데이터베이스 시스템을 위한 EBORD 성능 평가 방법론

정 회 진* · 이 상 호**

The EBORD Benchmark for Database Systems

Hoe Jin Jeong* · Sang Ho Lee**

Abstract

The paper presents the EBORD (Extended Benchmark for Object-Relational Databases) benchmark, which is an extension of the BORD benchmark for object-relational databases. The EBORD benchmark is developed to evaluate the database common functions that should be supported in modern database systems. Besides the 36 test queries already defined in the BORD benchmark, totally 22 test queries in five categories are newly defined in order to measure the index-relevant performance issues and database import capabilities. The EBORD benchmark also features scalability, use of a synthesized database, and a query-oriented evaluation. In order to show the feasibility of the proposed benchmark, we implement it with two commercial database systems. The experimental results and analyses are also reported.

Keywords : EBORD, BORD, Database Benchmark, Object-Relational DBMSs

논문접수일 : 2004년 10월 6일 논문게재확정일 : 2005년 5월 20일
※ 본 연구는 숭실대학교 교내 연구비 지원으로 이루어졌습니다.
* 숭실대학교 대학원 컴퓨터학과 박사과정
** 교신저자, 숭실대학교 컴퓨터학부 교수, (156-743)서울시 동작구 상도5동 1-1, Tel : 02-825-1092, e-mail : shlee@comp.ssu.ac.kr

1. 서 론

새로운 데이터베이스 시스템이 개발되거나 기존 데이터베이스 시스템에 새로운 기능이 추가되면 성능 평가 작업을 통해 다른 데이터베이스 시스템과의 비교 우위를 알아보거나 해당 데이터베이스 시스템의 문제점을 찾아낸다. 지금까지 개발된 데이터베이스 시스템 성능 평가로는 Wisconsin 성능 평가[DeWitt, 1993], TPC(Transaction Processing Performance Council) 시리즈 성능 평가[Transaction], OO7 성능 평가[Carey et al., 1993], BUCKY 성능 평가[Asgarian et al., 1997], BORD 성능 평가[Lee et al., 2000] 등이 있다. Wisconsin 성능 평가는 관계형 데이터베이스에 대한 기본적인 성능 평가를 문제 영역으로 다루며, OO7 성능 평가는 객체-지향형 데이터베이스에 대한 성능 평가를, TPC 시리즈 성능 평가는 데이터베이스 시스템의 트랜잭션 처리 능력에 대한 성능 평가를 문제 영역으로 다룬다.

1990년대 초에 UniSQL사와 Illustra사에 의해 객체-관계형 데이터베이스가 소개된 이래 많은 데이터베이스 시스템 공급 업체가 새로운 제품들을 개발하거나 기존 관계형 데이터베이스 시스템에 객체-지향형 데이터 모델의 다양한 기능을 추가하여 객체-관계형 데이터베이스 시스템 시장에 진입하였다. BUCKY 성능 평가와 BORD 성능 평가는 이러한 객체-관계형 데이터 모델에 기반을 둔 데이터베이스 시스템 성능 평가를 다루고 있다. BUCKY 성능 평가는 관계형 데이터베이스 시스템에서 제공하던 기능들이 객체-관계형 데이터베이스 시스템에서는 얼마나 효율적으로 수행되는가에 성능 평가 목적을 두고 있고, BORD 성능 평가의 성능 평가 목적은[Kim, 1996], [Stonebraker, 1996]을 참조하여 객체-관계형 데이터베이스 시스템이 제공하여야 하는 고유 기능을 정의하고, 선택 추출(selection) 질의를 이용하여 객체-관계형 데이터 모델에 의거하여 데이터베이스 시스템의 고유 기능에 대한 성능 평가를 수행하는 것이다.

데이터베이스 시스템은 데이터 모델에 기반을 두는 주요 질의 기능 외에 데이터 관리를 위한 부가적인 기능을 제공하여야 하며, 이에 대한 성능 평가 기준이 요구된다. 예를 들어, 인덱스 생성 및 삭제, 데이터 적재 기능 등은 데이터 모델과는 관련이 없는 기능이나 실제 데이터베이스 시스템 사용 환경에서는 유용하게 활용되는 데이터베이스 기능이다. 요구되는 성능 평가 기준을 위해 주요 질의 기능 외에 유용하게 사용되는 부가 기능들에 대한 성능 평가 방법론이 필요하다. BORD 성능 평가는 각 질의 별 응답 시간외에 분당 수행한 질의 수(queries per minutes, QPM)와 분당 수행한 질의 수 대비 비용 비율(price per QPM, P/QPM)을 성능 평가 측정 요소로 사용한다. 그러나 사용자가 특정 질의 형태들이 자주 사용되는 수행 환경에서 더 좋은 성능을 보이는 데이터베이스 시스템을 선택하기 위해 QPM을 성능 측정 요소로 사용하는 것은 부적합하다. 성능 평가를 구성하고 있는 시험 질의들의 특성 및 중요도를 고려하는 성능 평가 방법론의 개발이 요구된다.

본 논문은 데이터베이스 시스템을 위한 EBORD (Extended Benchmark for Object-Relational Databases) 성능 평가 방법론을 기술한다. EBORD 성능 평가 방법론은 BORD 성능 평가를 확장한 성능 평가 방법론으로서 BORD 성능 평가에 포함되지 않은 3개 영역을 포함하여 총 5개 영역 22개 질의 및 QPM 계산에 대한 새로운 정의를 포함한다. EBORD 성능 평가는 임의로 합성된 데이터(synthesized data) 사용, 데이터베이스 확장성 지원, 질의 기반 성능 평가 등 BORD 성능

평가의 특성을 유지한다.

단일 사용자 환경에서 객체-관계형 데이터베이스 시스템의 성능을 평가하는 EBORD 성능 평가는 BORD 성능 평가가 가진 적용 한계를 극복하고 다양한 사용자 환경에서의 데이터베이스 시스템에 대한 성능 평가 수행이 가능하다. EBORD 성능 평가는 질의 수행 동안 데이터 버퍼에 미리 저장된 데이터를 최대한 활용하도록 하는 웜 스타트 수행 방법을 사용한다. 이는 데이터베이스가 설치된 하드웨어의 재시작(reboot)을 통한 성능 평가 수행이나 대용량 데이터를 읽어 데이터베이스 시스템의 버퍼를 시험과 무관한 데이터로 채우는 콜드 스타트(cold start) 방법과 비교하여 일반 사용자들의 데이터베이스 시스템 수행 환경에 근접된 시험 환경을 구성하기 위함이다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문의 연구 시점이 되는 BORD 성능 평가에 대해 간략히 살펴본다. 3장에서는 EBORD 성능 평가의 설계 원칙, 시험 데이터베이스 구성, 시험 질의를 기술한다. 4장에서는 EBORD 성능 평가를 두 개의 객체-관계형 데이터베이스 시스템을 대상으로 구현하고 시험하여 그 결과를 기술하고, 5장에서 결론을 맺는다.

2. BORD 성능 평가

객체-관계형 데이터 모델에 관한 정확한 개념 및 기능에 관해서 학계나 산업계에서 널리 수용되는 문서 또는 표준안이 없던 때에 객체-관계형 데이터베이스 시스템이 UniSQL사와 Illustra사에 의해 개발되었다. BORD 성능 평가는 서로 다른 기능들을 채택하고 있는 객체-관계형 데이터베이스 시스템간의 성능 평가를 위해 객체-관계형 데이터베이스의 기능을 항목별로 정립하고, 정립된 기능을 평가하도록 개발되었다.

일반적인 대학교 환경을 모델링한 시험 데이터베이스에는 단일 상속만이 존재하고, 상속성의 깊이가 7이며, 최대 참조 홉(hop)의 수가 4인 17개의 클래스로 구성된다. 시험 데이터베이스는 인위적으로 생성된 합성 데이터베이스를 사용한다. 이는 시험 데이터베이스의 확장성과 시험 결과에 대한 검증 작업의 효과성을 높이기 위해서이다. 시험 데이터베이스의 크기는 확장 요소(scale factor)를 통해 변경이 가능하며, 인덱스 구조와 인덱스 생성이 가능한 데이터 형은 데이터베이스 시스템마다 차이가 있기 때문에 사용자가 가능한 많은 인덱스를 생성하는 것을 허용한다. 다양한 인덱스를 지원하는 시스템에게 보다 좋은 평가를 주기 위함이다.

BORD 성능 평가는 15개 영역에 대한 총 36개 질의를 통해 시험을 수행한다. 단일 인스턴스 검색 질의가 2개, 객체 식별자(object identifier, OID) 검색 질의가 2개, 단순 조인 질의가 3개, 클래스 참조에 관한 질의가 4개, 집합 관련 질의가 4개, 사용자 정의 메쏘드에 대해서는 3개의 질의가 수행된다. 또한 기본 연산자(built-in operator) 대 사용자 정의 연산자 비교 질의가 3개, 단일 클래스에 대한 편평화(flattening) 관련 질의가 1개, 일반 추상 데이터 형에 대한 질의가 추상 데이터 형 접근, 메쏘드 수행, 조인 수행 등 3가지 영역으로 나뉘어 총 6개 질의로 구성된다. 지리 정보 시스템에 대한 추상 데이터 형 관련 질의가 조건절에서의 선택도 및 공간 조건 유무, 함수 수행 여부, 조인 수행 여부 등 4가지 영역으로 나뉘어 총 8개 질의가 수행된다.

콜드 스타트 방법을 사용하여 성능 평가 시험이 수행되며, 다중 측정치로 데이터베이스 시스템의 성능을 평가한다. 각 질의별로 10번 수행에 대한 질의 응답 시간의 산술 평균값과 QPM, P/QPM을 성능 평가 측정 요소로 사용한다. 질의 응답 시간의 산술 평균값은 각 질의 별 성능 비교 시 사용하며, QPM과 P/QPM은 데이터베이스

시스템의 전반적인 성능을 비교할 때 사용한다. 특히 P/QPM은 대형 컴퓨터와 개인용 컴퓨터처럼 성능과 가격에 있어 일장일단을 가지는 상황에서 유용하게 사용할 수 있는 성능 측정 요소이며, 이를 활용하여 사용자는 소형에서부터 대형까지의 다양한 컴퓨터 환경에서 성능 평가를 수행하고 그 결과를 비교할 수 있다.

3. EBORD 성능 평가의 설계 전략 및 시험 질의

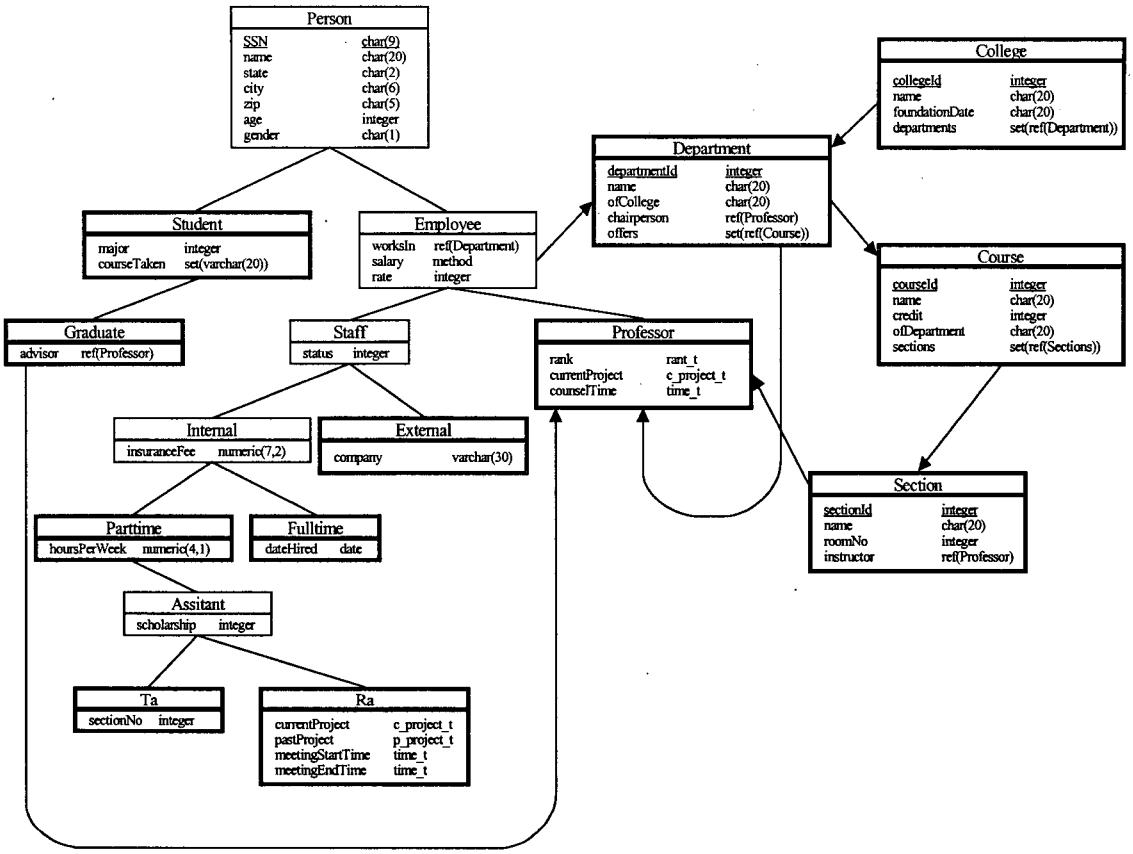
3.1 설계 전략 및 시험 데이터베이스

EBORD 성능 평가의 설계 전략은 BORD 성능 평가의 특성을 유지하는 한편 다양한 사용자 환경에서의 전반적인 객체-관계형 데이터베이스 시스템 성능 평가가 가능하도록 BORD 성능 평가를 확장하는 것이다. 이를 위해 EBORD 성능 평가의 시험 데이터베이스 구성 방법 및 인스턴스 생성 방법은 BORD 성능 평가의 방법을 응용하며, BORD 성능 평가의 적용 한계를 극복하기 위해 질의는 추가 구성한다. 데이터베이스 시스템의 주요 기능이지만 다른 주요 성능 평가 방법론에서는 거의 포함되지 않은 인덱스 생성이나 대량 적재(bulk loading) 등의 시험 질의를 포함하여 사용자가 데이터베이스 시스템의 다양한 기능을 성능 평가할 수 있도록 한다.

시험 데이터베이스로는 실 데이터베이스를 사용하지 않고 인위적으로 생성된 데이터베이스를 사용한다. 실 데이터베이스는 실세계 특정 분야의 데이터 특성을 반영한다는 장점을 가지고 있으나 확장성이 없고, 데이터 분포의 무작위성으로 인해 시험 결과에 대한 검증 작업이 불가능하다는 단점을 가지기 때문에 시험 데이터베이스의 확장과 시험 결과 검증이 필요한 성능 평가에는 사용하지 않는다. 시험 데이터베이스는 순차 분포(sequential distribution), 균등 분포(uniform distribution), 무작위 분포(random distribution)를 갖는 데이터로 구성된다. 순차 분포 데이터는 해당 인스턴스만큼 증가하므로 모든 값이 유일하여 키(key) 특성을 가진다. 균등 분포 데이터는 각 값의 데이터 개수가 균등하므로 일정 개수의 결과를 원하는 선택 추출 질의 수행에 유용하다. 무작위 분포 데이터는 일정한 순서로 정렬된 데이터베이스 인스턴스를 무작위로 분포시키기 위한 목적으로만 사용되며, 시험 질의에는 사용되지 않는다.

시험 데이터베이스의 스키마와 인스턴스는 <그림 1>과 같은 일반적인 대학교 환경을 모델링하고 있다. <그림 1>에서 일반적인 사각형은 인스턴스가 없는 클래스를 나타내고, 굵은 선의 사각형은 인스턴스가 있는 클래스를 나타낸다. 클래스는 이름, 속성, 각 속성의 데이터 형으로 표현된다. 각 클래스에서 밑줄이 있는 속성은 키 속성을 의미한다. 일반적인 직선은 상속 관계를 나타내며, 방향성을 갖는 직선은 참조 관계를 의미한다. 본 성능 평가에서 사용되는 시험 데이터베이스에는 총 17개의 클래스가 있으며 상속성의 깊이는 7이다.

주요한 인스턴스의 분포는 다음과 같다. "person" 클래스의 "SSN" 속성은 주 키 속성으로 중복되지 않는 유일값을 가지는 9자리 문자열로서 앞의 두 자리 문자열은 해당 클래스를 나타내고, 나머지 7자리 문자열은 클래스 내의 유일한 값으로 할당된다. 클래스의 "name" 속성은 스키마 내의 모든 클래스가 가지는 속성으로 무작위 값을 갖는 20자리 문자열로 이루어진다. "city", "state" 속성은 각각 1%와 10%의 키 평균 분포(key average distribution)를 갖는다. 예를 들어, "city" 속성 값이 서로 다른 100개로 균등 분포되어 있다면, 하나의 특정값을 조건으로 하여 선택 질의(selection query)를 수행할 경우 정확하게 1%의 인스턴스가 질의 조건을 만족하게 된다. "age"



〈그림 1〉 시험 데이타베이스 스키마

속성은 속성 값으로 20에서 60까지의 값을 가지며, 인스턴스의 1%는 반드시 60의 값을 가진다. "gender" 속성은 "M"이나 "F" 문자를 속성값으로 가지고, 50%의 키 평균 분포를 보인다. 키 평균 분포 값을 가지는 각 속성들은 특정 비율의 인스턴스 추출이 필요할 경우 사용된다.

. "student" 클래스의 "major" 속성은 "department" 클래스의 "departmentID" 속성을 참조하기 위한 외래키이다. "courseTaken" 속성은 최소 1개에서 최대 20개까지의 원소 개수를 가지는 집합형 데이터형으로, 원소 하나의 크기는 5바이트이다. "gradudate" 클래스의 "courseTaken" 속성은 최소 201개에서 최대 250개까지의 원소 개수를 가지는 집합형 데이터형으로, 원소 하나의 크기는 "student" 클래스에서와 동일하다. 집합형 속성 내의 원소값 분포 또한 질의문의 조건절에서 선택율을 조절할 수 있도록 생성되며, BORD 성능 평가에서만 사용되는 속성이다. "professor" 클래스의 "rank_t", "c_project_t" 자료형이나 "Ra" 클래스의 "time_t", "c_project_t", "p_project_t" 자료형은 BORD 성능 평가의 범용 추상 데이터형(generic abstract data type)에 대한 질의 수행을 위해 사용되는 자료형으로 EBORD 성능 평가에서는 사용되지 않는다.

EBORD 성능 평가는 확장 요소를 사용하여 시험 데이터베이스의 확장성을 제공한다. 확장 요소가 적용되는 각 클래스의 인스턴스 수는 <표 1>과 같고, 확장 요소로는 양의 자연수만 허락된다.

〈표 1〉 시험 데이타베이스 인스턴스(SF : 확장 요소)

클래스	인스턴스 수	클래스	인스턴스 수
Student	100,000 × SF	Professor	30,000 × SF
Graduate	60,000 × SF	External	40,000 × SF
Department	500 × SF	Fulltime	20,000 × SF
College	100 × SF	Parttime	10,000 × SF
Section	60,000 × SF	Ta	50,000 × SF
Course	30,000 × SF	Ra	50,000 × SF

3.2 시험 질의

성능 평가를 위한 시험 질의는 그 기능에 따라 5개 항목으로 분류되며 총 22개로 구성된다. 각 질의는 질의 내용을 설명하는 서술식 기술과 실제 상용 데이터베이스 시스템에서 사용되는 SQL에 기반을 두어 표현한다.

3.2.1 인덱스 생성 및 비 생성 시 단일 클래스 및 클래스 계층에서의 선택 추출

인덱스를 사용하지 않는 경우에는 인스턴스 저장 전략 차이와 테이블 전체 읽기(full table scan) 기능의 효과적인 구현 여부에 따라 선택 추출 성능 차이가 발생한다. 선택 추출 질의의 대상이 되는 클래스가 단일 클래스인지 클래스 계층인지의 여부도 선택 추출 성능에 영향을 미친다. 인덱스를 생성한 경우에는 클래스 계층 여부에 따른 인덱스 사용의 효율성을 시험한다. 인덱스는 'graduate' 클래스의 'SSN' 속성과 'person' 클래스의 'SSN' 속성에 모두 비클러스터된 고유 인덱스(nonclustered unique index)를 생성한다.

Q1-1 인덱스 비 생성 시 단일 클래스에서의 선택 추출 : 대학원생 중에서 주민번호가 '120000050'인 사람의 이름, 도, 도시, 우편번호, 나이, 성별을 검색한다.

```
SELECT name, state, city, zip, age, gender
FROM only(graduate)
WHERE SSN = '120000050' ;
```

Q1-2 인덱스 비 생성 시 클래스 계층에서의 선택 추출 : 모든 사람들 중에 주민번호가 '120000050'인 사람의 이름, 도, 도시, 우편번호, 나이, 성별을 검색한다.

```
SELECT name, state, city, zip, age, gender
FROM person
WHERE SSN = '120000050' ;
```

Q1-3 인덱스 생성 시 단일 클래스 선택 추출 : Q1-1과 동일한 질의를 사용한다.

Q1-4 인덱스 생성 시 클래스 계층 선택 추출 : Q1-2와 동일한 질의를 사용한다.

3.2.2 인덱스 생성 및 비 생성 시 데이터 변경(modification)

삽입(insertion)과 삭제(deletion), 갱신(update) 등 데이터 변경 작업은 데이터베이스 시스템의 기본 기능이다. 인덱스가 생성되지 않은 경우 집합이나 참조와 같은 객체-관계형 데이터 특성을 가지는 인스턴스가 삽입, 갱신, 삭제될 때 데이터 변경 관련 기능에 대한 수행 성능을 비교한다. 인덱스가 생성된 경우에는 데이터 변경에 따른 인덱스 관리가 필요하므로 이에 대한 수행 성능을 평가한다. 인덱스 생성은 질의 대상 클래스인 'graduate' 클래스의 세 가지 속성에 비클러스터된 비고유 인덱스(nonclustered nonunique index)를 생성한다. 삽입 연산인 경우에는 문자 데이터형을 가지는 'name' 속성에 인덱스를 생성하고, 삭제 연산인 경우에는 정수 데이터 형인 'age' 속

성, 갱신 연산인 경우에는 참조 데이터 형인 'advisor' 속성에 인덱스를 생성한다.

삽입 질의와 갱신 질의에 기술된 'professor'는 특정 교수의 OID를 가진다. 특정 인스턴스의 OID 값을 가지기 위해서는 이를 추출하는 선택 추출 질의를 사용하여야 한다. 삽입 질의에 표기된 'professor'는 'SSN' 속성 값이 '020000001'인 교수의 OID를 추출하는 선택 추출 질의로 변경하고, 갱신 질의에 표기된 'professor'는 'SSN' 속성 값이 '020029999'인 교수의 OID를 추출하는 선택 추출 질의로 변경한다. OID를 추출하는 선택 추출 질의는 데이터베이스 시스템마다 기술하는 방법의 차이가 있으므로 성능 평가 구현 시 시험 대상 데이터베이스 시스템에서 정의한 기술 방법에 따라 적절한 질의로 변경한다.

Q2-1 인덱스 비 생성 시 한개 인스턴스 삽입 : 주민번호가 '020000001'인 교수를 지도교수로 하는 새로운 대학원생 한명의 정보를 삽입한다. 대학원생의 주민번호는 '999999999'이고, 이름은 'abcdefghijklmnopqrst'이다. 대학원생은 우편번호가 '9966'인 'S0'도 'city00'시에 살고 있고, 나이가 20세인 여자이다. 학과번호가 59번인 학과에 재학중이고, 'aaaaa' 과목 코드를 가진 과목을 수강한다.

```
INSERT into graduate
VALUES ('999999999', 'abcdefghijklmnopqrst',
        'S0', 'city00', '9966', 20, 'F', 59,
        {'aaaaa'}. professor) ;
```

Q2-2 인덱스 비 생성 시 한개 인스턴스 갱신 : 주민번호가 '120000001'인 대학원생의 지도 교수를 주민번호가 '020029999'인 교수로 변경한다.

```
UPDATE graduate
SET advisor = professor
WHERE SSN = '120000001' ;
```

Q2-3 인덱스 비 생성 시 한개 인스턴스 삭제 : 주민번호가 '120000001'인 대학원생의 정보를 삭제한다.

```
DELETE FROM graduate
WHERE SSN = '120000001' ;
```

Q2-4 인덱스 생성 시 한개 인스턴스 삽입 : Q2-1과 동일한 질의를 사용한다.

Q2-5 인덱스 생성 시 한개 인스턴스 갱신 : Q2-2와 동일한 질의를 사용한다.

Q2-6 인덱스 생성 시 한개 인스턴스 삭제 : Q2-3과 동일한 질의를 사용한다.

3.2.3 인덱스 생성 및 비 생성 시 단일 클래스 및 클래스 계층간 조인

조인 연산은 정보 간에 물리적으로 전혀 연결 관계가 없을지라도 필요 시 논리적인 관계만으로 원하는 정보를 서로 연결하여 참조할 수 있도록 해 주는 데이터베이스 시스템의 기본적이고 중요한 연산이며 큰 수행 비용을 필요로 한다. 조인 연산 수행 시 인덱스 생성 여부에 따라 단일 클래스간 조인과 클래스 계층과의 조인 수행 능력 격차를 알아본다. 인덱스 생성이 필요한 경우에는 'department' 클래스의 'departmentID' 속성에 비클러스터된 고유 인덱스를 생성하고, 'student' 클래스의 'major', 'city', 'state' 속성에는 각각 비클러스터된 비고유 인덱스를 생성한다.

또한 인덱스 생성 여부 및 클래스 계층 여부에 따른 조인 연산 성능 외에 조인 선택율의 변화에 따른 조인 성능을 시험한다. 조인 선택율은 조인 대상 테이블의 전체 인스턴스 수에 대한 조인 참여 인스턴스 수의 비율이다. 시험 질의에 기술된 'city25'는 균등 분포를 가지는 'city' 속성의 값으로 사용자가 'city' 속성의 특정 값을 선택하면 해당 값을 가지는 인스턴스의 수는 전체 인스턴스 수의 1%가 되도록 데이터가 생성되어 있다. 'S5'

는 전체 인스턴스 수의 10%에 해당되는 인스턴스가 선택되도록 균등 분포로 데이터가 생성된 'state' 속성 값이다.

Q3-1 인덱스 비 생성 시 100% 조인 선택율을 적용한 단일 클래스간 조인 : 각 학부 학생의 학생 번호와 학과 이름을 검색한다.

```
SELECT s.SSN, d.name
FROM department d, only(student) s
WHERE d.departmentID = s.major ;
```

Q3-2 인덱스 비 생성 시 100% 조인 선택율을 적용한 클래스 계층과의 조인 : 모든 학생의 학생 번호와 학과 이름을 검색한다.

```
SELECT s.SSN, d.name
FROM department d, student s
WHERE d.departmentID = s.major ;
```

Q3-3 인덱스 생성 시 1% 조인 선택율을 적용한 단일 클래스간 조인 : 'city25' 시에 사는 학부 학생의 학생 번호와 학과 이름을 검색한다.

```
SELECT s.SSN, d.name
FROM department d, only(student) s
WHERE d.departmentID = s.major and s.city
                     = 'city25' ;
```

Q3-4 인덱스 생성 시 1% 조인 선택율을 적용한 클래스 계층과의 조인 : 'city25' 시에 사는 모든 학생의 학생 번호와 학과 이름을 검색한다.

```
SELECT s.SSN, d.name
FROM department d, student s
WHERE d.departmentID = s.major and s.city
                     = 'city25' ;
```

Q3-5 인덱스 생성 시 10% 조인 선택율을 적용한 단일 클래스간 조인 : 'S5' 도에 사는 학부 학생의 학생 번호와 학과 이름을 검색한다.

```
SELECT s.SSN, d.name
FROM department d, only(student) s
WHERE d.departmentID = s.major and s.state
                     = 'S0' ;
```

Q3-6 인덱스 생성 시 10% 조인 선택율을 적용한 클래스 계층과의 조인 : 'S5' 도에 사는 모든 학생의 학생 번호와 학과 이름을 검색한다.

```
SELECT s.SSN, d.name
FROM department d, student s
WHERE d.departmentID = s.major and s.state
                     = 'S0' ;
```

Q3-7 인덱스 생성 시 100% 조인 선택율을 적용한 단일 클래스간 조인 : Q3-1과 동일한 질의를 사용한다.

Q3-8 인덱스 생성 시 100% 조인 선택율을 적용한 클래스 계층과의 조인 : Q3-2와 동일한 질의를 사용한다.

3.2.4 데이터 적재

데이터 적재는 초기 데이터베이스 구축 시 필수 불가결한 요소며 사용자에게 편리함을 제공하는 중요한 기능이다. 사용자가 텍스트 파일 형태로 소유하고 있는 데이터를 데이터베이스에 적재하고자 할 때 가장 효과적으로 사용할 수 있는 방법이 데이터베이스 시스템 개발 업체에서 제공하는 데이터 대량 적재 도구를 사용하는 것이다. 데이터 대량 적재 도구를 활용하지 못하는 경우에는 SQL의 삽입 질의를 사용하여 인스턴스 각각을 적재하여야 하기 때문에 적재 도구를 활용하는 경우에 비해 많은 비용과 시간을 필요로 한다.

새로운 데이터 적재 알고리즘을 개발하고 다른 알고리즘들과의 성능 비교를 수행하여 해당 알고리즘의 우수성을 알리고자 하는 노력[Berchtold et al., 1998 ; Garcia et al., 1998 ; Kamel & Faloutsos, 1993]들은 다차원 인덱스(multidimen-

sional index)와 관련된 분야에서 일부 있었지만, 데이터베이스 시스템에서 제공하는 대량 적재 도구를 사용하여 데이터 적재 기능에 대한 성능 평가를 수행하고자 하는 노력은 미비하였다. EBORD 성능 평가에서는 초기 데이터베이스 구축 시 텍스트 데이터 형태로 준비된 데이터에 기반을 두어 데이터 적재 기능에 대한 성능 평가를 수행한다. 데이터베이스 시스템마다 데이터 대량 적재 도구의 이름이나 사용 방법이 상이하므로 데이터 적재에 관한 질의는 서술식으로만 기술한다.

Q4-1 문자 데이터 형, 정수 데이터 형 및 집합 데이터 속성을 가지는 'student' 클래스에 대해 데이터 적재를 수행한다.

3.2.5 인덱스 생성

인덱스는 질의 수행 시간 단축을 목적으로 생성된다. 지금까지 인덱스 활용 성능이나 인덱스 관리 성능에 대한 평가 노력은 있어 왔지만, 이들 기능에 비해 활용 빈도가 낮은 인덱스 생성에 관해서는 성능 평가가 거의 이루어지지 않았다. EBORD 성능 평가에서는 데이터베이스 시스템 내에서의 데이터 형에 따른 인덱스 생성 성능 차이를 알아보고, 또한 같은 데이터 형에 대해서는 데이터베이스 시스템간의 인덱스 생성 성능 비교를 수행한다.

Q5-1 문자 데이터 형 속성에 인덱스 생성 : 'graduate' 클래스의 'name' 속성에 비클러스터된 비고유 인덱스를 생성한다.

```
CREATE INDEX idx_graduate_name ON
        graduate(name) ;
```

Q5-2 정수 데이터 형 속성에 인덱스 생성 : 'graduate' 클래스의 'age' 속성에 비클러스터된 비고유 인덱스를 생성한다.

```
CREATE INDEX idx_graduate_age ON
        graduate(age) ;
```

Q5-3 참조 데이터 형 속성에 인덱스 생성 : 'graduate' 클래스의 'advisor' 속성에 비클러스터된 비고유 인덱스를 생성한다.

```
CREATE INDEX idx_graduate_advisor ON
        graduate(advisor) ;
```

3.3 성능 측정 방법

EBORD 성능 평가는 단일 사용자 환경에서 각 질의에 대한 데이터베이스 시스템의 응답 시간을 성능 측정 요소로 사용한다. 질의 응답 시간은 1/1000초 단위로 측정되며 양의 소수형(decimal type) 실수로 표현된다. 질의 응답 시간의 측정을 위해 웜 스타트 방법을 사용한다. 본 성능 평가에서는 질의 당 5회를 수행하여 초기 2회 질의 응답 시간을 제외한 나머지 3회의 질의 응답 시간에 대한 산술 평균값을 측정한다.

BORD 성능 평가에서는 콜드 스타트 방법을 사용하여 질의 응답 시간을 측정하였다. 콜드 스타트 방법은 질의 반복 수행 시 데이터베이스 시스템의 데이터 버퍼가 질의 내용과 상관없는 값으로 채워져 데이터베이스 시스템의 데이터 버퍼에 있는 값들이 다음 질의 수행에 영향을 미치지 않도록 하는 방법이다. 하지만, 데이터 버퍼 크기보다 훨씬 큰 데이터를 사용하여 콜드 스타트를 수행해 본 결과 그 효과를 증명할 수 없었으며, 이를 해결하기 위해 질의를 한번 수행할 때마다 데이터베이스가 설치된 장비를 반복하여 재시작하였으나 이는 많은 시험 시간을 필요로 하였다.

EBORD 성능 평가는 질의 응답 시간 외의 성능 평가 측정 요소로 QPM과 P/QPM을 고려한다. QPM은 성능 평가의 결과를 단일 수치로 표현하기 때문에 사용자가 성능 평가의 결과를 이해하기 쉽고 각 시스템과의 비교가 용이하다. 본 성능 평가에서의 QPM에 대한 정의는 BORD 성능 평가에서의 정의와 다르다. BORD 성능 평가

에서는 QPM 계산을 위해 수행된 질의 수를 각 질의 응답 시간에 대한 산술 평균값의 합으로 단순히 나누고 있으나, 본 성능 평가에서는 각 질의에 대한 가중 값을 사용자가 산정한 후 식 (1)을 통하여 소수점 이하 셋째 자리까지 계산한다. 모든 질의에 대한 가중 값의 합은 100이 되도록 산정한다. 'N'은 수행된 전체 질의 수이고, 'i'는 각각의 질의를 의미한다. 'RT_i'는 각 질의 응답 시간의 산술 평균값이며, 'W_i'는 각 질의에 대한 가중 값을 나타낸다.

$$\frac{N}{\sum_{i=1}^{N} RT_i \times \frac{W_i}{100}} \quad (1)$$

본 성능 평가에서의 QPM 계산 방식이 BORD 성능 평가에서의 계산 방식에 비해 다양한 사용자 환경에서 각 환경에 적절하게 수행이 되는 데이터베이스 시스템을 선택하는데 더욱 적합하다. P/QPM은 비용을 고려하여 계산된 QPM 결과를 정규화한 값이다. P/QPM에서의 비용은 5년 동안 하드웨어와 소프트웨어의 구매 및 유지보수에 소요된 모든 비용을 의미한다.

4. 구현 및 시험 결과 분석

본 성능 평가를 두 개의 상용 객체-관계형 데이터베이스 시스템을 대상으로 구현하였다. 구현 목적은 구현 대상이 된 데이터베이스 시스템의 성능 비교가 아니라 EBORD 성능 평가의 구현 가능성을 보이는 것이다. 결과 분석은 앞서 기술한 시험 질의를 기반으로 사용자가 성능 평가를 수행했을 경우 시험 결과를 쉽게 분석할 수 있도록 예를 보이기 위해 수행한다. 시험에 사용된 두 객체-관계형 데이터베이스 시스템의 제품명과 버전은 제품 설치 시 동의한 라이선스(license)로 인해 명시가 불가능하므로, 본 논문에서는 각각의 데이터베이스 시스템을 시스템 X와 시스템 Y로 표기한다.

4.1 구 현

시스템 X와 시스템 Y를 위한 스키마는 본 성능 평가의 사양을 지원하는 정도에 따라 생성되었다. 시스템 Y에서는 객체-관계형 특성 사용을 위해 객체 타입(object type)을 활용하여야 하고 각 테이블은 객체 타입 형태로 선언되어야 한다. 그러나 시스템 X는 클래스 개념 및 클래스 상속 개념을 원활히 지원하고 있어 스키마 생성이 시스템 Y에 비해 수월하다. 시스템 Y에서는 참조 기능의 사용을 위해 참조 범위 지정 여부를 결정해야 한다. 본 성능 평가 구현에서는 범위 지정 참조(scoped reference)를 사용한다. 시스템 Y에서는 같은 객체 타입을 사용하여 생성된 객체 테이블이 복수개가 존재할 수 있지만 그에 비해 참조는 객체 타입에 대해 수행되기 때문에 특정 테이블을 지정하여 참조를 수행해야만 참조 기능의 수행이 효과적으로 이루어진다.

시험 데이터베이스에서 'department' 클래스, 'course' 클래스, 'section' 클래스, 'professor' 클래스는 순환 참조(circular reference)의 관계를 가진다. 순환 참조 관계에 있는 클래스들은 자신의 클래스가 생성되기 위한 선행 조건으로 다른 클래스가 먼저 생성되어야 한다. 따라서 이러한 선행 조건으로 인해 어떤 클래스도 먼저 생성될 수 없다는 문제점을 가진다. 시스템 X에서는 특정 클래스의 특정 속성을 제외한 채 스키마 구성을 수행하고 추후 삭제된 속성들을 추가하는 방법을 사용하여 순환 참조 문제를 해결한다. 즉 가장 먼저 'department' 클래스를 생성하되 'course' 클래스를 참조하는 'offers' 속성과 'professor' 클래스를 참조하는 'chairperson' 속성을 제외한 채 생성한다. 이후 'professor' 클래스, 'section' 클래스,

'course' 클래스를 순서대로 생성하고, 마지막으로 SQL의 'alter' 구문을 사용하여 'department' 클래스에 'offers' 속성과 'chairperson' 속성을 추가한다. 시스템 Y에서는 객체 테이블에 대한 순환 참조 설정이 불가능뿐만 아니라 시스템 X와 같은 방법으로는 순환 참조 문제 해결을 할 수 없기 때문에 시험 질의에 사용되지 않는 'department' 클래스의 'offers' 속성과 'professor' 클래스의 'department' 속성을 삭제하고 순환 참조 관계에 있는 다른 테이블들을 생성한다.

시스템 X에서 인스턴스의 OID는 시스템이 생성하는 임의의 값을 가지고, 시스템 Y에서는 주 키(primary key)의 값을 가진다. 시스템 Y에서 OID 값으로 주 키 값을 가지는 이유는 데이터 적재 성능 평가를 위해 대량 적재 도구를 사용하기 위함이다. 이를 위해 시스템 Y에서의 모든 객체 테이블은 주 키를 가지게 되고 주 키 속성에는 인덱스가 자동으로 생성된다. EBORD의 시험 질의 중에는 인덱스를 사용하지 않아야 하는 질의도 포함되어 있으므로 인덱스를 생성하지 않고 질의를 수행해야 하는 경우에는 SQL의 'alter' 문장을 사용하여 주 키를 무효화(disable) 시킨다.

스키마에 따른 인스턴스 생성은 시스템 X와 시스템 Y에서 제공하는 대량 적재 도구를 사용하였다. 대량 적재 도구의 적절한 사용을 위해 아스키(ASCII) 파일 형식의 데이터 파일을 데이터 생성 도구 수행을 통해 생성하였다. 데이터 파일 내의 데이터는 일정한 형식을 가지게 되는데 각 인스턴스의 구분을 위해 캐리지 리턴(carrage return)이 사용되었고, 속성 값의 구분을 위해서는 공백이 사용되었다. 시스템 X에서 데이터 적재를 위한 제어 정보는 데이터 파일 맨 앞에 기술하였고, 시스템 Y에서는 외부 파일에 기술하고 외부 파일 정보를 적재 도구 사용 시 인자(argument)로 제공하였다. 시스템 X에서는 본 성능 평가에서 제시하는 다수의 제약 조건을 만족하는 데이터 이외에 인스턴스 번호를 추가로 생성하고 데이터 파일에 '참조 클래스 이름|인스턴스 번호' 형태로 데이터 파일에 기술하여 데이터 적재를 위해 사용한다. 시스템 Y에서는 데이터 적재를 위하여 참조하는 인스턴스의 주 키 값을 사용한다.

4.2 시험 결과 및 분석

시험은 펜티엄4 1.6GHz CPU를 사용하는 PC에서 수행되었다. PC의 메모리는 1GB이고, 하드 디스크는 160GB, 30GB 용량을 가지는 두 개로 구성되어 있다. 또한, 와우 리눅스 파란 릴리즈 2를 운영 체제로 사용하고 있다. 성능 평가 시험 도구는 C 언어를 사용하여 각 데이터베이스 시스템에서 제공하는 CLI(call level interface) 프로그래밍 방법으로 작성하였다. 시험에 사용된 데이터베이스는 원시 디스크(raw disk)를 사용하지 않고 리눅스 파일 시스템을 사용하여 저장하였으며, 인스턴스 생성 시 사용된 확장 요소는 1이다. 데이터 저장을 위해 4GB의 디스크 영역을 사용하였고, 인덱스 영역 및 임시 영역으로는 각각 2GB씩을 할당하여 사용하였다.

성능 평가 시험을 위해 시험 대상 데이터베이스 시스템의 데이터 페이지 크기에 관한 시스템 파라미터는 8KB로 설정하였고, 데이터 버퍼 크기는 208MB, 체크 포인트 발생 간격 최소 시간은 30분으로 설정하였다. 서버 프로세스 수행 요청 당 요구되는 개별 정렬 영역의 크기는 512KB로 설정하였고, 로그 버퍼 크기도 같은 크기로 설정하였다. 동시 제어(concurrency control)를 위한 고립화 수준(isolation level)은 'read committed' 수준을 사용하였다. 그 외의 다른 시스템 파라미터들은 시스템 X와 시스템 Y의 표준 설치 시 설정된 기본 값을 그대로 활용하였다. 실

험 환경의 각 값은 특정 목적을 위하여 설정된 값은 아니며, 본 실험이 특정 사용자 환경에서 이루어지는 실험이라 가정하고 해당 환경을 반영하여 설정한 값이다. 앞서 언급한 바와 같이 본 실험의 목적은 구현 대상이 된 데이터베이스 시스템의 성능 비교가 아니라 EBORD 성능 평가의 구현 가능성을 보이는 것이므로 실험 환경으로 설정된 각 특정 값이 중요한 의미를 지니지 않는다.

<표 2>는 인덱스 생성 및 비 생성 시 단일 클래스 및 클래스 계층에서의 선택 추출에 관한 시험 결과이다. 시스템 Y에서는 클래스 계층 질의 수행이 불가능하여 'N/A'(not applicable)로 표에 기술하였다. Q1-1의 시험 결과 비교를 통해 인덱스 비 생성 시 단일 클래스 상에서의 테이블 전체 읽기 기능은 시스템 Y가 시스템 X에 비해 약 145.9배 더 빠르다는 것을 알 수 있다. 인덱스 생성 시에는 그 차이가 줄었지만 시스템 Y가 시스템 X에 비해 약 4.0배 더 빠르다는 것을 Q1-3의 결과를 비교하여 알 수 있다. Q1-1과 Q1-3의 결과를 비교해 볼 때, 두 시스템 모두 단일 클래스 상에서의 인덱스 활용은 원활하게 이루어졌다. 하지만, Q1-2와 Q1-4의 결과는 시스템 X에서 클래스 계층에서의 인덱스 활용은 이루어지지 않았음을 보인다.

〈표 2〉 선택 추출 질의 시험 결과

질의 번호	시스템 X(초)	시스템 Y(초)
Q1-1	5.837	0.040
Q1-2	7.713	N/A
Q1-3	0.004	0.001
Q1-4	7.579	N/A

<표 3>은 인덱스 생성 및 비 생성 시 데이터 변경에 따른 시험 결과이다. 삽입 연산을 나타내는 Q2-1과 Q2-4의 결과를 비교해 볼 때 시스템 Y가 시스템 X보다 약 36.0배 더 빠르게 수행되었고, 갱신 연산(Q2-2, Q2-5)에서도 시스템 Y가 약 120.7배~약 207.2배 더 빠르게 수행되었다. 그러나 삭제 연산에서는 시스템 X가 시스템 Y에 비해 약 1.6배~약 1.7배 더 빠르게 수행되었음을 Q2-3과 Q2-6의 결과를 통해 알 수 있다. 인덱스에 대한 관리 오버헤드는 인덱스 비 생성 시의 수행 시간 대비 인덱스 생성 시 수행 시간의 증가량을 계산하여 측정하였다. 인덱스 관리 성능은 시스템 Y가 시스템 X에 비해 갱신 연산에서 약 1.3배, 삭제 연산에서는 약 2.5배 더 나은 성능을 보였다. 삽입 연산에서는 두 시스템 모두에서 인덱스 관리 오버헤드에 대한 수치를 얻을 수 없었다.

〈표 3〉 데이터 변경 질의 시험 결과

질의 번호	시스템 X(초)	시스템 Y(초)
Q2-1	0.036	0.001
Q2-2	6.008	0.029
Q2-3	5.874	9.685
Q2-4	0.036	0.001
Q2-5	6.035	0.050
Q2-6	5.902	9.795

인덱스 생성 여부에 따른 조인 질의 수행 결과는 <표 4>에 보인다. <표 4>의 질의 결과에 기술된 'N/A'는 시스템 Y가 클래스 계층 관련 질의를 지원하지 않음을 의미한다. 인덱스 비 생성 시 100% 조인 선택율을 적용한 단일 클래스간 조인 질의인 Q3-1의 결과를 보면 시스템 Y가 시스템 X에 비해 약 220.0배 더 나은 성능을 보였다. 시스템 X는 Q3-1 질의를 정렬 합병 조인으로 수행하였고, 시스템 Y는 해시 조인을 사용하였다. 조인 수행 방법은 각 데이터베이스 시스템에서 제공하는 SQL 수행 도구를 사용하여 획득하였다.

〈표 4〉 조인 질의 시험 결과

질의 번호	시스템 X(초)	시스템 Y(초)
Q3-1	406.789	1.849
Q3-2	435.489	N/A
Q3-3	0.178	0.013
Q3-4	6.698	N/A
Q3-5	5.777	0.061
Q3-6	19.699	N/A
Q3-7	409.272	2.293
Q3-8	438.125	N/A

Q3-3, Q3-5, Q3-7 질의의 수행 결과 비교를 통해 조인 선택율 증가에 따른 질의 응답 시간의 변화를 알 수 있다. 두 시스템 모두 조인 선택율 증가에 따라 응답 시간이 증가하였으나 시스템 Y가 시스템 X에 비해 응답 시간 증가 비율이 낮아 약 13.7배~약 178.5배 더 나은 성능을 보였다. 인덱스가 조인 대상 클래스에 모두 생성된 경우에도 시스템 X는 클래스 계층 조인 질의 수행 시 인덱스를 적용하여 질의를 수행치 못하고 테이블 전체 읽기로 수행하였다. 단일 클래스만 사용된 조인의 경우에는 조인 대상 클래스 모두에 대해 인덱스를 정상적으로 적용하여 수행하였다.

<표 5>는 데이터 적재에 관한 시험 결과를 나타내고, <표 6>은 인덱스 생성 질의에 대한 시험 결과를 나타낸다. 데이터 적재 기능은 시스템 X가 시스템 Y에 비해 약 2.7배 더 나은 성능을 보인다. Q5-1, Q5-2, Q5-3의 시험 결과 비교를 통해 인덱스 생성 성능은 시스템 Y가 시스템 X에 비해 약 10.9배~약 20.1배 더 우수하다는 사실을 알 수 있다. 시스템 X에서는 문자 데이터 형이 다른 데이터 형에 비해 인덱스 생성 시간이 약 5% 더 걸린 것을 제외하면 전반적으로 데이터 형에 관계없이 비슷한 인덱스 생성 시간을 보였다. 시스템 Y에서는 정수 데이터 형에 대한 인덱스 생성 시간이 다른 데이터 형에 대한 인덱스 생성 시간에 비해 약 45% 더 적게 걸렸다.

〈표 5〉 데이터 적재 시험 결과

질의 번호	시스템 X(초)	시스템 Y(초)
Q4-1	169.927	463.622

〈표 6〉 인덱스 생성 질의 시험 결과

질의 번호	시스템 X(초)	시스템 Y(초)
Q5-1	13.564	1.197
Q5-2	12.759	0.634
Q5-3	12.593	1.152

QPM은 성능 평가에 기술된 총 질의 수에 기반을 두어 계산한다. 하지만, 시험 대상 데이터베이스 시스템마다 수행된 질의 수에 차이가 있으므로 QPM 비교의 공정성 확보를 위해 대상 데이터베이스 시스템에서 동시에 수행된 질의만을 대상으로 한다. BORD 성능 평가에서 정의한 QPM을 계산하면 16개 질의에 대해서 시스템 X에서는 17.677분 동안 수행하여 0.905개가 되고, 시스템 Y에서는 8.174분에 수행하여 1.957개가 된다. 시스템 Y가 시스템 X에 비해 분당 수행 가능한 질의 수가 약 2.2배 더 많다.

〈표 7〉 각 질의 별 가중 값

질의 번호	가중 값	질의 번호	가중 값
Q1-1	11	Q3-1	21
Q1-3	4	Q3-3	4
Q2-1	16	Q3-5	4
Q2-2	4	Q3-7	4
Q2-3	4	Q4-1	4
Q2-4	4	Q5-1	4
Q2-5	4	Q5-2	4
Q2-6	4	Q5-3	4

EBORD 성능 평가에서는 QPM을 계산하기 위

해 각 질의에 대한 가중 값 선정이 필요하고 이를 위해 사용자가 <표 7>과 같이 정했다고 가정한다. 이는 인덱스를 생성하지 않은 수행 환경에서 100% 조인 선택율 적용 조인 질의가 가장 많이 사용되고, 그 뒤를 이어 삽입 연산과 선택 추출 질의가 기술한 순서대로 높은 사용 빈도를 가지며, 나머지 질의들은 비슷한 사용 빈도를 가지는 경우를 가정한 것이다.

각 질의별 가중 값 역시 비교 대상 데이터베이스 시스템에서 동시에 수행된 질의에 대해서만 산정하며, 이들 가중 값의 합이 100이 되도록 한다. 시험 대상 데이터베이스 시스템에 대해 수행되지 않은 질의가 있는 경우 해당 질의의 가중 값을 산정하지 않은 이유는 수행되지 않은 기능이 사용자 환경에서 중요한 기능이라면 사용자가 해당 데이터베이스 시스템을 선택하지 않을 것이고, 사용자 환경에서 중요한 기능이 아니라면 해당 기능의 성능 차가 데이터베이스 시스템을 선택하는데 영향을 미치지 않기 때문이다. 앞서 정의한 식 (1)에 의하여 시스템 X에서는 QPM이 8.572개, 시스템 Y에서는 48.159개로 계산된다. 시스템 Y가 시스템 X에 비해 분당 수행 가능한 질의 수가 5.6배 많아 BORD 성능 평가에서 정의한 QPM 계산 방식보다 주어진 사용자 환경에 더욱 적합한 데이터베이스 시스템을 선택하는데 유용하게 사용될 수 있음을 알 수 있다.

5. 결 론

기존 BORD 성능 평가는 인덱스 생성에 관한 무원칙 하에 객체-관계형 데이터베이스의 고유 기능에 대한 평가를 목적으로 개발되었다. 본 논문에서는 기존 BORD 성능 평가를 확장한 EBORD 성능 평가를 기술하였다. EBORD 성능 평가는 BORD 성능 평가 특성을 유지하면서 질의 영역 및 시험 질의를 추가하여 확장하고 QPM 측정 요소의 계산 방식을 개선하는 설계 전략에 기반을 두고 있다. 일반적인 대학교 환경을 모델링한 시험 데이터베이스를 대상으로 총 5개 영역에 대한 22개 질의로 구성된다. 또한 본 논문에서는 두 개의 상용 객체-관계형 데이터베이스 시스템을 대상으로 EBORD를 구현하여 구현 가능성을 보였다.

구현 대상 데이터베이스 시스템 중 한 시스템이 클래스 계층 질의를 지원하지 않아 본 성능 평가에서 제안한 모든 질의에 대한 결과를 보이지는 못했지만, BORD 성능 평가에서 다루지 않은 성능 평가 영역 및 질의를 추가하여 다양한 질의 영역에서의 성능 평가 결과를 도출하였다. 사용자가 지정한 질의 별 가중 값을 가정하여 QPM을 계산한 예를 통해 다양한 사용자 환경에서 사용자가 자신의 업무 영역에 적합한 객체-관계형 데이터베이스 시스템을 선택하는데 새로운 QPM 계산 방법이 도움이 될 수 있음을 보였다.

EBORD 성능 평가 방법론의 개발은 데이터베이스 시스템 개발 업체 중 한 곳의 요구 사항 분석을 거쳐 시작되었으며, 이후 관련 문헌 분석과 시행착오를 통해 질의 영역이나 시험 질의를 정의하였다. EBORD 성능 평가는 실제 데이터베이스 시스템 개발 업체에서의 사용을 위해 개발되어 단순히 연구 목적을 위해 개발된 성능 평가들과는 달리 실 사용자의 요구 사항을 반영한다는 장점을 가진다.

음성 데이터, 이미지 데이터, 비디오 데이터 등과 같은 멀티미디어 데이터는 오늘날의 데이터베이스 시스템에서 흔히 지원된다. 또한, 웹의 발달로 인해 전역 문헌 검색(full text search)의 필요성이 증대되고 사용자는 데이터베이스 시스템에서 이들 비구조적 문헌에 대한 검색 능력을 요구하고 있다. 상용 객체-관계형 데이터베이스 시스템에는 현재 관련 기능들이 추가되었거나 추가되고 있다. 객체-관계형 데이터베이스 시스템에서의 멀티미디어 데이터와 관련된 기능 및 전역 문헌 검색에 관한 성능 평가를 향후 과제로 남긴다.

참 고 문 헌

[1] Asgarian, M., Carey, M.J., DeWitt, D.J., Gehrke, J., Naughton, J.F. and Shah, D.N., "The BUCKY Object-Relational Benchmark", Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, 1997, pp. 135-146.

[2] Berchtold, S., Bohm, C. and Kriegel, H.P., "Improving the Query Performance of High-dimensional Index Structures by Bulk Load Operations", Proceeding of the International Conference on Extending Database Technology, 1998, pp. 216-230.

[3] Carey, M.J., DeWitt, D.J. and Naughton, J.F., "The OO7 Benchmark", Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, 1993, pp. 12-21.

[4] DeWitt, D., "The Wisconsin Benchmark : Past, Present, and Future", In : *The Benchmark Handbook 2nd Ed.*, J. Gray Ed., Morgan Kaufmann, 1993, pp. 269-315.

[5] Garcia, Y.J., Lopez, M.A. and Leutenegger, S.T., "A Greedy Algorithm for Bulk Loading R-trees", Proceedings of the 6th ACM Symposium on Advances in GIS, 1998, pp. 163-164.

[6] Kamel, I. and Faloutsos, C., "On Packing R-trees", Proceedings of the International Conference on Information and Knowledge Management, 1993, pp. 490-499.

[7] Kim, W., *Completeness Criteria for Object-Relational Database Systems*, UniSQL Inc., 1996.

[8] Lee, S.H., Kim, S.J. and Kim, W., "The BORD Benchmark for Object-Relational Databases", Proceedings of the 15th International Conference on Database and Expert Systems Applications, 2000, pp. 6-20.

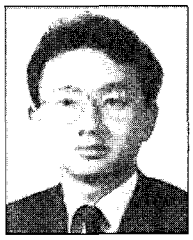
[9] Stonebraker, M., *Object-relational DBMSs*, Morgan Kaufmann, 1996.

[10] Transaction Processing Performance Council, http://www.tpc.org/.

□ 저자소개

정 회 진

저자는 현재 숭실대학교 대학원 컴퓨터학과 박사학위 취득 예정이며, 전주우석대학교 전산학과 이학사(1993), 숭실대학교 컴퓨터학과 공학석사(1995) 학위를 취득하였다. 2000년 6월까지 (주)핸디소프트에서 선임연구원으로 재직하며 그룹웨어 및 정보시스템 개발 업무를 수행하였다. 또한 2002년 3월부터 2003년 2월까지 숭실대학교 정보미디어연구소 전임연구원으로 재직하였다. 주요 관심 분야는 데이터베이스 시스템 성능 평가 및 튜닝이다.



이 상 호

저자는 현재 숭실대학교 컴퓨터학부 교수로 재직중이며, 서울대학교 전산공학과 공학사(1984), 미국 노스웨스턴대 전산학과 공학석사(1986), 공학박사(1989) 학위를 취득하였다. 1990년부터 1992년까지 한국전자통신연구원에 재직하였으며, 1999년에서 2000년까지 미국 조지메이슨대 소프트웨어정보공학과 교환교수로 근무하였다. 관심분야는 인터넷 데이타베이스, 데이타베이스 시스템 성능 평가이다.

◈ 이 논문은 2004년 10월 6일 접수하여 1차 수정을 거쳐 2005년 5월 20일 게재확정되었습니다.