韓國 컴퓨터情報學會 論文誌
第8卷 第1號, 2003. 3.

2003-8-1-1-1

능동규칙의 공유 술어를 이용한 조건 평가

이 기 욱*

Condition Evaluation Using Shared Predicates of Active Rule

Ki-wook Lee*

요 약

능동규칙의 조건부 평가는 사건이 발생할 때마다 수행되기 때문에 조건부 처리 방법에 따라 시스템의 성능에 중요한 영향을 미친다. 그래서 능동 데이터베이스의 성능 향상을 위해 능동규칙의 조건부에 관한 연구가 매우 중요한 요소가 되며, 조건부에서 발생되는 연산 처리시간을 최소화 시켜야만 시스템의 성능을 높일 수 있다.

본 논문에서는 조인연산과 선택연산을 효율적으로 처리할 수 있는 트리 처리 시스템을 제안한다. 실험결과 본 논문에서 제안한 트리 처리 시스템은 기존의 조건부 평가 방법보다 능동규칙의 조건부 처리에 대한 평가비용이 적게 소요됨을 알 수 있다.

Abstract

As the condition evaluation of the active rules is performed whenever an event occurs, the performance of the system is influenced greatly depending on the conditions processing method. Therefore, a study on evaluating the conditions of active rules becomes a very important element for enhancing the performance of active database, and the processing time of the calculation generated from the conditions must be minimized in order to improve the performance.

In this paper, the tree processing system which can effectively process the join and selection operations is proposed. The experimental results show that tree processing system has the lower evaluation cost for the conditions processing of the active rules than the existing methods.

본 연구는 2002년도 동명대학 학술 연구비로 조성되었습니다.
* 동명대학 정보통신계열 부교수

접수일자 : 2003. 1. 3
심사완료 : 2003. 2.20

Ⅰ. 서론

능동규칙(active rule)은 특정한 상황이 되었을 때 정해진 작업을 수행하는 일반적인 능동작업을 표현한다. 능동규칙은 데이터베이스 시스템에 강력한 능동성을 제공할 수 있기 때문에 능동 데이터베이스 시스템의 핵심이 되는 기술이라 할 수 있다[1].

데이터베이스에 능동규칙을 도입하는 중요한 이유는 자료의 변화를 감지하기 위해서 질의 형태인 응용 프로그램을 사용하는 것보다 데이터베이스에 규칙을 내장하는 것이 보다 효율적이기 때문이다.

능동 데이터베이스 시스템(active database system)은 능동규칙 처리 기반을 도입함으로써 특정한 상태를 탐지하여, 사건이 발생하면 규칙의 조건부에서 참·거짓을 판별하여 이 값에 따라 동작이 수행된다. 이와 같은 조건 평가는 사건이 발생할 때마다 수행되기 때문에 조건부를 처리하는 방법에 따라 시스템의 성능에 중요한 영향을 미친다. 그래서 능동 데이터베이스의 성능 향상을 위해 능동규칙의 조건부에 관한 연구가 매우 중요한 요소가 되며, 조건부에서 발생되는 연산 처리시간을 최소화 시켜야만 시스템의 성능을 높일 수 있다[2, 3].

본 논문은 능동규칙의 조건 술어들을 트리로 구축하여 처리비용이 큰 데이터베이스 연산들을 효율적으로 처리하여 능동 데이터베이스 시스템의 성능을 극대화시키는 트리 처리시스템을 제안한다.

Ⅱ. 능동규칙의 조건 평가

능동규칙의 조건 평가에는 점진적 평가[4, 5]와 판별 네트워크[6]를 이용한 평가 방법이 있다. 점진적 평가 방법은 데이터베이스의 변경된 자료만을 차이 릴레이션에 따로 보관하여 분할연산을 이용하여 조건부를 처리한다. 능동규칙의 조건부가 반복적으로 평가되는 특징을 이용하여 이전의 값과 새롭게 변경된 튜플들을 저장한 차이 릴레이션을 대상으로 분할연산을 수행함으로써 효율적으로 조건부를 평가 할 수 있는 장점을 갖는다. 그러나 차이 릴레이션을 이용한 조인연산 처리에서 삭제된 튜플과 삽입된 튜플의 조인을 무시함으로써 전체 조인의 결과에 오류를 발생하는 경우가 있다. 또한 조인과 같이 복잡한 연산의 경우 점진적 평가에서는 조인연산을 위한 특별한 구조를 갖지 못하고 단지 차이 릴레이션과의 분할연산만을 수행하기 때문에 조인연산의 처리 범위가 좁다.

Ariel에 적용된 TREAT는 조인연산에 참여하는 튜플들을 α-메모리에 저장함으로써 릴레이션 전체를 대상으로 조인연산을 수행하는 대신 α-메모리의 튜플로써 조인연산을 효율적으로 처리할 수 있다. 그러나 α-메모리에는 조건 술어에 따라 많은 튜플들을 보관하기 때문에 공간적 측면에서 효율성이 떨어진다. A-TREAT는 TREAT를 확장한 형태로 α-메모리에 저장되는 튜플 때문에 데이터의 중복이 심하게 발생한다. 게다가 능동규칙의 개수에 따라 판별네트워크가 구축되기 때문에 매우 많은 데이터 중복이 발생한다. 이것은 조건부 처리에서 비효율적인 공간처리가 될 수 있다. 이 문제를 해결하기 위해서 가상 α-메모리를 도입하여 저장될 튜플이 많을 경우 실제로 이 메모리에 튜플들을 저장하지 않고 조인연산을 처리할 때 기본 릴레이션에서 해당 튜플들을 가지고 온다. 그러나 이 방법 역시 기본 릴레이션에서 해당 튜플을 가지고 오는 처리시간이 조건부를 효율적으로 처리하는데 장애가 된다. 또한 A-TREAT의 구조는 중간 연산 결과를 보유하지 못하기 때문에 다른 조건 술어의 부분적인 결과에 대해 조건 평가를 할 수 없다. 앞에서 소개한 조건 평가 방법은 특수한 조건 술어들만 효과적으로 처리하기 때문에 보편적인 평가 방법이라고 볼 수 없다.

Ⅲ. 트리 구축

능동규칙은 사건, 조건, 그리고 동작으로 구성된다. 특히 규칙의 조건이 가지는 값에 따라 동작이 수행되기 때문에 능동규칙의 조건처리가 능동데이터베이스 성능에 중요한 역할을 한다. 그래서 능동규칙의 조건을 트리 처리시스템의 차

이트리와 분류트리를 적용하면 조건 평가를 향상시킬 수 있다.

능동규칙 조건의 술어를 평가하기 위하여 Emp 릴레이션에 삽입연산이 수행되었을 때 적용되는 조인연산을 갖는 능동규칙은 다음과 같다.

```
tr1: On Insert EMP
     If Emp.salary > 50000 AND Emp.dno =
       Dept.dno AND Dept.name = 'Sales'
     Then CALL SEND_MAIL(manager,"High
       Salary", Emp.name, Emp.salary)

tr2: On Insert EMP
     If Emp.salary < 20000 AND Emp.dno =
       Dept.dno AND Dept.name = 'Sales'
     Then CALL SEND_MAIL(manager,"Low
       Salary", Emp.name, Emp.salary)

tr3: On Insert EMP
     If Emp.salary > 100000 AND Emp.dno =
       Dept.dno AND Dept.name = 'Sales'
     Then CALL SEND_MAIL(manager,"Too
       High Salary", Emp.name, Emp.salary)
```

EMP 릴레이션에서 삽입연산이 수행되면 일반적인 평가방법에서는 능동규칙 tr1, tr2, 그리고 tr3의 조건술어들을 순차적으로 처리한다. 그런데 이들 규칙의 조건에서 "Emp.dno = Dept.dno AND Dept.name = 'Sales'"가 공통적으로 능동규칙의 조건에 있기 때문에 동일한 조건술어가 반복적으로 평가됨으로써 오버헤드가 발생한다. 이런 문제점들을 해결하기 위하여 능동규칙의 동일한 조건술어들을 공유하기 위해 차이트리를 도입한다. 그림 1은 능동규칙의 조건술어들을 차이트리로 구축한 것이다.

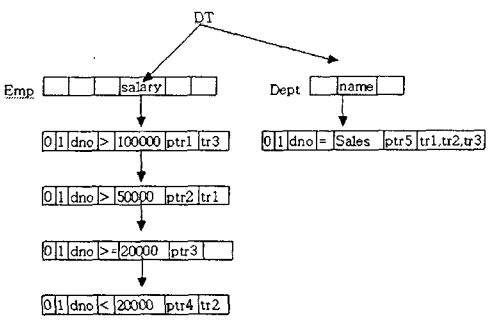


그림 1. 조건술어 차이트리
Fig. 1 Condition predicates delta tree

능동규칙의 조건술어인 "Emp.dno = Dept.dno AND Dept.name = 'Sales'"는 각 규칙의 조건에 공통적으로 사용되는데 이 조건술어를 차이트리로 구성하면 능동규칙의 조건평가를 단 한번만 실시함으로써 규칙을 효율적으로 처리할 수 있다. 결국 여러 능동규칙의 조건에 사용된 조건술어들을 차이트리로 공유함으로써 평가되는 조건술어들을 재 사용한다.

조인연산이 포함된 능동규칙의 조건술어는 비교적 많은 처리비용을 요구한다. 그래서 동일 조인연산을 반복적으로 처리하면 시스템 성능저하의 결정적인 요인이 될 수 있다. 분류트리는 조인연산으로 인한 오버헤드를 해결할 수 있다. 그림 2는 tr1, tr2, 그리고 tr3에 대한 분류트리이다.

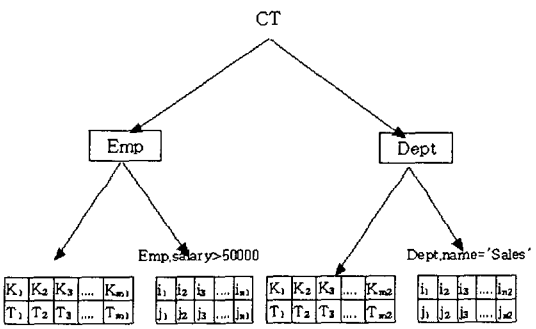


그림 2. 조건술어 분류트리
Fig. 2 Condition predicates classification tree

조인연산이 포함된 조건술어가 많을수록 능동규칙의 조건 평가는 많은 처리시간을 요구한다. 차이트리와 분류트리는 복잡한 조건술어로 구성된 능동규칙의 처리를 효과적으로 수행한다. 2개 이상의 조인연산을 갖는 능동규칙은 다음과 같다.

```
tr4 : On Insert EMP
      If Emp.salary > 50000 AND Emp.dno =
        Dept.dno AND Dept.name = 'Sales'
      Then CALL SEND_MAIL(manager,"High
        Salary", Emp.name, Emp.salary)

tr5 : On Insert EMP
      If Emp.salary > 50000 AND Emp.dno =
        Dept.dno AND Dept.name = 'Sales'
       AND Emp.jno=Job.no AND Job.title='Clerk'
      Then action5
```

tr6 : On Insert EMP
If Emp.salary 〉 50000 AND Emp.dno =
Dept.dno AND Dept.name = 'Sales'
AND Emp.jno=Job.no AND Job.title='Agent'
Then action6

능동규칙들을 효과적으로 처리하기 위해서는 차이트리와 분류트리로 구축해야 한다. 능동규칙에서 사용되는 릴레이션은 Emp, Dept, 그리고 Job이기 때문에 구축되는 차이트리에는 이들 릴레이션들이 반영되어진다. 그림 3은 능동규칙들에 대한 차이트리이다.

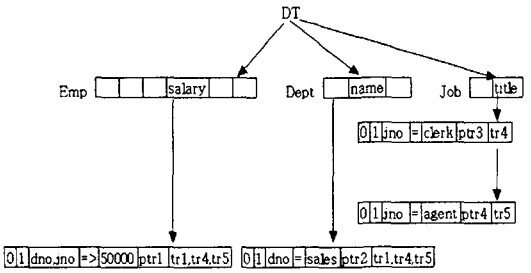


그림 3. 차이트리
Fig. 3 Delta tree

그림 3의 차이트리로 구축된 능동규칙들은 2개 이상의 조인연산으로 구성되었기 때문에 분류트리의 역할이 중요하다. 조인연산에 참여하는 릴레이션들은 Emp, Dept, 그리고 Job이고 이들 릴레이션들과 관련되는 애트리뷰트들을 분류트리에 사용한다. 앞에서 제시한 능동규칙의 조건술어를 분류트리로 구축한 것이 그림 4이다.

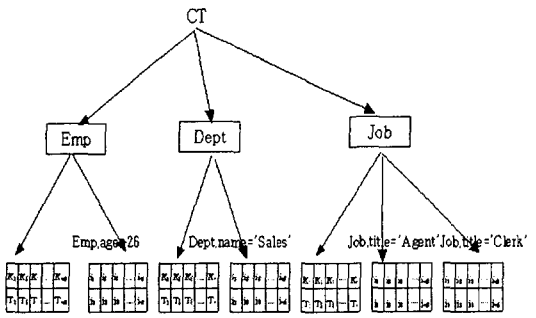


그림 4. 분류트리
Fig. 4 Classification tree

만약 tr1에서 tr6까지 6개의 능동규칙이 같이 사용되면 이미 구축된 분류트리에 Job 릴레이션 중에서 조인연산에 사용되는 title 애트리뷰트가 'Agent'와 'Clerk'을 추가시킨다. 조건술어들의 공유가 많을 경우 차이트리와 분류트리를 사용하면 능동규칙의 조건처리를 효율적으로 수행할 수 있다.

Ⅳ. 실험 및 평가

본 논문에서 제안한 차이트리와 분류트리를 구축하는 시스템을 구현하여 능동데이터베이스의 조건술어를 효과적으로 평가하는 것을 실험을 통하여 알아본다. 성능평가는 선택연산으로 구성된 조건술어 평가와 조인연산을 갖는 조건술어는 튜플 수와 밀접한 관련을 갖고 있기 때문에 튜플 수의 변화에 따른 처리능력을 측정한다. 실험은 OS를 Windows NT로 장착한 128MB 주기억장치를 가진 퍼스널 워크스테이션 상에서 Visual C++ 6.0으로 구현하였다.

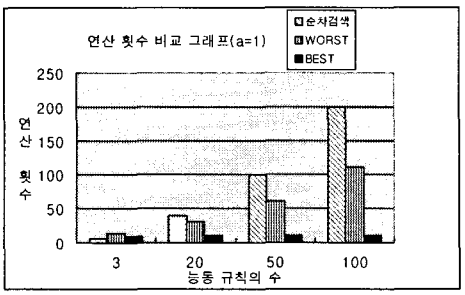
4.1 선택 술어

선택연산을 수행하는 조건술어의 평가는 기존의 데이터베이스 자료와 관련이 없고, 새로이 삽입되는 자료가 능동규칙의 조건술어를 만족하는 것을 결정한다. 즉 선택연산의 조건술어에 대한 처리는 능동규칙의 수에 의해 결정된다. 본 실험은 능동규칙의 수와 능동규칙의 조건이 단일 조건술어가 중복되는 경우와 중복되지 않는 경우를 대상으로 수행하였다. 그리고, 성능비교는 비교연산의 횟수를 이용한다. 실험에 사용되는 능동규칙의 조건들은 다음과 같다.

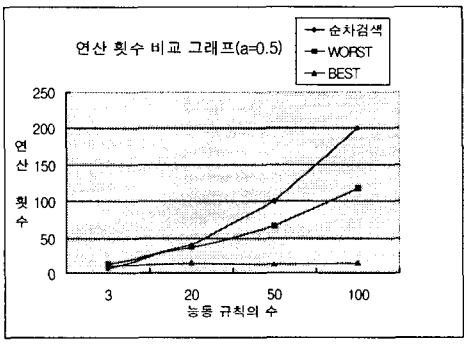
Emp.salary 〉= 11000 and Emp.eno 〉 2500
Emp.salary 〉= 10000 and Emp.salary 〈= 25000
Emp.age 〉 26 and Emp.salary 〉= 11000

조건술어를 갖는 능동규칙은 Emp 릴레이션에 대한 사건과 관련이 있고, 선택연산만을 포함하는 조건술어를 갖는다. 그리고 각 능동규칙은 두 개의 단일 조건술어를 갖고 있다. 순차검색과 트리시스템을 이용하여 조건술어를

평가한 결과는 그림 5이다.



(a) 술어공유 비율(a=1)



(b) 술어공유 비율(a=0.5)

그림 5. 선택 조건술어 처리 그래프
Fig. 5 Selection condition predicate processing graph

그림 5에서 a는 능동규칙들의 조건술어가 공유되는 비율이다. 예를 들어 a=1인 경우는 3개의 능동규칙의 단일 조건술어가 나머지 능동규칙의 조건술어에서도 사용되는 것을 의미한다. a=0.5는 능동규칙의 조건술어가 두 번씩 나타나는 경우인데 순차검색에서 평가되어야 하는 조건술어의 수는 2 x N개지만 서로 다른 조건술어의 수는 N개인 것을 의미한다. 만약 a=0인 경우, 능동규칙의 조건술어가 서로 다른 경우이다. 그러나 능동 데이터베이스시스템에서 사용되는 능동규칙들을 분석해 보면, 모든 조건술어가 서로 다를 경우는 없다. 따라서, 실험에서 a=0인 경우는 제외한다. N은 능동규칙의 수를 의미한다.

실험에서 a=1일 때 WORST는 삽입되는 자료가 차이트리의 각 애트리뷰트 노드 중 최하위 레벨에만 저장될 때 비교 횟수를 산출한 것이고, BEST는 능동규칙의 조건을 만족하지 않는 자료가 삽입될 때의 비교 횟수이다. a=0.5의 경우는 차이트리의 노드가 만족하는 능동규칙의 수가 동일하다는 가정 하에서 WORST와 BEST의 비교 연산 횟수를 산출하였다. 트리처리시스템은 전처리에서 공유되는 조건술어를 단일화시키기 때문에 능동규칙의 조건술어에 대한 공유비율이 높을수록 중복평가를 배제할 수 있어 규칙의 조건술어 처리성능이 좋다.

4.2 조인 술어

조인연산을 수행하는 조건술어의 평가는 전체 릴레이션을 대상으로 처리되기 때문에 기본 데이터베이스 릴레이션의 튜플 수가 많을수록 평가시간이 증가한다. 본 실험은 3개의 릴레이션 Emp, Dept, 그리고 Job이 조인연산에 참여하는 능동규칙의 조건술어를 대상으로 하였다. 그리고, 성능비교는 비교연산의 횟수를 이용한다. 실험에 사용되는 능동규칙의 조건은 다음과 같다.

Emp.salary>50000 AND Emp.dno=Dept.dno
AND Dept.name='sales' AND Emp.jno=Job.jno AND Job.title='clerk'

Emp 릴레이션에 삽입되는 튜플의 salary가 50000을 초과하면 dno 필드를 이용하여 Dept 릴레이션과 조인연산을 수행한다. 조인연산의 결과에서 튜플의 name이 'sales'이면 jno 필드를 이용하여 Job 릴레이션 전체와 다시 조인연산을 수행한다. 조인연산은 전체 릴레이션을 대상으로 작업을 수행하기 때문에 릴레이션의 튜플 수가 많으면 조인연산에 많은 시간이 필요하여, 조건술어를 평가하는 시간이 증가한다. 순차검색과 트리시스템을 이용하여 조인 조건술어를 평가한 결과는 그림 6이다.

a는 전체 릴레이션 중에서 조인대상이 되는 자료의 빈도수이다. a의 값이 클수록 트리처리시스템의 분류트리에 구축되어야 하는 자료가 증가한다. 실험에서 사용되는 조건술어 Dept.name='sales' 또는 Job.title='clerk'에 대해 조인대상 자료의 수가 전체 릴레이션의 1%, 10%, 그리고 20%를 차지하는 경우에 따라 a의 값을 다르게 정했다. a의 값이 작을수록 조인연산에 관련되는 자료가 적기 때문에 연산시간이 감소하고, 순차검색은 a의 값에 상관없이 연산시간이 증가한다. 본 논문에서 제안한 트리처리시스템은 전체 릴레이션에 대해서 조인연산을 처리하는 대신 연산 대상만을 미리 분류하여 조인연산의 수를 감소시킨다.

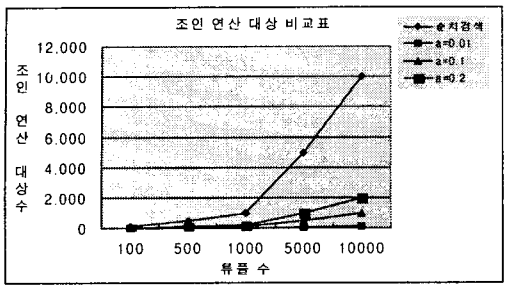


그림 6. 조인 조건술어 처리 그래프
Fig. 6 Join condition predicate processing graph

실험 결과를 분석하면 본 논문에서 제안한 트리처리시스템이 순차검색 방법에 비해 선택연산과 조인연산에 대한 처리능력이 우수함을 알 수 있다.

V. 결 론

본 연구에서는 능동데이터베이스시스템의 성능을 향상시키기 위해 능동규칙의 조건부를 효율적으로 처리하는 트리처리시스템을 제안하였다. 차이트리와 분류트리로 구성된 트리처리시스템은 일반적인 조건술어 처리 방법보다 처리 성능이 우수하다는 것을 실험 결과로 알 수 있다. 따라서 제안된 트리처리시스템은 선택연산 위주로 연구되어지는 판별네트워크 시스템과 극히 제한적으로 능동규칙의 조건부를 처리하는 점진적 평가 시스템보다 능동데이터베이스의 조건술어의 모든 처리 환경을 고려한 시스템이다.

본 연구의 향후 과제는 정적구조로 트리를 구성하고 있는 노드들을 동적인 구조 형태로 구축하여 능동규칙의 조건 처리를 향상시킬 필요가 있다.

References

[1] Zanio lo. C., Ceri. S., Faloutsos. S., Snodgrass. R., Subrahmanian. V. and Zicari. R. Advanced Database System. Morgan Kaufmann Publishers, Inc., pp. 7-36, 1997.

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

[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, 1993.

[4] 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.

[5] 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.

[6] 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. June 1992.

저 자 소 개

이 기 욱
1985년: 계명대학교 전자계산학과 (공학사)
1987년: 동국대학교대학원 전자계산학과(공학석사)
2001년: 계명대학교대학원 컴퓨터공학과 (공학박사)
1991년 ~ 현재: 동명대학 정보통신계열 부교수