論文2001-38CI-3-4

능동적 시간 데이터베이스에서 능동 규칙의 의미 모델

(Semantics of Active Rule in Active Temporal Databases)

南 光 祐 *, 柳 根 浩 *, 李 鍾 勳 **

(Kwang Woo Nam, Keun Ho Ryu, and Jong Hun Lee)

요 약

이 논문은 시간 데이터베이스에 능동 규칙 시스템을 결합한 능동적 시간 데이터베이스의 능동 규칙 및 실행 모델을 제안한다. 기존의 시간 데이터베이스 능동 규칙 연구들이 시간 릴레이션에 대한 조건-조치 형태의 규칙 지원에 머물렀던데 비하여 제안하는 능동 규칙 모델은 유효 시간 및 트랜잭션 시간을 지원하는 이원 시간 릴레이션에 대해 SQL3 트리거의 기능 모두를 시간 데이터베이스 상에서 지원한다. 즉, 전이 릴레이션 및 전이 변수, 조건-조치의 튜플 및 문 단위 수행을 포함한다. 또한, 능동 규칙의 유효 시간 정의 및 유효 시간 사건 등의 새로운 개념을 제시한다.

Abstract

This paper propose an active rule and its semantic model integrating active rule system into temporal database. Most of previous works have been focused on condition-action(CA) rule for temporal relations. However, proposed active rule model supports full SQL3 trigger functionalities including transition variable and table as well as tuple-level and statement-level active rules in E-CA granularity. A bi-temporal concept of active rule and valid-time event are also proposed newly.

Ⅰ. 서 론

능동적 시간 데이터베이스 시스템(active temporal database system)은 시간 데이터베이스 시스템[1]에 능동 규칙 시스템[2]을 결합한 진보된 데이터베이스 시스템으로 과거 데이터를 바탕으로 능동적으로 데이터베이스의 상황을 판단하여 적절한 조치를 취할 수 있다[3,4]. 이 시스템은 추세 분석, 상태 관찰을 통한 경보, 고장에 대한 예측, 금융 시스템 및 공정 제어 등과 같은 고도의 응용에 효과적으로 대응할 수 있다. 실제 플랜트 컴퓨터 시스템(plant computer system), 주식 관리 시스템(stock management system), 네트워크 데이터 관리 시스템(network data management system)과 같은 응용들은 능동적 시간 데이터베이스 기능을 필요로 하는 미래 데이터베이스 응용들로 지적되고 있다[5].

능동 데이터베이스 시스템에 시간 개념 지원을 확장하려는 연구들은 그 동안 계속적으로 진행되어 왔다. 비교적 근접한 형태의 연구는 Chomicki[6]와 Sistla[7,8]의 연구로 시간 데이터베이스가 아닌 일반 능동 데이터베이스 상에서 보조 릴레이션과 능동 규칙을 사용하여

* 正會員, 忠北大學校 컴퓨터科學科
(Dept. of Computer Science Chungbuk National Univ.)
** 正會員, 韓國電子通信研究院 컴퓨터소프트웨어研究所 映像處理研究部
(GIS Research ETRI)
※ 본 논문은 한국과학재단 특정기초(1999-2-303-006-3)과제와 한국전자통신연구원의 "4D 데이터제공자 컴포넌트 개발" 위탁과제의 연구비 지원에 의해 수행되었음.
接受日字:2000年2月14日, 수정완료일:2001年4月24日

PTL(past temporal logic) 등을 바탕으로 한 소급 갱신 및 선행 갱신 처리를 지원하고 있다. 또한 완전한 형태의 E-C-A(event-condition-action) 규칙이 아닌 C-A 형태를 지원한다.

시간 데이터베이스를 기반으로 능동 규칙 시스템을 결합하려는 연구는 Etzion 등이 시간 속성을 갖는 데이터 모델 상에서 능동 규칙을 지원하기 위한 PARDES[3]과 TALE[9] 등이 있다. 그러나, 이들에 의해 제시된 능동 시간 규칙은 C-A 형태를 가지며 사건을 발생시킨 튜플이나 테이블 상태를 조건, 조치 등에서 참조할 수 있는 전이 릴레이션이나 전이 튜플과 같은 기능을 제공하지 않는 초보적인 수준의 규칙만을 제공한다. 이외에 많은 연구들이 시간 데이터 모델을 고려하지 않은 일반 데이터베이스 상에서의 능동 규칙에 시간적 사건이나 시간 관련 표현을 포함하는 조건부의 표현을 가능하게 해주는 형태이다.

한편, 현재 데이터베이스 질의어 표준으로 제정된 SQL3는 능동 규칙을 지원하기 위하여 트리거를 포함하고 있다. 이와 함께 시간 데이터베이스 모델 및 질의어의 표준도 SQL3와 상호 변환 가능한 SQL/Temporal이 제안되기도 하였다[10]. 이 논문에서 제안하고 있는 능동 규칙 모델은 SQL/Temporal 데이터 모델을 기반으로 SQL3 트리거의 모든 기능을 시간 데이터베이스 상에서 지원하도록 확장한다. 즉, 유효 시간 릴레이션, 트랜잭션 시간 릴레이션, 이원 시간 릴레이션에 대하여 전이 릴레이션 및 전이 변수, 조건-조치의 튜플 및 문단위 수행을 지원 할 수 있다. 또한, 능동 규칙을 이원 시간 릴레이션에 저장함으로써 능동 규칙의 활성화 시간을 유효 시간으로 정의 할 수 있으며 트랜잭션 시간을 통해 능동 규칙의 변화 이력을 추적할 수 있는 기능을 제공한다. 뿐만 아니라, 시간 데이터베이스 상의 특정 유효 시간 영역에 대한 삽입, 삭제, 갱신에 대한 능동 규칙을 정의할 수 있도록 유효 시간 사건 등의 새로운 개념을 제시한다.

논문의 제 2장에서는 능동 규칙 지원을 위한 시간 데이터베이스의 기본 개념에 대한 정의와 사건을 유발하는 데이터 수정 연산문의 의미에 대해 살펴보고, 제 3장에서는 능동 규칙 언어의 의미, 그리고 능동 규칙에서 사용되는 전이 릴레이션과 전이 변수, 사건의 의미를 설명한다. 제 4장에서는 2장과 3장에서 제안된 의미를 이용한 실제 능동 규칙의 의미를 예제를 사용해서 보이고 있다. 또한 제 5장에서는 이 논문의 시간 데이터베이스 능동 규칙의 구현을 위한 모델을 제공한다. 제 6장은 기존의 연구와 비교를 수행하며, 제 7장에서 결론을 맺는다.

Ⅱ. 능동 규칙 지원을 위한 시간 데이터베이스

이원 시간 관계형 데이터 모델에서 객체는 유효 시간(valid time)과 트랜잭션 시간(transaction time)이라고 하는 두 개의 축 상에 있는 시간 값 쌍과 연관된다. 유효 시간은 시간의 흐름을 따라 변화하는 실세계의 사실을 모델링 함에 있어 연관된 시간을 모델 속에 표현하기 위해 사용되며 트랜잭션 시간은 그 데이터가 실제 데이터베이스에 반영되고 삭제된 시간을 모델링 하기 위해 사용되며 트랜잭션 로그의 시간이다. 데이터베이스에서 시간 값의 표현은 한정된 정확도를 가지며 그 가장 작은 시간 단위를 크로논(chronon)이라고 한다. 이 때 시간 도메인 $D_T=\{C_1,...,C_j\}$이며, 유효 시간 도메인 $D_{VT}=\{c_1', \cdots, c_k'\}\cup\{FOREVER\}$이고 $FOREVER$는 c_k보다 큰 값을 나타내기 위한 상수이다, 트랜잭션 시간 도메인 $D_{TT}=\{c_1'', \cdots, c_l''\}\cup\{UC\}$이고 UC는 아직 튜플의 트랜잭션 시간이 종료되지 않았음(until changed)을 표현하는 상수이다. 이 때 유효 시간과 트랜잭션의 시간 값에 대하여 다음에서 정의한다.

[정의 2.1] 유효 시간과 트랜잭션 시간 값은 값 내에 포함되는 크로논들의 하한 t^-와 상한 t^+를 갖는 시간격(period) $[t^- - t^+)$의 형태로 표현되며 다음과 같이 정의된다.

$$[t^- - t^+) = \begin{cases} \{t | t^- \le t < t^+\} & \text{if } t^- < t^+ \\ \varnothing & \text{otherwise} \end{cases}$$

즉, 시간격의 시작 값 t-가 끝 값 t+보다 작다면, 시간격은 t-는 포함하나 t+는 포함하지 않은 t-와 t+의 값들로 구성된다. t-가 t+보다 크다면 시간격은 공집합이다. 시간격에 대한 합집합과 교집합 연산은 다음과 같다.

[정의 2.2] $t_1^-, t_1^+, t_2^-, t_2^+$이 DT에 있을 때 시간격에 대한 차분(−)과 교집합(∩)은 다음과 같이 정의된다.

$$[t_1^- - t_1^+) - [t_2^- - t_2^+)$$
$$= \begin{cases} \{[t_1^- - t_2^-), [t_2^+ - t_1^+)\} & \text{if } t_1^- < t_2^+ \wedge t_2^- < t_1^+ \\ \{[t_1^- - t_1^-\} & \text{otherwise} \end{cases}$$
$$[t_1^- - t_1^+) \cap [t_2^- - t_2^+) = [\max(t_1^-, t_2^-), \min(t_1^+, t_2^+))$$

시간격의 차분은 첫 번째 라인의 경우처럼 겹침 부분이 존재할 경우 0,1 또는 2개의 시간격을 반환하게 되며 겹침이 존재하지 않으면 원 시간격을 반환하게 된다. 교집합 연산은 겹침 부분을 포함하는 시간격을 반환한다.

시간 데이터베이스의 릴레이션들은 유효 시간과 트랜잭션 시간에 대한능동간 값의 지원 여부에 따라 시간 지원을 갖지 않는 표준적인 일반 릴레이션, 유효 시간 값을 갖는 유효 시간 릴레이션(valid-time relation), 트랜잭션 시간 값을 갖는 트랜잭션 시간 릴레이션(transaction-time relation), 유효 시간과 트랜잭션 시간 값 둘 다를 갖는 이원 시간 릴레이션(bi-temporal relation)으로 구분된다. 이 논문에서는 각 릴레이션에 대하여 모두 기술하지 않고 두 시간값 모두를 갖고있는 이원 시간 릴레이션만을 대상으로 함으로써 나머지 세 개의 릴레이션에도 적용가능함을 보인다.

[정의 2.3] 명시적인 속성들의 집합 $A1, \ldots, An$, 튜플의 대리자 SURROGATE, 유효 시간 값을 위한 타임스탬프 VT, 트랜잭션 시간을 위한 타임스탬프 TT를 갖는 이원 시간 릴레이션 스키마 Rbt를 다음과 같이 정의한다.

$$R_{bt} = SURROGATE:(A_1, \ldots, A_n \| VT, TT)$$

여기서, 각 R_{bt}의 인스턴스들을 나타내기 위해 r_{bt}의 표기을 사용하며, 릴레이션 내의 튜플 인스턴스를 위해 $s\langle a_1, \ldots, a_n | vt, tt\rangle$의 형태를 사용한다. $SURROGATE$는 튜플간의 동등성 비교를 위한 식별자로 튜플의 oid나 rowid 등을 의미한다. 유효 시간 VT와 트랜잭션 시간 TT는 시간격 $[t_1 - t_2)$ 의 형태를 가지며 각 값의 하한과 상한값은 VT^-, VT^+, TT^-, TT^+의 형태로 표기된다. 또한 어떤 튜플 x의 특정 속성 값에 접근하고자 할 경우 t[속성이름]과 같은 표현을 사용한다. 예를 들면 튜플 x의 SURROGATE 속성값은 x[SURROGATE], 유효 시간 값은 x[VT]의 표현에 접근할 수 있다. 그리고, 일반 속성들의 집합 $\{A_1, \ldots, A_n\}$ 전부를 표현하고자 할 경우 약식 표기 A를 사용한다. 즉, $A = \{A_1, \ldots, A_n\}$이며, 이원 시간 튜플 t와 x가 있을 때 t[A]=x[A]와 같이 사용되었다면 이것은 1부터 n사이의 모든 i에 대해 $t[A_i] = x[A_i]$임을 나타낸다.

[정의 2.4] r'_{bt}와 r''_{bt}가 이원 시간 릴레이션이고 이원 시간 튜플이 t라고 할 때 합집합($\cup^{bt}$)연산자는 다음과 같이 정의된다.

$$r'_{bt} \cup^{bt} r''_{bt} = \{t | t \in r'_{bt} \wedge t \in r''_{bt}\}$$

이원 시간 합집합 연산자는 대상 릴레이션이 유효 시간과 트랜잭션 시간을 포함하는 이원 시간 릴레이션이라는 것을 제외하면 일반 관계 대수 합집합과 동일하다.

시간 데이터베이스의 데이터에 대한 수정 연산은 일반적인 관계형 데이터베이스와 같이 릴레이션에 대한 삽입, 삭제, 갱신절에 의해 수행되며 다수의 튜플들로 이루어진 튜플집합을 그 대상으로 한다. 이 논문에서는 시간 데이터베이스에서의 수정 연산절들을 보이고 그 연산의 의미를 명령절이 선언하고 있는 전체 대상 튜플 집합 선언의 의미와 실제 적용되는 각 튜플단위의 의미로 구분하며, 전자는 $op^{statement}$의 형태로 후자는 op^{row}의 표기를 사용한다.

이원 시간 릴레이션 r_{bt}에 대해 유효 시간 vt동안 유효한 튜플 $(a_1, \ldots, a_n)$의 삽입은 다음과 같은 일반적인 형식을 갖는다.

VALIDTIME vt INSERT INTO r_{bt} VALUES
$(a_1, a_2, \ldots, a_n)$

유효 시간 vt는 릴레이션의 특성에 따라 PERIOD '$[t_1$-$t_2)$'와 같은 시간격 표현이 오거나 DATE 't3'와 같은 시점 표현이 올 수 있으며, 만약 VALIDTIME 구문이 정의되지 않는다면 디폴트 유효 시간은 PERIOD '[CT-FOREVER)'를 의미한다. 삽입되는 튜플의 트랜잭션 시간은 유효 시간과 관계없이 현재 시간(CT)로부터 이 튜플이 다음 갱신이 될 때까지(UC)를 의미한다. 위의 삽입절의 의미는 다음과 같이 관계 해석(calculus)식으로 표현된다.

$$insert_{bt}^{statement}(r_{bt}, vt, (a_1, a_2, \ldots, a_n))$$

$$= {}^{\diamond} r'_{bt} \leftarrow r_{bt} \Upsilon^{bt} insert^{row}_{bt}(r_{bt}, vt, (a_1, a_2, \ldots, a_n))$$
$$= {}^{\diamond} r'_{bt} \leftarrow r_{bt} \Upsilon^{bt} \{\langle a_1, a_2, \ldots, a_n \| vt, [CT-UC) \rangle\}$$

식 (1)의 첫 번째 라인의 $insert^{statement}_{bt}$은 이원 시간 릴레이션에 대한 절 단위의 삽입 의미임을 보이고 있으며, 두 번째와 세 번째 라인에서 보이는 것과 같이 유효 시간 vt를 갖는 단일 튜플 (a_1, a_2, ..., a_n)을 기존의 이원시간 릴레이션 rbt에 합집합 연산을 수행하게 된다. 삽입절은 위의 일반적인 형식외에 INSERT ... SELECT와 같은 형태로 사용될 수도 있으나 이 논문에서는 논리의 단순화를 위해 INSERT ... VALUES 구문만을 대상으로 하지만 위의 삽입의미는 전자의 구문에 쉽게 확장가능하다.

이원 시간 릴레이션 r_{bt}에 대해 조건 c를 만족하는 튜플들의 유효 시간 vt를 삭제절은 다음과 같은 일반적인 형식을 갖는다.

VALIDTIME *vt* DELETE FROM r_{bt} WHERE *c*

유효 시간 *vt*는 삽입절 VALIDTIME 구문과 같이 시간격 표현이나 시점 표현이 올 수 있으며, 만약 VALIDTIME 구문이 정의되지 않는다면 디폴트 유효 시간은 *PERIOD '[CT-FOREVER)'*를 의미한다. 또한 삭제절의 대상은 이미 논리적으로 삭제된 튜플에는 수행될 수 없으므로 TT^+가 UC인 튜플들로 한정된다. 위의 삭제절의 의미는 다음과 같이 정의된다.

$$delete^{statement}_{bt}(r_{bt}.vt.c)$$
$$= {}^{\diamond} r'_{bt} \leftarrow \{t | t \in r_{bt}(\neg c(t)) \vee t[VT] \mathrm{I} vt = \phi \vee t[TT^+] \not\in UC\}$$
$$\Upsilon^{bt} \sum_{i=1..l} \mathrm{delete}^{\mathrm{row}}_{\mathrm{bt}}(r_{bt}, t_i, vt) \quad (2)$$

where, $restrict_{bt} =$
$$\{t | \exists s \in r_{bt}(c(t) \wedge t[VT] \mathrm{I} vt \not\in \phi \wedge t[TT^+] = UC)\}$$
$$t_i \in restrict_{bt},$$
$$l = \mathrm{count}(restrict_{bt})$$

식 (2)에서 삭제절의 대상 튜플 집합은 트랜잭션 시간 상한값 TT^+가 UC인 현재 유효한 튜플들 중 조건 c를 만족하면서 유효 시간 VT가 삭제절의 삭제 유효 시간 vt와 교집합이 존재하는 튜플들이다. 이 튜플 집합의 각 튜플에 대하여 튜플 단위의 삭제 연산 $delete^{row}_{bt}$을 사용하여 유효 시간의 겹침 부분의 삭제를 수행한다.

이원 시간 릴레이션 r_{bt}에 대해 조건 *c*를 만족하는 튜플들을 대상으로 유효 시간 vt의 속성 Ai의 값을 ai로 변경하는 수정절은 다음과 같은 일반적인 형식을 갖는다.

VALIDTIME *vt* UPDATE r_{bt} SET $A_i = a_i$ WHERE *c*

유효 시간 vt는 위의 VALIDTIME 구문과 같이 시간격 표현이나 시점 표현이 올 수 있으며, 만약 VALIDTIME 구문이 정의되지 않는다면 디폴트 유효 시간은 *PERIOD '[CT-FOREVER)'*를 의미한다. 또한 삭제절과 같은 이유로 갱신절의 대상은 TT^+가 UC인 튜플들로 한정된다. 위의 갱신절의 의미는 다음과 같이 정의된다.

$$update^{statement}_{bt}(r_{bt}.A_j, a_j, vt.c)$$
$$= {}^{\diamond} r'_{bt} \leftarrow \{t | t \in r_{bt}(\neg c(t)) \vee t[VT] \mathrm{I} vt = \phi \vee t[TT^+] \not\in UC\}$$
$$\Upsilon^{bt} \sum_{i=1..l} \mathrm{delete}^{\mathrm{row}}_{\mathrm{bt}}(r_{bt}, t_i, A_j, a_j, vt) \quad (3)$$

where, $restrict_{bt} = \{t | \exists s \in r_{bt}(c(t) \wedge t[VT] \mathrm{I} vt \not\in \phi \wedge t[TT^-] = CT \wedge t[TT^+] = YC)\}$
$t_i \in restrict_{bt}$,
$l = \mathrm{count}(restrict_{bt})$

식 (3)에서 갱신절의 대상 튜플 집합은 삭제절과 같이 트랜잭션 시간 상한값 TT^+가 UC인 현재 유효한 튜플들 중 조건 c를 만족하면서 유효 시간 VT가 갱신절의 갱신 유효 시간 vt와 교집합이 존재하는 튜플들이다. 이 튜플 집합의 각 튜플에 대하여 튜플 단위의 갱신 연산 $update^{row}_{bt}$을 사용하여 유효 시간의 겹침 부분을 삭제하고 Ai의 속성이 ai인 새로운 튜플을 삽입한다.

절 단위의 삽입, 삭제, 갱신절에 의해 사용되는 튜플 단위의 수정 연산 $insert^{row}_{bt}$, $delete^{row}_{bt}$, $update^{row}_{bt}$는 다음과 같이 정의된다.

$$insert^{row}_{bt}(r_{bt}.vt, (a_1, a_2, \ldots, a_n))$$
$$= {}^{\diamond} r'_{bt} \Upsilon^{vt} \{\langle a_1, a_2, \ldots, a_n \| vt, [CT-UC) \rangle\}] \quad (4a)$$
$$delete^{row}_{bt}(r_{bt}.x, vt)$$
$$= {}^{\diamond} r'_{bt} \{t | \exists x \in r_{bt}(x[SURROGATE] = t[SURROGATE]$$
$$\wedge t[A] = x[A] \wedge t[VT] = x[VT] \wedge t[TT^+] = CT\} \quad (4b)$$

$$\Upsilon'_{bt}\{t|\exists x\in r_{bt}(t[A]=x[A]\wedge t[VT]\prec(x[VT]-vt)$$
$$\phi\wedge t[TT^-]=cT\wedge t[TT^+]=UC)\}$$
$$update^{row}_{bt}(r_{bt}.x,A_i,a_i,vt)$$
$$=\diamond r'_{bt}\Upsilon^{bt}delete^{row}_{bt}(r_{bt},x,vt)\Upsilon^{bt}\{t|\exists_x\in r_{bt})t[A_{1..j-1}]$$
$$=x[A_{1..j-1}]\wedge t[A_i]=a_i\} \quad (4c)$$
$$\wedge t[A_{i+1..n}]=x[A_{i+1..n}]\wedge t[VT]=x[VT]\mathrm{I}vt\wedge t[TT^-]$$
$$=CT\wedge t[TT^+]-UC\}$$

식 (4a)의 이원 시간 릴레이션 r_{bt}에 대한 유효 시간 vt를 갖는 튜플 $(a_1,..,a_n)$의 삽입 연산 $insert^{row}_{bt}$는 유효 시간 vt와 트랜잭션 시간 $[CT\text{-}UC)$를 갖는 이원 시간 튜플을 r_{bt}에 대해 합집합 연산을 수행하는 것을 의미한다. 식(4b)의 $delete^{row}_{bt}$은 r_{bt}의 어떤 튜플 x로부터 유효 시간 vt를 삭제하기 위한 연산으로 현재 유효 튜플 x의 트랜잭션 시간 상한 TT^+에 현재 시간 CT를 갖게 하여 논리적 삭제를 수행하고 x의 유효 시간 $x[VT]$에서 삭제하고자하는 유효 시간 vt의 차분을 갖는 새로운 튜플들을 삽입한다. 이 때 삽입되는 튜플의 수는 차분에 따라 0,1 또는 2개가 존재할 수 있다. 식 (4c)에서 릴레이션 r_{bt}의 튜플 x의 속성 A_i의 값을 유효 시간 vt동안 a_i 값을 갖도록 하는 갱신 연산 $update^{row}_{bt}$ 는 우선 r_{bt}로부터 튜플 x의 유효 시간 vt를 삭제하는 연산 $delete^{row}_{bt}(r_{bt},x,vt)$를 수행하고 유효 시간 vt동안 유효한 새로운 튜플, 즉 $x[A_i]$가 a_i의 값을 갖는 튜플을 삽입한다는 것을 의미한다.

Ⅲ. 시간 능동 규칙 베이스

제 2장에서는 능동 규칙의 기반이 되는 시간 데이터베이스에 대하여 정의하고, 이원 시간 릴레이션에서의 삽입, 삭제, 갱신 연산의 실행 의미에 대하여 기술하였다. 이 장에서는 시간 데이터베이스에서의 능동 규칙 베이스의 의미에 대하여 기술한다.

1. 시간 능동 규칙 모델

시간 능동 규칙 언어는 시간 릴레이션에 대한 트리거를 지원하는 SQL3 표준 트리거 언어의 확장이다. 시간 데이터베이스 능동 규칙 언어의 구문은 다음에서 보이는 것과 같이 트리거의 유효시간과 시간 데이터베이스 사건, 조건, 조치, 전이 변수 선언, 조건-조치 처리 단위 부분으로 이루어진다.

VALIDTIME vt
CREATE TRIGGER *rulename*
E_t
$[TV_t]$
$[G_t]$
[WHEN C_T]
A_t

위의 시간 데이터베이스 능동 규칙 선언에서 vt는 트리거가 유효하게 활성화되어 있는 시간을 정의하며, E_t는 활성화 시간 동안 능동 규칙 모니터 해야 하는 시간 데이터베이스에서의 사건의 정의이다. C_t는 E_t에 정의된 사건이 발생 했을 때 데이터베이스의 현재 상태와 TV_t에 정의된 전이 변수에 의한 조건 판단에 대한 정의이며, A_t는 C_t의 조건을 만족할 때 수행해야 할 일련의 데이터 조작 연산으로 이루어진 조치들에 대한 정의이다. G_t는 E_t가 발생했을 때 C_t와 A_t를 수행하는 단위에 대한 정의이다. 위의 시간 능동 규칙 언어에 대한 실행 모델이 다음에서 정의된다.

[정의 3.1] 시간 능동 규칙의 실행 모델은 다음과 같이 정의된다.

$$[e_T\xrightarrow{\Delta_t,\delta_t}(g)(c_T\xrightarrow{\Delta_t,\delta_t}a_T)](VT)$$

시간 능동 규칙 TR은 유효한 활성화 시간을 정의하는 vt를 만족할 때 능동 규칙이 모니터 해야 하는 사건들의 집합 정의 E_t, 사건 E_t에 의해 트리거 되었을 때 처리해야하는 데이터베이스의 현재 상태와 규칙의 전이 릴레이션 Δ_t와 전이 변수 δ_t에 의한 조건 판단 정의 C_t, 조건 C_t를 만족할 때 수행해야할 일련의 데이터 조작 연산으로 이루어진 조치에 대한 정의 A_t의 네 개 구성요소에 의해 이루어진다. 이 논문에서는 각 규칙의 실행 의미를 표현하기 위해 다음과 같은 표현을 사용한다.

E_t: 사건 검출 정의어
C_t: 조건 평가 정의어
A_t: 조치 실행 정의어

위의 표현에서 보는 것과 같이 시간 능동 규칙의 선언은 트리거 유효성 검사부, 사건 검출부, 조건 평가부, 조치 실행부로 이루어진다.

[정의 3.2] 능동적 시간 데이터베이스 ATDB는 TDB가 시간 데이터베이스이고 TR이 DB를 위해 정의된 시간 능동 규칙의 집합일 때 다음과 같이 정의된다.

$$ATDB = <TDB, TR>$$

시간 데이터베이스의 시간 데이터는 사용자의 명시적인 수정 질의와 시간 능동 규칙의 처리 둘 다에 의해 수정될 수 있다. 또한 사용자의 명시적 수정 질의와 이 수정 질의에 의한 시간 능동 규칙 처리에 의해 수행되는 일련의 데이터 수정연산은 하나의 트랜잭션으로 간주된다.

[정의 3.3] 시간 능동 규칙 릴레이션 TRB의 스키마는 다음과 같이 정의된다.

$$TRB_{bt} = (rid,\ name,\ granularity,\ event,\ condition,\ action \| VT, TT)$$

시간 능동 규칙 데이터를 저장하기 위한 시간 능동 규칙 릴레이션 TRBbt는 규칙의 식별자 rid 속성, 규칙의 이름 name 속성, 조건과 조치부를 실행하기 위한 단위 granularity 속성, 규칙의 사건 정의 E_t를 저장하기 위한 event 속성, 조건부 C_t를 위한 condition 속성, 조치부 A_t를 위한 action 속성으로 구성되는 유효 시간과 트랜잭션 시간을 지원하는 이원 시간 릴레이션이다. 이 때 릴레이션 TRB_{bt}의 유효 시간 VT는 릴레이션의 인스턴스인 어떤 규칙이 데이터베이스에 대해서 유효하게 활성화되어 있는 시간을 의미하며, 트랜잭션 시간 TT는 그 규칙이 데이터베이스에 실제 삽입되고 삭제된 시간을 의미한다. 즉, 어떤 능동 규칙이 선언되었을 때 이 규칙은 다음과 같이 저장된다.

VALIDTIMEvt CREATE TRIGGER rulename...

$$= \diamond\, insert_{bt}^{statement}(TRB_{bt}, vt, (new\_rid(), rulename, (G_t, E_t, C_t, A_t))$$

시간 능동 규칙 릴레이션 TRB_{bt}과 시간 데이터베이스 내의 다른 릴레이션들과 두가지의 차이점이 존재한다. 첫 번째는 일반 릴레이션들이 INSERT, UPDATE, DELETE, SELECT 구문에 의해 접근가능한데 비하여 TRBbt는 시간 능동 규칙의 정의, 변경, 삭제 연산, 즉 CREATE TRIGGER, ALTER TRIGGER, DROP TRIGGER에 의해서만 수정할 수 있다는 것이다. 또 다른 차이점은 일반 이원 시간 릴레이션의 유효 시간 VT와 트랜잭션 TT가 상호 독립적인 값을 갖는데 비하여 TRBbt의 VT^-는 TT^- 보다 항상 크거나 같아야 한다.

$$TRB_{bt}[VT^-] \geq TRB_{bt}[TT^-]$$

유효 시간 하한값 VT^-가 시간 능동 규칙의 활성화 시작 시간, 즉 데이터베이스에 대한 삽입, 삭제, 갱신 연산의 모니터링을 시작하는 시간을 의미한다. 그런데 만약 VT^-가 TT^-보다 작은 값을 갖는다면 이것은 능동 규칙이 정의되기 이전 사건에 대한 회귀적(retroactive) 규칙 활성화를 의미하게 되는 것이다. 그러므로 항상 TRBbt의 VT^-는 TT^- 보다 같거나 작은 값을 갖도록 정의되어야 한다.

2. 전이 릴레이션 Δ_t와 전이 변수 δ_t

이원 시간 릴레이션에 대한 임의의 변경을 다루기 위해 Δ_t릴레이션의 개념을 도입한다. 다음은 Δ_t릴레이션의 정의이다.

[정의 3.4] 이원 시간 릴레이션 스키마 $R_{bt} = SURROGATE:(A_1, \ldots, A_n \| VT, TT)$라고 할 때 $\Delta_t R_{bt}$ 릴레이션의 스키마는 다음과 같이 구성된다.

$$\Delta_t R_{bt} = SURROGATE:(surrogate^{new}, A_1^{new}, A_1^{new}, VT^{new}\ TT^{new}, surrogate^{old}, A_1^{old}, \ldots, A_1^{old}, VT^{old}, TT^{old})$$

$\Delta_t R_{bt}$는 릴레이션 R_{bt}에 대한 변경 연산문의 실행에 의해 삽입, 삭제, 갱신되는 튜플 변경을 표현하기 위한 릴레이션이다. 이 릴레이션의 스키마는 새로 삽입되거나 삽입될 속성 값들을 참조하기 위해 사용되는 new 접미사를 갖는 속성들과, 삭제되거나 삭제될 속성 값들을 참조하기 위해 사용되는 old 접미사를 갖는 속성들로 구성된다. 즉, 위의 정의에서 $surrogate^{new}$는 삽입 튜플의 surrogate 값이고, $A_1^{new}, \ldots, A_n^{new}$는 그 튜플의 속성 값들이며, VT^{new}, TT^{new}는 삭제 튜플의 유효 시간과 트랜잭션 시간이다. $surrogate^{new}$의 값은 삽입 튜플이 실제 데이터베이스에 반영되기전 즉, BEFORE트

리거의 처리 시에는 NULL값을 가지며, 반영된 후 튜플의 실제 surrogate값을 갖게 된다. surrogate^{old}는 삭제 튜플의 surrogate 값이고, $A_1^{old},\ldots,A_n^{old}$는 그 튜플의 속성 값들이며, VT^{old}는 그 튜플에서 삭제되는 유효 시간 영역이고 TT^{old}는 삭제튜플의 트랜잭션 시간이다.

제2장의 식(1), 식(2)와 식(3)에서 기술 된 것과 같이 어떤 릴레이션 r_{bt}에 대한 삽입 연산문 $insert_{bt}^{statement}$, 삭제 연산문 $delete_{bt}^{statement}$, 갱신 연산문 $update_{bt}^{statement}$의 실제 데이터 변경은 각각의 튜플 수준 연산 $insert_{bt}^{statement}$, $delete_{bt}^{statement}$, $update_{bt}^{statement}$의 $\sum$ 또는 단일 연산에 의해 수행된다. 그러므로 릴레이션 $\Delta_t r_{bt}$에 대한 변경 연산문 S에 의한 릴레이션 $\Delta_t r_{bt}$는 튜플 수준 연산들의 $\sum$ 또는 단일 연산에 의해 구성된다. 이 때 각 튜플 수준 연산들의 $\Delta_t r_{bt}$에 대해 의미는 다음과 같다.

$$insert_{bt}^{row}(r_{bt}.vt,(a_1,a_2,\ldots,a_n))$$
$$=^{\diamond}\Delta_t r'_{bt}\leftarrow\Delta_t r_{bt}\mathrm{Y}\{t|t\in\Delta_t r_t(t[A_1^{new}]=a_1\wedge\ldots\wedge t[A_n^{new}]$$
$$=a_n\wedge t[VT^{new}]=vt\wedge t[TT^{-new}]=CT\wedge t[TT^{+new}]$$
$$=UC)\} \quad (5a)$$

$$delete_{bt}^{row}(r_{bt}.x,vt)$$
$$=^{\diamond}\Delta_t r'_{bt}\leftarrow\Delta_t r_{bt}\mathrm{Y}\{t|t\in\Delta_t r_t(t[surrogate^{old}]$$
$$=x[surrogate]\wedge t[A^{old}]=x[A]\wedge t[VT^{old}]$$
$$=x[VT]\mathrm{I}vt\wedge t[TT^{old}]=x[TT]\mathrm{I}[CT-UC))\} \quad (5b)$$

$$insert_{bt}^{row}(r_{bt}.x,A_i,a_i,vt)$$
$$=^{\diamond}\Delta_t r'_{bt}\leftarrow\Delta_t r_{bt}\mathrm{Y}\{t|t\in\Delta_t r_t(t[A_1^{new}]=x[A_1]\wedge t[A_i^{new}]$$
$$=a_i\wedge t[A_n^{new}]=x[A_n]\wedge t[VT^{new}]=[VT]\mathrm{I}vt$$
$$\wedge t[TT^{-new}]=CT\wedge t[TT^{+new}]=UC$$
$$\wedge t[surrogate^{old}]=x[surrogate]\wedge t[A^{old}]=x[A]$$
$$\wedge t[VT^{old}]=x[VT]\mathrm{I}vt\wedge t[TT^{old}]=x[TT]\mathrm{I}[CT-UC))\} \quad (5c)$$

식 (5a)에서 보이는 것과 같이 만약 S가 $insert_{bt}^{statement}$라면, $\Delta_t r_{bt}$는 S의 VALUES 구문에 의해 생성되는 튜플로 초기화 되며, new 접미사 속성들은 데이터베이스 내에 삽입되는 튜플들의 값을 포함하게 되며, old 접미사 속성들은 NULL이다. 식(5b)에서는 만약 S가 $delete_{bt}^{statement}$라면, $\Delta_t r_{bt}$는 S의 WHERE 구문과 VALIDTIME 구문을 만족하는 튜플들의 집합들에 의해 초기화 되며, old 접미사 속성들은 데이터베이스에서 삭제되는 튜플들의 유효 시간과 튜플들의 값을 포함하게 되고 new 접미사 속성들은 NULL이다. 식(5c)는 만약 S가 $update_{bt}^{statement}$라면, $\Delta_t r_{bt}$는 삭제연산과 같이 S의 WHERE 구문과 VALIDTIME 구문을 만족하는 튜플들의 집합들에 의해 초기화 되며, old 접미사 속성들은 데이터베이스에서 삭제되는 튜플들의 유효 시간과 튜플들의 값을 포함하며, new 접미사 속성들은 갱신에 의해 생성되는 튜플들의 값을 포함하게 된다. 다음의 예 3.1은 변경 연산에 의해 발생하는 Δ_t릴레이션을 보여준다.

[예 3.1] 다음의 릴레이션 EMP에 대하여 각 변경 연산문들에 의한 $\Delta_t EMP$을 보여라.

EMP

	name	salary	VT	TT
e1	Nam	100	[6-forever)	[2-4)
e2	Nam	100	[6-15)	[4-uc)
e3	Nam	100	[20-forever)	[4-uc)

위 릴레이션 EMP에 대하여 CT=10 일 때 다음의 삽입 연산문의 Δ_tEMP은 다음과 같다.

$$insert_{bt}^{statement}(EMP,[9-\text{for}\ ever),(\text{'}Rick\text{'},129))$$

$\Delta_t^i EMP$

	s^{new}	$name^{new}$	$salary^{new}$	VT^{new}	TT^{new}	s^{old}	$name^{old}$	$salary^{old}$	VT^{old}	TT^{old}
d1	e4	Rick	120	[9-forever)	[10-uc)					

위 삽입 연산문의 결과 릴레이션 EMP는 다음과 같다.

EMP

	name	salary	VT	TT
e1	Nam	100	[6-forever)	[2-4)
e2	Nam	100	[6-15)	[4-uc)
e3	Nam	100	[20-forever)	[4-uc)
e4	Rick	120	[9-forever)	[10-uc)

위 릴레이션 EMP에 대하여 CT=11 일 때 다음의 갱신 연산문이 수행되었다고 가정하면,

$$update_{bt}^{statement}(EMP,salary,150,[12-\text{for}\ ever],name$$
$$=\text{'}Rick\text{'}\wedge VT\mathrm{I}[12-\text{for}\ ever)\not\equiv\phi)$$

위 갱신 연산문을 수행에 의해 생성되는 △tEMP는 다음과 같다.

$\Delta_t^2 EMP$

	s^{new}	$name^{new}$	$salary^{new}$	VT^{new}	TT^{new}	s^{old}	$name^{old}$	$salary^{old}$	VT^{old}	TT^{old}
d2	e5	Rick	150	[12-forever)	[11-uc)	e4	Rick	120	[12-forever)	[11-uc)

위 갱신 연산문의 결과 릴레이션 EMP는 다음과 같다. 다음의 EMP릴레이션에서 e6는 제 2장에서 정의한 것과 같이 삭제 연산에 의해 발생된다.

EMP

	name	salary	VT	TT
e1	Nam	100	[6-forever)	[2-4)
e2	Nam	100	[6-15)	[4-uc)
e3	Nam	100	[20-forever)	[4-uc)
e4	Rick	120	[9-forever)	[10-11)
e5	Rick	150	[12-forever)	[11-uc)
e6	Rick	120	[9-12)	[11-uc)

조건 평가부와 조치부에서의 전이 릴레이션에 대한 참조를 지원하기 위해서 릴레이션 Rbt의 스키마의 속성 이름들로부터 new나 old 접미사를 재명명하기 위한 함수 newsuffix와 oldsuffix함수를 제공한다. R_{bt}의 스키마가 $surrogate(A_1, \ldots, A_n, VT, TT)$일 때, newsuffix($R_{bt}$)와 oldsuffix($R_{bt}$)는 다음과 같다.

$$newsuffix(R_{bt}) = (surrogate^{new}, A_1^{new}, \ldots, A_n^{new}, VT^{new}, TT^{new})$$

$$oldsuffix(R_{bt}) = (surrogate^{new}, A_1^{new}, \ldots, A_n^{new}, VT^{new}, TT^{new})$$

즉, newsuffix(R_{bt})는 R_{bt}의 스키마로부터 $\Delta_t R_{bt}$의 new접미사 속성들의 이름을 갖는 속성 리스트를 생성하는 함수이며, oldsuffix(R_{bt})는 old접미사 속성들의 이름을 갖는 속성 리스트를 생성하는 함수이다. 이 때, Δ_t릴레이션에 대한 프로젝션 연산 π_Δ는 $\Delta_t R_{bt}$로부터 특정 속성들을 프로젝트 하는 함수이며, $unsuffix_\Delta$는 대상 릴레이션의 속성이름들로부터 접미사를 제거하기 위한 함수이다. 위의 네 개 함수를 사용하여 전이 릴레이션 $\Delta_t^{new} R_{bt}$와 $\Delta_t^{old} R_{bt}$는 다음의 식(6a)와 식(6b)와 같이 정의될 수 있다.

$$\Delta_t^{new} R_{bt} \leftarrow unsuffix_\Delta(\pi_\Delta(\Delta_t R_{bt}, newsuffix(R_{bt}))) \quad (6a)$$

$$\Delta_t^{old} R_{bt} \leftarrow unsuffix_\Delta(\pi_\Delta(\Delta_t R_{bt}, oldsuffix(R_{bt}))) \quad (6a)$$

[예 3.2] 예 3.1의 $\Delta_t^2 EMP$의 $\Delta_t^{new} R_{bt}$는 식 (6b)에 의해 다음과 같다.

$\Delta_t^{new} EMP$

s	name	salary	VT	TT
e5	Rick	150	[12-forever)	[11-uc)

[정의 3.5] 릴레이션 R_{bt}에 대한 수정문 S에 의해 전이 릴레이션 $\Delta_t R_{bt}$이 생성되었을 때 δ_t는 $\Delta_t R_{bt}$내의 한 튜플이다. 이 때 삽입문과 갱신문에서 사용할 수 있는 표기 δ_t^{new}는 그 튜플의 new 접미사 부분 속성 값들을 의미하며 삭제문과 갱신문에서 사용할 수 있는 표기 δ_t^{old}는 old 접미사 부분 속성 값들을 의미한다.

위의 정의에서 δ_t^{new}. 속성이름 또는 δ_t^{old}.속성이름의 표기가 사용된다면 그 것은 각각 δ_t. 속성이름^{new}와 δ_t. 속성이름^{old}를 의미한다. 또한, 각 δ_t^{new}, δ_t^{old}.에 대하여 시간 능동 규칙의 조치부에서 속성 값에 대한 변경을 허용하며 각각 δ_t^{new}과.속성이름←속성값과 δ_t^{old}.속성이름←속성값의 표기를 사용한다.

이 절에서는 전이 릴레이션 Δ_t와 전이 변수 δ_t에 대하여 설명하였다. 아래의 예 3.3은 위의 정의가 어떻게 규칙에서 사용되는지를 보인다.

[예 3.3] 어떤 시간 능동 규칙이 EMP 릴레이션에 대한 salary속성에 대한 UPDATE 사건에 의해 트리거 되고 이 사건에 의해 변경되는 값들에 대해 조건부에서 새로운 값이 이전 값에 비해 10%이상 증가한 값인지를 평가하라.

$$C_i: \delta_t^{new}.salary > \delta_t^{old}.salary * 1.1$$

위 예는 정의 3.5를 이용하여 월급이 10%이상 증가한 값인지를 평가하고 있다.

3. 시간 데이터베이스에서의 사건

이미 앞에서 서술된 것과 같이 삽입 연산문 $insert_{bt}^{statement}$, 삭제 연산문 $delete_{bt}^{statement}$, 갱신 연산문 $update_{bt}^{statement}$들은 릴레이션 r_{bt}에 대한 변경을 수행하게 되며, 이 변경들은 능동 규칙의 사건을 트리거 하게 된

다. 이 때, 각 수정문들이 시간 능동 규칙을 트리거 시키는 사건은 다음과 같은 정규 형태를 갖도록 정의된다.

[정의 3.6] 시간 데이터베이스에서의 유효시간 사건은 다음과 같이 정의된다.

$$EVENT=(op, table, attr, vt, status)$$

위의 정의에서 op는 수정문들이 발생시키는 사건의 종류이며 {INSERT,UPDATE,DELETE}중 하나의 값을 갖는다. table은 수정문의 대상이 되는 릴레이션의 이름이며 attr은 그 릴레이션에서의 대상 속성이다. op가 INSERT, DELETE 일 경우 attr의 값은 항상 NULL이며, UPDATE 일 경우 값을 갖는다. vt는 수정문이 대상으로 하는 유효 시간 영역이다. status는 현재 수정문의 수행 상태를 의미하며 {BEFORE,AFTER} 중 하나의 값을 갖는다. 즉, 각 수정문에 대하여 실제 데이터베이스에 대한 반영 전(BEFORE)과 후(AFTER)에 각각 한번씩 사건이 발생하게 된다. EVENT에 대한 참조는 EVENT.속성의 형태로 수행된다. 예를 들면 EVENT.op는 현재 발생한 EVENT의 사건 종류를 참조한다.

[예 3.4] 어떤 시간 능동 규칙이 EMP 릴레이션의 데이터에 대해 유효시간 시간격 [20-30)사이의 튜플들에 대해 UPDATE 할 때 트리거되는 규칙이라면 이 사건은 다음과 같이 표현된다.

$$E_t: EVENT.op = UPDATE \wedge EVENT.table = 'emp' \wedge EVENT.attr = 'salary' \wedge EVENT.VTI[20-30) \not\equiv \phi$$

Ⅳ. 실행 의미 변환의 예

앞의 장들에서 지금까지 시간 능동 규칙 언어와 규칙의 의미, 사건과 조건-조치 사이의 실행 단위, 전이 릴레이션 Δ_t와 전이 변수 δ_t에 대하여 설명하였다. 이 장에서는 예를 통하여 시간 능동 규칙이 어떻게 이 논문에서 정의된 의미로 변환되는지 보인다.

[예 4.1] EMP 릴레이션에 대해 2000년 12월 25일부터 2001년 1월 2일까지 휴가기간 동안 데이터를 삭제하려는 시도에 대해 audit 릴레이션에 접근 기록을 저장하라는 능동 규칙은 다음과 같다.

```
VALIDTIME PERIOD '[2000/12/25-2001/01/02)'
CREATE TRIGGER vocation_modify_audit
BEFORE DELETE ON emp
FOR EACH STATEMENT
INSERT INTO audit VALUES( $USER, 'emp', CT);
```

위의 능동 규칙에서 WHEN절은 생략될 수 있다. 이 때 WHEN절은 항상 true를 의미한다. 또한, FOR EACH STATEMENT문에 의하여 사용자가 EMP릴레이션의 데이터를 수정하려는 접근에 대한 예 4.1의 능동 규칙은 각 수정문에 대하여 단 한번씩만 수행된다. 위의 시간 능동 규칙은 제 3장에의 정의들에서 서술된 것과 같이 사건 검출기에서 처리되는 E_t^1, 조건 평가기에서 처리되는 C_t^1, 조치 수행기에서 처리되는 A_t^1로 분리되며 다음과 같은 실행 의미를 갖는다.

$$E_t^1: EVENT.op = DELETE \wedge EVENT.status = BEFORE \wedge EVENT.table = 'emp'$$

$$C_t^1: true$$

$$A_t^1: insert(audit, (Suser, 'emp' CT));$$

또한, 이 규칙은 2000년 12월 25일부터 2001년 1월 2일까지 효력을 갖도록 능동 규칙 베이스 TRB_{bt}에 저장되기 의해 다음과 같은 삽입문을 수행한다.

$$insert_{bt}^{statement}(TRB_{bt}[2000/12/25-2001/01/02), (new\_rid(), STATEMENT, 'vocation\_modify\_audit', E_t^1, C_t^1, A_t^1))$$

[예 4.2] EMP 릴레이션에 대해 2000년부터 6월 1일부터 고용되는 사람의 월급이 현재부터 3년전의 고용자 평균 월급보다 2배 이상 크다면 그 고용자의 월급을 80%로 삭감하라, 이 능동 규칙이 2000년 3월 1일부터 활성화 되도록 하는 능동 규칙 선언은 다음과 같다.

```
VALIDTIME PERIOD '[2000/03/01 - forever)'
CREATE TRIGGER emp_salary_limit
```

```
BEFORE INSERT FOR VALIDTIME PERIOD
'[2000/06/01-forever)'
        ON emp
REFERENCING NEW AS new_emp
FOR EACH ROW
WHEN EXIST(VALIDTIME SELECT *
        FROM emp
        WHERE name=new_emp.name and new_
        emp.salary>emp.salary*2
                and VALIDTIME(emp) overlap
                CT-3year)
SET new_emp.salary = new_emp.salary*0.8
```

위의 시간 능동 규칙은 사건 검출기에서 처리되는 E_t^2, 조건 평가기에서 처리되는 C_t^2, 조치 수행기에서 처리되는 A_t^2로 분리되며 다음과 같은 의미를 갖는다.

$$E_t^2: EVENT.op = INSERT \wedge .EVENT.status = BEFORE \wedge EVENT.table = 'emp' \wedge EVENT.VT \mid [2000/06/01 - 2001/12 - 31) \not\in \phi$$

$$C_t^2 \{ < t.name, t.salary \mid VT, TT > \mid t \in EMP(t[name] = \delta_t^{new}.name \wedge \delta_t^{new}.salary > t[salary]*2 \wedge t[VT] \mid (CT - 3year) \not\in \phi) \not\in \phi$$

$$A_t^2: \delta_t^{new}.salary \leftarrow \delta_t^{old}.salary*1.1$$

위의 능동 규칙은 전이 변수 정의 NEW를 사용하였다. NEW는 제 3장의 정의 3.5에서 보이는 것과 같이 δ_t^{new}로 변환될 수 있다. 그러므로, 능동 규칙의 조건부 C_t^2는 δ_t^{new}를 갖는 해석식으로 변환되었다. 위의 능동 규칙 의미는 2000년 3월 1일부터 효력을 발생하도록 능동 규칙 베이스 TRB_{bt}에 저장되기 위해 다음과 같은 이원 시간 릴레이션 삽입문을 실행한다.

$$insert_{bt}^{statement}(TRB_{bt}, [2000/03/01 - forever), (new\_rid(), ROW, 'emp\_salary\_limit', E_t^2, C_t^2, A_t^2))$$

V. 구현 모델

시간 데이터베이스 규칙 시스템은 규칙 카탈로그, 규칙 관리자, 사건 검출기, 조건 평가기, 조치 수행기 등 다섯 가지 구성요소로 이루어진다. 이들 각 구성요소의 구체적 기능은 다음과 같다.

■ 규칙 카탈로그(TRB_{bt}) : 시간 능동 규칙을 저장하기 위한 이원 시간 릴레이션이다. 정의 3.3에서 설명된 것과 같이 규칙의 식별자 rid 속성, 규칙의 이름 name 속성, E-CA 수행 단위 granularity 속성, 규칙의 사건 정의 E_t를 저장하기 위한 event 속성, 조건부 C_t를 위한 condition 속성, 조치부 At를 위한 action 속성으로 으로 구성된다. 각 규칙 튜플의 유효시간은 규칙이 활성화 되어 있어야 되는 시간을 의미한다.

■ 규칙 관리자(rule manager) : 규칙 카탈로그 TRB_{bt}의 규칙들의 유효 시간을 체크하여 활성화 시간에 도달한 규칙들을 메인 메모리 상의 유효 규칙 캐쉬로 적재하며, 유효시간을 지난 규칙들을 캐쉬로부터 제거한다.

■ 사건 검출기(event detector) : 정의 3.6에 정의된 규칙들의 사건부 E_t에 대해 사건 트리(event tree)의 형태로 관리하며, 질의 수행기로부터 발생한 사건에 대한 메시지를 받고 사건 트리로부터 사건을 검출한다.

■ 조건 평가기(condition evaluator) : 데이터베이스의 현재 상태와 정의 3.4와 정의 3.5에 기술된 전이 릴레이션 및 전이 변수에 대한 질의를 통하여 조건을 평가한다. 조건 평가는 규칙 처리 중 가장 많은 시간을 소모하는 부분이다. 그러므로 점진적 조건 평가 알고리즘을 사용한다. 조건 평가를 위한 전이 릴레이션 및 전이 변수는 질의 수행기로부터 전달 받는다.

■ 조치 수행기(action executor) : 규칙의 조치부 C_t에 기술된 데이터베이스에 대한 검색 질의 및 데이터의 삽입, 삭제, 갱신 연산문 및 사용자 정의 함수를 호출한다. 조치의 수행 결과는 시간 데이터베이스의 데이터를 변경 시킬 수 있으며 이러한 변경은 또 다른 규칙을 트리거 시킬 수 있다. 이 때 규칙을 트리거 시킨 사건과 조치에 의해 발생하는 변경은 같은 트랜잭션으로 간주된다.

시간 데이터베이스의 능동 규칙 시스템에서 사건 검출기, 조건 평가기, 조치 수행기는 규칙 관리자의 유효 규칙 적재 및 제거 메시지에 의해 동적으로 처리해야

할 데이터를 추가하거나 삭제한다. 즉, 규칙 관리자가 유효 시간 시작에 도달한 규칙을 규칙 카탈로그 TRB_{bt}로부터 유효 규칙 캐쉬에 적재 할 때 사건 검출기에 적재되는 규칙의 사건부 E_t를 적재 메시지를 통해 보내며, 조건부 C_t는 조건 평가기에, 조치부 A_t는 조치 수행기에 보내진다. 또한, 유효 시간이 종료된 규칙을 캐쉬에서 제거 할 때 제거되는 규칙에 대한 메시지가 사건 검출기, 조건 평가기, 조치 수행기에 보내지며 메시지를 전달받은 각 시스템은 그 규칙에 대한 정보를 제거한다.

시간 데이터베이스의 데이터 수정문을 수행하는 질의 수행기는 알고리즘 5.1의 수행 알고리즘과 같이 각 삽입, 삭제, 갱신 연산문에 대한 실제 변경을 수행하며 각 수행단계에서 능동 규칙 시스템의 사건 검출기, 조건 평가기, 조치 수행기를 호출한다. 다음과 같이 네 개의 주요 함수로 구성된다.

○ compute_delta(query^{statement},type,target$_{bt}$,attr,vt, Δ_t)

compute_delta 함수는 수정문을 해석하고 전이 릴레이션 Δ_t를 계산하는 함수이다. 삭제, 갱신 수정문을 입력으로 받아서 질의 수정문의 타입 type, 대상이 되는 릴레이션 targetbt, 대상이 되는 속성 attr, 수정문이 대상으로 하는 유효시간 영역 vt, 영향을 받는 튜플 들의 전이 릴레이션 Δ_t를 반환한다. attr은 update에서 속성이 명시된 경우에 값을 가지며 그렇지 않을 경우 NULL값을 갖는다.

○ event_detect(type, targetbt, attr, vt, mode)

event_detect 함수는 사건 검출을 위해 규칙 시스템의 사건 검출기에 사건 발생 메시지를 보내는 함수이다. 메시지는 수정문의 타입 type, 대상이 되는 릴레이션 target$_{bt}$, 대상 속성 attr, 사건의 대상이 되는 유효시간 영역 vt, BEFORE와 AFTER 사건 처리를 위해 발생한 사건의 전인지 후인지를 표현하는 mode로 구성된다. 사건 검출기는 처리 결과로 위의 사건을 만족하는 규칙들의 리스트를 결과 값으로 반환한다.

○ condition_evaluation(tr, Δ_t, δ_t)

condition_evaluation 함수는 규칙 시스템의 조건 평가기에 조건 발생 메시지를 보내는 함수이다. 메시지는 조건 평가에 평가해야하는 규칙 tr, 사건에 의해 생성된 전이 릴레이션 Δ_t, 현재 처리중인 변수 δ_t로 구성된다. 만약 규칙이 STATEMENT 단위의 실행으로 설정되었다면 δ_t는 NULL이다. 조건 평가의 반환 값은 조건이 만족할 때 true, 만족하지 않을 때 false를 반환한다.

```
executevt (q_bt^statemen)
{
  compute_delta( q_bt^statemen, type, target_bt, attr, vt, Δ_t);
   TriggeredRules≤ event_dectect
  for each  tr∈ TriggeredRules
    if  tr[ granularity] = STATEMENT
      if(condition_evaluation( tr, Δ_t, NULL) = true)
        action_activation( tr, Δ_t, NULL);
    else/*if tr[granularity]=ROW* /
        for  eachδ_t∈Δ
          if(condition_evaluation ( tr, Δ_t, δ_t) = true)
            action_activation ( tr, Δ_t, δ_t);
  apply_delta( type, target_bt, attr, vt, AFTER);
   TriggeredRules≤ event_dectect
  for each  tr∈ TriggeredRules
    if  tr[ granularity] = STATEMENT
      if(condition_evaluation( tr, Δ_t, NULL) = true)
        action_activation( tr, Δ_t, NULL);
    else/*if tr[granularity]=ROW* /
        for each  δ_t∈Δ
          if(condition_evaluation ( tr, Δ_t, δ_t) = true)
            action_activation ( tr, Δ_t, δ_t);
}
```

알고리즘 5.1 수정 질의 수행기 알고리즘

Algorithm 5.1. Algorithm of modification executor

○ action_activation(tr, Δ_t, δ_t)

action_activation 함수는 규칙 시스템의 조치 수행기에 조치 수행 메시지를 보내는 함수이다. 메시지는 조치 수행에 사용되는 규칙 tr, 사건에 의해 생성된 전이 릴레이션 Δ_t, 현재 처리중인 전이 변수 δ_t로 구성된다. 조치문에서 SET new.attr=value 형태의 구문을 사용하여 δ_t의 값을 변경할 수 있다. 조치의 수행은 다른 규칙들을 트리거 할 수 있다. 단 BEFORE문에 의한 조치

문은 데이터의 회귀적 변경을 초래할 수 있으므로 조치문에 수정문이 허용되지 않는다.

알고리즘 5.1의 질의 수행기 알고리즘에서 만약 트리거 된 규칙이 E-CA가 명령문 단위로 수행되는 STATEMENT 모드라면 규칙은 트리거 된 사건에 대해 단 한번 수행된다. 그리고 튜플 단위의 수행 모드인 ROW 라면 각 전이 릴레이션에 있는 각 튜플당 한번씩 조건을 평가하고 조건을 만족할 경우 조치를 수행한다. 또한, event_detect 함수에 의해 트리거 되는 규칙이 하나 이상일 경우 규칙들은 규칙들이 TRBbt에 삽입된 트랜잭션 시간 순으로 수행된다. 이 것은 먼저 정의 된 규칙의 수행 결과에 대해 항상 후에 정의된 규칙의 조치가 수행되는 것을 보장한다.

Ⅵ. 기존 연구와의 비교

이 논문의 능동 시간 데이터베이스(active temporal database)는 시간 데이터베이스 상에서 능동 규칙을 지원하는 데이터베이스 시스템이다. 능동 시간 데이터베이스는 능동 데이터베이스 시스템에 시간 개념을 지원하려는 시간 능동 데이터베이스(temporal active database)에 대한 연구들과는 기반 데이터 모델 상에서 차이점을 갖는다. 즉, 시간 능동 데이터베이스는 시간 데이터 모델을 고려하지 않는 일반 데이터베이스 상에서 능동 규칙에 시간 사건을 포함하는 복합 사건에 중점을 두는데 비하여[11~13], 능동 시간 데이터베이스는 유효 시간과 트랜잭션 시간을 지원하는 시간 데이터베이스 상의 데이터를 바탕으로 능동 규칙을 통해 데이터베이스의 현 상황을 판단하여 적절한 조치를 취할 수 있다[3].

Sistla의 연구들[7,8]은 능동 시간 데이터베이스와의 시간 능동 데이터베이스의 중간점에 존재한다. 그의 연구들은 일반적인 능동 관계형 데이터베이스 상에서 능동 규칙을 이용하여 PTL(Past Temporal Logic) 및 FTL (Future Temporal Logic)을 통해 C-A 패턴의 능동규칙 언어를 제시하면서, 조건의 표현에 유효 시간과 트랜잭션 시간 개념이 가능하도록 했으며 규칙에서 시간과 연계된 조치를 표현할 수 있도록 하였다. 그러나, 시간 데이터베이스를 기반으로 하지 않고 관계형 데이터베이스 상에서 보조 릴레이션을 통해 이미 선언된 PTL과 FTL을 통해 시간 개념을 지원하므로 능동 규칙 선언 이후의 시간 데이터에 변경에 대해서만 조건 평가를 수행할 수 있다는 점에서 이 논문의 능동 시간 데이터베이스와 차이점을 갖는다.

Gal, Etzion 및 Segev는 [3]에서 시간 함수들, 특히 선행 갱신과 소급 갱신의 처리와 관련된 PARDES의 확장에서 능동적 시간 데이터베이스(active temporal databases) 모델을 제안하고 있다. 특히 데이터의 소급 갱신과 선행 갱신의 처리지원을 위한 data-driven 규칙의 확장을 고려하여 시간 데이터 모델의 확장을 추구했다. 이 후 [9]에서 PARDES 능동 데이터베이스 언어의 확장에 의한 시간 능동 데이터베이스 언어 및 실행 모델 TALE(Temporal Active Language and Execution Model)을 제시하였다. Gal 등은 TALE을 통해 데이터베이스의 시간지원 기능과 능동기능 간의 관계에 대하여 명확하고 정밀한 의미를 기술하며, 시간 데이터 모델을 대상으로 한 능동 규칙 연산에 대해 시간 능동 모델이 미치는 시간차원의 영향 유형을 정의하고 있다. 이 연구들은 능동 규칙의 수행시에 시간 데이터베이스상의 데이터들 간의 갱신에 의한 영향에 중점을 두었다. 현재의 능동 규칙 시스템이 대부분 지원하고 지원하고 있는 사건을 발생시킨 튜플이나 테이블 상태를 조건, 조치 등에서 참조할 수 있는 전이 릴레이션이나 전이 변수와 같은 기능을 제공하지 않는 초보적인 수준의 규칙만을 제공한다. 그러나, 이 논문의 모델에서는 전이 릴레이션과 전이 변수, 규칙의 수행 단위, 조치에서의 전이 변수 수정과 같은 기능에 중점을 둔 좀 더 진보적인 능동 시간 규칙을 지원하는 것이 큰 차이이다. 아울러 의미 모델과 실행 알고리즘을 구체적으로 제시하였다.

Ⅶ. 결 론

이 논문은 SQL/Temporal 데이터 모델을 기반으로 유효 시간 릴레이션, 트랜잭션 시간 릴레이션, 이원 시간 릴레이션에 대하여 전이 릴레이션 및 전이 변수, 조건-조치의 튜플 및 문 단위 수행을 지원하도록 하였고, 능동 규칙의 활성화 유효 시간과 유효 시간 사건 등의 새로운 개념을 갖는 능동 규칙 모델과 실행 모델을 제안하였다. 아울러 제안된 모델을 증명하기 위하여 예를 제시하고 제 5장에서 구현 알고리즘을 제시하였다.

제안된 능동규칙 언어는 표현 형태에 있어서 현재 ISO의 SQL92의 확장 표준화 안으로 작성된 SQL3의

SQL/Foundations에 기술된 트리거 표현 문법 형태를 따르면서 SQL/Temporal 모델을 확장하였다. 이것은 제시된 능동 규칙 언어가 시간 속성의 표현이나 능동 규칙의 표현에 있어서 표준인 SQL3를 따르도록 함으로써 사용자의 부담을 최소화하고 여타 응용 분야의 능동 규칙 표현이나 시간 속성을 갖는 데이터 관리 요구에도 일반적으로 적용될 수 있도록 하기 위해서이다.

한편 이 논문에서는 시간 데이터베이스 상에서의 규칙의 충돌에 대한 내용과 시간 무결성 제한자의 지원에 대한 문제점에 대한 해결책은 차후 연구로 남겨두었다.

참 고 문 헌

[1] R. Snodgrass and I. Ahn, "Temporal Databases," *IEEE Computer*, vol.19, no.9, pp.35~42, September 1986.

[2] J. Widom and S. Ceri, ed. *Active Database System, Triggers and Rules For Advanced Database Processing*, Morgan Kaufmann Publishers Inc., p.2, 1996.

[3] O. Etzion, A. Gal, and A. Segev, "Data Driven and temporal Rules in PARDES," *in Proc. of Conf. on Rules in Database Systems*, pp.92~108, Edinburgh, Scotland September 1993.

[4] 박정석, 신예호, 남광우, 류근호, "시간지원 능동 규칙의 점진적 조건 평가," 정보과학회 논문지 (B), 제 26권 제 4호, 462~472쪽, 1999년 4월.

[5] R. Chandra and A. Segev, "Managing Temporal Financial Data in an Extensible Database," *in Proc. of VLDB Conf.*, pp.302~313, Dublin, Ireland, August 1993.

[6] J. Chomicki and D. Toman, "Implementing Temporal Integrity Constraints using an Active DBMS," *IEEE Trans. on Knowledge and Data Engineering*, vol.7, no.4, pp.566~581, April 1995.

[7] Sistla, P. and O. Wolfson, "Temporal Conditions and Integrity Constraints in Active Database Systems," *in Proc. of ACM SIGMOD Conf.*, pp.269~280, San Jose, USA, May 1995.

[8] Sistla, P. and O. Wolfson, "Temporal Triggers in Active Databases," *IEEE Trans. on Knowledge and Data Engineering*, vol.7, no.3, pp. 471~486, June 1995,

[9] Gal, Avigdor, Opher Etzion, and Arie Segev, "TALE: A Temporal Active Language and Execution Model," *Lecture Note in Computer Science*, vol.1080, pp.61~81, May 1996.

[10] R. T. Snodgrass, M. Bohlen, C. Jensen, and A. Steiner, "Transitioning Temporal Support in TSQL2 to SQL3," *Lecture Note in Computer Science*, vol.1399, pp.150~194, June 1997.

[11] D. Klaus and S. Gatziu, "Events in an Active Object-Oriented Database System," *in Proc. of the Int. Workshop on Rules in Database Systems*, pp.23~39, Edinburgh, Scotland, September 1993.

[12] N. Gehani, H. Jagadish, and I. Mumick, "Event Specification in an Active Object-Oriented Database." *in Proc. of ACM SIGMOD Conf.*, pp.81~90, San Diego, USA, June 1992.

[13] E. N. Hanson and L. Noronha, "Timer-Driven Database Triggers and Alerters: Semantics and a Challenge," *SIGMOD Record*, vol.28, no.4, pp.11~16, December 1999.

저 자 소 개

南 光 祐(正會員)

1995년 충북대학교 전산학과 졸업(이학사). 1997년 충북대학교 대학원 전산학과 석사(이학석사). 1997년-현재 충북대학교 대학원 전산학과 박사과정. 관심분야 : 공간 데이터베이스, 시간 데이터베이스, 시공간 데이터베이스, Temporal GIS 등

柳 根 浩(正會員)

1976년 숭실대학교 전산학과 졸업(이학사). 1980년 연세대학교 산업대학원 전산전공(공학석사). 1988년 연세대학교 대학원 전산전공(공학박사). 1976년-1986년 육군 군수 지원사 전산실(ROTC장교), 한국전자통신연구소(연구원), 한국 방송대학교 전산학과(조교수) 근무. 1989년 - 1991년 University of Arizona, Research Staff(TempIS 연구원, Temporal DB). 1986년 - 현재 충북대학교 컴퓨터과학과 교수. 관심분야는 시간 데이터베이스, 시공간 데이터베이스, 지식기반 정보검색, Temporal GIS, 객체 및 지식베이스 시스템

李 鍾 勳(正會員)

1981년 연세대학교 토목공학과(공학사). 1981년 연세대학교 대학원 토목공학과(공학석사). 1987년 코넬대학교 원격탐사전공(공학석사). 1990년 코넬대학교 원격탐사전공(공학박사). 1990년 한국과학기술연구원 시스템공학센터(KIST/SERI) GIS그룹. 1992년 -현재 한국전자통신연구원 컴퓨터·소프트웨어연구소 GIS연구팀 팀장

관심분야는 원격탐사(RS), 공간정보시스템인 지리정보시스템(GIS), 위성측위시스템(GNSS), 공간영상정보시스템(SIIS), 지능형교통체계(ITS)