관계 데이터베이스와 객체 데이터베이스의 상호 질의를 위한 중개자의 설계

장인기[0] 배명남[1] 조완섭[2] 이충세[0] 최완[1]
충북대학교 전자계산학과[0], 한국전자통신연구원 실시간데이터처리팀[1], 충북대학교 경영정보학과[2],
cig2005@shinbiro.com, mnbae@etri.re.kr, wscho@trut.chungbuk.ac.kr, csrhee@cbucc.chungbuk.ac.kr,
wchoi@etri.re.kr

Design of a Mediator for Query Interoperation Between Object-Oriented Databases and Relational Databases

In-Ki Jang[0] Myung-Nam Bae[1] Wan-Sup Cho[2], Chung-Se Rhee[0] Wan Choi[1]
[0]Dept. of Computer Science, [2]Dept. of MIS, Chungbuk National Univ.,
[1]Real-Time DBMS Team, ETRI

요 약

응용 환경에서 동시에 이종의 모델을 지원하는 다수의 데이터베이스에 대하여 이들을 동시에 사용하거나 서로 간의 데이터 이전을 위해서는 스키마 변환과 질의 변환이 필요하다. 본 논문에서는 이종 모델 데이터베이스간의 상호 질의를 위해, 1)관계형 스키마를 객체지향 스키마로 변환하고, 2)관계 데이터베이스의 데이터를 객체 데이터베이스로 이전하며, 3)사용자의 관계형 질의를 객체 데이터베이스에서 인식할 수 있도록 변환하는 수행하는 중개자(Mediator)의 설계에 대해 기술한다. 제안된 중개자는 관계형 SQL 질의를 받아서, 데이터의 위치에 따라 적절한 질의로 변환한 후, 관계 데이터베이스와 객체 데이터베이스에 있는 데이터를 자동으로 처리한다. 결과적으로, 중개자는 서로 다른 데이터베이스를 동시에 사용할 수 있는 기초가 될 것이다.

1. 서론

관계 데이터베이스는 데이터를 저장하고 관리하며 검색하기 위해 널리 사용되어 왔다. 하지만, 객체지향 데이터 모델링에 대한 요구가 증가하면서 기존의 관계 데이터베이스와 새로이 구축되는 객체 데이터베이스 간의 상호 운용의 필요성이 증가하게 되었다. 이러한 경우, 데이터 모델의 차이로 인해서 스키마와 질의의 변환이 필요하며, 각 데이터베이스와 연관된 응용의 변경도 함께 필요하게 된다. 결과적으로, 데이터베이스와 관련한 관리 및 개발 비용의 증가를 초래하게 된다. 이러한 문제를 해결하기 위해서는 사용자에게 하나의 데이터베이스 인터페이스를 제공하면서 실질적으로는 두 개 이상의 데이터베이스가 연동될 수 있도록 하는 중개자[6]의 도입이 요구된다. 중개자는 두 데이터베이스 간의 스키마 변환을 자동으로 처리하며, 사용자에게 모델 투명성을 제공할 수 있어야 한다.

기존의 스키마 변환 연구[8,9,11,12]에서는 객체지향 스키마를 관계형 스키마로 변환하거나, 외래키의 변환을 위해서 참조(reference)나 집합(set) 속성을 사용하였다. 질의 변환 연구[5,7,12]에서는 객체 데이터베이스의 질의를 SQL 질의로 변환하는 연구가 주를 이루었다. 중개자에 대한 연구[6,7,9]는 어느 특정 응용에서 사용하기 위한 연구와 일부분만을 처리하는 방법에 대한 연구가 주로 이루어 졌다.

본 연구에서는 한국전자통신연구원에서 개발한 객체지향 모델 특성을 지닌 Tachyon[2] 데이터베이스관리시스템이 기존의 관계 데이터베이스 관리시스템과 함께 사용될 수 있도록 하는 중개자의 설계 및 구현을 다룬다. 먼저, 스키마 변환에서는 외래키의 변환을 위해 OID[10]에 대한 단일(single) 참조나 다중(multiple) 참조의 이용 이외에도, 역(inverse) 참조를 이용한 변환 방법[13]을 사용한다. 다음으로 질의 변환에서는 제약조건 그래프[13]에서 그래프 생성 방법과, 질의 변환 방법을 발전시킨, 질의 그래프(Query Graph; QG)를 이용한 경로 표현으로 질의 변환을 수행하게 된다.

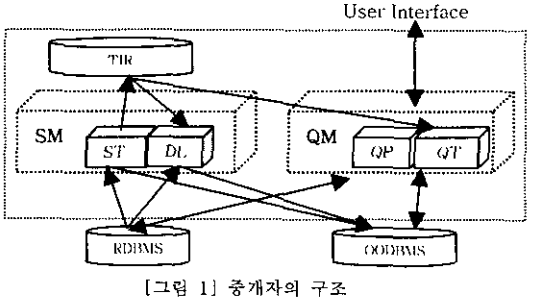
본 연구의 결과는 이종 모델 데이터베이스간의 연동을 위한 기초 연구가 될 수 있다. 특히, 관계 데이터베이스와 객체 데이터베이스를 동시에 사용하지만, 사용자에게는 객체 데이터베이스를 인식하지 않아도 되도록 투명성을 제공한다. 그리고 관계 데이터베이스에 대한 응용에서도 객체 데이터베이스로 인한 수정의 필요성을 제거해 준다.

본 연구에서 관계 데이터베이스로는 Oracle[1]을 이용하며, 메모리 상주 데이터베이스인 Tachyon(Main Memory-Resident Object-Relational Database)을 이용한다. Tachyon은 클래스, 질의, 상속과 같은 객체지향 특성을 지원하는데, OID와 관련하여, OID에 대한 단일 참조를 나타내는 *OID_REF*, 다중 참조를 나타내는 *OID_SET*, 그리고 각 참조에 대한 역 참조 등이 있다. 중개자의 설계 언어는 UML[3]을 사용하였고, 데이터베이스와의 연결은 JDBC[4]를 통해 이루어 진다.

본 논문의 2장에서는 중개자의 구조에 대해서 살펴본다. 3장에서는 스키마 변환 방법, 4장에서는 질의 변환의 방법에 대해 살펴본다. 그리고 5장에서는 본 논문의 결론 및 향후 과제에 대해서 살펴 본다.

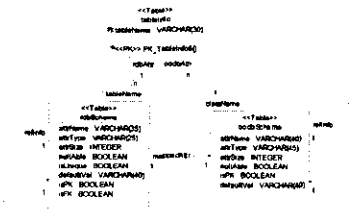
2. 중개자의 구조

[그림 1]은 중개자의 구조이다. 중개자는 크게 *TIR*(Translated Informations Repository), SM(Schema Manager), 그리고 QM(Query Manager)으로 구성된다. SM 내의 스키마 변환기(Schema Translator:ST)가 관계형 스키마를 객체지향 스키마로 사상하게 된다. 이때, 관계형 스키마와 변환된 객체지향 스키마는 사상 정보 저장소(*TIR*)에 저장된다. 데이터 로더(*Data Loader; DL*)는 *TIR*의 정보를 이용해서 관계 데이터베이스의 데이터를 객체 데이터베이스로 이전하게 된다. 데이터 이전이 완료된 후, 사용자의 SQL 질의는 질의 관리자(Query Manager:QM)에서 질의를 구별하여 각 데이터베이스로 보내게 된다. 이때, 질의가 객체지향 데이터베이스에 대한 질의일 경우에는 질의 분석기(*Query Parser:QP*)에서 질의를 분석하고, 이 정보를 가지고 질의 변환기 (Query Translator;QT)[6]에서 질의 변환을 수행하게 된다.



[그림 1] 중개자의 구조

TIR은 중개자에서 핵심 역할을 하게 되며, [그림 2]와 같이 세 부분으로 나누어져 있다. rdbSchema는 관계형 스키마 정보를 저장하며, oodbSchema에는 관계형 스키마에서 객체지향 스키마로 변환된 정보가 저장된다. tableInfo는 rdbSchema와 oodbSchema에서 사용된 테이블과 클래스의 정보를 저장한다. 이 세 개의 클래스는 상호 간의 참조 관계를 통해서 정보를 검색할 수 있도록 설계된다.

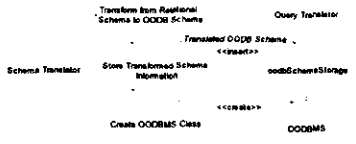


[그림 2] TIR의 구조

중개자는 물리적으로 데이터베이스시스템에 종속적이지 않고, Tachyon이 빠른 검색 속도를 제공하기 때문에 TIR를 Tachyon에 유지하도록 구현하였다. 이를 통해, TIR의 정보를 계속해서 사용할 수 있도록 했다.

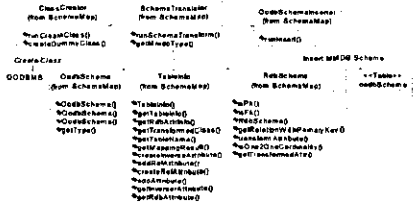
3. 스키마 변환기

스키마 변환은 [그림 3]과 같은 과정들로 이루어져 있으며, 크게 관계형 스키마를 객체지향 스키마로 변환하는 과정, 변환 스키마 정보를 TIR에 저장하는 과정, 그리고 실제 클래스를 생성하는 세 과정으로 나눌 수 있다.



[그림 3] 스키마 변환 과정

위의 세 과정은 모두 ST에서 이루어지게 되며, ST를 구성하고 있는 클래스들은 [그림 4]와 같이 정의할 수 있다.

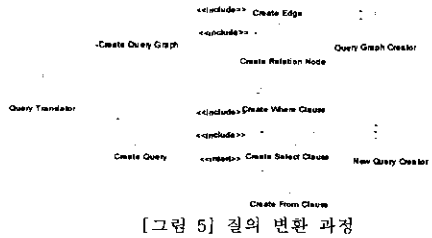


[그림 4] 스키마 변환기의 간략한 클래스 구조도

[그림 4]에서 각 클래스의 속성은 생략했으며, 각 클래스의 메소드(Method)들은 스키마 변환 알고리즘[13]과 실제 스키마를 변환하기 위해서 클래스간 상호 작용을 위해서 정의되었다.

4. 질의 변환

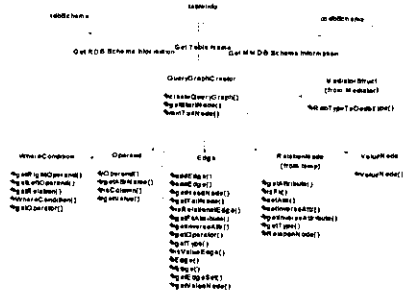
질의 변환은 QT에서 이루어 지며, [그림 5]와 같이, QG를 생성하는 과정과 이를 이용해 실제 질의 변환을 하는 두 과정으로 나눌수 있다.



[그림 5] 질의 변환 과정

4.1 질의 그래프 생성 및 시작 노드의 선정

QG를 생성하기 위한 클래스들은 [그림 6]과 같이 정의된다. 이 클래스들은 질의를 변환하는 전 과정에서 사용된다.



[그림 6] 질의 그래프 생성을 위한 클래스 구조도

질의 그래프에서 간선은 WHERE 절의 각 조건을 나타내고, 노드는 WHERE 절의 각 조건에서 사용된 릴레이션을 나타낸다. QG는 간선의 집합과 각 간선에 사용된 노드들의 집합을 이용해서 만들어 진다. 각 간선은 <외래키노드,주키노드> 혹은 <애트리뷰트노드, 값 노드>의 조합으로 이루어지며, 방향성이 있는 그래프가 된다. QG는 트리 형태의 그래프로, 모든 간선을 검색하기 위해서는 시작 노드를 선정해야 한다. 질의 그래프 생성과 시작노드를 선정하는 방법을 개략적으로 나타내면, 알고리즘-1과 같은 방법으로 만들어지게 된다.

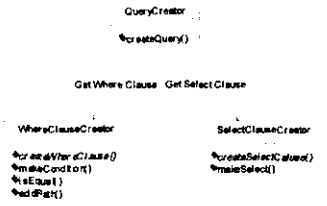
알고리즘-1 : 질의그래프 및 시작노드 선정

```
begin :
for 모든 WHERE 조건
  .각 조건의 Operand를 얻어온다.
  .각 Operand에서 사용된 Attribute와 Relation을 TIR에서 얻어
    온다.
  if(조건이 Column 조건)
    if(조건에서 사용된 Attribute가 Foreign Key)
       .이 Attribute와 Relation에 대한 INVERSE Attribute를
        TIR에서 얻어온다.
       .각 노드에서 사용된 Attribute와 Relation, INVERSE
        Attribute, 그리고 Relation Edge를 나타내는 정보를 포
        함하고, 이 노드를 Tail Node로 하여, Edge 집합에 새로
        운 Edge를 추가한다.
    end if
  else(조건이 Value Select일 경우)
    .이 조건에서 사용된 Value와 Operator를 얻어온다.
    .이 조건에서 사용된 Attribute의 변환된 Attribute를 TIR에
     서 얻어온다.
    .이 Attribute의 Data Type 정보를 TIR에서 얻어온다.
    .위의 모든 정보를 포함하여 Value를 Head Node로 하고,
     Value Edge를 나타내는 정보를 포함하여 Edge 집합에 새로
     운 Edge를 추가한다.
  end if
end for
  .Edge 집합에서 Head Node에는 사용되지 않고 Tail Node에만
   사용된 Node를 얻어온다.
  .이 Node 중에서 가장 적에 나타나는 노드를 선택하고, 두 개
   이상일 경우에는 첫번째 나타나는 노드를 선택한다.
  .이 노드를 QG의 시작노드(Root Node)로 선정한다.
End :
```

4.2 질의 변환

질의 변환은 크게 SELECT, FROM, WHERE 절의 세 부분으로 나눌 수 있다. 이 중에서 FROM 절에는 알고리즘-1에서 선정되는 시작노드를 정의해 준다. SELECT 절과 WHERE 절은 시작노드에서 시작되는 경로 표현을 통해서 변환된다. QG의 경로 탐색은 트리의 경로 탐색 방법을 재귀적으로(recursively) 수행하게 된다. 질의 변환을 위해서는 [그림 7]과 같은 추가적인 클래스들이 필요하게 되며,

[그림 6]의 클래스와 같이 작용한다. 알고리즘-2에서는 WHERE 절 및 SELECT 절을 변환하기 위한 방법을 간단한 설명한다.



[그림 7] 질의 변환을 위한 추가 클래

알고리즘-2 : WHERE 절의 변환 알고리즘

begin :

. 시작노드로 초기 WHERE절을 위한 PATH를 생성한다.
for Edge집합에 있는 모든 Edge
. Edge의 Head Node와 Tail Node를 얻어온다.
if(Egde가 Relational Edge이고 Tail Node가 WHERE절의 시작노드와 같다)
. Edge에서 Foreign Key 속성의 attribute를 얻어온다.
. 이 Foreign Key 속성의 Attribute를 PATH에 추가한다.
else if(Edge가 Relational Edge이고 Head Node가 WHERE절의 시작노드와 같다)
. Edge에서 INVERSE Attribute를 얻어온다.
. INVERSE Attribute를 PATH에 추가한다.
else if(Edge가 Value Edge이다)
. Edge에서 Value를 얻어온다.
. Edge에서 Operator를 얻어온다.
. Edge에서 Data Type을 얻어온다.
. 앞에서 만들어진 PATH와 Operator, Value, Data Type을 가지고 WHERE 절을 완성한다.
end if
end for

end :

4.3 질의 변환 예

질의 변환 예를 소개하기 위해 다음과 같은 관계형 스키마를 정의하였다. 여기에서, Primary Key는 밑줄로, Foreign Key는 괄호안에 정의한다. 예제 스키마는 Company에 대한 전형적인 모델이다.

Department : dnumber, dname, (mgrssn : Employee.ssn)
Employee : ssn, ename, (dno :Department.dnumber)
Project : pnumber, pname, (dnum : Department.dnumber)
Workon : (essn : Employee.ssn), (pno : Project.pnumber), hours

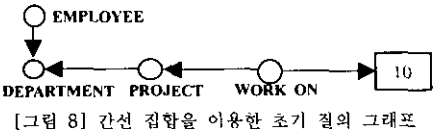
질의 : 이 스키마에서 10시간 이상 근무한 직원의 이름, 부서명, 참여한 과제명을 검색하라.

```
SELECT    Employee.ename, Department.dname, Project.pname
FROM      Employee, Department, Project
WHERE     Employee.dno = Department.dnumber
  AND     Department.dnumber = Project.dnum
  AND     Project.pnumber = Workon.pno
  AND     Workon.hours > 10;
```

WHERE 절의 조건에 대해서 간선의 집합 E는 다음과 같이 정의된다.

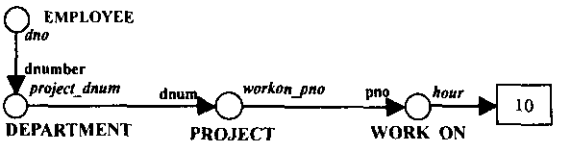
E ={<EMPLOYEE, DEPARTMENT>, <PROJECT, DEPARTMENT>, <WORKON, PROJECT>, <WORKON, 10> }

E를 그래프의 형태로 나타내면 [그림 8]과 같다.



[그림 8] 간선 집합을 이용한 초기 질의 그래프

위의 그래프에서 알고리즘-1의 결과로 시작노드는 EMPLOYEE가 되며, 알고리즘-1과 알고리즘-2를 적용한 후, 각 노드에 애트리뷰트를 명시하면 [그림 9]와 같은 최종 QG가 생성된다.



[그림 9] 완성된 질의 그래프 예

[그림 9]의 각 노드 중 " _ " 가 있는 애트리뷰트는 역 참조 속성의 애트리뷰트를 나타내며, 각 간선의 꼬리(tail) 노드에 명시된 애트리뷰트들이 경로 탐색에서 이용된다. 최종 QG를 이용한 최종 질의 변환 결과를 아래와 같다. FROM절에는 시작노드 선정된 Employee를 명시하며, SELECT절과 WHERE절은 QG의 경로 표현을 통해서 변환된 결과이다.

```
SELECT    ename, dno->dname, dno->project_dnum->pname
FROM      Employee
WHERE     dno -> project_dnum -> workon_pno -> hours > 10
```

5. 결론 및 향후 과제

본 논문에서는 기존의 연구를 발전시켜, 스키마 변환에서는 단일 참조와 다중 참조, 그리고 역참조를 통한 변환을 수행하였고, 질의 변환에서는 질의 그래프 생성시에 역참조 속성의 사용과 시작노드 선정을 통한 질의 변환을 수행하였다. 또한, 관계-객체 데이터베이스간의 상호 질의를 자동으로 변환하고, 수행할 수 있는 중개자를 설계하였다. 이를 통해, 상호 이질적인 데이터베이스 간의 스키마 변환을 자동으로 처리해 줄 수 있으며, 관계형 질의만으로도 객체 데이터베이스에 접근할 수 있게 되었다. 이는 서로 다른 모델의 데이터베이스를 동시에 사용하거나, 한 데이터베이스에서 다른 데이터베이스로의 데이터의 이전 등에 사용될 수 있다.

향후에는 중개자를 통해서 관계 데이터베이스에서 객체 데이터베이스로의 변환 뿐만 아니라, 객체 데이터베이스에서 관계 데이터베이스로의 변환에 대한 연구가 필요하다. 또한, 질의 그래프에서 시작 노드의 선정에 따른 비용 및 검색 성능에 대한 명확한 검증이 필요할 것이다.

5. 참고 문헌

[1] Oracle, White Paper, "http://www.oracle.com"

[2] Tachyon, White Paper, "http://tachyon.etri.re.kr"

[3] UML, White Paper, "http://www.omg.org/uml/"

[4] Java and JDBC, White Paper, "http://www.java.sun.com"

[5] C. Yu, Y. Zhang, W. Meng, W. Kim, G.. Wang, T. Pham, and S. Dao, "Translation of Object-Oriented Queries to Relational Queries," In Proc. Int'l Conf. on Data Engineering, pp. 90-97, 1995.

[6] G. Wiederhold, "Mediators in the Architecture of Future Information Systems," IEEE Computer, pp. 38-49, 1992.

[7] H. Huang, J. Kerridge, and S. Chen, "A Query Mediation Approach to Interoperability of Heterogeneous Databases," In Proc. on Database Conference, pp. 41-48, 1999.

[8] J. Fong, "Converting Relational to Object-Oriented Databases," SIGMOD Record, Vol. 26, No.1, pp. 53-58, Mar. 1997.

[9] L. Stoimenov, A. Mitrovic, D. Slobodanka, and D. Mitrovic, "Bridging objects and relations: a mediator for an OO front-end to RDBMSs," Information and Software Technology, Volume 41, pp. 57-66, 1999.

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

[11] W. Meng, A. Kamada, and Y. Chang, " Transformation of Relational Schemas to Object-Oriented Schemas," In Proc. Int'l Conf. on Computer Software and Applications, pp. 356-361, 1995.

[12] W. Meng, C. Yu, W. Kim, G. Wang, T. Pham, and S. Dao, "Construction of a Relational Front-end for Object-Oriented Database Systems," In Proc. Int'l Conf. on Data Engineering, pp. 476-483, 1993.

[13] 장인기, 공희경, 이충세, 조완섭, 최완, "관계형 데이터베이스에서 Tachyon 객체관계형 데이터베이스로의 변환기법," 한국정보과학회 추계학술발표회, pp. 254-256, 2000.