GIỚI THIỆU CẤU TRÚC CHỈ MỤC GR-TREE VÀ 4-R ĐỐI VỚI DỮ LIỆU THEO HAI LOẠI THỜI GIAN

Ngô Quỳnh Như, Hoàng Quang
Trường Đại học Khoa học, Đại học Huế

TÓM TẮT

Các cơ sở dữ liệu theo hai loại thời gian về cơ bản chỉ cho phép bổ sung nên chúng thường có kích cỡ rất lớn. Mặt khác, chúng thường chứa một phần đáng kể dữ liệu theo hai loại thời gian động nên vấn đề xử lý và tìm kiếm dữ liệu lại càng phức tạp và tốn nhiều thời gian. Một trong số các phương pháp giúp truy cập hiệu quả loại dữ liệu này là bổ sung một cấu trúc truy xuất phụ, được gọi là chỉ mục. Bài báo này giới thiệu hai cấu trúc chỉ mục hiệu quả nhằm cho phép truy lục dữ liệu theo hai loại thời gian, đặc biệt là thời gian động, đó là GR-tree và 4-R.

1. Giới thiệu

Thời gian là một thuộc tính của các hiện tượng trong thế giới thực. Vì vậy, phần lớn các ứng dụng cơ sở dữ liệu (CSDL) hiện nay đều quản lý dữ liệu thay đổi theo thời gian. Có hai loại thời gian thường được quan tâm là thời gian hợp lệ (Valid Time) và thời gian giao tác (Transaction Time). Thời gian hợp lệ của một sự kiện là thời gian khi sự kiện đó xảy ra đúng trong thực tế, trong khi thời gian giao tác là thời gian lúc sự kiện đó được lưu trữ trong CSDL. Dữ liệu hỗ trợ cả hai loại thời gian trên được gọi là dữ liệu theo hai loại thời gian.

Bài báo tập trung trình bày hai cấu trúc chỉ mục hỗ trợ dữ liệu theo hai loại thời gian động (thời gian thay đổi theo thời gian hiện tại) đó là GR-tree [1], [5] và 4-R [2], [5]. Bằng cách sử dụng các biến NOW và UC, GR-tree có thể mã hóa chính xác hình dạng các vùng theo hai loại thời gian trong các nút lá và các vùng giới hạn cực tiểu trong các nút nhánh. Các vùng giới hạn này tăng trưởng khi các vùng bên trong chúng tăng trưởng. Tuy nhiên, để cài đặt được GR-tree thì cần phải can thiệp vào nhân của hệ quản trị cơ sở dữ liệu (DBMS). Để khắc phục hạn chế trên, chỉ mục 4-R được đề xuất. Kỹ thuật sử dụng trong chỉ mục này là chuyển đổi dữ liệu theo hai loại thời gian động thành dữ liệu theo hai loại thời gian tĩnh, sau đó dùng chỉ mục sẵn có cho dữ liệu đã được chuyển đổi. Các truy vấn trên dữ liệu ban đầu cũng được chuyển đổi thành các truy vấn trên dữ liệu đã được chuyển đổi tương ứng.

Theo đó, trong phần 2, chúng tôi trình bày sự kết hợp giữa thời gian với dữ liệu

theo hai loại thời gian bằng cách sử dụng các vùng hai chiều. Các chỉ mục GR-tree và 4-R lần lượt được trình bày trong phần 3 và 4. Cuối cùng là phần kết luận.

2. Biểu diễn dữ liệu theo hai loại thời gian

Theo định dạng bốn nhãn thời gian của TQuel, mỗi bộ của một quan hệ theo hai loại thời gian có một số thuộc tính phi thời gian và bốn thuộc tính thời gian: VTbegin (thời gian bắt đầu hợp lệ), VTend (thời gian kết thúc hợp lệ), TTbegin (thời gian bắt đầu giao tác) và TTend (thời gian kết thúc giao tác). Chẳng hạn, xét quan hệ thời gian NV_PB được cho ở Bảng 1.

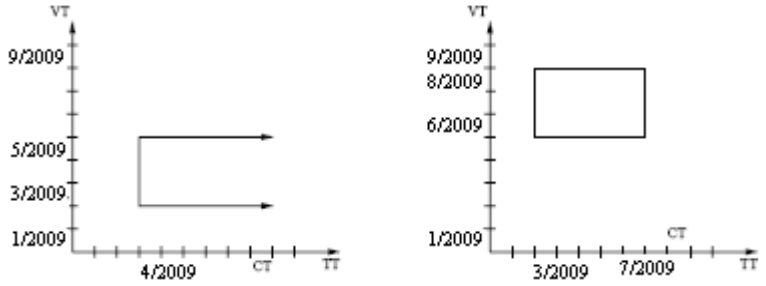
Bảng 1. *Quan hệ NV_PB*

	NV	PB	TTbeginn	TTend	VTbegin	VTend
(1)	Hùng	Quảng cáo	4/2009	UC	3/2009	5/2009
(2)	Nam	Quản lý	3/2009	7/2009	6/2009	8/2009
(3)	Mai	Kinh doanh	5/2009	UC	5/2009	NOW
(4)	Hoa	Kinh doanh	3/2009	7/2009	3/2009	NOW

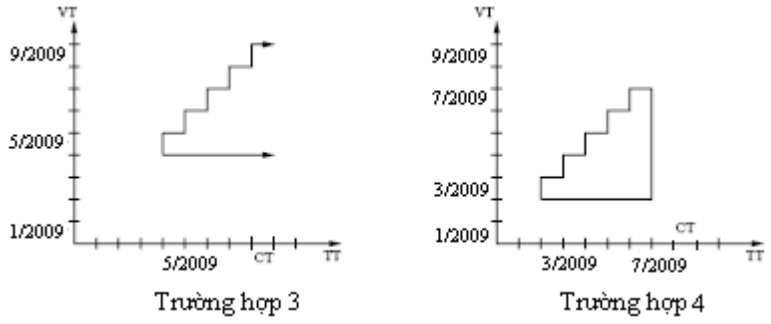
Hai biến UC (Until Changed) và NOW dùng để biểu diễn thời gian hiện hành, biến UC dùng cho TTend và biến NOW dùng cho VTend. Trong quan hệ trên, độ chi tiết thời gian là tháng và thời gian hiện hành (Current Time) là 9/2009.

Bộ (1) ghi nhận thông tin "Hùng làm việc ở bộ phận Quảng cáo" đúng từ 3/2009 đến 5/2009 và điều này được lưu trong CSDL từ 4/2009 và vẫn còn hiện hành. Bộ (3) ghi rằng "Mai làm việc ở bộ phận Kinh doanh" từ 5/2009 cho đến nay (NOW), điều này được lưu lại từ 5/2009 và nó vẫn còn là một phần của trạng thái CSDL hiện hành. Khi xóa hoặc sửa đổi một bộ hiện hành, giá trị UC của thuộc tính TTend sẽ thay đổi thành giá trị cố định khiến cho bộ đó không còn hiện hành nữa. Chẳng hạn bộ (2) đã bị xóa logic.

Mỗi bộ có thể được biểu diễn bởi một vùng theo hai loại thời gian trên một hệ trục tọa độ gồm hai chiều: thời gian giao tác (TT) và thời gian hợp lệ (VT). Chẳng hạn, các bộ (1) đến (4) lần lượt tương ứng với các trường hợp từ 1 đến 4 trong Hình 1.



Trường hợp 1 Trường hợp 2



Hình 1. *Các vùng theo hai loại thời gian*

Một khoảng thời gian giao tác hiện hành cung cấp một hình chữ nhật "tăng trưởng" theo chiều thời gian giao tác (trường hợp 1). Nếu khoảng thời gian hợp lệ và giao tác đều là hiện hành thì được biểu diễn bằng một vùng có dạng bậc thang tăng trưởng theo cả hai chiều thời gian trên (trường hợp 3). Đến một lúc nào đó, nếu một bộ không còn là hiện hành nữa thì vùng theo hai loại thời gian ngừng tăng trưởng (trường hợp 2, 4).

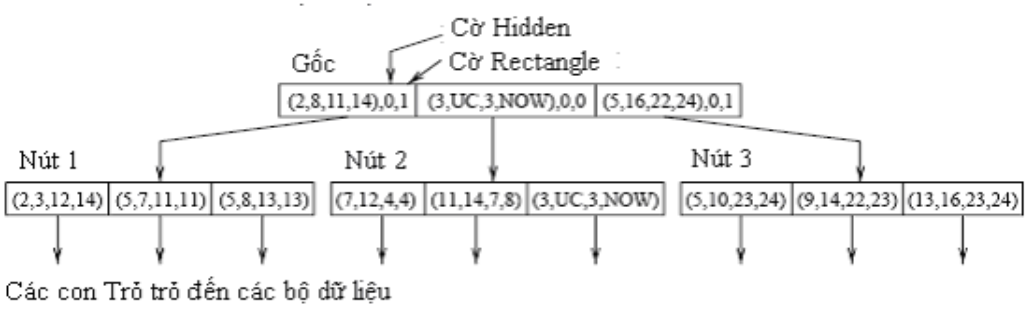
3. GR-TREE

Trong phần này, trước tiên chúng tôi trình bày cấu trúc của GR-tree, một phiên bản mở rộng của R*-tree [4]. Trên cơ sở đó để xây dựng thuật toán bổ sung một đầu vào mới cho cấu trúc GR-tree.

3.1. Cấu trúc của GR-tree

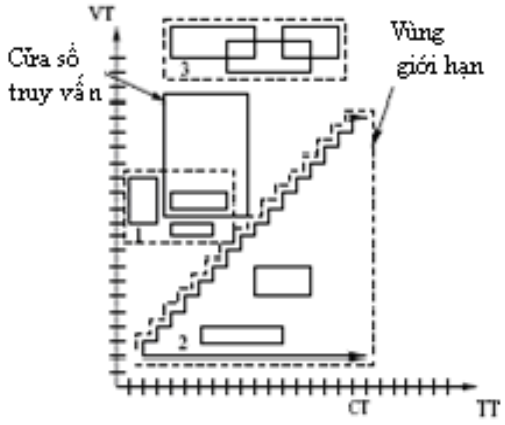
Bằng cách sử dụng biến NOW và UC, GR-tree có thể mã hóa hình dạng chính xác các vùng theo hai loại thời gian (phần 2). Với sự mở rộng này, mỗi đầu vào của một nút lá chứa 4 nhãn thời gian biểu diễn một vùng theo hai loại thời gian và một con trỏ trỏ tới dữ liệu theo hai loại thời gian thực sự được lưu trữ trong CSDL. Mỗi đầu vào của một nút nhánh chứa bốn nhãn thời gian biểu diễn một vùng giới hạn cực tiểu bao quanh các đầu vào của nút con, một cờ Hidden, một cờ Rectangle và một con trỏ trỏ đến nút con đó.

Hình 2 là ví dụ về một cấu trúc GR-tree đối với một quan hệ theo hai loại thời gian gồm 9 bộ.



Hình 2. *Ví dụ về cấu trúc GR-tree [1]*

Cờ Rectangle nhằm chỉ ra bốn nhãn thời gian mã hóa một hình chữ nhật giới hạn cực tiểu hay một hình bậc thang giới hạn cực tiểu trong trường hợp giá trị VTend là NOW và giá trị TTend là UC, nghĩa là cờ Rectangle là cần thiết để phân biệt giữa việc biểu diễn một hình bậc thang tăng trưởng với một hình chữ nhật tăng trưởng. Hình 3 biểu diễn các vùng theo hai loại thời gian của cây GR-tree tương ứng trong hình 2. Khi đó, nếu muốn tìm tất cả các vùng chồng lấn với cửa sổ truy vấn đã cho trong hình 3 thì chỉ cần truy lục nút 1.



Hình 3. *Biểu diễn bằng hình ảnh các vùng theo hai loại thời gian [1]*

Một hình bậc thang tăng trưởng nhỏ có thể được đặt cùng với các vùng khác trong một hình chữ nhật giới hạn lớn hơn có điểm kết thúc thời gian hợp lệ cố định (lớn hơn thời gian hiện hành). Đến một lúc nào đó, hình bậc thang sẽ vượt quá hình chữ nhật giới hạn của nó khiến cho hình chữ nhật này không còn hợp lệ. Cờ Hidden được sử dụng để xử lý các hình bậc thang như thế.

Lưu ý rằng nếu gọi M là số đầu vào tối đa mà một nút có thể chứa và m là số đầu vào tối thiểu phải có trong bất kỳ một nút không phải nút gốc. Khi đó chúng ta có $m \leq M/2$. Ngoài ra, GR-tree còn là cây cân bằng.

Để phục vụ cho việc tạo lập một cấu trúc GR-tree, cũng như việc bổ sung dữ liệu sau này, chúng ta cần quan tâm đến thuật toán bổ sung một đầu vào cho cấu trúc đó.

3.2. Thuật toán bổ sung một đầu vào cho cấu trúc GR-tree

Như chúng ta đã biết, thuật toán bổ sung (chèn) một đầu vào mới của R*-tree [4] được thiết kế cho các hình chữ nhật tĩnh. Trong khi đó, đối với GR-tree, các vùng giới hạn cực tiểu không chỉ là các vùng tĩnh mà có thể là các vùng tăng trưởng theo thời gian nên các tiêu chuẩn như: chọn một nút lá để chèn đầu vào mới (*ChooseSubtreee*), chọn p đầu vào để xóa (*RemoveTop*), phân tách một nút thành hai nút (*Split*) phải được cải tiến để có thể áp dụng cho GR-tree.

Thuật toán chèn của R*-tree trước tiên gọi thuật toán *ChooseSubtree* nhằm tìm

một nút lá thích hợp để đặt một đầu vào mới. Nếu nút được chọn đã chứa tối đa M đầu vào thì thuật toán *OverflowTreatment* sẽ được gọi. Nếu trong suốt quá trình chèn đầu vào mới, đây là lần gọi đầu tiên thuật toán *OverflowTreatment* tại mức đã cho của cây thì thuật toán *RemoveTop* cũng được gọi; trái lại thì thuật toán *Split* sẽ được gọi.

Thuật toán *RemoveTop* của R*-tree thực hiện việc xóa p đầu vào của nút được chọn và chèn lại chúng. Trong trường hợp xấu nhất, tất cả các đầu vào đó được chèn lại vào cùng một nút hoặc chúng có thể làm tràn một số nút khác thì thuật toán *OverflowTreatment* được gọi lại, và bây giờ nó sẽ gọi thuật toán *Split*. Việc phân tách một nút có thể gây ra quá tải tại nút cha. Nếu điều này xảy ra, thuật toán *OverflowTreatment* được áp dụng cho nút cha.

Từ ý tưởng trên, các thuật toán cải tiến sử dụng bốn kiểu vùng giới hạn theo hai loại thời gian cho các nút nhánh như sau:

Hình chữ nhật tĩnh và hình bậc thang tĩnh

Hình chữ nhật tăng trưởng theo một chiều (sử dụng biến UC)

Hình bậc thang tăng trưởng (sử dụng các biến UC và NOW, giá trị cờ Rectangle bằng False)

Hình chữ nhật tăng trưởng theo cả hai chiều (sử dụng các biến UC và NOW, giá trị cờ Rectangle bằng True).

Để không gian chồng lấn giữa các vùng giới hạn cực tiểu của các nút tại mỗi mức của cây và không gian không dùng được trong vùng giới hạn của mỗi nút (không gian được bao phủ bởi vùng giới hạn đó nhưng không được bao phủ bởi bất kỳ vùng giới hạn nào bên trong nó) là nhỏ, ta cần đạt được các kiểu nút nhánh theo thứ tự ưu tiên trên. Dựa trên sự ưu tiên này, một quy tắc heuristic tổng quát được áp dụng là: nhóm các đầu vào thành các nút sao cho kiểu nút thu được là tốt nhất, đồng thời giữ số lượng nút có kiểu không tốt ở mức thấp nhất có thể.

Cụ thể, quy tắc heuristic trên được áp dụng cho các thuật toán như sau.

Thuật toán *ChooseSubtree.* Thuật toán này xem xét kiểu của đầu vào mới và kiểu của nút mà đầu vào mới có thể được chèn vào. Nó sẽ chọn nhóm gồm các nút có cùng kiểu sao cho khi đầu vào mới được chèn vào bất kỳ nút nào của nhóm thì kiểu của nút đó vẫn như cũ. Nếu một số nhóm gồm các nút có kiểu khác nhau thỏa mãn điều kiện trên thì nhóm gồm các nút có kiểu tốt nhất sẽ được chọn. Nếu không có nhóm nào thỏa điều kiện trên, thì đầu vào mới sẽ làm cho nút được chọn có kiểu xấu hơn trước. Trong trường hợp này, thuật toán chọn một nhóm gồm các nút cùng kiểu sao cho khi chèn đầu vào mới vào bất kỳ nút nào của nhóm thì kiểu của nút trở nên xấu hơn là ít nhất. Nếu có nhiều nhóm như vậy thì thuật toán sẽ chọn nhóm gồm các nút có kiểu xấu nhất. Sau đó, chuyển nhóm này cho thuật toán *ChooseSubtree* của R*-tree, thuật toán này đưa ra quyết định dựa trên sự mở rộng phạm vi chồng lấn và phạm vi bao phủ.

Thuật toán *Split.* Có hai hướng tiếp cận:

Thứ nhất, thuật toán Split sẽ được cải tiến bằng cách xem xét các kiểu khác nhau của các đầu vào để phân tách chúng thành hai nút sao cho mỗi nút có thể được giới hạn bởi một vùng có kiểu tốt nhất đồng thời hạn chế phân bố các đầu vào cùng kiểu vào hai nút khác nhau. Hai nút được tạo ra sau khi phân tách có thể thuộc một trong bốn kiểu đã nêu, có mười khả năng kết hợp kiểu của hai vùng giới tương ứng với hai nút. Chúng ta ưu tiên các cặp đó theo mức độ tốt của chúng (Hình 4).

Hình 4. *Các cặp kiểu vùng giới hạn*

Một cặp vùng giới hạn x_1 và x_2 được xem là tốt hơn cặp vùng giới hạn y_1 và y_2 nếu:

$(type(x_1) \neq type(y_1) \vee type(x_2) \neq type(y_2)) \wedge ((type(x_1) \leq type(y_1) \wedge type(x_2) \leq type(y_2)) \vee (type(x_1) < \max(type(y_1), type(y_2)) \wedge type(x_2) < \max(type(y_1), type(y_2))))$

Thứ hai, thuật toán Split của R^*-tree có thể được sửa đổi nhằm kiểm tra thêm một số phân bố đầu vào bằng cách bổ sung hai cách sắp xếp. Cách thứ nhất, các đầu vào được sắp xếp theo giá trị VTend - TTbegin, giá trị này tỷ lệ với khoảng cách từ góc trên bên trái của vùng giới hạn (tương ứng với mỗi đầu vào) đến đường y = x. VTend được thiết lập bởi một giá trị cố định thích hợp nếu vùng giới hạn đó là một hình chữ nhật tăng trưởng có VTend là NOW. Đối với các hình bậc thang, giá trị 0 được sử dụng thay cho giá trị VTend - TTbegin bởi vì các bậc thang của vùng có dạng bậc thang luôn luôn nằm trên đường y = x. Tương tự, trong cách thứ hai, các đầu vào được sắp xếp theo giá trị VTbegin - TTend. Các cách sắp xếp mới có ưu điểm là tạo ra sự tách biệt giữa dạng chữ nhật và dạng bậc thang. Khi đó, khả năng phân bố các đầu vào có kiểu giống nhau vào cùng một nút sẽ cao hơn nên kiểu của hai nút thu được sau khi phân tách sẽ tốt hơn.

Thuật toán Split của R^*-tree có thể được sử dụng mà không cần sửa đổi đối với trường hợp tất cả các đầu vào là hình chữ nhật tĩnh hoặc hình bậc thang tĩnh.

Tuy GR-tree hỗ trợ hiệu quả dữ liệu theo hai loại thời gian nhưng các DBMS hiện có không hỗ trợ chỉ mục này. Hạn chế trên được khắc phục trong kỹ thuật chỉ mục sẽ trình bày ở phần tiếp theo.

4. Chỉ mục 4-R

Ý tưởng của kỹ thuật được sử dụng trong chỉ mục 4-R là áp dụng các chuyển đổi dữ liệu làm cho các vùng dữ liệu tăng trưởng liên tục theo hai loại thời gian trở nên tĩnh, sau đó lập chỉ mục cho các vùng dữ liệu tĩnh này bằng cách sử dụng bốn chỉ mục R^*-tree. Theo đó, các truy vấn trên dữ liệu ban đầu cũng được chuyển đổi thành các truy vấn trên dữ liệu đã được chuyển đổi.

Trong mục này, trước tiên chúng tôi trình bày việc chuyển đổi dữ liệu. Theo đó, việc chuyển đổi truy vấn sẽ được trình bày ở phần tiếp theo sau.

4.1. Chuyển đổi dữ liệu

Mục đích chính của việc chuyển đổi dữ liệu theo hai loại thời gian là khử các biến UC và NOW. Chúng ta phân biệt bốn loại dữ liệu theo hai loại thời gian phụ thuộc vào TTend và VTend lần lượt có bằng UC và NOW hay không.

Định nghĩa 1. Gọi miền giá trị của các nhãn thời gian là T và miền định danh của bộ là ID. $TT^{|-}$ và $TT^{-|}$ lần lượt kí hiệu cho TTbegin và TTend; $VT^{|-}$ và $VT^{-|}$ lần lượt kí hiệu cho VTbegin và VTend.

Miền giá trị của các bộ dữ liệu theo hai loại thời gian D^B và miền giá trị của các bộ dữ liệu theo hai loại thời gian tĩnh D^S được định nghĩa như sau:

$$D^B = \{< TT_r^{|-}, TT_r^{-|}, VT_r^{|-}, VT_r^{-|}, id_r > \in T \times T \cup \{UC\} \times T \times T \cup \{NOW\} \times ID |$$

$$(TT_r^{-|} = UC \vee TT_r^{|-} \le TT_r^{-|}) \wedge (VT_r^{-|} = NOW \vee VT_r^{|-} \le VT_r^{-|})\}$$

$$D^S = \{< TT_r^{|-}, TT_r^{-|}, VT_r^{|-}, VT_r^{-|}, id_r > \in T \times T \times T \times T \times ID \mid (TT_r^{|-} \le TT_r^{-|}) \wedge (VT_r^{|-} \le VT_r^{-|})\}$$

Lưu ý rằng chỉ số dưới "r" dùng để phân biệt hình chữ nhật dữ liệu với hình chữ nhật truy vấn được bàn đến sau này.

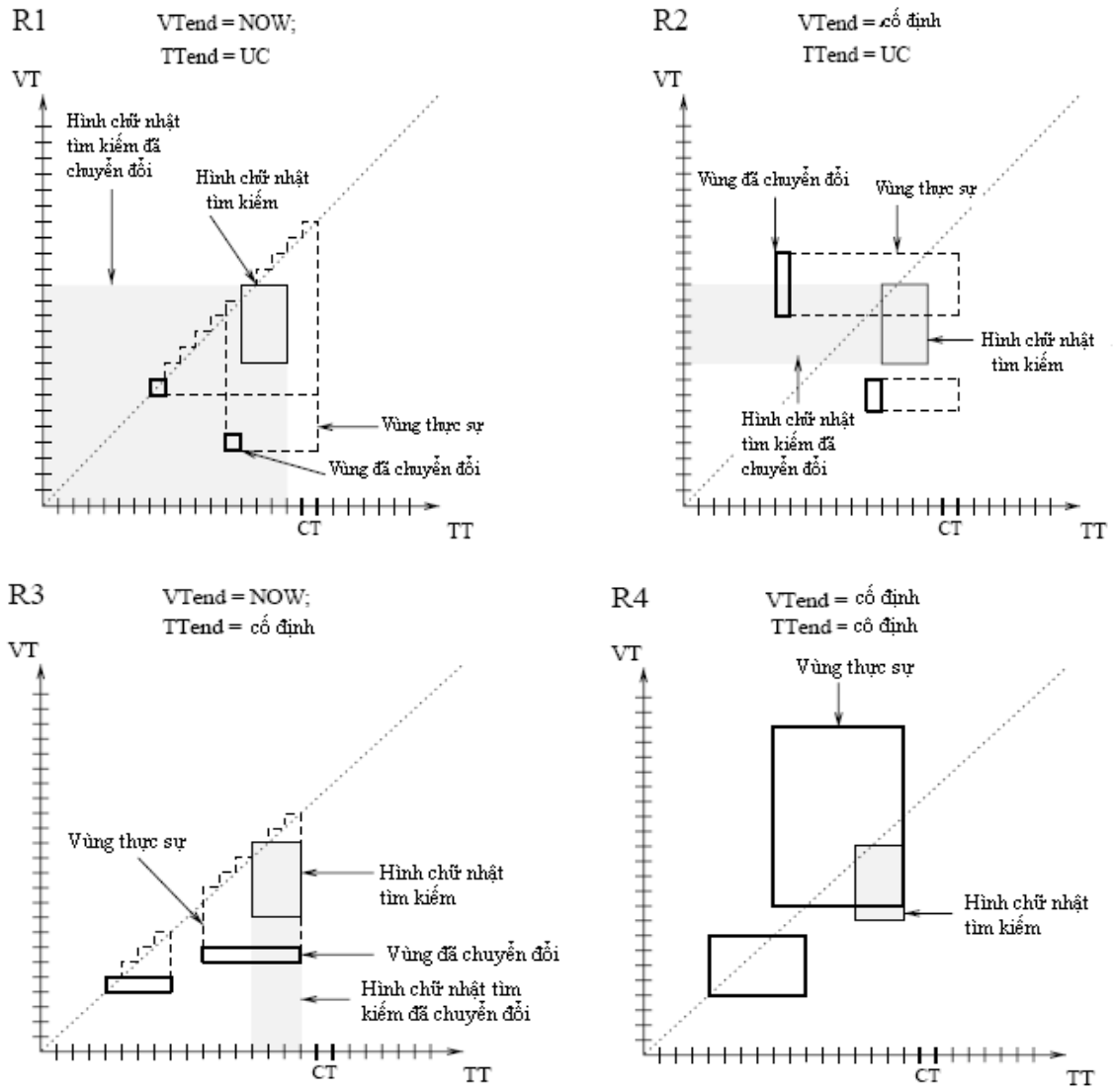
Việc chuyển đổi dữ liệu được định nghĩa sau đây nhằm xác định việc chuyển đổi một bộ theo hai loại thời gian nói chung thành một bộ theo hai loại thời gian tĩnh.

Định nghĩa 2. Cho $R \subseteq D^B$ và *Type* = {1, 2, 3, 4}. Khi đó phép chuyển đổi dữ liệu được định nghĩa như sau:

$$\tau_D(R) = \{\tau_r(\langle TT_r^{|-}, TT_r^{-|}, VT_r^{|-}, VT_r^{-|}, id_r \rangle) | \langle TT_r^{|-}, TT_r^{-|}, VT_r^{|-}, VT_r^{-|}, id_r \rangle \in R\}$$

Trong đó:

$$\tau_r(\langle TT_r^{\vdash}, TT_r^{\dashv}, VT_r^{\vdash}, VT_r^{\dashv}, id_r\rangle) = \begin{cases} \langle TT_r^{\vdash}, TT_r^{\vdash}, VT_r^{\vdash}, VT_r^{\vdash}, id_r, 1\rangle, khi: TT_r^{\dashv} = UC \wedge VT_r^{\dashv} = NOW \\ \langle TT_r^{\vdash}, TT_r^{\vdash}, VT_r^{\vdash}, VT_r^{\dashv}, id_r, 2\rangle, khi: TT_r^{\dashv} = UC \wedge VT_r^{\dashv} \neq NOW \\ \langle TT_r^{\vdash}, TT_r^{\dashv}, VT_r^{\vdash}, VT_r^{\vdash}, id_r, 3\rangle, khi: TT_r^{\dashv} \neq UC \wedge VT_r^{\dashv} = NOW \\ \langle TT_r^{\vdash}, TT_r^{\dashv}, VT_r^{\vdash}, VT_r^{\dashv}, id_r, 4\rangle, khi: TT_r^{\dashv} \neq UC \wedge VT_r^{\dashv} \neq NOW \end{cases}$$



Hình 5. *Lưu trữ dữ liệu và tìm kiếm trong bốn cây của chỉ mục 4-R*

Cây R1, tương ứng với trường hợp 1, nhằm lập chỉ mục cho các vùng mà trước khi chuyển đổi các vùng này có giá trị kết thúc của thời gian giao tác và hợp lệ đều không cố định. Do cả $VT^{\dashv}$ và $TT^{\dashv}$ đều gắn với thời gian hiện hành, nên để mã hóa các

vùng này chỉ cần lưu trữ một điểm ($TT^{|-}, VT^{|-}$) trong chỉ mục. Khi đó, để phù hợp với cấu trúc R-tree [3], các vùng sau khi chuyển đổi được biểu diễn bằng các điểm hai chiều $\langle TT_r^{|-}, TT_r^{|-}, VT_r^{|-}, VT_r^{|-} \rangle$.

Tương tự, cây R2, R3, R4 lần lượt tương ứng với trường hợp 2, 3, 4 trong Định nghĩa 2 được minh họa trong Hình 5.

Lưu ý rằng, cây R4 dành cho các vùng mà trước khi chuyển đổi có giá trị kết thúc thời gian giao tác và hợp lệ đều cố định. Chúng có dạng hình chữ nhật tĩnh nên không cần chuyển đổi và được lưu trữ trong chỉ mục dưới dạng $\langle TT_r^{|-}, TT_r^{-|}, VT_r^{|-}, VT_r^{-|} \rangle$

4.2. Chuyển đổi truy vấn

Chúng ta sẽ tìm hiểu kiểu truy vấn chỉ mục thông dụng nhất đối với dữ liệu theo hai loại thời gian được gọi là *truy vấn giao dạng chữ nhật*. Loại truy vấn này được biểu diễn bởi một hình chữ nhật tĩnh, được gọi là hình chữ nhật truy vấn. Gọi $\langle TT_q^{|-}, TT_q^{-|}, VT_q^{|-}, VT_q^{-|} \rangle \in T \times T \times T \times T$ kí hiệu cho hình chữ nhật truy vấn, trong đó T là miền giá trị của các nhãn thời gian. Giả sử: $TT_q^{|-} \le TT_q^{-|}; VT_q^{|-} \le VT_q^{-|}$ và $TT_q^{-|} \le CT$, trong đó CT là giá trị thời gian hiện hành. Ta có các định nghĩa sau:

Định nghĩa 3. Gọi r1 và r2 là hai vùng giới hạn chữ nhật tĩnh. Khi đó, giao của r1 và r2 được ký hiệu là:

$\langle TT_{r1}^{|-}, TT_{r1}^{-|}, VT_{r1}^{|-}, VT_{r1}^{-|} \rangle \cap \langle TT_{r2}^{|-}, TT_{r2}^{-|}, VT_{r2}^{|-}, VT_{r2}^{-|} \rangle$ nếu

$(TT_{r1}^{|-} \le TT_{r2}^{-|}) \wedge (TT_{r1}^{-|} \ge TT_{r2}^{|-}) \wedge (VT_{r1}^{|-} \le VT_{r2}^{-|}) \wedge (VT_{r1}^{-|} \ge VT_{r2}^{|-})$.

Ngoài ra, gọi $q = \langle TT_q^{|-}, TT_q^{-|}, VT_q^{|-}, VT_q^{-|} \rangle$ và R $\subseteq$ D^B. Khi đó, truy vấn giao dạng chữ nhật $Intersect^B$ trên R với các tham số là hình chữ nhật truy vấn q và giá trị thời gian hiện hành CT, được định nghĩa như sau:

$$Intersect^B[q, CT](R) =$$

$$\{id_r \mid \langle TT_r^{|-}, TT_r^{-|}, VT_r^{|-}, VT_r^{-|}, id_r \rangle \in R \wedge$$

$$(((TT_r^{-|} = UC) \wedge (VT_r^{-|} = NOW) \wedge (TT_q^{-|} \ge VT_q^{|-}) \wedge (q \cap \langle TT_r^{|-}, CT, VT_r^{|-}, CT \rangle)) \vee$$

$$((TT_r^{-|} = UC) \wedge (VT_r^{-|} \ne NOW) \wedge (q \cap \langle TT_r^{|-}, CT, VT_r^{|-}, VT_r^{-|} \rangle)) \vee$$

$$((TT_r^{-|} \ne UC) \wedge (VT_r^{-|} = NOW) \wedge (TT_q^{-|} \ge VT_q^{|-}) \wedge (q \cap \langle TT_r^{|-}, TT_r^{-|}, VT_r^{|-}, TT_r^{-|} \rangle)) \vee$$

$$((TT_r^{-|} \ne UC) \wedge (VT_r^{-|} \ne NOW) \wedge (q \cap \langle TT_r^{|-}, TT_r^{-|}, VT_r^{|-}, VT_r^{-|} \rangle))))\}$$

Điều kiện đầu tiên giới hạn các bộ kết quả là các bộ thuộc R.

Các bộ thỏa mãn một trong bốn điều kiện tiếp theo sẽ thỏa mãn truy vấn và thuộc tập kết quả. Điều kiện thứ nhất ứng với các vùng dạng bậc thang tăng trưởng, điều

kiện thứ hai ứng với các vùng dạng chữ nhật tăng trưởng, điều kiện thứ ba ứng với các vùng dạng bậc thang tĩnh và điều kiện thứ tư ứng với vùng dạng chữ nhật tĩnh. Trong trường hợp các vùng dạng bậc thang tăng trưởng hoặc tĩnh, ngoài việc kiểm tra sự giao nhau giữa các vùng này với hình chữ nhật truy vấn thì cần kiểm tra thêm điều kiện góc dưới bên phải của hình chữ nhật truy vấn có nằm bên dưới hoặc nằm ngay trên đường VT = TT hay không (tương ứng với điều kiện $TT_q^{\dashv} \geq VT_q^{\vdash}$).

Tương tự, chúng ta định nghĩa truy vấn giao dạng chữ nhật đối với dữ liệu theo hai loại thời gian tĩnh. Kết quả của truy vấn này là độc lập với thời gian hiện hành.

Định nghĩa 4. Gọi $q = \langle TT_q^{\vdash}, TT_q^{\dashv}, VT_q^{\vdash}, VT_q^{\dashv} \rangle$ và S $\subseteq$ D^S. Khi đó, truy vấn giao dạng chữ nhật *Intersect*^{S} trên S với tham số là hình chữ nhật truy vấn q được định nghĩa như sau:

$$Inter\sec t^S[q](S) = \{id_r \mid \langle TT_r^{\vdash}, TT_r^{\dashv}, VT_r^{\vdash}, VT_r^{\dashv}, id_r \rangle \in S \wedge q \cap \langle TT_r^{\vdash}, TT_r^{\dashv}, VT_r^{\vdash}, VT_r^{\dashv} \rangle\}$$

Từ Định nghĩa 3 và 4, chúng ta có thể định nghĩa ánh xạ chuyển đổi truy vấn 4-R để áp dụng với dữ liệu đã được chuyển đổi. Ánh xạ chuyển đổi truy vấn 4-R nhằm ánh xạ một truy vấn giao dạng chữ nhật trên dữ liệu ban đầu thành hai hoặc bốn truy vấn tương đương trên dữ liệu đã được chuyển đổi.

Định nghĩa 5. Cho:

$R \subseteq D^B$, $S = \tau_D(R)$,

$S_i = \{\langle TT_r^{\vdash}, TT_r^{\dashv}, VT_r^{\vdash}, VT_r^{\dashv}, id_r \rangle \mid \langle TT_r^{\vdash}, TT_r^{\dashv}, VT_r^{\vdash}, VT_r^{\dashv}, id_r, i \rangle \in S\}$ (với i = 1, 2, 3, 4),

$q = \langle TT_q^{\vdash}, TT_q^{\dashv}, VT_q^{\vdash}, VT_q^{\dashv} \rangle$,

$q_1 = \langle 0, TT_q^{\dashv}, 0, VT_q^{\dashv} \rangle$,

$q_2 = \langle 0, TT_q^{\dashv}, VT_q^{\vdash}, VT_q^{\dashv} \rangle$,

$q_3 = \langle \max(TT_q^{\vdash}, VT_q^{\vdash}), TT_q^{\dashv}, 0, VT_q^{\dashv} \rangle$,

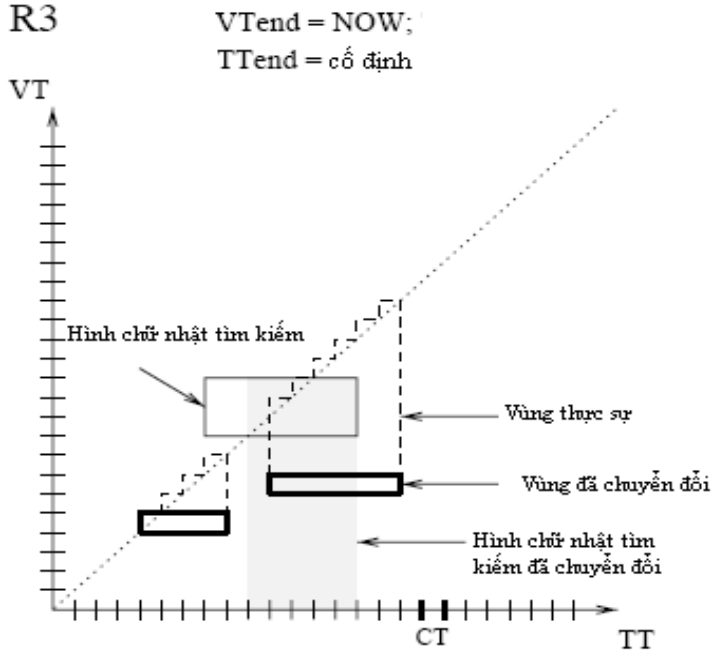
$q_4 = \langle TT_q^{\vdash}, TT_q^{\dashv}, VT_q^{\vdash}, VT_q^{\dashv} \rangle$,

Khi đó, ánh xạ chuyển đổi truy vấn 4-R, $\tau_q : [2^{D^B} \to 2^{ID}] \to [2^{D^S \times Type} \to 2^{ID}]$

được định nghĩa như sau:

$$\tau_q(Inter\sec t^B[q, CT])(S) = \begin{cases} \bigcup_{i=1,2,3,4} Inter\sec t^S[q_i](S_i), khi: TT_q^{\dashv} \geq VT_q^{\vdash} \\ \bigcup_{i=2,4} Inter\sec t^S[q_i](S_i), khi: TT_q^{\dashv} < VT_q^{\vdash} \end{cases}$$

Hình chữ nhật tìm kiếm ban đầu (hình chữ nhật truy vấn q) và hình chữ nhật tìm kiếm đã chuyển đổi tương ứng được minh họa trong Hình 5 và Hình 6.



Hình 6. *Các đặc điểm tìm kiếm đối với cây R3*

Khi tìm kiếm trên cây R1, hình chữ nhật tìm kiếm được mở rộng để bao phủ hết không gian kéo dài từ điểm bắt đầu của thời gian hợp lệ và giao tác đến góc trên bên phải của hình chữ nhật tìm kiếm.

Khi tìm kiếm trên cây R2, chúng ta không chỉ tìm kiếm các vùng dữ liệu chồng lấn với hình chữ nhật tìm kiếm ban đầu mà còn tìm kiếm các vùng bên trái hình chữ nhật tìm kiếm đó (mở rộng tìm kiếm sang trái).

Việc chuyển đổi hình chữ nhật tìm kiếm đối với cây R3 cũng tương tự nhưng bây giờ hình chữ nhật tìm kiếm được mở rộng xuống phía dưới thay vì sang trái. Mặt khác, khi một phần hình chữ nhật tìm kiếm nằm dưới đường VT = TT và một phần khác nằm trên thì chỉ phần nằm dưới được mở rộng (xem Hình 6); Trong Định nghĩa 5, hàm "max" nhằm đảm bảo chỉ phần dưới của hình chữ nhật tìm kiếm được mở rộng.

Do cây R4 lập chỉ mục cho các hình chữ nhật theo hai loại thời gian là không chuyển đổi, nên việc chuyển đổi hình chữ nhật tìm kiếm là không cần thiết.

Sau đây là một số trường hợp đặc biệt cần chú ý khi truy vấn trên bốn cây này. Trong các cây R1 và R3, các điểm và các khoảng được lập chỉ mục chỉ nằm phía dưới đường VT = TT và các vùng (trước khi chuyển đổi) được mã hóa bởi các điểm hoặc các khoảng đó cũng không mở rộng trên đường này. Vì vậy, việc tìm kiếm trong hai cây đó chỉ được thực hiện khi hình chữ nhật tìm kiếm có ít nhất một phần nằm dưới đường VT = TT.

Trường hợp đặc biệt khác là truy vấn lát cắt thời gian giao tác tại thời điểm hiện hành ($TT_q^{|-} = TT_q^{-|} = CT$) có thể xảy ra thường xuyên trong thực tế. Dữ liệu hiện hành chỉ nằm trong R1 và R2, do đó, truy vấn loại này không cần thiết phải tìm kiếm trong R3 và R4. Ngoài ra, nếu một truy vấn lát cắt thời gian giao tác tại thời điểm hiện hành có thời gian hiệu lực phía trên đường VT = TT thì chỉ có cây R2 là được tìm kiếm. Trường hợp đặc biệt cuối cùng là nếu lát cắt thời gian giao tác tại thời điểm hiện hành không có các ràng buộc về thời gian hợp lệ thì tất cả các bộ theo hai loại thời gian được lập chỉ mục bởi các cây R1 và R2 đều thỏa mãn và không cần đến bất kỳ tìm kiếm nào.

Cải tiến quan trọng của chỉ mục 4-R là có thể sử dụng lại các chỉ mục hiệu quả dựa trên R-tree (ví dụ như R*-tree) mà không cần bất kỳ sự sửa đổi nào do chỉ có các điểm, các khoảng và các hình chữ nhật tĩnh được lập chỉ mục. Ngoài ra, vì các điểm và các khoảng chiếm ít không gian lưu trữ nên một nút có thể chứa nhiều đầu vào hơn và độ cao của cây sẽ thấp hơn. Đồng thời, số chiều của dữ liệu hiện hành sẽ ít hơn nên vùng chồng lấn giữa các hình chữ nhật giới hạn cực tiểu sẽ nhỏ hơn. Hơn nữa, do sự biểu diễn dữ liệu hiện hành không mở rộng tới thời gian hiện hành nên không gian không dùng được giảm. Tuy nhiên, 4-R cũng có một số hạn chế như: việc chuyển đổi truy vấn làm cho hình chữ nhật truy vấn luôn luôn bị mở rộng; các hình chữ nhật giới hạn cực tiểu của các nút trong R2 kéo dài theo chiều thời gian hợp lệ trong khi các truy vấn đã chuyển đổi trong R2 lại mở rộng khá dài theo chiều thời gian giao tác;…

5. Kết luận

Bài báo này đã trình bày hai chỉ mục hỗ trợ hiệu quả cho dữ liệu theo hai loại thời gian động, đó là GR-tree và 4-R. Có thể xem chỉ mục 4-R là một trường hợp đặc biệt của GR-tree. Thuật toán bổ sung đầu vào của GR-tree tách các kiểu vùng dữ liệu khác nhau vào các nút khác nhau nhằm thu được một cây gồm các nhóm nút chứa các vùng dữ liệu cùng loại. Vì vậy, chỉ mục 4-R là một trường hợp rất đặc biệt của GR-tree, trong đó các nhóm nút này tạo thành bốn cây khác nhau.

Tuy nhiên, cả hai cấu trúc GR-Tree và 4-R đều không đề cập tới việc phân vùng dữ liệu theo giá trị khóa phi thời gian mà chỉ dựa trên các giá trị thuộc tính thời gian nên có thể dẫn đến tình trạng các bản ghi dữ liệu có giá trị khóa phi thời gian gần nhau sẽ bị lưu trữ trong nhiều khối dữ liệu khác nhau. Khi đó, các truy vấn chỉ liên quan đến khóa phi thời gian sẽ được xử lý thiếu hiệu quả. Đây là vấn đề đáng được quan tâm và là một trong những hướng nghiên cứu của chúng tôi trong lĩnh vực này.

TÀI LIỆU THAM KHẢO

1. Rasa Bliujute, Christian S.Jensen, Simonas Saltenis, and Giedrius Slivinskas, *R-Tree Based Indexing of Now-Relative Bitemporal Data, Temporal Database Management*, 2000, 1161-1186.

2. Rasa Bliujute, Christian S.Jensen, Simonas Saltenis, and Giedrius Slivinskas, *Light-Weight Indexing of General Bitemporal Data*, *Temporal Database Management*, 2000, 1187-1217.

3. Antonm Guttman, *R-trees: A Dynamic Index Structure for Spatial Searching,* ACM SIGMOD Conference, Boston, Massachusetts, 1984.

4. N. Beckmann [et al], *The R*-tree: An Efficient and Robust Access Method for Points and Rectangles,* ACM SIGMOD Conference, Atlantic City, NJ, 1990.

5. Stantic B., *Access Methods for Temporal Databases*, Bela Stantic's PhD Thesis, Griffith University, Australia, 2005.

INTRODUCTION TO GR-TREE AND 4-R INDEX STRUCTURES OF BITEMPORAL DATA

Ngo Quynh Nhu, Hoang Quang
College of Sciences, Hue University

SUMMARY

Because bitemporal databases are basically appended only, they are usually very large in size. On the other hand, they usually contain a significant part of now-relative bitemporal data, so processing and searching data are more complicated and time-consuming. One of the effective access methods to this type of data is creating an extra retrieval structure that is called index. This paper introduces two effective index structures that permit to retrieve bitemporal data, especially now-relative time, they are GR-tree and 4-R.