시공간 데이터베이스를 위한 *history* 집계 연산자

이 종 연
삼척대학교 정보통신공학과
jongyun@samchok.ac.kr

History Aggregate Operator for Spatio-Temporal Databases

Jong-Yun Lee
Dept. of Information and Communication Eng., Samchok National University

요 약 기존의 관계형 데이터베이스 시스템은 기본적으로 count, max, min, sum, avg의 집계 함수(aggregate functions)를 제공하며, DBMS에 따라 다양한 집계 연산자를 추가로 지원한다. 시공간 데이터베이스는 기본적인 공간 정보뿐만 아니라 시간 흐름에 따른 이력 정보를 취급하므로 데이터베이스로부터 자유로운 이력(history) 정보의 검색 기능이 요구되고 있다. 따라서, 본 연구에서는 시공간 데이터베이스로부터 이력을 자동으로 검색할 수 있는 새로운 집계 연산자, *'history'*를 제안하고, 그 처리 알고리즘과 SQL3에서 탐색 질의 표현법을 제안한다. 결과적으로, 제안된 이력 집계 연산자는 향후 SQL3의 질의 표현 능력의 제고에 기여할 것이다.

1. 서론

시공간 데이터베이스(Spatio_temporal databases)는 지금까지 주로 공간과 시간 데이터베이스의 독립적인 형태로 진행되어 왔다. 특히 공간 데이터베이스는 주로 지리정보 시스템(GIS)을 바탕으로 공간 자료의 저장, 분석 및 처리, 공간 질의 [1,2,3] 및 최적화, 공간 색인 구조, 시각화 등의 분야에서 연구되어 왔다. 뿐만 아니라 시간 데이터베이스(Temporal Databases)도 시간 데이터 모델[4], 시간 질의 [5,6] 및 질의 최적화, 시간 색인 등의 분야에서 연구가 진행되어 왔다. 하지만 최근에는 시공간 데이터베이스라는 새로운 통합 유형이 등장하면서 다시 시공간 데이터 모델 [7,8], 시공간 색인 구조와 시간 및 공간 연산자의 통합[9] 등의 여러 분야에서 연구가 활발히 진행되고 있다. 하지만 최근에는 공간 및 시간 질의 연구와 더불어 과거의 이력 질의를 자유롭게 할 수 있는 새로운 연산자의 필요성이 제기된 바 있다[9,10].

따라서, 본 논문에서는 시공간 데이터베이스에서 과거의 이력 정보를 자유롭게 검색할 수 있는 새로운 집계 연산자, *'history'*를 제안하고자 한다. 뿐만 아니라, 이력 집계 연산자(history aggregate operator)의 처리 알고리즘과 SQL3에서의 이력 질의 표현 방법을 제시할 것이다. 특히 제안된 이력 집계 연산자는 SQL3의 시공간 복합 질의에서 질의 표현의 표현력 제공에 기여할 것이다.

본 논문의 구성은 다음과 같다. 제 2장은 시공간 데이터베이스의 스키마 구조와 이력 집계 연산자를 정의하고 그 처리 알고리즘을 제안한다. 제 3장은 SQL3에서 이력 질의의 예를 통해 질의 표현 방법과 연산의 처리 과정을 검토한다. 끝으로 제 4장은 논문의 결론을 요약한다.

+ 이 논문은 2001년도 삼척대학교 자체학술연구비 지원에 의해 수행되었음.

2. 이력 집계 연산자

2.1 시공간 데이터베이스 구성

제안된 시공간 데이터베이스는 '공간 릴레이션(feature relation)', '공간 이력 릴레이션(feature history relation)', '합병 릴레이션(merge relation)'의 세 개 테이블로 구성되어 있다고 가정한다. 공간 릴레이션에서 객체는 공간과 속성 정보에 의해 기술되며, 이력 정보는 공간 이력 릴레이션에서 이력 포인터의 연결 정보를 통해 관리된다. 뿐만 아니라 지형객체의 합병 정보는 별도의 합병 릴레이션에 수록된다. 합병 릴레이션은 단지 공간 이력 릴레이션과 공간 릴레이션에서 객체간의 이력 포인터 정보와 그 유효기간(valid time)만을 포함한다. 이 세가지 릴레이션의 스키마 구조는 세부적으로 다음과 같이 정의한다.

- 공간 릴레이션(feature relation):
  (*oid : number, fid : character, type : number, mbr : coordinate, points : coordinate, prev : historical pointer of object, VTs : valid time period, TT : transaction time period*)
- 공간 이력 릴레이션(feature history relation):
  (*oid , fid, type, mbr, points, prev, VT, TT*)
- 합병 릴레이션(merge relation):
  (*oid, fid, hid : character, oid, VT, TT*)

여기서, 각 변수는 다음과 같은 의미를 지님.

oid: 릴레이션에서 객체 식별자
fid: 공간 또는 공간 이력 릴레이션에서 공간 객체의 식별자(예: 지번)
type: 공간 자료형 (예: 점, 선, 다각형, 텍스트)
mbr: 공간객체를 포함하는 최소경계 사각형의 최소 및 최대 (x, y) 좌표값들
points: 공간객체를 구성하는 일련의 좌표들의 집합
VTs: 유효시간의 시작시간
VTe: 유효시간의 종료시간

VT: 유효시간의 시작 및 종료시간을 표현하는 기간 *TT*: 데이터베이스 트랜잭션의 수록시간으로 로 시작 시간과 종료시간을 나타냄.

단, 공간 릴레이션은 유효기간의 시작시간(VTs)만을 포함하므로 시공간 데이터베이스 시스템는 기존의 질의 처리기(query processor)를 그대로 사용할 수 있도록 하였고, 모든 유효시간의 종료시간은 현재(now)로 간주한다. 여기서, 유효시간(VT)은 공간객체가 실질적으로 공간상에서 유효한 기간으로 시간값의 범위는 {t1, t2, t3, ..., tk, now}를 가지며, 'now'는 현재 시간을 나타낸다. 반면 거래시간(transaction time)은 데이터베이스 수록시간으로 {t1, t2, t3, ..., tk}∪{UC}의 시간 범위를 가지며 'UC'는 'Until Changed'의 약자이다. 하지만 이하의 릴레이션 스키마에서 mbr, TT의 속성은 간편성을 위해 고려하지 않는다.

2.2 이력 집계 연산

시공간 데이터베이스는 공간객체의 현재, 과거, 미래의 모든 정보를 포함할 수 있다. 특히 과거의 이력 데이터는 공간적 분할(split), 병합(merge), 재할당(re- allocate) 사건 등으로 발생되어 관리되며, 추후 사용자에 의해 자유로운 이력 검색(history retrieval)이 필수적이다. 따라서, 본 논문에서는 SQL3의 선택(select)문에서 과거 이력정보를 자유롭게 검색할 수 있는 새로운 집계 연산자 *'history'*를 제안하며, 그 세부적인 기능은 다음의 [정의1]과 같다.

【정의1】 이력 집계 연산자, *'history'*는 저장된 시공간 데이터베이스로부터 공간 객체의 모든 공간 및 속성 이력 정보를 검색하는 선택 연산으로, 그 처리 결과는 공집합(∅)이거나 다수의 이력 객체를 포함할 수 있다. □

[정의1]에서 기술한 바와 같이, 집계 연산자의 처리 결과는 객체의 이력 정보가 없는 경우 공집합이거나 유효기간에 포함되는 다수의 이력 객체(historical objects)를 포함할 수 있다. 제안된 시공간 데이터베이스 구조에서 이력 집계 연산자의 처리 알고리즘은 <그림 1>과 같다. <그림 1>에서 이력 집계 연산은 프로시저 *'allHistory()'*와 함수 *'History()'*에 의해 처리된다. 프로시저 *'allHistory()'*는 공간객체의 식별자와 원하는 속성 유형을 입력받고, 공간 릴레이션에서 원하는 공간객체가 발견되면, 그 객체의 이력 포인터를 통해 <공간 이력 릴레이션>으로부터 모든 이력을 검색한다 (step 1.1 - 1.2). 그렇지 않으면 <공간 이력 릴레이션>으로부터 입력된 공간객체를 탐색한다 (step 1.3). 만약 이력 정보가 릴레이션에 존재하면, 함수 *'History()'*는 이력 포인터가 'null'이거나 유효시간 'VT'가 유효(valid) 범위에 존재할 때까지 반복적으로 모든 이력 객체를 검색하여 반환한다 (단계 '2.3 - 2.10'). 단, 검색 과정에서 이력 포인터가 'MERGE'이면, 검색 연산은 <합병 릴레이션>으로부터 *'History()'* 함수의 순환적 호출(recursive calls)에 의해 실행된다 (step 2.5 - 2.9).

```
Procedure allHistory(char * fid, int designated_attribute_type)
Input  : feature identifier(fid) and designated attribute type of spatial object
Output : spatial or non-spatial historical information of feature
Step 1.0 Found = FALSE;
Step 1.1 Find the input record from the current feature relation;
Step 1.2 If (found == TRUE) /*Retrieve all the histories of the input record*/
           History(previous pointer of current feature, designated_attribute_type);
Step 1.3 Else {
           Find the input record from the feature history relation;
           If (found == TRUE)
             History(previous pointer of historical feature,
                     designated_attribute_type);
         }

Function int History(char id, int designated_attribute_type)
Step 2.0 Int oid; /* id is an identifier of pointer */
Step 2.1 If (id == NULL or VT is a valid time period)
           Return;
Step 2.2 Else {
Step 2.3     oid = OldFeature[id].ObjectIdentifier ;
Step 2.4     do { /*retrieve all the merged histories from merge relation*/
Step 2.5        If (id == MERGE) {
Step 2.6           For (I = 0; I < SIZE; I++) {
Step 2.7              If ((fid of oid is found from the merged relation) and
                        (mbr intersected by one another))
Step 2.8                designated_attribute_type);
                      }
Step 2.9              break; History(merge[i].PreviousPointer,
                   }
Step 2.10       Else {
                  Output all historical objects from the feature history relation;
                  id = OldFeature[id].PreviousPointer;
                } /* if-else statement */
Step 2.11    } while (id != NULL); /* do-while loop */
Step 2.12 } /* if-else statement */
Step 2.13 Return;
```

그림 1. 'history' 집계 연산의 처리 알고리즘

뿐만 아니라, 이력 집계 연산자 *'history'*의 인터페이스는 *'*.history'* 또는 *'history.*'*의 2가지 표현이 가능하지만, SQL3의 일반적인 표현 사례를 따라 후방 표현법(postfix notation)을 채택하였다. 다음의 절은 제안된 이력 집계 연산자를 이용한 시공간 질의 표현의 예를 통해 SQL3의 질의 표현 능력의 향상을 확인할 수 있다.

3. 질의 표현 및 평가

공간에서 공간적 크기 변화를 발생시키는 주요 공간 연산(major spatial operations)에는 분할, 합병, 재할당 등이 존재한다. 이 장에서는 이러한 주요 공간 연산으로 발생한 시공간 데이터베이스의 예(그림 2)를 통해 앞에서 제안한 이력 집계 질의의 수행 과정을 검토할 것이다. 먼저

제안된 시공간 데이터베이스는 초기에 (1, 100, polygon, (0,0,0,5,6,5,5,0), null, [96/1, 96/6))과 (5, 101, polygon, (5,0,6,5,8,5,8,0), null, [96/1, 98/2])를 시작으로 분할, 재할당, 합병, ID 변경의 연산을 거쳐 그림 2와 같다고 가정한다. 여기서 시공간 데이터베이스는 객체의 최소경계 사각형(mbr), 데이터베이스 수록 시간인 거래시간(transaction time) 속성을 제외한 나머지 요소들은 앞의 2.1절에서 제안된 스키마 구조와 일치한다.

<Feature relation>:
{(3, 100, polygon, (0,0,0,5,2,5,2,0), 6, 98/8),
(2, 101, polygon, (2,0,2,5,8,5,8,0), MERGE, 98/2)}
<Feature history relation>:
{(6, 100-1, polygon, (0,0,0,5,2,5,2,0), 2, [97/12, 98/8)),
(5, 101, polygon, (5,0,6,5,8,5,8,0), null, [96/1, 98/2,
(4, 100-2, polygon, (2,0,2,5,6,5,5,0), 3, [97/12, 98/2,
(3, 100-2, polygon, (3,0,2,5,6,5,5,0), 1, [96/6, 97/12,
(2, 100-1, polygon, (0,0,0,5,2,5,3,0), 1, [96/6, 97/12,
(1, 100, polygon, (0,0,0,5,6,5,5,0), null, [96/1, 96/6)
<Merge relation>:
{(1, 101, 101, 5,[96/1, 98/2)),
(2, 101, 100-2, [97/12, 98/2))

그림 2. 시공간 데이터베이스의 예

다음은 시공간 데이터베이스로부터 이력 집계 질의의 예로서 논문 [9]에서 제안한 시공간 참고 매크로(spatio-temporal reference macros)와 이력 집계 연산자 '*history*'를 사용한 SQL3 질의의 예를 통해 질의 표현력 방법을 검토할 것이다.

Query 1: *"Find all the histories of a spatial object,100, since 1996* in Fig. 2".

```
> SELECT *.HISTORY FROM parcels p
  WHERE fid = 100 and
  p.VALID OVERLAPS Period Jan-01-96, now;
```

위의 질의는 그림 2의 '공간 릴레이션'으로부터 식별자 '100'과 유효기간 'Jan-01-96, now'를 가진 레코드, (3, 100, polygon, (0,0,0,5,2,5,2,0), 6, 98/8)를 검색할 수 있다. 이 레코드의 이력 정보는 이력 포인터 '6'을 가지고, 다시 '공간 이력 릴레이션'으로부터 주어진 유효시간과 부합되는 한 반복적으로 이력을 탐색한다. 탐색 과정에서 이력 포인터가 'null'를 만나면 종료한다. 위의 질의의 최종 결과는 다음과 같다.

{(3,100, polygon, (0,0,0,5,2,5,2,0), 6, 98/8),
(6,100-1, polygon, (0,0,0,5,2,5,2,0), 2, [97/12, 98/8)),
(2,100-1, polygon, (0,0,0,5,2,5,3,0), 1, [96/6, 97/12)),
(1,100, polygon, (0,0,0,5,6,5,5,0), null, [96/1, 96/6))}

위의 질의를 통해 보았듯이 본 논문에서 제안한 이력 집계 연산자와 시공간 질의 매크로[9]를 통해 시공간 질의 표현의 편리성을 확인할 수 있다.

4. 결론

본 논문은 시공간 데이터베이스를 위한 시간 및 공간 질의의 통합(integration) 연구의 일환으로 새로운 이력 집계 연산자를 제안하고, 그 처리 알고리즘과 표현법을 제안하였다. 또한 SQL3에서 시공간 이력 질의의 표현을 통해 질의 수행 과정과 그 결과를 검토하였고, 결과적으로 SQL3에서 이력 질의 표현력의 향상과 편리성이 기대된다. 특히 지금까지 연구되어 온 시공간 데이터 모델, 공간 및 시간 질의의 통합, 시공간 색인와 더불어 공간 및 시간 데이터베이스의 질의 통합(query integration) 연구에 크게 기여하게 될 것이다. 앞으로의 연구로는 시간 및 공간 질의의 통합에 따른 시공간 질의 처리의 최적화 (Spatio-temporal Query Processing Optimization) 연구가 필요할 것으로 사료된다.

참고문헌

[1] M. J. Egenhofer, Spatial SQL: A Query and Presentation Language, IEEE Transactions on Knowledge and Data Engineering, Vol. 6, No. 1, Feb. 1994.

[2] R. H. Guting, Gral: An Extensible Relational Database System for Geometric Applications, Proc. of the Fifteenth International Conference on Very Large Data Bases, Amsterdam, pp.33-43, 1989.

[3] N. Roussopoulos, C. Faloutsos, and T. Sellis, An Efficient Pictorial Database System for PSQL, IEEE Transactions on Software Engineering, Vol. 14, No. 5, pp. 639-650, 1988.

[4] R. T. Snodgrass, The TSQL2 Temporal Query Language, The TSQL2 Language Design Committee, Kluwer Academic Publishers, 1995.

[5] Jensen, C. S., Soo, M. D., and Snodgrass, R. T., "Unifying Temporal Data Models via a Conceptual Model", Information Systems, Vol. 19, No. 7, pp. 513-547, 1994.

[6] Nikos A. Lorentzos and Yannis G. Mitsopoulos, "SQL Extension for Interval Data", IEEE Transactions on Knowedge and Data Engineering, Vol. 9, No.3, pp. 480-499, May/June 1997.

[7] D. Peuquet and N. Duan, An event-based spatiotemporal data model (ESTDM) for temporal analysis of geographical data, Information Systems, Vol. 9, No. 1, pp. 7-24, 1995.

[8] C. Claramunt, Managing Time in GIS: An Event-Oriented Approach, Recent Advances in Temporal Databases, in: J. Clifford, and A. Tuzhilin (Eds.), Workshops in Computing Series, Berlin:Springer-Verlag, pp. 23-42 (1995).

[9] Jong Yun Lee, Dong Ho Kim and Keun Ho Ryu, "Spatiotemporal Relational Model and Refere- nce Macros for Spatial and Temporal Data", 충북대학교 컴퓨터정보통신연구소 논문지, 제 6권 제 2호, pp.119-129, 1998.

[10] Jong Yun Lee, 공간객체의 이력 관리를 위한 시공간 데이터 모델링, 충북대학교 박사학위 논문, 1999.