Bài tập lớn 2: ArangoDB

Nguyễn Khánh Văn
IS211.O11.HTCL
Trường Đại học Công nghệ Thông tin
21522781@gm.uit.edu.vn

Lê Thị Kiều Lam
IS211.O11.HTCL
Trường Đại học Công nghệ Thông tin
21522275@gm.uit.edu.vn

Đặng Quang Nhật
IS211.O11.HTCL
Trường Đại học Công nghệ Thông tin
21522413@gm.uit.edu.vn

Hoàng Quốc Việt
IS211.O11.HTCL
Trường Đại học Công nghệ Thông tin
21522790@gm.uit.edu.vn

Tóm tắt — Trong bài tập lớn 2 này, nhóm chúng tôi sẽ tiến hành thiết kế một cơ sở dữ liệu phân tán bằng ngôn ngữ NoSQL ArangoDB. Sau đó sẽ tiến thành thực hiện truy vấn trên môi trường phân tán.

Từ khóa — Cơ sở dữ liệu phân tán, ArangoDB, Thiết kế cơ sở dữ liệu phân tán

I. GIỚI THIỆU

Hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) đã xuất hiện vào những năm 1970 và sử dụng ngôn ngữ truy vấn SQL và mô hình quan hệ để lưu trữ và quản lý dữ liệu. Trong quá trình phát triển công nghệ, đặc biệt là với sự phổ biến của web, khối lượng dữ liệu và thông tin mà các hệ thống phải xử lý đã tăng lên đáng kể. Các công nghệ truyền thống như SQL đã gặp một số hạn chế, bao gồm tính linh hoạt kém và khả năng mở rộng hạn chế. Để đáp ứng những yêu cầu mới trong lưu trữ và xử lý dữ liệu, hệ thống NoSQL đã xuất hiện.

NoSQL là một dạng hệ quản trị cơ sở dữ liệu phi quan hệ không sử dụng mô hình dữ liệu quan hệ và SQL. NoSQL hỗ trợ tốc độ và khả năng mở rộng cao hơn, cho phép xử lý lượng dữ liệu lớn và mở rộng ngang dễ dàng. Các hệ thống NoSQL thường có khả năng lưu trữ dữ liệu dưới nhiều hình thức khác nhau như key-value, document, column, và graph. Tuy nhiên, tính nhất quán dữ liệu có thể bị giảm xuống để đạt được hiệu suất và khả năng mở rộng cao hơn.

SQL và NoSQL có vai trò hỗ trợ và bổ sung cho nhau trong các dự án. SQL thường được sử dụng trong các ứng dụng có cấu trúc rõ ràng và yêu cầu tính nhất quán mạnh mẽ, trong khi NoSQL thích hợp cho các ứng dụng có yêu cầu về tốc độ và khả năng mở rộng cao hơn, và cho các dạng dữ liệu không cấu trúc hoặc có cấu trúc không rõ ràng. Sự lựa chọn giữa SQL và NoSQL phụ thuộc vào yêu cầu và tính chất của dự án cụ thể.

II. CƠ SỞ DỮ LIỆU NOSQL

A. Tổng quan

1) Cơ sở dữ liệu NoSQL là gì?

- Cơ sở dữ liệu NoSQL là hệ thống quản lý dữ liệu phi quan hệ được xây dựng dành riêng cho mô hình dữ liệu và có schema (lược đồ) rất linh hoạt.
- Việc xây dựng cơ sở dữ liệu phi quan hệ có mục đích dành cho các kho dữ liệu phân tán nhằm đáp ứng nhu cầu lưu trữ dữ liệu lớn.
- Cơ sở dữ liệu NoSQL được ứng dụng vào các web có thời gian giống với thời gian thực và có nguồn dữ liệu lớn thu thập hàng ngày như Google hay Facebook.

2) Tại sao nên sử dụng NoSQL?

- Nosql có tính linh hoạt cao: Cơ sở dữ liệu này sẽ cung cấp các sơ đồ linh hoạt giúp phát triển các công đoạn nhanh hơn cùng khả năng lặp lại.
- NoSQL có khả năng thay đổi quy mô: NoSQL được thiết kế nhằm tăng quy mô bằng các cụm phần cứng được phân phối thay vì bổ sung máy chủ mạnh và tốn kém.
- Hiệu năng của NoSQL cao: Cơ sở dữ liệu phi quan hệ sẽ được tối ưu hoá từng mô hình cụ thể và mẫu truy cập. Điều này sẽ giúp tăng hiệu năng so với việc đạt chức năng tương tự bằng cơ sở dữ liệu quan hệ.
- Tính thiết thực cao: Cơ sở dữ liệu phi quan hệ có cung cấp các kiểu dữ liệu thiết thực như API được xây dựng cho từng mô hình dữ liệu riêng.

3) Những dạng lưu trữ của NoSQL

- Key - Value: dữ liệu lưu trữ kiểu này chúng ta sẽ dựa vào key để lấy value. Dữ liệu dạng này có tốc độ truy vấn nhanh và đặc biệt Key - Value được sử dụng để làm cache cho dữ liệu (ví dụ như Redis).
- Document: Mỗi object được lưu trữ trong cơ sở dữ liệu dưới dạng document. Dữ liệu sẽ được lưu trữ dưới dạng BSON/JSON/XML dưới database. Với dạng này chúng ta có thể dễ dàng thêm, sửa, xóa trường một cách linh hoạt vì vậy nó khắc phục được cấu trúc cứng nhắc của Schema.
- Column - Family: Dữ liệu được lưu trữ dạng cột khác với SQL là dạng hàng, mỗi hàng có key/id riêng. Đặc biệt mỗi hàng trong một bảng lại có số lượng cột khác nhau. Với dạng lưu trữ này sẽ thích hợp cho việc ghi một số lượng lớn dữ liệu.
- Graph: Đây là kiểu cơ sở dữ liệu đồ thị, dữ liệu sẽ được lưu trữ theo từng node. Node chính là các thực thể hoặc là đối tượng. Properties là các thuộc tính liên quan đến từng Node, nó sẽ được đặt tên sao cho có quan hệ gần gũi với Node nhất. Edges là cạnh nối các Node, nó biểu thị cho quan hệ giữa các Node. Với dạng lưu trữ này thường được sử dụng trong các mạng noron, mạng xã hội.... Ưu điểm của nó là sử dụng các thuật toán duyệt node để tăng tốc độ truy vấn.

B. Đặc điểm

- NoSQL (Not Only SQL) là một hệ thống quản lý cơ sở dữ liệu phi quan hệ không sử dụng mô hình quan hệ và ngôn ngữ truy vấn SQL. Dưới đây là một số đặc điểm chính của hệ thống NoSQL:

  + Mô hình dữ liệu linh hoạt: Cho phép lưu trữ dữ liệu dưới nhiều dạng khác nhau như key-value, document, column, graph, và wide-column. Điều này cho phép các ứng dụng lưu trữ dữ liệu có cấu trúc không rõ ràng hoặc có cấu trúc linh hoạt mà không cần tuân theo một lược đồ cố định.

+ Khả năng mở rộng ngang: Được thiết kế để có khả năng mở rộng ngang dễ dàng. Điều này có nghĩa là có thể thêm máy chủ mới vào hệ thống để xử lý tải lớn và tăng khả năng chịu tải mà không làm giảm hiệu suất.
+ Tốc độ và hiệu suất: Thường được tối ưu hóa để đạt hiệu suất cao và xử lý tốc độ nhanh. Do không có mô hình quan hệ phức tạp và các ràng buộc ACID (Atomicity, Consistency, Isolation, Durability), NoSQL có thể đạt được tốc độ truy vấn và ghi dữ liệu cao hơn so với hệ thống SQL truyền thống.
+ Khả năng xử lý dữ liệu lớn: Thích hợp cho việc xử lý và lưu trữ lượng dữ liệu lớn, bao gồm cả dữ liệu có cấu trúc và không cấu trúc. Các hệ thống NoSQL thường có khả năng phân tán dữ liệu trên nhiều máy chủ và cho phép xử lý song song để đảm bảo khả năng xử lý dữ liệu hàng tỷ hoặc hàng triệu giao dịch mỗi giây.
+ Tính sẵn sàng và khả năng chịu lỗi: Thường được thiết kế để có tính sẵn sàng cao và khả năng chịu lỗi. Một số hệ thống NoSQL hỗ trợ sao lưu và khôi phục dữ liệu tự động, và có khả năng tự điều chỉnh để xử lý các trường hợp lỗi và sự cố mạng.

C. *Phân loại*

Cơ sở dữ liệu NoSQL được phân loại thành 5 loại chính sau:

- Cơ sở dữ liệu Key-Value (Key-Value Stores): Đây là dạng cơ sở dữ liệu đơn giản nhất trong hệ thống NoSQL. Dữ liệu được lưu trữ dưới dạng cặp khóa-giá trị, trong đó giá trị có thể là bất kỳ đối tượng nào. Ví dụ: Riak, DynamoDB
- Cơ sở dữ liệu Document (Document Stores): Cơ sở dữ liệu này lưu trữ dữ liệu dưới dạng tài liệu JSON hoặc XML. Các tài liệu có thể được tổ chức thành các tập hợp (collections) và được truy xuất bằng cách sử dụng các truy vấn dựa trên nội dung của tài liệu. Ví dụ: MongoDB, Couchbase.
- Cơ sở dữ liệu Column Family (Wide-Column Stores): Loại cơ sở dữ liệu này sử dụng mô hình cột họ (column family) để tổ chức dữ liệu. Các cột họ là nhóm các cột liên quan chứa các giá trị tương tự. Điều này cho phép lưu trữ và truy xuất hiệu quả cho các tập dữ liệu lớn và có cấu trúc biến đổi. Ví dụ: Cassandra, HBase.
- Cơ sở dữ liệu Đồ thị (Graph Databases): Loại cơ sở dữ liệu này tập trung vào mô hình dữ liệu đồ thị, trong đó các đối tượng được biểu diễn bằng các đỉnh (nodes) và các mối quan hệ giữa chúng được biểu diễn bằng các cạnh (edges). Đây là lựa chọn tốt cho các ứng dụng có yêu cầu phân tích mạng lưới phức tạp. Ví dụ: Neo4j, OrientDB.
- Cơ sở dữ liệu Time Series (Time Series Databases): Loại cơ sở dữ liệu này được tối ưu hóa cho việc lưu trữ và truy xuất dữ liệu chuỗi thời gian, như dữ liệu cảm biến, thông tin thống kê, dữ liệu giao dịch v.v. Các tính năng như nén dữ liệu và truy vấn dựa trên thời gian là những điểm mạnh của loại cơ sở dữ liệu này. Ví dụ: InfluxDB, Prometheus.

Mỗi loại cơ sở dữ liệu NoSQL có ưu điểm và hạn chế riêng, và lựa chọn phụ thuộc vào yêu cầu cụ thể của dự án hoặc ứng dụng.

III. GIỚI THIỆU VỀ ARANGODB

A. *Tổng quan*

ArangoDB là một hệ quản trị cơ sở dữ liệu đa mô hình (multi-model database) hỗ trợ ba mô hình dữ liệu: tài liệu, đồ thị và khóa-giá trị. Được phát triển bởi ArangoDB Inc. Điều này khiến nó trở thành một công cụ linh hoạt để xây dựng các ứng dụng yêu cầu mối quan hệ dữ liệu phức tạp và hiệu suất truy vấn nhanh.

ArangoDB đã được giới thiệu như là 1 universal database nhưng những người tạo ra thì gọi nó là cơ sở dữ liệu "native multi-model" để chỉ ra rằng nó được đặc biệt thiết kế để cho phép các kiểu dữ liệu key/value, document và graph được lưu trữ cùng nhau và được truy vấn bằng cùng 1 ngôn ngữ. Xây dựng các ứng dụng hiệu suất cao bằng cách sử dụng ngôn ngữ truy vấn giống SQL thuận tiện hoặc các tiện ích mở rộng JavaScript. Sử dụng các giao dịch ACID nếu bạn yêu cầu. Chia tỷ lệ theo chiều ngang và chiều dọc với một vài cú nhấp chuột.

B. *Tính năng*

Một số tính năng chính của ArangoDB:

1. **Multi-Model**:
    - ***Document***: Được sử dụng khi dữ liệu có cấu trúc linh hoạt, có thể được biểu diễn dưới dạng các tài liệu JSON hoặc BSON. Đây là mô hình thích hợp cho các ứng dụng có yêu cầu linh hoạt về cấu trúc dữ liệu.
    - ***Graph***: Được sử dụng khi dữ liệu có mối quan hệ phức tạp và được biểu diễn dưới dạng đồ thị, với các đỉnh và cạnh. Mô hình đồ thị thích hợp cho các ứng dụng liên quan đến mạng xã hội, định tuyến, phân tích mạng, và nhiều hơn nữa.
    - ***Key-Value***: Được sử dụng khi dữ liệu được lưu trữ dưới dạng các cặp khóa-giá trị. Mô hình này thích hợp cho các ứng dụng yêu cầu truy cập dữ liệu nhanh chóng bằng cách sử dụng khóa.
2. **Joins**: ArangoDB hỗ trợ các phép nối (join) giữa các bộ sưu tập (collection) khác nhau. Bằng cách sử dụng AQL (ArangoDB Query Language), bạn có thể thực hiện các truy vấn linh hoạt và kết hợp thông tin từ nhiều bộ sưu tập trong một truy vấn duy nhất. Điều này giúp truy vấn dữ liệu một cách thuận tiện và linh hoạt, đồng thời giảm thiểu dư thừa dữ liệu.
3. **Transactions**: ArangoDB hỗ trợ giao dịch (transactions) để đảm bảo tính nhất quán và an toàn của dữ liệu. Bạn có thể thực hiện các hoạt động ghi dữ liệu (insert, update, delete) trong một giao dịch và đảm bảo rằng các thay đổi chỉ được áp dụng nếu tất cả các hoạt động trong giao dịch thành công. Điều

này giúp đơn giản hóa việc quản lý dữ liệu và đảm bảo tính toàn vẹn của dữ liệu.

4. **Client Convenience**: ArangoDB cung cấp các thư viện và API dễ sử dụng để phát triển ứng dụng. Các thao tác truy vấn và thao tác dữ liệu được thực hiện thông qua các giao diện dễ hiểu và linh hoạt. Điều này giúp đơn giản hóa quá trình phát triển ứng dụng và tương tác với cơ sở dữ liệu từ phía client.

Dưới đây là một câu truy vấn AQL ví dụ về các tính năng trên:

```
FOR event IN events
FILTER 'Python' IN event.topics
FOR location IN OUTBOUND event takes_place
FOR attendee IN INBOUND location attends
FILTER attendee.notify
FOR cname IN location_names
FILTER cname.location == location._key AND
cname.lang == attendee.lang
INSERT { email: attendee.email, event: event._key,
location: cname.name }
INTO notifications
```

C. Transaction concepts

Transaction concepts (khái niệm giao dịch) là các khái niệm cơ bản được sử dụng trong lĩnh vực cơ sở dữ liệu để đảm bảo tính toàn vẹn và nhất quán của các thao tác dữ liệu. Một giao dịch (transaction) là một tập hợp các thao tác dữ liệu được thực hiện như một đơn vị không thể chia nhỏ. Các khái niệm quan trọng trong giao dịch bao gồm:

- Atomicity (Nguyên tử): Atomicity đảm bảo rằng một giao dịch được xem như một hoạt động duy nhất và không thể chia nhỏ. Nếu một phần của giao dịch không thể hoàn thành, toàn bộ giao dịch sẽ bị hủy bỏ và dữ liệu sẽ được khôi phục về trạng thái ban đầu (rollback), không để lại trạng thái trung gian không nhất quán.
- Consistency (Nhất quán): Consistency đảm bảo rằng một giao dịch khi hoàn thành sẽ đưa cơ sở dữ liệu từ một trạng thái hợp lệ (consistency) sang một trạng thái hợp lệ khác. Điều này đảm bảo rằng các ràng buộc và quy tắc của cơ sở dữ liệu không bị vi phạm sau khi giao dịch hoàn thành.
- Isolation (Cô lập): Isolation đảm bảo rằng các giao dịch được thực hiện độc lập với nhau, mỗi giao dịch không thể ảnh hưởng đến kết quả của các giao dịch khác. Các giao dịch được thực hiện song song mà không gây ra hiện tượng xung đột (conflict) hoặc đọc dữ liệu không nhất quán (inconsistent data).
- Durability (Bền vững): Durability đảm bảo rằng sau khi một giao dịch hoàn thành, kết quả của nó sẽ được lưu trữ một cách bền vững và không bị mất đi trong trường hợp xảy ra sự cố như mất điện hoặc hỏng hóc phần cứng. Dữ liệu đã được lưu trữ sẽ tồn tại và có sẵn cho các giao dịch và truy xuất sau này.

D. Mô hình lưu trữ

Trong ArangoDB, dữ liệu được lưu trữ dưới dạng JSON và được gọi là "document". Các document có thể có cấu trúc khác nhau và vẫn có thể được lưu trữ trong cùng một collection. Collection là một nhóm các document liên quan đến nhau.

Trong ArangoDB, không cần phải xác định các lược đồ (schema) trước như trong hệ quản trị cơ sở dữ liệu quan hệ (RDBMS). Điều này cho phép linh hoạt trong việc lưu trữ dữ liệu, vì các document có thể có cấu trúc khác nhau trong cùng một collection.

Tuy nhiên, vẫn có một cấu trúc dữ liệu nhất định để dễ dàng thao tác. Các document trong một collection thường chia sẻ các trường chung, ví dụ như các trường "name" hoặc "age", để dễ dàng truy cập và truy vấn dữ liệu.

ArangoDB cũng hỗ trợ hai dạng collection là document collection và edge collections. Document collection là các collection thông thường, trong đó mỗi document đại diện cho một đối tượng độc lập. Edge collections là các collection đặc biệt được sử dụng để tạo quan hệ giữa các document. Các document trong edge collections thường có hai thuộc tính đặc biệt là "_from" và "_to", để chỉ định quan hệ giữa các document trong collection. Thông qua các thuộc tính này, ta có thể tạo các mối quan hệ như "parent-child" hoặc "follows" giữa các document.

E. Ngôn ngữ truy vấn AQL

AQL (ArangoDB Query Language) là một ngôn ngữ truy vấn giống SQL được sử dụng trong ArangoDB. Nó hỗ trợ các hoạt động CRUD (Create, Read, Update, Delete) cho cả document (node) và edge, tương tự như SQL trong hệ quản trị cơ sở dữ liệu quan hệ.

Tuy nhiên, *AQL không phải là ngôn ngữ định nghĩa dữ liệu (DDL)*. Điều này có nghĩa là AQL không hỗ trợ các câu lệnh để tạo, sửa đổi hoặc xóa cấu trúc cơ sở dữ liệu, bao gồm các bảng, cột và ràng buộc. Các tác vụ như tạo collection, index hoặc graph schema được thực hiện bên ngoài AQL, thông qua API hoặc giao diện quản trị.

AQL cũng không hỗ trợ truy vấn không gian địa lý (spatial query), tức là truy vấn dựa trên vị trí địa lý của dữ liệu. Để thực hiện các truy vấn không gian địa lý, ArangoDB cung cấp các API và chức năng đặc biệt khác.

Cú pháp của AQL được trình bày theo hướng JSON, với việc sử dụng dấu chấm (.) để truy cập các giá trị của các khóa (keys). Điều này cho phép ngôn ngữ truy vấn dễ đọc và dễ hiểu, tương tự như cú pháp của JSON.

Ví dụ:

```
FOR doc IN collection
  FILTER doc.name == 'John'
  RETURN doc.age
```

F. Cơ chế phân tán

1) Master/Slave

Để thiết lập một bản sao Master/Slave trong ArangoDB, bạn cần ít nhất hai máy chạy cơ sở dữ liệu ArangoDB:

- Master: Đây là máy chủ (server) chịu trách nhiệm cho tất cả các hoạt động sửa đổi dữ liệu. Trên master, bạn có thể thực hiện các thao tác CRUD (Create, Read, Update, Delete) và các hoạt động khác liên quan đến dữ liệu.
- Slave: Đây là máy sao chép dữ liệu từ master. Dữ liệu được sao chép từ master sang slave theo cách không đồng bộ. Để thực hiện sao chép, bạn cần khởi động một trình nhân bản (replication applier) trên slave. Trình nhân bản sẽ lấy dữ liệu từ master và áp

dụng các hoạt động của nó trên slave, giúp đảm bảo rằng dữ liệu trên slave được cập nhật theo dữ liệu trên master.

a) Thành phần

Mô hình Master/Slave của ArangoDB, có hai thành phần quan trọng liên quan đến sao chép và đồng bộ dữ liệu: *Replication Logger* (ghi nhật ký sao chép) và *Replication Applier* (áp dụng sao chép).

- **Replication Logger**: Replication Logger ghi tất cả các hoạt động sửa đổi dữ liệu trên master vào một nhật ký ghi trước (log). Nhật ký này chứa thông tin về các sự kiện thay đổi dữ liệu, bao gồm tạo, cập nhật và xóa. Nhật ký ghi trước này có thể được clients đọc để chạy lại các sửa đổi dữ liệu trên một máy chủ khác, ví dụ như slave.
- **Replication Applier**: Replication Applier đọc dữ liệu từ nhật ký ghi trước (log) của master và áp dụng chúng một cách cục bộ trên slave. Applier kiểm tra định kỳ cơ sở dữ liệu master để xem có sự kiện sửa đổi mới nào. Nó chỉ yêu cầu master cung cấp các hoạt động xảy ra sau lần đồng bộ hóa trước đó (đồng bộ hóa gia tăng). Cơ sở dữ liệu master không thông báo trực tiếp cho Replication Applier về các hoạt động mới, mà thay vào đó sử dụng nguyên tắc đẩy. Do đó, có thể có một khoảng thời gian trễ (được gọi là độ trễ sao chép) trước khi các thao tác từ master được chuyển đến và áp dụng trên slave.

b) Hạn chế

- **Độ trễ sao chép**: Có thể có một độ trễ giữa dữ liệu trên master và dữ liệu trên slave. Do quá trình sao chép dữ liệu không đồng bộ, dữ liệu trên slave có thể không được cập nhật ngay lập tức sau khi có sự thay đổi trên master. Điều này có thể gây ra sự không nhất quán giữa dữ liệu trên master và slave trong một khoảng thời gian.
- **Độ trễ đồng bộ hóa**: Khi Replication Applier không thể kết nối hoặc giao tiếp với master, có thể xảy ra độ trễ trong quá trình đồng bộ hóa. Nếu có sự cố xảy ra trên master và Replication Applier không thể kết nối lại thành công sau một số lượng lần thử kết nối, dữ liệu trên slave sẽ không còn được cập nhật và không đồng bộ với master.
- **Single point of failure**: Master là điểm trung tâm có trách nhiệm cho toàn bộ hoạt động sửa đổi dữ liệu. Nếu master gặp sự cố hoặc không khả dụng, toàn bộ hệ thống sẽ bị ảnh hưởng. Điều này làm cho mô hình này có một điểm đơn lẻ có thể gây gián đoạn toàn bộ hệ thống.
- **Scale-out hạn chế**: Không đảm bảo khả năng mở rộng tuyến tính. Việc thêm các slave mới có thể giúp tăng khả năng chịu tải và đảm bảo sẵn sàng cao hơn, nhưng cấu trúc này không cho phép việc ghi dữ liệu đồng thời trên nhiều master. Điều này có thể làm hạn chế khả năng mở rộng và hiệu suất của hệ thống trong một số trường hợp.
- **Khả năng chỉ đọc trên slave**: Slave chỉ được phép đọc dữ liệu và không được phép thực hiện các hoạt động sửa đổi dữ liệu. Điều này có nghĩa là chỉ có master mới có thể thực hiện các thao tác CRUD. Nếu ứng dụng của bạn cần thực hiện các hoạt động sửa đổi dữ liệu trên slave, mô hình này có thể không phù hợp.

2) Master/Master

ArangoDB hỗ trợ mô hình Master/Master (Chủ/Chủ) cho phép nhiều máy chủ trong một Cluster hoạt động như các nút chính độc lập và có khả năng ghi (write) dữ liệu. Trong mô hình này, mỗi máy chủ được coi là một nút chủ (master node), và chúng có quyền ghi dữ liệu vào cơ sở dữ liệu.

a) Thành phần

Mô hình Master/Master của ArangoDB, có 3 thành phần chính đó là: *Múi chủ* (Coordinator), *Nút chủ* (Master Node), *Cơ chế đồng thuận* (Consensus Mechanism).

- **Múi chủ** *(Coordinator):* Múi chủ là thành phần chịu trách nhiệm quản lý và điều phối các hoạt động trong mô hình Master/Master. Nó nhận yêu cầu từ ứng dụng và quyết định nút chủ nào sẽ xử lý yêu cầu đó. Múi chủ cũng chịu trách nhiệm đồng bộ dữ liệu giữa các nút chủ và xử lý xung đột dữ liệu.
- **Nút chủ** *(Master Node):* Mỗi nút chủ là một máy chủ trong Cluster ArangoDB và có khả năng ghi dữ liệu. Mỗi nút chủ lưu trữ một bản sao của cơ sở dữ liệu và thực hiện các hoạt động ghi dữ liệu từ ứng dụng. Các nút chủ trong mô hình Master/Master có khả năng chủ động và đồng bộ dữ liệu với nhau.
- **Cơ chế đồng thuận** *(Consensus Mechanism):* ArangoDB sử dụng các thuật toán đồng thuận để đảm bảo tính nhất quán của dữ liệu giữa các nút chủ. Các thuật toán này giúp đồng bộ hóa các thay đổi dữ liệu và giải quyết xung đột dữ liệu khi có nhiều nút chủ cùng thay đổi dữ liệu.

b) Cluster

Khi việc sử dụng ArangoDB vượt quá khả năng phần cứng hiện có, chúng ta cần mở rộng quy mô hệ thống. Mặc dù có thể mở rộng theo chiều dọc bằng cách di chuyển ArangoDB sang một máy chủ mạnh hơn, nhưng mở rộng theo chiều ngang bằng cách tạo thành một Cluster từ nhiều máy chủ cơ sở dữ liệu có những lợi thế đáng kể.

Clustering ArangoDB không chỉ cải thiện hiệu suất và tăng dung lượng lưu trữ, mà còn cung cấp khả năng phục hồi thông qua sao chép tự động và chuyển đổi dự phòng. Điều này đảm bảo tính sẵn sàng cao và bảo đảm rằng dữ liệu vẫn sẽ có sẵn ngay cả khi xảy ra sự cố.

Hơn nữa, việc triển khai hệ thống tự động mở rộng quy mô lên và xuống theo nhu cầu là một lợi ích quan trọng của việc sử dụng clustering ArangoDB. Điều này cho phép hệ thống linh hoạt thích ứng với tải công việc biến đổi và tăng cường khả năng xử lý của nó khi cần thiết, đồng thời giảm thiểu việc sử dụng tài nguyên không cần thiết khi tải công việc giảm đi.

c) Kiến trúc

Theo lý thuyết CAP (Consistency, Availability, Partition tolerance), clusters trong ArangoDB sử dụng mô hình master/master với không điểm lỗi. ArangoDB ưu tiên tính nhất quán bên trong (consistency) hơn tính khả dụng (availability). Điều này có nghĩa là các clients có thể truy cập vào cơ sở dữ liệu giống nhau bất kể họ kết nối đến node nào

trong cluster, và nếu một máy chủ bị lỗi, cluster vẫn tiếp tục hoạt động.

Một ArangoDB cluster bao gồm các ArangoDB được liên kết với nhau trong cùng một mạng. Mỗi máy chủ trong cluster có vai trò khác nhau, bao gồm:

- **Agents**: Đây là Key/Value Store của một hoặc nhiều ArangoDB và đóng vai trò như một Agency để lưu trữ cấu hình của cluster. Nó thực hiện bầu cử lãnh đạo và cung cấp các tác vụ đồng bộ hóa khác. Mặc dù không được nhìn thấy từ bên ngoài, Agent là trung tâm của cụm. Để đảm bảo tính chịu lỗi tốt, các Agent sử dụng thuật toán Raft Consensus để đảm bảo không có xung đột trong cụm.
- **Coordinators**: Khi clients giao tiếp với một ArangoDB cluster, họ sẽ giao tiếp với các coordinators. Các coordinators này phối hợp các tác vụ và chạy dịch vụ Foxx. Chúng biết nơi lưu trữ dữ liệu và tối ưu hóa để thực hiện các truy vấn từ người dùng. Các coordinators không có trạng thái, vì vậy chúng có thể tắt và khởi động lại khi cần thiết.
- **Database Servers**: Các ArangoDB lưu trữ dữ liệu thực tế và là các server cơ sở dữ liệu. Mỗi server lưu trữ một mảnh dữ liệu và sao chép đồng bộ để hoạt động như một leader hoặc follower trên mỗi mảnh dữ liệu cụ thể. Sau đó, chúng thực hiện truy vấn một phần hoặc toàn bộ dữ liệu tuỳ thuộc vào yêu cầu từ người dùng.

d) Hạn chế

Mặc dù mô hình Master/Master trong ArangoDB có nhiều lợi ích, nhưng cũng có một số hạn chế cần được xem xét:

- **Xung đột ghi**: Khi có nhiều nút chủ cùng thay đổi dữ liệu, có thể xảy ra xung đột ghi. ArangoDB sử dụng các cơ chế đồng thuận để giải quyết xung đột này, nhưng vẫn có khả năng xảy ra xung đột không thể giải quyết hoặc gây mất tính nhất quán của dữ liệu.
- **Độ trễ đồng bộ**: Đồng bộ dữ liệu giữa các nút chủ trong mô hình Master/Master có thể tạo ra độ trễ. Khi một nút chủ thực hiện một thao tác ghi, dữ liệu phải được sao chép và đồng bộ hóa với các nút chủ khác trước khi yêu cầu được coi là hoàn thành. Điều này có thể gây trễ cho các hoạt động ghi và làm giảm hiệu suất hệ thống.
- **Phức tạp hóa**: Tạo ra sự phức tạp trong việc quản lý và triển khai hệ thống. Việc cấu hình và quản lý các nút chủ và cơ chế đồng thuận đòi hỏi kiến thức và kỹ năng chuyên sâu. Sự phức tạp này có thể làm tăng độ khó trong việc triển khai và bảo trì hệ thống.
- **Tốn kém**: Yêu cầu nhiều nút chủ hoạt động đồng thời và đồng bộ dữ liệu. Điều này có thể tạo ra tải công việc và tốn kém về tài nguyên, bao gồm bộ nhớ, CPU và băng thông mạng. Do đó, việc triển khai và duy trì một hệ thống Master/Master có thể đòi hỏi một nguồn lực đáng kể.
- **Khả năng mở rộng hạn chế**: Mô hình Master/Master trong ArangoDB có khả năng mở rộng, nhưng có giới hạn về số lượng nút chủ mà hệ thống có thể hỗ trợ. Khi số lượng nút chủ tăng, việc đồng bộ dữ liệu và quản lý xung đột trở nên phức tạp hơn và có thể gây ảnh hưởng đến hiệu suất hệ thống.

3) Sharded

Trong ArangoDB, sharding là một phương pháp phân tán dữ liệu thành các phần nhỏ hơn gọi là shard và phân phối chúng trên nhiều máy chủ. Sharding giúp tăng khả năng mở rộng của hệ thống, cải thiện hiệu suất và khả năng chịu lỗi.

a) Thành phần

Dưới đây là một số khái niệm và thành phần chính liên quan đến sharding trong ArangoDB:

- **Shard** *(phân đoạn):* Shard là một phần nhỏ của cơ sở dữ liệu ArangoDB. Mỗi shard là một tập hợp các tài liệu hoặc bộ sưu tập (collection) mà dữ liệu được chia thành. Mỗi shard được lưu trữ trên một máy chủ riêng biệt trong hệ thống.
- **Tập hợp** *(Collection):* Trong ArangoDB, dữ liệu được tổ chức thành các tập hợp, còn được gọi là bộ sưu tập. Mỗi tập hợp có thể được chia thành nhiều shard và được phân phối trên các máy chủ khác nhau.
- **Tạp chủ** *(Coordinator):* Tạp chủ là thành phần trong ArangoDB chịu trách nhiệm quản lý và điều phối các hoạt động liên quan đến sharding. Nó nhận yêu cầu từ ứng dụng và xác định shard tương ứng để thực hiện các hoạt động đọc và ghi dữ liệu.
- **Từ khóa shard** *(Shard key):* Shard key là một trường trong tài liệu được sử dụng để xác định shard mà tài liệu đó thuộc về. Khi tài liệu được thêm vào hoặc truy vấn, ArangoDB sẽ sử dụng giá trị của shard key để xác định shard tương ứng.
- **Hashing**: Trong quá trình sharding, ArangoDB sử dụng các thuật toán hashing để phân phối tài liệu vào các shard khác nhau. Các giá trị của shard key được chuyển đổi thành giá trị băm (hash value), từ đó xác định shard mà tài liệu thuộc về.
- **Replication** *(Sao chép)*: Trong sharding, ArangoDB hỗ trợ sao chép dữ liệu trên các shard khác nhau để đảm bảo tính sẵn sàng và độ tin cậy. Các bản sao của dữ liệu được lưu trữ trên các máy chủ khác nhau để đảm bảo rằng dữ liệu vẫn có sẵn trong trường hợp một máy chủ gặp sự cố.

b) Hạn chế

Mặc dù sharding trong ArangoDB mang lại nhiều lợi ích, nhưng cũng có một số hạn chế cần được xem xét:

- **Phân vùng dữ liệu**: Sharding trong ArangoDB yêu cầu chọn một shard key (hoặc nhiều shard key) để phân vùng dữ liệu. Việc lựa chọn shard key phải được thực hiện cẩn thận để đảm bảo sự phân phối đồng đều của dữ liệu và truy vấn hiệu quả. Nếu shard key được chọn không cân đối hoặc không phù hợp, có thể xảy ra hiện tượng hotspot (cháy nổ) dữ liệu trên một số shard hoặc truy vấn không hiệu quả.
- **Khả năng mở rộng**: Sharding trong ArangoDB cung cấp khả năng mở rộng dựa trên việc phân tán dữ liệu trên nhiều shard. Tuy nhiên, việc thêm shard mới và điều chỉnh cấu hình sharding có thể là một

quá trình phức tạp và đòi hỏi sự quản lý kỹ lưỡng. Khi hệ thống cần mở rộng, việc phân vùng lại dữ liệu và di chuyển tài liệu giữa các shard có thể tốn thời gian và tài nguyên.

- **Xử lý giao transaction** *(giao dịch):* Sharding trong ArangoDB đôi khi gặp khó khăn trong việc xử lý giao dịch (transaction). Vì dữ liệu trong một giao dịch có thể nằm trên nhiều shard khác nhau, việc đảm bảo tính nhất quán và đồng bộ giữa các shard có thể phức tạp. ArangoDB cung cấp các cơ chế để hỗ trợ giao dịch trên shard, nhưng việc thiết kế và triển khai giao dịch phải được xem xét kỹ lưỡng.
- **Quản lý metadata**: Trong sharding, ArangoDB phải duy trì thông tin về phân vùng dữ liệu và shard trên các tạp chủ (coordinator). Quản lý metadata có thể trở nên phức tạp khi hệ thống có nhiều shard và tài liệu. Việc duy trì và đồng bộ hóa metadata đòi hỏi tài nguyên và quản lý cẩn thận để đảm bảo tính nhất quán và hiệu suất của hệ thống.
- **Khả năng truy vấn phức tạp**: Khi dữ liệu được phân tán trên nhiều shard, việc thực hiện các truy vấn phức tạp có thể trở nên phức tạp hơn. Truy vấn cần truy cập dữ liệu từ nhiều shard và kết hợp các kết quả từ các shard khác nhau, điều này yêu cầu khả năng xử lý và tối ưu hóa truy vấn phân tán.

G. *Ưu điểm*

Một số lợi ích của việc sử dụng ArangoDB:

- Giảm thời gian phát triển: Khả năng của AQL để truy vấn tất cả ba mô hình dữ liệu trong một truy vấn duy nhất có thể giúp giảm thời gian phát triển bằng cách loại bỏ nhu cầu viết nhiều truy vấn cho các mô hình dữ liệu khác nhau.
- Tăng hiệu suất: Lưu trữ đồ thị gốc của ArangoDB và khả năng tối ưu hóa của AQL có thể dẫn đến hiệu suất truy vấn được cải thiện, đặc biệt là đối với các truy vấn dựa trên đồ thị.
- Tăng tính linh hoạt: Hỗ trợ nhiều mô hình dữ liệu của ArangoDB khiến nó trở thành một lựa chọn tốt cho các ứng dụng cần lưu trữ nhiều loại kiểu dữ liệu.
- Quản lý dữ liệu được đơn giản hóa: Ngôn ngữ truy vấn thống nhất và thiết kế không có lược đồ của ArangoDB có thể giúp đơn giản hóa việc quản lý dữ liệu.

H. *Nhược điểm*

ArangoDB vẫn có một số nhược điểm khi sử dụng:

- Học và sử dụng phức tạp: AQL, ngôn ngữ truy vấn của ArangoDB, là một ngôn ngữ khai báo mạnh mẽ nhưng cũng có thể khó hiểu. Ngoài ra, ArangoDB cung cấp một số tính năng nâng cao, chẳng hạn như các chỉ mục tùy chỉnh và các truy vấn phân tán, có thể khiến việc sử dụng trở nên phức tạp hơn.
- Không hiệu quả cho các ứng dụng có lượng truy vấn rất cao: ArangoDB được thiết kế để xử lý các truy vấn phức tạp, nhưng nó có thể không phải là lựa chọn tốt nhất cho các ứng dụng có khối lượng truy vấn cao, chẳng hạn như các ứng dụng thương mại điện tử.
- Có thể không phải là lựa chọn tốt nhất cho các ứng dụng cần hỗ trợ cho các mô hình dữ liệu cụ thể: ArangoDB cung cấp hỗ trợ cho một loạt các mô hình dữ liệu, nhưng nó không phải là lựa chọn tốt nhất cho tất cả các nhu cầu.

I. *Sử dụng*

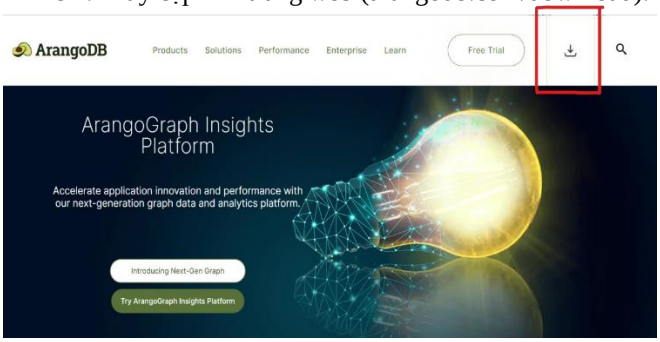
Dưới đây là một số ví dụ về cách ArangoDB có thể được sử dụng:

- Mạng xã hội: ArangoDB có thể được sử dụng để lưu trữ dữ liệu về người dùng, bạn bè, nhóm và bài đăng. Nó có thể được sử dụng để thực hiện các truy vấn như tìm kiếm người dùng, tìm tất cả bạn bè của một người dùng hoặc hiển thị danh sách bài đăng mới nhất.
- Phát hiện gian lận: ArangoDB có thể được sử dụng để lưu trữ dữ liệu về giao dịch, thẻ tín dụng và tài khoản. Nó có thể được sử dụng để thực hiện các truy vấn như xác định các giao dịch đáng ngờ, tìm kiếm các mẫu gian lận hoặc phát hiện các hành vi bất thường.
- Hệ thống đề xuất: ArangoDB có thể được sử dụng để lưu trữ dữ liệu về sản phẩm, người dùng và lượt mua. Nó có thể được sử dụng để thực hiện các truy vấn như đề xuất sản phẩm cho người dùng, tìm kiếm các sản phẩm tương tự hoặc xác định các sản phẩm phổ biến

IV. HƯỚNG DẪN CÀI ĐẶT

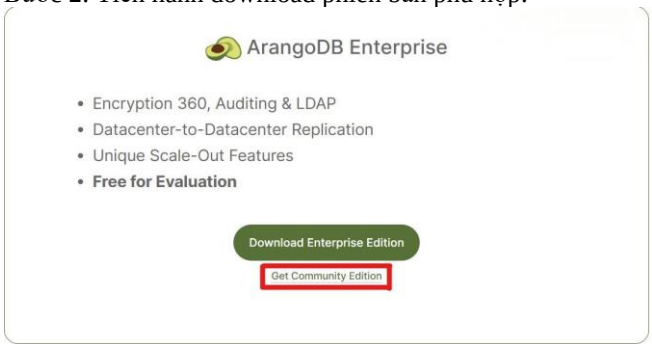
A. *Cài đặt trên một máy*

Bước 1: Truy cập đến trang web (arangodb.com/download).



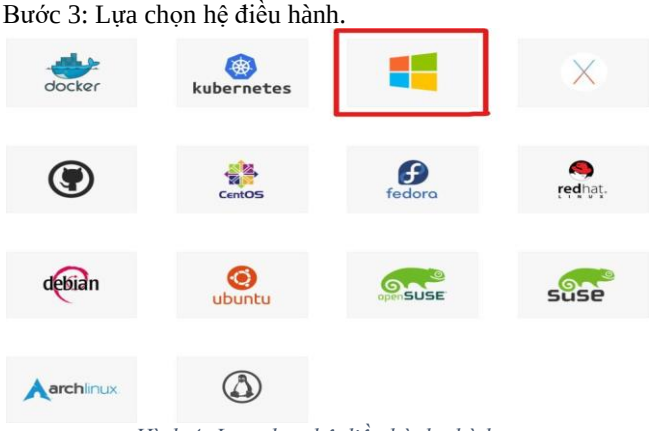
Hình 2: Giao diện khi truy cập trang web download ArangoDB

Bước 2: Tiến hành download phiên bản phù hợp.



Hình 3: Lựa chọn phiên bản download

Bước 3: Lựa chọn hệ điều hành.

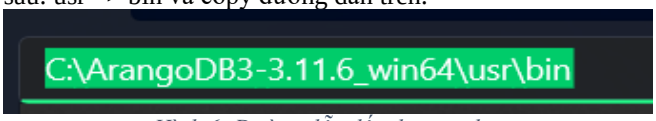


Hình 4: Lựa chọn hệ điều hành phù hợp

Bước 4: Chọn gói download.

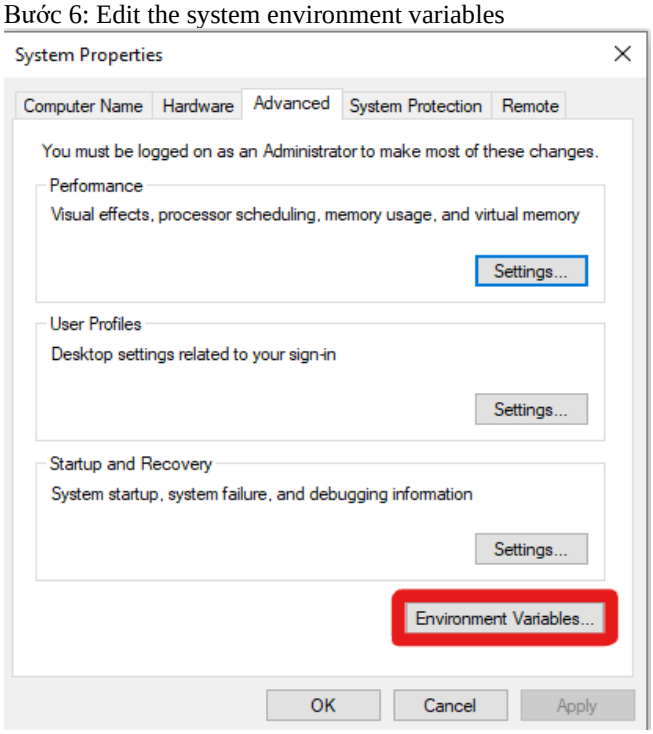
Hình 5: Lựa chọn gói download thích hợp

Bước 5: Giải nén file vừa down về, tiến hành vào đường dẫn sau: usr –> bin và copy đường dẫn trên.



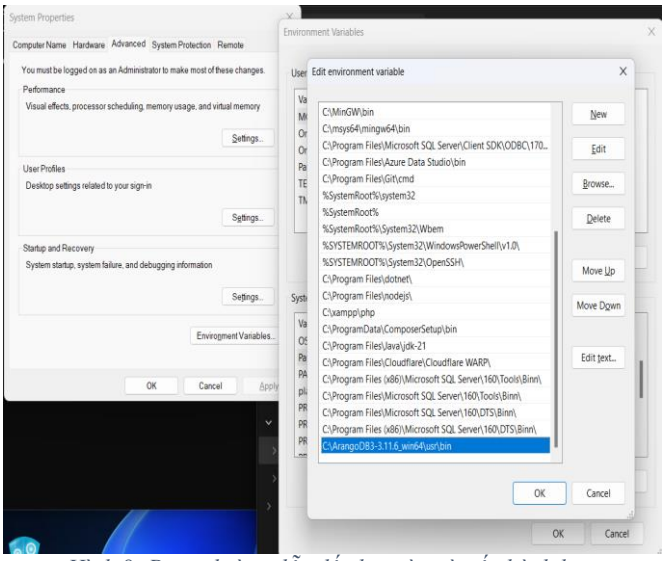
Hình 6: Đường dẫn đến thư mục bin

Bước 6: Edit the system environment variables



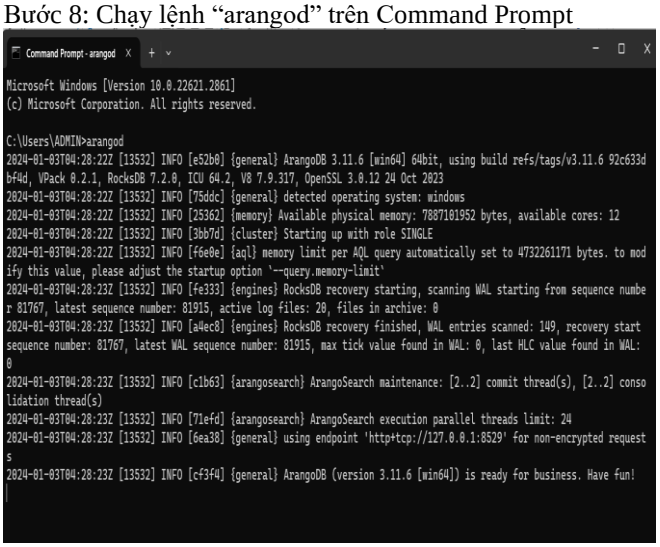
Hình 7: Hộp thoại Edit the system enviroment variables

Bước 7: Paste đường dẫn vào biến môi trường và lưu lại



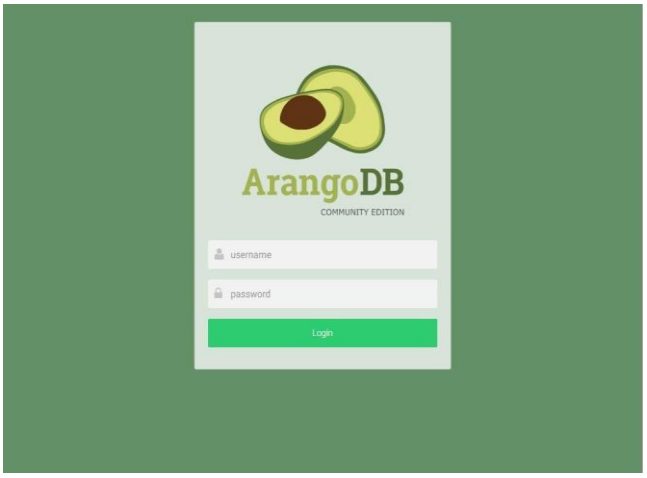
Hình 8: Paste đường dẫn đến bin vào và tiến hành lưu

Bước 8: Chạy lệnh "arangod" trên Command Prompt



Hình 9: Chạy lệnh trong Command Prompt

Bước 9: Mở giao diện đăng nhập ArangoDB bằng localhost 127.0.0.1:8529

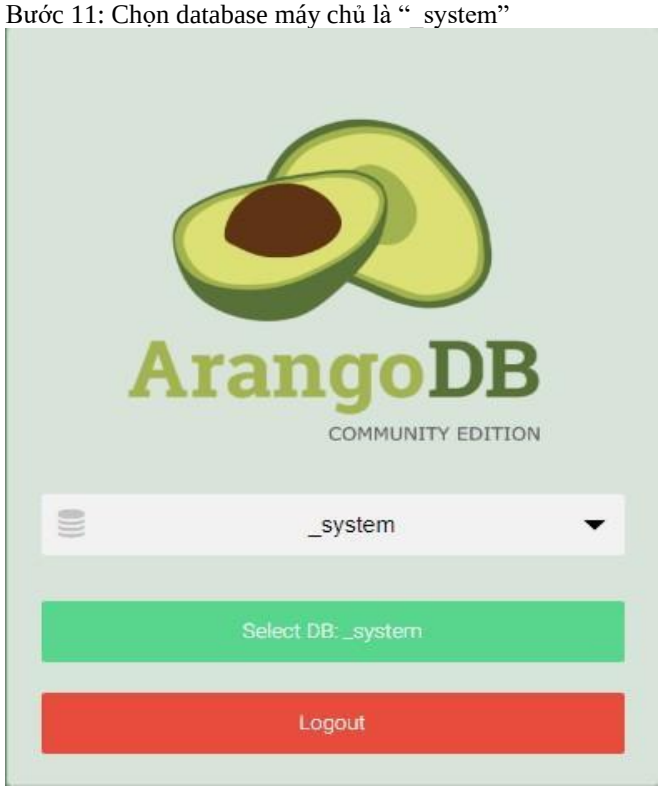


Hình 10: Giao diện trang đăng nhập ArangoDB

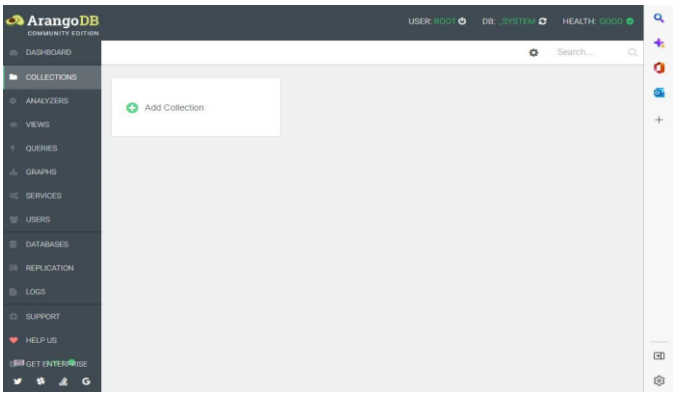
Bước 10: Đăng nhập bằng tài khoản root

Hình 11: Giao diện trang đăng nhập ArangoDB

Bước 11: Chọn database máy chủ là " _system"



Hình 12: Giao diện lựa chọn Database

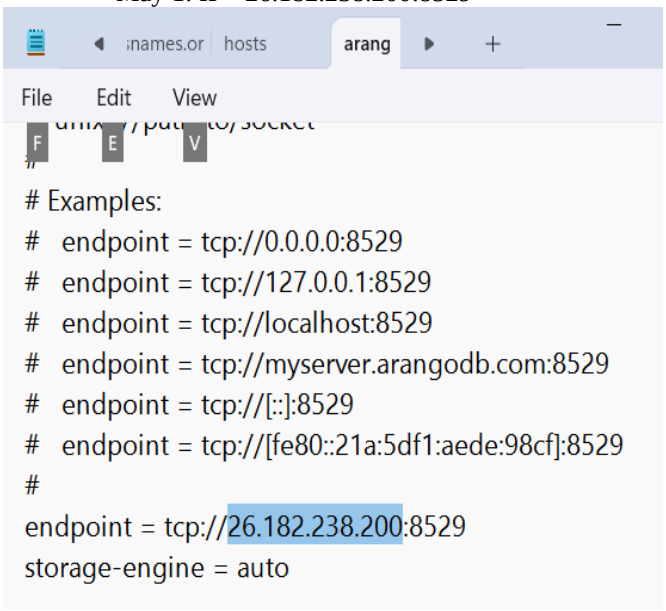


Hình13: Màn hình làm việc của ArangoDB

B. *Kết nối ba máy*

- Yêu cầu 3 máy đã cài ArangoDB
- Theo mặc định, các máy khi chạy ArangoDB thì sẽ chạy trên địa chỉ Localhost (127.0.0.1)
- Để các máy liên kết với nhau ta tiến hành chỉnh sửa endpoint thành IP máy:
  - Vào thư mục arangod.conf bằng đường dẫn ect -> arangodb3.
  - Tiến hành chỉnh sửa endpoint.

Máy 1: IP - 26.182.238.200:8529



Hình 1: Chỉnh sửa endpoint ở máy 1

Máy 2: IP - 26.83.35.244:8529

```
# Examples:
#   endpoint = tcp://0.0.0.0:8529
#   endpoint = tcp://127.0.0.1:8529
#   endpoint = tcp://localhost:8529
#   endpoint = tcp://myserver.arangodb.com:8529
#   endpoint = tcp://[::]:8529
#   endpoint = tcp://[fe80::21a:5df1:aede:98cf]:8529
#
endpoint = tcp://26.83.35.244:8529
storage-engine = auto
```

Hình 2: Chỉnh sửa endpoint ở máy 2

Máy 3: IP – 26.191.62.34:8529

```
File    Edit    View
#
# Examples:
#   endpoint = tcp://0.0.0.0:8529
#   endpoint = tcp://127.0.0.1:8529
#   endpoint = tcp://localhost:8529
#   endpoint = tcp://myserver.arangodb.com:8529
#   endpoint = tcp://[::]:8529
#   endpoint = tcp://[fe80::21a:5df1:aede:98cf]:8529
#
endpoint = tcp://26.191.62.34:8529
storage-engine = auto
```

Hình 3: Chỉnh sửa endpoint ở máy 3

1) Khởi tạo cluster cho 3 máy

Bước 1: Chạy lệnh khởi tạo

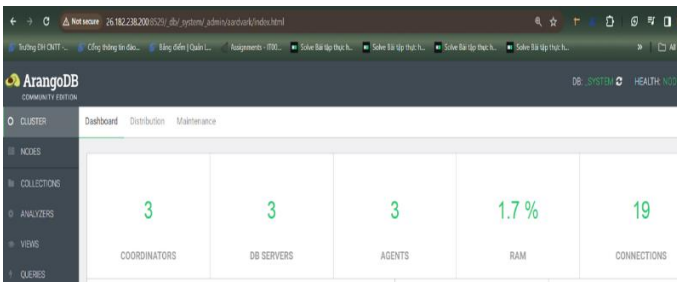
```
arangodb \
  --server.storage-engine=rocksdb \
  --starter.data-dir=data \
  --starter.join
26.182.238.200,26.83.35.244,26.191.62.34
```

Bước 2: Đăng nhập ArangoDB bằng địa chỉ IP của từng máy

- Máy 1: http://26.182.238.200:8529
- Máy 2: http://26.83.35.244:8529
- Máy 3: http://26.191.62.34:8529

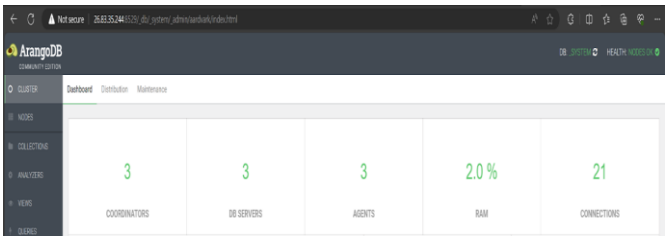
Bước 3: Kiểm tra tín hiệu kết nối

- Máy 1:



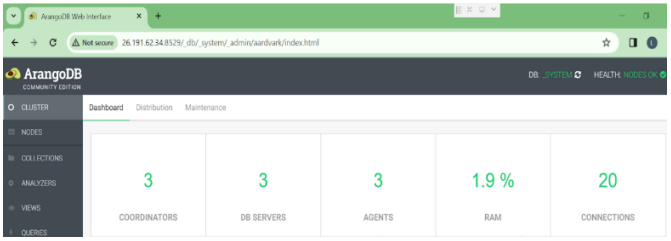
Hình 4: Máy 1 khi đăng nhập vào địa chỉ IP

- Máy 2:



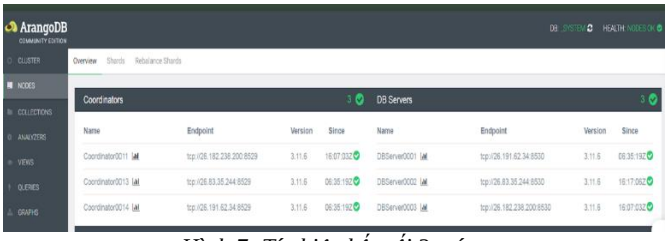
Hình 5: Máy 2 khi đăng nhập vào địa chỉ IP

- Máy 3:



Hình 6: Máy 3 khi đăng nhập vào địa chỉ IP

- Tín hiệu kết nối:



Hình 7: Tín hiệu kết nối 3 máy

V. THỰC NGHIỆM

A. Xây dựng mô hình phân tán

Mô hình quản lí cửa hàng vật liệu xây dựng với các quan hệ:

- **SANPHAM** (MASP, TENSP, GIA, SL): lượt đồ thể hiện các sản phẩm có trong của hàng. Bao gồm mã sản phẩm (MASP), tên sản phẩm (TENSP), giá của sản phẩm (GIA) và số lượng còn trong kho (SL).
- **HOADON** (MAHD, KHACHHANG, SDT, DIACHI, TONGHD): lượt đồ thể hiện hoá đơn bán hàng của cửa hàng với mã hoá đơn (MAHD), khách hàng (KHACHHANG), số điện thoại liên lạc (SDT), địa chỉ nhận hàng (DIACHI) và nhân viên phụ trách hoá đơn (MANV).
- **CTHD** (MAHD, MASP, SOLUONG, THANHTIEN): lượt đồ thể hiện chi tiết các sản phẩm trong hoá đơn với mã hoá đơn(MAHD), mã sản phẩm (MASP), số lượng mua (SOLUONG), tổng tiền của sản phẩm này (THANHTIEN).
- **NHANVIEN** (MANV, TENNV, CHUCVU): Lược đồ thể hiện thông tin của nhân viên trong cửa hàng với mã nhân viên (MANV), tên nhân viên (TENNV), chức vụ của nhân viên (CHUCVU).

***Yêu cầu:** triển khai mô hình trên 1 Cluster phân tán, Admin sẽ có quyền truy xuất và ghi vào tất cả các quan hệ. Đồng

thời, dữ liệu được ghi phải đồng bộ với nhau giữa các máy trong Cluster. Dữ liệu mẫu:

SANPHAM

MASP	TENSP	GIA	SL
CAT	Cát	140.000	10000
DA	Đá	230.000	20000
XM	Xi măng	74.000	50000

HOADON

MAHD	KHACHHANG	SDT	DIACHI	MANV
HD01	Nguyễn Đức Trí	0937859594	Ấp 3, Tân Thành, Thủ Thừa, Long An	NV04
HD02	Nguyễn Hoàng Nhân	028347852	Tân Mai, Biên Hoà, Đồng Nai	NV03
HD03	Đặng Hoàng Quân	036845271	Phường 15, Tân Bình,TP.HCM	NV04
HD04	Nguyễn Đức Trí	0937859594	Ấp 3, Tân Thành, Thủ Thừa, Long An	NV03

CTHD

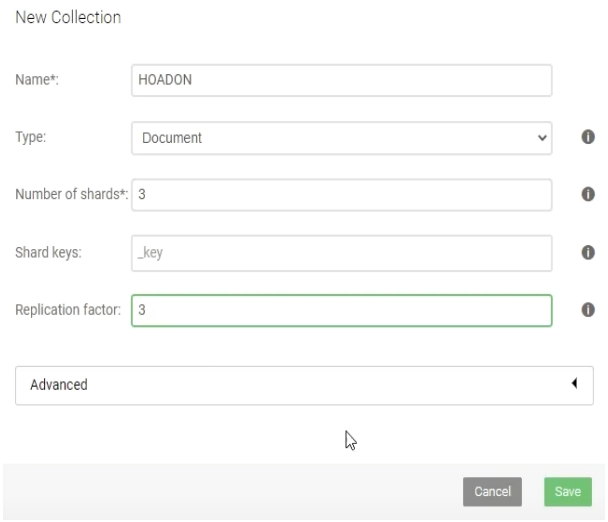
MAHD	MASP	SOLUONG	THANHTIEN
HD01	CAT	20	2.800.000
HD01	DA	10	2.300.000
HD02	XM	30	2.220.000
HD03	CAT	10	1.400.000
HD04	XM	10	740.000

NHANVIEN

MANV	TENNV	CHUCVU
NV01	Phạm Phú Phước	Quản lí
NV02	Nguyễn Thị B	Kế toán
NV03	Trần Văn A	Vận chuyển
NV04	Trần Văn B	Vận chuyển

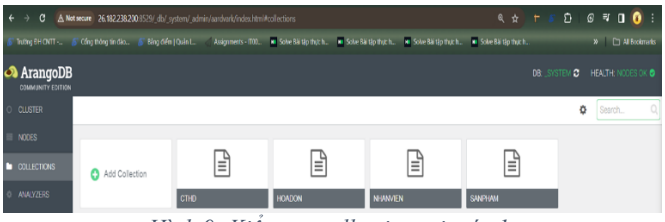
B. Tạo collection

- Sau khi thiết lập Cluster thành công, nhóm tiến hành tạo các Collection tương ứng với các quan hệ. Ta chỉ cần truy cập vào 1 máy trong Cluster, tại thanh Menu ta chọn COLLECTIONS -> Add Collections. Một hộp thoại xuất hiện để ta nhập các thông tin cần thiết cho Collection muốn tạo.
- Ở đây, nhóm chọn "Number of shards" và Replication Factor đều bằng 3, nghĩa là mỗi Collection sẽ được phân thành 3 đoạn.



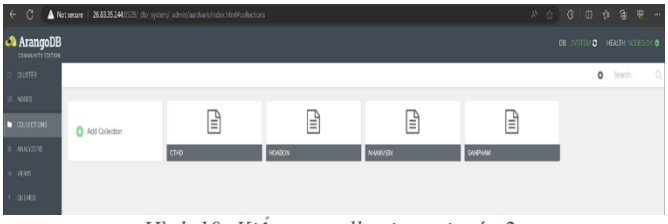
Hình 8: Tạo một collection mới

- Tương tự với 3 bảng còn lại (CTHD, NHANVIEN, SANPHAM)
- Các Collection sau khi được tạo sẽ có trên cả 3 máy, collections được thêm vào sẽ được đồng bộ qua lại với nhau. Ta có thể xem 3 máy này như 1 máy duy nhất.
- Máy 1:



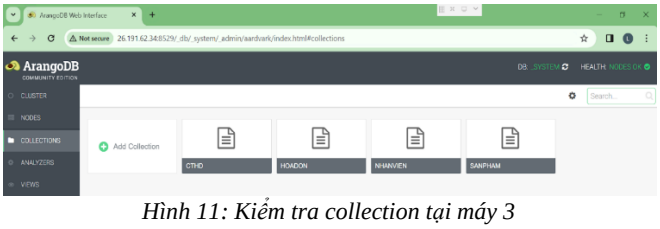
Hình 9: Kiểm tra collection tại máy 1

- Máy 2:



Hình 10: Kiểm tra collection tại máy 2

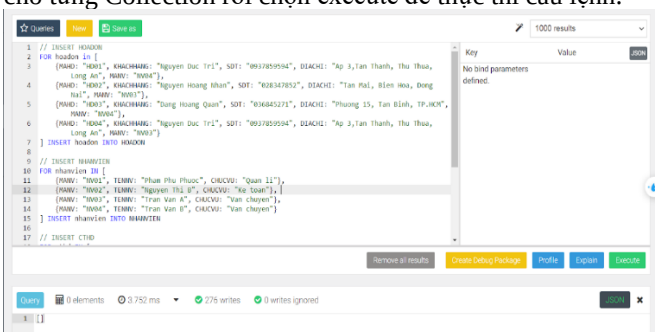
- Máy 3:



Hình 11: Kiểm tra collection tại máy 3

C. Tạo dữ liệu

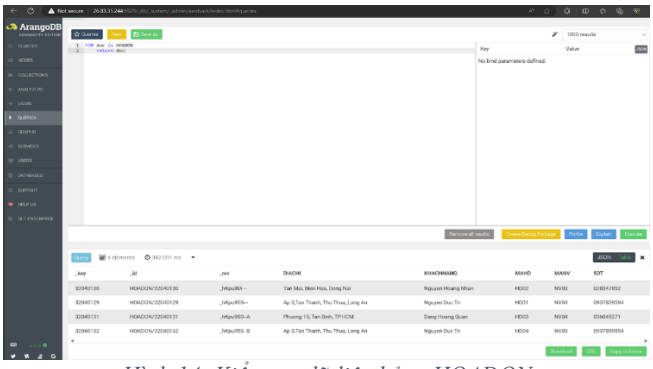
Tiếp theo, nhóm tiến hành sử dụng các lệnh AQL để tiến hành thêm dữ liệu cho từng Collection bằng cách nhấn vào mục QUERIES tại thanh Menu và nhập các lệnh AQL tương ứng cho từng Collection rồi chọn execute để thực thi câu lệnh:



Hình 12: Thêm dữ liệu cho bảng HOADON và NHANVIEN

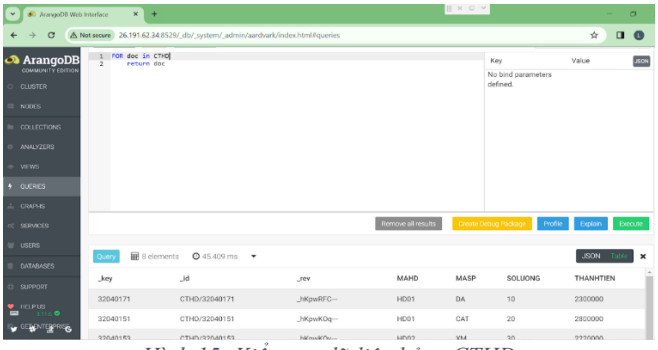
Hình 13: Thêm dữ liệu cho bảng CTHD và SANPHAM

- Kiểm tra dữ liệu vừa mới tạo trên 3 máy:
- Máy 2: Kiểm tra dữ liệu bảng HOADON



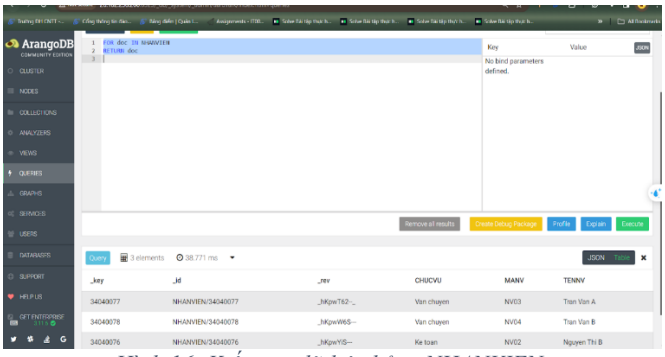
Hình 14: Kiểm tra dữ liệu bảng HOADON

- Máy 3: Kiểm tra dữ liệu bảng CTHD



Hình 15: Kiểm tra dữ liệu bảng CTHD

- Máy 1: Kiểm tra dữ liệu bảng NHANVIEN

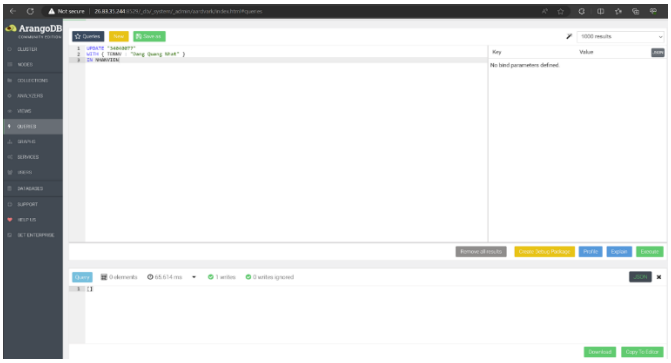


Hình 16: Kiểm tra dữ liệu bảng NHANVIEN

D. Thay đổi dữ liệu (update)

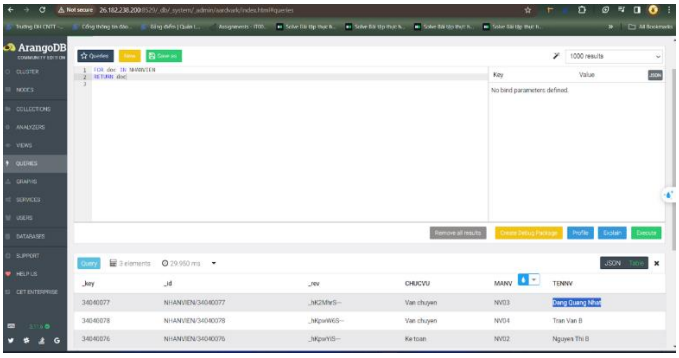
Sử dụng các lệnh AQL để thay đổi dữ liệu bằng cách nhấn vào mục QUERIES tại thanh Menu và nhập các lệnh AQL tương ứng với collection rồi chọn execute để thực thi câu lệnh:

- Máy 2: Thực hiện Update nhân viên có _key '34040077' thành tên Dang Quang Nhat



Hình 17: Máy 2 thực hiện update bảng NHANVIEN

- Máy 1: Kiểm tra lại dữ liệu

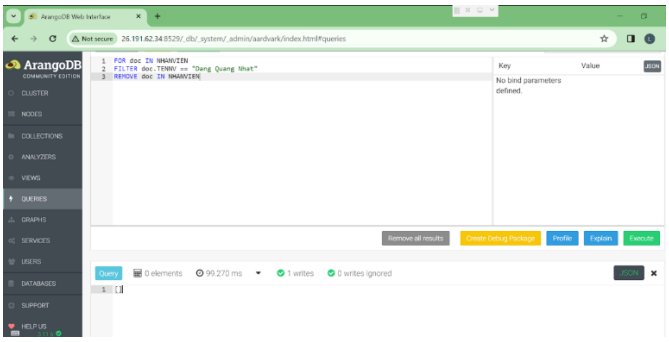


Hình 18: Máy 1 kiểm tra dữ liệu sau khi updatePhantom read

E. Xóa dữ liệu (Delete)

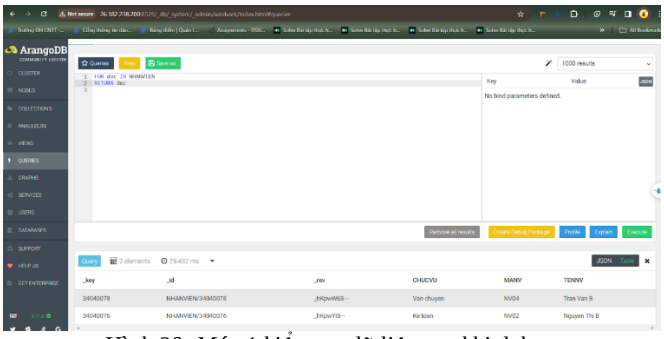
Sử dụng các lệnh AQL để xóa dữ liệu cho từng Collection bằng cách nhấn vào mục QUERIES tại thanh Menu và nhập các lệnh AQL tương ứng cho từng Collection rồi chọn execute để thực thi câu lệnh:

- Máy 3: Thực hiện Delete nhân viên có TENNV = "Dang Quang Nhat"



Hình 19: Máy 3 thực hiện delete dữ liệu bảng NHANVIEN

- Máy 1: Kiểm tra dữ liệu bảng NHANVIEN sau khi xóa

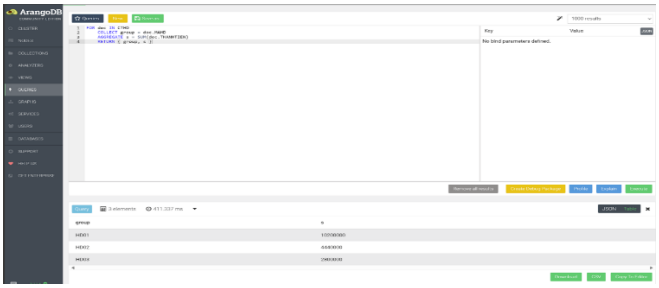


Hình 20: Máy 1 kiểm tra dữ liệu sau khi delete

F. Đọc dữ liệu

Sử dụng các lệnh AQL để đọc dữ liệu cho từng Collection bằng cách nhấn vào mục QUERIES tại thanh Menu và nhập các lệnh AQL tương ứng cho từng Collection rồi chọn execute để thực thi câu lệnh:

- Máy 2: Thực hiện đọc dữ liệu tại bảng CTHD hiển thị tổng tiền của từng hóa đơn.



Hình 21: Máy 2 thực hiện đọc dữ liệu tại bảng CTHD

VI. LỜI CẢM ƠN

Trước tiên, chúng tôi muốn dành những lời tri ân đặc biệt đến Thầy Nguyễn Minh Nhựt, một người giảng viên đã dày công truyền đạt tri thức và lòng đam mê của mình cho chúng tôi trong khóa học Cơ sở dữ liệu phân tán.

Những kiến thức mà Thầy truyền đạt đã trở thành tài sản vô giá, giúp chúng tôi vượt qua khó khăn và hoàn thành đề tài đồ án một cách xuất sắc hơn. Dựa trên những kiến thức được Thầy truyền đạt và sự cống hiến của cả nhóm trong việc tự nghiên cứu và khám phá các công cụ và kiến thức mới, chúng tôi đã dành thời gian để thiết kế một cơ sở dữ liệu phân tán.

Trong quá trình thực hiện đồ án, chúng tôi đã nỗ lực hết mình để đạt được kết quả tốt nhất. Tuy nhiên, không tránh khỏi những sai sót và hạn chế do sự hạn hẹp về thời gian và khả năng cá nhân của từng thành viên trong nhóm. Đồ án của chúng tôi, mặc dù đã có những nỗ lực, vẫn chỉ là một sản phẩm mang tính chất đồ án học tập và chưa thực sự hoàn thiện.

Chúng tôi trân trọng mong nhận được những góp ý giá trị từ Thầy, nhằm hoàn thiện những kiến thức đã được học và đồng thời làm bổ sung cho chúng tôi trong các dự án và công việc sắp tới. Chúng tôi biết ơn và trân trọng sự chia sẻ kiến thức và kỹ năng từ Thầy, đã giúp mỗi thành viên trong nhóm nâng cao vốn kiến thức hiện có, để chúng tôi có thể tiếp tục hoàn thiện các dự án và sản phẩm khác trong tương lai.

Một lần nữa, chúng tôi muốn gửi lời cảm ơn chân thành và tốt đẹp nhất đến Thầy cô và các bạn trong lớp!

TÀI LIỆU THAM KHẢO

[1] *"Database link with Radmin VPN"*, tài liệu thực hành Cơ sở dữ liệu phân tán, Trường Đại học Công nghệ thông tin

[2] Youtuber Trevor Sullivan *"Intro to #OpenSource ArangoDB Document & Graph #Database!"*

[3] Tài liệu SQL /AQL "SQL / AQL – Comparison"

[4] Tài liệu AQL "AQL Documentation"