Kỷ yếu Hội nghị Quốc gia lần thứ VIII về Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR); Hà Nội, ngày 9-10/7/2015
DOI: 10.15625/vap.2015.000213

TRUY VẤN HƯỚNG ĐỐI TƯỢNG DỰA TRÊN PHÂN CẤP TẬP TIN CHỮ KÝ VÀ CÂY SD-TREE

Trần Minh Bảo[1], Trương Công Tuấn[2]
[1, 2] Trường Đại học Khoa học, Đại học Huế
tmbaovn@gmail.com, tctuan_it_dept@yahoo.com

TÓM TẮT - *Truy vấn trực tiếp trên các đối tượng trong cơ sở dữ liệu hướng đối tượng rất tốn kém chi phí lưu trữ dữ liệu trong quá trình truy vấn và tốn nhiều thời gian để thực hiện truy vấn trên hệ thống dữ liệu thực. Gần đây, có nhiều nghiên cứu tập trung vào việc giải quyết vấn đề đó bằng cách xây dựng các chỉ mục trên các lớp đơn, phân cấp lớp, hoặc phân cấp đối tượng lồng nhau. Trong bài báo này, chúng tôi sẽ đề xuất một phương pháp lập chỉ mục mới. Phương pháp này dựa trên kỹ thuật sử dụng tập tin chữ ký và cây SD-tree trong đó các tập tin chữ ký được tổ chức theo phân cấp để nhanh chóng lọc dữ liệu không thích hợp và mỗi tập tin chữ ký được lưu theo cấu trúc cây SD-tree nhằm tăng tốc độ quét chữ ký. Kỹ thuật này giúp giảm đáng kể không gian tìm kiếm và do đó sẽ cải thiện đáng kể độ phức tạp thời gian truy vấn.*

Từ khóa - *Hệ thống cơ sở dữ liệu hướng đối tượng, chỉ mục, tập tin chữ ký, cây SD-Tree, truy vấn hướng đối tượng.*

I. MỞ ĐẦU

Truy vấn trực tiếp trên các đối tượng trong cơ sở dữ liệu hướng đối tượng rất tốn kém chi phí lưu trữ dữ liệu và tốn nhiều thời gian để thực hiện trên hệ thống dữ liệu thực. Bài toán đặt ra là cần mô tả lại hệ thống dữ liệu đơn giản hơn và xây dựng cấu trúc dữ liệu tương ứng để có thể giảm không gian tìm kiếm trong quá trình thực thi câu truy vấn mà vẫn đảm bảo được việc truy vấn được các đối tượng cần thiết.

Để giảm không gian truy vấn dữ liệu, các kỹ thuật chỉ mục sử dùng đánh giá truy vấn trong CSDL được đề xuất trong [6] đã được phát triển dựa trên cơ chế thăng bằng cây nhị phân thêm vào một số tính chất đặc biệt để giảm việc cân bằng cây hoặc để tối thiểu hóa các truy cập vào tập tin dữ liệu. Các kỹ thuật này đã được phát triển tiếp nhằm tăng tốc truy vấn trong các CSDL hướng đối tượng [10, 11, 12]. Ý tưởng chính ở đây là mỗi cây *SD-tree* trên một lớp trong phân cấp lớp vẫn được duy trì nhưng các chỉ mục sẽ được lồng ghép với nhau bằng mối quan hệ lớp con – lớp mục tiêu. Ngoài các chỉ mục theo cấu trúc phân cấp thừa kế còn có rất nhiều các phương pháp lập chỉ mục dùng cho truy vấn thuộc tính lồng nhau đã được đề xuất [1, 2, 3, 7, 9]. Thay vì tập trung vào phân cấp thừa kế các lớp, các nhà nghiên cứu khác đã khám phá ra sự phân cấp tổng hợp các lớp và đề xuất các cấu trúc lập chỉ mục khác nhau theo các thuộc tính lồng nhau [1, 2, 7, 9]… các cấu trúc lưu trữ tập tin chữ ký sẽ làm giảm không gian tìm kiếm và tối ưu quá trình truy vấn dữ liệu.

Để việc tìm kiếm hiệu quả hơn, cần xây dựng cấu trúc dữ liệu lưu trữ tập tin chữ ký. Cấu trúc lưu trữ tập tin chữ ký này có thể dưới dạng các tập tin chữ ký tuần tự, các tập tin chữ ký phân mảnh, cấu trúc cây chữ ký, cấu trúc dạng đồ thị chữ ký,… Trong đó, chi phí lưu trữ của tập tin chữ ký phân mảnh lại gấp đôi tập tin chữ ký tuần tự và chi phí cập nhật của tập tin chữ ký phân mảnh cũng gấp ba lần tập tin chữ ký tuần tự hoặc nhiều hơn [8]. Ưu điểm cơ bản của phương pháp tập tin chữ ký tuần tự nằm ở hiệu quả xử lý chèn và truy vấn mới lên các phần của từ. Tuy nhiên, khi so sánh với lập chỉ mục dựa trên cấu trúc cây thì các tập tin chữ ký liên tiếp lại bị hai nhược điểm, thứ nhất không thể được dùng để đánh giá các truy vấn phạm vi và thứ hai đối với mỗi truy vấn được xử lý thì toàn bộ tập tin chữ ký cần phải được quét làm tăng chi phí xử lý I/O.

Trong bài báo này, chúng tôi sẽ cố gắng cải thiện vấn đề thứ hai đến một mức độ nào đó. Đầu tiên, chúng tôi tổ chức các tập tin chữ ký tuần tự sang cấu trúc phân cấp dùng để giảm bớt không gian tìm kiếm trong quá trình đánh giá truy vấn. Tiếp theo, chúng tôi lưu trữ tập tin chữ ký dưới dạng cây *SD-tree*, nhằm tiến hành quét chỉ một tập tin chữ ký đơn nhất. Nếu tập tin chữ ký có kích thước lớn thì khối lượng thời gian tiết kiệm được bằng phương pháp này là rất đáng kể. Cây *SD-tree* được xây dựng dựa trên tập tin chữ ký. Do đó, nó có thể tăng tốc quá trình xác định vị trí chữ ký trong một tập tin chữ ký. Tuy nhiên, trong cây *SD-tree*, mỗi đường dẫn sẽ tương ứng với một định danh chữ ký có thể dùng để xác định duy nhất chữ ký tương ứng với nó trong tập tin chữ ký. Cách này giúp nhanh chóng tìm ra một tập hợp các chữ ký phù hợp với chữ ký truy vấn.

Bài báo này được tổ chức như sau. Trong phần II, chúng tôi đưa ra một số kiến thức cơ sở. Tại phần III, chúng tôi sẽ giới thiệu cấu trúc dữ liệu và thuật toán truy vấn. Phần IV đề xuất phương pháp kết hợp phân cấp tập tin chữ ký và cây *SD-tree*. Cuối cùng, phần V sẽ đưa ra một kết luận.

II. MỘT SỐ KHÁI NIỆM CƠ SỞ

A. Chữ ký thuộc tính

Trong một CSDL hướng đối tượng, mỗi đối tượng được biểu diễn bởi một bộ giá trị của các thuộc tính. Chữ ký của một giá trị thuộc tính là một chuỗi các bit được mã hóa bằng hàm băm. Cho trước một giá trị thuộc tính, ví dụ chữ "truong", phân giải nó thành một chuỗi các bộ ba chữ cái như sau: "tru", "ruo", "uon" và "ong". Sau đó, dùng hàm băm h, ánh xạ một bộ ba thành một số nguyên k với ý nghĩa là bit thứ k trong chuỗi sẽ được đặt giá trị 1.

Ví dụ, giả sử có h(tru) = 2, h(ruo) = 7, h(uon) = 10, and h(ong) = 11. Sau đó thiết lập một chuỗi bit: 010 000 100 110 làm chữ ký cho từ.

B. Chữ ký đối tượng, tập tin chữ ký

Chữ ký đối tượng được xây dựng bằng phép toán logic OR cho tất cả các chữ ký của các giá trị thuộc tính của đối tượng. Sau đây là một ví dụ về chữ ký của đối tượng:

Ví dụ 1. Xét một đối tượng có các giá trị thuộc tính lần lượt là "truong", "12345678", "giao su". Giả sử chữ ký của các đối tượng này là:

010 000 100 110

100 010 010 100

110 100 011 000

Lúc đó chữ ký của đối tượng là 110 110 111 110, nhận được từ chữ ký các giá trị thuộc tính bằng phép toán logic OR. Các chữ ký đối tượng của một lớp được lưu trữ trong một tập tin, gọi là tập tin chữ ký đối tượng.

C. Chữ ký truy vấn

Một truy vấn đối tượng sẽ được mã hóa thành một chữ ký truy vấn theo cùng hàm băm đã thực hiện đối với các đối tượng. Khi có một truy vấn cần thực hiện, các chữ ký đối tượng sẽ được quét và những đối tượng không phù hợp sẽ bị loại trừ. Lúc đó chữ ký truy vấn được so sánh đối với mọi chữ ký đối tượng trong tập tin chữ ký. Có ba trường hợp có thể xảy ra:

(i) Đối tượng phù hợp với truy vấn, nghĩa là đối với mọi bit trong chữ ký truy vấn s_q, bit tương ứng trong chữ ký đối tượng s cũng chính là nó, tức là $s_q \wedge s = s_q$, đối tượng thực sự chứa từ truy vấn;

(ii) Đối tượng không phù hợp với câu truy vấn, nghĩa là $s_q \wedge s \neq s_q$;

(iii) Chữ ký được đối sánh và cho ra một kết quả phù hợp nhưng đối tượng không phù hợp với điều kiện tìm kiếm trong truy vấn. Để loại ra trường hợp này, các đối tượng phải được kiểm tra sau khi các chữ ký đối tượng được đối sánh phù hợp.

Ví dụ 2. Ví dụ này minh họa việc truy vấn đối với chữ ký đối tượng ở ví dụ 1:

Truy vấn	Chữ ký truy vấn	Kết quả
truong	010 000 100 110	thành công
binh	011 000 100 100	không thành công
duong	110 100 100 000	thành công nhưng không phù hợp

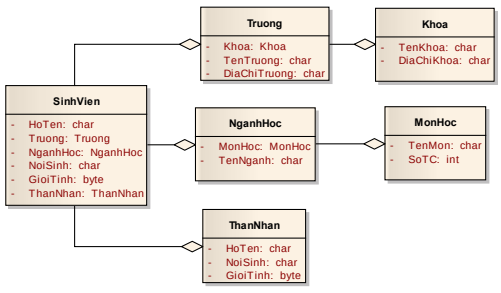
Nhận xét: Việc so sánh chữ ký truy vấn s_q với chữ ký đối tượng s là loại đối sánh không chính xác. Nghĩa là, chữ ký truy vấn s_q phù hợp chữ ký s nếu mọi bit 1 trong s_q, các bit tương ứng trong s cũng là bit 1. Tuy nhiên, đối với mọi bit 0 trong s_q thì không quan trọng bit tương ứng trong s là 0 hay 1.

III. CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN TRUY VẤN

A. Truy vấn trong cơ sở dữ liệu hướng đối tượng

Trong các hệ thống CSDL hướng đối tượng, một thực thể sẽ được biểu diễn dưới dạng đối tượng bao gồm các phương thức và thuộc tính. Các đối tượng có cùng tập thuộc tính và phương thức được nhóm lại với nhau trong cùng một lớp. Nếu một lớp C có một thuộc tính phức hợp với miền C' thì sẽ thiết lập quan hệ giữa C và C'. Quan hệ này được gọi là mối quan hệ tổng hợp. Khi dùng mũi tên kết nối các lớp này để biểu diễn mối quan hệ tổng hợp thì một phân cấp tổng hợp có thể được xây dựng để hiển thị cấu trúc lồng nhau của các lớp.

Ví dụ 3. Một ví dụ về hệ thống phân cấp đối tượng lồng nhau được minh họa như sau:



Hình 1. Một ví dụ về hệ thống phân cấp đối tượng lồng nhau

Nếu đối tượng *o* được tham chiếu là một thuộc tính của đối tượng *o'*, thì sau đó *o* được coi là được lồng nhau trong *o'* còn *o'* sẽ được gọi là các đối tượng cha của *o*.

Trong CSDL hướng đối tượng, điều kiện tìm kiếm trong truy vấn được biểu diễn dưới dạng một tổ hợp của các thuộc tính. Thuộc tính này có thể là thuộc tính lồng nhau của lớp mục tiêu.

Ví dụ 4. Truy vấn "tìm tất cả các sinh viên sinh ở Bến Tre của Khoa Công nghệ thông tin" có thể được biểu diễn như sau:

Select SinhVien

Where SinhVien.NoiSinh = "*Bến Tre*"

And SinhVien.Truong.Khoa.TenKhoa = "*Công nghệ Thông tin*"

Nếu không có cấu trúc chỉ mục thì truy vấn nói trên có thể được đánh giá theo trình tự trên xuống như sau. Đầu tiên, hệ thống phải tìm kiếm tất cả các đối tượng trong lớp "*SinhVien*" và lọc ra các đối tượng có nơi sinh ở "*Bến Tre*". Tiếp đó hệ thống sẽ tìm các đối tượng trường có tham chiếu là sinh viên có nơi sinh ở "*Bến Tre*" và kiểm tra tên khoa của trường. Cuối cùng, những sinh viên có nơi sinh ở "*Bến Tre*" và thuộc khoa "*Công nghệ Thông tin*" của trường sẽ được trả về.

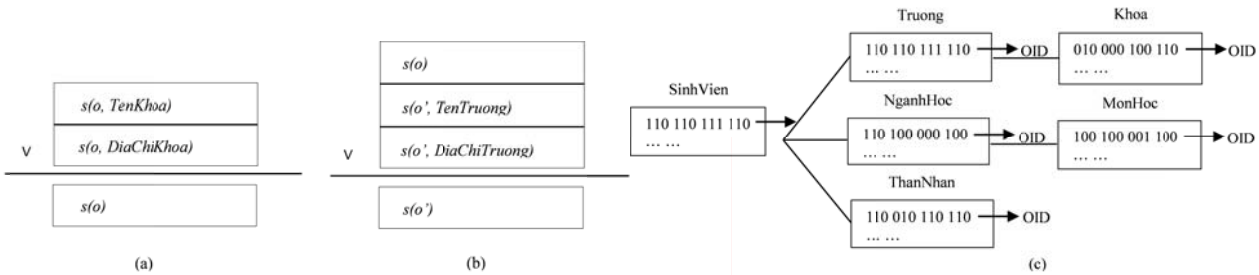
B. Phân cấp tập tin chữ ký và thuật toán truy vấn

1. Phân cấp tập tin chữ ký

Mục đích của việc sử dụng tập tin chữ ký là để lọc ra hầu hết các đối tượng không đủ điều kiện, nghĩa là một chữ ký không phù hợp với chữ ký truy vấn sẽ đảm bảo rằng đối tượng liên quan với chữ ký này sẽ bị bỏ qua. Do đó sẽ tránh được việc truy cập không cần thiết vào các đối tượng này. Đối với phân cấp tổng hợp, có thể xây dựng phân cấp tập tin chữ ký của nó như sau:

(i) Chữ ký của đối tượng được tạo ra bằng cách xếp chồng các chữ ký của tất cả các thuộc tính nguyên thủy và phức hợp của nó.
(ii) Chữ ký của thuộc tính nguyên thủy được tạo ra bằng cách băm các giá trị thuộc tính; chữ ký của thuộc tính phức hợp là chữ ký của đối tượng mà nó tham chiếu.
(iii) Giả sử *C* là ký hiệu lớp với o_1, ..., o_l là các đối tượng của lớp đó; tồn tại một tập tin chữ ký *S* mà o_i (i = 1, ..., l) có một mục <*osig, oid*> trong *S*.
(iv) Gọi S_i và S_j là hai tập tin chữ ký tương ứng liên kết với các lớp C_i và C_j. Nếu tồn tại một mũi tên từ C_i đến C_j, thì cũng ngầm định có một mũi tên từ S_i đến S_j.

Ví dụ 5. Chữ ký và phân cấp tập tin chữ ký được minh họa như sau:



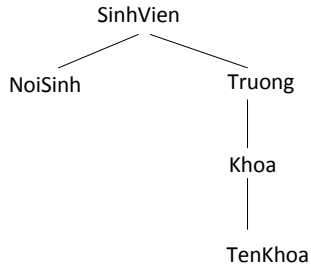
Hình 2. Chữ ký và phân cấp tập tin chữ ký

Xét lớp "*Khoa*" trong phân cấp lớp không các thuộc tính phức hợp tại hình 1. Chữ ký của đối tượng *o* được xây dựng bằng phương pháp như trong hình 2 (a), trong đó mỗi *s(o, x)* ký hiệu chữ ký được tạo ra cho giá trị thuộc tính *x* của *o* và *s(o)* ký hiệu chữ ký của *o*. Đối với lớp các thuộc tính phức hợp thì chữ ký của các đối tượng trong đó có thể được tạo ra theo cùng cách thức như đối với lớp chỉ các thuộc tính nguyên thủy. Sự khác biệt duy nhất là chữ ký của thuộc tính phức hợp là chữ ký của đối tượng mà nó tham chiếu được minh họa ở hình 2(b). Trong hình 2 (b), *o'* ký hiệu cho đối tượng của lớp "*Truong*" và đối tượng *o* của lớp "*Khoa*" là giá trị thuộc tính của "*Khoa*" của *o'*. Một phân cấp tập tin chữ ký mà có thể xây dựng cho một cơ sở dữ liệu được hiển thị trong hình 1 được minh họa ở hình 2(c).

2. Thuật toán truy vấn dựa trên tập tin chữ ký

Định nghĩa 1: (*Cây truy vấn*) Gọi $p_1 \wedge p_2 \ldots \wedge p_k$ là điều kiện tìm kiếm trong truy vấn *Q*, trong đó mỗi p_i là một vị từ có dạng: <*giá trị toán tử thuộc tính*>. Lúc đó, tất cả các đường dẫn xuất hiện trong điều kiện tìm kiếm sẽ tạo thành một cây truy vấn, ký hiệu là Q_t.

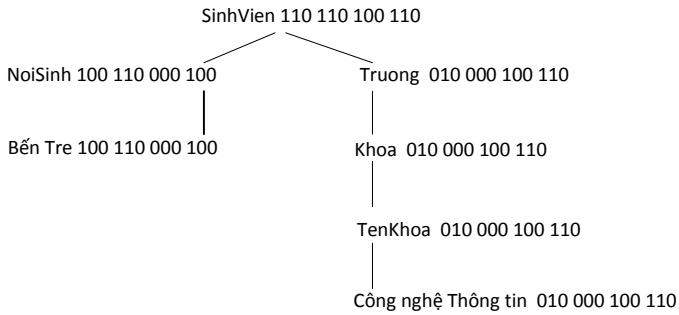
Ví dụ 6. Cây truy vấn được minh họa như sau:



Hình 3. Cây truy vấn

Định nghĩa 2: (*Cây chữ ký truy vấn*) Gọi $p_1.p_2 \ldots .p_n$ là một đường dẫn trong cây truy vấn Q_t. Gọi <$p_i \ldots .p_n$ *giá trị toán tử*> là một vị từ xuất hiện trong điều kiện tìm kiếm của Q. Lúc đó, chữ ký của p_n là s_{value}. Chữ ký của một nút không phải nút lá Q_t được tạo bằng phép toán logic OR cho các chữ ký của các nút con của nó. Cây chữ ký truy vấn được ký hiệu là $Q(s,t)$.

Vi dụ 7. Cây chữ ký truy vấn được minh họa như sau:



Hình 4. Cây chữ ký truy vấn

Sử dụng cây chữ ký truy vấn để giảm bớt không gian tìm kiếm. Phương pháp này, cần có hai cấu trúc ngăn xếp (*stack*) để điều khiển việc quét ưu tiên chiều sâu của các cấu trúc cây: $stack_q$ đối với $Q(s,t)$ và $stack_c$ đối với phân cấp lớp. Trong $stack_q$, mỗi thành phần là một chữ ký trong khi trong $stack_c$, mỗi thành phần là một tập các đối tượng thuộc về cùng một lớp có thể tiếp cận được bằng cách quét phân cấp lớp.

Thuật toán 1. [4] Tìm kiếm phân cấp theo kiểu trên xuống;

Vào: Một truy vấn đối tượng Q;

Ra: Một tập các OID_s thỏa mãn truy vấn.

Phương pháp:

Bước 1. Tính toán phân cấp chữ ký truy vấn $Q_{(s,t)}$ đối với Q.

Bước 2. Đưa chữ ký gốc của $Q_{(s,t)}$ vào $stack_q$; đẩy tập OID đối tượng của lớp mục tiêu vào $stack_c$.

Bước 3. Nếu $stack_q$ không rỗng, $s_q \leftarrow$ lấy ra $stack_q$; ngược lại chuyển sang (Bước 7).

Bước 4. $S \leftarrow$ lấy $stack_c$; với mỗi oid_i E S, nếu chữ ký của nó $osig_i$ không so sánh với s_q, thì loại ra khỏi S; đưa S vào S_{result}.

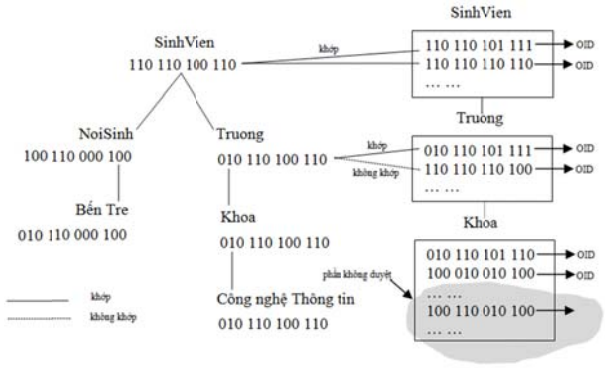
Bước 5. Gọi C là lớp các đối tượng của S; gọi $C_1, \ldots, C_k$ là các lớp con của C; sau đó phân chia tập OID của các đối tượng được tham chiếu bởi S vào $S_1, \ldots, S_k$ sao cho S_i thuộc C_i; đẩy $S_1, \ldots, S_k$ vào $stack_c$; đẩy các nút con của s_q vào $stack_q$.

Bước 6. Chuyển đến (Bước 3).

Bước 7. Đối với mỗi đối tượng lá, kiểm tra phù hợp nhưng không đúng.

Kỹ thuật này giúp đạt được tối ưu hóa khi thực hiện bước (4). Tại bước này, một số đối tượng được lọc bằng cách sử dụng chữ ký tương ứng trong cây chữ ký truy vấn. Trong bước (5), các đối tượng được tham chiếu và các chữ ký của nút con của cây chữ ký truy vấn được tương ứng đưa vào $stack_c$ và $stack_q$. Trong bước (7), thực hiện việc kiểm tra nhằm lẫn.

Ví dụ 8. Giả sử rằng một phần của phân cấp tập tin chữ ký được xây dựng cho một CSDL với sơ đồ được minh họa trong hình 1 thuộc dạng được mô tả trong hình 5:



Hình 5. Minh họa quá trình truy vấn

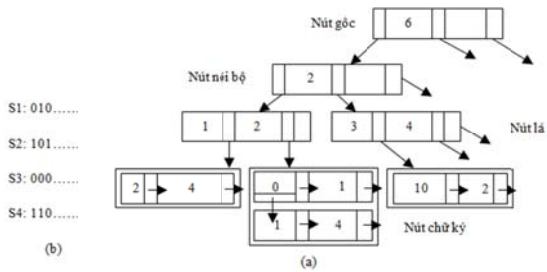
Khi cả hai chữ ký đầu của tập tin chữ ký cho "*SinhVien*" phù hợp với chữ ký tương ứng trong cây chữ ký truy vấn, các chữ ký được tham chiếu bởi chúng trong tập tin chữ ký cho "*Truong*" sẽ được kiểm tra bổ sung. Giả sử rằng chữ ký đầu tiên của "*Truong*" được tham chiếu bởi chữ ký đầu tiên trong "*SinhVien*" trong khi chữ ký thứ hai trong "*Truong*" được tham chiếu bởi chữ ký thứ hai trong "*SinhVien*". Có thể thấy rằng chữ ký thứ hai trong "*Truong*" không phù hợp với chữ ký tương ứng trong cây chữ ký truy vấn. Vì vậy tất cả các chữ ký của đối tượng "*Khoa*" được nó tham chiếu sẽ không bị kiểm tra (xem minh họa tại phần màu xám trong hình 5). Việc này là tối ưu khi so sánh với "*tìm kiếm trên xuống*" vì trong "*tìm kiếm trên xuống*" phải tiến hành kiểm tra tất cả các chữ ký đối tượng của "*Khoa*".

C. Cây SD-tree

1. Tổng quát về cấu trúc *SD-tree*

Kỹ thuật tạo chỉ mục trong hệ thống CSDL hướng đối tượng ở đây bằng cách sử dụng cây *SD-tree* (Phân cụm chữ ký). Trong công việc này, các vị trí của bit 0 và bit 1 trong chữ ký được phân bổ qua một tập hợp các nút lá. Sử dụng phương pháp này cho một chữ ký truy vấn cho sẵn thì tất cả các chữ ký trùng khớp đều có thể được truy vấn tích lũy trong một nút đơn. Việc truy vấn, con đường tìm kiếm tối ưu được tính toán để đẩy nhanh toàn bộ quá trình.

Ví dụ 9. Cây chữ ký *SD-tree* được minh họa như sau:



Hình 6. Cây chữ ký *SD-tree*

Để xử lý một chữ ký truy vấn S_q: (1) Đối với chữ ký có trọng lượng dưới 50% xét sự xuất hiện cuối cùng của bit 1 tại vị trí *i* trong S_q sẽ tìm thấy với sự tạo thành tiền tố trung gian *(B)*; (2) Đối với chữ ký có trọng lượng trên 50% xét sự xuất hiện cuối cùng của bit 0 tại vị trí *i* trong S_q sẽ tìm thấy với sự tạo thành tiền tố trung gian *(B)*. Sau đó, nút chữ ký của nút lá thứ *i* được truy cập từ gốc và tất cả các chữ ký với tiền tố *(B)* được truy vấn.

Ví dụ 10. Cho S_q = 011001000101. Để tìm tất cả các chữ ký khớp với S_q, cây *SD-tree* được xét kỹ từ gốc và các giá trị nút được so sánh với vị trí các bit của S_q. Sự xuất hiện cuối cùng của 1 trong S_q là ở vị trí 12. Tiền tố nhị phân được tạo thành cho S_q bằng cách sử dụng vị trí của bit 1 là 01100100010. Một nút với giá trị khóa là 12 được truy cập trong danh sách chữ ký lưu trữ bit 1 dưới dạng gom cụm và tất cả chữ ký trong danh sách chữ ký được kiểm tra giá trị tiền tố 01100100010. Do đó, không kể tới mẫu bit của S_q, tất cả các chữ ký trùng khớp đều được trả về trong một truy cập đơn.

Tương tự, chúng tôi sử dụng *SD-tree* để mô tả bit 0 dưới dạng gom cụm. Điều này giúp cải thiện thời gian tìm kiếm chữ ký có trọng lượng trên 50%. Bây giờ, giả sử rằng S_q= 011101110101. Sự xuất hiện cuối cùng của bit 0 là ở vị trí 11. Như vậy nút có giá trị là 11 sẽ được tìm kiếm trong danh sách chữ ký lưu trữ bit 0 dưới dạng gom cụm.

2. Thuật toán truy vấn trên cây *SD-tree*

Thuật toán sau đây phác thảo các bước tìm kiếm chữ ký khớp với chữ ký truy vấn cho trước S_q. Trong quá trình F← 0 các thuật toán đều đến trực tiếp các nút chữ ký hồi đáp lại 1 cuối cùng từ gốc.

Thuật toán 2. [14] Tìm kiếm(S_q)

Vào: Chữ ký truy vấn.

Ra: Danh sách các chữ ký khớp với chữ ký cho sẵn.

Phương pháp:

Bước 1. Tính trọng lượng chữ ký cho chữ ký truy vấn.

Bước 2. Nếu trọng lượng chữ ký lớn hơn 50% thì tìm kiếm chữ ký truy vấn trong các nút lá cho các bit 0.

Bước 3. Nếu không thì tìm kiếm chữ ký truy vấn trong các nút lá cho các bit 1.

Bước 4. Truy cập nút lá.

Bước 5. So sánh tiền tố của S_q.

Bước 6. Nếu tìm thấy thì đọc và xuất ra danh sách các chữ ký.

Bước 7. Nếu không thì báo cáo không có chữ ký phù hợp.

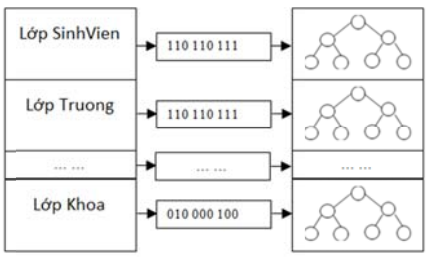
IV. PHƯƠNG PHÁP KẾT HỢP PHÂN CẤP TẬP TIN CHỮ KÝ VÀ CÂY SD-TREE

A. Mô hình cấu trúc dữ liệu truy vấn

Để cải thiện thời gian truy vấn trên cơ sở dữ liệu, cần mô tả lại hệ thống dữ liệu đơn giản hơn và xây dựng cấu trúc dữ liệu tương ứng để có thể giảm không gian tìm kiếm trong quá trình thực thi câu truy vấn mà vẫn đảm bảo được việc truy vấn được các đối tượng cần thiết bằng cách sử dụng cây chữ ký. Từ [4], để vấn đề truy vấn được tối ưu hơn cần kết hợp phân cấp tập tin chữ ký và cây chữ ký, vấn đề này đã được chứng minh là cải thiện thời gian truy vấn tốt hơn. Từ [13], độ phức tạp thời gian truy vấn trên cây *SD-tree* nhỏ hơn rất nhiều so với độ phức tạp thời gian truy vấn trên cây chữ ký. Do đó, chúng tôi vẫn sử dụng phân cấp tập tin chữ ký như trong [4], nhưng thay thế cây chữ ký bằng cây *SD-tree* để cải thiện thời gian truy vấn tốt hơn nữa.

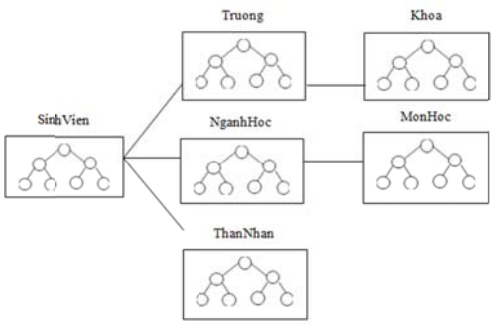
Dựa trên cơ sở lý thuyết và thuật toán đã đề nghị như trên, bài báo đề xuất phương pháp kết hợp phân cấp tập tin chữ ký và cây *SD-tree* như sau: (1) Mỗi tập tin chữ ký được lưu trữ dưới dạng cấu trúc cây *SD-tree* để tăng tốc quá trình quét tập tin chữ ký; (2) Tất cả các tập tin chữ ký được tổ chức theo cấu trúc phân cấp để tạo thuật lợi cho việc thực hiện kỹ thuật lọc theo từng bước. Trong mô hình này, thuật toán 1 nhằm xác định chữ ký thuộc lớp nào, thuật toán 2 xác định chính xác chữ ký trong lớp cụ thể.

Ví dụ 11. Tạo cây *SD-tree* được minh họa như sau:



Hình 7. Tạo cây *SD-tree*

Ví dụ 12. Kết hợp phân cấp tập tin chữ ký và cây *SD-tree* được minh họa như sau:



Hình 8. Phân cấp tập tin chữ ký và cây *SD-tree*

Cấu trúc dữ liệu được lưu trữ hoàn toàn bên trong bộ nhớ chính, trong trường hợp này, việc chèn và xóa một chữ ký trên cây *SD-tree* được thực hiện dễ dàng. Tuy nhiên, trong cơ sở dữ liệu các tập tin thường rất lớn, vì vậy cấu trúc dữ liệu sẽ không thể lưu trữ trên bộ nhớ chính mà phải được lưu trữ trên bộ nhớ ngoài. Đối với cơ sở dữ liệu hướng đối tượng, được lưu trữ và thực thi trên bộ nhớ ngoài. Cơ sở dữ liệu hướng đối tượng có nhiều lớp, mỗi lớp có nhiều đối tượng. Ứng với mỗi lớp sẽ được xây dựng thành một cấu trúc cây *SD-tree*, đồng thời mỗi đối tượng này sẽ

tạo ra một chữ ký đối tượng. Toàn bộ cơ sở dữ liệu hướng đối tượng sẽ được phân hoạch dưới dạng cấu trúc một bảng băm gồm các chữ ký của đối tượng để thực hiện quá trình truy vấn.

B. Xử lý truy vấn hướng đối tượng

Để thực hiện việc truy vấn một đối tượng trong cơ sở dữ liệu hướng đối tượng, đầu tiên phải chuyển đổi cơ sở dữ liệu hướng đối tượng thành cấu trúc dữ liệu như trên, ta thực hiện như sau:

Bước 1. Thuộc tính của đối tượng được băm thành chữ ký nhị phân và các thuộc tính tạo thành chữ ký đối tượng.

Bước 2. Các chữ ký đối tượng của cùng một lớp sẽ tạo thành cây *SD-tree*.

Bước 3. Tạo phân cấp tập tin chữ ký trong đó mỗi tập tin là cây *SD-tree*.

Sau khi có cấu trúc dữ liệu để truy vấn, ta thực hiện quá trình truy vấn đối tượng trong cơ sở dữ liệu hướng đối tượng như sau:

Bước 1. Mã hoá từ khóa cần truy vấn thành chữ ký nhị phân.

Bước 2. Thực hiện truy vấn chữ ký từ khóa để xác định thuộc lớp cần truy vấn.

Bước 3. Thực hiện truy vấn chữ ký từ khóa trên cây *SD-tree* tương ứng với các lớp đã xác định.

C. Đánh giá độ phức tạp

1. So sánh tìm kiếm theo phương pháp Yong và phân cấp tập tin chữ ký

Để ước tính số đối tượng được truy cập trong một truy vấn sử dụng hai phương pháp khác nhau: (1) phương pháp Yong được đề xuất trong [15]; (2) tìm kiếm phân cấp theo kiểu trên xuống [4].

(1) *Phương pháp Yong*

Phương pháp Yong, chữ ký của một đối tượng bị tham chiếu sẽ được lưu trữ trong đối tượng tham chiếu. Sau đó, có thể tiến hành kiểm tra thuộc tính đối với các chữ ký của chúng trước khi truy cập. Bằng cách này sẽ tiết kiệm được rất nhiều phép toán I/O.

(2) *Tìm kiếm phân cấp theo kiểu trên xuống*

Phương pháp này, có khả năng lọc mạnh mẽ hơn so với phương pháp Yong. Điều này là do mỗi sự kiểm tra đối với một nút trong một phân cấp chữ ký truy vấn, không chỉ thuộc tính liên quan đến nút hiện hành mà còn có một số thuộc tính khác các ảnh hưởng của chúng sẽ được đưa lên một số đường dẫn tới nút đó. Sử dụng phân cấp chữ ký truy vấn, có nhiều đối tượng trong lớp mục tiêu sẽ bị loại bỏ bằng cách kiểm tra tập tin chữ ký tương ứng, điều này dẫn tới giảm bớt đáng kể tổng số các đối tượng truy cập.

Trong [4], cho thấy rằng có thể đạt được hiệu suất cao bằng phương pháp tìm kiếm phân cấp trên xuống từ quan điểm trừu tượng, phân cấp chữ ký truy vấn là một bộ lọc "*toàn cục*" trong khi đó thì kỹ thuật nhân bản được Yong phát triển có thể coi là bộ lọc "*cục bộ*". Cả hai đều giúp giảm số đối tượng truy cập.

2. So sánh độ phức tạp thời gian của cây chữ ký và cây *SD-tree*

(i) *Phương pháp cây chữ ký*

Trong [13, 14], độ phức tạp thời gian chèn vào cây chữ ký là $O(nF)$, n là số chữ ký của tập tin và F là độ dài của chữ ký bao gồm bit 0 và bit 1. Với cây chữ ký, chiều cao của cây bị giới hạn là $O(log_2 n)$, n là số nút lá. Chi phí tìm kiếm cây chữ ký trung bình là $O(\lambda.log_2 n)$, trong đó λ là số đường đã đi qua.

(ii) *Phương pháp cây SD-tree*

Trong [13, 14], cây *SD-tree* sử dụng như cấu trúc chỉ mục cho tập hợp các dữ liệu lớn, giá trị F nhỏ thì sẽ giảm thời gian xây dựng *SD-tree*. Độ phức tạp thời gian chèn bị giới hạn là $O(n.m)$ trong đó n là số chữ ký trong tập tin và m là số bit 1 trong chữ ký cho trước. Một đặc tính hữu ích khác của *SD-tree* là với giá trị F lớn hơn, bằng cách biến đổi p, giá trị h, chiều cao của cây có thể được giữ ở mức nhỏ để thúc đẩy tìm kiếm nhanh hơn được giới hạn là $O(log_p(F/p-1))$. Thời gian tìm kiếm cho một truy vấn với tập hợp bit ở vị trí thứ i sau cùng là tổng thời gian truy cập nút lá (T_{li}) và thời gian tìm kiếm nút chữ ký (T_{si}) được tính như sau:

$$T_s = T_{li} + T_{si}.$$

Trong đó T_{li} không thay đổi cho tất cả các nút lá cho một cấu trúc cân bằng động như *SD-tree* và T_{si} tăng khi giá trị của i tăng. Do vậy, thời gian tìm kiếm bị giới hạn là $O(T_{li}+2^{i-1})$.

So sánh độ phức tạp thời gian tìm kiếm của cây chữ ký là $O(\lambda.log_2 n)$ và cây *SD-tree* là $O(T_{li}+2^{i-1})$, rõ ràng là giá trị T_{li} rất nhỏ so với giá trị λ, đó là một lợi thế của cây *SD-tree*.

V. KẾT LUẬN

Trong bài báo này, chúng tôi đã đề xuất một kỹ thuật lập chỉ mục mới. Phương pháp tiếp cận này là kết hợp phân cấp tập tin chữ ký với cây *SD-Tree*. Để tối ưu hóa việc quét phân cấp đối tượng, dựa trên các phân cấp tập tin chữ ký để giảm bớt số nhánh cây. Tuy nhiên, do tập tin chữ ký chỉ làm việc với tư cách là bộ lọc phi chính xác, nó không thể bị sắp xếp hay tìm kiếm nhị phân nên không thể được dùng để tăng tốc quá trình quét tập tin chữ ký. Do đó, chúng tôi đề xuất xây dựng một cây *SD-Tree* trên tập tin chữ ký xuất hiện với tư cách là một nút trong phân cấp

tập tin chữ ký. Kỹ thuật này có thể tránh phải tìm kiếm tuần tự giúp giảm đang kể thời gian cần thiết để tìm kiếm trên tập tin chữ ký.

VI. TÀI LIỆU THAM KHẢO

[1] Bertino, "Optimization of queries using nested indices", in Proceedings of International Conference on Extending Database Technology, 1990, pp. 44-59.

[2] Bertino and C. Guglielmani, "Optimization of object-oriented queries using path indices", in 2nd International Workshop on Research Issues on Data Engineering: Transaction and Query Processing, 1992, pp. 140-149.

[3] S. Choenni, E. Bertino, H. M. Blanken,and T. Chang, "On the selection of optimal index configuration in OO databases", in Proceedings of 10th International Conference on Data Engineering, 1994, pp. 526-537.

[4] Yangjun Chen, "Building Signature Trees into OODBs", Journal of Information Science and Engineering, 20(2), 2004, 275-304.

[5] Dervos, Y. Manolopoulos, and P. Linardis, "Comparison of signature file models with superimposed coding", Journal of Information Processing Letters, Vol. 65, 1998, pp. 101-106.

[6] R. Elmasri and S. B. Navathe, Fundamentals of Database Systems, Benjamin Cumming, California, 1989.

[7] Fotouhi, T. G. Lee, and W. I. Grosky, "The generalized index model for object-oriented database systems", in 10th Annual International Phonix Conference on Computers and Communication, 1991, pp. 302-308.

[8] Y. Ishikawa, H. Kitagawa, and N. Ohbo, "Evaluation of signature files as set access facilities in OODBs", in Procreedings of ACM SIGMOD International Conference on Management of Data, 1993, pp. 247-256.

[9] W. Kim, K. C. Kim, and A. Dale, "Indexing Techniques for Object Oriented Databases", Addison Wesley, 1989, pp. 371-394.

[10] Kemper and G. Moerkotte, "Access support relations: an indexing method for object bases", Information Systems, Vol. 17, 1992, pp. 117-145.

[11] C. C. Low, B. C. Ooi, and H. Lu, "H-trees: a dynamic associative search index for OODB", in Proceedings of 1992 ACM SIGMOD Conference on the Management of Data, 1992, pp. 134-143.

[12] Sreenath and S. Seshadri, "The hcC-tree: an efficient index structure for object oriented database", in Proceedings of International Conference on Very Large Database, 1994, pp. 203-213.

[13] I.E. Shanthi, R. Nadarajan, "Applying SD-Tree for Object-Oriented Query Processing", Informatica (Slovenia), 33(2), 2009, 169-179.

[14] Ms. Ankita Thakur, Ms. Meena Chauhan, "Optimizing Search for Fast Query Retrieval in Object Oriented Databases Using Signature Declustering", International Journal of Engineering Research and Development, 2012, pp. 46-50.

[15] S. Yong, S. Lee, and H. J. Kim,"Applying signatures for forward traversal query processing in object-oriented databases", in Proceedings of 10th International Conference on Data Engineering, 1994, pp. 518-525.

OBJECT-ORIENTED QUERIES BASE ON SIGNATURE FILE HIERARCHY AND SD-TREE

Tran Minh Bao, Truong Cong Tuan

ABSTRACT *– Direct query on objects in object-oriented database is time-consuming. Recently there are many studies focusing on resolving the problem by indexing on single class, class hierarchy or nested objects hierarchy. In this paper, we propose a new indexing approach by combining the signature files and SD-tree where signature files are organized in a hierarchy to quickly filter irrelevant data and each signature file is stored in a signature SD-tree to speed up signature scanning. This technique helps to reduce searching space, hence improves significantly time complexity of query.*