객체식별자를 사용한 항해기반의 데이터베이스 커서 개발

배명남[O] 박유미 한미경 최완
한국전자통신연구원 실시간데이터처리팀
{mnbae, parkym, mkhan, wchoi}@etri.re.kr

Development of Navigation-based Cursor Using Object Identifier on TACHYON

MyungNam Bae[O] YooMi Park MiKyong Han Wan Choi
Real-Time DBMS team, ETRI

요 약

교환기 시스템(switching system)과 같은 통신 시스템 환경에서는 매우 제한적인 시간 제약하에서 명시된 이벤트들이 반드시 완료되어야 한다. 따라서, 시스템내에 유지되는 응용 데이터에 대해, 매우 빠르고 균일한 접근 시간을 제공하여야 한다. 또한, 최근들어 통신 운용 환경과 교환기 소프트웨어의 복잡성의 증가로 다중테이블 조인과 객체지향 특성과 같은 고급 기능을 포함한 보다 향상된 데이터 모델링이 요구되고 있다. 이를 위해, 본 논문에서는 1) 통신 응용 환경을 보다 쉽게 모델링하도록 다양한 설계 모델링 개념을 제공하고, 2) 조인과 같은 연산을 위해 추가의 메모리공간이나 시간을 사용하지 않고, 객체식별자의 항해를 통해 직접 결과를 추출하는 방법에 대해 설명한다.

1. 서론

교환기 시스템(switching system)은 호처리, 트래픽 처리 등을 지원하기 위해 많은 응용 데이터의 관리가 요구된다. 하지만, 교환기는 매우 제한적인 시간 제약하에서 명시된 이벤트들이 수행되어야 하기 때문에, 이들에 대해 매우 빠르고 균일한 접근 시간을 제공할 수 있어야 한다[1]. 따라서, 교환기와 같은 안정되고 고성능이 요구되는 통신 시스템에는 운용될 모든 데이터세트를 주기억장치에 상주시키고, 이에 대한 일관성과 고성능 접근을 제공하기 위한 여러 방식들[2,3]이 제안되었다. 이들[2,3]은 데이터에 대한 접근 속도의 향상을 가장 주요한 목적으로 하고 있어, 고수준의 데이터 모델을 제공하지 않고 있다. 한 예로, 데이터들에 대한 조인(join)은 균일하지 않은 접근 시간과 부가적인 메모리의 요구로 통신 시스템의 제한적인 시스템 환경에는 적합하지 않았다.

최근, 하드웨어의 가격하락과 비약적인 성능 향상, 그리고 교환기 소프트웨어의 복잡성 증가, 통신 소프트웨어의 컴포넌트화 요구, 통신용 객체지향 언어의 지원[4] 등으로, 기존의 데이터 처리 방법들[2,3]과는 달리 데이터들간의 조인 뿐만 아니라 다중 테이블 상속, 상속 구조에 따른 질의 상속 등의 고수준의 데이터 모델링 기능들이 요구되고 있다.

TACHYON[5]은 ETRI에서 개발된 객체관계형 주기억장치 상주형 DBMS로, 통신 운용 환경에서 자주 사용되는 다양한 설계 시멘틱을 관계로 정의할 수 있고, 추가로 관계의 다중성(multiplicity), 방향성 뿐만 아니라 관계에 대한 참조 무결성을 보장하는 객체관계형 데이터 모델을 지원한다.

본 논문은 [5]에서 이러한 고수준의 모델링 기능을 제공하면서 동시에 추가의 메모리공간이나 시간을 소요하지않고, 객체식별자의 항해를 통해 직접 결과를 얻을 수 있도록 구현한 데이터베이스 커서에 대해 기술한다.

2. TACHYON에서 관계성의 표현

[5]는 모든 데이터들에 대해 유일한 객체식별자(Object Identifier, OID)를 할당하고, 데이터들간의 관계는 관련된 데이터의 OID를 속성으로 가짐으로서 성립된다. 예를 들어, employee 데이터 e1이 department 데이터 d1과 부서-직원 관계 r1이 있다면, r1은 e1내 d1의 OID를 d1은 e1내의 OID를 가짐으로서 구체화된다. 여기에서 e1과 d1이 r1으로 구체화되었다는 것은 e1.r1을 통해 e1의 부서 데이터에 접근가능하며, 역으로 d1.r1을 통해 부서에 속한 직원 데이터에 접근이 가능하다는 것을 의미한다.

다음 그림은 부서-직원 관계를 정의하는 스키마이다. oid_ref-oid_set inverse로 선언된 관계는 두 클래스가

n:1 관계를 가짐을 의미하며, 항상 관계에 대한 참조무결성을 보장한다.

```
create class employee          create class department
(                              (
  ...                            ...
  deptoid_ref department,        emp oid_set inverse employee.dept,
);                             );
```

그림 1. 관계성 정의 스키마

이때, 두 클래스의 죠인은 OID를 갖는 dept, emp 속성(attribute)을 사용한 항해로 이루어진다. 즉, "$\pi_{dept\_name,\ emp\_name}$(employee$\bowtie_{employee.oid=emp}$department)"과 같은 질의는 "select dept_name, emp->emp_name from department;" 형태로 주어지고, [5]에서는 이를 위해 두 클래스에 대한 물리적인 재구성없이 department의 emp 속성을 사용한 항해를 통해 죠인 결과를 얻는다(질의에서와 같이 OID를 통해 다른 데이터로의 참조는 곧 그 데이터와의 조인을 의미). 이와같이 OID를 통한 다른 객체의 참조를 위해 경로 연산자("->")를 사용한다. 경로 연산자는 자신과 관련된 다른 데이터를 지칭할 때 사용되며, 다시 지칭된 데이터내의 OID를 통해 다른 데이터와 반복적으로 죠인될 수 있다.

[5]에서는 연관(association)외에도 집성화(composition), 일반화(generalization)도 유사한 방법으로 항해를 통해 결과를 추출할 수 있다. 현재, [5]는 ESQL, ODBC/JDBC를 사용한 질의 인터페이스를 제공하고 있다.

3. 항해기반 커서

커서는 주어진 질의(Query)에 만족하는 데이터들의 순차적인 접근 연산을 위해 사용하는 구조이다. 기존의 커서는 재구성된 데이터들에 대한 순차 접근을 제공하지만, 이 장에서는 데이터에 직접 접근할 수 있는 OID를 사용해, 데이터의 물지적인 재구성 없이 항해를 통해 보다 융통성있고 효과적인 커서에 대해 기술한다. 현재, [5]는 표준 SQL[3]의 정적/동적 SQL을 지원하고 있다.

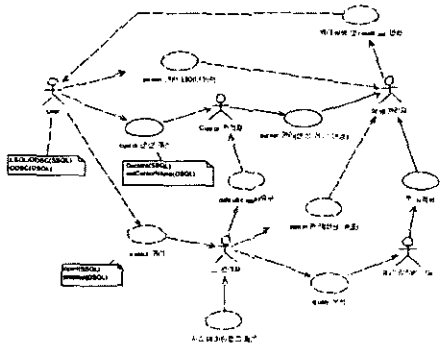


그림 2. 질의 처리 흐름

[5]에서 질의 처리는 크게 4개의 흐름으로 이루어진다. 1) 질의내에 파라메터 심볼을 추출하고, 이후 값과 대체할 구조를 준비하고(ParamFields 클래스), 2) 데이터들에 대한 메타정보들로 관련있는 클래스의 이름, 속성 번호, 별명(alias) 등(Cursor와 TablePtr 클래스등)을 추출한다. 3) 이를 바탕으로 최적화를 수행(Statement 클래스)하며, 마지막으로 4) 항해를 통해 실제결과를 생성하여 반환한다(SelectStatement 클래스). 그림 3은 검색 처리에 사용되는 구조이다.

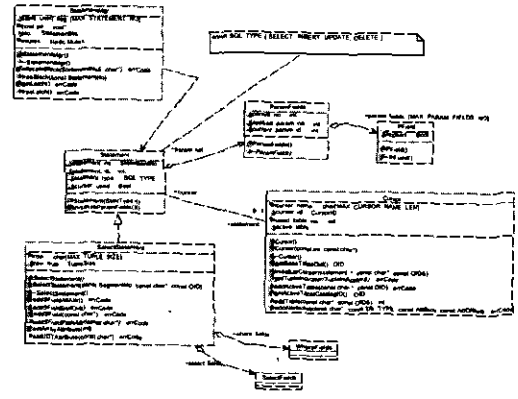
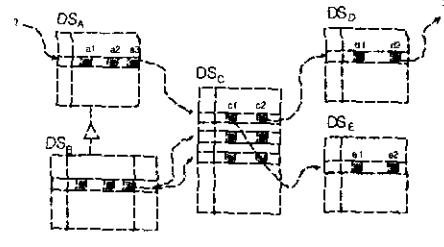


그림 3. 검색 질의 처리를 위한 구조

항해기반 커서는 질의내의 각 경로(path)를 구분하고, 경로에 따라 항해하기 위한 항해 제어 방법이 필요하다. 경로는 직/간접적인 자기참조(self-referencing)가 가능하기 때문에, 각 클래스에 대한 진입경로(in alias)와 진출경로(out alias)로 구분하여 유지한다. 질의내에서 경로를 유일하게 구분하기 위해, 경로는 (클래스id, 속성id, 진출경로id)로 구성한다.




```
SELECT   a3->c2->d2->a2, a3->c1
FROM     DSA
WHERE    a3->c2->d1 = 2630
```

그림 4. 질의 예제

예를 들어, 그림4의 질의를 파싱(parsing)한 결과, DS_C는 커서내에 한개의 진입경로($(DS_A, a_3, 0)$)와 두개의 진출경로($(DS_D, c_1, 0)$와 $(DS_E, c_2, 0)$)를 갖게 된다. 그 결과,

진출경로는 DS_C의 작업 데이터(active object)가 변경될 경우, 같이 변경되어야 할 데이터에 대한 정보를 제시하며, 또한 DS_C의 작업 데이터에 대한 항해가 완료했을 경우, 진입경로를 통해 DS_A(혹은 DS_B의)의 다음 데이터로의 항해를 요구하기 위해 필요하다. 이러한 제어에 따라 클래스들에 대해 물리적인 재구성 없이 조인 결과를 얻을 수 있다. 그림 5는 경로를 사용해 항해하기 위해 필요한 진입경로와 진출경로, 그리고 데이터에 접근하기 위한 세부 정보를 보이고 있다.

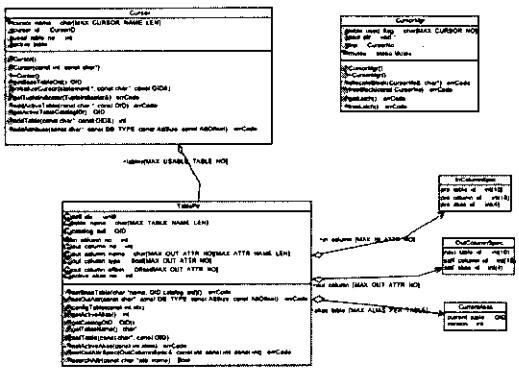


그림 5. 커서의 명세리스트 정보

다음으로, 질의에 명시된 경로는 비교절인가 검색절인가에 따라 항해하는 방식이 다르다. 비교절의 경우, 모든 항해 가능한 경로중에 하나의 식이라도 만족하지 않는 경우 해당 항해 경로는 질의의 결과가 아니다. 반면에 검색절은 가능한 모든 항해 경로가 질의의 결과가 된다. 또한, 비교절과 검색절에 같은 경로가 명시된다면 검색절은 비교절의 해석 방식에 따라야 한다.

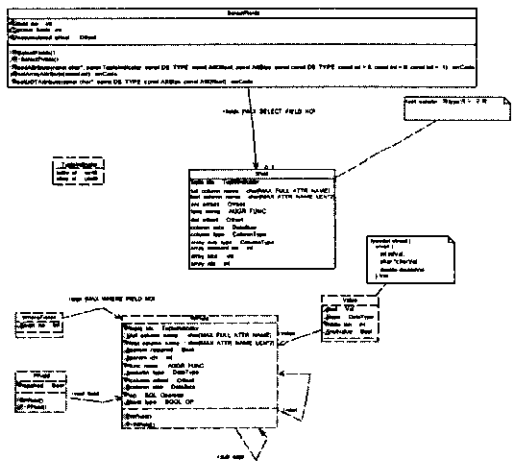


그림 6. 비교절과 검색절의 표현 구조

비교절 구조(WhereFields와 WField 클래스)는 검색절(SelectFields와 SField 클래스)에 비해 값(Value 클래스)과 파라메터 구조(ParamFields와 PField 클래스)를 추가로 가진다.

비교절은 결과의 조건을 명세하는 식들의 조합으로, 그림 7은 비교절의 파싱과정을 보인다. 만일, 식의 값이 파라메터('?')로 명시된다면 이후 입력될 값과 연계하기 위한 부가적인 정보(PField와 연동)를 생성한다.

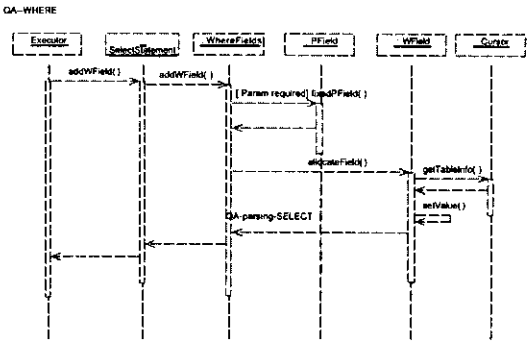


그림 7. 비교절 파싱 절차

질의내 파라메터의 사용은 교환기 데이터 초기화와 같이 동일한 구조의 다수 데이터 처리에 유용하게 사용될 수 있다.

4. 결론 및 향후 연구 과제

본 논문에서는 검색된 결과내에 존재하는 중첩된 구조를 갖는 데이터들에 대해 정형화 된 접근체계와 인터페이스를 제공할 수 있도록 한 항해기반 커서에 대해 기술하였다. 그 결과, 클래스에 대한 조인 연산시 주기억장치 상에 부가적인 저장구조를 만들지 않고, 항해를 통해 대상 데이터에 직접 접근함으로써 불필요한 저장구조의 생성으로 인한 저장 공간의 낭비 현상을 방지하고, 전체적으로 균일하고 빠른 실행 속도를 제공할 수 있다.

참고문헌

[1] Y.I. Yoon, et. al., "Scalable Distributed Real-time Database Management for Switching System," ISS 97, pp. 539-545, Sep. 1997.

[2] Centura Software Co., *Raima Database Manager++ In Real-time and Embedded Systems*, White Paper,

[3] C. T. Yu and W. Meng. *Principles of Database Query Processing for Advanced Applications*, San Francisco: Morgan Kaufmann Publishers, 1998.

[4] S. K. Kim et. al., "A Design and Implementation of Object-oriented Compiler for the Telecommunications System Software," ITC-CSCC, 1998.

[5] TACHYON 사용자 메뉴얼, http://tachyon.etri.re.kr/, ETRI, 2001.