조건 술어 분석을 이용한 능동규칙의 조건부 처리 시스템

이 기 욱† · 김 태 식††

요 약

능동 데이터베이스 시스템은 특정한 상태를 탐지하는 능동규칙을 도입한다. 능동규칙의 조건부 평가는 사건이 발생할 때마다 수행되기 때문에 조건부를 처리하는 방법에 따라 시스템의 성능에 중요한 영향을 미친다. 본 논문에서는 차이트리 구조, 분류트리, 그리고 집계함수 테이블을 생성하는 전처리 기능을 갖는 조건부 처리 시스템을 제안한다. 전처리는 능동규칙을 미리 파악할 수 있는 능동 데이터베이스의 특징 때문에 도입될 수 있다. 본 논문에서는 선택연산, 조인연산, 그리고 집계함수를 효율적으로 처리할 수 있는 차이트리를 제안하여 조건부의 처리 성능을 높인다. 그리고 조인연산을 효과적으로 처리하는 분류트리와 높은 처리비용을 요구하는 집계함수를 처리하는 집계함수 테이블을 제안한다. 본 논문의 조건부 처리 시스템은 전처리 기능에서 만들어진 조건부 처리 구조 때문에 조건 비교의 횟수를 감소시켜 능동규칙에서 조건부 처리의 성능 향상을 기대할 수 있다.

A Condition Processing System of Active Rules Using Analyzing Condition Predicates

Ki-Wook Lee† · Tae-Sik Kim††

ABSTRACT

The active database system introduces the active rules detecting specified state. As the condition evaluation of the active rules is performed every time an event occurs, the performance of the system has a great influence, depending on the conditions processing method. In this paper, we propose the conditions processing system with the preprocessor which determines the delta tree structure, constructs the classification tree, and generates the aggregate function table. Due to the characteristics of the active database through which the active rules can be comprehended beforehand, the preprocessor can be introduced. In this paper, the delta tree which can effectively process the join, selection operations, and the aggregate function is suggested, and it can enhance the condition evaluation performance. And we propose the classification tree which effectively processes the join operation and the aggregate function table processing the aggregate function which demands high cost. In this paper, the conditions processing system can be expected to enhance the performance of conditions processing in the active rules as the number of conditions comparison decreases because of the structure which is made in the preprocessor.

키워드 : 능동 데이터베이스(Active database), 차이트리(Delta tree), 분류트리(Classification tree), 집계함수 테이블(Aggregate function table)

1. 서 론

수동 데이터베이스에서 무결성 제약 조건과 트리거를 해결하는데는 두 가지 방법으로 접근할 수 있다. 첫 번째 방법은 데이터베이스에 대한 각각의 응용에 제약 조건 또는 트리거 조건을 조사하는 코드를 추가하는 방법이다. 이 방법은 여러 단점을 가진다. 자료의 추가, 수정, 그리고 제거 시 제약조건과 트리거는 모든 응용에서 관계된 코드를 찾고 변경 작업을 요구한다. 정확한 제약조건 검사나 트리거 행위는 모든 응용이 정확하게 구현될 때 보장되어 진다. 또한 모든 변경 후에는 추가적인 응용과 데이터베이스 연결이 요구되어 진다. 두 번째 방법은 각 제약조건과 트리거를 조사하는 데이터베이스에 정기적으로 폴링(polling)하는 특수한 응용프로세서를 추가하는 것이다. 첫 번째 방법에서 발생되는 문제점들을 해결하지만 폴링의 발생 빈도가 높으면 발생되는 데이터베이스 처리 성능에 부가비용이 발생한다. 또한 폴링 빈도를 낮추면 제약조건 또는 트리거 조건의 탐지가 비효율적이다. 따라서 다른 방법으로 해결하는 것보다 데이터베이스 시스템 자체에 능동행위를 직접 지원하는 방법을 사용하는 것이 유용하다[1].

능동 데이터베이스 시스템은 수동 데이터베이스 시스템에서 인코드되어야 하는 함수를 효율적으로 수행할 수 있

† 정 회 원 : 동명대학 정보통신계열 교수
†† 정 회 원 : 계명대학교 컴퓨터공학전공 교수
논문접수 : 2001년 8월 20일, 심사완료 : 2001년 10월 25일

으며, 수동 데이터베이스 시스템의 영역을 초과하는 응용들을 제시하여 처리한다. 그리고 수동 데이터베이스 시스템에 특수 목적 서브시스템(special-purpose subsystem)을 요구하는 작업을 수행할 수 있다.

능동 데이터베이스 시스템은 능동규칙 처리 기반을 도입함으로써 데이터베이스 상태에 따라 자동적으로 일정한 동작이 수행된다. 능동규칙은 사건-조건-동작(ECA) 패러다임에 기초로 한 구조로서 사건이 발생할 때, 조건이 만족하면 동작을 수행한다. 능동규칙은 데이터베이스 시스템에 강력한 능동성을 제공할 수 있기 때문에 능동 데이터베이스 시스템의 핵심이 되는 기술이라 할 수 있다. 능동규칙의 조건부는 데이터베이스 연산과 집계함수들로 구성되어 있고 특히 조인연산과 집계함수는 처리비용이 큰 연산들이다. 대부분의 조건부에는 이들 연산의 조합으로 구성되어 있기 때문에 조건부를 구성하는 조건 술어의 형태가 매우 복잡해질 수 있다. 이런 상황에서 조건 술어의 평가가 효율적으로 이루어지지 않으면 시스템 전체 성능에 막대한 영향을 주게 된다[2-4].

본 논문은 능동 데이터베이스 시스템의 능동규칙 조건부에서 발생되는 기본 데이터베이스 연산과 처리비용이 큰 집계함수를 효율적으로 처리하기 위하여 능동규칙의 조건부를 미리 파악할 수 있는 점을 효과적으로 이용한 전 처리 기법과 점진적 평가를 기초로 한 차이트리, 분류트리, 그리고 집계함수 테이블을 이용하여 조건부 평가를 효율적으로 처리하는 조건부 처리 시스템을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 능동규칙의 조건부 평가의 관련 연구를 알아보고, 3장에서는 조건부 평가를 위한 조건부 처리 시스템을 제안하고, 4장에서는 성능 평가를 위한 환경과 분석 결과를 제시하고, 5장에서는 결론과 향후 연구 과제를 기술한다.

2. 관련 연구

능동 데이터베이스 시스템에서 효과적으로 수행될 수 있는 함수는 일반적인 무결성 제약조건과 트리거들이다. 무결성 제약조건은 자료의 정밀성, 정확성, 그리고 유효성을 유지하도록 명시된 술어이다. 기존의 수동 데이터베이스에서는 무결성 제약조건 검사를 수정직후나 트랜잭션의 종료시점 그리고 트랜잭션 복귀 시 제약조건 위반에 대한 반응으로 제한한다. 능동 데이터베이스 시스템은 일반적인 데이터베이스에 대한 무결성 제약조건의 명세(specification)와 감시(moritors)를 지원하며, 조건검사에 대한 시간의 융통성을 허용하며, 트랜잭션 복귀 없이 제약조건 위반 수정에 대한 보상행위의 수행을 제공한다[5].

능동규칙의 조건부 평가를 높이기 위하여 크게 점진적 평가(incremental evaluation) [6-8]와 판별네트워크(discrimination network)[9, 10]이 사용되어 진다. 점진적 평가 방법에서는 차이 릴레이션(delta relation)을 도입하여 데이터베이스의 변경된 부분만 처리함으로써 전체 데이터베이스에 대한 처리비용을 줄이는 방법이다. 판별네트워크는 조건을 평가하기 위해 조건 술어를 만족하는 튜플들을 네트워크의 형태로 저장하여 해당 조건을 만족하는 자료를 찾아서 처리한다.

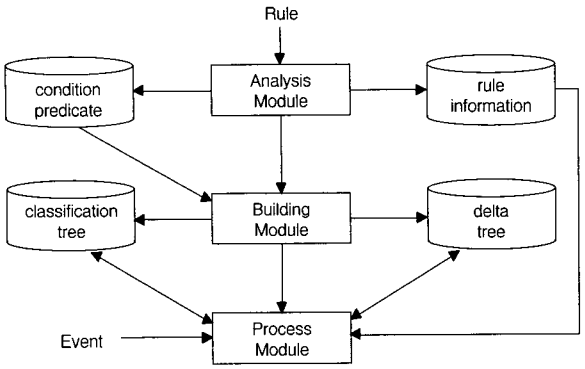
점진적 평가에서는 일부 집계함수를 사용한 단일 조건 또는 중첩된 연산들로 구성된 조건부를 처리하고 있다. 그러나 집계함수가 포함된 조인연산에서 조건부 평가를 효율적으로 처리하는 구조가 미비한 상황이기 때문에 연산 처리비용이 시스템에 부담을 준다[8]. 판별네트워크를 이용한 조건부 평가에서 사용되는 A-TREAT[9]는 인공지능 분야의 생성시스템에서 사용되던 Rete[11]와 TREAT[12] 알고리즘을 응용하여 만들어졌다. A-TREAT는 기본 릴레이션에서 해당 튜플을 가지고 오는 처리시간이 조건부를 효율적으로 처리하는데 장애가 된다. 또한 A-TREAT의 구조는 조인연산을 처리하기 위해 α-메모리를 사용하여 자료를 저장하는데, 조인연산에 참조가 되는 튜플들 뿐만 아니라 튜플 전체가 저장된다. 그래서 A-TREAT는 자료의 중복으로 인하여 메모리 낭비가 발생하고 중간 연산 결과를 보유하지 못하기 때문에 다른 조건 술어의 부분적인 결과에 대해 조건 평가를 할 수 없다. 또한 판별네트워크에서 사용할 수 있는 조건은 인터벌, 비교연산, 그리고 일반함수로 국한하고 있다. 일반함수는 함수의 반환 값을 참 또는 거짓 값으로만 한정한다. 예를 들면 isodd( ) 함수는 인자의 값이 홀수 또는 짝수인지를 판별하여 반환하는 일반함수이다. 따라서 판별네트워크는 집계함수를 처리하지 못하는 문제점을 가지고 있어 능동규칙 조건부가 중첩되는 경우, 집계함수가 자주 사용되므로 효율적인 집계함수 처리 방법이 제공되어야 한다.

3. 조건부 평가

능동 데이터베이스는 능동규칙을 미리 정의하여 데이터베이스 연산이 수행되면 규칙의 조건부에서 참·거짓을 판별하여 이 값에 따라 동작이 수행된다. 이런 특징은 데이터베이스에 특정한 사건이 발생하기 전에 능동규칙을 미리 파악할 수 있음을 의미하기 때문에 전처리 작업이 가능하다. 전 처리기 기능은 미리 정의된 능동규칙을 바탕으로 조건 술어만을 수집하여 조건 술어 평가에 적합한 형태로 구조화할 수 있다.

3.1 시스템 모델

전 처리 기능을 갖는 조건부 처리 시스템은 (그림 1)과 같다.



(그림 1) 조건부 처리 시스템 모듈 구조

조건부 처리 시스템은 분석모듈(analysis module), 구축모듈(building module), 그리고 처리모듈(process module)로 구성된다. 분석모듈은 능동 데이터베이스 시스템에서 정의된 능동규칙을 분리하여 규칙 정보 파일(rule information file)과 조건 술어 파일(condition predicate file)을 생성한다. 규칙 정보 파일은 규칙이름, 규칙의 사건 종류, 그리고 규칙의 조건부에 존재하는 조건 술어의 개수를 보유하여 능동규칙의 사건과 관련된 규칙의 조건부를 만족하는 여부를 판단할 때 사용된다.

다른 조건부 평가 방법에서는 사건에 관련된 능동규칙을 순차적으로 조사하여 조건부가 만족되는 규칙을 찾는데 반해 조건부 처리 시스템에서는 규칙 정보 파일을 이용하여 조건부를 만족하는 규칙들을 동시에 찾을 수 있다. 조건 술어 파일은 분석모듈을 통해 규칙이름과 규칙의 조건부에 있는 조건 술어들을 제공받는다. 조건 술어 파일에 있는 규칙이름과 조건 술어들은 구축모듈에 사용된다.

구축모듈은 조건 술어 파일의 정보를 제공받아 차이트리 구조, 분류트리, 그리고 집계함수 테이블을 생성한다. 구축모듈은 조건 술어 파일의 조건 술어 중에서 관계연산자를 포함하는 술어들을 계층적인 노드로 차이트리를 구축한다. 구축된 차이트리는 삽입되는 튜플들을 저장하는 구조로 되어 있다. 그리고 조인연산을 포함하는 조건 술어의 정보를 이용하여 연산에 관계된 애트리뷰트들을 분류트리로 구축한다. 또한 집계함수와 관계된 조건 술어들을 이용하여 집계함수 테이블을 만든다. 처리모듈은 실제 데이터베이스에 사건이 발생될 때 차이트리, 분류트리, 집계함수 테이블, 그리고 규칙 정보 파일을 이용하여 규칙의 사건과 관련된 튜플들을 차이트리에 저장하거나 각 사건에 따른 데이터베이스 변경 작업을 처리한다. 또한 능동규칙의 목록을 생성하여 규칙의 조건부를 평가한다.

3.2 조건 술어 처리

조건부 처리 시스템이 사용하는 능동규칙의 형태는 다음과 같다.

RuleName : On *Event*
If *Condition*
Then *Action*

능동규칙의 이름은 유일해야 하며, 사건의 종류에는 삽입, 삭제, 그리고 수정이 있다. 조건부에는 선택연산, 조인연산, 그리고 집계함수가 있으며 대부분의 조건부 형태는 이들 연산들이 결합되어 있다. 동작부는 임의의 SQL 문장이거나 응용프로그램 실행문이 될 수 있다. 능동규칙의 의미는 사건이 발생되고 조건이 만족되는 경우에 동작을 실행하는 것이다.

본 논문에서 사용되는 조건부 술어 구조는 다음과 같다.

⟨*Cond*⟩ :: = ⟨*Value Exp.*⟩ ⟨*Op*⟩ ⟨*Value Exp.*⟩
⟨*Value Exp.*⟩ : = ⟨*Single Column Ref.*⟩
| ⟨*Aggregate Function*⟩⟨*Multiple Column Ref.*⟩
⟨*Op*⟩ :: = > | < | ≤ | ≥ | = | ≠
⟨*Aggregate Function*⟩ :: = *sum* | *count* | *avg* | min | max

⟨*Single Column Ref.*⟩와 ⟨*Multiple Column Ref.*⟩는 각각 단일 칼럼과 다중 칼럼을 가리키는 ⟨*Query Exp.*⟩이며 'EXIT'나 'IN'과 같은 술어는 조인과 집계함수로 변형하여 사용한다. 위에서 제시하는 술어 구조로 나타낼 수 있는 능동규칙의 조건은 한개의 튜플에 한개의 컬럼만을 가리키는 표현뿐만 아니라 두개이상의 릴레이션이 적용되는 복잡한 표현이 될 수 있다.

전 처리 기능은 능동규칙의 조건부에서 사용되는 조건 술어를 분석한 후 각 연산을 처리할 수 있는 차이트리, 분류트리, 그리고 집계함수 테이블을 구축한다. 그리고 이런 구조들을 구축하기 위해서는 데이터베이스 스키마와 능동규칙이 필요하다. 데이터베이스 스키마와 능동규칙의 예는 (그림 2)와 같다. 데이터베이스의 스키마에서 Emp 릴레이션은 기본 릴레이션으로 지정되며 Dept 릴레이션과 Job 릴레이션은 조인연산에 사용된다. 6개의 능동규칙을 제시하였으며 대부분의 규칙의 사건은 삽입연산으로 이루어져 있다.

```
Emp(eno, name, age, salary, dno, jno)
Dept(dno, name, loc)
Job(jno, title)

R1 : On Insert Emp
     If Emp.eno > 500 AND
        Emp.eno < 2500 AND
        Emp.salary >= 11000
     Then Action1
R2 : On Update Emp
     If Emp.salary >= 10000 AND
```

```
        Emp.salary <= 25000
    Then Action2
R3 : On Insert Emp
    If (Emp.age > 26 AND
        Emp.dno = Dept.dno AND
        Dept.loc = 'PUSAN')
    Then Action3
R4 : On Insert Emp
    If SUM(Emp.salary where
        Emp.dno = Dept.dno AND
        Dept.loc = 'SEOUL') > 500000
    Then Action4
R5 : On Insert Emp
    If SUM(Emp.salary where
        Emp.age> 26 AND
        Emp.dno = Dept.dno AND
        Dept.loc = 'PUSAN') >= 250000
    Then Action5
R6 : On Delete Emp
    If SUM(Emp.salary) <= 1000000
    Then Action6
```

(그림 2) 스키마와 능동규칙 예

전 처리 기능인 분석모듈은 데이터베이스 시스템에 존재하는 능동규칙들을 분석하여 능동규칙 이름, 능동규칙 사건, 능동규칙의 조건부에 있는 술어의 개수를 규칙 정보 파일에 저장하고, 조건 술어 파일에는 각 규칙의 이름과 조건부의 조건 술어들을 분리하여 저장한다.

규칙 정보 파일의 목록은 데이터베이스를 변경하는 사건의 튜플들이 차이트리에 저장될 때 트리의 각 노드에 존재하는 규칙의 정보와 비교하여 조건부의 만족 여부를 조사하는 작업에 사용되어 진다. 능동규칙의 조건 술어에 대한 만족 여부는 규칙 정보 파일에 저장된 규칙 개수만큼의 능동규칙 목록이 발생되어야 판단할 수 있다. 예로 사용된 능동규칙을 분석모듈에 적용하였을 때의 결과는 (그림 3)과 같다.

```
R1 I 3
R2 U 2
R3 I 3
R4 I 1
R5 I 1
R6 D 1
```

(a) rule information file

```
R1 Emp.eno > 500
R1 Emp.eno < 2500
R1 Emp.salary >= 11000
R2 Emp.salary >= 10000
R2 Emp.salary <= 25000
R3 Emp.age > 26
R3 Emp.dno = Dept.dno
R3 Dept.loc = 'PUSAN'
R4 SUM(Emp.salary where
    Emp.dno = Dept.dno AND
    Dept.loc = 'SEOUL') > 500000
R5 SUM(Emp.salary where
    Emp.age > 26 AND
    Emp.dno = Dept.dno AND
    Dept.loc = 'PUSAN') >= 250000
R6 SUM(Emp.salary) <= 1000000
```

(b) condition predicate file

(그림 3) 분석모듈 처리 결과

규칙 정보 파일에는 6개의 능동규칙의 정보가 존재하고 규칙 R1은 삽입사건과 단일 조건 술어가 3개, 규칙 R2는 수정연산과 단일 조건 술어 2개, 규칙 R3은 삽입사건과 단일 조건 술어 3개, 규칙 R4는 삽입사건과 조건 술어 1개, 규칙 R5는 삽입사건과 조건 술어 1개, 그리고 규칙 R6는 삭제사건과 조건 술어 1개가 존재하는 정보를 갖는다. 조건 술어 파일에는 분석모듈에서 분리된 조건 술어들이 존재하고 규칙 R3는 조인연산을 갖는다. 특히 규칙 R4, R5, 그리고 R6는 집계함수, 조인연산, 그리고 선택연산으로 조합된 조건 술어를 갖는다.

3.3 트리 구축

구축모듈은 분석모듈에서 생성된 조건 술어 파일을 이용하여 릴레이션들의 애트리뷰트에 따라 조건 술어들을 분류하여 선택연산, 조인연산, 그리고 집계함수를 처리할 수 있는 계층화된 구조인 차이트리와 분류트리를 생성한다.

3.3.1 차이트리

차이트리는 능동규칙의 사건이 발생되면 해당되는 튜플들을 차이 릴레이션 대신에 구축되어 조건 술어와 연관된 노드에 삽입된 튜플들의 실제 내용이 애트리뷰트 별로 저장됨과 동시에 노드의 정보들을 참조하여 조건부 평가가 처리되는 구조이다. 차이트리의 구조는 조건부 처리 시스템의 전 처리 기능의 구축모듈에서 결정되기 때문에 차이트리를 구성하는 시간은 조건부 처리에 큰 영향을 주지 않고, 또한 조건부 평가를 효율적으로 처리할 수 있는 구조를 갖고있어 능동규칙의 처리를 향상시킬 수 있다.

차이트리는 능동규칙의 조건 평가를 위한 노드와 실제 자료를 저장하기 위한 배열로서 이루어진다. 능동규칙의 조건 평가를 위한 차이트리의 노드 구조는 (그림 4)이다.

차이트리는 릴레이션의 애트리뷰트별로 분류되어 구축되기 때문에 차이트리의 루트노드에는 각 애트리뷰트에 해당되는 트리의 노드들을 가르키는 포인터가 있어야 한다. 조

a_y	j_y	j_f	r_o	const	d_p	r_l

a_y : 집계함수 처리 여부
j_y : 조인연산 수행 여부
j_f : 조인연산의 애트리뷰트 정보
r_o : 관계연산자
const : 관계연산자의 상수 값
d_p : 애트리뷰트 저장 포인터
r_l : 규칙이름 목록

(그림 4) 노드 구조

건 술어 파일에 있는 조건 술어는 차이트리의 루트노드를 통하여 차이트리를 구성하며, 사건에 관련된 튜플들이 삽입될 때 루트노드를 통하여 조건 검사가 이루어지고 실제 값이 저장된다. (그림 5)는 Emp 릴레이션에 대한 차이트리의 루트노드 구조이다.

eno	name	age	salary	dno	jno

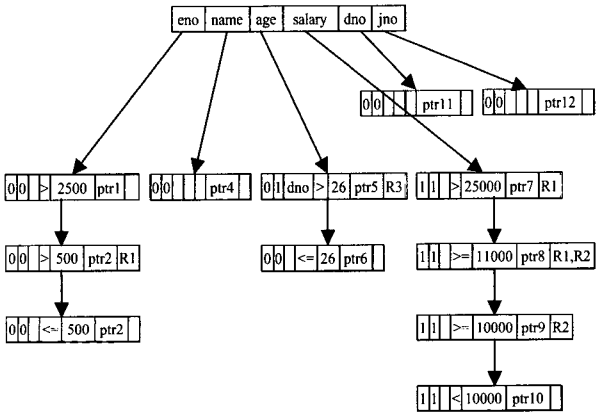
(그림 5) 루트노드 구조

구축모듈이 차이트리의 구조를 만들기 위해서는 능동규칙을 분석하는 분석모듈에서 생성된 조건 술어 파일을 사용한다. 분석모듈에 있는 조건 술어중 salary 필드를 갖는 조건 술어를 사용하여 일부분의 차이트리를 구축한다. 조건 술어 파일에서 salary 필드와 관련된 조건 술어들은 (그림 6)과 같다.

```
R1 Emp.salary >= 11000
R2 Emp.salary >= 10000
R2 Emp.salary <= 25000
R4 SUM(Emp.salary where
    Emp.dno = Dept.dno AND
    Dept.loc = 'SEOUL') > 500000
R5 SUM(Emp.salary where
    Emp.age > 26 AND
    Emp.dno = Dept.dno AND
    Dept.loc = 'PUSAN') >= 250000
R6 SUM(Emp.salary) <= 1000000
```

(그림 6)Salary 필드 조건 술어

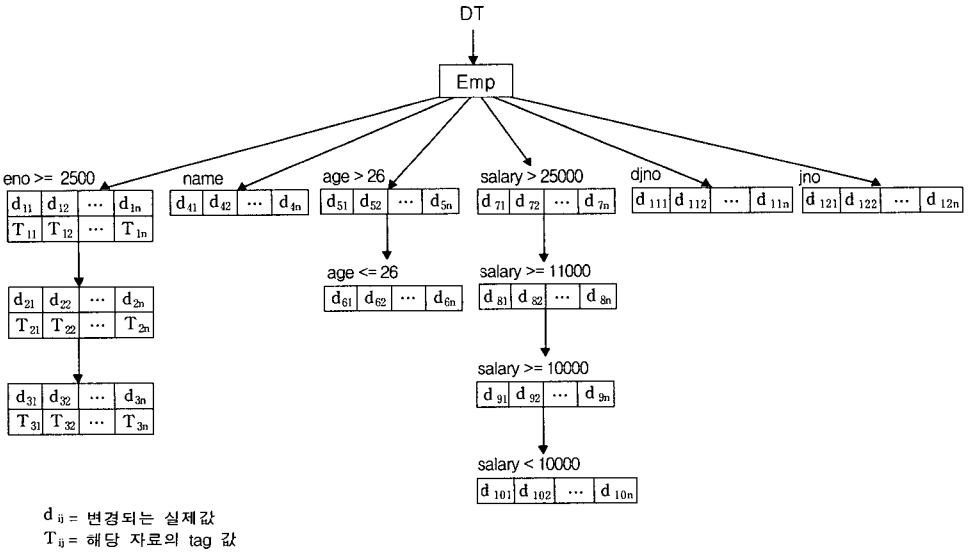
(그림 6)을 구축모듈에서 처리 된 차이트리의 구조는 (그림 7)이다.



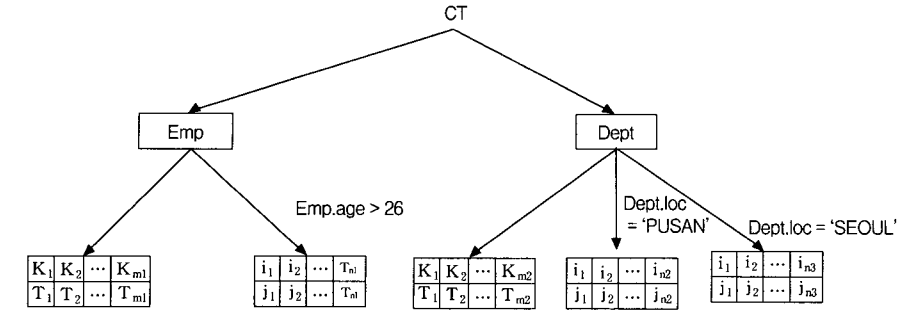
(그림 7) 차이트리 구조

구축모듈 수행 후의 실제 자료가 저장된 형태는 (그림 8)이다.

eno 노드의 1행에는 실제 값이 저장되고, 2행에는 삭제 태그 값을 저장하여 삭제연산이 발생하는 경우 튜플을 삭



(그림 8) 차이트리 자료 저장 구조



K_i = 기본 릴레이션의 key값
T_i = 해당자료의 tag값
i_i = 조인연산에 적용되는 필드들의 key값이 저장된 배열 인덱스
j_i = 해당 인덱스의 실제 key값

(그림 9) 분류트리 구조

제하지 않고 태그정보만을 0으로 변경하여 연산처리 시간을 단축시킨다. Emp 릴레이션에 대한 차이트리의 자료 저장 배열 구조로서 Emp 릴레이션에 새로운 튜플이 삽입될 경우 그 튜플의 애트리뷰트들이 차이트리의 노드에 저장된다.

3.3.2 분류트리

분류트리는 기본 릴레이션의 튜플에서 기본 키인 애트리뷰트와 조인연산에 사용될 애트리뷰트를 분리하여 저장하고 연산에 관련된 다른 릴레이션들의 애트리뷰트들을 트리 구조로 저장한다.

능동규칙의 조건부 평가에서 조인연산은 다른 데이터베이스 연산에 많은 처리비용이 요구된다. 그래서 이 문제를 해결하기 위하여 전 처리 기능의 구축모듈에서 조건 술어 파일을 이용하여 조인연산이 적용된 필드를 대상으로 분류트리를 만든다.

앞에서 제시한 규칙의 조건부 예를 사용하여 차이트리를 구축한 결과 조인연산이 적용된 필드는 age이며, 차이트리에서 age가 26보다 큰 노드에서 조인연산이 발생된다. 차이트리의 age 필드와 조인연산의 대상이 되는 자료는 Dept 릴레이션의 애트리뷰트 loc에서 'PUSAN'과 'SEOUL' 값을 갖는 자료이다. 따라서 기본 Emp 릴레이션에서 이 자료들만을 분리하여 기본 키 필드의 값과 태그정보를 분류트리에 추가한다. 예제로 사용되고 있는 능동규칙의 분류트리 구조는 (그림 9)이다.

3.3.3 집계함수 테이블

능동규칙의 조건부에 집계함수가 있을 경우 조건부 처리 시스템은 집계함수 테이블을 만든다. 점진적 평가처럼 이전 결과를 기억하여 데이터베이스 변경이 발생하는 경우, 이전 결과에 현재 변경되는 값만을 반영함으로써 새로운 집계함수 값을 쉽게 얻을 수 있도록 한다.

분석모듈에서 발생된 조건 술어 파일을 이용하여 구축모듈에서 집계함수 테이블이 생성된다. (그림 10)은 조건 술어 파일에서 인출된 조건 술어가 집계함수인 경우에 이를 처리하는 프로시저이다.

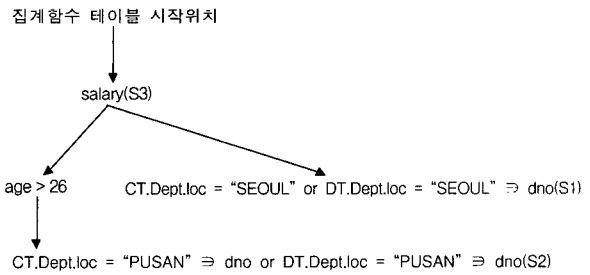
```
INPUT        Agg_ftn file is included in condition predicate file
OUTPUT       Aggregate function table
TEMPORARIES  ln - 집계함수 조건 술어 파일에서 하나의 조건 술
               어를 인출하여 저장
             attr_name - 인출된 조건 술어내의 애트리뷰트 이
               름 저장
PROCEDURE
 1. repeat
 2. Fetch a ln from Agg-ftn file.
 3. Fetch a attribute_name from ln which used to calculate func-
    tion value.
 4. if ln has not substring 'where' then
 5. Set to 1 aggfn_yn of all corresponding attribute's delta tree node.
 6. Calculate and write sum of database's attribute value.
 7. else if ln has not join operation then
 8. Implement delta tree node
 9. Set to 1 aggfn_yn of corresponding delta tree node.
10. Calculate and write sum of database's attribute value which
    satisfies condition.
11. else
12. Implement delta tree node and classification tree node
13. Set to 1 aggfn_yn of corresponding delta tree node.
14. Calculate and write sum of database's attribute value which
    satisfies condition.
15. end-if
16. Implement aggregate_function_process node.
17. until not EOF
END-PROCEDURE
```

(그림 10) 집계함수 테이블 생성 프로시저

집계함수의 경우도 값을 산출할 자료의 범위가 데이터베이스 전체 튜플, 선택연산을 만족하는 튜플, 그리고 조인연산 조건을 만족하는 튜플 등으로 나누어 볼 수 있다. 집계함수 테이블 생성 프로시저는 각각의 경우에 대해 판별하여

집계함수를 처리한다. 집계함수 값을 산출할 자료의 범위가 데이터베이스 전체 튜플인 것은 조건 술어 내에 'Where'가 없는 경우이고 선택연산과 조인연산의 판별은 참조되는 릴레이션의 수를 이용한다.

집계함수 테이블은 집계함수가 처리된 값, 이 값이 조건 술어와의 관계, 그리고 규칙이름으로 구성된다. 조건은 평가된 거짓의 값을 가지며, 규칙이름은 산출된 집계함수 값이 조건을 만족할 경우에 동작부를 수행하는 능동규칙이다. (그림 11)은 능동규칙 R4, R5, 그리고 R6의 조건부에 대한 집계함수 처리 구조이며 집계함수 테이블은 <표 1>과 같다.



(그림 11) 집계함수 처리 구조

〈표 1〉 집계함수 테이블

인자이름	값		
집계함수 처리값	S1	S2	S3
조 건	S1 > 500000	S2 >= 250000	S3 <= 1000000
규칙이름	R4	R5	R6

S1 : SUM(Emp.salary where Emp.dno = Dept.dno AND Dept.loc = 'SEOUL')
S2 : SUM(Emp.salary where Emp.age > 26 AND Emp.dno = Dept.dno AND Dept.loc = 'PUSAN')
S3 : SUM(Emp.salary)

3.4 사건 연산 처리

능동 데이터베이스 시스템은 사건을 탐지하여 조건을 만족하는 능동규칙을 수행하는 메카니즘을 갖고 있기 때문에 규칙의 사건과 조건부 평가는 서로 연관되어 있다. 규칙의 사건은 삽입, 삭제, 그리고 수정연산을 갖고 있으며, 차이트리는 이들 연산에 따라 여러 형태로 동작되면서 조건부를 평가한다.

능동규칙에서 삽입사건이 발생하면, 능동규칙의 조건부에 상관없이 사건에 관련된 자료가 차이트리에 삽입된다. 삽입된 튜플은 애트리뷰트별로 보관되는데, 차이트리 노드의 조건을 만족하는 위치를 찾아 저장된다.

삭제사건이 발생하면, 사건과 관련된 튜플을 기본 릴레이션 또는 차이트리에서 삭제해야 한다. 본 논문에서 제안한 조건부 평가 시스템은 릴레이션에 대한 삭제작업이 조건부 평가에 부담을 줄 수 있기 때문에 삭제사건과 관련된 튜플을 실제로 삭제하지 않고 차이트리와 기본 릴레이션에 연관된 분류트리를 검사하여 해당 기본 키값의 태그를 0으로 변경한다.

수정사건이 발생하면, 사건과 관련된 튜플을 삭제 처리하고 수정되는 튜플을 삽입연산으로 처리한다. 수정사건 역시 수정된 자료가 차이트리에 저장되므로, 수정자료가 저장되는 노드와 연결된 능동규칙 정보를 이용하여 동작부의 수행을 판단한다.

규칙의 사건이 발생하여 조건부를 처리하는 동안에 해당 튜플의 애트리뷰트에 의해 만족되는 능동규칙의 목록은 차이트리와 분류트리에 저장된 정보를 이용하여 자동적으로 알 수 있다. 능동규칙의 목록은 분석모듈에서 생성한 규칙 정보 파일의 사건의 유형과 단일 조건 술어의 수와 비교하여 조건부의 진위를 판단함으로써 능동규칙을 처리할 수 있다. 처리모듈의 프로시저는 (그림 12)와 같다.

INPUT Database operation
OUTPUT Result of database operation
PROCEDURE
1. **while** database operation is occurred
2. **if** database operation is 'insert' **then**
3. Compare each attribute value with delta tree node's const and save it to proper location.
4. **if** attribute's node has 1 at aggfn_yn **then**
5. Add value at aggregate function value.
6. **end-if**
7. **if** attribute's node has 1 at join_yn **then**
8. Search join_fld at classification tree and delta tree.
9. **end-if**
10. Compare event-type and number_of_rules with rule information file.
11. **end-if**
12. **if** database operation is 'delete' **then**
13. Search eno at classification tree and delta tree, and change tag value.
14. **if** deleted tuple is reflected to aggregate function value **then**
15. Subtract value from aggregate function value.
16. **end-if**
17. Compare event-type and number_of_rules with rule information file.
18. **end-if**
19. **if** database operation is 'update' **then**
20. Delete operation is performed for old tuple.
21. Insert operation is performed for new tuple.
22. Compare event-type and number_of_rules with rule information file.
23. **end-if**
20. **end-while**
21. **END-PROCEDURE**

(그림 12) 사건연산 처리 프로시저

4. 성능 평가

본 절에서는 본 논문에서 제안한 조건부 처리 시스템을 구현하여 전처리 기능이 없는 조건부 평가 기법과 비교하여 분석한다. 실험은 OS를 Windows NT로 장착한 128MB 주기억장치를 가진 퍼스널 워크스테이션 상에서 Visual C++ 6.0로 구현하였다. 그리고 처리시간을 측정하기 위해 clock()함수를 사용하였고, rand( )함수를 이용하여 각 애트리뷰트의 특성에 따라 임의로 연산을 발생시켰다.

본 실험에서는 rand( )함수를 사용하여 Emp 릴레이션의 튜플 10000개, Dept 릴레이션의 튜플 1000개, 그리고 Job 릴레이션의 튜플 20개를 만들었다. 또한, 조인연산이 적용되는 Dept 릴레이션에 대해 연산 대상이 되는 릴레이션의 튜플 수를 조정하였다. 예를 들어, 조인연산에서 Dept.loc = 'PUSAN'과 Dept.loc = 'SEOUL'이라는 조건 술어가 사용되는 경우 Dept 릴레이션의 전체 튜플의 수를 1000개를 만들어, 그 중에 위의 두 조건을 만족하는 튜플의 비율이 2%, 10%, 20%가 되도록 릴레이션을 생성하여 전 처리기가 없는 조건부 평가 방법과 본 논문에서 제안한 시스템이 처리하는 시간을 측정하였다.

전처리 기능이 없는 조건부 평가 방법과 본 논문에서 제안한 조건 처리 시스템의 조건부 처리 시간을 각각 NP, DT라 한다.

실험 1. 본 실험은 선택연산과 조인연산을 포함한 능동규칙의 조건부 처리 시간을 측정한다. 실험에 사용된 규칙의 조건부는 다음과 같다.

```
R1 : Emp.salary >= 15000 and Emp.dno = Dept.dno and Dept.loc
     = "PUSAN"
R2 : Emp.salary >= 10000 and Emp.dno = Dept.dno and Dept.loc
     = "SEOUL" and Emp.jno = Job.jno and Job.title = "Clerk"
```

<표 2>는 규칙의 조건부에 대한 만족 비율을 2%, 10%, 그리고 20%로 설정하여 연산수에 따른 R1과 R2를 처리한 시간을 초단위로 나타내었다.

〈표 2〉 조인연산이 포함된 조건부 처리시간

연산수	수행시간(초)					
	NP			DT		
	2%	10%	20%	2%	10%	20%
1000	3.74	3.36	3.34	0.33	0.35	0.37
3000	10.29	10.09	10.10	0.95	1.03	1.10
5000	17.08	16.82	16.85	1.59	1.65	1.77
7000	23.9	23.59	23.58	2.21	2.35	2.50
10000	30.7	30.45	30.30	2.83	2.99	3.22

실험 2. 본 실험은 집계함수와 조인연산을 포함한 능동규칙의 조건부 처리 시간을 측정한다. 실험에 사용된 규칙의 조건부는 다음과 같다.

```
R1 : SUM(Emp.salary where Emp.dno = Dept.dno and Dept.loc =
     "PUSAN") >= 1000000
R2 : SUM(Emp.salary >= 10000 and Emp.dno = Dept.dno and
     Dept.loc = "SEOUL" and Emp.jno = Job.jno and Job.title
     = "Clerk") >= 500000
```

실험 2에서 사용되는 조건부 R1은 Emp 릴레이션과 Dept 릴레이션에 대한 조인연산과 집계함수로 구성되어 있고, R2는 조인연산에 Job 릴레이션을 추가로 참여시켰다. R1과 R2에 대한 실험 결과는 <표 3>과 같다.

〈표 3〉 집계함수가 포함된 조건부 처리시간

연산수	수행시간(초)					
	NP			DT		
	2%	10%	20%	2%	10%	20%
1000	3.44	3.47	3.53	0.37	0.39	0.37
3000	10.34	10.58	10.53	0.98	1.11	1.13
5000	17.25	17.36	17.51	1.60	1.68	1.86
7000	24.17	24.28	24.49	2.27	2.32	2.61
10000	31.06	31.19	31.46	2.92	3.05	3.36

실험 3. 본 실험은 선택연산, 조인연산, 그리고 집계함수 연산이 포함되는 능동규칙의 조건부가 처리되는 시간을 측정한다. 실험에 사용된 규칙의 조건부는 다음과 같다.

```
R1 : Emp.eno >= 1000 and Emp.eno <= 2500
R2 : Emp.salary >= 15000 and Emp.dno = Dept.dno and Dept.loc
     = "SEOUL"
R3 : SUM(Emp.salary where Emp.dno = Dept.dno and Dept.loc =
     "PUSAN") >= 1000000
R4 : Emp.salary >=10000 and Emp.salary < 25000 and Emp.dno =
     Dept.dno and Dept.loc = "PUSAN" and Emp.jno = Job.jno
     and Job.title = "Clerk"
```

실험 3에서 사용되는 조건부 R1은 Emp 릴레이션에 대한 선택연산으로 구성되고, R2는 Emp 릴레이션과 Dept 릴레이션에 대한 조인연산이 추가된 형태이다. 그리고 R3는 집계함수와 조인연산으로 구성되어 있고, R4는 Emp 릴레이

〈표 4〉 복잡한 조건부 처리 시간

연산수	수행시간(초)					
	NP			DT		
	2%	10%	20%	2%	10%	20%
1000	3.44	3.43	3.51	0.47	0.51	0.47
3000	10.36	10.44	10.46	1.28	1.45	1.51
5000	17.30	17.37	17.41	2.11	2.46	2.44
7000	24.23	24.28	24.51	2.97	3.38	3.45
10000	31.17	31.18	31.54	3.84	4.35	4.40

션에 대한 Dept 릴레이션과 Job 릴레이션의 조인연산으로 구성되어 있다. 실험 3의 결과는 <표 4>와 같다.

5. 결 론

본 논문에서는 조건부를 평가하기 전에 능동규칙을 미리 알 수 있는 점을 이용하여 전처리 기능을 갖는 조건부 처리 시스템을 제안하였다. 조건부 처리 시스템은 능동규칙을 처리하기 전에 규칙의 정보를 이용하여 조건부 평가를 지원할 수 있는 구조들을 미리 구축함으로써 능동규칙을 처리한다.

능동규칙의 조건부에서 조인연산과 집계함수를 처리하는 사건이 발생하면, 대량의 데이터들이 보관되어 있는 릴레이션을 참조해야 하며 또한 이들 연산과 관련된 사건이 발생할 때마다 조인연산에 참조되는 튜플들을 찾기 위해 전체 릴레이션을 검색해야 한다. 이는 능동규칙 처리의 효율에 중요한 영향을 줄 수 있는 요소가 된다. 본 논문에서 제안한 분류트리는 조인연산의 대상이 되는 튜플의 키 값들을 전처리 과정에서 분류하여 보관함으로써 조인연산의 처리시간을 단축할 수 있으며, 조인연산에 참조되는 튜플의 수가 적을 경우 연산처리가 더욱더 효율적이라는 사실을 실험을 통해 알 수 있었다. 또한 집계함수의 값을 집계함수 테이블의 형태로 보관하여 능동규칙의 사건 즉, 삽입연산, 삭제연산, 그리고 수정연산에 따라 집계함수의 값을 재 계산함으로써 집계함수 처리시간을 줄일 수 있었다.

본 논문에서 제안된 시스템이 전처리 기능을 가지 않는 조건부 평가 기법보다 데이터베이스 연산과 집계함수의 평가비용이 적게 들 수 있음을 알 수 있다. 따라서 본 논문의 조건부 처리 시스템은 데이터베이스의 강력한 능동성을 갖는 능동 데이터베이스 시스템의 수행 성능을 향상시키는데 도움을 줄 수 있다.

앞으로 수행되어야 할 연구 과제는 다음과 같다. 본 논문에서 사용한 분류트리와 차이트리의 노드의 구조가 배열로 되어 있기 때문에 응용성이 부족하다. 그래서 트리의 노드를 동적인 구조 형태로 변경할 필요성이 있다.

본 논문에서 제안한 트리들은 능동규칙의 조건부에 따라 애트리뷰트들을 분류하여 저장하는 구조이므로 연산처리에 병렬성이 존재할 수 있다. 이런 구조의 특성을 활용하여 분산처리 시스템의 활용에 대한 연구가 있으면 다양한 데이터베이스 응용 처리에 적용할 수 있을 것이다.

참 고 문 헌

[1] Dayal. U., "The HiPAC Project : Combining Active Databases and Timing Constraints," *ACM SIGMOD record, 17(1)*, pp.51-70, 1988.

[2] Chaudhuri. S. and Shim. K., "Optimization of Queries with User-Defined Predicates," *ACM Transactions on Database System, 24(2)*, June, 1999.

[3] Fabret. F., Regnier. M. and Simon. E., "An Adaptive Algorithm for Incremental Evaluation of Production Rules in Databases," In *Proceedings of the 19th International Conference on Very Large Data Bases*, pp.455-467, Dublin, Ireland, August, 1993.

[4] Risch. T., "Monitoring Database Objects," In *Proceedings of the 15th International Conference on VLDB*, pp.445-453, 1989.

[5] Widom. J. and Ceri. S., *Active Database Systems : Triggers and Rules for Advanced Database Processing*. Morgan Kaufmann Publishers, Inc., pp.73-80, 1995.

[6] Rosenthal. A., Chakravarthy. S., Blaustein. B., and Blakeley. J., "Situation monitoring for active databases," In *Proceedings of the 15th International Conference on Very Large Data Bases*, pp.455-464, Amsterdam, The Netherlands, August, 1989.

[7] Simon. E., Kiernan. J. and de Maindreville. C., "Implementing High Level Active Rules on Top of a Relational DBMS," In *Proceedings of the 18th International Conference on Very Large Data Bases*, pp.315-326, Vancouver, British Columbia, August, 1992.

[8] Baralis. E. and Widom. J., "Using Delta Relations to Optimize Condition Evaluation in Active Databases," *RIDS'95 (Rules in Database Systems), Springer Lecture Notes in Computer Science*, pp.292-308, Athens, Greece, September, 1995.

[9] Hanson. E. N., "Rule Condition Testing and Action Execution in Ariel," In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp.49-58. San Diego, California, June, 1992.

[10] Hanson. E. N., Bodagala. S., Hasan. M., Kulkarni. G., and Rangarajan. J., "Optimized Rule Condition Testing in Ariel using Gator Network," Technical Report TR-95-027, University of Florida CIS Dept., 1995.

[11] Schor. M. I., Daly. T. P., Lee. H. S. and Tibbitts. B. R., "Advances in RETE Pattern Matching," *In Proceedings of the 5th National Conference on Artificial Intelligence*, pp. 226-232, Philadelphia, Pennsylvania, August, 1986.

[12] Miranker. D. P., "TREAT : A Better Match Algorithm for AI Production Systems," In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pp.42-47, August, 1987.

이 기 욱

e-mail : kwlee@yongma.tmc.ac.kr

1985년 계명대학교 전자계산학과 졸업 (공학사)

1987년 동국대학교 전자계산학과 졸업 (공학석사)

2001년 계명대학교 컴퓨터공학과 졸업 (공학박사)

1991년~현재 동명대학 정보통신계열 부교수

관심분야 : 능동데이터베이스, 데이터 웨어하우스, AI 응용, 전자상거래

김 태 식

e-mail : tskim@kmu.ac.kr

1984년 계명대학교 전자계산학과 졸업 (공학사)

1987년 미네소타대학교 전자계산학과 졸업 (공학석사)

1992년 노스다코다대학교 전자계산학과졸업 (공학박사)

1992년~현재 계명대학교 공학부 컴퓨터공학 교수

관심분야 : 카오스, AI 응용, 정보검색시스템