Tạp chí Tin học và Điều khiển học, T. 19, S. 3 (2003), 233–242

TỐI ƯU HÓA CÂU TRUY VẤN TRONG CƠ SỞ DỮ LIỆU SUY DIỄN BẰNG PHÉP BIẾN ĐỔI MA TẬP

LÊ MẠNH THẠNH, TRƯƠNG CÔNG TUẤN

Trường Đại học Khoa học Huế

Abstract. The magic-sets transformation was a general query optimization technique in deductive databases. However, magic-sets technique may not be the best query optimization strategy. In this paper, we discuss the drawback of magic sets and propose some improvements of magic-sets technique that allow efficient bottom-up computation of answers.

Tóm tắt. Phép biến đổi ma tập (magic sets transformation) được đánh giá là một trong những kỹ thuật tối ưu câu truy vấn rất có hiệu quả trong cơ sở dữ liệu suy diễn. Lý do quan trọng đối với sự thành công của kỹ thuật này là sự kết hợp các ưu điểm của kỹ thuật ước lượng trên xuống (top -down) và dưới lên (bottom-up), từ đó giảm thiểu được số các sự kiện cần tính và tìm kiếm trên cơ sở dữ liệu. Tuy nhiên, phép biến đổi ma tập chưa hẳn là một chiến lược định giá câu truy vấn tốt nhất và nó còn một số hạn chế. Bài báo tập trung tìm hiểu những mặt hạn chế của phép biến đổi ma tập và đề xuất một số cải tiến để nâng cao tính hiệu quả của nó.

1. MỞ ĐẦU

Trong lĩnh vực cơ sở dữ liệu suy diễn, một trong những nhiệm vụ chính là nghiên cứu các kỹ thuật tối ưu câu truy vấn. Nhiều phương pháp định giá câu truy vấn đã được đề xuất và có thể tìm thấy trong các công trình nghiên cứu ([2, 3, 4, 6, 10, 11]). Các phương pháp này có thể chia thành ba hướng: các tiếp cận trên xuống, các tiếp cận dưới lên và các tiếp cận có sự kết hợp các đặc trưng của phương pháp trên xuống và dưới lên.

Trong các phương pháp trên xuống, điểm khởi đầu của việc tính toán chính là từ đích truy vấn và sẽ không tính đến các sự kiện không liên quan với câu truy vấn. Tuy nhiên phương pháp này có thể dẫn đến lặp vô hạn ([9]). Các phương pháp dưới lên đảm bảo tính kết thúc trong quá trình tìm kiếm lời giải của câu truy vấn, nhưng đôi lúc tỏ ra không hiệu quả do thường không định hướng đích và nhiều sự kiện không liên quan đến câu truy vấn cũng được tính. Các chiến lược dưới lên không xem xét câu truy vấn trong suốt quá trình định giá câu truy vấn, tức là việc tính toán không được gắn liền với câu truy vấn như thường xãy ra trong các phương pháp trên xuống ([8]).

Trong thời gian gần đây, một số phương pháp mở rộng để trả lời câu truy vấn được đề xuất nhằm mục đích tạo ra một chiến lược tìm kiếm hướng đích đồng thời có tính hiệu quả và đảm bảo kết thúc quá trình tính toán câu trả lời truy vấn. Điển hình của các phương pháp này là phép biến đổi ma tập ([10, 11]). Phép biến đổi ma tập được đánh giá là một trong những kỹ thuật tối ưu câu truy vấn rất có hiệu quả trong cơ sở dữ liệu suy diễn. Lý do quan trọng đối với sự thành công của kỹ thuật này là nó kết hợp được các ưu điểm của kỹ thuật ước lượng trên xuống và dưới lên, từ đó giảm thiểu được số các sự kiện cần tính và tìm kiếm trên cơ sở dữ liệu. Việc định giá câu truy vấn theo các ma tập về cơ bản là tương đương với các tiếp cận trên xuống. Tính lôi cuốn của kỹ thuật ma tập được thể hiện ở tính

hiệu quả của nó. Tuy nhiên, phép biến đổi ma tập chưa hẳn là một chiến lược định giá câu truy vấn tốt nhất và nó còn một số hạn chế.

Bài báo tập trung thảo luận một số mặt hạn chế của kỹ thuật ma tập trong việc tối ưu câu truy vấn đối với chương trình Datalog, từ đó đề xuất một số cải tiến để nâng cao tính hiệu quả của nó.

Trong bài báo này, chúng tôi chỉ trình bày tóm tắt một số khái niệm cơ sở của phép biến đổi ma tập. Để có các chi tiết đầy đủ hơn cũng như một số khái niệm khác của cơ sở dữ liệu suy diễn có thể xem trong [11].

2. MỘT SỐ KHÁI NIỆM CƠ SỞ

Tô điểm: là cách chú thích trên các vị từ để cung cấp thông tin về các vị từ sẽ được sử dụng như thế nào trong quá trình định giá câu truy vấn.

Định nghĩa 2.1. Một đối của một đích con trong quy tắc r được gọi là buộc nếu trong suốt quá trình định giá câu truy vấn, mọi đích được tạo ra từ đích con này có giá trị hằng ở vị trí của đối này. Ngược lại, đối được gọi là tự do.

Định nghĩa 2.2. Một tô điểm của vị từ $p(t_1, t_2, ..., t_k)$ là một dãy các ký tự b, f có chiều dài k. Nếu ký hiệu thứ i của tô điểm là b thì đối thứ i của p là buộc, nếu ký hiệu thứ i của tô điểm là f thì đối thứ i của p là tự do. Chỉ có các vị từ IDB là được tô điểm.

Định nghĩa 2.3. Cho quy tắc $p \leftarrow q_1 \wedge q_2 \wedge ... \wedge q_n$ và w là tô điểm của vị từ p, tô điểm α_i của các đích con $q_i(t_{i,1}, ..., t_{i,n_i})$ được xác định như sau: Nếu $t_{i,j}$ là hằng hoặc biến đã xuất hiện trong đích con q_k trước đó $(k < i)$ hoặc ở một vị trí buộc của p thì $\alpha_i[j] = b$, ngoài ra thì $\alpha_i[j] = f$ (với $\alpha_i[j]$ là ký hiệu ở vị trí thứ j của tô điểm).

Định nghĩa 2.4. Cho chương trình P, chương trình tô điểm của P, ký hiệu là P^{ad}, gồm các quy tắc được tô điểm của mọi quy tắc trong P.

Định nghĩa 2.5. Tô điểm α của câu truy vấn $p(t_1, ..., t_n)$ được xác định bởi: $\alpha[i] = b$ nếu t_i là hằng và $\alpha[i] = f$ nếu ngược lại.

Truyền thông tin sang ngang: Phép biến đổi ma tập được thực hiện theo một chiến lược truyền thông tin sang ngang (Sips-Sideway Infomation Passing Strategy). Sips là một quyết định về cách thức để lan truyền thông tin trong quy tắc. Sips chỉ ra cách thức để các trị buộc trong đầu của quy tắc được truyền đến thân, thứ tự mà các đích con trong thân sẽ được tính và cách thức để các trị buộc này truyền sang ngang giữa các đích con trong thân quy tắc. Chiến lược truyền thông tin sang ngang có thể biểu diễn bằng các quy tắc được tô điểm. Trong phần bốn của bài báo chúng tôi sẽ thảo luận một số vấn đề liên quan đến Sips. Ta có định nghĩa sau:

Định nghĩa 2.6. Một chiến lược truyền thông tin sang ngang, ký hiệu Sips(r, α), đối với quy tắc r và tô điểm α của đầu quy tắc r là một đồ thị có hướng được xây dựng như sau:

+ Vị từ đầu quy tắc r tạo thành nút xuất phát của đồ thị và chỉ có các cạnh đi.
+ Mỗi vị từ trong thân quy tắc r được biểu diễn bởi một nút trong Sips(r, α).
+ Các cạnh có hướng $p \xrightarrow{s} q$ từ nút p đến nút q được gán nhãn là tập S các biến. Các cạnh và nhãn chỉ định cách thức thông tin được truyền giữa các đích con.

  Các cạnh của đồ thị chỉ ra một thứ tự để các đích con được ước lượng, các nhãn chỉ

định thông tin được truyền sang ngang từ đích con này đến đích con khác.

Ví dụ 2.1. Xem quy tắc sau:

$$(r) : p(X, Y) \leftarrow q(X, Z) \wedge s(Z, Y)$$

Giả sử đối X của vị từ p bị buộc bởi hằng 1, lúc đó Sips(r, α) được biểu diễn như sau:

$$p \xrightarrow{X=1} q \xrightarrow{X=1,Z} s$$

Quy tắc r được tô điểm thành quy tắc: $p^{bf}(X, Y) \leftarrow q^{bf}(X, Z) \wedge s^{bf}(Z, Y)$

3. TỐI ƯU CÂU TRUY VẤN TRÊN CHƯƠNG TRÌNH DATALOG TUYẾN TÍNH PHẢI

Phép biến đổi ma tập được đề xuất bởi Ullman ([11]) đã được đánh giá là một kỹ thuật tối ưu câu truy vấn tốt. Ý tưởng chính của phương pháp ma tập là mô phỏng sự lan truyền các trị buộc được tạo ra trong phương pháp ước lượng trên xuống khi định giá câu truy vấn. Sự lan truyền trị buộc nhận được bằng cách viết lại chương trình ban đầu và câu truy vấn, tạo thành một tập quy tắc mới. Trong mỗi quy tắc một điều kiện mới được thêm vào để hạn chế việc tính các sự kiện không liên quan đến câu truy vấn. Các điều kiện này được xem như là các quan hệ lọc. Từ đó, việc ước lượng trên chương trình viết lại này sẽ hiệu quả hơn so với việc định giá câu truy vấn trên chương trình ban đầu. Tuy nhiên, trên một số lớp con của chương trình Datalog thì phép biến đổi ma tập không được hiệu quả. Trong phần này, chúng tôi xem xét việc áp dụng kỹ thuật ma tập đã được cải tiến của R. Ramakrishnan ([10]) để định giá câu truy vấn trên lớp chương trình Datalog tuyến tính phải. Chiến lược Sip được sử dụng là từ trái sang phải.

Định nghĩa 3.1. Xét chương trình Datalog P chỉ gồm một vị từ IDB p và giả sử k đối đầu tiên $X_1, ..., X_k$ của vị từ p là bị buộc và h đối còn lại $Y_1, ..., Y_h$ là tự do. Chương trình P được gọi là tuyến tính phải nếu:

(i) Mỗi quy tắc đệ quy của P có dạng:

$$p(X_1, ..., X_k, Y_1, ..., Y_h) \leftarrow \text{first}(X_1, ..., X_k, Z_1, ..., Z_k) \wedge p(Z_1, ..., Z_k, Y_1, ..., Y_h) \quad \text{(R)}$$

(ii) Mỗi quy tắc không đệ quy của P có dạng:

$$p(X_1, ..., X_k, Y_1, ..., Y_h) \leftarrow \text{exit}(X_1, ..., X_k, Y_1, ..., Y_h) \quad \text{(E)}$$

trong đó first và exit là hội của các vị từ EDB.

3.1. Phép biến đổi ma tập đối với chương trình Datalog tuyến tính phải

Giả sử P là chương trình Datalog đệ quy tuyến tính phải và câu truy vấn $?\text{init}(X_1, .., X_k) \wedge p(X_1, ..., X_k, Y_1, ..., Y_h)$, trong đó k đối đầu tiên của vị từ p bị buộc vào một tập các giá trị khởi đầu của vị từ EDB init. Phép biến đổi ma tập ([10]) áp dụng cho lớp chương trình này và câu truy vấn sẽ được thực hiện qua hai bước:

Bước 1: Bước tô điểm chương trình: Biến đổi chương trình P ban đầu thành chương trình có tô điểm P^{ad} theo chiến lược Sip đã được chọn.

Bước 2: Bước biến đổi ma tập: Biến đổi chương trình P^{ad} thành chương trình MP^{ad}, được thực hiện như sau:

1) Đối với vị từ đệ quy p^{α} trong chương trình P^{ad} (tô điểm α biểu thị k đối đầu tiên của p là buộc) ta tạo ra một vị từ mag_p^{α} trong MP^{ad}.

2) Đối với mỗi quy tắc đệ quy tuyến tính phải có dạng (R) trong Định nghĩa 3.1, được biến đổi thành hai quy tắc:

$$(\text{mar}_1)\text{: mag\_}p^{\alpha}(Z_1, ..., Z_k) \leftarrow \text{mag\_}p^{\alpha}(X_1, ..., X_k) \wedge \text{first}(X_1, ..., X_k, Z_1, ..., Z_k)$$

(mar_2):
$p^{\alpha}(X_1, ..., X_k, Y_1, ..., Y_h) \leftarrow \text{mag\_}p^{\alpha}(X_1, ..., X_k) \wedge \text{first}(X_1, ..., X_k, Z_1, ..., Z_k) \wedge p^{\alpha}(Z_1, ..., Z_k, Y_1, ..., Y_h)$

3) Đối với mỗi quy tắc không đệ quy có dạng (E) trong Định nghĩa 3.1, được biến đổi thành quy tắc:

$$(\text{mar}_3)\text{: } p^{\alpha}(X_1, ..., X_k, Y_1, ..., Y_h) \leftarrow \text{mag\_}p^{\alpha}(X_1, .., X_k) \wedge \text{exit}(X_1, ..., X_k, Y_1, ..., Y_h)$$

4) Câu truy vấn: $?\text{init}(X_1, ..., X_k) \wedge p(X_1, ..., X_k, Y_1, ..., Y_k)$ được biến đổi thành quy tắc magic không đệ quy: (mar_4): $\text{mag\_}p^{\alpha}(X_1, ..., X_k) \leftarrow \text{init}(X_1, ..., X_k)$.

Việc ước lượng đối với chương trình viết lại bởi phép biến đổi ma tập sẽ cho kết quả của câu truy vấn. Tuy nhiên, kỹ thuật ma tập này chưa thực sự hiệu quả trên lớp chương trình này, ta xem ví dụ sau:

Ví dụ 3.1. Xét chương trình Datalog tuyến tính phải như sau:

r_1:$\text{anc}(X, Y) \leftarrow \text{par}(X, Y)$

r_2: $\text{anc}(X, Y) \leftarrow \text{par}(X, Z) \wedge \text{anc}(Z, Y)$

Câu truy vấn (Q) :? – $\text{anc}(x_0, Y)$.

Giả sử quan hệ của vị từ EDB par được cho bởi tập hợp: $E = \{(x_0, x_1), (x_1, x_2), ..., (x_{n-1}, x_n)\}$. Chương trình được viết lại bởi phép biến đổi ma tập là:

mar_1: $\text{anc}^{bf}(X, Y) \leftarrow \text{mag\_anc}^{bf}(X) \wedge \text{par}(X, Y)$

mar_2: $\text{anc}^{bf}(X, Y) \leftarrow \text{mag\_anc}^{bf}(X) \wedge \text{par}(X, Z) \wedge \text{anc}^{bf}(Z, Y)$

mar_3: $\text{mag\_anc}^{bf}(Z) \leftarrow \text{mag\_anc}^{bf}(X) \wedge \text{par}(X, Z)$

mar_4: $\text{mag\_anc}^{bf}(x_0)$

Sử dụng thuật toán ước lượng theo kiểu dưới lên, chẳng hạn nửa ngây thơ (semi-naive [11]) đối với chương trình viết lại này, ta tìm được lời giải của câu truy vấn chỉ gồm n bộ và số các bộ được tạo ra cho vị từ anc có thời gian thực hiện là $O(n^2)$. Tuy nhiên, để ý rằng chỉ với những bộ phát sinh đối với vị từ mag_anc đã bao gồm tất cả các bộ cần tìm của câu truy vấn, vì vậy chỉ cần áp dụng quy tắc mar_1, ta có thể tìm được tất cả lời giải của câu truy vấn. Trong khi đó quy tắc mar_2 lại tính các tổ tiên của những tổ tiên của x_0, rõ ràng điều này là thừa, vì vậy quy tắc mar_2 có thể loại bỏ, kết quả giảm được số các bộ cần tính đối với vị từ mag_anc xuống còn $O(n)$. Từ ý tưởng đó ta có thể xây dựng các phép biến đổi ma tập cải tiến sau đây:

3.2. Phép biến đổi ma tập tuyến tính phải

Giả sử P là chương trình Datalog đệ quy tuyến tính phải và câu truy vấn $p(c_1, ..., c_k, Y_1, ..., Y_h)$, với $c_1, ..., c_k$ là các hằng. Phép biến đổi ma tập tuyến tính phải sau đây biến đổi chương trình P thành chương trình, ký hiệu là RMP^{ad}, bao gồm các bước:

Bước 1. Biến đổi chương trình P thành chương trình tô điểm P^{ad} theo chiến lược Sip đã

được chọn.

Bước 2. Biến đổi chương trình P thành chương trình RMP^{ad} được thực hiện như sau:

1) Đối với vị từ đệ quy p^α trong chương trình P^{ad} ta tạo ra một vị từ mag_p^α trong chương trình RMP^{ad}.

2) Đối với mỗi quy tắc đệ quy tuyến tính phải có dạng (R) trong Định nghĩa 3.1, được biến đổi thành quy tắc:

$$(\text{mar}_1)\text{: mag\_}p^\alpha(Z_1, ..., Z_k) \leftarrow \text{mag\_}p^\alpha(X_1, ..., X_k) \wedge \text{first}(X_1, ..., X_k, Z_1, ..., Z_k)$$

3) Đối với mỗi quy tắc không đệ quy có dạng (E) trong Định nghĩa 3.1, được biến đổi thành quy tắc:

$$(\text{mar}_2)\text{: } p^\alpha(X_1, ..., X_k, Y_1, ..., Y_h) \leftarrow \text{mag\_}p^\alpha(X_1, ..., X_k) \wedge \text{exit}(X_1, ..., X_k, Y_1, ..., Y_h)$$

4) Câu truy vấn $p^\alpha(c_1, ..., c_k, Y_1, ..., Y_h)$ trở thành:

$$(\text{mar}_3)\text{: mag\_}p^\alpha(c_1, ..., c_k).$$

Phép biến đổi ma tập tuyến tính phải khác với phép biến đổi ma tập ở chỗ nó loại bỏ quy tắc đệ quy mar_2 trong phép biến đổi ma tập. Việc ước lượng trên chương trình RMP^{ad} sẽ sinh ra cùng kết quả với việc ước lượng chương trình P, mặt khác nó hiệu quả hơn vì bậc của vị từ đệ quy là nhỏ hơn.

Ví dụ 3.2. Áp dụng phép biến ma tập tuyến tính phải vào chương trình ở Ví dụ 3.1, nó loại bỏ quy tắc mar_2 được tạo ra bởi phép biến đổi ma tập, kết quả ta nhận được chương trình:

$$(\text{mar}_1)\text{: mag\_anc}^{bf}(Z) \leftarrow \text{mag\_anc}^{bf}(X) \wedge \text{par}(X, Z)$$

$$(\text{mar}_3)\text{: anc}^{bf}(X, Y) \leftarrow \text{mag\_anc}^{bf}(X) \wedge \text{par}(X, Y)$$

$$(\text{mar}_4)\text{: mag\_anc}^{bf}(x_0)$$

Chương trình này sẽ tính các bộ của vị từ mag_anc với thời gian là $O(n)$, từ đó lời giải của câu truy vấn $?p(x_0, Y)$ gồm n bộ sẽ được tính cũng với thời gian $O(n)$.

4. MỘT SỐ VẤN ĐỀ KHÁC LIÊN QUAN ĐẾN PHÉP BIẾN ĐỔI MA TẬP

Trong phần này chúng tôi tập trung thảo luận một số hạn chế của kỹ thuật ma tập và đề xuất một số giải pháp nhằm nâng cao tính hiệu quả của nó trong quá trình định giá câu truy vấn.

4.1. Xác định thứ tự tối ưu của các đích con trong quá trình thực hiện phép biến đổi ma tập

Phép biến đổi ma tập được thực hiện theo hai bước: Bước tô điểm và bước biến đổi chương trình. Một hạn chế của kỹ thuật ma tập là không xác định được thứ tự tối ưu giữa các đích con để thực hiện ở bước tô điểm. Câu truy vấn được tô điểm trong bước đầu tiên nhằm xác định các đối nào của một vị từ là bị buộc và các đối nào là tự do. Chiến lược truyền thông tin sang ngang được dùng trong suốt quá trình tô điểm và thứ tự các đích con trong quy tắc phải được chỉ ra để thực hiện tô điểm. Vì vậy kỹ thuật ma tập sẽ hiệu quả hơn nếu ta xác định được một thứ tự giữa các đích con trong quy tắc để hỗ trợ cho quá trình tô điểm. Tuy nhiên, thường thì không có nhiều thông tin để làm cơ sở đánh giá chi phí về thứ tự phép nối giữa các đích con và khi áp dụng kỹ thuật ma tập thì hoặc chấp nhận một thứ

tự giống như thứ tự của các đích con trong quy tắc hoặc sử dụng một thứ tự được chỉ ra. Ta xét ví dụ sau đây.

Ví dụ 4.1. Cho chương trình P gồm các quy tắc:

$$(\mathrm{r}_1):\ p(X,Y) \leftarrow q(X,Z) \wedge s(Z,Y)$$

$$(\mathrm{r}_2):\ q(X,Y) \leftarrow v(X,W,V,Y) \wedge w(Y,W,U,V)$$

$$(\mathrm{r}_3):\ s(X,Y) \leftarrow t(X,Z) \wedge u(X,Y)$$

Câu truy vấn: $(Q) :? - r(2,X) \wedge p(X,Y)$

Trong đó p, q, s là các vị từ IDB, r, u, v, w và t là các vị từ EDB.

Trong bước tô điểm, đích con $p(X,Y)$ trong truy vấn Q có tô điểm bf bởi vì đối thứ nhất của nó là đối thứ hai của vị từ EDB $r(2,X)$. Vì vậy, quy tắc r_1 được tô điểm với đầu là p^{bf}. Ta cần phải đưa ra một quyết định về thứ tự giữa các đích con q và s trong quy tắc r_1. Lúc này thông tin duy nhất có thể sử dụng để xác định thứ tự các đích con q và s là biến X bị buộc vào một tập các hằng và như vậy có thể xác định thứ tự các đích con trong quy tắc r_1 là (q, s). Kết quả ta nhận được chương trình tô điểm P^{ad} như sau:

$$(\mathrm{ar}_1):\ p^{bf}(X,Y) \leftarrow q^{bf}(X,Z) \wedge s^{bf}(Z,Y)$$

$$(\mathrm{ar}_2):q^{bf}(X,Y) \leftarrow v(X,W,V,Y) \wedge w(Y,W,U,V)$$

$$(\mathrm{ar}_3):s^{bf}(X,Y) \leftarrow t(X,Y) \wedge u(X,Y)$$

Câu truy vấn (Q) được tô điểm: $(AQ) :? - r(2,X) \wedge p^{bf}(X,Y)$

Bước thứ hai của phép biến đổi ma tập sẽ biến đổi chương trình P^{ad} thành chương trình MP^{ad} sau đây:

$$(\mathrm{mar}_1):p^{bf}(X,Y) \leftarrow \mathrm{mag\_}p^{bf}(X) \wedge q^{bf}(X,Z) \wedge s^{bf}(Z,Y)$$

$$(\mathrm{mar}_2):\ q^{bf}(X,Y) \leftarrow \mathrm{mag\_}q^{bf}(X) \wedge v(X,W,V,Y) \wedge w(Y,W,U,V)$$

$$(\mathrm{mar}_3):\ s^{bf}(X,Y) \leftarrow \mathrm{mag\_}s^{bf}(X) \wedge t(X,Y) \wedge u(X,Y)$$

$$(\mathrm{mar}_4):\ \mathrm{mag\_}p^{bf}(X) \leftarrow r(2,X)$$

$$(\mathrm{mar}_5):\ \mathrm{mag\_}q^{bf}(X) \leftarrow \mathrm{mag\_}p^{bf}(X)$$

$$(\mathrm{mar}_6):\ \mathrm{mag\_}s^{bf}(X) \leftarrow \mathrm{mag\_}p^{bf}(X) \wedge q^{bf}(X,Z)$$

Câu truy vấn: $(AQ) :? - r(2,X) \wedge p^{bf}(X,Y)$

Trong chương trình MP^{ad}, việc ước lượng trên quy tắc mar_1 phụ thuộc vào số các bộ nhận được của $\mathrm{mag\_}p^{bf}(X)$. Mặt khác, từ quy tắc mar_4, ta có thể xác định các bộ của $r(2,X)$, từ đó có thể tính được các bộ của $\mathrm{mag\_}p^{bf}(X)$. Như vậy, có thể thấy rằng nếu biết thông tin về các vị từ cung cấp trị buộc thì điều này sẽ giúp ta xác định được thứ tự của các đích con. Tuy nhiên, chỉ khi thực hiện xong bước thứ nhất thì ta mới nhận được vị từ này. Vấn đề này có thể được giải quyết bằng cách áp dụng vị từ magic đối với câu truy vấn Q ngay trong bước tô điểm, từ đó ta có phép biến đổi ma tập cải tiến như sau.

Thuật toán

Input. Chương trình Datalog P, câu truy vấn q.

Output. Chương trình Datalog MP^{ad} sao cho khi ước lượng MP^{ad} bằng các thuật toán kiểu dưới lên sẽ cho ra kết quả của câu truy vấn q.

Phương pháp

Bước 1: Sử dụng vị từ magic để biến đổi câu truy vấn, từ đó xác định một thứ tự giữa các đích con trong các quy tắc để biến đổi chương trình P thành chương trình tô điểm P^{ad}.

Bước 2: Thực hiện các bước biến đổi ma tập trên chương trình P^{ad} đã tạo ra ở bước 1.

Trong trường hợp đặc biệt, nếu P là chương trình Datalog không đệ quy thì trong bước 1, tất cả các quy tắc định nghĩa vị từ magic p có thể sử dụng để xác định được một thứ tự tối ưu giữa các đích con trong quy tắc định nghĩa vị từ p.

Ví dụ 4.2. Xem chương trình trong Ví dụ 4.1, từ câu truy vấn ta tạo ra quy tắc magic (mar_4) định nghĩa vị từ $\text{mag\_}p^{bf}$ và thêm vị từ magic này như là một đích con vào quy tắc r_1, ta nhận được chương trình P' như sau:

(r'_1): $p^{bf}(X,Y) \leftarrow \text{mag\_}p^{bf}(X) \wedge q(X,Z) \wedge s(Z,Y)$

(r_2):$q(X,Y) \leftarrow v(X,W,V,Y) \wedge w(Y,W,U,V)$

(r_3): $s(X,Y) \leftarrow t(X,Y) \wedge u(X,Y)$

(r_4): $\text{mag\_}p^{bf}(X) \leftarrow r(2,X)$

Để ý rằng các đích con trong quy tắc r'_1 không được tô điểm bởi vì chúng ta chưa xác định một thứ tự nối giữa các đích con q và s. Lúc này với các thông tin về vị từ $\text{mag\_}p^{bf}(X)$ cùng với những thông tin của các đích con q và s ta có thể xác định một thứ tự tối ưu của các đích con.

4.2. Hạn chế tính toán dư thừa trên chương trình viết lại bởi phép biến đổi ma tập

Khi kết thúc bước hai của phép biến đổi ma tập, ta nhận được một chương trình mới và việc tìm kiếm lời giải của chương trình viết lại này thường được thực hiện bởi các thuật toán ước lượng dưới lên, chẳng hạn thuật toán nửa ngây thơ, thuật toán này cho phép ngăn chặn việc tính toán lại các sự kiện đã được tính ở bước trước. Tuy nhiên, nó không ngăn chặn được việc dẫn xuất ra các vị từ magic dư thừa. Một vị từ magic tô điểm có thể chứa các vị từ magic tô điểm khác, chẳng hạn vị từ $\text{mag\_}p^{bff}(a)$ chứa vị từ $\text{mag\_}p^{bbf}(a,b)$, bởi vì $\text{mag\_}p^{bff}(a)$ tương ứng với đích $?p(a,X,Y)$, $\text{mag\_}p^{bbf}(a,b)$ tương ứng với đích $?p(a,b,Z)$ và $p(a,X,Y)$ chứa $p(a,b,Z)$. Như vậy, giữa các vị từ magic tô điểm, mặc dầu không có quan hệ về cú pháp nhưng về mặt ngữ nghĩa, một vị từ magic tô điểm có thể chứa các vị từ magic tô điểm khác. Vấn đề ở đây chính là sử dụng các thông tin trong các tô điểm để xác định xem một vị từ magic có chứa vị từ magic khác hay không. Việc kiểm tra quan hệ giữa các vị từ magic có thể thu hẹp thành việc kiểm tra giữa các vị từ tương ứng của chúng. Điều này được thực hiện qua thuật toán sau đây:

Thuật toán kiểm tra quan hệ giữa các vị từ magic được tô điểm

Input. Giả sử $\text{mag\_}p_1^{\alpha}(\overline{c})$ và $\text{mag\_}p_2^{\beta}(\overline{d})$ là hai vị từ magic lần lượt có tô điểm là α và β.

Output. Cho kết quả vị từ $\text{mag\_}p_1^{\alpha}(\overline{c})$ có chứa vị từ $\text{mag\_}p_2^{\beta}(\overline{d})$ hay không.

Phương pháp

Bước 1: Biến đổi các vị từ $\text{mag\_}p_1^{\alpha}(\overline{c})$ thành hạng thức $p_1(\overline{x})$, trong đó $\overline{x}$ bao gồm các hằng trong $\overline{c}$ tương ứng với ký tự 'b' trong α và các biến phân biệt đối với ký tự 'f' trong α. Tương tự biến đổi vị từ $\text{mag\_}p_2^{\beta}(\overline{d})$ thành hạng thức $p_2(\overline{y})$.

Bước 2: Nếu tồn tại hợp nhất tử tổng quát nhất $(mgu)\theta$ của $p_1(\overline{x})$ và $p_2(\overline{y})$ sao cho $p_1\theta$ chính là p_2 thì kết luận vị từ $\text{mag\_}p_1^{\alpha}(\overline{c})$ chứa vị từ $\text{mag\_}p_2^{\beta}(\overline{d})$, ngược lại thì $\text{mag\_}p_1^{\alpha}(\overline{c})$ không chứa $\text{mag\_}p_2^{\beta}(\overline{d})$.

Để ý rằng phép hợp nhất trở nên đơn giản hơn nhiều nếu một trong hai vị từ cần hợp nhất là nguyên tố nền. Trong trường hợp này thì phép hợp nhất thu hẹp thành phép đối sánh các hạng thức.

Ví dụ 4.2. Nguyên tố $p(a, Y, Z)$ và $p(a, b, c)$ có mgu θ là $\{Y/b, Z/c\}$, và ta có $p(a, Y, Z)\theta = p(a, b, c)$. Vậy nguyên tố $p(a, Y, Z)$ chứa $p(a, b, c)$.

Tóm lại, với việc kết hợp thuật toán kiểm tra quan hệ giữa các vị từ magic trong quá trình ước lượng chương trình viết lại bởi phép biến đổi ma tập, ta có thể loại bỏ được các tính toán dư thừa.

4.3. Nâng cao hiệu quả của việc ước lượng chương trình viết lại bởi phép biến đổi ma tập

Sau khi thực hiện phép biến đổi ma tập, các thuật toán dưới lên thường được sử dụng để ước lượng trên tập các quy tắc của chương trình viết lại theo các bước lặp. Quá trình lặp sẽ kết thúc khi không còn sự kiện mới nào được phát sinh. Việc tính toán theo cách này có nhược điểm là phải xem xét đến tất cả các quy tắc trong chương trình ở mỗi bước lặp. Chúng ta có thể cải tiến bằng cách phân rã quá trình tính toán thành một dãy các tính toán nhỏ hơn, trong đó việc tính toán được thực hiện theo từng mức. Ta có một số định nghĩa sau.

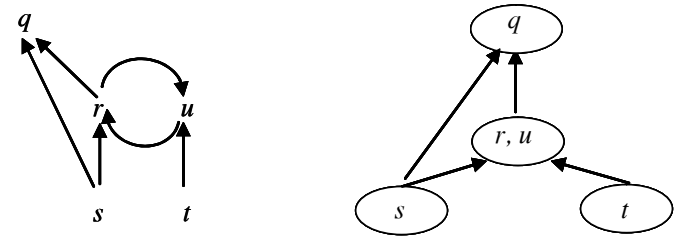
Định nghĩa 4.1. Cho chương trình Datalog P, đồ thị phụ thuộc của P là một đồ thị có hướng $G = (V, E)$, trong đó tập đỉnh V là tập các vị từ có mặt trong chương trình P, E là tập các cạnh (p, q) với $p, q \in V$ và q là vị từ đầu quy tắc chứa vị từ p trong thân quy tắc.

Định nghĩa 4.2

(i) Thành phần liên thông mạnh (SCC - Strongly Connected Component) của đồ thị phụ thuộc $G = (V, E)$ là tập lớn nhất các đỉnh của V sao cho giữa hai đỉnh u và v bất kỳ trong tập này đều tồn tại một đường đi từ u đến v và một đường đi từ v đến u.

(iii) Đồ thị phụ thuộc thu gọn của chương trình Datalog P là một đồ thị có hướng $G' = (V', E')$, trong đó tập đỉnh V' chính là tập các thành phần liên thông mạnh $\{S_1, S_2, ..., S_k\}$ của G. Một cạnh $(S_i, S_j) \in E, i \neq j$, nếu có một cạnh $(p, q) \in E$, trong đó p là vị từ thuộc S_i và q là vị từ thuộc S_j.

Rõ ràng đồ thị phụ thuộc thu gọn G' là một đồ thị có hướng phi chu trình.



Đồ thị phụ thuộc G của P Đồ thị phụ thuộc thu gọn G' của P

Hình 1

Ví dụ 4.3. Xem chương trình Datalog P gồm các quy tắc (để đơn giản ta bỏ qua các đối trong quy tắc) $q \leftarrow r$; $q \leftarrow s$; $r \leftarrow u \wedge s$; $u \leftarrow r \wedge t$. Lúc đó ta có các đồ thị phụ thuộc và đồ thị phụ thuộc thu gọn như trên hình 1.

Định nghĩa 4.3. Một sắp xếp tuyến tính trên đồ thị phụ thuộc thu gọn G' là một sự chỉ định thứ tự các đỉnh của G' sao cho nếu có một cạnh từ đỉnh i đến đỉnh j của G' thì $i < j$.

Với đồ thị phụ thuộc thu gọn G' ở ví dụ trên, gồm các đỉnh $S_1 = \{s\}$, $S_2 = \{t\}$, $S_3 = \{r, u\}$, $S_4 = \{q\}$, ta có S_1, S_2, S_3, S_4 là một sắp xếp tuyến tính của G'.

Định nghĩa 4.4. (Đánh số hiệu của mức cho các đỉnh của đồ thị phụ thuộc thu gọn). Mỗi đỉnh của đồ thị phụ thuộc thu gọn $G' = (V', E')$ được gán một số hiệu của mức, các vị từ EDB thuộc mức 0, một đỉnh S_i được gọi là có mức thấp hơn đỉnh S_j nếu có một đường đi từ S_i đến S_j trong G', các vị từ trong cùng một đỉnh được gán bởi số hiệu mức của đỉnh đó.

Trong Ví dụ 4.3 thì $S_1 = \{s\}$, $S_2 = \{t\}$ thuộc mức 0, $S_3 = \{r, u\}$ thuộc mức 1, $S_4 = \{q\}$ thuộc mức 2.

Thuật toán định giá câu truy vấn theo các mức được thực hiện như sau

1) Biến đổi chương trình Datalog P ban đầu và câu truy vấn thành chương trình MP theo phép biến đổi ma tập.

2) Ước lượng chương trình MP theo các mức: Thực hiện việc đánh số hiệu mức cho các đỉnh của đồ thị phụ thuộc thu gọn G' của MP. Gọi P' là tập các quy tắc trong MP có đầu thuộc vào đỉnh S của G', giả sử S có mức i. Khi đó các vị từ trong tất cả các mức $j (j < i)$ có thể xem là các vị từ EDB cho chương trình P'. Bởi vì các quy tắc có đầu ở mức i thì không chứa các vị từ ở mức $i + 1$ nên có thể tính các quan hệ tương ứng với các vị từ ở các mức này. Từ đó việc ước lượng chương trình MP có thể tiến hành theo từng mức. Mỗi mức được ước lượng bởi một thuật toán kiểu dưới lên, dữ liệu vào của mỗi mức là lời giải của mức trước đó. Ở mỗi mức tùy theo tính chất của các chương trình con mà ta có các thuật toán hiệu quả để tìm lời giải của nó ([7]).

Việc phân nhỏ tính toán theo từng mức có những thuận lợi sau đây:

- Có ít quy tắc được tính toán ở mỗi bước lặp.
- Không phải tất cả nguyên tố trong thân quy tắc là được tính trên mỗi bước lặp.

5. KẾT LUẬN

Bài báo đã tập trung thảo luận một số vấn đề liên quan đến việc tối ưu hoá câu truy vấn trong cơ sở dữ liệu suy diễn bằng phép biến đổi ma tập. Một số hạn chế của kỹ thuật ma tập để định giá câu truy vấn đã được xem xét và các giải pháp để khắc phục. Với một vài thay đổi, phép biến đổi ma tập đã thực sự hiệu quả hơn nhiều trên một lớp con của chương trình Datalog là chương trình Datalog tuyến tính phải. Ngoài ra, một số vấn đề liên quan đến phép biến đổi ma tập như việc xác định thứ tự tối ưu của các đích con trong thân quy tắc, việc loại bỏ tính toán dư thừa trên chương trình viết lại và kỹ thuật phân mức để tăng tính hiệu quả của các thuật toán định giá câu truy vấn cũng được bàn luận trong phần bốn của bài báo. Tuy nhiên, mặc dù vẫn còn một số hạn chế nhưng điều này hoàn toàn không thể phủ nhận phép biến đổi ma tập đã thực sự đóng vai trò rất quan trọng trong việc phát triển các kỹ thuật tối ưu câu truy vấn trong cơ sở dữ liệu suy diễn.

TÀI LIỆU THAM KHẢO

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *Data Structures and Algorithms,* Addison-Wesley, 1983.

[2]] S. Abiteboul, R. Hull, V. Vianu, *Foundations of Databases,* Addision Wesley Publishing, MA, 1995.

[3] A. V. Gelder, A Message Passing Framework for Logical Query Evaluation, *Association for Computing Machinery,* 1986 (155–165).

[4] S. Ceri, G. Gottlob, L.Tanca, *Logic Programming and Databases,* Springer-Verlag Berlin Heidelberg, 1990.

[5] H. Gallaire, J. Minker and J. Nicolas, Logic and Databases: A Deductive Approach, *Computing Survey* **16** (1984).

[6] J. W. Loyd, *Foundations of Logic Programming,* 2nd ed, Springer-Verlag, New York, 1987.

[7] Lê Mạnh Thạnh, Trương Công Tuấn, Một số phương pháp xác định mô hình của chương trình Datalog và mở rộng của nó, *Kỷ yếu Hội thảo tin học toàn quốc,* 1998.

[8] Lê Mạnh Thạnh, Trương Công Tuấn, Một số phương pháp ước lượng câu truy vấn trong cơ sở dữ liệu suy diễn, *Tạp chí Khoa học,* Đại học Huế, (7) (2001) 49–59.

[9] Hồ Thuần, Lê Mạnh Thạnh, Trương Công Tuấn, Phân tích một số phương pháp xử lý vòng lặp vô hạn trong quá trình ước lượng câu truy vấn đối với chương trình Datalog, *Kỷ yếu Hội thảo Khoa học,* Viện Công nghệ Thông tin, 2001. *Tạp chí Tin học và Điều khiển học* **17** (4) (2001) 87–96.

[10] R. Ramakrishnan, Magic Templates: A Spellbinding Approach to Logic Programs, *Journal of Logic Progamming* **11** (1991) 189–216.

[11] J. D. Ullman, *Principles of Database and Knowledge-Base Systems,* Vol. I and II, Computer Science Press, 1989.

Nhận bài ngày 15 - 9 - 2002
Nhận lại sau sửa ngày 10 - 10 - 2002