□ 기술해설 □

구현 사례

SRP(SUN RDB Platform)에서 SOP(SNU OODB Platform)까지
—관계형 및 객체 지향 DBMS의 개발[1]—

서울대학교 안정호 · 김형주*

목 차

1. 서 론

1950년대까지 모든 데이타 처리(data processing) 작업은 운영체제가 제공하는 화일 시스템 상에서 이루어졌다. 그러나 응용 시스템의 크기와 복잡도가 증가함에 따라서 이제는 대부분의 데이타 처리 응용 프로그램은 모두 데이타베이스 관리 시스템 상에서 구현되고 있다. 데이타베이스 관리 시스템은 데이타의 공유는 물론 데이타의 일관성과 무결성을 보장하며, 물리적 장치나 저장 구조와의 독립성을 제공함으로써 가장 중요한 소프트웨어 구성 요소 중 하나가 되었다.

초기의 데이타베이스 관리 시스템은 IMS, System 2000 등으로 대표되는 계층형 데이타베이스 관리 시스템(hierarchical DBMS)과 IDMS, TOTAL 등의 네트워크 데이타베이스 관리 시스템(network DBMS)이 주도를 하였다. 이후 IBM의 System/R과 Berkeley 대학의 Ingres로부터 시작된 관계형 데이타베이스 관리 시스템(relational DBMS)은 튼튼한 이론적 기반과 선언적 질의(declarative query)를 통한 사용의 편리함을 바탕으로 오늘날 거의 모든 사무 응용 환경에서 사용되고 있다.

그러나 컴퓨터 기술의 발전에 따라 등장하게 된 새로운 응용 분야에서는 더 이상 기존의 데이타베이스 관리 시스템이 적합치 않게 되었다. 흔히 차세대 응용 분야라고 불리는 CAD/CAM, CASE, OIS(office information system), 멀티미디어 시스템, 지식 데이타베이스 등에서는 과거 사무 중심의 응용에서 사용하고 있는 단순하고 고정된 형태의 데이타만을 요구하는 것이 아니라 복잡한 형태의 데이타와 연산을 요구하고 있다.

90년대에 이르러서 객체지향 패러다임은 거의 모든 컴퓨터 분야에서 사용되고 있으며, 데이타베이스 분야도 그 예외는 아니다. 객체지향 데이타베이스 관리 시스템은 풍부한 데이타 모델을

1) 상공부 공기반 과제 "대규모 방송정보시스템의 설계 및 구현" 및 공업기반기술개발 사업 위탁 기술개발 과제 "Object Manager 개발"의 연구비 지원에 의한 것임.

* 종신회원

기반으로 복합 객체(complex object), 버전 제어(version control), 스키마 변경(schema evolution), 장기간 트랜잭션 관리(long duration transaction management) 등을 제공함으로써 과거 관계형 데이타베이스 관리 시스템의 한계를 극복하고자 하고 있다.

지금까지 국내에서는 주로 관계형 DBMS를 중심으로 연구가 진행되어 왔는데 한국 과학기술원의 IM을 비롯하여 ETRI 주관의 BADA와 삼성의 CODA, 대우의 한바다 등을 예로 들 수 있다. 반면에 객체지향 DBMS 분야에서는 아직 프로토타입 정도만이 개발되어 있는 형편이다. 한편 국외에서는 1980년대 중반부터 미국 대학과 연구소 등을 중심으로 객체지향 DBMS에 관한 연구가 활발히 이루어졌으며, 대표적인 예로 ORION, O2, Gemstone, ObjectStore 등이 있다.

서울대학교 객체지향 시스템 연구실에서는 단일 사용자용 관계형 DBMS의 개발을 완료하였으며 이를 다중 사용자화 하고 있다. 그리고 관계형 DBMS 개발의 경험을 밑거름으로 하여 새로운 응용 분야를 지원하기 위한 객체지향 DBMS의 개발을 시작하였다

본 논문에서는 현재 객체지향 시스템 연구실에서 진행 중인 SRP(SNU Relational DBMS Platform) 과제와 SOP(SNU Object-Oriented DBMS Platform) 과제를 소개하고자 한다.

2. SRP—관계형 DBMS의 개발

본 연구실에서 개발된 관계형 DBMS는 크게 저장 시스템과 질의 처리기 모듈로서 구성되어 있다. 현재는 단일 사용자 버전이 완성되었으며, 저장 시스템의 다중 사용자화와 ESQL 처리기의 개발이 진행 중에 있다. 본 절에서는 지금까지 개발된 저장 시스템과 질의 처리기 모듈의 구조 및 특징과 함께 앞으로의 개발 계획을 소개하겠다.

2.1 저장 시스템(SESS: SNU Extensible Storage System)

본 연구실에서 개발한 저장 시스템 SESS(Snu Extensible Storage System)는 객체지향 개념을 사용하여 설계 구현한 확장 용이 저장 시스템이다. 즉, 새로운 응용이 등장함에 따라 요구되는 새로운 화일 구조를 쉽게 추가할 수 있으며 버퍼 관리 알고리즘이나 페이지 관리 알고리즘 등의 내부 구조도 쉽게 변경하거나 추가할 수 있는 저장 시스템이다. 이러한 저장 시스템의 특성은 확장성과 재사용성을 가장 큰 장점으로 하고 있는 객체지향 개념을 직접 저장 시스템의 설계와 구현에 사용함으로써 얻은 결과이다.

2.1.1 SESS의 구조

저장 시스템(storage system)에 대한 요구는 특정 키(key)의 순서에 따라 레코드를 순차적으로 검색하거나 삽입, 삭제하는 연산들로 이루어진다. 이를 위하여 저장 시스템은 인덱스를 유지 관리하는데, 하나의 데이타 화일에 대해서 여러 다른 키로서 인덱스를 생성할 수 있도록 인덱스 화일은 데이타 화일과 별개의 화일로 구성한다. 그리고 각 데이타 화일이나 인덱스 화일 등은 모두 여러 개의 블럭 또는 페이지로써 구성된다. 페이지는 디스크 장치와 주 기억 장치 사이의 데이타 이동 단위이며, 저장 공간의 할당과 반환의 기본 단위가 된다.

이에 따라 저장 시스템에서의 화일에 대한 관점은 다음과 같이 3 단계로 구분할 수 있다. 즉, 가장 하위 단계에서의 화일에 대한 관점은 화일을 단순히 페이지들의 집합으로서 보는 것이다. 따라서 이 단계에서는 페이지의 내부 구조나 연결 순서 등은 나타나지 않는다. 두번째 단계에서의 화일은 B-tree 구조나 해쉬(hash) 구조와 같이 일정한 데이타 구조를 이루는 레코드들의 집합으로서 존재한다 따라서 이 두번째 단계에서의 화일에 대한 연산은 페이지 단위가 아닌 레코드 단위로 이루어진다. 세번째 단계에서의 화일에 대한 관점은 이 두 데이타 화일과 인덱스 화일을 결합한 가상의 인덱스된 화일(indexed file)로서 보는 것이다. 이로서 사용자는 하나의 연산을 통해서 데이타 화일과 인덱스 화일에 대해 동시에 작업을 할 수 있는 것이다.

SESS는 이와 같은 3단계 화일 관점에 대칭하여 그림 1과 같은 3단계 구조를 갖는다.

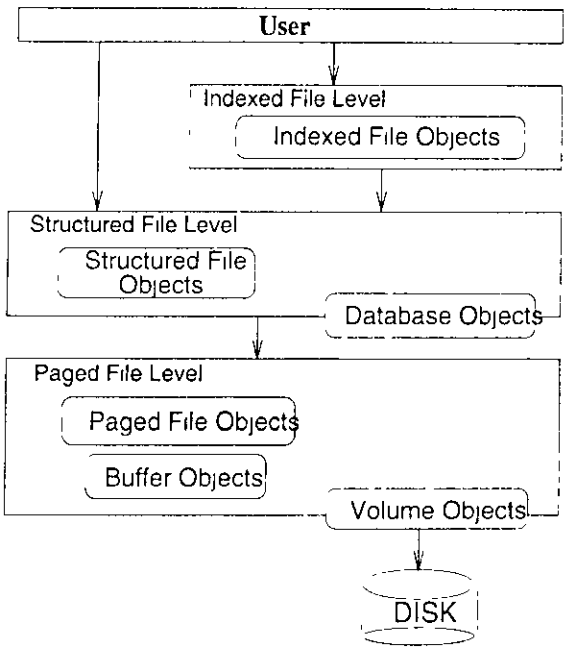


그림 1 SESS의 전체 구조

SESS의 각 단계는 각 단계에 해당하는 화일에 대한 관점을 그 상위 단계에 보여 준다. 먼저 페이지 화일 단계(paged file layer)는 페이지 화일 관점을 제공하는 단계로 페이지의 할당 및 반환과 디스크와 주기억장치 사이의 페이지 이동을 담당한다. 페이지 화일 단계는 볼륨 객체, 버퍼 풀 객체, 페이지 화일 객체 등으로 구성된다. 볼륨 객체는 디스크 장치를 연속된 페이지들의 집합으로서 추상화시키는 객체로 실제 물리적 입출력을 담당하는 디스크 객체와 화일 기술 목록(file descriptors)을 다루는 객체, 페이지 맵(page map)을 관리하는 객체 등으로 구성된다. 버퍼 풀 객체는 특성 버퍼 관리 알고리즘에 따라서 버퍼 풀을 관리한다. 이때 버퍼 관리 알고리즘은 치환(replacement)에 관한 부분만을 제외하고는 모두 일반화하여 구현함으로써 새로운 알고리즘의 추가를 용이하게 하였다. 페이지 화일 객체는 페이지 화일 단계의 인터페이스 객체로 상위 단계에 페이지 화일 관점에서의 화일 연산을 제공한다. 이때 페이지 화일 객체는 특정 버퍼 풀을 지정할 수 있도록 함으로써 화일의 종류에 따라서 적합한 버퍼 풀을 사용할 수 있도록 하였다 예를 들면 로그를 저장하기 위한 화일은 FIFO 알고리즘을 사용하는 특정 버퍼를 사용할 수 있으며, 또한 데이타 화일과 시스템 화일의 버퍼를 별도로 관리할 수도 있다.

다음 구조화된 화일 단계(structured file layer)는 다양한 화일 구조를 갖는 화일 객체들과 데이타베이스 객체들로 이루어진다. 먼저 구조화된 화일 객체는 페이지 위에서 B-트리나 해쉬와 같은 레코드 단위의 데이타 구조를 구현함으로써 페이지 화일을 논리적 데이타 구조를 갖는 구조화된 화일로써 만들어 준다 이 단계는 확장성이 가장 요구되는 단계라 할 수 있는데, 이 확장성은 그림 2와 같은 페이지 클래스 계층 구조를 통해 얻는다. 각 페이지 객체는 한 페이지 내에서의 레코드 삽입/삭제 등의 연산을 제공한다. 따라서 구조화된 화일 객체는 페이지 화일 객체가 제공하는 페이지를 특정 페이지로서 객체화하여 바라보고 연산을 행하며, 페이지 사이의 연결을 유지 관리한다. 그런데 많은 화일 구조가 페이지 내에서의 연산은 동일하거나 유사하기 때문에 새로운 화일 구조의 첨가는 페이지간의 연결만을 다루어 줌으로써 추가를 쉽게 한다 그리고 데이타베이스 객체는 페이지 화일 단계의 볼륨 객체에 디렉토리 관리를 추가한 것이다. 이로서 구조화된 화일 단계에서는 화일의 명칭으로 화일을 지정할 수 있다

가장 상위 단계인 인덱스 화일 단계(indexed file layer)는 구조화된 화일 단계에서 제공하는 데이타 화일과 인덱스 화일을 결합하여 가상의 인덱스된 화일로 보여주는 역할을 한다. 즉 이 단계에서는 일정한 범위 내의 레코드들을 순차적으로 검색하는 것을 도와주는 일종의 응용 단계(application layer) 이다

2.1.2 SESS의 특징 및 성능 평가

SESS의 가장 큰 특징이자 장점은 각 단계별로 그리고 각 모듈 별로 확장이나 수정이 매우 자유롭다는 것을 들 수 있다 C++ 언어[18]를 사용하여 구현된 SESS는 고정 길이 레코드 순차 화일과 가변 길이 레코드 순차 화일을 지원하고 있으며, 인덱스 구조로는 B+트리 구조와 해쉬 구조(extensible hash)를 지원한다. 인덱스 구조에서 사용하는 키는 임의로 사용자가 정의할 수

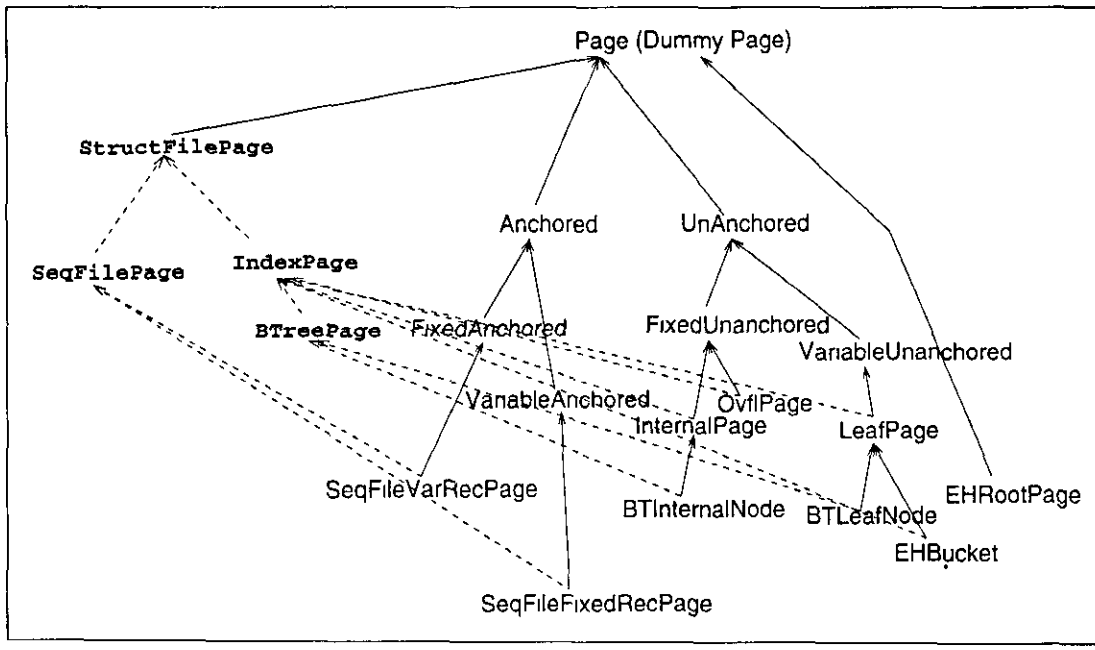


그림 2 페이지 클래스 계층 구조

항목	SESS	WiSS	평가 방법
가변길이 레코드 순차 화일	2 80초	3 20초	삽입(200회), 추가(486회), 삭제(159회), 읽기(314회), 쓰기(70회)
B+트리	137 40초	155 30초	삽입(40,000회), 검색(10,000회), 삭제(40,000회)
길이가 긴 데이타	3 20초	3 50초	1K~19K 크기의 블럭을 삽입(307회), 삭제(243회),읽기(198회), 쓰기(252회)

표 1 SESS의 성능 평가

있기 때문에 복합 키(*compound key*)의 사용이 용이하다. 그리고 B^+트리의 변형된 구조를 갖는 길이가 긴 데이타(long data) 구조도 지원하고 있어, 크기의 제한 없이 임의의 위치에 삽입, 삭제가 가능하다. SESS의 테스트는 WiSS의 테스트 프로그램을 확장하여 사용하였으며, 가변 길이 레코드 순차 화일을 비롯하여 B^+트리 인덱스, 길이가 긴 데이타에 대한 성능 평가를 실시하였다. 성능 평가는 동일한 프로그램을 각각 10번씩 수행한 후 이를 평균한 값을 사용하였다(표1 참조)[2].

2.2 SESS의 다중 사용자화 및 고장 회복 기능

현재 SESS의 다중 사용자화와 고장 회복 기능의 구현이 진행 중에 있다. 다중 사용자 SESS는 2 단계 잠금 규약(2 phase locking protocol)에 기반하여 동시성 제어를 한다. 시스템의 성능 향상을 위해 SESS에서는 UNIX O/S에서

2) 프로그램의 수행은 Sparc2(SunOS 4 1.2, 메인 메모리 32 *Mbytes*)에서 실시하였다

제공하는 세마포어(semaphore)를 사용하지 않고 직접 하드웨어 명령어 test-and-set을 이용하여 동기화와 상호 배제를 구현하고 있다. 고장 회복 관리자는 로깅(logging)을 사용하는 ARIES[11] 고장 회복 알고리즘에 근거하여 설계 및 구현 중에 있다 ARIES을 고장 회복 기법으로 사용한 이유는 ARIES 알고리즘이 강력하면서도 비교적 단순한 알고리즘이기 때문이다. 현재 목표로 하고 있는 고장 회복 기능이 페이지 단위 고장 회복이기 때문에 ARIES의 장점을 충분히 발휘할 수 없으리라 예상되지만, 고장 회복 단위가 페이지 단위보다 작은 레코드 단위로 확장될 경우를 고려하여 ARIES 알고리즘을 채택하였다.

2.3 질의 처리기(SNU SQL Engine)

SNU 질의 처리기는 확장성과 재사용성을 그 특징으로 하며 SESS 상에서 구현하였다

2.3.1 질의 처리기의 구성

SNU SQL Engine은 사용자로부터 받은 SQL 문을 질의 객체, 파싱 트리 객체, 질의 그래프 객체 등으로 변형시키면서 최종 결과를 얻는다 먼저 질의 객체는 SQL 문을 객체화한 것으로서 파싱을 통해 파싱 트리 객체를 생성한다. 파싱 트리 객체는 검색 조건들의 정규화, 정당성 검사, 뷰 변환(transformation) 등을 수행함으로써 질의 그래프 객체를 생성한다. 정당성 검사에서는 올바른 테이블과 컬럼, 데이타 값 등이 사용되었는지를 검사하고 사용자가 각 테이블과 컬럼에 대해 권한을 가졌는지 확인한다 이때 중첩된 SQL 문에 대한 파싱 트리를 단일 레벨 SQL 문의 파싱 트리로 변환하는 작업도 함께 이루어진다. SNU SQL Engine은 기본적으로 질의 변환 방법을 통해 뷰를 지원하고 있으나, 통계 뷰에 대해서는 count 방법[14]을 통한 저장 뷰(materialized view) 개념을 사용하고 있다. 통계 뷰는 AVG, MAX, MIN, SUM, COUNT 등과 같은 통계 함수를 이용해서 정의된 뷰를 일컫는데, 이들은 자주 사용되는 유용한 정보를 가지면서도 대개 테이블의 크기는 작아서 보다 효율적인 질의 처리가 가능하다. 이러한 과정을 거쳐 파싱

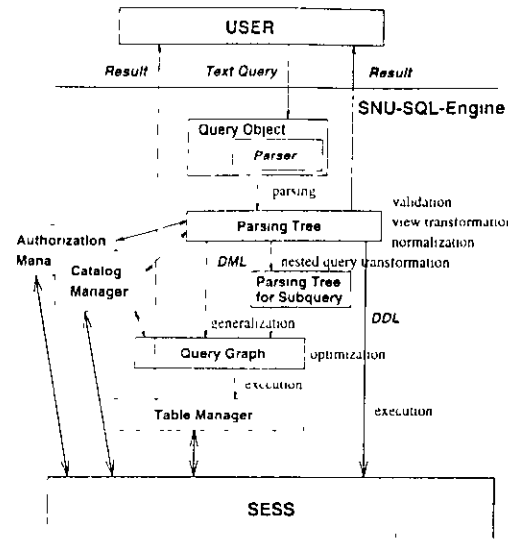


그림 3 SNU-SQL-Engine의 구조

트리 객체는 질의 그래프 객체로 변형되는데, 데이타 정의어(DDL)의 경우는 효율적인 실행을 위해서 질의 그래프 객체로 변환하지 않고 직접 파싱 트리 객체에서 수행한다. 질의 그래프 객체는 최적화를 한 후 실행함으로써 최종 결과 객체를 생성한다. 그림 3는 SNU SQL Engine의 전체 구조이다.

SNU-SQL-Engine에서 사용하는 시스템 카탈로그로는 데이타베이스 내의 모든 테이블에 대한 정보를 저장하는 테이블 카탈로그와 각 테이블 내의 모든 컬럼들에 대한 정보를 저장하는 컬럼 카탈로그, 각 테이블에 생성된 인덱스에 대한 정보를 저장하는 인덱스 카탈로그, 뷰를 정의하는 SQL문을 저장하는 뷰 정의 카탈로그, 뷰의 테이블들의 종속성을 저장하는 뷰 종속성 카탈로그, 그리고 테이블과 컬럼에 대한 사용자의 권한을 저장하는 테이블 권한 카탈로그와 컬럼 권한 카탈로그가 있다. 카탈로그 관리자는 이러한 카탈로그들을 저장 시스템에 유지, 관리하고 이를 메모리 내의 카탈로그 객체로 변환하는 작업을 담당한다. 테이블 관리자는 카탈로그 관리자를 사용하여 저장 시스템의 물리적인 테이블로부터 메모리 내의 논리적인 테이블 객체를 생성한다 SNU SQL Engine의 권한 관리자는 Grant/Revoke 모델에 기반한 권한 모델을 지원

표 2 SNU SQL Engine 성능 평가 결과

항 목	실험 결과(초)	
	SNU SQL Engine	INFORMIX
질의 1(인덱스를 사용하지 않음)	31	57
질의 1(학과 테이블에 B+트리 인덱스 사용)	30	17
질의 2(인덱스를 사용하지 않음)	316	768
질의 2(TA, TB에 B+트리인덱스 사용)	120	125
질의 3(인덱스를 사용하지 않음)	441	924
질의 3(모든 컬럼에 B+트리 사용)	180	280

하며, 테이블과 뷰 단위의 권한 관리 뿐만 아니라 각 컬럼 단위의 권한 관리를 지원한다.

2.3.2 질의 처리기의 특징 및 성능 평가

SNU SQL Engine은 객체지향 기법을 사용하여 설계, 구현함으로써 확장성과 재사용성을 그 장점으로 하고 있다. 구현 언어는 C++을 사용하였으며, ISO와 ANSI에서 표준화한 SQL1[6]을 지원하고 있다.

SNU SQL Engine의 적합성 평가에는 SQL1을 채택한 미국의 FIPS PUB 127-1의 적합성을 검사하기 위해서 NIST에서 개발한 SQL Test Suite [15]를 사용하였다. SQL Test Suite는 테스트 데이타베이스를 위한 스키마와 데이타, 그리고 다양한 SQL문과 그 실행 결과로서 구성된다. NIST SQL Test Suite의 대화형 SQL에 대한 검사용 SQL문에는 데이타 조작어에 관한 것과 제약 조건에 관한 것들이 있는데, 데이타 조작어에 관한 총 199가지 경우들 중에서 보다 확장된 기능을 검사하는 21가지 경우를 제외한 총 178가지 경우에 대하여 적합성 검사를 실시하였다. 그 결과 SNU SQL Engine은 129가지 검사를 통과하여 약 72.5%의 적합성을 보였다.

SNU SQL Engine의 성능 평가에는 서울대학교 정보시스템에서 실제로 사용되는 약 4.4 Mbyte 크기의 수강 편람 데이타베이스에 대하여 3가지 형태의 질의를 사용하였다. 질의는 단일 테이블에 대한 질의와 두개의 테이블에 대한 질의, 그리고 세개의 테이블에 대한 질의로 구성된다.

질의 1
```
SELECT COUNT(*)
FROM 강의시간
WHERE 학과 BETWEEN '400' and '600';
```

질의 2
```
SELECT COUNT(*)
FROM 교과과정 TA, 강의시간 TB
WHERE TA.과목번호=TB.과목번호;
```

질의 3
```
SELECT COUNT(*)
FROM 소속, 교과과정, 강의시간
WHERE 소속.학과=교과과정.개설학과
AND 강의시간.과목번호=교과과정.과목번호;
```

표-2에 나타난 수행 결과는 SNU SQL Engine과 INFORMIX I-SQL(ver. 4.0)에 대해 위 질의를 각각 10번씩 수행한 후 최소값과 최대값을 제외한 값들의 평균치이다[3]. 또한 각 질의에 대해서는 인덱스를 적용하는 경우와 그렇지 않은 경우에 대해서 성능 실험을 실시하였다.

앞으로 SNU SQL Engine의 보완점은 다음과 같다. 먼저 SQL1 적합성을 90%이상으로 높이고 SQL2로 확장하는 것이 요구된다. 그리고 시스템의 성능을 보다 향상시키기 위해서 인덱스를 이용한 질의 최적화 방법의 개선 등 보다 다양한 질의 최적화 기법이 필요하다.

2.4 ESQL 전위처리기

3) 프로그램의 수행은 Sparc2(SunOS 4.1.2, 메인 메모리 32 Mbytes)에서 실시하였다

내포형 데이타베이스 프로그래밍 언어는 기존의 프로그래밍 언어에 데이타 모델을 결합하여 확장시킴으로써 복잡한 데이타베이스 응용 프로그램의 작성을 도와준다. 본 ESQL 전위처리기는 SQL1 표준을 최대로 지원하고 강력한 오류 처리를 제공함으로써 응용 프로그램의 개발을 쉽게하는 것을 기본 목표로 하고 있다. 현재 프로그래밍 언어에 내포된 SQL의 처리는 구현되었으며, 데이타 타입 검사와 오류 처리 기능을 추가하고 있다.

3. SOP—객체지향 DBMS의 개발

이미 앞에서 언급한 바와 같이 현재 데이타베이스 분야에서는 객체지향 데이타 모델의 풍부한 성질을 기반으로 차세대 응용 분야에서 요구하는 복잡한 형태의 데이타와 연산을 지원하려는 시도가 있다.

본 연구실에서는 SRP 개발의 연구 결과를 토대로 객체지향 데이타베이스 관리 시스템의 개발을 시작하였다 본 연구실에서 목표로 하는 객체지향 DBMS는 CAD, OIS, GIS 등의 응용 분야에서 적합한 성능을 발휘하며, 또한 새로운 요구에 쉽게 확장하거나 수정할 수 있는 시스템이다. 그리고 객체지향 데이타 모델은 ODMG의 제안을 최대한 수용하기로 하였다. 이번 절에서는 SOP(SNU Object-Oriented DBMS Platform)에 대해서 주요 구성 모듈의 특징과 진행 방향을 중심으로 소개하겠다

3.1 객체 관리자(Object Manager)

객체지향 데이타 모델은 관계형 데이타 모델에 비해 매우 복잡하기 때문에 질의 처리기와 같은 상위 모듈이 직접 저장 시스템의 기능을 사용하여 구현하기에는 많은 무리가 따른다. 이에 객체지향 DBMS에서는 객체 관리자를 두어 시스템의 복잡도를 해결하고 있다. 즉, 객체 관리자란 객체지향 DBMS의 중심 모듈로 저장 시스템에서 제공하는 연산을 사용하여 질의 처리기나 스키마 관리자 등의 상위 모듈에게 추상화된 형태의 객체와 이에 따른 기본 연산을 제공하는 모듈이다. 지금까지 객체 관리자는 각 객체지향 DBMS의 내부 모듈로서 구현되거나, Mneme[12], ObServer [5]와 같이 객체 서버(object server)로서 개발되어 지속성 프로그래밍 언어(persistent programming language)의 후위 처리기 등으로 사용되고 있다.

본 객체 관리자의 개발 목표는 특정 데이타 모델에 의존하지 않고 여러 다양한 응용 분야를 지원할 수 있도록 확장성과 성능을 제공하는 것이다. 객체 관리자의 기능에는 객체 식별자(OID)의 생성과 관리, 객체의 접근과 조작, 객체 버퍼 관리, 객체 버젼 관리, 복합 객체 관리, 객체 명칭 관리 등이 있으며, 성능의 향상을 위해서 다음과 같은 특징을 갖는다.

객체 식별자란 객체의 저장 장소나 상태에 관계없이 유일하게 객체를 구별할 수 있는 수단으로 CAD나 CASE와 같은 설계 응용 환경에서는 반드시 요구되는 특성이며, 동시에 객체 식별자의 표현 방식은 객체지향 DBMS의 성능에 중요한 요소가 된다[8]. 본 객체 관리자에서는 객체 식별자를 물리적 주소로써 표현하여 디스크 상에서의 접근 속도를 향상시키고자 한다. 그러나 물리적 주소의 경우 시간적으로 객체 식별자의 유일성을 보장할 수 없기 때문에 실제 객체 식별자는 사용자의 요구에 따라 최대 4바이트 크기의 시간 식별자를 추가로 갖는다. 객체 버퍼 상의 객체에 대한 접근은 객체에 대한 포인터를 직접 제공함으로써 다른 일반 메모리 객체와 동일한 접근 속도와 투명성을 제공한다. 이를 위해 객체 관리자는 하드웨어와 운영체제에서 제공하는 가상 메모리(virtual memory) 관리 기능을 사용한다.

객체 버퍼 관리에 있어 또 다른 중요한 문제는 스위즐링(swizzling)이다[13]. 객체 사이의 참조는 OID를 통해 이루어지는데, 스위즐링이란 객체 버퍼 상에서 OID를 객체 버퍼 상의 실제 주소 값으로 변경하는 것을 말한다. 따라서 스위즐링 이후의 객체 참조는 OID를 객체의 주소값으로 변경할 필요 없이 바로 접근이 가능하므로 빠른 수행을 보장한다. 그러나

디스크로 다시 저장하기 위해서는 스위즐링된 주소 값들을 다시 OID로 변경시켜 주어야 하는 오버헤드가 따른다. 이에 본 객체 처리기에서는 사용자의 요구에 의해서만 스위즐링하여 불필요한 스위즐링을 최소화한다

객체 버젼 관리는 객체 버젼 제어를 위한 기본적인 연산만을 제공하고 상위 모듈에서 여러 다양한 형태의 버젼 관리 기법을 구현하도록 한다. 이외에 객체 관리자는 복합 객체 관리를 위한 동시성 제어, 권한 제어 등을 제공하며, 주어진 명칭으로부터 해당하는 OID의 값을 얻을 수 있는 객체 명칭 관리를 한다.

객체 관리자의 하위 모듈인 저장 시스템은 SRP 과제에서 개발된 SESS를 사용한다.

3.2 스키마 관리자(Schema Manager)

객체지향 DBMS의 주요한 응용 분야들은 그 특성상 스키마를 설계하는 단계에서 빈번한 스키마 변경(schema evolution)을 요구한다[24]. 그러나 스키마의 변경은 스키마 자체의 비일관성(inconsistency) 문제를 비롯하여 변경된 스키마와 기존 객체 사이 발생하는 구조적 불일치 문제, 그리고 스키마 변경에 따른 메소드와 응용 프로그램의 무효화(invalidation) 문제 등을 발생시키는데, 객체지향 DBMS에서는 스키마 관리자와 스키마 관리 도구를 통해서 이러한 문제를 해결하고 사용자의 스키마 변환을 도와주고 있다.

스키마 관리를 지원하는 객체지향 DBMS로는 ORION[9], Gemstone[16], Encore[17], O2[20]있다. 여기서는 스키마 관리와 관련한 앞의 여러 문제점들에 대한 여러 시스템들의 해결책과 문제점을 살펴본 후, SOP의 설계 내용을 제시하겠다.

스키마 변경으로 인한 스키마의 비일관성 문제를 해결하기 위해서 객체지향 DBMS는 가능한 스키마 변환 연산과 스키마가 만족해야 할 조건(invariant)들을 정의하고, 각 스키마 변환 연산에 대해 결과 스키마가 주어진 조건을 만족하도록 하는 시맨틱을 제공하고 있다. 그러나 대부분의 객체지향 DBMS에서 지원하는 스키마 변경 연산의 의미는 고정적이기 때문에 여러 다른 응용 환경에서 요구되는 스키마 변경을 모두 만족시키기가 어렵다. 예를 들면 클래스 A와 A의 상위클래스 B 사이의 상속 관계를 없애는 스키마 변환의 의미는 B로부터 상속받은 속성(attribute)을 클래스 A가 그대로 유지하고 있거나 없앨 수도 있는데, 이를 사용자가 스키마 변환 시 지정할 수 있어야 한다. 본 스키마 관리자는 이와 같이 스키마 변환 연산의 의미를 사용자가 좀 더 자유롭게 지정할 수 있는 방법을 제공한다.

스키마 변경으로 인한 스키마와 인스턴스 객체들 사이의 구조적 불일치 문제의 해결 방법은 크게 변환(conversion) 방식[9,16,20]과 에뮬레이션(emulation) 방식[17]으로 구분할 수 있다. 변환 방식은 객체의 구조를 바뀐 스키마에 맞게 물리적 구조를 바꾸는 방법이데, 변환 방식은 다시 변환 시점에 따라 동시 변환(immediate update), 지연 변환(deferred update) 방식으로 구분한다. 본 스키마 관리자는 동시 변환과 지연 변환을 모두 지원함으로써 객체 접근 양식에 따라 적절한 방법을 선택할 수 있도록 한다.

마지막으로 스키마 변경으로 인한 메소드와 응용 프로그램도 무효화 문제의 경우, 아직 대부분의 객체지향 DBMS에서 다루고 있지 않으며 이론적인 연구가 부족하다. 본 스키마 스키마 관리자는 객체의 인스턴스 변수와 메소드의 의존 관계와 메소드와 메소드 사이의 의존 관계를 유지 관리함으로써 스키마 변경 시 발생하는 메소드나 응용 프로그램의 무효화를 검사한다.

이밖에 SOP에서는 사용자의 스키마 설계와 변경 작업을 편리하게 하기 위해서 스키마 브라우저(schema browser)와 스키마 변경 지원 도구 등을 아울러 개발할 예정이다.

3.3 질의 처리기

질의어로는 ODMG-93[2]에서 제안한 OQL을 채택하였다. 그러나 ODMG-93에서 제안한 질의어는 모호한 점이 많기 때문에 질의 처리

기의 구현에 앞서 SOP의 질의어를 정의하는 것이 선행될 예정이다.

SOP의 질의 처리기는 SRP의 질의 처리기와 유사한 구조를 가지며, 파싱 트리 클래스 계층구조와 질의 그래프 클래스 계층구조 등 주요한 클래스 계층구조를 확장하여 재사용할 것이다. 그러나 SOP의 질의 처리기는 SRP의 질의 처리기와는 달리 질의 그래프 객체를 객체 대수(object algebra)의 논리적 연산자를 포함하는 연산자 그래프[7]로써 구성하여 데이타베이스에 대한 접근 방식을 표현한다. 논리적 연산자에는 selection, projection, join, set operations, unnest, materialize 등이 포함된다. 질의 최적화는 논리적 질의 그래프 객체에 대해서 이루어지며, 최적화된 질의 그래프 객체의 각 논리적 연산자들은 다시 물리적 연산자들로 변환된다[4]. 물리적 연산자는 segment scan, sort, merge, index scan, assembly 등으로 구성되며, 논리적 연산자를 물리적 연산자로 대응시키는 부분은 질의 최적화의 한 과정으로 질의 처리기의 성능에 많은 영향을 미친다. 질의의 결과는 이러한 변환 과정을 통해 얻은 물리적 연산자들의 네트워크를 수행하여 얻는다.

3.4 데이타베이스 프로그래밍 언어(DBPL)

SOP에서는 사용자 친숙도를 높이기 위해, 최근 들어 가장 널리 사용되는 C++ 언어[18]를 기본 DBPL로서 정하였다. DBPL의 개발 목표는 데이타베이스와 프로그래밍 언어와의 불일치를 극소화시키고 ODMG-93[2]을 최대한 수용하는 것이다. 이를 위해서 먼저 부족한 ODMG의 표준안을 확장하고 모호성을 배제하는 일이 요구된다. 또한 새로운 기능이나 프로그래밍 언어를 쉽게 추가 할 수 있도록 모델의 확장성을 제공할 예정이며, 최적화 방안 및 서로 다른 프로그래밍 언어 사이의 객체 호환성[10]에 관한 연구도 수행할 것이다.

4. 결 론

지금까지 본 객체지향 시스템 연구실에서 진행중인 SRP 과제와 SOP 과제에 대해서 소개하였다. 먼저 SRP 과제는 단일 사용자용 저장 시스템과 질의 처리기의 구현이 완료되었으며, 다른 시스템과의 성능 평가도 이루어 졌다. SRP의 저장 시스템과 질의 처리기는 모두 확장성을 가장 큰 장점으로 하고 있어 쉽게 새로운 화일 구조나 기능을 추가 할 수 있다. 현재 저장 시스템의 다중 사용자화 및 고장 복구 기능이 추가 중이며 ESQL 전위 처리기도 구현 중에 있다. 그리고 멀티미디어 자료를 위한 다중 키 인덱스 구조의 구현도 진행되고 있다.

SOP 과제는 관계형 DBMS의 한계를 극복하고 새로운 응용 분야를 지원하기 위한 객체지향 DBMS의 개발을 목표로 하고 있다. SOP 과제는 SRP 과제의 연구 결과를 기반으로 진행 중에 있다. SOP 과제는 크게 객체 관리자, 스키마 관리자, 질의 처리기, DBPL의 개발로 구성되며, 각 모듈은 SRP 과제에서와 같이 확장성과 성능을 최대 목표로 한다 또한 SOP에서는 ODMG-93에서 제안한 객체지향 데이타 모델을 최대한 수용하며, 개발된 객체지향 DBMS상에서의 멀티미디어 응용 및 CASE 응용 과제도 수행할 예정이다.

참고문헌

[1] J Banerjee, and Won Kim *et al*, "Data Model Issues for Object-Oriented Applications," ACM Transactions on Information Systems, Jan. 1987.

[2] R. G. G Cattel, *The Object Database Standard ODMG-93*, Morgan Kaufmann Publishers

[3] R G. G Cattell, *Object Data Management Object-Oriented and Extended Relational Databse Systems*, Addison-Wesley, 1991.

[4] Goetz Grafe, "Query Evaluation Techniques for Large Databases," ACM Computing Surveys, **25**(2), June 1993.

[5] Mark F Hornick, and Stanley B Zdonik, "A Shared, Segmented Memory System for an Object-Oriented Database," ACM Transactions

on Office Information Systems, **5**(1), pp. 70~95, Jan. 1987.

[6] ISO, "Information Technology-Database Language SQL," 1989.

[7] Matthias Jarke, and Jurgen Koch, "Query Optimization in Database Systems," ACM Computing Surveys **16**(2), June 1984.

[8] Setrag N. Khoshafian, and George P. Copeland, "Object Identity," Proceedings of the ACM OOPSLA Conference, pp. 406~416, 1986.

[9] Won Kim, *Introduction to Object-Oriented Database*, MIT press.

[10] C. L clude and P. Richard, "The O2 Database Programming Language," Object-Oriented Database System: The Story of O2, Morgan Kanfmann Publishers Inc., 1991.

[11] C. Mohan, Don Haderle, Bruce Lindsay, Hamid Pirahesh, and Peter Schwarz, "ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging," ACM Transactions on Database Systems, **17**(1), March 1992, Pages 94~162.

[12] J. Eliot B. Moss, "Design of the Mneme Persistent Object Store," ACM Transactions on Information Systems, **8**(2), pp. 103~129, April 1990.

[13] J. Eliot B. Moss, "Working with Persistent Objects: To Swizzle or Not to Swizzle," ACM Transactions on Software Engineering, **18**(8), pp. 657~673, Aug. 1992.

[14] Mumick, "Maintenance View incrementally," Proceedings of SIGMOD, **22**, 1993.

[15] NIST, "SQL Test Suite for FIPS 127-1 Ver. 2.0," 1989.

[16] D.J Penny, and J. Stein, "Class Modification in the Gemstone Object-Oriented DBMS," Proceedings of the ACM OOPSLA conference, 1987.

[17] A. Skarra, and S. Zdonik, "The Management of Changing Types in an Object-Oriented Database," Proceedings of the ACM OOPSLA conference, 1986.

[18] Bjarne Stroustrup, The C++ programming language, 2nd edition, Addison-Wesley, 1991.

[19] Fernando Vélez, Guy Bernard, and Vineeta Darnis, "The O2 Object Manager: An Overview", Object-Oriented Database System: The Story of O2, Morgan Kanfmann Publishers Inc., 1991.

[20] R. Zicari, "A Framework for Schema Updates in an Object-Oriented Database System," Proceedings 1991 IEEE Data Engineering conference.

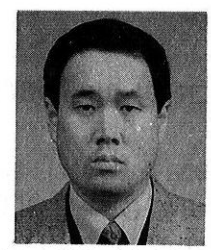
[21] 송용준, "객체지향 기법을 이용한 SQL 처리기의 설계와 구현", 서울대학교 석사 학위 논문, 1994.

[22] 안정호, "확장 용이 저장 시스템의 객체지향 설계," 서울대학교 석사 학위 논문, 1993.

[23] 이상원, "관계형 데이타베이스 시스템의 뷰와 권한 장치의 객체지향적 설계 및 구현," 서울대학교 석사 학위 논문, 1994.

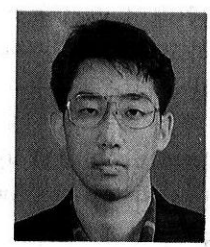
[24] "Panel on Schema Evolution and Version Management," Report on Object-Oriented Database Workshop, SIGMOD Record **18**(3), 1989.

김 형 주



1982 서울대학교 컴퓨터공학과 졸업
1985 Univ. of Texas at Austin, 전자계산학 석사
1988 Univ. of Texas at Austin, 전자계산학 박사
1988 Univ. of Texas at Austin, Post-Doc
1988 ~.1990 Georgia Institute of Technology, 조교수
1991 ~현재 서울대학교 컴퓨터공학과, 조교수
관심 분야 : 객체지향 시스템, 사용자 인터페이스, 데이타베이스

안 정 호



1991 서울대학교 컴퓨터공학과 졸업
1993 서울대학교 컴퓨터공학과, 공학석사
1993 ~현재 서울대학교 컴퓨터공학과 박사과정
관심 분야 : 객체지향 시스템, 데이타베이스