2D+Temporal 시공간 연산자의 설계 및 구현†

이진관° 김영삼 남광우 류근호
충북대학교 전자계산학과
{jklee,yskim,kwnam,khryu}@dblab.chungbuk.ac.kr

Design and Implementation of 2D+Temporal Spatio-Temporal Operators

Jin kwan Lee° Young Sam Kim Kwang Woo Nam Keun Ho Ryu
Dept. of Computer Science, Chungbuk Natioinal University

요 약

실세계의 객체들은 공간 정보뿐만 아니라 시간적 정보와도 연관을 갖는데 기존의 공간데이터베이스만으로는 시간 흐름에 따른 공간 객체의 정보를 효율적으로 관리해 주지 못하는 단점이 있다. 이러한 문제점을 해결하기 위하여 시공간 데이터베이스에 대한 데이터 모델과 시공간 연산자가 제시되었다. 그러나, 시공간 연산자에 대한 정의와 시그너처정도 만이 기술되었고 시공간 연산자에 대한 설계와 구현에 대한 사항은 제시되지 않았다. 이 논문에서는 시공간 데이터 모델과 시공간 연산자 그리고 공간 연산자 구현 기법인 Plane-Sweep 기법을 이용한 시공간 연산자인 Trajectory와 MPIntersection에 대한 설계와 구현 알고리즘을 제시하였다.

1. 서론

실세계의 객체들은 실제로 공간적인 정보 뿐만 아니라 시간적 정보와도 연관을 갖는다. 즉, 이러한 객체들의 공간상에서의 위치 또는 영역에 관한 정보는 시간에 따라 변하게 된다. 오직 현재 시점에서 유효한 공간 정보만을 처리 대상으로 하는 기존의 공간 데이터베이스는 객체에 대한 공간상의 변화가 발생하면 이전의 값을 삭제하고 현재시점의 값으로 대치한다. 이것은 시간의 흐름에 따라 객체에 저장된 공간 정보에 대한 효율적인 이력 관리가 매우 어려운 문제점을 갖는다.

이와 같은 문제점을 해결하기 위하여 시공간 데이터베이스는 현실세계에 존재하는 다양하고 복잡한 객체에 대하여 효율적인 공간 정보를 제공할 뿐만 아니라 시간의 흐름에 따라 변화하는 공간정보에 대한 이력을 동시에 효율적으로 관리해 준다. 이러한 시공간 데이터베이스에서 다루는 시공간 객체는 불연속적으로 움직이는 이력객체(discretely moving object) 와 연속적으로 움직이는 이동객체(continuously moving object) 로 구분[6]되는데 이 논문에서는 이동객체의 시공간 데이터 모델[5]에서 제시한 모델을 바탕으로 정의된 시공간 연산자 중에서 한 점의 이동 궤적을 구하는 trajectory 연산자를 설계하고 구현 알고리즘을 기술한다. 또한 두 이동점의 교차 여부를 연산하는 새로운 시공간 연산자인 MPIntersection에 대한 정의를 내리고 그 설계와 구현 알고리즘을 제시한다.

제2장에서는 시공간 데이터 모델과 시공간 연산자 그리고 공간 연산자 구현 기법이Plane-Sweep 기법을 관련연구로 기술하고, 제3장에서 2D+ Temporal 시공간연산자인trajectory 와 MPIntersection에 대한 설계와 구현 알고리즘을 설명한다. 제4장에서는 지금까지의 연구 결과와 향후 연구방향에 대해 기술한다.

2. 관련연구

2.1 시공간 데이터 모델

일반적으로 시간에 관계된 공간 객체를 시공간 객체(spatiotemporal object)라 하고 이것은 불연속적으로 움직이는 이력 객체(discretely moving object) 와 연속적으로 움직이는 이동객체(continuously moving object)로 구분된다[6]. 기 제시된 시공간 데이터 모델 중 몇몇 모델의 접근 방식에 따른 특성은 표1과 같다.

표 1. 시공간 데이터 모델

모델명	특징
시공간 복합 데이터 모델[1]	다수의 스냅샷 레이어를 하나의 시공간 복합레이어에 중첩시키는 방법으로 동일 이력을 갖는 공간객체의 구분이 불가능하다.
시간 스탬핑 모델[2]	각 객체에 대해 객체의 생성 및 삭제에 대한 시간 스탬프 쌍을 부여하여 객체의 시간정보를 관리.
사건 기반 시공간 데이터 모델[3]	각각의 사건을 리스트로 구성하고, 각 사건에 대한 시간 스탬프, 공간 속성 변화, 공간 위치 변화를 링크트 리스트로 구성.
이력 그래프 모델[2]	공간데이터 집합에서의 변화의 성질을 식별하기 위하여 시간 객체의 이력을 발생시키는 객체의 생성, 변경, 삭제, 재생, 병합, 분할의 여섯가지 형태로 이벤트를 구분하며, 이력객체간의 파생관계를 직접 표현할 수 있다.
삼원 모델[4]	의미, 공간, 시간영역을 서로 다른 테이블로 구성하고 이들간의 연결 링크를 제공함으로써 공간정보의 처리 및 현상을 표현

† 이 연구는 한국전자통신연구원 "4D 시공간 데이터제공자 콤포넌트 개발" 및 한국자원연구소 "3D GIS 인터넷 지원 공간 데이터관리 시스템 개발" 연구비 지원에 의함.

시공간 객체지향 모델[2]	객체, 클래스, 상속, 다형화등의 객체지향 패러다임을 기본으로 동일객체의 모든 이력 버전을 하나의 단일 개체내에 포함되도록 한다.
이동객체의 시공간 데이터 모델[5]	추상적 데이터 타입을 이용하여 이동 영역과 이동점에 대한 데이터 타입을 정의하고 각각의 데이터 타입은 점, 영역등의 개체와 함께 이동연산을 포함하며 시간요소는 공간객체에 통합됨으로써 변경 및 이동을 동시에 표현

지금까지의 시공간 데이터 모델들은 동일한 이력을 갖는 이력 객체간의 구별 문제[9], 동일한 객체의 이력의 중복 관리문제[3,9], 단일 객체의 이력 객체간 식별자가 서로 다르거나 이력 관리 단위가 지나치게 큰 문제[9]를 가지고 있다.

2.2 시공간 연산자

시공간 연산은 시공간 관계형 연산, 시공간 위상 관계 연산, 그리고 시공간 기하 연산으로 구분할 수 있다[10].

시공간 관계형 연산은 데이터베이스에 저장된 시공간 객체를 선정(selection)하거나 객체로부터 임의의 속성을 추출(projection)하거나 객체간 결합(union), 교차(intersection), 차분(difference0, 카테시안 곱(Cartesian product), 죠인(join) 및 제산(division) 연산을 수행한다. **시공간 위상 관계 연산**은 주어진 객체들에 대하여 시간 영역과 공간 영역상의 관계에 대하여 참/거짓을 반환한다. 시간 영역상의 객체간 관계는 객체의 공간 좌표값에 의해 결정된다. 대표적인 시공간 위상 관계 연산으로 공간 위상 관계 연산에는 contains, equal, intersect 등이 있고 시간 위상 관계 연산에는 precede, overlap, equal 등이 있다. **시공간 기하 연산**은 주어진 객체 또는 객체들로부터 시간 영역과 공간 영역을 기반으로 기하학 수치 값을 반환하거나 객체 자체의 시공간 정보를 변경하는 기능을 갖는다. 대표적인 공간 기하 연산에는 direction, distance, length 등이 있으며, 시간 기하 연산에는 overlap, extend 등이 있다.

[5]는 이동객체에 적용하기 위한 시공간 데이터 타입을 정의하였다. **이동객체를 위한 두개의 기본 타입**은 *mpoint*와 *mregion*이 있는데 *mpoint*는 time -> point로 표현되고, 여기서 time은 valid time 이다. 시간의 함수로서 위치를 기술하는 타입 mpoint의 값은 3 차원 공간에서 curve 로 표현된다. *mregion*은 time -> region 로 표현되며, 여기서의 time 도 valid time 이다. 타입 mregion의 값은 3 차원 공간에서의 볼륨의 집합이다. 위 정의된 두개의 타입과 point, time, real 을 사용하여 표현한 **이동객체를 위한 연산자들**은 다음 표2와 같다.

표 2. 이동객체를 위한 시공간 연산자

연산자 명	설명
At	특정 시간 포인트에서의 이동객체의 값을 리턴
Minvalue, maxvalue	이동객체의 최소와 최대값을 리턴
Duration	이동객체가 정의된 시간interval 의 전체 길이를 리턴
Const	시공간 객체의 생성
Mdistance	두개의 이동점 사이의 거리를 계속해서 계산하고 실수시간을 리턴
Visits	이동점이 이동 region 의 내부에 있을 때의 위치를 리턴
Trajectory	이동점을 평면상에 projection
Traversed	이동 region 을 평면상에 projection
Inside	점이 region 내부에 존재하는지를 검사
Length	line의 전체 길이를 리턴

2.3 Plane-Sweep 기법

Plane-Sweep 기법은 기하학적 집합 문제인 이차원 문제를 해결하기 위해 이차원 문제를 더 단순한 일련의 일차원 문제로 변환하여 해결하는 방식으로 연산 기하에서 많이 사용한다[6,7]. vertical sweep line은 평면을 좌에서 우로(또는 역으로) sweeping 하며 event point 에서 멈춘다. 이 event point는 일반적으로 큐에 저장되는데 event point schedule 이라 불린다. Plane-Sweep 기법에서 실제로 공간 연산은 event point 에서 이루어 진다. sweep line status 는 현재의 sweep line 위치에서 sweep 되는 기하 구조와 sweep line 과의 교차 상태를 자료구조에 수직 순서(vertical order)로 저장한 것이다. Sweep line 이 event point 와 만날 때 sweep line status 가 갱신되는데 sweep line 을 지나가는 event point 는 sweep line status 에서 제거된다. 일반적으로 효율적이고 완전히 동적인 자료구조가 event point schedule을 표현하기 위해 필요하다.

Guting[8]은 Plane-Sweep 기법을 이용한 공간 연산자 구현을 두 가지로 분류했다. 첫번째는 points 와 lines 객체와 regions 객체 사이의 관계를 나타내는 것인데, 여기서 사용된 알고리즘 기법은 regions 객체의 세그먼트들을 sweep line status 에 삽입하고 points 와 lines 객체의 원소들을 각각 질의 원소로 사용하는 것이다. 두 번째는 두개의 regions 객체의 관계를 고려하는데 overlap number 와 세그먼트 분류 개념을 사용한다. 즉 하나의 세그먼트는 overlap number 의 쌍 (m/n) 과 관련된다. 여기서 m 은 낮은(또는 오른쪽)을 n 은 높은(또는 아래쪽)을 나타낸다.

3. 2D+Temporal 시공간 연산자 설계 및 구현

이 절에서는 시공간 연산자중 이동점의 trajectory 와 Plane-Sweep 기법을 이용하여 두 이동점들의 intersection 연산에 대한 알고리즘을 제시한다. 먼저 이 알고리즘들을 기술하기 위한 가정과 제한사항, 그리고 이 알고리즘들을 지원하기 위한 연산들의 집합에 대해 정의 한다.

이동객체의 위치는 일정한 시간 간격(예를 들면, 5분)을 가지고 데이터베이스에 그 위치가 저장되며, 많은 이동객체에 대해 관리를 할 경우 실제 각 이동객체에 대한 위치가 저장되는 시간은 약간의 지연이 발생할 수 있다. 이동 객체들은 테이블에 저장되며 이 테이블에서 키는 (*mpid, v_time*)이며, 테이블의 구조는 표3과 같다.

표 3. 하나의 이동객체에 대한 테이블

mpid	X_pos	Y_pos	V_TIME
MP1	1	10	2000-09-01 08:50:03
MP1	2	20	2000-09-01 08:55:04
...	...	...	...

3.1 trajectory

이동점을 평면상에 projection 하는 연산으로 알고리즘은 다음과 같은 기본 단계로 이루어 진다.

1. 주어진 이동객체의 테이블에서 주어진 시간 범위 이내에 포함되는 튜플들을 선택한다.(단, 시간 범위가 주어지지 않으면 주어진 테이블 내의 모든 튜플들을 선택한다.)
2. 위 1 에서 얻어진 튜플들의 좌표를 평면상에 display 한다.
3. 위 1 에서 얻어진 튜플 집합을 반환한다.

다음은 trajectory 연산의 알고리즘이다.

Function Trajectory(mpid, S_time, E_time)

Input: 하나의 이동점ID(mpid)와 trajectory 를 구하고자 하는 시간범위의 시작 시간(S_time)과 끝시간(E_time)

Output: 하나의 이동점의 궤적을 나타내는 point set

Method: GetPointsFromDB는 주어진 mpid가 저장된 테이블로부터 Stime과 Etime의 범위 내에 존재하는 튜플들을 ResultSet으로 반환한다.

```
Begin
  /* Initialize */
  PointSet.points ← null, Vector RSet ← null, int i ← 0
  /* Get tuples of given mpid from DB */
  RSet = GetPointsFromDB(mpid, Stime, E_time)
  while(RSet.hasNext()) do
    PointSet.points(i).x = RSet.getInt()
    PointSet.points(i).y = RSet.getInt()
    PointSet.points(i).time = RSet.getString()
    i++
  end_while
  /* display the trajectory of moving point */
  PointSet.display()
  /* return the trajectory of moving point */
return PointSet
endTrajectory
```

3.2 MPIntersection

두 이동점이 주어진 시간 범위 이내에서 또는 시간 범위가 주어지지 않은 경우에는 주어진 이동점의 전체 시간에 대하여 한 점에서 만났는가를 결정하는 연산으로 MPIntersection이라 명명하였다. 이 연산을 구현하는 기본 알고리즘은 다음과 같은 단계로 이루어진다.

1. 주어진 두 이동점에 대해 주어진 시간내에서의 trajectory를 결정
2. Plane-Sweep 기법을 이용해 두 trajectory의 교차 여부를 결정
3. 교차가 발생한 경우
   - A. 두개의 이동점의 trajectory 가 하나의 포인트에서 교차한 경우: 두 포인트의 시간을 비교한다. 같으면 true 를 아니면 false를 반환한다.
   - B. 두개의 이동점의 trajectory 가 point 와 lines 교차한 경우: 포인트의 시간이 라인의 두 끝점의 시간의 중간값과의 오차가 주어진 범위내에 있으면 true를 반환 아니면 false를 반환
   - C. 두개의 이동점의 trajectory 가 두개의 라인선상에서 교차가 발생한 경우: 두개의 라인의 시작점과 끝점의 시간이 overlap하면 true를 반환 그렇지 않으면 false를 반환

다음은 MPIntersection의 알고리즘이다.

Function MPIntersection(mpid1, mpid2, S_time, E_time)

Input: 두개의 이동점ID(mpid1, mpid2) 와 교차가 발생했는지를 알고자 하는 시간 범위(S_time, E_time)

Output: 두 이동점이 만났으면 true를 아니면 false를 반환

Method: **Arrange**는 trajectory를 입력으로 받아서 x 와 y 의 좌표값에 따라 정렬된 값을 반환한다. **PlaneSweep**은 두개의 정렬된 trajectory를 입력으로 받아 교차되는지를 결정하고 교차점이 있으면 교차점의 시간을 비교하여 두 이동점이 만나는지를 결정하여 만나면 true 를 그렇지 않으면 false를 반환한다.

```
Begin
  /* Initialize */
  PointSet1 ← null, PointSet2 ← null, Flag ← 0
  /* Get the trajectory of two mpoint */
  PointSet1 = Trajectory(mpid1, S_time, E_time)
  PointSet2 = Trajectory(mpid2, S_time, E_time)
  if( PointSet1 && PointSet2) do
    /* sort trajectory by (x, y) order */
    NewPointSet1 = Arrange(PointSet1)
    NewPointSet2 = Arrange(PointSet2)
      /* determine the intersection of two trajectory */
      Flag = PlaneSweep(NewPointSet1, NewPointSet2)
  endif
  return Flag
  endMPIntersection
```

4. 결론 및 향후 연구과제

시공간 연산자는 시간 연산과 공간연산의 조합이라 할 수 있다. 따라서 본 논문에서는 시간연산은 TSQL2 에서 제시된 연산을 사용하였고 공간연산은 PlaneSweep 기법을 기반으로 하였다. 기 제시된 시공간 연산자중 이동점의 궤적을 구하는 trajectory 와 두 이동점이 주어진 시간 내에서 만나는가를 결정하는 MPIntersection 연산자를 PlaneSweep기법을 이용하여 그 알고리즘을 제시하였다.

앞으로 두 이동점 사이의 거리를 구하는 mdistance와 이동 regions 의 궤적을 구하는 연산자인 traversed 그리고 이동 점과 이동 regions 에 관계를 나타내는 visits, inside 에 대한 구현 알고리즘을 제시해 나갈 것이다.

참고 문헌

[1] G. Langran, "Time in Geographical Information Systems", ed. Taylor & Francis, London, 1992.

[2] A.Renolen, "History Graphs: Conceptual Modeling of Spationtemporal Data", In GIS Frontiers in Business and Science, Vol.2, International Cartographic Association 1996 .

[3] Donna J. Peuquet and Miu Duanm "An event-based spatiotemporal data model(ESTDM) for temporal analysis of geographic data ", Int. Journal of Geographic Information Systems, 1995 .

[4] M. Yuan, "Temporal Gis and Spatio-Temporal Modeling ", 3 rd Int. Conference/Workshop Intergrating GIS and Environmental Modeling, Jan 21-25, 1996.

[5] M.Erwig, R.H.Guting, M.Schneider, and M.Vazirgiannis, "Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases" , GeoInformatica, 1999.

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

[7] Edward P.F.Chan & Jimmy N.H.Ng, "A General and Efficient Implementation of Geometric Operators and Predicates", Proceedings of the 5th ACM International Symposium on Advances in Spatial Databases, Berlin, , 1997.

[8] R.H.Guting, T.D.Ridder, Markus Schneider, "Implementation of the ROSE Algebra: Efficient Algorithm for Realm-Based Spatial Data Types", Proc. of the 4th Intl. Symposium on Large Spatial Databases, Portland, 1995.

[9] 백주연, 이성종, 류근호, "고딕 GIS 도구를 이용한 시공간 객체의 표현과 이력관리", 한국정보과학회 논문지: 데이터베이스 통권 제27권 제1호, 2000년 3 월.

[10] 김동호, "시공간 데이터베이스 질의처리 시스템', 충북대학교 전자계산학과 박사학위논문, 1999년 2 월.