위치 기반 서비스를 위한 시공간 데이터모델에 관한 연구†

A Study on a Spatio-Temporal Data Model for Location-Based Service

정원일*, 배해영**
Warn-Ill Chung, Hae-Young Bae

요 약 차세대 무선 인터넷의 킬러 어플리케이션으로 주목받고 있는 위치기반서비스는 시간에 따른 시공간 객체의 위치 및 영역 변화에 대한 분석 기능이 필수적이다. 시공간 데이터베이스 시스템은 대용량 시공간 객체의 실시간 위치 정보를 효과적으로 저장하고 빠른 검색을 제공하는 시스템으로 그 필요성이 증가하고 있다. 또한, 시공간 데이터베이스 시스템에서는 시공간 객체의 비공간정보와 공간정보 및 시간정보를 통합 관리할 수 있고, 시간 정보와 관련된 연산을 효율적으로 처리할 수 있는 시공간 데이터모델에 대한 연구가 활발히 진행중이다. 본 논문에서는 시간 흐름에 따라 동적으로 변화하는 시공간 객체 정보들을 현재 시점의 상태와 과거의 변화 과정에 대한 정보를 효과적으로 관리할 수 있는 시공간 데이터 모델을 제안한다. 또한 제안하는 시공간 데이터 모델을 위한 다양한 시공간 연산을 설계하며, 시공간 데이터와 시공간 객체 연산의 무결성을 유지하기 위한 제약조건을 제시한다.

ABSTRACT Sptaio-temporal databases are important to store the real-time location information of large spatio-temporal objects efficiently and retrieve them rapidly. Accordingly, necessity for spatio-temporal database system that can manage spatial information, aspatial information and temporal information of spatio-temporal objects is increasing. Sptaio-temporal databases are important to store the real-time location information of large spatio-temporal objects efficiently and retrieve them rapidly. Accordingly, necessity for spatio-temporal database system that can manage spatial information, aspatial information and temporal information of spatio-temporal objects is increasing. Therefore, in this paper, we propose a spatio-temporal data model that is able to efficiently manage historical spatio-temporal objects that change dynamically their states as time. Also, various spatio-temporal operations and constraint conditions are defined to keep integrity of spatio-temporal data and spatio-temporal operations.

주요어 : LBS, 시공간 데이터모델, 시공간 데이터베이스
Key word : LBS, Spatio-Temporal Data Model, Spatio-Temporal Database

1. 서 론

최근 이동 전화와 휴대용 단말과 같은 이동식 전자장치의 보급이 확산되고 위치기반서비스에 대한 사용자의 요구가 점차 커지고 있다. 이러한 위치 기반 서비스의 주요 응용 대상인 차량, 휴대용 전화기, 노트북, PDA등은 모두 자유롭게 이동하면서 그 위치가 실시간으로 변화되는 특징을 가진다. 이와 같이 시간의 흐름에 따라 객체가 이동하면서 그 위치와 모양을 연속적으로 변경하는 특징을 가지는 데이터를 시공간 객체[1,13,16]라 하며, 위치기반서비스에서는 시공간 객체의 위치 정보를 효과적으로 관리하는 것이 필수적이다. 특히, 이러한 시공간 객체는 그 정보량이 매우 방대하게 증가하므로 대용량 실시간 시공간 객체를 관리하기 위한 시공간 데이터베이스시스템에 대한 연구가 수행되고 있으며[3,12], 이러한 시공간 데이터베

† 본 연구는 정보통신부의 대학 S/W 연구센터 지원사업의 연구 결과임.
* 인하대학교 대학원 전자계산공학과 박사과정 wnchung@dblab.inha.ac.kr
** 인하대학교 컴퓨터공학부 교수 hybae@inha.ac.kr

이스 시스템에서 시공간 객체의 위치 정보를 표현하기 위해 시공간 데이터모델 및 시공간 연산들을 사용한다[5,10,15,17,18]. 그러나 기존의 시공간 데이터 모델들은 시공간 객체에 대한 개념적인 데이터모델의 정의 수준에 머무르고 있으며, 또한 기존의 시공간 데이터모델을 기반으로 설계된 시공간 연산들[1,5,9,15]의 문제점은 시공간 객체와 관련된 시간정보는 년월일과 같이 여러 단위의 시간을 이용하여 표현되므로, 시간 단위에 따른 연산의 실행 방법에 대한 정의의 미비로 인해 상이한 시간 단위를 이용하여 시간정보를 표현한 시공간 객체들이 피연산자로 입력되는 경우 연산의 결과를 신뢰할 수 없다는 것과 공간정보나 시간정보와 관련된 다양한 연산을 제공하지 않으므로 시공간 객체의 변화를 이용한 다양한 분석을 수행할 수 없다는 것이다.

따라서 본 논문에서는 위치 기반 서비스에 효과적인 적용하기 위해 시간정보를 포함하는 시공간 객체의 정보를 효율적으로 관리할 수 있는 시공간 데이터모델을 제안한다. 또한 시간정보와 공간정보를 피연산자로 하는 다양한 시공간 연산의 설계와 시공간 데이터의 무결성과 연산의 무결성을 유지하기 위한 제약조건을 제시한다. 아울러 실험을 통해 제안 모델은 위치기반서비스에서 시공간 객체를 효과적으로 관리하기 위해 적용 가능하고, 시공간 객체의 과거, 현재, 미래의 모든 질의 시점에 대한 연산 처리를 지원함으로써 현재 위치를 기반으로 관심 지역의 검색, 특정 시공간 객체의 위치 추적 경로 서비스 등 다양한 위치기반 응용서비스에 적용할 수 있음을 보인다.

본 논문의 구성은 다음과 같다. 2장에서는 시공간 데이터베이스의 관련 연구동향을 살펴보고, 기존의 시공간 데이터모델에 대하여 알아본다. 3장에서는 시공간 객체를 정의하고, 시공간 데이터모델을 제안한다. 4장에서는 제안 모델에서의 시공간 데이터타입에 대해 기술하고, 5장에서는 공간정보와 시간정보를 이용하는 다양한 시공간 연산을 설계하고, 각 연산들의 무결성 유지를 위한 제약조건을 정의한다. 6장에서는 제안된 기법에 대한 평가를 수행하고, 마지막으로 7장에서는 논문의 결론과 함께 향후 연구과제에 대하여 논한다.

2. 관련 연구

Yuan[7]은 관계형 데이터모델을 기반으로 설계된 시공간 데이터모델을 레이어단위 시공간 데이터모델, 속성단위 시공간 데이터모델 그리고 공간객체 단위 시공간 데이터모델로 구분하였으며, 이들은 각각 기존의 문자기반 시간지원 시스템의 데이터모델인 테이블단위 시간지원 데이터모델, 튜플단위 시간지원 데이터모델, 속성단위 시간지원 데이터모델과 유사한 특징을 갖는다[8].

레이어단위 시공간 데이터모델의 대표적인 모델로는 Armstrong[10]의 스냅샷 모델이 있다. 스냅샷 모델에서는 공간 객체에 대한 공간정보의 변경이 발생할 때마다 변경된 공간 객체를 포함하는 레이어에 포함된 모든 공간 객체에 대한 새로운 스냅샷 레이어를 생성하여 관리한다. 이 모델은 어느 시점에 유효한 객체의 정보들은 모두 하나의 레이어에 스냅샷의 형태로 저장되어 있기 때문에 현재 시점이나 과거 시점에 유효한 객체의 상태를 검색하는 연산은 데이터베이스의 크기에 영향을 받지 않고 빠르게 수행할 수 있으며, 현재의 공간 데이터베이스 시스템을 이용하여 쉽게 구현할 수 있다. 그러나 공간정보의 변화가 없는 공간 객체의 데이터들도 새로 생성되는 레이어에 포함되기 때문에 심각한 자료의 중복이 발생하며, 이로 인하여 데이터베이스의 크기가 빠르게 증가하고, 동일한 레이어에 대한 정보가 여러 개의 테이블에 분산되어 저장되기 때문에 객체의 변화 과정을 검색하기 위해서는 객체의 변경된 회수와 비례하는 개수의 테이블에 대한 조인 연산을 수행하여야 한다[7].

속성단위 시공간 데이터모델의 대표적인 모델로는 Langran과 Chrisman이 제안한 시공 복합 모델이 있다[6]. 시공 복합 모델에서는 하나의 레이어에 변화된 모든 공간 객체들의 형상을 표현한다. 이 모델은 스냅샷 모델에 비하여 데이터 중복의 문제가 심하지 않지만, 공간 객체의 변경이 발생했을 경우 변경된 부분을 새로운 객체로 관리하기 때문에 공통된 속성을 가지는 하나의 공간 객체가 여러 개의 분할된 객체로 표현된다. 즉, 어느 한 시점에 유효한 객체의 정보가 여러 개의 튜플에 분산되어 저장되기 때문에 이를 검색하기 위해서는 여러 튜플의 조인이 필요하다. 또한, 레이어를 구성하는 많은 객체들 중에서 단 하나의 공간 객체가 변경되더라도 변경된 객체를 포함한 모든 객체들에 대한 시간정보를 재구성해야 한다[7].

공간객체단위 시공간 데이터모델의 대표적인 모델로는 Worboys의 시공간 객체 모델이 있다[2]. 이 모델은 기존의 2차원 공간에 시간 차원이 결합된 3차원 시공간 위에 공간 객체를 표현하며, 각 공간 객체들은 시공간 원자로 구성된다. 이 모델은 시간과 공간차원에 대한 시공간 객체의 속성 변화에 대하여 동시에 저장할 수 있기 때문에 시간의 흐름에 따른 객체의 변화 과정은 쉽게 기록하고 검색할 수 있지만, 전체적인 공

간정보의 변화 과정은 표현할 수 없다.

이와 같이 기존의 시공간 객체에 대한 연구는 연산의 수행 속도가 지속적으로 증가하는 데이터베이스의 크기로부터 받는 영향을 최소화하고, 시간의 흐름에 따라 동적으로 변화하는 공간 객체의 변화 과정을 이용한 분석을 효과적으로 수행할 수 있는 모델을 제안한다.

3. 시공간 데이터 모델의 제안

3.1 시공간 객체의 정의

본 논문에서 제안하는 모델에서의 시공간 객체는 비공간 속성을 표현하는 하나 이상의 비공간정보와, 2차원(또는 3차원) 공간에서 객체의 위치와 형태를 표현하는 공간정보 및 비공간정보와 공간정보가 실세계에서 유효한 시간의 범위를 표현하는 유효시간과 시스템에 저장되어 있던 시간의 범위를 표현하는 거래시간으로 구성한다. 시공간 객체의 정형 의미(formal semantics)는 다음과 같다.

$$o^{st}=(u^{(i+1+4)} \mid 1\le i\le n \wedge$$
$$u[1]=O^{st}.a_1 \wedge u[2]=O^{st}.a_2 \wedge \cdots \wedge u[n]=O^{st}.a_n \wedge$$
$$u[n+1]=O^{st}.\ge ometry \wedge$$
$$u[n+1+1]=O^{st}.vs \wedge u[n+1+2]=O^{st}.ve \wedge$$
$$u[n+1+3]=O^{st}.ts \wedge u[n+1+4]=O^{st}.ve \wedge$$
$$u[n+1+1]\le u[n+1+2] \wedge u[n+1+3]\le u[n+1+4])$$

$O^{st}.a_i$는 시공간 객체 O^{st}의 비공간 속성을 의미하며, $O^{st}.node$는 노드의 집합으로 구성되는 공간 속성을 의미한다. $O^{st}.vs$와 $O^{st}.ve$는 유효시작시간과 유효종료시간을 의미하며, $O^{st}.ts$와 $O^{st}.te$는 거래시작시간과 거래종료시간을 의미한다. 여섯번째 줄은 유효시간과 거래시간에 대한 제약조건을 기술한 내용이다. 노드의 집합으로 구성되는 $O^{st}.\ge geometry$의 정형 의미는 다음과 같다.

$$O^{st}.geometry=(K_i \mid k_i\in V_{geometry} \wedge 1\le i\le n$$
$$node=\{U^j \mid 1\le j\le m \wedge u[1]\}=R^j.c_1 \wedge u[2]$$
$$=R^j.c_2 \wedge \cdots \wedge u[j]=R^j.c_j\}$$

$V_{geometry}$는 노드(node)들의 집합을 나타내고, $R^j(1\le j\le m)$는 j차원의 유클리드 공간을 의미하며, $R^j.c_i(1\le i\le j)$는 j차원의 유클리드 공간에서 i번째 차원의 좌표값을 의미한다.

3.2 데이터 모델의 정의

제안하는 시공간 데이터에 대해 공간 정보를 포함하고 있는 시간지원 테이블을 위한 제안 모델의 정의는 다음과 같다.

시공간 테이블 스키마 R^{ST}는 비공간 정보를 위한 속성인 $A_i(1\le i\le n)$와 공간정보를 위한 속성인 S및 시간 정보를 위한 속성인 T로 구성되는 집합으로 다음과 같이 정의한다.

$$R^{ST}=\{A_1,\ A_2,\ \cdots,\ An,\ S,\ T\}$$

비공간속성 $A_i(1\le i\le n)$는 A_i의 도메인인 $D^A{}_i$의 집합이며, 비공간속성의 도메인 D^A는 다음과 같다.

$$DA=D^A{}_1\times D^A 2\times\cdots\times D^A{}_n$$

공간속성 S의 도메인 D^S는 x좌표와 y좌표의 쌍으로 구성되는 노드 $n_j(1\le j\le m)$의 집합으로 다음과 같이 정의한다.

$$D^S=\{n_1,\ n_2,\ \cdots,\ n_m\}$$

NOW는 현재 시간을 나타내는 시간 상수로, 유효시간의 도메인 D^{VT}와 거래시간의 도메인 D^{TT}를 각각 다음과 같이 정의할 때,

$$D^{VT}=\{vt_1,\ vt_2,\ \cdots,\ vt_k\} \cup \{NOW\}$$
$$D^{TT}=\{tt_1,\ tt_2,\ \cdots,\ tt_l\} \cup \{NOW\}$$

유효시간과 거래시간을 구성하는 시작시간과 종료시간의 도메인은 각각 다음과 같다.

- 유효시작시간의 도메인 $D^{VTs}=D^{VT}-\{NOW\}$
- 유효종료시간의 도메인 $D^{VTs}=D^{VT}$
- 거래시작시간의 도메인 $D^{TTs}=D^{TT}-\{NOW\}$
- 거래종료시간의 도메인 $DTT_E=D^{TT}$

따라서, 이력 공간 테이블과 롤백 공간 테이블 및 이원시간 공간 테이블의 시간 속성 T의 도메인은 각각 다음과 같이 정의된다.

- 이력 공간 테이블의 시간 속성 T의 도메인 $DT_H=D^{VT}$
- 롤백 공간 테이블의 시간 속성 T의 도메인 $DT_R=D^{TT}$
- 이원시간 공간 테이블의 시간 속성 T의 도메인 $DT_B=D^{VT}\times D^{TT}$

$DT_H\subset DT^B$이고, $DT_H\subset DT_B$이므로 시공간 테이블의 시간 도메인 D^T는 다음과 같다.

$$D^T=D^B=D^{VT}\times D^{TT}$$

$a_i\in D^A{}_i$이고, $s\in D^S$이며, $t_v\in D^{VTs}$, $t_{vs}\in D^{VTs}$, $t_{ve}\in DVT_E$, $t_{ts}\in D^{TTs}$, $t_{te}\in DTT_E$ 일 때, 스키마 R^{ST}에서 정의되는 개념적인 시공간 테이블 인스턴스 $r(R^{ST})$을 구

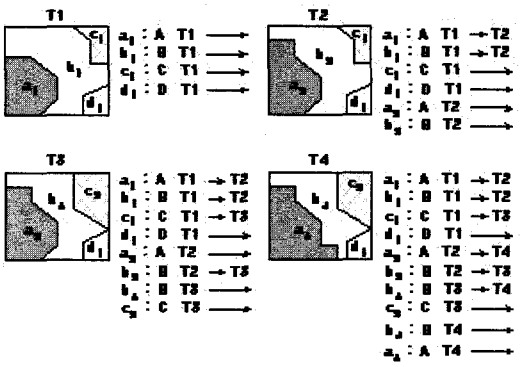
성하는 튜플 x는 〈표 1〉의 시공간 테이블에 대하여 각각 다음과 같다.

- 이력 사건 공간 테이블의 튜플
  $x^{he}=(a_1,a_2,\cdots a_n,s,t_v)$
- 이력 기간 공간 테이블의 튜플
  $x^{hi}=(a_1,a_2,\cdots a_n,s,t_{vs},t_{ve})$
- 롤백 공간 테이블의 튜플
  $x^{r}=(a_1,a_2,\cdots a_n,s,t_v,t_{te})$
- 이원시간 사건 공간 테이블의 튜플
  $x^{be}=(a_1,a_2,\cdots a_n,s,t_v,t_{ts},t_{te})$
- 이원시간 기간 공간 테이블의 튜플
  $x^{bi}=(a_1,a_2,\cdots a_n,s,t_{vs},t_{ve},t_{ts},t_{te})$

따라서, 시간 도메인 D^T에서 정의되는 시공간 테이블의 튜플 x^{st}는 다음과 같다.

$$x^{st}=(a_1,a_2,\cdots,a_n,s,t)$$

즉, 시공간 테이블을 구성하는 각 튜플들은 튜플에 포함된 정보들이 유효한 시간 범위를 갖는다. 이와 같이 시공간 테이블에 포함되는 다섯 가지 부류의 테이블들은 모두 비공간 도메인 D^A와 공간 도메인 D^S 및 시간 도메인 D^T에서 정의되며, '이원시간 기간 공간 테이블'이 가장 포괄적인 정의를 포함한다. 본 논문에서는 시공간 테이블의 범위를 이원시간 기간 공간 테이블로 한정하여 논문을 전개하며, 필요한 경우에만 각 테이블을 구분하기로 한다. 〈그림 1〉은 제안 시공간 데이터모델을 이용하여 표현한 시공간 객체의 예이다. 시간 T1에서 각 객체는 T1 시간을 가지며 시간 T2에서 상태가 변경된 a1과 b1에 대한 유효시간은 T2로 종료되고 a2와 b2는 시간 T2로 새로이 설정된다. 이후 과정은 동일한 과정으로 진행된다.



〈그림 1〉 제안 시공간 데이터모델을 이용한 시공간 객체 정보 표현

제안하는 시공간 데이터모델의 특징은 다음과 같다. 첫째, 어느 객체의 상태가 변경된 경우 이 객체에 대한 정보만 변경하기 때문에, 다른 객체들의 시간정보를 재구성할 필요가 없다. 둘째, 어느 시점에 유효한 객체의 상태는 하나의 튜플에 저장되기 때문에 튜플간의 조인이나 테이블간의 조인연산을 하지 않아도 이 객체에 대한 검색을 수행할 수 있다.

4. 시공간 데이터 타입

4.1 데이터 타입의 정의

시공간 객체는 비공간정보, 공간정보와 시간정보를 포함하며, 각 정보 표현을 위한 데이터 타입을 정의하여야 한다. 시공간 객체의 공간정보를 표현하기 위한 공간 데이터타입을 객체의 기하학적인 특성을 기준으로 점 타입(Point type), 선 타입(Line type), 영역 타입(Polygon type), 그리고 영역선 타입(Polyline type)로 나뉜다.

시공간 객체의 시간 정보를 표현하기 위한 시간 데이터 타입을 사건 타입과 기간 타입 및 간격 타입의 정의는 다음과 같다. 먼저 사건 타입(Event type)은 시공간 객체와 관련된 사건이 발생한 시간을 표현하기 위한 한 개의 타임스탬프(timestamp)를 갖는 시간 데이터 타입을 사건 타입이라고 한다. 기간 타입(Interval type)은 시공간 객체의 유효시간을 표현하기 위하여 두 개의 타임스탬프를 갖는 시간 데이터 타입을 기간 타입이라고 한다. 이는 종료시간에 사용된 타임스탬프의 의미에 따라 종료시간의 타임스탬프를 기간에 포함시키지 않는 기간타입인 열린기간타입(Open Interval Type)과 종료시간의 타임스탬프를 기간에 포함시키는 기간타입인 닫힌기간타입(Close Interval Type)으로 정의한다. 간격 타입(Duration type): 추상적인 시간의 길이를 표현하는 시간 데이터 타입으로, 시간 단위와 정수로 구성된다.

4.2 시간 데이터 타입의 제약조건

사건 타입, 기간 타입 그리고 간격 타입으로 표현되는 시간 정보의 무결성을 유지하기 위하여 다음과 같은 제약 조건을 정의한다.

【제약조건 1】 사건 타입으로 선언된 시간은 오직 한 개의 시간값을 갖는다.

【제약조건 2】 기간 타입으로 선언된 시간은 시작시간과 종료시간으로 구성되는 두 개의 시간값을 가지며, 두 시간 단위는 동일해야 하고, 종료시간이 시작시간보다 작을 수 없다.

【제약조건 3】 간격타입으로 선언된 시간은 시간단위를

표현하는 'YEAR'나 'MONTH', 'DAY', 'HOUR', 'MINUTE', 'SECOND'중 하나와 시간크기를 표현하는 양의 정수를 갖는다.

4.3 시간 변수의 정의

시공간 객체의 상태가 현재 시점에 유효한 경우 이 객체의 유효기간을 표현하기 위하여 시작 시간은 상수의 시간으로 표현이 되지만 종료 시간은 상수의 시간으로 표현할 수 없다. 즉, 유효종료시간은 실세계에서 시간의 흐름에 따라 계속 연장된다. 이러한 시간을 표현하기 위하여 유효종료시간과 거래종료시간에 모두 다음과 같이 시간 변수 NOW를 정의한다. 시간 변수 NOW는 현재의 시간을 의미하는 시간 변수로 실세계 시간의 흐름에 따라 계속해서 변하는 시간을 나타낸다.

5. 시공간 연산 및 제약 조건

5.1 공간 연산

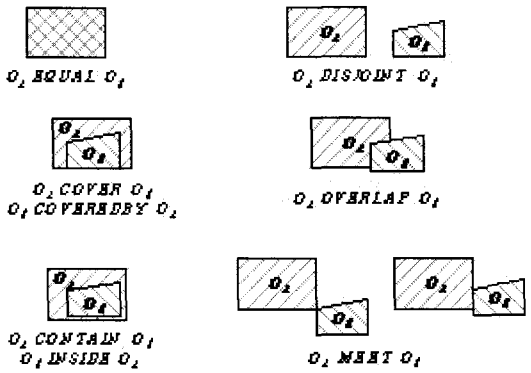
공간 연산이란 공간 객체의 기하학적인 특성을 이용하여 공간 객체간의 다양한 관계나, 공간 분석과정을 수행하는 연산이다. 본 논문에서는 연산의 결과로 반환하는 데이터의 형태에 따라 다음과 같이 네 가지로 분류한다. 공간 논리 연산에서 공간 위상 연산과 공간 기하 연산은 OpenGIS 명세서[14]에서 정의한 연산을 확장하였으며, 본 논문에서 공간 위상 연산은 영역 타입의 두 공간 객체에서 발생할 수 있는 위상 관계에 대해서만 명시한다. 본 논문에서는 공간 객체 O를 구성하는 x좌표와 y좌표의 집합을 각각 $O.x$와 $O.y$로 표기한다.

5.1.1 공간 논리 연산

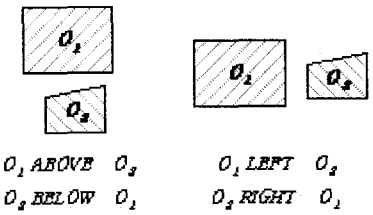
공간 위상 연산은 피연산자로 입력된 두 공간 객체의 위상관계를 연산하여 그 결과로 논리값을 반환하는 부울 연산이다. 〈그림 2〉는 폴리곤타입의 두 공간 객체간에 발생할 수 있는 위상 관계를 표현한 그림이다.

공간 위상 연산은 공간 객체 O를 구성하는 경계선들의 집합을 의미하는 *boundary*(O)와 객체의 내부를 나타내는 *interior*(O)와의 교집합을 이용하여 정의할 수 있다.

공간 위치 연산은 공간 객체간의 상하좌우 위치 관계를 파악하기 위한 연산으로, 〈그림 3〉과 같이 네 가지의 연산을 정의한다. 〈그림 3〉의 연산들은 두 번째 피연산자로 입력된 공간 객체의 관점에서 첫 번째 피연산자로 입력된 공간 객체의 위치를 판단하며, 이들은 공간 객체를 구성하는 좌표값들의 최대값과 최소값을 이용하여 정의할 수 있다.



〈그림 2〉 두 공간 객체간에 존재하는 공간 위상 관계



〈그림 3〉 두 공간 객체간의 공간 위치 관계

5.1.2 공간 수치 연산

공간 수치 연산은 피연산자로 입력된 공간 객체들을 구성하는 노드의 좌표값을 이용한 수치 연산을 수행하여 정수값이나 실수값을 반환하는 연산이다..

(1) *NUMNODE*(O): 공간 객체 O를 구성하는 노드의 개수를 반환한다.

(2) *LENGTH_X*(O): 공간 객체 O의 X축으로의 길이를 구한다.

$$LENGTH\_X(O) = max(O.x) - min(O.x)$$

(3) *LENGTH_Y*(O): 공간 객체 O의 Y축으로의 길이를 구한다.

$$LENGTH\_Y(O) = max(O.y) - min(O.y)$$

(4) *AREA*(O): 피연산자인 공간 객체 O가 폴리곤일 경우 공간 객체 O의 면적을 구한다.

(5) *PERIMETER*(O) : 공간 객체 O를 구성하는 세그먼트들의 길이의 합을 구한다.

$$PERIMETER(O) = \sum_{i=1}^{n} \sqrt{(O.x_{i+1} - O.x_i)^2 + (O.y_{i+1} - O.y_i)^2}$$

(6) *DISTANCE*(O_1, O_2): 피연산자 O_1과 O_2의 중심

접으로 두 공간 객체간의 거리를 구한다.

$$DISTANCE(O_1, O_2) = \sqrt{(\frac{\max(O_1.x)+\min(O_1.x)-\max(O_2.x)-\min(O_2.x)}{2})^2 + (\frac{\max(O_1.y)+\min(O_1.y)-\max(O_2.y)-\min(O_2.y)}{2})^2}$$

(7) *MIN_DISTANCE*(O_1,O_2): 피연산자 O_1과 O_2의 구성 노드중 최근접 노드간의 거리를 구한다.

(8) *MAX_DISTANCE*(O_1,O_2) : 피연산자로 입력된 두 공간 객체 O_1과 O_2를 구성하는 노드들 중 가장 멀리 떨어져 있는 노드간의 거리를 구한다.

공간 객체 O_1과 O_2가 각각 n개의 노드와 m개의 노드로 구성되어 있고, $A_i(1 \le i \le n)$와 $B_j(1 \le j \le m)$는 각각 객체 O_1의 i번째 노드와 객체 O_2의 j번째 노드이며, dist(a, b)를 노드 a와 노드 b의 거리라고 할 때 *MIN_DI STANCE*와 *MAX_DISTANCE*는 다음과 같이 구할 수 있다.

$$MIN\_DISTANCE(O_1, O_2) = min(dist(A_i, B_j))$$
$$MAX\_DISTANCE(O_1, O_2) = max(dist(A_i, B_j))$$

5.1.3 공간 기하 연산

공간 기하 연산은 피연산자로 입력된 공간 객체를 구성하는 노드나 세그먼트들의 일부를 반환하는 연산이다.

(1) *BOUNDARY*(O): 공간객체 O의 경계선을 구성하는 노드들을 반환하는 연산으로, 2차원상에 존재하는 k개의 노드로 구성된 공간 객체 O에 대한 *BOUNDARY*연산의 결과는 다음과 같다.

$$BOUNDARY(O)=\{(x\_coord, y\_coord)^k | x\_coord \in O.x \wedge y\_coord \in O.y \wedge k=NUMNODE(O)\}$$

(2) *MBR*(O): 공간객체 O의 최소경계사각형 (MBR:Minimum Bounding Rectangle)을 반환한다.

$$MBR(O)=\{coord^4 | coord[1]=min(O.x) \wedge coord[2]=min(O.y) \wedge coord[3]=max(O.x) \wedge coord[4]=min(O.y)\}$$

(3) *CENTROID*(O): 공간 객체 O의 MBR을 이용하여 중심점의 좌표를 반환한다.

$$CENTROID(O)=(x,y),\ x=(max(O.x)+min(O.x))/2,\ y=(max(O.y)+min(O.y))/2$$

(4) *COMMON_BORDER*(O_1,O_2): 피연산자인 두 공간 객체의 공통 경계선들의 집합을 구하는 연산으로, 두 공간연산의 위상관계가 MEET, COVER 또는 COVEREDBY일 경우에만 가능하다.

(5) *UNION*(O_1,O_2): 피연산자로 입력되는 두 공간 객체 O_1과 O_2의 합집합을 구하며, 이 연산은 두 공간 객체의 위상 관계가 *DISJOINT*가 아닐 경우만 가능하다.

(6) *INTERSECTION*(O_1,O_2): 피연산자인 O_1과 O_2가 겹치는 부분의 좌표값을 구한다.

(7) *DIFFERENCE*(O_1,O_2): 공간 객체 O_1에서 공간 객체 O_2의 영역을 제외한 부분을 반환한다.

5.1.4 공간 변환 연산

피연산자로 입력된 공간 객체의 위치, 크기, 방향, 형태 등을 변환하는 연산으로, 잘못 입력된 공간 객체들을 일괄적으로 수정하는 경우 필요한 연산들이다.

(1) *ROTATE*(O,Θ,*Rx*,*Ry*): 공간객체 O를 구성하는 모든 노드들을 좌표 *Rx*와 *Ry*를 회전축으로 하여 Θ만큼 회전한다. 공간 객체 O를 구성하는 x 좌표와 y 좌표는 다음과 같이 변환된다.

$$(O.x,\ O.y) = (O.x \cdot \cos 2\Theta + O.y \cdot \sin 2\Theta,\ O.x \cdot \sin 2\Theta - O.y \cdot \cos 2\Theta)$$

(2) *MOVE_X*(O, a): 공간객체 O를 구성하는 모든 노드들을 X축으로 평행하게 피연산자로 입력된 실수 a만큼 이동한다.

$$(O.x,\ O.y) = (x + a,\ O.y)$$

(3) *MOVE_Y*(O, b): 공간 객체 O를 구성하는 모든 노드들을 Y축으로 평행하게 피연산자로 입력된 실수 b만큼 이동한다.

$$(O.x,\ O.y) = (O.x,\ O.y + b)$$

(4) *SCALE_X*(O, a): 공간 객체 O를 구성하는 모든 노드들을 X축으로 평행하게 피연산자로 입력된 실수 a만큼 확대/축소한다.

$$(O.x\ ,\ O.y\ ) = (a \cdot O.x\ ,\ O.y)$$

(5) *SCALE_Y*(O, b): 공간 객체 O를 구성하는 모든 노드들을 Y축으로 평행하게 피연산자로 입력된 실수 b만큼 확대/축소한다.

$$(O.x,\ O.y) = (O.x,\ b \cdot O.y)$$

(6) *SHEAR_X*(O, a): 공간 객체 O를 구성하는 모든 노드들을 X축으로 평행하게 피연산자로 입력된 실수 a를 이용하여 변환한다.

$$(O.x,\ O.y) = (O.x + a \cdot O.y,\ O.y)$$

(7) *SHEAR_Y*(O, b): 공간 객체 O를 구성하는 모든 노드들을 Y축으로 평행하게 피연산자로 입력된 실수 b를 이용하여 변환한다.

$$(O.x,\ O.y) = (O.x,\ b \cdot O.x + O.y)$$

5.2 시간 연산

시간 연산이란 피연산자로 입력된 공간 객체와 관련된 시간정보를 이용하여 공간 객체간의 시간적인 관계

나, 시간을 이용한 수치 연산을 수행할 수 있도록 하는 연산으로, 연산의 결과로 반환되는 데이터의 형태에 따라서 시간 수치 연산과 시간 논리 연산으로 분류한다. 제안하는 시간 연산들은 피연산자로 객체의 유효시간과 거래시간이 모두 사용될 수 있다.

5.2.1 시간 수치 연산

시간수치연산은 피연산자로 입력된 공간 객체의 유효시간이나 거래시간을 반환하거나, 이들 시간을 이용한 연산을 수행하여 또 다른 시간을 반환하는 연산이다.

(1) *START_OF*()/*END_OF*(): 피연산 객체의 유효시간이나 거래시간의 시작시간과 종료시간을 반환한다. 피연산자 객체가 이벤트 타입으로 저장되어 있는 경우는 두 연산의 결과가 동일하다.

(2) *VALIDTIME*(): 입력된 공간객체의 시간타입이 사건타입일 경우에는 사건이 발생한 시간을 의미하는 유효시간을 시간단위와 함께 반환하며, 기간타입일 경우는 피연산자의 유효 시작시간과 유효 종료시간을 시간 단위와 함께 반환한다.

(3) *TRANSACTIONTIME*(): 입력된 피연산자의 거래시작시간과 거래종료시간을 반환한다.

(4) *TEMP_DURATION*(): 피연산자로 시간단위를 포함한 기간타입의 시간과 변환하고자 하는 간격타입의 시간단위가 입력되며, 입력된 기간을 두 번째 피연산자로 입력된 시간단위의 간격(duration)으로 변환하여 반환한다. 두 번째 피연산자는 생략할 수 있으며 이 때는 첫 번째 피연산자의 시간 단위로 간격이 계산되어 반환된다.

(5) *TEMP_GRANULE_CHANGE*(): 시간단위를 포함한 시간과 시간단위가 피연산자로 입력되며, 첫 번째 피연산자로 입력된 시간을 두 번째 피연산자로 입력된 시간 단위로 변경하여 반환한다. 연산에 입력되는 시간은 세 가지 타입의 시간들이 모두 가능하다. 연산의 수행은 첫 번째 피연산자의 시간 단위와 시간 타입, 그리고 두 번째 피연산자의 시간 단위의 관계에 따라 다양하게 수행되어야 한다.

(6) *FIRST*() /*LAST*(): 집단화된 결과에서 시간상 최초 또는 최후의 정보를 반환한다.

(7) *GROUP_OVERLAPTIME*(): 집단화된 결과의 유효시간이나 거래시간들이 모두 겹치는 기간을 반환한다.

(8) *GROUP_CONCATENATE*(): 집단화된 결과의 유효시간이나 거래시간을 하나의 기간 타입으로 연결하여 반환한다. 집단화된 결과의 시간들은 최소한 하나 이상의 다른 시간과 OVERLAP이나 MEET의 시간 위상 관계가 성립하여야 하며, 그렇지 않은 경우에는 널(null)을 반환한다.

(9) *TEMP_CONCATENATE*(): 피연산자로 입력된 두 개의 기간 타입의 시간을 하나의 기간 타입으로 연결(concatenate)하여 반환하며, 피연산자로 입력된 두기간 타입의 시간 위상 관계는 TEMP_OVERLAP이나 TEMP_MEET의 관계가 성립하여야 한다.

(10) *OVERLAPTIME*(): 두 기간이 겹치는 기간을 반환하며, 피연산자로 입력되는 두 기간의 시간 위상 관계는 TEMP_EQUAL, TEMP_OVERLAP, TEMP_INCLUDE, TEMP_EXCLUDE의 관계가 성립하여야 한다.

5.2.2 시간 논리 연산

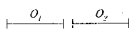
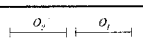
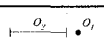
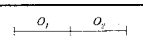
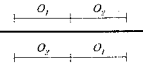
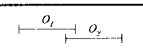
시간 논리 연산은 시간 위상 연산과 시간 비교 연산으로 분류되며, 시간 위상 연산은 데이터베이스에 저장되어 있는 공간 객체들의 유효시간이나 거래시간간의 관련성을 검사하여 부울값을 반환하는 연산으로, 〈표 2〉와 〈표 3〉은 동일한 시간 단위로 표현된 시간정보를 갖는 두 공간 객체간에 존재할 수 있는 시간 위상 관계를 나타낸 표이다. 〈표 2〉와 〈표 3〉에서 I는 간격타입을 의미하고 E는 이벤트 타입을 의미한다.

시간 위상 관계는 동일한 시간 단위로 표현된 시간정보를 포함하고 있는 공간 객체들간에만 파악이 가능하지만, 본 논문에서 정의하는 시간 위상 연산들은 연산에 입력되는 피연산자들의 시간 타입이나 시간 단위에 대한 제약을 최소로 한다. 즉, 〈표 2〉에서 허용하지 않는 상이한 시간 타입이나 상이한 시간 단위를 이용하여 표현된 시간들이 피연산자로 입력되는 경우에도 의미상 두 공간 객체간의 시간 위상 관계를 파악할 수 있는 경우에는, 시간 위상 연산을 실행하기 전에 피연산자들의 시간 단위를 변경한 다음 〈표 2〉의 시간 타입 관계를 적용하여 시간 위상 연산을 실행하도록 한다. 본 절에서는 동일한 시간 단위로 표현된 피연산자들이 입력되는 경우에 대해서만 다루기로 한다. 시간비교연산은 기존의 비교연산자를 이용하여 간격타입

〈표 2〉 시간단위가 동일한 시간 위상 연산에서 허용하는 피연산자의 시간 타입 관계

구 분	I-I	I-E	E-I	E-E
TEMP_EQUAL	○	×	×	○
TEMP_BEFORE	○	○	○	○
TEMP_AFTER	○	○	○	○
TEMP_MEET_END	○	○	×	×
TEMP_MEET_START	○	○	×	×
TEMP_OVERLAP_END	○	×	×	×
TEMP_OVERLAP_START	○	×	×	×
TEMP_INCLUDE_END	○	×	×	×
TEMP_INCLUDE_START	○	×	×	×
TEMP_INCLUDE_PROPER	○	○	×	×
TEMP_EXCLUDE_END	○	×	×	×
TEMP_INCLUDE_START	○	×	×	×
TEMP_EXCLUDE_PROPER	○	×	○	×

〈표 3〉 동일한 시간단위의 시간정보를 갖는 두 공간 객체간에 존재하는 시간위상관계

구 분		I-I	I-E	E-I	E-E
O_1 *TEMP_EQUAL* O_2					
O_1 *TEMP_DISJOINT* O_2	O_1 *TEMP_BEFORE* O_2				
	O_1 *TEMP_AFTER* O_2				
O_1 *TEMP_MEET* O_2	O_1 *TEMP_MEET_END* O_2				
	O_1 *TEMP_MEET_START* O_2				
O_1 *TEMP_OVERLAP* O_2	O_1 *TEMP_OVERLAP_END* O_2				
	O_1 *TEMP_OVERLAP_START* O_2				
O_1 *TEMP_INCLUDE* O_2	O_1 *TEMP_INCLUDE_END* O_2				
	O_1 *TEMP_INCLUDE_START* O_2				
	O_1 *TEMP_INCLUDE_PROPER* O_2				
O_1 *TEMP_EXCLUDE* O_2	O_1 *TEMP_EXCLUDE_END* O_2				
	O_1 *TEMP_EXCLUDE_START* O_2				
	O_1 *TEMP_EXCLUDE_PROPER* O_2				

으로 표현된 두 시간의 크기를 비교하는 연산이다. 피연산자로 이용되는 두 간격타입의 시간단위는 동일하여야 한다.

5.3 시공간 연산

시공간 연산은 시공간 객체의 시간 정보와 공간 정보를 이용하여 시간의 흐름에 따른 객체의 변화과정을 분석하기 위한 연산으로, 단순 시공간 검색 연산과 시공간 분석 연산으로 구성된다.

5.3.1 단순 시공간 검색 연산

(1) *GEO_PROJECTION(O)*: 시공간객체 O의 공간정보를 추출한다. 즉, 비공간정보, 공간정보, 시간정보로 구성된 시공간 객체 O의 정보들 중에서 공간정보만 추출하여 반환한다.

(2) *TEMP_PROJECTION(O)*: 시공간객체 O의 시간정보를 추출하며, 유효시간과 거래시간을 모두 반환한다.

5.3.2 시공간 분석 연산

(1) *ST_AREA_CHANGE_RATE(O, Time-1, Time-2)* : 연산에 입력된 두 시점간에 시공간 객체 *O*의 면적에 대한 변화량을 측정한다.

(2) ST_LENGTH_CHANGE_RATE(*O*, Time-1, Time-2): 연산에 입력된 두 시점간에 시공간 객체 *O*의 길이에 대한 변화량을 측정한다.

(3) *ST_DISTANCE(O, Time-1, Time-2)*: 연산에 입력된 두 시점간에 시공간 객체 *O*가 이동한 거리를 측정하여 반환한다.

(4) *ST_SPEED(O, Time-1, Time-2)* : 연산에 입력된 두 시점간에 시공간 객체 *O*의 이동 속도를 계산하여 반환한다.

5.4 시공간 연산의 제약조건

시공간 연산의 제약조건은 연산의 무결성을 지원하기 위하여 정의되며, 각 연산의 수행을 위한 피연산자의 타입에 대한 제약조건과 연산의 결과로 반환되는 결과의 타입에 대한 제약조건으로 구성된다. 정의되는 제약조건에는 연산에 입력되는 피연산자의 타입이나 연산의 결과에 대한 타입의 제약조건과 함께 연산의 특징에 따라서 시간 단위에 대한 제약조건이 포함된다. 제약조건의 정의에 사용된 '*st*'는 연산에 입력되는 피연산자를 의미하며, *tg(st)*는 피연산자 st의 시간 단위를 의미한다.

5.4.1 시간 수치 연산의 제약조건

시간 수치 연산은 연산의 종류와 연산에 입력되는

【제약조건 4】 시간 수치 연산에 입력되는 피연산자의 타입 제약조건

시간 수치 연산	피연산자의 타입 제약조건
START_OF(st)	$st \in Event$ or $st \in Interval$
END_OF(st)	$st \in Event$ or $st \in Interval$
VALIDTIME(st)	$st \in Event$ or $st \in Interval$
TRANSACTIONTIME(st)	$st \in Event$ or $st \in Interval$
TEMP_DURATION(st1,st2,st3)	$st1, st3 \in Temporal\ Granularity$, $st2 \in Interval$ and $tg(st2) = st1$
FIRST(st)	$(st = \{t_i \mid t_i \in Event \wedge 1 < i < n\}$ or $st = \{t_i \mid t_i \in Interval \wedge 1 < i < n\})$ and $\exists t_i \in st \forall t_j \in st (i \neq j$ and $tg(t_i) = tg(t_j))$
LAST(st)	$(st = \{t_i \mid t_i \in Event \wedge 1 < i < n\}$ or $st = \{t_i \mid t_i \in Interval \wedge 1 < i < n\})$ and $\exists t_i \in st \forall t_j \in st (i \neq j$ and $tg(t_i) = tg(t_j))$
GROUP_OVERLAPTIME(st)	$(st = \{t_i \mid t_i \in Interval \wedge 1 < i < n\}$ and $\exists t_i \in st \forall t_j \in st (i \neq j$ and $tg(t_i) = tg(t_j))$
GROUP_CONCATENATE(st)	$(st = \{t_i \mid t_i \in Interval \wedge 1 < i < n\}$ and $\exists t_i \in st \forall t_j \in st (i \neq j$ and $tg(t_i) = tg(t_j))$
TEMP_GRANULE_CHANGE(st1,st2,st3)	$st1, st3 \in Temporal\ Granularity$ ($st2 \in Event$ or $Interval$ or $Duration$) and $tg(st2) = st1$
TEMP_CONCATENATE(st1, st2)	$st1, st2 \in Interval$ and $tg(st1) = tg(st2)$
OVERLAPTIME(st1, st2)	$st1, st2 \in Interval$ and $tg(st1) = tg(st2)$

피연산자의 타입에 따라서 다양한 타입의 결과를 반환한다. 따라서 시간 수치 연산에 입력되는 피연산자의 타입에 대한 제약조건과 연산의 수행 결과에 대한 제약조건을 정의하여야 한다.

다음은 시간 수치 연산에 입력되는 피연산자의 타입 제약조건으로 피연산자로 시간의 집합이 입력되는 연산들과 두개 이상의 시간 단위가 입력되는 연산들은 피연산자의 타입에 대한 제약조건과 함께 시간 단위에 대한 제약조건을 정의하였다.

다음은 시간 수치 연산 결과에 대한 제약조건으로 두 개 이상의 시간 단위가 입력되는 연산들은 피연산자의 타입에 대한 제약조건과 함께 시간 단위에 대한 제약조건을 정의하였다.

【제약조건 5】 시간 수치 연산 결과의 타입 제약조건

시간 수치 연산	피연산자의 타입 제약조건
r = START_OF(st)	r = Event
r = END_OF(st)	r = Event
r = VALIDTIME(st)	if st ∈Event then r = Event if st ∈Interval then r = Interval
r = TRANSACTIONTIME(st)	if st ∈Event then r = Event if st ∈Interval then r = Interval
r = TEMP_DURATION (st1, st2, st3)	r = Duration and tg(r) = st3
r = FIRST(st)	r = Event 단, if st∈Event and ∃e∈st then e TEMP_EQUAL r if st∈Interval then ∃i∈st then i TEMP_MEET_START r
r = LAST(st)	r = Event 단, if st∈Event and ∃e∈st then e TEMP_EQUAL r if st∈Interval and ∃i∈st then i TEMP_MEET_END r
r = GROUP_OVERLAPTIME(st)	if∀t∈st ∃i∈Interval(t TEMP_OVERLAPi) then r = i else r = NULL
r = GROUP_CONCATENATE(st)	if∃t_i∈st ∃t_j∈st(i≠j and (t_i TEMP_OVERLAP t_j or t_i TEMP_MEET t_j)) then r = Interval else r = NULL
r = TEMP_GRANULE_CHANGE (st1, st2, st3)	if st ∈Event then r = Event if st ∈Interval then r = Interval if st ∈Duration then r = Duration tg(r) = st3
r = TEMP_CONCATENATE (st1, st2)	if (st1 TEMP_OVERLAP st2) or (st1 TEMP_MEET st2) then r = Interval else r = NULL
r = OVERLAPTIME(st1, st2)	if st1 TEMP_OVERLAP st2 then r = Interval else r = NULL

5.4.2 시간 위상 연산의 제약조건

시간 위상 연산에 입력되는 피연산자의 시간 단위는 원칙적으로 동일한 경우만 허용한다. 시간 단위 변경에 의하여 상이한 시간 단위의 피연산자가 입력된 경우에도 연산의 실행이 가능한 경우가 발생하기 때문에 시간 위상 연산에 입력되는 피연산자의 데이터 타입에 대한 제약조건을 두 피연산자의 시간 단위가 동일한 경우와 상이한 경우로 나누어 정의한다. 또한, 시간 위상 연산은 연산의 수행 결과로 논리값인 참이나 거짓을 반환하기 때문에 연산 결과에 대한 타입 제약조건은 정의하지 않는다. 연산에 입력되는 두 피연산자의 시간 단위가 동일한 경우에 피연산자의 타입 제약조건은 【제약조건 6】과 같이 정의하며, 두 피연산자의 시간 단위가 상이한 경우의 제약 조건은 【제약조건 7】과 같이 정의한다. 【제약조건 7】에서 사용한 tg(st)는 피연산자 st의 시간 단위를 의미하며, 두 시간 단위의 사이에 사용된 기호 ◀는 왼쪽 피연산자의 시간 단위가 오른쪽 피연산자의 시간 단위보다 더 조밀함을 의미하고, 기호 ▶는 반대의 경우를 의미한다.

【제약조건 6】 시간 위상 연산에 입력되는 피연산자의 타입 제약조건(시간단위 동일)

시간 위상 연산	피연산자의 타입 제약조건
st1 TEMP_EQUAL st2	(st1 ∈Interval and st2 ∈Interval) or (st1 ∈Event and st2 ∈Event)
st1 TEMP_BEFORE st2	(st1 ∈Interval and (st2 ∈Interval or st2 ∈Event)) or (st1 ∈Event and (st2 ∈Interval or st2 ∈Event))
st1 TEMP_AFTER st2	(st1 ∈Interval and (st2 ∈Interval or st2 ∈Event)) or (st1 ∈Event and (st2 ∈Interval or st2 ∈Event))
st1 TEMP_MEET_END st2	(st1 ∈Interval and (st2 ∈Interval or st2 ∈Event))
st1 TEMP_MEET_START st2	(st1 ∈Interval and (st2 ∈Interval or st2 ∈Event))
st1 TEMP_OVERLAP_END st2	st1 ∈Interval and st2 ∈Interval
st1 TEMP_OVERLAP_START st2	st1 ∈Interval and st2 ∈Interval
st1 TEMP_INCLUDE_END st2	st1 ∈Interval and st2 ∈Interval
st1 TEMP_INCLUDE_START st2	st1 ∈Interval and st2 ∈Interval
st1 TEMP_INCLUDE_PROPER st2	(st1 ∈Interval and (st2 ∈Interval or st2 ∈Event))
st1 TEMP_EXCLUDE_END st2	st1 ∈Interval and st2 ∈Interval
st1 TEMP_EXCLUDE_START st2	st1 ∈Interval and st2 ∈Interval
st1 TEMP_EXCLUDE_PROPER st2	(st2 ∈Interval and (st1 ∈Interval or st1 ∈Event))

【제약조건 7】 시간 위상 연산에 입력되는 피연산자의 타입 제약조건(시간단위 상이)

시간 위상 연산	피연산자의 타입 제약조건
st1 TEMP_EQUAL st2	(st1 ∈ Interval and st2 ∈ Interval) or ((st1 ∈ Interval and st2 ∈ Event) and (tg(st1) ◀ tg(st2))) or ((st1 ∈ Event and st2 ∈ Interval) and (tg(st1) ▶ tg(st2))) or (st1 ∈ Event and st2 ∈ Event)
st1 TEMP_BEFORE st2	(st1 ∈ Interval and (st2 ∈ Interval or st2 ∈ Event)) or (st1 ∈ Event and (st2 ∈ Interval or st2 ∈ Event))
st1 TEMP_AFTER st2	
st1 TEMP_MEET_END st2	(st1 ∈ Interval and (st2 ∈ Interval or st2 ∈ Event)) or ((st1 ∈ Event and (st2 ∈ Interval or st2 ∈ Event)) and (tg(st1) ▶ tg(st2)))
st1 TEMP_MEET_START st2	
st1 TEMP_OVERLAP_END st2	(st1 ∈ Interval and st2 ∈ Interval) or ((st1 ∈ Interval and st2 ∈ Event) and (tg(st1) ◀ tg(st2))) or ((st1 ∈ Event and st2 ∈ Interval) and (tg(st1) ◀ tg (st2)))
st1 TEMP_OVERLAP_START st2	
st1 TEMP_INCLUDE_END st2	(st1 ∈ Interval and st2 ∈ Interval) or ((st1 ∈ Interval and st2 ∈ Event) and (tg(st1) ◀ tg(st2))) or ((st1 ∈ Event and st2 ∈ Interval) and (tg(st1) ▶ tg(st2)))
st1 TEMP_INCLUDE_START st2	
st1 TEMP_INCLUDE_PROPER st2	(st1 ∈ Interval and (st2 ∈ Interval or st2 ∈ Event)) or ((st1 ∈ Event and (st2 ∈ Interval or st2 ∈ Event)) and (tg(st1) ▶ tg(st2)))
st1 TEMP_EXCLUDE_END st2	(st1 ∈ Interval and st2 ∈ Interval) or ((st1 ∈ Interval and st2 ∈ Event) and (tg(st1) ◀ tg(st2))) or ((st1 ∈ Event and st2 ∈ Interval) and (tg(st1) ▶ tg(st2)))
st1 TEMP_EXCLUDE_START st2	
st1 TEMP_EXCLUDE_PROPER st2	(st1 ∈ Interval and st2 ∈ st2 ∈ Interval) or ((st1 ∈ Interval and st2 ∈ Event) and (tg(st1) ◀ tg(st2))) or (st1 ∈ Event and st2 ∈ Interval) or ((st1 ∈ Event and st2 ∈ Event) and (tg(st1) ◀ tg(st2)))

5.4.3 시간 비교 연산 제약조건

시간비교연산은 기존의 비교연산자(comparison operator : $>$, $<$, $=$, $\geq$, $\leq$)를 이용하여 간격 타입으로 표현된 두 시간의 크기를 비교하는 연산으로 연산의 종류에 관계없이 모두 아래의 제약조건을 만족하여야 하며, 두 피연산자의 시간 단위에 대한 제약조건은 없다.

【제약조건 8】 시간 비교 연산에 입력되는 피연산자의 타입 제약조건

(st1 시간비교연산 st2)에서 피연산자 st1과 st2의 타입은 다음과 같아야 한다.

(st1 Interval and (st2 Interval or st2 Duration))

5.4.4 시공간 연산 제약조건

시공간 연산의 피연산자에 대한 제약조건은 다음과 같이 피연산자로 입력되는 객체가 저장된 데이터베이스의 타입을 이용하여 정의한다.

【제약조건 9】 시공간 연산에 입력되는 피연산자의 타입 제약조건

시공간 연산	피연산자의 타입 제약조건
GEOPROJECTION(O)	O ∈ spatial database
TEMPPROJECTION(O)	O ∈ historical event spatial database or O ∈ historical interval spatial database or O ∈ rollback spatial database or O ∈ bi-temporal event spatial database or O ∈ bi-temporal interval spatial database
STAREACHANGERATE (O, st1, st2)	O ∈ POLYGON st1, st2 ∈ Event and st1 < st2 and tg(st1) = tg(st2)
STLENGTHCHANGERATE (O, st1, st2)	O ∈ POLYLINE st1, st2 ∈ Event and st1 < st2 and tg(st1) = tg(st2)
STDISTANCE(O, st1, st2)	O ∈ POINT st1, st2 ∈ Event and st1 < st2 and tg(st1) = tg(st2)
STSPEED(O, st1, st2)	O ∈ POINT st1, st2 ∈ Event and st1 < st2 and tg(st1) = tg(st2)

【제약조건 10】 시공간 연산 결과의 타입 제약조건

시공간 연산	피연산자의 타입 제약조건
r = GEOPROJECTION(O)	r = spatial data type
r = TEMPPROJECTION(O)	if O∈historical event spatial database then r = Event if O∈historical interval spatial database then r = Interval if O∈rollback spatial database then r = Interval if O∈bi-temporal event spatial database then r = (Event, Interval) if O∈bi-temporal interval spatial database then r=(Interval, Interval)
r = STAREACHANGERATE (O, st1, st2)	r = {(Interval, float)$_i$ | 1 ≦ i ≤ n}
r = STLENGTHCHANGERATE (O, st1, st2)	r = {(Interval, float)$_i$ | 1 ≦ i ≤ n}
r = STDISTANCE(O, st1, st2)	r = {(Interval, float)$_i$ | 1 ≦ i ≤ n}
r = STSPEED(O, st1, st2)	r = {(Interval, float)$_i$ | 1 ≦ i ≤ n}

6. 평가

본 장에서는 제안 데이터 모델을 이용하여 돌풍 관제 시스템에 구현한다. 실험에 적용한 돌풍 관제 시스템에 대한 응용은 공간 데이터베이스 관리 시스템인 GMS[24], MS Widnwos2000, 그리고 Visual C++를 이용하여 구현하였다. 실험에 이용한 이동 객체 데이터는 2003년 4월 23일 0시부터 2003년 4월 23일 18시를 유효 시간 구간 동안 사람과 구급차에 대해 2분, 돌풍에 대해서는 10분의 폴링 타임 주기를 설정하여 임의로 생성하여 GMS에 유지하며, 실험에 사용된 공간 테이블들에 대한 명세는 다음과 같다.

지한다. 이렇게 이동 객체의 정보를 분리 저장함으로써 실제 이동 객체 질의에서 현재 정보를 기반으로 하는 질의와 과거 이력 정보를 기반으로 하는 질의를 효과적으로 처리할 수 있다. GMS에서는 [22]에서 명시한 다양한 공간 객체들을 지원한다.

아래에서는 다양한 시공간 질의에 대해 공간 SQL로 변환하여 실행된 결과를 보인다.

(질의 1)에서는 홍길동의 현재 위치를 찾고, 찾은 위치와 구급차들간의 최소 거리값을 구한다. 다시 홍길동의 위치와 구급차간의 거리 값들 중에 최소 거리값보다 작거나 같은 거리값을 가지는 객체의 쌍을 찾는다. 이러한 과정은 현재 홍길동의 위치에서 가장 가까운 구급차를 구하는 연산이 된다. 이를 통해 가장

```
STIDTable (OID:INT, Name:STRING, Type:INT)
STPsnHisTable (OID:INT, DATE:DATE, Geometry:Spatial)
STAmbHisTable (OID:INT, DATE:DATE, Geometry:Spatial)
STTndHisTable (OID:INT, DATE:DATE, Geometry:Spatial)
STPsnCurTable (OID:INT, DATE:DATE, Geometry:Spatial)
STAmbCurTable (OID:INT, DATE:DATE, Geometry:Spatial)
STTndCurTable (OID:INT, DATE:DATE, Geometry:Spatial)
```

(질의 1) "현재 홍길동의 위치에서 가장 가까이 있는 구급차를 검색하라."

```
SELECT Name
FROM STIDTable
WHERE OID = (
   SELECT DISTINCT Amb.OID
   FROM STAmbCurTable Amb, STPsnCurTable Psn, STIDTable ID
   WHERE ID.Name = '홍길동' AND ID.OID = Psn.OID AND
            distance(Amb.Geometry, Psn.Geometry) <= (
      SELECT min( distance(Amb.Geometry, Psn.Geometry) )
      FROM STAmbCurTable Amb, STPsnCurTable Psn
      WHERE Psn.OID = (
            SELECT DISTINCT ID.OID
            FROM STIDTable ID, STPsnCurTable Psn
            WHERE ID.Name = '홍길동' AND ID.OID = Psn.OID ));
```

STIDTable은 시공간 객체의 식별자, 이름, 그리고 지오메트리 타입을 나타내는 속성 스키마이고, STPsnHisTable, STAmbHisTable, STTndHisTable은 사람, 구급차, 돌풍 등 시공간 객체의 이력 정보를 표현하기 위한 스키마이며, STPsnCurTable, STAmbCurTable, STTndCurTable 스키마는 가장 최근에 폴링된 이동 객체의 정보를 유

가까운 구급차의 식별자를 얻게 되고 이 값으로 MOIDTable에서 구급차의 이름을 검색하게 된다.

<그림 4>는 (질의 1)에 해당하는 공간 SQL 구문을 GMS의 입력으로 사용하여 수행한 결과 화면으로, 지도상에 홍길동의 현재 위치에서 가장 가까운 구급차를 표시하고 있다.

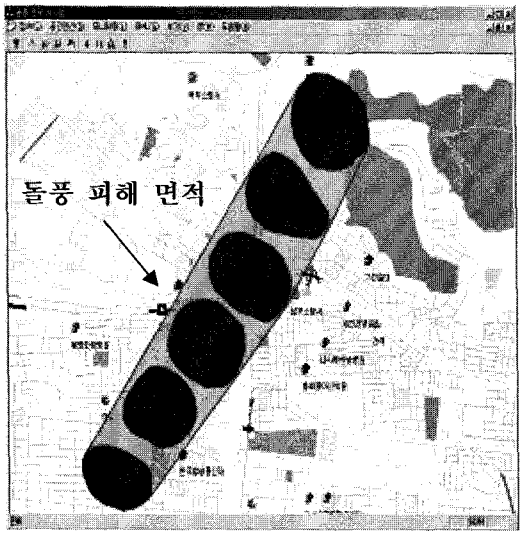
〈그림 4〉 질의 1 수행 결과

(질의 2)에서는 MOTndHisTable에서 돌풍 'T'가 최초 폴링된 시점부터 현재까지 이동한 영역들을 검색하여 각 폴링된 수만큼의 이동 영역들을 얻는다. 이 이동 영역들을 입력으로 convexhull 연산을 수행하면 이동 영역들의 최외곽의 볼록한 점으로만 구성되는 지오메트리(convex hull)가 생성된다. 이 지오메트리(convex hull)의 면적을 구함으로써 실제 돌풍이 지나온 영역을 검색할 수 있고 이는 돌풍의 피해 면적을 의미한다.

〈그림 5〉는 최초 돌풍이 측정된 시점부터 현재까지 6개의 돌풍 영역으로부터 MOTraverse 연산을 통해 전체 돌풍이 지나간 영역을 알 수 있다.

(질의 2) "현재까지 돌풍 T의 피해 면적을 검색하라."

```
SELECT convexhull(Geometry)
FROM STIDTable ID, STTndHisTable Tnd
WHERE ID.Name = 'T' AND ID.OID = Tnd.OID AND
        Tnd.DATE BETWEEN(MIN(DATE),MAX(DATE))
```



〈그림 5〉 질의 2 수행 결과

(질의 3)에서는 현재 'T'의 영역과 사람들의 위치를 입력으로 contains 연산을 수행하여 'T'영역내의 있는 사람들을 검색한다.

〈그림 6〉은 (질의 3)에 대한 실행 결과로 현재 돌풍 지역은 붉은 영역으로 표시되고 있으며, 이 영역안에 존재하는 사람들은 contains 연산을 통해 찾을 수 있다.

7. 결론

기존의 시공간 데이터 모델에서 시공간 연산들의 경우 시공간 객체와 관련된 시간정보는 시간 단위에 따른 연산 방법에 대한 정의의 미비로 연산 결과를 신뢰할 수 없고, 공간정보나 시간정보와 관련된 다양한 연산을 제공하지 않기 때문에 시공간 객체의 변화과정을 이용하는 다양한 분석기능을 수행할 수 없으며, 공간정보나 속성정보의 변화로 인하여 더 이상 현재시점에 유효하지 않는다는 문제점을 갖는다. 본 논문에서는 시간 정보를 포함하는 시공간 객체의 정보를 효율적으로 표현하고 저장할 수 있는 시공간 데이터 모델을 제안하였다. 제안 시공간 데이터모델에서의 시공간 연산은 시간정보와 공간정보를 피연산자로 하는 다양한 시공간 연산을 설계하였으며, 시공간 데이터의 무결성과 연산의 무결성을 유지하기 위한 제약조건을 제시하였다.

향후 연구과제로는 제안된 데이터모델을 이용하여 시공간 객체를 효율적으로 저장할 수 있는 기법에 대한 연구가 필요하며, 비공간정보에 대한 조건과 공간정보에 대한 조건, 그리고 시간정보에 대한 조건을 포함하는 질의를 간결하게 표현할 수 있는 시공간 질의어에 대하여 연구가 수행되어야 한다.

〈그림 6〉 질의 3 수행 결과

(질의 3) "현재 Tornado T의 영향권내에 속하는 사람을 검색하시오."

```
SELECT Name
FROM STIDTable
WHERE OID = (
    SELECT DISTINCT Psn.MOID
    FROM STPsnCurTable Psn, STTndCurTable Tnd, STIDTable ID
    WHERE ID.Name='T' AND ID.OID=Tnd.OID AND
          contains(Tnd.Geometry, Psn.Geometry) );
```

참고문헌

[1] R.H. Guting and et. Al., "A Foundation for Representing and Querying Moving Objects", ACM Transactions on Database Systems, Vol. 25, No.1, 2000, pp.1-42.

[2] M. Worboys, "A Unified Model for Spatial and Temporal Information", The Computer Journal, Vol. 37, No. 1, 1994.

[3] O. Wolfson, B. Xu, S. Chamberlain and L. Jiang, "Moving Objects Databases: Issues and Solutions", Proceedings of the 10th International Conference on Scientific and Statistical Database Management (SSDBM98), Capri, Italy, 1998.

[4] N.L. Sarda, "Extensions to SQL for Historical Databases", IEEE Transactions on Knowledge and Data Engineering, Vol. 2, No. 2, 1990, pp.220-230.

[5] A.P. Sistla and et. Al, "Modeling and Querying Moving Objects", Proceedings of the Thirteenth Int'l Conf. on Data Engineering (ICDE13), Birmingham, UK, 1997.

[6] G. Langran, Time in Geographic Information Systems, London: Taylor & Francis, 1993.

[7] M. Yuan, "Temporal GIS and Spatio-Temporal Modeling", Third International Conference/Workshop on Integrating GIS and Environmental Modeling, 1996.

[8] A. Tansel and et. al., Temporal Databases: Theory, Design, and Implementation, The Benjamin/Cummings, 1993.

[9] O. Wolfson and et. Al, "DOMINO: Databases fOr MovINg Objects tracking", Proc. of the International Conference on SIGMOD, 1999, pp.547-549.

[10] M.P. Armstrong, "Temporality in Spatial Databases", Proc. GIS/LIS '88, Vol. 2, 1988, pp. 880-889

[11] R.T. Snodgrass, I. Ann, "Temporal databases", IEEE Computer Magazine, 1986, pp. 35-42.

[12] K.K. Al-Taha, R.T. Snodgrass and M.D. Soo, Bibliography on Spatiotemporal Databases, SIGMOD RECORD, Vol. 22, No. 1, 1993.

[13] L. Forlizzi, R.H. Guting, E. Nardelli and M. Schneider, "A Data Model and Data Structures for Moving Objects Databases", In Proc. of the ACM SIGMOD Conf., 2000, pp.319-330.

[14] Open GIS Consortium, Simple Feature Specification for OLE/COM Revision 1.1, OpenGIS Project Document 99-050, 1999.

[15] M. Erwig, R. H. Guting, M. Schneider and M. Vazirgiannis, "Abstract and Discrete Modeling of Spatio-Temporal Data Types", 6th ACM Symp. on Geographic Information Systems (ACMGIS '98), 1998, pp. 131-136.

[16] M.Erwig and et. Al, "Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases", GeoInformation, Vol3, No.3, 1999. pp.269-296.

[17] S. Ram and J .S. Park, "Modeling Spatial and Temporal Semantics in a Large Heterogeneous GIS Database Environment", Proceedings of the 2nd Americas Conference on Information Systems (AIS '96), Phoenix, Arizona, 1996, pp. 683-685.

[18] H.A. Lee et. Al, "Design of a spatiotemporal object model for 2D geographic objects", KIPS Journal VOL.9-DNO.01 2002, pp.43~56.

정원일

1998년 인하대학교 전자계산공학과 졸업(공학사)

1998년~현재 인하대학교 대학원 전자계산공학과 박사과정

관심분야: 공간데이터베이스 클러스터, 이동체 데이터베이스, GML, LBS 등

배해영

1974년 인하대학교 응용물리학과 졸업(공학사)

1978년 연세대학교 대학원 전자계산학과 졸업(공학석사)

1989년 숭실대학교 대학원 전자계산학과 졸업(공학박사)

1985년 Univ. of Houston 객원 교수

1992년~1994년 인하대학교 전자계산소 소장

1982년~현재 인하대학교 전자계산공학과 교수

1999년~현재 지능형 GIS 연구센터 소장

2000년~현재 중국 중경우전대학교 대학원 명예 교수

관심분야 : 공간데이터베이스, 지리정보시스템, 분산 데이터베이스, 데이터베이스 클러스터 멀티미디어 데이터베이스 등