효율적인 ECA Rule설계를 위한 Simulator의 구현 및 연구

김병욱 권순덕 고재진
울산대학교 컴퓨터정보통신공학부
{gom, trueguy, jjkoh}@cic.ulsan.ac.kr

A study and implementation of simulator which supports efficient ECA rule design

Kim Byung Wook° Kwon Sun Deok Koh Jae Jin
School of Computer Engineering and Information Technology

요 약

능동데이터베이스(active database)는 일반 데이터베이스와 능동 규칙을 결합한 형태로서, 데이터베이스 스스로 상태 변화에 대응한 조치를 취할 수 있는 시스템을 말한다. 이러한 능동데이터베이스에서는 규칙의 설계가 데이터베이스 활용의 큰 비중을 차지한다. 이러한 규칙의 효율적인 설계를 위해서 simulator의 도움이 필요하다. 이에 본 논문에서는 이러한 능동데이터베이스의 규칙을 효율적으로 설계하기 위한 도구의 설계에 대해서 연구한다.

1. 서론

능동데이터베이스는 일반 데이터베이스와 능동 규칙(rule)을 결합한 형태로서. 데이터베이스 스스로 상태 변화에 대응한 조치를 취할 수 있는 시스템이다.

여기에서 규칙은 능동데이터베이스에서 능동성을 담당하는 요소이며, 이는 사건(event), 조건(condition), 조치(action)로 구성되어 있다. [1]

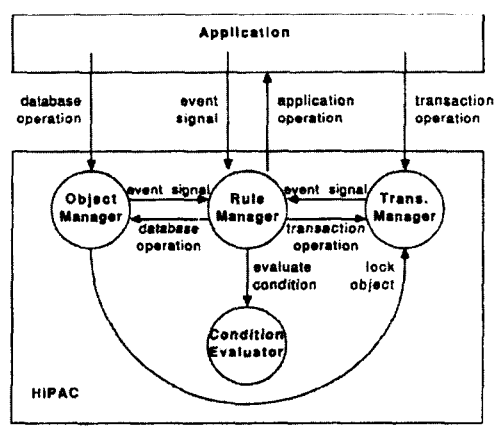

그림 1 능동 데이터베이스 구조(HiPAC)

능동 데이터베이스에서 사용되어지는 각각의 구성요소에 대해서 간단하게 살펴본다.

(1)Object Manager - 객체들의 형식과 연산들에 대한 정의를 정의한다.

(2)Rule Manager - 해당하는 이벤트가 감지되어지면 적당한 규칙을 실행시킨다.

(3)Transaction Manager - 트랜잭션의 생성과 종료와 병행성 제어를 담당한다. 지원되는 연산은 Create Transaction, Commit Transaction, Abort Transaction으로 3가지의 연산이 지원된다.

(4)Condition Evaluator - multiple query실행때에 조건들이 만족이 되었는지 효율적으로 결정을 하는 것을 담당한다. 지원되는 연산은 Add Rule, Delete Rule, Evaluate Rule의 연산이 지원된다.

(5)Event Detector - 기본적인 이벤트(DB-event, temporal-event, external-event)의 발생 감지를 적용한다. 지원되는 연산은 Define Event, Delete Event, Enable Event, Disable Event의 연산을 지원한다.[3]

현실의 응용프로그램에서 능동데이터베이스 시스템은 ECA(Event, Condition, Action)규칙을 사용하여 구현하고, 디버깅, 그리고 수많은 수의 규칙들을 관리한다. 그러한 능동데이터베이스 시스템에 있어서 규칙의 유용성과 의미들은 설계와 정확성의 검증을 어렵게 만든다.

이러한 규칙의 효율적인 설계를 위해서 simulator의 도움이 필요하다. 규칙들을 설계하고 가상적으로 규칙들의 적용을 확인함으로서 잠재적으로 일어날 수 있는 오류를 줄일 수 있다.

효과적인 능동데이터베이스 시스템의 도입에는 규칙들의 상호 작용(규칙들, 사건들, 데이터베이스의 객체들)을 쉽게 이해하고 검증하기 위해서 simulator가 필요하다[4]

능동 데이터베이스의 Simulator는 기존의 디버거는 적당하지 않다.

기존의 디버거는 낮은 수준의 변수와 함수, 포인터 등과 같은 세부적인 것들에 치중하였지만, 능동데이터베이스의 디버거 혹은 simulator는 상호작용을 이해하는 것에 도움을 주는 것을 목적으로 한다. 예를 들어 규칙들과 데이터베이스 개체들 사이에 일어나는 상호작용에 대한 이해가 필요하다. 이것은 기존의 디버거 프로그램과 반대되는 성격을 가지고 있다.

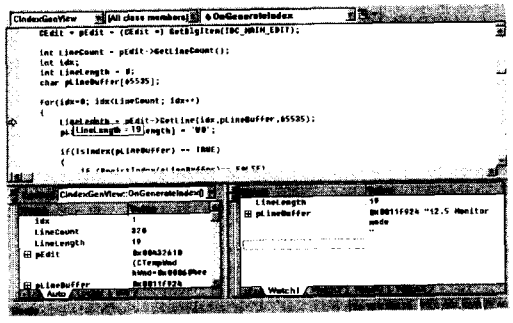

그림 2 기존의 디버거 작업 화면

그렇지만 이것은 기존의 디버거를 대신한다고는 말할 수 없다. 대신 이것은 기존의 디버거 보다 능동 데이터베이스에서 규칙들의 설계, 구현, 디버깅에 있어서 고차원적인 수준의 자료 추상화를 제공하는데 도움을 준다고 말할 수 있다.

이에 본 논문에서는 이러한 능동데이터베이스의 규칙을 효율적으로 설계하기 위한 도구의 설계와 구현에 대해서 연구한다.

2. 모델

전형적인 프로그래밍 디버깅 환경에서는 변수들의 값, 서브루틴 호출, 예외상황, 스택의 상태, 포인터 참조들이 중요한 요소였다. 그러한 환경에서 어떠한 문제가 발생하면 사용자는 그 문제를 해결한다. 그리고 디버거의 도움으로 사용자는 오류가 발생한 위치를 찾아낼 수 있다. 디버거는 단지 낮은 수준의 도움을 주고 사용자는 이 도움으로 오류를 발견할 수 있다.

여기서 기존의 디버거와 구별이 되어지는 부분을 살펴보기로 한다.

(1) Database Component

Database가 무엇을 하고 있는지를 정확히 알기 위해서 locking정보, transaction정보 등을 사용자에게 자세히 제공해 줄 수 있어야 한다.

(2) Rule and event interaction

규칙들 사이의 상호작용, 데이터베이스 객체와 규칙들 사이의 상호작용을 도식적으로 나타낼 수 있어야 한다.

특히 병행 transaction의 경우에는 발생하는 사건과 적용되는 규칙들 사이의 관계를 나타낼 수 있어야 한다. [2]

그림 3 Sentinel에서의 visualization

3. 설계

구현하고자하는 Simulator의 구조는 다음과 같다.

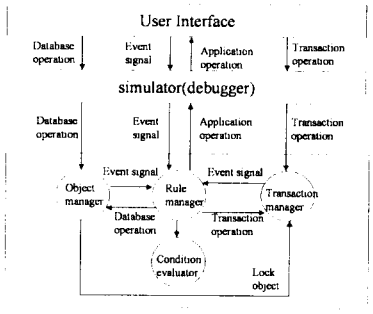

그림 4 Simulator의 위치

그림에서 보는 것과 같이 사용자는 사용자 인터페이스를 통해서 Simulator에게 명령을 내리고 Simulator는 그 명령을 해석해서 능동 데이터베이스와 통신을 한 다음에 그 결과를 사용자 인터페이스에 맞게 시각적으로 보여준다.

이 결과를 통해서 사용자는 자신이 원하는 결과와 진행사항을 한눈에 알 수 있다.

또한 사용자가 요구한 Rule들에 대해서 Rule Graph를 작성해서 사용자에게 Rule실행과정을 이해하기 쉽게 만든다.

사용되어지는 Database는 모든 능동규칙을 지원하는 Database에 대해서 지원을 하는 것을 목표로 하고 있지만, 그 중에서 능동 객체지향 데이터베이스(AOODB)를 지원하는 것으로 한다. 본 논문에서는 그 중에서 AT&T에서 개발한 AOODB인 ODE를 대상으로 하였다.

그리고 Simulator로 가상의 작업을 한 다음에도 원래의 데이터베이스에는 영향을 미치지 않아야 하는데, 이를 위해서 Rollback과 데이터베이스 테이블을 복사해서 작업하고 취소를 시키는 방법이 있을 수 있다.

작업이 시작되면 새로운 임시 테이블을 만드는 방법은 확실하게 작동하는 방법이지만, 테이블을 작성하는것에 걸리는 부담이 크기 때문에 현재의 설계에 있어서는 제외한다. 현재 설계대상이 되는 ODE에서는 Rollback을 지원하기 때문에 Rollback 방법을 적용한다.

데이터베이스와 디버거 사이의 통신을 위해서는 여러 가지 통신 방법이 존재할 수 있지만, socket 통신과, named pipe방법을 사용한다.

그렇지만 그것들을 제외하더라도 작업 내용에 대해서 log file들을 기록하는 것이 바람직하다. [2]

사용자로부터 규칙의 입력, 행동의 정의를 위해서 SQL을 이용한다.

4. 결론 및 향후과제

본 논문에서는 효율적으로 능동 규칙을 설계하는데 도움을 주고, 설계자의 실수를 최소화시키기 위한 Simulator의 설계에 대해서 기술하였다.

실제의 능동 규칙을 사용자 위주로 살펴보면서 원하는 행위들이 어떻게 발생되어 지는지를 확인해서 Rule설계자의 실수를 최소화시킬 수 있다.

하지만 아직 Simulator는 원본 데이터베이스를 변경하지 않고 작동해야 하기 때문에 이 부분에서 부가적인 작업들이 많이 일어나게 되는데, 이 곳에서의 부가적인 작업들을 최소화시키는 방법과 데이터베이스 서버와 Simulator의 통신을 방법들도 개선이 필요하다.

참고문헌

[1] 김동욱 , 김명호, 이윤준
복잡한 트리거 조건을 위한 점진적 평가 방법,1996

[2] T. Coupaye, C. L. Roncancio, C. Bruley
3D Visualization Of Rule Processing In Active Databases

[3] Dennis R. McCarthy
The Architecture Of An Active Data Base Management System

[4] S. Chakravarthy,Z.Tamizuddin,J.Zhou
A Visualization and Explanation Tool for Debugging ECA Rules in Active Database ,1995