VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

DATABASE SYSTEM (CO201B)

Assignment

"IN-MEMORY DATABASE SYSTEMS"

Instructor(s): Trương Tuấn Anh

Students: Nguyễn Xuân Huy Hoàng - 2311070

HO CHI MINH CITY, JANUARY 2026

Contents

List of Figures

List of Tables

1 GIỚI THIỆU

2 KIẾN TRÚC ỨNG DỤNG

3 SO SÁNH VÀ THỰC NGHIỆM

3.1 Data Storage & Management

3.2 Indexing

3.3 Query Processing

3.3.1 Tổng quan về Xử lý truy vấn

Xử lý truy vấn là một quy trình phức tạp bao gồm nhiều bước nhằm chuyển đổi một câu truy vấn cấp cao thành một chiến lược thực thi hiệu quả để truy xuất dữ liệu từ cơ sở dữ liệu.

Một câu truy vấn trong ngôn ngữ cấp cao như SQL mang tính *khai báo* (declarative), nghĩa là nó chỉ định *những gì* người dùng muốn lấy ra chứ không mô tả *cách thức* để truy xuất dữ liệu. Do đó, DBMS cần xác định một chiến lược thực thi tối ưu hoặc gần tối ưu nhất nhằm đảm bảo hiệu năng truy vấn.

Quy trình xử lý truy vấn

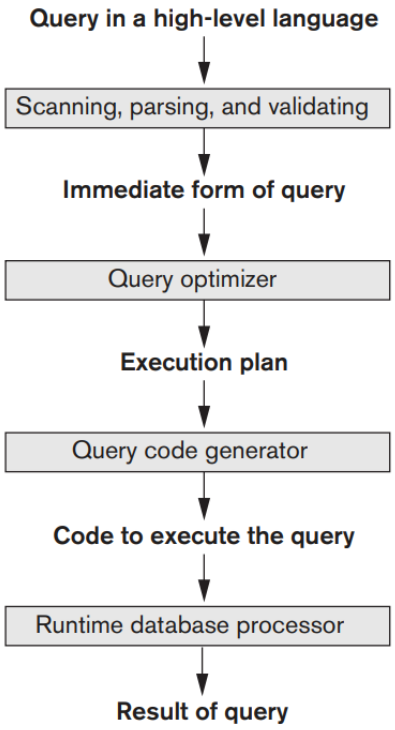

Figure 1: Quy trình xử lý truy vấn trong hầu hết các hệ quản trị cơ sở dữ liệu

Quy trình xử lý truy vấn trong DBMS thường bao gồm các giai đoạn sau:

1. **Quét, phân tích cú pháp và kiểm tra hợp lệ (Scanning, Parsing, and Validation)**: Giai đoạn này nhận diện các đơn vị từ vựng (tokens) như tên quan hệ, tên thuộc tính và các từ khóa SQL. Sau đó, hệ thống kiểm tra cú pháp nhằm đảm bảo truy vấn tuân thủ các quy tắc ngữ pháp của ngôn ngữ truy vấn, đồng thời kiểm tra tính hợp lệ về mặt ngữ nghĩa.
2. **Biểu diễn nội bộ (Internal Representation)**: Sau khi phân tích, truy vấn được chuyển đổi thành một cấu trúc dữ liệu nội bộ, phổ biến nhất là cây truy vấn (query

tree) hoặc đồ thị truy vấn (query graph). Cây truy vấn tương ứng với một biểu thức đại số quan hệ mở rộng. Trong thực tế, truy vấn SQL thường được phân rã thành các khối truy vấn (query blocks), là đơn vị cơ bản để dịch sang các toán tử đại số và thực hiện tối ưu hóa.

3. **Tối ưu hóa truy vấn (Query Optimization)**: Đây là giai đoạn quan trọng nhất trong xử lý truy vấn, nơi DBMS lựa chọn chiến lược thực thi phù hợp. Quá trình này được thực hiện bởi mô-đun *Query Optimizer*. Hai kỹ thuật chính được sử dụng bao gồm:
   - *Tối ưu hóa dựa trên quy tắc (Heuristic Optimization)*, sử dụng các quy tắc kinh nghiệm để biến đổi cây truy vấn ban đầu thành một cây tương đương nhưng hiệu quả hơn.
   - *Tối ưu hóa dựa trên chi phí (Cost-based Optimization)*, dựa trên việc ước lượng chi phí thực thi của các chiến lược khác nhau và lựa chọn chiến lược có chi phí thấp nhất.
4. **Sinh mã thực thi (Code Generation)**: Sau khi kế hoạch thực thi tối ưu được lựa chọn, bộ sinh mã truy vấn (Query Code Generator) sẽ tạo ra mã thực thi tương ứng. Mã này bao gồm các lời gọi đến các thuật toán vật lý cụ thể để thực hiện các phép toán cơ sở dữ liệu.
5. **Thực thi (Runtime Execution)**: Bộ xử lý cơ sở dữ liệu tại thời điểm chạy (Runtime Database Processor) sẽ thực thi mã được sinh ra, ở chế độ biên dịch hoặc thông dịch, để tạo ra kết quả truy vấn. Nếu xảy ra lỗi trong quá trình thực thi, hệ thống sẽ phát sinh thông báo lỗi tương ứng.

Các thuật toán xử lý phép toán đại số quan hệ

Để xử lý truy vấn, các hệ quản trị cơ sở dữ liệu sử dụng các thuật toán vật lý tương ứng với từng phép toán trong đại số quan hệ.

1. **Sắp xếp ngoài (External Sorting)**: Sắp xếp là một thuật toán cơ bản, cần thiết đối với các truy vấn có mệnh đề `ORDER BY`, loại bỏ trùng lặp (`DISTINCT`), hoặc hỗ trợ các thuật toán kết như *sort-merge join*. Đối với các tập dữ liệu lớn không vừa trong bộ nhớ chính, DBMS thường sử dụng thuật toán *external sort-merge*.
2. **Phép chọn (Selection)**: Phép chọn là thao tác tìm kiếm các bản ghi thỏa mãn một điều kiện nhất định. Các phương pháp phổ biến bao gồm quét tuyến tính (linear scan), tìm kiếm nhị phân (binary search) khi dữ liệu đã được sắp xếp, hoặc sử dụng chỉ mục (index scan) như chỉ mục sơ cấp, chỉ mục băm hoặc chỉ mục thứ cấp (B^+-tree). Các phương pháp dựa trên chỉ mục đặc biệt hiệu quả đối với truy vấn đẳng thức và truy vấn phạm vi.
3. **Phép kết (Join)**: Phép kết là một trong những thao tác tốn chi phí nhất trong xử lý truy vấn. Các chiến lược phổ biến bao gồm:
   - *Nested-loop join*: Với mỗi bản ghi trong bảng ngoài, hệ thống quét toàn bộ bảng trong để tìm các cặp khớp.
   - *Index-based nested-loop join*: Sử dụng chỉ mục trên thuộc tính kết của bảng trong để giảm chi phí quét.
   - *Sort-merge join*: Hiệu quả khi cả hai bảng đã được sắp xếp theo thuộc tính kết.
   - *Partition hash join*: Phân hoạch hai bảng bằng cùng một hàm băm, sau đó thực hiện kết nối trên các phân hoạch tương ứng.

Tối ưu hóa truy vấn
Tối ưu hóa truy vấn chủ yếu dựa trên hai kỹ thuật chính:

1. **Tối ưu hóa dựa trên quy tắc (Heuristic Optimization)**: Phương pháp này sử dụng các quy tắc kinh nghiệm để biến đổi cây truy vấn ban đầu thành một cây tương đương nhưng hiệu quả hơn. Một số đặc điểm chính bao gồm:
   - Thực hiện các phép `SELECT` và `PROJECT` càng sớm càng tốt nhằm giảm kích thước dữ liệu trung gian.
   - Cây truy vấn (Query Tree) biểu diễn các phép toán, trong đó các nút lá là các quan hệ đầu vào và các nút trong là các phép toán như σ, π, $\bowtie$.
   - Áp dụng các quy tắc biến đổi để đẩy các phép lọc xuống thấp nhất có thể trong cây truy vấn.
2. **Tối ưu hóa dựa trên chi phí (Cost-based Optimization)**: Phương pháp này ước lượng chi phí thực thi của các chiến lược khác nhau và lựa chọn chiến lược có chi phí thấp nhất.
   - Thành phần chi phí bao gồm chi phí truy cập đĩa (I/O), chi phí tính toán CPU, chi phí lưu trữ và chi phí truyền dữ liệu (đối với hệ phân tán).
   - Bộ tối ưu hóa sử dụng thông tin thống kê từ *system catalog* như số lượng bản ghi, kích thước bản ghi, số lượng khối và hệ số phân khối.
   - Độ chọn lọc (selectivity) được sử dụng để ước lượng kích thước các kết quả trung gian, thường dựa trên histogram.

Do số lượng thứ tự thực hiện phép kết (join ordering) có thể rất lớn ($n!$), các bộ tối ưu hóa thường giới hạn không gian tìm kiếm bằng cách ưu tiên các cấu trúc *left-deep tree*. Trong cấu trúc này, toán hạng bên phải của mỗi phép kết luôn là một quan hệ cơ sở, thuận lợi cho việc thực thi theo đường ống (pipelining).

Đánh giá và thực thi truy vấn
Đánh giá truy vấn (Query Evaluation) là quá trình chuyển đổi một kế hoạch thực thi truy vấn thành các hoạt động cụ thể để truy xuất dữ liệu. Hai chiến lược chính được sử dụng bao gồm:

1. **Materialized Evaluation**: Mỗi phép toán trong cây truy vấn được thực thi tuần tự và kết quả trung gian được lưu trữ tạm thời trên đĩa trước khi chuyển sang phép toán tiếp theo. Cách tiếp cận này gây ra chi phí I/O đáng kể do phải ghi và đọc lại các kết quả trung gian.
2. **Pipelining Evaluation**: Các phép toán được kết nối thành một luồng xử lý liên tục. Ngay khi một bộ kết quả được tạo ra, nó được chuyển trực tiếp sang phép toán tiếp theo mà không cần lưu trữ trung gian. Cách tiếp cận này giúp giảm chi phí I/O và cho phép hệ thống trả về kết quả sớm hơn. Trong thực tế, các phép toán vật lý thường được cài đặt dưới dạng các *iterator*.

3.3.2 Xử lý truy vấn trong PostgreSQL

PostgreSQL là một hệ quản trị cơ sở dữ liệu quan hệ tuân thủ chuẩn SQL, sử dụng pipeline xử lý truy vấn nhiều giai đoạn kết hợp giữa phân tích cú pháp, tối ưu hóa dựa trên chi phí và cơ chế thực thi hiệu quả. Quá trình xử lý truy vấn trong PostgreSQL bao gồm các bước chính sau:

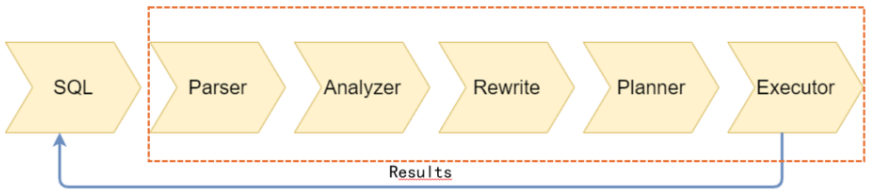

Figure 2: Quy trình xử lý truy vấn trong PostgreSQL

1. **Parsing**: Truy vấn SQL đầu vào được phân tích cú pháp nhằm kiểm tra tính hợp lệ về mặt ngữ pháp và cấu trúc. PostgreSQL chuyển truy vấn thành cây cú pháp (parse tree), phản ánh cấu trúc logic của câu lệnh SQL, bao gồm các mệnh đề `SELECT`, `FROM`, `WHERE`, `JOIN` và `GROUP BY`.

2. **Analysis và Query Rewrite**: Ở giai đoạn này, PostgreSQL thực hiện phân tích ngữ nghĩa bằng cách xác minh sự tồn tại của các đối tượng cơ sở dữ liệu như bảng, cột, view và kiểm tra kiểu dữ liệu cũng như quyền truy cập. Đồng thời, bộ rewrite áp dụng các luật chuyển đổi truy vấn, bao gồm mở rộng view, biến đổi các biểu thức con như `IN`, `EXISTS` và áp dụng chính sách bảo mật ở mức hàng (Row Level Security).

3. **Query Optimization (Planner)**: Bộ tối ưu hóa của PostgreSQL sử dụng mô hình tối ưu hóa dựa trên chi phí (Cost-Based Optimizer). Dựa trên các thống kê như số lượng bản ghi, phân bố dữ liệu và độ chọn lọc của chỉ mục, hệ thống sinh ra nhiều kế hoạch thực thi khả thi. Mỗi kế hoạch được ước lượng chi phí I/O và CPU, từ đó lựa chọn kế hoạch có chi phí thấp nhất, quyết định thứ tự join, thuật toán join (Nested Loop, Hash Join, Merge Join) và phương pháp truy xuất dữ liệu.

4. **Execution Engine**: Kế hoạch thực thi được chuyển cho bộ máy thực thi, nơi các toán tử được xử lý theo mô hình iterator (Volcano model). Các phép toán như quét dữ liệu, lọc, nối, sắp xếp và tổng hợp được thực hiện theo đơn vị bộ (tuple-at-a-time). PostgreSQL cũng hỗ trợ thực thi song song để tận dụng tài nguyên đa lõi.

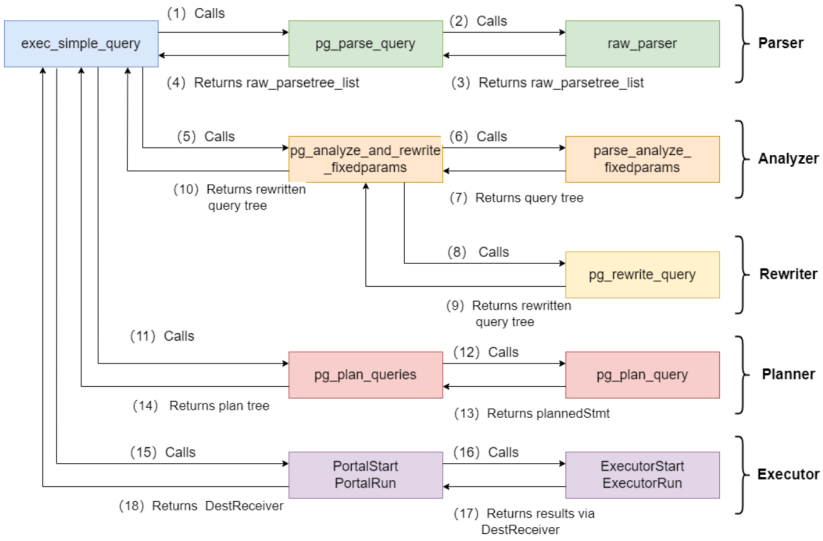

Figure 3: Vòng đời xử lý truy vấn trong PostgreSQL

3.3.3 Xử lý truy vấn trong MongoDB

Quy trình xử lý truy vấn trong MongoDB được thiết kế nhằm đảm bảo hệ thống luôn lựa chọn kế hoạch thực thi hiệu quả nhất, ngay cả khi khối lượng dữ liệu hoặc phân bố dữ liệu thay đổi theo thời gian.

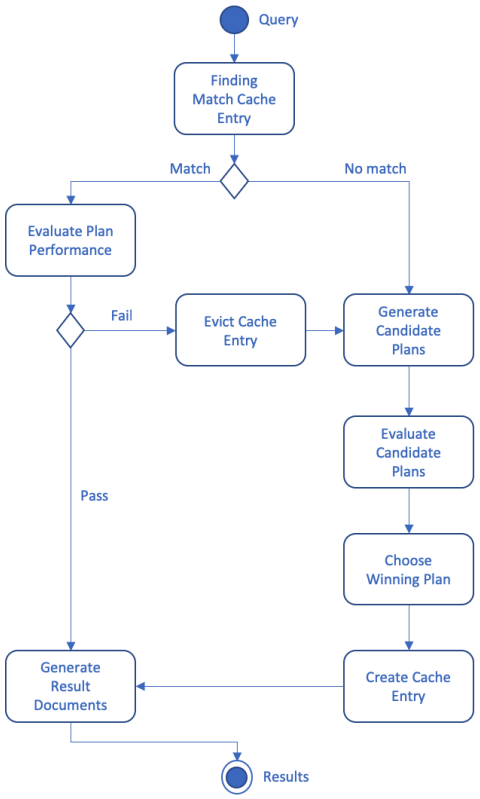

Figure 4: Quy trình xử lý truy vấn trong MongoDB

1. **Phân tích Truy vấn (Query Parsing)**: Mọi quy trình đều bắt đầu từ một **Query**. Truy vấn MongoDB được biểu diễn dưới dạng BSON và có thể bao gồm các toán tử như `$match`, `$and`, `$or`, `$gt`. Hệ thống tiến hành phân tích cú pháp truy vấn nhằm

xác định tập collection mục tiêu, các điều kiện lọc dữ liệu và các yêu cầu truy xuất, xử lý dữ liệu liên quan.

2. **Kiểm tra Bộ nhớ đệm Kế hoạch (Finding Match Cache Entry)**

   Trước khi sinh ra kế hoạch thực thi mới, MongoDB kiểm tra *plan cache* để xác định xem một truy vấn có cùng hình dạng (query shape) đã từng được xử lý hay chưa.

   - **Match (Có sẵn):**
     - MongoDB sử dụng kế hoạch trong cache và tiến hành *Evaluate Plan Performance* để đánh giá hiệu năng thực tế.
     - **Pass:** Nếu kế hoạch vẫn đạt hiệu năng tốt, hệ thống tiếp tục sử dụng kế hoạch này và chuyển sang bước sinh kết quả.
     - **Fail:** Nếu hiệu năng suy giảm (ví dụ do phân bố dữ liệu thay đổi), kế hoạch sẽ bị *Evict Cache Entry* và hệ thống tiến hành lập kế hoạch lại.
   - **No Match (Chưa có):** MongoDB chuyển trực tiếp sang giai đoạn lập kế hoạch mới.

3. **Lập Kế hoạch và Tối ưu hóa Khi Thực thi (Query Planning & Runtime Optimization)**

   Khi không tồn tại kế hoạch phù hợp trong cache, MongoDB thực hiện:

   - **Generate Candidate Plans:** Sinh ra nhiều kế hoạch thực thi ứng viên dựa trên các chỉ mục (indexes) hiện có, bao gồm index scan, collection scan hoặc các chiến lược kết hợp.
   - **Evaluate Candidate Plans:** Các kế hoạch ứng viên được chạy thử trong một khoảng thời gian ngắn (trial execution) nhằm đo lường các chỉ số thực tế như số lượng tài liệu được quét và thời gian xử lý.

4. **Lựa chọn và Lưu Kế hoạch (Plan Selection & Caching)**

   Sau quá trình đánh giá, MongoDB thực hiện:

   - **Choose Winning Plan:** Kế hoạch có hiệu năng tốt nhất được chọn làm kế hoạch chiến thắng.
   - **Create Cache Entry:** Kế hoạch chiến thắng được lưu vào plan cache để tái sử dụng cho các truy vấn tương tự trong tương lai.

5. **Thực thi và Trả Kết quả (Execution Engine & Cursor)**

   MongoDB thực thi truy vấn dựa trên kế hoạch đã được lựa chọn:

   - **Generate Result Documents:** Truy vấn được xử lý theo mô hình *document-at-a-time*, trong đó mỗi tài liệu BSON được xử lý như một đơn vị độc lập.
   - **Results:** Kết quả được trả về cho client thông qua *cursor*, cho phép truy xuất dữ liệu từng phần, giúp tối ưu bộ nhớ cho cả phía server và client.

Đối với các truy vấn nâng cao, MongoDB hỗ trợ:

- **Aggregation Pipeline:** Dữ liệu được xử lý qua chuỗi các stage như lọc, nhóm, sắp xếp và tính toán.
- **Distributed Query (Scatter–Gather):** Trong môi trường sharded cluster, truy vấn được phân tán đến nhiều node (scatter) và kết quả được tổng hợp lại (gather) trước khi trả về cho người dùng.

3.4 Transaction

3.5 Concurrency

4 DEMO ỨNG DỤNG

5 KẾT LUẬN

6 REFERENCES