Tạp chí Tin học và Điều khiển học, T.22, S.1 (2006), 14–24

KHÁM PHÁ PHỤ THUỘC ĐA TRỊ DỰA VÀO MA TRẬN PHỤ THUỘC

HỒ THUẦN[1], HOÀNG THỊ LAN GIAO[2]

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

Abstract. The discovery of dependencies and approximate dependencies in databases is one of the problems that were interested by many researchers, such as Hong Yao, Howard J. Hamilton and Cory J. Butz [2], H. Mannila, H. Toivonen, and A. I. Verkamo, [4] and Yka Huhtala, Juha Karkainen, Pasi Porkka [3,6]. In this paper, we propose an algorithm to determine whether a dependency holds or not by building the dependency matrices. So, we can find all multivalued dependencies $X \xrightarrow{U} Y$, with a given set X. These algorithms based on partitions of tuples of the relation with respect to values of the attributes.

Tóm tắt. Sự khám phá những phụ thuộc và phụ thuộc xấp xỉ từ cơ sở dữ liệu là một trong những hướng nghiên cứu được nhiều người quan tâm. Có thể đơn cử một số nhóm tác giả như: Hong Yao, Howard J. Hamilton and Cory J. Butz [2], H. Mannila, H. Toivonen, and A. I. Verkamo [4], và Yka Huhtala, Juha Karkkainen, Pasi Porkka, Hannu Toivonen [5,6]. Trong bài báo này chúng tôi đưa ra một thuật toán kiểm tra phụ thuộc và phụ thuộc xấp xỉ bằng cách xây dựng ma trận phụ thuộc, từ đó tìm tất cả phụ thuộc tối tiểu $X \xrightarrow{U} Y$, với X là một tập các thuộc tính cho trước. Các thuật toán này đều dựa vào sự phân hoạch của quan hệ tương ứng với giá trị của các thuộc tính.

1. MỞ ĐẦU

Sự phụ thuộc giữa các thuộc tính của cơ sở dữ liệu quan hệ biểu diễn hình dạng của cấu trúc trong quan hệ đó. Chẳng hạn như một phụ thuộc hàm $X \xrightarrow{U} Y$ biểu diễn sự phụ thuộc của giá trị thuộc tính Y theo giá trị thuộc tính X: những giá trị của thuộc tính X xác định duy nhất giá trị của thuộc tính Y. Hay phụ thuộc đa trị $X \xrightarrow{U} Y$ là sự phụ thuộc của Y vào X độc lập với các thuộc tính còn lại.

Việc phát hiện các phụ thuộc trong cơ sở dữ liệu rất cần thiết cho mục đích khám phá tri thức và khai phá dữ liệu. Nhiều tác giả [2–6] đã tìm cách xây dựng các thuật toán nhằm khám phá những mối liên hệ giữa các dữ liệu. Trong bài báo này chúng tôi đưa ra thuật toán tìm tất cả vế phải của phụ thuộc đa trị $X \xrightarrow{U} Y$ và thuật toán kiểm tra phụ thuộc đa trị xấp xỉ chấp nhận được với sai số bé hơn hoặc bằng một giá trị ngưỡng không âm cho trước. Để xây dựng các thuật toán này, chúng tôi đã đưa vào khái niệm ma trận phụ thuộc và các đặc trưng của nó cung cấp một công cụ mới cho việc kiểm định phụ thuộc xấp xỉ chấp nhận được một cách dễ dàng.

2. PHỤ THUỘC VÀ PHỤ THUỘC XẤP XỈ

Trong bài này ta luôn xét $\mathcal{A} = (U, A)$ là một hệ thống thông tin với U là tập các đối tượng và A là tập các thuộc tính. Với mỗi $u \in U$ và $a \in A$ ta ký hiệu $u(a)$ là giá trị thuộc tính a của đối tượng u. Nếu $X \subseteq A$ là một tập các thuộc tính ta ký hiệu $u(X)$ là bộ gồm các giá trị $u(a)$ với $a \in X$. Vì vậy, nếu u và v là hai đối tượng thuộc U, ta sẽ nói $u(X) = v(X)$ nếu $u(a) = v(a)$ với mọi thuộc tính $a \in X$.

Định nghĩa 2.1. Cho $X, Y \subseteq A$, ta nói Y phụ thuộc hàm vào X trên U và ký hiệu $X \xrightarrow{U} Y$ nếu

$$\forall u, v \in U : u(X) = v(X) \Rightarrow u(Y) = v(Y).$$

Định nghĩa 2.2. Cho $X, Y \subseteq A$. Đặt $Z = A \setminus (X \cup Y)$. Ta nói Y phụ thuộc đa trị vào X trên U và ký hiệu $X \xrightarrow{U} Y$ nếu

$$\forall u, v \in U : u(X) = v(X) \Rightarrow \exists t \in U : \begin{cases} t(X) = u(X), \\ t(Y) = u(Y), \\ t(Z) = v(Z). \end{cases}$$

Trong nhiều trường hợp ta không nhận được $X \xrightarrow{U} Y$ nhưng tồn tại một tập con $V \subset U$ sao cho $X \xrightarrow{V} Y$. Nếu tập V như vậy nhận được bằng cách loại bỏ một số rất ít các bộ trong U, thì ta nói Y "phụ thuộc đa trị xấp xỉ" vào X trên U. Một cách chặt chẽ, ta đưa vào khái niệm "sai số phụ thuộc" được định nghĩa như sau.

Định nghĩa 2.3. Cho $X, Y \subseteq A$. Ta gọi giá trị sau là sai số của phụ thuộc đa trị $X \xrightarrow{U} Y$

$$e(X \xrightarrow{U} Y) := \frac{\min\{\text{Card}(U \setminus V) \mid V \subseteq U; X \xrightarrow{V} Y\}}{\text{Card}(U)}.$$

Rõ ràng, $X \xrightarrow{U} Y$ khi và chỉ khi $e(X \xrightarrow{U} Y) = 0$. Trong nhiều trường hợp, ngoài các phụ thuộc đa trị, chúng ta cũng cần xác định các phụ thuộc xấp xỉ có sai số không vượt quá một giá trị ngưỡng $\epsilon \in [0, 1)$ cho trước.

Cho $V \subseteq U$ và $X \subseteq A$. Ta gọi tập hợp

$$\text{dom}(V, X) = \{u(X) \mid u \in V\}$$

là miền giá trị của V trên X. Mặt khác, trên V tồn tại một quan hệ tương đương $\text{IND}^V(X)$ xác định bởi

$$\forall u, v \in V : (u, v) \in \text{IND}^V(X) \Leftrightarrow u(X) = v(X).$$

Họ các lớp tương đương trên V, tương ứng với $\text{IND}^V(X)$, được ký hiệu là $[X^V]$.

Bây giờ cho $X, Y \subseteq A$ là hai tập con các thuộc tính. Giả sử

$$[X^U] = \{X_1, X_2, ..., X_m\}; \quad [Y^{X_i}] = \{Y_1^i, Y_2^i, ..., Y_{n_i}^i\}; \quad 1 \leqslant i \leqslant m.$$

Nghĩa là, với mọi bộ $u, v \in U$, ta có

$$u(X) = v(X) \Leftrightarrow \Big(\exists i : u, v \in X_i\Big) \tag{1}$$

$$u(X \cup Y) = v(X \cup Y) \Leftrightarrow \Big(\exists i, j : u, v \in Y_j^i\Big). \tag{2}$$

Định lý sau cho ta các đặc trưng của phụ thuộc hàm và phụ thuộc đa trị.

Định lý 2.1. *Cho $X, Y \subseteq A$. Đặt $Z = A \setminus (X \cup Y)$. Khi đó,*

a) $X \xrightarrow{U} Y \Leftrightarrow [Y^{X_i}] = \{X_i\}$ *(hay $n_i = 1$), với mọi $1 \leqslant i \leqslant m$.*

b) $X \xrightarrow{U} Y \Leftrightarrow \mathrm{dom}(X_i, Z) = \mathrm{dom}(Y_j^i, Z)$, *với mọi $1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant n_i$.*

Chứng minh

a) Hiển nhiên đúng xuất phát từ định nghĩa của phụ thuộc hàm và (1)-(2).

b) ($\Rightarrow$) Giả sử $X \xrightarrow{U} Y$. Vì bao hàm thức $\mathrm{dom}(X_i, Z) \supset \mathrm{dom}(Y_j^i, Z)$ là hiển nhiên, nên ta chỉ cần chứng minh $\mathrm{dom}(X_i, Z) \subset \mathrm{dom}(Y_j^i, Z)$, với mọi i, j. Thật vậy, nếu $\alpha \in \mathrm{dom}(X_i, Z)$ thì $\exists v \in X_i$ sao cho $v(Z) = \alpha$. Lấy $u \in Y_j^i$ suy ra $u \in X_i$ hay $u(X) = v(X)$. Vì $X \xrightarrow{U} Y$ nên tồn tại $t \in X_i$ sao cho $t(Y) = u(Y)$ (nên $t \in Y_j^i$) và $t(Z) = v(Z) = \alpha$. Vậy $\alpha \in \mathrm{dom}(Y_j^i, Z)$.

($\Leftarrow$) Giả sử $\mathrm{dom}(X_i, Z) = \mathrm{dom}(Y_j^i, Z)$, với mọi i, j. Lấy hai bộ bất kỳ $u, v \in U$ mà $u(X) = v(X)$, gọi i và j là các chỉ số sao cho $u, v \in X_i$ và $u \in Y_j^i$. Vì $v \in X_i$ nên $\alpha = v(Z) \in \mathrm{dom}(X^i, Z) = \mathrm{dom}(Y_j^i, Z)$. Nghĩa là tồn tại bộ $t \in Y_j^i$ sao cho $t(Z) = \alpha$. Vì $t, u \in Y_j^i$ nên $t(X) = u(X) = v(X)$ và $t(Y) = u(Y)$. Hơn nữa $t(Z) = v(Z)$. Vậy $X \xrightarrow{U} Y$.

3. ĐẶC TRƯNG PHỤ THUỘC BẰNG MA TRẬN PHỤ THUỘC

Trong mục trước, chúng ta đã đưa ra điều kiện cần và đủ để $X \xrightarrow{U} Y$ đúng. Dựa vào kết quả đó, chúng ta sẽ xây dựng thuật toán khả thi để kiểm tra một phụ thuộc đúng bằng sự phân hoạch trên các giá trị thuộc tính. Với mỗi i cố định, ta ký hiệu $\mathrm{dom}(X_i, Z) = \{\alpha_1, \alpha_2, \ldots, \alpha_p\}$ và $[Y^{X_i}] = \{Y_1^i, Y_2^i, \ldots, Y_q^i\}$ (tức là $q = n_i$). Chúng ta sẽ thiết lập một ma trận phụ thuộc $D^i = (d_{rs})_{p \times q}$ được định nghĩa bởi

$$d_{rs} = \begin{cases} 1 & \text{nếu } \alpha_r \in \mathrm{dom}(Y_s^i, Z), \\ 0 & \text{nếu } \alpha_r \notin \mathrm{dom}(Y_s^i, Z), \end{cases}$$

và ký hiệu

$$\|D^i\| = \sum_{r=1}^{p} \sum_{s=1}^{q} d_{rs} = \sum_{s=1}^{q} \sum_{r=1}^{p} d_{rs}.$$

Từ định nghĩa suy ra $d_{rs} = 1$ khi và chỉ khi có một bộ $t \in Y_s^i$ mà $t(Z) = \alpha_r$. Do đó tổng các số trên hàng thứ r của ma trận $\sum_{s=1}^{q} d_{rs}$ chính là số các bộ (nằm rãi rác trong các Y_s^i) có giá trị tại thuộc tính Z bằng α_r, còn tổng các số trên cột thứ s: $\sum_{r=1}^{p} d_{rs}$ lại chính là $\mathrm{Card}(Y_s^i)$. Hiển nhiên $\|D^i\| = \mathrm{Card}(X_i)$. Nói cách khác, số bộ của X_i chính là số các số 1 xuất hiện trong ma trận D^i. Ta nói ma trận D^i là *dầy đặc* nếu $d_{rs} = 1$, với mọi r, s. Một điều thú vị là phụ thuộc đa trị $X \xrightarrow{U} Y$ có thể được đặc trưng hoàn toàn bởi các ma trận D^i. Điều đó được khẳng định trong định lý sau.

Định lý 3.1. $X \xrightarrow{U} Y \Leftrightarrow D^i$ *là ma trận dầy đặc với mọi $i \in \{1, 2, \ldots, m\}$.*

Chứng minh. Giả sử $X \xrightarrow{U} Y$. Theo Định lý 2.1 , với mọi i ta có

$$\mathrm{dom}(Y_s^i, Z) = \mathrm{dom}(X_i, Z) = \{\alpha_1, \alpha_2, \ldots, \alpha_p\}, \quad 1 \leqslant s \leqslant q = n_i.$$

Do đó, $d_{rs} = 1$ với mọi r, s, nên D^i là ma trận dầy đặc.

Ngược lại, giả sử các D^i đều là ma trận dầy đặc. Với mọi $\alpha_r \in \mathrm{dom}(X_i, Z)$ và Y_s^i $(1 \leqslant s \leqslant q)$ ta có $d_{rs} = 1$ nên $\alpha_r \in \mathrm{dom}(Y_s^i, Z)$. Vậy $\mathrm{dom}(X_i, Z) = \mathrm{dom}(Y_s^i, Z)$. Tức là $X \xrightarrow{U} Y$.

Dễ thấy $X \xrightarrow{U} Y$ khi và chỉ khi $X \xrightarrow{X_i} Y$, $\forall i = \overline{1..n}$. Định lý 3.1 thực ra khẳng định rằng $X \xrightarrow{X_i} Y$ nếu và chỉ nếu D^i là ma trận dầy đặc. Từ đó, nếu $X \overset{U}{\not\longrightarrow} Y$ thì có ít nhất một ma trận D^i không dầy đặc, hay nói cách khác, tồn tại X_i mà $X \overset{X_i}{\not\longrightarrow} Y$. Một câu hỏi khá tự nhiên là nếu các D^i "gần dầy đặc" thì Y có phụ thuộc xấp xỉ X hay không? Để đánh giá khả năng "gần dầy đặc"của họ ma trận phụ thuộc, dưới đây chúng tôi sẽ đưa ra khái niệm về độ dầy đặc của một họ ma trận.

Cho C và D là các ma trận chỉ chứa các giá trị 0 hoặc 1. Ta nói C là ma trận con của D và ký hiệu là $C \prec D$ nếu C có thể thu được từ D bằng cách xóa đi một số hàng và cột nào đó. Nếu D không phải là ma trận dầy đặc, ta cần tìm ma trận con dầy đặc lớn nhất của D. Cụ thể, ta ký hiệu

$$k(D) := \max\big\{\|C\| \mid C \prec D \text{ và } C \text{ dầy đặc}\big\}.$$

với $\|C\|$ là tổng giá trị các phần tử của ma trận C. Bây giờ tương ứng với họ các ma trận $\{D^1, D^2, \ldots, D^m\}$, ta gọi *độ dầy đặc* của họ ma trận này là giá trị sau

$$s(D^1, D^2, \ldots, D^m) := \frac{\sum_{i=1}^{m} k(D^i)}{\sum_{i=1}^{m} \|D^i\|}.$$

Rõ ràng, $s(D^1, D^2, \ldots, D^m) \leqslant 1$ và, từ Định nghĩa 2.3 và Định lý 3.1 ta có

$$s(D^1, D^2, \ldots, D^m) = 1 \Leftrightarrow X \xrightarrow{U} Y \Leftrightarrow e(X \xrightarrow{U} Y) = 0.$$

■

Tổng quát hơn, ta có kết quả sau.

Định lý 3.2.

$$s(D^1, D^2, \ldots, D^m) = 1 - e(X \xrightarrow{U} Y).$$

Trước khi chứng minh định lý này, chúng ta cần làm rõ ý nghĩa của các ma trận con $C^i \prec D^i$. Như nhận xét sau Định lý 3.1, D^i không dầy đặc khi và chỉ khi $X \overset{X_i}{\not\longrightarrow} Y$. Lúc đó, ta cần xóa một số bộ trong X_i để thu được tập $V_i \subset X_i$ mà $X \xrightarrow{V_i} Y$. Theo Định lý 2.1, tồn tại giá trị $\alpha_r \in \mathrm{dom}(X_i, Z)$ mà $\alpha_r \notin \mathrm{dom}(Y_s^i, Z)$, với một s nào đó. Như vậy, để thu được tập hợp $V_i \subset X_i$ có khả năng thỏa mãn phụ thuộc $X \xrightarrow{V_i} Y$ ta cần thực hiện một trong hai thao tác cơ bản sau:

(A): Xóa tất các cả bộ t trong lớp Y_s^i.

(B): Xóa tất cả các bộ $t \in X_i$ mà $t(Z) = \alpha_r$.

Nếu thực hiện thao tác (A) thì tập $V_i \subset X_i$ thu được sẽ tương ứng với ma trận con C^i nhận được từ D^i bằng cách xóa đi cột s. Còn nếu thực hiện thao tác (B) thì tập $V_i \subset X_i$ thu được sẽ tương ứng với ma trận con C^i nhận được từ D^i bằng cách xóa đi hàng r. Như

vậy, để có được phụ thuộc đúng $X \xrightarrow{V_i} Y$ chúng ta cần phải xóa đi các dòng và cột trên ma trận phụ thuộc D^i để có được ma trận con dầy đặc C^i. Số bộ phải xóa chính là tổng số phần tử có giá trị 1 trên các dòng và cột của D^i bị xoá. Tức là $\mathrm{Card}(X_i \setminus V_i) = \|D^i\| - \|C^i\|$, hay $\mathrm{Card}(V_i) = \|C^i\|$. Tóm lại, việc tìm tập con V_i lớn nhất của X_i thỏa mãn $X \xrightarrow{V_i} Y$ tương đương với việc tìm ma trận dầy đặc $C^i \prec D^i$ có $\|C^i\|$ cực đại (tức là $\|C^i\| = k(D^i)$).

Chứng minh Định lý 3.2. Từ những phân tích trên ta thấy, nếu gọi V_i là tập con có số phần tử lớn nhất của X_i, thỏa mãn $X \xrightarrow{V_i} Y$, thì $\mathrm{Card}(V_i) = k(D^i)$. Mặt khác, để ý rằng $\mathrm{Card}(U) = \sum_{i=1}^{m} \|D^i\|$. Nên từ Định nghĩa 2.3 ta có

$$e(X \xrightarrow{U} Y) = \frac{\mathrm{Card}\left(U \setminus \bigcup_{i=1}^{m} V_i\right)}{\mathrm{Card}(U)} = 1 - \frac{\sum_{i=1}^{m} \mathrm{Card}(V_i)}{\sum_{i=1}^{m} \|D^i\|} = 1 - s(D^1, D^2, \ldots, D^m).$$

Định lý được chứng minh. ■

Để minh họa, chúng ta xét hệ thống cho trong Bảng 1. Trong đó,
$[X] = \{X_1, X_2\}$, $X_1 = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}\}$, $X_2 = \{t_{11}, t_{12}\}$,
$[Y^{X_1}] = \{Y_1^1, Y_2^1, Y_3^1\}$,
$Y_1^1 = \{t_1, t_2, t_3, t_4\}$, $Y_2^1 = \{t_5, t_6, t_7\}$, $Y_3^1 = \{t_8, t_9, t_{10}\}$,
$\mathrm{dom}(X_1, Z) = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5\}$,
$\mathrm{dom}(Y_1^1, Z) = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$,
$\mathrm{dom}(Y_2^1, Z) = \{\alpha_3, \alpha_4, \alpha_5\}$,
$\mathrm{dom}(Y_3^1, Z) = \{\alpha_1, \alpha_2, \alpha_4\}$,
$[Y^{X_2}] = \{Y_1^2, Y_2^2\}$; $Y_1^2 = \{t_{11}\}$; $Y_2^2 = \{t_{12}\}$,
$\mathrm{dom}(X_2, Z) = \mathrm{dom}(Y_1^2, Z) = \mathrm{dom}(Y_2^2, Z) = \{\alpha_5\}$.

Bảng 1

U	X	Y	Z
t_1	1	A	α_1
t_2	1	A	α_2
t_3	1	A	α_3
t_4	1	A	α_4
t_5	1	B	α_3
t_6	1	B	α_4
t_7	1	B	α_5
t_8	1	C	α_1
t_9	1	C	α_2
t_{10}	1	C	α_4
t_{11}	2	B	α_5
t_{12}	2	C	α_5

Hai ma trận D^1 và D^2 lần lượt được thiết lập như sau

$$D^1 = \begin{pmatrix} & Y_1^1 & Y_2^1 & Y_3^1 \\ \alpha_1 & 1 & 0 & 1 \\ \alpha_2 & 1 & 0 & 1 \\ \alpha_3 & 1 & 1 & 0 \\ \alpha_4 & 1 & 1 & 1 \\ \alpha_5 & 0 & 1 & 0 \end{pmatrix}; \quad D^2 = \begin{pmatrix} & Y_1^2 & Y_2^2 \\ \alpha_5 & 1 & 1 \end{pmatrix}.$$

Rõ ràng, D^2 dầy đặc. Tuy vậy, D^1 không dầy đặc nên $X \overset{U}{\nrightarrow} Y$. Có thể kiểm chứng được các ma trận con dầy đặc lớn nhất của D^1 và D^2 là

$$C^1 = \begin{pmatrix} & Y_1^1 & Y_3^1 \\ \alpha_1 & 1 & 1 \\ \alpha_2 & 1 & 1 \\ \alpha_4 & 1 & 1 \end{pmatrix}; \quad C^2 = \begin{pmatrix} & Y_1^2 & Y_2^2 \\ \alpha_5 & 1 & 1 \end{pmatrix}.$$

Điều đó tương ứng với việc xóa hoàn toàn lớp $Y_2^1 = \{t_5, t_6, t_7\}$, các bộ trong X_1 có $t(Z) = \alpha_3$ là $\{t_3, t_5\}$, có $t(Z) = \alpha_5$ là $\{t_7\}$ (tổng số có 4 bộ cần xoá). Độ dầy đặc của $\{D^1, D^2\}$ là

$$s(D^1, D^2) = \frac{\|C^1\| + \|C^2\|}{\|D^1\| + \|D^2\|} = \frac{8}{12} = \frac{2}{3}.$$

Do đó, ta có sai số phụ thuộc $e(X \overset{U}{\longrightarrow} Y) = 1 - \dfrac{2}{3} = \dfrac{1}{3}$.

Từ Định lý 3.2 và ví dụ trên ta thấy, chỉ cần làm việc trên các ma trận phụ thuộc, cụ thể là từ giá trị độ dầy đặc của họ ma trận phụ thuộc, chúng ta có thể tính được sai số phụ thuộc và từ đó khẳng định phụ thuộc có chấp nhận được hay không. Trong phần sau chúng ta sẽ bàn chi tiết vấn đề này.

4. THUẬT TOÁN KIỂM ĐỊNH PHỤ THUỘC XẤP XỈ

Từ định nghĩa ta thấy việc xác định $s(D^1, D^2, \ldots, D^m)$ chủ yếu dựa vào việc tính giá trị $k(D^i)$. Vì vậy, trước tiên chúng tôi sẽ xây dựng thuật toán tìm ma trận dầy đặc $C^i \prec D^i$ sao cho $\|C^i\| = k(D^i)$. Từ đó thiết lập thuật toán kiểm định một phụ thuộc xấp xỉ có chấp nhận được không.

4.1. Ý tưởng thuật toán

Để dễ hình dung, chúng tôi trình bày ý tưởng thuật toán thông qua một ví dụ cụ thể. Chúng ta hãy khảo sát lại hệ thống được cho trong Bảng 1. Ta cần phải xác định các ma trận dầy đặc C^1 và C^2, lớn nhất có thể, sao cho $C^1 \prec D^1$ và $C^2 \prec D^2$. Trong ma trận D^1, chúng ta cần cân nhắc xem nên giữ lại số d_{rs} nào cho ma trận C^1. Rõ ràng, các giá trị $d_{rs} = 0$ (như $d_{1,2}, d_{2,2}$...) bắt buộc phải được loại bỏ. Đối với các $d_{rs} = 1$ chúng ta thử tính toán xem *cần loại bỏ tối thiểu bao nhiêu số* 1 *khác trong ma trận nếu quyết định giữ lại vị trí này.* Chẳng hạn, đối với $d_{1,1}$, nếu cho phép $d_{1,1} \in C^1$ thì bắt buộc phải loại đi cột 2 (Y_2^1) và dòng 5 (α_5), do đó số giá trị 1 cần xóa ít nhất là 3. Ta đặt $g_{1,1} = 3$ và gọi đó là *hệ số loại bỏ của* $d_{1,1}$. Tương tự, đối với $d_{5,2}$, nếu muốn giữ lại vị trí này cho C^1 ta cần xóa các cột 1, 3 và các dòng

1, 2, nên phải xóa ít nhất là $g_{5,2} = 7$ số 1. Rõ ràng là những vị trí có hệ số loại bỏ càng bé càng cần được ưu tiên giữ lại cho ma trận C^1. Một cách tự nhiên, ta gán $g_{rs} = \infty$ cho các vị trí $d_{rs} = 0$. Cuối cùng ta nhận được ma trận sau, gọi là *ma trận hệ số loại bỏ* của D^1:

$$G^1 = \begin{pmatrix} & Y_1^1 & Y_2^1 & Y_3^1 \\ \alpha_1 & 3 & \infty & 4 \\ \alpha_2 & 3 & \infty & 4 \\ \alpha_3 & 4 & 5 & \infty \\ \alpha_4 & \underline{\mathbf{1}} & 4 & 3 \\ \alpha_5 & \infty & 7 & \infty \end{pmatrix}$$

Nhìn vào ma trận này ta thấy vị trí $(4, 1)$ có hệ số loại bỏ bé nhất ($g_{4,1} = 1$) nên quyết định giữ lại vị trí này. Điều đó đồng nghĩa với việc xóa đi dòng 5, ma trận D^1 trở thành (số 0 in đậm cho biết vị trí mới bị xoá).

$$D^1 = \begin{pmatrix} & Y_1^1 & Y_2^1 & Y_3^1 \\ \alpha_1 & 1 & 0 & 1 \\ \alpha_2 & 1 & 0 & 1 \\ \alpha_3 & 1 & 1 & 0 \\ \alpha_4 & \underline{\mathbf{1}} & 1 & 1 \\ \alpha_5 & 0 & \underline{\mathbf{0}} & 0 \end{pmatrix}$$

Từ đây, tính lại ma trận hệ số G^1, chọn giữ lại vị trí có hệ số (dương) bé nhất, xóa các dòng cột thích hợp rồi viết lại ma trận D^1, tính ma trận G^1... Quá trình lặp này kết thúc khi ta nhận được ma trận G^1 chỉ chứa các giá trị 0 và ∞. Cụ thể, tiếp tục thao tác đối với ma trận trên ta được

$$\hookrightarrow G^1 = \begin{pmatrix} & Y_1^1 & Y_2^1 & Y_3^1 \\ \alpha_1 & \underline{\mathbf{2}} & \infty & 3 \\ \alpha_2 & 2 & \infty & 3 \\ \alpha_3 & 3 & 5 & \infty \\ \alpha_4 & 0 & 4 & 2 \\ \alpha_5 & \infty & \infty & \infty \end{pmatrix} \to D^1 = \begin{pmatrix} & Y_1^1 & Y_2^1 & Y_3^1 \\ \alpha_1 & \underline{\mathbf{1}} & 0 & 1 \\ \alpha_2 & 1 & 0 & 1 \\ \alpha_3 & 1 & \underline{0} & 0 \\ \alpha_4 & 1 & \underline{\mathbf{0}} & 1 \\ \alpha_5 & 0 & 0 & 0 \end{pmatrix} \to$$

$$\hookrightarrow G^1 = \begin{pmatrix} & Y_1^1 & Y_2^1 & Y_3^1 \\ \alpha_1 & 0 & \infty & \underline{1} \\ \alpha_2 & 0 & \infty & 1 \\ \alpha_3 & 3 & \infty & \infty \\ \alpha_4 & 0 & \infty & 1 \\ \alpha_5 & \infty & \infty & \infty \end{pmatrix} \to D^1 = \begin{pmatrix} & Y_1^1 & Y_2^1 & Y_3^1 \\ \alpha_1 & 1 & 0 & \underline{\mathbf{1}} \\ \alpha_2 & 1 & 0 & 1 \\ \alpha_3 & \underline{\mathbf{0}} & 0 & 0 \\ \alpha_4 & 1 & 0 & 1 \\ \alpha_5 & 0 & 0 & 0 \end{pmatrix} \to$$

$$\hookrightarrow G^1 = \begin{pmatrix} & Y_1^1 & Y_2^1 & Y_3^1 \\ \alpha_1 & 0 & \infty & 0 \\ \alpha_2 & 0 & \infty & 0 \\ \alpha_3 & \infty & \infty & \infty \\ \alpha_4 & 0 & \infty & 0 \\ \alpha_5 & \infty & \infty & \infty \end{pmatrix}.$$

Cuối cùng ta nhận được ma trận con dầy đặc C^1 gồm sáu số 1. Đó cũng chính là số các giá trị 0 còn lại trong ma trận G^1. Lúc đó, $k(D^1) = \|C^1\| = 6$. Đối với D^2 thì cũng bằng kỹ thuật như trên ta thu được $G^2 = (0 \quad 0)$. Nên $C^2 = D^2$ và $k(D^2) = \|C^2\| = 2$.

4.2. Thuật toán tổng quát xác định độ dầy đặc

Bây giờ chúng tôi sẽ đưa ra thuật toán tìm các $k(D^i)$. Trước hết, đối với mỗi ma trận phụ thuộc D^i, chúng ta cần xây dựng ma trận hệ số phụ thuộc G^i (thủ tục Compute_G^i). Một cách tổng quát, ma trận $G^i = (g_{rs})$ được xây dựng như sau: Nếu $d_{rs} = 0$ thì đặt $g_{rs} = \infty$, ngược lại, nếu $d_{rs} = 1$ thì g_{rs} là tổng số các số 1 tối thiểu cần phải loại để giữ lại số 1_{rs} trong ma trận phụ thuộc D^i, những số 1 chắc chắn bị loại là những số nằm trên dòng u mà $d_{us} = 0$ hoặc nằm trên cột v mà $d_{rv} = 0$. Nói cách khác, khi giữ lại số 1_{rs} thì phải loại các số 1_{uv} nếu $d_{us} * d_{rv} = 0$. Tóm lại, ta có

$$g_{rs} = \begin{cases} \infty & \text{nếu } d_{rs} = 0, \\ \displaystyle\sum_{d_{us} \times d_{rv} = 0} d_{uv} & \text{nếu } d_{rs} = 1. \end{cases}$$

Rõ ràng, $g_{rs} = \infty$ nếu $d_{rs} = 0$, $g_{rs} = 0$ nếu $d_{rs} = 1$ và việc giữ lại vị trí này không đòi hỏi phải xóa đi bất kỳ số 1 nào. Cuối cùng, $g_{rs} = l > 0$ nếu $d_{rs} = 1$ và việc giữ lại vị trí này đòi hỏi phải xóa đi ít nhất l số 1 khác.

Nếu ma trận G^i chỉ gồm các giá trị 0 và ∞ thì các vị trí tương ứng với giá trị 0 sẽ xác định một ma trận con dầy đặc của D^i, thuật toán dừng. Trong trường hợp ngược lại, chúng ta cần xác định vị trí 1_{hv} với $g_{hv} > 0$ đáng được giữ lại nhất. Với ý nghĩa của các g_{rs} như đã phân tích trên, rõ ràng vị trí này cần thỏa mãn điều kiện sau

$$g_{hv} = \min\{g_{rs} \mid g_{rs} > 0\}.$$

Với quyết định giữ lại giá trị 1_{hv} ta phải xóa đi các số 1_{rs} liên quan (tức là thỏa mãn $d_{hs} * d_{rv} = 0$) và nhận được ma trận D^i mới (thủ tục Modify_D^i). Quá trình tiếp tục cho đến khi nhận được ma trận G^i chỉ gồm các giá trị 0 và ∞. Thuật toán được trình bày chi tiết như sau:

Thuật toán 1. (Xác định $k(D^i)$)
Input D^i.
Output $k(D^i)$.
B1. $k := \|D^i\|$.
B2. Compute_G^i.
B3. Xác định $g_{hv} = \min\{g_{rs} \mid g_{rs} > 0\}$.
B4. Nếu $g_{hv} = \infty$ thì Output k và dừng;
Ngược lại, sang B5.
B5. Modify_D^i; $k := k - g_{hv}$; Quay lại B2.
Proc Compute_G^i
For $r \in \{1, ..., p_i\}$; $s \in \{1, ..., q_i\}$ do
If[1] $d_{rs} = 0$ then $g_{rs} := \infty$
Else
$g_{rs} := 0$;
For $u \in \{1, ..., p_i\}$; $v := \{1, ..., q_i\}$ do

If^2 $d_{us} * d_{rv} = 0$ then $g_{rs} := g_{rs} + d_{uv}$ EndIf^2

EndIf^1

EndProc Compute_G^i

Proc Modify_D^i

For $r \in \{1..p_i\}; s \in \{1..q_i\}$ do

If $d_{rv} * d_{hs} = 0$ then $d_{rs} := 0$.

EndProc Modify_D^i

Giả sử ma trận D^i có kích thước $p_i \times q_i$. Lúc đó, thủ tục Compute_G^i có độ phức tạp tính toán $O(p_i^2 q_i^2)$, còn thủ tục Modify_D^i có độ phức tạp tính toán $O(p_i q_i)$. Do đó, độ phức tạp của mỗi vòng lặp là $O(p_i^2 q_i^2)$ và thuật toán xác định $k(D^i)$ có độ phức tạp tính toán trong trường hợp xấu nhất là $O(p_i^3 q_i^3)$. Chú ý rằng, nếu ký hiệu $x_i = \text{Card}(X_i) = \|D^i\|$ thì ta có ước lượng: $x_i \leqslant p_i q_i \leqslant x_i^2$.

4.3. Thuật toán kiểm định phụ thuộc xấp xỉ

Trong phần này, chúng ta sẽ đánh giá xem giữa hai tập thuộc tính đã cho $X, Y \subseteq A$, phụ thuộc $X \overset{U}{\longrightarrow} Y$ có chấp nhận được với sai số ϵ cho trước không.

Từ Định lý 3.1 và Định lý 3.2, chúng ta thấy rằng, phụ thuộc $X \overset{U}{\longrightarrow} Y$ là chấp nhận được nếu $s(D^1, D^2, \ldots, D^m) \geqslant 1 - \epsilon$. Tức là

$$\sum_{i=1}^{m} k(D^i) \geqslant (1 - \epsilon)\text{Card}(U).$$

Thuật toán được xây dựng như sau

Thuật toán 2. (Kiểm tra phụ thuộc chấp nhận được)

Input Hệ thống thông tin $\mathcal{A} = (U, A)$,

Các tập $X, Y \subseteq A$; Giá trị ngưỡng $\epsilon \geqslant 0$.

Output $e(X \overset{U}{\longrightarrow} Y) \leqslant \epsilon$?

B1. $s := 0$; OK:= TRUE.

B2. For $i \in \{1, ..., m\}$ do

Xây dựng D^i;

$s := s + k(D^i)$;

If $s \geqslant (1 - \epsilon) \times \text{Card}(U)$ then Exit.

EndFor.

OK:= FALSE.

Trong thuật toán trên, m chính là số lớp tương đương của IND(X). Giả sử Card(U) $= n$ và mỗi ma trận D^i có kích thước $p_i \times q_i$. Khi đó việc xây dựng các ma trận D^i có thể thực hiện bằng cách sắp xếp các đối tượng trong U, vì vậy độ phức tạp của thuật toán xây dựng dãy các ma trận D^i là $O(n \log n)$ và độ phức tạp của thuật toán kiểm chứng phụ thuộc xấp xỉ được xác định bởi vòng lặp For ở B2. Thực chất ở đây chúng ta thực hiện hai vòng lặp, một vòng lặp xây dựng các D^i và một vòng lặp tính sai số phụ thuộc. Vòng lặp tính sai số phụ thuộc có độ phức tạp trong trường hợp xấu nhất là $O(\sum_{i=1}^{m} p_i^3 q_i^3)$. Vì $p_i q_i \leqslant x_i^2$, nên độ phức tạp tính toán của Thuật toán 2 không vượt quá $O(n \log n + \sum_{i=1}^{m} x_i^6)$.

5. THUẬT TOÁN TÌM KIẾM PHỤ THUỘC TỐI TIỂU

Một phụ thuộc $X \xrightarrow{U} Y$ được gọi là tối tiểu về phải và ký hiệu $X \xrightarrow{U}_{\min} Y$ nếu với mọi $Y' \subset Y$, $X \not\xrightarrow{U} Y'$. Khi đó, nếu $X \xrightarrow{U} Y$ thì $Y = Y_1 \cup Y_2 \cup \cdots \cup Y_n$ với $X \xrightarrow{U}_{\min} Y_i$ với mọi $1 \leqslant i \leqslant n$. Vì vậy, trong phần này chúng tôi chỉ đặt vấn đề tìm kiếm tất cả các phụ thuộc tối tiểu về phải với vế trái X là một tập con các thuộc tính cho trước .

Thuật toán chúng tôi sắp đề nghị sử dụng một phần ý tưởng của Tane [5] khi các tác giả này thiết kế thuật toán tìm các phụ thuộc hàm không tầm thường dạng $X \setminus \{C\} \to C$. Cụ thể, chúng tôi sẽ tìm các tập con $Y \subseteq A$ thỏa mãn $X \xrightarrow{U}_{\min} Y$ theo thứ tự tăng dần của $\mathrm{Card}(Y)$. Nghĩa là, việc tìm kiếm vế phải của phụ thuộc xuất phát từ họ các tập có một phần tử, tập có hai phần tử ...

Gọi L_i là họ các tập thuộc tính Y có kích thước i và có khả năng thỏa $X \xrightarrow{U} Y$. Vì $X \xrightarrow{U}_{\min} Y$ thì $X \cap Y = \emptyset$ nên chúng ta khởi tạo $L_1 := \{\{a\} \mid a \in A \setminus X\}$. Hơn nữa, nếu $Y \in L_i$ và $X \xrightarrow{U}_{\min} Y$ thì Y không thể là tập con của tập $Y' \in L_j$ với $j > i$ mà $X \xrightarrow{U}_{\min} Y'$ hay Y sẽ không tham gia vào việc xây dựng họ L_j, với mọi $j > i$. Vì vậy ở đây chúng tôi dùng thêm họ M_i lưu các tập $Z \in L_i$ mà $X \not\xrightarrow{U} Z$ và xây dựng họ L_{i+1} dựa vào M_i. Cụ thể, sau khi tìm tất cả các phụ thuộc $X \xrightarrow{U}_{\min} Y$ với $Y \in L_i$, chúng ta xây dựng L_{i+1} bằng cách lấy hợp hai tập bất kỳ $Y_1, Y_2 \in M_i$, nếu hợp này cho ta một tập có $i+1$ phần tử thì đưa nó vào L_{i+1}. Rõ ràng, M_i khởi tạo bằng L_i, khi có một phụ thuộc $X \xrightarrow{U} Y$ đúng với $Y \in L_i$ thì loại Y khỏi M_i.

Ngoài ra, nếu $Z = A \setminus (X \cup Y)$ thì từ $X \xrightarrow{U} Y$ ta cũng có $X \xrightarrow{U} Z$ và $\mathrm{Card}(Y) + \mathrm{Card}(Z) = \mathrm{Card}(A \setminus X)$. Vì vậy quá trình tìm kiếm của chúng tôi chỉ diễn ra trong các họ L_i với $i \leqslant \mathrm{Card}(A \setminus X)/2$.

Thuật toán 3.
Input Hệ thống thông tin $\mathcal{A} = (U, A)$,
Tập con $X \subseteq A$.
Output Tất cả phụ thuộc $X \xrightarrow{U}_{\min} Y$.
B1. $i := 1$; $L_1 := \{\{a\} \mid a \in A \setminus X\}$.
B2. While $i \leqslant \mathrm{Card}(A \setminus X)/2$ do
$M_i := L_i$;
For $Y \in M_i$ do
If $X \xrightarrow{U} Y$ then
Output $X \xrightarrow{U}_{\min} Y$;
$M_i := M_i \setminus \{Y\}$.
EndIf
EndFor
Compute_L_{i+1}
$i := i + 1$
EndWhile.

Proc Compute L_{i+1}
$L_{i+1} := \emptyset$;

For $X, Y \in M_i$, $X \neq Y$ do
$Z := X \cup Y$;
If Card$(Z) = i + 1$ then
$L_{i+1} := L_{i+1} \cup \{Z\}$
EndIf
EndFor

EndProc Compute _L_{i+1}

Độ phức tạp tính toán của thuật toán này được xác định bởi vòng lặp While ở B2. Tại mỗi bước của vòng lặp, chúng ta xét tất cả các phần tử của M_i và kiểm tra phụ thuộc, sau đó tính L_{i+1} theo M_i. Chú ý rằng Card$(M_i) \leqslant C_k^i$ ($k =$ Card$(A \setminus X)$). Mặt khác, độ phức tạp của thuật toán kiểm tra phụ thuộc là $O(n \log n + \sum_{i=1}^{m} x_i^6)$, còn độ phức tạp của thủ tục Compute_L_{i+1} là $O(C^2_{\text{Card}(M_i)}) = O(\text{Card}(M_i)^2)$. Vì vậy độ phức tạp tính toán của Thuật toán 3 trong trường hợp xấu nhất là $O(\sum_{i=1}^{k/2} C_k^i \times (n \log n + \sum_{i=1}^{m} x_i^6)) = O(2^{k/2} \times (n \log n + \sum_{i=1}^{m} x_i^6))$. Trong đó, $k =$ Card$(A \setminus X)$, $n =$ Card(U), m là số lớp tương đương của IND(X) và x_i là lực lượng của lớp tương đương X_i, $1 \leqslant i \leqslant m$.

TÀI LIỆU THAM KHẢO

[1] Hồ Thuần, Contribution to the theory of relational databases, *Tanulmanyok 184/1986, Studies 184/1986,* Budapest, Hungary.

[2] Hong Yao, Howard J. Hamilton, and Cory J. Butz, FD_Mine: Discovering functional dependencies in a database using equivalences, *University of Regina Computer Sciences Department Technical Report CS-02-04, ISBN ăă0-7731-0441-0.*

[3] Iztok Savnik and Peter A. Flash, Discovery of multivalued dependencies from ralations, *Intelligent Data Analysis* **4** (3,4) (2000) 195–211.

[4] H. Mannila, H. Toivonen, and A.I. Verkamo, Discovery of frequency episodes in event sequences, *Data Mining and Knowledge Discovery* **I** (1997) 259–289.

[5] Yka Huhtala, Juha Karkkainen, Pasi Porkka, and Hannu Toivonen, Tane: An efficient algorithm for discovering functional and approximate dependencies, *The Computer Journal* **42** (2) (1999).

[6] Yka Huhtala, Juha Karkkainen, Pasi Porkka and Hannu Toivonen, Efficient discovery of functional and approximate dependencies using partitions, *Proc, 14th Int. Conf. on Data Engineering, IFFE Computer Society Press* (ICDE'98).

Nhận bài ngày 28 - 5 - 2005
Nhận lại sau sửa ngày 13 - 1 -2006