BỘ GIÁO DỤC VÀ ĐÀO TẠO
ĐẠI HỌC ĐÀ NẴNG

TRẦN NGỌC HIẾN LÊ

XÂY DỰNG CHƯƠNG TRÌNH SINH TỰ ĐỘNG MÃ CÁC TRIGGER THỰC HIỆN CẬP NHẬT GIA TĂNG CÁC BẢNG KHUNG NHÌN THỰC NỐI NGOÀI

Chuyên ngành : Khoa học máy tính
Mã số : 60.48.01.01

TÓM TẮT LUẬN VĂN THẠC SĨ KỸ THUẬT

Đà Nẵng - Năm 2015

Chương trình được hoàn thành tại

ĐẠI HỌC ĐÀ NẴNG

Người hướng dẫn khoa học: TS. NGUYỄN TRẦN QUỐC VINH

Phản biện 1: TS. Huỳnh Công Pháp

Phản biện 2: GS.TS. Nguyễn Thanh Thủy

Luận văn đã được bảo vệ trước Hội đồng chấm Luận văn tốt nghiệp Thạc sĩ Kỹ thuật họp tại Đại học Đà Nẵng vào ngày 18 tháng 7 năm 2015

Có thể tìm hiểu luận văn tại:

– Trung tâm Thông tin - Học liệu, Đại học Đà Nẵng

– Trung tâm Học liệu, Đại học Đà Nẵng

MỞ ĐẦU

1. Tính cấp thiết của đề tài

Khung nhìn thực là một đối tượng cơ sở dữ liệu chứa các kết quả của một truy vấn, giúp trả lời các truy vấn nhanh chóng thay vì lấy dữ liệu từ các bảng gốc và xử lý. Trong kho dữ liệu, dữ liệu biên niên sử ở các hệ thống ngân hàng, bán lẻ và thanh toán; trong các ứng dụng ảo hóa; trong các hệ thống di động; kiểm soát các ràng buộc toàn vẹn; tối ưu hoá truy vấn, hiệu quả việc sử dụng KNT được thể hiện rõ hơn.

Một ví dụ điển hình về tính hiệu quả của việc ứng dụng KNT. Một tập đoàn có nhiều đại diện tại nhiều vùng thuộc nhiều quốc gia cung cấp cho nhiều khách hàng khác nhau một số lượng lớn các sản phẩm. Như vậy, CSDL trung tâm của tập đoàn này có thể chứa hàng triệu hoặc nhiều hơn các dòng dữ liệu về chi tiết bán hàng. Bây giờ, người ta cần thống kê số lượng sản phẩm được bán cũng như tổng doanh thu cho từng loại sản phẩm tại mỗi vùng theo quốc gia. Truy vấn được thực thi và kết quả được trả lại sau một khoảng một thời gian T_1 nào đó. Kết quả này được lưu lại trong một bảng – KNT bao gồm 200 dòng dữ liệu. Sau này, mỗi khi xuất hiện truy vấn đó, thay vì thực thi lại từ đầu bằng việc quét và xử lý hàng triệu dòng dữ liệu, HQT CSDL đọc bảng KNT chứa chỉ 200 dòng dữ liệu và trả lại kết quả trong khoảng thời gian T_2 (thường rất nhỏ so với T_1), thường là vài ms. Thậm chí, KNT có thể được dùng để trả lời các truy vấn tương tự nhưng cho trường hợp cả thế giới, hoặc một vài vùng nào đó, hoặc trường hợp chỉ cần tính hoặc doanh thu hoặc số lượng sản phẩm. Tính năng này được gọi là viết lại truy vấn (query rewrite) [4].

Tuy nhiên, KNT không cho phép nâng cao năng suất trong tất cả các trường hợp, hiệu quả ứng dụng chúng có thể giảm đi rõ rệt nếu thường xuyên xảy ra thay đổi dữ liệu trong các bảng gốc sử dụng để tạo KNT (hay KNT sử dụng).

Khung nhìn thực được ứng dụng từ những năm 1980, tuy nhiên đến nay chỉ có 3 HQTCSDL hàng đầu thế giới là Oracle, IBM DB2 và MS SQL Server triển khai thành công. Còn đối với những HQTCSDL khác, nhất là HQTCSDL mã nguồn mở như PostgeSQL thì khó khăn được xác định là ở bước thực hiện cập nhật gia tăng cho KNT. Trước đó đã có những nghiên cứu về việc sử dụng trigger mã nguồn PL/pgSQL để cập nhật gia tăng đồng bộ KNT trong PostgreSQL, nhưng chỉ dừng lại ở KNT kiểu SPJ (Select – Project - Join), là KNT dựa trên truy vấn chỉ chứa phép chọn, phép chiếu và phép nối trong, không bao gồm các phép toán thống kê như SUM, COUNT, AVG, MIN, MAX,… Tuy nhiên, mã nguồn PL/pgSQL được đánh giá là có hiệu năng chưa cao ở một số xử lý (ví dụ như xử lý vòng lặp).

PostgreSQL được viết hoàn toàn bằng ngôn ngữ C, có các cấu trúc dữ liệu trong C tương ứng với các kiểu dữ liệu trong SQL và một giao diện lập trình giúp cho hàm trigger có thể thực hiện truy vấn đến CSDL được dễ dàng. Vì vậy, việc thực thi phần mở rộng PostgreSQL thực sự nhanh hơn khó có thể đạt được bằng cách sử dụng bất cứ ngôn ngữ gì khác ngoài C. Viết mã bằng ngôn ngữ C có thể không phải là cách nhanh nhất của việc thực thi các tính năng, nhưng việc thực hiện các hàm sẽ không bị ảnh hưởng nhiều mà các ngôn ngữ lập trình khác gây ra.

Xuất phát từ những lý do trên, tôi đề xuất chọn đề tài luận văn tốt nghiệp thạc sỹ ngành khoa học máy tính: *"Xây dựng chương*

trình sinh tự động mã các trigger thực hiện cập nhật gia tăng các bảng khung nhìn thực nối ngoài".

2. Mục tiêu nghiên cứu

Mục tiêu chung của đề tài là: Xây dựng chương trình sinh tự động mã các trigger thực hiện cập nhật gia tăng các bảng khung nhìn thực nối ngoài nhằm làm tăng tốc độ thực thi các truy vấn.

Để đạt được mục tiêu chung đó, tác giả nghiên cứu các mục tiêu cụ thể như sau:

- Nghiên cứu tổng quan về KNT.
- Nghiên cứu cơ sở lý thuyết về cập nhật gia tăng, đồng bộ KNT nối ngoài.
- Nghiên cứu tổng quan về trigger trên C trong PostgreSQL.
- Xây dựng thuật toán cập nhật gia tăng KNT nối ngoài.
- Xây dựng module sinh mã tự động các trigger.

3. Đối tượng và phạm vi nghiên cứu

3.1. Đối tượng nghiên cứu

- Khung nhìn thực
- Trigger

3.2. Phạm vi nghiên cứu

- Cập nhật gia tăng, đồng bộ khung nhìn thực nối ngoài.
- Hệ quản trị cơ sở dữ liệu PostgreSQL.
- Trigger trong ngôn ngữ C.

4. Phương pháp nghiên cứu

4.1. Phương pháp lý thuyết

Thu thập, chọn lọc, phân loại, ghi chú và nghiên cứu các tài liệu (sách, bài báo, luận văn, trang web) có liên quan đến khung nhìn thực, HQTCSDL PostgreSQL, cập nhật gia tăng KNT, đồng bộ KNT, sinh mã trigger trong ngôn ngữ C.

4.2. Phương pháp thực nghiệm

Dựa trên lý thuyết đã nghiên cứu, tiến hành xây dựng chương trình sinh tự động mã các trigger thực hiện cập nhật gia tăng khung nhìn thực nối ngoài trong hệ cơ sở dữ liệu PostgreSQL; thử nghiệm trên máy đơn và đánh giá tốc độ cập nhật dữ liệu trên các BG có trigger cập nhật KNT.

5. Ý nghĩa khoa học và thực tiễn của đề tài

5.1. Ý nghĩa khoa học

- Đề xuất thuật toán cập nhật gia tăng KNT nối ngoài.

- Đề xuất thuật toán sinh tự động mã nguồn các trigger trong ngôn ngữ C phục vụ cập nhật gia tăng các bảng khung nhìn thực nối ngoài trong hệ quản trị cơ sở dữ liệu PostgreSQL.

5.2. Ý nghĩa thực tiễn

Áp dụng chương trình vào cơ sở dữ liệu PostgreSQL nhằm làm tăng tốc độ thực thi các truy vấn dữ liệu mà vẫn đảm bảo ràng buộc toàn vẹn dữ liệu. Chương trình sinh tự động mã trigger C thực hiện CNGT KNT có tính ứng dụng cao, giúp tiết kiệm thời gian và công sức của người quản trị CSDL khi làm việc với HQTCSDL PostgreSQL cũng như các HQTCSDL mã nguồn mở có hỗ trợ trigger trong ngôn ngữ C. Chương trình sinh tự động mã nguồn các trigger có thể hỗ trợ các lập trình viên sinh mã trigger. Lập trình viên chỉ cần điều chỉnh trigger được sinh ra theo ý muốn thay vì phải lập trình từ đầu.

6. Bố cục đề tài

Ngoài phần mở đầu và kết luận, cấu trúc nội dung của luận văn bao gồm 3 chương:

Chương 1: Tổng quan nghiên cứu, chương này giới thiệu tổng quan về KNT; cập nhật gia tăng, đồng bộ KNT. Trình bày các vấn đề liên quan đến trigger trên C trong PostgreSQL.

Chương 2: Cập nhật gia tăng khung nhìn thực nối ngoài, chương này trình bày thuật toán cập nhật gia tăng KNT nối ngoài; xây dựng thuật toán sinh tự động mã nguồn các trigger.

Chương 3: Xây dựng chương trình sinh tự động mã nguồn các trigger thực hiện cập nhật gia tăng khung nhìn thực nối ngoài, chương này trình bày giao diện chương trình đã được xây dựng; đánh giá kết quả đạt được.

CHƯƠNG 1
TỔNG QUAN NGHIÊN CỨU

1.1. TỔNG QUAN VỀ KHUNG NHÌN THỰC

1.1.1. Giới thiệu chung

Một khung nhìn có thể được cụ thể hóa bằng cách lưu trữ các bộ dữ liệu của các khung nhìn trong cơ sở dữ liệu, được gọi là ***khung nhìn thực***. KNT là hiện thân tự nhiên của ý tưởng tính toán lại và bộ nhớ đệm trong cơ sở dữ liệu. Thay vì tính toán một truy vấn từ đầu từ dữ liệu cơ bản, hệ thống cơ sở dữ liệu có thể sử dụng kết quả đã được tính toán, lưu trữ và duy trì.

Giống như một bộ nhớ cache (bộ nhớ đệm – nơi lưu trữ các dữ liệu nằm chờ các ứng dụng hay phần cứng xử lý), một KNT cung cấp truy cập dữ liệu nhanh; sự khác biệt tốc độ có thể là rất quan trọng trong các ứng dụng mà tốc độ truy vấn cao và khung nhìn rất phức tạp rằng nó không thể tính toán lại khung nhìn cho mỗi lần truy vấn. KNT là hữu ích trong các ứng dụng như kho dữ liệu, máy chủ sao chép, biên niên sử hoặc các hệ thống ghi dữ liệu, trực quan dữ liệu và hệ thống điện thoại di động. Kiểm tra tính ràng buộc toàn vẹn và tối ưu hóa truy vấn cũng có thể được lợi ích từ KNT [5, tr.1].

Tuy nhiên, việc sử dụng KNT cũng có những nhược điểm; đó là: KNT làm tiêu tốn không gian lưu trữ và phải được cập nhật khi các bảng chi tiết cơ bản được sửa đổi.

1.1.2. Phân loại khung nhìn thực

a. KNT bản sao (snapshot materialized view)

b. KNT hăm hở (eager materialized view)

c. KNT rất lười (very lazy materialized view)

d. KNT lười (lazy materialized view)

1.2. TỔNG QUAN VỀ CẬP NHẬT GIA TĂNG, ĐỒNG BỘ KHUNG NHÌN THỰC

1.2.1. Tổng quan về cập nhật gia tăng

Có ba phương pháp cập nhật KNT, đó là hoàn toàn (COMPLETE), gia tăng (FAST hay còn gọi là INCREMENTAL) và ép buộc (FORCE)

Để duy trì các bảng KNT trong trạng thái thực tiễn, cần phải cập nhật chúng mỗi khi có sự thay đổi dữ liệu trong các bảng gốc. Phụ thuộc vào thời hạn đưa các thay đổi vào các bảng KNT, các cơ chế cập nhật được phân ra đồng bộ và không đồng bộ.

1.2.2. Đồng bộ khung nhìn thực

Cập nhật đồng bộ được thực thi không chậm trễ ngay khi có thay đổi dữ liệu trong bảng gốc như một phần của giao tác thực hiện thay đổi đó. Ngược lại, cập nhật không đồng bộ được thực hiện vào một thời điểm nào đó sau khi các giao tác sửa đổi dữ liệu trong bảng gốc đã được cố định. Cập nhật không đồng bộ khai thác thời gian trì hoãn cho phép trong cập nhật dữ liệu được xác định bởi công nghệ xử lý dữ liệu trong các hệ thống thông tin và ý nghĩa của các bài toán giải quyết trong đó [3].

1.3. CẬP NHẬT KHUNG NHÌN THỰC BẰNG TRIGGER TRÊN C TRONG POSTGRESQL

1.3.1. Khái niệm về trigger

Trigger là một thủ tục đặc biệt mà việc thực thi của nó tự động khi có sự kiện xảy ra, các sự kiện gọi thủ tục đặc biệt này được định nghĩa trong câu lệnh, thông thường được thực hiện với các sự kiện liên quan đến Insert, Update, Delete dữ liệu. Trigger được sử dụng trong việc bảo đảm toàn vẹn dữ liệu theo quy tắc xác định, được quản lý theo bảng dữ liệu hoặc khung nhìn.

1.3.2. Thủ tục tạo trigger bằng ngôn ngữ C trong HQT CSDL PostgreSQL

1.3.3. Ví dụ tạo trigger trên C trong PostgreSQL

1.4. MỞ RỘNG POSTGRESQL VỚI HÀM TÙY CHỌN

1.4.1. Hàm ngôn ngữ truy vấn

1.4.2. Hàm sử dụng ngôn ngữ lập trình C

1.5. TIỂU KẾT CHƯƠNG 1

Trong phần này, chúng ta đã tập trung trình bày tổng quan về các vấn đề về khung nhìn thực, phương pháp tạo khung nhìn thực và phân loại theo cấu trúc các loại khung nhìn thực trong các HQT CSDL. Ngoài ra, chúng tôi cũng đã trình bày tổng quan về cập nhật gia tăng trong các HQT CSDL và khái niệm về đồng bộ khung nhìn thực. Cuối chương 1, chúng tôi cũng đã đi nghiên cứu các vấn đề về trigger, phương pháp tạo trigger trên C trong PostgreSQL và đã trình bày một ví dụ bằng chính ngôn ngữ C cho phép tạo trigger. Phần tiếp theo của luận văn này, chúng tôi sẽ đi nghiên cứu các giải thuật cập nhật gia tăng khung nhìn thực nối ngoài.

CHƯƠNG 2
CẬP NHẬT GIA TĂNG KHUNG NHÌN THỰC NỐI NGOÀI VÀ SINH MÃ TỰ ĐỘNG TRIGGER

2.1. THUẬT TOÁN CẬP NHẬT GIA TĂNG KHUNG NHÌN THỰC NỐI NGOÀI

Truy vấn dữ liệu chứa nối ngoài là phổ biến trong các ứng dụng kho dữ liệu. Các KNT nối ngoài có thể tăng tốc độ rất nhanh nhiều truy vấn nhưng hầu hết các hệ thống cơ sở dữ liệu không cho phép nối ngoài trong KNT. Đó là bởi vì KNT nối ngoài không thể được duy trì một cách hiệu quả khi các bảng cơ bản được cập nhật. Trong luận văn này, chúng tôi sẽ chỉ ra cách để duy trì hiệu quả các KNT nối ngoài nói chung. Các ràng buộc khóa ngoài được khai thác để giảm thiểu tổng chi phí duy trì.

2.1.1. Tổng quan về KNT nối ngoài

KNT có thể tăng tốc độ xử lý truy vấn rất nhiều, nhưng để nhận ra lợi ích thì hai bài toán con phải được giải quyết: so khớp khung nhìn và cập nhật gia tăng khung nhìn. Mục tiêu của so khớp khung nhìn là để xác định, tại thời điểm tối ưu hóa, không biết và làm thế nào một phần hoặc tất cả của một truy vấn có thể được tính từ một khung nhìn. Cập nhật gia tăng khung nhìn là cần thiết để mang lại hiệu quả một khung nhìn cập nhật khi các bảng cơ sở được cập nhật.

Cập nhật có thể được chia thành hai bước: tính toán và áp dụng *delta chính* và *delta thứ cấp*. Bước đầu tiên rất giống với việc duy trì khung nhìn nối trong. Bước thứ hai là một bước "làm sạch" và luận văn này cho thấy làm thế nào để thực hiện bước này một

cách hiệu quả. Luận văn cũng mô tả cách các ràng buộc khóa ngoài có thể được khai thác để giảm thiểu chi phí duy trì.

2.1.2. Cơ sở lý thuyết KNT nối ngoài

a. Một số định nghĩa và ký hiệu

b. Dạng thức thông thường kết nối – phân ly

c. Biểu đồ gộp

d. Cơ chế đóng góp một số hạng kiểu mạng lưới

2.1.3. Phương pháp duy trì khung nhìn thực

a. Các số hạng bị ảnh hưởng bởi một cập nhật

b. Phương pháp duy trì KNT

Giả sử bảng T đã được cập nhật và chúng tôi cần duy trì một khung nhìn V tham chiếu đến T. Đầu tiên chúng ta tính toán đồ thị duy trì và phân loại các số hạng mà bị ảnh hưởng trực tiếp, bị ảnh hưởng gián tiếp và không bị ảnh hưởng. Không mất tính tổng quát, giả định rằng khung nhìn có *n* số hạng, trong đó các số hạng *1, 2, · · · , k* bị ảnh hưởng trực tiếp, các số hạng $k + 1 , k + 2 , \cdots , k + m$ bị ảnh hưởng gián tiếp , và các số hạng $k + m + 1, k + m + 2 , \cdots , n$ không bị ảnh hưởng. Sau đó chúng ta có thể viết lại biểu thức khung nhìn trong dạng thức sau:

$$V = V^D \uplus V^I \uplus V^U ,$$

trong đó

$$V^D = \uplus_{i=1}^{k} D_i , \; V^I = \uplus_{i=k+1}^{k+m} D_i , \; V^U = \uplus_{i=k+m+1}^{n} D_i .$$

Từ dạng thức này của biểu thức, rõ ràng là để cập nhật khung nhìn chúng ta cần tính toán hai biểu thức delta.

$$\Delta V^D = \uplus_{i=1}^{k} \Delta D_i , \; \Delta V^I = \uplus_{i=k+1}^{k+m} \Delta D_i ,$$

Chúng ta gọi ΔV^D là *delta chính* và ΔV^I là *delta thứ cấp*.

Tóm lại, duy trì một khung nhìn V sau khi cập nhật một trong những bảng cơ sở cơ bản của nó được thực hiện theo hai bước:

- Nếu có các số hạng ảnh hưởng trực tiếp, tính delta chính ΔV^D và áp dụng nó đến khung nhìn.
- Nếu có các số hạng bị ảnh hưởng gián tiếp, tính delta thứ cấp ΔV^I và áp dụng nó đến khung nhìn.

Nếu cập nhật là một việc chèn (xóa), delta chính được chèn vào (xóa từ) khung nhìn và delta thứ cấp bị xóa từ (đưa vào) khung nhìn. Trong phần tiếp theo, chúng tôi mô tả cách hiệu quả để tính toán delta chính và delta thứ cấp, tương ứng.

2.1.4. Phương pháp tính toán delta chính

Thuật toán: Xây dựng biểu thức ΔV^D

Đầu vào: Biểu thức khung nhìn V ban đầu, bảng cập nhật T.

Đầu ra: Biểu thức để tính toán ΔV^D .

1. Xét kỹ toàn bộ cây toán tử cho V dọc theo đường đi từ T đến gốc. Trên bất kỳ toán tử kết nối gặp phải, áp dụng quy tắc giao hoán để đảm bảo rằng các đầu vào tham chiếu T là bên trái.

2. Xét kỹ toàn bộ đường đi từ T đến gốc của V. Chuyển đổi bất kỳ toán tử phép kết ngoại đầy đủ thành phép kết ngoại bên trái và bất kỳ toán tử phép kết ngoại bên phải thành phép kết nội.

3. Thay thế T bởi ΔT

Bước 1 là một cách viết lại thông thường của biểu thức khung nhìn và không làm thay đổi kết quả. Bước 2 chỉnh sửa biểu thức để nó loại bỏ tất cả các dòng dữ liệu mà không thể trở thành một phần của V^D. Sau Bước 2, các toán tử trên đường đi từ T đến gốc chỉ bao gồm các phép chọn, phép kết nội và phép kết ngoại bên trái và biểu thức delta luôn luôn là đầu vào bên trái. Sự chính xác của Bước 3 theo sau từ quy tắc lan truyền delta sau đây.

$$\sigma_p(e_1 \pm \Delta e_1) = \sigma_p e_1 \pm \sigma_p \Delta e_1$$

$$(e_1 \pm \Delta e_1) \ltimes_p e_2 = e_1 \ltimes_p e_2 \pm \Delta e_1 \ltimes_p e_2$$

$$(e_1 \pm \Delta e_1) \ltimes^{l_0}_p e_2 = e_1 \ltimes^{l_0}_p e_2 \pm \Delta e_1 \ltimes^{l_0}_p e_2$$

Trong đó ± đại diện cho hoặc một bộ hợp hoặc một bộ hiệu. Các quy tắc cho các phép chọn và phép kết nội là hiển nhiên. Quy tắc cho phép kết ngoại bên trái có thể được tìm thấy trong [9].

a. Phép kết ngoại bên trái chuyển đổi sang một cây sâu – trái

b. Luật kết hợp cho phép kết ngoại trái

2.1.5. Phương pháp tính toán delta thứ cấp

a. Rút ra các delta số hạng từ ΔV^D

b. Tính toán ΔV^I sử dụng khung nhìn

2.1.6. Phương pháp khai thác các khóa ngoại CSDL

a. Đơn giản hóa việc tính ΔV^D

b. Đơn giản hóa việc tính ΔV^I

2.2. PHÂN TÍCH THUẬT TOÁN CẬP NHẬT GIA TĂNG KHUNG NHÌN THỰC NỐI NGOÀI

Để hiểu rõ hơn thuật toán cập nhật gia tăng khung nhìn thực nối ngoài, chúng ta xem xét lại ví dụ đầu tiên ở mục 2. 1. 1. Gán bảng Customer, Orders, Employee tương ứng với C, O, E.

Khung nhìn *oj_view* được viết lại như sau:

$$V = (E \overset{lo}{\ltimes}_{p(e,o)} O) \overset{fo}{\ltimes}_{p(o,c)} C .$$

Chúng ta chèn bộ dữ liệu mới vào bảng orders với các dòng dữ liệu mới được chứa trong bảng new_orders.

2.2.1. Biểu đồ gộp

Chuyển biểu thức khung nhìn V đến dạng thức thông thường:

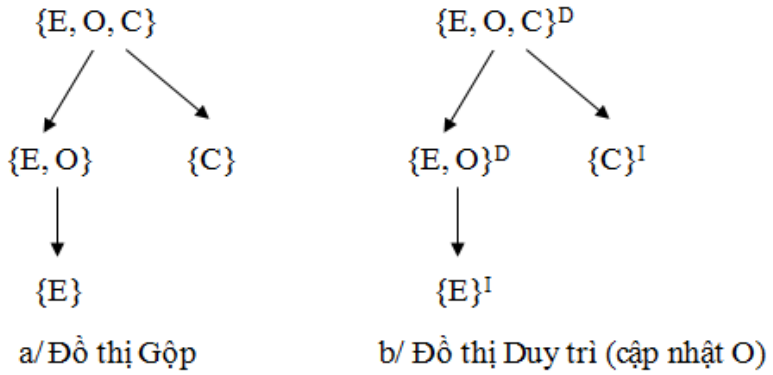
$$V = (\sigma_{p(e,o)}(E \times O) \oplus E) \overset{fo}{\bowtie}_{p(o,c)} C$$

Bởi vì kết nối là một phép kết ngoại đầy đủ, thêm các số hạng từ đầu vào bên trái và bên phải. Điều này tạo ra dạng thức thông thường sau:

$$V = (\sigma_{p(e,o)}(E \times O) \oplus E) \overset{}{\bowtie}_{p(o,c)} C \oplus \sigma_{p(e,o)}(E \times O) \oplus E \oplus C$$

Vì E được mở rộng rỗng trên O (vì $E \overset{lo}{\bowtie}_{p(e,o)} O$) và kết nối thuộc tính p(o,c) là từ chối rỗng nên các dòng dữ liệu từ C không thể kết nối với E, ta có dạng thức kết nối – phân ly bằng:

$$V = (\sigma_{p(e,o) \Lambda p(o,c)}(E \times O \times C) \oplus \sigma_{p(e,o)}(E \times O) \oplus E \oplus C$$



Hình 2.4. Đồ thị Gộp và Duy trì cho khung nhìn V (cập nhật O)

2.2.2. Tính delta chính

Kết nối E $\bowtie^{lo}_{p(e,o)}$ O có thể tạo ra ba loại các dòng dữ liệu: EO, E. Tất cả các dòng dữ liệu E được mở rộng rỗng trên O và do đó, không bao giờ trở thành một phần của V^D vì không có dòng dữ liệu nào trong V^D được mở rộng rỗng trên O. Loại bỏ dòng dữ liệu E bằng cách thay đổi kết nối trái bằng kết nối trong:

$$V = (O \bowtie_{p(e,o)} E) \bowtie^{fo}_{p(o,c)} C$$

Kết nối tiếp theo trên đường đi là $\bowtie^{fo}_{p(o,c)}$. Bởi vì, kết nối là phép kết ngoại đầy đủ, nó duy trì các dòng dữ liệu không phù hợp từ đầu vào phải. Tuy nhiên, chúng được mở rộng rỗng trên O và do đó, không thể trở thành một phần của VD. Chúng ta có thể loại bỏ các dòng dữ liệu không phù hợp bằng cách thay đổi kết nối từ phép kết ngoại đầy đủ thành một phép kết ngoại bên trái.

Khi O được cập nhật, thuật toán cập nhật gia tăng khung nhìn thực nối ngoài tạo ra biểu thức sau đây cho ΔV^D:

$$\Delta V^D = (\Delta O \bowtie_{p(e,o)} E) \bowtie^{lo}_{p(o,c)} C$$

2.3. THUẬT TOÁN SINH MÃ TỰ ĐỘNG TRIGGER

Mã nguồn trigger được chia làm 2 phần, phần mã cố định và phần mã tùy biến. Mã cố định là những dòng mã giống nhau cho mọi trigger, được xây dựng thành khung (frame) và viết thẳng vào file trong quá trình sinh mà không cần phân tích. Phần mã tùy biến là phần chính trong quá trình sinh tự động mã trigger. Để xác định được phần này, cần thực hiện phân tích câu truy vấn gốc (TVG – là câu

truy vấn ban đầu cần tạo KNT), từ đó xác định các thông tin cần thiết cho việc sinh mã CNGT.

a. Mã cố định

Mã cố định bao gồm một số dòng mã như sau:

- Khai báo các thư viện.
- Khai báo các biến điều khiển.
- Chạy các thủ tục, macro được quy ước khi khai báo hàm ngôn ngữ C chạy trong PostgreSQL. (kiểm tra lời gọi hàm, dòng dữ liệu NULL, giá trị trả về…)
- Kết nối/ngắt kết nối đến SPI (Server Programming Interface – cho phép trigger có thể chạy các truy vấn trong thân hàm).

b. Mã tùy biến

Trước tiên ta cần phải phân tách TVG thành các thành phần nhỏ, sau đó thực hiện phân tích các thông tin như danh sách các trường khóa, kiểu dữ liệu của từng trường, vị trí của trường trong bảng…(nhờ vào giao diện lập trình mà PostgreSQL cung cấp để truy vấn đến CSDL), từ đó xác định các biến dữ liệu và biểu thức điều kiện ban đầu cho từng bảng tham gia vào truy vấn, rồi áp dụng những khuôn mẫu CNGT từ thuật toán đã đề cập [4] để sinh mã trigger cho từng sự kiện trên mỗi bảng.

- **Danh sách biến dữ liệu**
- **Điều kiện ban đầu**
- **Các khuôn mẫu CNGT cho từng sự kiện**

Điều quan trọng trong việc dùng trigger để CNGT là phải xác định được thời gian gọi trigger cho từng sự kiện tương ứng. Đối với sự kiện insert, trigger phải được gọi sau khi sự kiện xảy ra; đối với sự kiện delete, trigger phải được gọi trước khi sự kiện xảy ra (và

có thể được gọi lại sau khi sự kiện xảy ra đối với trường hợp có xuất hiện hàm thống kê min hoặc max, theo thuật toán CNGT đồng bộ KNT [4]); tuy nhiên đối với sự kiện update, do thuật toán CNGT đồng bộ KNT [4] xem sự kiện này như 2 thao tác delete rồi insert riêng biệt, nên trigger cho sự kiện update phải được gọi cả trước và sau khi sự kiện này xảy ra.

Dựa theo các thông tin đã phân tích được và thuật toán CNGT đồng bộ KNT [4], giả sử phần khung mã cứng đã có, đối với từng sự kiện ta sinh mã mềm như sau:

❖ **Sự kiện Insert**

1. Khai báo biến dữ liệu.
2. Gán giá trị cho biến ứng với các cột ở dòng thêm mới.
3. Kiểm tra biểu thức điều kiện ban đầu.
4. Chạy lại TVG với mệnh đề where có thêm vào các trường khóa với dữ liệu cụ thể.
5. Cấp phát bộ nhớ và lưu kết quả truy vấn ở bước (4), với mỗi dòng dữ liệu đã lưu, thực hiện bước (6).
6. Truy vấn đến bảng KNT lấy giá trị trường COUNT(*). Nếu truy vấn không trả về dòng kết quả nào, tức giá trị của nhóm dữ liệu vẫn chưa xuất hiện trong KNT, thực hiện thêm mới dòng dữ liệu đó vào KNT, trường COUNT(*) có giá trị là 1. Ngược lại, nếu truy vấn trả về ít nhất 1 dòng kết quả, thực hiện bước 7.
7. Thực hiện cập nhật KNT, các trường có chứa các biểu thức thống kê cần được cập nhật giá trị phù hợp với thuật toán CNGT, trường COUNT(*) được cập nhật tăng lên 1.

❖ **Sự kiện delete**

1. Khai báo biến dữ liệu.
2. Gán giá trị cho biến ứng với các cột ở dòng đang xóa.
3. Kiểm tra biểu thức điều kiện ban đầu.
4. Đối với trường hợp không có xuất hiện hàm thống kê min và/hoặc max, thực hiện sinh mã từ bước 5 đến bước 8. Ngược lại, nếu có, thực hiện sinh mã từ bước 5 đến bước 8 với thời gian gọi trigger trước khi sự kiện xảy ra, thực hiện sinh mã từ bước 9 đến 11 với thời gian gọi trigger sau khi sự kiện xảy ra.
5. Chạy lại TVG với mệnh đề where có thêm vào các trường khóa với dữ liệu cụ thể.
6. Cấp phát bộ nhớ và lưu kết quả truy vấn ở bước (5), với mỗi dòng, thực hiện bước (6).
7. Truy vấn đến bảng KNT lấy giá trị trường COUNT(*). Trong trường hợp delete, xem như luôn có dòng dữ liệu trong KNT, so sánh trường đếm COUNT(*) trong KNT, gọi là x và giá trị trường đếm ở bước (5) gọi là y, nếu x = y thì thực hiện xóa dòng dữ liệu đó trong KNT, nếu x > y thì thực hiện bước (8).
8. Thực hiện cập nhật KNT, các trường có chứa các biểu thức thống kê cần được cập nhật giá trị phù hợp với thuật toán CNGT, trường đếm COUNT(*) trong KNT được cập nhật x = x-y.
9. Thực hiện lại TVG với mệnh đề where được thêm vào các cặp điều kiện trường – giá trị thỏa điều kiện: trường được thêm vào phải có mặt trong BG và mệnh đề group by của TVG.

10. Cấp phát bộ nhớ và lưu kết quả truy vấn ở bước (9), với mỗi dòng, thực hiện bước (11).
11. Cập nhật lại các trường thống kê min/max từ kết quả truy vấn đã có.

❖ **Sự kiện update**

1. Khai báo các biến dữ liệu.
2. Kiểm tra thời điểm gọi trigger, nếu trigger được gọi trước sự kiện update thì thực hiện bước (3), nếu trigger được gọi sau sự kiện update thì thực hiện bước (4).
3. Trigger gọi trước (before update): thực hiện tương tự như cho trường hợp delete đã đề cập ở trên từ bước (2) trở đi.
4. Trigger gọi sau (after update): thực hiện tương tự như cho trường hợp insert đã đề cập ở trên từ bước (2) trở đi (và có thể cùng với một phần của trigger delete nếu có xuất hiện hàm thống kê min/max, phần này, nếu có, sẽ được đặt trước đoạn mã trigger cho sự kiện insert).

2.4. TIỂU KẾT CHƯƠNG 2

Trong phần này, chúng tôi trình bày tổng quan về KNT, các phương pháp để tính delta chính và delta thứ cấp, phương pháp khai thác các khóa ngoại trong các HQT CSDL để đơn giản hóa việc tính delta chính và delta thứ cấp. Đặc biệt, trong chương 2 này chúng tôi còn phân tích chi tiết thuật toán cập nhật gia tăng KNT nối ngoài qua việc minh họa bằng một ví dụ thực tế. Dựa trên thuật toán cập nhật gia tăng KNT nối ngoài đã trình bày, chúng tôi xây dựng thuật toán sinh mã tự động các trigger cho từng sự kiện insert, delete và update.

CHƯƠNG 3
XÂY DỰNG CHƯƠNG TRÌNH SINH TỰ ĐỘNG MÃ NGUỒN CÁC TRIGGER THỰC HIỆN CẬP NHẬT GIA TĂNG KHUNG NHÌN THỰC NỐI NGOÀI

Trên cơ sở tìm hiểu thuật toán cập nhật gia tăng KNT nối ngoài và xây dựng được thuật toán sinh mã tự động các trigger đã trình bày, chúng ta sẽ triển khai xây dựng chương trình sinh tự động mã nguồn các trigger thực hiện cập nhật gia tăng KNT nối ngoài.

3.1. XÂY DỰNG ỨNG DỤNG

3.1.1. Công cụ và ngôn ngữ lập trình

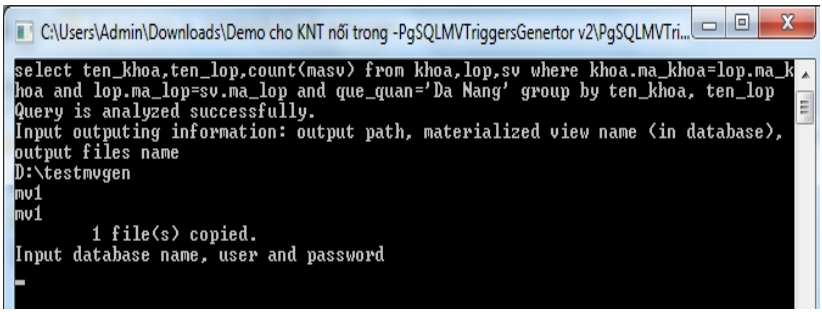
Chương trình được viết bằng ngôn ngữ C trên nền Visual Studio 12.0, sử dụng hệ quản trị cơ sở dữ liệu PostgreSQL 9.3.

3.1.2. Mô hình hoạt động của ứng dụng

3.1.3. Cài đặt ứng dụng

Ứng dụng được cài đặt trên máy đơn chạy hệ điều hành Windows 7, cài đặt Visual Studio 12.0, sử dụng hệ quản trị cơ sở dữ liệu PostgreSQL 9.3.

3.1.4. Giao diện của ứng dụng



Hình 3.3. Màn hình chương trình sau khi nhập đường dẫn thư mục chứa ouput, tên bảng KNT trong CSDL, tên của các file được sinh ra

3.2. ĐÁNH GIÁ KẾT QUẢ THỰC NGHIỆM

3.2.1. Tạo cơ sở dữ liệu thực nghiệm

Từ thư mục lưu file QLSV.sql, ta mở file bằng chương trình Notepad, copy toàn bộ nội dung trong file. Khởi động PostgreSQL, chọn cơ sở dữ liệu, chọn Execute arbitrary SQL queries trên Tool bar, và chép toàn bộ nội dung vừa copy vào cửa sổ SQL Editor.

Sau khi thực thi lệnh Execute query, các bảng bao gồm: KHOA, LOP, SV sẽ được cài đặt trên hệ quản trị cơ sở dữ liệu PostgreSQL.

3.2.2. Chương trình sản phẩm

Mở file .sln của chương trình bằng Visual Studio 12.0, thực hiện build chương trình. Chương trình có giao diện console.

a. Input

Đầu vào nhập trên luồng input chuẩn bao gồm:

- Truy vấn gốc: select ten_khoa, ten_lop, count(ma_sv) from khoa, lop, sv where khoa.ma_khoa = lop.ma_khoa and lop.ma_lop = sv.ma_lop and que_quan = 'Da Nang' group by ten_khoa, ten_lop
- Đường dẫn thư mục chứa ouput: D:\testmvgen
- Tên bảng KNT trong CSDL:mv1
- Tên của các file được sinh ra: mv1
- Tên CSDL chứa TVG: QLSV
- Username và password tài khoản quản trị trong HQTCSDL PostgreSQL:
    + Username: postgres
    + Password: 011184

b. Output

Sau khi nhập đầy đủ thông tin đầu vào, chương trình tự động sinh ra 3 file nằm trong thư mục D:\testmvgen mà người dùng đã chỉ

định sẵn ở input. Chúng được dùng để cài đặt KNT và các trigger cập nhật nó:

- mv1_mvsrc.sql: file mã SQL.
- ctrigger.h: file header định nghĩa các thủ tục macro dùng trong trigger
- mv1_triggersrc.c: file mã nguồn C của trigger.

c. Cài đặt KNT dựa trên output

Từ các file được sinh ra, ta thực hiện cài đặt KNT và các trigger để cập nhật nó như sau:

- Biên dịch file mã nguồn C (*.c) thành file thư viện chia sẻ (*.dll trên hệ điều hành Windows hoặc *.so cho các hệ điều hành Linux).
- Đặt file thư viện vào thư mục \lib trong cây thư mục cài đặt PostgreSQL trên máy tính (thường nằm ở thư mục C:\Program Files\PostgreSQL9.3\).
- Chạy mã trong file SQL trong HQTCSDL PostgreSQL để tạo KNT và khai báo các trigger cho các BG liên quan.

3.2.3. Đánh giá kết quả thực nghiệm

a. Chương trình sản phẩm

Chương trình được viết bằng C nên dễ dàng tích hợp vào HQTCSDL PostgreSQL dưới dạng module. Ngoài ra các thông tin của TVG đã được phân tích thành các kiểu cấu trúc, thuận tiện cho việc xây dựng thêm module tổng hợp kết quả cho các truy vấn liên quan từ dữ liệu trên KNT. Chương trình đáp ứng được vấn đề đặt ra đó là sinh mã nguồn trigger bằng ngôn ngữ C thực hiện CNGT KNT. Tuy nhiên, do thời gian còn hạn chế, nên các kiểu dữ liệu cũng như các hàm xử lí thông tin trong SQL vẫn chưa được hỗ trợ đầy đủ trong

chương trình. Trong tương lai, chương trình sẽ tự động hóa quá trình cài đặt KNT trong CSDL.

b. Đánh giá tốc độ cập nhật dữ liệu trên các BG có trigger cập nhật KNT

TVG đã được thay thế bằng câu lệnh select đơn giản truy vấn đến bảng KNT nên tốc độ thực hiện TVG xem như đạt đến mức tối đa. Tuy vậy, ở các BG, những trigger cập nhật gia tăng đồng bộ cho KNT làm cho thao tác cập nhật KNT trở thành một phần trong quá trình cập nhật dữ liệu trên BG, cho nên, thao tác cập nhật dữ liệu trên BG sẽ bị chậm đi tỉ lệ thuận với khối lượng dữ liệu trên các BG, độ phức tạp của TVG và số lượng KNT được sử dụng. Phần đánh giá này sẽ giúp ước lượng được phần nào thời gian bị chậm đi khi cập nhật dữ liệu trên BG có trigger cập nhật KNT.

- TVG:

```
SELECT
    "KHOA"."MA_KHOA","KHOA"."TEN_KHOA",
"LOP"."TEN_LOP","LOP"."NIEN_KHOA","SV"."MA_SV
", SV"."HO_TEN", "SV"."NGAY_SINH", "SV"."QUE_QUAN"
FROM "KHOA", "LOP", "SV"
WHERE "KHOA"."MA_KHOA" = "LOP"."MA_KHOA"
AND "LOP"."MA_KHOA" = "SV"."MA_KHOA"
GROUP BY "SV"."QUE_QUAN"
```

- Số lượng dòng dữ liệu của các bảng tham gia truy vấn:

KHOA: 8

LOP: 100

SV: 5000

- Đánh giá thời gian thực thi các thao tác cập nhật trên BG

Đối với các lệnh insert/delete: Thời gian thực thi khi không có trigger trung bình dao động từ 20-25 ms/1 lệnh; thời gian thực thi khi có trigger cập nhật KNT trung bình dao động từ 30-50ms/1 lệnh.

Đối với lệnh update: Thời gian thực thi khi không có trigger trung bình dao động từ 260ms – 280s/1 lệnh; thời gian thực thi khi có trigger cập nhật KNT dao trung bình dao động từ 270ms – 280ms/1 lệnh.

Các thông số kĩ thuật trên máy đánh giá:

Phiên bản HQTCSDL: PostgreSQL 9.3 (32-bit)

CPU: Intel core i3

RAM: 2GB

HĐH: Windows 7 64-bit

3.3. TIỂU KẾT CHƯƠNG 3

Trong chương này, luận văn sẽ trình bày quá trình xây dựng module sinh tự động mã nguồn các trigger thực hiện cập nhật gia tăng khung nhìn thực cho các sự kiện thêm mới, xóa, cập nhật các bảng cơ sở. Từ đó xây dựng chương trình sinh tự động mã nguồn các trigger đó bằng ngôn ngữ C. Chúng tôi còn đưa ra một số hình ảnh khi chạy chương trình và cách cài đặt chương trình.

KẾT LUẬN

1. KẾT LUẬN

Luận văn này trình bày các kết quả nghiên cứu của chúng tôi về khung nhìn thực nối ngoài và phương pháp luận để cập nhật gia tăng khung nhìn thực nối ngoài trong Hệ quản trị CSDL PostgreSQL và ngôn ngữ C. Qua đó, chúng tôi đã trình bày tổng quan các nghiên cứu liên quan đến khung nhìn thực, các phương pháp để cập nhật gia tăng, trong đó trình bày sâu các phương pháp để tính toán delta chính và delta thứ cấp, và phương pháp khai thác các khóa ngoại trong các HQT CSDL.

Kết quả của luận văn này, chúng tôi xây dựng được module hỗ trợ sinh mã tự động các trigger thực hiện cập nhật gia tăng các bảng khung nhìn thực nối trong liên quan đến các truy vấn như INSERT, UPDATE, DELETE.

2. HƯỚNG PHÁT TRIỂN

- Chương trình phát triển cho khung nhìn thực nối ngoài dựa trên thuật toán đã trình bày.

- Tự động hóa quá trình cài đặt khung nhìn thực trong CSDL.