능동 데이터베이스에서 병렬처리를 고려한 규칙실행전략

정회원 오 성 균*, 이 재 호**, 임 해 철***

Strategies for Rule Execution Considering Parallel Processing in Active Database

Sung Kyun Oh*, Jae Ho Lee**, Hae Chull Lim*** *Regular Members*

요 약

능동 데이터베이스에서 규칙을 병렬실행 시킬 경우에 기존의 병렬 제어구조는 단순하여 현실세계의 다양한 형태의 시멘틱을 충분하게 반영하지 못하고 있는 실정이다. 본 논문에서는 규칙실행에 필요한 구성요소와 실행형태를 분석하고, 실세계 환경의 복잡한 상황을 정확하게 표현할 수 있도록 병렬규칙실행 제어구조를 확장하였다. 이러한 제어구조를 기반으로 실세계 업무상황에서 이용될 수 있는 그래프-기반 규칙실행 그래프를 제시하였다. 그리고 트리거 규칙이 병렬로 실행될 경우, 내포된 규칙의 수와 기타 정보에 따라 트리거그룹을 생성하여 우선순위를 결정하는 정책과 알고리즘을 제안하였다. 또한 유한개의 프로세서를 다수의 트리거 규칙에 할당할 경우, 트리거그룹 우선순위 정책을 기반으로 한 프로세서 할당 및 해제정책을 제시하였다. 논문에서 제안한 규칙실행 전략과 기존의 실행전략에 대하여 처리시간 기준으로 성능을 분석하였다.

ABSTRACT

Various real-world semantics have not been fully provided due to the simplicity of conventional parallel control construct in parallel rule execution model of Active Database. This paper extends to represent correctly the complex situations in real world environment and to analyse the necessary constituent and the form of rule execution. Based on these control constructs, we present graph-based rule execution graph considering real world situation. And, we propose an algorithm and the policy which create priority trigger group according to the number of nested rule and other information for the parallel execution of trigger rule. In case that multi-processors are assigned to the large number of trigger rules, the policy is suggested to assign and to release processor in multi-processors environment based on trigger group priority policy. Performance analysis is performed for the proposed rule execution strategy compared with conventional execution strategy in terms of the processing time.

Ⅰ. 서 론

대부분의 능동 데이터베이스 응용에서는 트랜잭션을 미리 지정하고 설계된 트랜잭션에 따라 반복적으로 수행하게 된다. 이러한 트랜잭션들은 많은 프리미티브 규칙(primitive rule)들로 구성되어 복잡하게 수행한다. 따라서 트랜잭션을 구성하는 많은 프리미티브 규칙들을 어떻게 효율적으로 트리거(trigger)시킬 것인가가 중요한 문제로 남는다. 능동 데이터베이스에서 트랜잭션의 성능 향상에 관한 연구로 규칙실행 분야를 들 수 있다. 기존의 규칙실행 모델은 주로 중앙집중 또는 순차실행 형태로 이용되거나 구현되었지만 최근 데이터베이스의 경향은 분산 환경과 클라이언트-서버 환경으로 전환됨에 따

* 서일대학교 전자계산과 ** 인천교육대학교 컴퓨터공학과
*** 홍익대학교 컴퓨터공학과(Skoh@CS.hongik.ac.kr)
논문번호:99051-0218, 접수일자: 1999년 2년 18일
※ 본 연구는 '98 한국과학재단 특정기초연구의 지원에 의하여 연구되었음(과제번호 : 98-0102-09-01-3)

라 병렬규칙실행 모델에 대한 트랜잭션 수행시의 성능 향상에 초점이 맞추어지고 있다. 따라서 기존의 병렬규칙실행 모델에서 제어구조를 이용한 실행 형태는 단순하고 간단하여, 다양하고 복잡한 현실 세계의 시멘틱을 충분히 반영하지 못하고 있는 실정이다.

따라서 본 논문에서는 트랜잭션 성능 향상의 일환으로 프리미티브 규칙실행에 필요한 구성요소와 실행형태를 분석하고, 실세계 환경을 정확하고 다양하게 표현할 수 있도록 병렬규칙실행 제어구조를 확장하여 제안하였다. 이러한 제어구조를 기반으로 실세계 업무상황에서 이용되는 규칙응용 시나리오와 그래프기반 규칙실행 그래프를 생성하여 제시하였다. 병렬규칙실행 모델에서 트리거규칙이 병렬로 실행될 경우, 내포된 규칙의 수와 기타 정보에 따라 트리거그룹을 생성하여 우선순위를 결정하는 정책과 알고리즘을 제안하였다. 또한 멀티프로세서를 다수의 트리거규칙에 할당할 경우, 트리거그룹 우선순위 정책을 기반으로 한 프로세서 할당 및 해제정책을 제안하였다. 마지막으로, 본 논문에서 제안한 규칙실행 전략과 기존 실행전략에 대하여 처리시간을 기준으로 성능 분석을 시행하였다.

Ⅱ. 능동 DB에서 규칙구성과 실행형태

본 절에서는 능동 데이터베이스에서의 규칙실행에 필요한 규칙구성, 실행형태 그리고 규칙제어흐름에 대해 분석한다.

2.1 규칙구성

능동 데이터베이스에서의 규칙은 이벤트, 조건절, 액션절 부분으로 구성되며, 이벤트의 종류는 원시이벤트와 복합이벤트로 구분한다. 원시이벤트는 데이터를 조작하는 데이터조작 이벤트(insert, update, retrieval), 트랜잭션 수행 시 완료나 철회를 담당하는 트랜잭션 이벤트(commit, abort), 시간 이벤트(absolute time, relative time, period time), 객체지향 개념의 메소드(method) 이벤트 등이 있고, 복합이벤트에는 원시이벤트에 복합오퍼레이터(disjunction:OR, conjuntion:AND, sequence:SEQ)를 추가한 이벤트가 있다.

조건절 부분은 액션절을 수행하기 위한 조건으로, 만족하면 액션절을 수행하고, 그렇지 않으면 수행하지 않는다. 이러한 조건절은 데이터베이스 상에서의 프리디키트(predicate)나 SQL의 where절에 해당된다고 할 수 있다. 액션절 부분은 재수행 행위를 명시하는 부분으로 데이터베이스 수정과 검색, 트랜잭션에 대한 연산, 프로시저호출(procedure call), 규칙연산자(rule operator)를 명시하는 부분이다[1, 2, 11, 12].

2.2 규칙실행 형태와 제어흐름

규칙실행 형태는 트랜잭션 수행 중 이벤트가 발생할 경우 하나의 규칙에 의해서만 실행되는 단일규칙실행, 한 규칙에서 여러 개의 규칙을 트리거 하는 다중규칙실행, 규칙액션 부분에서 또 다른 규칙이나 이벤트를 실행하는 내포규칙실행 등으로 구분한다. 내포규칙실행 형태에서 제어흐름은 한 규칙내부에서 이벤트탐색, 조건평가, 액션실행 순서로 제어흐름이 행하여지는 규칙-내부(intra-rule) 제어흐름, 한 규칙에서 또 다른 규칙으로 제어이동이 행하여지는 규칙-간(inter-rule) 제어흐름, 그리고 규칙실행 시 규칙들간의 상호의존도에 따라 발생되는 규칙상호작용(rule interaction) 등으로 구분한다.

Ⅲ. 규칙실행 모델과 제어구조 분석

규칙실행 모델은 한 트랜잭션에서 트리거 규칙을 순차적으로 처리하는 순차규칙실행 모델과 멀티프로세서를 이용하여 병렬로 여러 개의 규칙을 동시에 실행시키는 병렬규칙실행 모델로 구분할 수 있다. 본 절에서는 규칙실행 모델과 규칙실행 제어구조를 분석한다.

3.1 규칙실행 모델의 분석

순차규칙실행 모델은 평면 트랜잭션 모델 또는 그래프기반 트랜잭션 모델에서 트리거된 규칙을 순차적으로 실행하여 직렬성을 보장받으나 규칙실행 시간이 느리다는 단점이 있다. 그리고 병렬규칙실행 모델은 유한개의 프로세서를 이용하여 여러 개의 트리거 규칙을 동시에 처리할 수 있는 모델이다. 이러한 방식은 동기접근 방식과 비동기접근 방식으로 구분된다. 동기접근 방식은 레벨별로 그룹화(수평적 그룹화)하여 그룹내의 모든 규칙이나 타스크들이 완료되어야만 다음 레벨의 규칙이나 타스크로의 실행이 가능한 방식이고 반면에 비동기접근 방식은 레벨별로 그룹화한 규칙이나 타스크들 모두가 실행이 완료되지 않아도 다음 레벨의 규칙이나 타스크를 실행할 수 있는 방식이다.

3.2 기존의 규칙실행 제어구조

규칙실행 모델에서 이용된 기존의 제어구조는 순차 구조(precede construct), 순차-셀 구조(precede -set construct), 동기 구조(sync-at construct)로 구분할 수 있다. 순차 구조는 한 규칙실행 후 다음 규칙을 순차적으로 실행시키는 제어구조이며, 순차-셀 구조는 병렬구조라고도 하며, 어느 한 규칙을 완료한 후 레벨별로 그룹화 한 모든 규칙을 병렬처리를 수행하는 제어구조이다. 동기 구조는 레벨별로 그룹화 한 모든 규칙실행을 완료한 후, 다음 레벨에 있는 하나의 규칙을 트리거하는 구조이다[7, 8]. 그림 1의 (a), (b), (c) 내의 제어구조 표현은 순차 구조를 "S"(serial)로, 순차-셀 구조를 "P"(parallel)로, 동기 구조를 "Y"(sync-at)로 표기한 것이다. 이러한 기존 제어구조의 특징은 시멘틱이 단순하게 정의되어 있어, 실세계 환경에서 복잡하고 다양한 상황을 정확하게 표현할 수 없는 단점이 있다.

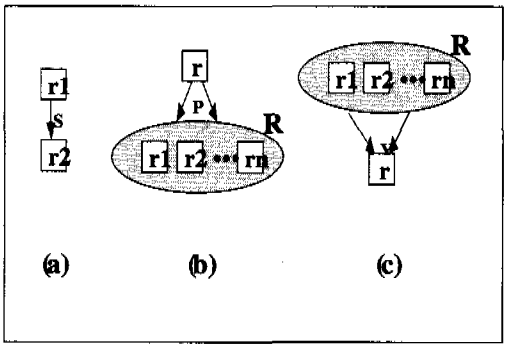

그림 1. 기존의 병렬규칙실행 모델에서 이용한 제어구조

Ⅳ. 기존 병렬규칙실행 모델에서의 개선 방향

서론에서도 언급하였듯이 최근의 능동 데이터베이스 규칙실행 모델의 경향은 중앙집중 환경의 순차규칙실행 모델에서 분산 환경과 클라이언트-서버 환경에서 이용할 수 있는 병렬규칙실행 모델로 전환되고 있는 추세이다. 하지만 기존의 병렬규칙실행 모델로는 복잡하고 다양한 형태의 내포다중규칙을 실행하는 경우, 여러 가지 측면에서 병렬규칙실행 상의 성능 향상을 기대할 수 없다. 이러한 면을 고려하여 기존의 병렬규칙실행 모델에서 개선시켜야할 방향을 제시하면 다음 표 1과 같다.

표 1. 기존 규칙실행 모델에서의 개선 방향

실행모델 형태	문제점	개선방향
순차규칙 실행모델	· 규칙 실행 시 성능 저하	· 병렬 규칙 실행 시 제어 구조를 시멘틱이 풍부하고 다양한 형태로 확장 · 병렬 규칙 실행 그래프 생성 · 규칙 실행 그래프를 기반으로 한 병렬규칙 실행 구문 명세 설계 · 병렬 규칙 실행시 프로세서 우선 순위할당과 해제 전략 제시 · 병렬 규칙 실행 전략 제시 · 기존 모델과 제안 모델과의 성능비교 평가
병렬규칙 실행모델 (1) 동기방식[7] (2) 비동기 방식[8]	· 기존의 제어 구조가 단순하고 간단함. 즉, 다양한 형태의 규칙 실행 관계성 부족 · 병렬 프로세서 우선 순위 할당 전략과 알고리즘 부재 · 병렬 규칙 실행 모델에서의 성능향상을 위한 병렬 규칙 실행 전략 부재	

Ⅴ. 병렬처리를 고려한 규칙실행전략

본 절에서는 멀티프로세서 환경 하에서 병렬트리거 규칙을 실행하고자 할 경우, 규칙처리에 필요한 실행환경을 정의하고 기존 제어구조를 확장하여 정의한 제어구조와 이의 제어구조를 바탕으로 한 시나리오를 제시한다. 또한 멀티프로세서 우선순위 할당 정책의 일환인 트리거그룹 생성전략과 멀티프로세서 우선순위 할당 및 해제전략을 수립한다.

5.1 병렬규칙처리를 위한 실행환경 정의

병렬규칙 처리를 고려할 경우, 유한개의 프로세서($n \geq 2$)가 실행된다고 가정하고, 내포형태의 규칙실행은 한 트랜잭션 내에서 규칙-내부 제어흐름만을 고려한다고 가정한다. 이러한 내포규칙실행 형태의 제어구조 관계성은 기존의 평면 트랜잭션모델이나 내포 트랜잭션모델에서는 표현되기 어렵기 때문에 본 논문에서는 복합구조 관계성을 표현하고, 규칙 사이의 다양한 제어구조의 관계성을 가질 수 있는 그래프-기반 트랜잭션모델을 이용한다. 이는 방향성을 갖는 비순환 그래프(DAG) 형태로 표현된 트랜잭션 모델이다. 능동규칙 병렬처리실행 모듈은 규칙관리자, 병렬규칙실행 관리자, 이벤트탐색 관리자, 충돌규칙해결 모듈 등으로 구성된다. 이 실행모듈 중의 병렬규칙실행 관리자는 병렬처리를 고려한 규칙에 대한 메시지 패싱(message passing) 관리와 규

칙을 병렬처리가 가능하도록 담당하는 기능, 중요도에 따른 트리거 규칙에 우선순위를 부여하는 프로세서 할당 및 해제관리 기능 등을 코디네이터 프로세서(coordinator processor : CP)가 관리하도록 하는 역할을 담당한다.

5.2 확장한 제어구조 정의

병렬규칙실행 모델에서는 능동 규칙수행 시 트리거 과정이 복잡하기 때문에 기존의 단순한 제어구조만을 가지고는 실세계의 다양한 상황을 정확하게 표현할 수 없다. 본 절에서는 규칙들 간의 다양한 관계성을 표현할 수 있는 제어구조를 정의하였다. 이에 따른 규칙이나 타스크는 원기호가 이용되고, 방향을 가지는 화살표 기호는 규칙이나 타스크에서 다음 레벨의 규칙을 트리거하는 역할을 가진다. 그리고 분기제어 레이블이 사용되고, 트리거 규칙이 병렬로 실행될 경우, 중요도에 따라 우선순위를 부여하는 상대 우선순위 값이 이용된다. 규칙실행에 대한 확장된 제어구조 정보는 다음과 같이 정의한다.

1) "S"인 경우

순차(serial)실행의 의미를 가지며, 한 규칙실행 후 다음 레벨 규칙을 순차적으로 실행하는 제어정보이다.

2) "P"인 경우

완전병렬(complete parallel)실행의 의미를 가지며, 멀티프로세서를 이용한 병렬실행으로 트리거그룹 생성정책을 기반으로 프로세서에 우선순위를 할당하는 제어정보이다.

3) "Os"인 경우

순서적(ordered sequence)실행의 의미를 가지며, 상대 우선순위 값에 의거하여 규칙을 트리거하는 제어정보이다.

4) "Ex"인 경우

배타적(exclusive)실행의 의미를 가지며, 여러 개의 트리거 규칙 중 배타적으로 하나의 규칙만을 트리거하는 제어정보이다.

5) "Op"인 경우

선택(optional) 실행의 의미를 가지며, 트리거 규칙을 선택적으로 트리거하는 제어정보이다. 즉, "Op-P"는 완전병렬, "Op-Os"은 순서적 실행, "Op-Ex"는 배타적 실행 중의 하나를 선택하여 실행하는 정보이다.

6) "Ya"인 경우

전체동기처리(sync-at-all)의 의미를 가지며, 트리거된 규칙들이 모두 도착되면 다음 레벨의 규칙을 트리거하는 제어정보이면서, 프로세서 우선순위 및 도착순서를 적용하는 제어정보이다.

7) "Yp"인 경우

부분동기처리(sync-at-partial)의 의미를 가지며, 트리거된 규칙들 중 일부만 도착되어도 다음 레벨의 규칙을 트리거하는 제어정보이면서, 프로세서 우선순위 및 도착순서를 적용하는 제어정보이다.

5.3 규칙응용 시나리오

확장된 제어구조 정보를 기반으로 실세계 업무를 적용한 규칙응용 시나리오를 그림 2에 제시하였다. 이 시나리오는 증권 업무에서 신규 투자자가 어떤 주식종목에 대한 매입상황 일부를 표현한 것이다. 예를 들면 어떤 주식종목이 100원 이상 상승하거나 100원 미만으로 하락할 경우, 고객의 현금 또는 신용 계좌에서 주식매도 또는 매수를 신속·정확하게 요청하도록 적용한 예이다.

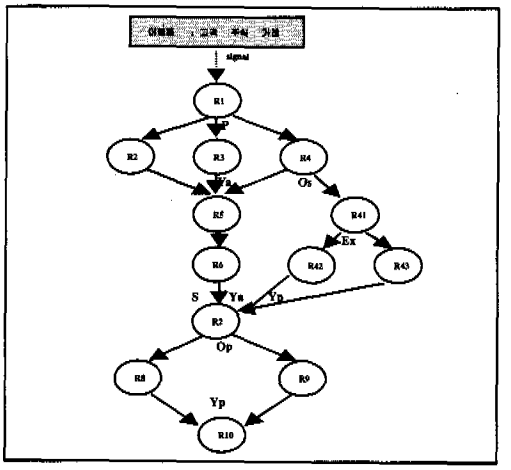

그림 3. 규칙응용 시나리오를 이용한 규칙실행 그래프

이러한 시나리오는 어떤 주식종목에 대하여 신규 투자자가 매입하고, 주식매입에 따른 분석 결과가 양호하다고 가정하였다. 또한 주식 매입시기가 적절하고, 재무구조 리스트와 매입지분을 분리처리 하여 내포된 규칙실행으로 처리된다고 가정하였다. 이러

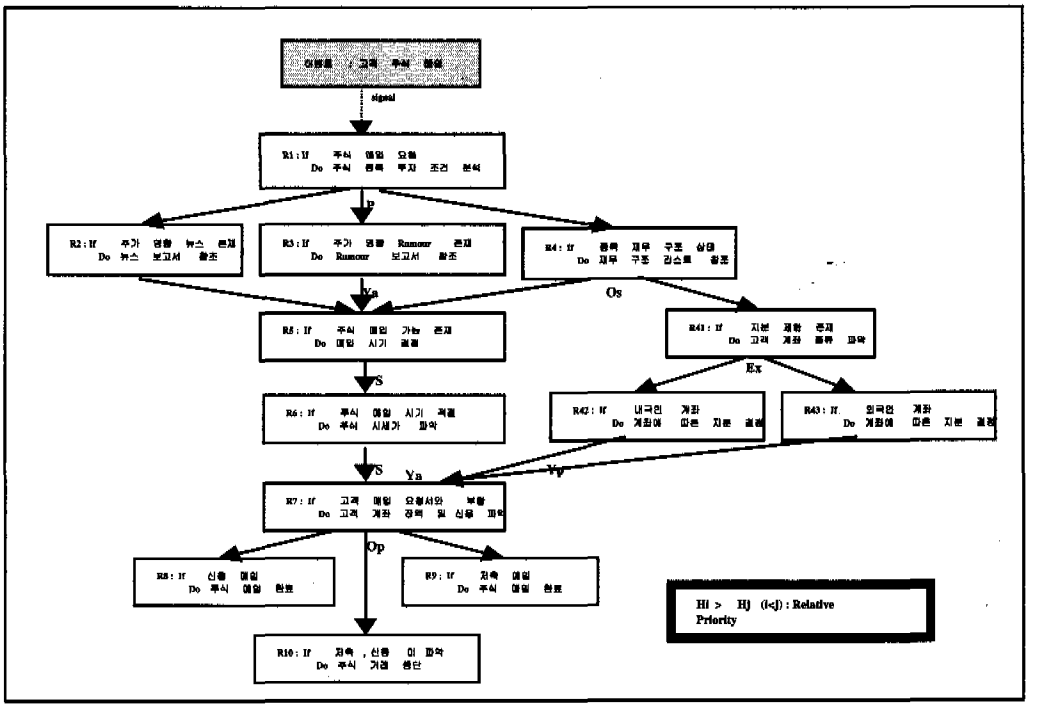

그림 2. 주식 매입을 위한 규칙응용시나리오

한 상태 하에서 이벤트가 발생하면 규칙실행은 주식 종목 분석(R2, R3, R4 규칙)을 통하여 병렬규칙 실행을 수행하며, 고객계좌의 잔고 상황(R7)에 따라 선택실행을 수행하는 주식매입 상황이다.

그림 3의 규칙실행그래프는 규칙응용시나리오가 실세계의 상황과 정확하게 일치하는지 피드백 시키면서 실세계를 포괄적으로 표현하기 위한 간단한 표현법을 나타낸 것이다.

5.4 멀티프로세서 할당을 위한 트리거그룹 생성정책

본 절에서는 병렬규칙실행 모델에서의 규칙실행 그래프를 기반으로 트리거 규칙을 병렬처리 하는 경우, 트리거할 규칙 수가 프로세서 수보다 많이 존재한다고 가정하면 프로세서 우선순위 할당정책이 필요하다. 이는 내포규칙 수가 많은 트리거 규칙이나 장시간 실행을 하는 트리거 규칙인 경우에는 유한개의 프로세서를 우선순위로 할당하여 트랜잭션 실행 성능을 향상할 수 있는 정책을 제시한다.

5.4.1 트리거그룹의 필요성

트리거그룹을 생성하는 이유는 병렬로 실행되는 다수의 내포 트리거규칙에 유한개의 프로세서를 할당할 경우, 어느 트리거규칙에 먼저 프로세서를 할당할 것인가 또는 어느 트리거규칙을 먼저 수행시킬 것인가에 대한 정책이 필요하다. 예를 들면, 프로세서는 2개인데 병렬로 실행될 트리거규칙이 3개일 경우, 어느 규칙에 우선적으로 프로세서를 할당할 것인가라는 문제이다. 이러한 프로세서 우선순위 할당정책의 일환으로 트리거그룹 생성전략이 필요하다. 이러한 방안은 트리거규칙에 따른 내포트리거규칙의 수 또는 내포 트리거규칙이 갖는 제어정보, 연산자 그리고 데이터 수에 따라 전반적인 성능에 영향을 미칠 수 있다. 이러한 성능요소에 대한 규칙수와 활동 실행시간을 추정하여 수행시간이 가장 길 것으로 예상되는 트리거그룹에 높은 우선순위의 프로세서를 할당한다.

5.4.2 트리거그룹 생성과정

트리거그룹을 생성하는 경우는 부모-타스크의 제어정보가 병렬로 실행되는 제어정보("P", "Os", "Op-P", "Op-Os")일 때 트리거규칙 수에 따라 트리거그룹 수가 생성되고, 동기제어 정보("Ya", "Yp")일 때 트리거그룹 범위를 종료한다. 그림 4에서 R1 규칙실행이 완료된 후 부모-타스크의 정보가 "P"이므로 병렬로 처리할 트리거 수는 R2, R3, R4

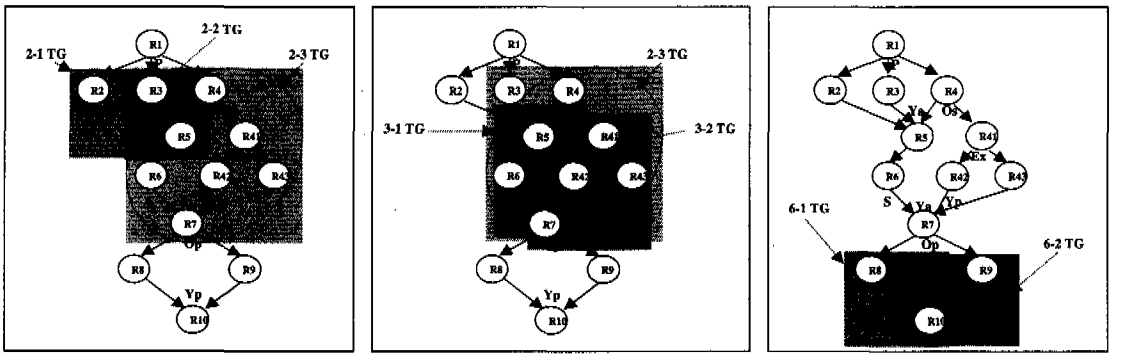

그림 5. 리거그룹 생성 시 단계별 그룹화의 예

규칙이므로 트리거그룹 수는 3개가 형성되는데 그에 해당되는 트리거그룹의 이름은 R2. R3, R4 트리거 그룹으로 명명된다.

예를 들어 R2 트리거그룹 생성은 R2 규칙실행 완료 후 다음 레벨로 트리거 할 규칙의 제어정보가 "Ya"에 해당되므로 R2의 트리거그룹이 생성된다. 이 때 R2의 트리거그룹 생성 시 기타정보(규칙 제어정보의 수, 연산자 수, 데이터 수)가 저장된다. R3의 트리거그룹 생성 과정도 R2의 트리거그룹과 마찬가지 방식으로 생성된다. 그리고 R4의 트리거그룹을 생성할 때에는 R4의 타스크 제어정보가 "Os"에 해당되므로 또 다른 내포 트리거그룹(트리거그룹 수 : 2)을 생성한다. 즉, R4 트리거그룹 내부에 또 다른 내포 트리거그룹 R5와 R41의 트리거그룹이 생성된다. 이 때 R5 트리거그룹은 단계번호 5에서 "Ya"를 만나게 될 때 비로소 R5 트리거그룹이 생성된다.

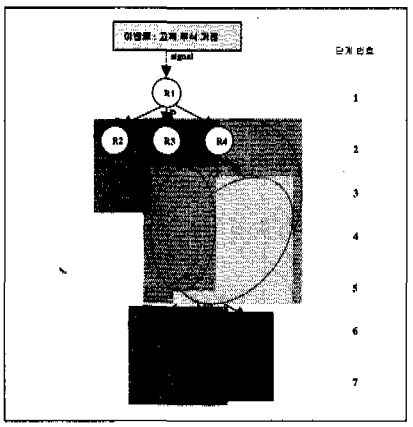

그림 4. 트리거그룹생성 예

또 다른 트리거그룹인 R41인 경우는 타스크 제어정보가 배타적인 트리거규칙에 해당되기 때문에 또 다른 2개 이상의 내포 트리거그룹은 생성되지 않고 하나의 트리거그룹만을 생성한다.

이 때 R4의 트리거그룹 정보에 R4 트리거그룹 정보와 R41 트리거그룹 정보를 같이 누적하여 저장시킨다. 나머지 트리거그룹 생성은 위와 같은 방식을 적용한다. 그림 5은 트리거그룹 생성 과정을 1단계(a)에서부터 3단계(c)까지를 단계별로 생성한 예를 표현한 것이다.

5.4.3 트리거그룹 생성 알고리즘

트리거그룹 생성 알고리즘에서는 한 규칙실행 후 다음 레벨 트리거규칙의 수가 3을 초과하지 않고, 레벨단계는 9 단계 이내로 제한하였다.

그림 6에 기술한 트리거그룹 생성 알고리즘의 동작개념은 트리거그룹내의 트리거규칙 수를 계산할 배열을 단계배열(level-array)과 내포배열(nested-array)로 구분하여 이용한다. 단계배열은 단계별로 실행되는 트리거규칙 수가 저장되는 영역이고, 내포배열은 트리거그룹내의 또 다른 트리거그룹이 존재할 때 이용되는 배열이다.

또한 배열생성 시, 2개의 스택(stack-1, stack-2) 정보를 사용하여 레벨포인터(Lp)와 트리거포인터(Tp)의 정보를 같이 저장한다. stack-1에서는 배열이 종료되면 Lp정보와 Tp정보는 삭제되지만 stack-2에서는 삭제하지 않고 그대로 Lp, Tp정보를 보관해 둔다. 이 개념은 트리거그룹내의 또 다른 내포 트리거그룹을 생성할 경우, stack-2의 정보를 이용하여 상위 트리거그룹에 대한 트리거규칙 수를 계산하기 위한 목적으로 이용된다. 모든 트리거그룹 생성을

종료한 경우, 내포배열에 있는 정보를 단계배열에 저장하고 나머지 배열은 제거시킨다.

5.5 트리거그룹에 따른 프로세서 우선순위 할당 및 해제정책

본 절에서는 생성된 트리거그룹 정보를 기반으로 병렬규칙처리 시 우선순위 부여 규칙과 제어정보에 따른 멀티프로세서 할당 및 해제정책을 제시한다.

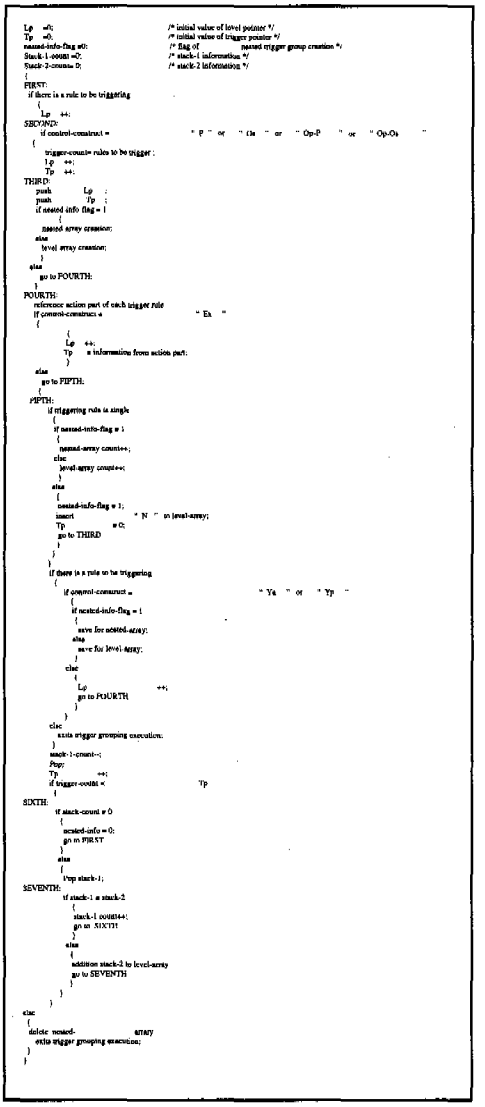

그림 6. 트리거그룹 생성 알고리즘

5.5.1 우선순위 부여 규칙

트리거그룹이 생성되면 그에 따른 우선순위를 레벨단계별로 트리거규칙에 부여하고, 부여한 우선 순위를 기준으로 프로세서를 할당한다. 트리거그룹생성 시, 기타정보가 저장됨으로서 우선순위를 부여하는 규칙은 다음과 같다.

규칙 1) 트리거그룹내의 규칙 수가 같은 경우, 제어정보의 수로 비교하고, 제어정보의 우선순위는 다음과 같이 정의한다.

"P" > "Op-P" > "Os" > "Op-Os" > "Ex" > "Op-Ex" > "S" > "Ya" > "Yp"

규칙 2) 트리거그룹내의 제어정보의 수가 같은 경우, 규칙연산자 수로 비교하고, 규칙연산자 우선순위는 다음과 같이 정의한다.

"U"(update) > "I"(insert) > "R"(retrieval)

규칙 3) 트리거그룹내의 규칙연산자의 수가 같은 경우, 임의의 규칙을 처리한다.

5.5.2 단계별 프로세서 할당 및 해제정책

능동 병렬규칙실행 모듈에서 병렬규칙실행 관리자의 역할 중 하나가 멀티프로세서 할당과 해제 시 CP로 하여금 프로세서를 관리하도록 하는 것이다. CP는 트리거그룹 생성정보를 기반으로 트리거규칙에 프로세서를 할당하며, 트리거규칙의 실행이 완료된 경우 다시 CP가 프로세서를 회수한다. 즉, CP는 병렬규칙실행 시 트리거그룹 생성 우선순위 정책에 의한 프로세서 할당과 자식-타스크의 제어정보로 프로세서를 해제하는 정책을 담당한다. 초기 규칙에 이벤트 시그널이 발생되는 경우, 트리거규칙에 프로세서를 할당하고 해제하는 정책을 제어정보 종류의 따라 분류하여 요약한 것이 표 2이다.

표 2. 프로세서 할당을 위한 제어정보

제어구조	의 미
"P"	트리거그룹생성우선순위정책에의해유한개의멀티프로세서를 병렬트리거규칙에할당
"Op-P"	선택적인실행정보로서프로세서를할당한다 그리고위에서 정의한 P같은방식으로할당
"Os"	트리거그룹우선순위정책과는상관없이유한개의멀티프로세서를상대우선순위값에의거하여트리거규칙에할당한다
"Op-Os"	선택적인실행정보로서프로세서를할당한다 그리고위에서정의한 Os와 같은방식으로할당
"Ex"	부모-타스크정보의조건에따라여러개 트리거규칙중 하나의규칙에만프로세서를할당한다
"Op-Ex"	선택적인실행정보로서프로세서를할당한다 그리고위에서 정의한 Ex와 같은방식으로할당
"S"	프로세서를부모-타스크기할당받으면다음레벨에있는 트리거규칙으로프로세서를상속하여할당
"Ya"	할당받은프로세서가도착하면이에대한프로세서실행해제를 CP에게알리고, 프로세서를할당받은모든트리거규칙이 도착되어야다음레벨의트리거규칙에CP가 프로세서를할당
"Yp"	할당받은프로세서가도착하면이에대한프로세서실행해제를 CP에게알리고, 프로세서를할당받은일부의트리거규칙이 도착되어또다음레벨의트리거규칙에CP가 프로세서를할당

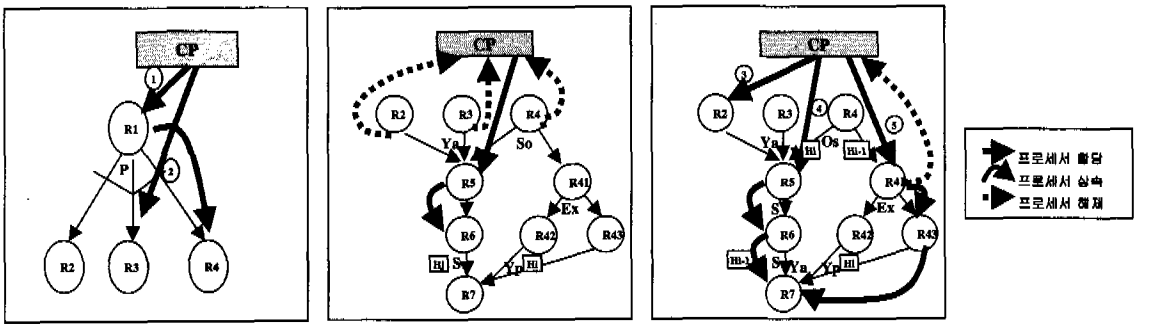

그림 7. 제어구조 정보에 따른 프로세서 할당 예

그리고 프로세서 할당정책을 적용한 예를 그림 7에 나타내었다. 할당 과정은 그림과 같이 3단계로 구분하여 프로세서가 2개만 있다고 가정하고 실행하면 다음과 같다.

1단계(a)로서 CP에서 트리거규칙이 존재하는 경우 우선적으로 프로세서를 할당하고 실행 완료 시 제어정보가 병렬 제어구조일 경우 트리거그룹 생성의 우선순위 정책에 의해 우선순위가 제일 높은 트리거규칙(R4)에 상위 레벨에서 실행했던 프로세서를 상속하고 그 다음 우선순위 트리거규칙(R3)에 CP는 나머지 하나의 프로세서를 할당한다.

2단계(b)로서 할당받은 트리거규칙이 실행이 완료된 경우의 프로세서는 CP가 해제(R4 규칙 점선 표시)하고 그 나머지 트리거규칙에 할당한다. 이와 같은 방식으로 제어정보가 "S"인 경우와 단일 트리거규칙(제어정보가 "Ex")인 경우는 프로세서를 상속하게 한다.

마지막 3단계(c)에서는 규칙응용 시나리오를 이용한 규칙실행 그래프의 프로세서 할당 방식을 전반적으로 도시한 예이다.

VI. 성능 평가

앞 절에서 제안한 규칙실행 전략과 기존의 규칙실행 방법과의 실행시간을 비교 분석한다. 이를 위하여 5.3절에서 제안한 규칙응용 시나리오를 이용하였으며, 트리거규칙의 실행시간은 실제업무의 난이도를 고려하여 할당하였다.

추정한 트리거규칙 실행시간은 표 3에서 가정한 규칙연산자의 실행시간과 표 4의 공정관리의 PERT 기법을 이용하여 규칙활동 추정과 기대시간을 바탕으로 표 5와 같이 규칙 실행시간을 산출한다.

표 3. 규칙연산자의 실행시간 가정

규칙 연산 \ 시간	a	m	b
규칙 검색	0.8	1.0	1.2
규칙 삽입	1.4	1.6	1.8
규칙 수정	1.6	1.8	2.0

(단위 : sec)

표 4. 규칙활동 추정과 기대시간

규칙활동시간 추정	규칙활동을 완료하는 기대시간	규칙활동 추정치의 분포
낙관치 : a 기대치 : m 비관치 : b 데이터수 : c	$T_E = \frac{a+4m+b}{6} \times c$	$\sigma = \frac{b-a}{6} \times c$

표 5. 규칙활동 기대시간

규칙 \ 시간	a(낙관치)	m(기대치)	b(비관치)	c(데이터수)	규칙 활동 기대시간
R1	0	0	0	-	-
R2	1.4	1.6	1.8	3	4.8
R3	0.8	1.0	1.2	2	2.0
R4	0.8	1.0	1.2	4	4.0
R5	1.6	1.8	2.0	3	5.4
R6	0.8	1.0	1.2	1	1.0
R7	1.6	1.8	2.0	2	3.6
R8	1.4	1.6	1.8	1	1.6
R9	1.6	1.8	2.0	2	3.6
R10	0.8	1.0	1.2	1	1.0
R41	0.8	1.0	1.2	2	2.0
R42	0.8	1.0	1.2	1	1.0
R43	1.6	1.8	2.0	1	1.8

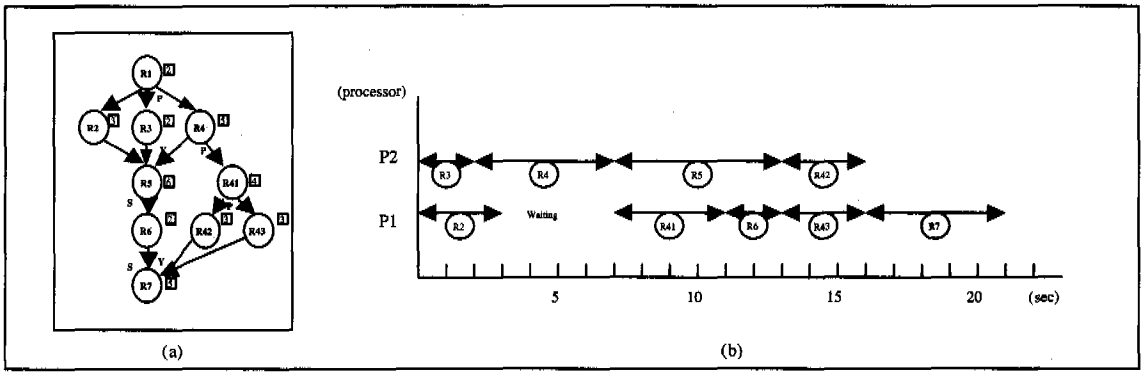

그림 8. 기존 제어구조의 응용 규칙실행 예

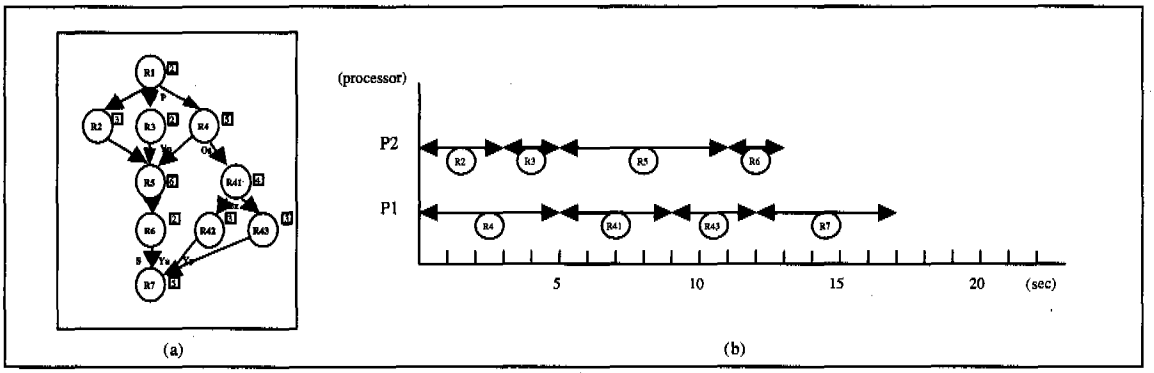

그림 9. 제안한 제어구조의 응용규칙실행 예

예를 들면 R2 트리거규칙에 데이터 수가 3이고 규칙활동 기대치가 1.6이라고 하면 규칙활동 기대시간은 4.8이 된다. 본 논문에서는 실행시간을 임의의 정수로 가정하여 그래프노드의 우측에 위치한 사각형으로 표현하였다. 그리고 프로세서는 2개로 실행된다고 가정하였다.

이러한 응용규칙 실행시간을 바탕으로 기존의 제어구조와 제안한 제어구조를 이용하여 실행시킨 결과를 그림 8의 (b)와 그림 9의 (b)로 나타내었다.

예를 들면, 기존의 제어구조를 이용한 그림 8의 (a)에서는 병렬실행 형태가 임의의 방식으로 트리거규칙을 선택하였다. 즉, R2 규칙을 P1에 R3 규칙을 P2에 할당하였다고 가정하자, 이러한 실행과정에서 P2가 P1보다 빨리 실행을 완료하였기 때문에 P2에 다시 R4의 트리거규칙을 실행하면 P1의 R2 규칙실행이 완료되었음에도 다른 규칙을 할당하지 못하고 대기상태로 들어간다. 그 이유 중 하나는 기존 제어구조정보 "Y"는 R5의 규칙을 트리거할 경우, 할당된 모든 프로세서가 도착되어야 하기 때문(R4가 실행 중이기 때문에 R5를 트리거할 수 없음)이고, 또 다른 하나는 R4가 실행 완료가 되지 않았기 때문에 R41 규칙을 트리거할 수 없다는 이유 때문에 P1은 대기상태로 들어가 있다.

또 다른 예를 들면, R41의 규칙실행이 완료된 경우 기존의 제어구조는 "Ex" (Exclusive)가 표현하지 못하기 때문에 제어정보 "P"라고 표현되었다고 가정하면, R42와 R43의 트리거규칙 모두를 실행하여야만 한다.

이런 측면으로 그림 8와 그림 9의 실행 결과에서 보듯이 기존 제어구조를 이용한 병렬 트리거규칙을 임의로 프로세서 할당 방식으로 실행할 때와 제안한 제어구조를 이용하고 트리거그룹 우선순위 기반인 프로세서 할당정책을 시행했을 때의 실행 시간을 비교한 결과 상당한 차이가 있음을 나타내었다. 표 6에서 기존 모델과 제안 모델을 비교 분석하였다.

표 6. 기존 모델과 제안 모델의 비교 분석

장단점 / 모델	단 점	장 점
기존모델	· 제어구조가 간단함 · 실 업무의 응용 예 부재 · 프로세서 우선 순위 할당 부재 (프로세서 할당 스케줄링 필요) · 병렬 규칙 실행 시 성능 저하	· 그래프 기반 트랜잭션 모델 이용 · 제어 구조의 단순성 →모델링 용이
제안모델	· 규칙 실행 활동 시간 추정 오버헤드 · 직렬성 유지 관리 부재	· 다양한 시멘틱에 의한 제어 구조 제시 · 프로세서 우선 순위 할당 방식 제시 (트리거 그룹 생성 정책에 의함) · 준, 동적 병렬 규칙 실행 시 성능 향상

Ⅶ. 결 론

기존의 병렬실행 제어구조는 단순하여 시멘틱을 충분하게 제공하지 못하기 때문에 본 논문에서는 실세계의 다양한 상황을 정확하게 표현할 수 있도록 시멘틱을 강화한 규칙실행 제어구조를 확장하여 제안하였다.

이러한 제어구조를 기반으로 실세계 업무에 이용한 규칙응용 시나리오와 그래프기반 규칙실행 그래프를 제시하였고 내포된 트리거규칙이 병렬처리로 실행되는 경우, 내포된 규칙 수와 기타정보를 기반으로 한 트리거그룹 생성정책과 알고리즘을 제시하였다. 또한 트리거그룹 우선순위 정책에 의한 멀티 프로세서 할당 및 해제 방안에 대한 예를 제시하였다. 그리고 본 논문에서 제안한 규칙실행 전략과 기존의 실행전략을 처리시간 기준으로 성능 분석을 시행하였다.

향후의 연구 과제로는 분산 환경에서의 규칙 간 제어 흐름을 기반으로 한 병렬규칙 실행전략과 실시간 환경에서 규칙의 활동 실행 시간을 고려한 병렬규칙 실행전략에 대해 연구가 진행되어야 한다.

참 고 문 헌

[1] Anca Vaduva. Rule Development for Active Database Systems: Ph.D. Thesis, Department of Computer Science, University of Zurich, Switzerland 1998.

[2] A.P Buchmann, J. Zimmermann, J.A. Blakeley, and D.L. Wells. Building an Integrated Active OODBMS: Requirements, Architecture, and Design Designs. In Proceedings of the 11th International Conference on Data Engineering, pages 117-128, Taipei, Taiwan, 1995.

[3] Prashant V. Cherukuri. A Task Manager for Parallel Rule Execution in Multi-processor Environments. Master's thesis: Department of Electrical Engineering, University of Florida, 1993.

[4] Umeshwar Dayal. Active Database Management System: In Proceedings of the Third International Conference on Data and Knowledge Base, page 150-169, Jerusalem, June 1988.

[5] Andreas Geppert, Stella Gatziu, Klaus R. Dittrich. Architecture & Implementation of the Active Object-Oriented Database Management System SAMOS: University Zurich, Technical Report, 1995.

[6] Meichun Hsu, Rivka Ladin, and Dennis R. McCarthy. An Execution Model For Active Database Management Systems: In Proceedings of the Third International Conference on Data and Knowledge Base, page 171-179, Jerusalem, June 1988.

[7] Ramamohanrao Jawadi, Stanley Y.W. Su. A Graph-based Transaction Model for Active Database and its Parallel Implementation: University of Florida, Technical Report, 1994.

[8] TR94-003, Ramamohanrao Jawadi, Stanley Y.W. Su. Incorporating Flexible and Expressive Rule Control in a Graph-based Transaction Framework: University of Florida, Technical Report, TR94-030, 1994.

[9] Stanley Y. W. Su, Ramamohanrao Jawadi, Prashant Cherukuri, Qiang Li, Richard Nartey. OSAM*.KBMS/P: A Parallel, Active, Object-oriented Knowledge Base Server: University of Florida, Technical Report, UF-CIS-TR-94-031, 1994.

[10] Yucel Sygin, Ozgur Ulusoy, Sharma Chakra-

varthy. Concurrent Rule Execution In Active Databases: University of Florida, Technical Report, 1997.

[11] Anca Vaduva, Stella Gatziu, Klaus R. Dittrich. Investigating Termination in Active Database Systems with Expressive Rule Languages: University Zurich, Technical Report, April 1997.

[12] Jennifer Widom, Stefano Ceri. Active Database Systems Triggers and Rules For Advanced Database Processing: Morgan Kaufmann Publishers, Inc. Page 78-206, San Francisco, California, 1996.

[13] 오성균, 임해철, 능동 데이터베이스의 병렬규칙 실행 모델에서 멀티프로세서 할당 정책, 한국정보과학회 '99 한국 데이터베이스 학술대회 논문집, 제15권, 1호, pp.253-260, 2. 1999.

오 성 균(Sung Kyun Oh) 정회원

1981년 : 홍익대학교 전자계산학과(이학사)
1984년 : 연세대학교 산업대학원 전자계산학과(공학석사)
1995년 : 홍익대학교 전자계산학과 박사과정 수료
1987년~현재 : 서일대학 전자계산과 조교수

<주관심 분야> 능동 데이터베이스, 분산 데이터베이스, 멀티미디어 데이터베이스

이 재 호(Jae Ho Lee) 정회원

1987년 : 홍익대학교 전자계산학과(이학사)
1989년 : 홍익대학교 전자계산학과(이학석사)
1996년 : 홍익대학교 전자계산학과 박사
1996년~현재 : 인천교육대학 컴퓨터교육과 조교수

<주관심 분야> 컴퓨터교육, 분산 데이터베이스, 웹 데이터베이스

임 해 철(Hae Chull Lim) 정회원

1976년 : 서울대학교 계산통계학과(이학사)
1978년 : 한국과학기술원 전자계산학과(이학석사)
1988년 : 서울대학교 컴퓨터공학과 박사
1978년~1981년 : 현대엔지니어링
1989년~1990년 : 미국 플로리다대학 방문교수
1981년~현재 : 홍익대학교 컴퓨터공학과 교수

<주관심 분야> 객체지향, 실시간, 분산, 멀티미디어 데이터베이스