TGQL : Tachyon ODBMS를 위한 그래픽 질의어의 설계 및 구현†

안명상[1] ,이충세[1] ,경원현[2] ,조완섭[3]
[1]충북대학교 전자계산학과, [2]충북대학교 정보산업공학과, [3]충북대학교 경영정보학과

A Design and Implementation of GQL for Tachyon ODBMS

Myoung-Sang Ahn[1], Chung-Sei Rhee[1] , Won Hyun Kyung[2], Wan-Sup Cho[3],
[1]Dept. of Computer Science, ChungBuk Nat'l Univ.,[2]Dept. of IIE, ChungBuk Nat'l Univ., [3]Dept. of MIS, ChungBuk Nat'l Univ.

요 약

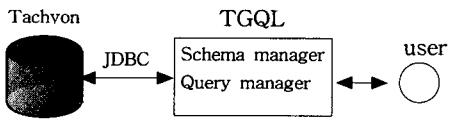
데이터베이스 전문 사용자가 아닌 일반 유저들도 쉽게 데이터베이스에 접근하고자하는 요구들이 늘어나고 있다. 이러한 요구사항을 만족시킬 수 있는 방법 중 하나가 Graphical Query를 사용하는 것이다. Graphical Query는 유저에게 Schema browsing 메커니즘을 제공하여 텍스트를 기반의 질의보다 쉽게 데이터베이스에 접근할 수 있기 때문이다. 본 논문에서는 Tachyon OODBMS를 위한 그래픽 질의어 TGQL(Tachyon Graphical Query Language)의 설계 및 구현에 대해 서술한다. TGQL은 객체 DBMS를 위한 그래픽 질의어이므로 객체지향 개념을 어떻게 그래픽 질의어에 시각적인 요소들로 반영시키는가 라는 문제가 중요하게 다루어진다. 본 논문은 이러한 측면에서 TGQL의 특징들을 기술한다.

1. 서론

본 논문은 Tachyon 그래픽 질의어(TGQL:Tachyon Graphical Query Languge)의 설계 및 구현에 대해서 서술한다. 텍스트 기반의 질의어의 단점은 전문적인 사용자가 아니면 문법이 복잡하며, 또한 사용지는 데이터베이스 스키마에 대한 정확한 이해가 필요하다. 그래픽 질의어는 텍스트 질의어인 SQL보다 편리한 사용자 인터페이스를 제공한다. 즉 GQL(Graphical Query Language)사용자는 데이터베이스 스키마에 대한 정보나 SQL구문을 모르는 상태에서도 질의어를 작성할 수 있다. 특히 Tachyon 실시간 ODBMS의 경우 기존의 SQL을 확장한 문법이 사용되기 때문에 일반 사용자에게 구체적인 Tachyon 질의 문법을 알지 못하더라도 쉽게 사용하게 할 수 있는 질의어 작성 Tool이 필요하다. TGQL은 위와 같은 목적을 위해 설계, 구현되었다.

2. 시스템 구조

Tachyon GQL 시스템의 구조는 크게 Schema manager, Query manager 이루어진다. Schema manager는 데이터베이스 스키마를 생성 관리하는 모듈이며, Query manager는 사용자에게 질의를 작성하고 실행하도록 지원한다. [그림1]은 시스템의 전체적인 구조를 보여주고 있다.



[그림 1] TGQL 시스템 구조

[그림 1]에서 사용자는 JDBC을 통하여 Tachyon ODBMS에 연결 후 질의 처리를 요구하고, 질의 처리 결과를 사용자에게 전달하는 과정을 보여주고 있다.

2.1 Schema Manager

이 모듈은 데이터베이스에 클래스와 타입 및 인덱스를 생성하고 , 이들을 삭제하는 기능을 제공한다. 사용자는 database를 선택하고 (하나의 데이터베이스만을 지원하는 시스템의 경우 이 단계를 생략함), 선택된 데이터베이스에 대하여 Schema Manager를 선택하면 Database에 대하여 가능한 스키마 관리 명령어를 선택할 수 있다. 스키마 관리 명령어로는 클래스 생성 (create class), 클래스 삭제 (drop class), 타입의 생성 (create type), 타입의 삭제 (drop type), 인덱스의 생성(create index), 인덱스의 삭제 (drop index)가 있다. 클래스, 타입 및 인덱스를 생성하는 Tachyon SQL 문장의 예는 아래와 같으며, TGQL에서는 적절한 그래픽 인터페이스를 사용하여 이러한 SQL문장이 자동으로 생성되도록 한다.

Q1: 클래스 생성 예

```
Create Class C1 as Under C2(
        A1 int[10] Primary,
        A2 char(20) Default '0',
        A3 de_ref Inverse C3.A4 );
```

Q2: 타입 생성 예

```
Create Type T1(
        A1 int not Null,
        A2 char(10) Default 'chongju',
        A3 bit(10) );
```

Q3: 인덱스의 생성 예

```
Create Unique index i1 on c1 (a1,a2)  Using TTree;//or Hash
```

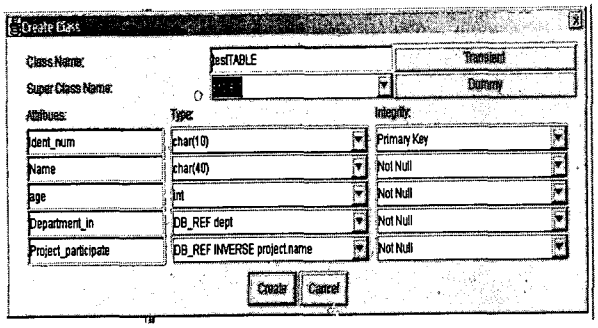
Q4: 삭제 예

```
Drop Class c1;
Drop Type t1;
Drop Index i1;
```

2.1.1 클래스 생성

클래스를 생성하고자 하는 경우 사용자는 Create Class 명령을 선택

하며, 사용자는 [그림2]와 같이 클래스 생성 창을 제공받게 된다. 클래스 생성 창은 기본적으로 Create Class 문장에서 사용자가 입력해야 하는 정보를 대화식으로 받아들일 수 있게 설계되었다. 즉, 클래스와 수퍼 클래스의 이름을 명시할 수 있으며(이 경우 GQL로부터 클래스-이름 리스트를 제공받을 수 있음), 또한 속성에 관한 정보를 입력할 수 있다. 사용자가 속성을 정의하지 않은 채 클래스 이름만 생성하고자 하는 경우에는 Dummy 버튼을 사용하면 된다. 이 경우 속성 정보 입력 부분은 생성되지 않는다. 속성 정보를 입력하는 경우에는 마우스를 조작하여 TGQL이 제공하는 속성 정보 (타입과 Integrity) 리스트로부터



[그림 2] 클래스 생성 인터페이스

선택할 수 있다. 특히, 속성의 타입이 db_ref 혹은 db_set인 경우에는 추가적으로 클래스 이름과 속성 (db_ref Inverse인 경우에 해당함)을 추가로 제공해야 한다. 이 경우에도 TGQL은 클래스 이름(혹은 속성)의 리스트를 제공하고, 사용자로 하여금 선택할 수 있도록 한다.

2.1.2 타입의 생성

타입 생성 과정은 클래스 생성과 거의 유사한 과정을 거친다. 두 경우의 차이점으로는 Create Type 명령에서는 Primary key를 지정할 수 없으며, 데이터 타입의 종류도 Create Class 문장과 달리 다음과 같이 제한된다는 점이다.

```
int             int[?]    // array of integer
smallint        smallint[?]
double          double[?]
real            real[?]
char            char(?)
byte            byte(?)
boolean         boolean[?]
bit(?)          range(? : ?)
```

2.1.3 Create Index 문

인덱스 생성은 클래스와 속성(들)을 선택한 후 인덱스의 성질인 Unique 여부를 지정하고, Hash 혹은 Ttree를 선택함으로써 완성된다. 사용자가 Create Index를 선택하면 GQL 시스템은 사용자에게 인덱스 이름을 요구하고, 클래스 리스트를 제시하면서 인덱스가 생성될 클래스를 선택하도록 요구한다. 사용자가 class를 선택하면 시스템은 다시 class의 속성 리스트를 제시하고 인덱스가 구성될 속성(들)을 선택하도록 요구한다. 클래스와 속성이 선택된 후에는 인덱스의 특성을 지정하는 것으로 Unique 여부와 인덱스 구조를 선택하도록 한다(TTree와 Hash 중의 하나).

2.1.4 Drop Class (Type, Index)

클래스와 타입 혹은 인덱스를 삭제하는 명령은 간단하게 이루어진다. 사용자가 기존의 클래스 리스트중에서 하나를 선택하면 된다. 이 때, 옵션 사항으로 ALL을 선택하면 서브 클래스까지 모두 삭제함을 의미한다.

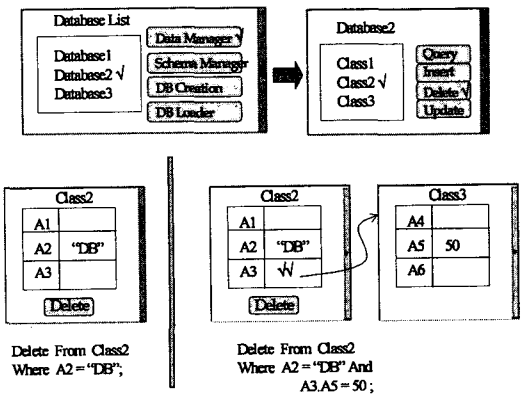
2.1.5 Insert

클래스에 객체를 입력하는 경우에 사용되며, 값을 제공하여 입력하는 경우와 질의 결과를 입력하는 두 경우로 나누어진다. 사용자가 클래스 이름을 선택한 후 Insert를 선택하면 선택된 클래스의 속성 리스트가 나타나고 사용자는 각 속성의 값을 입력한 후, insert 버튼을 눌러서 실행한다. 집합값을 가지는 속성인 경우 집합을 나타내는 괄호 ('{'와 '}')를 사용하여 원소를 나열한다. 속성의 도메인이 다른 클래스인 경우 더블 클릭하여 도메인 클래스 창을 띄운 후, 도메인 클래스의 객체를 먼저 insert하고, 그 다음에 원래 클래스의 객체를 입력한다. 이 경우, db_set 타입인 경우 다수의 창을 띄워 다수의 객체를 여러개 입력하도록 한다. 한편, 질의의 결과를 객체의 속성 값으로 insert 하는 경우 먼저 "Insert With Query" 버튼을 선택하며, 그 결과 insert 문과 select 문이 결합된 형태로 GQL이 나타난다. 사용자가 속성창에서 오른쪽 버튼을 누르면 질의 창이 생성되고, 질의 창에 질의를 완성한 후 실행 버튼을 누르면 질의에 해당하는 SQL문이 작성되고, 이것이 Insert 문장에 포함되어 사용자가 의도했던 Insert 문을 완성한다.

2.1.6 Delete 문장

GQL의 Delete 문은 조건을 만족하는 객체(들)을 모두 삭제하는 것이다. 삭제될 객체에 해당하는 Where 조건은 다음 두 가지로 구분할 수 있으며, Where 절 자체에 대한 GQL 형태는 Query 문장의 Where 절과 동일한 창을 이용한다.

- 단일 클래스에 대한 조건 [그림3] 하단의 왼쪽
- 경로식에 대한 조건 [그림3]하단의 오른쪽



[그림 3] 단일 클래스 및 경로식에 대한 조건

2.1.7 Update 문장

GQL의 Update 문은 조건을 만족하는 객체(들)을 선택한 후 주어진 값으로 속성값을 변경하는 명령이다. Update 문장에서도 변경될 객체에 해당하는 조건은 Delete에서와 마찬가지로 단일 클래스에 대한 조건과 경로식에 대한 조건 두 가지 형태로 명시할 수 있다.

2.2 Query Manager

GQL에서 질의 작성 과정은 데이터베이스 선택, 클래스 선택, 질의 조건 작성의 세 단계를 거쳐서 이루어진다. 질의 조건의 작성은 Select 문을 구성하는 다섯 개의 절을 차례로 작성함으로써 완성된다. 다섯 개의 절 중에서 Where, Group By, Having, Order By 절은 선택적이며, 실제로 Tachyon 에서는 이들 중에서 Select 절, From 절, Where 절만 지원된다. 사용자가 클래스를 선택하고 Query를 클릭하면 시스템은 All 혹은 Only를 추가로 선택하도록 요구한다. All의 경우 선택된 클래스의 하위 클래스까지 검색 대상으로 지정하는 경우이고, Only의 경우는 하위 클래스를 검색 대상에서 제외함을 의미한다 (default로는 only임). 다음으로 사용자는 Select 문의 나머지 절들을 선택하여 질의 작성을 완성한다.

2.2.1 Where 조건의 작성

사용자가 Where 버튼을 누르면 질의 조건을 작성하는 창으로 이전한다. TGQL에서는 Where절의 조건들이 모두 AND 연산자로 연결되었다고 가정하며, 각 조건은 "a1.a2.a3 = 30"과 같이 경로식에 대한 조건(간단히 **경로 조건** 이라고 부름)의 형태이다. 경로는 다음과 같이 단일값 경로와 집합값 경로로 구분된다. 단일값 경로란 경로에 포함된 모든 속성들이 단일값을 가지는 경우로써 단일값이 반환된다. 반면, 집합값 경로는 경로에 포함된 속성중에서 어느 하나라도 집합값 속성인 경우(즉, 중간 속성의 타입이 db_set이거나 마지막 속성이 배열인 경우)로써 집합값을 반환한다.

```
dept.manager.age // 단일값 경로식
dept.employee.age // 집합값 경로식
```

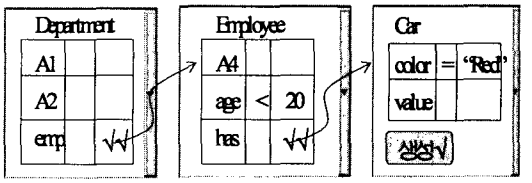
경로식에 대한 조건은 단일값 경로식과 집합값 경로식을 구분하여 작성해야 한다. 단일값 경로식에 대한 조건은 스칼라 연산자를 사용하며, 집합값 경로식에 대한 조건은 집합 연산자를 사용한다. 스칼라 연산자는 =, !=, >=, >, =<, < 가 있으며, 집합값 연산자에는 superset, superseteq, subset, subseteq, seteq, setnoteq 등이 있다. 아래는 위의 두 경로식에 대한 경로 조건의 예이다.

```
dept.manager.age > 30;// 스칼라 연산자
dept.employee.age ⊇ 19;// 집합 연산자
```

GQL에서 질의를 작성하는 경우, 단일값 경로에 대한 조건과 집합값 경로에 대한 조건을 구분하여 작성해야 한다. 전자의 경우 스칼라 연산자를, 후자의 경우 집합 연산자를 사용해야 하기 때문이다. 집합 연산자의 의미를 ODMG 등의 표준에 적합하도록 지원하기 위하여 집합의 각 원소에 대한 조건도 명시할 수 있어야 한다. 예를들어, 다음 질의어는 20세 미만의 미성년자를 (한 명이라도) 고용한 부서를 찾고, 그 미성년자가 가진 자동차의 color가 ""Red"이면, 부서명과 그 미성년자 이름을 리턴하는 질의이다. 여기서 객체변수 e는 특정 department에 소속된 직원 개개인과 바인딩되며, c는 e가 가진 자동차 각각에 대하여 바인딩된다. 질의 처리에서는 e.age와 e.color가 Where 절에 명시된 조건을 만족하면 그 때의 department 이름과 employee이름을 리턴한다.

```
Select   d.name, e.name, c.value
From     Department d, d.emp e, e.has c
Where    e.age < 20 AND c.color = Red
// 집합 원소에 대한 질의 예
```

이러한 질의의 GQL 표현은 아래 그림과 같이 표현된다. [그림 4]에서는 Department의 emp 속성이 집합값을 가지더라도 Employee.age에 대한 조건을 스칼라 연산자로 표시하여야 한다. 즉, 집합에 대한 연산이 아니라 집합의 각 원소에 대한 연산을 표현한 것이다.



[그림 4] 집합값에 대한 표현식

2.2.2 Projection 작성

프로젝션 절에는 From 절에 나타난 클래스의 경로식/속성(들), 경로식/속성에 대한 집계 함수(COUNT, MAX, MIN, AVG, SUM), 경로식/속성 혹은 그들에 대한 연산이 올 수 있으나 Tachoyn에서는 다음으로 한정하고 있다.

*, OID, column_name or path, COUNT(*), AVG, MAX, MIN, SUM

여기서 * (OID)는 From 절에 나타난 클래스에서 조건을 만족하는 객체의 속성전부를 의미한다. path (경로식)는 From 절에 나타난 클래스로부터 속성-도메인 관계를 사용하여 작성한 경로식을 의미한다. 사용자가 Project 버튼을 선택하면 TGQL은 From 절에 나타난 클래스의 속성과 그 도메인을 보여준다. 사용자는 제시된 클래스에서 속성을 선택하여 그 속성을 projection list에 첨가하거나, 도메인 부분을 더블 클릭하여 도메인에 해당하는 클래스를 띄워서 도메인 클래스의 속성을 선택할 수도 있게 한다. 또한 사용자는 이미 projection list에 포함된 속성 (혹은 경로)에 대하여 집계 함수를 추가로 지정할 수 있다.

3. 결론 및 향후 과제

본 논문에서 Tachyon OR-DBMS을 위한 Graphical Query Tool 설계 및 구현에 대하여 설명하였다. 향후, 연구 방향으로는 질의 결과로 리턴된 객체들에 대하여 속성의 값으로 포함된 OID를 따라서 항해(navigation)하는 방법에 대한 연구가 필요하다. 또한 UML을 이용하여 데이터베이스 Schema를 관리할 수 있는 User Interface가 필요하다.

[참고 문헌]

[1] Herman Lam,H. More Chen " A Graphical INterface for an Object-Oriented Query Language,". in IEEE, 1900.
[2] Richard Gary epstein " A Grahical Query language for Object-Oriented Data Models." IN IEEE 1900
[3]Seong-Woo Chang, Suk-Ho Lee Hyoung-Joo Kim "SOPView: A Visaul Query and Object Browing Environment for SOP OODBMS." in IEEE 1996