TÌM VẾT NGUỒN GỐC DỮ LIỆU

Vũ Thanh Nguyên

ABSTRACT

Data ware house, from many data sources which were used, mixed, distributed to mine and analyze data for many problems. In that process original data were changed on transactions, algebra operations, and data clean procedures. To need to find traces of original data from source data on data ware house. In this paper, we research general transformation using attributes and schema mapping. In addition, we also study some algorithms to trace for original data using general transformation. This result was used for tracing of original data for exam data ware house of Private Information Technology College Ho Chi Minh City.

TÓM TẮT

Các hệ thống nhà kho dữ liệu tích hợp thông tin dữ liệu từ các nguồn dữ liệu thao tác, hỗn hợp, phân tán và từ xa vào kho trung tâm để khai khoáng và phân tích thông tin tích hợp ([1],[2],[4],[7]). Nhưng trong suốt quá trình tích hợp, nguồn dữ liệu trải qua hàng loạt các biến đổi có thể thay đổi từ các thao tác hay các kết hợp đại số đơn giản đến các thủ tục làm sạch dữ liệu phức tạp. Vậy trong môi trường nhà kho dữ liệu, vấn đề truy vết nguồn gốc có nghĩa là truy tìm lại nguồn gốc dữ liệu mà nó xuất phát. Trong bài báo này, chúng tôi nghiên cứu về phép biến đổi tổng quát dựa vào các thuộc tính biến đổi và ánh xạ lược đồ. Ngoài ra, chúng tôi nghiên cứu các thuật giải theo vết nguồn gốc dữ liệu qua phép biến đổi tổng quát. Ứng dụng kết quả nghiên cứu trên chúng tôi đã thực hiện truy vết nguồn gốc trong nhà kho tuyển sinh của trường Cao Đẳng Công Nghệ Thông Tin TP.HCM và cho nhiều kết quả lý thú.

I. GIỚI THIỆU

Hệ thống nhà kho dữ liệu tích hợp thông tin dữ liệu từ các nguồn dữ liệu thao tác, hỗn hợp, phân tán và từ xa vào kho trung tâm để khai khoáng và phân tích thông tin tích hợp. Trong môi trường nhà kho dữ liệu, dữ liệu được dùng rất phổ biến cho việc xử lý phân tích trực tuyến (OLAP) ([3],[5],[6],[8],[9]), các hệ thống hỗ trợ quyết định, công bố và phục hồi thông tin trực tuyến, và các thư viện số. Tuy nhiên, ngoài việc phân tích dữ liệu trong kho đôi khi chúng ta cũng cần kiểm tra thông tin trong kho được dẫn xuất từ những nguồn dữ liệu như thế nào. Cho một bộ t trong nhà kho, tìm tập các mẩu dữ liệu nguồn mà t dẫn xuất thì vấn đề này được gọi là nguồn gốc dữ liệu ([10],[11]).

Việc xác định một mẩu dữ liệu từ đâu đến và quy trình mẫu dữ liệu này tham gia vào cơ sở dữ liệu như thế nào ngày càng trở nên quan trọng hơn, đặc biệt trong cơ sở dữ liệu khoa học hiểu được nguồc gốc ở đâu là điều quyết định cho sự chính xác về dữ liệu hiện hành ([8]). Ví dụ, trong lãnh vực sinh học giả sử có hơn 500 cơ sở dữ liệu phổ biến nhưng chỉ có một ít là dữ liệu nguồn trong cơ sở dữ liệu được lấy từ phòng thí nghiệm ([12]). Tất cả những cơ sở dữ liệu khác trong những khung nhìn cụ thể vừa là dữ liệu nguồn vừa là ở khung nhìn khác. Vấn đề người dùng phải đương đầu về cơ sở dữ liệu này là phải biết được nguồn gốc dữ liệu này đến từ đâu. Thông tin này cần thiết cho những ai thật sự quan tâm về mức độ chính xác và hợp lý của dữ liệu.

Để hiểu được nguồn gốc và quá trình tham gia của một mẩu dữ liệu là vấn đề phức tạp. Giả sử cho một khung nhìn cơ sở dữ liệu V = Q(D) và một câu truy vấn Q xây dựng từ cơ sở dữ liệu D và ta thử đặt một câu hỏi về nguồn gốc của mẩu dữ liệu d trong Q(D) như sau: "những phần nào của cơ sở dữ liệu D đã tham gia vào mẩu dữ liệu d". Câu hỏi chính ở đây có nghĩa là "tham gia vào", vậy nên kiểm tra nguồn gốc trong tập tổng quát để phân biệt giữa

"nguồn gốc dữ liệu này ở đâu" và "tại sao nó ở trong gốc này". Xét ví dụ sau:

```
SELECT HoTen, NgaySinh, HoKhau
FROM TuyenSinh
WHERE DiemTC>=All (SELECT DiemTC FROM TuyenSinh)
```

Nếu có một bộ <Le Minh Phuong, 15/02/1983, 2> ở tập đầu ra, ta có thể đặt ra câu hỏi như sau: "những bộ dữ liệu nào đã đưa ra kết quả đó", nếu thay đổi bất cứ bộ nào trong quan hệ TuyenSinh thì sẽ ảnh hưởng đến kết qủa hiện tại của <Le Minh Phuong, 15/02/1983, 2> không. Đây chính là nguồn gốc tại sao ([10]). Mặt khác, giả sử có một câu hỏi khác "Hộ khẩu 2 trong bộ <Le Minh Phuong, 15/02/1983, 2> từ đâu đến", thì câu trả lời đơn giản nhất là "2 là hộ khẩu của Le Minh Phuong trong tập đầu vào" điều này nghĩa là ta đã xác định khóa của quan hệ TuyenSinh chính là HoTen. Nhưng nếu nó không phải thì ta cần những điều kiện khác để xác định bộ trong gốc, bởi vì ngôn ngữ SQL không khử đi các bản sao (chúng ta chỉ dùng câu trả lời SELECT duy nhất sẽ xác định được tập các địa điểm). Thực ra "nguồn gốc ở đâu" yêu cầu chúng ta xác định nơi lưu trữ dữ liệu gốc: Việc nắm rõ về nguồn gốc ở đâu là điều rất quan trọng để hiểu các lỗi sai trong cơ sở dữ liệu (nguồn gốc dữ liệu nào để Le Minh Phuong kiểm tra nếu anh ta phát hiện ra hộ khẩu của mình bị sai trong khung nhìn).

II. PHƯƠNG PHÁP GIẢI QUYẾT

1. Phép biến đổi

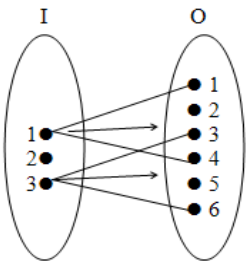
Phép biến đổi T là quá trình nhận các tập dữ liệu đầu vào và sinh ra các tập dữ liệu đầu ra. Cho bất kỳ tập dữ liệu đầu vào I, ứng dụng của T đến I dẫn đến tập đầu ra O, có nghĩa T(I) = O.

Nếu phép biến đổi đã cho xác định toán tử quan hệ hay khung nhìn chuẩn, thì có thể xác định và lấy được chính xác nguồn gốc dữ liệu cho các mẩu đầu ra bằng cách sử dụng các kỹ thuật trong tìm vết gốc khung nhìn dữ liệu trong môi trường nhà kho dữ liệu ([10]). Ngược lại, nếu chúng ta không biết gì về một phép biến đổi thì nguồn gốc của tập đầu ra phải được xác định như toàn bộ tập đầu vào. Thực ra, các phép biến đổi thường nằm giữa hai trạng thái – nó không phải là các toán tử đại số chuẩn, nhưng nó có một vài cấu trúc và thuộc tính để giúp chúng ta xác định nguồn gốc tìm vết dữ liệu.

2. Lớp biến đổi ([11])

Mỗi phép biến đổi là lớp biến đổi dựa vào cách nó ánh xạ các mẩu dữ liệu đầu vào đến các mẩu dữ liệu đầu ra.

2.1. Phép tách (dispatcher)



Hình 1.a. Phép tách

Phép biến đổi T là một phép tách nếu với mỗi mẩu dữ liệu đơn trong tập đầu vào, thì T sinh ra zero hay nhiều mẩu dữ liệu đầu ra.

$$\forall I, T(I) = \bigcup_{i \in I} T(\{i\})$$

Nguồn gốc của mẩu đầu ra o theo phép tách T được định nghĩa:

$T^*(o,I) = \{i \in I \mid o \in T(\{i\})\}$.

Hình 1.a minh họa một phép tách, trong đó mẩu đầu vào 1 thì sinh ra mẩu đầu ra từ 1- 4, mẩu đầu vào là 3 thì sinh ra mẩu đầu ra từ 3-6, còn mẩu đầu vào là 2 thì không sinh ra các mẩu đầu ra nào.

Thuật giải được phát biểu như sau:

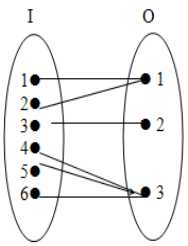
```
Procedure  TraceDS(T,O*,I)
1. I* ← ∅;
2. For each i ∈ I do
3. If T({i}) ∩ O* ≠ ∅ then
4. I* ← I* ∪ {i};
5. Return I*
```

TraceDS quét toàn bộ tập dữ liệu đầu vào. Mỗi tập dữ liệu đầu vào i, gọi phép biến đổi T qua {i}.

Phép tách đặc biệt (Phép lọc - filter): Phép lọc (trường hợp đặc biệt của

dispatcher) nếu mỗi mẩu đầu vào sinh ra chính nó hay không sinh ra gì cả: $\forall i \in I$, $T(\{i\}) = \{i\}$ hay $T(\{i\}) = \varnothing$. Do đó nguồn gốc của bất kỳ mẩu dữ liệu đầu ra giống với mẩu trong tập dữ liệu đầu vào: $\forall o \in O$, $T^*(o) = \{o\}$.

2.2. Phép gộp (aggregator)



Hình 1.b. Phép gộp

Một phép biến đổi T là một phép gộp nếu T đầy đủ, và tất cả I và $T(I) = O = \{o_1,\ldots, o_n\}$, tồn tại duy nhất phần phân chia rời nhau $I_1,\ldots, I_n$ của I như thế $T(I_k) = \{o_k\}$ (k=1..n). $I_1,\ldots,I_n$: sự phân chia đầu vào, I_k: là nguồn gốc của o_k theo T: $T^*(o_k,I) = I_k$.

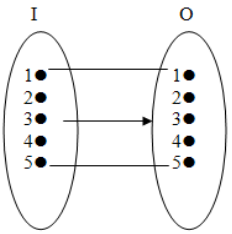
Một phép biến đổi T là đầy đủ nếu mỗi mẩu dữ liệu đầu vào luôn tham gia vào mẩu dữ liệu đầu ra nào đó: $\forall I \neq \varnothing, T(I) \neq \varnothing$.Hình 1.b minh họa phép gộp, nguồn gốc của mẩu đầu ra 1 là từ các mẩu đầu vào {1, 2}, mẩu đầu ra 2 chính là đầu vào {3}, mẩu đầu ra 3 chính là {4, 5, 6}.

Thuật giải được phát biểu như sau:

```
Procedure  TraceAG(T, O*, I)
L ← tất cả tập con của I được lưu trữ bằng kích thước;
For each I* ∈ L (tăng theo thứ tự) do
  If T(I*) = O* then
  If T(I – I*) = O – O* then
  break;
Else  L = tất cả tập con cao nhất  của I* được sắp xếp
Return I*
```

TraceAG (T, O^*, I) liệt kê các tập con của đầu vào I. Trả về tập con duy nhất I^*. Do đó I^* sinh ra chính xác O^* ($T(I^*) = O^*$), và phần còn lại của tập đầu vào sinh ra phần còn lại của tập đầu ra ($T(I – I^*) = O – O^*$). Trong quá trình liệt kê kiểm tra các tập có kích thước tăng. Nếu tìm thấy tập con I^* nghĩa là $T(I^*) = O^*$ nhưng $T(I – I^*) \neq O – O^*$, sau đó chỉ cần kiểm tra tập cao nhất của I^*.

2.3. Hộp đen (black - box)



Hình 1.c. Hộp đen

Toàn bộ tập dữ liệu đầu vào là nguồn của mỗi mẩu đầu ra:$\forall o \in O$, $T^*(o,I)= I$. Do đó thủ tục tìm vết đối với phép biến đổi hộp đen đơn giản trả về toàn bộ đầu vào.

3. Ánh xạ lược đồ (trường hợp cải tiến của lớp biến đổi): sử dụng thông tin lược đồ để cải tiến truy vết nguồn gốc cho phép tách và phép gộp. Xét phép biến đổi T có lược đồ đầu vào là A, khoá đầu vào là A_{key}, lược đồ đầu ra B, và khoá đầu ra là B_{key}.

3.1. Ánh xạ khóa xuôi

T là ánh xạ khóa xuôi forward key – map (fkmap) nếu nó đầy đủ ($\forall I \neq \varnothing$, $T(I) \neq \varnothing$) và có lược đồ ánh xạ xuôi đến khoá đầu ra: $f(A) \xrightarrow{T} B_{key}$.*

```
Procedure TraceFM(T, O*, I)
//let f(A) → Bkey
I* ← ∅
For each i∈ I do
 If f(i.A)∈ πkey(O*) then I* ← I* ∪ {i}
Return I*
```

TraceFM(T, O^*, I) trình bày tìm vết nguồn gốc cho ánh xạ khoá xuôi nhưng TraceFM(T, O^*, I) áp dụng hàm f đến một số thuộc tính của mỗi mẩu dữ liệu.

3.2. Ánh xạ khóa ngược

T là ánh xạ khóa ngược backward key – map (bkmap) nếu có ánh xạ lược đồ ngược đến khoá đầu vào: $A_{key} \xleftarrow{T} g(B)$.*

```
Procedure TraceBM(T, O*, I)
```

* f và g là các hàm từ các bộ giá trị thuộc tính đến các bộ giá trị thuộc tính

// let $A_{key}{}^{T} \leftarrow g(B)$

$I^* \leftarrow \varnothing$;

For each $i \in I$ do

If $i.key \in \pi_{g(B)}(O^*)$ Then $I^* \leftarrow I^* \cup \{i\}$;

Return I^*

Thủ tục TraceBM (T, O^*, I) trình bày tìm vết nguồn gốc cho ánh xạ khóa ngược nhưng hiệu quả hơn TraceDS(T, O^*, I).

3.3. Ánh xạ khóa hoàn toàn ngược

T là ánh xạ hoàn toàn ngược – backward total – map (btmap) nếu có một ánh xạ lược đồ ngược đến tất cả các thuộc tính đầu vào: $A \overset{T}{\leftarrow} g(B)$.

Procedure TraceTM(T, O^*)

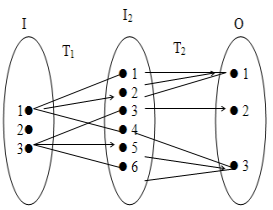
//let $A^{T} \leftarrow g(B)$

return $\pi_{g(B)}(O^*)$;

TraceTM(T, O^*) trình bày tìm vết nguồn gốc cho ánh xạ hoàn toàn ngược, thủ tục này hiệu quả vì nó không cần quét tập dữ liệu đầu vào và không gọi phép biến đổi.

4. Truy vết nguồn gốc qua một chuỗi biến đổi

4.1. Nguồn dữ liệu qua chuỗi biến đổi: Tìm vết nguồn gốc cho 1 phép biến đổi đơn chỉ có 1 tập đầu vào và 1 tập đầu ra.

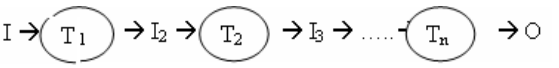


Hình 2. T_1 o T_2

Xét một chuỗi đơn có 2 phép biến đổi, như T_1 o T_2 trong hình trên lấy từ hình hình 1.a và hình 1.b.

Đối với tập dữ liệu đầu vào I, đặt $I_2 = T_1(I)$ và $O = T_2(I_2)$. Đưa ra một mẩu dữ liệu đầu ra $o \in O$, nếu $I^*_2 \subseteq I_2$ là gốc của o theo T_2, và $I^* \subseteq I$ là nguồn gốc của I^*_2 theo T_1, thì I^* là nguồn gốc của o theo T_1 o T_2. Ví dụ, trong hình 2 nếu $o \in O$ là mẩu 3, thì I^*_2 là các mẩu {4, 5, 6} trong I_2 và I^* là các mẩu {1, 3} trong I.

Xét một chuỗi biến đổi T_1 o …o T_n minh họa trong hình sau



Hình 3: Chuỗi biến đổi

Trong đó, mỗi I_k là các kết quả trung gian đầu ra từ T_{k-1} và đầu vào T_k, theo cách tiếp cận Bruteforce là lưu trữ tất cả các kết quả trung gian $I_2,\ldots,I_n$, sau đó quay ngược lại vết gốc thông qua phép biến đổi.

Cách tiếp cận này không hiệu quả do chi phí lưu trữ tất cả các kết quả trung gian thông qua các phép biến đổi trong chuỗi nhiều, và do các thủ tục tìm vết quá lớn được gọi lặp lại qua tất cả các phép biến đổi trong chuỗi. Nói chung trong chu trình tìm vết chuỗi càng dài thì càng ít hiệu quả, và đối với các chuỗi biến đổi thật thì chi phí có thể quá cao. Hơn nữa nếu phép biến đổi T trong chuỗi là một black-box, ta sẽ kết thúc tìm vết nguồn gốc của toàn bộ tập đầu vào đến T cho dù phép biến đổi theo T trong chuỗi như thế nào.

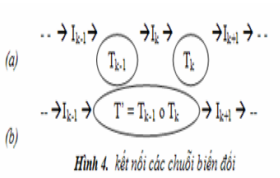
Để làm nhẹ bớt vấn đề này bằng cách nối các phép biến đổi ở gần nhau trong một chuỗi, một vài kết quả trung gian có thể được bỏ đi. Tiến hành sử dụng cách tiếp cận như sau:

- Bước 1: Khi một chuỗi biến đổi S = T_1 o…..o T_n được xác định, đầu tiên phải bình thường hoá chuỗi bằng cách nối các phép biến đổi trong S khi nó có lợi. Sau đó xác định các kết quả trung gian cần lưu để tìm vết nguồn gốc, dựa vào các thuộc tính tốt nhất để duy trì các phép biến đổi.

- Bước 2: Khi dữ liệu được nạp qua phép biến đổi, thì các kết quả trung gian cần thiết được lưu lại.

- Bước 3: Sau đó tìm vết nguồn gốc của bất kỳ mẩu dữ liệu đầu ra o trong nhà kho thông qua chuỗi biến đổi được bình thường hoá sử dụng thuật giải tìm vết có lặp.

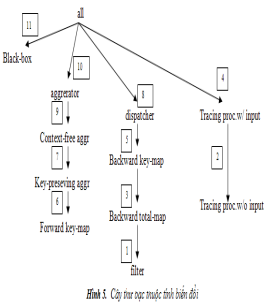
4.2. Bình thường hóa chuỗi biến đổi



Hình 4. kết nối các chuỗi biến đổi

Nối các phép biến đổi T_{k-1} và T_k hình 4.a thay thế 2 phép biến đổi thành phép biến đổi đơn $T'=T_{k-1}$ o T_k

Hình 4.b, giới hạn kết quả trung gian I_k và tìm vết thông qua nối phép biến đổi trong bước 1.



Hình 5. Cây thứ tực thuộc tính biến đổi

Quyết định kết nối mỗi cặp biến đổi có lợi và sử dụng các phép biến đổi nối kết để tìm vết nguồn gốc, bước đầu tiên cần xác định các thuộc tính của một phép biến đổi nối kết dựa vào các thuộc tính của các phép biến đổi cấu thành của nó.

Sau đó, liên kết mỗi phép biến đổi T_k đến tất cả thông tin lược đồ (lược đồ đầu vào $T_k.A$, khoá đầu vào $T_k.A_{key}$, lược đồ đầu ra $T_k.B$, khoá đầu ra $T_k.B_{key}$) và tất cả ánh xạ lược đồ (các ánh xạ xuôi T_k.fmapping và các ánh xạ ngược T_k.bmapping). T_k đầy đủ (T_k.complete) và tập T_k.properties của tất cả các thuộc tính đã cho T_k liệt kê trong cây thứ bậc ở hình bên cạnh.

Thuật giải Combine(T_1,T_2) sau thiết lập các đặc điểm này để nối các phép biến đổi $T = T_1$ o T_2 dựa vào các đặc điểm T_1 và T_2. Thuật giải cần tất cả các đặc điểm này để xác định thuộc tính T.properties từ T_1 và T_2.

Combine(T_1, T_2)

$T \leftarrow T_1$ o T_2;

$T.A \leftarrow T_1.A$; $T.A_{key} \leftarrow T_1.A_{key}$;

$T.B \leftarrow T_2.B$; $T.B_{key} \leftarrow T_2.B_{key}$;

T.fmapping ← ∅ ; T.bmapping ← ∅ ; T.properties ← ∅;

T.complete ← T1.complete and T2.complete;

For each property p in {aggregator, dispatcher, filter, tracing-pro w/o input} do

If $p \in T_1$.properties and $p \in T_2$.properties then add p to T.properties;

For each $f_1(A) \rightarrow A'$ in T_1.fmapping do

If $\exists\ f_2(A') \rightarrow B$ in T_2.fmapping then add f_1 o $f_2(A) \rightarrow B$ to T.fmapings;

For each $A \leftarrow g_1(A')$ in T_1.bmapping do

If $\exists A' \leftarrow g_2(B)$ in T_2.bmapping then add $A \leftarrow g_2$ o $g_1(B)$ to T.bmappings;

If $\exists f(A) \rightarrow T.B_{key}$ in T.fmapping and T.complete then add fkmap to T.properties;

If $\exists T.A_{key} \leftarrow g(B)$ in T.bmapping then add bkmap to T.properties;

If $\exists T.A \leftarrow g(B)$ in T.bmapping then add btmap to T.properties;

Return T;

Theo lý thuyết, ta có thể nối các phép biến đổi gần nhau trong một chuỗi. Thật ra, nó có thể làm sụp đổ toàn bộ chuỗi trong 1 phép biến đổi lớn, nhưng các phép biến đổi kết nối ít thuộc tính mong muốn hơn các phép biến đổi thành phần, dẫn đến việc tìm vết ít hiệu quả và ít chính xác. Do đó, chúng ta nên nối các phép biến đổi chỉ khi nó có lợi. Việc đưa ra một chuỗi biến đổi, xác định cách tốt nhất để nối các phép biến đổi trong một chuỗi là vấn đề kết nối khó khăn, vì phải giải quyết nó chính xác hay thậm chí xác định chính xác khi nó thật hiệu quả để nối hai phép biến đổi, thì sẽ yêu cầu mẫu chi phí chi tiết để đưa vào tính toán các thuộc tính biến đổi gồm chi phí áp dụng phép biến đổi, chi phí lưu các kết quả trung gian và ước tính khối lượng công việc.

YingWei Cui đã đưa ra thuật giải tham lam Normalize như sau:

Normalize(S= T_1 o….oT_n)

While (k ← BestCombo(S)) # 0

Thay thế T_k và T_{k+1} trong S với T ← Combine(T_k, T_{k+1});

If S chứa trong các phép biến đổi black –box Then J ← chỉ mục thấp nhất của black–box trong S;

Thay thế phép biến đổi T_j,…, T_n trong S với T ← T_j o…o T_n;

BestCombo(S = T_1 o….o T_n)

K ← 0; N[1..5] ← [0, 0, 0, 0, 0];

For j=1..n do g ← group(T_i); N[g] ← N[g] + 1//vector khởi tạo

For j=1..n-1 do

CurN ← N;

T ← Combine(T_i, T_{i+1});

g ← group(T);

If g<5 then // T không phải là black - box

g_1 ← group(T_i); g_2 ← group(T_{i+1});

curN[g] ← curN[g] +1 ;

curN[g_1] ← curN[g_1] – 1;

curN[g_2] ← curN[g_2] –1;

if curN<N then k← j ; N ← curN;

return k;

Thuật giải tìm kiếm lặp lại các nối kết có lợi của các cặp biến đổi trong chuỗi, nối cặp tốt nhất cho đến khi không còn các nối kết tốt hơn nữa. Nói chung, một sự nối kết được xem là có lợi với điều kiện nó làm giảm chi phí tìm vết toàn bộ trong khi cải tiến và duy trì tìm vết chính xác. Chúng ta xác định nó có lợi để nối hai phép biến đổi chỉ dựa vào thuộc tính sử dụng hai phương pháp heuristics sau:

- Đầu tiên, ta không nối các phép biến đổi vào black-boxes, trừ khi chúng ta chắc rằng các nối kết sẽ không suy biến chính xác kết quả gốc, mà chỉ được xác định như bước cuối cùng của thuật giải Normalize.

- Thứ hai, ta không nối các phép biến đổi nếu các thành phần kết nối của nó xấu cho việc truy vết nguồn gốc. Vì nó thường có chi phí truy vết cao hoặc dẫn đến kết quả ít chính xác.

Do đó, nên chia các thuộc tính trong hình 5 thành 5 nhóm: nhóm 1 chứa các thuộc tính 1 - 3, nhóm 2 chứa các thuộc tính 4 - 8, nhóm 3 chứa thuộc tính 9, nhóm 4 chứa thuộc tính 10, nhóm 5 chứa thuộc tính 11. Trong mỗi nhóm, hiệu quả và sự chính xác của thủ tục tìm vết là khá giống nhau trong khi các nhóm có ý nghĩa khác nhau. Nhóm của phép biến đổi T, ký hiệu là group (T), là nhóm mà các thuộc tính tốt nhất của T thuộc về nó. Các nhóm còn lại thì thấp hơn, thì chọn T để tìm vết nguồn gốc, và ta xét nó có lợi để nối kết hai phép biến đổi T_1 và T_2 chỉ với điều kiện nếu nhóm (T_1 o T_2) <= max(group(T_1), group(T_2)).

Dựa vào cách tiếp cận trên, thủ tục BestCombo được gọi bởi Normalize, tìm ra cặp biến đổi gần nhau nhất để kết nối vào chuỗi S, và trả lại chỉ mục của nó. Thuật giải trả lại 0 nếu nối kết không có lợi. Xét các nối kết có lợi là tốt nhất nếu nối kết bỏ đi các phép biến đổi xấu nhất trong chuỗi, so với các phần tử tham gia khác. Bình thường, liên kết S là một vector N[1..5], trong đó N[j] là các phép biến đổi trong S thuộc về nhóm j (vì thế tổng của N[j](j=1..5) bằng với chiều dài S). Hai chuỗi S_1 và S_2 có vector N_1 và N_2, k là chỉ mục cao nhất mà trong đó N_1[k] khác với N_2[k]. Chúng ta nói $N_1 < N_2$ nếu $N_1[k] < N_2[k]$, điều đó ý nói S_1 có một vài phép biến đổi xấu hơn S_2. Nói chung nối kết tốt nhất là nối kết mà dẫn đến vector N thấp nhất của chuỗi kết quả.

Sau khi hoàn thành nối kết được mô tả ở trên, giả sử chuỗi vẫn chứa một hay nhiều phép biến đổi black-box. Suốt trong quá trình tìm vết, ta sẽ kết thúc tìm vết nguồn gốc của toàn bộ đầu vào đến black-

box T sớm nhất trong S bất chấp các phép biến đổi theo T như thế nào. Do đó, bước cuối cùng chúng ta nối T với tất cả các phép biến đổi mà cho phép T loại trừ các chi phí lưu trữ và tìm vết không cần thiết.

Thủ tục Normalize có độ phức tạp $O(n^2)$ cho chuỗi biến đổi có chiều dài n. Mặc dù, chúng ta sử dụng thuật giải tham lam và heuristics để ước tính lợi nhuận của các phép biến đổi đang nối kết. Cách tiếp cận của chúng ta hoàn toàn hiệu quả để cải tiến tìm vết cho các chuỗi.

III. ĐÁNH GIÁ KẾT QUẢ

Từ yêu cầu xây dựng nhà kho mới trong phần 1 qua 6 phép biến đổi chúng tôi tiếp cận lý thuyết lớp biến đổi và ánh xạ lược đồ ta đi gán cho mỗi phép biến đổi là một thuộc tính tương ứng tốt nhất, rồi đi tìm vết qua hai phương pháp:

❖ Tìm vết qua từng phép biến đổi trung gian: dựa vào các thuộc tính biến đổi.

❖ Tìm vết bằng cách nối các phép biến đổi ở gần nhau trong chuỗi.

Một số ưu và nhược điểm của 2 phương pháp nêu trên:

Ưu điểm:

▪ Tìm vết qua từng phép biến đổi trung gian tương đối chính xác.

▪ Tìm vết bằng cách nối các phép biến đổi ở gần nhau trong chuỗi giảm không gian lưu trữ và giảm được chi phí thời gian.

Nhược điểm:

▪ Tìm vết qua từng phép biến đổi trung gian tốn nhiều thời gian lưu trữ và thời gian thực hiện.

▪ Tìm vết bằng cách nối các phép biến đổi ở gần nhau trong chuỗi xác định thuộc tính tốt nhất khi nối các chuỗi lại với nhau để tránh trường hợp chuỗi có thể bị bẻ gãy.

IV. KẾT LUẬN

Kết quả bài báo được thực nghiệm từ nguồn dữ liệu tuyển sinh qua các năm của Trường Cao Đẳng Dân Lập Công Nghệ Thông Tin TP.HCM nhằm thống kê yêu cầu xây dựng kho dữ liệu trung tâm của phòng quản lý đào tạo để gởi lên Bộ, các ban ngành khác thực hiện qua các đợt kiểm tra của các ban ngành hay đánh giá kết quả tuyển sinh của từng năm luôn thực hiện qua từng đợt hay từng kỳ. Mục đích là xác định trong những tập thống kê phố biến đó có chính xác với dữ liệu gốc ban đầu tuyển sinh ở đầu vào hay không? Hay còn những đầu vào từ những nguồn khác nữa… Sự chính xác, dữ liệu thật sự rất có ích cho công tác tuyển sinh và đào tạo như: việc thanh tra và kiểm tra của Bộ giáo dục để phát hiện những vi phạm trong quá trình tuyển sinh hay quá trình truy tìm bằng giả... Kết quả nghiên cứu có thể ứng dụng cho nhiều đơn vị nhằm xác định nguồn gốc dữ liệu.

TÀI LIỆU THAM KHẢO

[1] Erhard Rahm và Hong Hai Do (2000). Data cleaning: problems and current approaches. University of Leipzip, Germany.

[2] Felix Naumann (2004). Information ETL & data lineage.

[3] H.Galhardas, D.Florescu, D.Shasha, E.Simon, và C.Saita (2001). Improving data cleaning quality using a data line facility. In Proc. of the third International Workshop on Design and Management of Data Warehouses, Interlaken, Switzerland.

[4] Hao Fan and Alexandra Poulovassilis. Tracing Data Lineage Using Schema Transformation Pathways. School of Computer Science and Information Systems, Birkbeck College, University of London.

[5] Hao Fan (11-2005). Investigating a Heterogeneous Data Integration Approach for Data Warehouse. School of Computer Science and Information Systems, Birkbeck College.

[6] Jennifier Widom (1995). Research problems in data warehousing. Standford University.

[7] Jefferey D. Ullman, biên dịch Trần Đức Quang (2002). Nguyên lý các hệ Cơ Sở Dữ Liệu và Cơ Sở Tri Thức. NXB Thống kê.

[8] Peter Buneman, Sanjeev Khanna, Wang- Chiew Tan (2001). Why and where: a characterization of data provenance, University of Pennsylvania.

[9] Peter Buneman, Sanjeev Khanna, Wang - Chiew Tan (2001). Data provenance: Some basic Ussues. University of Pennsylvania.

[10] Yingwei Cui and Jennifier Widom (2000). Lineage the tracing of view data in a warehousing Environtment. Standford University.

[11] Yingwei Cui and Jennifier Widom (2001). Lineage Tracing for General Data Warehouse Transformation. Standford University.

[12] Infobiogen. DBCAT, The Public Catalog of Databases. http://www.infobiogen.fr/ services/bdcat/, 5 June 2000.