BAN CƠ YẾU CHÍNH PHỦ

HỌC VIỆN KỸ THUẬT MẬT MÃ

BÁO CÁO TỔNG KẾT

ĐỀ TÀI NGHIÊN CỨU KHOA HỌC SINH VIÊN

ĐỀ TÀI

TĂNG TỐC TRUY VẤN CƠ SỞ DỮ LIỆU SỬ DỤNG GPU

Ngành: Công nghệ thông tin

Sinh viên thực hiện:

Nguyễn Thị Hồng Ngân - CT060229

Tô Quang Viễn - CT060146

Nguyễn Ngọc Tuyền - CT060145

Bùi Đức Khánh - CT060119

Giảng viên hướng dẫn:

ThS. Cao Thanh Vinh

Khoa Công Nghệ Thông Tin - Học viện Kỹ thuật Mật Mã

Hà Nội, 2025

BAN CƠ YẾU CHÍNH PHỦ

HỌC VIỆN KỸ THUẬT MẬT MÃ

BÁO CÁO TỔNG KẾT

ĐỀ TÀI NGHIÊN CỨU KHOA HỌC SINH VIÊN

ĐỀ TÀI

TĂNG TỐC TRUY VẤN CƠ SỞ DỮ LIỆU SỬ DỤNG GPU

Ngành: Công nghệ thông tin

Sinh viên thực hiện:

Nguyễn Thị Hồng Ngân - CT060229

Tô Quang Viễn - CT060146

Nguyễn Ngọc Tuyền - CT060145

Bùi Đức Khánh - CT060119

Giảng viên hướng dẫn:

ThS. Cao Thanh Vinh

Khoa Công Nghệ Thông Tin - Học viện Kỹ thuật Mật Mã

Hà Nội, 2025

CHƯƠNG 1. THỰC NGHIỆM

1.1. Cấu hình máy chủ thực nghiệm

1.1.1. Lựa chọn Cấu hình Máy chủ AWS EC2

Để đảm bảo việc đánh giá hiệu năng giữa HeavyDB (tối ưu cho GPU) và DuckDB (tối ưu cho CPU) được thực hiện một cách chính xác và phản ánh đúng tiềm năng của từng hệ thống trong các kịch bản thực tế, hai cấu hình máy chủ AWS EC2 riêng biệt đã được lựa chọn. Mỗi cấu hình được chọn lựa cẩn thận nhằm tối ưu hóa cho loại workload và hệ quản trị cơ sở dữ liệu tương ứng.

1.1.1.1. Cấu hình cho HeavyDB (Nền tảng GPU)

Instance `g5.2xlarge` được lựa chọn để triển khai HeavyDB, nhằm khai thác tối đa khả năng xử lý song song mạnh mẽ của kiến trúc GPU hiện đại:

- **Loại Instance:** `g5.2xlarge`
- **Bộ xử lý (CPU):** 8 vCPU (thường là AMD EPYC 7R32 trên thế hệ G5)
- **Bộ xử lý đồ họa (GPU):** 1 x NVIDIA A10G Tensor Core
- **Bộ nhớ GPU (VRAM):** 24 GiB GDDR6 - Dung lượng đủ lớn để chứa các bảng dữ liệu quan trọng của TPC-H/DS ở các Scale Factor mục tiêu (lên đến 50GB), giảm thiểu việc truy cập bộ nhớ hệ thống.
- **Bộ nhớ hệ thống (RAM):** 32 GiB - Đóng vai trò bộ đệm khi dữ liệu vượt quá VRAM.
- **Mục đích:** Cung cấp nền tảng GPU kiến trúc Ampere mạnh mẽ, phù hợp cho các tác vụ tính toán, phân tích dữ liệu phức tạp và xử lý song song quy mô lớn mà HeavyDB được thiết kế để tận dụng.

1.1.1.2. Cấu hình cho DuckDB (Nền tảng CPU)

Instance `c7a.8xlarge` được lựa chọn để triển khai DuckDB, đảm bảo hệ thống này được vận hành trên một nền tảng CPU hiệu năng cao, đại diện cho sức mạnh tính toán của các bộ xử lý hiện đại:

- **Loại Instance:** `c7a.8xlarge`

- **Bộ xử lý (CPU):** 32 vCPU (AMD EPYC 9R14 - Thế hệ 4 "Genoa") - Số lượng nhân lớn và kiến trúc mới nhất đảm bảo khả năng xử lý song song và hiệu suất đơn luồng cao.
- **Tần số Turbo tối đa:** Lên đến 3.7 GHz.
- **Bộ nhớ hệ thống (RAM):** 64 GiB - Dung lượng lớn, đủ để DuckDB (vốn là hệ thống in-memory hiệu quả) xử lý các bộ dữ liệu TPC-H/DS mục tiêu mà không bị giới hạn bởi bộ nhớ (ít nhất ở các SF nhỏ và vừa).
- **Mục đích:** Tạo ra một môi trường CPU mạnh mẽ nhất có thể để DuckDB, với các kỹ thuật tối ưu hóa vector hóa tiên tiến, thể hiện được tiềm năng tối đa của mình. Đây là cơ sở để có một phép so sánh công bằng và ý nghĩa về hiệu năng khi đối đầu với giải pháp GPU.

1.1.1.3. Lý do lựa chọn hai cấu hình riêng biệt và khác nhau

Việc lựa chọn hai cấu hình riêng biệt và không hoàn toàn tương đương về tài nguyên CPU/RAM là một quyết định có chủ đích và chiến lược, dựa trên phân tích, kinh nghiệm thực nghiệm ban đầu và mục tiêu nghiên cứu:

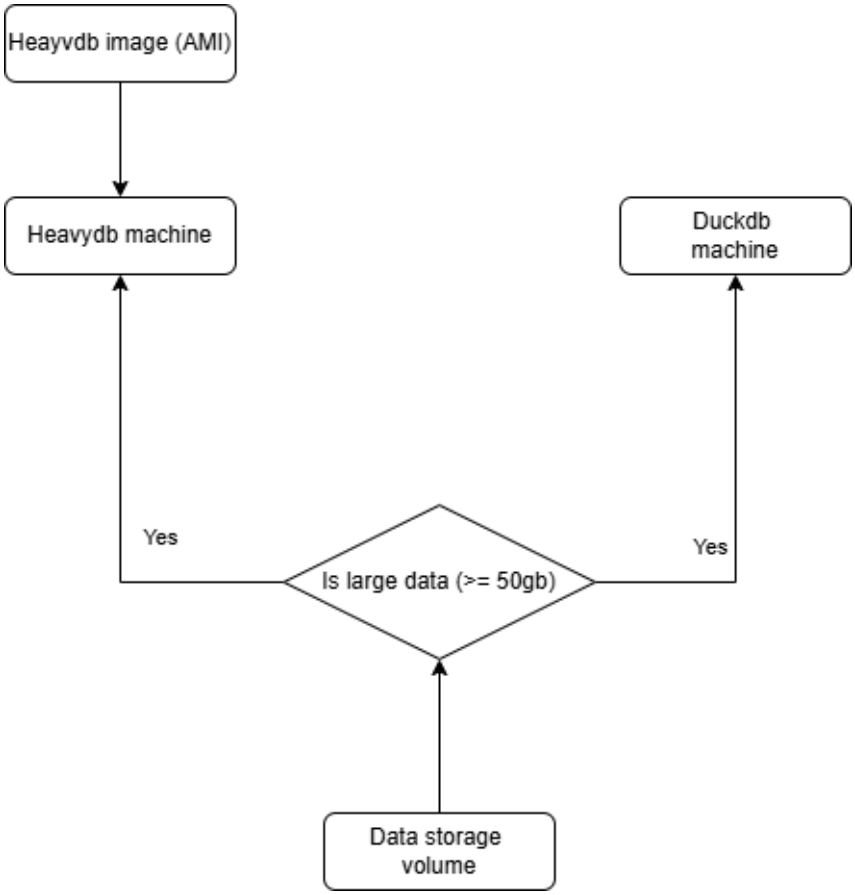
- **Đảm bảo Môi trường Tối ưu cho Từng Công nghệ:** Mục tiêu cốt lõi là so sánh hiệu quả của hai hướng tiếp cận công nghệ khác nhau (GPU-native vs CPU-optimized) khi mỗi hướng được phát huy tối đa. HeavyDB cần một GPU mạnh (A10G) để thể hiện giá trị, trong khi DuckDB cần một CPU đa nhân, hiện đại (EPYC Gen 4) để không bị giới hạn. Việc ép cả hai vào một cấu hình "tương đương" hoặc chạy trên CPU yếu của máy G5 sẽ làm sai lệch kết quả và không phản ánh đúng tiềm năng của ít nhất một trong hai hệ thống.
- **Phản ánh Bài toán Lựa chọn Thực tế:** Thay vì một so sánh hàn lâm trên phần cứng giả định, phương pháp này mô phỏng kịch bản thực tế nơi người dùng phải lựa chọn giữa việc đầu tư vào một hệ thống trang bị GPU chuyên dụng (như G5) hoặc một hệ thống CPU mạnh mẽ (như C7a) cho nhu cầu phân tích dữ liệu của họ.
- **Tạo ra Phép so sánh Cạnh tranh và Ý nghĩa:** Bằng cách đặt HeavyDB/G5 đối đầu với DuckDB/C7a, nghiên cứu đảm bảo rằng giải pháp GPU

đang được so sánh với một đối thủ CPU thực sự mạnh mẽ và được tối ưu hóa cao độ. Điều này làm cho bất kỳ lợi thế nào mà GPU thể hiện được trở nên thuyết phục hơn, và ngược lại, cũng cho thấy rõ những lĩnh vực mà CPU hiện đại vẫn chiếm ưu thế.

- **Tập trung vào Giải pháp Tổng thể:** Nghiên cứu này đánh giá hiệu năng của hệ thống CSDL cụ thể (HeavyDB, DuckDB) được triển khai trên nền tảng phần cứng phù hợp nhất với kiến trúc của chúng, thay vì chỉ so sánh "GPU vs CPU" một cách trừu tượng trên phần cứng không tối ưu cho cả hai.

Mặc dù có sự khác biệt về cấu hình hỗ trợ (CPU/RAM), lựa chọn này cho phép tiến hành một cuộc đối đầu công bằng hơn về mặt tiềm năng công nghệ và rút ra những kết luận sâu sắc, thực tế hơn về vai trò và hiệu quả của GPU trong việc tăng tốc cơ sở dữ liệu phân tích hiện đại.

1.1.2. Setup môi trường thực nghiệm



Hình 1: Sơ đồ luồng cài đặt môi trường thực nghiệm

Trong quá trình thiết lập môi trường thực nghiệm, chúng tôi đã sử dụng các máy ảo AWS và thực hiện các bước như sau:

- **Hệ điều hành**: Các máy ảo (HeavyDB machine và DuckDB machine) được cài đặt hệ điều hành **Ubuntu 22.04**. Phiên bản này được lựa chọn vì tính ổn định, hiệu suất cao và khả năng tương thích tốt với các công cụ cần thiết như HeavyDB và DuckDB.
- **Tạo AMI cho HeavyDB**: Một AMI (Amazon Machine Image) chứa sẵn HeavyDB và các cấu hình cần thiết được tạo trên nền tảng Ubuntu 22.04 để tiết kiệm thời gian và tránh việc phải cài đặt lại HeavyDB nhiều lần trong quá trình thử nghiệm.
- **Cài đặt DuckDB**: DuckDB được cài đặt trực tiếp trên máy ảo DuckDB mỗi khi cần, không sử dụng AMI riêng do việc cài đặt DuckDB khá nhanh chóng và đơn giản. Điều này giúp tiết kiệm dung lượng lưu trữ và thời gian tạo AMI.
- **Tạo Data Storage Volume**:
    - Một **Data Storage Volume** được tạo để lưu trữ các bộ dữ liệu lớn (>= 50GB). Đây là nơi dữ liệu được sinh ra chung cho cả HeavyDB và DuckDB sử dụng trong các bài thử nghiệm benchmark.
    - Đối với các bộ dữ liệu nhỏ hơn (1GB, 5GB, 10GB, 20GB), dữ liệu được tạo trực tiếp trên máy ảo HeavyDB hoặc DuckDB mà không cần sử dụng Data Storage Volume.
- **Quy trình xử lý dữ liệu lớn**:
    - Đối với các bộ dữ liệu lớn (>= 50GB), thời gian sinh dữ liệu khá lâu (khoảng >= 20 phút). Vì vậy, Data Storage Volume được sử dụng để lưu trữ các bộ dữ liệu này.
    - Khi cần sử dụng, máy ảo HeavyDB hoặc DuckDB chỉ cần **mount** Data Storage Volume vào và thực hiện các tác vụ cần thiết.
- **Luồng hoạt động**:
    - Máy ảo HeavyDB được sử dụng để chạy các tác vụ liên quan đến GPU.

- Máy ảo DuckDB được sử dụng để chạy các tác vụ liên quan đến CPU.
- Cả hai máy ảo có thể truy cập chung vào Data Storage Volume để sử dụng các bộ dữ liệu lớn khi cần thiết. Khi cần dùng thì HeavyDB machine hoặc DuckDB machine sẽ mount Data Storage Volume vào và thực hiện các tác vụ cần thiết. Việc này giúp tiết kiệm thời gian và dung lượng lưu trữ, đồng thời đảm bảo tính nhất quán của dữ liệu giữa hai hệ thống.

Cách thiết lập này đảm bảo tính linh hoạt, tiết kiệm thời gian và tối ưu hóa hiệu suất trong quá trình thực nghiệm.

1.2. Dữ liệu thử nghiệm

Dữ liệu thử nghiệm trong nghiên cứu được sinh ra từ hai bộ công cụ chuẩn hóa là TPC-H và TPC-DS, thông qua tính năng tích hợp sẵn của hệ quản trị cơ sở dữ liệu DuckDB. Việc sử dụng dữ liệu từ các chuẩn này nhằm đảm bảo tính khách quan, tính mô phỏng thực tế và khả năng so sánh kết quả giữa các hệ thống.

1.2.1. Nguồn dữ liệu

TPC-H là một bộ benchmark phổ biến trong việc đánh giá hiệu suất của các hệ quản trị cơ sở dữ liệu hướng phân tích (OLAP). Bộ dữ liệu này mô phỏng các nghiệp vụ phân tích truyền thống của doanh nghiệp như quản lý đơn hàng, khách hàng, và sản phẩm. Mô hình dữ liệu được xây dựng theo dạng quan hệ chuẩn hóa, giúp phản ánh đúng các phép toán JOIN phức tạp trong truy vấn phân tích.

TPC-DS được thiết kế để phản ánh các truy vấn phân tích phức tạp hơn, đặc biệt trong môi trường kho dữ liệu doanh nghiệp. Mô hình dữ liệu trong TPC-DS có cấu trúc dạng ngôi sao (star schema) và bông tuyết (snowflake schema), phù hợp để kiểm tra khả năng xử lý dữ liệu lớn, đa chiều và sự tối ưu hóa của hệ thống trong các bài toán truy vấn thực tế.

DuckDB được sử dụng như công cụ tạo dữ liệu, với khả năng hỗ trợ sinh dữ liệu TPC-H và TPC-DS thông qua câu lệnh tích hợp. Việc sử dụng DuckDB

giúp rút ngắn quy trình chuẩn bị dữ liệu và đảm bảo tính tương thích với hệ thống thử nghiệm.

1.2.2. Kích thước dữ liệu

Dữ liệu thử nghiệm được tạo với các kích thước khác nhau để kiểm tra khả năng xử lý và hiệu suất của hệ thống:

- **1GB**: Dữ liệu nhỏ, phù hợp để kiểm tra hiệu suất cơ bản.
- **5GB**: Dữ liệu trung bình, kiểm tra khả năng xử lý của hệ thống.
- **10GB**: Dữ liệu lớn, kiểm tra hiệu suất tối ưu.
- **20GB**: Dữ liệu rất lớn, kiểm tra khả năng xử lý dữ liệu lớn của CPU và GPU.
- **50GB**: Dữ liệu cực lớn, kiểm tra khả năng mở rộng tối đa của hệ thống.
- **100GB**: Dữ liệu khổng lồ, kiểm tra giới hạn hiệu suất của hệ thống trong các bài toán phân tích dữ liệu lớn.

Dưới đây là bảng tổng hợp kích thước dữ liệu và thời gian sinh dữ liệu cho TPC-H và TPC-DS (Chạy trên máy c7a.8xlarge):

Kích thước (GB)	Tổng số bản ghi TPC-H	Tổng số bản ghi TPC-DS	Thời gian sinh dữ liệu TPC-H (s)	Thời gian sinh dữ liệu TPC-DS (s)
1	8.661.245	19.557.579	9	19
5	43.299.825	79.615.123	41	75
10	86.586.082	191.500.208	86	155
20	173.194.638	131.673.124	164	239
30	259.798.402	200.436.595	241	369
50	433.005.841	337.148.059	406	597
100	866.037.932	959.031.513	800	1268

Bảng 1: Tổng số bản ghi và thời gian sinh dữ liệu cho TPC-H và TPC-DS

1.2.3. Cách tạo bộ dữ liệu

Dữ liệu thử nghiệm được tạo tự động bằng cách sử dụng API của DuckDB thông qua script `generate_data.sh`. Script này hỗ trợ sinh dữ liệu từ hai bộ benchmark chuẩn là TPC-H và TPC-DS với các kích thước khác nhau. Quy trình cụ thể như sau:

- **Công cụ sử dụng**:
  - DuckDB được sử dụng làm công cụ chính để sinh dữ liệu. DuckDB hỗ trợ tích hợp các plugin TPC-H và TPC-DS, cho phép tạo dữ liệu trực tiếp thông qua các hàm SQL.
  - Script `generate_data.sh` được viết để tự động hóa quá trình tạo dữ liệu, đảm bảo tính nhất quán và dễ dàng mở rộng.

- **Tham số đầu vào**:
  - Script nhận hai tham số:
    - `TYPE`: Loại benchmark (1 cho TPC-H, 2 cho TPC-DS).
    - `SCALE_FACTOR`: Kích thước dữ liệu cần tạo (ví dụ: 1, 5, 10, 20, 50, 100).

- **Quy trình tạo dữ liệu**:
  - Dựa trên tham số `TYPE`, script xác định loại benchmark:
    - Với `TYPE=1`, script sử dụng plugin TPC-H (`tpch`) và hàm `dbgen` để tạo dữ liệu.
    - Với `TYPE=2`, script sử dụng plugin TPC-DS (`tpcds`) và hàm `dsdgen` để tạo dữ liệu.
  - Script kiểm tra xem file cơ sở dữ liệu DuckDB đã tồn tại hay chưa. Nếu có, file sẽ bị xóa để đảm bảo dữ liệu được tạo mới hoàn toàn.
  - DuckDB được chạy với các lệnh SQL sau:

```
INSTALL <plugin>;
LOAD <plugin>;
SELECT * FROM <gen_function>(sf=<SCALE_FACTOR>);
```

Trong đó:

- `<plugin>` là `tpch` hoặc `tpcds` tùy thuộc vào loại benchmark.
- `<gen_function>` là `dbgen` hoặc `dsdgen`.
- `<SCALE_FACTOR>` là kích thước dữ liệu cần tạo.

- **Ví dụ sử dụng**:
  - Để tạo dữ liệu TPC-H với kích thước 10GB:

```
./generate_data.sh 1 10
```

  - Để tạo dữ liệu TPC-DS với kích thước 50GB:

```
./generate_data.sh 2 50
```

- **Lưu trữ dữ liệu**:
  - Dữ liệu được lưu trữ trong các file DuckDB với cấu trúc thư mục như sau:

```
tpc-h/tpc-h_nckh.duckdb
tpc-ds/tpc-ds_nckh.duckdb
```

  - Các file này chứa toàn bộ dữ liệu được sinh ra, sẵn sàng để sử dụng trong các bài thử nghiệm benchmark.

- **Ưu điểm của quy trình**:
  - Tự động hóa hoàn toàn việc tạo dữ liệu, giảm thiểu sai sót thủ công.
  - Hỗ trợ linh hoạt cho cả hai bộ benchmark TPC-H và TPC-DS.
  - Dễ dàng mở rộng với các kích thước dữ liệu khác nhau thông qua tham số `SCALE_FACTOR`.

Phương pháp này đảm bảo dữ liệu thử nghiệm được tạo ra một cách nhất quán, nhanh chóng và phù hợp với các yêu cầu của bài toán phân tích dữ liệu lớn.

1.2.4. Danh sách các bảng dữ liệu thuộc bộ TPC-DS

Khi tạo bộ dữ liệu bằng cách sử dụng plugin TPC-DS trong DuckDB với lệnh `SELECT * FROM dsdgen(sf=1);` có 24 bảng được sinh ra, các bảng này sẽ tuân theo tiêu chuẩn của TPC-DS. Dữ liệu bao gồm các **bảng sự kiện (Fact Tables) và bảng chiều (Dimension Tables)** như sau:

1.2.4.1. Bảng sự kiện (Fact Tables)

Các bảng sự kiện chứa dữ liệu giao dịch hoặc sự kiện chính, gồm 7 bảng:

- **store_sales**: Doanh số bán hàng tại cửa hàng.
- **store_returns**: Hàng hóa trả lại tại cửa hàng.
- **web_sales**: Doanh số bán hàng trực tuyến.
- **web_returns**: Hàng hóa trả lại từ bán hàng trực tuyến.
- **catalog_sales**: Doanh số bán hàng qua danh mục.
- **catalog_returns**: Hàng hóa trả lại từ bán hàng qua danh mục.
- **inventory**: Thông tin tồn kho sản phẩm.

1.2.4.2. Bảng chiều (Dimension Tables)

Các bảng chiều chứa thông tin mô tả liên quan đến các bảng sự kiện, gồm 17 bảng:

- **customer**: Thông tin khách hàng.
- **customer_address**: Địa chỉ khách hàng.
- **customer_demographics**: Thông tin nhân khẩu học của khách hàng.
- **date_dim**: Thông tin về ngày tháng.
- **time_dim**: Thông tin về thời gian.
- **item**: Thông tin sản phẩm.
- **promotion**: Thông tin về các chương trình khuyến mãi.
- **store**: Thông tin về cửa hàng.
- **web_site**: Thông tin về trang web bán hàng.
- **web_page**: Thông tin về các trang web cụ thể.
- **warehouse**: Thông tin về kho hàng.
- **ship_mode**: Thông tin về phương thức vận chuyển.

- **call_center**: Thông tin về trung tâm chăm sóc khách hàng.
- **income_band**: Phân loại thu nhập của khách hàng.
- **reason**: Lý do trả lại hàng.
- **household_demographics**: Thông tin nhân khẩu học của hộ gia đình.
- **catalog_page**: Thông tin về các trang danh mục sản phẩm

1.2.4.3. Tổ chức dữ liệu

- **Bảng sự kiện** thường chứa các giao dịch lớn và liên kết với các bảng chiều thông qua khóa ngoại (foreign key).
- **Bảng chiều** cung cấp thông tin chi tiết để phân tích dữ liệu từ các bảng sự kiện.

1.2.5. Danh sách các bảng dữ liệu thuộc bộ TPC-H

Khi tạo bộ dữ liệu bằng cách sử dụng plugin TPC-H trong DuckDB với lệnh `SELECT * FROM dbgen(sf=1);` có 8 bảng được sinh ra. Các bảng này tuân theo tiêu chuẩn của TPC-H và được chia thành hai loại chính: **bảng sự kiện (Fact Tables)** và **bảng chiều (Dimension Tables)**.

1.2.5.1. Bảng sự kiện (Fact Tables)

Bảng sự kiện chứa dữ liệu giao dịch hoặc sự kiện chính, gồm 2 bảng:

- **lineitem**: Chi tiết các mục trong đơn hàng, bao gồm thông tin về sản phẩm, số lượng, giá cả, chiết khấu, và trạng thái giao hàng.
- **orders**: Thông tin về đơn hàng, bao gồm khách hàng, ngày đặt hàng, trạng thái đơn hàng, và tổng giá trị.

1.2.5.2. Bảng chiều (Dimension Tables)

Bảng chiều chứa thông tin mô tả liên quan đến các bảng sự kiện, gồm 6 bảng:

- **customer**: Thông tin khách hàng, bao gồm tên, địa chỉ, phân khúc thị trường, và quốc gia.
- **nation**: Thông tin về các quốc gia, bao gồm tên quốc gia và khu vực liên kết.

- **region**: Thông tin về các khu vực, bao gồm tên khu vực và mô tả.
- **part**: Thông tin về sản phẩm, bao gồm tên, loại, kích thước, và nhà cung cấp.
- **supplier**: Thông tin về nhà cung cấp, bao gồm tên, địa chỉ, quốc gia, và tài khoản.
- **partsupp**: Thông tin về mối quan hệ giữa sản phẩm và nhà cung cấp, bao gồm giá cả và số lượng hàng tồn kho.

1.2.5.3. Tổ chức dữ liệu

- **Bảng sự kiện** thường chứa các giao dịch lớn và liên kết với các bảng chiều thông qua khóa ngoại (foreign key).
- **Bảng chiều** cung cấp thông tin chi tiết để phân tích dữ liệu từ các bảng sự kiện.

1.2.5.4. Mối quan hệ giữa các bảng

- Bảng **orders** liên kết với bảng **customer** thông qua khóa ngoại `custkey`.
- Bảng **lineitem** liên kết với bảng **orders** thông qua khóa ngoại `orderkey`.
- Bảng **customer** liên kết với bảng **nation** thông qua khóa ngoại `nationkey`.
- Bảng **nation** liên kết với bảng **region** thông qua khóa ngoại `regionkey`.
- Bảng **partsupp** liên kết với bảng **part** và **supplier** thông qua các khóa ngoại `partkey` và `suppkey`.
- Bảng **supplier** liên kết với bảng **nation** thông qua khóa ngoại `nationkey`.

Cấu trúc dữ liệu này được thiết kế để hỗ trợ các truy vấn phân tích phức tạp, đặc biệt là các phép toán JOIN giữa các bảng sự kiện và bảng chiều.

1.3. Tạo các bộ truy vấn thử nghiệm

Trong quá trình thực nghiệm, các bộ truy vấn thử nghiệm được tạo ra dựa trên hai bộ kit chuẩn là **TPC-H** và **TPC-DS**. Các bộ kit này cung cấp các truy vấn chuẩn hóa, được thiết kế để đánh giá hiệu suất của các hệ quản trị cơ sở dữ liệu

trong các bài toán phân tích dữ liệu lớn. Dưới đây là chi tiết cách tạo các bộ truy vấn thử nghiệm:

1.3.1. Bộ truy vấn TPC-H

- **Nguồn bộ kit**: Bộ truy vấn TPC-H được lấy từ [TPC-H Kit trên GitHub] (https://github.com/gregrahn/tpch-kit).
- **Đặc điểm**:
    - Bộ truy vấn TPC-H bao gồm 22 truy vấn chuẩn hóa, được thiết kế để kiểm tra hiệu suất của các hệ quản trị cơ sở dữ liệu trong các bài toán phân tích dữ liệu truyền thống.
    - Các truy vấn này tập trung vào các phép toán JOIN phức tạp, tính toán tổng hợp (aggregation), và các phép lọc dữ liệu (filtering).
- **Cách sử dụng**:
    - Bộ truy vấn TPC-H được sử dụng chung cho tất cả các bộ dữ liệu (1GB, 5GB, 10GB, 20GB, 50GB, 100GB).
    - Điều này đảm bảo tính nhất quán trong việc so sánh hiệu suất giữa các kích thước dữ liệu khác nhau.
- **Triển khai**:
    - Bộ truy vấn TPC-H được sinh ra từ công cụ qgen trong tpch-kit, dựa trên file danh sách truy vấn (query.lst) và thư mục chứa các template (queries).
    - Các truy vấn được sinh ra có định dạng SQL chuẩn, và được điều chỉnh (nếu cần) để tương thích với cú pháp của hệ thống truy vấn đang được benchmark (ví dụ: sửa đổi một số cú pháp không tương thích với DuckDB hoặc HeavyDB).
    - Các truy vấn được lưu trong thư mục `sql/queries/` và được thực thi tuần tự trên các bộ dữ liệu khác nhau.

1.3.2. Bộ truy vấn TPC-DS

- **Nguồn bộ kit**: Bộ truy vấn TPC-DS được lấy từ [TPC-DS Kit trên GitHub] (https://github.com/gregrahn/tpcds-kit).
- **Đặc điểm**:
    - Bộ truy vấn TPC-DS bao gồm 99 truy vấn chuẩn hóa, được thiết kế để kiểm tra hiệu suất của các hệ quản trị cơ sở dữ liệu trong môi trường kho dữ liệu doanh nghiệp.
    - Các truy vấn này tập trung vào các bài toán phân tích phức tạp, bao gồm các phép toán JOIN đa chiều, tính toán tổng hợp, và các phép lọc dữ liệu phức tạp.
- **Cách sử dụng**:
    - Đối với TPC-DS, mỗi bộ dữ liệu (1GB, 5GB, 10GB, 20GB, 50GB, 100GB) sẽ có một bộ truy vấn tương ứng.
    - Điều này đảm bảo rằng các truy vấn được tối ưu hóa cho từng kích thước dữ liệu cụ thể, phản ánh đúng thực tế khi triển khai các bài toán phân tích dữ liệu lớn.
- **Triển khai**:
    - Các truy vấn được sinh tự động bằng cách sử dụng script `dsqgen` trong bộ kit TPC-DS.
    - Ví dụ, để tạo bộ truy vấn cho bộ dữ liệu 10GB, lệnh sau được sử dụng:

```
./dsqgen -DIRECTORY ../query_templates -INPUT ../query_templates/
templates.lst -SCALE 10 -OUTPUT_DIR ../queries_10gb
```

    - Các truy vấn được lưu trong thư mục tương ứng, ví dụ: `queries_1gb/`, `queries_10gb/`, `queries_100gb/`.

1.3.3. So sánh cách tạo bộ truy vấn giữa TPC-H và TPC-DS

- **TPC-H**:

    - Sử dụng chung một bộ truy vấn cho tất cả các kích thước dữ liệu.
    - Phù hợp với các bài toán phân tích truyền thống, nơi các truy vấn không phụ thuộc vào kích thước dữ liệu.

- **TPC-DS**:
    - Tạo bộ truy vấn riêng cho từng kích thước dữ liệu.
    - Phù hợp với các bài toán phân tích phức tạp, nơi các truy vấn cần được tối ưu hóa cho từng kích thước dữ liệu cụ thể.

1.3.4. Kết luận

Việc sử dụng hai bộ kit TPC-H và TPC-DS với cách tiếp cận khác nhau trong việc tạo bộ truy vấn giúp đảm bảo tính toàn diện trong quá trình đánh giá hiệu suất của hệ thống. TPC-H cung cấp một cách tiếp cận đơn giản và nhất quán, trong khi TPC-DS phản ánh các bài toán thực tế phức tạp hơn, yêu cầu tối ưu hóa truy vấn theo kích thước dữ liệu.

1.4. Cách triển khai benchmark

1.4.1. Benchmark trên DuckDB

Một hệ thống benchmark tự động đã được xây dựng để đánh giá hiệu năng DuckDB một cách khoa học, khách quan và dễ tái tạo trên các máy ảo AWS, tạo nền tảng so sánh với HeavyDB.

1.4.1.1. Quy trình thực hiện benchmark

- **Cài đặt DuckDB**:
    - Sử dụng script `install_duckdb.sh` để cài đặt DuckDB trên hệ thống.
    - Script này đảm bảo phiên bản DuckDB được cài đặt là phiên bản mới nhất và ổn định.

- **Chuẩn bị dữ liệu và chạy benchmark**:

◦ Sử dụng script chính `main.sh` để lấy dữ liệu đã được tạo sẵn và tiến hành benchmark.

◦ Cú pháp sử dụng: `./main.sh <type> <scale_factor> <num_runs> <aws_instance>`

– `type`: `1` cho TPC-H, `2` cho TPC-DS
– `scale_factor`: Kích thước bộ dữ liệu (ví dụ: 1, 5, 10, 20, 30, 50, 100)
– `num_runs`: Số lần lặp lại benchmark để đảm bảo kết quả ổn định
– `aws_instance`: Loại instance AWS đang sử dụng (ví dụ: on_c7a_8xlarge, on_g4dn_xlarge)

◦ **Vị trí dữ liệu**:

– Dữ liệu cho TPC-H được lưu tại: `/mnt/data/storage/tpch/<scale_factor>GB.duckdb`
– Dữ liệu cho TPC-DS được lưu tại: `/mnt/data/storage/tpcds/<scale_factor>GB.duckdb`

◦ **Vị trí file truy vấn (các file tương tự như của HeavyDB)**:

– Các truy vấn TPC-H: `tpc-h/sql/queries_<scale_factor>/`
– Các truy vấn TPC-DS: `tpc-ds/sql/query<scale_factor>/splited/`

- **Thực thi benchmark**:
  - Script `benchmark.sh` được gọi tự động bởi `main.sh` cho mỗi lần chạy.
  - Có thể chạy trực tiếp script này với cú pháp: `./process/benchmark.sh <type> <scale_factor> <run_number> <aws_instance>`
- **Lưu trữ kết quả**:

- Kết quả benchmark được lưu tại thư mục `../benchmark_result/<aws_instance>/duckdb/`.
- Định dạng kết quả bao gồm thời gian thực thi của từng truy vấn và tổng thời gian chạy.

1.4.1.2. Ví dụ thực hiện

Ví dụ, để thực hiện benchmark TPC-H với scale factor 10, lặp lại 3 lần trên instance c7a.8xlarge, sử dụng lệnh:

```
./main.sh 1 10 3 on_c7a_8xlarge
```

Lệnh này sẽ:

- Kiểm tra xem dữ liệu TPC-H với scale factor 10 đã tồn tại chưa
- Nếu chưa, script sẽ tự động tạo dữ liệu
- Thực hiện 3 lần benchmark trên bộ dữ liệu này
- Lưu kết quả vào thư mục `../benchmark_result/on_c7a_8xlarge/duckdb/`

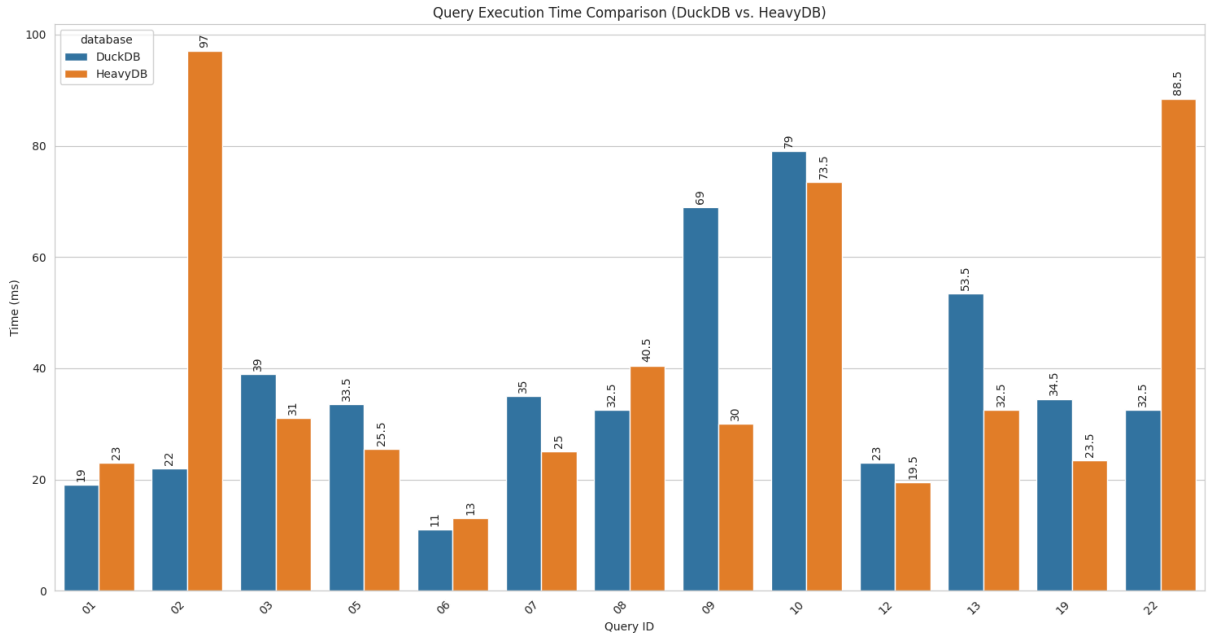
1.4.1.3. Ưu điểm của phương pháp benchmark

- **Tự động hoá cao**: Toàn bộ quy trình từ cài đặt, chuẩn bị dữ liệu đến thực thi benchmark được tự động hóa, giảm thiểu sai sót do con người.
- **Khả năng tái tạo**: Có thể dễ dàng tái tạo kết quả benchmark trên các môi trường khác nhau.
- **Linh hoạt**: Hỗ trợ cả hai bộ benchmark phổ biến (TPC-H và TPC-DS) với nhiều kích thước dữ liệu khác nhau.
- **Phân loại kết quả**: Kết quả được tổ chức theo loại instance AWS, giúp dễ dàng so sánh hiệu năng giữa các cấu hình phần cứng khác nhau.
- **Độ tin cậy**: Việc chạy nhiều lần (thông qua tham số `num_runs`) giúp đảm bảo kết quả ổn định và đáng tin cậy.

1.5. Kết quả thực nghiệm

1.5.1. Kết quả từ bộ benchmark TPC-H

1.5.1.1. Đối với bộ dữ liệu 1GB



Hình 2: Biểu đồ so sánh thời gian truy vấn trung bình giữa HeavyDB và DuckDB đối với bộ dữ liệu 1GB

- **Phân tích Tổng Quan**: Trên bộ dữ liệu TPC-H 1GB, kết quả benchmark cho thấy sự phân hóa về hiệu suất giữa DuckDB (CPU) và HeavyDB (GPU) tùy thuộc vào từng truy vấn. Không có một hệ thống nào chiếm ưu thế tuyệt đối ở quy mô dữ liệu này.
- **Phân tích Chi Tiết về Thời Gian Thực Thi Truy Vấn**:
    - **HeavyDB chậm hơn rất nhiều so với DuckDB (từ 4 lần trở lên):** Q04, Q16, Q18, Q20.
    - **HeavyDB chậm hơn đáng kể so với DuckDB (từ 2 đến dưới 4 lần):** Q02, Q17, Q22.
    - **HeavyDB chậm hơn so với DuckDB (từ 1 đến dưới 2 lần)** Q08.
    - **Thời gian thực thi gần như tương đương:** Q10.
    - **HeavyDB nhanh hơn so với DuckDB (từ 1 đến dưới 2 lần):** Q01, Q03, Q05, Q06, Q12, Q19.

- **HeavyDB nhanh hơn đáng kể so với DuckDB (từ 1.5 lần trở lên):** Q07, Q09, Q13.

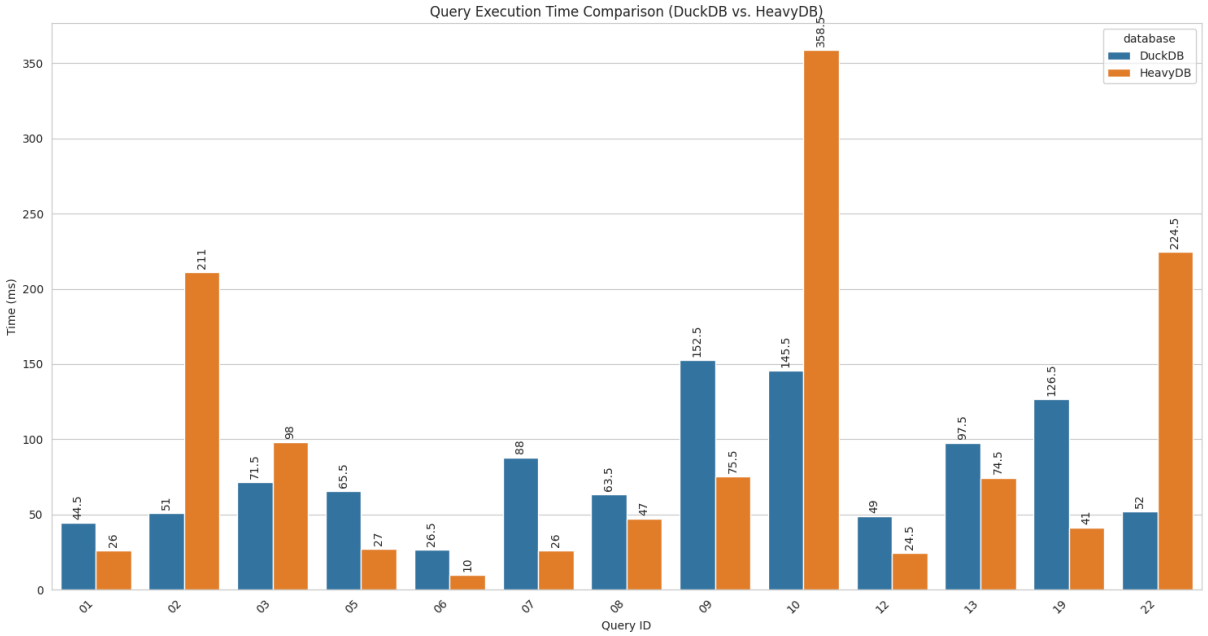
- **Nhận xét Chi Tiết**:
  - Các truy vấn Q04, Q16, Q18, Q20 cho thấy kiến trúc CPU của DuckDB và các kỹ thuật tối ưu hóa cho CPU hoạt động hiệu quả hơn nhiều so với HeavyDB (GPU) trên bộ dữ liệu 1GB.
  - Ở các truy vấn Q02, Q17, Q22, DuckDB vẫn nhanh hơn đáng kể, cho thấy có thể có những đặc điểm truy vấn mà CPU xử lý tốt hơn ở quy mô này.
  - HeavyDB chỉ chậm hơn một chút so với DuckDB ở Q08.
  - Hiệu suất của hai hệ thống là gần như tương đương ở Q10.
  - HeavyDB thể hiện lợi thế về tốc độ, dù không quá lớn, ở các truy vấn Q01, Q03, Q05, Q06, Q12, và Q19, gợi ý rằng kiến trúc GPU có thể khai thác được một mức độ song song hóa nhất định.
  - Đáng chú ý, HeavyDB nhanh hơn đáng kể ở các truy vấn Q07, Q09, và Q13. Điều này cho thấy có những loại truy vấn cụ thể mà khả năng song song hóa của GPU mang lại lợi ích rõ rệt, vượt qua được cả overhead ở quy mô dữ liệu 1GB.

- **Đánh giá**:
  - Benchmark trên bộ dữ liệu TPC-H 1GB cho thấy hiệu suất của DuckDB (CPU) và HeavyDB (GPU) phụ thuộc nhiều vào đặc điểm của từng truy vấn. Không có một hệ thống nào chiếm ưu thế chung ở quy mô này.
  - Kiến trúc CPU tỏ ra rất hiệu quả cho một số lượng lớn các truy vấn, đặc biệt là những truy vấn mà HeavyDB chậm hơn đáng kể.
  - Tuy nhiên, GPU cũng chứng minh khả năng tăng tốc đáng kể ở một số truy vấn khác, cho thấy tiềm năng khi xử lý các workload phù hợp.

- Quy mô dữ liệu 1GB có thể là một yếu tố hạn chế khả năng thể hiện toàn diện sức mạnh của HeavyDB do các chi phí overhead liên quan đến GPU.

- **Kết luận**: Kết quả benchmark trên bộ dữ liệu TPC-H 1GB cho thấy sự phức tạp trong việc so sánh hiệu suất giữa kiến trúc CPU và GPU. Hiệu suất tối ưu phụ thuộc vào sự tương thích giữa đặc điểm truy vấn và kiến trúc phần cứng. Việc tiếp tục benchmark trên các bộ dữ liệu lớn hơn sẽ cung cấp cái nhìn rõ ràng hơn về khả năng mở rộng và ưu thế thực sự của HeavyDB (GPU) trong các bài toán phân tích dữ liệu lớn.

1.5.1.2. Đối với bộ dữ liệu 5GB



Hình 3: Biểu đồ so sánh thời gian truy vấn trung bình giữa HeavyDB và DuckDB đối với bộ dữ liệu 5GB

- **Phân tích Tổng Quan**: Trên bộ dữ liệu TPC-H 5GB, kết quả benchmark tiếp tục cho thấy sự khác biệt về hiệu suất giữa DuckDB (CPU) và HeavyDB (GPU) tùy thuộc vào từng truy vấn. Tuy nhiên, so với bộ dữ liệu 1GB, có một sự chuyển dịch đáng chú ý, với HeavyDB thể hiện hiệu suất tốt hơn ở nhiều truy vấn hơn.

- **Phân tích Chi Tiết về Thời Gian Thực Thi Truy Vấn**:

- **HeavyDB chậm hơn rất nhiều so với DuckDB (từ 4 lần trở lên):** Q04, Q16, Q18, Q20.
- **HeavyDB chậm hơn đáng kể so với DuckDB (từ 2 đến dưới 4 lần):** Q17, Q22.
- **HeavyDB chậm hơn so với DuckDB (từ 1 đến dưới 2 lần):** Q02, Q10.
- **Thời gian thực thi gần như tương đương:** Q03.
- **HeavyDB nhanh hơn so với DuckDB (từ 1 đến dưới 2 lần):** Q01, Q05, Q06, Q08, Q12, Q13.
- **HeavyDB nhanh hơn đáng kể so với DuckDB (từ 2 lần trở lên):** Q07, Q09, Q19.

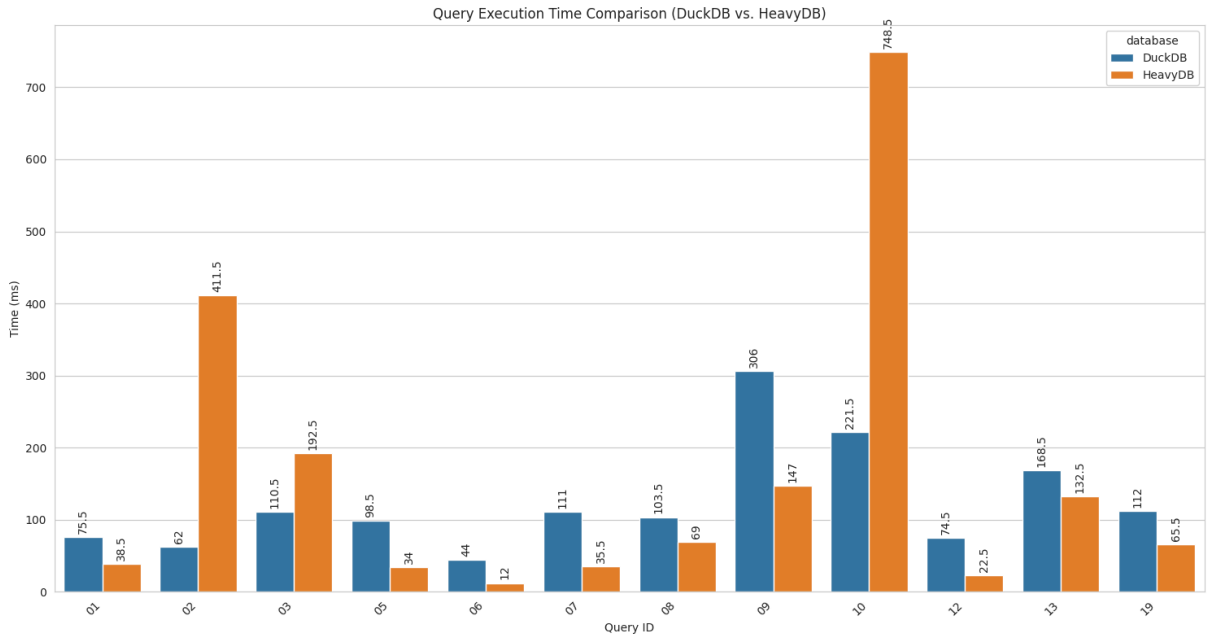
- **Nhận xét Chi Tiết**:
  - Các truy vấn Q04, Q16, Q18, Q20 vẫn cho thấy DuckDB (CPU) có hiệu suất vượt trội, với mức độ chênh lệch thậm chí còn lớn hơn so với bộ dữ liệu 1GB. Điều này gợi ý rằng những truy vấn này có thể đặc biệt phù hợp với cách CPU xử lý hoặc gặp phải những hạn chế trong việc tối ưu hóa trên GPU.
  - Ở các truy vấn Q17 và Q22, DuckDB vẫn nhanh hơn đáng kể, nhưng tỷ lệ chậm hơn của HeavyDB đã giảm so với 1GB, cho thấy có sự cải thiện về hiệu suất của GPU khi kích thước dữ liệu tăng lên.
  - HeavyDB chậm hơn DuckDB ở Q02 và Q10, nhưng mức độ chênh lệch đã giảm đáng kể so với benchmark 1GB. Q03 có thời gian thực thi gần như tương đương giữa hai hệ thống.
  - Đáng chú ý, HeavyDB (GPU) đã vượt trội hơn DuckDB (CPU) ở các truy vấn Q01, Q05, Q06, Q08, Q12, và Q13, với mức độ nhanh hơn từ nhẹ đến đáng kể. Điều này cho thấy khả năng song song hóa của GPU bắt đầu phát huy hiệu quả rõ rệt hơn khi kích thước dữ liệu tăng lên.
  - Các truy vấn Q07, Q09, và Q19 tiếp tục cho thấy HeavyDB nhanh hơn đáng kể so với DuckDB, với tỷ lệ tăng tốc cao hơn so với kết

quả 1GB, củng cố thêm nhận định về sự phù hợp của GPU với các loại truy vấn này.

- **Đánh giá**:
    - Với bộ dữ liệu TPC-H 5GB, HeavyDB (GPU) bắt đầu thể hiện lợi thế hiệu suất rõ rệt hơn so với DuckDB (CPU) ở nhiều truy vấn. Số lượng truy vấn mà HeavyDB nhanh hơn đã tăng lên đáng kể so với 1GB.
    - Kiến trúc GPU cho thấy khả năng tận dụng tốt hơn khối lượng dữ liệu lớn hơn để tăng tốc độ xử lý ở nhiều loại truy vấn, đặc biệt là những truy vấn có tính song song hóa cao.
    - Tuy nhiên, kiến trúc CPU của DuckDB vẫn giữ vững ưu thế ở một số truy vấn nhất định, cho thấy sự khác biệt trong cách hai kiến trúc này xử lý các loại workload khác nhau.
    - Overhead của GPU dường như đã trở nên ít đáng kể hơn so với thời gian tính toán thực tế ở quy mô 5GB đối với nhiều truy vấn.

- **Kết luận**: Benchmark trên bộ dữ liệu TPC-H 5GB cho thấy HeavyDB (GPU) đang dần thể hiện ưu thế về hiệu suất so với DuckDB (CPU) trên nhiều truy vấn khi kích thước dữ liệu tăng lên. Tuy nhiên, DuckDB vẫn là một lựa chọn hiệu quả cho một số loại workload cụ thể. Xu hướng này nhấn mạnh tầm quan trọng của việc đánh giá hiệu suất trên nhiều quy mô dữ liệu để hiểu rõ khả năng của từng hệ thống. Các benchmark tiếp theo trên các bộ dữ liệu lớn hơn sẽ làm rõ hơn về điểm chuyển giao hiệu suất giữa CPU và GPU trong bối cảnh TPC-H.

1.5.1.3. Đối với bộ dữ liệu 10GB



Hình 4: Biểu đồ so sánh thời gian truy vấn trung bình giữa HeavyDB và DuckDB đối với bộ dữ liệu 10GB

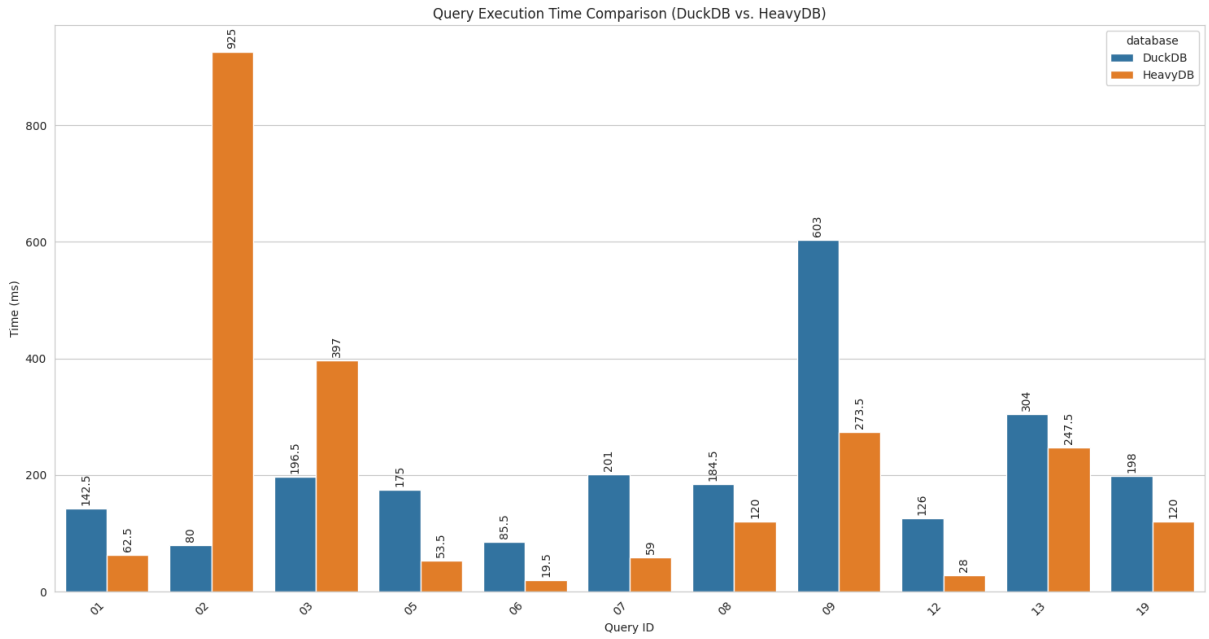
- **Phân tích Tổng Quan**: Xu hướng hiệu suất đã bắt đầu thay đổi rõ rệt hơn so với các bộ dữ liệu nhỏ hơn. HeavyDB (GPU) tiếp tục cho thấy sự cải thiện hiệu suất đáng kể ở nhiều truy vấn khi kích thước dữ liệu tăng lên.
- **Nhận xét Chi Tiết (10GB)**:
    - **HeavyDB tiếp tục cải thiện:** Số lượng truy vấn mà HeavyDB (GPU) nhanh hơn DuckDB (CPU) tiếp tục tăng lên. Mức độ tăng tốc ở nhiều truy vấn cũng trở nên rõ rệt hơn.
    - **Ưu thế rõ ràng của HeavyDB:** Các truy vấn Q01, Q03, Q05, Q06, Q07, Q08, Q09, Q12, Q13, và Q19 cho thấy HeavyDB nhanh hơn đáng kể, với tỷ lệ tăng tốc ngày càng cao.
    - **DuckDB vẫn giữ ưu thế ở một số truy vấn:** DuckDB vẫn nhanh hơn ở các truy vấn Q02, Q04, Q10, Q16, Q17, Q18, Q20, và Q22. Tuy nhiên, mức độ chênh lệch có xu hướng tăng lên ở hầu hết các truy vấn này, cho thấy HeavyDB gặp khó khăn hơn với các loại workload này khi dữ liệu lớn hơn.

- **Sự khác biệt ngày càng lớn ở các truy vấn "khó" cho GPU:** Thời gian thực thi của HeavyDB ở các truy vấn Q04, Q10, Q16, Q18, và Q20 tiếp tục tăng đáng kể so với DuckDB, cho thấy những truy vấn này có thể không tận dụng được kiến trúc GPU hoặc bị ảnh hưởng nặng nề bởi overhead ở quy mô này.

- **Đánh giá (10GB)**:
  - Với kích thước dữ liệu 10GB, HeavyDB (GPU) đã trở thành lựa chọn hiệu quả hơn về hiệu suất cho đa số các truy vấn so với DuckDB (CPU). Lợi thế của GPU trong việc xử lý song song lượng lớn dữ liệu ngày càng rõ ràng.
  - Tuy nhiên, DuckDB (CPU) vẫn duy trì hiệu suất tốt hơn đáng kể ở một số truy vấn cụ thể, cho thấy sự khác biệt cơ bản trong cách hai hệ thống xử lý các loại workload khác nhau.
  - Sự gia tăng đáng kể về khoảng cách hiệu suất ở các truy vấn mà DuckDB nhanh hơn cho thấy có những giới hạn nhất định trong khả năng xử lý của HeavyDB (hoặc cách nó được tối ưu hóa) cho các loại truy vấn này khi dữ liệu lớn hơn.

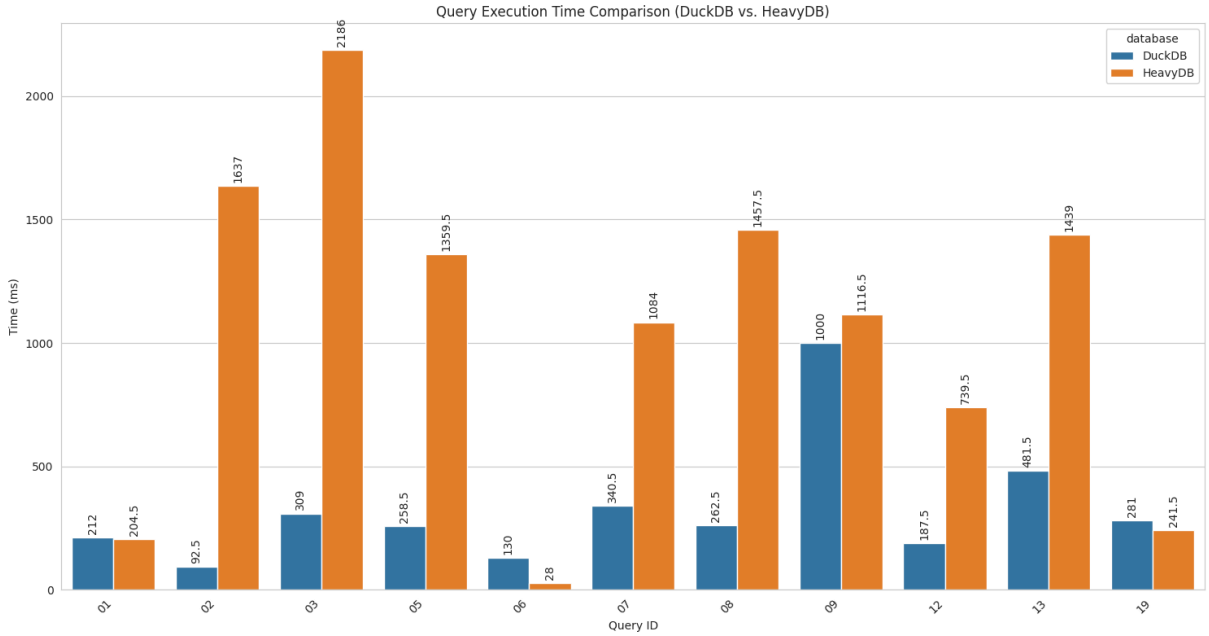
- **Kết luận (10GB)**: Ở quy mô dữ liệu 10GB, HeavyDB (GPU) đang dần khẳng định ưu thế về hiệu suất tổng thể so với DuckDB (CPU) trong benchmark TPC-H. Tuy nhiên, sự khác biệt lớn ở một số truy vấn vẫn cho thấy rằng CPU vẫn có vai trò quan trọng tùy thuộc vào đặc điểm của workload. Các benchmark trên các bộ dữ liệu lớn hơn nữa sẽ tiếp tục làm sáng tỏ xu hướng này.

1.5.1.4. Đối với bộ dữ liệu 20B



Hình 5: Biểu đồ so sánh thời gian truy vấn trung bình giữa HeavyDB và DuckDB đối với bộ dữ liệu 20GB

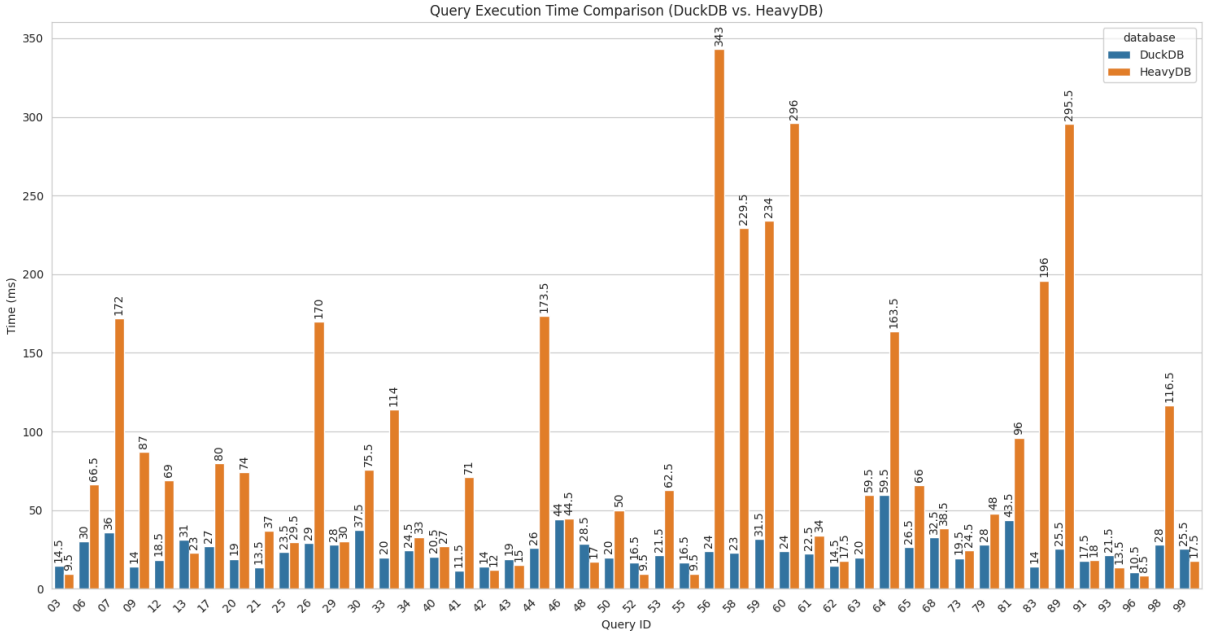
1.5.1.5. Đối với bộ dữ liệu 30B



Hình 6: Biểu đồ so sánh thời gian truy vấn trung bình giữa HeavyDB và DuckDB đối với bộ dữ liệu 20GB

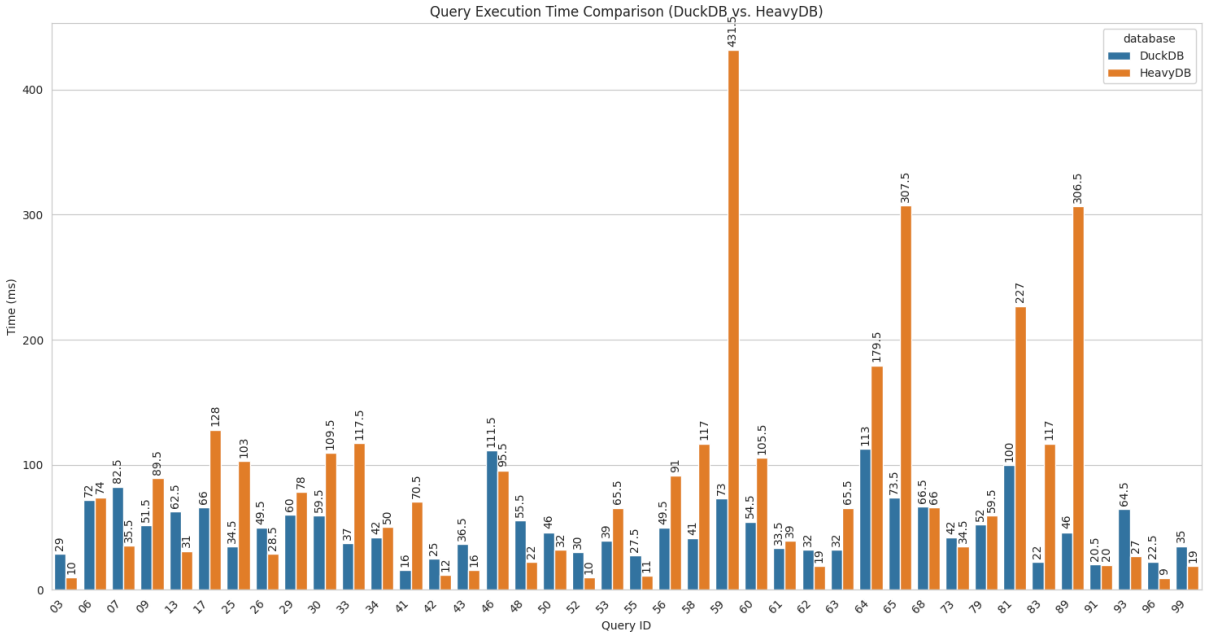
1.5.2. Kết quả từ bộ benchmark TPC-DS

1.5.2.1. Đối với bộ dữ liệu 1GB



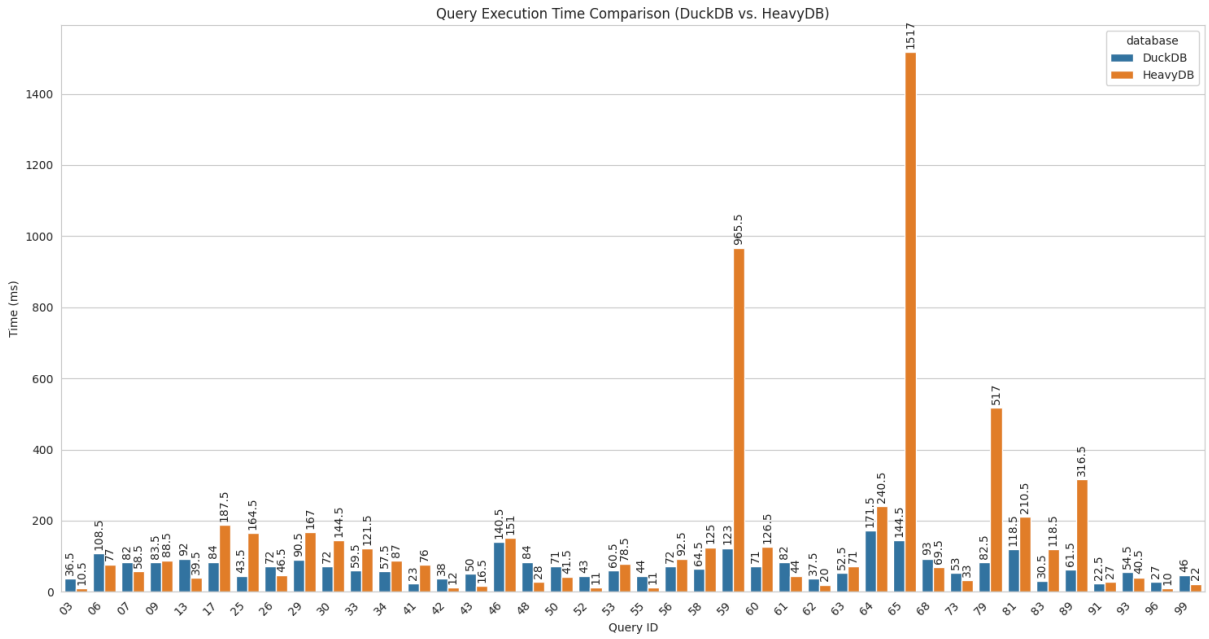
Hình 7: Biểu đồ so sánh thời gian truy vấn trung bình giữa HeavyDB và DuckDB đối với bộ dữ liệu 1GB

1.5.2.2. Đối với bộ dữ liệu 5GB



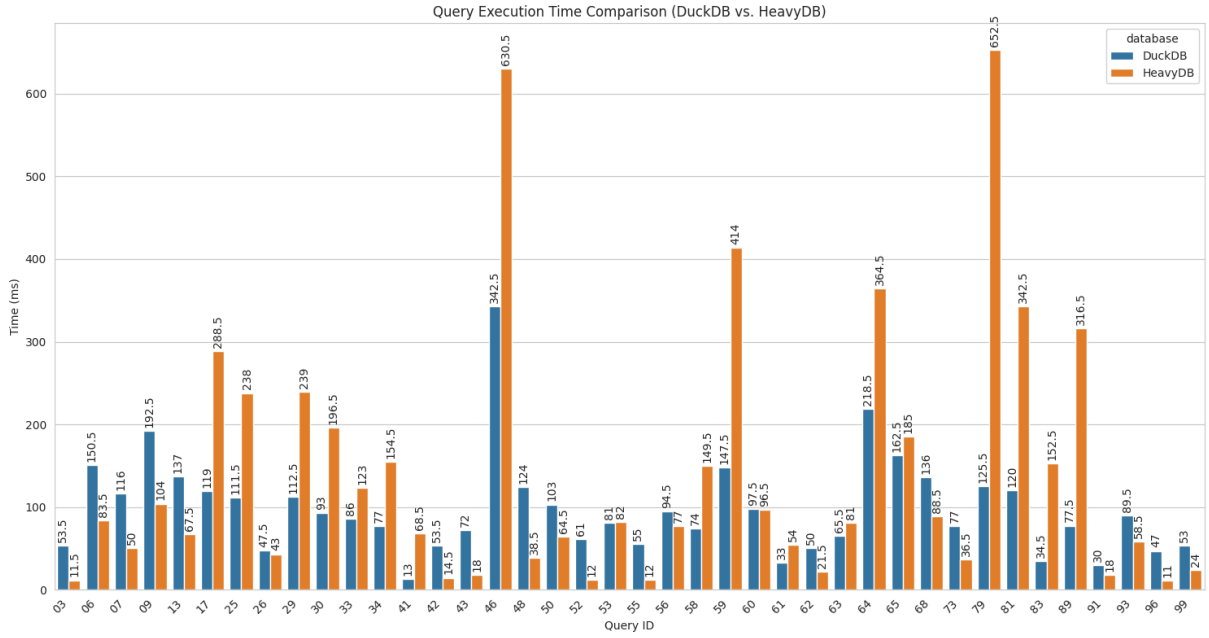
Hình 8: Biểu đồ so sánh thời gian truy vấn trung bình giữa HeavyDB và DuckDB đối với bộ dữ liệu 5GB

1.5.2.3. Đối với bộ dữ liệu 10GB



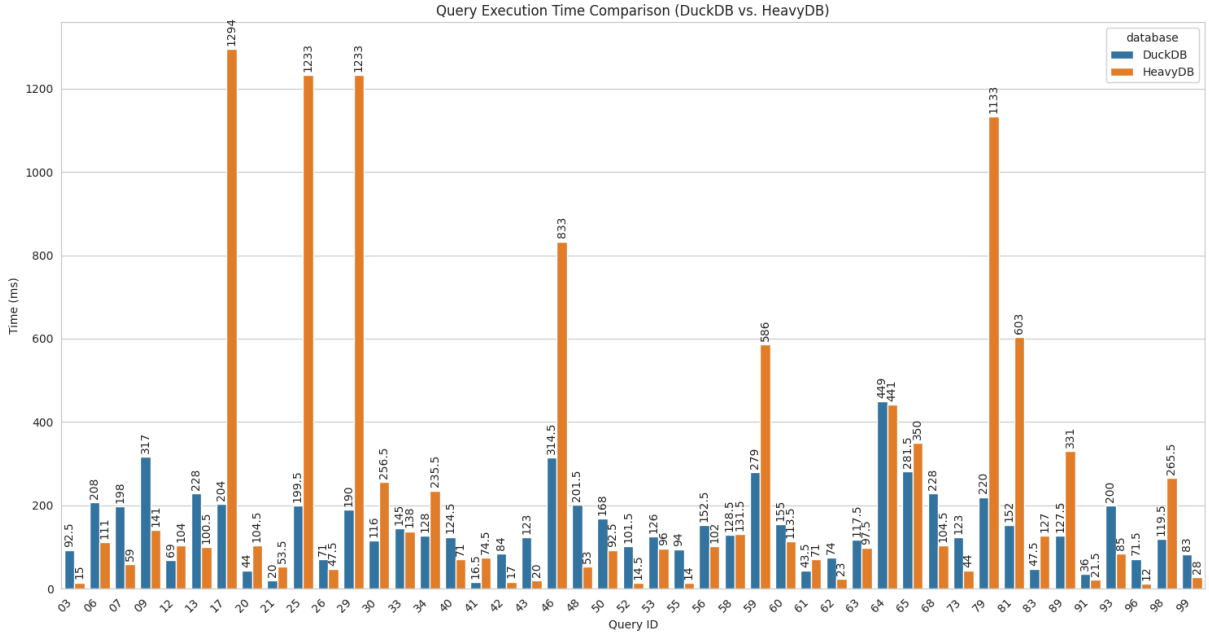
Hình 9: Biểu đồ so sánh thời gian truy vấn trung bình giữa HeavyDB và DuckDB đối với bộ dữ liệu 10GB

1.5.2.4. Đối với bộ dữ liệu 20B



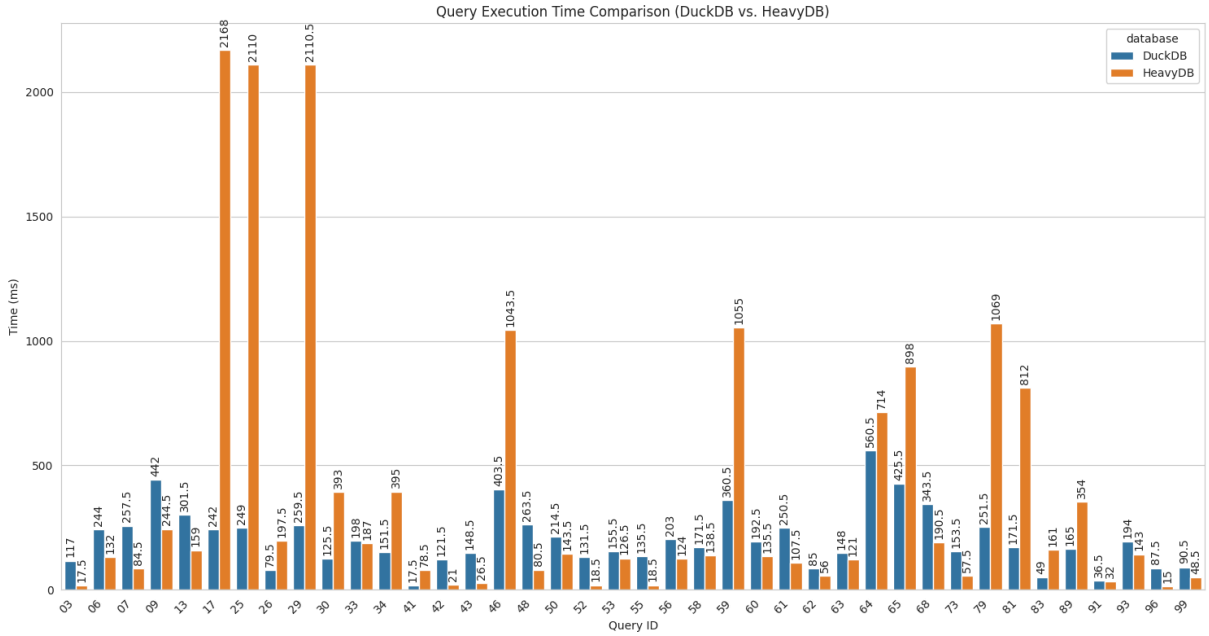
Hình 10: Biểu đồ so sánh thời gian truy vấn trung bình giữa HeavyDB và DuckDB đối với bộ dữ liệu 20GB

1.5.2.5. Đối với bộ dữ liệu 30B



Hình 11: Biểu đồ so sánh thời gian truy vấn trung bình giữa HeavyDB và DuckDB đối với bộ dữ liệu 20GB

1.5.2.6. Đối với bộ dữ liệu 50B



Hình 12: Biểu đồ so sánh thời gian truy vấn trung bình giữa HeavyDB và DuckDB đối với bộ dữ liệu 50GB

1.6. Kết luận

1.7. Hướng phát triển