MỘT VÀI THUẬT TOÁN THỰC HIỆN PHÉP CHIA TRONG MÔ HÌNH QUAN HỆ

NGUYỄN THANH THỦY
Đại học bách khoa Hà Nội
NGUYỄN XUÂN HUY
Viện tính toán và điều khiển

1. MỞ ĐẦU

Quan hệ R với tập các thuộc tính $\alpha(R) = \{a_1, ..., a_h\}$ và các miền giá trị D_i, $i = \overline{1, h}$ được xem như là tập các ánh xạ $\{r : \alpha(R) \to \bigcup_{i=1}^{h} D_i \mid \forall i = \overline{1,h}\}$ $r(a_i) \in D_i$ hay là tập các bộ $\{r = \langle a_i . \alpha_i \rangle i = \overline{1,h} \mid \alpha_i \in D_i \ \forall i = \overline{1,h}\}$.

Giả sử $M \subseteq \alpha(R)$ là tập con các thuộc tính

$$R[M] \xrightarrow{đ.n} \{r . M \mid r \in R\}.$$

Nói rằng S và R [M] là khả hợp nếu như $S \cup R[M]$ vẫn là miền giá trị có thể nhận được của các thuộc tính trong M trong các miền giá trị tương ứng.

Ta định nghĩa phép chia như sau

$$R : M : S = \{y \in R[\overline{M}] \mid \forall x \in S \ \langle y, x \rangle \in R\}.$$

Trong [1] đã đưa ra một thuật toán tìm R : M : S có lưu ý đến thủ thuật đánh nhãn. Để đánh giá thuật toán đó, tác giả đã đưa vào định nghĩa quan hệ tương đương ρ như sau

$$r, r' \in R, \ r \rho r' \xleftrightarrow{đ.n} r.\overline{M} = r'.\overline{M}$$

Thuật toán [1] là như sau:

Thuật toán 1

```
n := card (S);
{loop 1:} for each r ∈ R do
          begin k := 0;
             construct t := < r . M̄ >;
          {loop 2:} for each q ∈ R with q . M̄ = t do
             begin if (q . M in S) then k := k + 1;
                   label q in R
             end;
{test:} if k = n then add t to T
        end
```

Trong [1], tác giả đã chỉ ra độ phức tạp thời gian của thuật toán 1 là

$$cost = \sum_{i=1}^{m} f(i) - \sum_{k=1}^{v} f\left(m - \sum_{j=o}^{k-1} m_j\right) + mf(n)$$

$m = card(R)$, $n = card(S)$, $v = card(R/\rho)$,
m_j, $j = \overline{1,v}$ là số phần tử trong lớp tương đương thứ j, $m_o = 0$.

Dưới đây ta sẽ đưa ra một vài nhận xét về thuật toán trên và nêu lên một thuật toán cải tiến.

2. MỘT THUẬT TOÁN KHÁC TÌM R : M : S

Ta có các nhận xét sau:

1) Nếu $S = \phi$ (card (S) = 0) thì $R : M : S = R[\overline{M}]$ (1)

2) Nếu $S \neq \phi$ và $S \notin R[M]$ thì $R : M : S = \phi$ (2)

Vì trong trường hợp ngược lại, nếu $\exists y\ y \in R : M : S$
thì $\forall x \in S \langle y, x \rangle \in R$. Do vậy $S \subseteq R[M]$.

3) Giả sử $\langle t, x \rangle \in R$; $\langle t, y \rangle \in R$; $t \in R[\overline{M}]$, $x, y \in R[M]$ thì $x \neq y$ (3)

Từ đây suy ra rằng nếu như ta xét các bộ $\langle t, x \rangle \in R$ với $t \in R[\overline{M}]$ cho trước, đến một lúc nào đó đã chỉ ra được rằng $S \subseteq t_M = \{x \in R[M] \mid \langle t, x \rangle \in R$ thì sự xuất hiện về sau của các bộ $\langle t, x \rangle$ không ảnh hưởng gì và có thể không cần xét chúng.

Từ nhận xét trên ta suy ra rằng đoạn chương trình

```
k : = 0;
construct  t : = <r . M̄>;
for each q ∈ R with q . M̄ = t do
    begin if q . M in S then k : = k + 1;
                        label q in R;
    end
```

sẽ tính giá trị k thỏa mãn điều kiện sau:

(i) k là số phần tử của t_M thuộc vào S

(ii) $k = n$ khi và chỉ khi $S \subseteq t_M$ (4)

Một cải tiến trực tiếp của thuật toán 1 là cho lệnh {test:} vào trong vòng lặp {loop2:}

Thuật toán T.L.1

```
n : = card (S);
{loop1 :} for each r ∈ R do
    begin k : = 0; construct t : = <r.M̄>;
                   construct p : = <r.M>;
                   label r in R;
                   if p in S then k : = k+1;
    {loop 2:} While k < n and not eof (R) do
          begin getnext q from R;
          if q.M̄ = t then
          begin label q in R;
          if q.M in S then
              begin k : = k+1;
              if k=n then add t to T
              end
          end
    end
```

Từ nhận xét 3) ta suy ra rằng thuật toán T.L.1 cũng tính đúng kết quả mong muốn tức là tìm được đúng $T = R : M : S$.

Ta chứng tỏ rằng thuật toán T.L.1 làm việc tốt hơn thuật toán 1.

Thật vậy

i) Đối với các bộ $t \in R[\overline{M}]$ không thuộc R : M : S thì theo (4) ta có $k < n = card (S)$ và do vậy làm việc của hai thuật toán đối với t là như nhau.

ii) Đối với các bộ $t \in R[\overline{M}]$ và $t \in R : M : S$, thuật toán 1 làm việc với thời gian là

$$A = \sum_{i=1}^{m_p} f(m - \sum_{j=0}^{p-1} m_j - i) + m_p f(n)$$

với m_p là số phần tử của lớp tương đương thứ p trong R/ρ chứa t.

Ta xem xét làm việc của thuật toán T.L.1 một cách chi tiết.

Giả sử k = n sau khi đã duyệt θ_p phần tử của lớp tương đương thứ p, dễ thấy $\theta_p \leqslant m_p$ $C_p = m_p - \theta_p$ phần tử còn lại của lớp thứ p sẽ được xem xét về sau bởi vòng lặp {loop1 : }. Tất nhiên là theo nhận xét 3), k luôn luôn giữ giá trị k = 0.

Để duyệt $C_p = m_p - \theta_p$ phần tử đó trong lần duyệt mới ta phải mất

$$\sum_{i=1}^{C_p} f(m - \sum_{j=0}^{p-1} m_j - \gamma_p - i) + (m_p - \theta_p)\, f(n)$$

ở đây γ_p là số phần tử của tập R được duyệt kể từ phần tử đầu tiên (kể cả nó) của lớp thứ p đến phần tử thứ $\theta_p + 1$.

Dễ thấy $\gamma_p \geqslant \theta_p$.

Đối với θ_p phần tử đầu của lớp thứ p ta phải mất

$$\sum_{i=1}^{\theta_p} f(m - \sum_{j=0}^{p-1} m_j - i) + \theta_p f(n)$$

Do vậy, giá tổng cộng để xử lý lớp thứ p là

$$B = m_p f(n) + \sum_{i=1}^{\theta_p} f(m - \sum_{j=0}^{p-1} m_j - i) + \sum_{i=1}^{m_p - \theta_p} f(m - \sum_{j=0}^{p=1} m_j - \gamma_p - i)$$

Ta sẽ chứng tỏ rằng $B \leqslant A(S)$.

Thật vậy, do tính đơn điệu của hàm f và $\gamma_p \geqslant \theta_p$

$$B = m_p f(n) + \sum_{i=1}^{\theta_p} f(m - \sum_{j=0}^{p-1} m_j - i) + \sum_{i=1}^{m_p - \theta_p} f(m - \sum_{j=0}^{p-1} m_j - \theta_p - i)$$

$$= m_p f(n) + \sum_{i=1}^{\theta_p} f(m - \sum_{j=0}^{p-1} m_j - i) + \sum_{i=\theta_p+1}^{m_p} f(m - \sum_{j=0}^{p-1} m_j - i)$$

$$\leqslant m_p f(n) + \sum_{i=1}^{m_p} f(m - \sum_{j=0}^{p-1} m_j - i) = A$$

Mệnh đề 1

Giá của thuật toán T.L.1 không vượt quá giá của thuật toán 1.

Ta lưu ý rằng đối với thuật toán T.L.1 ta vẫn chưa khắc phục được nhược điểm cơ bản là: Vì nhằm mục đích xử lý từng lớp tương đương nên phải chi phí

$$\sum_{i=1}^{m} f(i) - \sum_{p=1}^{v} f(m - \sum_{j=0}^{p-1} m_j)$$

để kiểm tra điều kiện

$q \cdot \overline{M} = t$ đối với thuật toán 1 hay thậm chí

$$\sum_{p=1}^{v} (\sum_{i=1}^{\theta_p} f(m - \sum_{j=1}^{p-1} m_j - i) + \sum_{i=1}^{m_p - \theta_p} f(m - \sum_{j=0}^{p-1} m_j - \gamma_p - i)).$$

Một ý tưởng là:

Đối với mỗi lớp tương đương của quan hệ ρ, như đã xác định ở trên, ta dùng một phần bộ nhớ để chứa $t \in R[\overline{M}]$. Gọi mảng đó là mảng left (Thực ra trong thực hành nếu không thể lưu trữ left trong bộ nhớ trong thì nên tổ chức một tệp repret ở thiết bị nhớ ngoài và left sẽ chứa các con trỏ tới các phần tử của repret).

Mỗi khi duyệt một bộ $r = <r_1, r_2>$, $r_1 \in R[\overline{M}]$, $r_2 \in R[M]$, ta xem xét $r_1 \in R[\overline{M}]$ đã thuộc left chưa. Nếu có rồi thì kiểm tra xem r_2 có thuộc S không, nếu đúng thì tăng cái đếm tương ứng lên 1. Nếu chưa có r_1 trong left thì tạo ra là phần tử mới và cho cái đếm tương ứng là 1 nếu như r_2 thuộc S, bằng 0 nếu ngược lại.

Như vậy, trong cách làm này các lớp tương đương đều được xử lý hết lớp này rồi mới đến lớp kia, như trong thuật toán 1 và thuật toán T.L.1.

Chú ý đến các nhận xét 1) và 2) và tư tưởng trên, ta đi đến thuật toán sau (Thuật toán T.L.2).

Chú thích

Phần tử mảng left (i) chứa $t \in R[\overline{M}]$ khác nhau, đại diện cho lớp tương đương thứ i trong R/ρ. Thực ra khi cài đặt, mảng left nên được tổ chức động nhờ cấu trúc danh sách móc nối. Tuy nhiên cũng có thể mảng left chứa các con trỏ tương ứng đến các phần tử của tập repret như đã nói ở trên. Khi đó một vài thay đổi nhỏ trong thuật toán cần phải được thực hiện.

Count là mảng và Count (i) đếm số phần tử của lớp tương đương thứ i có trong S.

Hàm find(t) với $t \in R[M]$ cho chỉ số k của mảng left sao cho left(k) = t (nếu tìm được) và find(t) = 0 trong trường hợp ngược lại.

Thuật toán T.L. 2

Vào: Quan hệ R, tập thuộc tính $\alpha(R)$

$M \subseteq \alpha(R)$.

Quan hệ S khả hợp với R [M].

Ra: $T = R : M : S = \{y \in R[\overline{M}] \mid \forall x \in S < y, x > \in R\}$

Phương pháp

```
begin n: card (S);
  if n = 0 then T = R[M̄] else
   begin đếm: = 0
        { đếm dùng để chỉ số phần tử hiện có của mảng left }
      for each r ∈ R do
        begin Construct t: = <r . M̄>;
              Construct p: = <r . M>;
              k; = find(t);
         if k = 0 then { diền thêm đại diện vào cuối của left }
           begin đếm: = đếm + 1;
                 left (đếm): = t;
                   k: = đếm;
                 count (k): = 0
```

```
            end
            {k thỏa mãn điều kiện left(k) = t}
            if count (k) < n then
                    if p in S then
            begtn count (k) : = count (k) + 1;
            if count (k) = n then add t to T
            end
          end
    end
end
```

Mệnh đề 2

Thuật toán T.L.2 tìm đúng T = R : M : S

Mệnh đề 3

Nếu không chú ý đến điều kiện count (k) ≠ n trong lệnh kiểm tra thì thuật toán T.L.2 có độ phức tạp thời gian là

$$(m - v)f(v) + \sum_{i=1}^{v} f(i) + mf(n), \quad \text{ở đây}$$

$$m = \text{card}(R), \quad n = \text{card}(S), \quad v = \text{card}(R/\rho).$$

Nếu S = R [M] thì độ phức tạp thời gian của nó là

$$(m - v)\, f(v) + \sum_{i=1}^{v} f(i).$$

Chứng minh:

Để xây dựng danh sách feft ta phải dùng đến hàm find và như vậy phải tốn $\sum_{i=1}^{v} f(i)$.

Sau khi đã có các phần tử đại diện left(i) $i = \overline{1, v}$ của các lớp tương đương trong R/ρ, các phần tử còn lại của R sẽ được tìm bởi hàm find với thời gian (m − v)f(v).

Nếu $S \notin R[M]$ thì để kiểm tra biểu thức logic p *in* S ta phải tốn mf(n).

Do vậy, thời gian tổng cộng là $\sum_{i=1}^{v} f(i) + (m - v)\ f(v) + mf(n)$

Khi S = R[M] thì phần chi phí mf(n) để kiểm tra xem p *in* S hay không là không cần thiết, do vậy trong trường hợp này ta mất $\sum_{i=1}^{v} f(i) + (m - v)\ f)v)$.

Mệnh đề 4

Thuật toán T.L.2 có độ phức tạp thời gian là

$$\sum_{i=1} f(i) + (m - v)f(v) + \left(m - \sum_{p=1}^{v} (m_p - \theta_p)\right) f(n)$$

$$\text{Count}(p) = n$$

khi S không bao hàm trong R [M] và là $\sum_{i=1}^{v} f(i) + (m - v)\ f(v)$ khi S = R [M],

$$m = \text{card}\ (R), \quad n = \text{card}\,(S), \quad v = \text{card}\ (R\ \rho),$$

m_p là số phần tử của lớp tương đương thứ p,

θ_p là số phần tử của lớp tương đương thứ p được xét đủ để kết luận rằng count (p) = n.

Chứng minh:

Lý luận như trong chứng minh mệnh đề 3, để thực hiện thủ tục find đối với các hộ $r \in \mathbb{R}$ ta phải chi phí

$$\sum_{i=1}^{v} f(i) + (m - v) f(v)$$

Bây giờ ta chuyển sang đánh giá thời gian thực hiện phép kiểm tra p *in* S.

Thực vậy, nhờ phép kiểm tra count (k) $<$ n ta sẽ tránh được phép kiểm tra thừa đối với các bộ $<$ t, p $>$ = r t $\in$ R[$\overline{M}$], p $\in$ R[M] để xem rằng p *in* S, khi đã có count (k) = n, k = find (t).

Ta lưu ý rằng để xác định count (k) = n ta phải duyệt θ_k phần tử của lớp tương đương thứ k. Đối với $m_k - \theta_k$ phần tử còn lại không cần phải xem xét xem phần thuộc R[M] có nằm trong S hay không. Từ đây suy ra thời gian để thực hiện phép kiểm tra θ_k phần tử của lớp k là θ_k f(n). Do đó thời gian tổng cộng là

$$\sum_{\substack{k=1 \\ count(k) < n}}^{v} m_k f(n) + \sum_{\substack{k=1 \\ count(k) = n}}^{v} \theta_k f(n) = \sum_{\substack{k=1 \\ count(k) < n}}^{v} m_k f(n) + \sum_{\substack{k=1 \\ count(k) = n}}^{v} m_k f(n) - \sum_{\substack{k=1 \\ count(k) = n}}^{v} (m_k - \theta_k) f(n) =$$

$$= mf(n) - \sum_{\substack{k=1 \\ count(k) = n}}^{v} (m_k - \theta_k) f(n).$$

Thời gian thực hiện của thuật toán T.L. 2 là

$$\sum_{i=1}^{v} f(i) + (m - v) f(v) + [m - \sum_{\substack{k=1 \\ count(k) = n}}^{v} (m_k - \theta_k)] f(n).$$

Nhận ngày 10-12-1984

TÀI LIỆU THAM KHẢO

1. **Nguyễn Xuân Huy**, Toán tử đánh nhãn và các phép toán quan hệ. Khoa học tính toán và điều khiển, N3, 1985.

РЕЗЮМЕ

Некоторые алгоритмы деления в реляционной моделе данных

Предлагаются два алгоритмы реализации операции деления двух отношений. алгоритм с использованием оператора «метка» и алгоритм с использованием списком. Даны оценки перечисленных алгоритмов:

$$C1 \leqslant \sum_{i=1}^{m} f(i) - \sum_{p=1}^{v} f(m - \sum_{j=0}^{p-1} m_j) + mf(n)$$

$$C2 = (m-v)f(v) + \sum_{i=1}^{v} f(i) + mf(n).$$