Tạp chí Tin học và Điều khiển học, T.20, S.3 (2004), 257–269

SỰ TƯƠNG ĐƯƠNG TRONG BIỂU DIỄN GIỮA NGÔN NGỮ TRUY VẤN OQL VÀ ĐẠI SỐ ĐỐI TƯỢNG

ĐOÀN VĂN BAN[1], LÊ MẠNH THẠNH[2], HOÀNG BẢO HÙNG[2]

[1] *Viện Công nghệ thông tin*
[2] *Khoa CNTT, Trường Đại học Khoa học Huế*

Abstract. In inheriting the success and popularity from the SQL relational query language, most object - oriented database (OODB) query languages have been using a syntax which is similar to that of SQL. OQL is the OODB query language that was proposed in ODMG-93. The last version of OQL, as in ODMG-93 (Release 1.2), was the superset of SQL92. In this paper, we wish to present the establishment of OODB queries in OQL, and object algebra as well as to prove a equivalent expression between the OODB query language OQL and the object algebra. These results are the bases for researching on processing and optimizing OODB queries.

Tóm tắt. Kế thừa sự thành công và tính phổ dụng của ngôn ngữ truy vấn quan hệ SQL, nên hầu hết các ngôn ngữ truy vấn CSDL hướng đối tượng được đề xuất đều sử dụng một cú pháp tương tự như SQL. Trong đó, OQL là ngôn ngữ truy vấn cơ sở dữ liệu (CSDL) hướng đối tượng đã đề xuất trong ODMG-93. Phiên bản cuối cùng của OQL, như trong ODMG-93 (Release 1.2), là siêu tập (superset) của SQL92. Trong giới hạn và khuôn khổ bài báo chúng tôi muốn trình bày thiết lập các truy vấn CSDL hướng đối tượng trong OQL, đại số đối tượng, chứng minh sự tương đương trong biểu diễn giữa ngôn ngữ truy vấn OQL và đại số đối tượng, điều này sẽ là cơ sở cho quá trình nghiên cứu về xử lý truy vấn và tối ưu hoá truy vấn CSDL hướng đối tượng.

1. OQL - NGÔN NGỮ TRUY VẤN CSDL HƯỚNG ĐỐI TƯỢNG

1.1. Giới thiệu

OQL là một ngôn ngữ truy vấn CSDL hướng đối tượng đã đề xuất trong ODMG-93. Phiên bản cuối cùng của OQL, như trong ODMG-93 (Release 1.2), là siêu tập của SQL92. OQL không giới hạn cú pháp 3 mệnh đề select-from-where, hầu hết các truy vấn OQL không tầm thường đều có trên ba mệnh đề.

Với các kỹ thuật tối ưu hoá truy vấn hướng đối tượng được trình bày trong [4] - tối ưu hoá truy vấn bằng các phép biến đổi đại số đối tượng, nhằm làm cơ sở cho vấn đề ước lượng các biểu thức đại số đối tượng thông qua việc xác định thứ tự ưu tiên thực hiện các phép toán đại số đối tượng, việc chứng minh sự tương đương trong chuyển đổi giữa các biểu thức đại số đối tượng và truy vấn trong ngôn ngữ truy vấn hướng đối tượng là cần thiết, từ đó, chúng tôi chọn ngôn ngữ truy vấn OQL, là ngôn ngữ truy vấn CSDL hướng đối tượng khá "thân thiện" và đại số đối tượng của nó (đã được trình bày trong [1, 2, 3, 5]) là ngôn ngữ truy vấn sử dụng trong quá trình nghiên cứu về xử lý truy vấn và tối ưu hoá truy vấn CSDL

hướng đối tượng.

1.2. Truy vấn CSDL hướng đối tượng OQL

Cú pháp của truy vấn CSDL hướng đối tượng OQL ([1, 2, 3]):

Truy vấn:

select [distinct] q
from (q as x,..., q as x)
where q
$q ::= b|f|i|c|s$
$|x$
$|bag(q, ..., q)|set(q, ..., q)|list(q, ..., q)|array(q, ..., q)$
$|struct(\ell : q, ..., \ell : q)$
$|C(\ell : q, ..., \ell : q)|q.\ell|(C)q$

Định nghĩa biến vùng $d ::= define \times as\ q$
$|define\ (x, ..., x)\ as\ q$

Ký hiệu b, f, i, c, s tương ứng là các kiểu dữ liệu *boolean, float, integer, character và string*, x là tập đếm được các định danh, ℓ là tập đếm được các nhãn và C là tập đếm được các tên lớp.

2. CÁC THÀNH PHẦN CỦA TRUY VẤN OQL

2.1. Biểu thức đường dẫn

Kiểu miền của một thuộc tính thuộc lớp có thể là một lớp khác. Điều này xác nhận có "đường đi" giữa các đối tượng của lớp này đến các đối tượng của một lớp khác thông qua các liên kết hợp thành. Trong ngôn ngữ truy vấn CSDL hướng đối tượng, ký hiệu dấu chấm ' .' được sử dụng để xác định các đường đi như vậy.

2.2. Bộ phận của một phân cấp lớp

Cho C là một siêu lớp cha của $C_1, C_2, ..., C_n$. Từ phạm vi của C là một siêu lớp của phạm vi với mỗi lớp $C_i,\ i = 1, ..., n$ nên việc tìm kiếm trên C sẽ được thực hiện với tất cả các đối tượng trong phân cấp lớp có gốc tại C. Có nhiều phương pháp truy vấn khác nhau cho phép chúng ta chỉ tìm kiếm các đối tượng ở một vài lớp chứ không phải tất cả các lớp. Các truy vấn như vậy được hỗ trợ với các phép toán tập hợp: *union, difference và except.*

2.3. Các phương thức tham chiếu

Có 2 kiểu phương thức được sử dụng trong một truy vấn CSDL hướng đối tượng. Kiểu thứ nhất là phương thức thuộc tính - suy dẫn được dùng cho phương thức có thể áp dụng để tính một giá trị đối với mỗi đối tượng trong lớp. Trong một truy vấn, phương thức thuộc tính - suy dẫn có thể được xét như một thuộc tính.

Kiểu phương thức thứ hai là hàm vị từ trả về một giá trị Boolean đối với mỗi đối tượng trong lớp. Chẳng hạn, giả sử ta có phương thức "large():Boolean" trên lớp Department, phương thức này sẽ trả về giá trị *true* nếu trường có trên 30 khoa thành viên hoặc trên 300 sinh viên và *false* nếu ngược lại.

Ví dụ 1. select $f.name$ from Department as $d, d.Chairperson$ as f
where $d.large()$

2.4. Cấu trúc trích xuất

Mô hình dữ liệu hướng đối tượng chỉ cho phép sử dụng các kiến trúc đối tượng như tập, danh sách và bộ để thiết lập các đối tượng phức từ các đối tượng đơn giản. Ngôn ngữ truy vấn OQL cho phép thiết lập các giá trị phức trong kết quả của một truy vấn bằng việc áp dụng cùng tập các phép toán thiết lập ở mệnh đề select.

Ví dụ 2. select $tuple(SSN : s.SSN, Name : s.Name,$
$CS\_Courses : selectcfroms.Courseasc$
where $c.Department.Name = `ComputerScience')$
$fromUG\_Studentass$
$wheres.GPA > 3$

3. CÁC PHÉP TOÁN ĐỐI TƯỢNG

Các phép toán đại số được chia làm 6 loại: phép toán đối tượng, phép toán bộ, phép toán tập hợp, phép toán *bag*, phép toán danh sách và phép toán mảng ([5]).

3.1. Phép toán đối tượng

Mỗi đối tượng được biểu diễn như một bộ ba (oid, class_name, value), oid là định danh duy nhất (OID) của đối tượng và class_name là tên lớp chứa đối tượng (). Giá trị của đối tượng do người sử dụng định nghĩa thường là một bộ. Có 3 phép toán trên các đối tượng:

- Chiếu lấy OID: $\pi_0()$, phép toán này nhận một đối tượng và trả về OID của đối tượng.

- Chiếu lấy giá trị: $\pi_V()$, phép toán này nhận một đối tượng và trả về giá trị của đối tượng.

- Chiếu lấy đối tượng: $\pi_D()$, phép toán này nhận một OID và trả về đối tượng có OID.

3.2. Phép toán bộ

- Thiết lập bộ: $tuple(a_1 : v_1, ..., a_n : v_n)$. Phép toán này nhận một số các thuộc tính và các cặp giá trị $(a_i : v_i)$ và trả về một bộ.

- Chiếu bộ: $\pi_{(Attrs)}()$, phép toán này nhận một bộ và trả về một bộ con với tên các thuộc tính được mô tả trong Attrs.

- Trích xuất thuộc tính: $\pi_{Attr}()$, nhận vào một bộ và trả về giá trị của thuộc tính mô tả.

- Nối bộ: $tuple\_cat()$, nhận vào hai bộ và nối chúng vào một bộ mới. Phép toán này được dùng để thay thế tích Đề các.

3.3. Phép toán tập hợp

- Thiết lập tập hợp: $set()$ hoặc $\{\}$. Phép toán thiết lập một tập khởi đầu của một số phần tử.

- Phép toán hợp, hiệu: set_union, set_diff.

- Phép toán chọn tập hợp: $\sigma^s_{\lambda s.f}()$, phép toán nhận một tập (thường là một tập đối tượng) và trả về một tập (đối tượng) sao cho mỗi phần tử ở tập kết quả thoả mãn điều kiện mô tả

trong công thức f.

- Phép toán làm phẳng tập ($setflatten$) : $set\_flat()$. Phép toán này nhận vào một tập các tập và trả về một tập chứa phần hợp của các tập lồng nhau.

- Phép toán áp dụng tập ($setapplication$) : $set\_apply_{\lambda s.e}()$. Phép toán này nhận vào một tập và áp dụng biểu thức đại số e vào mỗi phần tử trong tập.

3.4. Phép toán bag. Có 7 phép toán trên kiểu dữ liệu bag, 6 trong 7 phép toán này là tương tự như các phép toán tập hợp ngoại trừ các phép toán bag cho phép trùng lặp:

- Thiết lập, hợp, hiệu, chọn, làm phẳng, áp dụng: $bag()$, $bag\_union()$, $bag\_diff()$, $\sigma^b_{\lambda s.f}()$, $bag\_flat()$, $bag\_apply()$.

- Chuyển đổi về tập: $bagtoset()$, phép toán này chuyển đổi một bag về tập hợp bằng cách loại bỏ các trùng lặp trong bag.

3.5. Phép toán danh sách. Phép toán thiết lập danh sách, lấy phần tử đầu tiên, lấy phần tử cuối cùng, nối danh sách, chọn, làm phẳng, áp dụng, lần lượt là: $list()$, $first()$, $last()$, $list\_cat()$, $\sigma^l_{\lambda s.f}()$, $list\_flat()$, $list\_apply_{\lambda s.e}()$.

3.6. Phép toán mảng

- Thiết lập mảng, ghép mảng, áp dụng: $array()$, $array\_cat()$, $array\_apply_{\lambda s.e}()$.

- Trích xuất phần tử: $\pi_i()$. Phép toán này trả về phần tử thứ i trong mảng đã cho.

- Chiếu mảng: $\pi_{i,j}()$, $j > i$. Phép toán trả về một mảng con chứa các phần tử có chỉ số từ phần tử thứ i đến phần tử thứ j của mảng đã cho.

4. SỰ TƯƠNG ĐƯƠNG GIỮA ĐẠI SỐ ĐỐI TƯỢNG VÀ NGÔN NGỮ TRUY VẤN OQL

Định nghĩa. Nếu E là biểu thức đại số đối tượng và Q là truy vấn trong ngôn ngữ truy vấn OQL cùng xác định một tập đối tượng thì ta nói E biểu diễn Q hay Q biểu diễn E. Với kỹ thuật được trình bày trong [4], chúng ta sử dụng chứng minh sự chuyển đổi giữa đại số đối tượng và truy vấn trong ngôn ngữ truy vấn CSDL hướng đối tượng OQL là tương đương.

4.1. Các biểu thức đại số đối tượng được biểu diễn tương đương bằng truy vấn trong ngôn ngữ truy vấn CSDL hướng đối tượng OQL

Định lý 1. *Mọi biểu thức đại số đối tượng luôn tìm được sự biểu diễn tương đương bằng truy vấn trong ngôn ngữ truy vấn OQL.*

Chứng minh: Một biểu thức đại số E chứa một hoặc nhiều định danh (biến, hằng, hàm,...) các đối tượng CSDL ở mức đỉnh và các phép toán đại số (có thể không có phép toán nào). Trong chứng minh sau đây, chúng ta sử dụng phép chứng minh quy nạp, tức là, nếu chúng ta chứng minh tập các phép toán đại số đối tượng có thể được diễn dịch dưới dạng một truy vấn OQL tương đương thì một biểu thức đại số đối tượng bất kỳ có thể được viết thành một truy vấn CSDL hướng đối tượng bằng ngôn ngữ truy vấn OQL.

Trường hợp cơ sở: Giả sử biểu thức đại số đối tượng E không chứa phép toán nào.

Trong trường hợp này, $E = R$, là định danh hoặc là một đối tượng CSDL ở mức đỉnh. Truy vấn trong OQL sẽ là: $selectR$.

Trường hợp quy nạp: Giả sử trong biểu thức đại số E có chứa một hoặc nhiều phép toán.

Giả sử rằng tất cả các biểu thức có ít hơn n phép toán ($n \geq 1$), chúng ta có các truy vấn OQL tương ứng với 32 phép toán trong đại số đối tượng như sau.

(i) Các phép toán đối tượng

1. Chiếu lấy OID: $\pi_0(S)$.

select $0.OIDfrom\ S\ as\ 0$

2. Chiếu lấy giá trị: $\pi_V(S)$.

select $(0.a_1, 0.a_2, ..., 0.a_n)from\ S\ as\ 0$

trong đó, a_i là các thuộc tính của đối tượng $0(i = 1, ..., n)$.

3. Chiếu lấy đối tượng: $\pi_D(oid\_object)$.

select 0

where $oid\_objectin(select0.OIDfromSaso)$

(ii) Các phép toán bộ

1. Thiết lập bộ: $tuple(a_1 : v_1, ..., a_n : v_n)$.

select $tuple(a_1 : v_1, ..., a_n : v_n)$

2. Chiếu bộ: $\pi_{(Attrs)}(S),\ (Attrs) = \{a_1, a_2, ..., a_m\}$.

select $tuple(a_1 : v.v_1, ..., a_m : v.v_m)fromSasv$

3. Trích xuất thuộc tính: $\pi_{Attr}(S)$.

select $v.AttrsfromSasv$

4. Nối bộ: $tuple\_cat(S_1, S_2)$.

select $tuple(a_1 : u.u_1, ..., a_n : u.u_n,\ b_1 : v.v_1, ..., b_m : v.v_m)$

from $S_1asu,\ S_2asv$

(iii) Các phép toán tập hợp

1. Thiết lập tập hợp: $set(S)$ hoặc $\{\}$.

select $set(s)fromSass$

2. Phép toán hợp: $set\_union(S_1, S_2)$

select $set(s)from(S_1unionS_2)ass$

3. Phép toán hiệu: $set\_diff(S_1, S_2)$.

select $set(s)from(S_1exceptS_2)ass$

4. Phép toán chọn tập hợp: $\sigma^s_{\lambda s.f}(S)$.

select $set(u)fromSasu$

where $finu$

5. Phép toán làm phẳng tập: $set\_flat(S)$.

select $set(v)$

where $vin(selectdistinctufromSasu)$

6. Phép toán áp dụng trên tập: $set\_apply_{\lambda s.e}(S)$.

define $s\_elementas < type(element(S)) >$

define function $e() : s\_element$

select $set(u) fromSasu$

where $u.e()$

(iv) Các phép toán bag

Các phép toán trên kiểu dữ liệu bag được chứng minh tương tự như các phép toán trong (c). Chúng ta chỉ xét phép toán $bagtoset(S)$ với truy vấn OQL sẽ là:

select $set(u)$

from (select distinct u from S as u)

(v) Các phép toán danh sách

1. Thiết lập danh sách: $list(S)$.

select $list(s)$ from S as s

2. Lấy phần tử đầu tiên: $first(S)$.

select $s[0]$ from S as s

3. Lấy phần tử cuối cùng: $last(S)$.

select $s[count(selectsfromSass) - 1]$

4. Các phép toán nối danh sách: $list\_cat(S_1, S_2)$, chọn $\sigma^l_{\lambda s.f}(S)$, làm phẳng: $list\_flat(S)$, áp dụng: $list\_apply_{\lambda s.e}(S)$ được chứng minh tương tự như các phép toán trong tập hợp và bag.

(vi) Các phép toán mảng

1. Thiết lập mảng: $array(S)$.

select $array(s) fromSs$

2. Trích xuất phần tử: $\pi_i(S)$. Phép toán này trả về phần tử thứ i trong mảng đã cho.

select $s[i] fromSs$

3. Chiếu mảng: $\pi_{i,j}(S),\ j > i$.

select $array(s[x]) fromSs$

where $(x >= i) and (x <= j)$

4. Các phép toán nối mảng: $array\_cat(S_1, S_2)$, áp dụng: $array\_apply_{\lambda s.e}(S)$ được chứng minh tương tự như trong tập các phép toán tập hợp và danh sách.

Đến đây, chúng ta hoàn tất chứng minh cho trường hợp quy nạp, hoàn thành điều phải chứng minh: Mọi biểu thức đại số đối tượng đều có thể biểu diễn bằng một truy vấn trong truy vấn CSDL hướng đối tượng OQL. ■

4.2. Truy vấn trong ngôn ngữ truy vấn CSDL hướng đối tượng OQL được biểu diễn bằng biểu thức đại số tương đương

Định lý 2. *Mọi truy vấn trong ngôn ngữ truy vấn CSDL hướng đối tượng OQL đều được biểu diễn bằng một biểu thức đại số đối tượng.*

Để chứng minh định lý 2, chúng ta sẽ chứng minh thông qua các bổ đề ở phần sau. Bước thứ nhất, chúng ta xét truy vấn "select..." trong OQL với 3 mệnh đề chính "select_clause",

"from_clause", "where_clause" và chứng minh các thành phần trong 3 mệnh đề này đều được biểu diễn một cách tương đương bởi các biểu thức đại số. Bước thứ hai, dựa vào các kết quả đã chứng minh ở bước 1 để chứng minh cho truy vấn tổng quát có thể biểu diễn bằng một biểu thức đại số tương đương.

Để thuận tiện trong trình bày, chúng ta biểu diễn cú pháp của truy vấn CSDL hướng đối tượng OQL như sau:

Truy vấn:

$$select[distinct] < final\_res >$$
$$from < from\_clause >$$
$$where < where\_clause >$$

trong đó,

$$< final\_res >::=< res\_list >$$
$$< res\_list >::= (< result >,) < result >$$
$$< result >::= (< r\_component > .) < r\_component >$$
$$< r\_component >::= (< object > | < var >)([< int\_const > [:< int\_const >]])$$
$$< from\_clause >::=< target > [< set\_op >< target >]as < var >$$
$$< target >::= (< t\_component > .) < t\_component >$$
$$< t\_component >::=< object > ([< int\_const > [:< int\_const >]])$$
$$< where\_clause >::=< where\_clause > (and|or) < where\_clause >$$

4.2.1. Mệnh đề "from_clause" được biểu diễn tương đương bằng một biểu thức đại số đối tượng

Trước hết chúng ta xét "from_clause" và chứng minh lần lượt các bổ đề đối với từng thành phần trong "from_clause" ($\langle t\_component\rangle, \langle target\rangle, \langle var\rangle$) và sau đó là "from_clause".

Bổ đề 1. *t_component có một biểu thức đại số tương đương.*

Chứng minh. Bằng phương pháp quy nạp trên các thành phần của các phép toán liên quan đến chỉ số mảng xuất hiện trong thành phần.

Trường hợp cơ sở: t_component = C (không có chỉ số mảng). Trong trường hợp này thành phần là một định danh và phải là tên của một đối tượng ở mức đỉnh. Vì vậy, truy vấn đại số đối với việc tìm đối tượng này đơn giản là C.

Trường hợp quy nạp:

(i) $t\_component = C[...]...[...][x : y]$. Theo giả thiết quy nạp, với mọi *t_component* cho trước chỉ số cuối được biểu diễn theo đại số bởi một biểu thức CE nào đó. Nếu CE là một tham chiếu, thì nó được thay thế bởi $\pi_D\pi_0(CE)$. Truy vấn đối với $t\_component : \pi_{x,y}(CE)$.

(ii) $t\_component = C[...]...[...][x]$. Theo giả thiết quy nạp, với mọi *t_component* cho trước chỉ số cuối được biểu diễn theo đại số bởi một biểu thức CE nào đó. Nếu CE là một tham chiếu, thì nó được thay thế bởi $\pi_D\pi_0(CE)$. Truy vấn đối với $t\_component : \pi_x(CE)$. ■

Bổ đề 2. *target trong from_clause với thành phần cuối không có chỉ số sẽ được biểu diễn bằng một biểu thức đại số.*

Chứng minh. Quy nạp trên số thành phần trong *target*.

Trường hợp cơ sở: $target = C(C$ là một thành phần). Theo bổ đề 1 điều này đã được chứng minh.

Quy nạp: $target = C_1...C_n.C = (Attrs)$. Với giả thiết quy nạp và bổ đề 1, lấy E_1 là một biểu thức đại số cho $C_1...C_n$. Chúng ta xét 6 trường hợp tương ứng với các kiểu của E_1 :

(i) E_1 là một bộ. Truy vấn đại số là: $\pi_{(Attrs)}(S)$.

(ii) E_1 là một tham chiếu đến bộ. Truy vấn là: $\pi_{(Attrs)}(\pi_D(\pi_0(E_1)))$.

(iii) E_1 là một tập các bộ. Nếu C là một thuộc tính giá trị - tập, ta có truy vấn đại số: $set\_flat(set\_apply_{\lambda s\pi_{(Attrs)}(\pi_V(s))}(E_1))$. Ngoài ra, ta có truy vấn đại số như sau: $set\_apply_{\lambda s\pi_{(Attrs)}(\pi_V(s))}(E_1)$.

(iv) E_1 là một tập tham chiếu đến bộ. Nếu C là thuộc tính giá trị - tập, thì ta có truy vấn đại số: $set\_flat(set\_apply_{\lambda s\pi_{(Attrs)}(\pi_V(\pi_D(\pi_0)))}(E_1))$. Ngoài ra, ta có truy vấn đại số như sau: $set\_apply_{\lambda s\pi_{(Attrs)}(\pi_V(\pi_D(\pi_0)))}(E_1))$.

(v) E_1 là một mảng các bộ. Nếu C là một thuộc tính giá trị - tập, ta có truy vấn đại số: $array\_cat(array\_apply_{\lambda s\pi_{(Attrs)}(\pi_V(S))}(E_1))$. Ngoài ra, ta có truy vấn đại số như sau: $array\_apply_{\lambda s\pi_{(Attrs)}(\pi_V(S))}(E_1))$.

(vi) E_1 là một mảng các tham chiếu đến bộ. Nếu C là một thuộc tính giá trị tập, thì ta có truy vấn đại số: $array\_cat(array\_apply_{\lambda s\pi_{(Attrs)}(\pi_V(\pi_D(\pi_0(S))))}(E_1))$.

Ngoài ra, ta có truy vấn đại số như sau: $array\_apply_{\lambda s\pi_{(Attrs)}(\pi_V(\pi_D(\pi_0(S))))}(E_1)$. ■

Bổ đề 3. *Thành phần target của from_clause trong truy vấn có một biểu thức đại số tương đương.*

Chứng minh. Chúng ta chỉ xét trường hợp mà bổ đề 2 chưa đề cập, là các thành phần cuối của target có các chỉ số mảng.

(i) Giả sử rằng *target* có dạng $target = C_1...C_n.C[x : y]$ thì theo bổ đề 2, chúng ta biết rằng có một biểu thức E_1 tương đương với $C_1...C_n.C$. Nếu E_1 là một tham chiếu, nó sẽ được thay thế bằng $\pi_D(\pi_0(E_1))$. Vì vậy truy vấn đại số là: $\pi_{x,y}(E_1)$.

(ii) Tương tự, ta giả sử $target = C_1...C_n.C[x]$. Theo bổ đề 2, chúng ta biết rằng có một biểu thức E_1 tương đương với $C_1...C_n.C$. Nếu E_1 là một tham chiếu, nó sẽ được thay thế bằng $\pi_D(\pi_0(E_1))$. Ta có truy vấn $\pi_x(E_1)$. ■

Bổ đề 4. *Biến* ($\langle var \rangle$) *được khai báo ở from_clause có một biểu thức đại số tương đương.*

Chứng minh. Gọi T, T_1 và T_2 là các thành phần (*target*) trong *from_clause*, với các truy vấn đại số tương đương là E, E_1 và E_2. Chúng ta xét 3 trường hợp:

(i) $var = T$. Theo bổ đề 3 chúng ta có một biểu thức đại số tương đương.

Trong các trường hợp sau, nếu E, E_1 hoặc E_2 là một tham chiếu, nó sẽ được thay thế tương đương bằng $\pi_D(\pi_0(E)), \pi_D(\pi_0(E_1))$ hoặc $\pi_D(\pi_0(E_2))$.

(ii) $var = T_1$ *union* T_2. Nếu T_1 hoặc T_2 cũng biểu diễn 1 tập thì ta có truy vấn là: E_1 *set_union* E_2. Ngoài ra, truy vấn tương đương là: $array\_cat(E_1, E_2)$.

(iii) $var = T_1$ *except* T_2. Nếu T_1 hoặc T_2 cũng biểu diễn 1 tập thì truy vấn là: $E_1 set\_diff\ E_2$.

■

4.2.2. Mệnh đề "select_clause" được biểu diễn tương đương bằng một biểu thức đại số đối tượng

Tiếp theo, chúng ta xét "select_clause" và chứng minh lần lượt các bổ đề cho các thành phần ở "select_clause" ($\langle r\_component\rangle, \langle result\rangle, \langle res\_list\rangle, \langle final\_res\rangle$) và sau đó là mệnh đề "select".

Bổ đề 5. *r_component có một biểu thức đại số tương đương.*

Chứng minh. Với phương pháp quy nạp trên các phép toán liên quan đến chỉ số mảng xuất hiện trong *r_component*.

Trường hợp cơ sở: r_component = C (không có chỉ số mảng). Trong trường hợp này *r_component* cũng là một định danh của một đối tượng ở mức đỉnh. Vì vậy, truy vấn đại số đối với việc tìm đối tượng này đơn giản là C, hoặc tên của biến ở mệnh đề "from_clause", theo bổ đề 4 ta có một biểu thức truy vấn đại số tương đương trong trường hợp này.

Trường hợp quy nạp:

(i) *r_component* = $C[...]\cdots[...][x:y]$. Theo giả thiết quy nạp, với mọi *r_component* cho trước chỉ số cuối được biểu diễn theo đại số bởi một biểu thức CE nào đó. Nếu CE là một tham chiếu, thì nó được thay thế bởi $\pi_D(\pi_0(CE))$. Truy vấn đối với *r_component* : $\pi_{x,y}(CE)$.

(ii) *r_component* = $C[...]\cdots[...][x]$. Theo giả thiết quy nạp, với mọi *r_component* cho trước chỉ số cuối được biểu diễn theo đại số bởi một biểu thức CE nào đó. Nếu CE là một tham chiếu, thì nó được thay thế bởi $\pi_D(\pi_0(CE))$. Truy vấn đối với *r_component* : $\pi_x(CE)$. ■

Bổ đề 6. *result có thành phần cuối cùng không có chỉ số có một biểu thức đại số tương đương.*

Chứng minh. Quy nạp theo số thành phần của *result*.

Trương hợp cơ sở: result = C. Theo bổ đề 5 điều này đã được chứng minh.

Quy nạp: result = $C_1...C_n.C = (Attrs)$. Với giả thiết quy nạp và bổ đề 5, lấy E_1 là một biểu thức đại số tương đương với $C_1...C_n$. Chúng ta xét các trường hợp sau, tuỳ thuộc vào kiểu của E_1 :

(i) E_1 là một bộ. Truy vấn đại số là: $\pi(Attrs)(S)$.

(ii) E_1 là một tham chiếu đến bộ. Truy vấn đại số là: $\pi(Attrs)(\pi_D(\pi_0(E_1)))$.

(iii) E_1 là một tập các bộ. Truy vấn đại số là $set\_apply_{\lambda s\pi_{(Attrs)}(\pi_V(s))}(E_1)$.

(iv) E_1 là một tập tham chiếu đến bộ. Truy vấn đại số: $set\_apply_{\lambda s\pi_{(Attrs)}(\pi_V(\pi_D(\pi_0(s))))}(E_1)$.

(v) E_1 là một mảng các bộ. Truy vấn đại số: $array\_apply_{\lambda s\pi_{(Attrs)}(\pi_V(s))}(E_1)$.

(vi) E_1 là mảng các tham chiếu đến bộ. Truy vấn đại số: $array\_apply_{\lambda s\pi_{(Attrs)}(\pi_V(\pi_D(\pi_0(s))))}(E_1)$.

■

Bổ đề 7. *result được biểu diễn bằng một biểu thức đại số tương đương.*

Chứng minh. Chúng ta chỉ xét trường hợp bổ sung cho bổ đề 6 là các thành phần cuối của result có chỉ số mảng. Vì vậy chứng minh ở đây tương tự như trường hợp chứng minh quy nạp của bổ đề 1.

(i) Giả sử *result* = $C_1...C_n.C[x:y]$. Theo bổ đề 6, chúng ta biết rằng có một biểu thức E_1

tương đương với $C_1...C_n.C$. Nếu E_1 là một tham chiếu, nó sẽ được thay thế bằng $\pi_D(\pi_0(E_1))$. Vì vậy truy vấn trở thành $\pi_{x,y}(E_1)$.

(ii) Giả sử $result = C_1...C_n.C[x]$. Thì theo bổ đề 2, chúng ta biết rằng có một biểu thức E_1 tương đương với $C_1...C_n.C$. Nếu E_1 là một tham chiếu, nó sẽ được thay thế bằng $\pi_D(\pi_0(E_1))$. Vì vậy truy vấn trở thành $\pi_X(E_1)$. ■

Bổ đề 8. *res_list được biểu diễn bằng một biểu thức đại số tương đương.*

Chứng minh. Lấy $res\_list = R_1, R_2, ..., R_n$ và E_i là biểu thức đại số tương đương với R_i (bổ đề 7). Chúng ta xét 2 trường hợp sau:

(i) Nếu R_i không phải kiểu tập hoặc mảng, ta có biểu thức đại số tương đương:

$tuple\_cat(tuple(E_1),\ tuple\_cat(tuple(E_2), ..., tuple\_cat(tuple(E_{n-1}),\ tuple(E_n))))$.

(ii) Nếu có ít nhất một trong các R_i là một tập, ta có: $E_1 \times (E_2 \times (E_3 \cdots (E_{n-1} \times E_n)))$. Nếu E_i là một tham chiếu đến tập, thì nó được thay bằng $\pi_D(\pi_0(E_1))$. ■

Bổ đề 9. *final_res có một biểu thức đại số tương đương.*

Chứng minh. Giả sử (sử dụng bổ đề 8) biểu thức đại số tương đương với $final\_resR$ là E. Chúng ta xét 4 trường hợp xảy ra như sau:

(i) $final\_res = set(R)$. Truy vấn là $set(E)$ (hoặc $\{E\}$).

(ii) $final\_res = array(R)$. Truy vấn là $array(E)$ (hoặc $[E]$).

(iii) $final\_res = (R)$. Truy vấn là $tuple(E)$.

(iv) $final\_res = struct(R)$. Truy vấn là $\pi_0(E_1)$. ■

4.2.3. Mệnh đề "where_clause" được biểu diễn tương đương bằng một biểu thức đại số đối tượng

Bổ đề 10. *where_clause được biên dịch thành một tân từ trong ngôn ngữ vị từ.*

Chứng minh. Chúng ta chứng minh quy nạp trên số các vị từ xuất hiện trong mệnh đề "where_clause".

Trường hợp cơ sở: Một mệnh đề so sánh đơn giản có dạng $T_1\langle op\rangle T_2$. Theo bổ đề 4 và 7, T_1 và T_2 sẽ có các biểu thức đại số tương đương là E_1 và E_2. Ta chỉ xét 4 trường hợp sau:

(i) T_1 *is* T_2. Ta có tân từ: $E_1 = E_2$.

(ii) $T_1 = T_2$. Ta có tân từ $\pi_D(\pi_0(E_1)) = \pi_D(\pi_0(E_2))$ nếu T_1 và T_2 là các tham chiếu và $E_1 = E_2$ đối với những trường hợp còn lại.

(iii) $T_1 in T_2$ thì tân từ sẽ là $E_1 \in E_2$ nếu T_2 là một tập và E_1 E_2 đối với các trường hợp còn lại. Nếu E_2 là một tham chiếu đến một tập hoặc mảng, nó được thay thế bởi $\pi_D(\pi_0(E_2))$. Nếu E_1 là một tham chiếu, nó được thay thế bằng $\pi_D(\pi_0(E_1))$ nếu kiểu của các phần tử của E_2 không phải là kiểu tham chiếu.

(iv) $T_1 \leq T_2$. Tân từ là $E_1 \subseteq E_2$ nếu T_2 là một tập và E_1 E_2 trong các trường hợp còn lại. Nếu E_1 hoặc E_2 là một tham chiếu, nó sẽ được thay thế bằng $\pi_D(\pi_0(E_1))$ hoặc $\pi_D(\pi_0(E_2))$.

Trường hợp quy nạp: Ta xét 3 trường hợp

(i) $forall/exists\ in\ S_1 : S_2$. Sử dụng bổ đề 7 và giả thiết quy nạp ta có các biểu thức E_1 và E_2 tương đương với S_1 và S_2. Tân từ tương đương sẽ là $apply_{\lambda s.E_2}(E_1)$. Nếu E_1 là một tham chiếu, nó sẽ được thay thế bằng $\pi_D(\pi_0(E_1))$.

(ii) S_1 *and* S_2. Theo giả thiết quy nạp ta có các biểu thức E_1 và E_2 tương đương với S_1 và S_2. Vì vậy, ta có tân từ: $E_1 \wedge E_2$.

(iii) *not*S. Theo giả thiết quy nạp ta có các biểu thức E tương đương với S. Ta có tân từ $\neg E$.

4.2.4. Truy vấn tổng quát "select..." trong OQL được biểu diễn bằng biểu thức đại số đối tượng tương đương

Bổ đề 11. *Truy vấn tổng quát select của OQL không chứa từ khoá "distinct" hoặc các kết nhập (aggregate) có một biểu thức đại số tương đương.*

Chứng minh. Với các bổ đề đã chứng minh ở trên, chúng ta biết rằng mỗi biến V_i (var) trong "from_clause" có một biểu thức đại số tương đương, ta ký hiệu là QV_i. Và với mỗi *result* R_i trong *res_list* có một biểu thức đại số tương đương, ký hiệu là QR_i.

Để đảm bảo các *result* trong *res_list* là tương quan với nhau, chúng ta sẽ có một khái niệm mở rộng của "biến vùng" . Chúng ta xem mỗi thành phần của *result* trong *res_list* như là một biến vùng và sẽ chỉ rõ một trong các thành phần đó được xem như là một "thành phần chính". Ví dụ: ta có truy vấn OQL:

select ($s.Name.firstname$, $s.Name.lastname$) from $G\_Students$

where $s.Age > s.Advisor.Age$

Ở danh sách kết quả (*res_list*) : "($s.Name.firstname$, $s.Name.lastname$)", rõ ràng kết quả ta có được là một tập các cặp gồm giá trị của 2 thuộc tính ($firstname$, $lastname$). Vì vậy thành phần chính ở đây (biến vùng) là "$s.Name$" ($G\_Student.name$). Nếu chúng ta chọn "$G\_Student$" làm thành phần chính thì kết quả trả về sẽ là một tập các tên và một tập các họ của các sinh viên, nhưng tên và họ trong 2 tập này lại không có quan hệ ràng buộc gì với nhau.

Với hai kết quả R_1 và R_2, chúng ta sẽ có thành phần chung nhất chính là "đường dẫn" chung nhất đi trước các kết quả. Ta sẽ xác định thành phần chính của mỗi *result* trong *res_list* bằng đoạn chương trình giả mã như sau:

For each $resultR_1$ **in** *res_list* **do**

For each $resultR_2$ **in** *res_list* **do**

$G := GCC(R_1, R_2)$ //GCC: thành phần chung nhất.

If $(G > primary\_component(R_1))$

Then $primary\_component(R_1) := G$

Đối với *res_list* có n result thì có thể có ít hơn n thành phần chính duy nhất. Chúng ta xem xét tính tương quan giữa dữ liệu được tìm kiếm với các *result* có cùng thành phần chính. Với mỗi thành phần chính duy nhất PCi trong *res_list*, ta thực hiện như sau:

(i) Giả sử PC_i là thành phần chính của các kết quả $Rr_1, ..., Rr_n$.

(ii) Vì mỗi Rr_i có cùng thành phần chính, do đó chúng sẽ có một biểu thức con đại số chung tương đương với thành phần chính này. Nếu xoá biểu thức con chung này trong mỗi QRl_i, chúng ta sẽ nhận được các biểu thức đại số riêng biệt $PQRl_i$, chúng hoàn toàn khác biệt nhau và được xem như là đầu vào của biểu thức con đại số chung.

(iii) Thiết lập biểu thức đại số riêng $APCE_i$ như sau: Nếu $n = 1$, biểu thức riêng là $PQRl_1$. Nếu $n > 1$, biểu thức riêng là $tuple\_cat(tuple(PQRl_1), ..., tuple\_cat(tuple(PQRl_{n-1}), tuple(PQRl_n)))$.

Đến đây, chúng ta có một biểu thức để áp dụng cho mỗi phần tử của vùng thuộc thành

phần chính tương ứng để tìm dữ liệu thích hợp từ thành phần đó. Trước khi thực hiện, chúng ta sẽ áp dụng đối với tân từ của "where_clause" . Theo bổ đề 10, ta thiết lập một biểu thức đại số cho một tân từ. Áp dụng tân từ này đối với biểu thức đại số PE cho các thành phần chính của *res_list* như sau (theo bổ đề 7: mỗi thành phần chính PC có một biểu thức đại số tương đương PCE):

(i) Nếu *res_list* chỉ có một thành phần chính thì $PE = PCE$.

(ii) Còn không, nếu không tồn tại một PC_i nào có giá trị - tập hoặc giá trị - mảng thì:

$PE = tuple\_cat(tuple(PCE_1), ..., tuple\_cat(tuple(PCE_{m-1}), tuple(PCE_m)))$, ($m$ là số thành phần chính trong *res_list*).

(iii) Còn không, nếu tồn tại ít nhất một PC_i có giá trị - tập, thì:

$PE = PCE_1 \times (PCE_2 \times \cdots \times (PCE_{m-1} \times PCE_m))$. Nếu PCE_i là một tham chiếu, nó được thay thế bởi $\pi_D(\pi_0(PCE_i))$.

Tiếp theo, áp dụng tân từ P của "where_clause" đối với PE. Chúng ta xét các trường hợp thiết lập PQ - biểu thức tương đương với "where_clause" áp dụng cho PE :

(i) PE là một giá trị, một bộ hoặc một tham chiếu thì $PQ = apply_{\lambda s.P}(PE)$.

(ii) PE là một tập thì $PQ = set\_apply_{\lambda s.P}(PE)$.

(iii) PE là một mảng thì $PQ = array\_apply_{\lambda s.P}(PE)$.

Sử dụng các kết quả trên để áp dụng $APCE_i$ đối với PQ để đi đến kết quả cuối cùng của truy vấn. Xét các trường hợp:

(i) PQ là một giá trị, một bộ hoặc một tham chiếu. Nếu chỉ có một thành phần chính và tất cả các kết quả tương ứng của nó là đơn trịthì truy vấn cuối cùng là $APCE_1(PQ)$.

(ii) PQ là một tập thì mỗi phần tử của tập có một phần tương ứng với dữ liệu từ mỗi thành phần chính. Truy vấn cuối cùng sẽ là: $set\_apply_E(PQ)$, mà E là một biểu thức áp dụng cho mỗi $APCE_i(PQ)$ đối với thành phần tương ứng của nó sử dụng phép trích xuất bộ π (nếu cần thiết) và phép nối các kết quả vào một bộ sử dụng các phép toán *tuple*, *tuple_cat*.

(iii) PQ là một mảng thì mỗi phần tử của tập có một phần tương ứng với dữ liệu từ mỗi thành phần chính. Truy vấn cuối cùng sẽ là: $array\_apply_E(PQ)$, mà E là một biểu thức áp dụng cho mỗi $APCE_i(PQ)$ đối với thành phần tương ứng của nó sử dụng phép trích xuất bộ π (nếu cần thiết) và phép nối các kết quả vào một bộ sử dụng các phép toán *tuple*, *tuple_cat*.

■

Bổ đề 12. *Truy vấn select của OQL không chứa từ khoá "distinct" có một biểu thức đại số tương đương.*

Chứng minh. Nếu truy vấn không chứa kết nhập, bổ đề 11 đã chứng minh. Lưu ý rằng mỗi loại phép toán tập G (như "min", "max",...) định nghĩa một phép toán đại số mới. Thành phần chính của một kết quả kết nhập được xác định là biến trong *from_clause* (nếu nó có một thành phần) hoặc thành phần chính của danh sách thuộc tính yêu cầu mô tả bên trong kết nhập. Nếu danh sách thuộc tính này có nhiều hơn một thành phần chính, thành phần chính của kết quả kết nhập trở thành tích Đề các của các thành phần chính của danh sách thuộc tính.

Theo các bổ đề ở trên, lấy PCE là biểu thức đại số tương đương với thành phần chính của kết quả kết nhập. Trong phần còn lại của việc chứng minh bổ đề này, chúng ta xét nếu E hoặc các phần tử của tập là các tham chiếu, thì chúng phải được loại bỏ tham chiếu với

phép toán π_D. Theo bổ đề 11, chúng ta có một biểu thức đại số tương đương E có thể được áp dụng đối với thành phần chính của kết quả kết nhập để thu được dữ liệu với việc áp dụng hàm kết nhập G. Vì vậy G cũng là một phép toán đại số, $G(E)$ là một biểu thức và có thể sử dụng như một trong $PQRl_i$ trong bổ đề 11, đã được chứng minh. Nếu thành phần chính của kết quả kết nhập là một mảng, thì ta dùng phép toán *array_apply* thay thế phép toán *set_apply*. ■

Bổ đề 13. *Truy vấn tổng quát "select..." củaOQL có một biểu thức đại số đối tượng tương đương.*

Chứng minh. Nếu truy vấn không có từ khoá "distinct" , bổ đề 12 đã chứng minh. Nếu truy vấn là "select distinct Q", theo bổ đề 11 ta có một biểu thức đại số E biểu diễn Q. Nếu kiểu kết quả của E là một bộ, một tham chiếu hoặc một giá trị đơn thì biểu thức đối với truy vấn đầu vào là E. Nếu kiểu của E là tập hợp thì biểu thức đại số là $set\_flat(E)$. Ngoài ra, biểu thức đại số là $array\_apply_{\lambda s.e}(E)$. ■

Với việc hoàn tất chứng minh bổ đề 13, chúng ta hoàn thành việc chứng minh cho định lý 2. Và kết luận rằng sự biểu diễn giữa ngôn ngữ đại số đối tượng và truy vấn CSDL hướng đối tượng OQL là tương đương.

5. KẾT LUẬN

Vấn đề nghiên cứu về ngôn ngữ truy vấn CSDL hướng đối tượng OQL và đại số đối tượng của nó là cơ sở của việc lựa chọn ngôn ngữ truy vấn thể hiện, để chúng ta có thể xem xét việc xử lý và tối ưu hoá các truy vấn hướng đối tượng trên CSDL hướng đối tượng. Ngôn ngữ truy vấn CSDL hướng đối tượng OQL cung cấp khá nhiều kiểu dữ liệu thích hợp trong mô hình CSDL hướng đối tượng, đảm bảo cho sự đặc tả các đối tượng phức một cách phong phú và linh hoạt. Với kết quả đạt được trong bài báo này, chúng tôi hy vọng đây sẽ là cơ sở tốt để có thể đề xuất một số phương pháp tối ưu hoá truy vấn hướng đối tượng mà chúng tôi đang nghiên cứu đối với ngôn ngữ truy vấn CSDL hướng đối tượng OQL và đại số đối tượng của nó.

TÀI LIỆU THAM KHẢO

[1] G. M. Bierman, A. Trigoni, *Towards A Formal Type System For ODMG OQL,* Technical Report 497, University of Cambridge, Computer Laboratory, October 2000.

[2] Trigoni, Agathoniki, *Semantic Optimization of OQL Queries,* Technical Report, Number 547, University of Cambridge, Computer Laboratory, UCAM-CL-TR-547, ISSN 1476-2986, October 2002.

[3] A. Trigoni, G. M. Bierman, *Inferring the Principal Type and the Schema Requirements of an OQL Query*, In 18th British National Conference on Databases (BNCOD), 2001, pages 185- 201.

[4] Vanderberg, Scott Lee, *Algebras for Object - Oriented Query Languages, A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy* (Computer Sciences), at the University of Winconsin-Madison, 1993.

[5] Yu, T. Clement, Meng, Weiyi, *Principles of Database Query Processing for Advanced Applications,* Morgan Kaufmann Publishers, Inc. San Francisco, California, 1998.

Nhận bài ngày 05 - 11 - 2003