트랜잭션 처리 시스템을 위한 동시성 제어의 기능시험 기법

홍석희*

Functional Testing Techniques for Concurrency Control in Transaction Processing Systems

Seok-Hee Hong*

이 논문은 2003학년도 경성대학교 특별과제연구비에 의하여 연구되었음

요 약

정해진 시간 내에 사용자 요구 조건을 충족시키도록 복잡한 소프트웨어를 개발하는 경우 시스템 시험은 중요한 요인이다. 데이터베이스 관리 시스템의 모듈 중 트랜잭션 처리 시스템은 다중 사용자 환경에서 트랜잭션의 수행과 데이터 일관성을 유지하는 중요한 기능을 한다. 본 논문은 트랜잭션 처리 시스템의 동시성 제어 기능이 요구조건을 만족하는지를 체계적이고 자동적으로 시험하기 위한 기법을 제안한다. 다섯 가지 잠금 모드를 지원하는 2단계 잠금 기법을 적용하는 동시성 제어 모듈의 기능을 자동적으로 시나리오 기반의 기능시험을 하고 시험 결과를 사용자에게 제시한다. 또한, 본 논문에서 제안한 기능시험 기법을 실제 데이터베이스 관리 시스템의 모듈 시험 과정에 적용하여 그 타당성을 확인한다.

ABSTRACT

System test is an important factor in the development process of good quality complex software in time, ensuring user requirements. Transaction processing module of database management systems schedules multiple transactions effectively and ensures that each transaction preserves data consistency. In this paper, we propose automatic functional testing techniques which ensure systematically that the implemented concurrency control function confirms to its requirements. The proposed testing technique tests functions of concurrency control module based on scenario without user intervention, and displays the result of functional testing. Finally, we utilize the proposed functional testing technique in the testing process of a database management system.

키워드

데이터베이스, 트랜잭션, 동시성 제어, 기능시험

Ⅰ. 서 론

복잡한 소프트웨어를 정해진 계획대로 적절한 비용으로 개발하기 위해서는 요구조건 검증이나 시스템 시험과 같은 중요한 요인들이 필요하다. 특히, 소프트웨어 개발과정 중에서 코딩 단계 이후에 다양한 종류

*경성대학교 컴퓨터과학과

접수일자 : 2003. 2. 10

의 시스템 시험(testing)을 거쳐서 소프트웨어의 오류를 찾아내고 수정해야 한다. 시스템 시험이 소프트웨어 개발에 중요한 과정임에도 불구하고 많은 경우에 비체계적이고 많은 시간이 걸리는 방법으로 오류 확인과 수정을 수작업으로 하고 있다. 상용 소프트웨어 시스템 개발의 경우 시스템 시험 과정은 상당히 내부적인 작업이고 특정 기능에 적합한 시험 방법으로 일반적으로 사용되기는 힘들다[1].

본 논문은 트랜잭션 처리 시스템의 동시성 제어 모듈에 대한 기능시험을 체계적이고 효율적으로 수행하기 위한 트랜잭션 모델을 제시하고 기능시험 기법을 제안한다. 동시성 제어 기능은 트랜잭션들 사이에 공유되는 데이터를 일관성 있게 사용하도록 트랜잭션들의 작업을 통제하는 역할을 한다. 동시에 실행되는 트랜잭션들 사이의 상호관계를 통제하기 때문에 오류를 발견하기가 쉽지 않다. 제안하는 기능시험 기법을 활용하여 잠금 기법(locking protocol)을 기반으로 한 동시성 제어 모듈의 오류를 효율적으로 찾을 수 있다.

II. 관련 연구

2.1. 시스템 시험

복잡한 소프트웨어를 정해진 시간 내에 오류 없이 개발하기 위해서 필요한 많은 요소들이 있다. 이 요소들 중에서 요구조건 검증(requirements validation)과 시스템 시험(system testing) 등은 소프트웨어 품질에 직접적인 영향을 준다[1, 3]. 요구조건 검증은 소프트웨어의 개발 목적과 요구조건이 완벽하게 만족되는지를 보장하는 과정으로 실제 소프트웨어의 코딩 전에 완료되지 않으면 수정하기가 힘들게 된다.

시스템 시험은 구조적(structural) 방법과 기능적 방법으로 분류할 수 있다. 구조적 방법은 백색상자(white box) 방식으로 소프트웨어의 내부 구조를 이해하고 분석하여 시험하는 방식이며, 기능적(functional) 방법은 흑색상자(black box) 방식으로 시험 대상의 내부 구조를 무시하고 정해진 기능규격 및 요구조건을 기반으로 소프트웨어의 실행 결과를 분석하여 시험하는 방식이다[2, 3]. 시스템 시험의 최근 연구 결과로 SCENT는 시스템 시험과 요구조건 검증을 위해서 시나리오 기반의 시험 도구를 제공한다[11]. SCENT는 스크립트 언어 형식과 그래프 표현방식으로 요구조건과 시험 사례를 제공하여 체계적인 소프트웨어 개발 과정을 제공한다. 그러나, 요구조건과 시험 사례의 형식이 상당히 제한되어 있기 때문에 트랜잭션 처리 시스템 등과 같은 다중 사용자 환경에는 적합하지 않다.

2.2. 트랜잭션 처리 시스템의 시험

안전하고 효율적으로 다수의 사용자에게 데이터베이스를 제공하는 데이터베이스 관리 시스템은 상당히 복잡한 소프트웨어다. 특히, 데이터베이스 관리 시스템의 트랜잭션 처리 모듈은 다중 사용자 환경에서 복잡한 방식으로 트랜잭션들 사이의 상호관계를 통제하기 때문에 오류를 찾기가 상당히 힘들다. 데이터베이스 시스템의 하부 구조인 트랜잭션 처리 기능을 포함하고 있는 저장 시스템인 MiDAS을 위한 기능 시험 도구를 개발한 연구가 있다[10]. MTS(MiD AS Test Suite)라는 이 시스템은 MiDAS에서 제공되는 모든 API에 대한 체계적이고 자동화된 기능시험 환경을 제공한다. 임의의 기능시험을 기술하는 시나리오를 MTS가 처리하면서 MiDAS에 대한 기능시험을 실행하여 오류를 탐지한다. 그러나, MTS는 가장 상위의 API에 집중된 시험을 하기 때문에 트랜잭션 처리 시스템에 대한 집중적인 기능시험을 지원하기는 쉽지 않다.

다중 트랜잭션들 사이의 데이터 일관성을 시험하기 위해서는 체계적이고 자동화된 시험 기법이 필요하지만 대부분의 상용 DBMS 업체에서 사용하는 시험 기법은 공개되지 않는다. 또한, 범용 소프트웨어 시스템에 대한 시스템 시험 기법의 개발에 대한 연구 성과는 많았으나 트랜잭션 처리 시스템과 같은 소프트웨어의 시험에 적용되기는 힘들다.

III. 트랜잭션 처리 시스템의 모델링

본 논문에서 가정하는 트랜잭션 처리 시스템의 동시성 제어 모듈이 제공하는 잠금 모드는 크게 공유 잠금(shared lock)과 배타 잠금(exclusive lock) 등의 두 가지이다. 공유 잠금은 데이터 항목을 읽을 목적으로만 사용함을 의미하기 때문에 동시에 여러 트랜잭션들이 공유 잠금을 획득할 수 있다. 배타 잠금은 데이터 항목을 변경하기 위해서 사용하는 모드로 다른 트랜잭션들과 해당 데이터 항목을 공유할 수 없음을

의미한다.

데이터베이스 상의 데이터는 논리적인 공간에 따라서 다양한 형태로 저장된다. 본 논문에서 가정하는 데이터베이스가 제공하는 논리적인 형태는 필드, 레코드, 파일, 볼륨 등으로 볼 수 있으며 필드가 가장 작은 단위의 데이터 형태이다. 동시성 제어 모듈은 잠금의 대상이 되는 데이터 항목의 형태를 파일과 레코드 수준으로만 한정짓는다.

본 논문에서 가정하는 동시성 제어 모듈이 제공하는 잠금 모드는 S, X, IS, IX, SIX, 등 총 5 가지이다. S와 X 모드는 각각 공유와 배타 잠금 모드에 해당한다. IS와 IX는 파일에 대해서 사용할 수 있는 잠금 모드로 해당 파일 내의 특정 레코드를 공유 또는 배타적으로 사용할 의도가 있음을 알리는 의도 잠금 모드(intention lock mode)이다. SIX 잠금 모드는 S 모드와 IX 모드의 통합형으로 해당 파일에 공유 모드로 잠금을 획득하면서 특정 레코드에 대해서 배타적 잠금을 사용할 의도가 있음을 표시한다. 잠금 관리기에 대해서 시험하기 위해서는 이들 5 가지 잠금 모드를 모두 사용하는 시험 과정을 지원해야 한다. 트랜잭션 처리 시스템은 트랜잭션들 사이의 직렬화 가능 스케줄(serializable schedule)을 보장하기 위해서 2 단계 잠금 규칙(2 phase locking protocol)을 만족시켜야 한다. 따라서, 동시성 제어 모듈에 대한 시험을 하기 위해서는 2 단계 잠금 규칙을 고려해야 한다.

IV. 동시성 제어 모듈의 기능 시험

4.1. 개 념

동시성 제어 모듈의 기능시험은 시험 사례(test case)를 표현하는 시나리오를 기반으로 하여 사용자 간섭을 배제한 상태에서 자동적으로 실행된다. 개발자는 다양한 트랜잭션의 동시성 상황을 시나리오의 형태로 기술하고, 이 시나리오를 기능시험 도구를 통해서 실행한다. 시나리오의 실행은 트랜잭션들이 임의의 데이터베이스 작업을 하도록 하며 동시성 제어 모듈을 수행시키면서 기능시험을 실행한다. 시나리오의 실행이 완료되면 해당 시험사례에 대한 기능시험의 오류 여부를 개발자에게 알려준다.

다음은 동시성 제어 모듈을 시험하기 위해서 고려해야 할 동시성 제어 모듈의 주요 기능이다.

(1) 파일과 레코드 수준의 데이터 항목에 대해서 IS, IX, S, SIX, X 등의 5 가지 잠금 모드가 잘 동작하는지 시험한다.

(2) 파일과 레코드 수준의 데이터 항목에 이미 잠금이 획득되어 있을 때 새로운 잠금을 요청하는 경우 잠금 상승(lock escalation)이 잘 동작하는지 시험한다.

(3) 파일과 레코드 수준의 데이터 항목에 설정된 잠금을 해제하는 경우 다섯 가지 잠금 모드가 잘 동작하는지 시험한다.

제안하는 기능시험 기법을 사용하여 트랜잭션 처리 시스템을 시험하는 전체적인 과정은 그림 1에서 볼 수 있다. 그림에서 실선은 사용자에 의한 작업이며 점선은 시험도구에 의한 작업을 나타낸다.

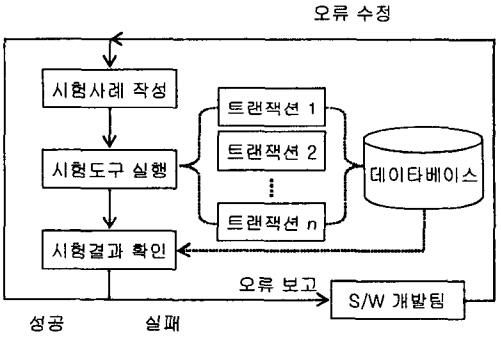

그림 1. 기능시험 과정
Figure 1. Functional Testing Process

시험자는 임의의 시험사례를 작성한 후 시험도구를 통하여 실행한다. 시험도구는 지정된 수만큼의 트랜잭션들로 기능시험을 실행하며 디스크 상의 데이터베이스에 결과 데이터가 저장된다. 시험도구는 데이터베이스를 분석하여 시험사례의 결과가 실패로 판별되면 소프트웨어 개발 팀에 오류를 보고한 후 첫 단계부터 다시 시작한다.

4.2. 시험사례의 형식

동시성 제어 모듈의 시험은 시나리오 단위로 실행되며, 각 시나리오는 트랜잭션들이 데이터베이스에 대해서 실행할 연산들에 대한 스크립트다. 시나리오의 내용인 스크립트를 시험사례(test case)라 하며 기능시험 도구의 입력 데이터로 사용된다. 그림 2에서 시

험사례의 예를 볼 수 있다.

```
trans[10] read[7]
trans[1] update[10, 2-13]
trans[5] write[8, 7]
trans[3] update[3, 9-10]
trans[2] read[10]
```

그림 2. 시험 사례의 예
Figure 2. Example of a Test Case

시험사례의 예제에서 trans, update, write, read 등의 총 4 가지 예약어를 볼 수 있다. 각 예약어는 인자들을 필요로 하며 인자에 의해서 여러 가지 기능이 결정된다. 예를 들어, 첫 번째 줄은 10개의 트랜잭션들이 동시에 읽기 연산을 7번 반복하는 작업을 의미한다. 다음은 각 예약어에 대한 구체적인 설명이다.

- ☐ trans[10] : 시험 과정에서 동시에 실행되는 트랜잭션의 총 수는 trans 예약어의 인자 값의 총합으로 구해질 수 있다. 트랜잭션의 총 수는 입력 파일을 실행하는 과정에서 생성되는 데이터와 실행 종료 후 검증되는 과정에서 중요한 요소가 된다.
- ☐ update[10, 2-13] : 동시성 제어 모듈을 시험할 때 파일에 저장된 레코드가 변경연산의 대상이 된다. 인자인 10은 해당 트랜잭션이 변경 할 레코드의 수이다. 각 트랜잭션은 사용 할 레코드 내에 저장된 필드 값만을 변경한다. 그 다음 인자인 2-13은 레코드 내에 포함된 필드 값을 2에서 13까지 1씩 증가시키는 연산을 나타낸다. 이 연산은 레코드에 저장된 필드 값을 읽어서 1씩 증가하는 변경 기능을 실행한다. 이 연산의 대상이 되는 파일에 대해서는 SIX 잠금 모드가 획득되어야 하고 변경 대상이 되는 레코드에 대해서는 S 잠금 모드에서 X 잠금 모드로 잠금 상승을 필요로 한다.
- ☐ write[8, 7] : 첫 번째 인자인 8은 해당 트랜잭션이 사용 할 레코드의 수이다. 각 트랜잭션은 기록 할 레코드 내에 저장된 필드 값만을 기록한다. write 연산과 update 연산의 차이점은 update 연산은 기존에 저장된 필드 값을 읽어서 1을 증가한 후 다시 동일한 위치에 저장한다. 그러나, write 연산은 단순히 같은 값을 반복적으로 해당 필드 공간에 덮어쓴다. 두 번째 인자인 7은 정수 7을 7번 반복적으로 해당 필드 공간에 기록하는 것을 나타낸다. 이 연산의 대상이 되는 파일에 대해서는 IX 잠금 모드가 획득되어야 하고 변경 대상이 되는 레코드에 대해서는 X 잠금 모드가 획득되어야 한다.
- ☐ read[10] : 해당 트랜잭션을 위한 레코드들을 10번 반복해서 읽기만 하는 연산을 실행한다. 이 연산은 대상이 되는 파일에 대해서 IS 잠금 모드를 필요로 하고 해당 레코드에 대해서 S 잠금 모드를 획득하면 된다.

4.3 기능시험의 검증 방법

(1) 레코드의 구조

시험사례에 의해서 실행되는 트랜잭션은 데이터베이스의 레코드와 파일을 접근한다. 시험 대상이 되는 파일에 저장되는 각 레코드는 그림 3과 같은 구조를 가진다.

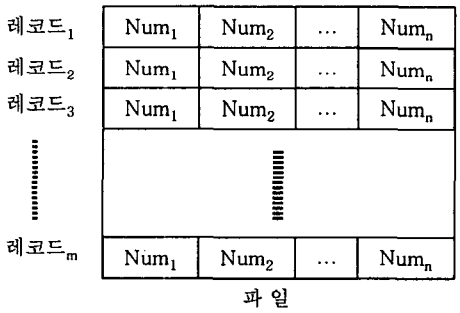

그림 3. 레코드와 파일 구조
Figure 3. Structure of Records and Files

각 레코드의 구성 요소 중 각 Numi는 정수형 숫자를 저장하는 필드며 총 n개로 구성된다. n은 시험사례에서 결정된 트랜잭션의 총 수와 일치한다. 시험과정에서 트랜잭션 i는 해당 레코드 내의 필드인 Numi의 값만을 사용하게 한다.

시험에 필요한 레코드의 수인 m은 시험 사례에 기술된 update, write, read 연산 등에 의해서 사용되는 레코드의 수에 의해서 결정된다. 필요한 레코드의 수는 시험사례의 각 연산에서 필요로 하는 레코드들의

수 중 가장 큰 수로 결정된다. 그림 2의 시험 사례 예제에서 update, write, read 등의 연산에 필요한 레코드 수는 7, 10, 8, 3, 10 등으로 레코드의 수 m은 가장 큰 수인 10으로 결정된다.

(2) 시험 과정

그림 2의 시험 사례는 총 21개의 트랜잭션들을 10개의 레코드들이 저장되어 있는 파일에 대해서 지정된 연산을 실행하게 한다. 21개의 트랜잭션들이 모든 연산을 완료하면 시험 대상이 된 파일 내의 필드 값을 분석하여 시험 사례의 실행이 성공인지 실패인지를 판단한다.

그림 2의 시험 사례를 실행하는 경우 먼저 시험에 필요한 파일을 생성한 후 레코드를 추가해야 한다. 생성한 파일에 10개의 레코드를 추가하고 각 레코드가 21개의 필드로 구성되도록 한다. 레코드가 추가된 후 실제 연산을 실행 할 21개의 트랜잭션들을 생성하여 동시에 실행시킨다.

트랜잭션 i가 update(또는 write) 연산을 실행하는 경우 단순히 지정된 레코드의 각 Numi 값만을 변경(또는 기록)하지 않게 한다. 해당 레코드의 Numi 값만 변경(또는 기록)하도록 하면 이 작업은 주기억장치 내에 저장된 레코드의 Numi 값에 대응하는 공간만 영향을 받는다. 동시성 제어 모듈의 데이터 일관성을 정확하게 시험하기 위해서는 update(또는 write) 연산의 경우 대상이 되는 레코드 전체를 읽어서 Numi 값만 수정하고 변경된 레코드 전체를 한번에 덮어쓰도록(overwrite) 해야 한다. 이와 같은 방식으로 레코드를 변경(또는 기록)하면 이 레코드를 여러 트랜잭션이 동시에 update(또는 write) 연산을 실행하는 경우에 대한 더 완전한 시험을 할 수 있다. 따라서, update(또는 write) 연산 시 레코드 전체를 한번에 덮어쓰는 방식을 사용함으로써 각 잠금 모드에 대한 데이터 일관성을 시험할 수 있다.

(3) 시험 사례의 검증

각 트랜잭션이 해야 할 과정이 시험 사례에 정확하게 표현되기 때문에 기능시험의 검증 과정은 비교적 단순화될 수 있다. 그림 2의 예제에서 트랜잭션 i가 성공적으로 해당 연산을 종료한 경우 모든 레코드의 필드 Numi가 어떤 값을 가져야 하는지를 예측할 수 있다.

Numi 값을 변경하는 update와 write 등의 연산에 대한 최종적인 필드 값만 다음과 같이 검증하면 된다.

① trans[10] read[7] : 레코드 1번에서 7번까지의 Numi (i=1~10) 값을 계속 읽기만 하므로 각 Numi (i=1~10) 값은 변화 없다.

② trans[1] update[10, 2-13] : 레코드 1번에서 10번까지의 각 Num11은 13이 된다.

③ trans[5] write[8, 7] : 레코드 1번에서 8번까지의 Num12에서 Num16 까지는 7이 된다.

④ trans[3] update[3, 9-10] : 레코드 1번에서 3번까지의 Num17에서 Num19 까지는 10이 된다.

⑤ trans[2] read[10] : Num20와 Num21 값을 계속 읽기만 하므로 Num20와 Num21 값은 변화 없다.

시험 사례가 정상적으로 실행이 종료된 후 모든 레코드의 Numi의 값을 확인하여 위와 같은 결과가 나온다면 시험 사례의 실행은 성공했다고 검증할 수 있다.

(4) 직렬화가능 스케줄과 교착상태

동시성 제어 기능은 트랜잭션의 고립성과 일관성을 만족하기 위해서 필요하다. 데이터 일관성을 유지하기 위해서는 트랜잭션들 사이의 수행순서를 직렬화 가능하도록 유지해야 한다. 그러나, 제안하는 동시성 제어 모듈의 시험에 사용되는 레코드의 구조 때문에 직렬화가능 스케줄을 고려할 필요는 없다. 각 트랜잭션이 변경하는 데이터 항목은 공유되는 레코드 내의 각 트랜잭션만을 위한 필드인 Numi 공간에 대해서만 적용된다. Numi 필드 공간은 트랜잭션 i를 제외한 다른 트랜잭션은 읽거나 변경의 대상이 되지 않는 배타적인 사용을 보장한다. 따라서, 직렬화가능 스케줄을 만족시키기 위한 2단계 잠금과 같은 규칙을 고려할 필요가 없다.

잠금을 사용하는 경우 교착상태가 발생할 수 있다. 교착상태가 발생할 수 있는 조건은 트랜잭션이 잠금을 획득한 상태에서 또 다른 잠금을 요청하는 경우이다[8, 9]. 기능시험 프로그램에서는 이와 같은 Hold-and-Wait 상황이 발생하지 않도록 잠금을 소유하고 있는 상태에서 다른 잠금을 요청하지 않도록 설계한다. 따라서, 동시성 제어 기능의 시험 시에는 교착상태가 발생할 수 없다.

4.4 기능시험 도구의 구현

제안한 기능시험 기법을 인텔 CPU 기반의 Linux 시스템과 SUN의 Solaris 8.0 운영체제에서 C 프로그래밍 언어로 구현되었다. 기능시험 도구는 한국전자통신연구원에서 개발한 자료저장 시스템인 MiDAS[4, 5]의 내부 API와 함께 링크되어 트랜잭션 처리 모듈의 기능시험에 적용해보았다. 기능시험을 적용한 결과 MiDAS의 동시성 제어 모듈의 몇 가지 오류를 발견하였으며 대부분의 시험사례가 성공적인 결과를 보여주었다. 또한, 제안한 시험 기법을 동시성 제어 모듈에 대한 압박 시험(stress testing)을 할 수도 있었다. 시험 사례의 트랜잭션 수를 상당히 크게 하고 시험 대상 레코드의 수도 상당히 증가시킴으로써 동시성 제어 모듈의 기능 및 과부하 상황에서의 압박 시험을 실행할 수 있었다.

다음은 그림 2의 시험 사례로 MiDAS에 대해서 기능시험 도구를 실행시킨 예를 보여준다. 시험 사례를 분석하여 21개의 트랜잭션을 실행시키는 과정을 볼 수 있다.

```
0'th child transaction started
0'th Process Finished Successfully
2'th child transaction started
2'th Process Finished Successfully
1'th child transaction started
1'th Process Finished Successfully
3'th child transaction started
4'th child transaction started
4'th Process Finished Successfully
3'th Process Finished Successfully
5'th child transaction started
5'th Process Finished Successfully
.... .... ....
-- the Contents of 8'th Record
[ 0] 7 [ 1] 7 [ 2] 7 [ 3] 7 [ 4] 7 [ 5] 7 [ 6]
7 [ 7] 7 [ 8] 7 [ 9] 7 [10] 13 [11] 7 [12] 7
[13] 7 [14] 7 [15] 7 [16] 0 [17] 0 [18] 0 [19]
10 [20] 10
-- the Contents of 9'th Record
[ 0] 7 [ 1] 7 [ 2] 7 [ 3] 7 [ 4] 7 [ 5] 7 [ 6]
7 [ 7] 7 [ 8] 7 [ 9] 7 [10] 13 [11] 0 [12] 0
[13] 0 [14] 0 [15] 0 [16] 0 [17] 0 [18] 0 [19]
10 [20] 10
-- the Contents of 10'th Record
[ 0] 7 [ 1] 7 [ 2] 7 [ 3] 7 [ 4] 7 [ 5] 7 [ 6]
7 [ 7] 7 [ 8] 7 [ 9] 7 [10] 13 [11] 0 [12] 0
[13] 0 [14] 0 [15] 0 [16] 0 [17] 0 [18] 0 [19]
10 [20] 10
Testing Success !!!
```

위 실행 예에서 10개 레코드들 중에서 마지막 3개 레코드의 내용을 볼 수 있다. 시험 사례에서 요구하는 레코드의 내용과 실제 파일에 저장된 레코드의 내용이 일치되었음을 확인하였기 때문에 시험 사례의 실행은 오류가 없음을 알 수 있다.

V. 결론

본 논문에서는 트랜잭션 처리 시스템의 중요한 모듈인 동시성 제어 기능을 시험하기 위한 트랜잭션 모델과 기능시험 기법을 제안하였다. 다중 사용자 환경에서 동시성 제어 기능을 체계적이고 효율적으로 시험하게 함으로써 DBMS 개발에 필요한 기간을 단축시킬 수 있다. 사용자가 예약어와 인자 값을 적절히 기술하여 동시성 제어 기능을 시험할 수 있는 다양한 상황을 생성할 수 있다. 제안한 기능시험 기법은 시험 사례에 따라서 자동적으로 동시성 제어 모듈을 시험하며 최종적으로 오류가 존재하는지를 사용자에게 보여준다.

향후 연구 계획은 좀 더 다양한 잠금 모드를 시험할 수 있는 모델과 기법을 연구하고 잠금 기법 외의 다양한 동시성 제어 기법에 대한 기능 시험 모델도 제안하고자 한다.

참고문헌

[1] Jorgensen, P.C., "Software Testing", CRC Press, 2002.

[2] Myers, G.J., "The Art of Software Test ing", John Wiley & Sons, New York, 1979.

[3] Boehm, B.W., "Software Engineering Economics", Englewood Cliffs, N.J.:Prentice Hall, 1981

[4] 마경호, "멀티미디어 데이터 처리를 위한 확장 관계 DBMS의 설계 및 구현", 동계 데이터베이스 학술대회 논문집, Vol. 11, No. 1, 1995.
[5] 김명준 외, 데이터베이스 서비스 시스템 개발, 최종 연구 개발 보고서, 한국전자통신연구원, 1997.
[6] P. Bernstein, V. Hadzilacos, and N. Goodman, Concurrency Control and Recovery in Database Systems, Addison-Wesley, 1987.
[7] P.C. Kim, H.I. Choi, Y.J. Lee, S.H. Lee, and M.J. Kim, "MIDAS: Design philosophy and Internals," IPCCC 1992.
[8] J. Gray and A. Reuter, "Transaction Processing: Concepts and Techniques", Morgan Kaufmann, 1993.
[9] A. Silberschatz and P. B. Galvin, "Opera ting System Concepts", 5th Ed., John Wiley & Sons Inc., 1999.
[10] MTS-IV(MiDAS Test Suite) 개발, 최종연구보고서, 한국전자통신연구원, 1999.
[11] J. Ryser and M. Glinz, "SCENT: A Meth od Employing Scenarios to Systematically Derive Test Cases for System Test", Tech nical Report, Inst. fur Informatik, Univ. Zurich, 2000.

저자 소개

홍석희(Seok-Hee Hong)

한국해양정보통신학회 논문지 제6권 제2호 참조