Tạp chí Tin học và Điều khiển học, T.25, S.4 (2009), 355–370

VỀ MỘT QUY TRÌNH PHI CHUẨN HÓA CƠ SỞ DỮ LIỆU

ĐẶNG HỮU ĐẠO, NGUYỄN HOÀNG HÀ, NGUYỄN MINH TUẤN,
HOÀNG ĐỖ THANH TÙNG

Viện Công nghệ thông tin, Viện Khoa học và Công nghệ Việt Nam

Abstract. Databases are typically normalized to ensure data consistency, but data capacity and access frequency are not adequately taken into account, which makes it difficult to access and generate reports from the databases. In this paper, we propose a denormalization process for speeding up the report generation and access. A software implementation of the process is described.

Tóm tắt. Các cơ sở dữ liệu (CSDL) thường được chuẩn hóa khi xây dựng để đảm bảo tính toàn vẹn thông tin, nhưng khối lượng, tần suất truy xuất dữ liệu lưu trữ chưa được cân nhắc dẫn đến tốc độ kết xuất các báo cáo từ CSDL ngày càng chậm và khó khai thác. Vậy, để cải thiện tình hình này, chúng tôi đề xuất một quy trình phi chuẩn hoá CSDL và xây dựng phần mềm thực hiện quy trình đó nhằm cải tiến tốc độ truy xuất các báo cáo.

MỞ ĐẦU

Đối với các hệ thống thông tin quản lý, báo cáo đóng vai trò rất quan trọng vì báo cáo là kênh thông tin đầu ra chủ yếu. Thực chất của việc tạo báo cáo là kết xuất thông tin từ CSDL thành các dạng tài liệu có nội dung và hình thức hữu ích với người dùng. Ban đầu, khi các hệ thống mới được triển khai, lượng dữ liệu còn ít thì tốc độ tạo báo cáo thường rất nhanh. Nhưng sau một thời gian hoạt động, lượng dữ liệu ngày càng lớn, có những bảng chứa hàng triệu bản ghi thì tốc độ kết xuất báo cáo thường chậm dần, thậm chí chậm đến mức không chấp nhận được.

Tuy các bảng có thể có rất nhiều bản ghi, nhưng thực sự số bản ghi ảnh hưởng trực tiếp đến kết quả báo cáo thường không phải là nhiều, do vậy vấn đề chính ở đây là làm sao tổ chức thông tin để đảm bảo truy xuất hiệu quả.

Việc thiết kế CSDL có ảnh hưởng rất lớn đến tốc độ kết xuất báo cáo. Các CSDL thường được chuẩn hóa, nhờ đó, thông tin được bảo đảm tính toàn vẹn và tránh được tối đa sự sai sót hoặc mất mát thông tin. Tuy nhiên, thực hiện chuẩn hóa làm số bảng tăng lên, dẫn đến khi cần tổng hợp thông tin chúng ta thường phải kết hợp (JOIN) nhiều bảng với nhau. Điều này làm tăng độ phức tạp của câu truy vấn dẫn đến làm giảm tốc độ kết xuất. Đặc biệt tốc độ thực hiện các câu truy vấn có nhiều JOIN giảm xuống rõ rệt trong trường hợp các bảng thành phần có số lượng bản ghi lớn.

Để khắc phục tình trạng này, trong những trường hợp cụ thể người ta tiến hành phi chuẩn hóa. Đây là công việc trái ngược với chuẩn hóa. Quá trình phi chuẩn hóa đặt một

thông tin vào nhiều chỗ [2] nhằm tăng tốc độ truy vấn. Tuy phi chuẩn hóa làm dư thừa dữ liệu, nhưng nhờ thông tin cần truy xuất được đặt gần nhau nên câu truy vấn dữ liệu cho báo cáo sẽ đơn giản hơn, giảm thiểu các phép JOIN, nhờ đó tốc độ tạo báo cáo sẽ nhanh hơn. Đã có những nghiên cứu tổng kết các chiến lược phi chuẩn và đưa ra các đánh giá định lượng về hiệu quả của phi chuẩn [1].

Bài báo này này nhằm trình bày một quy trình phi chuẩn hoá CSDL nhằm cải thiện tốc độ truy xuất báo cáo bao gồm quy trình và phần mềm hỗ trợ thực hiện quy trình. Bài báo gồm hai phần: phần I trình bày quy trình phi chuẩn hoá CSDL, phần II trình bày phần mềm hỗ trợ thực hiện quy trình này.

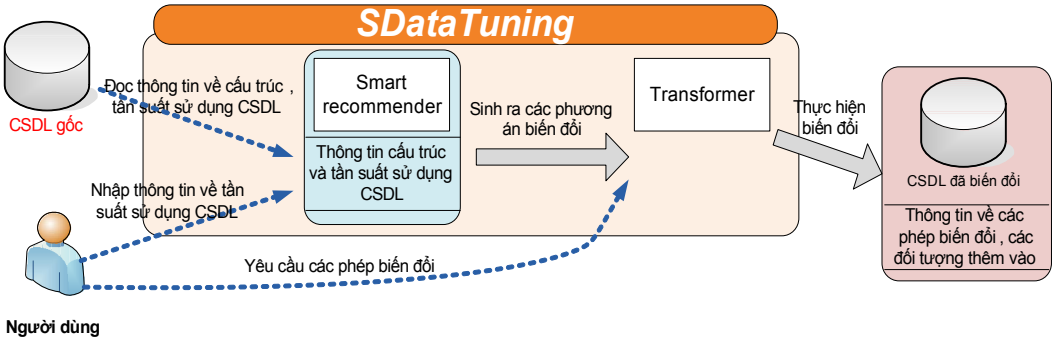
1. SDataTuning - MỘT QUY TRÌNH PHI CHUẨN HÓA CƠ SỞ DỮ LIỆU

Quy trình phi chuẩn hoá CSDL gồm hai bước cơ bản:

1. Lựa chọn phương án biến đổi
- Xác định các phép biến đổi sẽ được thực hiện để phi chuẩn hoá
- Xác định một phương án biến đổi bao gồm các phép biến đổi được lựa chọn để thực hiện nhằm phi chuẩn hoá CSDL.

2. Thực hiện các phép biến đổi theo phương án đã lựa chọn

Trong phần này chúng tôi trình bày cụ thể một quy trình phi chuẩn hoá CSDL nhằm cải thiện tốc độ kết xuất báo cáo được đặt tên là SDataTuning (Hình 1).



Hình 1. Sơ đồ quy trình SDataTuning

1.1. Lựa chọn phương án chuyển đổi

1.1.1. Các phép biến đổi CSDL

Các phép biến đổi được lựa chọn để phi chuẩn hoá CSDL trong SDataTuning bao gồm:
- Phép kết hợp các bảng.
- Phép phân hoạch dọc bảng.
- Phép phân hoạch ngang bảng.
- Phép thêm thuộc tính suy dẫn.

a) *Phép kết hợp các bảng (Collapsing Relations)*

Phép biến đổi này nhằm tạo ra bảng tích của các bảng cần JOIN với các cột phù hợp cho báo cáo. Nhờ thế dữ liệu luôn luôn được sẵn sàng cho báo cáo, không cần thực hiện các phép JOIN tiêu tốn nhiều tài nguyên máy (phép JOIN làm việc với số lượng bản ghi bằng tích số lượng bản ghi các bảng tham gia) làm chậm tốc độ truy xuất. Các cột ở bảng phi chuẩn là vừa đủ cần cho báo cáo, không xuất hiện các cột dư thừa, do đó kích thước bản ghi được tối ưu dẫn đến giảm thiểu thao tác vào ra.

Trường hợp phổ biến nhất của kết hợp là dạng kết hợp hai bảng *một-một*. Ví dụ hai bảng CUSTOMER (CustomerID, CustomerName, Addess) và CUSTOMER_ACCOUNT (CustomerID, AccountBal, MarketSegment) sẽ được kết hợp thành bảng CUSTOMER_DN (CustomerID, CustomerName, Address, AccountBal, MarketSegment). Dạng kết hợp *một-một* không gây ra sự dư thừa dữ liệu. Mỗi bản ghi ở bảng gốc tương ứng với một bản ghi ở bảng phi chuẩn. Đây là dạng ghép bảng ít hạn chế nhất.

Hai bảng có quan hệ dạng *một-nhiều* cũng có thể thực hiện ghép bảng trong trường hợp số bản ghi của bảng phía một là tương đối ít và các bản ghi của cả hai bảng ít có thao tác cập nhật. Xét hai bảng ở dạng chuẩn 3 là CUSTOMER (CustomerID, CustomerName, Addess, Zip) và bảng CITY (Zip, City, State) chúng ta sẽ phi chuẩn tạo ra bảng CUSTOMER_DN (CustomerID, CustomerName, Addess, Zip, City, State). Vì số lượng bản ghi của bảng CITY là ít hơn đáng kể so với bảng CUSTOMER nên thông tin dư thừa (các cột City, State) ở bảng CUSTOMER_DN không lớn. Giả sử rằng các Address của khách hàng ít cập nhật thì các bản ghi của bảng CUSTOMER_DN cũng ít cập nhật.

Dạng ghép bảng thứ ba được xét tới là ghép hai bảng ở quan hệ *nhiều-nhiều*. Dạng quan hệ này được biểu diễn trong cơ sở dữ liệu quan hệ qua một bảng trung gian thứ ba chứa tham chiếu đến khóa của hai bảng đó. Dạng này được sử dụng khi số lượng bản ghi của ít nhất một trong hai bảng không lớn và dữ liệu của chúng ít thay đổi. Ví dụ xét quan hệ hai bảng EMPLOYEE (EmployeeID, EmployeeName, Address) và bảng PROJECT (ProjectID, ProjectName) được thể hiện qua bảng trung gian thứ ba là EMPLOYEE_PROJECT(EmployeeID, ProjectID, WorkingTime). Trong trường hợp này, các bản ghi của bảng PROJECT ít khi được cập nhật và số lượng bản ghi của nó cũng ít. Vì vậy có thể phi chuẩn hóa bảng EMPLOYEE_PROJECT thành bảng EMPLOYEE_PROJECT_NEW(EmployeeID, ProjectID, ProjectName, WorkingTime). Bảng phi chuẩn này chứa đầy đủ thông tin về các dự án, do đó không cần tham chiếu đến bảng PROJECT mỗi khi cần truy vấn thông tin về nhân viên làm việc cho các dự án.

b) *Phép Phân hoạch dọc bảng (Vertical Partitioning)*

Trong phép biến đổi này, một bảng mới được tạo ra từ bảng ban đầu với số bản ghi giống nhau nhưng loại bỏ đi một số cột ít sử dụng. Điều này giúp cho kích thước bản ghi của bảng dùng cho báo cáo được thu gọn, do đó giảm thao tác vào ra đối với phương tiện lưu trữ.

Khi thực hiện tách dọc bảng, đưa các cột ra bảng mới, chúng ta luôn luôn phải mang các cột định danh đi theo để đảm bảo tính xác định và truy ngược dữ liệu.

Ví dụ, xét bảng PRODUCT (Product_ID, Width, Length, Height, Weight, Price, Stock,

Supplier_ID, Warehouse). Nếu hầu hết các truy vấn đối với bảng PRODUCT được chia làm hai loại tương ứng với hai nhóm thuộc tính, nhóm 1 là các thuộc tính mô tả sản phẩm bao gồm (Width, Length, Height, Weight, Price) và nhóm thứ hai gồm các thuộc tính liên quan đến thông tin lưu trữ sàn phẩm (Price, Stock, Supplier_ID, Warehouse) thì khi chúng ta tạo ra các bảng mới PRODUCT_SPECT (Product_ID, Width, Length, Height, Weight) và PRODUCT_INVENTORY (Product_ID, Price, Stock, Supplier_ID, Warehouse), các truy vấn được thực hiện trực tiếp trên mỗi bảng.

c) *Phép phân hoạch ngang bảng (Horizontal Partitioning)*

Phép phân hoạch ngang bảng tạo ra các bảng mới có cấu trúc giống hệt bảng gốc, nhưng dữ liệu là một tập con các bản ghi của bảng gốc và thỏa một điều kiện lọc nào đó.

Phép biến đổi này thường được dùng đối với các bảng có số lượng bản ghi lớn, trong đó nhiều bản ghi mang tính "lịch sử", tức là rất ít khi thay đổi. Nếu chúng ta tách các bản ghi "lịch sử" này ra thành các bảng mới thì rất ít khi phải thực hiện thao tác đồng bộ dữ liệu giữa bảng mới và bảng gốc. Do đó hầu như không ảnh hưởng đến tốc độ cập nhật dữ liệu ở bảng gốc.

Lợi ích của phép biến đổi này là tốc độ thực hiện truy vấn trên bảng thu được sẽ được cải thiện so với trên bảng gốc vì số lượng bản ghi của bảng mới rõ ràng nhỏ hơn so với bảng gốc.

Ví dụ có bảng ban đầu SALE_HISTORY (Sale_ID, Timestamp, Store_ID, Customer_ID, Amount). Hầu hết các bản ghi có Timestamp trong các miền thời gian 1, 2 và 3 (ví dụ trong 3 năm trước) đều hầu như không thay đổi. Chúng ta có thể thực hiện sinh ra các bảng mới như sau:

• SALE_HISTORY_Period1 (Sale_ID, Timestamp, Store_ID, Customer_ID, Amount). Điều kiện tách của bảng này là Timestamp trong miền thời gian 1.

• SALE_HISTORY_Period2 (Sale_ID, Timestamp, Store_ID, Customer_ID, Amount). Điều kiện tách của bảng này là Timestamp trong miền thời gian 2.

• SALE_HISTORY_Period3 (Sale_ID, Timestamp, Store_ID, Customer_ID, Amount). Điều kiện tách của bảng này là Timestamp trong miền thời gian 3.

d) *Phép thêm thuộc tính suy dẫn (Derived Attributes)*

Thực tế trong các báo cáo chúng ta vẫn thường hay phải tổng hợp dữ liệu từ những dữ liệu đã có để đưa ra những thông tin mới. Nếu chúng ta thực hiện việc thêm các bảng chứa sẵn các thông tin tổng hợp đó thì mỗi khi lên báo cáo, hệ thống không cần tính toán lại, chỉ cần lấy từ CSDL, như vậy tốc độ sẽ được cải thiện đáng kể. Thực hiện phép biến đổi này có hai dạng:

• Dạng thứ nhất: Bổ sung một số cột vào bảng đã có. Trong dạng này, thông tin tổng hợp chỉ liên quan đến các cột dữ liệu của cùng một bản ghi. Nếu bảng cũ có dạng $R(c_1, c_2, ...c_n)$ thì bảng mới được tạo ra có dạng $Rnew(c'_1, c'_2, ...c'_m,\ f(c'_1, c'_2, ...c'_m))$ trong đó $c'_1, ...c'_m$ là một trong các $c_1, c_2, ...c_n$; tất nhiên $m \leq n$. Ví dụ bảng PRODUCT_STOCK (ProductID, Quantity, Unit_price), nếu chúng ta thường phải tính thông tin tổng giá trị mỗi sản phẩm

Total_value = Quantity * Unit_price thì có thể tạo thêm một cột cho bảng đó thành bảng PRODUCT_STOCK_NEW (ProductID, Quantity, Unit_price, Total_value).

• Dạng thứ hai: Đưa một số cột của bảng đã có cùng các cột thông tin suy dẫn thành bảng mới. Trong dạng này thông tin tổng hợp liên quan đến một nhóm các bản ghi của một bảng. Chúng ta phải dùng các hàm nhóm gộp trong SQL như SUM, MIN, MAX, AVG, COUNT và các tính toán khác để làm dữ liệu cho báo cáo. Các hàm này khi chạy sẽ khiến cho máy tính phải xử lý trên rất nhiều bản ghi rồi mới tính toán cho ra dữ liệu cần thiết. Đây là một trong những nguyên nhân gây tiêu tốn nhiều thời gian. Trong dạng thứ hai chúng ta tách các cột nhóm gộp và thông tin suy dẫn đưa ra bảng mới. Ví dụ bảng SALE_HISTORY (Sale_ID, Timestamp, Store_ID, Customer_ID, Amount) thường xuyên phải thực hiện truy vấn tổng lượng hàng bán trong 1 kho nào đấy dưới dạng câu lệnh SQL như sau: SELECT StoreID, SUM(Amount) FROM SALE_HISTORY GROUP BY StoreID. Lúc đó ta nên tạo bảng mới tính sẵn giá trị tổng lượng hàng bán theo từng kho TOTAL_SALE_EACH_STORE (StoreID, Total_amount).

1.1.2. Xác định một phương án biến đổi

Xác định một phương án biến đổi bao gồm việc lựa chọn các phép biến đổi cần thực hiện nhằm phi chuẩn hoá CSDL. Căn cứ để lựa chọn phương án biến đổi là dựa vào cấu trúc CSDL, thông tin về tần suất truy vấn, cập nhật các bảng, các cột và thông tin về liên kết các bảng (gọi tắt là tần suất sử dụng CSDL). Cấu trúc CSDL được lấy từ hệ quản trị CSDL. Còn thông tin về tần suất sử dụng CSDL hoặc được lấy từ một agent (tác tử) chạy trên server CSDL có chức năng thống kê hoặc do người dùng trực tiếp xác định.

Từ những thông tin đó, cần lựa chọn phương án biến đổi CSDL một cách hợp lý nhằm làm tăng hiệu suất sử dụng CSDL. Ở đây cần cân nhắc giữa lợi ích tăng tốc truy vấn kết xuất báo cáo và nhược điểm dư thừa dữ liệu dẫn đến cập nhật sửa đổi chậm. Hệ thống có thể đưa ra gợi ý về phương án, hoặc người sử dụng tự lựa chọn phương án.

Trong SDataTuning đã đề xuất thuật toán lựa chọn cho hai loại phép biến đổi là phép kết hợp bảng và phép thêm thuộc tính suy dẫn.

Input: thông tin về cấu trúc và tần suất sử dụng (truy vấn, cập nhật) CSDL.

Để thuận tiện cho việc mô tả các tham số, chúng tôi quy ước đặt tên các biến số tần suất sử dụng CSDL qua bảng dưới. Xét từng cặp bảng T_1 và T_2 thuộc CSDL ban đầu, các tần suất liên quan đến bảng T_1 được đặt tên bắt đầu bằng $x(x_1, x_2, x_3...)$, các tần suất liên quan đến bảng T_2 được đặt tên bắt đầu bằng $y(y_1, y_2, y_3...)$, các tần suất liên quan đến phép kết hợp T_1 JOIN T_2 theo điều kiện α nào đó (ví dụ T_1.Col1 = T_2.Col1) được đặt tên bắt đầu bằng $z(z_1, z_2, z_3...)$. Giả sử T_1 có k cột $c_1, c_2...c_k$.

Đối tượng	Truy vấn	Cập nhật	Thêm mới	Xóa	Có điều kiện lọc	Công thức suy dẫn
Bảng T_1	x_1	x_2	x_3	x_4	x_5	x_6
Bảng T_2	y_1	y_2	y_3	y_4	y_5	y_6
T_1 JOIN T_2 (điều kiện α)	z_1	NA	NA	NA	NA	NA

NA: thông tin không tồn tại.

Output: đưa ra các gợi ý lựa chọn phép biến đổi CSDL.

a) *Điều kiện thực hiện phép kết hợp bảng*

Để xét xem có nên tạo bảng mới là kết quả của phép kết hợp cặp bảng T_1 và T_2 hay không chúng ta cần xét hai tỷ số $z_1/(x_1 + y_1)$ và $z_1/(x_2 + x_3 + x_4 + y_2 + y_3 + y_4)$.

Nếu tỉ số $z_1/(x_1 + y_1)$ đủ lớn (khoảng từ 3 trở lên) chứng tỏ rằng thông tin các bảng T_1 và T_2 luôn luôn đi cùng với nhau trong các truy vấn.

Nếu tỷ số $z_1/(x_2 + x_3 + x_4 + y_2 + y_3 + y_4)$ đủ lớn (khoảng từ 5 trở lên) chứng tỏ rằng số lần thay đổi dữ liệu các bảng T_1 và T_2 là ít so với số lần lấy thông tin tổng hợp từ T_1 và T_2, điều này giúp cho phép kết hợp của chúng ta ít khi phải thực hiện công việc đồng bộ dữ liệu của bảng mới $T_{\text{new}} = T_1$ JOIN T_2, do vậy chi phí cập nhật không đáng kể so với lợi ích đem lại là tăng tốc truy vấn từ bảng T_{new} so với lấy thông tin từ T_1 JOIN T_2.

Nếu hai điều kiện trên thỏa mãn quy trình sẽ đưa ra gợi ý cho người dùng nên kết hợp bảng T_1 với T_2 với điều kiện α.

b) *Điều kiện thực hiện thêm thuộc tính suy dẫn*

Về trực quan, chúng ta thêm các thuộc tính dư thừa khi tần suất các truy vấn sử dụng một công thức nào đó là khá lớn so với tổng tần suất truy vấn của một bảng và lớn hơn khá nhiều so với số thao tác cập nhật dữ liệu.

Vì vậy để thêm cột $c_{k+1} = f(c_1, c_2, c_3...c_k)$ vào bảng T_1 hai điều kiện đưa ra ở đây là:

• Tỷ số $x_6/x_1 \geq \Phi_1$. Ở đây Φ_1 là một hằng số được ước lượng dựa vào khảo sát, trong công cụ SDataTuning, ta chọn $\Phi_1 = 3$.

• Tỷ số $x_6/(x_2 + x_3 + x_4) \geq \Phi_2$. Tương tụ như Φ_1 thì Φ_2 cũng là một hằng số được ước lượng, ta chọn $\Phi_2 = 5$.

1.2. Thực hiện phương án biến đổi

Sau khi đã có phương án biến đổi CSDL, cần tiến hành thực hiện các phép biến đổi thuộc phương án. Kết quả thực hiện toàn bộ quy trình này là một CSDL mới đảm bảo tính "tương thích" với CSDL cũ và thông tin về các phép biến đổi, các đối tượng được thêm vào. Nói tương thích tức là CSDL mới hoàn toàn có thể thay thế cho CSDL cũ mà không phải sửa đổi chương trình. Để có được điều này, khi thực hiện các phép biến đổi thì quy trình chỉ thêm vào CSDL các bảng mới hoặc bổ sung các cột vào các bảng sẵn có. Thực chất các đối tượng dữ liệu mới đó chỉ chứa thông tin dư thừa được suy dẫn từ CSDL cũ. CSDL mới cũng chứa các cơ chế để đảm bảo tính đồng bộ dữ liệu, tức là khi thông tin các đối tượng ở CSDL cũ thay đổi thì các thông tin dư thừa được cập nhật theo một cách nhất quán. Ngoài ra CSDL mới còn chứa metadata (dữ liệu mô tả dữ liệu), đây là dữ liệu mô tả về CSDL mới, đặc biệt là các phép biến đổi, các đối tượng mới. Metadata này được tổ chức dưới dạng XML và lưu trực tiếp vào CSDL mới. XML là một dạng dữ liệu phi cấu trúc độc lập với nền tảng nên rất dễ để các hệ thống khác đọc và xử lý. Với metadata, người làm báo cáo sẽ rất thuận tiện khi thực hiện công việc của họ vì họ sẽ có hiểu biết sâu sắc về CSDL mới.

Như vậy, khi SDataTuning thực hiện chuyển đổi, cần đảm bảo tính nhất quán về ngữ nghĩa giữa dữ liệu mới thêm vào và dữ liệu gốc đối với cả 4 phép biến đổi kể trên.

Thuật toán đảm bảo tính toàn vẹn

Tất cả các phép chuyển đổi đều cần thực hiện trong hai giai đoạn:

- Chuyển (tạo) dữ liệu cho bảng mới khi vừa tạo ra bảng.
- Đồng bộ dữ liệu của bảng mới so với các bảng gốc khi dữ liệu bảng gốc thay đổi.

Vẫn dùng quy ước gọi tên bảng như phần trước, bảng T_1 và T_2 là các bảng cũ của CSDL, bảng T_{new} là bảng được thêm vào. Sau đây, chúng tôi trình bày thuật toán để thực hiện hai điều trên trong mỗi phép chuyển đổi.

a) *Phép kết hợp bảng*

Giả sử chúng ta thực hiện kết hợp hai bảng T_1 và T_2 với quan hệ $T_1.JoinCol_1 = T_2.JoinCol_2$, bảng đích tên là T_{new}.

* *Sinh dữ liệu cho bảng phi chuẩn*

Input: $T_1, T_2, T_1.JoinCol_1 = T_2.JoinCol_2,\ T_{\text{new}}$

Output: Dữ liệu của bảng T_{new} phù hợp với hai hảng T_1 và T_2

```
Begin
    var danhSachBanGhi = truy vấn T1 JOIN T2 ON T1.JoinCol1 = T2.JoinCol2
    Foreach (banGhi in danhSachBanGhi)
    Begin
        INSERT banGhi INTO Tnew
    End
End
```

**Đảm báo nhất quán khi chèn dữ liệu vào bảng* T_1

Input: $T_1, T_2,\ T_1.JoinCol_1 = T_2.JoinCol_2,\ T_{\text{new}}$, các bản ghi được chèn vào bảng T_1 gọi là Inserted

Output: Chèn bản ghi vào bảng T_{new} cho phù hợp

```
Begin
    var danhSachBanGhi = truy vấn Inserted JOIN T2 ON Inserted.JoinCol1 = T2.JoinCol2
    Foreach (banGhi in danhSachBanGhi)
    Begin
        INSERT banGhi INTO Tnew
    End
End
```

**Đảm báo nhất quán khi cập nhật dữ liệu vào bảng* T_1

Input: $T_1, T_2, T_1.JoinCol_1 = T_2.JoinCol_2,\ T_{\text{new}}$, bảng tạm chứa các giá trị bị sửa của bảng T_2 gọi là Deleted, bảng tạm chứa các giá bản ghi của bảng T_2 bị sửa gọi là Updated.

Output: cập nhật các bản ghi của bảng T_{new} sao cho phù với bảng T_1

Begin
Foreach (banGhi in Deleted)
Begin
UPDATE T_{new}
SET các cột của T_1 = giá trị tương ứng trong bảng Inserted
End
End

*Đảm báo nhất quán khi xóa dữ liệu bảng T_1.

Input: $T_1, T_2, T_1.JoinCol_1 = T_2.JoinCol_2$, T_{new}, bảng tạm chứa các giá trị bị xóa của bảng T_2 gọi là Deleted
Output: Xóa các bản ghi T_{new} tương ứng
Begin
Foreach (banGhi in Deleted)
Begin
DELETE FROM T_{new}
WHERE các_cột = các giá trị tương ứng của banGhi
End
End

b) *Phép tách ngang bảng*

* *Sinh dữ liệu cho bảng phi chuẩn*
Input: T_1, T_{new}, điều kiện lọc Condition để lấy T_1 từ T_{new}
Output: Dữ liệu của bảng T_{new} phù hợp với hảng T_1
Begin
Foreach (banGhi in của bảng T_1)
Begin
If(banGhi phù hợp Condition)
INSERT banGhi với các cột phù hợp INTO T_{new}
End
End

*Đảm báo nhất quán khi chèn dữ liệu vào bảng T_1

Input: T_1, T_{new}, bảng tạm Inserted chứa các bản ghi được chèn vào bảng T_1, điều kiện lọc Condition để lấy T_1 từ T_{new}
Output: Chèn dữ liệu vào bảng T_{new} phù hợp với bảng T_1
Begin
Foreach (banGhi in của bảng Inserted)
Begin
If(banGhi phù hợp Condition)
INSERT banGhi với các cột phù hợp INTO T_{new}

End

End

*Đảm báo nhất quán khi cập nhật dữ liệu vào bảng T_1

Input: T_1, T_{new}, bảng tạm Inserted chứa các bản ghi là giá trị mới cập nhật vào bảng T_1, bảng tạm Deleted chứa các bản ghi là giá trị cũ được sửa của bảng T_1, điều kiện lọc Condition để lấy T_1 từ T_{new}

Output: cập nhật các bản ghi của bảng T_{new} sao cho phù với bảng T_1

Begin

Foreach (banGhi in Deleted)

Begin

DELETE FROM T_{new}

WHERE các cột tương ứng của T_{new} = giá trị tương ứng trong bản Inserted

End

Foreach (banGhi in Inserted)

Begin

If(banGhi phù hợp Condition)

INSERT banGhi với các cột phù hợp INTO T_{new}

End

End

*Đảm báo nhất quán khi xóa dữ liệu bảng T_1

Input: T_1, T_{new}, bảng tạm Deleted chứa các bản ghi bị xóa khỏi bảng T_1

Output: Xóa các bản ghi T_{new} tương ứng

Begin

Foreach (banGhi in Deleted)

Begin

DELETE FROM T_{new}

WHERE các cột tương ứng của T_{new} = giá trị tương ứng trong bản Inserted

End

End

c) *Phép tách dọc bảng*

* *Sinh dữ liệu cho bảng phi chuẩn*

Input: T_1, T_{new}

Output: Dữ liệu của bảng T_{new} phù hợp với hảng T_1

Begin

Foreach (banGhi in của bảng T_1)

Begin

INSERT banGhi với các cột phù hợp INTO T_{new}

End

End

*Đảm báo nhất quán khi chèn dữ liệu vào bảng T_1

Input: T_1, T_{new}, bảng tạm Inserted chứa các bản ghi được chèn vào bảng T_1
Output: Chèn dữ liệu vào bảng T_{new} phù hợp với bảng T_1
Begin
    Foreach (banGhi in của bảng Inserted)
    Begin
        INSERT banGhi với các cột phù hợp INTO T_{new}
    End
End

*Đảm báo nhất quán khi cập nhật dữ liệu vào bảng T_1

Input: T_1, T_{new}, bảng tạm Inserted chứa các bản ghi là giá trị mới cập nhật vào bảng T_1, bảng tạm Deleted chứa các bản ghi là giá trị cũ được sửa của bảng T_1
Output: cập nhật các bản ghi của bảng T_{new} sao cho phù với bảng T_1
Begin
    Foreach (banGhi in Deleted)
    Begin
        UPDATE T_{new}
        SET các cột tương ứng của T_1 = giá trị tương ứng trong bản Inserted
    End
End

*Đảm báo nhất quán khi xóa dữ liệu bảng T_1

Input: T_1, T_{new}, bảng tạm Deleted chứa các bản ghi bị xóa khỏi bảng T_1
Output: Xóa các bản ghi T_{new} tương ứng
Begin
    Foreach (banGhi in Deleted)
    Begin
        DELETE FROM T_{new}
        WHERE các cột = các giá trị của banGhi
    End
End

d) *Phép thêm thuộc tính dư thừa vào bảng cũ*

* *Sinh dữ liệu cho bảng phi chuẩn*

Input: T_1, Các cột mới và công thức suy dẫn
Output: Dữ liệu của các cột mới trong bảng T_1 phù hợp với công thức
Begin
    Foreach (banGhi in của bảng T_1)
    Begin

UPDATE T_1
SET Các cột mới = Các giá trị suy dẫn
WHERE Giá trị các cột cũ = Giá trị các cột cũ của banGhi
End
End

**Đảm báo nhất quán khi chèn hoặc cập nhật dữ liệu vào bảng T_1*

Input: T_1, bảng tạm Inserted chứa các dòng được chèn vào. Các cột mới và công thức suy dẫn
Output: Cột mới có dữ liệu phù hợp với giá trị chèn vào
Begin
Foreach (banGhi in của bảng Inserted)
Begin
UPDATE T_1
SET Các cột mới = Các giá trị suy dẫn
WHERE Giá trị các cột cũ = Giá trị các cột cũ của banGhi
End
End

e) *Tạo bảng mới từ một số cột của bảng cũ, thêm thuộc tính suy dẫn*

Ở dạng biến đổi này, một bảng mới được tạo ra, các cột của nó là một số cột từ bảng cũ (gọi là cột cũ) và các cột suy dẫn từ các cột cũ đó.

* *Sinh dữ liệu cho bảng phi chuẩn*

Input: T_1, T_{new}, bảng tạm Inserted chứa các dòng được chèn vào. Các cột mới và công th ức suy dẫn
Output: Dữ liệu của bảng T_{new} phù hợp với hảng T_1
Begin
Foreach (banGhi in của bảng T_1)
Begin
INSERT banGhi, các giá trị suy dẫn từ công thức INTO T_{new}
End
End

**Đảm báo nhất quán khi chèn dữ liệu vào bảng T_1*

Input: T_1, T_{new}, bảng tạm Inserted chứa các dòng được chèn vào. Các cột mới và công thức suy dẫn
Output: Chèn dữ liệu vào bảng T_{new} phù hợp với bảng T_1
Begin
Foreach (banGhi in của bảng T_1)
Begin
INSERT banGhi, các giá trị suy diễn từ công thức INTO T_{new}

End

End

*Đảm báo nhất quán khi xóa dữ liệu bảng T_1

Input: T_1, T_{new}, bảng tạm Deleted chứa các bản ghi bị xóa khỏi bảng T_1

Output: Xóa các bản ghi T_{new} tương ứng

Begin

Foreach (banGhi in Deleted)

Begin

DELETE FROM T_{new}

WHERE Các cột tương ứng của T_{new} = Giá trị tương ứng trong bảng Inserted

End

End

*Đảm báo nhất quán khi cập nhật dữ liệu vào bảng T_1

Input: T_1, T_{new}, bảng tạm Inserted chứa bản ghi là giá trị mới cập nhật vào bảng T_1, bảng tạm Deleted chứa bản ghi là giá trị cũ được sửa của bảng T_1

Output: cập nhật các bản ghi của bảng T_{new} sao cho phù với bảng T_1

Begin

UPDATE T_{new}

SET Các cột tương ứng của T_1 = Giá trị tương ứng trong bảng Inserted,

các cột suy diễn = công thức tính toán,

WHERE Các cột cũ = Giá trị các cột cũ trong bảng Inserted

End

f) *Tạo bảng mới, thêm thuộc tính tổng hợp*

Trong dạng biến đổi này, một số cột của bảng cũ được chọn làm điều kiện nhóm gộp, từ đó các bản ghi của bảng cũ được phân thành các tập hợp, thực hiện tính toán tổng hợp các giá trị đại diện cho tập hợp đó, sau đó đặt giá trị tổng hợp cùng các cột nhóm gộp thành một bảng mới.

* *Sinh dữ liệu cho bảng phi chuẩn*

Input: T_1, T_{new}, điều_kiện_nhóm_gộp, công_thức_tổng_hợp

Output: Dữ liệu của bảng T_{new} phù hợp với hảng T_1

Begin

Foreach (giaTriNhomGop của bảng T_1 theo điều_kiện_nhóm_gộp)

Begin

INSERT giaTriNhomGop, giaTriTongHop = công_thức_tổng_hợp INTO T_{new}

End

End

*Đảm báo nhất quán khi chèn dữ liệu vào bảng T_1

Input: T_1, T_{new}, bảng tạm Deleted chứa bản ghi bị xóa, điều_kiện_nhóm_gộp, công_thức_tổng_hợp

Output: Chèn dữ liệu vào bảng T_{new} phù hợp với bảng T_1

Begin

DELETE FROM T_{new}

WHERE Các cột cũ = Các cột tương ứng trong Inserted

INSERT giaTriNhomGop của Inserted, giaTriTongHop = công_thức_tổng_hợp

INTO T_{new}

End

**Đảm báo nhất quán khi xóa dữ liệu bảng T_1*

Input: T_1, T_{new}, bảng tạm Deleted chứa bản ghi bị xóa, điều_kiện_nhóm_gộp, công_thức_tổng_hợp

Output: Chèn dữ liệu vào bảng T_{new} phù hợp với bảng T_1

Begin

DELETE FROM T_{new}

WHERE Các cột cũ = Các cột tương ứng trong Deleted

INSERT giaTriNhomGop của Deleted,

giaTriTongHop = công_thức_tổng_hợp INTO T_{new}

End

**Đảm báo nhất quán khi cập nhật dữ liệu vào bảng T_1*

Input: T_1, T_{new}, bảng tạm Inserted chứa các dòng được chèn vào, Bảng tạm Deleted chứa các dòng được chèn vào, điều_kiện_nhóm_gộp, công_thức_tổng_hợp

Output: Chèn dữ liệu vào bảng T_{new} phù hợp với bảng T_1

Begin

DELETE FROM T_{new}

WHERE Các cột cũ = Các cột tương ứng trong Deleted

IF (Giá trị các cột cũ Inserted ¡¿ Giá trị các cột cũ tương ứng Deleted)

Begin

DELETE FROM T_{new}

WHERE Các cột cũ = Các cột tương ứng trong Inserted

End

INSERT giaTriNhomGop của Deleted,

giaTriTongHop = công_thức_tổng_hợp INTO T_{new}

IF (Giá trị các cột cũ Inserted ¡¿ Giá trị các cột cũ tương ứng Deleted)

Begin

INSERT giaTriNhomGop của Inserted,

giaTriTongHop = công_thức_tổng_hợp INTO T_{new}

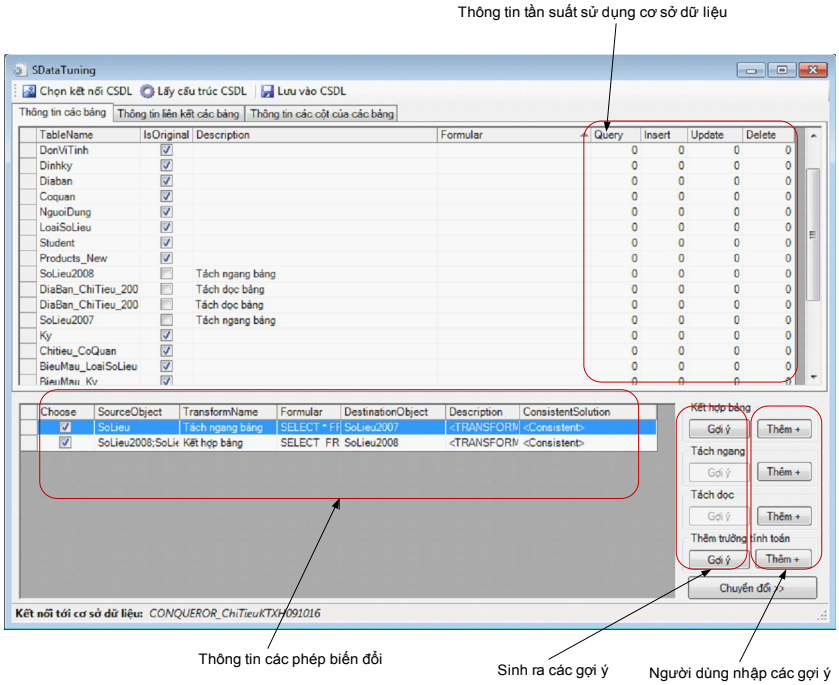
End

End

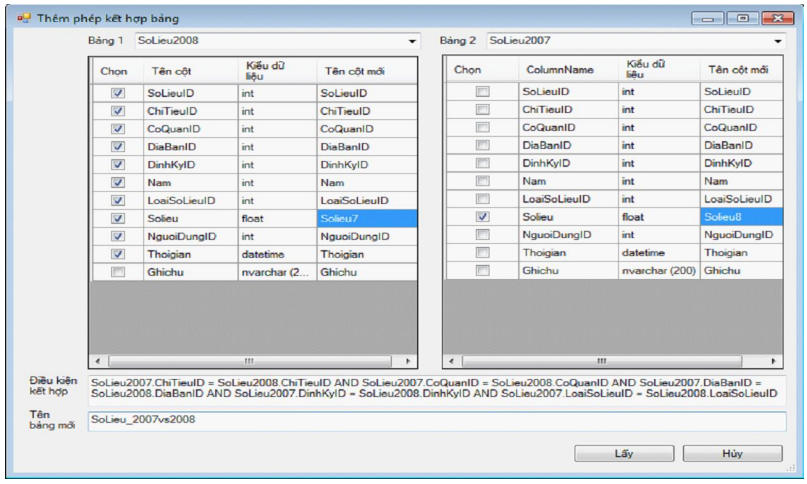
2. PHẦN MỀM HỖ TRỢ THỰC HIỆN SDataTuning

Một phần mềm hỗ trợ thực hiện SDataTuning đã được xây dựng. Ban đầu hoạt động, phần mềm sẽ lấy thông tin cấu trúc CSDL khi người dùng ấn nút "Chọn kết nối" và "Lấy cấu trúc CSDL".

Người dùng có thể nhập thông tin tần suất sử dụng CSDL vào các grid trong các tab "Thông tin các bảng", "Thông tin liên kết các bảng", "Thông tin các cột của các bảng". Sau khi nhập xong, để lưu lại thông tin về CSDL hiện tại, người dùng ấn nút "Lưu vào CSDL".

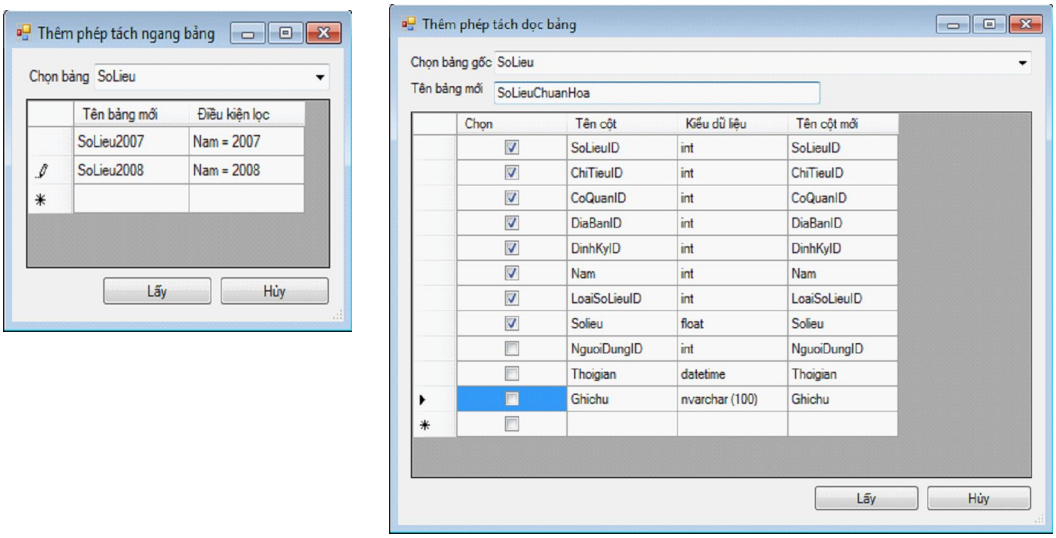


Hình 2. Giao diện phần mềm SDataTuning

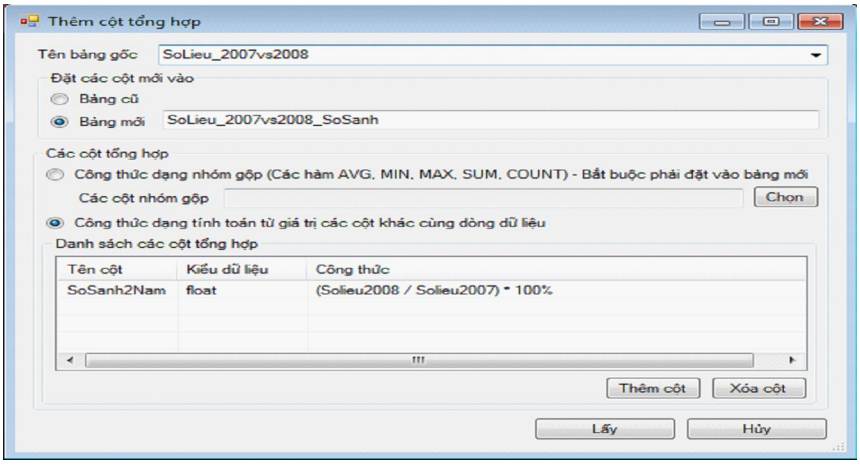


Hình 3. Giao diện nhập phép kết hợp bảng

Để sinh ra các gợi ý về chuyển đổi, người dùng ấn vào nút gợi ý trong các phép chuyển đổi tương ứng, hệ thống sẽ sinh ra phương án đề xuất chuyển đổi và hiển thị ở grid "Thông tin các phép biến đổi" (Hình 2).



Hình 4. Giao diện nhập phép Tách ngang và Tách dọc bảng



Hình 5. Giao diện nhập phép Thêm cột tổng hợp

Để tạo các phép biến đổi tùy chọn, người dùng ấn vào các nút "Thêm" ở từng phương pháp biến đổi, chương trình sẽ hiện ra các giao diện nhập phép biến đổi: kết hợp bảng (Hình 3), tách ngang (Hình 4), tách dọc (Hình 4) và thêm cột tổng hợp (Hình 5).

Sau khi đã chọn xong các phép biến đổi, ấn vào nút "Chuyển đổi" để SDataTuning thực hiện biến đổi CSDL. Sau khi biến đổi, các đối tượng mới của CSDL (bảng, cột) sẽ được hiển thị lên grid thông tin CSDL (Hình 2).

Tiến hành thử nghiệm các phép biến đổi đối với một số CSDL, kết quả cho thấy:

+ Đối với chức năng sinh ra các phương án biến đổi, nói chung công cụ đưa ra các gợi ý

chưa thực sự mang tính thuyết phục cao, chỉ duy nhất phép kết hợp bảng thì công cụ sinh ra được các gợi ý khá hợp lý.

+ Đối với chức năng thực hiện các phép biến đổi, công cụ đã thực hiện đúng như yêu cầu. Các dữ liệu dư thừa được thêm vào luôn luôn đảm bao phù hợp với dư liệu gốc. CSDL sau khi được biến đổi đã phục vụ đắc lực cho việc làm báo cáo, thể hiện được hai ưu điểm: cải thiện tốc độ xuất báo cáo và tăng tính dễ hiểu, dễ thao tác, dễ làm báo cáo.

KẾT LUẬN

Bài báo đã mô tả quy trình SDataTuning và phần mềm hỗ trợ thực hiện phi chuẩn hóa CSDL nhằm cải thiện tốc độ kết xuất báo cáo. Chúng tôi sẽ tiếp tục nghiên cứu thực hiện thêm các phép biến đổi khác, tối ưu các gợi ý biến đổi và tiến hành đưa ra các đánh giá hiệu năng các phép biến đổi để người dùng có thể dễ dàng hơn đưa ra quyết định thực hiện biến đổi. Ngoài ra, chúng tôi cũng sẽ phát triển module agent để thu thập thông tin về tần suất sử dụng CSDL làm thông tin đầu vào cho công cụ SDataTuning, khi đó toàn bộ quy trình cải tiến CSDL do chúng tôi đề xuất ở trên sẽ mang tính tự động cao hơn.

TÀI LIỆU THAM KHẢO

[1] Seung Kyoon Shin, and G. Lawrence Sanders, Denormalization strategies for data retrieval from data warehouses, *Decision Support Systems* **42** (1) (2006) (ISSN: 0167-9236) 267–282.

[2] Craig S. Mullins, Denormalization guidelines, The data administration newsletter, http://www.tdan.com.

[3] Microsoft Corp, *MSDN Library DVD*, 2008.

[4] Đặng Hữu Đạo, Nguyễn Minh Tuấn, Lê Huy Thập, Hoàng Đỗ Thanh Tùng, Nghiên cứu xác định các dạng báo cáo tổng hợp linh hoạt, *Tạp chí Tin học & Điều khiển học* **25** (2) (2009) p.188.

Nhận bài ngày 10 - 11 - 2009