CHƯƠNG 1. PHƯƠNG PHÁP NGHIÊN CỨU

1.1. Phương pháp nghiên cứu

Trong nghiên cứu này, phương pháp thực nghiệm được sử dụng để so sánh hiệu suất giữa CPU và GPU trong việc xử lý các truy vấn cơ sở dữ liệu. Cụ thể, các truy vấn được thực hiện trên hai hệ quản trị cơ sở dữ liệu:

- **DuckDB**: Chạy trên CPU, được thiết kế để xử lý các truy vấn phân tích dữ liệu (OLAP) trên tập dữ liệu vừa và nhỏ.
- **HeavyDB**: Chạy trên GPU, được tối ưu hóa để xử lý các truy vấn phân tích dữ liệu lớn với tốc độ cao.

Phương pháp nghiên cứu bao gồm các bước sau:

- **Bước 1: Chuẩn bị dữ liệu**: Sử dụng các bộ tiêu chuẩn benchmark TPC-H và TPC-DS để tạo dữ liệu thử nghiệm với các kích thước khác nhau (1GB, 2GB, 5GB, 10GB, 20GB). Dữ liệu được tổ chức theo mô hình quan hệ (TPC-H) và mô hình bông tuyết (snowflake schema) (TPC-DS).
- **Bước 2: Thiết kế truy vấn thử nghiệm**: Sử dụng các truy vấn chuẩn của TPC-H và TPC-DS để kiểm tra khả năng xử lý của DuckDB và HeavyDB. Các truy vấn bao gồm các phép nối (joins), tổng hợp (aggregations), và lọc dữ liệu (filters).
- **Bước 3: Chạy truy vấn**: Thực hiện các truy vấn trên DuckDB (CPU) và HeavyDB (GPU), đồng thời đo thời gian thực thi và thu thập thông tin tài nguyên (CPU, GPU, RAM).
- **Bước 4: Phân tích kết quả**: So sánh thời gian thực thi, mức tiêu thụ tài nguyên, và khả năng mở rộng giữa CPU và GPU.

Phương pháp này giúp đánh giá hiệu suất của DuckDB và HeavyDB trong các tác vụ phân tích dữ liệu, từ đó đưa ra kết luận về hiệu quả của việc sử dụng GPU để tăng tốc truy vấn.

1.2. Lý do lựa chọn công cụ và cấu hình thử nghiệm

1.2.1. Lý do chọn DuckDB và HeavyDB

Mục tiêu cốt lõi của đề tài là so sánh hiệu quả truy vấn giữa hai nền tảng tính toán: CPU truyền thống và GPU song song. Trong bối cảnh đó, DuckDB và HeavyDB được lựa chọn vì chúng đại diện tiêu biểu cho hai hướng tiếp cận kỹ thuật tương ứng, đồng thời đáp ứng các yêu cầu về hiệu năng, khả năng tích hợp và hỗ trợ kỹ thuật cần thiết cho quá trình thực nghiệm.

DuckDB là một HQT CSDL nhúng (in-process), được thiết kế chuyên biệt cho các truy vấn phân tích dữ liệu (OLAP) chạy trên CPU. Hệ thống này sử dụng kiến trúc xử lý theo cột (columnar execution) và khai thác tối đa khả năng tối ưu hóa cấp độ vector hóa, nhờ đó mang lại hiệu suất đáng kể trong các truy vấn dạng quét bảng lớn, kết nối và tổng hợp dữ liệu. DuckDB đặc biệt phù hợp với các khối lượng công việc có quy mô vừa phải, nơi mà truy vấn cần thực hiện nhanh chóng trên một máy đơn lẻ mà không cần triển khai phức tạp. Ngoài ra, DuckDB hỗ trợ tích hợp trực tiếp với các ngôn ngữ lập trình như Python hoặc R mà không cần cài đặt máy chủ độc lập, giúp đơn giản hóa quy trình thực nghiệm và cho phép tập trung vào đo lường hiệu năng truy vấn. Từ đó, DuckDB được sử dụng làm chuẩn tham chiếu cho môi trường xử lý dựa trên CPU.

Ở chiều ngược lại, HeavyDB (trước đây là OmniSciDB) là một HQT CSDL được thiết kế tối ưu hóa cho việc khai thác GPU, hướng đến các ứng dụng phân tích dữ liệu lớn và phân tích thời gian thực. HeavyDB sử dụng kiến trúc xử lý song song hàng loạt, tận dụng hàng nghìn lõi xử lý GPU để thực thi các truy vấn SQL phức tạp với tốc độ cao. Khả năng xử lý theo vector song song trên GPU giúp hệ thống đạt được băng thông truy xuất dữ liệu lớn hơn nhiều lần so với CPU, đặc biệt trong các tác

vụ như lọc dữ liệu, kết nối bảng (join), tổng hợp (aggregate), và xử lý truy vấn không gian (spatial queries). Ngoài ra, HeavyDB cũng cung cấp khả năng truy cập dữ liệu từ bộ nhớ GPU (GPU-resident data), giảm thiểu độ trễ do truyền dữ liệu qua lại giữa CPU và GPU. Nhờ những đặc điểm này, HeavyDB được chọn làm đại diện cho môi trường xử lý truy vấn dựa trên GPU.

Sự kết hợp giữa DuckDB và HeavyDB cho phép thiết lập một hệ đánh giá toàn diện, phản ánh rõ nét sự khác biệt về kiến trúc và hiệu năng giữa hai hướng tiếp cận: xử lý tuần tự dựa trên CPU và xử lý song song dựa trên GPU. Cả hai hệ thống đều hỗ trợ ngôn ngữ truy vấn SQL, đảm bảo tính tương thích khi xây dựng bộ truy vấn thử nghiệm. Bên cạnh đó, việc lựa chọn hai HQT này còn dựa trên khả năng triển khai đơn giản, tính ổn định và mức độ hỗ trợ cộng đồng cao, phù hợp cho các thử nghiệm định lượng trong môi trường nghiên cứu.

1.2.2. Lý do chọn máy AWS EC2 g4dn.xlarge

Việc lựa chọn môi trường triển khai thử nghiệm đóng vai trò then chốt trong nghiên cứu "Tăng tốc truy vấn cơ sở dữ liệu sử dụng GPU". Trong bối cảnh ngân sách nghiên cứu hạn chế và yêu cầu đánh giá hiệu năng thực tế trên cả hai nền tảng CPU và GPU, giải pháp sử dụng máy ảo đám mây (cloud-based instances) được đánh giá là phù hợp hơn so với việc đầu tư hạ tầng vật lý.

So sánh giữa chi phí xây dựng hệ thống phần cứng riêng và chi phí thuê trên nền tảng đám mây cho thấy sự khác biệt rõ rệt. Cụ thể, để xây dựng một máy chủ vật lý có hiệu năng tương đương với cấu hình g4dn.xlarge — bao gồm GPU NVIDIA Tesla T4 (16GB VRAM), CPU Intel Xeon Platinum, và 16GB RAM — cần ngân sách tối thiểu từ 40–60 triệu VND. Trong khi đó, sử dụng máy ảo EC2 g4dn.xlarge chỉ tốn khoảng 13,000 VNĐ/giờ (tương đương $0.526), cho phép triển khai thử nghiệm linh hoạt, không yêu cầu chi phí trả trước và không phát sinh thêm chi phí vận hành như điện, làm mát hay bảo trì.

AWS được chọn làm nền tảng triển khai do khả năng cung cấp tài nguyên GPU đa dạng, mức độ ổn định cao và hỗ trợ kỹ thuật mạnh mẽ. So với các nền tảng đám mây khác như Google Cloud hay Azure, AWS có ưu thế về số lượng vùng khả dụng (availability zones), thời gian khởi tạo máy nhanh, và hỗ trợ các instance tối ưu hóa cho tác vụ tính toán song song như dòng g4dn, p3 và p4. Việc sử dụng AWS cũng giúp đảm bảo tính lặp lại của thử nghiệm, dễ dàng tạo bản sao môi trường (AMI), đồng thời tích hợp tốt với các công cụ như Docker hoặc Terraform để tự động hóa hạ tầng.

Trong số các lựa chọn GPU instance trên AWS, g4dn.xlarge được ưu tiên do đạt được cân bằng tối ưu giữa chi phí và hiệu năng. So với các cấu hình mạnh hơn như g4dn.2xlarge hoặc p3.2xlarge, g4dn.xlarge có mức giá thấp hơn đáng kể nhưng vẫn sở hữu GPU Tesla T4 — một trong những dòng GPU phổ biến nhất hiện nay cho các tác vụ inference, phân tích dữ liệu và xử lý song song. Mặc dù chỉ có 1 GPU và 2 vCPU, cấu hình này đã đủ để đánh giá hiệu năng hệ quản trị cơ sở dữ liệu GPU trong môi trường thử nghiệm với quy mô dữ liệu chuẩn hóa như TPC-H 1GB đến 10GB. Đồng thời, CPU đi kèm (Intel Xeon Platinum 8259CL) cũng có hiệu năng đủ cao để triển khai các truy vấn trên hệ quản trị DuckDB nhằm so sánh trực tiếp giữa hai nền tảng tính toán.

Việc lựa chọn cấu hình g4dn.xlarge thay vì các máy không có GPU (như t3 hoặc m5) cũng giúp đánh giá được rõ ràng lợi thế của GPU trong xử lý dữ liệu song song. Ngược lại, nếu chọn các cấu hình GPU cao cấp hơn như p3 hoặc p4 với giá từ $3 đến $12/giờ, tổng chi phí thử nghiệm sẽ vượt xa ngân sách cho phép mà không mang lại giá trị tương ứng trong bối cảnh quy mô dữ liệu chưa lớn.

Tóm lại, lựa chọn máy AWS EC2 g4dn.xlarge là kết quả của việc cân nhắc giữa chi phí thuê, hiệu năng phần cứng, khả năng triển khai linh hoạt và độ phù hợp với mục tiêu của đề tài. Cấu hình này đảm bảo các yêu cầu kỹ thuật cần thiết để đánh giá hiệu quả tăng tốc truy vấn khi sử dụng GPU, đồng thời hỗ trợ mở rộng linh hoạt trong các thử nghiệm tiếp theo mà không cần đầu tư hạ tầng cố định.

1.3. 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.

1.3.1. 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.

1.3.2. 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.
- **2GB**: Dữ liệu trung bình, kiểm tra khả năng xử lý của hệ thống.
- **5GB**: Dữ liệu lớn hơn, kiểm tra khả năng mở rộng.
- **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.

1.3.3. Cách tạo dữ liệu

Dữ liệu được tạo bằng cách sử dụng DuckDB với plugin TPC-DS. Plugin này cung cấp công cụ tích hợp để sinh dữ liệu thử nghiệm theo tiêu chuẩn TPC-DS với các scale factor khác nhau. Các bước thực hiện như sau:

Bước 1: Cài đặt DuckDB:

- DuckDB cần được cài đặt trên hệ thống. Nếu chưa cài đặt, có thể sử dụng lệnh sau để cài đặt DuckDB qua `pip`:

```
pip install duckdb
```

Bước 2: Cài đặt và tải plugin TPC-DS:

- Sau khi cài đặt DuckDB, mở DuckDB và chạy các lệnh sau:

```
INSTALL tpcds;
LOAD tpcds;
```

- `INSTALL tpcds`: Cài đặt plugin TPC-DS.
- `LOAD tpcds`: Tải plugin TPC-DS vào phiên làm việc hiện tại.

Bước 3: Sinh dữ liệu TPC-DS:

- Sử dụng lệnh `dsdgen` để tạo dữ liệu với scale factor mong muốn. Ví dụ:

```
SELECT * FROM dsdgen(sf=1);
```

- `sf=1`: Sinh dữ liệu với scale factor 1 (tương ứng với 1GB dữ liệu). Có thể thay đổi giá trị `sf` để tạo dữ liệu lớn hơn, ví dụ `sf=10` cho 10GB.

Bước 4: Kiểm tra các bảng dữ liệu:

- Sau khi sinh dữ liệu, có thể kiểm tra danh sách các bảng được tạo bằng lệnh:

```
SHOW TABLES;
```

Bước 5: Truy vấn dữ liệu:

- Sau khi dữ liệu được tạo, bạn có thể thực hiện các truy vấn SQL để kiểm tra dữ liệu. Ví dụ:

```
SELECT * FROM store_sales LIMIT 10;
```

Bước 6: Xuất dữ liệu ra file CSV (nếu cần):

- Nếu cần xuất dữ liệu ra file CSV để sử dụng trong các hệ thống khác, có thể sử dụng lệnh:

```
COPY (SELECT * FROM table_name) TO 'file_path.csv' (HEADER, DELIMITER ',');
```

Ví dụ tạo dữ liệu: Dưới đây là một ví dụ cụ thể để tạo dữ liệu TPC-DS với scale factor 1GB:

```
INSTALL tpcds;
LOAD tpcds;
SELECT * FROM dsdgen(sf=1);
SHOW TABLES;
```

1.3.4. Danh sách các bảng dữ liệu

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:

1.3.4.1. 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.

1.3.4.2. 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

1.3.4.3. 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.

1.4. Các truy vấn thử nghiệm

Các truy vấn thử nghiệm được thiết kế dựa trên hai bộ tiêu chuẩn benchmark phổ biến là **TPC-H** và **TPC-DS**. Các truy vấn này được sử dụng để kiểm tra hiệu suất xử lý dữ liệu của DuckDB (CPU) và HeavyDB (GPU) trong các tác vụ phân tích dữ liệu. Dưới đây là mô tả chi tiết về cách các truy vấn được tạo ra, mục đích và đặc điểm của từng bộ truy vấn.

1.4.1. Bộ truy vấn TPC-H

Cách tạo truy vấn

- Bộ truy vấn TPC-H được lấy từ phiên bản **TPC-H v2.4.0**, một tiêu chuẩn benchmark phổ biến cho các hệ thống xử lý dữ liệu phân tích (OLAP).
- Các truy vấn được sinh ra từ bộ công cụ chính thức của TPC-H, bao gồm 22 truy vấn chuẩn.
- Các truy vấn này được thiết kế để hoạt động trên tập dữ liệu được tổ chức theo mô hình quan hệ (Relational Model), với các bảng như `CUSTOMER`, `ORDERS`, `LINEITEM`, `PART`, v.v.

Mục đích

- Đánh giá hiệu suất của hệ thống cơ sở dữ liệu trong các tác vụ phân tích dữ liệu truyền thống.
- Kiểm tra khả năng xử lý các phép nối phức tạp, tổng hợp dữ liệu, và lọc dữ liệu trên các bảng lớn.
- Đánh giá khả năng tối ưu hóa truy vấn của hệ thống.

Đặc điểm

- Số lượng truy vấn: 22 truy vấn chuẩn.
- Loại tác vụ:
  - Phép nối (joins) giữa các bảng lớn.
  - Tổng hợp dữ liệu (aggregations) như tính tổng, trung bình, tối đa, tối thiểu.
  - Lọc dữ liệu (filters) dựa trên các điều kiện cụ thể.
- Tính phức tạp:
  - Các truy vấn có độ phức tạp trung bình đến cao, với nhiều phép nối và điều kiện lọc.

Ví dụ truy vấn: Dưới đây là câu truy vấn 1 được tạo ra từ bộ dữ liệu 1GB

```
select
  l_returnflag,
  l_linestatus,
  sum(l_quantity) as sum_qty,
  sum(l_extendedprice) as sum_base_price,
```

```
  sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
  sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
  avg(l_quantity) as avg_qty,
  avg(l_extendedprice) as avg_price,
  avg(l_discount) as avg_disc,
  count(*) as count_order
from
  lineitem
where
  l_shipdate <= date '1998-12-01' - interval '90' day (3)
group by
  l_returnflag,
  l_linestatus
order by
  l_returnflag,
  l_linestatus;
```

1.4.2. Bộ truy vấn TPC-DS

Cách tạo truy vấn:

- Bộ truy vấn TPC-DS được lấy từ phiên bản TPC-DS v2.10.0, một tiêu chuẩn benchmark dành cho các hệ thống kho dữ liệu (Data Warehousing).
- Các truy vấn được sinh ra từ bộ công cụ TPC-DS kit, bao gồm 99 truy vấn chuẩn.
- Các truy vấn này được thiết kế để hoạt động trên tập dữ liệu được tổ chức theo mô hình ngôi sao (Star Schema) hoặc bông tuyết (Snowflake Schema), với các bảng sự kiện (Fact Tables) như store_sales, web_sales, và các bảng chiều (Dimension Tables) như customer, item, date_dim.

Mục đích:

- Đánh giá hiệu suất của hệ thống cơ sở dữ liệu trong các tác vụ phân tích dữ liệu phức tạp.
- Kiểm tra khả năng xử lý các truy vấn đa chiều, phân tích doanh số bán hàng, và dự báo tồn kho.
- Đánh giá khả năng mở rộng của hệ thống khi kích thước dữ liệu tăng lên.

Đặc điểm:

- Số lượng truy vấn: 99 truy vấn chuẩn.
- Loại tác vụ:
  - Phân tích doanh số bán hàng qua các kênh khác nhau (cửa hàng, trực tuyến, danh mục).
  - Phân tích khuyến mãi và hành vi khách hàng.
  - Dự báo và tối ưu hóa tồn kho.
- Tính phức tạp: Các truy vấn có độ phức tạp cao, với nhiều phép nối, tổng hợp, và điều kiện lọc.

Ví dụ truy vấn: Dưới đây là câu truy vấn 1 được tạo ra từ bộ dữ liệu 1GB

```
WITH customer_total_return AS (
    SELECT
        sr_customer_sk AS ctr_customer_sk,
        sr_store_sk AS ctr_store_sk,
        SUM(sr_fee) AS ctr_total_return
    FROM
        store_returns,
        date_dim
    WHERE
        sr_returned_date_sk = d_date_sk
        AND d_year = 2000
    GROUP BY
        sr_customer_sk,
```

```
        sr_store_sk
)

SELECT
    c_customer_id
FROM
    customer_total_return ctr1,
    store,
    customer
WHERE
    ctr1.ctr_total_return > (
        SELECT
            AVG(ctr_total_return) * 1.2
        FROM
            customer_total_return ctr2
        WHERE
            ctr1.ctr_store_sk = ctr2.ctr_store_sk
    )
    AND s_store_sk = ctr1.ctr_store_sk
    AND s_state = 'TN'
    AND ctr1.ctr_customer_sk = c_customer_sk
ORDER BY
    c_customer_id
LIMIT 100;
```

1.5. Các giả thuyết nghiên cứu

Hiệu suất của GPU vượt trội hơn CPU trong các bài toán xử lý dữ liệu lớn: GPU có khả năng xử lý song song mạnh mẽ, giúp giảm thời gian thực thi các truy vấn phức tạp.

Chi phí vận hành của GPU cao hơn CPU, nhưng hiệu quả hơn trong dài hạn: GPU tiêu thụ nhiều tài nguyên hơn, nhưng thời gian thực thi ngắn hơn giúp giảm tổng chi phí trong các bài toán lớn.

DuckDB phù hợp cho các bài toán nhỏ và trung bình, trong khi HeavyDB phù hợp cho các bài toán lớn: DuckDB được tối ưu cho CPU và dữ liệu vừa và nhỏ, trong khi HeavyDB tận dụng GPU để xử lý dữ liệu lớn.

1.6. Thang đo đánh giá

Thang đo đánh giá được sử dụng để so sánh hiệu suất giữa DuckDB (CPU) và HeavyDB (GPU) trong các tác vụ phân tích dữ liệu. Các tiêu chí đánh giá được lựa chọn nhằm đảm bảo tính khách quan và toàn diện, bao gồm:

1.6.1. Thời gian thực thi (Execution Time)

- **Mục tiêu**: Đánh giá tốc độ xử lý của hệ thống khi thực thi các truy vấn.
- **Phương pháp đo lường**:
  - Đo thời gian thực thi từng truy vấn từ lúc bắt đầu đến khi hoàn thành.
  - Sử dụng công cụ tích hợp trong DuckDB và HeavyDB để ghi nhận thời gian thực thi.
- **Đơn vị đo**: Milliseconds (ms).
- **Ý nghĩa**:
  - Thời gian thực thi càng thấp, hiệu suất xử lý càng cao.

1.6.2. Mức tiêu thụ tài nguyên (Resource Utilization)

- **Mục tiêu**: Đánh giá mức sử dụng tài nguyên hệ thống (CPU, GPU, RAM) trong quá trình thực thi truy vấn.
- **Phương pháp đo lường**:
  - Sử dụng công cụ giám sát như `nvidia-smi` để theo dõi mức sử dụng GPU.
- **Đơn vị đo**:
  - CPU: % sử dụng.
  - GPU: % sử dụng và dung lượng bộ nhớ GPU (VRAM) sử dụng (MB).
  - RAM: Dung lượng bộ nhớ sử dụng (MB).
- **Ý nghĩa**:
  - Hệ thống sử dụng tài nguyên hiệu quả hơn sẽ có mức tiêu thụ thấp hơn trong khi vẫn đảm bảo thời gian thực thi tốt.

1.6.3. Khả năng mở rộng (Scalability)

- **Mục tiêu**: Đánh giá khả năng của hệ thống khi kích thước dữ liệu tăng lên.
- **Phương pháp đo lường**:
  - Tạo dữ liệu thử nghiệm với các scale factor khác nhau (1GB, 2GB, 5GB, 10GB, 20GB).
  - Đo thời gian thực thi và mức tiêu thụ tài nguyên tương ứng với từng kích thước dữ liệu.
- **Ý nghĩa**:
  - Hệ thống có khả năng mở rộng tốt sẽ duy trì hiệu suất ổn định khi kích thước dữ liệu tăng lên.

1.6.4. Độ ổn định (Stability)

- **Mục tiêu**: Đánh giá khả năng của hệ thống trong việc xử lý các truy vấn liên tục mà không gặp lỗi hoặc giảm hiệu suất.
- **Phương pháp đo lường**:
  - Chạy lặp lại các truy vấn nhiều lần trên cùng một bộ dữ liệu.
  - Ghi nhận các lỗi xảy ra (nếu có) và sự thay đổi về thời gian thực thi giữa các lần chạy.
- **Ý nghĩa**:
  - Hệ thống ổn định sẽ có thời gian thực thi nhất quán và không gặp lỗi trong quá trình thử nghiệm.

1.6.5. Độ phức tạp của truy vấn (Query Complexity)

- **Mục tiêu**: Đánh giá khả năng xử lý các truy vấn có độ phức tạp khác nhau.
- **Phương pháp đo lường**:
  - Phân loại các truy vấn theo độ phức tạp (thấp, trung bình, cao) dựa trên số lượng phép nối, tổng hợp, và điều kiện lọc.
  - Đo thời gian thực thi và mức tiêu thụ tài nguyên cho từng loại truy vấn.
- **Ý nghĩa**:
  - Hệ thống hiệu quả sẽ xử lý tốt cả các truy vấn phức tạp mà không làm tăng đáng kể thời gian thực thi hoặc mức tiêu thụ tài nguyên.

1.6.6. Hiệu suất tổng thể (Overall Performance)

- **Mục tiêu**: Đánh giá hiệu suất tổng thể của hệ thống dựa trên các tiêu chí đã nêu.
- **Phương pháp đo lường**:
  - Tổng hợp kết quả từ các tiêu chí trên (thời gian thực thi, mức tiêu thụ tài nguyên, khả năng mở rộng, độ ổn định, độ phức tạp của truy vấn).
  - Sử dụng các biểu đồ và bảng để so sánh hiệu suất giữa DuckDB và HeavyDB.
- **Ý nghĩa**:

- Hiệu suất tổng thể cao cho thấy hệ thống có khả năng xử lý dữ liệu tốt, ổn định, và hiệu quả.

1.7. Kết luận chương

Chương 3 đã trình bày phương pháp nghiên cứu, lý do lựa chọn công cụ và cấu hình thử nghiệm, cũng như các giả thuyết nghiên cứu. Các phân tích về chi phí và hiệu suất giữa CPU và GPU cung cấp cơ sở để đánh giá hiệu quả của hai giải pháp này trong các bài toán phân tích dữ liệu.