BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM

NGUYỄN TRẦN BẢO LONG

TỐI ƯU HOÁ TRUY VẤN TRONG HỆ CƠ SỞ DỮ LIỆU PHÂN TÁN

LUẬN VĂN THẠC SĨ

Chuyên ngành: CÔNG NGHỆ THÔNG TIN

Mã số ngành: 60480201

TP. HỒ CHÍ MINH, tháng 03 năm 2015

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM

NGUYỄN TRẦN BẢO LONG

TỐI ƯU HOÁ TRUY VẤN TRONG HỆ CƠ SỞ DỮ LIỆU PHÂN TÁN

LUẬN VĂN THẠC SĨ

Chuyên ngành: CÔNG NGHỆ THÔNG TIN

Mã số ngành: 60480201

CÁN BỘ HƯỚNG DẪN KHOA HỌC:

TS. NGUYỄN ĐÌNH THUÂN

TP. HỒ CHÍ MINH, tháng 3 năm 2015

LỜI CAM ĐOAN

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi. Các số liệu, kết quả nêu trong Luận văn là trung thực và chưa từng được ai công bố trong bất kỳ công trình nào khác.

Tôi xin cam đoan rằng mọi sự giúp đỡ cho việc thực hiện Luận văn này đã được cảm ơn và các thông tin trích dẫn trong Luận văn đã được chỉ rõ nguồn gốc.

Học viên thực hiện Luận văn

(Ký và ghi rõ họ tên)

Nguyễn Trần Bảo Long

LỜI CÁM ƠN

Lời đầu tiên tác giả xin chân thành cảm ơn Ban Giám Hiệu và toàn thể cán bộ nhân viên, giảng viên trường Đại Học Công Nghệ TPHCM-HUTECH, Ban lãnh đạo Phòng Quản Lý Khoa Học và Đào Tạo Sau Đại Học, khoa Công Nghệ Thông Tin đã tạo điều kiện thuận lợi cho tác giả học tập và nghiên cứu trong suốt quá trình học cao học.

Xin cảm ơn các thầy cô đã trực tiếp giảng dạy, hướng dẫn tác giả trong suốt quá trình học tập: PGS.TS Lê Hoài Bắc, PGS.TS Nguyễn Xuân Huy, TS. Nguyễn Chánh Thành, TS. Nguyễn Thị Thanh Sang, TS. Tân Hạnh, TS. Nguyễn Đình Thuân, TS. Lê Mạnh Hải, TS. Nguyễn Tuấn Đăng, TS Lư Nhật Vinh.

Tác giả đặc biệt biết ơn và tri ân sâu sắc thầy TS. Nguyễn Đình Thuân - Trưởng khoa Hệ Thống Thông Tin, trường đại học Công Nghệ Thông Tin ĐHQG-TPHCM đã rất tận tình và nghiêm túc hướng dẫn tác giả trong suốt quá trình thực hiện nghiên cứu này.

Cuối cùng tác giả xin chân thành cảm ơn ba mẹ, các bạn bè và đồng nghiệp đã quan tâm, động viên tinh thần và tiếp sức giúp đỡ tác giả vượt qua những khó khăn để hoàn thành luận văn một cách tốt đẹp.

Tác giả

Nguyễn Trần Bảo Long.

TÓM TẮT

Nhu cầu truy xuất dữ liệu trên một hệ thống có khả năng phân bố dữ liệu tại nhiều vị trí là một nhu cầu rất lớn. Hệ thống cơ sở dữ liệu phân tán đã chứng minh được tầm quan trọng của nó khi được áp dụng rộng rãi trong các hệ thống cơ sở dữ liệu hiện nay. Vấn đề tối ưu hóa truy vấn trên hệ phân tán trở nên là một nhu cầu cấp thiết. Tuy nhiên, vấn đề tối ưu hóa truy vấn không phải là một bài toán đơn giản mà nó được xếp vào dạng NP-hard khi phải đưa ra quyết định tối ưu.

Trong quá trình tối ưu hóa truy vấn, phần quan trọng nhất là xử lý quá trình tìm kiếm trong không gian rộng lớn các kế hoạch thực thi tương đương, chọn lựa và đưa ra được kế hoạch thực thi có chi phí tối ưu nhất. Mặc khác, một điều không kém phần quan trọng là thời gian tìm ra kế hoạch thực thi tối ưu phải được cân bằng với tổng thời gian hệ thống có thể phản hồi kết quả cho ứng dụng.

Luận văn tập trung nghiên cứu các mục tiêu sau:

- Tìm hiểu quá trình tối ưu hóa truy vấn trong hệ cơ sở dữ liệu phân tán.
- Vấn đề ước tính chi phí trên hệ phân tán cho phép đánh giá kế hoạch thực thi, xác định thời gian hồi đáp truy vấn.
- Trình bày các thuật toán hiện có DP, IDP1, DPccp cho phép lựa chọn kế hoạch truy vấn tối ưu.
- Kết hợp hai thuật toán IDP1 và DPccp để tạo ra thuật toán IDP1ccp hiệu quả hơn.
- Thực nghiệm so sánh kết quả của ba thuật toán IDP1, DPccp và IDP1ccp.

ABSTRACT

Accessing data on a system which is capable of distributed data in multiple locations is a huge demand. The distributed database systems demonstrated the importance of it by widely applied in the current database system. Optimizing queries in distributed database system becomes an urgent problem. But the query optimization is not a simple problem, it is classified as a NP-hard problem when to make optimal decisions.

During query optimization, the most important part is handling the search process in the vast space of equivalent execution plans, select and give the plan with the optimal implement costly. On the other hand, it is equally important, the time to find the optimal execution plan must be balanced with the amount of the system time can respond to applications.

Thesis focused on the following objectives:

- Understanding the process of query optimization in a distributed database systems.
- Estimated cost problem in a distributed system enables to evaluate query execution plan, identify the query response times.
- Presentation of the existing algorithms DP, IDP1, DPccp allows choosing the optimal query plan.
- Combination two algorithms IDP1 and DPccp to generate more efficient algorithm IDP1ccp.
- Experiments to compare the results of three algorithms IDP1, DPccp and IDP1ccp.

MỤC LỤC

DANH MỤC CÁC TỪ VIẾT TẮT/TIẾNG ANH

DBMS: hệ quản trị cơ sở dữ liệu.

CSDL: cơ sở dữ liệu.

CSDLPT: cơ sở dữ liệu phân tán.

Cost-based optimizer (CBO): bộ tối ưu chi phí.

Query Execution Plan (QEP): kế hoạch thực thi truy vấn.

Select (σ): Phép chiếu.

Project (π): Phép chọn.

Join (⋈): Phép kết.

Candinarity: tập ứng viên, lực lượng quan hệ.

Operation: phép toán, toán tử.

Optimizer: bộ tối ưu.

Schedule: lịch trình.

Catalog: danh mục hệ thống.

Join graph: đồ thị kết hợp.

Chain: Truy vấn dạng chuỗi

Cycle: Truy vấn dạng vòng

Star: Truy vấn dạng sao

Clique: Truy vấn dạng chùm

Bushy plan: kế hoạch với các nút lá tăng trưởng nhanh

Iterative Dynamic Programming (IDP): Thuật toán quy hoạch động lặp.

Dynamic Programming connected subset complement pair (DPccp): Thuật toán quy hoạch động cặp đồ thị con liên thông bù.

DANH MỤC CÁC HÌNH

DANH MỤC CÁC BẢNG

CHƯƠNG 1: MỞ ĐẦU

1.1 Giới thiệu

1.1.1 Lý do chọn đề tài

Ngày nay cùng với sự phát triển công nghệ Doanh nghiệp thông minh (Business Intelligence) và kho lưu trữ dữ liệu (Data Warehouse), số lượng dữ liệu cần phải xử lý trong cơ sở dữ liệu của các công ty là rất lớn. Các hệ thống kho dữ liệu lại cần phải được tích lũy dữ liệu trong rất nhiều năm cho nên số lượng dữ liệu càng lúc càng lớn hơn nữa. Bên cạnh yếu tố lưu trữ dữ liệu là yếu tố vị trí xử lý dữ liệu. Các công ty lớn thường có nhiều nhân viên làm việc ở các văn phòng đặt tại nhiều vị trí khác nhau. Chính vì vậy, nhu cầu lưu trữ dữ liệu phân tán là rất lớn. Tuy nhiên, khi lưu trữ dữ liệu trên hệ thống phân tán, quá trình tối ưu hóa dữ liệu trở nên phức tạp hơn rất nhiều so với hệ thống cơ sở dữ liệu tập trung.

Từ sự phức tạp đó, vấn đề tối ưu hoá truy vấn trong hệ cơ sở dữ liệu phân tán trở nên là một vấn đề cấp thiết. Tối ưu hóa truy vấn sẽ góp phần làm tăng tốc độ xử lý của hệ thống và giảm thiểu một loạt các hoạt động liên quan đến việc truy cập cơ sở dữ liệu. Đề tài tập trung nghiên cứu quá trình tối ưu hóa truy vấn và các thuật toán lựa chọn kế hoạch thực thi tối ưu trên hệ thống cơ sở dữ liệu phân tán. Khi các truy vấn trên hệ phân tán được tối ưu, điều này sẽ giúp cho các công ty khai thác được sức mạnh của hệ phân tán, tận dụng hiệu quả hệ thống mạng và hệ thống cơ sở dữ liệu đang tồn tại.

1.1.2 Mục đích, đối tượng và phạm vi nghiên cứu

- **Mục đích**

Mục đích của luận văn là nghiên cứu quá trình tối ưu hóa truy vấn và các vấn đề liên quan đến việc tính toán chi phí, lựa chọn kế hoạch thực hiện truy vấn trên hệ cơ sở dữ liệu phân tán. Quá trình tối ưu truy vấn được xem là thành phần quan trọng nhất của một hệ thống quản lý cơ sở dữ liệu. Bộ tối ưu hóa có trách nhiệm phân tích câu truy vấn của người sử dụng, tìm kiếm kế hoạch thực hiện và trả về kế hoạch với

chi phí thấp nhất. Giữa các kế hoạch thực hiện truy vấn tương đương có thể có nhiều chi phí khác nhau, do đó nhiệm vụ của quá trình tối ưu là phải tìm ra được kế hoạch thực thi không gây ra các công việc không cần thiết cho bộ xử lý.

Khi tối ưu hóa truy vấn cơ sở dữ liệu được áp dụng lên hệ thống phân tán, quá trình tối ưu hóa phải xem xét thêm các khía cạnh khác của việc chọn lựa, ví dụ như vị trí để thực hiện bắt đầu thực hiện truy vấn, vị trí sử dụng kết quả, dung lượng dữ liệu truyền dẫn qua mạng, ... Các khía cạnh này làm gia tăng thêm kích thước không gian tìm kiếm và góp phần làm cho độ phức tạp của các vấn đề tối ưu hóa ngày càng tăng. Tuy nhiên, cũng do đặc điểm dữ liệu tồn tại trên nhiều địa điểm và khả năng thực hiện truy vấn đồng thời ở nhiều địa điểm, hệ phân tán lại tạo ra cơ hội cho các xử lý song song. Việc bổ sung thêm yếu tố lựa chọn các địa điểm bắt đầu thực thi truy vấn cũng là một sự bổ sung thêm năng lực xử lý, khi đó tất cả các vị trí có thể cùng được sử dụng góp phần tăng tốc xử lý đồng thời và làm giảm thiểu thời gian xử lý truy vấn cho người dùng.

- **Đối tượng và phạm vi nghiên cứu**

Đề tài được đầu tư nghiên cứu các kỹ thuật tối ưu cho các dạng truy vấn khác nhau trong hệ phân tán. Phạm vi nghiên cứu giới hạn chỉ xem xét các truy vấn trong đại số quan hệ liên quan đến sự kết hợp của các phép toán như là phép chọn, phép chiếu và phép kết. Luận văn tập trung vào các phần sau:

- Trình bày quá trình xử lý trên hệ phân tán, vấn đề khó khăn trong việc đánh giá xử lý song song trong một kế hoạch thực thi.
- Trình bày ưu khuyết điểm của ba thuật toán lựa chọn kế hoạch thực thi tối ưu: DP, IDP1, DPccp.
- Kết hợp hai thuật toán IDP1 và DPccp để tạo ra thuật toán hiệu quả hơn là IDP1ccp.
- Trình bày thực nghiệm áp dụng các thuật toán tối ưu trên các đồ thị truy vấn dạng chuỗi, dạng vòng, dạng sao, dạng chùm.

1.2 Ý nghĩa của đề tài nghiên cứu

Luận văn có ý nghĩa thực tiễn trong việc giảm thiểu thời gian xử lý và hồi đáp truy vấn dữ liệu cho người dùng và ứng dụng. Khi người dùng hoặc ứng dụng tạo truy vấn đến cơ sở dữ liệu để rút trích thông tin thì thời gian hồi đáp là rất quan trọng. Truy vấn được tối ưu không những làm giảm thời gian xử lý truy vấn mà còn góp phần giảm thiểu thời gian xử lý của hàng loạt các hoạt động phía sau truy vấn. Đối với các hệ thống lớn, việc lựa chọn kế hoạch xử lý truy vấn không tối ưu có thể dẫn đến một truy vấn thay vì có thể được giải quyết trong một vài phút thì lại phải tốn đến hàng giờ xử lý. Khi truy vấn được tối ưu, tất cả các yếu tố liên quan đến tài nguyên hệ thống như CPU, RAM, đĩa lưu trữ, chi phí truyền mạng cũng được sử dụng hiệu quả, tiết kiệm hơn và như vậy có nghĩa là tài nguyên hệ thống được tận dụng tốt hơn, có khả năng phục vụ cho nhiều truy vấn hơn.

Về mặt ý nghĩa khoa học, khi nghiên cứu các thuật toán tối ưu hóa truy vấn, đề tài đã góp phần tổng kết và trình bày thêm giải pháp cho bài toán khó trong việc lựa chọn quyết định tối ưu. Luận văn đã đưa ra giải pháp áp dụng kết hợp giữa thuật toán quy hoạch động lặp và thuật toán mô phỏng để tạo ra thuật toán tối ưu hiệu quả hơn. Kĩ thuật tối ưu hóa áp dụng thuật toán quy hoạch động lặp chia nhỏ kế hoạch thực hiện truy vấn thành các phần nhỏ, tối ưu các phần nhỏ và xử lý loại bỏ các phép toán gây hao tốn chi phí xử lý. Và kĩ thuật tối ưu hóa của thuật toán mô phỏng dựa theo đồ thị kết hợp của truy vấn để xử lý các đồ thị con làm giảm chi phí quét toàn bộ không gian tìm kiếm của các kế hoạch xử lý tương đương. Hai kĩ thuật này đã được phân tích và áp dụng phối hợp thành một thuật toán tối ưu hơn giúp cải thiện được quá trình xử lý các truy vấn trên hệ thống cơ sở dữ liệu phân tán.

1.3 Tóm tắt bố cục trình bày của luận văn

Luận văn gồm các phần sau

- Chương 1: Mở đầu

  Trong chương này sẽ giới thiệu sự cấp thiết của đề tài nghiên cứu. Đồng thời nội dung chương cũng cho thấy được mục đích, đối tượng và phạm vi giới

hạn nghiên cứu của đề tài. Các ý nghĩa khoa học và ý nghĩa thực tiễn của đề tài nghiên cứu cũng được nêu rõ. Cuối chương là phần trình bày tóm tắt bố cục các phần của luận văn để thể hiện toàn cảnh của luận văn.

- Chương 2: Tổng quan

Các khái niệm liên quan đến tối ưu hóa truy vấn mang lại cái nhìn tổng quan cho đề tài sẽ được trình bày trong phần này. Tác giả hệ thống một số công trình nghiên cứu đã có, nêu bật quá trình xử lý truy vấn và các thách thức cần giải quyết của hệ cơ sở dữ liệu phân tán. Đồng thời chương này cũng sẽ trình bày hướng nghiên cứu giải quyết của đề tài.

- Chương 3: Các thuật toán tối ưu hóa truy vấn

Nội dung chương sẽ tập trung vào phần mô tả quá trình hoạt động và các bước thực hiện của các chương trình thuật toán: Dynamic Programming, Iterative Dynamic Programming, DPccp, IDP1ccp. Phân tích những ưu khuyết điểm của các thuật toán và cung cấp các ví dụ minh họa góp phần làm rõ quá trình thực hiện của các thuật toán.

- Chương 4: Thực nghiệm và đánh giá

Đây là chương mô tả các bước chuẩn bị trước khi làm thực nghiệm, cấu trúc các tập tin danh mục và tập tin truy vấn. Quá trình các giai đoạn thực nghiệm các thuật toán trên các dạng truy vấn khác nhau. Phần cuối của chương ghi nhận kết quả thực nghiệm, đánh giá và so sánh kết quả đạt được thông qua các biểu đồ minh họa.

- Chương 5: Kết luận

Sau quá trình tìm hiểu và nghiên cứu, những kết quả đạt được của đề tài sẽ được tổng kết lại và đưa ra kết luận cho đề tài. Ngoài ra, trong chương này cũng sẽ nói rõ những điểm giới hạn của luận văn. Sau phần trình bày những điểm giới hạn này, một số hướng nghiên cứu tiếp theo cũng được gợi ý, góp phần mở rộng hướng nghiên cứu của đề tài trong tương lai.

CHƯƠNG 2: TỔNG QUAN

2.1 Tổng quan tối ưu hóa truy vấn

Tối ưu hóa truy vấn được coi là thành phần quan trọng nhất của một hệ thống quản lý cơ sở dữ liệu. Sau khi hệ thống tiếp nhận truy vấn, bộ tối ưu truy vấn của hệ thống có trách nhiệm phân tích truy vấn của người sử dụng, xây dựng các kế hoạch thực thi tương đương của truy vấn, ước tính chi phí và chọn lựa ra kế hoạch thực hiện truy vấn với chi phí tối ưu nhất. Kế hoạch thực thi truy vấn tối ưu sẽ được bộ tối ưu chuyển đến bộ thực thi để thực hiện truy vấn.

Mỗi kế hoạch thực thi truy vấn, mặc dù đều dẫn đến một kết quả giống nhau nhưng chi phí thực hiện của mỗi kế hoạch lại khác nhau đáng kể. Khi hệ thống cơ sở dữ liệu tiếp nhận truy vấn, với mỗi câu truy vấn tiếp nhận, tập hợp các kế hoạch thực thi có thể xảy ra để xử lý truy vấn là rất lớn. Tập hợp các kế hoạch thực thi này gọi là không gian tìm kiếm kế hoạch thực thi. Không gian tìm kiếm kế hoạch thực thi thường là rất lớn cho nên việc tìm kiếm kế hoạch tốt nhất gần như là không thể. Do đó, điều quan trọng của việc tối ưu hoá truy vấn không phải là tìm ra được kế hoạch thực thi tốt nhất mà đúng hơn là giúp cho hệ thống tìm ra một kế hoạch thực thi tối ưu, tránh sử dụng phải kế hoạch thực thi không tốt.

Một vấn đề tiếp theo của quá trình tối ưu hóa truy vấn phải đối mặt đó là thứ tự xử lý các truy vấn của người dùng. Thứ tự kết hợp các quan hệ cũng là một vấn đề cốt yếu mà bộ tối ưu phải giải quyết để tạo ra các kế hoạch tối ưu. Trong các hệ phân tán, đối với truy vấn trên dữ liệu không lớn, các hệ quản trị cơ sở dữ liệu thường xử lý bằng các thuật toán hiện có như thuật toán tối ưu hóa quy hoạch động (Dynamic Programming) cổ điển. Tuy nhiên, đối với các truy vấn phức tạp (có số lượng quan hệ nhiều hơn hoặc các truy vấn được thực thi phân bố trên nhiều địa điểm khác nhau), vấn đề tối ưu hóa sẽ trở nên khó khăn hơn và các thuật toán tối ưu hóa tập trung cổ điển sẽ rất khó khăn khi xử lý sự phức tạp này.

Sau khi đã xác định thứ tự thực hiện truy vấn theo các quan hệ trên các địa điểm khác nhau, bộ tối ưu truy vấn phải dựa trên một mô hình chi phí để quyết định lựa chọn kế hoạch thực thi nào là tối ưu nhất. Mô hình chi phí trong hệ tập trung khác với mô hình chi phí trong hệ phân tán vì hệ tập trung không quan tâm đến các phép toán có thể thực thi đồng thời trên nhiều địa điểm. Việc tìm ra một mô hình chi phí cho phép tính toán chi phí các phép toán có thể tiến hành song song trong hệ phân tán là một điều hết sức cần thiết.

Luận văn sẽ tập trung nghiên cứu các vấn đề liên quan đến quá trình tối ưu hóa truy vấn bao gồm các mục tiêu chính như không gian tìm kiếm kế hoạch thực thi truy vấn, thứ tự kết hợp các quan hệ trong truy vấn, mô hình chi phí và các thuật toán tối ưu hóa truy vấn trong hệ tập trung và hệ phân tán.

2.2 Các nghiên cứu liên quan

Trong quá trình tối ưu hóa truy vấn, một trong những vấn đề quan trọng cần tối ưu là phải làm giảm kích thước không gian tìm kiếm các kế hoạch thực thi tương đương của truy vấn. Từ đó làm giảm thời gian liệt kê và lựa chọn kế hoạch thực thi tối ưu đối với truy vấn đầu vào. Đã có rất nhiều công trình nghiên cứu và các thuật toán liên quan để giải quyết vấn đề này. Các thuật toán tối ưu hóa truy vấn (trên hệ tập trung) thường được xếp vào một trong ba loại thuật toán liệt kê sau đây: liệt kê toàn diện, liệt kê phỏng đoán kinh nghiệm và liệt kê ngẫu nhiên.

- **Thuật toán tìm kiếm toàn diện**

Đây là thuật toán có thời gian chạy xấu nhất theo hàm mũ và độ phức tạp không gian tìm kiếm theo cấp số nhân. Khi không gian tìm kiếm của truy vấn lớn, thuật toán có thể dẫn đến tình trạng tràn bộ nhớ và không thể tối ưu hóa truy vấn, bởi vì chi phí thực hiện việc phân tích tìm kiếm quá lớn. Các thuật toán tìm kiếm toàn diện thường liệt kê trên toàn bộ không gian tìm kiếm nên các thuật toán sẽ luôn luôn tìm thấy những kế hoạch tối ưu nhất định. Tuy nhiên, rất khó để áp dụng phương pháp vét cạn này để tìm kiếm kế hoạch tốt nhất đối với các truy vấn lớn có nhiều quan hệ [10].

Thuật toán tiêu biểu cho thuật toán tìm kiếm toàn diện là thuật toán quy hoạch động (Dynamic Programming).

- **Thuật toán phỏng đoán**

Thuật toán phỏng đoán được đề xuất với mục đích giải quyết các vấn đề thời gian chạy theo cấp số nhân của các thuật toán liệt kê toàn diện. Các thuật toán phỏng đoán dựa theo kinh nghiệm hoặc các quy tắc cụ thể để điều hướng tìm kiếm vào tập hợp con của toàn bộ không gian tìm kiếm.

Các thuật toán thuộc loại phỏng đoán tiêu biểu là thuật toán độ lựa chọn tối thiểu (Minimum Selectivity), phỏng đoán tham lam (Greedy heuristics) [11] và quy hoạch động lặp (IDP) biến thể [10]. Nguyên tắc chung khi sử dụng công nghệ phỏng đoán hầu hết đều là tìm kiếm kế hoạch tốt dựa trên kết quả của các tập hợp con trung gian nhỏ. Tuy nhiên, các thuật toán phỏng đoán vẫn có trường hợp chạy thời gian xấu và không gian hoạt động phức tạp.

- **Thuật toán ngẫu nhiên**

Thuật toán ngẫu nhiên xem xét không gian tìm kiếm là một tập hợp các phần tử, mỗi phần tử trong tập hợp đó tương ứng với một kế hoạch thực thi duy nhất. Một số thuật toán ngẫu nhiên tiêu biểu như Cải thiện lặp (Iterative Improvement-II) [24], mô phỏng luyện kim (Simulated Annealing-SA) [24], tối ưu hóa 2-pha (2PO) [27] và thuật toán di truyền [28]. Các thuật toán ngẫu nhiên thường chọn ngẫu nhiên một phần tử khởi đầu trong không gian và di chuyển đến các phần tử kế cận cũng được chọn ngẫu nhiên. Thuật toán ghi nhận lại tập hợp các phần tử cùng với chi phí di chuyển. Thuật toán lặp lại quá trình tìm kiếm với một phần tử khởi đầu mới. Sau một thời gian kiểm thử và so sánh, kế hoạch với chi phí thấp nhất sẽ được trả về.

Một số phương pháp khác đã được đề xuất như kỹ thuật viết lại truy vấn [25] và kỹ thuật đơn giản hóa truy vấn [26] sử dụng đồ thị truy vấn được giới hạn để nỗ lực làm giảm sự phức tạp của việc tối ưu hóa.

2.3 Quá trình xử lý truy vấn hệ phân tán

2.3.1 Danh mục hệ thống

Trong quá trình xử lý truy vấn, khi muốn dự đoán chi phí của các kế hoạch thực thi truy vấn, hệ quản trị cơ sở dữ liệu cần phải có các thông tin tham khảo như số lượng các quan hệ, số bộ dữ liệu trong một quan hệ, số lượng trang dữ liệu mà quan hệ đó đang chiếm giữ… Để có được các thông tin này, hệ quản trị cơ sở dữ liệu phải sử dụng thông tin được lưu trữ trong danh mục hệ thống.

Đối với hệ thống cơ sở dữ liệu tập trung, danh mục được sử dụng để lưu trữ chủ yếu là lược đồ (schema), bao gồm thông tin về các quan hệ, các chỉ mục (indexes) và khung nhìn (view). Thông tin của các quan hệ bao gồm tên quan hệ, tên thuộc tính, kiểu dữ liệu và các ràng buộc toàn vẹn cũng được lưu trữ trong danh mục. Ngoài ra, các số liệu thống kê khác cũng được lưu trữ trong danh mục như số lượng các khóa và lực lượng quan hệ.

Các thống kê này là cơ sở hỗ trợ cho bộ tối ưu truy vấn trong việc ước tính kích thước của kết quả trung gian của kế hoạch thực hiện, nghĩa là cho phép bộ tối ưu xác định ước tính chi phí của kế hoạch. Thông tin về tình trạng hệ thống hiện tại cũng được lưu trữ sẵn sàng trong danh mục, bao gồm số lượng các trang đệm trong vùng đệm dữ liệu và kích thước trang hệ thống [3].

Trong hệ quản trị cơ sở dữ liệu phân tán, danh mục hệ thống lưu trữ thêm các thông tin bổ sung bao gồm vị trí của các quan hệ và vị trí bản sao của các quan hệ. Danh mục cũng bao gồm các thông tin hệ thống như số lượng và định danh của các địa điểm trong hệ thống. Khi đề cập đến danh mục của hệ phân tán, chúng ta thấy một vấn đề quan trọng phát sinh đó chính là nơi để lưu trữ danh mục trong hệ thống. Có hai lựa chọn lưu trữ danh mục trong hệ thống bao gồm lưu trữ danh mục tại một địa điểm duy nhất hoặc sao chép nó đến tất cả địa điểm trong hệ phân tán.

Lưu trữ danh mục tại một địa điểm duy nhất sẽ gây ra nguy cơ tiềm ẩn nguy hiểm cho hệ thống. Nếu địa điểm chứa danh mục bị lỗi thì toàn bộ hệ thống sẽ không thể truy cập vào danh mục và có thể làm cho toàn bộ hệ thống phải ngừng lại. Lưu

trữ danh mục tại một địa điểm duy nhất còn có thể dẫn đến tình trạng suy giảm hiệu suất hệ thống, khi địa điểm chứa danh mục phải tiếp nhận quá nhiều truy vấn danh mục từ các địa điểm khác.

Lưu trữ danh mục tại nhiều địa điểm bằng cách tạo ra nhiều bản sao của danh mục ở mỗi địa điểm, sẽ giúp tăng tính sẵn sàng cho danh mục hệ thống và cũng làm giảm giao dịch mạng vì tất cả địa điểm không cần phải truy cập vào danh mục từ xa. Tuy nhiên, trong một hệ thống mà danh mục thường xuyên thay đổi, việc cập nhật danh mục cho toàn hệ thống sẽ mất rất nhiều thời gian. Lúc này hệ thống sẽ bị suy giảm hiệu năng vì phải liên tục đồng bộ hóa danh mục cho nhiều địa điểm [4].

2.3.2 Chi phí truyền tải mạng

Khi đánh giá truy vấn trong một hệ phân tán, chúng ta không những cần phải xem xét chi phí I/O trên đĩa cục bộ mà còn phải xem xét chi phí truyền tải mạng. Chi phí truyền tải mạng có sự ảnh hưởng đáng kể đến tổng thể chi phí thực hiện của một truy vấn. Trong hệ phân tán, địa điểm xuất phát của một truy vấn S_q rất quan trọng. Bất kỳ mô hình chi phí nào được sử dụng trong hệ phân tán cũng phải đưa vào tính toán điểm xuất phát của truy vấn. Vị trí địa điểm xuất phát truy vấn không chỉ ảnh hướng đến chi phí di chuyển dữ liệu từ các địa điểm khác đến địa điểm xuất phát truy vấn để làm dữ liệu đầu vào mà còn ảnh hưởng đến chi phí di chuyển kết quả đến địa điểm sử dụng kết quả truy vấn phân tán.

Luận văn giới hạn xem xét trường hợp các quan hệ không bị phân mảnh và được lưu trữ hoàn toàn tại các địa điểm. Như vậy các phép toán chiếu, chọn và kết hợp sẽ được thực hiện trên một quan hệ tại một địa điểm cụ thể. Sau đó, các kết quả sẽ được chuyển đến các địa điểm cần sử dụng hoặc trả về địa điểm xuất phát S_q [3].

Khi thực hiện việc kết hợp giữa hai quan hệ R_1 và R_2 trong một hệ phân tán, vấn đề địa điểm xuất phát, địa điểm thực hiện và địa điểm sử dụng kết quả kết hợp cần được xem xét kĩ càng. Vị trí các địa điểm này sẽ ảnh hưởng đến việc di chuyển, trao đổi toàn bộ dữ liệu của các quan hệ giữa các địa điểm và ảnh hưởng trực tiếp đến quá trình tạo ra phương án thực thi hiệu quả của quá trình kết hợp.

Các tình huống dẫn đến di chuyển dữ liệu giữa hai quan hệ R_1 và R_2:

- Trường hợp cả hai quan hệ R_1, R_2 có mặt tại cùng một địa điểm, quá trình kết hợp có thể được thực hiện hoàn toàn tại địa điểm cục bộ. Lúc này, chi phí của việc kết hợp chỉ liên quan đến chi phí truy cập đĩa, không bao gồm chi phí truyền thông. Do đó, kế hoạch thực thi truy vấn trong trường hợp này thường có chi phí thấp nhất.
- Trường hợp cả hai quan hệ R_1 và R_2 thuộc hai địa điểm khác nhau S_1 và S_2. Lúc này, địa điểm thực hiện quá trình kết hợp có thể được chọn là một trong hai địa điểm S_1 hoặc S_2. Giả sử S_1 là địa điểm xuất phát truy vấn. Kế hoạch thực thi truy vấn thường được áp dụng trong trường hợp này là di chuyển toàn bộ dữ liệu R_2 từ S_2 đến S_1 và thực hiện kết hợp tại S_1. Tuy nhiên, nếu xem xét tình huống dữ liệu của R_2 là rất lớn, dữ liệu của R_1 là rất nhỏ và dữ liệu kết quả của việc kết hợp (R_1, R_2) là rất nhỏ, thì trong trường hợp này, một kế hoạch tối ưu hơn là di chuyển dữ liệu R_1 đến S_2 và thực hiện kết hợp với R_2 tại S_2. Sau đó, kết quả (dữ liệu rất nhỏ) sẽ được di chuyển trả lại cho điểm truy vấn.
- Trường hợp cả hai quan hệ R_1 và R_2 thuộc hai địa điểm khác nhau S_1 và S_2. Nhưng địa điểm kết hợp S không nằm cùng một địa điểm với cả hai quan hệ R_1,R_2. Lúc này, kế hoạch thực thi sẽ phải di chuyển dữ liệu của cả hai quan hệ đến S. Trong trường hợp S cũng là điểm sử dụng truy vấn, chi phí di chuyển kết quả đến địa điểm sử dụng truy vấn sẽ được bỏ qua. Kế hoạch thực thi truy vấn lúc này sẽ được tối ưu trong trường hợp kết quả của phép kết có kích thước lớn hơn nhiều so với kích thước của các quan hệ đầu vào.

  Trong trường hợp S không phải là điểm sử dụng truy vấn, kế hoạch thực thi lúc này rất có thể sẽ trở thành một kế hoạch có chi phí lớn vì phải tính thêm chi phí di chuyển kết quả đến địa điểm sử dụng truy vấn. Tuy nhiên, trong trường hợp kết hợp giữa hai quan hệ chỉ là một phần của một truy vấn lớn, kế hoạch di chuyển này vẫn có thể mang lại lợi ích cho một kế hoạch thực thi tổng thể.

2.4 Các thách thức của hệ phân tán

Khi thiết kế tối ưu cho cơ sở dữ liệu phân tán, chúng ta sẽ đối mặt với các vấn đề mang tính thách thức cần phải giải quyết. Vấn đề đầu tiên liên quan đến kích thước của không gian tìm kiếm. Kích thước của không gian tìm kiếm trong một hệ thống tập trung đã rất lớn nhưng lúc này các quan hệ chỉ tồn tại tại một địa điểm. Trong cơ sở dữ liệu phân tán, các quan hệ tồn tại ở nhiều địa điểm và mỗi địa điểm tồn tại nhiều các bản sao của các quan hệ từ đó góp phần làm cho không gian tìm kiếm kế hoạch thực thi ngày càng lớn hơn nữa.

Vấn đề thứ hai khó khăn thứ hai là việc xác định các khả năng thực hiện các phép toán song song trong một kế hoạch nhất định để có thể xác định tổng thời gian thực hiện tối thiểu một cách chính xác.

2.4.1 Kích thước không gian tìm kiếm

Trong hệ phân tán, kích thước không gian tìm kiếm kế hoạch thực thi sẽ lớn hơn hệ tập trung đáng kể vì truy vấn có thể thực hiện phép kết tại một vị trí bất kỳ và sử dụng các bản sao của các quan hệ tại nhiều địa điểm khác nhau.

Xét không gian tìm kiếm của một truy vấn có $n+1$ quan hệ trong hệ phân tán. Trường hợp trường hợp xấu nhất, không gian tìm kiếm sẽ bao gồm tất cả các quan hệ của hệ thống có mặt ở tất cả các địa điểm. Chúng ta tiến hành xem xét ba cấp độ chú thích trên cây kế hoạch của một truy vấn có 3 quan hệ.

- Cấp độ 1 của cây kế hoạch chỉ gồm các nút kết hợp và tên quan hệ mà không có bất kỳ chú thích địa điểm quan hệ. Ví dụ về các kế hoạch ở cấp độ 1 được thể hiện trong *Hình 2.1*.
- Cấp độ 2 của cây kế hoạch có thêm chú thích địa điểm tại các nút lá (ví dụ như trong *Hình 2.2*).
- Cấp độ 3 là cây kế hoạch hoàn chỉnh có thêm chú thích di chuyển dữ liệu (ví dụ như trong *Hình 2.3*). Trong *Hình 2.3*, chú thích $O_{S_i \to S_j}$ biểu thị dữ liệu được di chuyển từ S_i sang S_j.

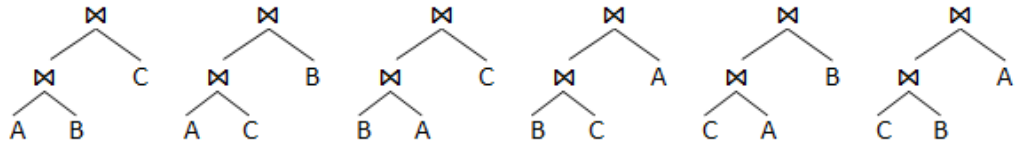
Ở cấp độ 1, số lượng các kế hoạch duy nhất với $n + 1$ quan hệ cấp lá sẽ tương đương với kích thước không gian tìm kiếm kế hoạch trong hệ thống cơ sở dữ liệu tập trung và được tính bằng $\frac{(2n)!}{n!}$ [2]

Cấp độ 2 thể hiện trong bối cảnh phân tán, chúng ta có thêm khả năng sử dụng quan hệ từ bất kỳ vị trí nào. Vì vậy, với mỗi cấu trúc cây cấp 1 có thể có được s^{n+1} trường hợp cây cấp 2 bằng cách xem xét tổ hợp tất cả các địa điểm liên quan. Nghĩa là số lượng kế hoạch ở cấp độ 2 được tính bằng công thức

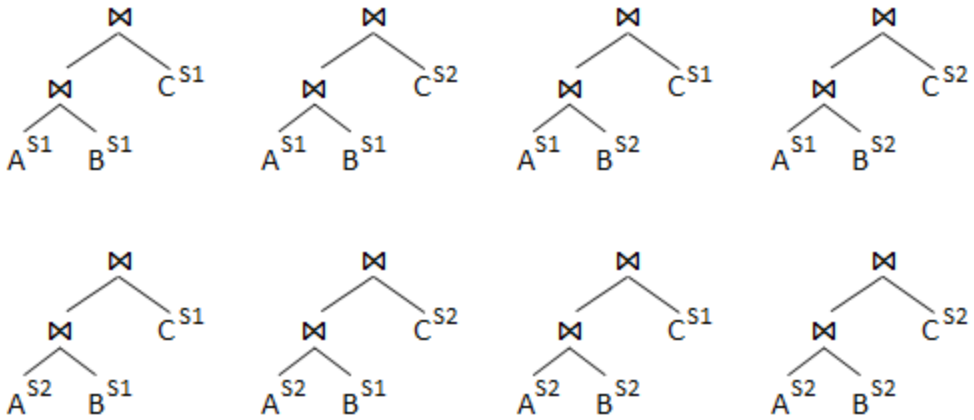
$$\frac{(2n)!}{n!} s^{n+1}$$

Với s là số lượng các địa điểm có trong hệ thống.

Ví dụ cây kế hoạch cấp 2 trong *Hình 2.2*, được áp dụng cho trường hợp hệ thống phân tán có 3 quan hệ *A, B, C* trên 2 địa điểm S_1 và S_2.



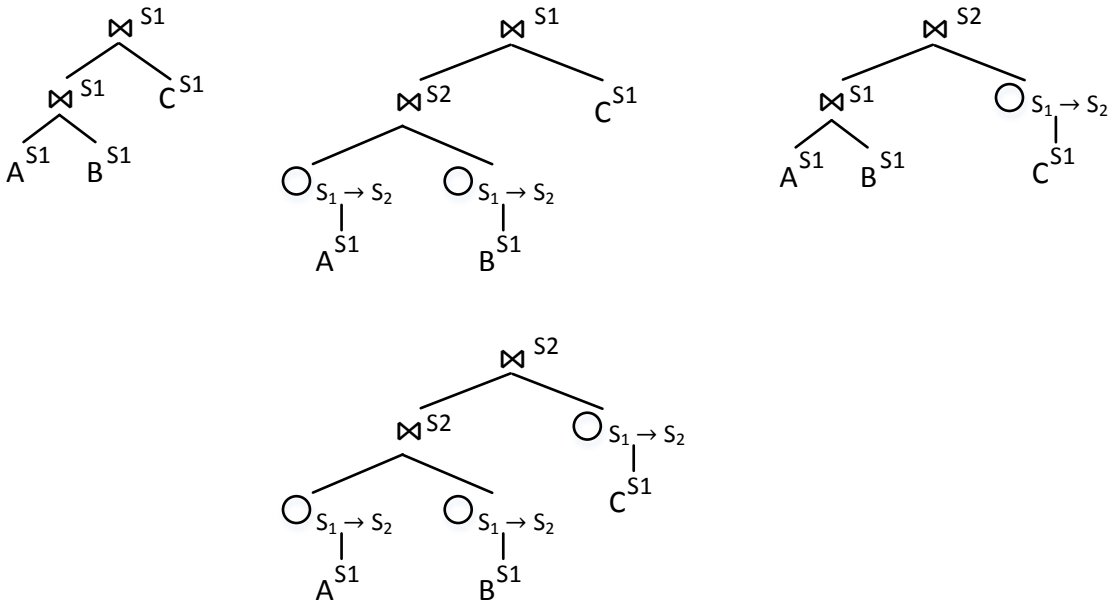
Hình 2.1 Cấu trúc cây nhị phân hoán vị các nút lá



Hình 2.2 Cây hoán vị quan hệ với chú thích địa điểm tại các nút lá

Đối với mỗi trường hợp cây ở cấp độ 2, chúng ta xem xét các địa điểm mà tại đó kết quả phép kết được chuyển đi. Bằng cách thực hiện tổ hợp tất cả vị trí kết hợp của n phép kết cho mỗi trường hợp cây ở cấp độ 2, chúng ta có được s^n kế hoạch cấp 3. Do đó, công thức tổng số lượng kế hoạch cấp độ 3 được tính như sau:

$$|S| \geq \frac{(2n)!}{n!} s^{n+1} s^n = \frac{(2n)!}{n!} s^{2n+1}$$



Hình 2.3 Cây hoán vị quan hệ với chú thích đầy đủ

2.4.2 Chi phí thiết lập kế hoạch truy vấn

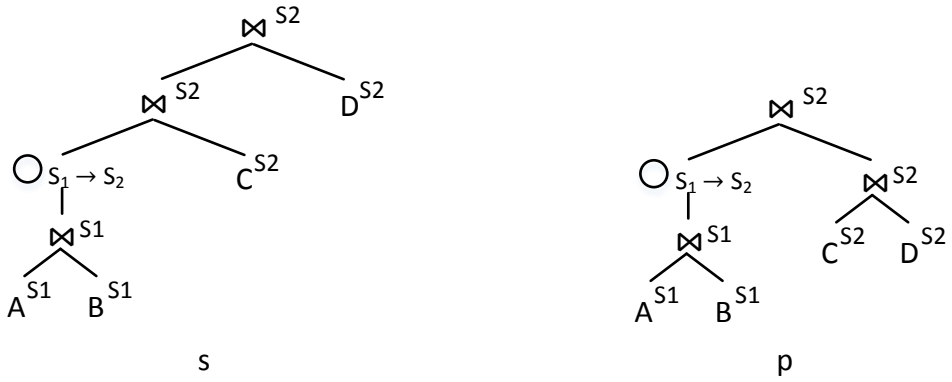
Trong hệ thống tập trung, chi phí của kế hoạch được định nghĩa bằng cách tổng hợp chi phí I/O ước tính của từng hoạt động riêng lẻ. Chi phí vào ra dữ liệu I/O cũng được sử dụng như là một thước đo của tổng thời gian thực thi hoặc tổng thời gian hồi đáp của kế hoạch. Điều quan trọng của một mô hình chi phí là phải phản ánh đúng các chi phí thực hiện thực tế của kế hoạch. Nếu mô hình chi phí không sát với thực tế, các kế hoạch thực thi tốt có thể bị bỏ qua trong quá trình liệt kê tìm kiếm kế hoạch tối ưu.

Trong một hệ quản trị cơ sở dữ liệu phân tán, nếu mô hình chi phí chỉ xem xét chi phí I/O là không đủ. Chi phí việc gửi nhận dữ liệu của các quan hệ giữa các địa điểm qua mạng cũng là một chi phí quan trọng tác động đến thời gian thực hiện tổng thể. Do đó chi phí này phải được đưa vào mô hình chi phí. Điều này có nghĩa là tốc độ truyền mạng và lưu lượng mạng cũng cần được xem xét.

Ngoài ra, một điểm cần lưu ý trong hệ quản trị cơ sở dữ liệu phân tán là sự tồn tại khả năng thực hiện song song các phần của một kế hoạch. Có hai nguồn song song chính: nội toán tử (intra-operator) và liên toán tử (inter-operator):

- Nội toán tử song song đề cập đến tình huống một phép toán được thực hiện bởi nhiều bộ xử lý một cách đồng thời. Điều này chỉ có thể có trong trường hợp quan hệ đã được phân mảnh trên các địa điểm.
- Liên toán tử là một thuật ngữ được dùng cho trường hợp hai phép toán được thực hiện đồng thời. Liên toán tử gồm hai trường hợp:
    - Trường hợp 1, dữ liệu phụ thuộc vào hai phép toán điều khiển thông qua một quan hệ ống dẫn (pipelining). Dữ liệu phụ thuộc nghĩa là một phép toán phải chờ đợi các dữ liệu đầu vào của nó trước khi nó có thể tiến hành.
    - Trường hợp 2, dữ liệu độc lập với hai phép toán độc lập thực hiện đồng thời. Khi đó sẽ có tình huống dẫn đến tranh chấp tài nguyên xảy ra khi mà nhiều tiến trình muốn truy cập vào cùng một tài nguyên.

Như đã đề cập trong một hệ tập trung, thời gian hồi đáp của một kế hoạch thực hiện cụ thể có thể được ước tính bằng cách xem xét tài nguyên tiêu thụ tức là tình trạng sử dụng đĩa. Tuy nhiên, trong một hệ phân tán, do tồn tại khả năng thực hiện song song trên nhiều địa điểm nên cách tính tài nguyên tiêu thụ không thể hiện được thời gian đáp ứng kế hoạch. Điều này được minh họa trong ví dụ sau. Ví dụ sẽ so sánh tính toán chi phí thời gian đáp ứng của hai kế hoạch tuần tự và kế hoạch song song.



Hình 2.4 So sánh kế hoạch tuần tự và kế hoạch song song

Các ký hiệu sử dụng trong ví dụ này:

• *s:* kế hoạch tuần tự.

• *p:* kế hoạch song song.

• y_i*:* số lượng I/O đĩa được thực hiện bởi kế hoạch $i, i \in \{p, s\}$

• *x:* số lượng các trang dữ liệu chuyển qua mạng.

• t_d*:* thời gian cần thiết để đọc/ghi một trang dữ liệu vào đĩa.

• t_n*:* thời gian cần thiết để chuyển một trang dữ liệu qua mạng từ một địa điểm này đến địa điểm khác.

• *a, b:* các yếu tố được sử dụng để đo các số liệu tài nguyên tiêu thụ.

• $C_{RC}(l)$*:* chi phí tài nguyên tiêu thụ của kế hoạch *l*.

• $C_T(l)$*:* chi phí thời gian đáp ứng của kế hoạch *l*.

• *h:* kế hoạch con (subplan) $A^{S1} \bowtie^{S1} B^{S1}$

Xét trường hợp tính chi phí của các kế hoạch trong Hình 2.4 dựa trên tài nguyên tiêu thụ, hai yếu tố *a* và *b* được sử dụng để chuyển đổi số lượng I/O và truyền mạng tương ứng khi so sánh kết quả. Các công thức tính chi phí tiêu thụ tài nguyên C_{RC} của mỗi kế hoạch được thể hiện ở công thức (2.1) và (2.2). Lưu ý rằng cả hai kế hoạch chuyển giao cùng một lượng dữ liệu giữa các địa điểm, do đó chúng ta có chi phí truyền mạng *xb* như nhau.

Giả sử rằng kế hoạch con $A^{S1} \bowtie^{S1} B^{S1}$ có chi phí thực hiện I/O ít hơn kế hoạch con $C \bowtie D$, và xét tình huống $C_{RC}(s) < C_{RC}(p)$. Mô hình chi phí dựa trên tài nguyên

tiêu thụ, sẽ cho thấy kế hoạch *s* là kế hoạch có chi phí thực hiện tốt hơn so với kế hoạch *p*, vì *s* sử dụng tài nguyên ít hơn.

$$C_{RC}(s) = ay_s + xb \quad (2.1)$$

$$C_{RC}(p) = ay_p + xb \quad (2.2)$$

$$C_{RC}(h) = ay_h \quad (2.3)$$

$$C_{RC}(s) < C_{RC}(p) < C_{RC}(s) + C_{RC}(h) \quad (2.4)$$

Bất đẳng thức 2.4 có nghĩa là

$$y_s < y_p < y_s + y_h \quad (2.5)$$

Để xác định chi phí của một kế hoạch dựa trên cách tính thời gian đáp ứng, chúng ta chỉ đơn giản là nhân số I/O với thời gian thực hiện đọc/ghi một trang dữ liệu và nhân số trang dữ liệu truyền qua mạng với thời gian truyền tải một trang dữ liệu tương ứng. Công thức xác định chi phí thời gian đáp ứng được tính như sau:

$$C_T(s) = y_s t_d + xt_n \quad (2.6)$$

$$C_T(h) = y_h t_d \quad (2.7)$$

$$\begin{aligned} C_T(p) &= y_p t_d + xt_n - C_T(h) \quad (2.8)\\ &= y_p t_d + xt_n - y_h t_d \\ &= (y_p - y_h) t_d + xt_n \\ &< y_s t_d + xt_n \text{ (do 2.5)} \\ &= C_T(s) \end{aligned}$$

Trong kế hoạch *p*, có thể thực hiện phép kết $A \bowtie B$ và $C \bowtie D$ đồng thời tại hai địa điểm khác nhau S_1 và S_2. Điều này có nghĩa rằng chi phí của kế hoạch *p* sẽ bao gồm khoản thời gian tối đa cần thiết để hoàn thành cả hai phép kết $A^{S1} \bowtie^{S1} B^{S1}$ và $C^{S2} \bowtie^{S2} D^{S2}$ cùng với thời gian chuyển mạng $S_1 \rightarrow S_2$ và thời gian thực hiện phép kết cuối cùng. Do đó, thời gian đáp ứng của *p* sẽ được tính bằng thời gian thực hiện tuần tự các bước của *p* trừ cho thời gian ($C_T(h)$), công thức (2.8). Trong ví dụ này, chúng ta thấy rằng mặc dù kế hoạch *s* có một chi phí tài nguyên tiêu thụ thấp hơn ($C_{RC}(s) < C_{RC}(p)$), nhưng kế hoạch *p* vẫn có chi phí thấp hơn theo khía cạnh thời gian đáp ứng ($C_T(p) < C_T(s)$).

Như vậy mô hình chi phí trong hệ phân tán phải có khả năng xác định sự di chuyển mạng và các kế hoạch thực thi song song giữa nhiều địa điểm của một truy vấn. Từ đó mô hình chi phí mới có thể tính được thời gian đáp ứng chính xác.

2.5 Hướng nghiên cứu của đề tài

Đề tài giải quyết các thách thức và khó khăn của việc tối ưu hóa truy vấn trong hệ phân tán thông qua việc nghiên cứu, mở rộng các thuật toán tối ưu hóa truy vấn trên hệ tập trung thành thuật toán tối ưu hóa truy vấn sử dụng được trên hệ phân tán. Các thuật toán tối ưu hóa truy vấn trên hệ tập trung như DP, IDP và DPccp vẫn còn một số điểm yếu:

- Thuật toán quy hoạch động lặp (Iterative Dynamic Programming-IDP) [10] đã được đề xuất để khắc phục những vấn đề phức tạp không gian của DP. Tuy nhiên, cả hai thuật toán DP và IDP đều không xem xét các cấu trúc truy vấn hoặc đồ thị kết hợp của truy vấn, dẫn đến trường hợp phải xem xét các tích chéo. Tích chéo thực hiện việc kết hợp toàn bộ các quan hệ với nhau nên rất tốn kém và không hiệu quả.
- Thuật toán DPccp [18] đề xuất sử dụng đồ thị kết hợp của một truy vấn để thực hiện quy hoạch động mà không xem xét các tích chéo. DPccp đã có sự cải tiến rõ rệt so với thuật toán DP. Tuy nhiên DPccp vẫn có trường hợp có cùng thời gian thực thi kém như DP.

Trong phần tiếp theo đầu tiên đề tài sẽ tìm hiểu các ưu khuyết điểm của các thuật toán tối ưu hóa truy vấn trên hệ tập trung và phương pháp mở rộng để tối ưu hóa truy vấn trên hệ phân tán. Sau đó, đề tài sẽ trình bày thuật toán IDP1ccp kết hợp hai thuật toán IDP1 và DPccp để tạo nên một thuật toán hiệu quả hơn.

CHƯƠNG 3: CÁC THUẬT TOÁN TỐI ƯU HÓA TRUY VẤN

3.1 Thuật toán quy hoạch động

3.1.1 Mô tả thuật toán

Bản chất của thuật toán quy hoạch động truyền thống (Dynamic Programming-DP) là một thuật toán tối ưu hóa truy vấn trên hệ tập trung. Tuy nhiên, thuật toán hoàn toàn có khả năng mở rộng để sử dụng tối ưu hóa truy vấn cho hệ phân tán. Thuật toán quy hoạch động DP thuộc vào loại thuật toán tìm kiếm toàn diện phổ biến, đã được sử dụng trong một số hệ thống quản lý cơ sở dữ liệu thương mại ví dụ như là hệ thống IBM DB2 [13]. Mục tiêu của thuật toán là xây dựng cây kế hoạch tối ưu từ những kế hoạch con tối ưu theo hướng từ dưới lên trên (bottom up).

Gọi *C(p)* là chi phí của một kế hoạch con bắt nguồn từ nút p. Nguyên tắc tối ưu được phát biểu rằng nếu hai kế hoạch chỉ khác nhau bởi một kế hoạch con, thì kế hoạch với *C(p)* thấp hơn, cũng sẽ có một kế hoạch chi phí thấp hơn [16], ví dụ so sánh kế hoạch $D \bowtie (A \bowtie B)$ và kế hoạch $D \bowtie (A \bowtie C)$, nếu phép kết $(A \bowtie C)$ có chi phí thấp hơn phép kết $(A \bowtie B)$ thì kế hoạch $D \bowtie (A \bowtie C)$ sẽ có chi phí thấp hơn.

Chương trình thuật toán quy hoạch động được thể hiện trong Hình 3.1, thuật toán có đầu vào là tập hợp các quan hệ (các bảng) có trong truy vấn. Kết quả của thuật toán là kế hoạch thực thi tối ưu của truy vấn đầu vào. Thuật toán thực hiện như sau:

- Ở giai đoạn đầu tiên, thuật toán tìm kiếm các kế hoạch tối ưu cho một quan hệ. Các kế hoạch truy cập *accessPlans* cần tối ưu trên một quan hệ thông thường là các chi phí như quét toàn bảng hoặc quét theo chỉ mục được áp dụng trên một quan hệ. Hàm *prunePlans* tiếp theo ở dòng 4 Hình 3.1, dùng để loại bỏ các kế hoạch truy cập có chi phí cao khi thực hiện trên một quan hệ. Các *accessPlans* tối ưu của các quan hệ sau đó được đưa vào các kế hoạch tối ưu *optPlans* dùng làm các khối xây dựng cho các kế hoạch tiếp theo.
- Trong giai đoạn thứ hai từ dòng 6-15, thuật toán tối ưu hóa các kế hoạch con tạo ra từ 2 quan hệ cho đến *n* quan hệ. Thuật toán xem xét tất cả các tổ hợp kết hợp của các quan hệ. Trước tiên, thuật toán liệt kê tất cả kế hoạch kết hợp 2-

quan hệ bằng cách gọi hàm *joinPlans* để xây dựng kế hoạch kết hợp từ những kế hoạch *accessPlans* tối ưu của một quan hệ đã tạo ra trong giai đoạn trước đó. Tiếp theo, các kế hoạch có chi phí cao sẽ được cắt tỉa bằng hàm cắt tỉa *prunePlans* (dòng 12 Hình 3.1), chỉ để lại các kế hoạch kết hợp của 2-quan hệ với chi phí thấp nhất trong các kế hoạch kết hợp.

Từ kế hoạch kết hợp 2-quan hệ đã được tối ưu và các kế hoạch *accessPlans* tối ưu một quan hệ, thuật toán lại tạo ra các kế hoạch kết hợp 3-quan hệ và tiếp tục tối ưu bằng hàm cắt tỉa để lại các kế hoạch kết hợp 3-quan hệ tốt nhất. Tương tự, thuật toán tiếp tục tạo ra các kế hoạch kết hợp 4-quan hệ, 5-quan hệ, 6-quan hệ cho đến các kế hoạch kết hợp *n*-quan hệ.

- Giai đoạn thứ 3 từ dòng 16-18, kế hoạch kết hợp *n*-quan hệ được xử lý bởi hàm *finalizePlans* (dòng 16 Hình 3.1) để trở thành kế hoạch hoàn chỉnh. Hàm *finalizePlans* sẽ bổ sung thêm vào kế hoạch thực thi các phép toán ngoài phép kết như phép chiếu, phép chọn với ghi chú thực hiện theo phương pháp tuần tự hoặc ống dẫn (materialized/pipelined). Ở bước cuối cùng, khi kế hoạch tổng thể đã hình thành, hàm *prunePlans* (dòng 17 Hình 3.1) chỉ cho phép một kế hoạch tốt nhất được tồn tại. Và kế hoạch tốt nhất này được dùng làm kết quả trả về của thuật toán (dòng 18 Hình 3.1).

3.1.2 Mở rộng trong môi trường phân tán

Một điểm nổi bật của thuật toán quy hoạch động là khả năng mở rộng dễ dàng. Thuật toán quy hoạch động không những có thể áp dụng cho hệ thống tập trung mà còn có thể mở rộng để tối ưu cho hệ phân tán. Trong hệ phân tán, ngoài việc phải quyết định dùng cách thức truy cập quan hệ nào, thứ tự kết hợp nào và phương pháp kết hợp nào như trong hệ tập trung, thì bộ tối ưu truy vấn phải quyết định địa điểm nào để thực hiện các phép toán.

Các hàm *accessPlans*, *joinPlans*, *finalizePlans* và *prunePlans* sẽ được mở rộng như sau để sử dụng trên hệ phân tán [10]:

- Nếu quan hệ đã được nhân bản, hàm *accessPlans* phải được phát sinh kế hoạch truy cập cho mỗi địa điểm mà quan hệ được nhân bản. Ví dụ quan hệ A được nhân bản tại 2 vị trí S_1 và S_2 thì các kế hoạch truy cập *accessPlans* cho quan hệ A có thể là quét bảng A ở S_1, quét bảng A ở S_2, quét chỉ mục bảng A ở S_1 và quét chỉ mục bảng A ở S_2.
- Hàm *joinPlans* phải phát sinh các kế hoạch kết hợp khác nhau để xác định kết hợp có thể thực hiện với các quan hệ trong cùng một địa điểm hoặc ở ngoài địa điểm kết hợp.
- Hàm *finalizePlans* phải tính thêm hoạt động di chuyển mạng, nếu phép toán của kế hoạch không thực hiện tại địa điểm kết hợp hoặc kết quả của truy vấn cần được trả về địa điểm khác.
- Hàm *prunePlans* cũng cần được điều chỉnh thận trọng khi cắt tỉa các kế hoạch có kết quả ở địa điểm khác. Ví dụ bảng A được lưu tại S_1, S_2 và bảng B chỉ lưu tại S_1, chúng ta không cắt tỉa kế hoạch truy cập quét bảng A tại S_1 trong truy vấn A ⋈ B ngay cả khi kế hoạch này lớn hơn quét bảng A tại S_2, bởi vì kết hợp A và B tại S_1 tạo ra kế hoạch tốt hơn vì không phải tốn thêm chi phí di chuyển kết quả. Tuy nhiên kế hoạch này có thể được cắt tỉa nếu kết quả cần di chuyển đến S_2.

3.1.3 Chương trình thuật toán DP

❖ **Thuật toán DYNAMIC-PROGRAMING**($R=\{R_1,R_2,R_3,\ldots,R_n\}$) [10]

Mục tiêu: *Xây dựng kế hoạch tối ưu từ các kế hoạch con sử dụng quy hoạch động*
Đầu vào: *Tập hợp các quan hệ trong một truy vấn* $R=\{R_1,R_2,R_3,\ldots,R_n\}$
Kết quả: *Kế hoạch thực thi truy vấn tối ưu.*

```
1  //Giai đoạn tạo ra các kế hoạch tối ưu cho 1 quan hệ
2  for i = 1 to n do {
3          optPlan({Ri}) = accessPlans({Ri})
4          prunePlans(opt-plan({Ri}))
5  }
```

```
6   for i=2 to n do {
7           //Giai đoạn tạo ra kế hoạch con tối ưu có i quan hệ
8           for all S ⊂ R such that |S| = i do{
9                   optPlan(S) =∅
10                  for all O ⊂ R where O≠ ∅ do{
11                          optPlan(S)= optPlan(S) ∪
                                    joinPlans(optPlan(O),optPlan(S\O))
12                          prunePlans(optPlan(S))
13                  }
14          }
15  }
16  finalizePlans(optPlan(R))
17  prunePlans(optPlan(R))
18  return optPlan(R)
```

Hình 3.1 Thuật toán Quy hoạch động truyền thống

3.2 Thuật toán quy hoạch động lặp

Hầu hết các bộ tối ưu hóa truy vấn ngày nay đều dựa trên thuật toán quy hoạch động để tối ưu hóa truy vấn. Bên cạnh các ưu điểm như dễ dàng mở rộng và tìm kiếm được kế hoạch thực thi tối ưu của thuật toán quy hoạch động, thuật toán cũng có hạn chế vì độ phức tạp cao khi áp dụng trên các truy vấn phức tạp và trong môi trường phân tán. Thuật toán tối ưu hóa truy vấn quy hoạch động lặp (Iterative Dynamic Programming-IDP) được xem như là một thuật toán cải tiến của thuật toán quy hoạch động. Thuật toán IDP có thể tạo ra kế hoạch tối ưu cho hầu hết các bài toán tối ưu truy vấn khi mà áp dụng thuật toán quy hoạch động sẽ có độ phức tạp cao. IDP có thể dễ dàng tích hợp vào bộ tối ưu đang sử dụng quy hoạch động.

Hạn chế chủ yếu của thuật toán DP (mục 3.1) là phát sinh không gian tìm kiếm kế hoạch thực thi rất lớn. Từ đó thuật toán DP dễ dàng lấp đầy hoàn toàn bộ nhớ chính khi áp dụng trên một truy vấn khá lớn (> 20 quan hệ). Trong trường hợp này, chương trình tối ưu hóa có thể làm hiệu suất hệ thống suy giảm đáng kể. Thuật toán quy hoạch động lặp (IDP) đã được đề xuất để giải quyết vấn đề phức tạp không gian của DP.

IDP kết hợp quy hoạch động với thuật toán phỏng đoán tham lam (greedy heuristic) giúp lựa chọn giá trị tốt nhất ở mỗi bước lặp để khắc phục những hạn chế về bộ nhớ. Trong phần sau sẽ trình bày các nguyên tắc cơ bản của IDP dựa trên biến thể kế hoạch tốt nhất tiêu chuẩn (standard best plan) là IDP1 [10].

3.2.1 Mô tả thuật toán

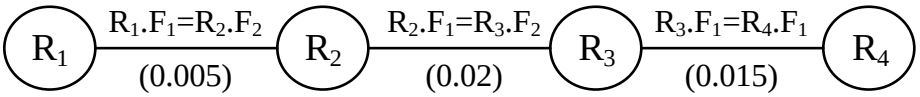
Xét một truy vấn có n quan hệ $R = \{R_1, R_2, ..., R_n\}$ cần được tối ưu hóa trong một hệ thống cơ sở dữ liệu tập trung. Giai đoạn đầu của thuật toán IDP1 vẫn giống như thuật toán DP. Các kế hoạch tốt sẽ được giữ trong *optPlan* và hủy bỏ các kế hoạch chi phí cao. Giả sử rằng bộ nhớ có sẵn của hệ thống chỉ đủ để lưu trữ tất cả các kế hoạch tối đa *k-quan hệ* sau khi đã cắt tỉa, tức là các kế hoạch bao gồm *2-quan hệ*, *3-quan hệ*, ..., *k-quan hệ*, với $k < n$. Lúc này, nếu tiếp tục sử dụng thuật toán DP sẽ không còn hiệu quả vì bộ nhớ của máy tính đã cạn kiệt và không thể xây dựng tiếp kế hoạch *(k+1)-quan hệ*. Thay vì tạo ra các kế hoạch *(k+1)-quan hệ*, IDP1 sẽ ngắt ra khỏi giai đoạn quy hoạch động và áp dụng hàm đánh giá chi phí *eval* (dòng 17 Hình 3.8) dựa trên thuật toán phỏng đoán tham lam để lựa chọn một kế hoạch *P* có giá trị thấp nhất trong tất cả các kế hoạch *k-quan hệ*.

Từ thời điểm này trở đi, kế hoạch *P* được chuyển thành một kế hoạch tạm thời *T*. Tập hợp các quan hệ trong *T* được ký hiệu là R_T (dòng 18 Hình 3.8). Lúc này, tất cả các *accessPlans* và *joinPlans* liên quan đến các quan hệ có trong kế hoạch *P* sẽ được hủy bỏ. Ví dụ: IDP1 đã lựa chọn được kế hoạch *P* tối ưu gồm có 3 quan hệ {A, B, D} thì nó sẽ hủy bỏ các kế hoạch 1-quan hệ như A, B, D và các kế hoạch 2-quan hệ và 3-quan hệ có chứa A, B, D. Kế hoạch tốt nhất của *P* đã được chuyển thành kế hoạch *T*.

Sau đó thuật toán lặp lại giai đoạn quy hoạch động giữa các quan hệ còn lại trong *R* và quan hệ tạm thời R_T. Quá trình lại ngắt ra khỏi giai đoạn quy hoạch động khi đạt đến bước *k-quan hệ*, thuật toán tiếp tục sử dụng phỏng đoán tham lam để tìm ra kế hoạch tối ưu. Quá trình sẽ dừng lại khi không còn quan hệ trong *R*. Thuật toán IDP sẽ được lặp lại ít nhất 2 lần ví dụ với *n=5, k=3* thì cần 2 lần lặp; với *n=10, k=4* thì cần đến 3 lần lặp. [10].

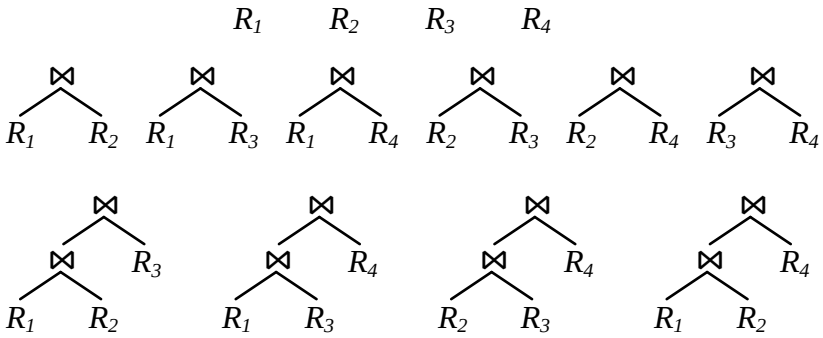
3.2.2 Ví dụ thuật toán IDP1 kế hoạch tốt nhất tiêu chuẩn

Ví dụ sử dụng thuật toán IDP1 để tối ưu hóa truy vấn 4 quan hệ (*n=4*) trong Hình 3.2. Cạnh nối giữa 2 quan hệ được ghi chú thông tin về điều kiện kết hợp và giá trị độ lựa chọn, các thông tin này trình bày trong mục 4.1.2.



Hình 3.2 Ví dụ đồ thị kết hợp

Giả sử rằng bộ nhớ hệ thống chỉ có thể giữ tất cả các kế hoạch giới hạn đến kế hoạch 3-quan hệ, tức là $k = 3$. Và giả sử kế hoạch được giữ lại trong cấu trúc *optPlan* sau khi cắt tỉa trong giai đoạn DP đầu tiên thể hiện trong Hình 3.3.

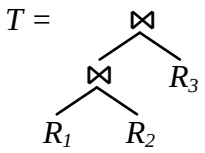


Hình 3.3 Các kế hoạch trong giai đoạn đầu DP

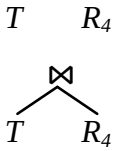
Đến giai đoạn kế hoạch 3-quan hệ đã được tạo ra, thuật toán áp dụng phỏng đoán tham lam lựa chọn phương án 3-quan hệ *P* có chi phí đánh giá thấp nhất. Giả sử *P* được chọn tương ứng với kế hoạch chứa các quan hệ R_1, R_2 và R_3 trong Hình 3.4, sau đó *P* được thay thế bằng quan hệ tạm thời *T*. Tất cả các kế hoạch có quan hệ liên quan đến R_1, R_2, R_3 được loại bỏ khỏi bộ nhớ.

Thuật toán lặp lại thực hiện giai đoạn DP lần 2 giữa *T* và *accessPlans* R_4. Kết quả thu được từ giai đoạn DP lần 2 thể hiện trong Hình 3.5. Cuối cùng bằng cách thay thế *T* bằng kế hoạch *P*, chúng ta có được kế hoạch tối ưu cho các quan hệ ban đầu Hình 3.6.

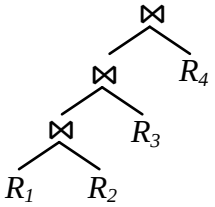
.



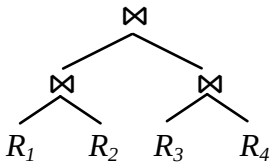
Hình 3.4 Kế hoạch được chọn bởi hàm chi phí



Hình 3.5 Giai đoạn DP lần hai



Hình 3.6 Kế hoạch thu được từ thuật toán IDP1 kế hoạch tốt nhất tiêu chuẩn



Hình 3.7 Kế hoạch tối ưu

3.2.3 Biến thể IDP cân bằng

Khi sử dụng thuật toán IDP1 kế hoạch tốt nhất tiêu chuẩn, do giá trị *k* làm giới hạn kích thước các khối xây dựng (các kế hoạch con), kế hoạch tối ưu cuối cùng sẽ không bao giờ đạt được kế hoạch với các nút lá tăng trưởng nhanh (bushy plan) thể hiện trong Hình 3.7. Phiên bản biến thể IDP1 dòng tốt nhất cân bằng (balanced best row) sẽ sử dụng giá trị cân bằng *b*, thay thế cho giá trị *k*, làm giới hạn kích thước các khối xây dựng để tạo ra được kế hoạch như trong Hình 3.7.

Các tiêu chí giới hạn kích thước khối xây dựng dựa trên các yêu cầu sau [10].

- Số lượng các quan hệ trong một kế hoạch con được chọn phải là một số chẵn
- Số lượng các quan hệ trong một kế hoạch con $n_{R_{subplan}} \leq \frac{d}{2}$, với *d* là số lượng các quan hệ khi bắt đầu giai đoạn DP hiện hành (các quan hệ trong danh sách *ToDo*).

Trong biến thể IDP1 dòng tốt nhất cân bằng, khi xác định kế hoạch *P* với giá trị *eval* thấp nhất, thay vì chỉ lựa chọn một kế hoạch *P* trong *optPlan* làm quan hệ tạm *T*, thuật toán IDP1 dòng tốt nhất sao chép tất cả các kế hoạch *P* trong *optPlan* cho *T*. Điều này có nghĩa là thuật toán sẽ có nhiều cơ hội hơn lựa chọn được kết quả tối ưu.

❖ **Thuật toán IDP1 dòng tốt nhất cân bằng** [10].

Đầu vào: *Tập hợp các quan hệ trong một truy vấn* $R=\{R_1,R_2,R_3,\ldots,R_n\}$
và số khối tối đa $k>1$

Kết quả: *Kế hoạch thực thi truy vấn tối ưu.*

Thuật toán IDP1($R=\{R_1,R_2,R_3,\ldots,R_n\},k$)

```
1  for i = 1 to n do {
2          optPlan({Ri}) = accessPlans({Ri})
3          prunePlans(opt-plan({Ri}))
4  }
5  toDo=R
6  while |toDo| > 1 do{
7          b=Balanced-Parameter(|toDo|,k)
8          for i=2 to b do {
9                  for all S ⊂ R such that |S| = i do{
10                         optPlan(S) =∅
11                         for all O ⊂ R where O≠ ∅ do{
12                                 optPlan(S) = optPlan(S ∪
                                           joinPlans(optPlan(O),optPlan(S-O))
13                                 prunePlans(optPlan(S))
14                         }
15                 }
16         }
17         find P,N with P ∈ optPlan(N), N ⊂ toDo,|N|=b such that
                   eval(P)=min{eval(P') | P' ∈ optPlan(W),W ⊂ toDo,|W|=b}
18         generate new symbol: T
19         optPlan({T})= optPlan(N)
20         toDo = toDo – N ∪ {T}
21         for all O ⊂ N do{
22                 delete(optPlan(O))
23         }
24 }
```

```
25  finalizePlans(optPlan(R))
26  prunePlans(optPlan(R))
27  return optPlan(R)
```

Hình 3.8 Chương trình thuật toán IDP1 dòng tốt nhất cân bằng

❖ **Thủ tục tính tham số cân bằng b** [10].

Đầu vào: *Số lượng các quan hệ d và số khối tối đa k*

Kết quả: *Tham số cân bằng b*

Thủ tục: ***BALANCED-PARAMETER***(*d,k*)

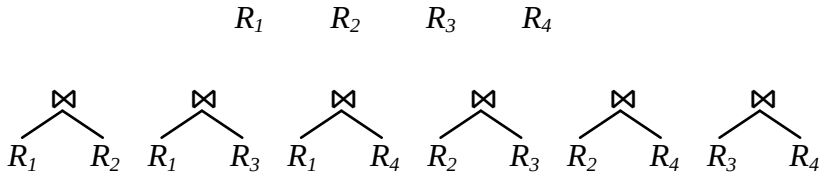
```
1  b=d
2  if b>k do{
3          b=ceil(b/2)
4          if b ≡ 1 mod 2 do{
5                  b=b-1
6          }
7          b=min(b,k)
8  }
9  return b
```

Hình 3.9 Thuật toán tính tham số cân bằng b

3.2.4 Ví dụ thuật toán IDP1 kế hoạch tốt nhất cân bằng

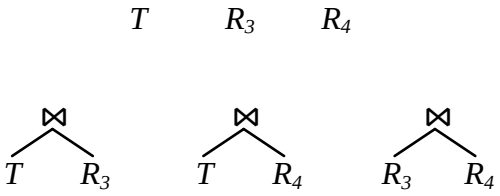
Giả sử bộ nhớ hệ thống chỉ có thể lữu trữ tất cả các kế hoạch *k-quan hệ* với k = 3. Tham số cân bằng b là số nguyên được tính bằng d là số lượng các quan hệ khi bắt đầu giai đoạn DP hiện hành với $b \leq \frac{d}{2}$ và $b < k$ Hình 3.9.

Giai đoạn DP đầu tiên trong ví dụ này d = 4, k =3 nên b = 2. Thay vì xây dựng tất cả các kế hoạch *3-quan hệ*, như kế hoạch IDP1 kế hoạch tốt nhất tiêu chuẩn, chương trình chỉ xây dựng đến kế hoạch *2-quan hệ*. Nội dung của cấu trúc *optPlan* trong giai đoạn DP đầu tiên thể hiện trong Hình 3.10.



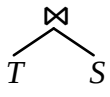
Hình 3.10 Giai đoạn DP đầu tiên của IDP1 kế hoạch tốt nhất cân bằng

Sau đó, chương trình lựa chọn phương án với giá trị *eval* thấp nhất là kế hoạch *P* chứa R_1 và R_2. Tất cả các kế hoạch liên quan đến các quan hệ R_1 và R_2 trong kế hoạch *P* được loại bỏ khỏi bộ nhớ và quan hệ tạm thời *T* được sử dụng để làm đại diện cho *P*. Các quan hệ chờ đợi để được tối ưu hóa lúc này là *T*, R_3 và R_4. Thuật toán lặp lại giai đoạn DP thứ hai với thông số cân bằng lúc này là 2 ($d=2,k=3$). Nội dung cấu trúc *optPlan* giai đoạn DP thứ hai (sau khi cắt tỉa) được thể hiện trong Hình 3.11.

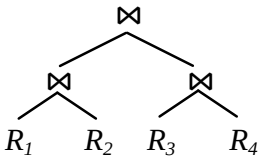


Hình 3.11 Giai đoạn DP thứ hai của IDP1 kế hoạch tốt nhất cân bằng

Sau đó, thuật toán lựa chọn phương án *2-quan hệ P'* với giá trị *eval* thấp nhất, trong trường hợp này *P'* là kế hoạch này liên quan đến R_3 và R_4. Thuật toán loại bỏ tất cả các kế hoạch khác liên quan đến R_3, R_4 và chuyển *P'* thành quan hệ tạm thời *S*. Giai đoạn DP cuối cùng sẽ chỉ có 2 quan hệ tạm thời *T* và *S* Hình 3.12. Kế hoạch tối ưu cuối cùng thể hiện trong Hình 3.13.



Hình 3.12 Giai đoạn DP cuối cùng của IDP1 kế hoạch tốt nhất cân bằng



Hình 3.13 Kế hoạch tối ưu của IDP1 kế hoạch tốt nhất cân bằng

3.3 Thuật toán quy hoạch động cặp đồ thị con liên thông bù

Thuật toán quy hoạch động cặp đồ thị con liên thông bù DPccp (Dynamic Programming connected subset complement pair) cũng là một thuật toán liệt kê quy hoạch động nhưng cách liệt kê các kế hoạch thực thi khác với hai thuật toán DP và IDP. Thuật toán DPccp xem xét truy vấn cần tối ưu như một đồ thị liên thông có các đỉnh là các quan hệ trong câu truy vấn. Trong khi đó, cả hai thuật toán DP và IDP xem xét truy vấn như là một tập hợp các quan hệ $R = \{R_1, R_2, ..., R_n\}$ và xem xét tất cả các tổ hợp kết hợp của các quan hệ có thể xảy ra. Cách liệt kê này rất phù hợp khi thực hiện với một truy vấn có đồ thị kết hợp dạng chùm.

Tuy nhiên, trong trường hợp thực hiện với một truy vấn dạng chuỗi hoặc vòng, cách liệt kê này có thể gây ra các trường hợp tích chéo dư thừa không cần thiết. Tích chéo hay còn gọi là tích Đề các (Cartesian product) là một hoạt động tốn kém khi xây dựng kế hoạch tối ưu. Thuật toán DPccp khắc phục được điểm yếu này, thuật toán DPccp chỉ liệt kê tổ hợp các quan hệ tồn tại trên đồ thị truy vấn, do đó thuật toán không bị trường hợp xem xét các tích chéo [18]. Nhờ vậy, thuật toán DPccp có thể tiết kiệm thời gian đáng kể khi tối ưu hóa truy vấn dạng chuỗi và dạng vòng.

3.3.1 Các định nghĩa liên quan

- **Tập hợp con/đồ thị con liên thông**

Xét đồ thị $G = (V, E)$ của một truy vấn có các đỉnh là các quan hệ $R = \{R_1, R_2, ..., R_n\}$. Khi đó nếu S là một tập hợp con của R, S sẽ có một đồ thị con của G là $S_G = (V', E')$ trong đó $V' = \{v_1, ..., v_{|S|}\}$ là các đỉnh tương ứng với các quan hệ trong S. Cạnh E' của S_G là tập hợp $\{(v_1, v_2) \mid v_1, v_2 \in V', (v_1, v_2) \in E\}$. *$S$ sẽ được định nghĩa là một tập hợp con liên thông nếu đồ thị con S_G liên thông*.

Số lượng các tập hợp con/đồ thị con liên thông của một truy vấn bao gồm n quan hệ được ký hiệu là ***#csg(n)***. Nói một cách khác, việc kết hợp các đỉnh dựa trên đồ thị truy vấn sẽ tạo nên đồ thị con liên thông S và một trong những nhiệm vụ của thuật toán DPccp là phải xác định số lượng đồ thị con *#csg(n)* phát sinh trong quá trình tìm kiếm kế hoạch tối ưu.

- **Cặp đồ thị con liên thông bù trừ**

Cho S_1 và S_2 là tập con liên thông khác rỗng của R tạo nên các đồ thị con liên thông S_{1G} và S_{2G}. Cặp (S_1, S_2) được định nghĩa là một cặp đồ thị con liên thông bù trừ (csg-cmp-pair) nếu thõa mãn điều kiện:

- $S_1 \cap S_2 = \emptyset$ *(3.1)*
- $\exists v_1 \in S_{1G}$ và $\exists v_2 \in S_{2G}$ mà $(v_1, v_2) \in E$. *(3.2)*

Điều kiện đầu tiên nhấn mạnh rằng hai tập hợp S_1 và S_2 là rời nhau. Điều kiện thứ hai có nghĩa là kết nối giữa đỉnh $v_1 \in S_1$ và đỉnh $v_2 \in S_2$ phải là một cạnh của đồ thị truy vấn. Trong chữ viết tắt cặp đồ thị con liên thông bù trừ (csg-cmp-pair), csg là đồ thị con liên thông, cmp là đồ thị bù trừ. Cặp đồ thị con liên thông bù trừ được ký hiệu là ***ccp***. Số lượng của tập hợp *ccp* được ký hiệu là ***#ccp***. Lưu ý rằng nếu (S_1, S_2) là cặp csg-cmp thì (S_2, S_1) cũng là cặp csg-cmp, do đó *#ccp* là số lượng đã bao gồm cả các cặp đối xứng. Giá trị *#ccp* phụ thuộc vào đồ thị truy vấn và được dùng làm giá trị chặn dưới về số lượng của các phép kết hợp cần được xem xét trong quá trình liệt kê.

3.3.2 Công thức tính #csg and #ccp

Công thức tính số lượng các đồ thị con liên thông *#csg(n)* và số lượng cặp đồ thị con liên thông bù trừ *#ccp(n)*, được trình bày trong Bảng 3.1 [18]. Nhận xét rằng đối với truy vấn dạng sao và chùm, *#ccp(n)* là hàm số mũ *n*, trong khi đó truy vấn chuỗi và vòng, *#ccp(n)* là một đa thức bậc 3.

Ví dụ muốn liệt kê các cặp-csg-cmp của một truy vấn có 10 quan hệ thì *#ccp* của truy vấn chuỗi là 330, trong khi *#ccp* cho một truy vấn chùm là 57002. Nếu áp dụng thuật toán DP (Hình 3.1) số lượng các kế hoạch được liệt kê sẽ luôn bằng số lượng *#ccp* của truy vấn chùm bất chấp cấu trúc truy vấn. Do đó, mặc dù khi áp dụng thuật toán DP với một truy vấn chuỗi, thuật toán vẫn sẽ liệt kê 56.672 (57002-330) kế hoạch dư thừa không cần thiết khi xây dựng kế hoạch tối ưu. Thuật toán DPccp liệt kê số lượng kế hoạch cần xem xét ít hơn thuật toán DP vì đã loại bỏ các kế hoạch không cần thiết do tích chéo gây ra.

Bảng 3.1 Công thức tính #csg và #ccp

Cấu trúc đồ thị kết hợp	*#csg(n)*	*#ccp(n)*
Chuỗi	$\frac{n(n+1)}{2}$	$\frac{n^3-n}{3}$
Vòng	n^2-n+1	n^3-2n^2+n
Sao	$2^{n-1}+n-1$	$(n-1)2^{n-2}$
Chùm	2^n-1	$3^n-2^{n+1}+1$

3.3.3 Mô tả thuật toán

Ý tưởng chính của thuật toán là xem xét các cặp-csg-cmp của đồ thị, liệt kê hiệu quả tất cả các cặp (S_1,S_2) mỗi cặp chỉ một lần duy nhất. Tất cả các tập con không rỗng của S_1 và S_2 sẽ được tạo ra trước, mỗi khi một cặp (S_1,S_2) được tạo ra. Thuật toán lặp lại quá trình liệt kê tất cả các cặp-csg-cmp (S_1,S_2) kết hợp các cặp (S_1,S_2) và xem xét kế hoạch tốt nhất kết nối với nó. Thuật toán được thể hiện trong Hình 3.14.

Lưu ý rằng *access-plans, prune-plans, join-plans* và *finalize-plans* được thực thi giống như khi sử dụng với thuật toán DP Hình 3.1. Trong trường hợp muốn thuật toán này được áp dụng trong môi trường phân tán, chúng ta thực hiện các thay đổi theo các phương pháp đề xuất trong mục 3.1.2.

Chương trình thuật toán DPccp Hình 3.14 có giai đoạn đầu (dòng 1-4) và giai đoạn cuối (dòng 16-18) giống như thuật toán DP. Sự khác biệt chính xảy ra trong giai đoạn thứ hai (dòng 5-15). Thủ tục *Enumerate-CSG* dùng để liệt kê các tập con liên thông và ứng với mỗi tập con S_1 thủ tục *Enumerate-CMP* dùng để phát sinh ra tất cả các đồ thị con bù S_2 của S_1.

Để đảm bảo chỉ liệt kê cặp-csg-cmp duy nhất, thuật toán thăm dò tất cả khả năng kết hợp các kế hoạch của các đồ thị con theo hai trật tự đảo nhau (dòng 10 và 11).

❖ **Chương trình thuật toán DPccp** [18]

Đầu vào: *Đồ thị kết hợp G chứa các đỉnh V={v_1,v_2,v_3,...,v_n} và các cạnh E.*

Kết quả: *Kế hoạch thực thi truy vấn tối ưu (không xem xét tích chéo)*

Thuật toán DPCCP(*G*)

```
1  for i = 1 to n do {
2          optPlan({vi}) = accessPlans({vi})
3          prunePlans(opt-plan({vi}))
4  }
5  Enumerate-CSG(G,A)
6  for all S1 ∈ A do{
7          Enumerate-CMP(G,S1,ℬ)
8          for all S2 ∈ ℬ do{
9                  S = S1 ∪ S2
10                 optPlan(S) = optPlan(S) ∪ joinPlans(optPlan(S1),optPlan(S2))
11                 optPlan(S) = optPlan(S) ∪ joinPlans(optPlan(S2),optPlan(S1))
12                 prunePlans(optPlan({S}))
13         }
14         ℬ = ∅
15 }
16 finalizePlans(optPlan(V))
17 prunePlans(optPlan(V))
18 return optPlan(V)
```

Hình 3.14 Chương trình thuật toán DPccp

3.3.4 Thuật toán liệt kê các tập con liên thông Enumerate-CSG

Cho một đồ thị kết hợp *G = (V, E)*. Với đỉnh $v \in V$, tập hợp đỉnh kề $\mathcal{N}$ của *v* được định nghĩa như sau:

$$\mathcal{N}(v) = \{v' | (v, v') \in E\}$$

Với một tập con liên thông $S \subseteq V$ có đồ thị con liên thông $S_G = (V', E')$, tập hợp đỉnh kề của *S* được định nghĩa như sau.

$$\mathcal{N}(S) = (\cup_{v \in V'} \mathcal{N}(v)) - V'$$

Xét một tập con liên thông S và tập đỉnh kề $N \subseteq \mathcal{N}(S)$. Khi đó, $S \cup N$ cũng là một tập con liên thông. Như vậy tập con liên thông có thể được mở rộng khi thêm vào bất kì tập con của tập đỉnh kề của nó. Phương pháp để tạo ra tất cả các tập con liên thông được thực hiện như sau:

Đầu tiên thuật toán xác định mỗi đỉnh riêng biệt $\{v_i\}$ của đồ thị con liên thông S. Tiếp theo thuật toán liệt kê tất cả tập con đỉnh kề $S' \subseteq \mathcal{N}(\{v_i\})$ và đệ quy tìm kiếm tập hợp con liên thông $S' \cup \{v_i\}$ tương đương với $S' \cup S$.

Vấn đề chính của phương pháp này là vấn đề trùng lặp. Để khắc phục vấn đề này mỗi đỉnh trong đồ thị kết hợp, được đánh số sử dụng duyệt đồ thị theo chiều rộng, bắt đầu từ đỉnh bất kỳ. Cho các đỉnh của đồ thị kết hợp G là tập hợp $V = \{v_1, v_2, ..., v_n\}$, trong đó đỉnh v_i là đỉnh thứ i gặp được trong quá trình duyệt theo chiều rộng bắt đầu từ v_1. Để tránh trùng lặp, chỉ có tập con S thu được bằng cách đệ quy tìm kiếm tập đỉnh kề của $\{v_i\}$ được xem xét, như vậy tất cả các đỉnh trong S sẽ có chỉ số $> i$. Điều kiện cần thiết dùng để hạn chế sự trùng lặp $\mathcal{B}_i$ được ký hiệu sau đây:

$$\mathcal{B}_i = \{v_j \mid j \leq i\}$$

❖ **Thủ tục Enumerate-CSG** [18]:

Đầu vào:

- *Đồ thị kết hợp G chứa các đỉnh $V=\{v_1,v_2,...,v_n\}$ và các đỉnh E.*
- *A chứa đồ thị con liên thông của G.*

***Điều kiện**: Các đỉnh trong G đã được sắp xếp duyệt theo chiều rộng.*

***Kết quả**: Tất cả tập hợp con của V tạo ra đồ thị con liên thông của G*

***Thủ tục ENUMERATE-CSG**(G,A)*

```
1  for i = n downto 1 do{
2          A = A ∪ {{vi}}
3          ENUMERATE-CSG-REG(G,{vi},Bi,A)
4  }
```

❖ **Thủ tục Enumerate-CSG-REC** [18]

Đầu vào:

- *Đồ thị kết hợp liên thông G chứa các đỉnh V={v_1,v_2,...,v_n} và các đỉnh E.*
- *S là tập hợp các đỉnh của đồ thị con hiện hành*
- *X là tập hợp các đỉnh cấm S mở rộng, dùng để chống trùng lắp*
- *A chứa đồ thị con liên thông của G.*

***Điều kiện**: Các đỉnh trong G đã được sắp xếp duyệt theo chiều rộng.*

***Kết quả**: Đệ qui tất cả tập hợp con của V tạo ra đồ thị con liên thông của G*

***Thủ tục: ENUMERATE-CSG-REC**(G,S,X,A)*

```
1  N = 𝒩(S) − X
2  For all S'⊆ N, S' ≠ ∅, enumerate subsets first{
3          A=A ∪ {S ∪ S'}
4  }
5  For all S'⊆ N, S' ≠ ∅,  enumerate subsets first{
6          ENUMERATE-CSG-REC(G, S ∪ S', X ∪ N,A)
7  }
```

Hình 3.15 Thủ tục Enumerate-CSG và Enumerate-CSG-REC

3.3.5 Ví dụ minh họa Enumerate-CSG

Ví dụ trong Hình 3.16 trình bày các bước của thủ tục Enumerate-CSG dùng để minh họa khả năng tránh được trùng lặp các quan hệ trong quá trình liệt kê đồ thị con liên thông. Hình 3.16 chỉ hiển thị hai vòng lặp cuối cùng của Enumerate-CSG (khi n = 2 và n = 1) và liệt kê giá trị của các biến N và A trong các lần gọi thủ tục đệ quy Enumerate-CSG-REC.

Giai đoạn áp dụng với đỉnh {R_1}, thủ tục Enumerate-CSG-REC không có sự hạn chế khi thăm dò các đỉnh kề cận. Do lúc này điều kiện hạn chế sự trùng lặp $\mathcal{B}_i$ bằng {R_1}, tất cả các tập con liên thông có chứa các đỉnh R_1 đều được liệt kê. Tuy nhiên, ở giai đoạn áp dụng với đỉnh {R_2}, thủ tục Enumerate-CSG-REC bị ngăn chặn trùng lặp với điều kiện $\mathcal{B}_i$ bằng {R_1, R_2}. Điều kiện này ngăn cản bất kỳ thăm dò có

chứa đỉnh R_1. Kết quả là tất cả các tập con bao gồm cả R_2 đều được liệt kê, ngoại trừ tập con $\{R_1, R_2\}$ do đã được tạo ra trong giai đoạn thăm dò đệ quy từ đỉnh R_1.

$A \leftarrow A \cup \{R_1\}$
Enumerate-Csg-Rec(G,{R1},{R1},A)
    N←{R2}
    $A \leftarrow A \cup \{R_1, R_2\}$
    Enumerate-Csg-Rec(G,{R1,R2},{R1,R2},A)
        N←{R3}
        $A \leftarrow A \cup \{R_1, R_2, R_3\}$
        Enumerate-Csg-Rec(G,{R1,R2,R3},{R1,R2,R3},A)
            N←{R4}
            $A \leftarrow A \cup \{R_1, R_2, R_3, R_4\}$
            Enumerate-Csg-Rec(G,{R1,R2,R3,R4},{R1,R2,R3,R4},A)
                $N \leftarrow \emptyset$
A={{R1},{R1,R2},{R1,R2,R3},{R1,R2,R3,R4}}

$A \leftarrow A \cup \{R_2\}$
Enumerate-Csg-Rec(G,{R2},{R1,R2},A)
    N←{R3}
    $A \leftarrow A \cup \{R_2, R_3\}$
    Enumerate-Csg-Rec(G,{R2,R3},{R1,R2,R3},A)
        N←{R4}
        $A \leftarrow A \cup \{R_2, R_3, R_4\}$
        Enumerate-Csg-Rec(G,{R2,R3,R4},{R1,R2,R3,R4},A)
            $N \leftarrow \emptyset$
A={{R1},{R1,R2},{R1,R2,R3},{R1,R2,R3,R4},{R2},{R2,R3},{R2,R3,R4}}

Hình 3.16 Liệt kê các bước Enumerate-CSG

3.3.6 Thủ tục liệt kê các tập con bù Enumerate-CMP

Tạo ra các tập con liên thông là bước quan trọng đầu tiên nhưng thuật toán cần tạo ra toàn bộ các cặp csg-cmp. Thủ tục *Enumerate-CSG* tạo ra thành phần S_1 của mỗi cặp csg-cmp, sau đó với mỗi S_1, sẽ tạo ra tất cả thành phần bù S_2 của nó.

Khi tạo ra phần bù S_2 cũng cần phải đảm bảo không bị trùng lắp giống như quá trình tạo ra tập con liên thông. Có nghĩa là chúng ta có thể tái sử dụng các thủ tục đệ quy *Enumerate-CSG-REC* với các thông số khác để có được tất cả phần bù của một tập hợp liên thông. Áp dụng kĩ thuật tương tự khi liệt kê tất cả các tập con liên thông, thuật toán sử dụng duyệt đồ thị theo chiều rộng để liệt kê tất cả các đồ thị bù S_2. Để tránh tạo ra các cặp trùng lặp, phần bù S_2 chỉ được chứa đỉnh v_j mà $j > i$ trong đó i là chỉ số của đỉnh bất kỳ trong S_1. Cho một tập hợp liên thông S_1, điều kiện hạn chế các phần bù trùng lặp là $\mathcal{B}_{min(S1)}$ với $min(S_1) = min(\{i \mid v_i \in S_1\})$.

❖ **Thủ tục Enumerate-CMP** [18]

Đầu vào:

- *Đồ thị kết hợp liên thông G chứa các đỉnh $V=\{v_1,v_2,\ldots,v_n\}$ và các đỉnh E.*
- *S là đồ thị con liên thông của G*
- *A chứa đồ thị con liên thông của G.*

***Điều kiện**: Các đỉnh trong G đã được sắp xếp duyệt theo chiều rộng.*

***Kết quả**: Tất cả đồ thị con liên thông H mà (S,H) là cặp-csg-cmp*

***Thủ tục: ENUMERATE-CMP**(G,S,A)*

```
1  X = B_min(S) ∪ S
2  N = 𝒩(S) − X
3  For all v_i ∈ N by descending i {
4        A=A∪{{v_i}}
5        ENUMERATE-CSG-REC(G, {v_i}, X ∪ (B_i ∩ N), A)
6  }
```

Hình 3.17 Thủ tục Enumerate-CMP

Thuật toán quan tâm đến việc xác định những đỉnh kề N của S, các đỉnh kề này phải có chỉ số cao hơn đỉnh có chỉ số nhỏ nhất trong S. Thuật toán lặp lại trên các

đỉnh kề $v_i \in N$ đáp ứng tiêu chí này và ghi nhận kết quả đầu ra A của mỗi đỉnh là một tập hợp con bù. Thuật toán tiếp tục tìm kiếm đệ quy những đỉnh kề xung quanh các đỉnh v_i này bằng cách sử dụng đệ quy thủ tục *Enumerate-CSG-REC* với tập hợp giới hạn $X \cup (\mathcal{B}_i \cap N)$. Tập hợp giới hạn này dùng để ngăn chặn trùng lắp do tính giao hoán.

3.3.7 Ví dụ minh họa Enumerate-CMP

Ví dụ sử dụng thủ tục Enumerate-CMP để tạo ra tất cả các đồ thị con liên thông bù của đồ thị con liên thông $\{R_2, R_3\}$. Các bước của thuật toán trình bày trong Hình 3.18.

Lưu ý rằng chỉ có duy nhất một thành phần bù của tập hợp con liên thông được tạo ra là ($\{R_4\}$). Một thành phần bù khác của $\{R_2, R_3\}$ là $\{R_1\}$ không được liệt kê bởi vì nó sẽ được tạo ra khi thủ tục enumerate-cmp được cung cấp đầu vào $\{R_1\}$.

Enumerate-cmp(G,{R2,R3},A)

X ← {R1,R2} ∪ {R2,R3} = {R1,R2,R3}

N ← {R1,R4} − {R1,R2,R3} = {R4}

A ← A ∪ {{R4}}

Enumerate-csg-rec(G, {R4}, {R1,R2,R3,R4},A)

N=∅

A = {{R4}}

Hình 3.18 Ví dụ Enumerate-cmp

3.4 Thuật toán kết hợp IDP1ccp

Thuật toán IDP1ccp là một thuật toán cải tiến từ hai thuật toán IDP1 (mục 3.2) và DPccp (mục 3.3). Tên của thuật toán IDP1ccp được ghép từ tên của thuật toán IDP1 và DPccp với nhau. Điều này có nghĩa là các điểm mạnh của hai thuật toán IDP1 và DPccp sẽ được phối hợp tận dụng trong thuật toán IDP1ccp.

Như vậy, thuật toán IDP1ccp sẽ vừa có điểm mạnh của thuật toán IDP1 là sử dụng quy hoạch động kết hợp với phỏng đoán tham lam để làm giảm thời gian hoạt động và độ phức tạp không gian tìm kiếm kế hoạch thực thi tối ưu. Đồng thời thuật

toán cũng sẽ có được điểm mạnh của thuật toán DPccp là khả năng tối ưu hóa truy vấn sử dụng quy hoạch động bằng cách liệt kê trên số lượng tối thiểu của các tập con liên thông của đồ thị kết hợp.

3.4.1 Chương trình thuật toán IDP1ccp

❖ **Thuật toán IDP1ccp** [2]

***Đầu vào**: Đồ thị kết hợp G chứa các đỉnh V={ $v_1, v_2, v_3, \ldots, v_n$ } tương ứng với các quan hệ R={ $R_1, R_2, R_3, \ldots, R_n$ } và các cạnh E. Số khối tối đa k*

***Kết quả**: Kế hoạch thực thi truy vấn tối ưu (không xem xét tích chéo)*

Thuật toán IDP1ccp(*G,k*)

```
1  for i = 1 to n do {
2          optPlan({Ri}) = accessPlans({Ri})
3          prunePlans(opt-plan({Ri}))
4  }
5  J=∅
6  While |V| > 1 do{
7          b=Balanced-Parmeter(|V|,k)
8          Enumerate-CSG'(G, A, b-1)
9          for all S1 ∈ A do{
10                 Enumerate-CMP'(G,S1,B,b)
11                 for all S2 ∈ B do{
12                         S = S1 ∪ S2
13                         if |S|=b do{
14                                 J = J ∪ {S}
15                         }
16                         optPlan(S) = optPlan(S) ∪ joinPlans(optPlan(S1),optPlan(S2))
17                         optPlan(S) = optPlan(S) ∪ joinPlans(optPlan(S2),optPlan(S1))
18                         prunePlans(optPlan({S}))
19                 }
20                 B = ∅
21         }
```

```
22          find P,N with P ∈ optPlan(N), N ∈ J such that
                    eval(P)=min{eval(P') | P' ∈ optPlan(W), W ∈ J}
23          generate new symbol: T
24          optPlan({T})= {P}
25          H = Merge-Vertices(G,N,T)
26          A=∅
27          Enumerate-CSG(H,A)
28          for all O ⊂ A do{
29                  delete(optPlan(O))
30          }
31          J=∅
32  }
33  finalizePlans(optPlan(V))
34  prunePlans(optPlan(V))
35  return optPlan(R)
```

Hình 3.19 Chương trình thuật toán IDP1ccp

3.4.2 Mô tả thuật toán

Thay vì sử dụng các quan hệ $R = \{R_1, R_2, ..., R_n\}$ làm quan hệ đầu vào như thuật toán IDP1, thuật toán IDP1ccp sử dụng đầu vào là đồ thị kết hợp *G*. Giai đoạn đầu tiên (dòng 1-4) và giai đoạn cuối cùng (dòng 33- 35) đều giống với các thuật toán IDP1 và DPccp. Thuật toán IDP1ccp không cần lập tập *toDo* như trong IDP1 để lưu trữ các quan hệ chờ đợi tham gia vào kế hoạch thực thi. Thay vào đó, thuật toán sử dụng đồ thị *G* để lưu trữ các quan hệ trong *R* tương ứng 1-1 với các đỉnh *V* trong đồ thị truy vấn.

Trong giai đoạn thứ hai của thuật toán IDP1 kế hoạch tốt nhất cân bằng, thuật toán tạo ra tất cả kế hoạch 2-quan hệ, 3-quan hệ, ..., *b*-quan hệ, thoát ra khỏi vòng lặp DP và lựa chọn kế hoạch *b*-quan hệ *P* tốt nhất. Ở đây, tham số *b* thu được bằng cách áp dụng thủ tục tính tham số cân bằng (trong phần 3.2.3) với số lượng các quan hệ hiện hành và giá trị *k*. Thuật toán IDP1ccp giống với khái niệm này, thuật toán sử

dụng các thủ tục *Enumerate-CSG* và *Enumerate-CMP* để tạo ra các cặp-csg-cmp tối ưu dựa trên đồ thị truy vấn, thay vì liệt kê tất cả các tập con có thể.

Tuy nhiên, thuật toán không cần thiết phải tạo tất cả các cặp-csg-cmp như thuật toán DPccp. Thuật toán IDP1ccp chỉ quan tâm đến việc tạo ra cặp (S_1, S_2) sao cho $|S1| + |S2| = b$, tức là thuật toán chỉ muốn tạo ra kế hoạch giới hạn đến kế hoạch b-quan hệ. Do đó, các thủ tục Enumerate-CSG, Enumerate-CSG-REC và Enumerate-CMP cần được thay đổi để phù hợp với IDP1ccp. Các thủ tục đã sửa đổi là *Enumerate-CSG', Enumerate-CSG-REC'* và *Enumerate-CMP'* có thể được xem thêm trong Phụ lục III.

Cấu trúc *J* được sử dụng để lưu trữ tất cả các tập con liên thông đã được liệt kê có chứa *b* đỉnh. Sau khi thuật toán đã tạo ra tất cả các tập con liên thông có kích thước *b*, thuật toán thoát ra khỏi giai đoạn DP và lựa chọn kế hoạch *P* tốt nhất của các tập con liên thông trong *J*. Lúc này thuật toán xem *P* như là một kế hoạch tiếp cận (*accessPlan*) của quan hệ tạm thời *T*.

Bước tiếp theo (dòng 25 Hình 3.19) dùng để cập nhật các quan hệ còn lại sẵn sàng cho giai đoạn lặp lại thuật toán DP. Tuy nhiên lúc này thuật toán không dùng tập hợp các quan hệ *R* để cập nhật, mà dùng đồ thị *G* để cập nhật. Khi xử lý trên đồ thi, việc cập nhật các quan hệ trở thành việc sáp nhập tất cả các đỉnh tương ứng với các quan hệ liên quan đến *P* thành một đỉnh tạm thời *T*.

Quá trình sáp nhập này được thực hiện bởi thủ tục *Merge-Vertices* Hình 3.20 và tạo ra đồ thị con *H*. Khi thuật toán loại bỏ đồ thị con *H* thì tất cả các kế hoạch liên quan đến các quan hệ trong *H* cũng sẽ được xóa. Thuật toán sử dụng *Enumerate-CSG* để liệt kê tất cả các tập con liên thông của *H*. Đối với mỗi tập con liên thông $h \in H$, thuật toán xóa các mục *optPlan* tương ứng của *h*.

❖ **Thủ tục Merge-Vertices** [2]

Đầu vào:

- *Đồ thị kết hợp G chứa các đỉnh* $V=\{v_1,v_2,v_3,\ldots,v_n\}$, *tương ứng với các quan hệ* $R=\{ R_1,R_2,R_3,\ldots,R_n \}$ *và các cạnh E.*
- *N: tập hợp các đỉnh sẽ được thay thế bởi T.*

***Mục đích**: Thay thế N thành đỉnh T và làm mới cạnh khảo sát.*

***Kết quả**: Đồ thị đã loại bỏ các đỉnh và cạnh.*

***Thủ tục: Merge-Vertices**(G,N,T)*

```
1  V = V − N ∪ T
2  E'= {(a,b)|a ∈ N, b ∈ N, (a,b) ∈ E }
3  for all (a,b) ∈ {(c,d) |c ∈ 𝒩(N), d ∈ N, (c,d) ∈ E} do{
4          b = T
5  }
6  for all (a,b) ∈ {(c,d) |d ∈ 𝒩(N), c ∈ N, (c,d) ∈ E} do{
7          a = T
8  }
9  return (N,E')
```

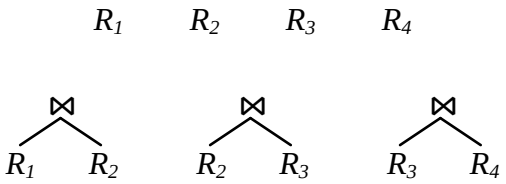
Hình 3.20 Thủ tục Merge-Vertices

3.4.3 Ví dụ minh họa IDP1ccp

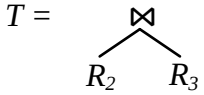
Xét ví dụ áp dụng thuật toán IDP1ccp với $k = 2$ cho đồ thị kết hợp chuỗi $G = (V, E)$ thể hiện trong Hình 3.2. Vì |V| = 4 lựa chọn kích thước khối xây dựng b cho giai đoạn đầu tiên DP sẽ là 2, ($b \leq \frac{|V|}{2} = 2$).

Giai đoạn đầu tiên của thuật toán gọi là giai đoạn DP (dòng 1-4 Hình 3.19). Thuật toán bắt đầu tìm các kế hoạch truy cập *accessPlans* cho các quan hệ R_1, R_2, R_3, R_4 tương đương với tạo các kế hoạch 1-quan hệ. Sau đó các *accessPlans* này sẽ được dùng để xây dựng tiếp các kế hoạch 2-quan hệ. Ví dụ các kế hoạch có thể được lưu trữ trong cấu trúc *optPlan* sau giai đoạn đầu tiên DP được thể hiện trong Hình 3.21.

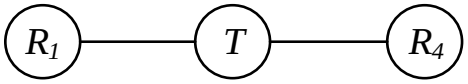
Giả sử rằng kế hoạch P được lựa chọn sau giai đoạn DP đầu tiên là kế hoạch liên quan đến các quan hệ như trong Hình 3.22. Đồ thị kết hợp lúc này được cập nhật bằng thủ tục *Merge-Vertices* kết hợp các đỉnh {R_2, R_3} thành một đỉnh mới tương ứng với quan hệ tạm thời T. Kết quả đồ thị kết hợp được cập nhật thể hiện trong Hình 3.23.



Hình 3.21 Các kế hoạch lưu trữ trong giai đoạn DP đầu tiên (b=2)

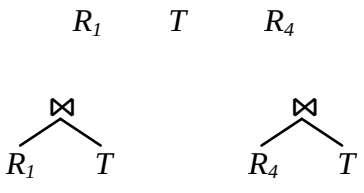


Hình 3.22 Kế hoạch được chọn sử dụng phỏng đoán tham lam

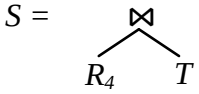


Hình 3.23 Đồ thị kết hợp sau giai đoạn sáp nhập

Sau đó thuật toán lặp lại giai đoạn DP thứ hai với các quan hệ R_1, T và R_4. Đến giai đoạn này, kích thước khối xây dựng b được tính lại vẫn là 2. Ví dụ các kế hoạch 2-quan hệ được lưu trữ trong cấu trúc *optPlans* trong giai đoạn DP thứ 2 được thể hiện trong Hình 3.24. Giả sử kế hoạch có giá trị *eval* thấp nhất là kế hoạch chứa quan hệ $\{R_4, T\}$ Hình 3.25.



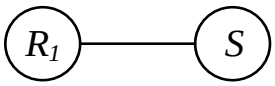
Hình 3.24 Các kế hoạch trong giai đoạn DP thứ hai



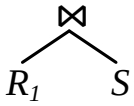
Hình 3.25 Kế hoạch được chọn sử dụng phỏng đoán tham lam

Thuật toán lại sử dụng *Merge-Vertices* để sáp nhập quan hệ $\{R_4,T\}$ thành quan hệ tạm thời tương ứng với đỉnh S. Đồ thị kết hợp giai đoạn sáp nhập thứ 2 được cập nhật được thể hiện trong Hình 3.26.

Thuật toán lặp lại giai đoạn thứ 3, lúc này số đỉnh của đồ thị kết hợp bây giờ 2, giá trị *b* là bằng *k*. Do đó, đây là giai đoạn tạo ra các kế hoạch cuối cùng. Kế hoạch cuối cùng xây dựng trên các quan hệ R_1 và quan hệ tạm thời *S* được thể hiện trong Hình 3.27.

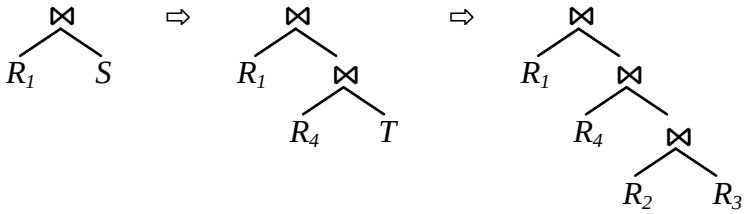


Hình 3.26 Đồ thị kết hợp sau giai đoạn sáp nhập thứ 2



Hình 3.27 Kế hoạch cuối cùng của các quan hệ tạm thời

Thay thế các quan hệ cơ sở vào kế hoạch cuối cùng, chúng ta có được kế hoạch tối ưu hóa cuối cùng của thuật toán IDP1ccp cho các quan hệ cơ sở ban đầu Hình 3.28.



Hình 3.28 Kế hoạch cuối cùng thu được từ IDP1ccp

CHƯƠNG 4: THỰC NGHIỆM VÀ ĐÁNH GIÁ

Trong phần trước đã trình bày nội dung và ý nghĩa các thuật toán tối ưu hóa truy vấn trong hệ phân tán như DP, IDP1, DPccp, IDP1ccp. Trong phần này trình bày quá trình thực nghiệm và so sánh kết quả thực hiện của các thuật toán trong các trường hợp khác nhau. Luận văn sẽ thực nghiệm trên các lớp của ngôn ngữ lập trình Java nhằm tạo ra một hệ quản trị cơ sở dữ liệu tổng quát. Phần này cũng trình bày các tập tin chứa đựng các thông số đầu vào như danh mục hệ thống, đồ thị kết hợp, kế hoạch thực thi truy vấn.

Thực nghiệm được thực thi trên máy tính có cấu hình như sau:

- CPU Intel Core i3 2,13GHz.
- Hệ điều hành Windows 8.1
- RAM 4GB.

4.1 Chuẩn bị các tập tin dữ liệu đầu vào

4.1.1 Cấu trúc tập tin danh mục

Ý nghĩa của danh mục hệ thống đối với hệ thống CSDL tập trung và hệ thống CDSL phân tán đã được trình bày trong mục 2.3.1. Trong phần này sẽ mô tả cấu trúc tập tin danh mục và cách phát sinh tập tin danh mục áp dụng vào mô hình thực nghiệm.

Trước khi thực hiện các bước thực nghiệm kiểm tra các thuật toán đã được trình bày, chúng ta cần phải xác định số quan hệ hiện có trong hệ thống, số lượng địa điểm cũng như số lượng quan hệ tại một địa điểm… Những thông tin này sẽ được lưu giữ trong tập tin cấu hình *catalog.conf*. Định danh của các địa điểm (địa chỉ IP) được lưu trong tập tin *system.conf*. Do đó, hai tập tin *system.conf* và *catalog.conf* cần phải tạo ra trước thi đưa vào chương trình thực nghiệm.

Do tập trung vào nghiên cứu thuật toán tối ưu hóa truy vấn để có thể áp dụng trên hệ thống cơ sở dữ liệu tổng quát, luận văn không xem xét thực hiện truy vấn trên một hệ quản trị cơ sở dữ liệu cụ thể. Tập tin danh mục hệ thống *catalog.conf* và tập tin danh sách các địa điểm chứa các quan hệ *system.conf* được phát sinh từ các lớp

được lập trình bằng ngôn ngữ java. Các thông số như tên của các quan hệ, lực lượng các quan hệ, các trường dữ liệu của một quan hệ … sẽ được tạo ra theo phương pháp lấy ngẫu nhiên. Cụ thể là để tạo tập tin cấu hình *catalog.conf* và tập tin *system.conf*, chương trình thực nghiệm sẽ gọi lớp *CatalogGenerator (sizeOfQuery, numberOfSites, relationsResideAtNumberOfSites, dirname).*

Lớp *CatalogGenerator* gồm có các thông số đầu vào như sau:

- *sizeOfQuery*: Số lượng các quan hệ có trong hệ thống
- *numberOfSites*: Số lượng các địa điểm
- *relationsResideAtNumberOfSites*: Số lượng quan hệ tại một địa điểm
- *dirname*: Đường dẫn lưu trữ tập tin catalog

Để thêm các thông tin về các địa điểm vào *catalog.conf*. Chương trình thực nghiệm sẽ sử dụng lớp *Catalog*. Lớp *Catalog (relationEntries, sitesInSystem, querySite, pageSize, bufferSizeInPages, ioTimeConstantForPage, relationKeys, networkTimeToSendAByte)* chứa một số tham số tiêu biểu như sau:

- *pageSize*: Kích thước của một trang đĩa
- *sitesInSystem*: Địa điểm thực hiện trong hệ thống
- *networkTimeToSendAByte*: Thời gian để truyền dữ liệu trên mạng
- *ioTimeConstantForPage*: Thời gian vào/ra đĩa.

Các thông tin về mối liên hệ giữa các quan hệ trong hệ thống được tạo bởi lớp *RelationEntry*. Lớp *RelationEntry* (*name, numberOfTuples, sizeOfTuplesInBytes, sites, schema*) chứa thông tin của các quan hệ như tên quan hệ, lực lượng quan hệ, kích thước bộ dữ liệu, các địa điểm có chứa quan hệ và các lược đồ (schema). Schema bao gồm một số trường cùng với *DomainType* tương ứng. Các *DomainType* được lấy ngẫu nhiên một trong bốn loại ký hiệu là A, B, C, D. Trong đó mỗi loại có kích thước khác nhau dùng để tạo ra kích thước cho các trường.

Tóm lại, đầu tiên chương trình tạo ra danh sách ngẫu nhiên tên của các quan hệ và tập ứng viên của các quan hệ cho danh mục. Tiếp theo danh mục được bổ sung các trường thuộc tính cho quan hệ. Mỗi quan hệ được gán từ 5 đến 10 trường. Mỗi trường được gán kích thước theo DomainType. Kích thước của mỗi domain được ghi

nhận tại Bảng 4.1. Cuối cùng danh mục được gán địa điểm cho các quan hệ. Số lượng địa điểm của một quan hệ cũng được tạo ra ngẫu nhiên.

Bảng 4.1 Domain Type

Domain Type	Bytes
A	2
B	10
C	15
D	8

Ví dụ một phiên bản của tập tin danh mục sử dụng trong hệ thống được trình bày trong Phụ lục I. Cấu trúc tập tin danh mục là một danh sách chứa các cặp dòng (*l1*, *l2*).

- Dòng thứ nhất *l1* chứa tên một quan hệ, kích thước quan hệ, kích thước của mỗi bộ theo byte và danh sách các địa điểm chứa quan hệ đó.
- Dòng thứ hai *l2* bao gồm một danh sách các cặp DomainType và tên của trường tương ứng của quan hệ thiết lập ở dòng *l1*.

Ví dụ:

Dòng 1: A1 40353 60 163.1.88.1 163.1.88.2

Dòng 2: D _A1.F1 B A1.F2 A A1.F3 C A1.F4 C A1.F5 B A1.F6

Được hiểu như sau:

Dòng 1	A1	40353	60	163.1.88.1 163.1.88.2
	Tên quan hệ	Số lượng bộ dữ liệu (numberOfTuples)	Kích thước quan hệ được lấy ngẫu nhiên	Địa điểm chứa quan hệ
Dòng 2	D	_A1.F1	B A1.F2 A A1.F3 C A1.F4 C A1.F5 .F6	
	Domain Type Ngẫu nhiên	Tên trường	Các cặp Domain Type-Tên trường còn lại	

Trong ví dụ này Kích thước quan hệ bằng tổng các Domain Type

D+B+A+C+C+B= 8 + 10 + 2 + 15 + 15 + 10= 60 byte

4.1.2 Phát sinh truy vấn

Sau khi đã có tập tin danh mục hệ thống, việc tiếp theo để thực hiện thí nghiệm chương trình cần phải có một tập hợp các truy vấn thể hiện các dạng cấu trúc kết hợp khác nhau. Thí nghiệm xem xét hệ thống phân tán gồm có *numberOfSites* vị trí và số lượng quan hệ lớn nhất trong một truy vấn là *sizeOfQuery*. Vì vậy, số lượng quan hệ trong danh mục *numberOfRelations* phải lớn hơn *sizeOfQuery*.

Tập tin truy vấn được tạo ra do lớp *QueryGenerator*. Lớp *QueryGenerator* sẽ phát sinh tập tin truy vấn ngẫu nhiên theo một cấu trúc đồ thị kết hợp. Trong lớp *QueryGenerator*, có các thủ tục *generateChain(),generateCycle(), generateStar(), generateClique()* dùng để phát sinh các truy vấn tương ứng với các kiểu truy vấn chuỗi, vòng, sao, chùm. Ví dụ một phiên bản của một tập tin truy vấn được trình bày trong Phụ lục II.

Điều kiện kết hợp của đồ thị sẽ được chọn ngẫu nhiên giữa một trường của một quan hệ với một trường của một quan hệ khác trong danh mục. Hai quan hệ R_1, R_2 kết hợp với nhau sẽ tạo thành một cạnh của đồ thị. Khi cấu trúc kết hợp đã được tạo ra, thông tin độ lựa chọn $\sigma(R_1, R_2)$ sẽ được ghi chú lên cạnh của đồ thị kết hợp.

Giả sử như điều kiện kết hợp giữa hai quan hệ R_1 và R_2 được lấy ngẫu nhiên là các trường $R_1.F_i$ và $R_2.F_j$, Gọi *dom(Fi)* là giá trị *DomainType* của thuộc tính F_i trên tổng số lượng bộ dữ liệu *numberOfTuples* của quan hệ. Thông tin ghi lên cạnh (R_1,R_2) sẽ là độ lựa chọn được tính toán như sau:

$$\sigma(R_1, R_2) = \frac{1}{\max\left(dom(F_i), dom(F_j)\right)} \qquad (4.1)$$

Truy vấn sau khi được phát sinh sẽ được lưu trữ vào tập tin truy vấn. Dòng đầu tiên của tập tin truy vấn chứa một chuỗi các kí tự tách biệt tương ứng với tên của các quan hệ có trong câu truy vấn. Sau khi dòng đầu tiên, kể từ dòng thứ hai trở đi mỗi dòng tiếp theo sẽ lần lượt lưu trữ một đỉnh (một quan hệ trong truy vấn), danh sách các đỉnh kề của nó, điều kiện kết hợp và độ lựa chọn của cạnh giữa hai đỉnh.

Ví dụ tập tin truy vấn dạng chùm (truy vấn có dạng từ một đỉnh có thể kết nối đến các đỉnh còn lại) của 3 quan hệ có nội dung như sau:

A1 B2 C3

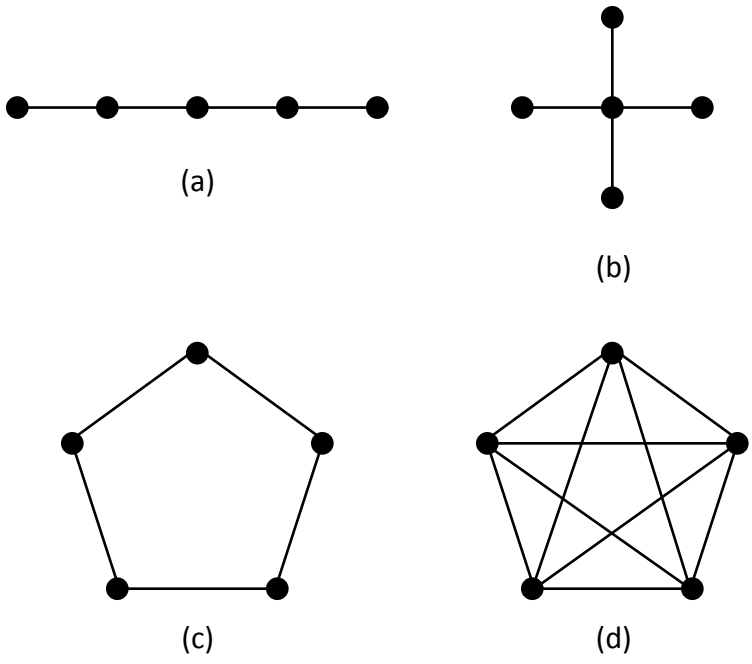
A1 B2 A1.F4=_B2.F1 1.3923 C3 _A1.F1=C3.F2 1.3927

B2 A1 _B2.F1=A1.F4 1.3923 C3 _B2.F1=C3.F5 1.3923

C3 B2 C3.F5=_B2.F1 1.3923 A1 C3.F2=_A1.F1 1.3927

Được hiểu là

<table>
<tr><td rowspan="2">Dòng 1</td><td colspan="3">A1 B2 C3</td></tr>
<tr><td colspan="3">Các quan hệ trong truy vấn</td></tr>
<tr><td rowspan="2">Dòng 2</td><td>A1</td><td>B2 A1.F4=_B2.F1 1.3923</td><td>C3 _A1.F1=C3.F2 1.3927</td></tr>
<tr><td>Tên đỉnh</td><td>Đỉnh kề; Điều kiện kết hợp; Độ lựa chọn</td><td>Đỉnh kề tiếp theo</td></tr>
<tr><td>Dòng 3 trở đi</td><td colspan="3">B2 A1 _B2.F1=A1.F4 1.3923 C3 _B2.F1=C3.F5 1.3923
C3 B2 C3.F5=_B2.F1 1.3923 A1 C3.F2=_A1.F1 1.3927
(tương tự dòng 2)</td></tr>
</table>



Hình 4.1 Các dạng truy vấn (a) chuỗi, (b) sao, (c) vòng, (d) chùm

4.1.3 Đồ thị kết hợp

Sau khi có tập tin truy vấn, đồ thị kết hợp dựa vào các thông tin trong tập tin truy vấn để xây dựng đồ thị mô tả dạng đồ thị của câu truy vấn. Đồ thị kết hợp gồm một lớp JoinGraph <String r1, Map <String r2, JoinInfo >>. Các khóa String r1 và r2 tương ứng với tên của các quan hệ và giá trị JoinInfo bao gồm độ lựa chọn và điều kiện kết hợp giữa các quan hệ r1 và r2. Dựa vào giá trị này của hai quan hệ, chúng ta có thể kiểm tra được thông tin kết hợp và sự tồn tại của một cạnh trong đồ thị kết hợp. Thông tin JoinInfo được lưu trữ hai lần, một lần cho cạnh (v1, v2) và một lần cho cạnh (v2, v1). Tuy nhiên, lưu ý rằng thông tin này chỉ dùng để thuận tiện cho việc kiểm tra đồ thị và hai giá trị này vẫn chỉ được tính là thông tin của một cạnh, không phải lưu trữ thông tin một cạnh thành hai lần.

4.1.4 Kế hoạch thực thi truy vấn

Chương trình sử dụng cấu trúc cây để thể hiện cho một kế hoạch Plan. Các lớp dùng để hiện thực kế hoạch thực thi truy vấn gồm có hai lớp chính là lớp Plan và PlanNode.

Lớp PlanNode cung cấp các hàm chức năng tính toán kết quả của kế hoạch như số lượng của các trang dữ liệu của kết quả (getNumberOfOutputPages()), tổng dung lượng kết quả theo byte (getTotalOutputBytes()). Các nút có thể có mặt trong một kế hoạch thực hiện truy vấn bao gồm JoinNode, RelationNode, SendNode, ReceiveNode và DelimiterNode. Mục đích của DelimiterNode là dùng để cô lập các nhiệm vụ có cùng lịch trình của một kế hoạch.

Lớp Plan xây dựng kế hoạch từ nút gốc PlanNode và cung cấp hàm *getCost()* để xác định chi phí của kế hoạch. Khi hàm *getCost()* được gọi trên một đối tượng Plan thì một đối tượng TaskTree được tạo ra. Chi phí của mỗi nhiệm vụ được đánh giá và lưu trữ trong trường thời gian *duration* của các nhiệm vụ. Sau đó, kết quả của TaskTree được chuyển sang cho lớp Scheduler, để thực hiện duyệt cây theo thứ tự duyệt sau và tạo ra lịch trình cuối cùng.

Trường chính của lớp Scheduler là trường schedule có cấu trúc là Map<Site, List<ScheduleEntry>>. Cấu trúc này lưu trữ mỗi địa điểm Site gắn với một danh sách các mục lịch trình ScheduleEntry. Một mục lịch trình ScheduleEntry bao gồm một nhiệm vụ và khoảng thời gian biểu thị khi nhiệm vụ được bắt đầu và kết thúc.

4.2 Các giai đoạn thực nghiệm

- **Giai đoạn 1**: Giai đoạn này dùng để thiết lập các thông số đầu vào, tạo thư mục chứa các tập tin đầu vào và kết quả.

  Thông số đầu vào gồm có các thông tin như:

  - Số lượng các quan hệ: *_numberOfRelations*
  - Số lượng các địa điểm: *_numberOfSites*
  - Kích thước của tham số *k*: *_sizeOfK*
  - Số quan hệ tại từng địa điểm: *_relationsResideAtNumberOfSites*
  - Số lượng truy vấn của một loại đồ thị truy vấn: *_numberOfQueries*

  Trong một thư mục tạo ra sẽ chứa các tập tin sau

  - Tập tin danh mục: *Catalog.conf*
  - Tập tin địa điểm: *SystemSites.conf*
  - Tập tin truy vấn: *Query.txt* (thay đổi tùy theo loại truy vấn Chain, Clique, Cycle, Star)
  - Tập tin kết quả: *Result.txt* ghi nhận thời gian thực hiện truy vấn (thay đổi tùy theo thuật toán áp dụng) .

- **Giai đoạn 2**: Phát sinh tập tin danh bạ

  Sau khi đã có tập tin địa điểm *Catalog.conf* và *SystemSites.conf,* ở bước này thí nghiệm sẽ tiến hành phân tích hai tập tin cấu hình, ánh xạ địa điểm và các thông số hệ thống như số lượng bộ dữ liệu, khóa kết nối giữa các quan hệ … vào các biến dùng để tính toán chi phí cho các kế hoạch như: *_relationEntrie, _sitesInSystem, _querySite, _pageSize, _bufferSizeInPages, _networkTimeToSendAByte,_relationKeys*

- **Giai đoạn 3**: Phát sinh truy vấn

  Để giả lập các loại truy vấn cần tối ưu, giai đoạn này thí nghiệm gọi thủ tục *QueryGenerator(sizeOfQuery,filename)* thực hiện tạo các truy vấn theo các dạng khác nhau như chuỗi, vòng, sao, chùm.

- **Giai đoạn 4**: Phát sinh đồ thị truy vấn

  Mỗi truy vấn sẽ được mô tả cấu trúc bằng một đồ thị truy vấn. Giai đoạn này ứng với từng loại tập tin truy vấn đã tạo ra ở giai đoạn 3, thí nghiệm gọi thủ tục *JoinGraph(queryFile)* để tạo các đồ thị truy vấn tương ứng. Các đồ thị này chủ yếu được dùng cho thuật toán DPccp và IDP1ccp.

- **Giai đoạn 5**: Thực hiện thuật toán tối ưu và ghi nhận kết quả

  Đến giai đoạn này, thí nghiệm đã có đầy đủ thông tin cần thiết về địa điểm, số lượng quan hệ, các truy vấn, đồ thị truy vấn... Thí nghiệm sẽ giữ nguyên các thông số đầu vào này và lần lượt áp dụng trên các thuật toán tối ưu khác nhau. Chi tiết các bước thực hiện của các thuật toán tối ưu sẽ dựa trên các chương trình thuật toán đã trình bày ở chương 3. Các bước cơ bản gồm có:

  - Chạy giai đoạn một: liệt kê các kế hoạch 1-quan hệ
  - Chạy giai đoạn hai: liệt kê các kế hoạch 2-quan hệ, … *k*-quan hệ
  - Tính toán chi phí để chọn kế hoạch tốt nhất, cắt bỏ các kế hoạch dư thừa cho lần lặp lại tiếp theo.
  - Ghi nhận thời gian thực hiện và kế hoạch tối ưu cuối cùng.

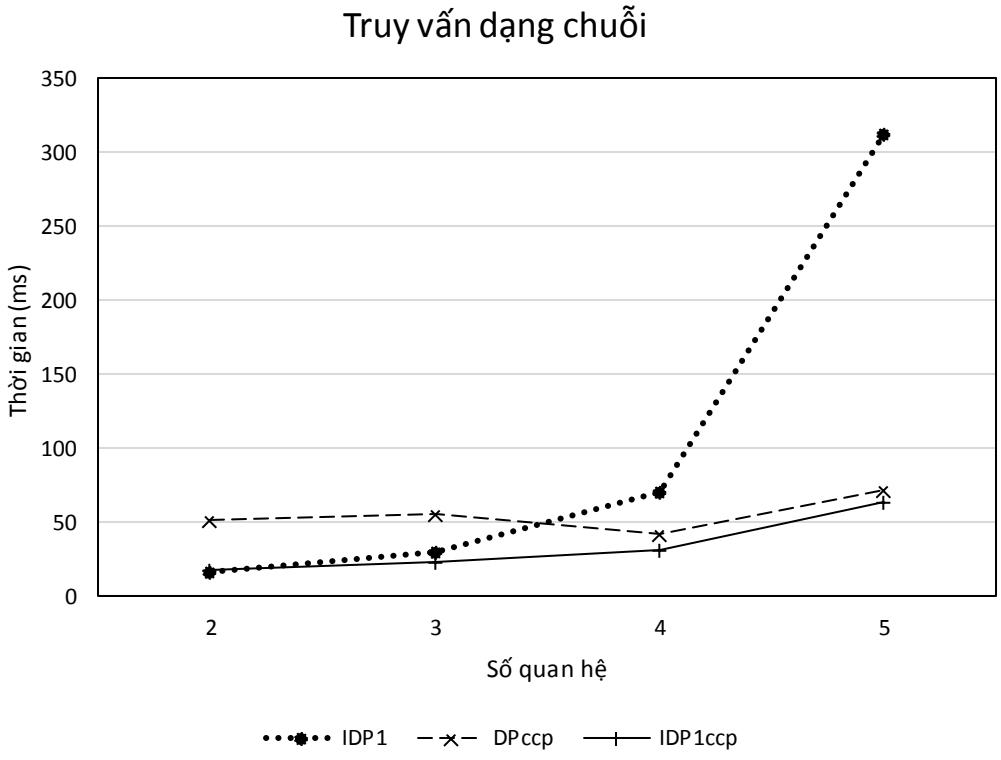
4.3 Kết quả thực nghiệm

Chương trình thực nghiệm với số lượng các quan hệ lần lượt thay đổi từ 2 cho đến 10 quan hệ, số lượng địa điểm được cố định là 3 địa điểm, kích thước của tham số *k* được thay đổi tối đa theo số quan hệ. Các dạng truy vấn lần lượt được thử nghiệm là dạng chuỗi, dạng vòng, dạng sao, dạng chùm. Ở đây tính khách quan của bộ dữ liệu thực nghiệm được đảm bảo vì bộ dữ liệu được sử dụng từ danh sách ngẫu nhiên tập ứng viên của các quan hệ trong tập tin danh mục. Đồng thời số lượng địa điểm của một quan hệ hiện diện cũng được tạo ra ngẫu nhiên.

Kết quả được ghi nhận như sau:

Bảng 4.2 Kết quả truy vấn dạng chuỗi

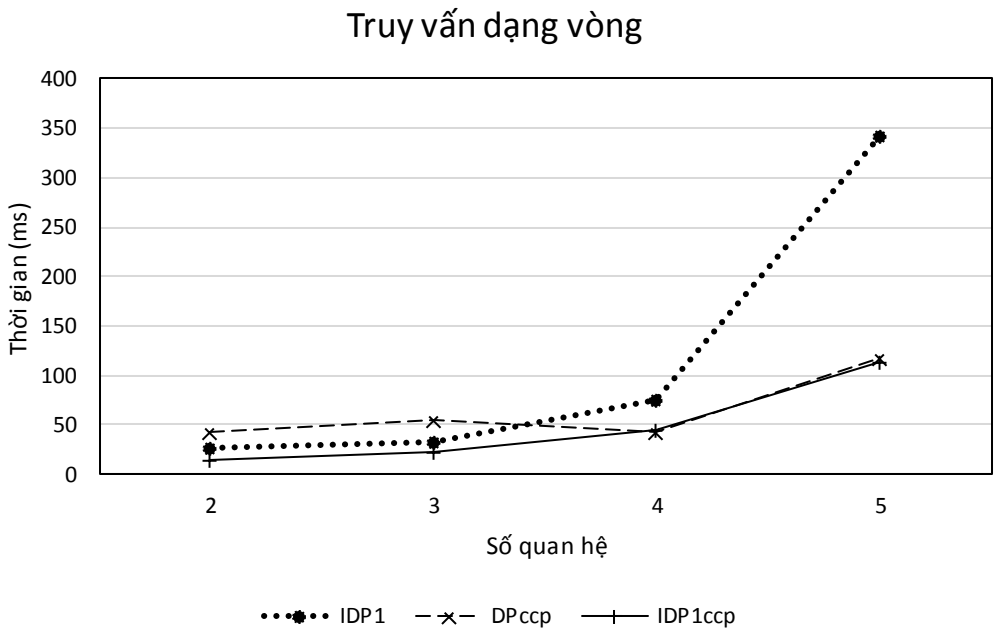
Số địa điểm	Số quan hệ	Tham số k	Thời gian thực hiện thuật toán (ms)		
			IDP1	DPccp	IDP1ccp
3	2	2	17	52	18
3	3	3	30	56	23
3	4	4	70	42	31
3	5	5	313	72	64
3	6	6	1219	141	119
3	7	7	3870	225	160
3	8	8	14228	279	273
3	9	9	49508	519	465
3	10	10	176642	758	1022



Hình 4.2 Biểu đồ kết quả truy vấn dạng chuỗi

Bảng 4.3 Kết quả truy vấn dạng vòng

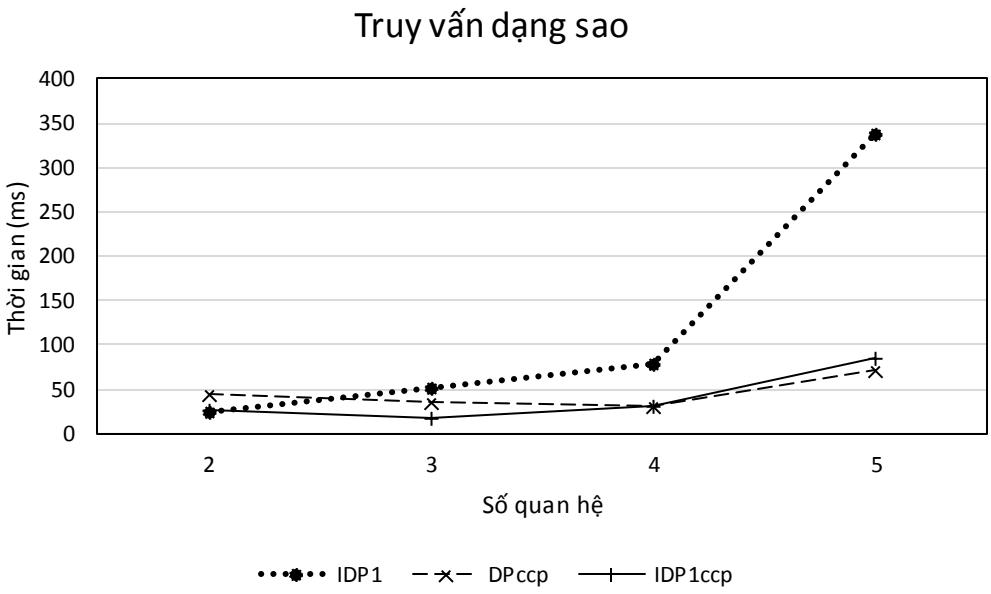
Số địa điểm	Số quan hệ	Tham số k	Thời gian thực hiện thuật toán (ms)		
			IDP1	DPccp	IDP1ccp
3	2	2	26	42	14
3	3	3	33	55	22
3	4	4	75	42	45
3	5	5	343	118	113
3	6	6	1323	237	256
3	7	7	4527	518	424
3	8	8	17019	740	801
3	9	9	71742	1579	2003
3	10	10	259571	3046	2385



Hình 4.3 Biểu đồ kết quả truy vấn dạng vòng

Bảng 4.4 Kết quả truy vấn dạng sao

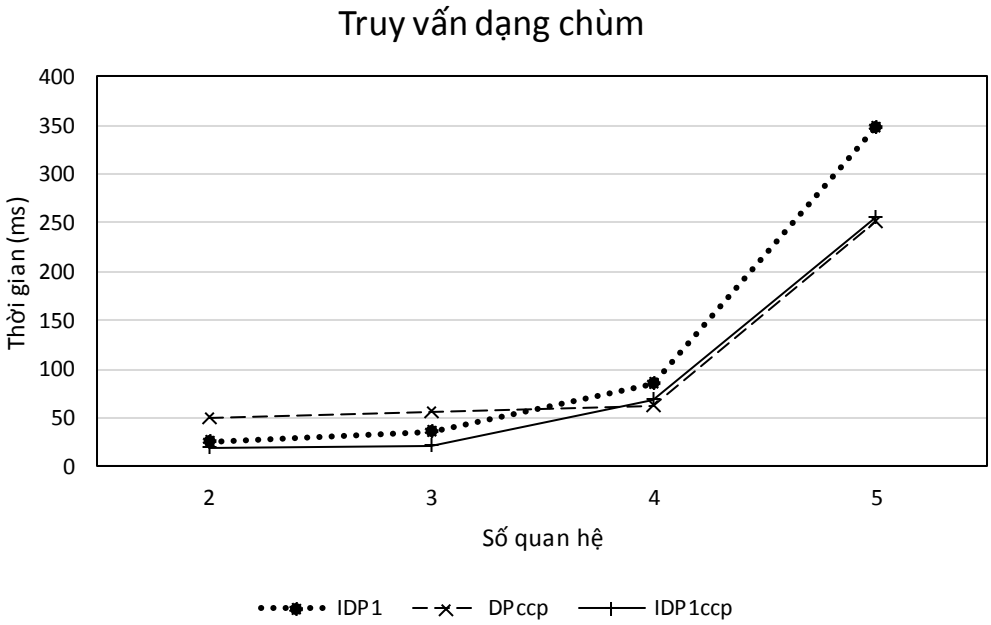
Số địa điểm	Số quan hệ	Tham số k	Thời gian thực hiện thuật toán (ms)		
			IDP1	DPccp	IDP1ccp
3	2	2	24	44	26
3	3	3	52	36	17
3	4	4	78	31	31
3	5	5	338	72	85
3	6	6	1397	189	200
3	7	7	4389	592	470
3	8	8	14858	1184	1292
3	9	9	65124	3407	3495
3	10	10	226849	8698	7314



Hình 4.4 Biểu đồ kết quả truy vấn dạng sao

Bảng 4.5 Kết quả truy vấn dạng chùm

Số địa điểm	Số quan hệ	Tham số k	Thời gian thực hiện thuật toán (ms)		
			IDP1	DPccp	IDP1ccp
3	2	2	26	50	18
3	3	3	36	55	20
3	4	4	85	63	68
3	5	5	348	251	255
3	6	6	1819	1347	824
3	7	7	4448	3127	3167
3	8	8	18150	12976	12076
3	9	9	62352	43375	43935
3	10	10	218641	152554	157066



Hình 4.5 Biểu đồ kết quả truy vấn dạng chùm

Thực nghiệm có thể mở rộng để kiểm chứng với các giá trị lớn hơn vì số lượng địa điểm, số quan hệ và số bộ dữ liệu trên một quan hệ là các biến có thể thay đổi giá trị theo yêu cầu khảo sát.

4.4 Nhận xét và đánh giá kết quả

Qua phân tích số liệu và sơ đồ minh họa chúng ta nhận thấy thuật toán IDP1 đều cho thời gian thực thi xấu nhất trong hầu hết các dạng đồ thị truy vấn khác nhau. Trong giai đoạn đầu khi số lượng quan hệ trong truy vấn còn ít, hai thuật toán DPccp và thuật toán IDP1ccp cho thấy kết quả gần như tương tự nhau, mặc dù vậy kết quả của thuật toán IDP1ccp vẫn có phần nhỉnh hơn một chút. Càng về cuối khi số lượng quan hệ trong truy vấn tăng lên thì thuật toán cải tiến IDP1ccp chứng tỏ sự vượt trội so với kết quả của hai thuật toán IDP1 và DPccp.

Trong các dạng đồ thị truy vấn thì đồ thị truy vấn dạng chuỗi và dạng vòng có cấu trúc gần giống nhau nên thời gian hồi đáp gần như là như nhau. Trong khi đó đồ thị truy vấn dạng sao và dạng chùm có độ phức tạp cao hơn nên thời gian hồi đáp truy vấn cũng dài hơn. Do cấu trúc phức tạp nên truy vấn dạng chùm có thời gian xử lý lâu nhất, trong khi đó do cấu trúc đơn giản nên truy vấn dạng chuỗi có thời gian xử lý nhanh nhất.

Như vậy so với thuật toán DP truyền thống, thuật toán IDP1 mặc dù đã có cải tiến trong việc áp dụng thuật toán phỏng đoán tham lam để thu nhỏ không gian tìm kiếm kế hoạch thực thi vẫn cho kết quả hồi đáp chưa tốt. Thuật toán DPccp với cải tiến dùng kĩ thuật đồ thị để hạn chế lựa chọn các tổ hợp các cặp quan hệ không liên thông và không tuân theo đồ thị truy vấn đã cho thời gian thực thi tốt hơn. Nhưng thuật toán DPccp vẫn còn điểm yếu do phải xem xét quá nhiều đồ thị con không cần thiết. Thuật toán IDP1ccp đã có sự kết hợp hai ưu điểm của hai thuật toán IDP1 và DPccp, kết hợp thuật toán phỏng đoán tham lam và kĩ thuật đồ thị, từ đó tối ưu và cải thiện được được thời gian xử lý. Nhưng chính vì do IDP1ccp bắt nguồn từ thuật toán IDP1 nên thuật toán vẫn còn một số hạn chế tương tự như IDP1. Thuật toán chỉ tốt nhất khi áp dụng cho các truy vấn có số lượng quan hệ nhỏ và vẫn có thời gian chạy xấu nhất khi tối ưu hóa truy vấn dạng chùm. Tuy nhiên, nhìn về tổng thể thuật toán IDP1ccp vẫn có thời gian hoạt động thực tế tốt hơn so với IDP1 và DPccp. Thuật toán IDP1ccp vẫn được xem là một giải pháp tốt trong việc tối ưu các truy vấn trong hệ tập trung lẫn hệ phân tán.

CHƯƠNG 5: KẾT LUẬN

5.1 Những kết quả đạt được

Sau quá trình tìm hiểu và nghiên cứu, luận văn đã đạt được một số điểm sau:

- Góp phần tổng hợp lý thuyết và phân tích quá trình xử lý tối ưu hóa truy vấn trong hệ tập trung cũng như hệ phân tán để đưa ra một bức tranh tổng quát về quá trình tối ưu hóa truy vấn. Trong quá trình phân tích, những điểm mạnh và điểm yếu của hệ quản trị cơ sở dữ liệu phân tán cũng đã được trình bày.
- Mô tả phương pháp xây dựng mô hình chi phí đánh giá truy vấn dựa trên nhiều tác vụ song song trong hệ phân tán khi thực hiện lựa chọn kế hoạch tối ưu.
- Trình bày được nội dung và ý nghĩa của các thuật toán tối ưu hóa trên hệ phân tán cải tiến từ thuật toán Dynamic Programming cổ điển như thuật toán IDP1, thuật toán DPccp.
- Kiểm chứng thuật toán kết hợp IDP1ccp đã tận dụng được điểm mạnh của các thuật toán IDP1 và thuật toán DPccp để mang lại hiệu quả tối ưu truy vấn tốt hơn trên hệ phân tán.
- Thực nghiệm minh họa so sánh thời gian thực thi của các thuật toán IDP1, DPccp và IDP1ccp trên các loại truy vấn khác nhau như truy vấn dạng chuỗi, truy vấn dạng sao, truy vấn dạng vòng, truy vấn dạng chùm, trong điều kiện phân tán trên nhiều địa điểm khác nhau.

Các thuật toán trình bày trong luận văn đều có khả năng xác định được kế hoạch thực hiện mang lại chi phí tốt nhất, cân bằng giữa thời gian tìm kiếm kế hoạch thực thi với thời gian tổng thời gian thực thi truy vấn để trả lại kết quả cho người dùng và ứng dụng. Đề tài sau khi nghiên cứu có thể ứng dụng để tối ưu hóa truy vấn, xử lý thứ tự thực hiện các quan hệ trong câu truy vấn, cho các tổ chức có cơ sở dữ liệu lớn được phân tán trên nhiều vị trí, tăng khả năng chia tải và khả năng kết hợp thực thi đồng thời giữa nhiều máy chủ quản lý cơ sở dữ liệu.

5.2 Các hạn chế và hướng nghiên cứu tiếp theo

Luận văn chỉ xem xét trường hợp các quan hệ được lưu trữ và nhân bản hoàn toàn tại các địa điểm, có nghĩa là các quan hệ không bị phân mảnh. Nhưng có rất nhiều mô hình cơ sở dữ liệu phân tán vẫn thường được sử dụng trên các quan hệ phân mảnh. Vì vậy, một hướng nghiên cứu tiếp theo là áp dụng các thuật toán tối ưu đã trình bày trên các quan hệ phân mảnh.

Phép kết hợp sử dụng trong thuật toán được giới hạn thực hiện dựa trên phép kết bằng giữa các quan hệ. Các phép kết khác áp dụng trên hệ phân tán nhằm làm giảm chi phí sử dụng mạng như Semi joins và Bloom joins chưa được xem xét. Đây cũng có thể là một hướng nghiên cứu tiếp theo.

Trong quá trình xây dựng mô hình chi phí để giảm bớt sự phức tạp của mô hình, luận văn đã giả định rằng mỗi địa điểm trong hệ thống chỉ có một bộ xử lý và chỉ thực hiện một hoạt động duy nhất tại một thời điểm. Hướng nghiên cứu xây dựng mô hình chi phí phức tạp trên hệ thống có nhiều bộ xử lý và tại một địa điểm có nhiều phép toán xảy ra đồng thời cũng là một bài toán cần giải quyết khi mở rộng đề tài.

Ngoài ra, hướng nghiên cứu đồ thị truy vấn dạng pha trộn, tức là đồ thị kết hợp theo từng cặp hoặc theo tất cả các loại đồ thị truy vấn kiểu chuỗi, vòng, sao, chùm cũng được để lại cho các phần nghiên cứu trong tương lai.

TÀI LIỆU THAM KHẢO

[1]. TS.Nguyễn Đình Thuân (2013), Bài giảng Distributed Database, Chương 4,5,7, Trường Đại học Công nghệ Thông tin ĐHQG-HCM.

[2]. Robert Taylor (2010), Query Optimization for Distributed Database Systems, Master Thesis, University of Oxford.

[3]. G. Ramakrishnan (2003), Database Management Systems, Third Edition, McGrawHill.

[4]. D. Kossmann (2000), The state of the art in distributed query processing, pages 422–469, ACM Computing Surveys.

[5]. M. Steinbrunn, G. Moerkotte, and A. Kemper (1997). Heuristic and randomized optimization for the join ordering problem, pages 191–208, VLDB Journal.

[6]. Y. E. Ioannidis (1996), Query optimization, pages 103–114, ACM Computing Surveys.

[7]. L. Angrave (2009), Completing Physical Query Plan Finale, from: <http://www.cs.uiuc.edu/class/fa06/cs411/slides/>.

[8]. Graham - Knuth – Patashnik (1990), Concrete Mathematics, Addison Wesley Publishing Company.

[9]. Benjamin Nevarez (2010), Optimizing Join Orders, from: <http://www.benjaminnevarez.com/2010/06/optimizing-join-orders/>.

[10]. D. Kossmann and K. Stocker (2000), Iterative dynamic programming: A new class of query optimization algorithms. ACM Transactions on Database Systems.

[11]. A. Swami (1989), Optimization of large join queries: Combining heuristics andcombinatorial techniques, pages 367–376, ACM SIGMOD Conference on Management of Data, Portland.

[12]. K. W. R. James F. Kurose (2009), Computer Networking: A Top-Down Approach Featuring the Internet. Addison Wesley.

[13]. C.S.Mullins (2003), Tuning DB2 SQL access paths. IBM, from:< http://www.ibm.com/developerworks/data/library/techarticle/0301mullins/0301mullins.html>.

[14]. K. Ono and G. M. Lohman (1990), Measuring the complexity of join enumeration in query optimization. IRM Almaden Research Center.

[15]. B. Vance and D. Maier (1996), Rapid bushy join-order optimization with cartesian products, pages 35–46, Proc. of the ACM SIGMOD Conf. on Management of Data.

[16]. S. Ganguly, W. Hasan, and R. Krishnamurthy (1992), Query optimization for parallel execution, pages 9–18, SIGMOD Conference.

[17]. P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, I. A. Lorie, and T. G. Price (1979), Access path selection in a relational database management system. ACM SIGMOD International Conference on Management of Data.

[18]. G. M. Thomas Neumann (2006), Analysis of two existing and one new dynamic programming algorithm for the generation of optimal bushy join trees without cross products, pages 930–941, VLDB Endowment.

[19]. Garima Mahajan (July 2012), Query Optimization in DDBS Vol. 1, No. 1, International Journal of Computer Applications & Information Technology, from:< www.ijcait.com>.

[20]. Ridhi Kapoor, Dr. R. S.Virk (June-2013), Selectivity & Cost Estimates in Query Optimization, Department of Computer Science & Engineering, Guru Nanak Dev University Amritsar, Punjab, India.

[21]. Farnoush Banaei-Kashani (April 2008), "Distributed Databases", excerpt from "Principles of Distributed Database Systems" by M. Tamer Özsu and Patrick Valduriez.

[22]. Magda Balazinska - CSE 444 (Spring 2013), Lecture 10 Query Optimization (part 1), from:< https://courses.cs.washington.edu/courses/cse444/13sp/>

[23]. Johann Gamper (2008), Optimization of Distributed Queries, Chapter 7, Faculty of Computer Science, Free University of Bozen – Bolzano.

[24]. E. W. Yannis E. Ioannidis (1987), Query optimization by simulated annealing, pages 9–22, ACM SIGMOD Conf Management of Data, San Francisco, Calif.

[25]. C. Z. Yingying Tao, Qiang Zhu and W. Lau, (2003), Optimizing large star schema queries with snowflakes via heuristic-based query rewriting. Proceedings of the 2003 conference of the Centre for Advanced Studies on Collaborative research, pages 279–293, Toronto, Canada. IBM Press.

[26]. T. Neumann (June 2009), Query simplification: Graceful degradation for join order optimization, pages 403–414, Association for Computing Machinery (ACM).

[27]. Y. C. K. Yannis E. Ioannidis (1990). Randomized algorithms for optimizing large join queries, pages 312–321, ACM SIGMOD Conf Management of Data, Atlantic City, NJ.

[28]. K. Bennett, M. C. Ferris, and Y. E. Ioannidis (1991). A genetic algorithm for database query optimization, pages 400–407, Morgan Kaufmann Publishers.

PHỤ LỤC

I. Tập tin danh mục

A1 40353 60 163.1.88.1 163.1.88.2

D _A1.F1 B A1.F2 A A1.F3 C A1.F4 C A1.F5 B A1.F6

B2 32570 83 163.1.88.0 163.1.88.2

D _B2.F1 B B2.F2 B B2.F3 B B2.F4 C B2.F5 B B2.F6 B B2.F7 B B2.F8

C3 58745 65 163.1.88.0 163.1.88.1 163.1.88.2

B _C3.F1 B C3.F2 B C3.F3 D C3.F4 C C3.F5 A C3.F6 B C3.F7

D4 38590 58 163.1.88.0

C _D4.F1 B D4.F2 C D4.F3 B D4.F4 D D4.F5

II. Tập tin truy vấn

A1 B2 C3 D4 E5

A1 B2 _A1.F1=B2.F6 1.742372E-5 C3 _A1.F1=C3.F3 1.742372E-5 D4

_A1.F1=D4.F5 1.742372E-5 E5 _A1.F1=E5.F2 1.742372E-5

B2 A1 B2.F6=_A1.F1 1.742372E-5

C3 A1 C3.F3=_A1.F1 1.742372E-5

D4 A1 D4.F5=_A1.F1 1.742372E-5

III. Thủ tục CSG', CMP'

❖ **Thủ tục ENUMERATE-CSG'** [2]

***Đầu vào**:*

- *Đồ thị kết hợp liên thông G chứa các đỉnh $V=\{v_1,v_2,...,v_n\}$ và các đỉnh E.*
- *A là đồ thị con liên thông của G, k là số khối tối đa.*

***Điều kiện**: Các đỉnh trong G đã được sắp xếp duyệt theo chiều rộng.*

***Kết quả**: Tập hợp con của V có đồ thị con liên thông của G mà $|V|\leq k$*

***Thủ tục: ENUMERATE-CSG'**(G,A,k)*

```
1  For i=n downto 1 do{
2        A=A∪{{vi}}
3        ENUMERATE-CSG-REC'(G, {vi},Bi,A,k)
4  }
```

❖ **Thủ tục ENUMERATE-CSG-REC'** [2]

Đầu vào:

- *Đồ thị kết hợp liên thông G chứa các đỉnh V={v1,v2,...,vn} và các đỉnh E.*
- *S là tập hợp các đỉnh của đồ thị con hiện hành*
- *X là tập hợp các đỉnh cấm S mở rộng, dùng để chống trùng lắp*
- *A là đồ thị con liên thông của G, k là số khối tối đa.*

***Điều kiện**: Các đỉnh trong G đã được sắp xếp duyệt theo chiều rộng.*

***Kết quả**: Đệ qui tất cả tập hợp con của V tạo nên đồ thị con liên thông của G mà $|V| \leq k$*

***Thủ tục: ENUMERATE-CSG-REC'**(G,S,X,A,k)*

1 $N = \mathcal{N}(S) - X$

2 ***For all*** $S' \subseteq N, S' \neq \emptyset, |S'| \leq min(k-|S|,|N|)$ *enumerate subsets first {*

3 $A = A \cup \{S \cup S'\}$

4 *}*

5 ***For all*** $S' \subseteq N, S' \neq \emptyset, |S'| \leq min(k-|S|,|N|)$ *enumerate subsets first {*

6 *ENUMERATE-CSG-REC'(G, S ∪ S', X ∪ N,A,k) }*

❖ **Thủ tục ENUMERATE-CMP'** [2]

Đầu vào:

- *Đồ thị kết hợp liên thông G chứa các đỉnh V={v1,v2,...,vn} và các đỉnh E.*
- *S là đồ thị con liên thông của G*
- *A dùng để lưu kết quả, k là số khối tối đa.*

***Điều kiện**: Các đỉnh trong G đã được sắp xếp duyệt theo chiều rộng.*

***Kết quả**: Tất cả đồ thị con liên thông H mà (S,H) là cặp-CSG-cmp và $|S| + |H| = k$*

***Thủ tục: ENUMERATE-CMP'**(G,S,A,k)*

1 $X = \mathcal{B}_{min(S)} \cup S$

2 $N = \mathcal{N}(S) - X$

3 ***For all*** $v_i \in N$ *by descending i {*

4 $A = A \cup \{\{v_i\}\}$

5 *ENUMERATE-CSG-REC'(G,* $\{v_i\}, X \cup (\mathcal{B}_i \cap N), A, k - |S|)$ *}*