객체지향 DBMS 기능 시험 도구의 프로토타입 개발+

김은영*, 이상호**, 전성택***

Development of an OODBMS Functionality Testing Tool Prototype

Eun Young Kim, Sang Ho Lee, and Sung Taeg Jun

〈요 약〉

본 논문에서는 멀티미디어 처리를 제공하는 객체지향 데이타베이스 시스템에 관한 기능 시험도구의 설계 전략에 대하여 설명한다. 시험도구의 설계시 시험 데이터베이스의 스키마는 추상화, 상속, 집단화(aggregation) 등의 객체 지향적인 성질을 시험할 수 있도록 하며, 포함되는 인스턴스는 사용자의 이해를 돕기 위해 의미있는 값으로 인위적으로 구성한다. 기능 시험항목은 경계값 분석, 동치 분할 등의 기법을 이용한 블랙박스 시험방법으로 작성되며, 각 시험항목은 서로 독립적으로 구현되어 실행 순서에 영향받지 않도록 한다. 시험항목이 수행된 후에는 성공적인 수행, 제외된 기능, 올바르지 않은 수행 등의 시험결과에 대한 목록이 출력되도록 한다. 이러한 설계 전략에 의하여 UniSQL/X를 위한 기능 시험도구를 C++로 개발하였다. 시험도구의 수행은 스키마의 생성, 인스턴스 삽입, 시험프로그램의 컴파일과 링크, 시험프로그램 실행 및 결과확인, 시험 데이타베이스 제거 순으로 이루어진다. 본 시험도구는 UniSQL/C++의 멤버 함수에 대응하는 90여개의 시험프로그램을 이용하여 140여개의 시험항목을 제공한다.

In this paper, we present design philosophy and implementation issues of a functionality testing tool for object-oriented database systems. A testing tool has been developed to validate UniSQL/X functionalities with C++ interface. A testing tool is designed under consideration of scaleability, simplicity and extendibility. The schema is deliberately constructed to verify the object-oriented functionalities such as abstraction, inheritance and aggregation. Each test item has been derived under various black box techniques such as equivalent partitioning and boundary-value analysis. The testing tool consists of six phases, namely, database creation, database population, construction of testindex, compilation and link, execution and result reporting, and final cleanup. The prototype provides more than 140 test items at 90 programs.

+본 논문은 한국전자통신연구소(과제명 : 멀티미디어 DBMS 시험도구 개발)위탁과제 연구비에 의하여 지원받았음.
* 한국오라클, ** 숭실대학교 컴퓨터학부, *** 한국전자통신연구소

1. 서론

데이타베이스 시스템이 상용화되기 시작한 1970년대 이후 계속적으로 그 중요성이 증가해온 데이타베이스의 시험은, 크게 성능 시험과 기능 시험으로 나눌 수 있다. 데이타베이스 시스템의 성능은 트랜잭션을 수행하는 시스템의 속도 등으로 표시되어 다소 그 기준이 분명하고 시험하기도 용이하다.

그러나 제품의 명세(specification)에 기술된 기능을 시스템이 제공하고 있는지를 시험하는 기능 시험은, 시스템마다 상이한 인터페이스로 인해 공인된 시험방법이 존재하지 않고, 각 시스템마다 자체적인 방법으로 시험하고 있다. 이러한 이유로 1980년대부터 활발하게 개발되어진 데이타베이스 시험도구는 성능 시험도구가 주를 이루며, 또한 대부분이 관계형 데이타베이스 시스템을 위한 것이다. 그러나 관계형 데이타베이스는 멀티미디어나 공학(engineering) 등의 처리에 있어 비효율성을 드러냈고, 객체지향 데이타베이스[6,7]가 차세대 데이타베이스로 부각되면서 객체지향 데이타베이스 시스템을 위한 시험도구가 필요하게 되었다. 현재 개발되어 사용되고 있는 몇몇 객체지향 데이타베이스 시험도구는 대부분 성능 시험도구로, 기능 시험은 그 중요성은 인식하고 있으나 일반적으로 공인된 시험방법은 존재하지 않고 있는 실정이다.

본 논문에서는 멀티미디어 처리를 포함하는 객체지향 데이타베이스 시스템의 기능을 시험하는 시험도구를 제시한다. 먼저 관련 연구로서 시험방법론과 기존의 데이타베이스 시험도구들에 대해서 언급한 후, 객체지향 DBMS 기능 시험도구의 설계 및 구현에 대해서 기술하고 결론을 맺는다.

2. 관련 연구

2.1 시험 방법론

시험 단계를 단위(unit) 시험, 통합(integration) 시험, 적합성(validation) 시험, 시스템 시험으로 나눌 때, 본 논문에서 수행하는 데이타베이스 시스템의 시험은 시스템의 기능이 요구사항 명세에 정의되어 있는대로 실행되는 지를 확인하는 적합성 시험에 해당된다.

소프트웨어 적합성 시험은 소프트웨어 기능이 바르게 작동하는지, 즉 입력이 바르게 받아들여지고 출력이 올바르게 생성되는지, 외부 정보의 무결성이 유지되는지를 증명하기 위해 블랙박스 시험(black box testing)을 통해 이루어진다. 시험의 완전성을 보장하는 여러가지의 소프트웨어의 시험 항목 설계 방법은 블랙 박스 시험과 화이트 박스 시험(white box testing)으로 분류될 수 있으며, 블랙 박스 시험은 제품의 기능을 시험하기 위해, 화이트 박스 시험은 제품의 내부적 연산과 내부 구성요소를 시험하기 위해 사용되어진다. 블랙 박스 시험은 최소한의 합리적인 시험 항목 수를 사용하여 최대 한의 오류를 발견하기 위해, 동치분할(equivalence partitioning), 경계값 분석(boundary-value analysis), 원인-결과 그래핑(cause-effect graphing) 등과 같은 기법을 사용한다.

적합성 시험은 시험 계획과 시험 절차로 이루어지며, 시험 계획은 수행되어져야 할 시험 종류에 대한 윤곽을 결정하고 시험 절차는 요구사항에 대한 적합성을 증명하기 위해서 사용되는 상세한 시험항목들을 정의한다. 각 적합성 시험항목이 수행된 후에는 기능 혹은 성능상의 요구사항에 대한

만족여부 리스트가 만들어진다[10].

2.2 데이타베이스 시험도구

데이타베이스 시험도구는 크게 성능 시험도구와 기능 시험도구로 나누어질 수 있으며, 이미 서론에서 언급한 것과 같이 기능 시험은 시스템마다 제공하는 기능과 인터페이스가 다르고, 정확한 기능 측정도 용이하지 않은 관계로 존재하는 대부분의 시험도구는 성능 시험도구이다.

1980년대에 최초의 표준 벤치마크라 할 수 있는 Wisconsin 벤치마크[1,4]를 비롯하여 Debit/Credit, AS3AP, 그리고 1989년 TPC-A 벤치마크를 시작으로 발표된 TPC-B, TPC-C 벤치마크는 모두 관계형 데이타베이스를 위한 성능 시험도구들이다[5]. 특히, 온라인 트랜잭션 프로세싱 환경의 작업들을 수행하는데 필요한 시스템 요소들을 시험하는 TPC 벤치마크들[11,12,13]은 사실상 표준 벤치마크가 되었다. 이러한 관계형 데이타베이스를 위한 성능 시험도구의 발전에 비해 기능 시험도구의 수는 적으며, 대표적인 기능 시험도구인 NIST SQL Test Suite[9]는 완전한 기능의 시험도구라기보다는 SQL문의 적합성을 시험하는 도구이다.

객체 지향 데이타베이스 시스템에 대한 시험도구에 대한 중요성도 점차 증가하여 Engineering Database 벤치마크(OO1)[3]와 OO7벤치마크[2] 등의 객체지향 데이타베이스용 벤치마크들이 개발되었다. Sun 벤치마크이라고도 불리는 OO1 벤치마크는 확장성 있는 데이타베이스 설계와 검색, 탐색, 삽입의 간단한 연산을 사용하여 시스템의 성능을 측정하며, 시험에 대한 결과는 단일한 값으로 나타난다. 이에 비해 OO7 벤치마크는 복잡하고 다양한 데이타베이스를 사용하여, 다양한 측면의 성능을 측정한다. OO7 벤치마크가 측정하는 성능은 OO1 벤치마크와 비교하여 다음과 같은 특성이 있다.

(1) 캐쉬되어 있는 데이타(cached data), 디스크에 존재하는 데이타, 희소 탐색(sparse traversal), 밀집 탐색(dense traversal) 등을 포함하는 다양한 종류의 포인터 탐색(pointer traversal) 속도를 측정한다.

(2) 인덱스되어 있거나 되어 있지 않은 객체 필드(object field)의 갱신, 반복 갱신(repeated update), 희소 갱신(sparse update), 캐쉬되어 있는 데이타(cached data)의 갱신, 객체의 생성과 제거와 같은 여러 종류의 변경(update) 연산의 효율성을 측정한다.

(3) 다양한 형태의 질의에 대한 질의 처리 성능을 측정한다.

OO7 벤치마크의 성능 결과는 단일값이 아닌 여러 값으로 표시되어, OO1에 비해 좀 더 정확하고 현실적인 시험 결과를 제공한다[5]. 객체지향 데이타베이스에 대한 성능 시험도구로는 이외에도 HyperModel 벤치마크, ACOB 벤치마크 등이 존재하나, 기능 시험도구로는 공인되어 사용되는 것이 존재하지 않는다.

3. 기능 시험도구 설계

객체지향 데이타베이스 시스템을 위한 기능 시험도구 설계시 중요하게 고려한 사항은 다음과 같다.

(1) 대상시스템의 객체지향 성질, 멀티미디어 기능을 포함한 모든 기능을 시험할 수 있도록 시

험데이타베이스를 구성한다(domain-specific).

(2) 시험데이타베이스는 다양한 플랫폼(platform)에서 사용이 가능해야 한다(scaleability).

(3) 시험의 내용과 결과를 사용자가 쉽게 이해할 수 있어야 한다(simplicity).

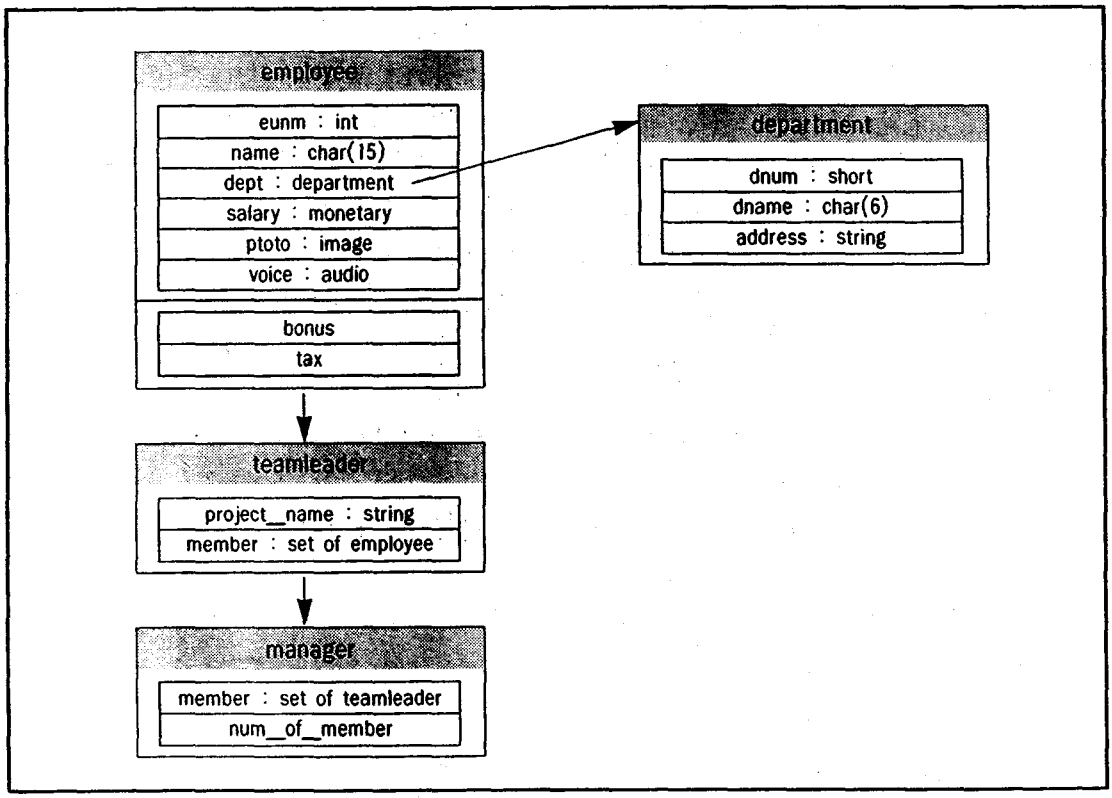
(4) 추가적인 기능을 위한 새로운 시험항목의 첨가가 용이하도록 한다(extendibility).

(5) 시험도구 수행의 각 단계는 서로 독립적으로 구성되며, 시험 프로그램내의 시험항목은 수행에 영향을 미치지 않아야 한다(independence).

3.1 시험 데이타베이스

객체지향 데이타베이스를 위한 기능 시험도구에서 사용할 데이타베이스는 일반화(generalization), 집단화(aggregation), 상속 등의 객체지향 특징을 반영하면서도 간단하여 이해하기 쉽도록 설계하였다. 기본적인 스키마 구조는 〈그림 1〉과 같으며 그 중에서도 employee 클래스와 department 클래스가 기본 클래스가 되어 많은 시험항목에 사용된다.

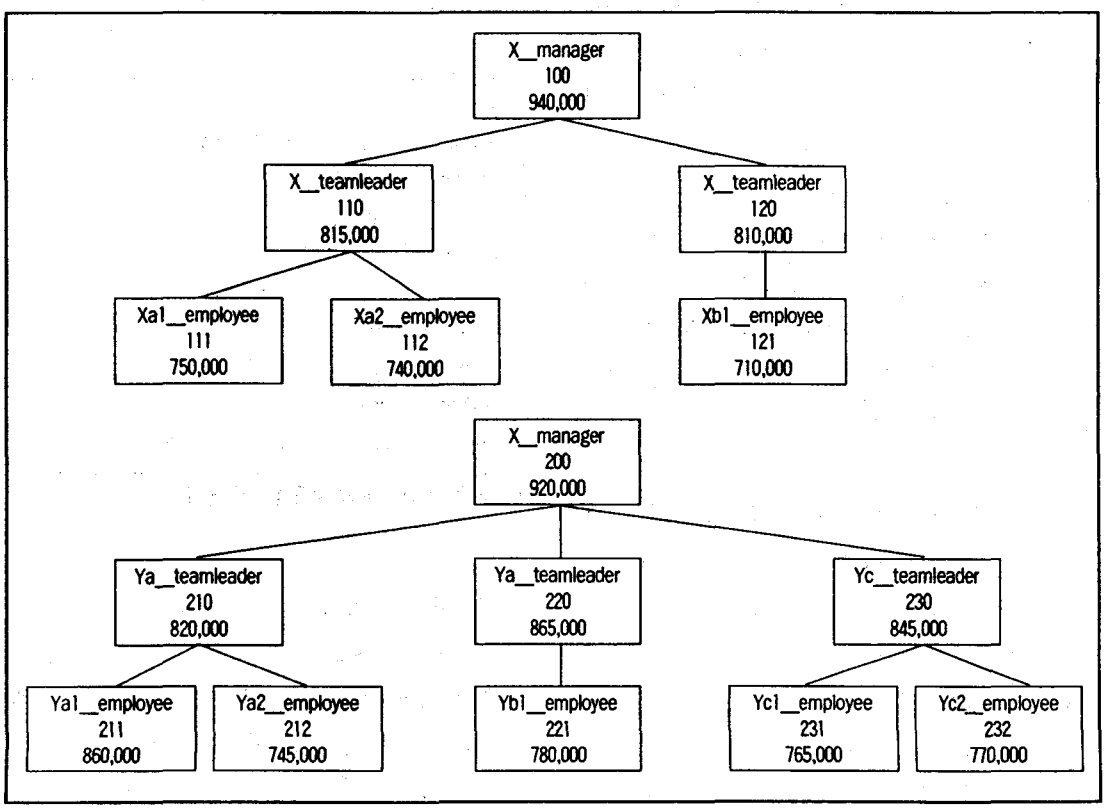
(그림 1)의 스키마를 살펴보면, teamleader 클래스를 employee 클래스의 서브클래스이며 manager 클래스의 수퍼 클래스가 되도록 하여 상속을 시험할 수 있도록 하였다. 집단화(aggregation)의 시험을 위해서는 employee 클래스의 dept 속성의 값의 도메인(domain)을 department 클래스로 지정하였다. teamleader의



(그림 1) 시험도구의 기본 클래스

member는 각 teamleader에 속하는 employee 객체의 집합을 그 값으로 하며, manager의 member는 teamleader 객체의 집합을 포함한다. salary를 기준으로 특정한 달에 지급하는 금액을 나타내는 employee의 bonus와 tax, manager가 관리하고 있는 사원의 형식에 맞추어 출력하는 num_of_member는 메소드(method)이다. 이러한 클래스에 포함하는 객체는, 기능 시험도구가 처리 속도와 같은 성능을 시험하지 않으므로 그 수를 많지 않도록 한다.

employee, teamleader, manager에 속하는 인스턴스들의 name, enum, salary의 값은 (그림 2)와 같이 구성된다. 시험프로그램의 수행 결과에 대한 이해를 돕기 위해, 각 인스턴스는 인위적으로 구성되었다. (그림 2)의 두 트리는 각각 한 부서의 사원들을 나타내며 사원들의 enum의 첫자리가 그 사원이 속한 부서 번호를 나타낸다. 두 트리의 루트 노드는 manager를 나타내며, 다음 레벨의 노드는 teamleader, 마지막 리프(leaf)노드는 employee 객체를 나타낸다. manager들의 번호와 이름을 기준으로 teamleader의 번호와 이름이 결정되며, employee는 자신이 속한 팀의 team leader의 번호와 이름을 기준으로 결정된다. manager의 salary는 900000원 이상이며 teamleader는 800000원대, employee는 700000원대로 구성하였다.



〈그림 2〉 기본 인스턴스의 속성값

3.2 시험항목 설계

시험항목은 주어진 입력 값에 대해 원하는 값이 출력되는지를 검사하는 블랙박스 시험 형태로 이루어지며, 다음과 같은 규칙에 의하여 생성한다.

(1) 멤버 함수의 구문(syntax)내에 포함된 선택사항(option)들의 변화마다 하나의 시험항목을 작성한다.

(2) 명세(specification)에 오류로 지정된 각 사항들에 대해서 하나의 시험항목을 생성한다.

(3) 블랙 박스 기법 중 하나인 동치 분할 기법을 적용해 입력값이 범위(range), 특별한 값(value), 집합(set), 혹은 불리언(Boolean) 조건인가에 따라 적당한 클래스를 정의한 후 그 클래스내의 값을 사용하여 시험항목을 생성한다. 이때 하나 혹은 둘의 유효한(valid) 클래스와 무효한(invalid) 클래스가 정의된다.

(4) 동치 분할 기법에 의해 정의된 클래스내에서 시험항목에 사용할 값의 추출은 경계값 분석을 사용한다. 즉 각 입력 조건에 따라 최대값과 최소값, 그리고 그 경계값보다 크거나 작은 값을 사용하여 시험항목을 정의한다.

이러한 규칙에 의해 작성된 모든 시험항목은 기능의 시험 후에 데이타베이스를 원래의 상태로 환원시켜 시험항목들이 서로 영향을 받지 않도록 한다. 이렇게 시험항목을 서로 독립적으로 작성함에 따라 각 시험항목은 순서에 무관하게 결과를 산출하게 되므로, 원하는 기능의 집합을 임의로 선택하여 시험하는 것이나 새로운 시험항목의 추가 등이 용이해진다.

3.3 시험 결과 보고(report)

기능 시험 후에는 다음과 같은 결과 목록이 산출되도록 한다.

(1) REPORT 1 : 사용자가 시험하고자 한 모든 시험항목

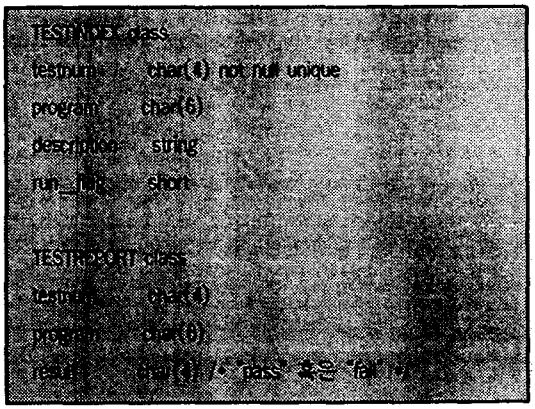
(2) REPORT 2 : 시스템이 수행한 시험항목

(3) REPORT 3 : 성공적으로 수행된 시험항목

(4) REPORT 4 : 바르지 않는 수행으로 인해 실패한 시험항목

(5) REPORT 5 : 시스템이 제공하지 않는 기능의 시험항목

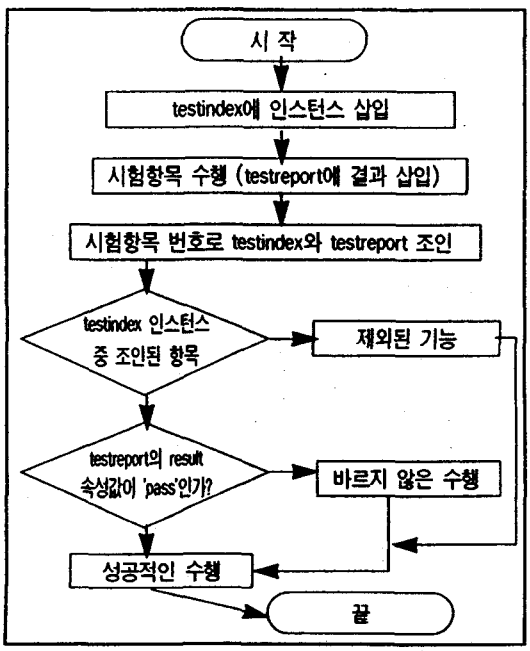
(6) REPORT 6 : 시험결과 요약 (REPORT 1 ~ 5에 포함된 각 시험항목의 수)



〈그림 3〉 시험 결과의 보고를 위한 클래스

이러한 시험 결과의 보고(report)를 위하여 (그림 3)과 같은 testreport, testindex 클래스를 이용한다. testindex 클래스는 모든 시험항목에 관한 정보를 포함하며, 속성은 시험항목 번호(testnum), 시험프로그램 이름(program), 시험항목의 내용(description), 실행여부를 나타내는

플래그(run_flag)로 구성된다. 이 중 run_flag는 사용자가 시험항목을 선택하면 설정되는 속성으로 사용자가 시험하고자 원하는 시험항목(REPORT 1)을 구별하기 위해 사용된다. testreport 클래스는 시험항목이 실행됨에 따라 시험항목의 번호(testnum)와 pass 혹은 fail의 시험 결과값을 가지는 result 등으로 구성하여, 수행된 시험항목(REPORT 2)을 알 수 있도록 한다. 구문오류, 즉 시스템이 제공하지 않는 기능에 대한 시험항목(REPORT 5)은 수행되지 않으므로 testindex의 인스턴스로는 존재하나, testreport의 인스턴스에는 제외된다. testreport의 result 값이 pass인 시험항목(REPORT 3)은 성공적인 수행, fail인 시험항목(REPORT 4)은 올바르지 않은 기능을 나타낸다. 이 두 클래스를 이용하여 시험결과의 목록을 생성하는 알고리즘을 흐름도로 살펴보면 (그림 4)와 같다.



〈그림 4〉 시험 결과 보고 프로그램의 알고리즘

4. 기능 시험도구의 구현

앞에서 언급한 설계를 바탕으로 UniSQL/X 2.4.1 객체지향 데이타베이스 시스템의 적합성을 시험하는 기능 시험도구를 개발하였다. UniSQL/X는 관계형과 객체지향의 성질을 모두 제공하는 데이타베이스 시스템으로, 이미지나 오디오 등의 멀티미디어 처리 기능도 제공한다. UniSQL/X는 객체지향적 SQL (SQL/X), API(application programming interface), C++, ODBC(open database connectivity) 등 여러 인터페이스를 제공하나 본 시험도구에서는 C++ 인터페이스를 사용하며, 단일 사용자 모드(standalone mode)와 클라이언트/서버 모드(client/server mode) 중 클라이언트/서버 모드 하에서 제공되는 기능을 시험하였다.

UniSQL/C++는 내장된 클래스와 그 클래스 안의 멤버 함수를 이용하여 데이타를 관리하며, 본 시험도구에서는 이러한 클래스들을 〈표 1〉과 같이 유사한 기능별로 분류하여 각 목록에 해당하는 시험 프로그램을 개발하였다.

〈표 1〉 UniSQL/C++에서 제공하는 클래스의 분류

내장된 클래스의 종류	클래스의 수	멤버 함수의 수
프로그램 초기화 클래스	2	36
기본 클래스	7	52
질의어 클래스	1	20
스키마 관리 클래스	1	37
권한 클래스	1	9
멀티미디어 클래스	11	73
트리거 클래스	1	18
합 계	24	245

시험도구를 수행시키기 위해서 사용자는 다섯

단계를 거쳐야 한다. 각 단계의 지정된 명령어를 입력하면 자동으로 그 내용이 수행된다. 시험도구를 구성하는 각 단계의 실행 화일 이름과 내용은 다음과 같다.

(1) create_sch : 시험 데이타데이스인 companydb 데이타베이스를 생성한 후 서버를 구동시킨다. 클래스를 정의한 company.sql 화일을 실행하여 클래스를 생성한다.

(2) ins_obj : employee, teamleader, manager, department에 포함되는 인스턴스들를 생성하고 트리거 시험을 위해 check_salary라는 트리거를 생성한다. 인스턴스 생성 후에는 선택(selcet) 연산을 이용하여 데이타베이스가 바르게 생성되었는지 사용자가 확인할 수 있도록 한다.

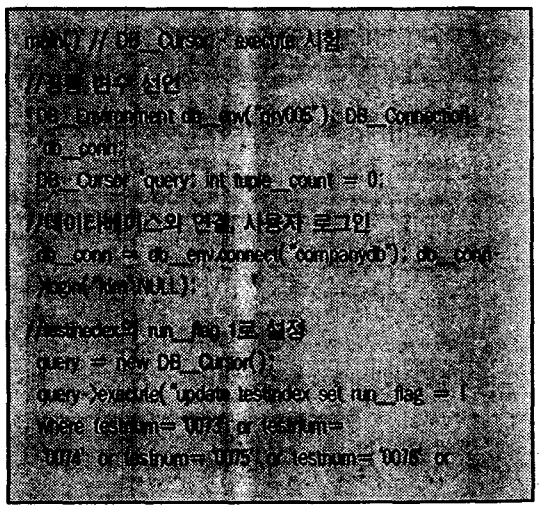
(3) ins_tdx : 전체 시험항목에 대한 정보를 포함하는 testindex 클래스의 인스턴스를 생성한다. ins_obj와 같이 testindex내의 인스턴스를 화면에 출력한다.

(4) compile_link : 시험프로그램을 컴파일하고 링크한다. 전체 프로그램 혹은 임의의 시험프로그램을 선택할 수 있도록 메뉴를 제공한다. UniSQL/X나 UniSQL/C++에서 제공하지 않는 기능으로 인해 컴파일이나 링크가 실패한 시험항목은 화면에 출력된다.

(5) run_test : 실제 사용자가 시스템을 시험하는 단계이다. 사용자는 임의의 원하는 시험프로그램만을 순서에 상관없이 선택할 수 있으며, 원하는 시기에 이제까지 수행된 시험항목들의 결과를 확인할 수 있다. 이러한 사용자의 선택은 메뉴 방식으로 이루어지며, 시험결과는 화면뿐 아니라 화일(file)로도 저장될 수 있다. 결과를 확인한 후에는 시험결과 클래스들의 값이 초기화되어 다음 시험프로그램의 결과를 저장하게 된다.

(6) clean_up : 시험데이타베이스인 companydb 데이타베이스를 제거하고 서버(server)의 구동을 정지(shutdown)시킨다.

〈표 1〉에 나타낸 UniSQL/C++의 각 멤버 함수는 하나의 시험 프로그램에 대응되며, 각 시험프로그램은 3.2의 시험항목 설계에서 설명한 규칙에 의해 생성된 하나 이상의 시험항목들로 구성된다. 실제 개발된 시험프로그램의 예는 (그림 5)와 같다. 그림은 qry005.C 프로그램의 일부분으로 DB_Cursor클래스의 execute 멤버 함수를 시험하며, 포함된 6개의 시험항목 중 계승(inheritance)을 시험하는 시험항목을 중심으로 나타내었다. 시험프로그램은 공통변수 선언부분, 데이타베이스와 연결, testindex의 run_flag 설정, 하나 이상의 시험프로그램, 데이타베이스의 연결 삭제 부분으로 이루어져 있으며, 각 시험항목은 자신만이 사용하는 변수 선언, 시험 수행, 결과 출력, testreport 클래스에 결과 저장으로 구성된다.



```
testnum= '0077' or testnum= '0078' );
db_conn->commit_transaction(); delete query;
//하나 이상의 시험항목
cout << "\n----------test0073----------\n";
// test0073 ~ test0075 시험항목 포함
cout << "\n----------test0076----------\n";
// generalization : contain its subclass test
//각 시험항목에 포함되는 변수 선언
tuple_count = 0;
query = new DB_Cursor();
query->execute("select * from all employee except
teamleader where salary >
700000"); tuple_count = query->row_count(); delete
query;
//시험결과 출력
cout << "expected tuple_count : 10\n";
cout << "     tuple_count : " << tuple_count << endl;
// 시험결과 저장
if(tuple_count == 10) new Testreport("0076","qry005",
"pass");
else new Testreport("0076","qry005","fail");
db_conn->commit_transaction();
//test0077 ~ test0078 시험항목 포함
//----------end test0078----------
// 데이터베이스와 연결 삭제
db_conn->disconnect();
delete db_conn; }
```

(그림 5) 시험 프로그램의 예 (qry005.C)

〈표 2〉 현재 개발된 프로그램과 시험항목의 수

프로그램의 분류	개발된 프로그램	시험항목 수
INI (initialization class)	ini001 ~ ini008	13
BAS (basic class)	bas001 ~ bas029	53
QRY (query class)	qry001 ~ qry013	22
SML (schema management language)	sml001 ~ sml011	18
AUT (authorization class)	aut001 ~ aut006	10
MTM (multimedia class)	mtm001 ~ mtm008	17
TRG (trigger class)	trg001 ~ trg010	10
합 계	90	143

시험프로그램은 시험의 완전성을 위하여 UniSQL/C++에서 제공하는 모든 멤버함수를 시험하는 것을 목표로 하고 있으며, 현재까지 개발된 시험프로그램과 시험항목의 수는 〈표 2〉와 같다.

5. 결론

객체지향 데이타베이스 시스템이 상용화되고 기능 시험도구의 중요성이 인식되고 있으나, 현재 개발자와 소비자를 위한 알려진 객체지향 DBMS의 기능 시험도구가 존재하지 않고 있다. 본 논문에서는 멀티미디어 기능을 포함하는 객체지향 DBMS의 기능 시험도구에 대한 설계전략 및 구현에 대하여 기술하였으며, 객체지향 DBMS의 하나인 UniSQL/X를 대상으로 구현된 객체지향 기능 시험도구의 프로토타입(prototype)을 제시하였다. 현재는 245개의 멤버함수에 대해서 90개의 시험프로그램만이 개발되어 있으며, 완전한 시험도구의 개발을 위해서는 각 멤버함수마다 시험프로그램을 작성하여 추가시켜야 한다.

본 시험도구가 호환성을 유지하기 위해서는 앞으로 객체지향 시스템의 기능과 인터페이스에 대한 표준이 제정되어 요구사항 명세에 대한 적합성이 아닌 표준에 대한 적합성을 시험하여야 할 것이며, 단일사용자가 아닌 다중 사용자를 위하여 동시성 제어, 오류 회복성, 버퍼 관리 등의 보다 완전한 기능을 시험하는 기능 시험도구의 개발이 요구되어진다.

〈참고문헌〉

[1] D. Bitton, D.J. DeWitt, and C. Turbyfill, "A Retrospective on the Wisconsin Benchmark," *Readings on Database Systems*, Morgan Kaufmann, 1988.

[2] M.J. Carey, D.J. DeWitt, and J. Naughton, "The 007 Benchmark," University of Wisconsin Computer Science Technical Report TR-92.99, 1992.

[3] R.G.G. Cattell, and J. Skeen, "Engineering Database Benchmark," In *the Benchmark Handbook for Database and Transaction Processing Systems*, Third Edition, ed. J. Gray, Morgan Kaufmann, 1993.,

[4] D.J. DeWitt, "The Wisconsin Benchmark : Past, Present, and Future," In *the Benchmark Handbook for Database and Transaction Processing Systems*, Third Edition, ed. J. Gray, Morgan Kaufmann, 1993.

[5] J. Gray, *The Benchmark Handbook for database and transaction processing systems*, Third Edition, Morgan Kaufmann, 1993.

[6] W. Kim, *Introduction to Object-Oriented Databases*, MIT Press, 1990.

[7] W. Kim and F. H. Lochovsky, *Object-Oriented Concepts, Databases, and Applications*, Addison Wesley, 1989.

[8] J. McCall, P. Richards, and G. Walters, "Factors in Software Quality," three volumes, NTIS AD-A049-014, 015, 055, Nov. 1977.

[9] National Institute of Standards and Technology, "NIST SQL Test Suite for FIPS 127-2," Version 4.0, Dec. 1993.

[10] R.S. Pressman, *Software Engineering-A Practitioner's Approach*, Third Edition, McGraw-Hill, 1992.

[11] Transaction Processing Performance Council(TPC), "TPC BENCHMARKTM A -Standard Specification," Nov. 1989.

[12] Transaction Processing Performance Council(TPC), "TPC BENCHMARKTM B -Standard Specification," Aug. 1990.

[13] Transaction Processing Performance Council(TPC), "TPC BENCHMARKTM C -Standard Specification," Aug. 1992.

[14] UniSQL, Inc., *C++ interface to UniSQL/X-User's Manual*, 1993.