal object model for 2D geographic objects

Hyun Ah Lee[†]·Kwang Woo Nam[†]·Keun Ho Ryu[††]

ABSTRACT

Most of works have been performed on representation of spatiotemporal objects from various points of view. Most of them represent spatiotemporal objects using approaches from GIS, temporal databases, object-oriented databases or data type. Spatiotemporal objects can be classified as objects whose position and shape changes discretely over time, objects whose position changes continuously and objects whose shape changes continuously as well as position. Previous works on spatiotemporal model have focused on only one of them. In this paper, we propose a spatiotemporal model that can represent three types of objects in Euclidean plan. For this purpose, we represent both discrete and continuous moving objects by defining temporal model extended from valid time and by defining relationship between two consecutive versions of objects. The proposed spatiotemporal object model is based on open GIS specification so that it has compatibility with existing spatial data model.

키워드 : 2차원 시공간 모델링(2D spatiotemporal modeling), 시공간 객체(Spatiotemporal object), 객체 모델(Object model)

1. 서 론

실세계의 객체들은 공간적인 정보 뿐만 아니라 시간적 정보와도 연관을 갖는다. 이러한 객체들은 공간 상에서의 위치 또는 영역에 관한 정보 뿐만 아니라 형태에 대한 정보들도 시간에 따라 변하게 된다. 그럼에도 불구하고 과거의 연구는 공간 데이터베이스와 시간 데이터베이스의 두 분야로 서로 독립적으로 연구되어 왔다. 시간 데이터베이스에서는 실세계의 과거 상태 그리고 데이터베이스의 과거 상태라는 두 가지 관점에서 데이터베이스의 정보를 확장하려고 하였고 공간 데이터베이스에서는 일반적으로 공간객체를 point, line, region의 추상화를 통해 데이터베이스의 공간적 표현을 확장하였다[1]. 그러나, 최근의 많은 응용분야는 시간에 따라 변하는 공간객체 즉, 시간과 공간을 통합한 시공간 객체를 다룰 수 있는 데이터베이스를 요구하게 되었다[2-4]. 이러한 시공간 객체를 다루는 데이터베이스를 시공간 데이터베이스(spatiotemporal database)라고 한다.

여기서, 시공간 객체는 세 가지 형태로 나누어 질 수 있다[5]. 첫 번째 경우, 위치나 모양과 같은 공간적 특성들이 시간에 따라 불연속적으로 변하는 객체로써 예를 들면 토지(land parcels)와 같은 것을 고려할 수 있다. 이런 경우 다소 빈번한 갱신을 통해 비교적 쉽게 이 객체의 움직임 즉 위치

※ 본 연구는 KOSEF의 목적기초(R01-1999-00243)연구비와 한국전자통신연구원의 "4D 시공간 데이터 제공자 컴포넌트 개발" 위탁과제의 연구비 지원에 의해 수행되었음.
† 준 회 원 : 충북대학교 대학원 컴퓨터과학과
†† 종신회원 : 충북대학교 전기전자및컴퓨터공학부 교수
논문접수 : 2001년 9월 19일, 심사완료 : 2001년 11월 5일

또는 영역의 변화를 데이터베이스에서 유지하고 다룰 수 있으며, 불연속적으로 변하는 공간 속성(위치 또는 영역)을 획득하고 질의하기 위한 많은 방법들[6-10]이 제안되었다. 두 번째 경우, 객체의 모양은 변하지 않지만 그 위치는 연속적으로 변하는 객체로써 움직이고 있는 자동차와 같은 예를 들 수 있다. 마지막으로 객체의 위치 뿐만 아니라 그 모양까지 연속적으로 변하는 객체로써 태풍과 같이 시간에 따라 그 영역과 함께 그 위치도 연속적으로 변하는 이동객체가 있다.

이 논문에서는 2차원 유클리드 평면 상에서 세 가지 형태의 시공간 객체를 모두 표현할 수 있는 시공간 객체 모델을 제안한다. 개방형 GIS 명세서의 2차원 공간 객체 모델에 시간을 확장하여 불연속적으로 변하는 공간 객체는 물론 연속적으로 위치 또는 위치와 모양이 모두 변하는 객체를 표현한다. 또한 제안한 모델로 구축된 데이터에 대해 분석 및 질의의 기능을 제공하기 위한 연산자를 함께 정의한다.

효율적인 전개를 위해 제 2장에서 시공간 객체의 표현에 대한 문제를 정의하고, 제 3장에서는 시공간 객체 모델을 구성하는 공간 모델과 시간 모델을 설명한다. 시공간 모델은 이력 객체에 대한 모델과 이동 객체에 대한 모델을 구분하여 제 4장에서 정의한다. 제 5장에서는 정의된 모델을 시나리오에 적용하여 객체가 구성되는 형태와 질의의 수행과정을 보이고, 관련된 연구들에 대해 제 6장에서 비교하여 정리한다. 마지막으로 제 7장에서 결론을 맺는다.

2. 문제 제기

시공간 데이터의 구성은 유효한 공간 데이터와 해당 타임 스탬프로 이루어진다. 이 때 연속된 타임 스탬프들의 간격동안 저장된 공간 속성이 어떻게 다루어지느냐에 따라 지리 객체가 이동 객체인가 혹은 이력 객체인가로 구분된다. 실제 지리 객체는 토지 구획 등과 같이 저장된 공간 데이터가 변경되는 다음 시점까지 유지될 수도 있고, 이동하는 자동차와 같이 시간의 흐름에 따라 계속 공간 데이터가 변경될 수도 있다.

다음 (그림 1)은 동일한 위치 정보에 대한 이력 객체 관점과 이동 객체 관점에 대한 차이점을 잘 설명하고 있다. t1, t2 그리고 t3 시점에서의 각 공간 정보가 (그림 1(a))와 같이 저장되어있다고 가정하자. 이 때 (그림 1(a))의 시공간 객체는 각 시점에 따른 해당 공간 정보의 해석 방법이 이력 객체에 대한 것인가, 혹은 이동 객체에 대한 것인가에 따라 (그림 1(b)) 또는 (그림 1(c))와 같이 해석될 수도 있다. 예를 들어 $f(t)$를 t(단, $t1 \le t \le t3$)에서 객체의 공간 정보를 반환하는 함수라고 가정하고 $t1 < t' < t2$를 만족하는 t'가 존재한다면, (b)의 경우 $f(t') = f(t1)$이고, (c)의 경우 $f(t') \neq f(t1)$이 된다.

이와 같은 결과는 주어진 시공간 객체 (그림 1(a))의 해석에 대해 다음 두 가지 문제를 야기시킨다.

1) 이력 시공간 (그림 1(b))과 이동 시공간 (그림 1(c))의 해석 방식이 서로 다르다.
2) 한 가지에만 초점을 맞춘 방식으로는 다른 방식의 현상을 올바르게 해석할 수 없다.

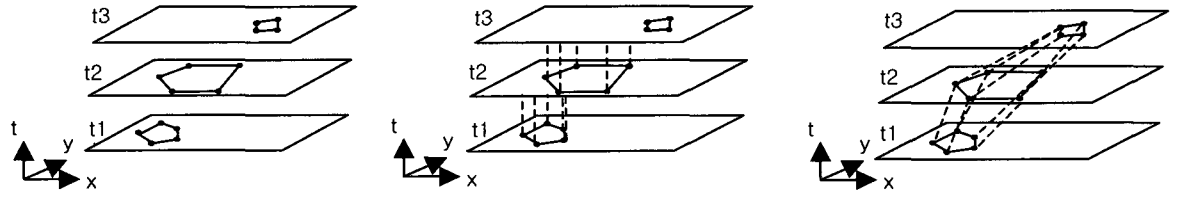
따라서, 이 논문에서 제안하고 있는 모델링방법은 위 문제들에 대한 해결 방안을 제시한다. 즉 기존의 이력 객체를 위한 객체 지향적 모델링방법의 단점을 보안하고 이동 객체에 대해 확장함으로써 이동 객체와 이력 객체의 모델을 통합시킨다.

3. 2차원 공간 모델과 시간 모델

다음은 여기서 제안하는 시공간 모델의 기반이 되는 공간 모델을 설명한다. 그리고 공간 모델을 시공간으로 확장하기 위해 정의한 시간 모델을 기술한다.

3.1 2차원 공간 모델

OpenGIS 명세서에서는 Geometry라는 추상 클래스로부터 Point, Curve, LineString, Polygon등의 단일 2차원 공간 객체를 정의하고, 단일 객체의 집합(collection) 형태로 Multipoint, MultiLineString 그리고 MultiPolygon를 정의한다. 이러한 공간 객체들 간의 공간 관계 연산자들은 ISpatialRelation 인터페이스를, 공간 객체들 간의 공간 분석 연산자들은 ISpatialOperator 인터페이스를 통하여 Geometry 객체에 제공된다[11]. 공간 객체에 대한 클래스 계층도와 각 메소드들에 대한 설명은 OpenGIS의 구현 명세서에서 자세히 소개



(a) 시공간 Polygon (b) 이력 시공간 Polygon (c) 이동 시공간 Polygon

(그림 1) 동일한 시공간 객체 정보의 해석에 대한 관점들

되며, 관련된 공간 관계 연산자들은 4장에서 시공간 연산자와 함께 다시 언급될 것이다.

3.2 시간 모델

데이터베이스에서 시간의 형태는 유효시간, 거래시간, 사용자정의시간 등을 사용할 수 있다. 특히 유효시간과 거래시간은 현실 세계에서 발생하는 객체의 이력을 정확하게 기록할 수 있도록 한다[12]. 시간은 구조적으로 선형시간, 분기시간, 순환시간으로 정의될 수 있으며 시간의 명확한 표현을 위해 세 가지 측면을 모두 고려해야 하지만, 대부분의 시간 모델은 가장 일반적인 형태이며 구현도 용이한 선형시간만을 고려하여 설계된다. 이 논문에서도 시간에 대한 표현을 단순화하기 위해 선형시간 구조만을 가정하고, 이동 객체에 대한 표현을 위해 우선적으로 현실 세계에 존재하는 객체에 대한 논리적인 시간인 유효시간을 고려한다.

유효시간은 일반적으로 사용자에 의해서 결정되는데 사용자가 위치한 시간시스템에 의해서 상대적인 값으로 표현된다. 2차원 공간 데이터를 시간으로 확장하기 위해 유효 시간을 표현하는 Validtime 클래스를 정의한다. Validtime 클래스는 객체의 유효시간 지원을 위한 추상 클래스이며, 시간에 관련되는 모든 기본 연산자들이 정의된다. 여기에서 제안되는 모든 시공간 객체는 이 유효시간 객체를 상속받음으로써 유효시간만을 갖는다.

Validtime 클래스는 객체의 생성 시간과 소멸 시간을 나타내기 위한 From과 To의 2가지 속성과 <표 1>과 같은 4가지 기본 메소드를 갖는다. 그리고 유효시간 객체간의 시간 관계를 평가하는 [13]의 시간 연산자들을 포함하고, 이에 대한 정의는 뒤에서 자세히 설명한다.

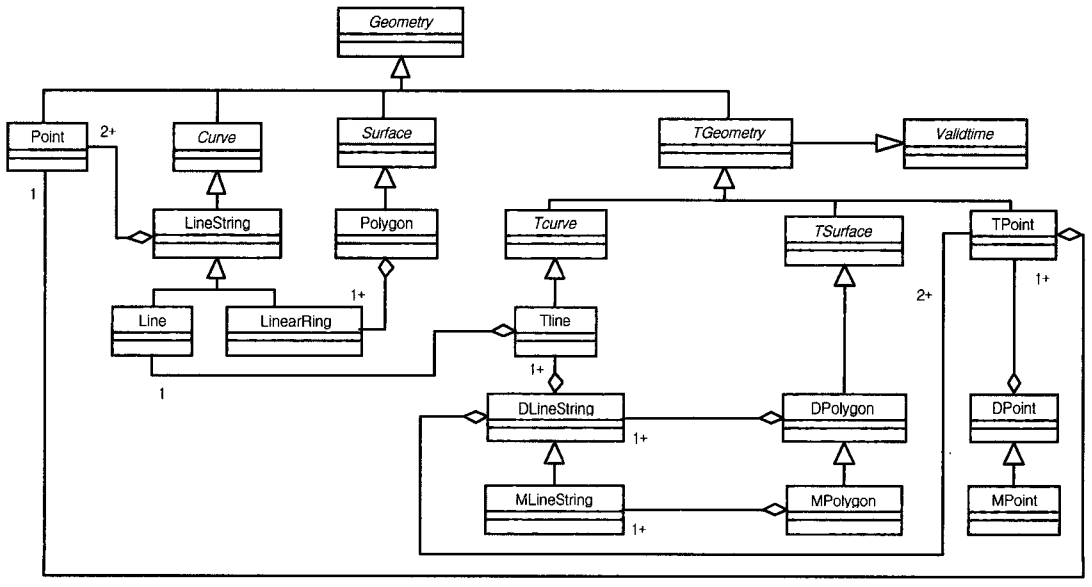
〈표 1〉 Validtime 클래스의 기본 메소드들

메 소 드	설 명
IsPeriod( )	A.From < A.To일 경우 true
getInterval( )	A의 전체 유효시간 구간 계산하여 결과 값을 반환
IntervalToMin( )	시간 구간 값을 분으로 환산하여 결과 값을 반환
IntervalToSec( )	시간 구간 값을 초로 환산하여 결과 값을 반환

4. 2차원 시공간 모델

제안하는 2차원 시공간 모델은 Validtime 이라는 추상클래스를 정의함으로써 공간 모델을 시간 차원으로 확장하는 방식으로 설계한다. 각 공간 속성은 해당하는 유효시간을 갖고, 해당 객체의 버전들간의 관계를 정의함으로써 이력 객체와 이동 객체를 표현한다. 이 때 고려해야 할 점은 공간의 어떤 레벨이 시간 참조와 연관이 있는가의 문제를 명확히 정의해야 한다는 것이다. 이에 대하여 [14]는 공간과 시간 참조 병합에 있어서 다음과 같은 세 가지 접근 방법을 제시하였다.

- 전체 지리객체에 타임스탬핑을 하는 것으로 저장구조 측면에서 가장 적은 비용이 들지만 객체의 시간 특성에 대한 제한된 표현을 제공한다.
- 시간과 공간에 대한 병합을 기본적인 수준의 공간 객체 즉, point, string, polygon에 타임스탬핑을 하는 것으로 비록 비싼 비용의 저장구조를 요구하지만 객체에 대한 시간적 측면의 표현력은 풍부하게 된다. 즉 공간 객체에 대한 더 많은 시간 질의처리에 대한 정보를 가지고 있게 된다.
- 마지막으로 시간과 공간의 병합을 point 수준에서 하고 이것을 시공간 클래스들의 새로운 집합에 전파하는 방법



(그림 2) 2차원 객체의 시공간 클래스 계층도

으로 이 접근 방법은 자세하게 개발되어지고 있지 않다.

이 논문에서는 세 가지 시간과 공간의 병합 방법 중 두 번째 접근 방법을 선택하여 기본 공간 객체인 Point와 Line에 대한 유효시간을 저장한다. 이력 객체 타입으로는 DPoint, DLineString, DPolygon 등이 존재하며 이러한 타입들은 모두 TPoint와 TLine의 집합으로 구성된다[15]. 이동 객체 타입은 시공간 객체간의 공간적 변화를 정의한다.

(그림 2)는 OpenGIS 공간 모델에 유효시간을 지원하도록 확장한 2차원 객체의 시공간 타입 클래스의 계층도이다.

4.1 TGeometry

각 하위 클래스에 공간과 시간의 통합 개념을 제공하기 위해 TGeometry가 정의된다. TGeometry는 Geometry 클래스와 Validtime 클래스로부터 각각 공간과 시간의 속성을 이중 상속받은 2차원 시공간 객체에 대한 추상 클래스이다. 하위 클래스로써 TPoint, TCurve, TSurface가 존재한다.

기본 속성으로 시공간 차원을 나타내는 tDimension, 시공간적으로 비어있는 상태를 나타내는 tIsEmpty, 그리고 시공간적으로 닫혀있는 상태를 나타내는 tIsSimple이 있고, <표 2>와 같은 기본 메소드를 갖는다. TGeometry에서 제공되는 시공간 연산자들은 뒤에서 다시 언급된다.

〈표 2〉 TGeometry 클래스의 기본 메소드들

메 소 드	설 명
tClone( )	TGeometry의 복사본을 반환
tEnvelope( )	TGeometry의 시공간 최소 경계 사각형을 TGeometry 형으로 반환
tExtent3D( )	TGeometry의 시공간 최소 경계 사각형의 좌표 값 반환
tsetEmpty( )	TGeometry를 빈 TGeometry로 설정

4.2 2차원 이력 객체 모델

2차원 공간 인스턴스는 Point, LineString, Polygon 등이 존재한다. 각 공간 객체에 대한 이력을 표현하기 위해 시간으로 확장한 클래스는 DPoint, DLineString, DPolygon 등이 있다. 이러한 클래스들은 시공간에 대한 추상 클래스인 TGeometry를 상속받는다.

4.2.1 2차원 이력 객체

DPoint는 시간에 따라 공간상의 위치가 불연속적으로 변하는 점 객체를 표현하기 위한 객체이며, TPoint들의 집합으로 구성된다. TPoint는 유효시간 객체를 상속받은 Point 객체로 정의되고, 해당 유효 시간에 대한 Point 객체의 공간 속성을 나타내는 하나의 버전이다. 또한 모든 시공간 객체를 구성하는 기본 요소이다. 따라서 DPoint는 Point 객체의 시간에 대한 버전들의 집합이고, 동일한 DPoint의 구성 버전인 TPoint들 간의 유효시간에 대한 교집합은 존재하지 않아야 한다. 이는 객체의 공간 정보는 동시에 다른 공간에 존재할 수 없음을 보장한다.

[정의 1] $DPoint \Leftrightarrow \{tp_1, \cdots, tp_n\} = \{(x_1, y_1, VT_1), \cdots, (x_n, y_n, VT_n)\}$

단, tp_i는 TPoint 객체이고, VT_i는 (From, To)의 두 타임스탬프로 구성된 유효시간이며, $\forall i, j \in N$일 때, $VT_i \cap VT_j = \phi$이다. □

1차원 공간객체인 LineString은 여러 점들과 그 사이의 선분들로 구성되며 하위 객체로 Line과 LinearRing을 갖는다. 시공간 객체를 정의할 때는 TPoint와 함께 시공간 객체를 표현하는 기본단위로써 TLine을 먼저 정의한다. TLine은 유효시간 객체를 상속받은 Line이고, 각각 하나의 유효시간과 공간 속성을 갖는다. TLine과 TPoint들의 집합으로 DLineString 객체를 구성하여 LineString을 이루는 각 세그먼트들에도 유효시간을 할당하고 세그먼트 단위까지 질의 영역을 확장하였다. DLineString의 모든 유효시점에 대해 시작점과 끝점이 만난다면 이는 DRing으로 정의한다. 다음의 <표 3>은 DLineString 객체를 구성하기 위해 필요한 메소드들이다.

〈표 3〉 DLineString 클래스의 기본 메소드들

메 소 드	설 명
StLength( )	주어진 시간 t에서 LineString의 총 길이 반환
StStartPoint( )	주어진 시간 t에서 LineString의 시작점 반환
StEndPoint( )	주어진 시간 t에서 LineString의 끝점 반환
NumTLines( )	TLine의 개수를 반환
TLine( )	색인 값에 대응하는 TLine반환
NumTPoints( )	TPoint의 개수 반환
TPoint( )	색인 값에 대응하는 TPoint반환

[정의 2] $DLineString \Leftrightarrow \{dl_1, \cdots, dl_n\}$,
$dl_i \Leftrightarrow \{tp_{is}, tl_i, tp_{ie}\} = \{(p_{is}, VT_{is}), (p_{is}, p_{ie}\ VT_i), (p_{ie}, VT_{ie})\}$
단, tp는 TPoint이고, tl은 TLine이다. $\forall i, j \in N$일 때 $VT_{total} = VT_{1s} \cup \cdots \cup VT_{ne}$, 이다. □

DPolygon은 2차원 공간상에 면적을 가진 객체 Polygon에 대한 시공간 객체로써 시간에 따른 Polygon의 불연속적인 변화를 표현한다. 시간을 고려하지 않은 Polygon은 공간 상에 시작점과 끝점이 동일한 LineString인 Ring들의 집합이다. 이러한 개념을 시간 차원으로 확장을 할 경우 시간에 대한 한 가지 제약사항이 더 추가됨으로써 DPolygon을 정의할 수 있다. 즉, 모든 유효시점에 대해 공간 상으로 시작점과 끝점이 동일한 LineString의 집합, 즉 DRing들의 집합을 경계로 갖는 2차원 객체로 정의된다. DPolygon은 <표 4>와 같은 기본 메소드들을 갖는다.

〈표 4〉 DPolygon 클래스의 기본 메소드들

메 소 드	설 명
ExteriorDRing( )	DPolygon의 외부 경계 DRing을 반환
NumInteriorDRings( )	DPolygon의 내부 DRing들의 개수 반환
InteriorDRing( )	주어진 인덱스에 해당하는 DPolygon의 내부 DRing 반환

[정의 3] DPolygon $\Leftrightarrow < dr_{outer}, \{dr_{inner1}, \cdots, dr_{innern}\} >$
단, dr_{outer}, dr_{inneri}는 DRing이며, $VT_{total} = VT_{outer} \cup VT_{inner1} \cup \cdots \cup VT_{innern}$이다. □

4.2.2 2차원 이력 객체를 위한 연산자

데이터베이스에서 가장 중요하고 또한 기본적인 기능은 데이터의 저장과 저장된 데이터의 조회이다. 따라서 데이터 모델 설계 시 대상 데이터의 표현 방법 뿐만 아니라 데이터를 효율적으로 조회할 수 있는 연산자도 함께 고려되어야 한다. 제안된 시공간 객체 모델에서 지원되는 연산자는 Geometry, Validtime, TGeometry 클래스를 통해 각각 공간, 시간 그리고 시공간을 통합적으로 분석할 수 있는 연산들을 제공한다.

공간 연산의 경우, 이 모델의 기반이 된 OGC의 명세서에 따라 Geometry 클래스에서 정의되는 위상 관계 연산자 및 기하 연산자들을 채용한다. 위상 관계 연산자로는 Contains( ), Crosses( ), Disjoint( ), Equals( ), Intersects( ), Overlap( ), Touches( ), Within( ), 그리고 기하 연산자로는 Boundary( ), Buffer( ), ConvexHull( ), Difference( ), Distance( ), Union( ), Intersection( ), SymmetricDifference( ) 등이 존재한다.

시간 연산의 경우, 시공간 객체간의 시간 관계를 평가하기 위한 [13]의 7가지 연산자들이 정의되어 하위 클래스로 상속된다. 다음 <표 5>에 나타난 각 연산자들에 대한 정의는 아래와 같은 가정을 전제로 한다.

(가정 1)

1. A, B는 각각 Validtime 클래스의 객체를 나타낸다.
2. 객체 A의 생성 시간은 A.From, 소멸 시간은 A.To으로 나타낸다.
3. A.From ≤ A.To

〈표 5〉 시공간 객체에 대한 시간 관계 연산자들

연 산 자	설 명
Before( )	A.To < B.From일 경우 true
Equals( )	(A.From = B.From)∧(A.To = B.To)일 경우 true
Meets( )	A.To = B.From일 경우 true
Overlaps( )	(A.To ≤ B.To)∧(A.From ≤ B.From)일 경우 true
During( )	(A.From > B.From)∧(A.To < B.To)일 경우 true
Starts( )	(A.From = B.From)∧(A.To < B.To)일 경우 true
Finishes( )	(A.From > B.From)∧(A.To = B.To)일 경우 true

〈표 6〉 이력 객체를 위한 시공간 연산자 분류

분 류	메 소 드
시공간 객체 연산	instanceAt( ), instancePeriod( ), validtime( ), location( )
시공간 위상 관계 연산	tEquals( ), tDisjoint( ), tIntersects( ), tTouches( ), tCrosses( ), tWithin( ), tContains( ), tOverlaps( )
시공간 기하 연산	tBoundary( ), tDistance( ), tBuffer( ), tIntersection( )

〈표 7〉 2차원 이력 객체의 시공간 연산자들

연 산 자	설 명
tWithin( )	$(\alpha.validtime \cap \beta.validtime = \alpha.validtime) \wedge (\alpha.spatial \cap \beta.spatial = \alpha.spatial) \wedge (\alpha.spatial(I) \cap \beta.spatial(E) \neq \phi)$
tCrosses( )	α/β가 각각 DPoint/DLineString, DPoint/DPolygon, DLineString/DLineString 또는 DLineString/DPolygon 일 때, $(\alpha.validtime \cap \beta.validtime \neq \phi) \wedge ((\dim(\alpha.spatial(I) \cap \beta.spatial(I)) < \max(\dim(\alpha.spatial(I)), \dim(\beta.spatial(I)))) \wedge (\alpha.spatial \cap \beta.spatial \neq \alpha.spatial) \wedge (\alpha.spatial \cap \beta.spatial \neq \beta.spatial)$
tDisjoint( )	$(\alpha.validtime \cap \beta.validtime = \phi) \wedge (\alpha.spatial \cap \beta.spatial = \phi)$
tEquals( )	$(\alpha.validtime = \beta.validtime) \wedge (\alpha.spatial = \beta.spatial)$
tIntersects( )	$(\alpha.validtime \cap \beta.validtime \neq \phi) \wedge (\alpha.spatial \cap \beta.spatial \neq \phi)$
tOverlaps( )	α/β가 각각 DPoint/DPoint, DLineString/DLineString 또는 DPolygon/DPolygon 일 때, $(\alpha.validtime \cap \beta.validtime \neq \phi) \wedge ((\dim(\alpha.spatial(I)) = \dim(\beta.spatial(I)) = \dim(\alpha.spatial(I) \cap \beta.spatial(I))) \wedge (\alpha.spatial \cap \beta.spatial \neq \alpha.spatial) \wedge (\alpha.spatial \cap \beta.spatial \neq \beta.spatial)$
tTouches( )	$(\alpha.validtime \cap \beta.validtime \neq \phi) \wedge (\alpha.spatial(I) \cap \beta.spatial(I) = \phi \wedge (\alpha.spatial \cap \beta.spatial \neq \phi)$
tContain( )	$(\alpha.validtime \cap \beta.validtime = \beta.validtime) \wedge (\alpha.spatial \cap \beta.spatial = \beta.spatial) \wedge (\alpha.spatial(I) \cap \beta.spatial(E) \neq \phi)$
tBoundary( )	tEnvelope을 이루는 각 요소를 TGeometry형으로 반환
tBuffer( )	주어진 시간 t에서 주어진 거리 d에 속하는 Geometry 객체들을 반환
tDistance( )	다른TGeometry의 한 Point와 시간적으로 동등(equals)할 때 공간적으로 최소 거리를 반환
tIntersection( )	주어진 시간에서 다른 TGeometry객체와 겹친 부분을 Geometry 형식으로 반환
validtime( )	TGeometry의 Validtime 반환
instanceAt( )	주어진 시점(time point)에서의 Geometry 반환
instancePeriod( )	주어진 시간구간(time period)에서의 TGeometry 반환
location()	시간에 따른 Geometry의 변화를 Geometry 집합으로 반환

마지막으로 시공간 연산은 공간 연산을 시간에 대해 확장하여 정의하고 이력 객체로부터 공간 속성 및 시간 속성에 대한 정보를 추출한다[16]. 이러한 이력 객체에 대한 시공간 연산들은 관계형 데이터 모델을 기반으로 <표 6>과 같이 분류된다[17].

이 때, 시공간 위상 관계 연산과 시공간 객체 연산 및 시공간 기하 연산은 TGeometry 클래스에서 정의되고 각 하위 클래스에 제공된다. <표 7>은 다음과 같은 가정 하에 각 연산자들에 대해 설명하고 있다.

(가정 2)

1. α, β는 각각 TGeometry 클래스의 객체를 나타낸다.
2. 객체 α의 시간 속성은 α.validtime로 나타낸다.
3. 객체 α의 공간 속성은 α.spatial로 나타낸다.
4. 객체 α의 공간 속성의 외부, 내부, 경계는 α.spatial(E), α.spatial(I), α.spatial(B)로 나타낸다.
5. dim(α)는 객체 α의 공간적 차원을 반환하는 함수이다.

시공간 위상관계 연산자인 tIntersect( )는 원본 객체(source object)와 대상 객체(target object)가 시간적으로 그리고 공간적으로 교차할 때 참을 반환한다. 이 때, 원본 객체란 비교의 기준이 되는 객체이며 대상 객체란 원본 객체와 비교하기 위한 객체이다. 각 시공간 연산은 대상 객체에 대해 원본 객체의 메소드에서 수행된다. 또한 시공간 객체는 버전들의 집합으로 구성되므로 실제 연산은 버전 단위로 진행된다.

```
algorithm tIntersect(targetObject)
input : tGeometry 타입의 대상 객체인 targetObject
output : 대상 객체와 원본 객체가 시공간적으로 교차할 경우 TRUE,
         그렇지 않으면 FALSE 반환
method :
tIntersect는 대상 객체와 원본 객체의 1 : 1 비교이다. 원본 객체인
sourceObject는 DLineString 타입이고, 이 경우 메소드인 tIntersect
는 DLineString에 대한 연산에 알맞은 알고리즘으로 정의된다.

    let sObject and tObject are TLineString type, initially null
    get a geometry type of targetObject
    for each version of sourceObject do
      set a version of sourceObject on sObject
      for each version of targetObject do
        set a version of targetObject on tObject
        if a validtime of sObject overlaps a validtime of tObject then
          if sObject.Intersect(tObject) is true then true
          else return false
        else continue
end tIntersect
```

(알고리즘 1) DLineString의 tIntersect 메소드

```
algorithm Intersect(targetObject)
input : 대상 객체가 되는 targetObject(Geometry 타입의 공간 객체)
output : 대상 객체와 원본 객체가 공간적으로 교차할 경우 TRUE, 그
         렇지 않으면 FALSE 반환
method :
Intersect는 대상 객체와 원본 객체의 1 : 1 비교이다. 원본 객체인 source-
Object는 TLineString 타입이고, 이 경우 메소드인 Intersect는
LineString에 대한 연산에 알맞은 알고리즘으로 정의된다.

    get a geometry type of targetObject
    for each point belonging to a LineString of sourceObject do
      get one point and the next point
      set up a linear equation using two points
      if the geometry type of targetObject is Point then
        if targetObject belong to a coordination range of the linear
        equation of sourceObject then
          substitute targetObject to the linear equation
          if the linear equation materialize then return true
          else return false
      if the geometry type of targetObject is LineString then
        for each point belonging to a LineString of targetObject do
          get one point and the next point
          set up a linear equation using two points
          if two gradients of sourceObject and targetObject equal
          then
            compare two y-intercepts of sourceObject and targetObject
            if y-intercepts equal then return true
            else return false
          else find a crossing of two linear equations
            if the crossing belong to a range of a linear equation
            of targetObject then
              if the crossing belong to a range of a linear equa-
              tion of sourceObject then
                return true
              else return false
    if the geometry type of the targetObject is Polygon then
      // if a sourceObject intersect only one boundary of target
      // Object, then two objects intersect
        for each point belonging to a LineString of targetObject
        do
          get one point and the next point
          set up a linear equation using two points
          if two gradients of sourceObject and targetObject equal
          then
            compare two y-intercepts of sourceObject and target-
            Object
            if y-intercepts equal then return true
            else return false
          else find a crossing of two linear equations
            if the crossing belong to a range of a linear equation
            of targetObject then
              if the crossing belong to a range of a linear equa-
              tion of sourceObject then
                return true
              else return false
end Intersect
```

(알고리즘 2) TLineString의 Intersect 메소드

시공간 기하 연산자인 tDistance( )는 주어진 시간에 유효한 객체의 TPoint 간의 최단 거리를 반환한다. 이 때 대상 객체는 DPoint로 한정하고, 원본 객체는 거리를 계산하는 원점이 된다. 그러나 원본 객체가 DPoint가 아닐 경우 원점이 될 수 있는 TPoint가 여러 개 존재하게 되므로 DLintString인 경우 모든 구성 Point를 원점으로 설정하여 거리를 계산 후 최단 거리를 선택한다. DPolygon인 경우 외부 경계인 DRing

을 구성하는 각 TPoint를 원점으로 하여 거리를 계산하고 최단 거리를 선택하는 방식을 택한다.

```
algorithm tDistance(targetObject)
input : 대상 객체가 되는 targetObject(DPoint 타입으로 한정), source-
        Object의 타입은 DLineString
output : 대상 객체와 원본 객체의 최단 거리와 최단 거리의 결과가
         유효한 시간
method :
tDistance는 대상 객체와 원본 객체의 1 : 1 비교이다. 원본 객체인
sourceObject는 DLineString 타입이고, 이 경우 메소드인 tDistance
는 DLineString에 대한 연산에 알맞은 알고리즘으로 정의된다.
   for each version of sourceObject do
      for each version of targetObject do
         if a validtime of sourceObject overlaps a validtime of
         targetObject then
            for each point of sourceObject do
               compute the distance between point of sourceObejct
               and targetObject
               set previous distance on A and set the next distance
               on B
               store smaller distance then the other and the vali-
               dtime of targetObject with A
   return A
end tDistance
```

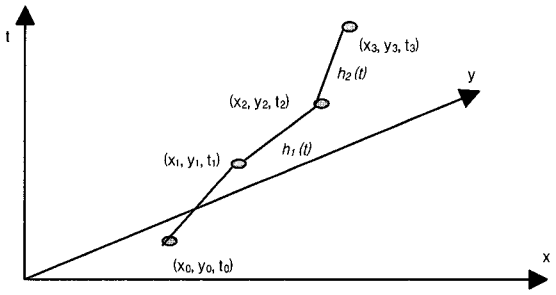
(알고리즘 3) DLineString의 tDistance 메소드

4.3 2차원 이동 객체 모델

2차원 이동 객체는 이력 객체로부터 상속된 클래스로부터 정의되며, (그림 2)에서 보이는 바와 같이 DPoint, DLine-String, DPolygon을 상속받는 이동 객체 MPoint, MLine-String, MPolygon이 존재한다. 이것은 이력 객체와 이동 객체의 내부 자료 구조가 비슷하며 동일한 정보를 저장하고 있음을 의미하지만, 제 2장에서 논의한 것처럼 같이 이력 객체와 이동 객체는 저장된 정보에 대한 해석에는 차이가 있다. 따라서, 이력 객체 클래스로부터 이동 객체 클래스를 정의할 때 고려해야 할 점은 버전간의 관계를 어떤 방식으로 정의할 것인가 하는 점이다. 이력 객체는 이전 버전의 공간 속성이 다음 버전까지 그대로 유지되는 반면, 이동 객체는 이전 버전부터 연속되는 다음 버전까지 공간 속성이 계속 변하고 있기 때문에 이러한 내부적 관계를 해석하기 위한 메소드뿐만 아니라 이동 객체를 위한 연산자의 정의가 필요하다.

4.3.1 2차원 이동 객체

버스, 트럭, 택시, 비행기나 배와 같은 다양한 운송 수단들의 이동은 시간에 따라 계속 변하기 때문에 이력 객체를 사용하여 표현할 경우 매우 빈번한 갱신이 일어나거나 정확한 유효시간을 기록하지 못하는 등의 문제가 발생한다. 따라서 이러한 연속적인 위치의 변화는 MPoint라는 이동 점 객체를 사용하여 나타낸다. MPoint의 데이터는 DPoint와 마찬가지로 해당 객체에 대한 버전들의 집합으로 저장되므로 버전간의 관계를 표현할 수 있는 함수를 정의하여 저장되지 않은 위치 정보를 추출한다. 여기서는 스플라인 함수를 이용하여 연속적인 두 절점 상의 구간을 보간하는 방법을 사용한다.



(그림 3) 이동 점 객체 MP의 이동

t_1에서 t_4까지의 시간 동안 객체 MP가 (그림 3)과 같이 이동한다고 가정하자. 우선 x, t 좌표축만을 고려하여 La-grange 다항식을 사용하면 부분구간 1차 곡선은 다음과 같이 나타낼 수 있다.

$$S_k(t) = x_k + d_k(t - t_k),\ \text{단}\ d_k = (x_{k+1} - x_k)/(t_{k+1} - t_k) \quad (1)$$

따라서 위치 변화 함수 $h_x(t)$는 식 (2)와 같이 정의되고, 어떤 시간 t에 대해 $t_k \le t \le t_{k+1}$일 때, $h_x(t)$는 식 (3)으로 일반화할 수 있다.

$$h_x(t) = \begin{cases} x_0 + d_o(t - t_0),\ t \in [t_0,\ t_1], \\ x_1 + d_1(t - t_1),\ t \in [t_1,\ t_2], \\ \quad \vdots \\ x_k + d_k(t - t_k),\ t \in [t_k,\ t_{k+1}], \\ \quad \vdots \\ x_{N-1} + d_{N-1}(t - t_{N-1}),\ t \in [t_{N-1},\ t_N], \end{cases} \quad (2)$$

$$h_x(t) = h_{xk}(t) = x_k + d_k(t - t_k),\ t_k \le t \le t_{k+1} \quad (3)$$

y축에 대해서도 동일한 원리로 $h_y(t)$를 정의하면, 결국 임의의 시간 t에 대한 이동 점의 위치는 다음과 같다.

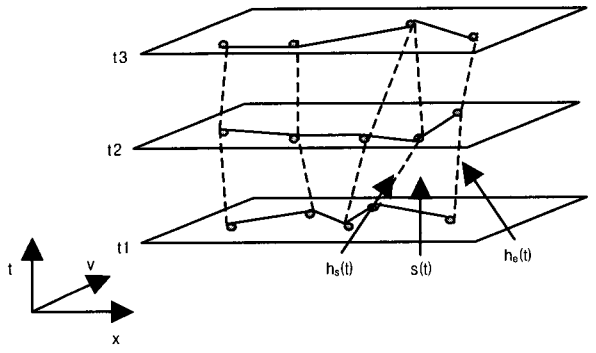
$$h(t) = (h_{xk}(t),\ h_{yk}(t)),\ t_k \le t \le t_{k+1} \quad (4)$$

이 방법을 고차다항식으로 확장할 수 있다. 예를 들어, 홀수개의 절점 $t_0, t_1, \cdots, t_{2M}$이 주어진다면, 각각의 부분구간 $[t_{2k}, t_{2k+2}]$(단 $k = 0, 1, \cdots, M-1$)에서 부분구간 2차 다항식을 구할 수 있다. 이런 2차 스플라인의 단점은 짝수번째 절점 t_{2k}에서 곡률이 갑자기 변하기 때문에 원하지 않는 굴곡이나 찌그러짐이 생길 수 있다는 것이다. 즉 2차 스플라인의 2차 도함수는 짝수번째 절점에서 불연속이다. 부분구간 3차 다항식을 사용하는 경우에는 다항식의 1차 도함수와 2차 도함수

가 모두 연속이어야 하지만 주어진 점들을 지나면서 오차에 영향을 받지 않고 비교적 정확한 이동 객체의 위치를 추출할 수 있다. 그러나 여기서는 2차 스플라인이나 3차 스플라인을 그릴 수 있는 조건 반드시 만족한다는 가정을 하지 않으므로 부분구간 선형보간법을 사용하여 이동 객체의 위치를 표현한다.

이와 같은 위치 변화 함수 $h(t)$에 대한 구체적인 구현은 MPoint를 구성하는 메소드들 중에서 instanceAt()에서 이루어지게 된다. instanceAt() 메소드뿐만 아니라 DPoint로부터 상속된 메소드들은 MPoint의 특성에 맞춰 $h(t)$ 함수를 사용하여 재구현 된다.

다른 시공간 객체와는 달리 MLineString의 경우 실세계 객체에 대한 응용 예제를 찾기가 어렵다. 즉, 이 클래스 타입이 적용될 수 있는 시공간 객체는 극히 한정적이라는 것이다. 따라서, MLineString은 자체의 시공간적 의미보다는 MPolygon에 응용될 수 있다는 의미가 더 크다고 할 수 있다. 다음 (그림 4)는 2차원 공간에서 LineString이 연속적으로 변하는 MLineString 객체인 ML을 보여준다.



(그림 4) 이동 선 객체 ML의 이동

ML의 이동을 표현하기 위해서는 각 세그먼트들에 대한 변화가 표현되어야 하고, 세그먼트의 변화는 세그먼트를 구성하는 시작점과 끝점의 변화로써 나타난다. 이는 각각 위치 변화 함수 $h_s(t)$와 $h_e(t)$로 표현한다. 따라서 n번째 세그먼트 변화의 함수 $s_k(t)$는 유효 시간 t를 인수에 대해 다음 식으로 표현된다.

$$s_k(t) = (h_{sk}(t), h_{ek}(t)),\ t_k \le t \le t_{k+1} \quad (5)$$

단, 세그먼트의 회전에 대한 것은 표현하지 않는다.

결국 임의의 시간 t에서 MLineString의 위치 함수 $L(t)$는 식 (6)과 같이 정의되고, 반환 값은 LineString이다. $L(t)$ 함수에 대한 구현은 상위 클래스의 instanceAt( )에 대한 재정의를 통하여 이루어진다.

$$L_m(t) = (s_1(t), s_2(t), \cdots, s_n(t)),\ t_m \le t \le t_{m+1} \quad (6)$$

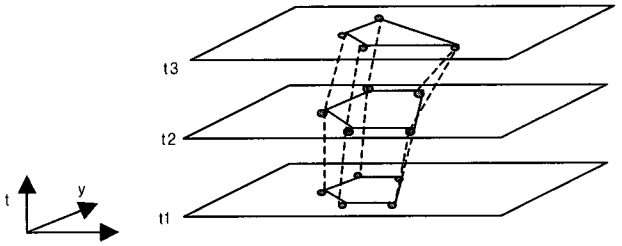
$$L(t) = L_k(t),\ t_k \le t \le t_{k+1} \quad (7)$$

MPolygon은 시간에 따라 객체의 위치 및 영역이 연속적으로 변하는 객체에 대한 모델로써 초기 시점에서의 공간적 속성이 LinearRing인 객체를 말한다. MPolygon은 태풍, 산불지역, 또는 유조선의 기름 유출과 같은 시공간 응용의 모델로 이용될 수 있다. 다음 (그림 5)는 Polygon의 시간에 따른 변화를 표현하고 있다.

Polygon은 내부에 0개 이상의 hole을 포함할 수 있으므로 MPolygon은 시작점과 끝점이 만나는 1개 이상의 MLine-String으로 이루어진다. 그러므로 임의의 시점 t에서 MPolygon 형태의 시공간 객체의 위치 및 영역을 나타내는 함수 $p(t)$는 다음과 같이 나타낼 수 있다.

$$p(t) = L_k(t),\ t_k \le t \le t_{k+1} \quad (8)$$

위치 및 영역 변화 함수 $p(t)$의 반환 값은 Polygon이 된다. MPoint의 경우와 같이 $p(t)$함수는 instanceAt( ) 메소드에서 구현된다.



(그림 5) MPolygon 객체

4.3.2 2차원 이동 객체를 위한 시공간 연산자

시공간 객체를 위한 연산자는 4.2의 (2)절에서 자세히 설명하였다. 그러나 이는 이력 객체를 대상으로 하고 있으며, 이동 객체의 경우 연속적인 변화라는 특성을 평가하고 질의하기 위해 이미 언급된 연산자들 외에도 추가적인 연산자들이 필요하다. 이동 객체에 대한 연산들은 [18]에서 다양한 타입 시스템을 기반으로 분류하고 정의되어 있다. 그러나, 이 논문에서는 시공간 이동 객체에 대한 예제 질의를 표현하기 위해 필요한 관련 연산자만을 아래의 <표 8>에서 요약한다.

〈표 8〉 이동 객체를 위한 시공간 연산자들

연 산 자	설 명
instanceAt( )	이동 객체로부터 어떤 시점에 해당하는 공간 속성을 반환
Trajectory( )	MPoint의 이동 경로를 DLineString으로 반환
Traversed( )	MPolygon의 이동 경로를 DPolygon으로 반환

여기서 instanceAt( )은 시공간 데이터에 대해 어떤 임의의 시점에 대한 공간 데이터를 얻기 위해 사용하는 연산자이고

이력 객체로부터 상속된다. 이력 객체를 위한 instanceAt( ) 연산의 알고리즘은 간단하다. 그러나 이동 객체를 위한 instanceAt( ) 연산은 임의의 시간에서의 객체 위치를 계산하기 위한 함수를 사용하여 재정의되어야 한다. 다음은 instanceAt( )을 이력 객체를 위한 알고리즘과 이동 객체를 위한 알고리즘을 비교하여 정의하였다.

```
algorithm instanceAt(vt)
input : 객체의 유효한 공간 정보를 얻어오기 위한 임의의 시점 vt
        (validtime 타입)
output : vt 시점에서 유효한 point set(Geometry 타입)
method :
instanceAt은 주어진 유효 시간과 유효한 원본 객체의 공간 속성을 검
색한다. 원본 객체인 sourceObject는 각 이력 객체이고, 이 때 적용되
는 instanceAt의 알고리즘은 동일하다.

   for each version of sourceObject do
     if a validtime of sourceObject overlaps vt then
       return a geometry of sourceObject
     else return empty point set
end instanceAt
```

(알고리즘 4) 이력 객체의 instanceAt 메소드

```
algorithm instanceAt (vt) //override operator for continuous
      // moving object
input : 유효한 공간 정보를 얻어오기 위한 임의의 시점 vt(validtime
        타입)
output : vt 시점에서 유효한 point set(Geometry 타입)
method :
instanceAt은 주어진 유효 시간과 유효한 원본 객체의 공간 속성을 검
색한다. 원본 객체인 sourceObject는 MPoint 타입이고, 이 경우 메소
드인 instanceAt은 MPoint에 대한 연산에 알맞은 알고리즘으로 정의
된다.

   for each version of sourceObject do
     if a validtime of sourceObject overlaps vt then
       set a geometry belonging to TPoint of source on A
       set a geometry belonging to the next TPoint of source on B
       //Compute a geometry for vt using the formula (4) for
   // MPoint
       result = h(A, B, vt)
       return result
   else return empty point set
end instanceAt
```

(알고리즘 5) MPoint의 instanceAt 메소드

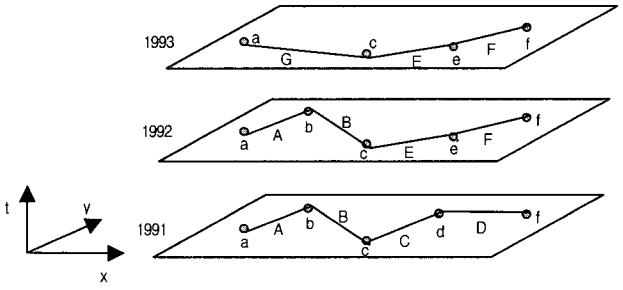
어떤 특정 시점에서 이동객체의 위상관계는 시간적 위상관계와 공간적 위상관계를 통하여 성립될 수 있다. 하지만 이동 객체는 시간에 따라 객체의 위치 또는 영역 뿐만 아니라 다른 이동 객체와의 위상 또한 시간에 따라 변하게 된다. 그러므로 이동 객체들의 위상 관계에 대한 새로운 개념의 연구가 진행되고 있다.

5. 적용 및 평가

이 장에서는 지금까지 설명한 시공간 모델을 적용하기 위해 3가지 시나리오를 제시한다. 각 시나리오는 이력 객체, 이동 객체 그리고 이력 객체와 이동 객체를 동시에 다루고 있으며 적절한 시공간 연산자를 사용하여 원하는 데이터를 추출한다.

5.1 시나리오 1 (고속도로의 건설)

일반적으로 고속도로의 건설은 구간 별로 나누어 진행되고 이미 건설된 도로도 여러 가지 이유로 인해 부분 구간을 재건설 되어 고속도로의 전체 모양이 바뀔 수도 있다. 이러한 고속도로 객체는 DLineString으로 표현될 수 있고, 각 구간은 하나의 TLine과 두개의 TPoint로 구성된다. 다음 (그림 6)은 1991년부터 1993년간의 건설 상황에 따른 Sky 고속도로의 형태가 어떻게 변하는지를 보여준다.



(그림 6) Sky 고속도로의 재건설

- 1991년 A, B, C, D구간으로 이루어진 Sky 고속도로가 완공되었다.
- 1992년 C, D 구간을 E, F로 재건설하였다.
- 1993년 A, B구간에 대하여 G구간을 재건설함으로써 고속도로의 전체 구간은 G, E, F구간으로 변경이 되었다.

(그림 6)의 데이터를 저장하기 위해 (그림 7)과 같이 Highway 테이블을 선언한다.

```
CREATE TABLE Highway (
        ID          INT         NOT NULL,
        NAME        VARCHAR (15),
        LOCATION DLineString
)
```

(그림 7) Highway 테이블의 선언

(그림 6)에서 주어진 Sky 고속도로의 시공간 속성을 나타내는 DLineString은 다음과 같이 표현된다.

```
{<a, 1991-1, NOW>, <b, 1991-1, 1992-12>, <c, 1991-1, NOW>,
 <d, 1991-1, 1991-12>, <f, 1991-1, NOW >, <e, 1992-1, NOW>,
 <ab, 1991-1, 1992-12>, <bc, 1991-1, 1992-12>, <cd, 1991-1, 1991-12>,
 <df, 1991-1, 1991-12>, <ce, 1992-1, NOW>, <ef, 1992-1, NOW>,
 <ac, 1993-1, NOW>}
```

여기서 a, b, c, d, f, e는 Point를 나타내며 ab, bc, cd, df, ce, ef, ac는 Line을 나타낸다.

예제 질의 1) "Sky 고속도로에 대한 현재 공간 정보를 검색하라"

```
SELECT h.LOCATION.instanceAt(now)
FROM Highway h
WHERE h.NAME = "Sky"
```

질의의 결과는 현재 Sky 고속도로를 구성하는 TPoint와 TLine의 공간 속성들의 집합인 LineString 형태가 된다. 즉, 현재 값 NOW와 관련된 TPoint와 TLine은 다음과 같다.

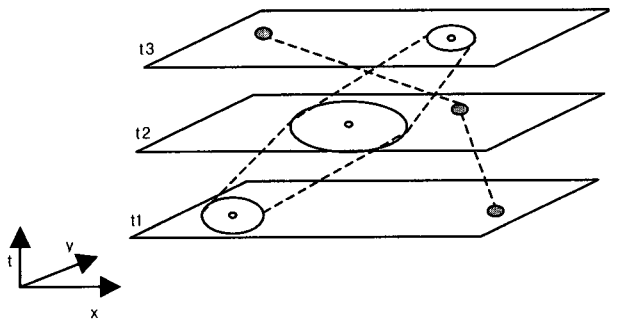
```
{<a, 1991-1, NOW>, <c, 1991-1, NOW>, <f, 1991-1, NOW>,
 <e, 1992-1, NOW>, <ce, 1992-1, NOW>, <ef, 1992-1, NOW>,
 <ac, 1993-1, NOW>}
```

그리고 위의 결과 집합으로부터 Point의 집합들로 이루어진 LineString객체는 다음과 같이 구성된다.

```
{⟨a⟩, ⟨c⟩, ⟨e⟩, ⟨f⟩}
```

5.2 시나리오 2(항공 관제 시스템의 비행기 위치 관리)

항공 관제 시스템의 경우 운항 중인 비행기의 위치를 지속적으로 파악하고 있어야 한다. 이런 경우 시간의 특정 시점에서 비행기의 위치들이 기록되어 지고 기록되지 않은 시간 사이에서도 비행기는 지속적으로 움직이고 있다고 가정한다. 이와 같이 시간에 따라 위치가 연속적으로 변하는 비행기와 같은 객체는 MPoint로써 표현되어질 수 있다. 또한 비행기의 운항은 안개 지역 또는 태풍의 영향권과 같은 기상 상태와 밀접한 관계를 갖으며, 여기서 태풍에 대한 영향권은 MPolygon으로 표현 될 수 있다. 즉 생성 단계로부터 소멸에 이르기까지 연속적으로 위치와 영역이 변하게 된다. (그림 8)은 서울에서 출발한 비행기가 태풍 SARA의 영향권을 지나는 경로를 보여준다.



(그림 8) 비행기와 태풍 SARA의 이동 경로

데이터 베이스에서 비행기와 태풍에 대한 정보를 저장하기 위해 (그림 9)와 같이 테이블을 선언한다.

```
CREATE TABLE Airplane (
        ID          INT  NOT NULL,
        NAME        VARCHAR (15),
        LOCATION    MPoint
)
CREATE TABLE Typhoon (
        ID          INT  NOT NULL,
        NAME        VARCHAR (15),
        RANGE       MPolygon
)
```

(그림 9) Airplane 테이블과 Typhoon 테이블의 선언

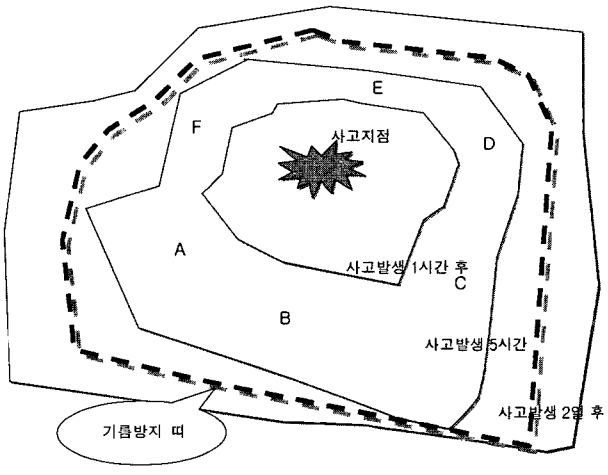
예제 질의 2) "태풍 SARA(ID : 24)의 영향권을 지나는 비행기를 검색하라".

```
SELECT a.ID
FROM   Airplane a, Typhoon t
WHERE  Traversed(t.RANGE).tIntersects(Trajectory
       (a.LOCATION)) AND t.ID = 24
```

위의 질의문에서 태풍 SARA가 이동한 궤적을 구하기 위해 Traversed 연산이 사용되었고, 비행기의 이동 경로를 구하기 위해 Trajectory 연산이 사용되었다. 태풍과 비행기의 궤적은 각각 DPolygon과 DLineString으로 반환되고, 이 두 객체에 대해 시공간적 교차는 태풍과 비행기의 이동 경로의 교차를 의미한다.

5.3 시나리오 3(유조선 사고 관리)

(그림 10)은 유조선의 사고와 사고 지점으로부터의 기름 확산 모습을 보여준다.



(그림 10) 유조선 사고

유조선에서 기름 유출 사고가 일어나면 손실 비용 및 환경

피해를 최소화하기 위해 유출된 기름의 처리를 위한 신속한 대처가 필요하다. 이 때 기름 유출 범위는 시간에 따라 계속 확산되므로 MPolygon으로 표현할 수 있고, 사고 위치나 사고 수습 기관인 해양 경찰서의 위치는 DPoint와 같은 이력 객체로 나타낼 수 있다. 이러한 데이터를 저장하기 위해 (그림 11)과 같이 테이블들을 선언한다.

```
CREATE TABLE Accident (
        ID          INT  NOT  NULL,
        LOCATION  DPoint
)
CREATE TABLE Station (
        ID          INT  NOT  NULL,
        NAME       VARCHAR (15),
        RANGE      DPoint
)
CREATE TABLE Pollution (
        ID          INT  NOT  NULL,
        AID         INT,
        RANGE      MPolygon
)
```

(그림 11) Accident와 Station 테이블 선언

예제 질의 3) "1995년 8월 24일 오전 3시에 발생한 기름 유출 사고(ID=19950824)에 대해, 사고시간으로부터 1시간 후의 기름 확산 해역을 표시하라".

```
SELECT p.ID, p.RANGE.instanceAt(4AM Aug 24, 1995)
FROM Accident a, Pollution p
WHERE a.ID = p.AID AND a.ID = 19950824
```

오염 구역은 이동 영역이기 때문에 주어진 시점에서의 해역을 구하기 위해 사용한 instantAt( )는 예제 질의 1에서 사용된 것과는 달리 식 (8)을 이용하여 재정의된 연산자이다.

예제 질의 4) "사고 지점과 가장 가까운 해양경찰서를 검색하라"는 질의는 다음과 같다.

```
SELECT s.ID
FROM Station s, Accident a
WHERE min(s.LOCATION.tDistance(a.LOCATION)) AND
        a.ID = 19950827
```

이 질의를 수행하기 위해 2가지 연산자가 사용되었다. tDistance ( )는 현재 유효한 데이터 중 사고 지점과 모든 해양 경찰서들과의 최단 거리를 구하고, 계산된 거리 중 가장 작은 값을 찾음으로써 사고지점과 가장 가까운 해양 경찰서를 찾아낼 수 있다.

지금까지의 예제 시나리오에서 보인 바와 같이 시간에 따라 변하는 공간 요소의 표현은 현재 데이터뿐만 아니라 과거로부터의 이력을 관리함으로써 정책 수립 및 수행에 응용할 수 있다. 세 가지의 시나리오 외에도 행정 구역이나 그린벨트 구역의 관리, 소방 관제, 이동 서비스 관리 그리고 군사 작전 시뮬레이션 등의 다양한 분야에 적용할 수 있다.

5.4 평가

지금까지 많은 시공간 모델이 제안되어 왔으며 각각은 적용 분야에 대해 나름대로의 장단점을 가지고 있다. 그러나 제 5장에서 제안한 세 가지 시나리오는 이 논문에서 제안한 모델은 기존의 시공간 모델들과 4가지 차이점을 보여준다.

첫째, 기본 공간 요소에 대한 타임스탬프는 공간에 대한 다양한 질의처리를 가능하게 하였다. 시나리오 1에서 보인 바와 같이 하나의 고속도로를 구성하는 요소들에 대해 연도별 건설 구간 또는 이미 폐쇄된 구간 등을 각각 검색할 수 있다. 그러나 하나의 객체에 타임스탬프를 할당하는 경우에 비해 훨씬 더 많은 저장 공간을 요구하며 이러한 문제를 해결하기 위해 적절한 저장 구조 및 데이터 접근 방식에 대한 연구가 필요하다.

둘째, 이력 객체와 이동 객체를 함께 관리함으로써 실세계의 객체들을 적절하게 표현할 수 있게 하였다. 이력 객체나 이동 객체 중 한 가지에만 초점을 맞춘 기존의 시공간 모델로는 실세계의 다양한 객체를 정의할 수 없다. 여러 응용 분야에 적용하기 위해 두 종류의 시공간 객체를 모두 지원함으로써 시나리오 3에서처럼 유조선이나 기름띠와 같은 이동 객체와 해양경찰서나 사고 지점과 같은 이력 객체를 동시에 관리하는 것이 가능하였다.

셋째, 다양한 시공간 연산을 지원함으로써 강력한 시간 및 공간 분석력을 지원하였다. 저장된 데이터를 효율적으로 분석하고 질의하기 위해서는 다양한 연산이 필요하다. 각 시나리오에서 제시된 예제 질의들에서 획득하고자 하는 데이터는 여기서 제안한 여러 시공간 연산들을 통해 적절하게 얻을 수 있었다. 지금까지의 시공간 모델들은 이러한 시공간적 관계성 및 위상 분석과 같은 연산자에 대한 정의가 언급되어 있지 않았다. 그러나 이 논문에서 제안하는 시간의 할당 방식으로 인해 시공간 연산 비용이 크게 증가할 수 있으며, 이러한 단점을 해결하기 위해 연산자의 효율적인 알고리즘 제시 및 최적화가 요구된다.

마지막으로 이 논문에서 제안한 모델은 표준 모델로써의 운용이 가능하도록 설계하였다. 각 시나리오들은 서로 상이한 목적을 갖는 시스템에서 사용되며 이러한 시스템들에서 방대한 양의 데이터를 상호 운용하기 위해 통합 모델의 필요성이 요구된다. 이를 위해 개방형 GIS 컨소시엄에서는 국제 표준으로써 2차원 공간 모델을 제시하였다. 이러한 표준 공간 모델을 기반으로 시공간 모델을 정의하여 통합 모델로서

의 필요조건을 만족시키고 있으며 새로운 시공간 표준 모델로써의 활용이 가능하다.

6. 관련 연구와의 비교

시간 데이터와 공간 데이터를 동시에 처리하기 위한 시공간 모델을 정의는 시간 모델로부터 시공간으로 일반화하는 방법과 공간 모델로부터 시공간으로 일반화하는 방법이 있다. 이 논문에서 제안하는 시공간 모델은 두 번째 방법을 사용하여 시공간 모델을 정의한다.

[19]도 공간 모델로부터 시공간 모델로 확장하여 정의하였다. 이 모델에서 시공간 객체(ST-Object)는 공간과 이원 시간 영역을 모두 가지고 있는 유일한 객체이고, 이를 기반으로 시공간 연산자들이 제공된다. 특히 [19]에서는 객체지향 개념을 처음으로 시공간 모델링에 도입하고 있는데, 이러한 객체지향 개념은 동일한 객체의 이력들을 하나의 단일 엔티티에 내장 시키는 것을 가능하게 하였다. 그러나 이 모델은 우선 기본적으로 제한된 연산 집합을 갖는 ST-complex라는 단일 타입만을 사용함으로써 표현력이 제한되었고 0, 1, 2-simplex 형태의 새로운 공간 객체 타입을 제안하여 표준 모델과의 호환을 보장하지 않았다. 제안하는 모델은 표준 모델을 기반으로 확장하여 풍부한 표현력과 호환성을 제공한다.

최근에는 시간에 따라 변하는 공간객체, 즉 일반적 의미로서 시공간 객체를 이동 객체의 개념을 이용하여 시공간 데이터베이스를 모델화하려는 접근 방법으로 움직이는 객체를 다루기 위한 연구가 진행되고 있다[20, 21]. 이동 객체의 경우 [19]에서 설명한 바와 같이 정의된 시공간 객체를 그대로 적용할 경우, 매우 빈번한 갱신과 대용량의 저장 공간, 갱신 오류 등의 문제를 유발할 수 있다. 따라서 이동 객체의 모델링에 대한 여러 연구들은 시공간 객체를 구성하는 버전들의 관계를 시간에 대한 함수로 정의하며, 이 논문에서도 이와 같은 방법을 사용하여 이동 객체를 이력 객체와 함께 정의하고 있다.

[22]의 이동 객체 모델 역시 함수를 사용하여 정의된다. [19]의 방식에 따라 공간 객체를 시간에 따른 버전으로 표현하고 버전들간의 관계를 표현하기 위해 행위 함수(behavioral function)를 사용했다. 시간을 공간에 대한 하나의 축으로 가정하여 2차원 시공간 객체를 3차원 객체로 표현하고 이를 바탕으로 연산자를 설계했다. 행위 함수는 미리 정의된 형태의 규칙 함수, 계단 함수, 선형 함수, 그리고 보간 함수 등이 존재하며, 특히 일반적인 이력 객체는 계단 함수와 함께 저장된다. 이와는 달리 이 논문에서의 모델은 이동 객체의 타입에 따라 위치 추출 함수를 정의하고, 이력 객체의 저장 구조는 함수를 포함하고 있지 않는다.

[23]은 연속적으로 움직이고 있는 객체 즉 시간에 따라 위치 또는 위치와 모양 모두가 연속적으로 변하는 객체에 대한 표현을 위하여 '단편화된 표현(sliced representation)' 개념을 도입하고 시간 간격(interval)과 그 시간 간격에 대해 정의된 단순 함수(simple function)의 쌍을 갖는 단위 타입(unit type)을 정의했다. 그러나 이 논문에서는 각 좌표 값에 대한 함수를 정의하고 객체의 형태에 따라 응용하는 방법을 사용한다. 또한 기본적으로 이 모델은 데이터 타입 접근 방법으로 이 논문에서 제안하는 객체 지향 모델링과는 접근 방법이 다르다. [23]과 같이 시공간 객체 추상형 데이터 타입을 정의하고 기존의 데이터베이스에 이 타입들을 확장하는 방법도 다양하게 연구되고 있다.

이 외에도 과거 데이터를 대상으로 하는 모델들과는 달리 현재와 예측할 수 있는 가까운 미래의 데이터를 대상으로 이동 객체를 표현하고자 하는 연구도 진행되고 있다. [21]에서 이동 객체 데이터베이스를 위해 제안한 MOST(Moving Object Spatiotemporal) 모델은 위치보다 덜 빈번하게 변하는 이동 정보 즉 속도, 방향의 변화를 데이터베이스에 저장함으로써 미래의 위치 추정을 가능케 한다.

7. 결 론

실세계의 개체들은 실제로 공간적인 정보 뿐만 아니라 시간적 정보와도 연관을 갖는다. 그렇기 때문에 현실 세계의 시간과 공간 개념이 동시에 요구되는 데이터들을 효율적으로 관리 할 수 있는 시스템이 요구되고 있으며, 이를 위해서는 시공간 데이터에 대한 모델링이 선결되어야 한다. 이 때, 시공간 객체는 공간 속성이 시간에 따라 불연속적으로 변하는가, 위치는 연속적으로 변하는가, 또는 위치와 모양이 연속적으로 변하는가에 따라 분류할 수 있으며, 이런 방식으로 분류된 시공간 객체는 각각 다른 방식의 해석 방법이 필요하다.

이 논문에서 실세계의 시간과 연관된 공간 객체를 객체 지향이라는 패러다임을 이용하여 대상 이력 객체와 이동 객체를 동시에 표현할 수 있는 통합 모델을 설계하였다. 제안된 모델은 개방형 GIS에서 정하고 있는 2차원 공간객체 모델을 기본 공간모델로써 사용하였으며, 유효시간만을 고려한 시간 모델에 따른 시간 클래스와 시간 관계 분석 연산자를 정의하고 있다. 그리고 이력 객체간의 시공간 관계성을 위한 시공간 연산자들과 이동객체 자체 내에 적용될 수 있는 시공간 연산자들에 대하여 정의하였다.

향후에는 제안한 모델의 연산 비용 개선을 위한 연산 알고리즘 및 데이터 접근 방법의 최적화가 수행되어야 한다. 그리고 이동 객체에 대하여 객체의 특성을 고려한 객체간 관계성 분석의 연구와 이에 기반한 이동 객체간의 시공간 연산자

를 추가함으로써 제안된 모델링에 대하여 표현력을 증가시킬 수 있다. 아울러 응용에 따른 시간 크기의 적용이 추가적으로 보완 연구가 필요하다.

참 고 문 헌

[1] M. Yuan, "Modeling Semantical, Temporal, and Spatial Information in Geographic information Systems," M. Craglia and H. Couclelis eds. Geographic Information Research : Bridging the Atlantic, Taylor & Francis, pp.334-347, 1996.

[2] C. Claramunt and M. Thuriault, "Managing Time in Gis : An Event-Oriented Approach," Recent Advances on Temporal Database, J.Clifford and A.Tuzhilin eds., Springer-Verlag, Zurich, pp.23-42, 1995.

[3] D. H. Kim, K. H. Ryu, H. S. Kim, "A Spatiotemporal Database Model and Query Language," International Journal of Systems and Software, 2000.

[4] D. H. Kim, K. H. Ryu, C. H. Park, "Design and Implementation of A Spatiotemporal Query Processing System," International Journal of Systems and Software, 2001.

[5] N. Tryfona and C. S. Jensen, "Conceptual data modeling for spatiotemporal applications," Chorochronous Technical Report, No.CH-98-08, September, 1998.

[6] A. Pavlopoulos, B. Theodoulidis, "Review of SpatioTemporal Data Models," TimeLab Technical Report, TR-98-3, October, 1998.

[7] M. E. Adiba, and B. G. Linsay, "Database snapshots," Proceeding of the Conference on Very Large Databases," pp.86-91, October, 1980.

[8] M. H. Bohlen, R. Busatto, and C. S. Jensen, "Point-Versus Interval-Based Temporal Data Models," Proceedings of International Conference on Data Engineering, 1998.

[9] D. J. Peuquet and N. Duan, "An Event-Based Spatiotemporal Data Model (ESTDM) for Temporal Analysis of Geographical Data," International Journal of Geographical Information Systems, Vol.9, No.1, pp.7-24, 1995.

[10] A. Renolen, "History Graphs : Conceptual Modeling of Spatio-temporal Data," In GIS Frontiers in Business and Science, Vol.2, International Cartographic Association, Brno, Czech Republic, 1996.

[11] Open GIS Consortium, Inc. OpenGIS "Simple Features Specification For OLE/COM Revision 1.1," OpenGIS Project Document 99-050, 1999.

[12] R. Snodgrass, and I. Ahn, "A taxonomy of time in databases," Proceeding of ACM SIGMOD International Conference on Management of Data, pp.236-246, May, 1985.

[13] F. J. Allen, "Maintaining Knowledge About Temporal Intervals," Communications of the ACM 26, pp.832-843, 1983.

[14] M. F. Worboys, H. M. Hearhshow, D. J. Maguire, "Object-Oriented Modeling for Spatial Database," Int. journal of GIS Vol.4, No.4, 1990.

[15] 이현아, 임헌기, 김영일, 남광우, 류근호, "3D+Temporal 시공간 객체 모델링", 한국정보처리학회 추계 학술발표논문집, 제7권 제2호, pp.89-92, 2000.

[16] 강구, 김상호, 류근호, "컴포넌트 기반의 2차원 시공간 위상관계 연산자 설계", 한국정보과학회 추계 학술발표논문집, 제28권 제1호, pp.79-81, 2001.

[17] 조영소, 김동호, 류근호, "시공간 데이터 모델에서 시공간 연산자의 관계 수식적 정형의미", 한국정보처리학회논문지, 제6권 제1호, pp.11-19, 1999.

[18] R. H. Guting, M. H. Bohlen, M. Erwig, C. S. Jensen, et al, "A Foundation for Representing and Querying Moving Object," Chorochronos TR, CH-98-3, 1998.

[19] M. F. Worboys, "A Unified Model for Spatial and Temporal Information," The Computer Journal Vol.37 No. 1, pp.36-34, 1994.

[20] M. Erwig, R. H. Guting, M. Schneider and M. Vazirgiannis, "Spatio-Temporal Data Types : An Approach to Modeling and Querying Moving Objects in Databases," Chorochronos Technical Report CH-97-8, 1997.

[21] O. Wolfson, B. Xu, S. Chamberlain, L. Jiang "Moving Object Database : Issues and Solutions," Proceedings of the Statistical and Scientific Database Management Conference, pp.111-122, 1998.

[22] T-S. Yeh, B. de Cambray, "Modeling Highly Variable Spatio-Temporal Data," PRiSM Technical Report, 1994/50, 1994.

[23] L. Forlizzi, R. Guting, E. Nardelli, and M. Schneider, "A Data Model and Data Structures for Moving Objects Databases," Proceedings of the ACM SIGMOD International Conference on Management of Data, pp.319-330, 2000.

이 현 아

e-mail : halee@dblab.chungbuk.ac.kr

1999년 충북대학교 전기전자공학부 졸업(공학사)

2000년~현재 충북대학교 대학원 컴퓨터과학과 석사과정

관심분야 : 시간 데이터베이스, 시공간 데이터베이스, 객체지향 데이터베이스, GIS 등

남 광 우

e-mail : kwnam@dblab.chungbuk.ac.kr

1993년 충북대학교 컴퓨터과학과 졸업 (이학사)

1995년 충북대학교 대학원 컴퓨터과학과 (이학석사)

1997년~현재 충북대학교 대학원 컴퓨터과학과 박사과정

관심분야 : 공간 데이터베이스, 시간 데이터베이스, 시공간 데이터베이스, Temporal GIS 등

류 근 호

e-mail : khryu@dblab.chungbuk.ac.kr

1976년 숭실대 전산학과 졸업(공학사)

1980년 연세대학교 산업대학원 전산전공 (공학석사)

1988년 연세대 대학원 전산전공(공학박사)

1976년~1986년 육군군수지원사전산실 ROTC장교, 한국전자통신연구소 연구원, 한국방송통신대 전산학과 조교수

1989년~1991년 Univ. of Arizona Research Staff(TempIS 연구원, Temporal DB)

1986년~현재 충북대학교 전기전자 및 컴퓨터공학부 교수

관심분야 : 시간 데이터베이스, 시공간 데이터베이스, Temporal GIS, 객체 및 지식기반 시스템, 지식기반 정보검색 시스템, 데이터 마이닝, 데이터베이스 보안 및 Bio-Informatics 등