韓國 OA學會 論文誌
第3卷 第4號, 1998. 12.

98-3-4-2-3

능동규칙의 조건부 처리 기법

이 기 욱*

An Technique for the Active Rule Condition

Ki-Wook Lee*

요 약

능동규칙의 조건부 처리에 사용되는 데이터베이스 연산들은 처리되는 시간들이 상당히 크기 때문에 능동 데이터베이스 시스템의 성능에 중요한 영향을 미친다. 그래서 조건부에서 발생되는 연산들의 처리 시간을 최소화 시켜야만 시스템의 성능을 높일 수 있다. 기존 연구의 처리 기법들은 기본 데이터베이스 연산과 한정된 일부 집계함수만을 효율적으로 처리하고 있다.

본 연구에서는 릴레이션의 구조화와 상태테이블을 사용한 처리 기법을 제시한다. 릴레이션들을 분류트리로 구조화시키고, 삭제 정보 테이블을 도입함으로써 기본 데이터베이스 연산의 처리시간을 줄일 수 있다. 또한 이진 검출트리와 릴레이션의 상태테이블을 도입함으로써 처리비용이 큰 집계함수를 효율적으로 처리할 수 있어 능동 데이터베이스 시스템의 이용을 극대화시킬 수 있다.

Abstract

AS it takes a considerable time for database operations for processing the condition part of active rule, the operations have an important effect on the efficiency of active database system. The processing time of operations should be minimized in order to improve the efficiency of system. The previous works are limited to basic database operations and the partial aggregate functions.

In this paper, the processing technique using the structuralization and the state table of relations is suggested. The processing time for basic database operations can be reduced with the structuralization of relations to classification tree and the introduction of deletion information table. With the introduction of binary search tree and relation state table, the aggregate function which has a big of processing cost can be processed effectively and the function of the active database system can be maximized.

* 동명대학 사무자동화과 부교수
논문접수 : 98.11.6. 심사완료 : 98.12.7.

I. 서 론

데이터베이스를 비롯한 컴퓨터 시스템 응용분야의 확대와 처리능력에 대한 수요자의 요구가 점차 고차원화 되면서 시스템이 자율적으로 능동성을 가지고 처리 할 수 있는 이른바 인텔리전트한 시스템이 점차 요구되어지고 있다.

기존의 데이터베이스 시스템들은 수동적 시스템으로서 사용자나 응용프로그램에 의해 외부적으로 요구되는 질의나 트랜잭션만을 수행한다. 많은 응용들에서는 특정한 상황을 감시하여 그 상황이 발생하는 경우 적절한 시간 내에 트리거(trigger)를 해야하는 경우가 있다. 예를 들어, 재고 관리 시스템은 재고 보유량을 감시하여 어떤 품목의 보유량이 한계 값 이하가 될 경우, 재 주문 동작을 발생해야 한다. 이 동작은 수동적 데이터베이스 시스템(passive database system)에 추가하는 방법으로 구현되는데, 두 가지 방법이 사용되고 있다. 첫 번째 방법은 조건 검사의 문법을 재고 데이터베이스를 수정하는 모든 프로그램에 포함시키는 경우로 소프트웨어 엔지니어링(software engineering)적인 측면에서는 부적절한 방법이다. 두 번째 방법으로 조건 검사 응용 프로그램이 데이터베이스에 추가되어 주기적으로 적절한 조건을 검사하도록 하는 것이다. 그러나, 만약 이 프로그램의 수행빈도가 너무 높다면 이는 비효율적이고, 수행빈도가 낮다면 조건에 대한 적절한 시간 내의 탐지가 되지 않을 것이다[1].

능동 데이터베이스 시스템은 능동규칙처리 기반을 도입함으로써 특정한 상태를 탐지하여, 사건이 발생하면 규칙 조건부에서 참, 거짓의 값을 산출하여 이 값에 따라 동작이 수행된다. 이와 같은 조건부의 평가는 사건이 발생할 때마다 수행되기 때문에 처리하는 방법에 따라 시스템의 성능에 중요한 영향을 미친다[2].

본 연구는 능동 데이터베이스 시스템의 능동규칙 조건부에서 발생되는 기본적인 데이터베이스 연산들 뿐 만 아니라 처리비용이 큰 집계함수를 효율적으로 처리할 수 있는 조건부 처리 기법을 제시하여 능동 데이터베이스 시스템의 이용을 극대화 시키고자 한다.

II. 능동 규칙 시스템

능동 데이터베이스 시스템이 제공하는 기능은 크게 무결성 제약 조건과 사건-조건-동작(EventCondition-Action:ECA)규칙인 능동규칙 시스템(active rule system)으로 나눌 수 있다.

무결성 제약 조건은 데이터베이스가 항상 유지해야 하는 상태를 명시하는 것으로서 이전부터 처리되어 왔다. 능동규칙 시스템은 특정한 상황이 되었을 때 정해진 작업을 수행하도록 하는 일반적인 능동작업을 표현한다.

2.1 능동 규칙

일반적으로 인공지능 시스템(artificial intelligence system)에서의 생성규칙은 '조건 → 동작'의 형식을 갖는다. 시스템에서 모든 규칙을 통한 추론 엔진 사이클은 작업 메모리 안의 자료를 가진 규칙의 조건 부분과 연결된다. 연결되는 모든 규칙 중의 한 규칙이 충돌 분해정책에 의해 선택되어 지고, 이 선택된 규칙은 수행된다. 즉, 그 규칙의 동작부가 실행된다. 동작부는 연결된 자료에 따라 작업 메모리를 수정하고, 동작 사이클은 규칙들이 더 이상 연결되지 않을 때까지 계속된다.

능동 규칙(active rule)은 사건-조건-동작(Event-Condition-Action:ECA) 패러다임에 기초를 둔다. 이들의 형태는 다음과 같다.

on event if condition then action

사건은 SQL 데이터 조작의 기본 구성 요소이며, 삽입(INSERT), 삭제(DELETE) 그리고 갱신(UPDATE)연산이 있다. 조건은 SQL안에 표현되는 데이터베이스 상태의 논리 값이다. 동작은 선택, 삽입, 삭제 그리고 갱신연산등을 포함하는 임의의 SQL 질의에 의해 수행될 수 있으며, 규칙 조작 문장(rule manipulation statements)과 업무 명령(transactional instruction)을 포함할 수 있다.

ECA 패러다임은 간단하며, 직관적인 의미를 갖는다. 사건이 발생했을 때, 조건이 만족되면 동작을 수행한다.

이러한 기본적인 의미는 대부분의 능동규칙 시스템이 추구한다. 즉, 규칙에서 적절한 사건이 발생하면 '트리거된', 조건이 평가되면 '고려된' 그리고 동작이 수행되면 '수행된'이라고 한다. 그러나 다른 시스템에 활동 규칙의 트리거링, 고려 그리고 수행의 의미를 정의하는 것은 약간씩 차이가 있다.

능동규칙은 데이터베이스 시스템에 강력한 능동성을 제공할 수 있기 때문에 능동 데이터베이스 시스템의 핵심이 되는 기술이라 할 수 있다[3].

2.2 능동규칙 조건부 처리

능동규칙의 조건부에는 선택(select), 프로젝션(projection), 조인(join) 그리고 집계함수(aggregate function) 연산들 즉 SUM, AVG, COUNT, MIN, MAX 등이 있는데, 이들은 처리되는 시간이 상당히 큰 데이터베이스 연산들이므로, 전체 데이터베이스 처리에 영향을 미친다. 그래서 능동 데이터베이스의 성능 향상을 위해 능동규칙의 조건부에 관한 연구가 매우 중요한 요소가 되며[4], 조건부에서 발생되는 연산들의 처리시간을 최소화 시켜야만 시스템의 성능을 높일 수 있다.

규칙수행에서 조건평가의 특성으로, 첫째, 규칙의 수행은 데이터베이스의 상태변화에 대한 시스템의 요청에 의하여 수행되고, 둘째, 수행요청이 반복적이며, 셋째, 질의 내용이 미리 알려져 있고, 넷째, 수행 목적이 튜플 검색이 아닌 조건의 진위 판별에 있음을 밝혔다. 이러한 특성은 효율적 조건평가를 위한 중요한 단서를 제공한다.

점진적 평가(incremental evaluation)방법은 두 가지 방향에서 접근될 수 있다. 하나는 데이터베이스적 접근방법으로 뷰 구체화에 대한 연구들로부터 파생된 방법이다. 이 방법에서는 차이 릴레이션을 이용하여 데이터베이스의 상태변경에 대한 이전의 상태와 바뀐 이후의 상태를 모두 유지하고, 이들 사이의 차이를 토대로 조건평가를 수행하는 방법이다[5]. 다른 하나는 인공지능 분야의 규칙언어에서 이용하는 인식망(discrimination network)방법으로서, 조건 절에 명시된 내용을 인식망 구조 속에 등록시킨 후 데이터베이스의 상태 변경 시 상태 변경을 제공한 튜플을 인식망에 삽입하게 되고, 삽입된 튜플이 망의 하부로 전파되는 과정에서 조건평가가 이루어지도록 구성하는 방법이다. 망을 이용한 점진적 평가는 집계함수를 효과적으로 처리하지 못하는 문제점을 갖고 있다. 증첩된 능동규칙 조건부에는 집계함수가 자주 사용되므로 효율적인 집계함수 처리 방법이 제공되어야 한다.

2.3 조건 평가 연구

능동규칙의 조건부 처리를 위한 여러 기법들이 제안되고 있다. [6]의 연구에서는 조건부 평가를 위해 A-TREAT라는 규칙조건 검사 망을 사용하였다. 이 방법은 조건연산의 수를 감소시켜 규칙처리의 성능을 높여주는 기법을 적용하였다. A-TREAT는 하나의 릴레이션에 대하여 규칙을 처리하였고, 가상 α-메모리 노드를 만들어 다른노드와 관련된 술어를 가진다. 가상 α-메모리 노드는 술어의 값이 참인 것만 저장하여 망에 저장되는 정보의 양을 줄여준다. [7]은는 Gator를 이용하여 규칙조건 처리의 성능을 향상시켰다. Gator망은 일반적인 트리 구조로서 Rete와 TREAT의 특별한 형태이다. 위에서 언급한 A-TREAT망을 이용한 규칙조건 처리 방법과 Gator를 이용한 처리 방법은 같은 메카니즘을 가지나 인식 망 구조 자체는 조금 다르다. A-TREAT는 가상 α-메모리 노드를 가지나 Gator는 가상 α-메모리 뿐 만 아니라 중간단계에서 처리된 술어 값을 가지는 β-메모리 노드를 가진다. Gator망은 A-TREAT보다 다소 효율성을 가진 인식 망이다. 그래서 Gator망을 이용한 규칙조건 처리는 A-TREAT보다 성능이 우수하다고 알려져 있다. 그러나 이들 연구들은 조인과 같이 복잡한 연산의 경우 점진적 평가 기법을 적용하여도 여전히 처리비용이 큰 연산이 되는 미 해결과제가 남아있다. [8]의 연구에서는 능동조정자 객체시스템에서 능동규칙 처리를 부분 차이(partial differencing)의 개념을 제안하여 복잡한 규칙조건들을 처리하였으나 역시 그 효율성에 관해서는 아직 미비한 점이 남아있다. 국내에서는 벡터 표현기법을 이용한 연구가 시도되었다[9]. 이 연구는 하나의 릴레이션을 하나의 벡터로 표현하고 데이터베이스 연산을 벡터 연산으로 변환시켜 조건부 평가 초기에 진리 값이 거짓인 규칙 조건들을 적은 연산 비용으로 찾아내는 방법을 제시하였다. 그러나 집계함수인 MAX와 MIN이 포함된 규칙조건과 같은 의미를 지니는 벡터연산을 찾기 어렵기 때문에 벡터 표현 기법을 범용적인 곳에 직접 적용하기 어렵다는 단점을 가지고 있다. 이와 같이 기존의 능동규칙의 조건부 평가에 대한 대부분의 연구들은 접근 방법은 다르지만 점진적 평가 기법(incremental evaluation tech-

nique)을 기초로 하고 있다[10].

기존의 연구에서는 능동규칙의 조건부를 기본 데이터베이스 연산과 일부 집계함수를 사용한 단일 조건들로 구성하여 처리하거나, 중첩된 연산들로 구성된 조건부를 처리하도록 하고 있으나 기본 데이터베이스 연산과 한정된 일부 집계함수만을 효율적으로 처리하고 있다.

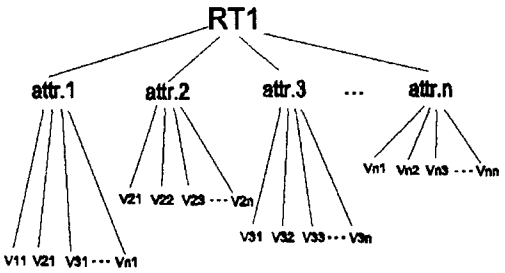

그림 1. 릴레이션 분류 트리
Fig 1. Classification Tree of Relation

III. 제안된 처리 기법

3.1 릴레이션 구조화

차이릴레이션을 이용한 점진적 평가방법에서는 일반 릴레이션의 수가 늘어나면 차이릴레이션의 수도 늘어난다. 그래서 조인연산과 선택연산은 해당 애트리뷰트를 비교해야 하기 때문에 높은 처리비용을 요구한다. 또한 분할 연산의 수도 증가하므로 조건부를 처리하는데 많은 시간을 요한다. 다음은 연산비용이 큰 연산자와 집계함수로 구성된 능동규칙의 예 이다.

```
CREATE RULE Salary Control OF Emp & Org
ON INSERT, DELETE, UPDATE
IF EMP.pay 〉 100 where Emp.sno = Org.sno
      and Org.name = 'kildong'
THEN UPDATE Emp
      SET Pay = .9 * Pay
```

위의 능동규칙에서는 연산비용이 큰 조인연산과 선택연산을 사용되었다. 이와같은 능동규칙의 조건부 처리를 효율적으로 처리하기 위해서 기존릴레이션을 각 애트리뷰트로 분류한 트리로 구조화시킨다. 그림 1은 기존릴레이션에 대한 분류트리(classification tree)이다. 각 애트리뷰트 별로 분류가 되어 있으면 조건부 연산에서 필요로 하는 조인연산이 빠르게 처리되며, 또한 해당 애트리뷰트들을 신속하게 검색할 수 있다. 또한 사건부의 갱신명령을 효율적으로 처리할 수 있다.

사건부의 삽입명령으로 인하여 기존릴레이션에 삽입될 튜플들을 기존릴레이션의 분류트리에 삽입하면 삽입알고리즘이 만들어져야 한다. 또한 기존릴레이션의 분류트리의 재배치 작업으로 인하여 처리 성능이 저하된다. 그래서 삽입튜플들을 위한 삽입릴레이션의 분류트리를 구조화시킨다. 삽입릴레이션의 분류트리는 기존릴레이션의 분류트리와 같은 형태이다.

사건부의 삭제명령으로 기존릴레이션에서 삭제된 튜플들의 처리를 위해 삭제알고리즘이 필요하다. 그러나 삭제알고리즘으로 인한 기존 분류트리의 재배치 작업으로 인하여 조건부 처리비용이 높아진다. 그래서 분류트리에 대한 삭제알고리즘 대신에 기존릴레이션에 대한 삭제 정보를 갖는 삭제 정보테이블을 만든다. 표 1은 기존 튜플과 삭제된 튜플의 정보를 갖는다.

표 1. 삭제 정보 테이블
Table 1. Deletion Information Table

튜플 인덱스	1	2	3	. . .	n
삭제 태그	t_1	t_2	t_3	. . .	t_n

삭제 정보테이블의 삭제 태그(tag)는 삭제된 튜플의 인덱스에 대해 삭제태그가 1로 표시되고 나머지 튜플의 삭제태그에는 0으로 표시된다. 기존 분류트리와 삭제 정보테이블을 이용하면 삭제명령으로 인한 기존릴레이션의 변화를 쉽게 파악할 수 있어 조건부 처리를 효율적으로 수행할 수 있다.

3.2 집계함수 처리

집계함수 중 MAX와 MIN연산의 처리는 연산의 대상이 되는 애트리뷰트들을 이진 검출트리(binary search tree)로 구성한다. 이진 검출트리는 정의 1의 성질을 갖는다.

정의 1. 이진 검출트리 T는 각 노드가 한 개의 키를 갖는 이진트리이며, 각 노드의 키 값이 다음과 같도록 구성한다.

1) 좌측 서브트리의 노드들이 갖는 키 값들은 근노드의 키 값보다 모두 작다.
2) 우측 서브트리의 노드들이 갖는 키 값들은 근노드의 키 값보다 모두 크다.

능동규칙의 조건부에서 사용되는 MAX, MIN연산은 정확한 값의 산출을 목적으로 하지 않고 참 또는 거짓을 판별하기 때문에 정확한 최고 값을 알 수 없다면 최고값이 어느 경계 값에 대해 초과하는지, 같은지, 또는 미만 값 인지의 여부만 판단해도 능동규칙의 조건부를 처리 할 수 있다. 따라서 MAX, MIN연산의 특성 상 애트리뷰트들을 이진 검출트리로 구축하면 연산처리를 쉽게 할 수 있다.

다음의 능동규칙은 집계함수인 AVG연산의 예이다. AVG연산을 처리하기 위해서는 SUM, COUNT연산이 필요하다.

```
CREATE RULE Salary Control OF Emp
ON INSERT, DELETE, UPDATE
IF AVG(Pay) FROM Emp > 100
THEN UPDATE Emp
SET Pay = .9 * Pay
```

각 릴레이션의 수치적 애트리뷰트들에 대한 상태테이블을 도입하면 위의 예에서 사용되는 집계함수를 효율적으로 처리할 수 있다. 표 2는 기존릴레이션의 상태테이블이다. 삽입릴레이션의 상태테이블은 표3의 형태와 같다.

표 2. 기존 릴레이션 상태테이블
Table 2. State Table of Relation

	attr.1	attr.2	· · ·	attr.N
SUM	s1	s2	· · ·	sn

상태테이블(state table)은 계산의 성질을 갖고 있는 애트리뷰트들의 SUM 값을 가지고 있다. 각 애트리뷰트들의 수는 튜플의 수와 같기 때문에 상태테이블은 애트리뷰트들의 수를 가질 필요가 없다. 상태테이블은 튜플의 변화가 있을 때, 즉 삭제와 갱신으로 인해 기존릴레이션에 변화가 발생하면 상태테이블의 정보가 변경된다. 각 상태테이블에는 집계함수 AVG에 대한 언급이 없다. AVG는 SUM과 COUNT연산으로 값이 산출될 수 있기 때문에 상태테이블에 첨가하지 않았다.

각 연산에 대해 상태테이블의 변화를 관찰하면, 삽입명령시 삽입릴레이션에 삽입된 튜플의 수치적 성격을 띠는 애트리뷰트의 값들이 삽입릴레이션의 SUM열에 가산이 되고, 삭제 명령시 기존릴레이션 상태테이블의 SUM열에 감산이 된다. 갱신 명령에서는 식 1과 식 2를 이용하여 상태테이블들에서 SUM열에 해당되는 항목의 값이 변경된다.

$\Delta i = ui - si$ (1)
$si = si + \Delta i$ (2)
ui : 애트리뷰트 변경 값
Δi : 차이 값
si : 상태테이블의 누적 값

집계함수 SUM의 처리는 높은 비용을 요구하기 때문에 상태테이블을 이용하면 집계함수의 연산을 효과적으로 처리할 수 있다.

IV. 결 론

능동 데이터베이스시스템은 스스로 사용자에게 필요한 자료를 처리해 주며 또한, 데이터베이스가 스스로 판단하여 사용자에게 필요한 자료를 제공해 주는 기능을 가진

다. 그래서 처리 속도의 문제점들을 갖고 있는 능동규칙의 조건부를 효과적으로 처리하면 무결성 제약, 트리거와 경고기의 구현, 유도된 자료 유지, 접근 제약 실시, 퀴리의 최적화, 그리고 데이터베이스 재구성 등에도 효율적으로 적용할 수 있을 것이다.

본 연구에서 제시된 조건부 처리 기법은 점진적 평가 방법에서 사용된 차이 릴레이션을 배제한다. 릴레이션의 애트리뷰트들을 분류 처리하기 적합한 구조로 만들고, 각 릴레이션에 대해 상태테이블을 구축하여 집계함수인 SUM, COUNT 그리고 AVG연산에 상태테이블을 사용하면 재 연산이 필요가 없기 때문에 연산 영역을 줄일 수 있어 중첩된 능동규칙의 조건부에 대한 기본 데이터베이스 연산과 집계함수를 효율적으로 처리 할 수 있다.

참고문헌

[1] W. Kim. MODERN DATABASE SYSTEMS. Addison Wesley, pp434-476, 1995.

[2] T. Risch. Monitoring Database Objects, Proceed-ings of the 15th International Conference on VLDB, pp. 445-453, 1989.

[3] C. Zaniolo,S.Ceri, C. Faloutsos, R. Snodgrass, V.Subrahmanian, R. Zicari. Advanced Database System. Morgan Kaufmann Publishers, Inc., pp7-36, 1997.

[4] F. Fabret, M. Reginer, and E. Simon. An daptive algorithm for incremental evaluation of production rules in databases. In Proceedings of the Nineth International Conference on VLDB , pp455-467, 1993.

[5] E. Simon, J. Kiernan, and C. de Maindreville. Implementing High Level Active rules on top of a relational DBMS. Proceedings of the 18th conference on VLDB, pp.315-326, 1992.

[6] J. Widom and S. Ceri. Active Database Systems: Triggers and Rules for Advanced Database Processing. Morgan Kaufmann Publishers, Inc., pp73-80, 1995.

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

[8] M.Sköld and T. Risch. Using Partial Differencing for Efficient Monitoring of Deferred Complex Rule Conditions. 12th International Conference on Data Engineering, 1996.

[9] Dong-Wook Kim, Myoung-Ho Kim, Yoon-Joon Lee, 벡터 표현을 이용한 능동규칙 조건의 효과적 여과 기법, 정보과학회논문지(B), Vol.25, No1, pp.26-36, 1998.

[10] J. Blakeley, P Larson, and F. Tompa. Efficiently Upeating Materialized views. Proceedings of ACM SIGMOD, pp.61-71, 1986.

이기욱

1985년 : 계명대학교 전자계산학과 (공학사)

1987년 : 동국대학교 대학원 전자계산학과 (공학석사)

1998년 : 계명대학교 대학원 전자계산학과 박사과정 수료

1991년 ~ 현재: 동명대학 사무자동화과 부교수