주기억장치 상주형 데이터베이스를 위한 능동 커널의 설계 및 구현

한혁*, 차상균*, 진성일*
*충남대학교 컴퓨터과학과
e-mail : hhan@cs.cnu.ac.kr, kkandol@cs.cnu.ac.kr, sijin@cs.cnu.ac.kr

Development of Active Kernel for a Memory Resident Database(KAIROS)

Hyoeok Han*, Sang-Gyun Cha*, Sung-Il Jin*
*Dept. of Computer Science, Chung-Nam National University
e-mail : hhan@cs.cnu.ac.kr, kkandol@cs.cnu.ac.kr, sijin@cs.cnu.ac.kr

요 약

실시간 응용의 객체 관리에 적합한 주기억 장치 상주형 데이터베이스에서 능동적 자료의 처리를 위한 모니터링 기능은 실시간 시스템 응용 분야에서 제공되어야 하는 필수적인 기능 요소라 할 수 있다. 본 논문에서는, 현재 질의어 국제 표준으로 제정되고 있는 SQL3 에서 제시된 트리거 기능과 실시간 응용을 위한 시간 사건 기능을 주기억 장치 상주형 DBMS 인 KAIROS[6]를 확장하기 위한 능동커널의 설계 방안을 제시하였다.

1. 서론

정보화 사회의 다양한 요구 속에서 실시간 객체관리 데이터베이스 시스템의 응용은 그 동안 여러 분야에서 많은 연구가 진행되어왔다. 이러한 연구는 최근의 급변하는 하드웨어 및 소프트웨어의 환경에 따라 새로운 응용과 연계되어 새로운 기술의 접목이 요구되고 있다. 특히 실시간 처리(Real-Time Computing)를 위한 응용에서는 효과적이고 체계적인 자료의 관리뿐만 아니라 입력된 자료에 대한 능동적이고도 정확한 자료의 처리들 지원하는 모니터링 기능을 요구하고 있다. 예를 들면, 증권관리 시스템에서의 매매제어, 환자의 상태를 알리는 의료 응용분야, 또는 교통혼잡이 예상되는 운송 응용분야 등의 다양한 응용에서 지속적인 데이터의 관리 및 모니터링 기능은 필수적이라 할 것이다. 따라서 본 연구에서는 실시간 응용에서 주목받고 있는 주기억장치 상주형 데이터베이스 시스템의 개념을 확장하여 모니터링 기능을 수행할 수 있는 능동 커널을 설계 및 구현함으로써 시스템의 기술적, 기능적 성능의 향상을 가져오고자 한다.

본 논문은 소프트웨어연구센터의 지원을 받아 수행.

능동 데이터베이스 시스템은 데이터베이스의 상태에 능동적으로 대처할 수 있는 기능을 제공하는 데이터베이스 시스템으로써, 이를 실현하기 위해 능동규칙(Active Rule)을 정의하여 데이터베이스에서 발생한 사건(Event)을 검출하고 이를 평가한 후 적절한 조치를 수행하는 일련의 반응적 행동(Reactive Behavior)을 제공하는 데이터베이스 시스템이다[3].

현재 능동 데이터베이스에 관한 연구는 시.공간 데이터베이스 및 연역(Deductive) 데이터베이스에 관한 연구와 함께 현대 데이터베이스(Modern Database)에서 주요 연구 분야로써 80 년대 후반기부터 본격적인 연구가 진행되어 왔다. 현재 미국과 유럽의 여러대학 및 연구소를 중심을 연구용 시제품이 시험 및 운용되고 있고, 데이터베이스 질의어 표준으로 제정되고 있는 SQL3 에서 데이터베이스의 능동적 행동을 지원할 수 있는 트리거(Trigger)를 정의하여 상용 데이터베이스 시스템에서도 그 기능성을 제공하도록 하고 있다[5].

능동 데이터베이스 시스템은 기반으로 하고 데이터 모델에 따라 크게 분류할 수 있는데, 관계형 데이터베이스 시스템을 기반으로 능동 기능을 구현한 시스템으로는 Starbust, POSTGRES, Ariel 등이 있으며, 객체지

향 데이터베이스 시스템을 기반으로 하는 시스템들로는 HiPAC, SAMOS, Sentinel, REACH 등이 대표적인 시스템들이다[3]. 이 외에 오라클을 비롯한 대부분의 상용 데이터베이스 시스템에서는 SQL3 에서 정의하고 있는 트리거가 구현되어 사용되고 있다[5].

본 연구에서는 현재 상용화 및 연구용 데이터베이스로 사용되고 있는 주기억장치 상주형 데이터베이스 시스템인 KAIROS 를 확장하여 능동 기능을 수행할 수 있도록 SQL3 표준안을 시스템에 구현하며, 트리거 시스템에서는 취약한 시간 사건에 대한 능동적 기능으로 현재 실시간 능동 객체 모델로 주목받고 있는 TMO(Time-triggered and Message-triggered Object)[1]를 이용하여 시간 사건지원을 위한 커널 모듈을 구현하고자 한다. 본 논문의 구성은 다음과 같다. 먼저 2 장에서 국제 표준인 SQL3 에서 제안된 트리거와 실시간 능동 객체인 TMO 를 소개하고, 3 장에서는 전체 시스템 구조 및 트리거 시스템의 구조를 설명하며, 4 장에서는 KAIROS 시스템에 구현하는 트리거 문법, 트리거 시스템을 위한 추가된 카탈로그와 시간 관리자에에 대해 기술한다. 마지막으로 본 연구에 대한 결론과 향후 연구방향에 관해 논의한다.

2. 관련연구

이 장에서는 SQL 국제표준인 SQL3 에서 제안된 트리거 문법을 소개하고 시간사건 지원을 위한 실시간 능동 객체인 TMO 를 살펴본다.

2.1 SQL3 에서의 트리거

국제 표준으로 정의하고 있는 트리거는 능동 데이터베이스 시스템에서 사용하고 있는 것과 같은 규칙(rule)을 따른다. 사건(evnet)-조건(condition)-조치(action)로 이루어 지는 능동 규칙에서, 사건은 모니터링되고 있는 데이터베이스 연산이며, 조건은 임의의 프리디킷이고, 조치는 일련의 SQL 프로시져들이다. 사건이 발생하고 조건이 참이되면 조치에 명시된 프로시져들이 순차적으로 수행된다. 아래의 [그림 2-1]은 국제 표준의 트리거 생성문 및 삭제문이다[7].

```
CREATE TRIGGER [schema_name.] trigger_name
    trigger_time trigger_event [trigger_priority]
    trigger_target
    [referencing_clause]
    [trigger_condition]
    trigger_action
    trigger_granularity;
trigger_time : BEFORE|AFTER|INSTEAD OF
trigger_event : INSERT|DELETE|UPDATE
            [OF column_name_list]
referencing_clause : REFERENCING
    OLD AS old_value_tuple_name |
    NEW AS new_value_tuple_name |
    OLD_TABLE AS old_value_table_name |
    NEW_TABLE AS new_value_table_name
Trigger_condition : WHEN ( condition )
Trigger_action :  INSERT statement |
                  DELETE statement |
                  UPDATE statement
Trigger_granularity : FOR EACH {ROW|STATEMENT}

DROP TRIGGER [schema_name.] trigger_name ;
```

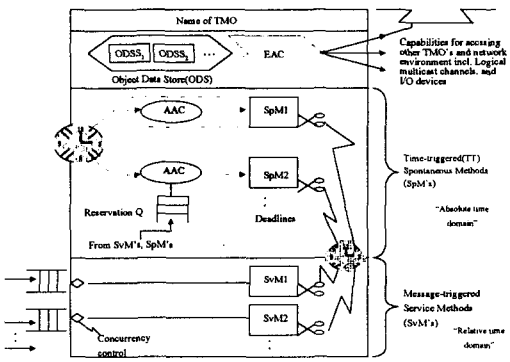
[그림 2-1] SQL3 트리거 문법

여기서 trigger_event 는 능동 규칙의 사건부이고 trigger_condition 과 trigger_action 이 각각 조건과 조치에 관한 명세를 기술하도록 되어 있다. trigger_time 은 트리거를 유발하는 시기를 기록하는 부분으로 BEFORE 인 경우, 트리거를 유발하는 사건, 즉 데이터베이스 연산에 우선하여 정의된 트리거를 수행하도록 지정한다. 트리거를 유발할 때는 항상 대상이 되는 테이블이 명시되어야 하는데 trigger_target 에서 이를 정의한다. 지정된 테이블에서 데이터베이스 연산이 영향을 미치는 각 행에 대하여 매번 트리거를 유발할 것인지, 하나의 데이터베이스 연산에 대하여 트리거를 유발할 것인지를 명시하는 부분이 trigger_granularity 이다.

생성된 트리거를 삭제하는 문장은 일반 적인 SQL 문장과 유사하다. DROP 문을 사용하여 트리거 이름 명시하여 줌으로써 생성된 트리거를 삭제한다.

2.2 시간 사건 지원을 위한 TMO 모델

TMO 모델은 기존의 객체 모델에 대한 확장으로, 기본적인 구조는 [그림 2-2]에 잘 나타나 있다. TMO 모델은 시간적인 분석을 용이하게 해 주는 TAC(TMO Access Capability)과 실질적인 구성 요소인 ODSS, SpM, 그리고 SvM 등 네 가지의 요소로 구성된다. TAC, ODSS, SpM 그리고 SvM 에 대한 기술은 다음과 같다



[그림 2-2] TMO 모델

- ✓ TAC : 어떤 TMO 의 SpM 이나 SvM 에서 다른 TMO 에 있는 SvM 에 대한 호출이 있을 경우, 이를 위한 통신 채널(Communication Channel) 을 할당 받아 가지고 있어야 한다. 채널을 할당 받기 위해서는 호출하려는 SvM 의 이름과 이 SvM 이 속한 TMO 의 이름 등이 필요하다. TAC 에 포함되는 정보에는 메시지를 주고 받기 위한 채널의 ID 와 데이터를 주고 받기 위한 저장 장소 등이 있다.
- ✓ ODSS : 어떤 TMO 모델 내의 SpM 과 SvM 이

서로 공유하는 데이터를 저장하기 위한 공간인 ODS(Object Data Store)가 있는데, 이 ODS 는 세그먼트 단위로 공유된다. 이를 ODSS 라 한다. 그리고 이러한 데이터는 유효기간이 지나면 무효한 데이터가 된다.

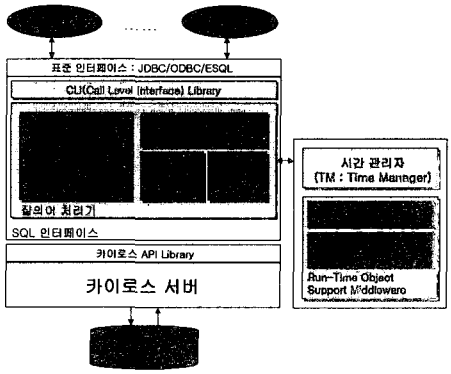
- ✓ SpM : 어떤 RTO 에서 해야 할 작업들은 메소드로 표현이 되는데, 이 중 주기성을 띠거나 시간성을 갖는 메소드 그룹이 SpM 이다. 그리고 이러한 시간적인 특성은 AAC(Autonomous Activation Condition)에 자세히 기술한다.
- ✓ SvM : 어떤 RTO 의 메소드 중에서 다른 메소드로부터 온 서비스를 수행해 주기 위한 메소드 그룹이 SvM 이다. 이러한 SvM 은 고유한 deadline 을 갖는다.

이렇게 TMO 를 구성하는 모든 구성 요소들이 가지는 시간적인 특성을 설계 단계에서부터 정확히 명시해 줌으로써, 그 TMO 에 대한 시간적인 특성의 분석을 용이하게 해 주고, 시스템의 설계 단계에서부터 시스템에 대한 실시간성을 보장해 준다[1].

3. 시스템 구조

3.1 통합 시스템 구조

SQL 은 데이터베이스 접근을 위한 표준 언어로서 대부분의 데이터베이스 시스템에서 지원된다. SQL 인터페이스의 제공으로 인해 프로그램 개발자들은 보다 쉬운 인터페이스를 통해 프로그램의 개발 및 유지보수를 보다 빠르고 쉽게 개발할 수 있다. Kairos 시스템의 전체 구조도는 [그림 3-1]과 같다. 트리거를 비롯한 SQL 처리를 위한 인터페이스가 시스템 커널에 포함되어 있고, 시간 사건 지원을 위한 시간관리자가 TMO 지원 미들웨어와 함께 본 시스템과 분리되어 있다. 이러한 통합 구조[2]는 실시간성을 지원하는 TMO 모델을 일반적인 트리거 시스템에 적용하기 위한 구조로써 데이터베이스 시스템과 실시간 객체 지원 시스템과의 연동을 위한 것이다.

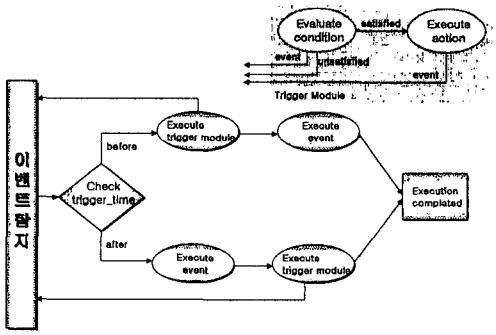


[그림 3-1] 통합 시스템 구조

트리거를 비롯한 SQL 의 처리 흐름을 살펴보면 다음과 같다. 먼저 표준 인터페이스를 이용해서 응용 프로그램을 작성하고, 표준 인터페이스는 CLI 와 대응된다. CLI 에서는 질의어 처리 관리자로 질의어들을 보내며, 질의어 처리 관리자는 이러한 질의어를 SQL 파서로 전달한다. SQL 파서는 구문들을 분석한 후 LLI 라이브러리를 호출하게 된다. 이 LLI 라이브러리는 Kairos API 를 이용하게 Kairos 서버에 접근하여 질의를 처리한다. SQL 파서에서 호출된 LLI 라이브러리의 함수들은 결과를 CLI(Call Level Interface)에 전달하게 되며, CLI(Call Level Interface) 인터페이스에서는 결과값을 유지하고 있다. 이러한 결과값을 표준 인터페이스를 통해 응용 프로그램에서 사용하게 된다.

3.2 트리거 시스템 수행 구조

트리거 시스템의 동작은 사건 탐지로부터 시작된다. 응용으로부터의 데이터베이스 연산이나 시간 관리자에로부터 보고된 사건 탐지 후, 탐지된 이벤트에 해당하는 트리거 정보를 시스템 카탈로그로부터 디스패치한다. 카탈로그 항목 중에서 trigger_time 을 검사하여 before trigger 인 경우, 트리거 모듈을 우선 수행한다. 이때, 트리거 모듈의 조건이나 조치 수행 중에 발생하는 또 다른 사건들은 이벤트 탐지 모듈로 보고되며 순환적으로 트리거를 수행한다. 트리거 모듈이 수행된 후에는 트리거를 유발했던 원래의 연산들이 수행되며, 하나의 트리거 수행에서 발생한 모든 사건에 대한 트리거 완료 시점에서 최초 트리거를 유발했던 연산의 종료가 이루어진다. [그림 3-2]는 이러한 트리거 시스템의 수행 모델을 도식화 한 것이다.



[그림 3-2] 트리거 시스템의 수행 모델

4. API 설계 및 구현

4.1 KAIROS 트리거 문법 및 API

KAIROS 데이터베이스 시스템에서 적용한 트리거 문법은 아래와 같다. SQL3 와 마찬가지로 능동 규칙의 사건은 INSERT, DELETE, UPDATE 와 같은 데이터베이스 연산들이며, 조치부는 데이터베이스 조작문과 화면 출력 같은 몇 개의 함수로 구성된다. KAIROS 트리거 시스템은 SQL3 와는 달리 사건-조치로 이루어지는 규칙을 사용한다. 조건부에 해당하는 프리디켓을

기반 시스템에서 충분이 제공하지 않는 관계로 차후 구현 부분으로 남겨놓았음을 밝혀둔다.

```
Create Trigger [schema_name.] trigger_name
    trigger_time trigger_event
    trigger_target
    trigger_granularity
    trigger_action;
trigger_time : BEFORE | AFTER
trigger_event : INSERT | DELETE |
            UPDATE [OF column_name_list]
trigger_target : ON [schema_name.]table_name
trigger_granularity : FOR EACH STATEMENT
trigger_action : trigger_action_API |
          INSERT statement |
          DELETE statement |
          UPDATE statement

DROP TRIGGER [schema_name.] trigger_name ;
```

[그림 4-1] KAIRS 트리거 문법

다음은 KAIROS SQL 인터페이스 중에서 트리거 지원을 위한 CLI(Call Level Interface)를 정의한 것이다. 응용프로그램에서는 아래의 인터페이스를 사용하여 트리거를 정의하고 삭제할 수 있으며, 이를 통한 트리거 생성문 및 삭제문은 질의 분석기를 통하여 시스템 인터페이스로 전달된다.

```
int KairosCreateTrigger( int, char*, int)
int KairosDropTrigger(int, char*)
int KairosCreateTTrigger(int, char*, int)
int KairosDropTTrigger(int, char*)
```

[그림 4-2] 트리거를 위한 CLI 인터페이스

4.2 트리거를 위한 시스템 카탈로그

현재 KAIROS 에서 제공하는 API 는 각 필드나 테이블에 대한 정보를 물리적인 값 즉 byte 단위로 처리해야 하며 DB, 테이블, 인덱스, 필드 등에 대한 정보를 프로그래머가 기억하고 있어야 하는 불편함이 있다. 이런 정보를 사용자가 이해하기 쉬운 값들로 저장 관리 하기 위하여 시스템 카탈로그를 사용한다. [그림 4-3]은 트리거 지원을 위해 시스템 카탈로그 관리자에 새로이 추가된 트리거 카탈로그 자료구조이다.

```
typedef struct triggers *triggers_t
typedef struct triggers
{
    int trigger_schema_ID;
    int trigger_ID;
    char* trigger_name;
    int trigger_time;
    int trigger_event;
    int target_table_ID;
    int trigger_granulariry;
    char* trigger_action;
    int trigger_status;
    int trigger_valid;
}triggers_info;
```

[그림 4-3] 트리거를 위한 시스템 카탈로그 자료구조

응용으로부터 SQL 문을 사용하여 데이터베이스 연산이 수행되면 먼저 질의 분석기를 통하여 요청된 질의를 파싱하고 시스템 카탈로그를 검색하여 해당 질의에 대한 트리거 존재 여부를 검사한 후 이를 실행한다.

4.3 시간 관리자 설계

시간 관리자는 하나의 TMO 객체로 구성된다. 시간 관리자에 등록할 수 있는 시간 사건으로는 절대시간(Absolute Time), 상대시간(Relative Time), 주기시간(Peoriodic Time)을 정의 할 수 있으며[4], 시간지원 트리거 생성문을 통하여 트리거를 생성하며 질의 분석기를 거쳐 시간 관리자에 등록된다. TimedTMO(시간관리자)의 SpM 에 등록된 시간 사건들은 사건 발생시에 트리거 시스템의 사건 탐지 모듈로 트리거 ID 와 함께 전달한다.

5. 결론 및 향후 연구과제

주기억 장치 데이터베이스 시스템인 KAIROS 에서 능동 기능의 지원은 그 응용분야와 활용에 있어 필수적인 요소라 할 수 있을 것이다. 따라서 본 논문에서는 이의 구현에 적용한 시스템의 구조와 인터페이스에 대하여 살펴보았고, 시간 지원 트리거 시스템의 구현을 위하여 실시간 능동 객체로 각광 받고 있는 TMO 모델 적용에 대하여 살펴보았다.

실제 데이터베이스 시스템의 커널부분과 TMO 지원 미들 웨어와의 인터페이스 부분이 좀 더 상세하게 연구되어져야 할 부분이며, 이러한 시스템간의 연동으로 인한 성능 저하를 최소로 하기 위한 방안이 연구되어져야 할 것이다.

참고문헌

[1] Kim, K.H. et al., "Distinguishing Features and Potential Roles of the RTO.k Object Model", Proc. 1994 IEEE CS Workshop on Object-oriented Real-time Dependable Systems(WORDS), Oct. '94, Dana Point, pp.36-45.

[2] D.R. McCarthy, U. Dayal, "The Architecture of An Active Data Base Management System," *ACM SIGMOD 18(2)*, pp.215-224, 1989.

[3] N.W. Paton, O. Diaz,"Active Database Systems", *http://www.cs.man. ac.uk/~norm/.*

[4] C. Ryu, Hyeok Han, Y.K Kim "Kernel Structuring using Time-Triggered Message-Triggered Objects for Real-Time Active DBMS in Layered Architecture", ISORC 2000 , pp148-155.

[5] 백진아,이미영,박영, "바다-II 에서 트리거의 설계 및 구현" 데이터베이스 연구회지, 제 14 권, 제 1 호, 1998.2

[6] 이경모, 임정옥, 김경배, "Mr.RT 3.0: 고성능 실시간 트랜잭션 처리를 위한 주기억장치 상주형 실시간 데이터베이스 시스템," 한국정보과학회 추계학술논문집 제 25 권 제 2 호, 1998, pp.208-210

[7] Jim Melton, "ISO-ANSI Working Draft Database Language SQL3",ISO-ANSI,1991.