ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

NGUYỄN DUY CHINH

KHAI THÁC TOP-K ĐỒNG XUẤT HIỆN VỚI BITTABLE

Ngành HỆ THỐNG THÔNG TIN

Mã số 12.12.014

ĐỒ ÁN TỐT NGHIỆP THẠC SĨ HỆ THỐNG THÔNG TIN

NGƯỜI HƯỚNG DẪN KHOA HỌC:

TS. NGUYỄN NGỌC THẢO

TP. Hồ Chí Minh - 2017

LỜI CẢM ƠN

Tôi xin gửi lời cảm ơn chân thành, lòng biết ơn sâu sắc đến tất cả các quý thầy cô khoa Công Nghệ Thông Tin trường đại học Khoa Học Tự Nhiên TP.HCM đã giảng dạy bằng tất cả sự nhiệt tình trong quá trình tôi học tại đây.

Đặc biệt tôi xin gởi lời cảm ơn chân thành nhất và lòng biết ơn sâu sắc nhất đến giảng viên hướng dẫn, TS. Nguyễn Ngọc Thảo vì đã tận tình hướng dẫn và giúp đỡ tôi trong quá trình làm đồ án.

Xin cảm ơn Cha Mẹ đã sinh ra con, nuôi nấng và dạy dỗ con, động viên con trong học tập cũng như công việc.

Cảm ơn vợ tôi đã luôn động viên và là động lực cho tôi hoàn thành đồ án này.

Cuối cùng, tôi xin gửi lời cảm ơn đến các bạn của tôi ở lớp cao học khóa 23 đại học Khoa Học Tự Nhiên TP.HCM, đã giúp đỡ và động viên tôi rất nhiều để tôi có thể hoàn thành được đồ án tốt nghiệp này.

Nguyễn Duy Chinh

MỤC LỤC

[1.1. Khai phá dữ liệu 11](#_Toc497137754)

[1.2. Khai thác tập sự kiện phổ biến (FIM) 11](#_Toc497137755)

[1.3. Khai thác top-k sự kiện đồng xuất hiện 12](#_Toc497137756)

[1.3.1. Thuật toán NT và NTI 13](#_Toc497137757)

[1.3.2. Thật toán NT-TA và NTI-TA 13](#_Toc497137758)

[1.3.3. Thuật toán PT và PT-TA 14](#_Toc497137759)

[1.4. Khai thác top-k sự kiện đồng xuất hiện với BitTable 15](#_Toc497137760)

[2. Cơ sở lý thuyết 16](#_Toc497137761)

[2.1. Giới thiệu 16](#_Toc497137762)

[2.2. Định nghĩa 1 18](#_Toc497137763)

[2.3. Định nghĩa 2 18](#_Toc497137764)

[2.4. Định nghĩa 3 18](#_Toc497137765)

[2.5. Định nghĩa 4 18](#_Toc497137766)

[2.6. Định nghĩa 5 (Top-k sự kiện đồng xuất hiện) 18](#_Toc497137767)

[2.7. Phát biểu bài toán 18](#_Toc497137768)

[2.8. Thuật toán (Navive Hunting algorithm) 19](#_Toc497137769)

[2.9. Thuật toán (Naive Hunting algorithm with Inverted list index) 19](#_Toc497137770)

[2.10. Cấu trúc BitTable 20](#_Toc497137771)

[3. Khai thác top-k đồng xuất hiện sử dụng cấu trúc bittable 22](#_Toc497137772)

[3.1. Thuật toán và 22](#_Toc497137773)

[3.2. Thuật toán (Bittable-based with Inverted list index in Vertical) 23](#_Toc497137774)

[4. Kết quả đạt được 25](#_Toc497137775)

[4.1. Cơ sở dữ liệu 26](#_Toc497137776)

[4.2. Phương pháp thực nghiệm 27](#_Toc497137777)

[4.3. Thời gian tiền xữ lý 27](#_Toc497137778)

[4.4. Thời gian xử lý trên cơ sở dữ liệu Connect 27](#_Toc497137779)

[4.5. Thời gian xử lý trên cơ sở dữ liệu Accidents 29](#_Toc497137780)

[5. Kết luận và hướng phát triển 31](#_Toc497137781)

DANH MỤC CÁC BẢNG

[Bảng 1. Ví dụ cơ sở dữ liệu giao tác 12](#_Toc497138198)

[Bảng 2. BitTable theo chiều ngang 21](#_Toc497138199)

[Bảng 3. BitTable theo chiều dọc 21](#_Toc497138200)

[Bảng 4. Cơ sở dữ liệu sau khi loại bỏ các giao tác không chứa 24](#_Toc497138201)

[Bảng 5. Bảng BitTable dạng dọc của tất cả những giao tác đều chứa 24](#_Toc497138202)

[Bảng 6. Bảng BitTable mà sẽ xử lý 25](#_Toc497138203)

[Bảng 7. Đặc điểm của các cơ sở dữ liệu sử dụng trong thực nghiệm 26](#_Toc497138204)

DANH MỤC CÁC HÌNH VẼ, ĐỒ THỊ

[Hình 1. Pi-Tree cho Bảng 1 15](#_Toc497138387)

[Hình 2. Thời gian xử lý trên cơ sở dữ liệu Connect với K=1 27](#_Toc497138388)

[Hình 3. Thời gian xử lý trên cơ sở dữ liệu Connect với K=5 28](#_Toc497138389)

[Hình 4. Thời gian xử lý trên cơ sở dữ liệu Connect với K=10 28](#_Toc497138390)

[Hình 5. Thời gian xử lý trên cơ sở dữ liệu Connect với K=15 29](#_Toc497138391)

[Hình 6. Thời gian xử lý trên cơ sở dữ liệu Accidents với K=1 29](#_Toc497138392)

[Hình 7. Thời gian xử lý trên cơ sở dữ liệu Accidents với K=5 30](#_Toc497138393)

[Hình 8. Thời gian xử lý trên cơ sở dữ liệu Accidents với K=10 30](#_Toc497138394)

[Hình 9. Thời gian xử lý trên cơ sở dữ liệu Accidents với K=15 31](#_Toc497138395)

DANH MỤC CÁC TỪ VIẾT TẮT

|  |  |
| --- | --- |
| **Từ Viết Tắt** | **Diễn Giải** |
| FIM | Frequent Itemset Mining |
| NT | Naive Hunting algorithm |
| NTI | Naive Hunting algorithm withInverted list index |
| NT-TA | with pruning |
| NTI-TA | with pruning |
| PT | Pi-Tree-based algorithm |
| PT-TA | Pi-Tree-based algorithm with TA pruning |
| BT | BitTable based algorithm |
| BTI | BitTable base algorithm with Inverted list index |
| BTIV | BitTable based algorithm with Inverted list index in Vertical |

MỞ ĐẦU

**Lời nói đầu**

Ngày nay dữ liệu được tạo ra từ mọi thiết bị, hơn 90% lượng dữ liệu trên thế giới được tạo ra trong hai năm gần đây. Có rất nhiều thiết bị và hệ thống tạo ra dữ liệu đặc biệt là các máy vi tính và thiết bị di động, các mạng xã hội. Dữ liệu này phản ánh mọi hoạt động của con người. Do đó trong những năm gần đây khai thác dữ liệu nổi lên như một ngành quan trọng của khoa học máy tính nhằm tìm kiếm các tri thức có giá trị trong khối dữ liệu khổng lồ đó, nhất là trong những lĩnh vực như xử lý số liệu, phân lớp dữ liệu, dự báo,…

Sự đa dạng và phong phú của dữ liệu hình thành nên nhiều mô hình dữ liệu khác nhau: mô hình dữ liệu giao tác (transaction), mô hình dữ liệu chuỗi (sequence), mô hình dữ liệu thời gian thực (real-time), …

Khai phá dữ liệu đang dần trở thành một ngành khoa học quan trọng trong đời sống của con người, giúp chúng ta có thể hiểu rỏ hơn quá khứ cũng như dự đoán tương lai. Đặc biệt trong kinh doanh, hiểu được hành vi của khách hàng là một lợi thế vô cùng lớn quyết định thành công của doanh nghiệp. Từ khi mới ra đời, bài toán phân tích các giao dịch của khách hàng để tìm ra các quy luật hành vi mua sắm của khách hàng đã trở thành bài toán tiêu biểu trong khai phá dữ liệu và đã có rất nhiều nghiên cứu hằng năm liên quan đến đề tài này. Tuy nhiên, không phải lúc nào người ta cũng quan tâm đến tất cả các quy luật liên kết trong toàn bộ cơ sở dữ liệu mà họ chỉ quan tâm đến những những hành vi liên quan đến một sự kiện nhất định. Từ đó, xuất hiện bài toán “*Tìm top-k sự kiện đồng xuất hiện*” được đề xuất vào năm 2015 bởi Zhi-Hong Deng. Ví dụ, khi người dùng chọn mua ba món hàng A,B và C, chúng ta sẽ cần quan tâm đến một số mặt hàng mà thường được mua cùng với ABC nhất, từ đó ta có thể đưa ra những gợi ý hữu ích cho khách hàng nhằm bán được nhiều mặt hàng hơn. Từ khi bài toán được đề xuất đến nay chưa có thêm công trình nghiên cứu mới nào được đề xuất để giảm thời gian xử lý bài toán. Do đó, đề tài “*Khai thác top-k sự kiện đồng xuất hiện với BitTable*” được đề xuất nhằm cải tiến thời gian xử lý bài toán.

**Bố cục của đồ án**

Đồ án được trình bày trong 5 chương.

Chương 1 giới thiệu sơ lược về đề tài thực hiện, lý do đề xuất đề tài và các hoàn cảnh thực tế có thể áp dụng đề tài đồng thời mô tả các công trình nghiên cứu đã được thực hiện trước đây, có liên quan đến đề tài thực hiện sẽ được thảo luận nhằm tìm kiếm giải pháp cho đề tài, các công trình này là nền tảng cho việc nghiên cứu nhằm đề xuất các cấu trúc dữ liệu và giải thuật mới. Chương này cũng nêu hướng tiếp cận nghiên cứu và đóng góp của đề tài.

Chương 2 trình bày cơ sở lý thuyết của bài toán khai thác top-k sự kiện đồng xuất hiện và các giai đoạn giải quyết bài toán. Khai thác top-k sự kiện đồng xuất hiện là một bài toán mới, được đề xuất từ năm 2015 bởi Zhi-Hong Deng. Tác giả đã đưa ra bốn thuật toán cơ bản và hai thuật toán dựa trên một cấu trúc dữ liệu đặc biệt có tên là Pi-Tree. Trong chương này chúng ta sẽ nhắc lại những định nghĩa, các thuật toán cơ bản (, ) và khái quát bài toán khai thác top-k sự kiện đồng xuất hiện.

Chương 3 trình bày chi tiết các thuật toán mà đề tài đề xuất, hai thuật toán sử dụng cấu trúc BitTable theo chiều ngang và thuật toán thứ ba dùng cấu trúc BitTable theo chiều dọc. Ba thuật toán lần lược là (BitTable based algorithm), (BitTable base algorithm with Inverted list index) và (BitTable based algorithm with Inverted list index in Vertical). khai thác bằng cách chuyển cơ sở dữ liệu sang BitTable theo chiều ngang, sau đó quét tất cả các bit trong BitTable để tìm ra các item đồng xuất hiện. khai thác tương tự như , nhưng trước khi chuyển cơ sở dữ liệu sang BitTable, loại bỏ tất cả các giao tác (transaction) không mong đợi bằng kỹ thuật Tid-set. khai thác bằng cách loại bỏ các giao tác không mong đợi sau đó chuyển cơ sở dữ liệu đã được xử lý thành BitTable dạng dọc, sau đó BTIV sẽ đếm số bit được set bằng 1 để tìm ra các item đồng xuất hiện.

Chương 4 trình bày kết quả thực nghiệm khai thác trên các cơ sở dữ liệu giao dịch phổ biến, thường được dùng trong các nghiên cứu khai thác dữ liệu thực tế như Connect, Accidents,… Từ kết quả thực nghiệm chúng ta sẽ có cái nhìn chính xác hơn về hiệu quả và lượng bộ nhớ tiêu tốn của từng thuật toán.

Chương 5 trình bày kết luận và định hướng phát triển của đề tài. Cuối cùng, phần tham khảo trình bày các bài báo, sách được tham khảo, trích dẫn trong luận văn.

Như vậy đồ án sẽ được trình bày như sau

* Chương 1: Tổng quan
* Chương 2: Cơ sở lý thuyết
* Chương 3: Khai thác top-k sự kiện đồng xuất hiện với BitTable
* Chương 4: Kết quả thực nghiệm
* Chương 5: Kết luận và hướng phát triển

# Tổng quan

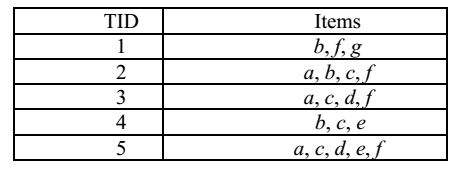
# Khai phá dữ liệu

Khai phá dữ liệu (data mining) Là quá trình tính toán để tìm ra các mẫu trong các bộ dữ liệu lớn liên quan đến các phương pháp tại giao điểm của máy học, thống kê và các hệ thống cơ sở dữ liệu. Đây là một lĩnh vực liên ngành của khoa học máy tính… Mục tiêu tổng thể của quá trình khai thác dữ liệu là trích xuất thông tin từ một bộ dữ liệu và chuyển nó thành một cấu trúc dễ hiểu để sử dụng tiếp. Ngoài bước phân tích thô, nó còn liên quan tới cơ sở dữ liệu và các khía cạnh quản lý dữ liệu, xử lý dữ liệu trước, suy xét mô hình và suy luận thống kê, các thước đo thú vị, các cân nhắc phức tạp, xuất kết quả về các cấu trúc được phát hiện, hiện hình hóa và cập nhật trực tuyến. Khai thác dữ liệu là bước phân tích của quá trình "khám phá kiến thức trong cơ sở dữ liệu".

# Khai thác tập sự kiện phổ biến (FIM)

Khai thác tập sự kiện là một nhánh quan trọng trong lĩnh vực khai phá dữ liệu, khám phá những mẫu hữu ích hoặc thú vị trong cơ sở dữ liệu giao tác. Nó được giới thiệu đầu tiên vào năm 1993 bởi Agrawal và Srikant với tên gọi là khai phá tập sự kiện lớn (*large itemset mining*). Công việc của FIM có thể được định nghĩa như sau: cho một cơ sở dữ liệu giao dịch của khách hàng, FIM khám phá những nhóm mặt hàng mà khách hàng mua thường xuyên dựa vào một ngưởng minsup cho trước. Ví dụ, người ta có thể phân tích một cơ sở dữ liệu giao dịch của khách hàng và khám phá ra rằng khách hàng thường mua Taco Shell với ớt. Khám phá sự liên hệ giữa những mặt hàng rất hữu ích để xác định hành vi của khách hàng.

Mặt dù FIM ban đầu được đề xuất là để phân tích dữ liệu khách hàng, nhưng bây giờ nó đã được xem như là công việc khai phá dữ liệu tiêu biểu mà được ứng dụng trong nhiều lĩnh vực khác nhau. Trên thực tế, cơ sở dữ liệu giao dịch khách hàng có thể được xem như là một cơ sở dữ liệu của các thể hiện các đối tượng (giao dịch), trong đó mỗi đối tượng được mô tả bằng cách sử dụng các giá trị thuộc tính danh nghĩa (các mặt hàng). Do đó, FIM có thể được định nghĩa tương đương như là nhiệm vụ tìm các giá trị thuộc tính thường xảy ra đồng thời trong cơ sở dữ liệu. Vì nhiều loại dữ liệu có thể được biểu diễn dưới dạng cơ sở dữ liệu giao dịch, FIM có nhiều ứng dụng trong nhiều lĩnh vực như tin sinh học, phân loại hình ảnh, phân tích lưu lượng mạng, phân tích đánh giá của khách hàng, theo dõi hoạt động, phát hiện phần mềm độc hại và e-learning12… Bảng 1 bên dưới là ví dụ mô tả một cơ sở dữ liệu giao tác. Cơ sở dữ liệu này sẽ là đầu vào cho những ví dụ cũng như mô phỏng thuật toán của toàn bộ đề tài.



1. Ví dụ cơ sở dữ liệu giao tác

Lĩnh vực khai phá tập sự kiện là một lĩnh vực nghiên cứu rất năng động và có hàng trăm thuật toán được đề xuất mỗi năm.

# Khai thác top-k sự kiện đồng xuất hiện

Khai thác top-k sự kiện đồng xuất hiện giống với khai thác tập sự kiện phổ biến ở chổ là nó tìm mối liên hệ giữa những sự kiện này với những sự kiện khác. Nhưng khác với khai thác tập sự kiện phổ biến FIM ở chổ là, FIM sẽ tìm ra tất cả các luật liên kết giữa các sự kiện trong toàn bộ cơ sở dữ liệu, còn khai thác top-k sự kiện đồng xuất hiện chỉ tập trung khai thác các sự kiện liên quan đến một hay một tập sự kiện cụ thể nào đó. Do đó dữ liệu đầu vào của hai bài toán này cũng khác nhau. Trong một số trường hợp người ta không quan tâm đến việc tìm tất cả các mối liên hệ giữa các sự kiện trong toàn bộ cơ sở dữ liệu mà họ chỉ quan tâm đến một sự kiện hoặc một nhóm sự kiện cụ thể nào đó. Ví dụ, khi phân tích các mặt hàng trong cơ sở dữ liệu giao dịch của khách hàng, người ta muốn biết là top 5 sản phẩm khách hàng thường mua chung với sữa và đường là gì, hoặc trong các mạng xã hội, người ta quan tâm top 2 sự kiện thường xảy ra với một sự kiện nào đó… Do đó đầu vào của 2 bài toán này sẽ khác nhau. FIM chỉ cần mội ngưỡng minsup cho trước, còn top-k sự kiện đồng xuất hiện thì cần một tập sự kiện (itemset) cần quan tâm và một số k là số sự kiện đồng xuất hiện cần tìm.

Bài toán khai thác top-k sự kiện đồng xuất hiện được đưa ra lần đầu tiên bởi Zhi-Hong Deng, trong một bài báo khoa học có tên là “*Mining Top-K Co-Occurrence Items*” năm 2015. Trong nghiên cứu của mình, ông đã trình bày bốn thuật toán cơ bản và giới thiệu một cấu trúc dữ liệu đặt biệt có tên là Pi-Tree. Dựa trên Pi-Tree, tác giả đã đề xuất hai thuật toán tên là (Pi-Tree-based algorithm) and (Pi-Tree-based algorithm with TA pruning). Bốn thuật toán cơ bản lần lượt có tên là (Naive Hunting algorithm), (Naive Hunting algorithm withInverted list index), - ( with pruning), và -( with pruning)

# Thuật toán và

là thuật toán cơ bản nhất. sẽ quét tất cả các sự kiện trong cơ sở dữ liệu để tìm số lần đồng xuất hiện của từng sự kiện. Rỏ ràng việc làm này sẽ không hiệu quả khi mà phải quét quá nhiều giao tác không mong đợi (không chứa tập sự kiện đầu vào cần quan tâm). Để trách không phải xử lý những ứng viên không mong đợi như vậy, một bước cải tiến đó là thuật toán . sẽ loại bỏ tất cả các giao tác không mong đợi dựa vào một kỹ thuật gọi là Tid-set, chuyển toàn bộ cơ sở dữ liệu sang dạng dọc rồi giao tất cả các Tid-set lại để cho ra kết quả là một cơ sở dữ liệu chỉ bao gồm các giao tác chứa tập sự kiện cần khai thác. Do xử lý trên cơ sở dữ liệu đã được rút tỉa nhỏ hơn nên tốc độ xử lý sẽ nhanh hơn.

# Thật toán - và -

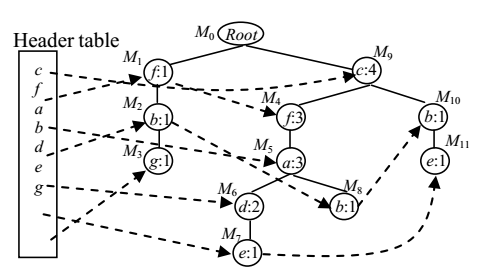
- và - là hai thuật toán mở rộng tương ứng của và bằng cách áp dụng nguyên lý (the threshold algorithm) của Fagin để có thể ngừng thuật toán ngay khi tìm ra được kết quả mà không cần phải quét hết toàn bộ cơ sở dữ liệu. Khi đó, thuật toán sẽ có hai danh sách, một danh sách chứa những item kết quả () và một danh sách chứa những item ứng ứng viên (). Giá trị đồng xuất hiện lớn nhất của một item bằng giá trị đồng xuất hiện tại thời điểm đang xét cộng với số giao tác còn lại (chưa được duyệt) trong cơ sở dữ liệu. Do đó, nếu giá trị đồng xuất hiện nhỏ nhất trong lớn hơn giá trị đồng xuất hiện lớn nhất trong cộng số giao tác còn lại chưa duyệt trong cơ sở dữ liệu thì có thể ngừng thuật toán mà không cần phải duyệt tiếp nữa.

# Thuật toán và -

Lần đầu tiên một cấu trúc dữ liệu đặc biệt được giới thiệu đó là Pi-Tree, sử dụng một cấu trúc gọn nhẹ để chứa thông tin của một cơ sở dữ liệu. Một Pi-Tree là một cấu trúc cây tiền tố được định nghĩa như sau:

1. Nó được tạo thành từ một gốc có nhãn là “*Root*”, một tập của những cây con của root, và một bảng header.
2. Mỗi node của cây chứa 4 trường dữ liệu: *lable*, *count*, *children-link*, và *parent-link*. Trường *lable* đăng ký item mà node thể hiện. Trường *count* đăng ký số lượng giao tác được trình bày bởi một phần đường dẫn đi qua node. Trường *children-link* đăng ký tất cả các node con của node. Trường *parent-link* đăng ký node cha của node.
3. Mỗi thành phần trong bảng header bao gồm hai trường: *label* và *node-link.* Trường label đăng ký cho một item. Trường node-link đăng ký tất cả các node thể hiện item đó. Trong bản header, các thành phần được xếp thứ tự giảm dần bởi độ hổ trợ (support count). Bảng header được tạo ra để tạo thuận lợi cho việc duyệt cây Pi-Tree.

Theo định nghĩa trên Pi-Tree trông giống như FP-Tree. Tuy nhiên, có hai điểm khác biệt quan trọng giữa chúng. Đầu tiên, Pi-Tree không yêu cầu những item được đăng ký trong các node phải là những item phổ biến trong khi FP-Tree chỉ tập trung vào những item phổ biến. Thứ hai, những node của Pi-Tree có liên kết đến node cha (*parent-link*) còn FP-Tree thì không.



1. Pi-Tree cho Bảng 1

Trong khuôn khổ của đồ án sẽ không trình bày chi tiết cụ thể nội dung của thuật toán PT và PTA.

# Khai thác top-k sự kiện đồng xuất hiện với BitTable

Từ khi bài toán được giới thiệu năm 2015 đến nay, chưa có thêm một nghiên cứu nào đề xuất các phương pháp mới để giải quyết bài toán. Do đó, đồ án muốn đóng góp một phương pháp mới đó là dùng BitTable để giải quyết bài toán và kết quả thực nghiệm của phương pháp mới này cho ra kết quả cũng rất khả quan.

BitTable hay còn gọi là bảng bit, chứa các giá trị 0 hoặc 1. Mỗi bit 1 biểu thị cho sự xuất hiện của một sự kiện trong cơ sở dữ liệu. Khi chuyển cơ sở dữ liệu từ dạng text sang bảng bit chúng ta sẽ có lợi thế là tiết kiệm bộ nhớ và xử dụng được các phép toán trên bit khiến cho tốc độ xử lý trên máy tính nhanh hơn nhiều. Với ý tưởng đó mà tôi quyết định thực hiện đề tài này “khai thác top-k sự kiện đồng xuất hiện với BitTable”.

Trong quá trình nghiên cứu, đồ án đề xuất ba thuật toán: (Bittable-based), (Bittable-based with Inverted list index) và (Bittable-based with Inverted list index in Vertical). Hai thuật toán đầu và không hiệu quả như mong đợi, mặc dù khi chuyển sang BitTable có tiết kiệm được bộ nhớ nhưng thời gian xử lý trong một số trường hợp còn chậm hơn so với hai thuật toán góc và . Còn thuật toán cuối cùng là tận dụng được phép đếm bit nên cho hiệu năng xử lý vô cùng cao.

Đóng góp của đồ án là đã đề xuất ba phương pháp mới để xử lý bài toán tìm top-k sự kiện đồng xuất hiện, đồng thời rút ra một số lưu ý khi xử dụng BitTable để giải quyết bào toán.

# Cơ sở lý thuyết

# Giới thiệu

Khai thác tập sự kiện phổ biến được đề xuất đầu tiên bởi Agrawal, nhằm phân tích các giao dịch của khách hàng để rút ra tất cả các luật liên kết giữa các mặt hàng. Kể từ khi việc khai thác dữ liệu này được đề xuất và những thuật toán khai thác hữu ích liên quan tới nó, đã có hàng nghìn công trình nghiên cứu theo sau trên nhiều lĩnh vực và ứng dụng. Khai thác tập phổ biến đã xuất hiện như một chủ đề quan trọng trong lĩnh vực khai thái dữ liệu. Nó đã được chứng minh đóng một vai trò thiết yếu trong nhiều công việc khai thác dữ liệu.

Trong khai thác tập phổ biến, một tập được gọi là phổ biến nếu tuần suất xuất hiện của nó trong cơ sở dữ liệu không nhỏ hơn một ngưỡng cho trước (minsup) . Có nghĩa là, tập sự kiện phổ biến là một khái niệm bao trùm trên phạm vi tất cả cơ sổ dữ liệu và chúng không có bất kỳ liên quan gì đến những tập sự kiện khác. Tuy nhiên, trong một số ứng dụng như các hệ thống gợi ý hoặc mạng xã hội, người ta có lẻ sẽ quan tâm đến những sự kiện có liên quan với nhau nhiều hơn. Ví dụ trong hệ thống đưa ra gợi ý của một cửa hàng trực tuyến, bất cứ khi nào một người dùng thanh toán một vài sản phẩm, hệ thống phải đưa ra những gợi ý về những sản phẩm khác mà thường mua cùng với những sản phẩm đó nhằm mục đích tăng doanh số. Những việc đó có thể được minh họa bằng ví dụ dưới đây.

Ví dụ: hãy xét một cơ sở dữ liệu chứa những giao dịch mua hàng được thể hiện ở Bảng 1. và Items là những thuộc tính thuộc miền sau và . Mỗi giao tác trong có một định danh duy nhất () là một tập con của . Để đơn giãn, chúng ta thể hiện một giao tác với bởi . Ví dụ là ký hiệu cho giao tác thứ hai trong Bảng 1. Top-2 những sự kiện đồng xuất hiện với tập sự kiện là sự kiện và vì có ba giao tác mua hàng chứa và hai giao tác mua hàng chứa trong khi những sự kiện khác đồng xuất hiện với hầu hết có một giao tác. Tương tự, top-2 sự kiện đồng xuất hiện với tập sự kiện là sự kiện và .

Mặc dù nhiều thuật toán đã được đề xuất để khai thác các tập sự kiện phổ biến và top-k tập sự kiện phổ biến, nhưng không khả thi để dễ dàng áp dụng những thuật toán đó để khai thác top-k sự kiện đồng xuất hiện. Thứ nhất, khai thác top-k sự kiện đồng xuất hiện không cần thiết lập một ngưỡng hổ trợ nhỏ nhất như khai thác tập sự kiện phổ biến hay top-k tập sự kiện phổ biến, nơi mà một tập sự kiện phổ biến được định nghĩa như một tập có độ hổ trợ không nhỏ hơn ngưỡng hổ trợ nhỏ nhất cho trước. Vì thế, không thể áp dụng trực tiếp các thuật toán cho khai thác tập sự kiện phổ biến hay top-k tập sự kiện phổ biến vào khai thác top-k sự kiện đồng xuất hiện khi mà không khả thi để thiết lập một ngưỡng phù hợp để tìm top-k sự kiện đồng xuất hiện của mỗi tập sự kiện. Chú ý rằng, độ hổ trợ tối thiểu của top-k sự kiện đồng xuất hiện của những tập sự kiện khác nhau thì vô cùng, chúng ta không thể nào biết trước được. Thứ hai, top-k sự kiện đồng xuất hiện không phải downward closure property,

Cho một tập sự kiện truy vấn, một hướng tiếp cận đơn giản để tìm top-k những sự kiện đồng xuất hiện là xem xét tất cả những sự kiện không liên quan đến những tập sự kiện trong cơ sở dữ liệu. Chắc chắn cách tiếp cận này sẽ gặp phải vấn đề không gian tiềm kiếm lớn, đặc biệt khi cơ sở dữ liệu rất lớn và một con số lớn của những tập sự kiện truy vấn cần được tìm kiếm top-k những sự kiện đồng xuất hiện của chúng. Vì thế, để giảm không gian tiềm kiếm và tìm top-k những sự kiện đồng xuất hiện một cách hiệu quả là một thách thức mới.

Cho là tập tất cả các sự kiện và là một cơ sở dữ liệu giao tác, với mỗi là một giao tác là tập các sự kiện mà . Chúng ta cũng có thể gọi là một itemset nếu là tập các sự kiện .

# Định nghĩa 1

Cho là một itemset. Một giao tác được gọi là chứa nếu và chỉ nếu .

# Định nghĩa 2

Độ hổ trợ của một itemset, được ký hiệu là , là số giao tác chứa trong .

# Định nghĩa 3

Cho itemset, một sự kiện là một sự kiện đồng xuất hiện của nếu và chỉ nếu một hoặc nhiều giao tác chứa . Tập tất cả những sự kiện đồng xuất hiện của được ký hiệu là .

# Định nghĩa 4

Cho itemset và , độ đếm đồng xuất hiện của và , được ký hiệu là , được định nghĩa là độ hổ trợ của .

# Định nghĩa 5 (Top-k sự kiện đồng xuất hiện)

Cho itemset , một sự kiện được gọi là Top-k sự kiện đồng xuất hiện của nếu có ít hơn sự kiện mà độ điếm đồng xuất hiện của chúng lớn hơn .

# Phát biểu bài toán

Cho một cơ sở dữ liệu giao tác , một itemset , và số mong muốn, bài toán tìm top-k sự kiện đồng xuất hiện của là tìm sự kiện mà xảy ra phổ biến nhất với trong .

Giả định chúng ta muốn biết những sự kiện nào xảy ra phổ biến nhất với trong Bảng 1. Từ định nghĩa 4, ta biết rằng bởi vì chỉ chứa . Tương tự, ta có , , và . Nếu ngưỡng được thiết lập là 1, thì top-1 sự kiện đồng xuất hiện của là . Nếu , thì top-2 những sự kiện đồng xuất hiện của là và .

# Thuật toán (Navive Hunting algorithm)

Cho một itemset và cơ sở dữ liệu giao tác, quét tất cả các giao tác trong . Nếu một giao tác chứa , tăng số lượng đồng xuất hiện của mỗi sự kiện , không chứa trong trong giao tác, lên 1. Bằng cách duyệt tất cả các giao tác, có được số lượng đồng xuất hiện của tất cả các sự kiện đồng xuất hiện của . Bằng việc sắp xếp giảm dần tất cả các sự kiện đồng xuất hiện theo số lượng đồng xuất hiện, tìm thấy tất cả top-k sự kiện đồng xuất hiện của .

|  |
| --- |
| Thuật toán |
| **Input**: cơ sở dữ liệu giao tác , itemset , và ngưỡng .  **Output**: , Top-k sự kiện đồng xuất hiện của  1: **foreach** **do**  2: **if** **then**  3: **foreach** **do**  4: **if** is fist visited **then**  5:  6: **else**  7:  8: is one of top-k biggest elements of  9: **return** |

# Thuật toán (Naive Hunting algorithm with Inverted list index)

Rỏ ràng, không hiệu quả khi nó quét tất cả giao tác trong cơ sở dữ liệu mà trong đó có nhiều giao tác không chứa itemset dẫn đến nó phải quét những giao tác không mong đợi. Để loại bỏ việc duyệt những giao tác không mong đợi, đầu tiên sẽ tìm tất cả các giao tác chứa itemset bằng cách sử dụng Tid-set[37], một cấu trúc dữ liệu được tổ chức từ danh sách được đảo ngược.Tid-set của một itemset là tập TIDs của tất cả các giao tác có chứa itemset đó. Như đã phát biểu trong [37], Tid-set có thể được tính toán bằng cách giao tất cả các Tid-set của những sự kiện mà được chứa bởi itemset đó. Khi đó, tìm ra top-k sự kiện đồng xuất hiện trong một cơ sở dữ liệu được hoạch định, là tập tất cả các giao tác chứa , như đã làm. Bởi vì chỉ tìm top-k sự kiện đồng xuất hiện trong một cơ sở dữ liệu nhỏ hơn, nên nó sẽ hiệu quả hơn .

|  |
| --- |
| Giải thuật của thuật toán |
| **Input**: cơ sở dữ liệu giao tác , itemset , và ngưỡng .  **Output**: top-k sự kiện đồng xuất hiện của .  1: **foreach** **do**  2: **foreach** **do**  3: **if** is first visited **then**  4: TID\_list()  5: **else**  6: TID\_list() TID\_list()  7: TID\_list() TID\_list(*first-item*  8: **foreach** first-item **do**  9: TID\_list() TID\_list() TID\_list();  10: TID\_list()}  11: chạy thuật toán nhưng thay bằng ; |

# Cấu trúc BitTable

Bittable là cấu trúc dữ liệu dạng bảng mà mỗi bit biểu diễn một sự kiện xảy ra hay không. Nếu giá trị là 1 có nghĩa là sự kiện có xảy ra. Gọi cơ sở dữ liệu giao tác là ta ký hiệu bảng BitTable theo chiều ngang là và bảng BitTable theo chiều dọc là . Ví dụ ta có thể chuyển Bảng 1 từ cơ sở dữ liệu giao tác thành cấu trúc bittable như sau:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **a** | **b** | **c** | **d** | **e** | **f** | **g** |
| **1** | *0* | *1* | *0* | *0* | *0* | *1* | *1* |
| **2** | *1* | *1* | *1* | *0* | *0* | *1* | *0* |
| **3** | *1* | *0* | *1* | *1* | *0* | *1* | *0* |
| **4** | *0* | *1* | *1* | *0* | *1* | *0* | *0* |
| **5** | *1* | *0* | *1* | *1* | *1* | *1* | *0* |

1. BitTable theo chiều ngang

Đối với bảng BitTable theo chiều ngang, mỗi cột tương ứng với một sự kiện, mỗi dòng tương ứng với một giao tác. Khi đó, nếu ta có một cơ sở dữ liệu giao tác với sự kiện và giao tác thì ta sẽ có một bảng BitTable với kích thước . Bảng BitTable theo chiều dọc cũng tương tự nhưng đổi cột thành dòng.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** |
| **a** | *0* | *1* | *1* | *0* | *1* |
| **b** | *1* | *1* | *0* | *1* | *0* |
| **c** | *0* | *1* | *1* | *1* | *1* |
| **d** | *0* | *0* | *1* | *0* | *1* |
| **e** | *0* | *0* | *0* | *1* | *1* |
| **f** | *1* | *1* | *1* | *0* | *1* |
| **g** | *1* | *0* | *0* | *0* | *0* |

1. BitTable theo chiều dọc

# Khai thác top-k đồng xuất hiện sử dụng cấu trúc bittable

# Thuật toán và

Tương tự như , nhưng khác ở chổ là xử lý trên cơ sở dữ liệu giao tác dạng text , còn xử lý trên bảng BitTable dạng ngang . cũng duyệt từng dòng của bảng BitTable để đếm số lần đồng xuất hiện của tất cả các sự kiện. Ví dụ đầu vào của là Bảng 1 thì của là Bảng 2.

Do mỗi dòng là một giao tác dạng bit, nên để kiểm tra itemset có được chứa trong giao tác hay không, chỉ cần thực hiện phép để kiểm tra. Ví dụ, xét bảng BitTable Bảng 2, để kiểm tra itemset có chứa trong giao tác hay không ta chỉ cần kiểm tra bit tại và bằng cách đơn giản là thực hiện phép AND giữa chuỗi bit 1110010 với 1010000 nếu kết quả bằng với 1010000 thì kết luận chứa trong . Tồng quát ta có .

|  |
| --- |
| Thuật toán |
| **Input**: cơ sở dữ liệu giao tác , itemset , và ngưỡng .  **Output**: , Top-k sự kiện đồng xuất hiện của  1: Chuyển thành  2: Tạo từ  3: **foreach** **do**  4: **if** **then**  5: **foreach** **do**  6: **if** **then**  **7:**  8: **if** **then**  9:  10: **else**  11:  12: is one of top-k biggest elements of  13: **return** |

Rỏ ràng, không hiệu quả khi nó quét tất cả các dòng giao tác không mong đợi(không chứa itemset ). Để loại trừ điều đó, sử dụng Tid-set[37] tương tự như để xây dựng một cơ sở dữ liệu chỉ bao gồm những giao tác có chứa itemset . sẽ xây dựng trước, sau đó sẽ chuyển thành bảng BitTable theo chiều ngang để xử lý.

|  |
| --- |
| Giải thuật của thuật toán |
| **Input**: cơ sở dữ liệu giao tác , itemset , và ngưỡng .  **Output**: top-k sự kiện đồng xuất hiện của .  1: **foreach** **do**  2: **foreach** **do**  3: **if** is first visited **then**  4: TID\_list()  5: **else**  6: TID\_list() TID\_list()  7: TID\_list() TID\_list(*first-item*  8: **foreach** first-item **do**  9: TID\_list() TID\_list() TID\_list();  10: TID\_list()}  11: chạy thuật toán nhưng thay bằng ; |

# Thuật toán (Bittable-based with Inverted list index in Vertical)

xử lý dựa vào bảng BitTable theo chiều dọc thay vì theo chiều ngang giống và . Đồng thời cũng loại bỏ những giao tác không chứa trước khi xử lý. Ví dụ, đối với cơ sở dữ liệu Bảng 1, cho itemset . sẽ loại bỏ những giao tác không chứa trước, khi đó ta sẽ có như bảng sau:

|  |  |
| --- | --- |
| TID | Items |
| 2 |  |
| 3 |  |
| 5 |  |

1. Cơ sở dữ liệu sau khi loại bỏ các giao tác không chứa

Sau đó sẽ chuyển Bảng dữ liệu trên thành BitTable dạng dọc

|  |  |  |  |
| --- | --- | --- | --- |
|  | **2** | **3** | **5** |
| **a** | *1* | *1* | *1* |
| **b** | *1* | *0* | *0* |
| **c** | *1* | *1* | *1* |
| **d** | *0* | *1* | *1* |
| **e** | *0* | *0* | *1* |
| **f** | *1* | *1* | *1* |
| **g** | *0* | *0* | *0* |

1. Bảng BitTable dạng dọc của tất cả những giao tác đều chứa

Tiếp theo loại bỏ những dòng dữ liệu của những sự kiện chứa trong itemset là và

|  |  |  |  |
| --- | --- | --- | --- |
|  | **2** | **3** | **5** |
| **b** | *1* | *0* | *0* |
| **d** | *0* | *1* | *1* |
| **e** | *0* | *0* | *1* |
| **f** | *1* | *1* | *1* |
| **g** | *0* | *0* | *0* |

1. Bảng BitTable mà sẽ xử lý

Để đếm số lần đồng xuất hiện của mỗi sự kiện, chỉ việc đếm số bit được set là 1 trong mãng bit tương ứng mà thôi. Rỏ ràng việc đếm số lượng bit 1 trong một mãng bit sẽ đơn giản và nhanh hơn rất nhiều so với việc duyệt từng bit để đếm số lượng đồng xuất hiện của mỗi sự kiện.

|  |
| --- |
| **Thuật toán** |
| **Input**: cơ sở dữ liệu giao tác , itemset , và ngưỡng .  **Output**: top-k sự kiện đồng xuất hiện của .  1: Xây dựng từ ;  2: Chuyển thành bảng BitTable theo chiều dọc ;  3: **foreach** **do**  4: loại bỏ ;  5: **foreach** **do**  6: ;  7: is one of top-k biggest elements of  8: **return** |

So với BT và BTI, BTIV chỉ tốn một vòng lặp để ra được kết quả, so với hai vòng lặp nhự BT và BTI thì tốc độ của BTIV sẽ nhanh hơn rất nhiều.

# Kết quả đạt được

Trong phần này, chúng ta đánh giá hiệu suất của 3 thuật toán. Thử nghiệm được thực hiện trên một laptop cá nhân với cấu hình 4G Ram và bộ vi xử lý Intel Core i5-4200U 1.6 GHz. Tất cả code được hiện thực bằng ngôn ngữ C# và chạy trên hệ điều hành Win 8 64bit.

# Cơ sở dữ liệu

Hai cơ sở dữ liệu thực tế và hai cơ sở dữ liệu tổng hợp được dùng để chạy thử nghiệm. Bảng X thể hiện những thông số của các cơ sở dữ liệu. Cột cuối cùng của bảng X, chúng ta chia số lượng giao tác cho số lượng sự kiện để đánh giả tỷ trọng của cơ sở dữ liệu. Tỷ trọng càng lớn có nghĩa là cơ sở dữ liệu càng dày đặc.

Connect và Accidents là hai cơ sở dữ liệu thực tế. Chúng thường được sử dụng trong những công trình nghiên cứu trước về khai thác tập sự kiện phổ biến. Để kiểm tra các thuật toán trên cơ sở dữ liệu lớn, chúng ta tạo những cơ sở dữ liệu tổng hợp bằng chương trình SPMF, một chương trình mã nguồn mở viết bằng Java. Cơ sở dữ liệu tổng hợp được đặc tên là Syn\_data1 và Syn\_data2 được sử dụng trong thử nghiệm. Để tạo ra Syn\_data1, kích thước trung bình mỗi giao tác, số giao tác, số sự kiện khách nhau được thiết lập lần lượt là 49, 1000K và 1K. Những tham số để tạo Syn\_data2 cũng tương tự nhưng số mẫu thì được thiết lập là 100, để làm cho Syn\_data2 thưa thớt hơn Syn\_data1.

Bên dưới đây, chúng ta sẽ trình bày và phân tích các kết quả thực nghiệm trên nhiều phương diện khác nhau bao gồm thời gian tiền xữ lý, thời gian xữ lý và khả năng mở rộng.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CSDL** | **Số Lượng Gia o Tác** | **Số Items** | **Chiều Dài Trung Bình Mỗi Giao Tác** | **Tỉ Trọng** |
| Connect | 67,557 | 129 | 43 | 524 |
| Accidents | 340,183 | 468 | 34 | 727 |

1. Đặc điểm của các cơ sở dữ liệu sử dụng trong thực nghiệm

# Phương pháp thực nghiệm

Thực nghiệm sẽ chạy lần lượt trên 2 cơ sở dữ liệu: Connect và Accidents. Ba thuật toán gốc của tác giả NT,NTI,PT và ba thuật toán mới được đề xuất BT,BTI và BTIV sẽ được chạy lần lượt trên mỗi cơ sở dữ liệu. Theo kết luận của tác giả bài báo gốc thì các thuật toán mở rộng TA không đạt hiệu quả nên chúng tôi không chạy thực nghiệm các thuật toán đó. Mỗi thuật toán sẽ chạy lần lược với thông số và query length có độ dài . Các mẫu itemset đầu vào cho các thuật toàn là giống nhau. Mỗi cặp và query length cho chạy 100 lần với 100 itemset khác nhau.

# Thời gian tiền xữ lý

# Thời gian xử lý trên cơ sở dữ liệu Connect

1. Thời gian xử lý trên cơ sở dữ liệu Connect với K=1
2. Thời gian xử lý trên cơ sở dữ liệu Connect với K=5
3. Thời gian xử lý trên cơ sở dữ liệu Connect với K=10
4. Thời gian xử lý trên cơ sở dữ liệu Connect với K=15

# Thời gian xử lý trên cơ sở dữ liệu Accidents

1. Thời gian xử lý trên cơ sở dữ liệu Accidents với K=1
2. Thời gian xử lý trên cơ sở dữ liệu Accidents với K=5
3. Thời gian xử lý trên cơ sở dữ liệu Accidents với K=10
4. Thời gian xử lý trên cơ sở dữ liệu Accidents với K=15

# Kết luận và hướng phát triển