객체-관계 랩퍼에서의 OQL 처리 전략

임기성, 김홍기, 현순주
한국정보통신대학원대학교 공학부

Query Processing Strategies in an Object-Relational Wrapper System

Gisung Lim, Hongki Kim, SoonJoo Hyun
School of Engineering, Information and Communications University(ICU)

요 약

복잡해지고 있는 응용 프로그램의 스키마를 객체형 모델링 기법을 통해 효과적으로 정의함으로써 풍부한 의미관계를 유지하면서, 보편적인 하부 저장시스템 솔루션으로 제공되는 관계형 DBMS를 활용할 수 있는 객체-관계 접근 방법은 현실적인 선택으로 각광을 받고 있다. 이러한 접근 방법은 객체-관계 랩퍼(Object-Relational Wrapper)의 사용을 통해 가능할 수 있다. 본 논문에서는 사용자의 질의를 처리하는데 있어서 보다 효율적으로 관계형 모델에 기반을 둔 객체간의 관계(Association)를 바탕으로 한 질의 처리기와 이를 지원해 주는 새로운 객체-관계 랩퍼의 설계와 구현에 대해 설명한다. 자바로 개발된 질의 처리기와 객체-관계 랩퍼는 DBMS와 플랫폼에 독립적인 시스템으로 제공되며, 객체간 상속의 무결성 제약 조건을 유지하기 위해 메타 정보를 활용하는 특성을 가지고 있다.

1. 서 론

사회의 환경이 복잡해짐에 따라, 대부분의 컴퓨팅 환경에서 복잡한 구조의 데이터를 처리하도록 요구되어지고 있다. 이런 복잡한 데이터를 처리하는 응용 프로그램에서는 데이터들이 서로 긴밀하게 관계를 맺고 있으므로, 이러한 관계를 유지하고 이용하면서 응용 프로그램을 작성하는 것이 중요해 진다. 또한, 이런 복잡한 스키마를 표현하는데 있어서 관계형 데이터 모델이 적절하지 않다는 것은 널리 알려진 사실이다. 따라서, 객체 지향형 데이터베이스 시스템과 같은 새로운 형식의 데이터베이스 시스템이 이러한 응용 프로그램을 지원하기 위해서 제안되었다. 반면에 관계형 모델은 사용자의 입장에서 간단한 기술 능력과 시스템의 관리을 이루는 수학적으로 우수한 이론으로 인하여 데이터베이스를 기반으로 하는 각종 응용 프로그램에서 널리 사용되고 있다. 널리 사용되고 있는 기존의 관계형 데이터베이스 시스템을 객체 지향형 데이터베이스 시스템으로 교체하는 것은 경제적인 이유와 두 모델의 차이점으로 인한 여러 문제점을 가지고 있다. 그러므로, 객체 지향형 모델과 관계형 모델간의 차이를 채울 수 있는 접근 방법이 필요하다.

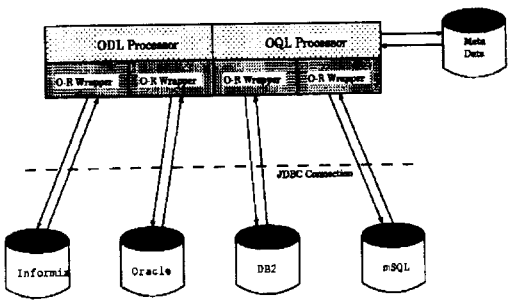
실질적으로 이러한 차이를 채울 수 있도록 여러 상용 관계형 데이터베이스 시스템 개발회사들은 확장된 관계형 모델을 제공하고 있다. 대표적인 예가 오라클(Oracle), 아이비엠(IBM)의 DB2, 인포믹스(Informix)와 같은 객체 관계 데이터베이스 시스템으로, 이들은 중복된 관계를 수용하며 기존의 관계형 데이터베이스 시스템에서 제공하는 것보다 복잡한 데이터 스키마를 다룰 수 있도록 하고 있다. 그러나, 이러한 확장된 관계형 데이터베이스 시스템들도 객체 지향형 데이터베이스 시스템과 비교하여 볼 때, 실세계를 표현하는데 있어서 부족한 점이 많이 나타나고 있는 것이 사실이다. 그러므로, 널리 사용되고 있는 관계형 데이터베이스 시스템을 이용하면서 객체 지향형 모델을 사용할 수 있도록 하여 객체 지향형 모델과 관계형 모델의 한계를 극복하려는 시도로 객체 관계형 랩퍼의 사용이 요구되고 있다[1]. 이러한 관점의 객체-관계형 랩퍼를 구현함에 있어서 가장 고려해야할 것은 객체 지향형 개념을 일반적인 관계형 데이터베이스 시스템에서 사용할 수 있는 환경을 구성하는 것이다. 객체 지향형 데이터베이스의 기준을 제안하고 있는 ODMG의 ODL과 OQL의 기능은 객체-관계 랩퍼의 구현에 있어서 핵심적인 기능이 될 것이다. ODL과 이 기능을 지원하기 위한 랩퍼의 개발은 [2]의 논문에서 자세히 언급되어 있다. 본 논문에서는 OQL과 이 기능을 지원하기 위한 랩퍼의 개발에 대한 설명을 하고자 한다.

본 논문의 구성은 다음과 같다. 제 2장에서는 시스템 구조를 통해 OQL 프로세서와 이것을 지원하는 객체-관계 랩퍼의 구현과 관련된 세부 사항을 설명한다. 제 3장에서는 OQL 질의문 처리를 위한 전략에 대해 설명한다. 그리고 제 4장에서 결론을 맺는다.

2. 시스템 구조

아래의 그림은 한국정보통신대학원대학교에서 시행하고 있는 AIMS(능동 정보 관리 시스템)의 한 부분인 객체-관계 랩퍼의 시스템의 구조이다. 여기서의 객체-관계 랩퍼는 자바와 JDBC를 이용하여 구현되어 있으므로 여러 플랫폼에 종속되지 않는 독립적인 특징을 가지고 있다. 그러므로, 본 랩퍼는 분산된 이기종의 관계형 데이터베이스 시스템을 이용한 서비스를 지원할 수 있으다. 그림에서의 ODL 프로세서는 ODMG 표준 ODL과 호환성을 지닌 파서로 개발되었으며, 자바 바인딩 코드를 생성하는 기능을 가지고 있다. 객체-관계 랩퍼에서 객체를 관계형 모델의 테이블로 매핑하는 과정과 내용에 대한 언급은 [2]에 표현되어 있다. OQL 프로세서는 사용자의 질의를 간단한 검색 질의와 복잡한 패턴을 지닌 질의로 나누어 처리할 수 있다. 간단한 검색 질의란 하나의 클래스에 관한 연산을 위한 처리를 하는 것을 말한다. 여기서의 복잡한 패턴 질의란 여러 클래스의 인스턴스간의 관계, 즉 패턴을 가지고 있는 연산에 대한 처리를 하는

것을 말한다. 사용자의 질의는 OQL 프로세서에 의해서 위와 같은 두가지 형식으로 분류되고 처리되어 진다. 이러한 기능을 가진 OQL 프로세서의 모듈은 아래에 설명된다.



2.1. OQL 프로세서

본 연구에서 구현된 OQL 프로세서는 다음과 같은 4가지의 하위 모듈로 구성되어 있다.

- 사용자 인터페이스 - 사용자의 질의를 받아서 최종 처리 결과를 전달한다.
- 어휘 구문 분석기 (OQL 파서) - 사용자 질의의 구문적 오류를 검사하여 결과를 전달한다.
- 검색 변환기 - 간단한 검색 명령어에 대해서는 SQL문의 검색 명령어로 변화하여 하위의 관계형 데이터베이스에게 전달한다.
- 패턴 프로세서 - 사용자 질의의 패턴 표현을 처리한다.

현재의 개발 상태에서 사용자 인터페이스는 문자 기반의 인터페이스를 제공하고 있다. 사용자는 이것을 통해 기존의 상용 데이터베이스에서 제공하는 것처럼 콘솔상에서 임의의 질의문을 사용하여 원하는 결과의 데이터를 얻어낼 수 있다. OQL 파서는 ODMG 2.0 표준의 OQL BNF 문법을 기반으로 하여 렉스(LEXical analyzer generator)와 야크(Yet Another Complier Complier)을 이용하여 개발되었다. 단순 검색 변환기는 패턴 표현을 가지고 있지 않은 단일 클래스의 정보를 추출하기 위해 사용자의 OQL 질의문을 관계형 모델에서 사용할 수 있는 SQL 질의문으로 변환하도록 하는 기능을 제공하고 있다. 마지막으로 복잡한 패턴 프로세서는 패턴을 가지고 있는 질의문을 처리하기 위한 모듈이다.

2.2 OQL 프로세서를 지원하기 위한 객체-관계 랩퍼

OQL 프로세서를 지원하기 위한 객체-관계 랩퍼의 역할은 하위의 관계형 데이터베이스에 저장된 모든 정보를 이용하여 객체 뷰(object view)로 나타나도록 하는 역할이다. OQL 프로세서에서 보는 모든 메타 정보는 관계형 모델이 아닌 객체 지향형으로 인식할 수 있도록 변환하여 주는 것을 의미한다. 즉, OQL 프로세서를 지원하기 위한 객체-관계 랩퍼에서는 기존의 관계형 데이터베이스 시스템 자체에서 사용되는 메타 정보를 유지해 주는 역할과 시스템 구조에서 보듯이 하위의 여러 데이터베이스에 연결을 유지 및 관리하는 역할도 동시에 제공하고 있다.

2.3 하위의 관계형 데이터베이스 시스템

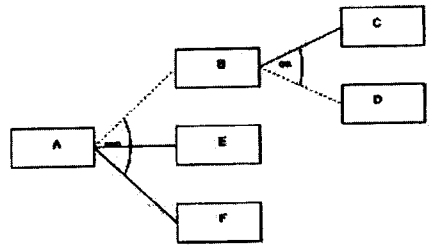
본 연구에서 사용되었던 관계형 데이터베이스는 오라클(Oracle)사의 Oracle 8i, 아이비엠(IBM)사의 DB2, 그리고 경량량 데이터베이스로 사용되고 있는 Hughes의 mSQL을 대상으로 시험되었다. 실제 객체의 저장은 일반 객체지향형 데이터베이스 시스템에서 제공되는 일반적인 방식이라할 수 있는 ODL 파서를 통해 생성된 Java 바인딩된 코드를 이용하여 이루어진다.

3. 객체-관계 랩퍼를 이용한 질의 처리 전략

3.1. 복잡한 질의의 패턴 처리

패턴 처리는 OSAM*.KBMS 모델[3]에서 소개되었다. 본 연구에서는 OSAM*.KBMS 모델의 패턴 처리를 포함한 확장된 ODMG/OQL을 지원한다. OQL 프로세서에서 사용하는 질의의 구문은 표준 ODMG/OQL을 반영하므로 일반적인 사용자가 사용하기 쉽도록 하는 반면, 의미적으로는 OSAM*와 A-algebra[3]의 군집 모델에 근간을 두어 ODMG 표준에서 지원하지 않는 질의 표현 능력을 제공할 수 있도록 한다.

일반적으로 ODMG/OQL에서는 젬스톤(GEMSTONE)에서 도입한 패스 표현(path expression)을 사용한다. 이것은 객체간의 연결을 통한 관계를 의미하기 위한 것으로 암시적(Implicit) 조인을 통해 사용하는 방식이다. 그러나, 이러한 방식을 사용했을 때, 여러 클래스의 객체들이 많은 관계를 가지고 있는 경우 일반적인 사용자가 긴 패스 표현을 사용하는 것은 부담이 된다.



예를 들어서 다음과 같이 점선의 경우, 2개의 클래스간의 관계를 의미하고, 실선은 연관관계를 나타낼 때, 전체 클래스의 패스를 ODMG/OQL의 표준에 의해 표현했을 경우는 다음과 같다.

```
from (select A from A where not exist (A.link_A_to_B)
                            and exist (A.link_A_to_E)
                            and exist (A.link_A_to_F) as A,
      (select B from B where not exist (B.inverse_link_A_to_B)
                            and (not exist(B.link_B_to_D)
                            or exist (B.link_B_to_C))) as B,
      B.link_B_to_C as C,
      (select D from D where not exist (D.inverse_link_B_to_D)) as D,
      A.link_A_to_E as E,
      A.link_A_to_F as F
```

반면, 본 연구에서 사용하고자 하는 OSAM*/OQL의 패턴 처리 방식을 도입하여 사용하였을 경우는 다음과 같다.

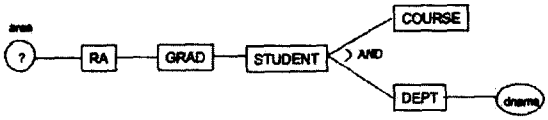
```
from A andBranch (!B orBranch(*C, !D), *E, *F)
```

즉, 사용자는 위와 같은 연관관계(Association) 표현법을 사용하여 클래스간의 여러 관계를 표현함에 있어서 보다 간편하고 유연성 있게 사용할 수 있을 것이다. 위의 패스 표현에서 사용된 '*', '!' 등의 심볼은 OSAM*의 연관 대수(Association Algebra)에 정의된 것으로 본 연구에서 구현된 OQL 프로세서는 OSAM*의 모든 연산자를 도입하여 복잡한 패턴을 간결하게 나타내도록 하였다.

3.2. 효율성을 위한 2단계 처리

본 연구에서 개발되는 OQL 프로세서는 하부에 객체-관계 랩퍼를 가지고 있는 층(layer) 구조를 가지고 있다. 일반적으로 층 구조는 하부 기능과의 통신비용의 문제로 인하여 성능면에서 느릴 수 있다. 그러나, 객체-관계의 랩퍼를 통해 하부 저장 시스템으로 관계형 데이터베이스를 상위의 사용자 인터페이스부분은 객체 지향형 모델을 제시하는 경우, 단일 시스템에 비하여 성능 면에서 오버헤드(overhead)가 발생하는 점은 피할 수 없다. 그러나, 우리가 주목한 것은 패턴의 처리를 위해 관계형 데이터베이스에서 일반적으로 일반적으로 사용하는 조인 연산이 갖는 부담이었다. 즉, 조인 연산으로 인한 내부의 메모리 소비나 처리 속도의 저하를 최소화하는 것이 중요한 문제였다. 따라서, 본 OQL 프로세서에서는 이러한 문제점들을 해결하기 위해 앞에서 설명했듯이 사용자의 질의를 간단한 검색 질의와 복잡한 패턴 처리, 두가지로 분류하여 처리하도록 고려하였다.

간단한 검색 질의의 경우는 하나의 클래스의 객체를 검색하는 과정이므로 크게 고려할 사항이 없었으나, 복잡한 패턴 처리의 경우 내부적으로 조인 연산을 피하기 위해 본 연구에서는 패턴 처리를 위해 웨이브 프론트(wave-front) 알고리즘[4]을 도입하였다. 이 알고리즘은 객체의 전체 데이터를 가지고 처리하는 것이 아니고 객체의 유일성이 보장되는 객체 식별자(Object Identifier, OID)만을 이용하여 패턴 처리를 시행하는 것이다. 즉, 복잡한 패턴의 경우 객체가 가진 전체 정보에 비하여 상당히 작은 양의 객체 식별자만을 이용하여 필요한 연산을 수행하고, 수행의 결과로 얻어진 질의를 충족시키는 객체의 식별자를 이용해서 실제 사용자가 원하는 정보를 추출(Projection)하는 것이다. 본 연구에서의 시스템 구조처럼 2단계로 하위 데이터베이스에 접근하는 경우, 많은 양의 객체의 정보가 이동하므로 발생되는 통신비용과 조인 연산을 사용하였을 경우 발생하는 오버헤드(overhead)를 줄일 수 있다. 그러므로, 층(layer) 구조를 사용하므로 발생되는 overhead의 단점을 줄일 수 있게 되는 것이다. 이제부터 본 연구에서 도입한 웨이브 프론트 알고리즘을 예제를 들어 설명하겠다.



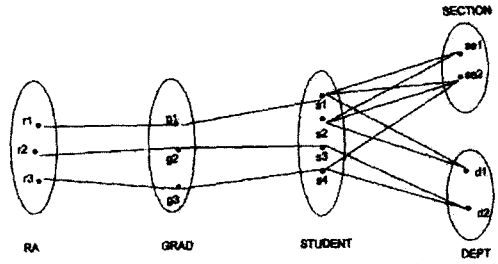
위와 같은 구조를 가지고 있는 하위 데이터베이스에 COURSE와 CS라는 DEPARTMENT에 참여하는 대학원 학생으로 RA인 사람의 전공을 찾는 질의가 있을 경우, 위의 질의를 본 시스템의 OQL 질의문을 이용하여 표현하면 아래와 같다.

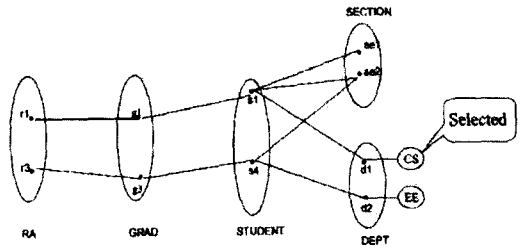
```
select RA.area
from RA * GRAD * STUDENT andBranch (*COURCE, *DEPT)
where DEPT.dname = 'CS'
```

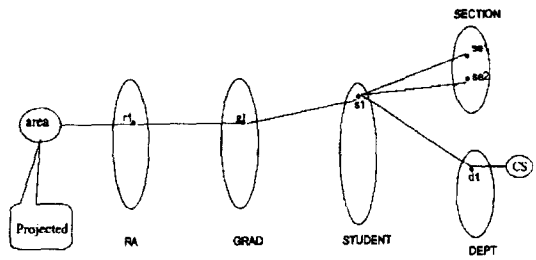
이때, 하위 데이터베이스에서의 패턴 처리를 위해서 실제 인스턴스간의 관계를 그림으로 도식해 보면 아래와 같다고 가정하자.

이와 같은 개념적인 클래스간의 관계를 알게 되면, OQL 프로세서는 웨이브 프론트 알고리즘을 이용하여 다음의 그림과 같은 논리적인 뷰를 만들게 된다.

OQL 질의문의 from 조건에 의한 객체의 연결성이 유지되었으므로, 이제는 where 조건을 부합시키는 DEPARTMENT 객체와 관계를 맺고 있는 모든 객체의 정보를 추출할 수 있는 것이다.







4. 결론

객체 지향형 모델링 기법을 지원하면서 이기종의 관계형 데이터베이스와 연동하기 위한 API를 제공해 주는 객체-관계 랩퍼는 많이 개발되고 있다. 우리는 본 연구에서 제안한 OQL 프로세서와 이것을 지원하는 객체-관계 랩퍼, [2]에서 제안한 ODL과 ODL 파서를 지원하는 랩퍼의 사용으로 사용자가 데이터베이스를 사용하는 응용 프로그램을 작성하는데 있어서 훨씬 능률적이고 표현력이 좋은 DBMS의 지원을 받게 되리라 생각한다.

참고 문헌

[1] M.J.Carey, D.J.DeWitt and L.L.Vandenberg. "A Data Model and Query Language for EXODUS" Univ,Wisconsin, CS-TR-734, 1987.

[2] 김규백, 김홍기, 현순주, "능동적 객체간 의미 제약성을 제공하는 자바기반 객체 관계 랩퍼의 구현", 99 추계 정보과학회 발표논문집, p355-357, 1999

[3] AmM.Alashqur, S.Y.W.Su, and H.Lam, "OQL:A Query Language for manipulating Object Oriented Database." VLDB, Amsterdam, Netherand, August 1989, pp.433-442

[4] Y.H.Chen and S.Y.W.Su, "Implementation and evaluation of Parallel query processing algorithms and data partitioning heuristics in Object oriented databases." journal of Distributed and Parallel Databases, Vol.4, 1996, pp.107-142