BỘ CÔNG THƯƠNG

**TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP. HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

---------------------------



**ĐỒ ÁN MÔN HỌC**

**NOSQL – Tìm hiểu công cụ ArangoDB Web Interface để quản trị CSDL đồ thị “Quản lý lộ trình xe buýt công cộng tại TP.HCM”. Xây dựng ứng dụng minh họa.**

**GVHD: Nguyễn Thị Định**

**2044226229\_ Võ Ngọc Nguyên Chương**

**2001220536\_ Cao Thiên Chi**

**2001221445 \_ Trương Mỹ Hoa**

*TP. HỒ CHÍ MINH, tháng 11 năm 2025*

BỘ CÔNG THƯƠNG

**TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP. HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

---------------------------



**ĐỒ ÁN MÔN HỌC**

**NOSQL – Tìm hiểu công cụ ArangoDB Web Interface để quản trị CSDL đồ thị “Quản lý lộ trình xe buýt công cộng tại TP.HCM”. Xây dựng ứng dụng minh họa.**

**GVHD: Nguyễn Thị Định**

**2044226229\_ Võ Ngọc Nguyên Chương**

**2001220536\_ Cao Thiên Chi**

**2001221445 \_ Trương Mỹ Hoa**

*TP. HỒ CHÍ MINH, tháng 11 năm 2025*

**BẢNG PHÂN CÔNG VIỆC NHÓM CHO CÁC THÀNH VIÊN**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **MSSV** | **HỌ TÊN** | **CÔNG VIỆC ĐƯỢC GIAO** | **ĐÁNH GIÁ** |
| 1 | 2044226229 | Võ Ngọc Nguyên Chương | * Biên soạn nội dung Chương 1, 2, 3, 4. * Hiệu chỉnh câu từ, mở rộng và hoàn thiện Chương 5. * Soạn thảo và trình bày báo cáo Word. * Hỗ trợ lập trình/code. * Thuyết trình phần Tổng quan đề tài và Dataset. | 100% – A |
| 2 | 2001221445 | Trương Mỹ Hoa | * Biên soạn nội dung Chương 6. * Chuẩn bị báo cáo PowerPoint. * Hỗ trợ lập trình/code. * Thuyết trình về Kỹ thuật và công nghệ sử dụng. | 100% – A |
| 3 | 2001220536 | Cao Thiên Chi | * Biên soạn nội dung Chương 5. * Thuyết trình về Code. * Thực hiện phần Code chính. | 100% – A |

**LỜI CAM ĐOAN**

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

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

**Sinh viên thực hiện Báo cáo**

*(Ký và ghi rõ họ tên)*

**LỜI CÁM ƠN**

Trước hết, nhóm xin gửi lời cảm ơn chân thành đến cô Nguyễn Thị Định đã tận tình hướng dẫn, hỗ trợ nhóm trong suốt quá trình thực hiện báo cáo môn học này. Nhờ vào sự chỉ dẫn và góp ý quý báu của cô mà nhóm đã hoàn thiện được báo cáo với đề tài "Tìm hiểu công cụ ArangoDB Web Interface để quản trị CSDL đồ thị “Quản lý lộ trình xe buýt công cộng tại TP.HCM”. Xây dựng ứng dụng minh họa".

Nhóm cũng xin cảm ơn nhà trường và khoa Công Nghệ Thông Tin đã tạo điều kiện thuận lợi cho nhóm có cơ hội tìm hiểu và nghiên cứu chuyên sâu về lĩnh vực này.

Cuối cùng, nhóm xin cảm ơn bản thân các thành viên trong nhóm vì đã cố gắng hoàn thành tốt nhiệm vụ và học hỏi được nhiều kiến thức bổ ích trong quá trình thực hiện báo cáo này.

Võ Ngọc Nguyên Chương

Trương Mỹ Hoa

Cao Thiên Chi

# 

# MỤC LỤC

[MỤC LỤC iii](#_Toc213014126)

[DANH MỤC CÁC HÌNH ẢNH vii](#_Toc213014127)

[DANH MỤC BẢNG BIỂU x](#_Toc213014128)

[DANH MỤC TỪ VIẾT TẮT xi](#_Toc213014129)

[TÀI LIỆU THAM KHẢO 74](#_Toc213014223)

# 

# DANH MỤC CÁC HÌNH ẢNH

# 

# DANH MỤC BẢNG BIỂU

# 

# DANH MỤC TỪ VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| **Từ viết tắt** | **Nghĩa Tiếng Anh** | **Nghĩa Tiếng Việt** |

# CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI

## Lý do chọn đề tài

Hệ thống giao thông công cộng bằng xe buýt tại TP.HCM là một trong những mạng lưới phức tạp nhất cả nước với hàng trăm tuyến xe buýt, hàng nghìn trạm dừng, hàng ngàn lượt xe vận hành mỗi ngày. Việc quản lý, theo dõi, vận hành và điều phối một hệ thống quy mô lớn như vậy đòi hỏi một công cụ hỗ trợ mạnh mẽ và hiệu quả.

Tuy nhiên, dữ liệu trong hệ thống xe buýt mang tính kết nối cao:

* Trạm này liên kết với trạm kia.
* Tuyến đi qua nhiều trạm khác nhau.
* Mỗi xe phục vụ những tuyến nhất định.

Những mối quan hệ dạng mạng lưới (graph) này khiến các hệ quản trị CSDL quan hệ truyền thống gặp nhiều hạn chế, đặc biệt khi xử lý các truy vấn phức tạp như: tìm đường đi ngắn nhất, truy vấn các tuyến kết nối, hay phân tích sự phụ thuộc giữa các trạm.

Từ thực tiễn này, đồ án lựa chọn ArangoDB, một hệ quản trị cơ sở dữ liệu đa mô hình (Multi-model Database), đặc biệt mạnh trong việc xử lý dữ liệu dạng đồ thị. ArangoDB có ArangoDB Web Interface – giao diện quản trị trực quan, hiện đại, giúp thao tác với dữ liệu dễ dàng, phù hợp cho hệ thống quản lý lộ trình xe buýt.

Do đó, việc nghiên cứu và xây dựng ứng dụng minh họa “Hệ thống Quản lý Xe buýt TP.HCM” bằng ArangoDB là cấp thiết, vừa mang giá trị thực tiễn, vừa thể hiện tính ứng dụng cao của công nghệ CSDL đồ thị.

# 1.2. Mục tiêu nghiên cứu

* Xây dựng mô hình cơ sở dữ liệu đồ thị cho mạng lưới xe buýt: Mô hình hóa các thực thể chính của hệ thống (Trạm – Tuyến – Phương tiện – Lịch trình), xây dựng các quan hệ đồ thị (Graph Edges) để phản ánh sự kết nối giữa các trạm, tuyến và phương tiện, thiết kế cấu trúc dữ liệu phù hợp cho việc truy vấn đường đi, phân tích mạng lưới và thống kê.
* Xây dựng hệ thống quản trị dữ liệu (CRUD) trên nền tảng ArangoDB Web Interface: Sử dụng Web Interface để thực hiện các thao tác với dữ liệu: tạo, xem, cập nhật, xóa. Đảm bảo dữ liệu được tổ chức logic, đồng nhất và dễ mở rộng.
* Phát triển các tính năng thông minh dựa trên Graph Database: Khai thác sức mạnh đồ thị của ArangoDB để thực hiện các chức năng nâng cao: Tìm kiếm đường đi (Journey Planner) sử dụng thuật toán shortest path, truy vấn nâng cao bằng AQL nhằm phân tích sâu dữ liệu (các tuyến kết nối, các trạm quan trọng, sự phân bố mạng lưới).
* Phát triển bộ phân tích và trực quan hóa dữ liệu: Xây dựng dashboard tổng quan tình trạng hệ thống, cung cấp biểu đồ thống kê về tuyến, trạm, phương tiện và lịch trình, hỗ trợ người quản trị đưa ra quyết định tốt hơn dựa trên dữ liệu.
* Tích hợp chức năng quản trị hệ thống và người dùng: Xây dựng mô hình xác thực và phân quyền người dùng, cho phép Admin quản lý tài khoản người dùng và vai trò trong hệ thống, đảm bảo tính bảo mật trong thao tác quản trị dữ liệu.

# 1.3. Phạm vi và đối tượng nghiên cứu

* Đối tượng nghiên cứu
  + Mạng lưới xe buýt TP.HCM: tuyến, trạm, phương tiện và các quan hệ kết nối.
  + Công cụ ArangoDB Web Interface và cơ chế hoạt động của Graph Database.
* Phạm vi nghiên cứu
  + Tập trung vào dữ liệu của hệ thống xe buýt trong đô thị TP.HCM.
  + Chỉ xây dựng mô hình CSDL và ứng dụng minh họa quy mô nhỏ.
  + Không đi sâu vào quy trình vận hành thực tế, yếu tố kinh doanh hay tối ưu vận tải ở cấp độ doanh nghiệp.

# 1.4. Phương pháp thực hiện

* Thu thập và phân tích yêu cầu
  + Tìm hiểu đặc điểm hệ thống xe buýt TP.HCM.
  + Xác định các thực thể cần quản lý (trạm, tuyến, xe,…).
    - * Nghiên cứu ArangoDB và Web Interface
        + Tìm hiểu mô hình đa mô hình (Multi-model), đặc biệt là Graph Model.
        + Nghiên cứu giao diện Web để quản trị dữ liệu.

Thiết kế mô hình dữ liệu đồ thị

Xây dựng các tập vertex (trạm, tuyến, phương tiện).

Xây dựng các tập edge thể hiện mối quan hệ (tuyến–trạm, trạm–trạm, xe–tuyến).

Triển khai hệ thống mẫu

Tạo database, collections, graph trên Web Interface.

Nhập dữ liệu mẫu và viết truy vấn AQL.

Xây dựng ứng dụng minh hoạ

Đánh giá và đề xuất hướng phát triển

Đánh giá tính hiệu quả của mô hình.

Đề xuất khả năng mở rộng trong tương lai.

**1.5. Cấu trúc báo cáo**

# CHƯƠNG 2: TỔNG QUAN VỀ ARANGODB VÀ CSDL ĐỒ THỊ

# 2.1. Tổng quan về CSDL đồ thị

Cơ sở dữ liệu đồ thị (Graph Database) là dạng cơ sở dữ liệu được thiết kế nhằm lưu trữ và truy vấn những dữ liệu có mối quan hệ phức tạp, trong đó thông tin được biểu diễn dưới dạng các thực thể (vertex hoặc node) và mối quan hệ (edge) nối giữa chúng.[1] Mỗi node có thể chứa các thuộc tính mô tả, trong khi edge thể hiện loại quan hệ, hướng và có thể kèm theo thông tin bổ sung như trọng số hoặc thời gian. Nhờ mô hình hóa dữ liệu theo cấu trúc mạng lưới tự nhiên, CSDL đồ thị được sử dụng rộng rãi trong các lĩnh vực như mạng xã hội, bản đồ giao thông, phân tích gian lận, hệ thống gợi ý và quản lý vận tải.

Ưu điểm của CSDL đồ thị bao gồm:

* Mô hình hóa dữ liệu theo đúng bản chất kết nối, phản ánh tự nhiên các mạng lưới phức tạp như: *trạm → tuyến → phương tiện*.[2]
* Truy vấn các liên kết và đường đi nhiều bước (multi-hop) nhanh và trực quan, nhờ khả năng duyệt đồ thị (graph traversal) trực tiếp mà không cần JOIN nhiều bảng như trong hệ CSDL quan hệ truyền thống.
* Linh hoạt trong thay đổi mô hình dữ liệu (schema-less hoặc schema-flexible), cho phép dễ dàng thêm node, thêm loại quan hệ hoặc mở rộng cấu trúc dữ liệu mà không ảnh hưởng đến toàn hệ thống.[3]
* Hiệu suất cao trong các bài toán phân tích mạng, như tìm đường đi ngắn nhất (shortest path), xác định điểm trung chuyển, phát hiện cộng đồng (community detection), hoặc phân tích quan hệ bất thường trong dữ liệu lớn.

Với các hệ thống có cấu trúc kết nối phức tạp và mật độ quan hệ cao, đặc biệt như mạng lưới xe buýt (gồm nhiều trạm, nhiều tuyến, mỗi tuyến kết nối qua nhiều trạm và phương tiện), việc sử dụng CSDL đồ thị giúp quản lý dữ liệu dễ dàng, mô hình hóa trực quan, hỗ trợ tốt cho bài toán phân tích lộ trình và tối ưu đường đi. Điều này vượt trội hơn so với việc lưu trữ toàn bộ quan hệ trong bảng quan hệ truyền thống vốn gây tốn chi phí JOIN và khó mô hình hóa tuyến - trạm - phương tiện một cách tự nhiên.

# 2.2. Tổng quan về ArangoDB

## 2.2.1. Giới thiệu

ArangoDB là một hệ quản trị cơ sở dữ liệu đa mô hình (multi-model database) mã nguồn mở, được thiết kế nhằm hỗ trợ nhiều mô hình dữ liệu khác nhau trong cùng một hệ thống, bao gồm Document, Graph và Key–Value [4]. Nhờ đó, ArangoDB cho phép lưu trữ và truy vấn dữ liệu linh hoạt, đồng thời giảm sự phức tạp khi phải kết hợp nhiều loại cơ sở dữ liệu khác nhau trong một ứng dụng lớn. ArangoDB sử dụng công cụ truy vấn AQL (ArangoDB Query Language) - một ngôn ngữ truy vấn thống nhất dùng cho cả document, graph và kiểu dữ liệu phụ trợ.

Một trong những đặc điểm nổi bật của ArangoDB là khả năng xử lý đồ thị mạnh mẽ, thông qua hai mô hình đồ thị chính:

* General Graph - cho phép người dùng tự định nghĩa tập các collection biểu diễn *đỉnh* (vertex collections) và *cạnh* (edge collections).
* Named Graph - một mô hình đồ thị được định nghĩa sẵn trong hệ thống, giúp đơn giản hóa quản lý và truy vấn.

ArangoDB hỗ trợ nhiều thuật toán đồ thị tích hợp như *shortest path*, *k-shortest paths*, *neighbors*, *traversal nhiều bước*, giúp tối ưu cho các bài toán phân tích kết nối phức tạp.



Hình: Logo của ArangoDB

## 2.2.2. Lịch sử phát triển

ArangoDB được bắt đầu phát triển vào năm 2011 bởi công ty TRIAGNO GmbH (sau này đổi tên thành ArangoDB GmbH). Trong giai đoạn đầu, hệ cơ sở dữ liệu này được phát hành dưới tên AvocadoDB, với định hướng xây dựng một hệ quản trị cơ sở dữ liệu đa mô hình thống nhất, hỗ trợ document, graph và key–value trong cùng một nền tảng.[5]

Đến năm 2012, AvocadoDB chính thức được đổi tên thành ArangoDB. Việc đổi tên được thực hiện để tạo ra thương hiệu dễ nhận diện hơn và phù hợp hơn với định hướng mở rộng sản phẩm. Theo ghi chép lịch sử, *“arango”* là tên một loại đơn vị tiền tệ cổ, được làm từ đá carnelian thô và từng được sử dụng trong trao đổi thương mại tại một số khu vực châu Phi trong nhiều thế kỷ trước.[6] Việc chọn tên này được nhà phát triển xem như một ẩn dụ cho *“đơn vị trao đổi dữ liệu”* - thể hiện mục tiêu cung cấp một hệ thống thống nhất cho nhiều dạng dữ liệu khác nhau.

Trong các năm tiếp theo, ArangoDB liên tục phát triển và được cộng đồng mã nguồn mở đón nhận nhờ:

* Kiến trúc đa mô hình duy nhất trên thị trường vào thời điểm ra mắt.
* Ngôn ngữ truy vấn AQL thống nhất.
* Hỗ trợ clustering và sharding, tối ưu cho dữ liệu lớn.
* Triển khai các tính năng đồ thị nâng cao, bao gồm shortest path, traversals và phân tích graph.

Từ năm 2015 trở đi, ArangoDB trở thành một trong những cơ sở dữ liệu đa mô hình nổi bật, được sử dụng trong các ứng dụng doanh nghiệp, phân tích mạng xã hội, IoT, log phân tán, giao thông vận tải, và các bài toán yêu cầu xử lý đồ thị phức tạp.[7]

## 2.2.3. Đặc điểm

ArangoDB cung cấp các truy vấn có thể mở rộng khi làm việc với dữ liệu đồ thị (graph). Sử dụng JSON làm định dạng lưu trữ mặc định nhưng bên trong sử dụng VelocyPack của ArangoDB - một định dạng nhị phân nhanh và nhỏ gọn để tuần tự hóa và lưu trữ. ArangoDB có thể lưu trữ một đối tượng JSON lồng nhau như 1 data entry bên trong 1 collection.

Do đó, không cần phải tháo rời các đối tượng JSON kết quả. Nhờ vậy, dữ liệu được lưu trữ sẽ đơn giản kế thừa cấu trúc cây của dữ liệu JSON. ArangoDB hoạt động trong 1 cluster phân tán và là DBMS đầu tiên được chứng nhận cho Hệ điều hành Datacenter (DC/OS).

DC/OS cho phép người dùng triển khai ArangoDB trên hầu hết các hệ sinh thái hiện có: Amazon Web Services (AWS), Google Compute Engine và Microsoft Azure. Hơn nữa, nó cung cấp khả năng triển khai chỉ với 1 cú nhấp chuột cho cluster của người dùng. ArangoDB cung cấp tích hợp với các native JavaScript microservices trực tiếp trên DBMS bằng cách sử dụng Foxx framework, tương tự như Node.js đa luồng.

Cơ sở dữ liệu này có AQL (Ngôn ngữ truy vấn ArangoDB) của riêng nó và cũng cung cấp GraphQL để viết các dịch vụ native web linh hoạt trực tiếp trên DBMS. Tính năng mới ArangoSearch là 1 công cụ tìm kiếm trong phiên bản 3.4. Công cụ tìm kiếm kết hợp khả năng truy xuất boolean với các thành phần ranking tổng quát cho phép truy xuất dữ liệu dựa trên 1 mô hình không gian vectơ chính xác (precise vector space).

# 2.3. Kiến trúc ArangoDB

Kiến trúc của ArangoDB được xây dựng nhằm hỗ trợ mô hình đa dữ liệu, xử lý phân tán và đảm bảo hiệu năng cao trong các ứng dụng yêu cầu nhiều kiểu dữ liệu kết hợp. ArangoDB vừa có thể chạy ở chế độ đơn máy, vừa hỗ trợ môi trường phân tán lớn, bao gồm replication, sharding, và cluster theo chuẩn Raft. Các thành phần trong kiến trúc được thiết kế để hoạt động đồng bộ nhưng độc lập, giúp mở rộng linh hoạt và an toàn.[8]

## 2.3.1. Các khái niệm cơ bản

* Multi-Model: document, graph và các cặp key-value - lập mô hình dữ liệu khi thấy phù hợp với ứng dụng của mình.
* Joins: Nối những gì thuộc về nhau một cách thuận tiện để truy vấn đặc biệt linh hoạt, ít dư thừa dữ liệu hơn.
* Transactions: Phát triển ứng dụng dễ dàng giữ cho dữ liệu nhất quán và an toàn. Không có rắc rối trong client

Dưới đây là một truy vấn AQL bao gồm tất cả các tính năng đó:



Hình . Câu truy vấn sử dụng các chức năng của ArangoDB

## 2.3.2. Mô hình lưu trữ

ArangoDB lưu trữ dữ liệu theo định dạng JSON, và các dữ liệu này được gọi là document. Các document được nhóm lại thành collection, và nhiều collection tồn tại bên trong một database.[9]

So với hệ quản trị CSDL quan hệ (RDBMS), nơi mỗi bảng cần xác định rõ các thuộc tính (columns) và quan hệ giữa chúng, trong ArangoDB thì không bắt buộc xác định schema cố định - các document có cấu trúc khác nhau vẫn có thể lưu cùng một collection.Tuy nhiên, vẫn có thể áp dụng giới hạn hoặc schema-validation nếu muốn để dễ thao tác.

ArangoDB hỗ trợ hai dạng collection chính liên quan tới mô hình đồ thị:

* Document Collection: là collection thông thường, chứa các document đại diện cho thực thể (vertices) hoặc dữ liệu dạng tài liệu.[9]
* Edge Collection: là collection đặc biệt để lưu trữ các document biểu diễn *cạnh* (edges) trong đồ thị. Mỗi document trong edge collection có thêm hai thuộc tính đặc biệt là \_from và \_to, dùng để xác định mối quan hệ giữa hai document khác (tức là hai vertex).[9]

## 2.3.3. Các phiên bản ArangoDB

Các phiên bản ArangoDB OASIS: phiên bản cloud của ArangoDB. Cung cấp cơ sở dữ liệu ArangoDB dưới dạng dịch vụ. Cho phép sử dụng toàn bộ chức năng của ArangoDB mà không cần phải tự cài đặc và quản lý hệ thống. Các phiên bản cài đặc khác trên các hệ điều hành khác nhau như Windows, Ubuntu, Docker, … gồm các phiên bản sau:

* ArangoDB Community Edition là một cơ sở dữ liệu native multi-model miễn phí được viết bằng C ++ với giấy phép nguồn mở (Apache 2).
* Ngoài Phiên bản Cộng đồng, ArangoDB Enterprise là một bản trả phí bao gồm SmartGraphs, Satellite Collections và nhiều tính năng bảo mật cấp doanh nghiệp. Ngoài ra còn có 1 số sản phẩm ở lĩnh vực khác như: BI Connector, ArangoML.

# 2.4. Lợi ích của ArangoDB trong quản lý giao thông đô thị

Ngoài ra, ArangoDB còn có các ưu điểm quan trọng:

* Hỗ trợ đa mô hình trong một engine duy nhất, giúp giảm độ phức tạp so với việc phải sử dụng nhiều hệ quản trị riêng (như MongoDB + Neo4j).
* Hỗ trợ sharding và clustering, đảm bảo khả năng mở rộng theo chiều ngang khi dữ liệu tăng trưởng lớn.[10]
* AQL trực quan, hỗ trợ JOIN, khắc phục hạn chế JOIN trong các hệ NoSQL đơn mô hình.
* Hiệu suất cao, nhờ cơ chế tối ưu bộ nhớ và cấu trúc dữ liệu được thiết kế cho workloads lớn.

Trong bối cảnh bài toán quản lý mạng lưới xe buýt, ArangoDB đặc biệt phù hợp nhờ:

* Mô hình Document giúp lưu trữ thông tin trạm, tuyến và phương tiện dưới dạng JSON.
* Mô hình Graph giúp mô tả trực quan quan hệ giữa các trạm và các tuyến.
* AQL hỗ trợ truy vấn đường đi (shortest path), tìm trạm lân cận, phân tích kết nối - đây là các thao tác cốt lõi để xây dựng ứng dụng phân tích và tối ưu lộ trình giao thông.

Nhờ sự kết hợp giữa tính linh hoạt, khả năng mở rộng và mô hình đồ thị tích hợp, ArangoDB trở thành một nền tảng phù hợp để xây dựng các hệ thống có tính kết nối cao, ví dụ như quản lý giao thông công cộng, hệ thống logistics, bản đồ đô thị hoặc phân tích mạng phức tạp.

# 

# CHƯƠNG 3. CÀI ĐẶT VÀ LÀM QUEN ARANGODB WEB INTERFACE

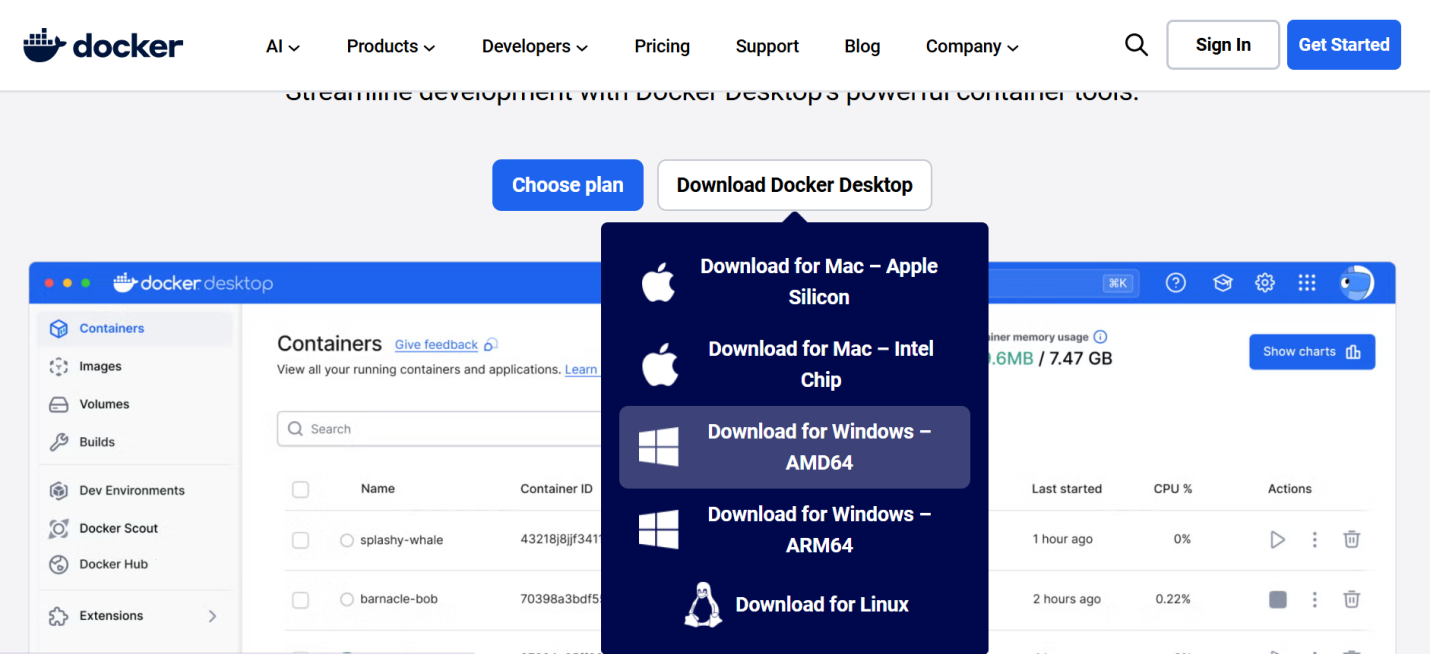
# 3.1. Yêu cầu hệ thống & môi trường cài đặt

Để triển khai ArangoDB phục vụ quá trình xây dựng và thử nghiệm hệ thống, cần chuẩn bị môi trường cài đặt phù hợp. Trong đề tài này, ArangoDB được triển khai thông qua Docker, giúp đảm bảo quá trình cài đặt đơn giản, nhanh chóng, đồng nhất giữa các máy và dễ dàng quản lý phiên bản.

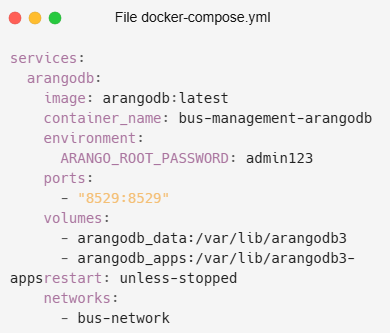
* Yêu cầu phần cứng tối thiểu
  + RAM: 4 GB (khuyến nghị 8 GB để chạy dataset lớn hoặc graph traversal)
  + Dung lượng ổ đĩa: tối thiểu 5 GB trống
  + Hệ điều hành: Windows 10/11, Ubuntu, macOS hoặc bất kỳ OS nào hỗ trợ Docker
    - * Yêu cầu phần mềm

Để chạy ArangoDB bằng Docker, cần cài đặt các phần mềm sau:

* Docker Desktop
  + Docker Desktop cho phép chạy container trên Windows, macOS hoặc Linux, tải tại: <https://www.docker.com/products/docker-desktop>, chọn loại phù hợp với hệ điều hành và nhấn download.



* + Sau khi tải file Docker Desktop Installer.exe về nhấn Install để cài đặt Docker.
* Docker-compose
  + Trong thư mục triển khai, tạo một file có tên docker-compose.yml chứa cấu hình dịch vụ ArangoDB. Nội dung file như hình sau:



Hình : File cấu hình docker-compose

Việc cài đặt ArangoDB thông qua Docker mang lại nhiều lợi ích:

* Không cần cài đặt thủ công ArangoDB vào hệ điều hành.
* Dễ dàng cấu hình thông qua file docker-compose.yml.
* Tránh lỗi phụ thuộc (dependency conflict).
* Có thể chạy nhiều phiên bản song song.
* Dễ backup dữ liệu bằng Docker volume.

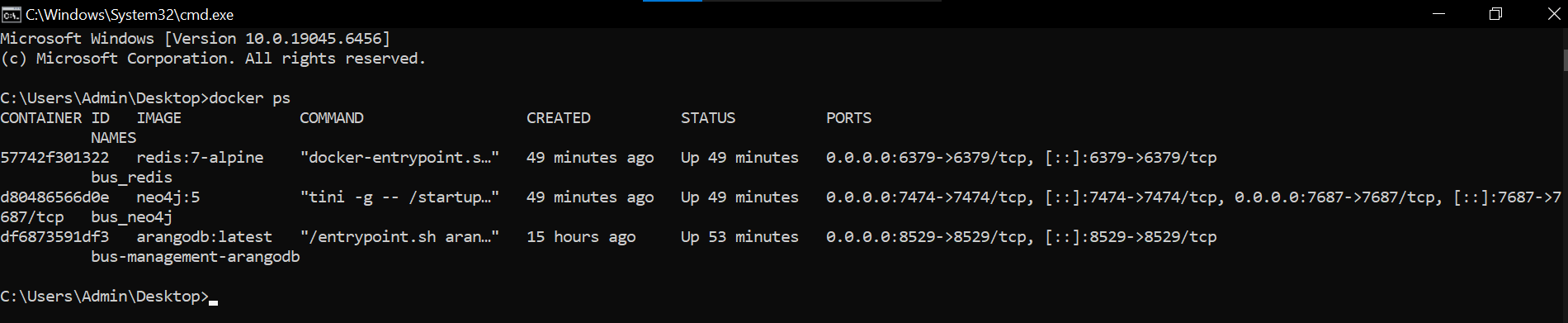
# 3.2. Cài đặt ArangoDB Web Interface

* Bật Docker Desktop lên
* Tại nơi lưu file docker-compose.yaml, chạy cmd với nội dung sau: *docker-compose up -d*



Hình : Chạy lệnh docker-compose up -d để cài đặt ArangoDB

* Đợi tải ArangoDB Web Interface

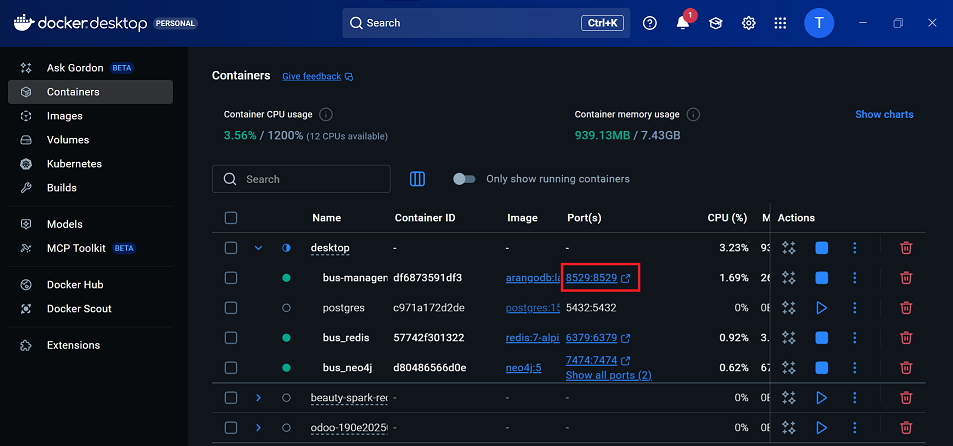


Hình : Kiểm tra trạng thái container ArangoDB bằng lệnh docker - ps

Nếu container có tên bus-management-arangodb xuất hiện → cài đặt thành công.

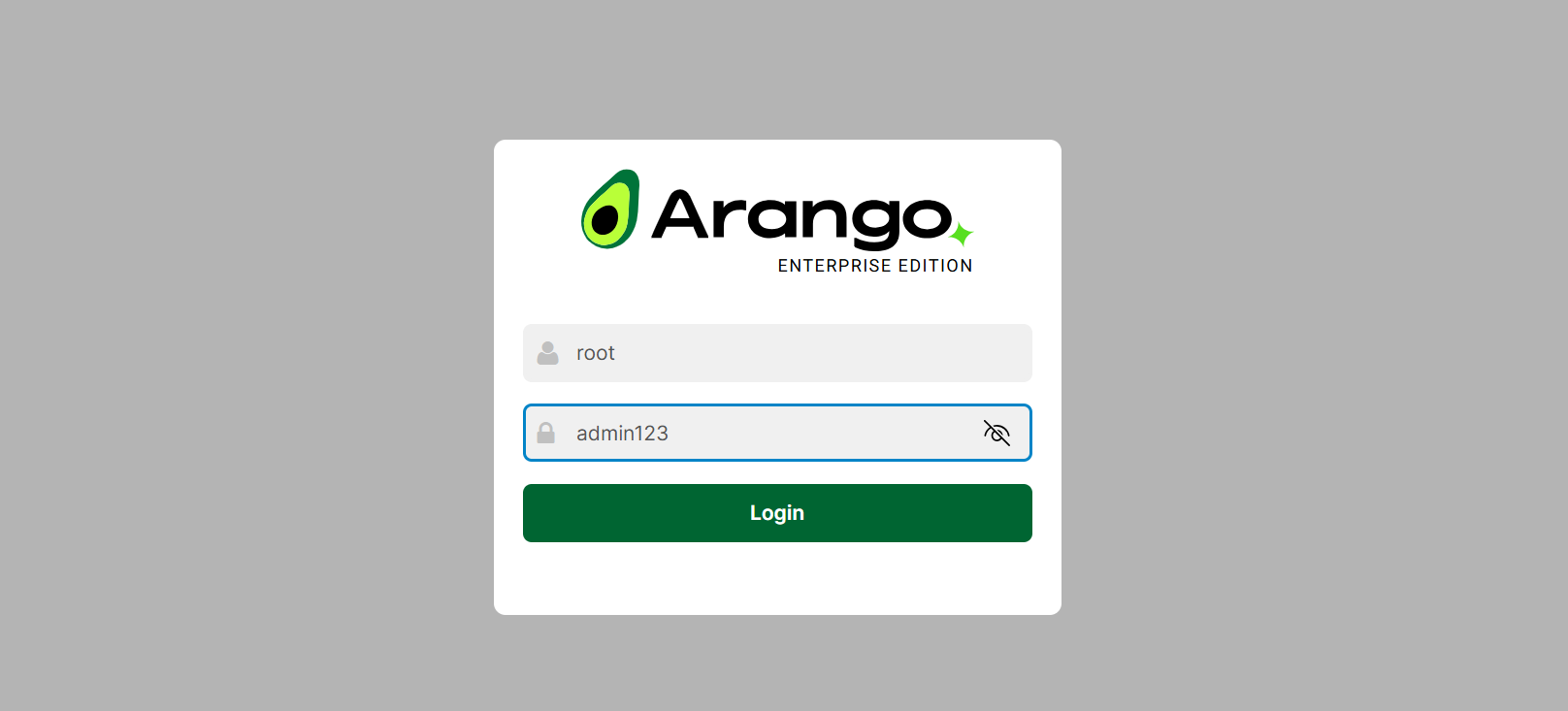
# 3.3. Truy cập và đăng nhập ArangoDB Web Interface

* Mở Dockertop, tại port của bus-management-arangodb nhấn vào đó sẽ hiện ra trang login của ArangoDB



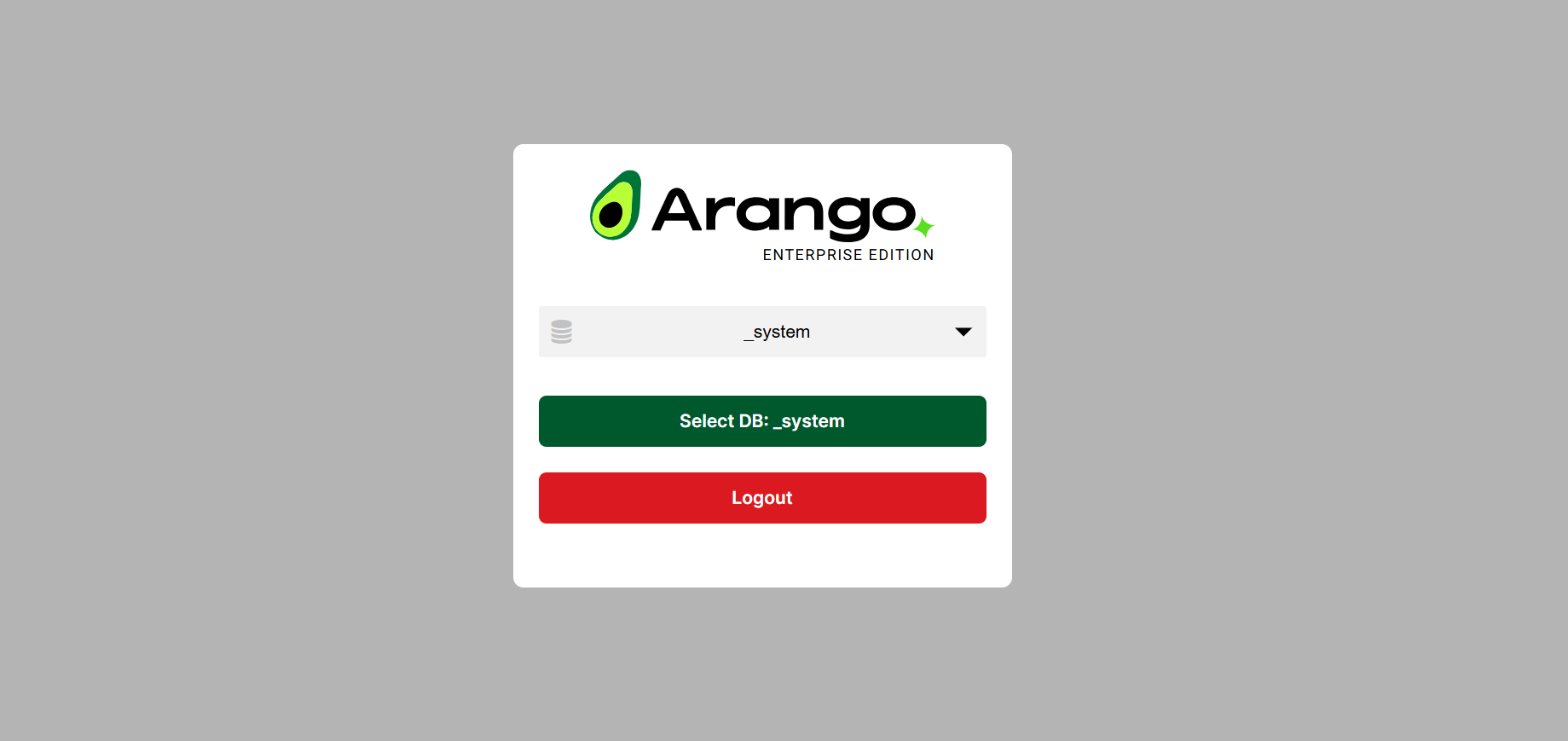
Hình : Giao diện Docker Desktop

* Tại trang login đăng nhập *user root và password: admin123*



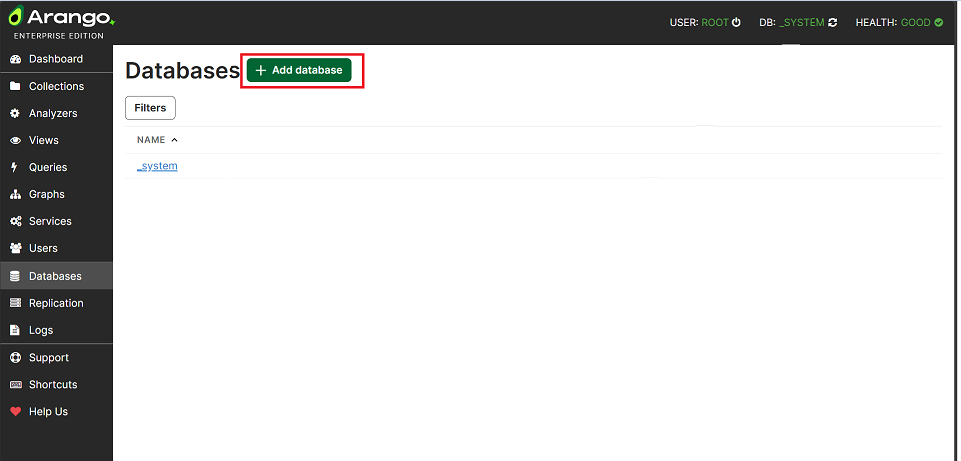
Hình : Giao diện đăng nhập của Arangodb

* Chọn nút SelecDB: \_system



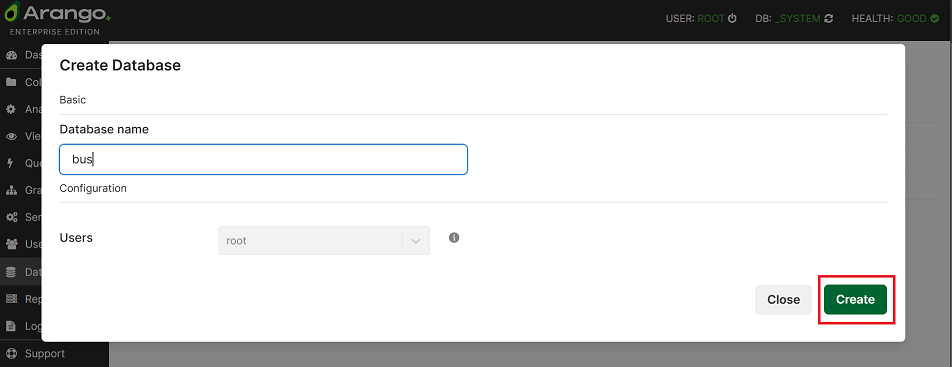
Hình : Giao diện chọn database của Arangodb

* Tại mục Database chọn *“Add database”*



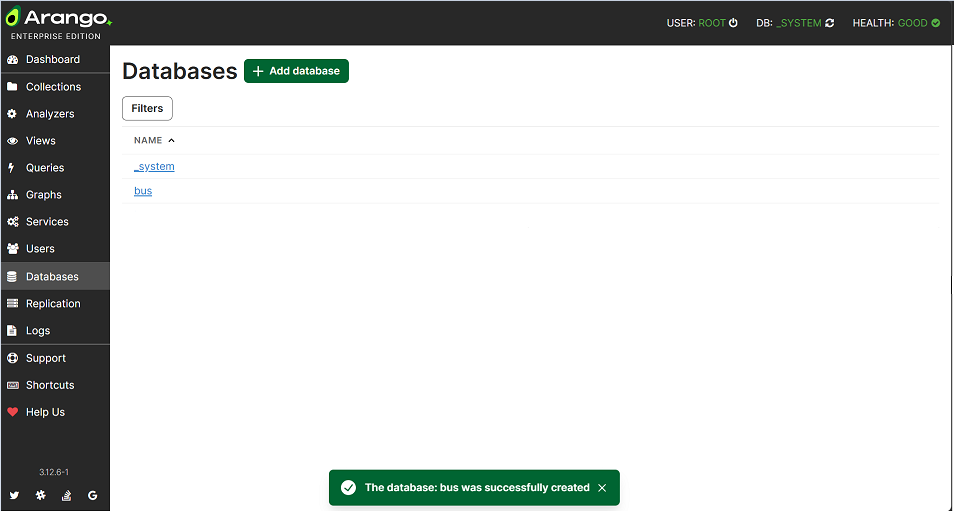
Hình : Giao diện chính của Arangodb

* Đặt tên database rồi chọn *“Create”*



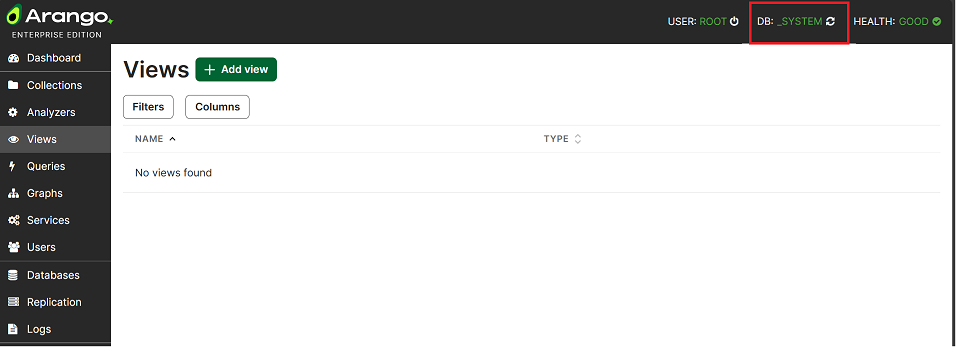
Hình : Giao diện tạo database của Arangodb

* Tạo thành công database



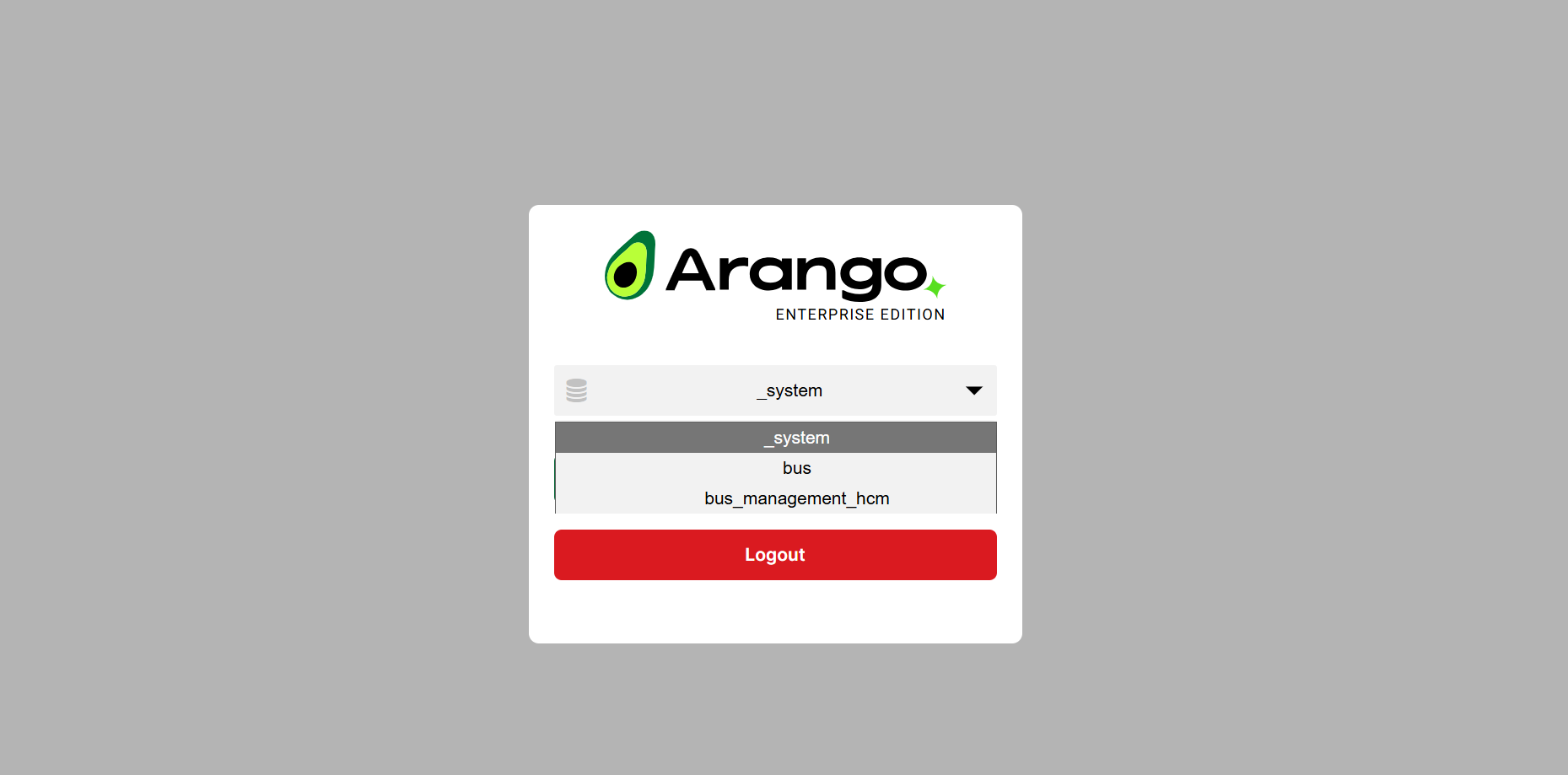
Hình : Danh sách database của Arangodb

* Chọn mục *“DB:\_System”*

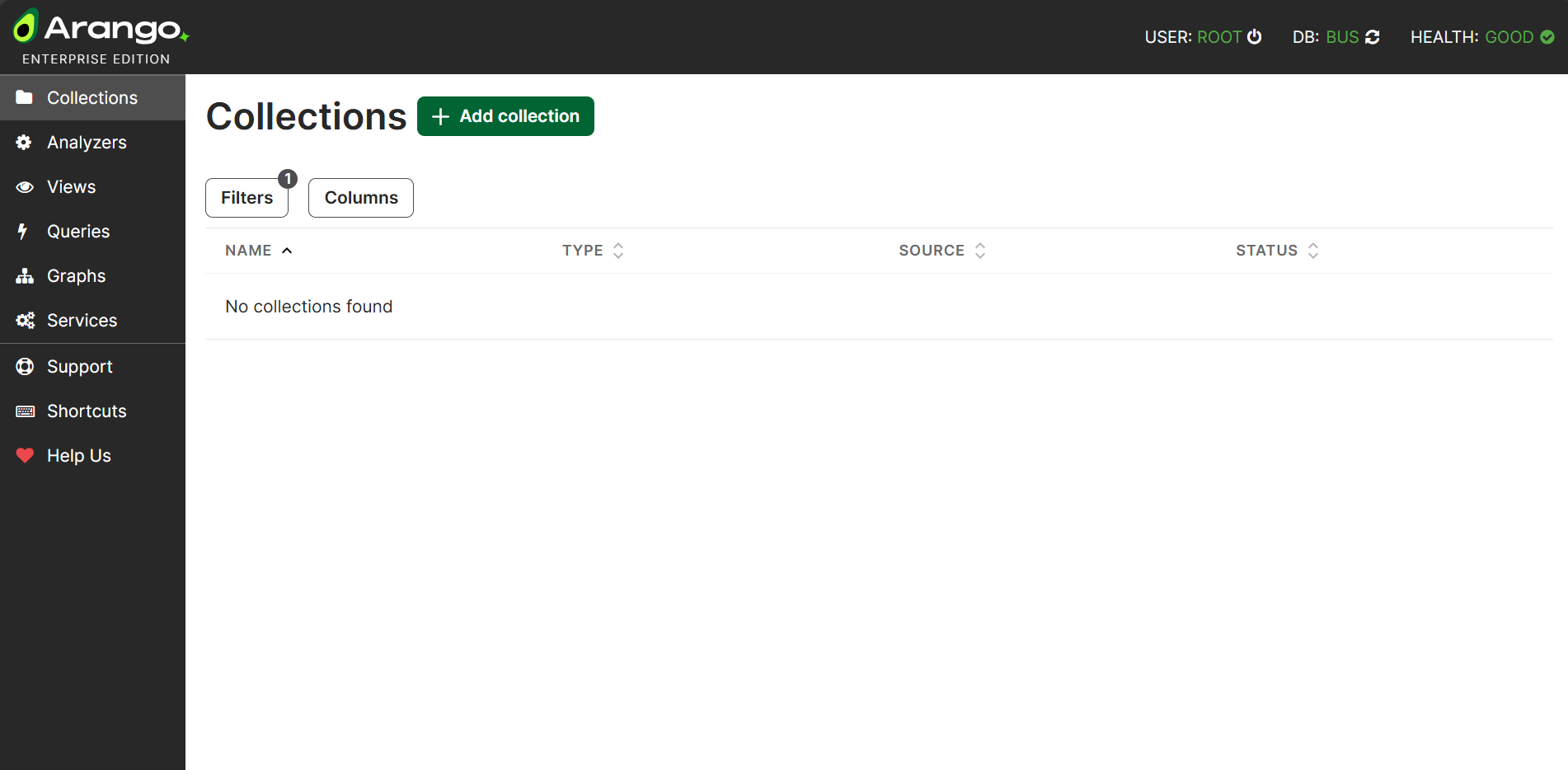


Hình : Giao diện chính \_system của Arangodb

* Chọn database cần quản lý, nhấn nút *“Select DB”*



Hình : Giao diện chọn database quản lý của Arangodb



Hình : Giao diện chính database Bus của Arangodb

# CHƯƠNG 4. CHỨC NĂNG QUẢN TRỊ TRÊN ARANGODB WEB INTERFACE

# 4.1. Quản lý Database

Trong ArangoDB, database là một “không gian lưu trữ độc lập” dùng để chứa toàn bộ dữ liệu của một ứng dụng.  
Mỗi database gồm:

* Collections (bảng / tập dữ liệu).
* Documents (tài liệu JSON).
* Edges (quan hệ trong mô hình đồ thị).
* Graphs (đồ thị).
* User, permission.
* AQL queries liên quan

Mỗi database hoạt động tách biệt với nhau → dữ liệu của database này không ảnh hưởng đến database khác.

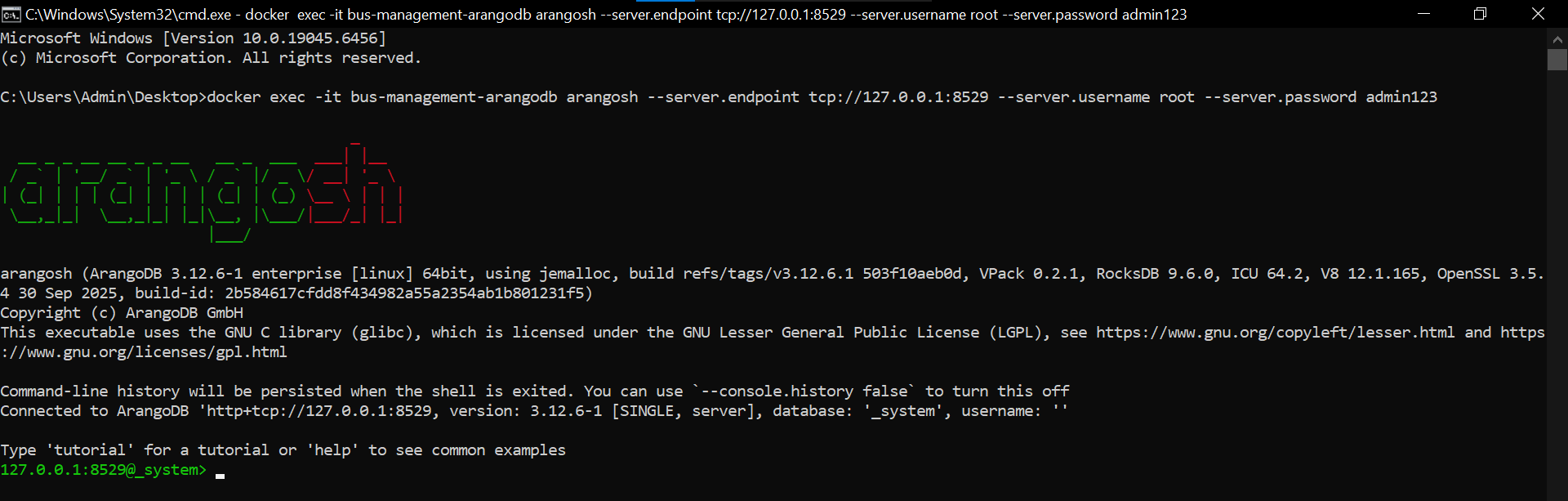
Để bắt đầu tạo database phải đã chạy “*docker-compose up -d”* và container bus-management-arangodb đang chạy - mật khẩu “*root”* của ArangoDB là “*admin123”.*

Có 2 cách tạo và quản lý database

* *Cách 1: Tạo database trên ArangoDb Web Interface - tham khảo mục 3.3*
* *Cách 2: Tạo database trên Terminal/Command Prompt - thực hiện các bước sau*

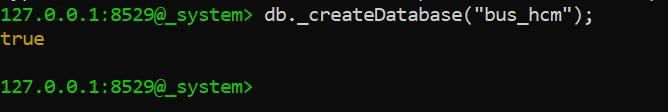
***Bước 1:***Mở Terminal/Command Prompt - mở một cửa sổ Terminal (trên Linux/macOS) hoặc Command Prompt/PowerShell (trên Windows).

*Bước 2:* Tại cmd chạy *“docker exec -it bus-management-arangodb arangosh --server.endpoint tcp://127.0.0.1:8529 --server.username root --server.password admin123”*



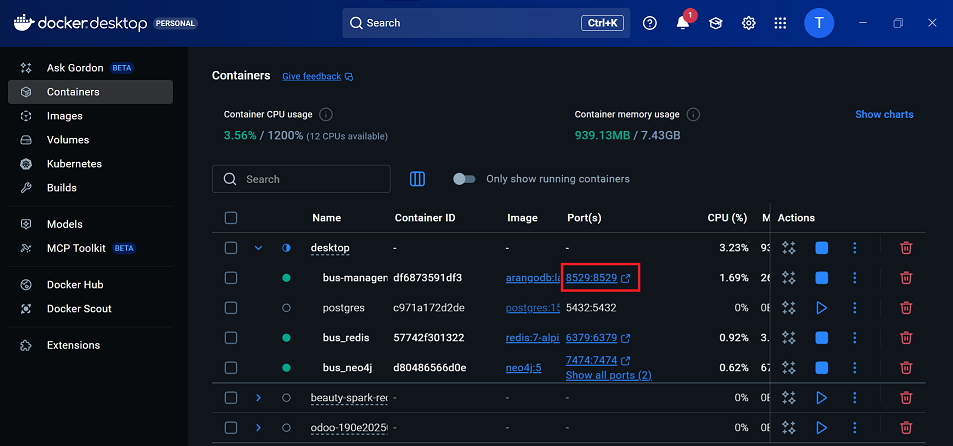
Hình : Kết nối với ArangoDB ở chế độ dòng lệnh

*Bước 3:* Tạo database với dòng lệnh *“db.\_createDatabase(“Ten\_database”);*”



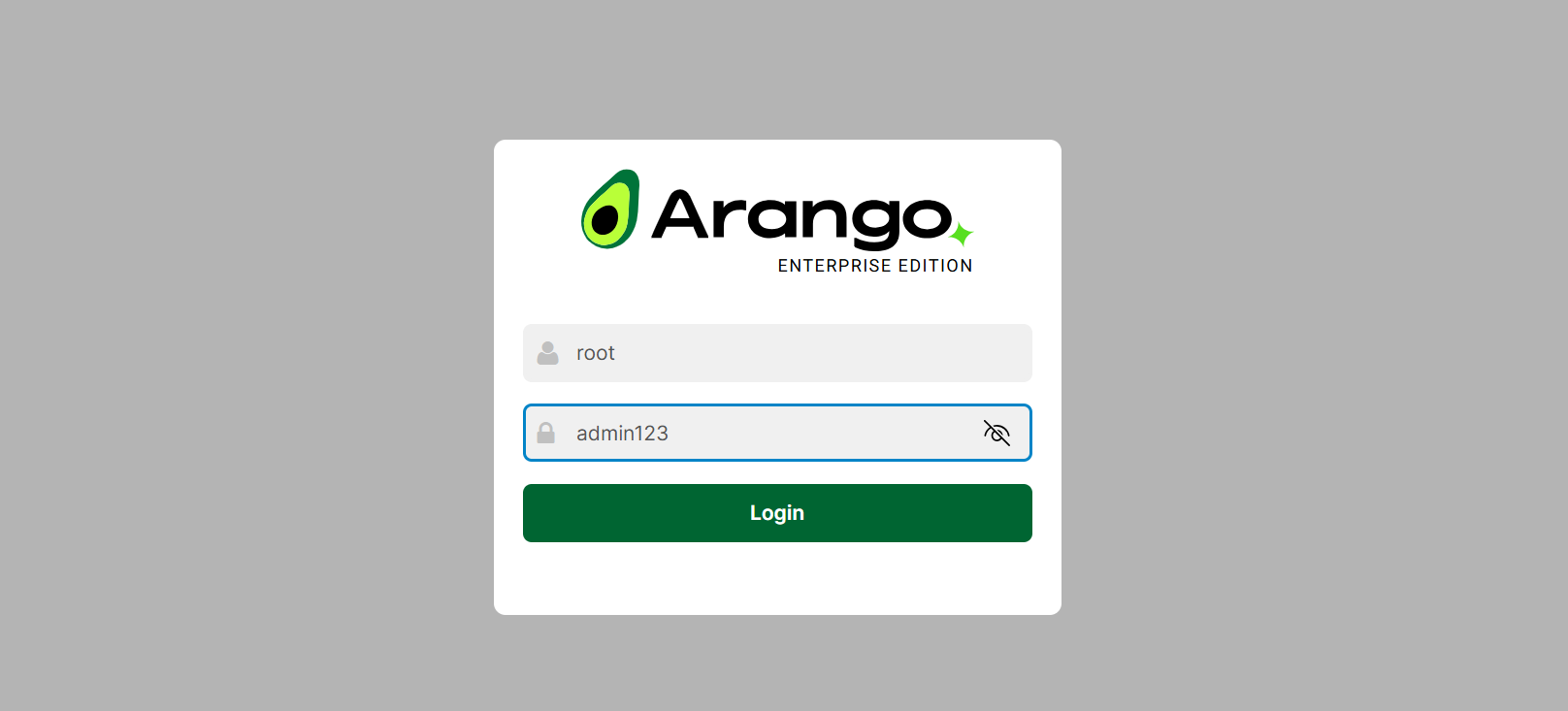
Hình : Tạo database thành công ở chế độ dòng lệnh

*Bước 4:* Mở Dockertop, tại port của bus-management-arangodb nhấn vào đó sẽ hiện ra trang login của ArangoDB



Hình : Giao diện Docker Desktop

*Bước 5:* Tại trang login đăng nhập *user root và password: admin123*

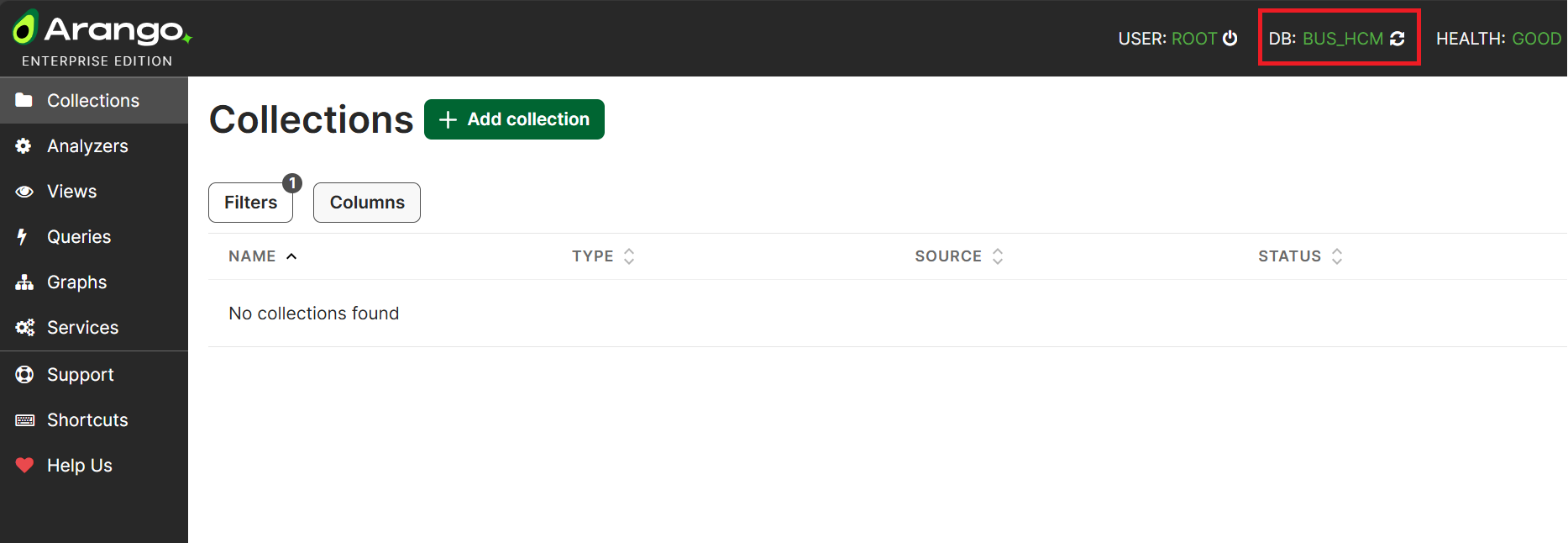


Hình : Giao diện đăng nhập của Arangodb

*Bước 6:* Chọn database cần quản lý và nhấn nút *“Select DB: bus\_hcm”*



Hình : Giao diện chọn database của Arangodb

**

Hình : Giao diện collections của Arangodb với database BUS\_HCM

# 4.2. Quản lý Collections (Document & Edge)

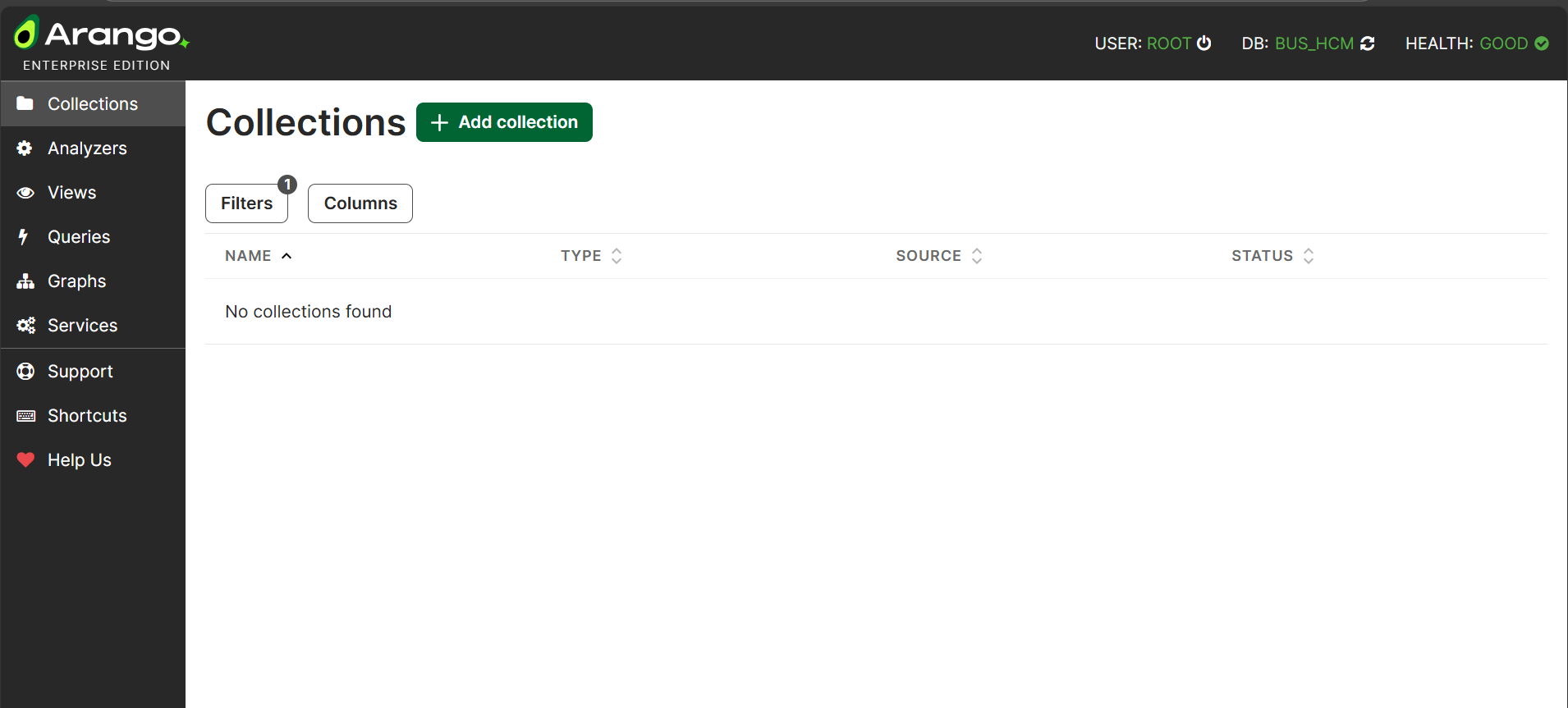
Trong ArangoDB, collection là một tập dữ liệu được dùng để lưu trữ các documents (tài liệu JSON) hoặc edges (các quan hệ trong đồ thị).  
 Nó tương đương với:

* Bảng trong SQL
* Collection trong MongoDB

Để bắt đầu tạo collections phải đã đăng nhập vào database cần tạo collection.

Có 2 cách tạo và quản lý collections

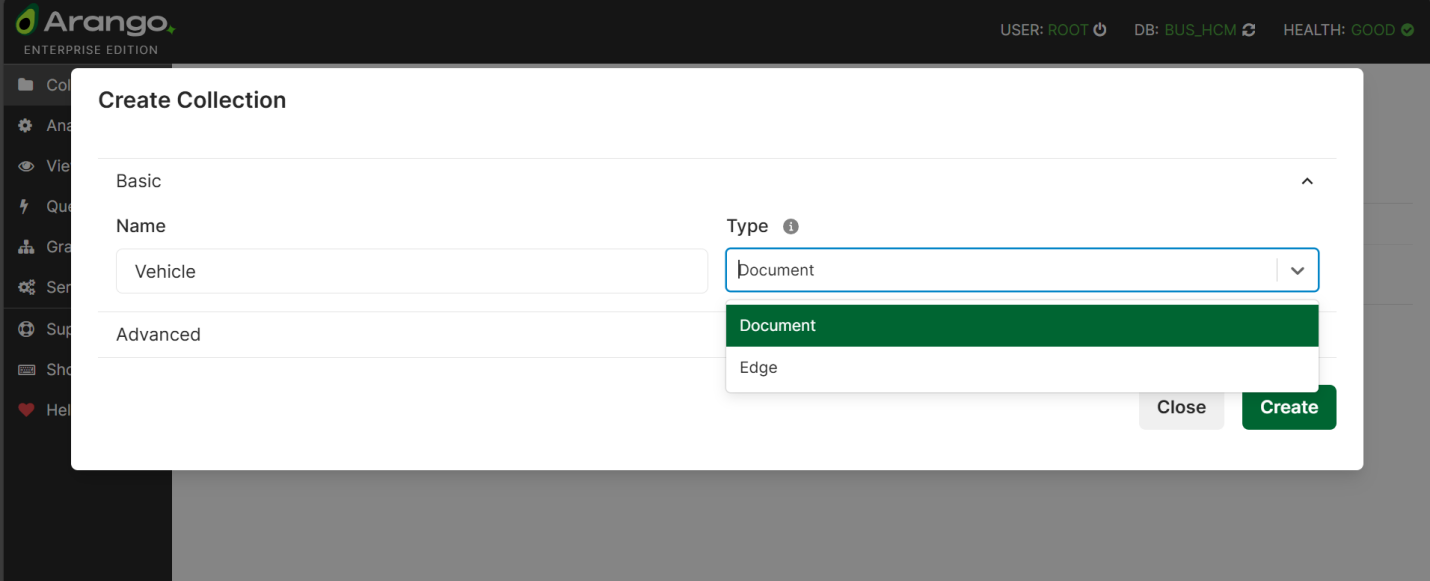
* *Cách 1: Tạo collections trên ArangoDb Web Interface - thực hiện các bước sau:*

*Bước 1:* Tại giao diện *Collections* của ArangoDB, chọn *“+Add collection”*

Hình : Giao diện collections chính với database BUS\_HCM

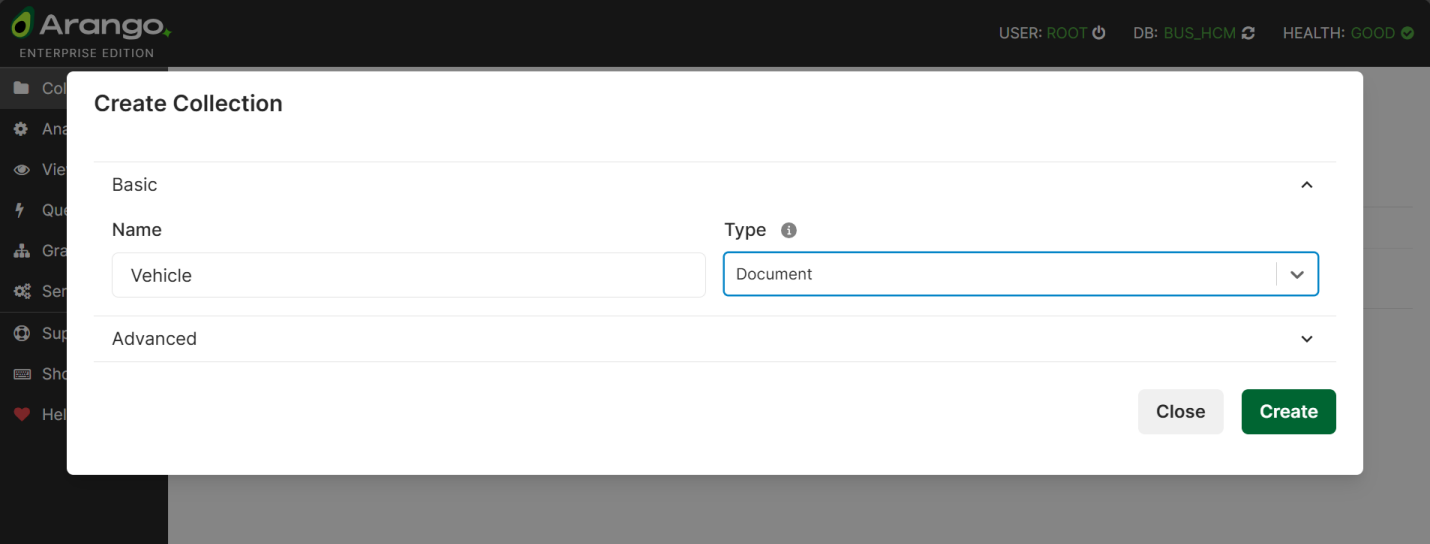
Bước 2: Tại mục *Create Collection*, đặt tên collection tại mục *name* - *type* có 2 loại Documents (tài liệu JSON), Edges ( quan hệ giữa dữ liệu)

Ví dụ: Đặt tên name là Vehicle, với type là document

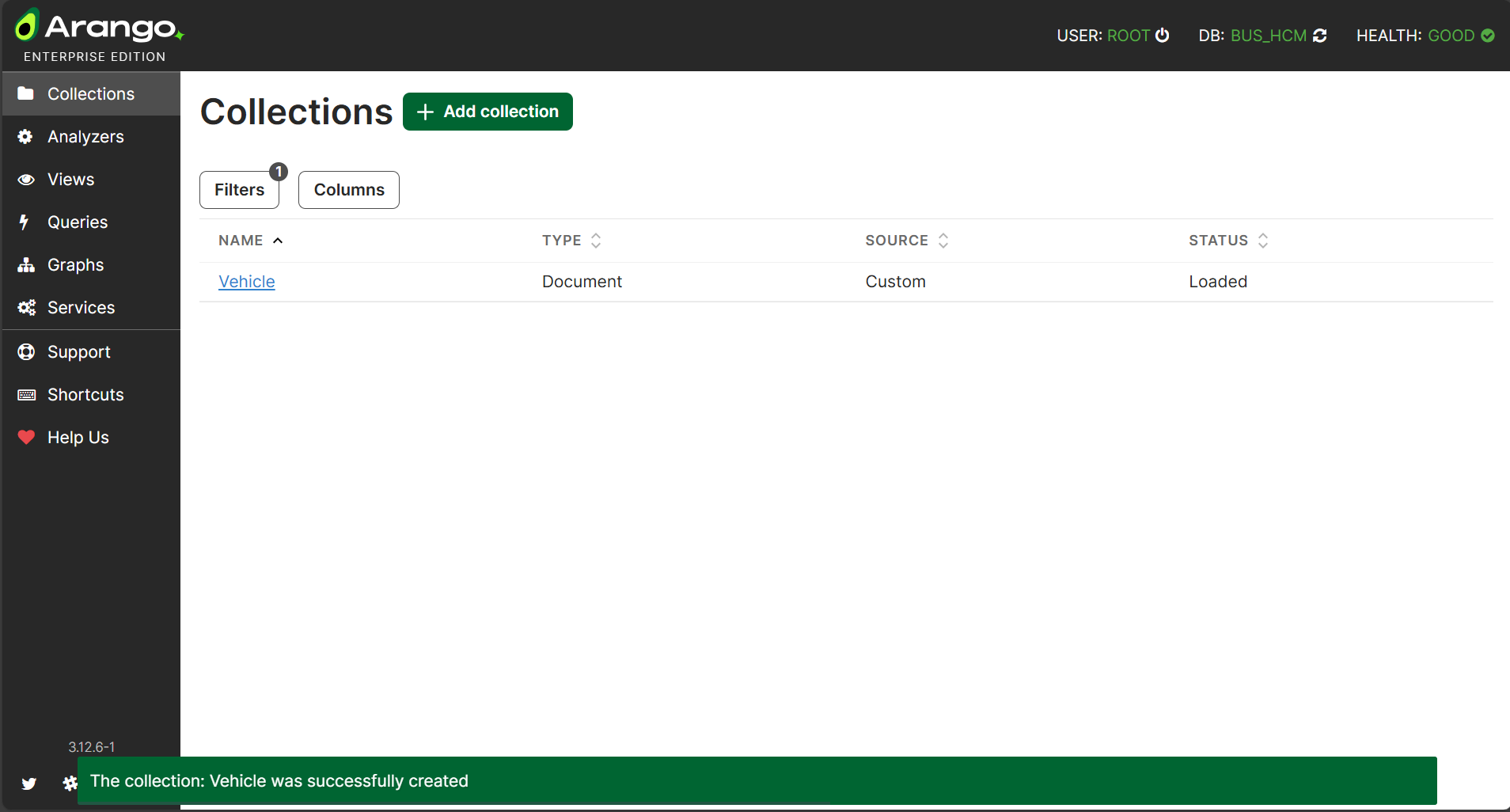


Hình : Collection ‘Vehicle’ với loại dữ liệu Document trong ArangoDB Web Interface

Bước 3: Nhấn nút *“Create”* để tiến hành tạo collection



Hình : Tạo collection ‘Vehicle’ trên ArangoDB Web Interface

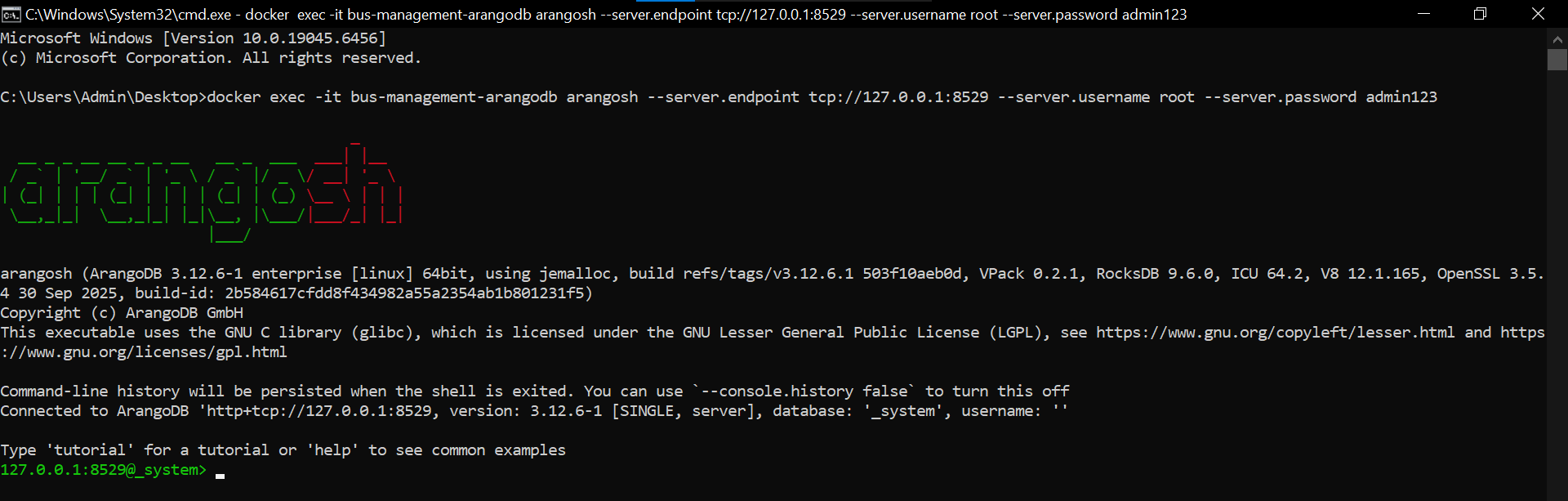


Hình : Giao diện Collections hiển thị collection 'Vehicle' sau khi tạo thành công

* *Cách 2:Tạo collection trên Terminal/Command Prompt - thực hiện các bước sau:*

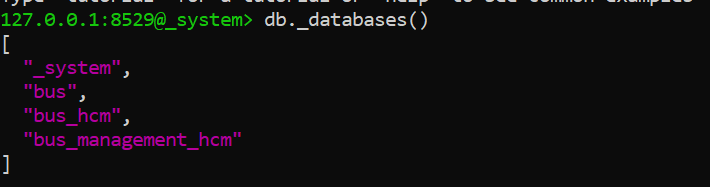
*Bước 1:* Mở Terminal/Command Prompt - mở một cửa sổ Terminal (trên Linux/macOS) hoặc Command Prompt/PowerShell (trên Windows)

*Bước 2:* Tại cmd chạy *“docker exec -it bus-management-arangodb arangosh --server.endpoint tcp://127.0.0.1:8529 --server.username root --server.password admin123”*



Hình : Kết nối với ArangoDB ở chế độ dòng lệnh trên cmd

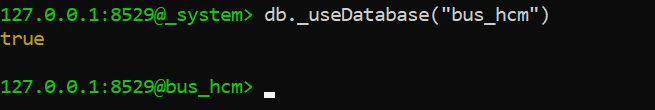
Bước 3: Chạy lệnh *“db.\_databases()”* để kiểm tra tất cả các database của Arangodb



Hình : Kết quả hiển thị danh sách database ở chế độ dòng lệnh

Bước 4: Chạy lệnh *“db.\_useDatabase(“ten\_database”)”* để truy cập vào đúng database

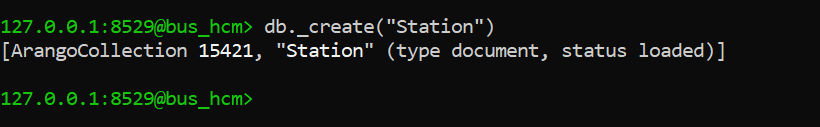
Ví dụ: Thực thi lệnh *“db.\_useDatabase(“bus\_hcm”)”* - để truy cập vào database bus\_hcm



Hình : Chuyển sang database bus\_hcm bằng arangosh

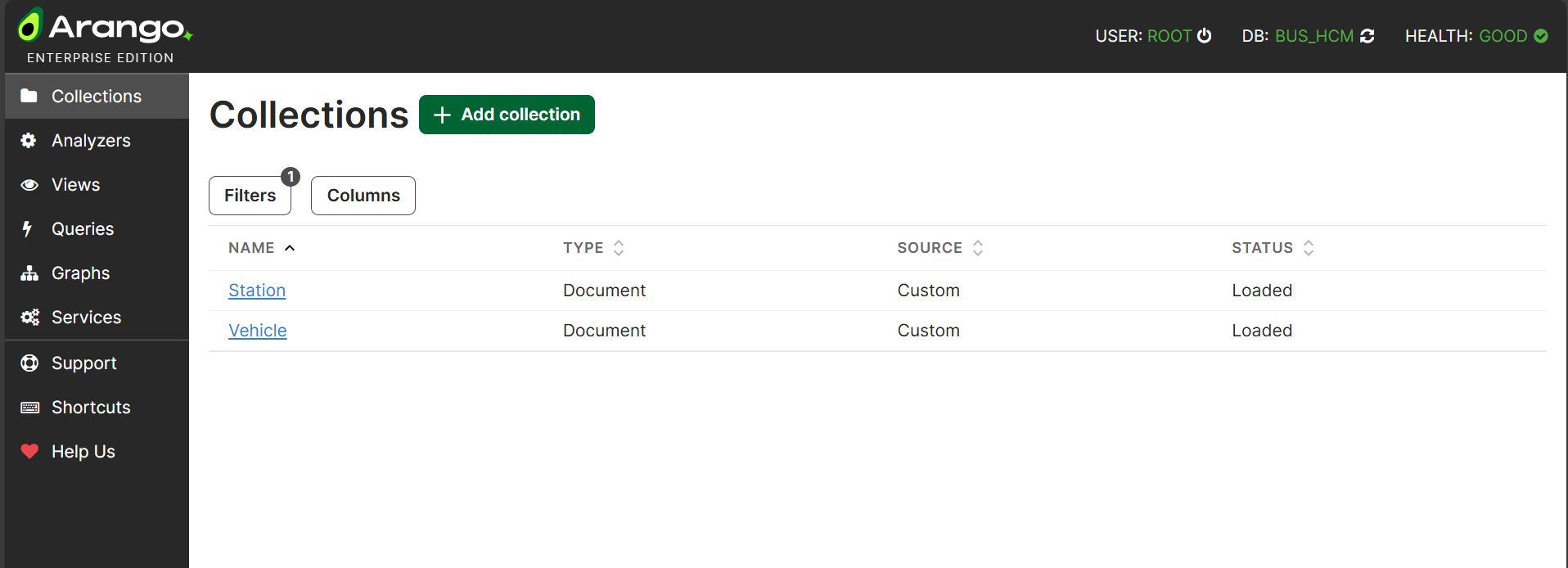
Bước 5: Chạy lệnh *“db.\_create(“ten\_collection”)”* để tạo 1 collection

Ví dụ: Thực thi lệnh *“db.\_create(“Station”)”* để tạo collection tên Station



Hình : Tạo collection Station ở chế độ dòng lệnh

Bước 6: Đăng nhập vào database *“bus\_hcm”,* tại mục *Collections* hiển thị danh mục collection đã được tạo



Hình : Danh sách collections sau khi tạo thành công

# 4.3. Quản lý Graph

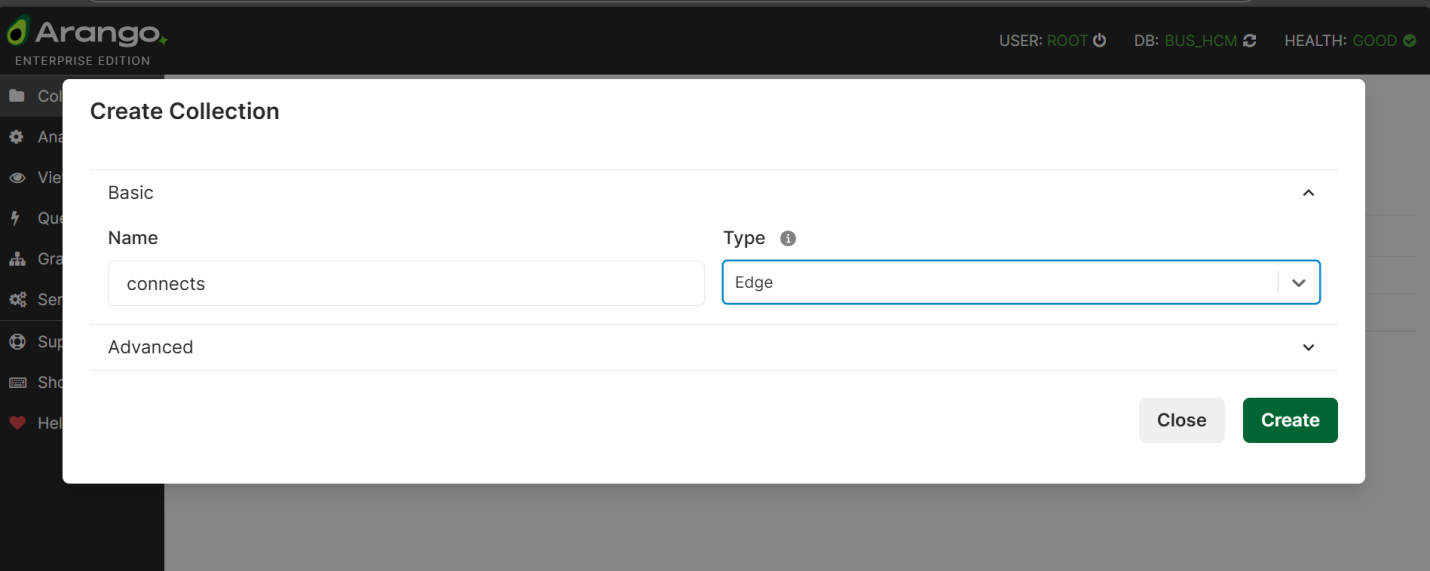
Trong ArangoDB, Graph (đồ thị) là một mô hình dữ liệu dùng để biểu diễn các đối tượng và mối quan hệ giữa chúng một cách trực quan và linh hoạt. Một graph bao gồm hai thành phần chính: vertex (đỉnh, lưu trong các *document collections*) và edge (cạnh, lưu trong *edge collections*). Vertex đại diện cho các thực thể như trạm xe buýt, tuyến xe, phương tiện… trong khi edge thể hiện mối quan hệ giữa các thực thể này, ví dụ: “tuyến đi qua trạm”, “xe thuộc tuyến”, hoặc “trạm kết nối với trạm”.

Để bắt đầu tạo graph phải đã đăng nhập vào database và đã tạo collections

Có 2 cách tạo và quản lý graph

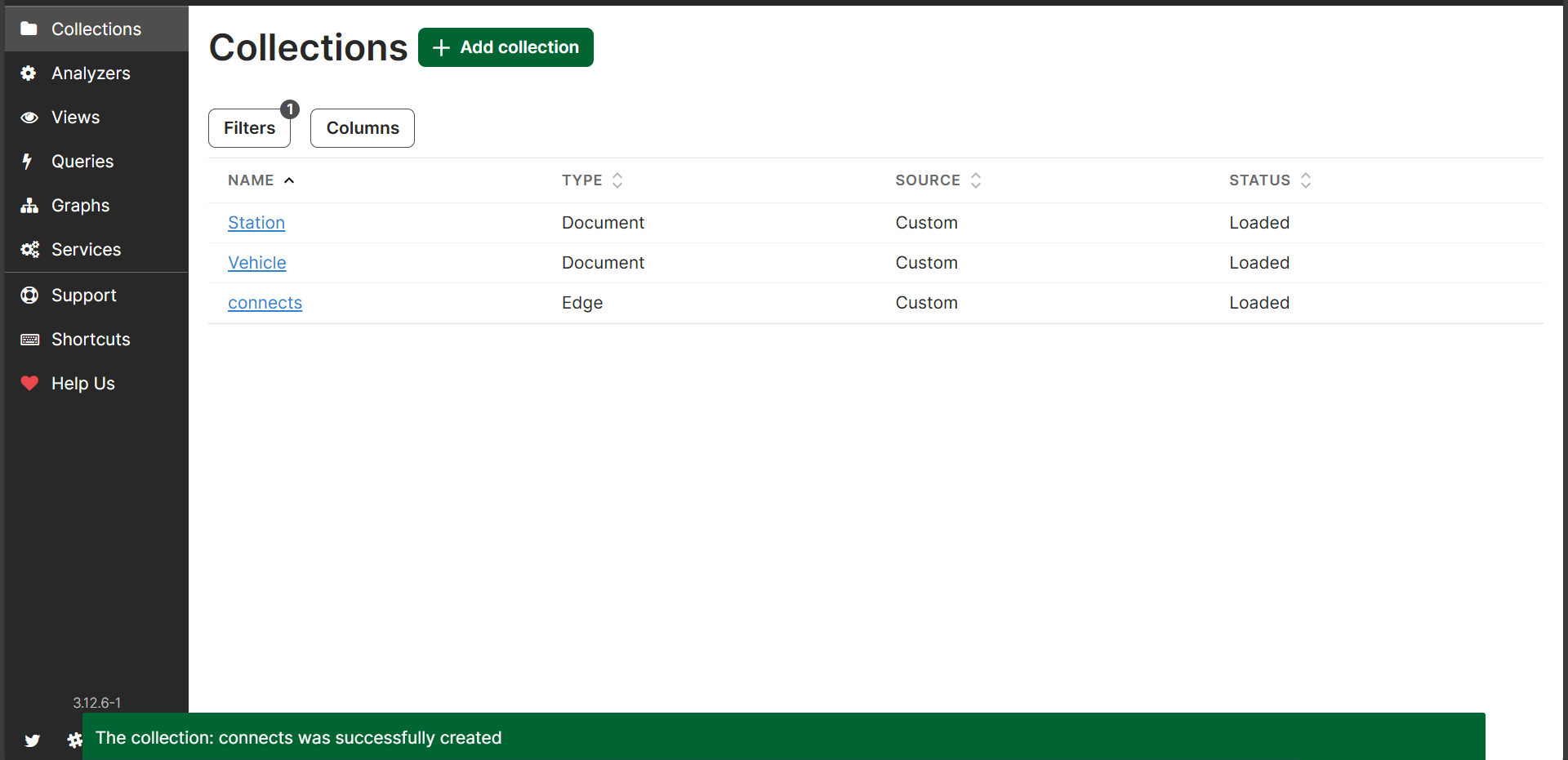
* *Cách 1: Tạo graph trên ArangoDb Web Interface - thực hiện các bước sau:*

*Bước 1:* Tại giao diện *Collections,* tạo 1 collection với type *Edge*

**

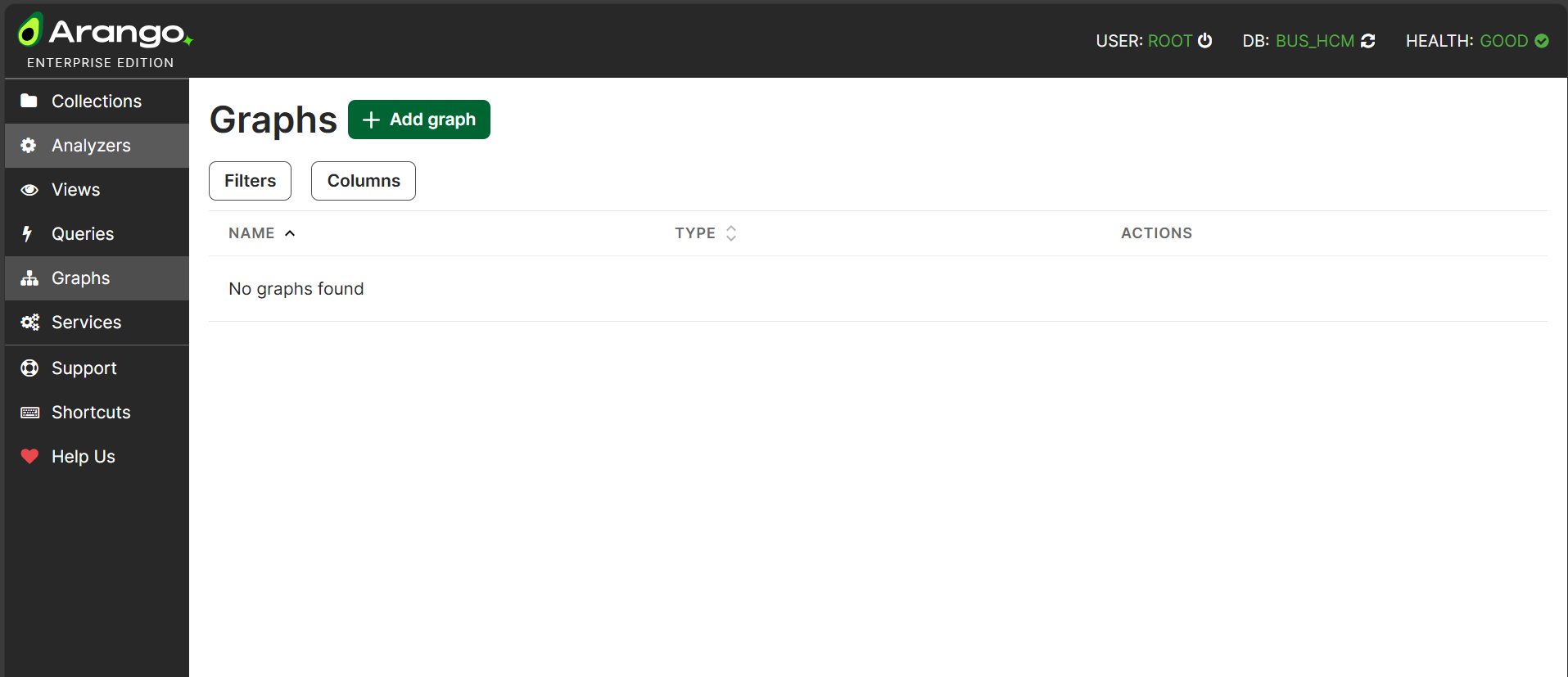
Hình : Tạo collection với type edge

*Bước 2:* Nhấn nút “*Create”,* để tạo thành công collection

**

Hình : Danh sách collection tại mục Collections

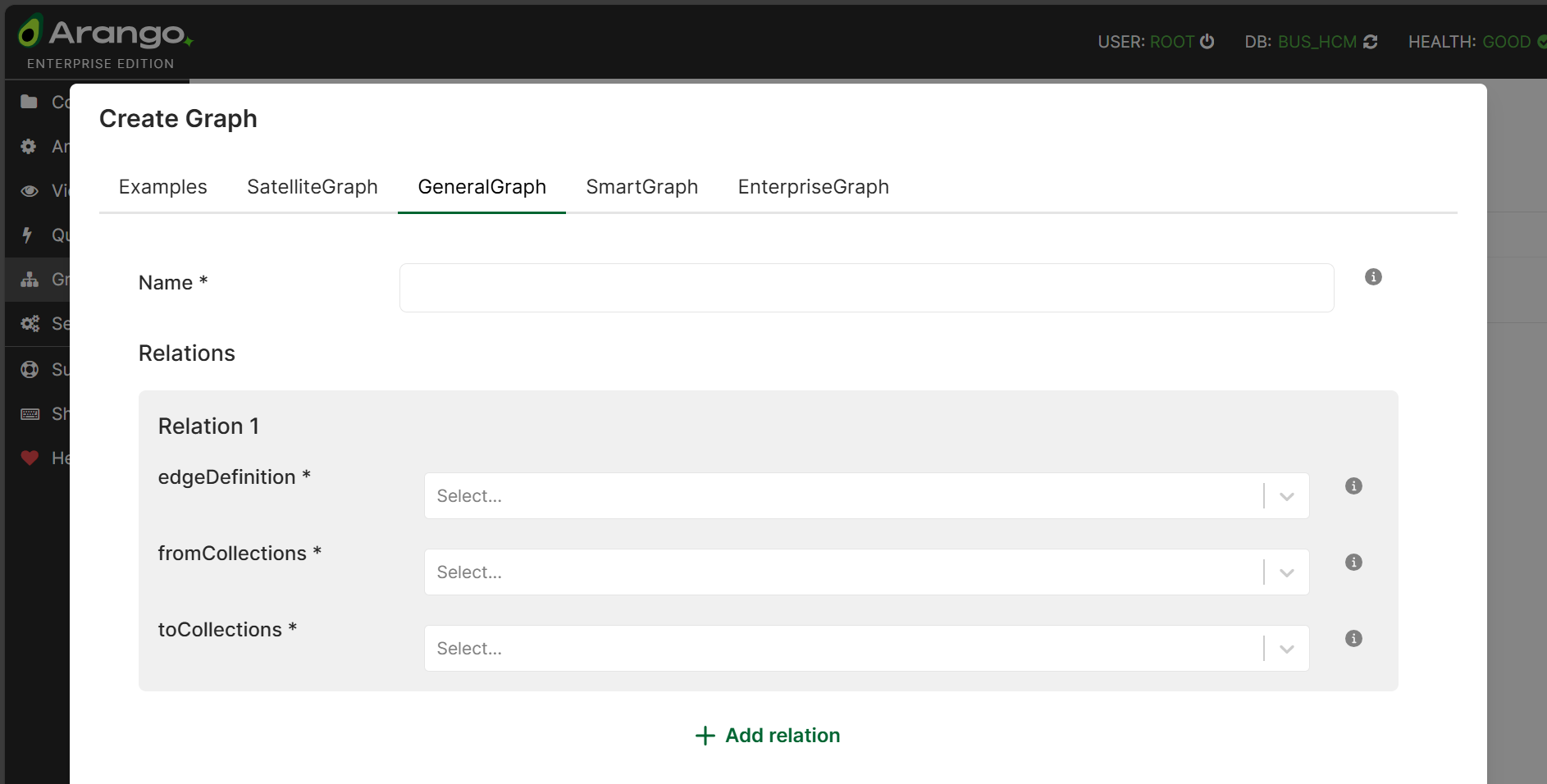
*Bước 4:* Tại giao diện *Graphs* của Arango, nhất nút*“+Add graph”*



Hình : Giao diện Graphs trên ArangoDb Web Interface

Bước 5: Tại giao diện *Create Graph,* chọn tab *“General Graph”*

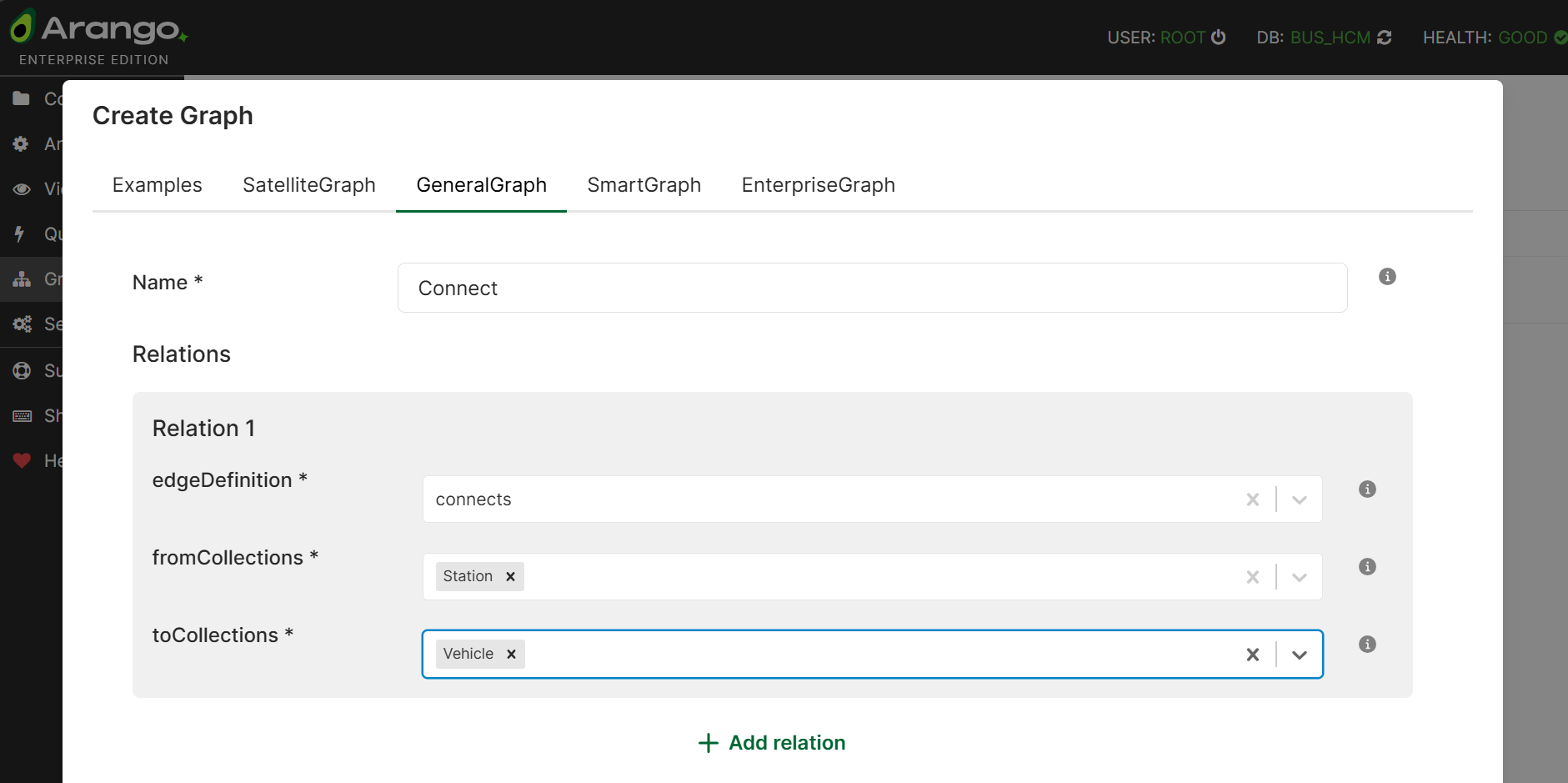
* Examples: *Đ*ây là graph mẫu do ArangoDB tạo sẵn - dùng để học, tham khảo cấu trúc graph.
* Satellite Graph:Chỉ có ý nghĩa khi chạy ArangoDB trong cluster (nhiều server).
* General Graph: Đây là loại graph cơ bản nhất của ArangoDB - đồ thị chuẩn, dùng trong hầu hết các ứng dụng.
* SmartGraph: SmartGraph là graph tối ưu trong phiên bản Enterprise + Cluster.
* Enterprise Graph:Chỉ dùng cho enterprise deployment.



Hình : Giao diện Create Graphs trên ArangoDb Web Interface

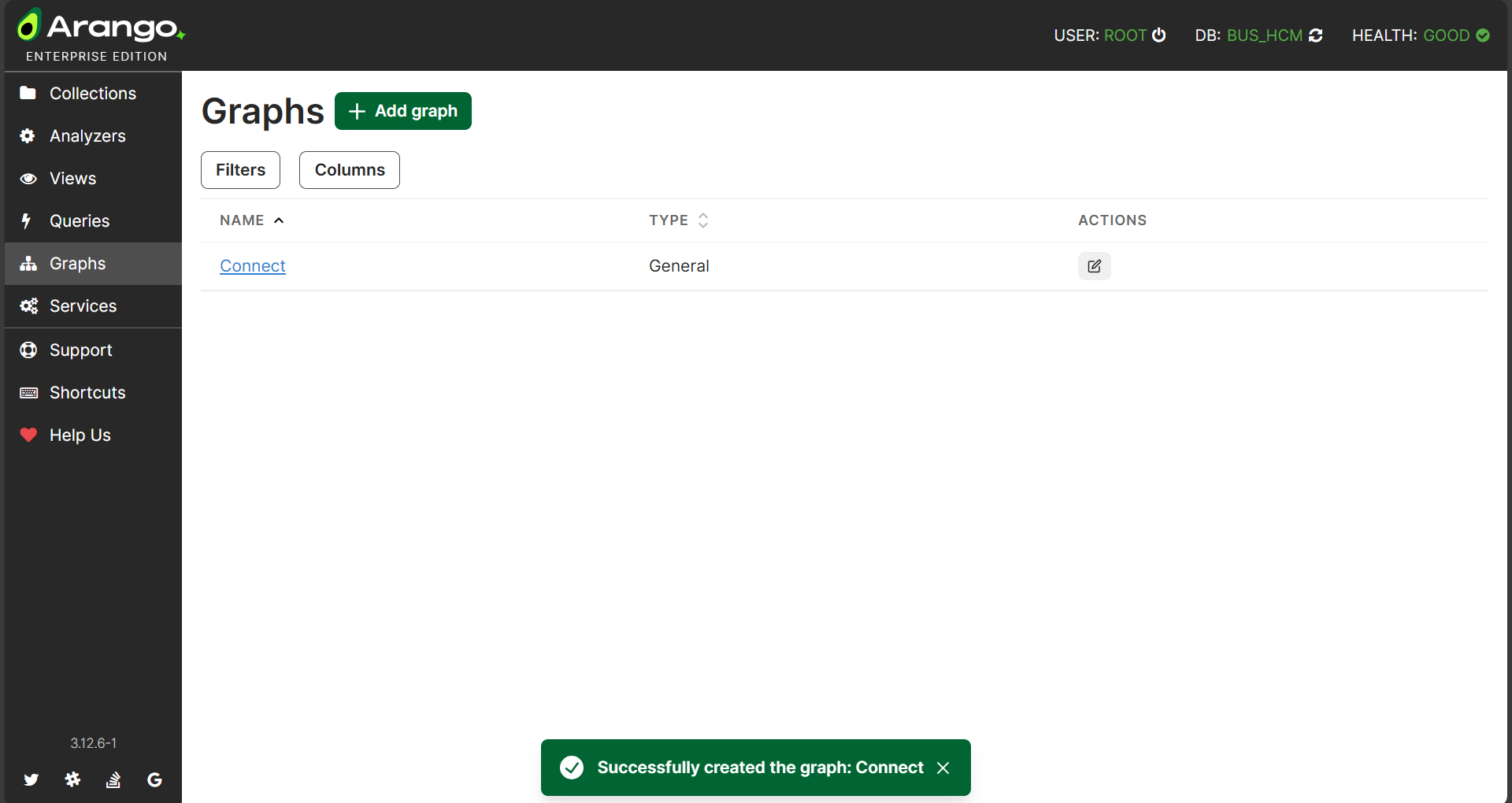
*Bước 6:* Nhập *name* và các mục của *relation*  và nhấn nút *“Create”*

Lưu ý: Nếu muốn thêm relation, thì chọn nhấn *“ + Add relation”*



Hình : Thực hiện tạo Graphs trên ArangoDb Web Interface

*Bước 7:* Danh sách các graphs đã tạo

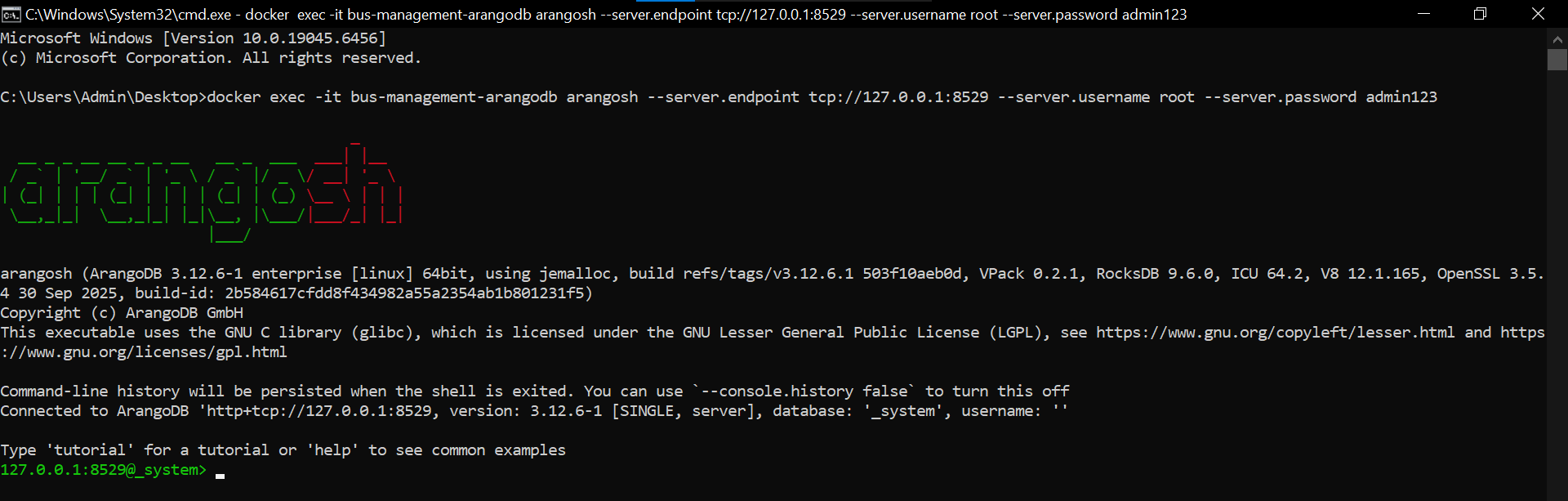


Hình : Danh sách Graphs trên ArangoDb Web Interface

* *Cách 2:Tạo collection trên Terminal/Command Prompt - thực hiện các bước sau:*

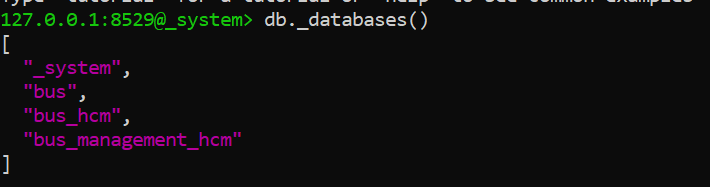
*Bước 1:* Mở Terminal/Command Prompt - mở một cửa sổ Terminal (trên Linux/macOS) hoặc Command Prompt/PowerShell (trên Windows)

*Bước 2:* Tại cmd chạy *“docker exec -it bus-management-arangodb arangosh --server.endpoint tcp://127.0.0.1:8529 --server.username root --server.password admin123”*



Hình : Kết nối với ArangoDB ở chế độ dòng lệnh trên cmd

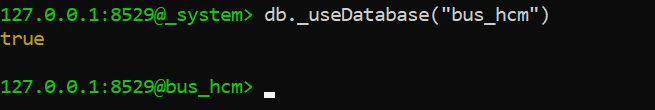
Bước 3: Chạy lệnh *“db.\_databases()”* để kiểm tra tất cả các database của Arangodb



Hình : Kết quả hiển thị danh sách database ở chế độ dòng lệnh

Bước 4: Chạy lệnh *“db.\_useDatabase(“ten\_database”)”* để truy cập vào đúng database

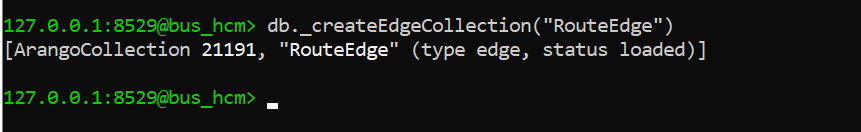
Ví dụ: Thực thi lệnh *“db.\_useDatabase(“bus\_hcm”)”* - để truy cập vào database bus\_hcm



Hình : Chuyển sang database bus\_hcm bằng arangosh

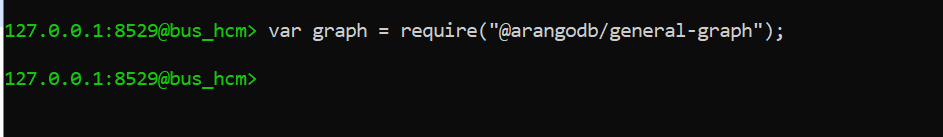
Bước 5: Chạy lệnh *“db.\_createEdgeCollection(“ten\_collectionEdge”)”* để tạo 1 collection với type Edge

Ví dụ: Thực thi lệnh *“db.\_createEdgeCollection(“RouteEdge”)”* để tạo collection với type tên RouteEdge



Hình : Tạo collection RouteEdge với type Edge ở chế độ dòng lệnh

Bước 6: Nạp module quản lý Graph



Hình : Nạp module tạo Graph trong ArangoDB

Bước 7: Tạo graph với tên “BusNetworkGranp” với dòng lệnh

var graph = require("@arangodb/general-graph");

graph.\_create(

"BusNetworkGraph",

[

{

collection: "RouteEdge",

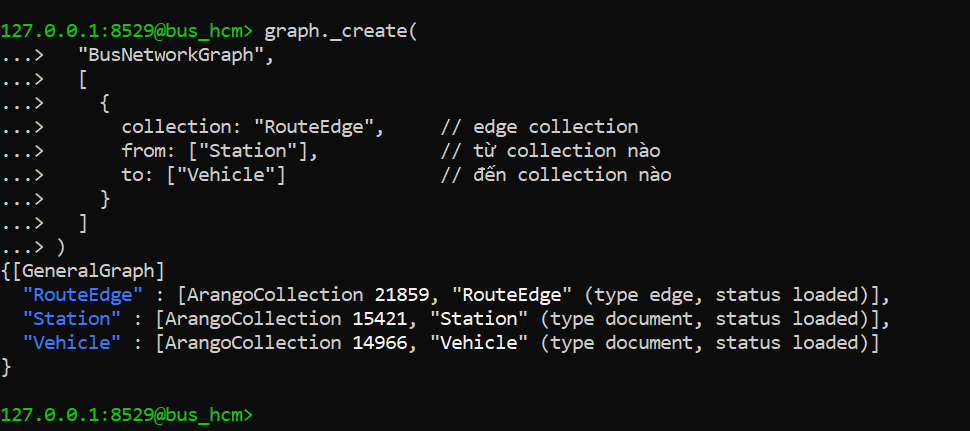
from: ["Station"],

to: ["Vehicle"]

}

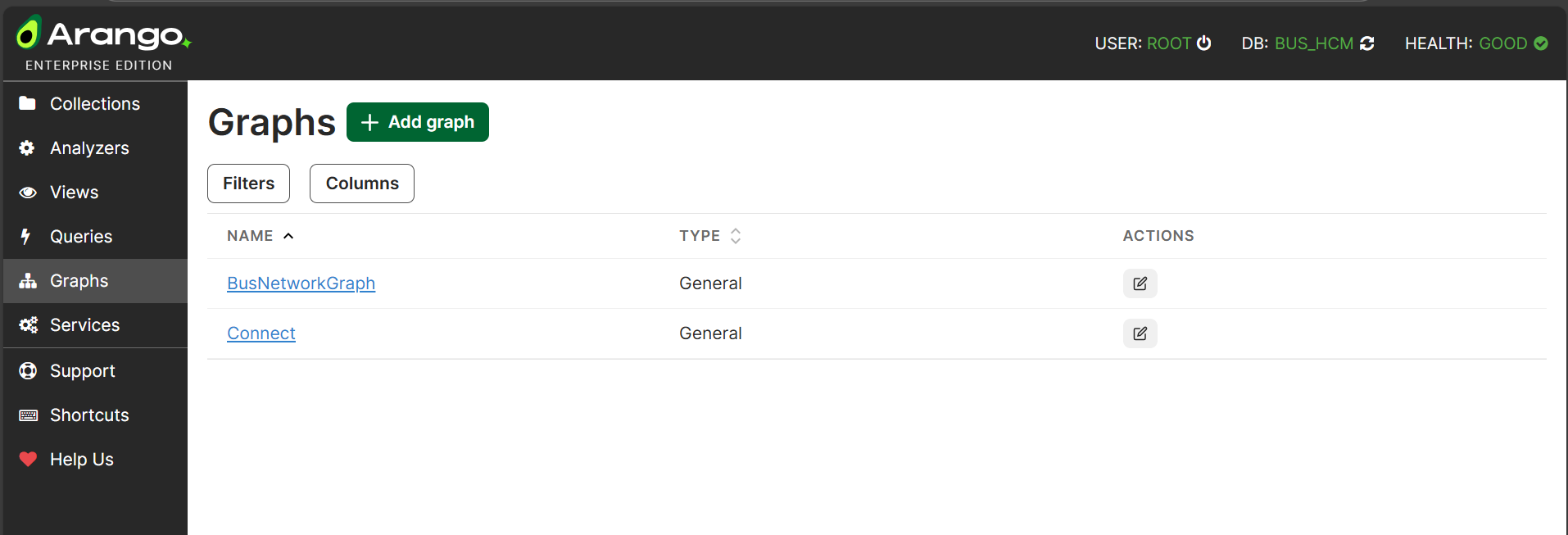
]

)



Hình: Cấu trúc tạo Graph trong CMD

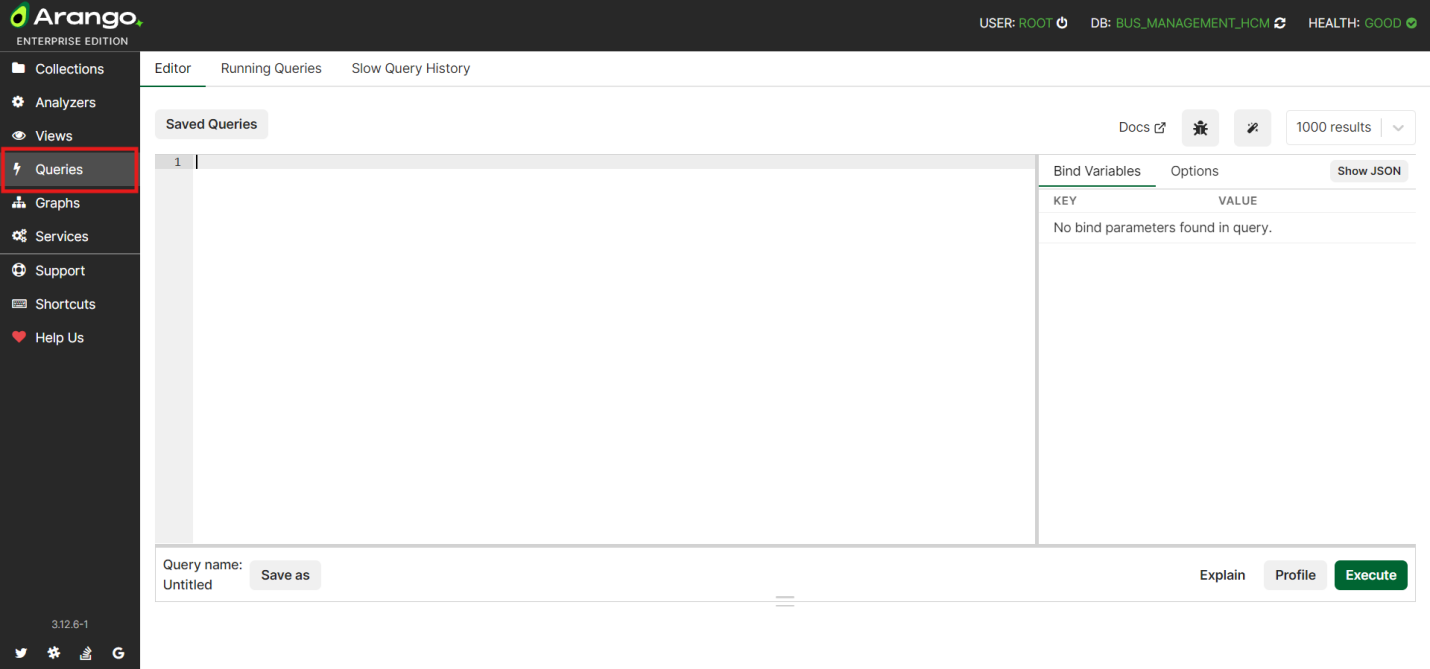
Bước 8: Đăng nhập vào ArangoDb Web Interface vào mục *Graph*, hiển thị danh sách graph đã tạo



Hình : Danh sách Graphs của db bus\_hcm trên ArangoDb Web Interface

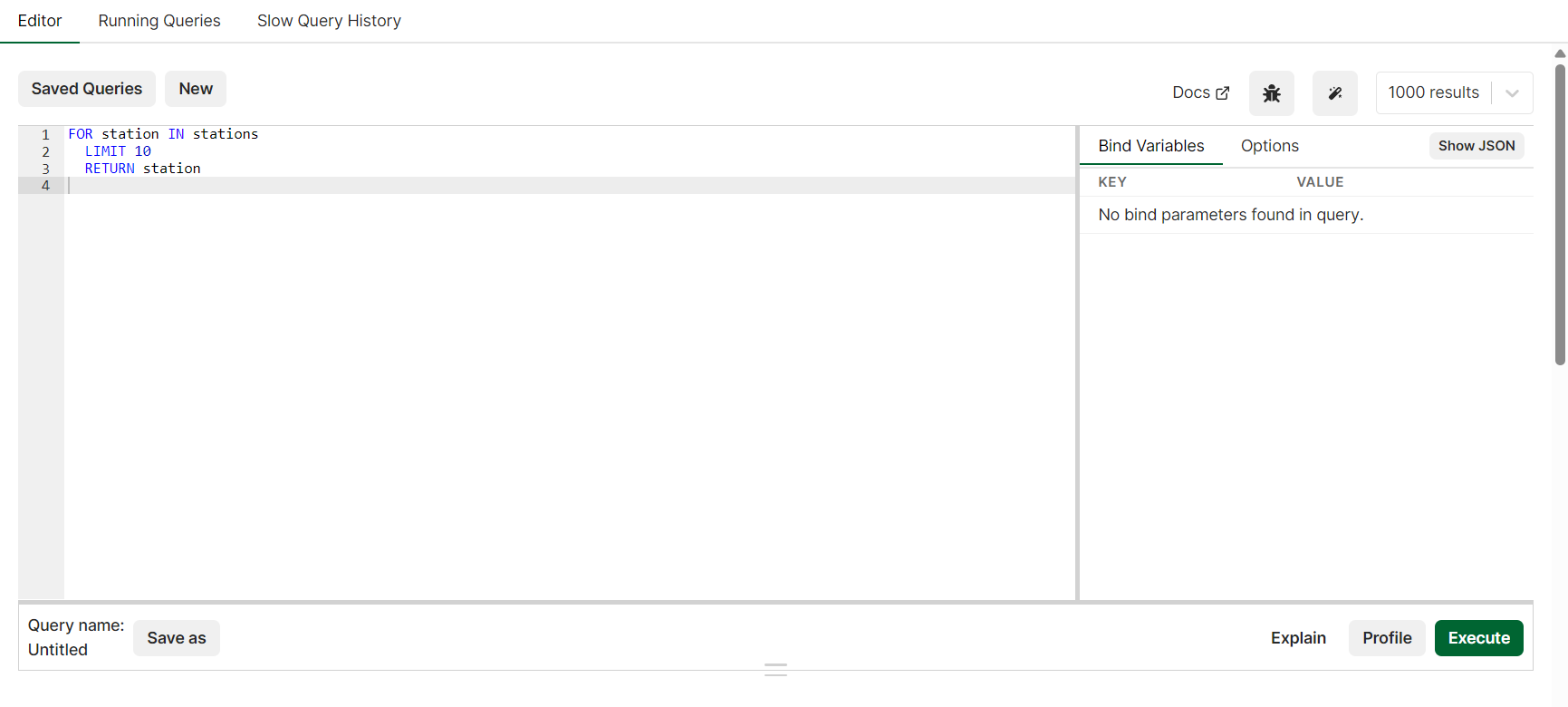
4.4. AQL Query Editor (Trình truy vấn dữ liệu)  
AQL Query Editor là một trong những công cụ mạnh mẽ và được sử dụng thường xuyên nhất trong ArangoDB Web Interface. Nó cung cấp một môi trường tương tác để viết, thực thi, phân tích và gỡ lỗi các truy vấn AQL (ArangoDB Query Language). Đây là công cụ không thể thiếu trong quá trình phát triển, giúp lập trình viên kiểm thử logic truy vấn trước khi tích hợp vào ứng dụng backend.  
*A. Giao diện chính của AQL Query Editor*Để truy cập, ta chọn tab QUERIES từ menu bên trái trong Web Interface. Giao diện chính được chia thành các khu vực chức năng rõ ràng:

* Vùng soạn thảo (Query Editor): Nơi để viết các câu lệnh AQL. Trình soạn thảo này hỗ trợ tô sáng cú pháp (syntax highlighting), tự động hoàn thành (autocomplete) tên các collection, hàm AQL, giúp việc viết query trở nên nhanh chóng và chính xác hơn.
* Vùng kết quả (Result): Hiển thị kết quả trả về sau khi thực thi truy vấn. Kết quả có thể được xem dưới nhiều định dạng khác nhau.
* Bảng Bind Parameters: Cho phép định nghĩa các biến ràng buộc (@variable) để truyền tham số vào truy vấn một cách an toàn, tránh được các lỗi tấn công AQL Injection.

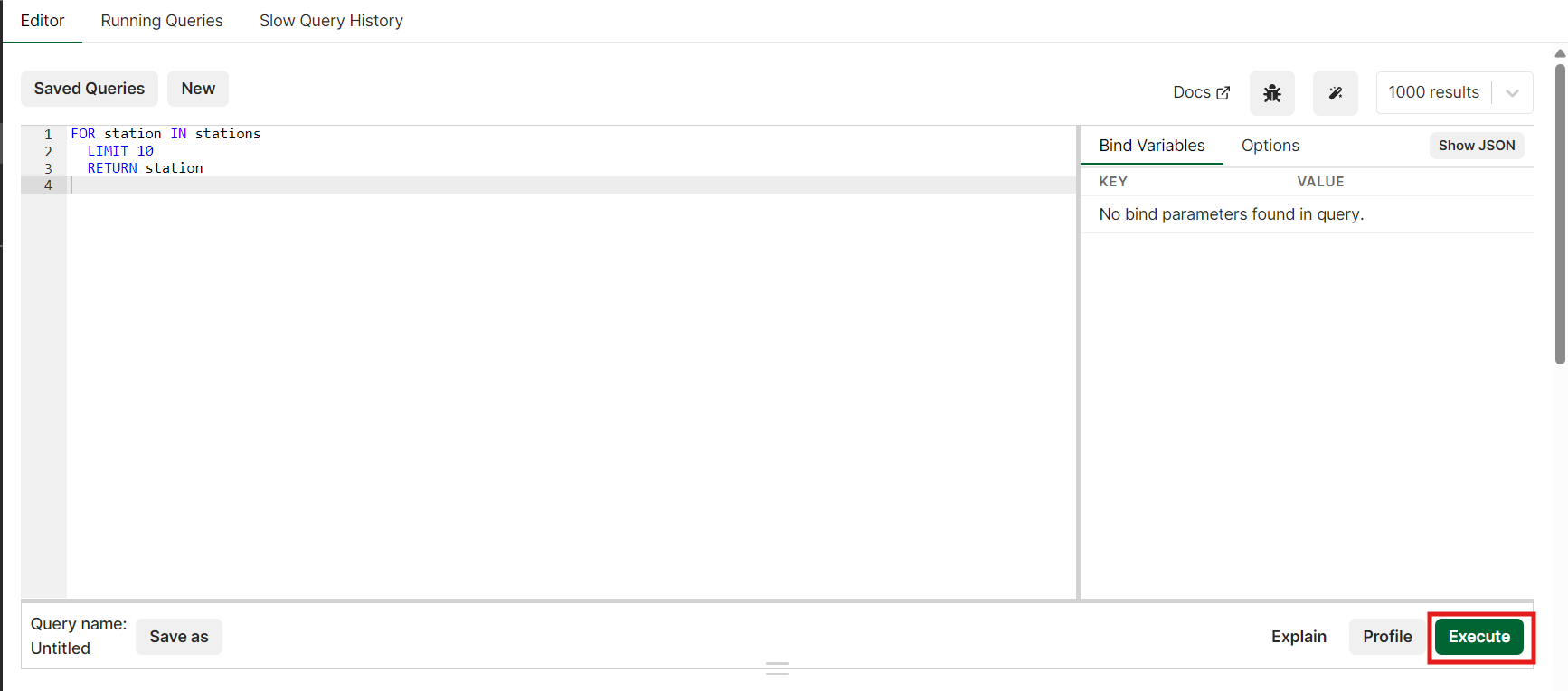
  
Các nút chức năng:

* Execute: Chạy truy vấn hiện tại.
* Explain: Phân tích và hiển thị kế hoạch thực thi (execution plan) của truy vấn mà không thực sự chạy nó.
* Save: Lưu truy vấn lại để tái sử dụng sau này.
* Query History: Xem lại lịch sử các truy vấn đã thực thi.

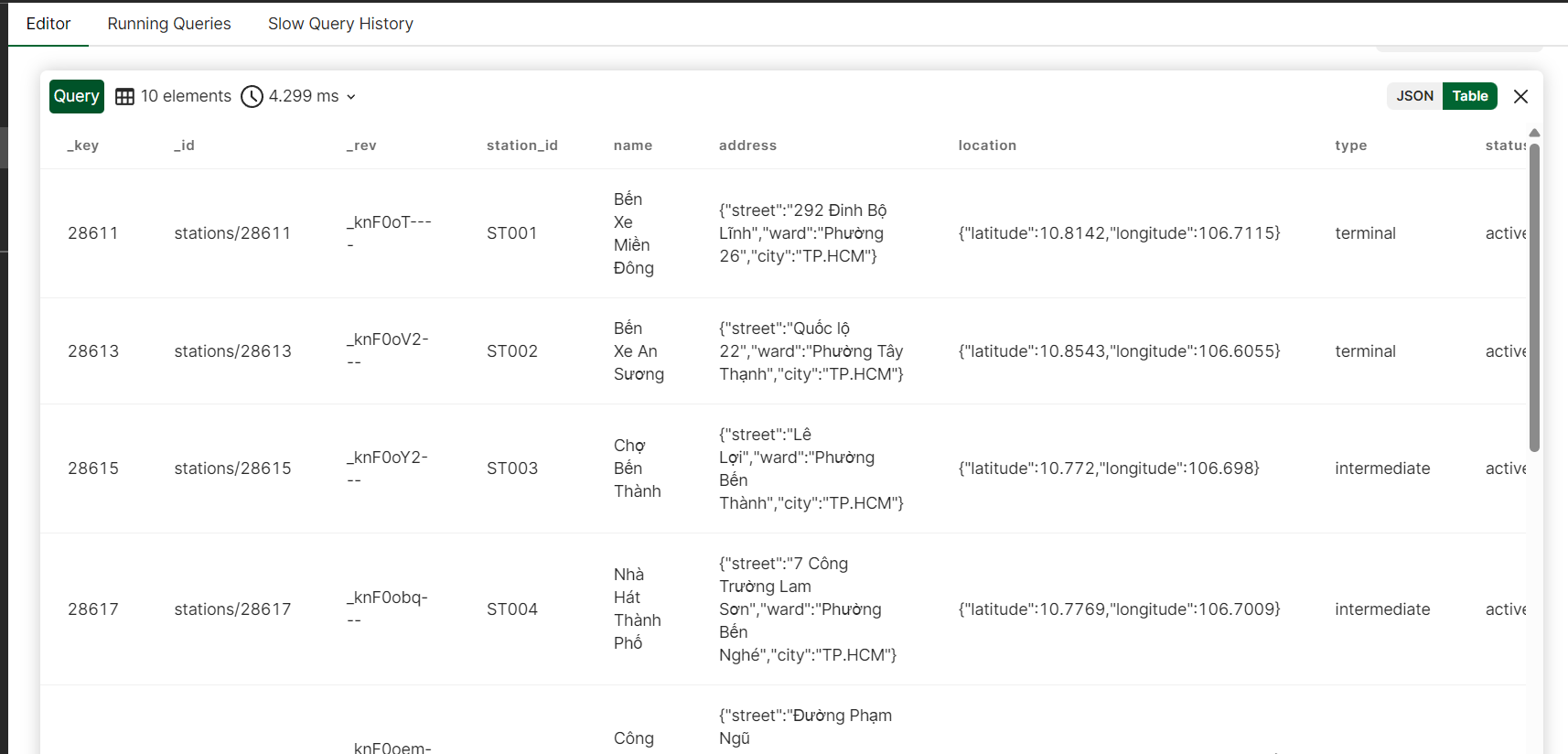
*B. Thực thi một truy vấn cơ bản*Bước 1: Viết truy vấn  
Trong vùng soạn thảo, nhập một câu lệnh AQL đơn giản. Ví dụ, để lấy 10 trạm xe buýt đầu tiên:  
*FOR station IN stations  
LIMIT 10  
RETURN station*

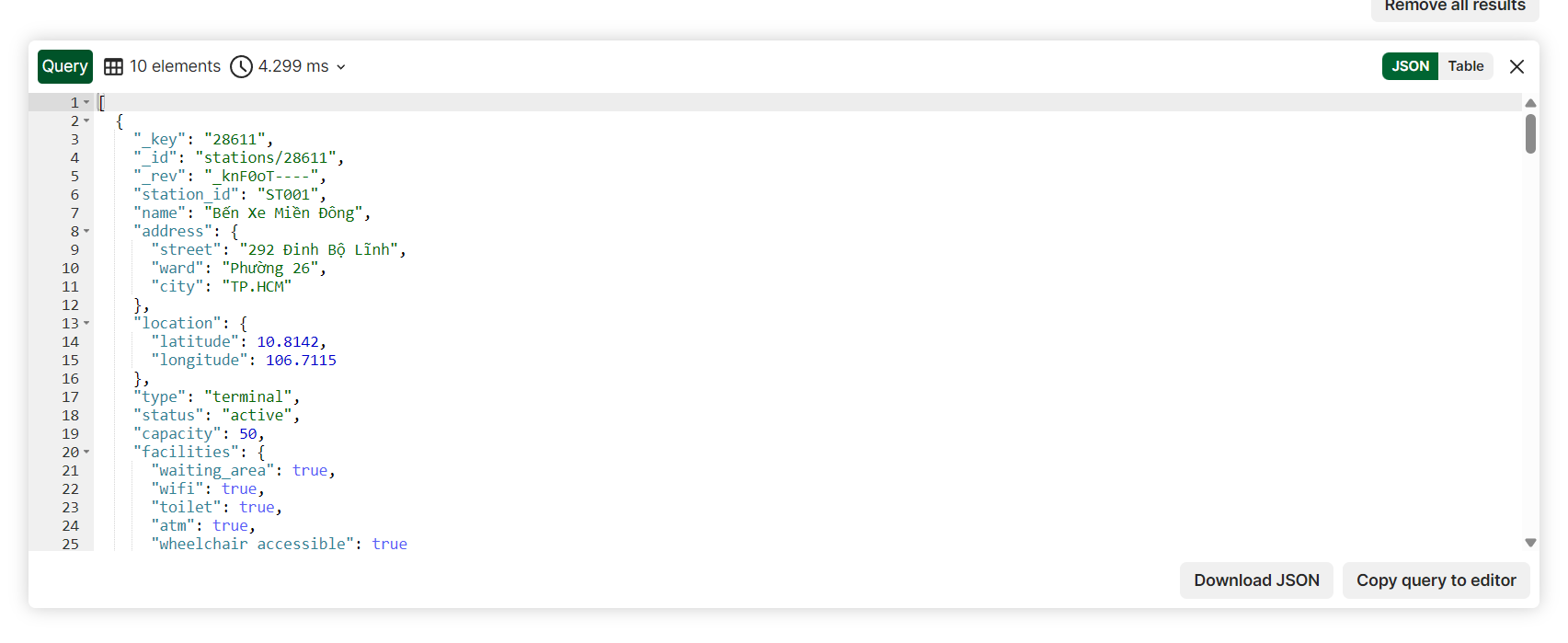
**

Bước 2: Thực thi  
Nhấn nút Execute (hoặc dùng phím tắt Ctrl + Enter).

Bước 3: Xem kết quả  
Kết quả sẽ được hiển thị ngay lập tức trong vùng bên dưới. Bạn có thể chuyển đổi giữa các chế độ xem:

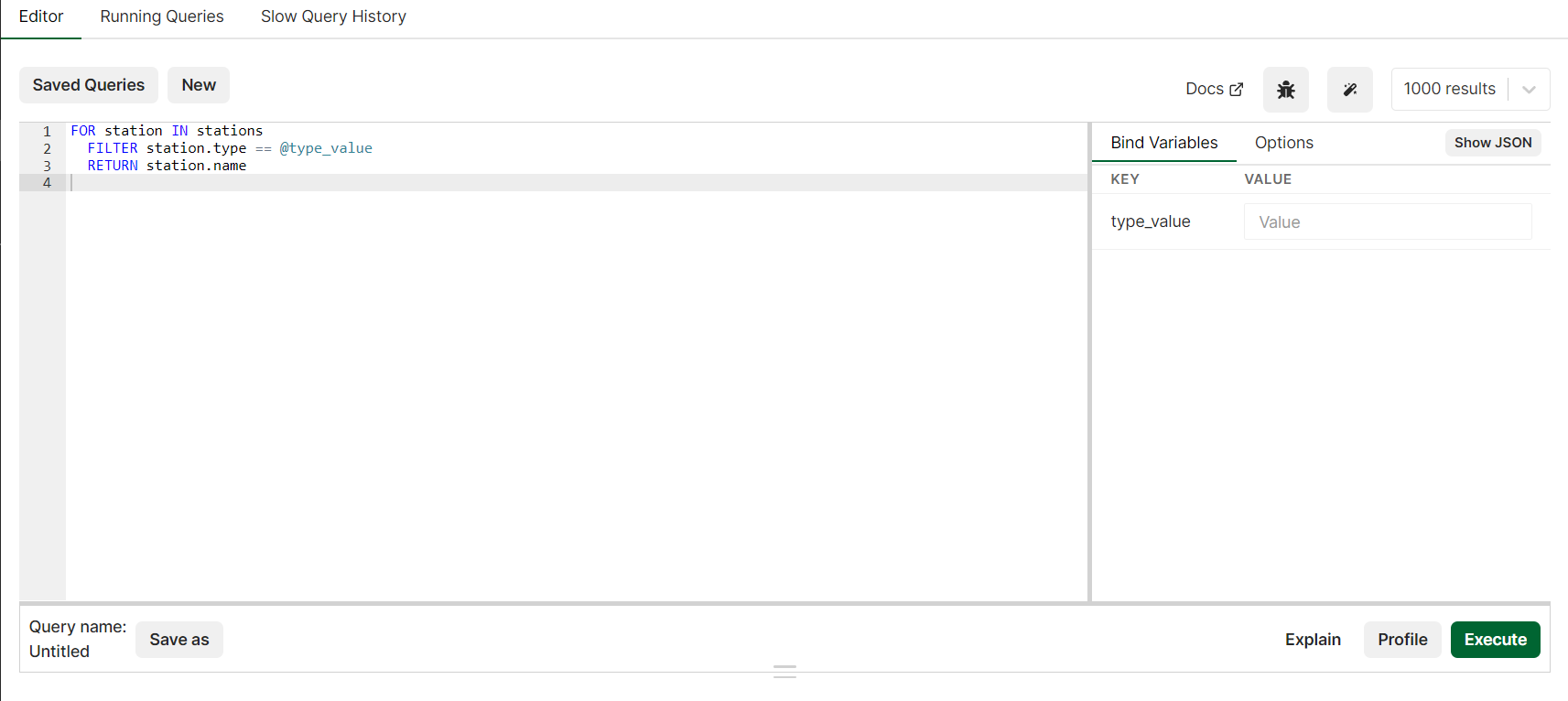
* JSON: Hiển thị kết quả dưới dạng JSON, dễ đọc và copy.
* Table: Hiển thị kết quả dưới dạng bảng, phù hợp cho dữ liệu có cấu trúc phẳng.
* Graph: Nếu kết quả trả về là một đồ thị (các đỉnh và cạnh), chế độ này sẽ trực quan hóa đồ thị đó. Đây là tính năng cực kỳ hữu ích để gỡ lỗi các truy vấn traversal.



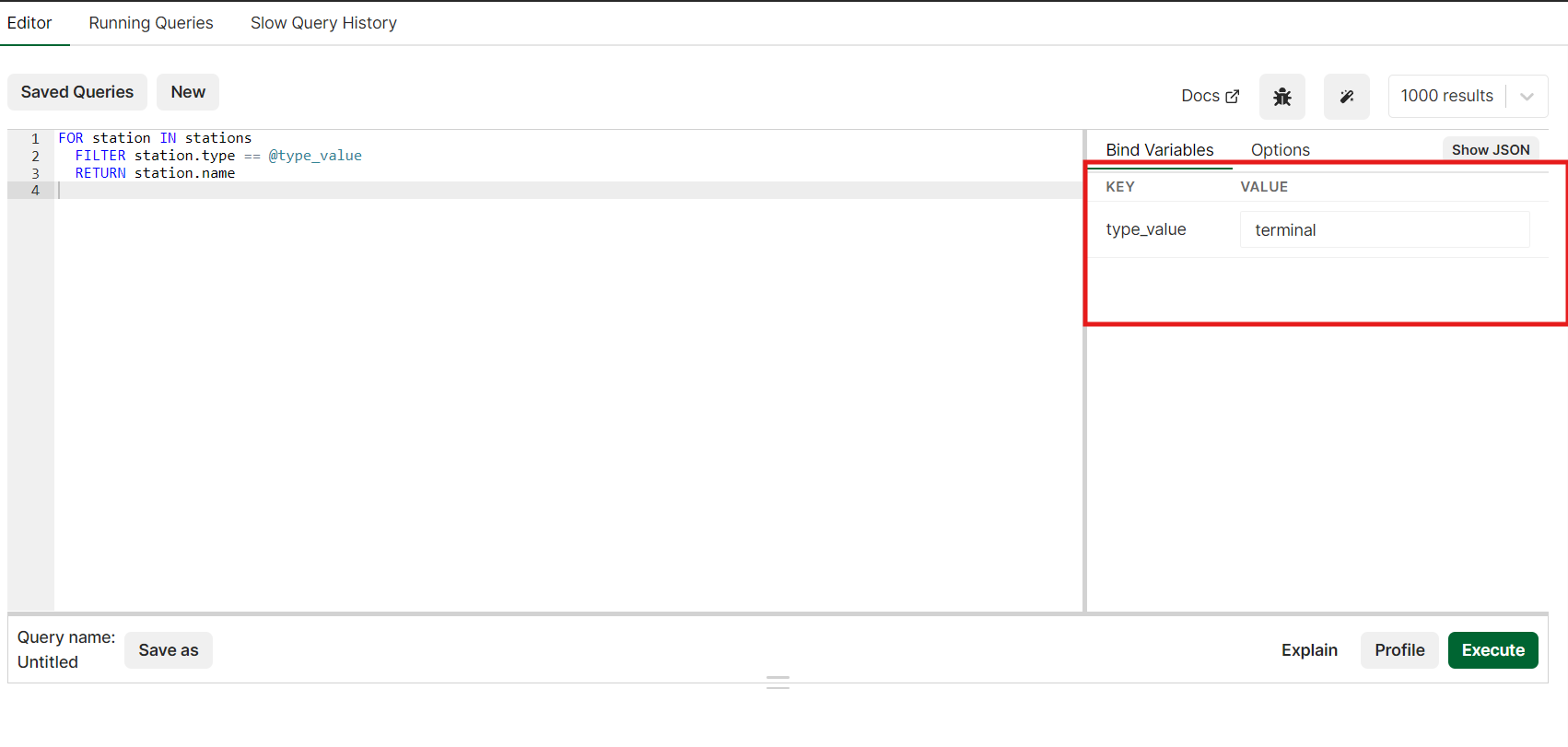


*C. Sử dụng Bind Parameters để tăng tính bảo mật và tái sử dụng*Việc nối chuỗi để tạo câu truy vấn là một hành vi không an toàn. Bind Parameters cho phép truyền giá trị vào truy vấn một cách an toàn.  
Bước 1: Viết truy vấn với biến ràng buộc

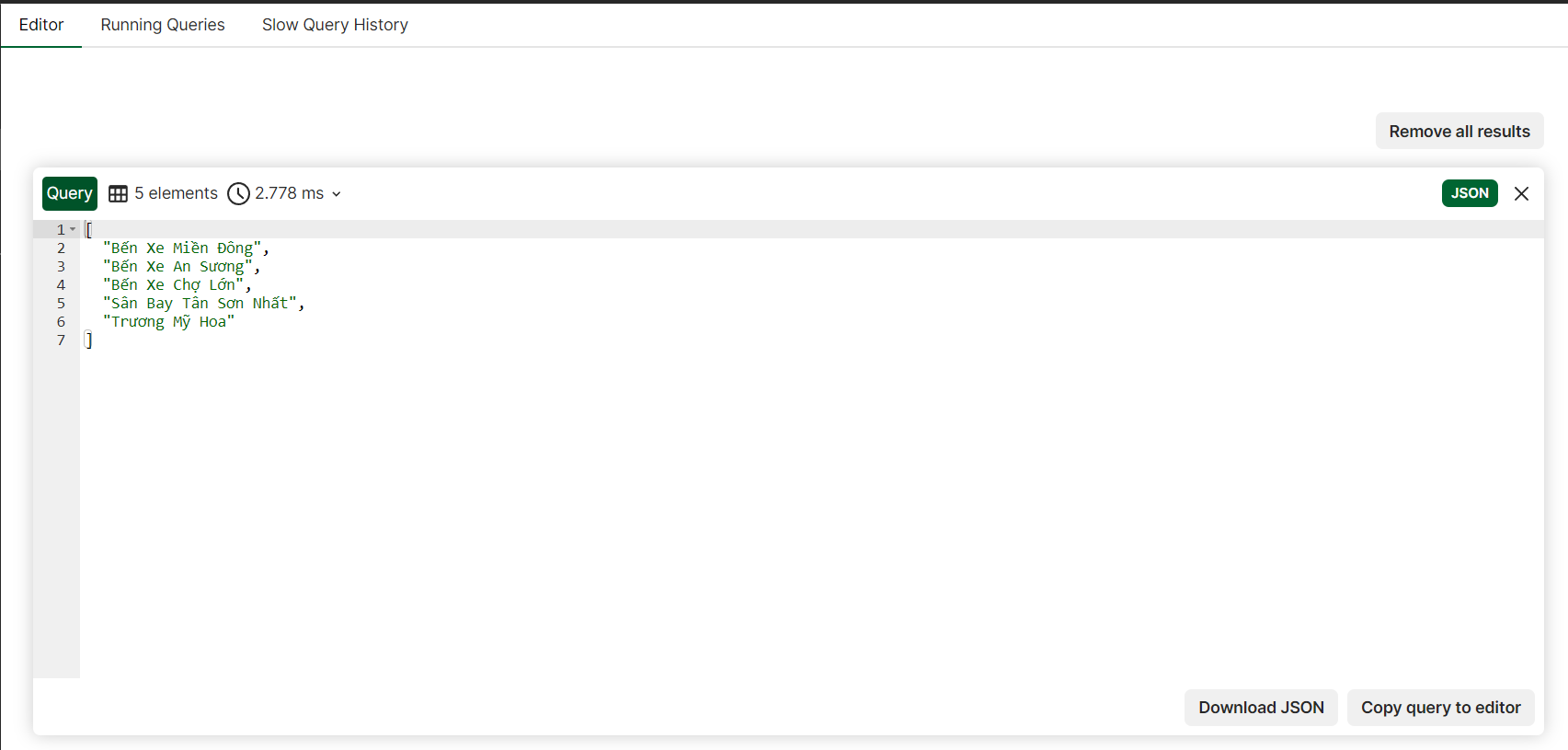
Viết một truy vấn tìm kiếm trạm xe buýt theo loại, sử dụng biến @type\_value.  
*FOR station IN stations  
FILTER station.type == @type\_value  
RETURN station.name*



Bước 2: Định nghĩa giá trị cho biến  
Trong bảng Bind Parameters, nhập type\_value vào cột Key và terminal vào cột Value.



Bước 3: Thực thi  
Nhấn Execute. ArangoDB sẽ thay thế @type\_value bằng giá trị terminal một cách an toàn và thực thi truy vấn.

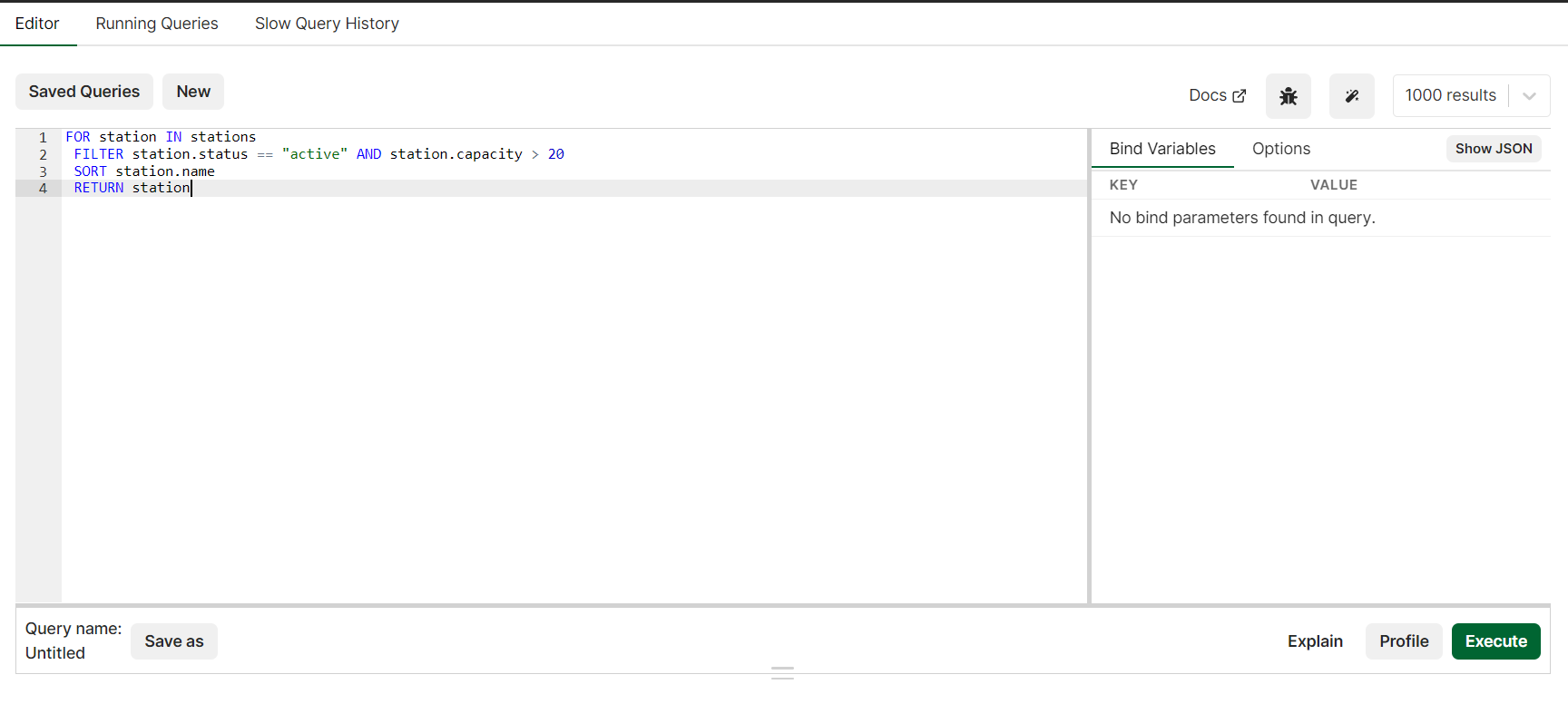


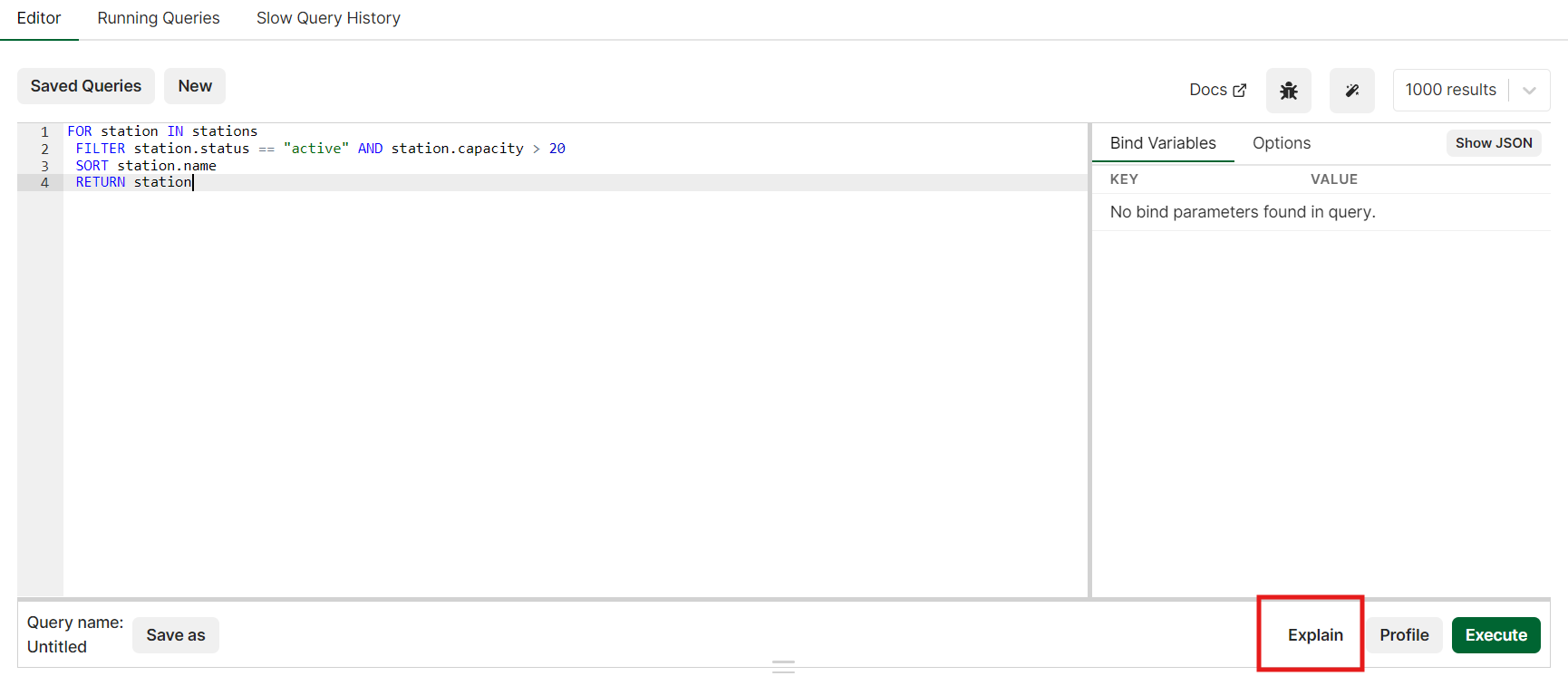
*D. Phân tích và Tối ưu hóa truy vấn với*Tính năng Explain là công cụ mạnh mẽ nhất để hiểu cách ArangoDB thực thi một câu truy vấn và tìm ra điểm nghẽn về hiệu năng.  
**Bước 1**: Nhập truy vấn cần phân tích  
Ví dụ, một truy vấn lọc phức tạp:  
*FOR station IN stations*

*FILTER station.status == "active" AND station.capacity > 20*

*SORT station.name*

*RETURN station*

**Bước 2**: Nhấn nút   
Thay vì chạy, ArangoDB sẽ phân tích và hiển thị Execution Plan (Kế hoạch thực thi).

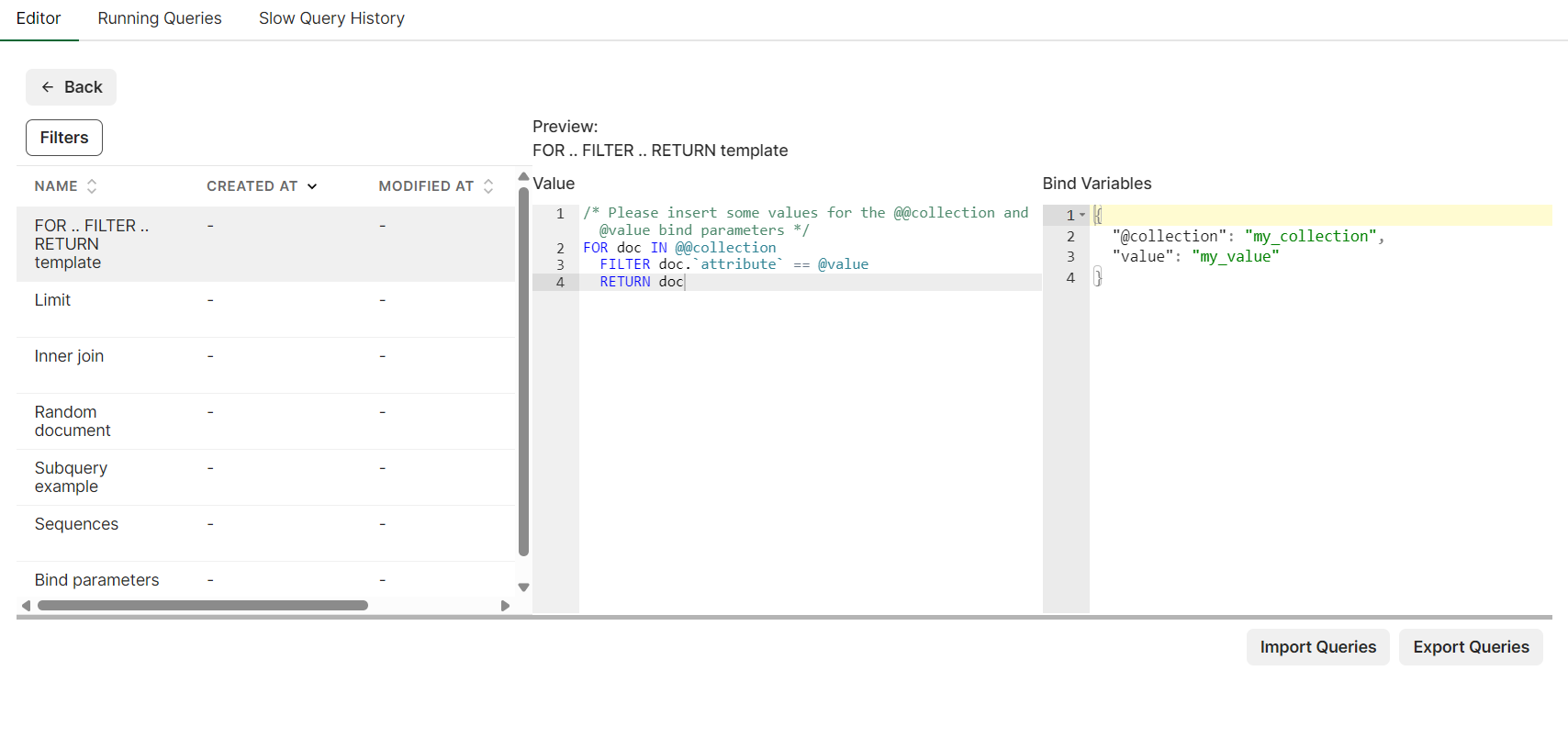
  
 Hình : Kế hoạch thực thi (Execution Plan)   
**Bước 3**: Đọc và phân tích kế hoạch



Kế hoạch thực thi cho biết:

* Các bước thực thi (Nodes): ArangoDB sẽ làm gì tuần tự (ví dụ: quét toàn bộ collection, lọc, sắp xếp).
* Số lượng item ước tính: Số lượng document được xử lý ở mỗi bước.
* Sử dụng Index: Quan trọng nhất, nó cho biết truy vấn có đang sử dụng index hay không. Nếu một bước FilterNode có dòng chữ "uses index", điều đó có nghĩa là truy vấn đang được tối ưu. Ngược lại, nếu nó phải quét toàn bộ collection (Full Collection Scan), đó là dấu hiệu cho thấy bạn cần thêm một index phù hợp.

→ Ví dụ, nếu kế hoạch cho thấy một "Full Collection Scan" với hàng triệu document chỉ để lọc ra vài document, người quản trị sẽ biết ngay cần phải tạo một skiplist index trên trường status và capacity để tăng tốc truy vấn.  
*E. Lưu và Tái sử dụng Truy vấn*Đối với các truy vấn phức tạp hoặc thường xuyên sử dụng, bạn có thể lưu chúng lại.  
Bước 1: Viết và kiểm thử truy vấn hoàn chỉnh.  
Bước 2: Nhấn nút Save. Đặt tên cho truy vấn (ví dụ: "Find\_Busiest\_Stations").  
Bước 3: Để mở lại, click vào mục Custom Queries ở menu bên trái của AQL Editor. Tất cả các truy vấn đã lưu sẽ xuất hiện ở đó.

**4.5. Import / Export dữ liệu**

# Import và Export là hai tác vụ cơ bản nhưng cực kỳ quan trọng trong vòng đời của một dự án. Chúng cho phép di chuyển dữ liệu giữa các môi trường hoặc tích hợp với các hệ thống khác. ArrangoDB hỗ trợ 2 phương thức chính cho việc này:

# *Cách 1: Sử dụng ArangoDB Web Interface*

# Phương pháp này phù hợp cho việc nhập (import) các file dữ liệu nhỏ một cách nhanh chóng.

# Bước 1: Chuẩn bị file dữ liệu

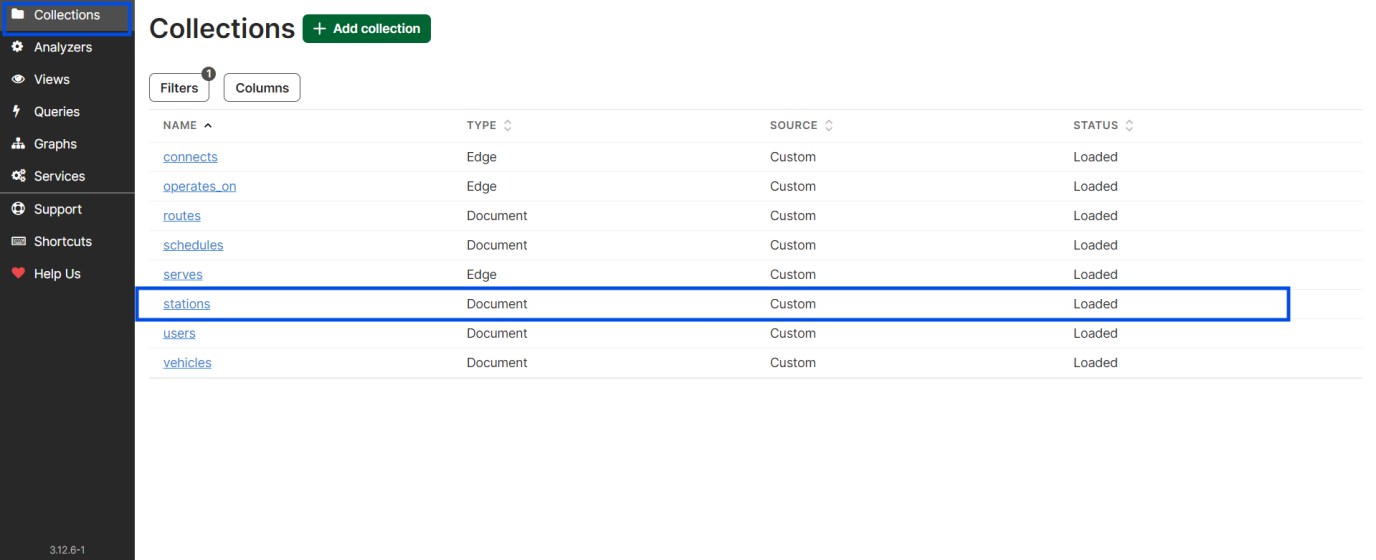
# Tạo một file JSON chứa các document cần import, ví dụ stations\_import.json. Mỗi document nằm trên một dòng riêng biệt (JSONL format).

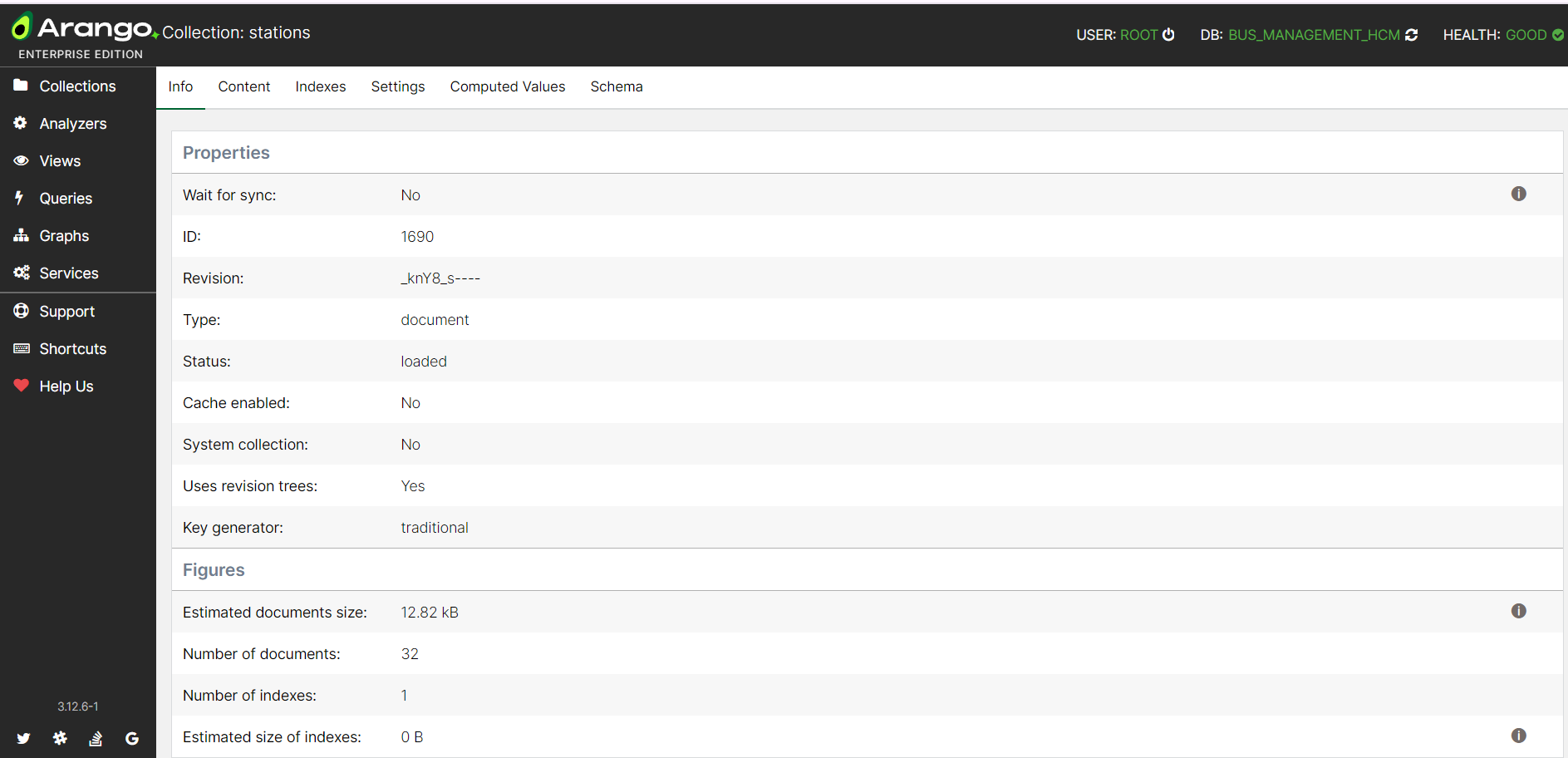
*{"\_key":"ST101", "name":"Trạm Mẫu 1", ...}*

*{"\_key":"ST102", "name":"Trạm Mẫu 2", ...}*

# Bước 2: Truy cập Collection

# Trong Web Interface, vào tab COLLECTIONS và chọn collection bạn muốn import dữ liệu vào (ví dụ: stations).





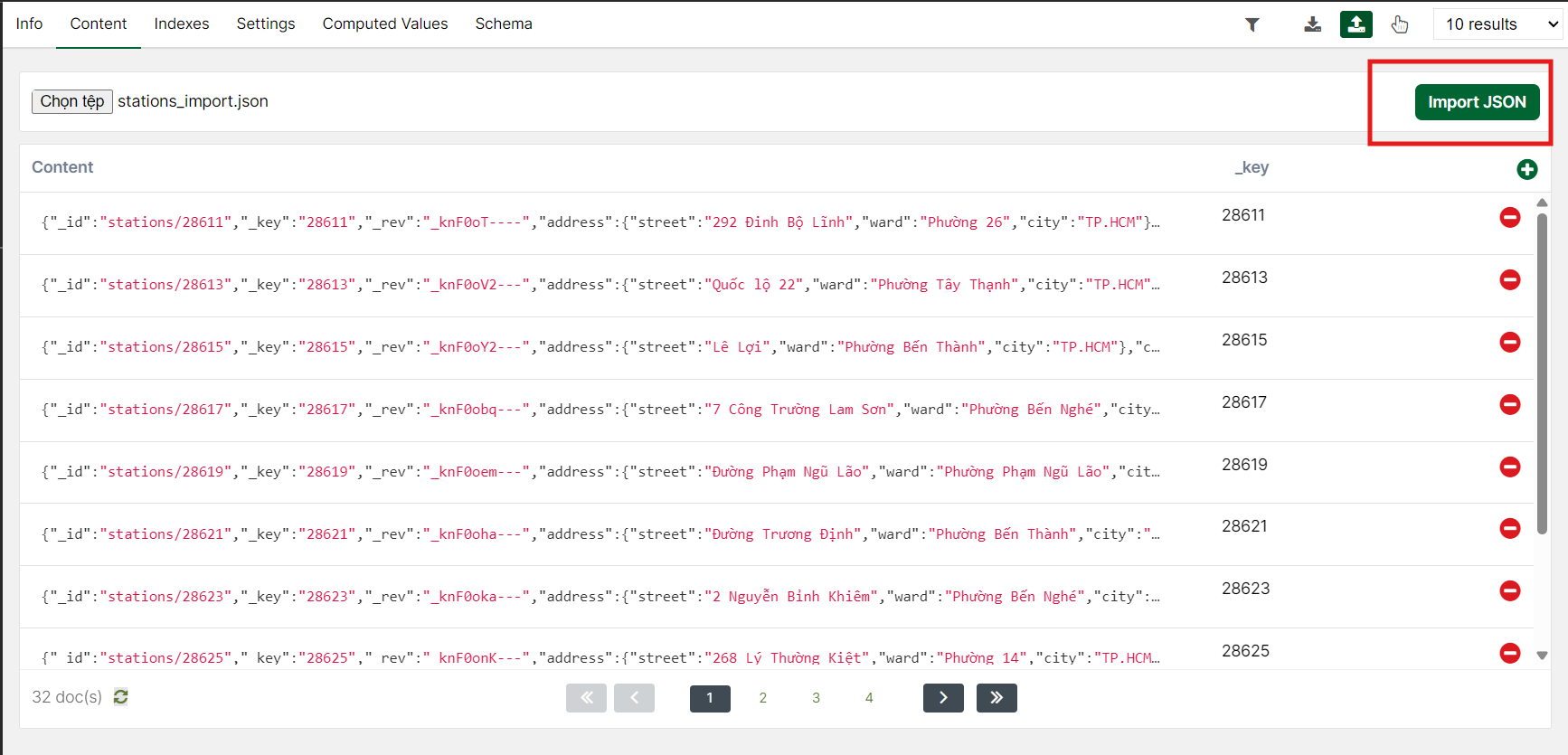
# Bước 3: Vào tab Content và tải dữ liệu lên

# Nhấn vào biểu tượng upload (mũi tên đi lên) ở góc trên bên phải của danh sách document.



# Bước 4: Chọn file và Import

# Một hộp thoại sẽ hiện ra, cho phép bạn chọn file từ máy tính và cấu hình các tùy chọn. Sau khi chọn file, nhấn nút Import.

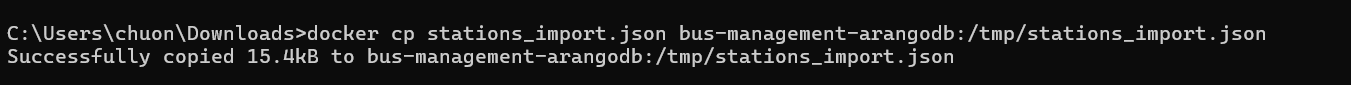


# *Cách 2: Sử dụng Công cụ dòng lệnh (arangoimport) với Docker*

# Bước 1: Chuẩn bị file và sao chép vào container

# Giả sử bạn có file stations\_import.json trên máy tính. Đầu tiên, cần sao chép file này vào bên trong Docker container.

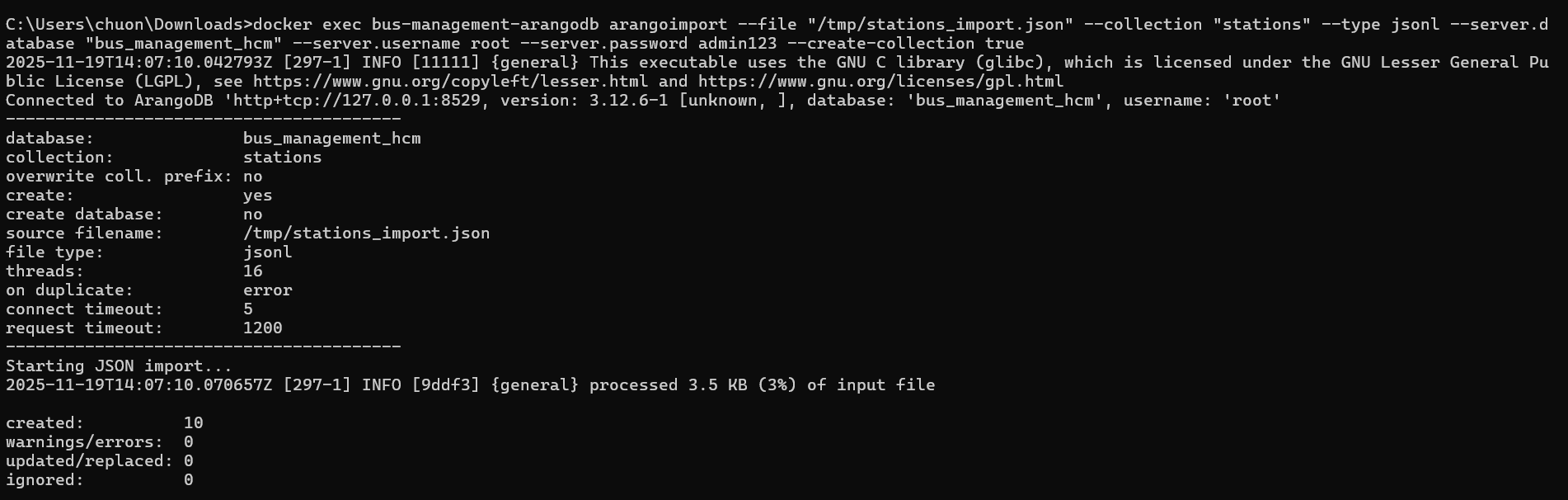
# Mở terminal và chạy lệnh: *“docker cp stations\_import.json bus-management-arangodb:/tmp/stations\_import.json”*



Bước 2: Thực thi lệnh arangoimport

Sử dụng docker exec để chạy arangoimport bên trong container.

“*docker exec bus-management-arangodb arangoimport --file "/tmp/stations\_import.json" --collection "stations" --type jsonl --server.database "bus\_management\_hcm" --server.username root --server.password admin123 --create-collection true”*

  
**4.6. Backup / Restore dữ liệu**

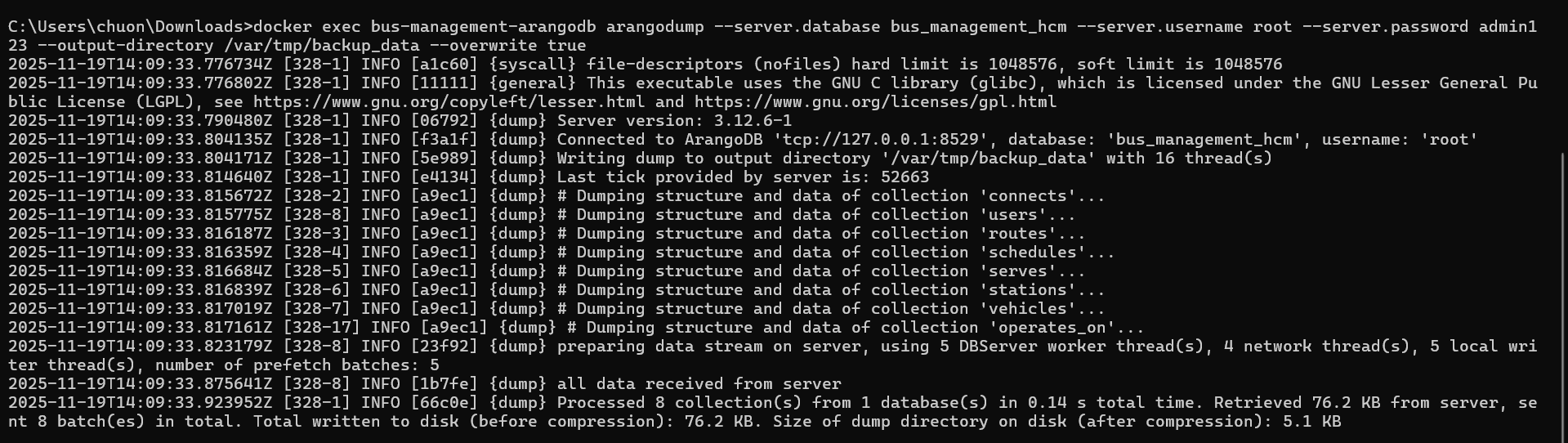
# *Cách 1: Backup dữ liệu (arangodump) với Docker*

Bước 1: Tạo thư mục backup bên trong container Thực thi lệnh arangodump để tạo một thư mục chứa dữ liệu sao lưu bên trong container chạy lệnh

“

*docker exec bus-management-arangodb arangodump --server.database bus\_management\_hcm --server.username root --server.password admin123 --output-directory /var/tmp/backup\_data --overwrite true*

*"*”



Lúc này, dữ liệu sao lưu đang nằm trong thư mục /var/tmp/backup\_data bên trong container.

Bước 2: Sao chép thư mục backup ra máy tính

Để lưu trữ bản backup an toàn, chúng ta cần sao chép nó ra khỏi container bằng lệnh

“

*docker cp bus-management-arangodb:/var/tmp/backup\_data C:\Users\chuon\Downloads\db\_backup\_20251120*

”



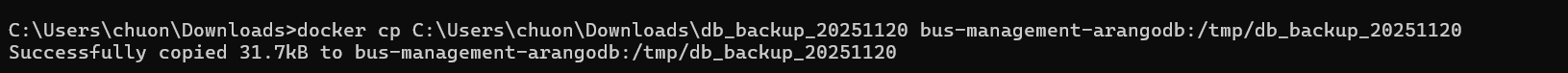
Bây giờ bạn đã có một thư mục db\_backup\_20251120 trên máy tính chứa toàn bộ dữ liệu sao lưu.

# *Cách 1: Restore dữ liệu (arangorestore) với Docker*

**Bước 1**: Sao chép thư mục backup vào container

Giả sử bạn cần phục hồi từ thư mục db\_backup\_20251120. Đầu tiên, sao chép nó vào container bằng cách chạy lệnh

“*docker cp C:\Users\chuon\Downloads\db\_backup\_20251120 bus-management-arangodb:/tmp/db\_backup\_20251120*”



**Bước 2**: Tạo database mới (nếu chưa có)

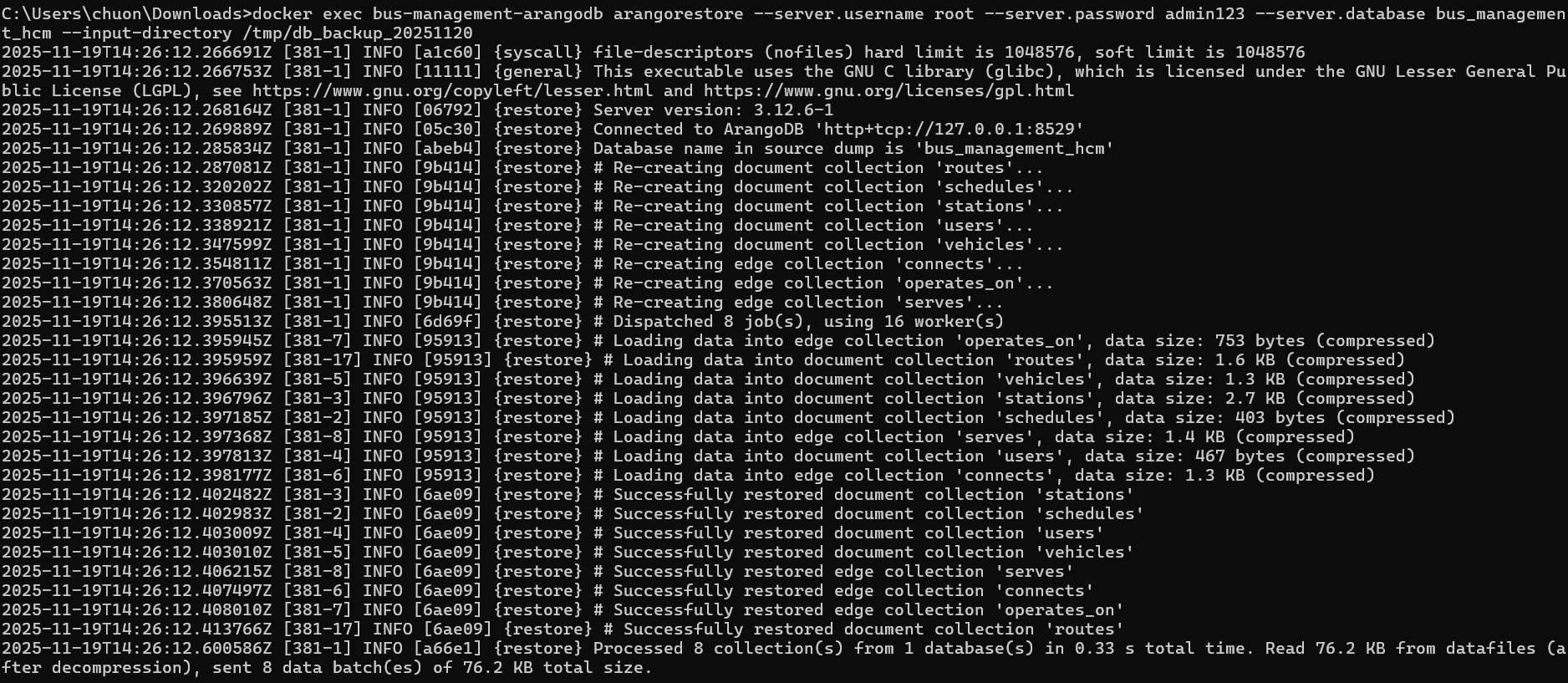
“docker exec -it bus-management-arangodb arangosh --server.username root --server.password admin123 --javascript.execute-string "db.\_createDatabase('bus\_management\_hcm');"”

**Bước 3**: Thực thi lệnh arangorestore

Sử dụng docker exec để chạy arangorestore restore dữ liệu vào database.

“*docker exec bus-management-arangodb arangorestore --server.username root --server.password admin123 --server.database bus\_management\_hcm --input-directory /tmp/db\_backup\_20251120*

”



**4.7. User Management – phân quyền người dùng**

ArangoDB có một hệ thống quản lý người dùng và phân quyền mạnh mẽ ở cấp độ CSDL, tách biệt hoàn toàn với collection users trong ứng dụng của chúng ta. Chức năng này cho phép kiểm soát ai có thể truy cập vào database nào và với quyền hạn gì.

Có hai phương pháp chính để thực hiện việc quản lý người dùng:

* Sử dụng Giao diện Web trực quan
* Sử dụng công cụ dòng lệnh arangosh.

**Cách 1: Sử dụng Giao diện Web (Web Interface)**

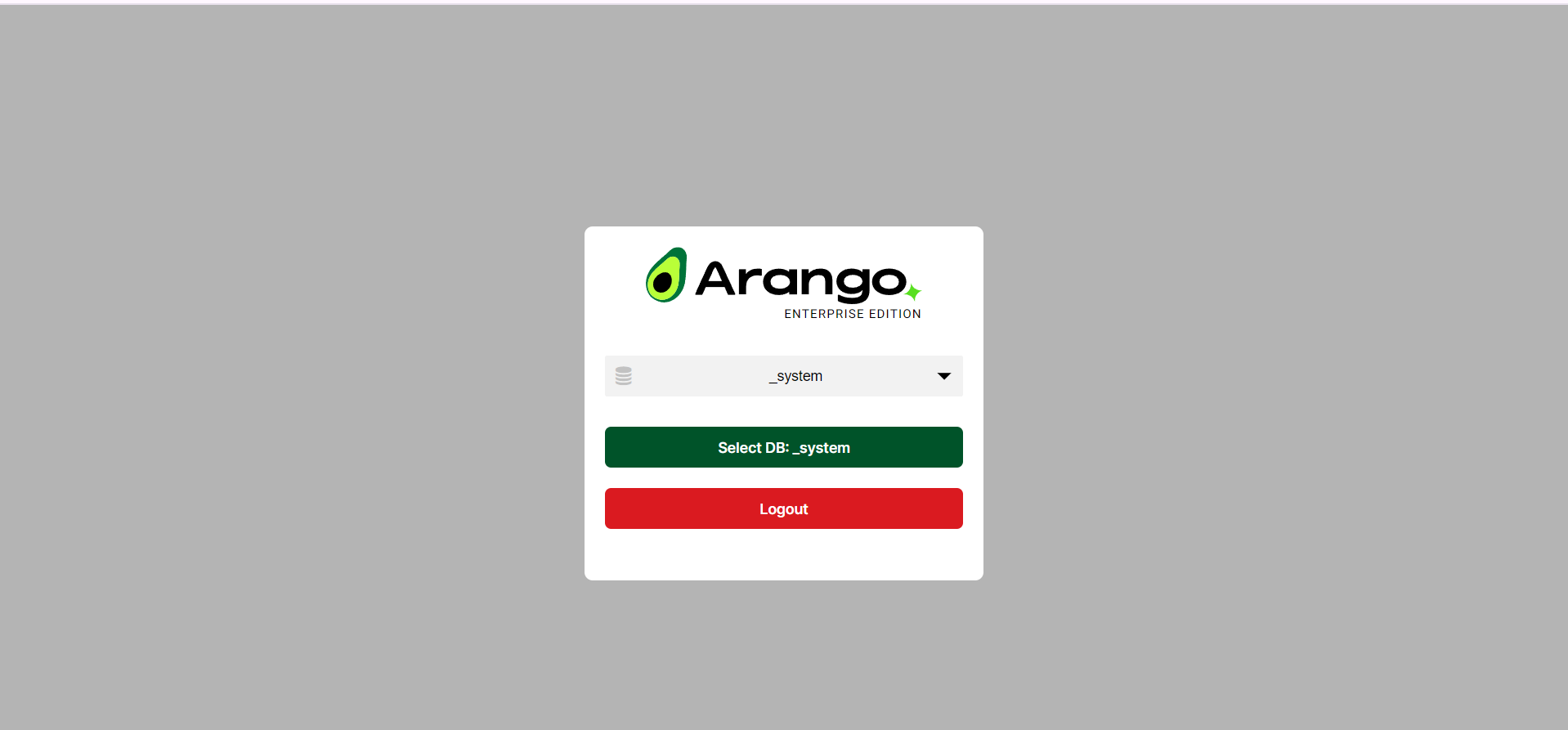
Đây là cách trực quan nhất, phù hợp cho người mới bắt đầu hoặc khi cần thao tác nhanh trên một vài tài khoản.

**Bước 1**: Đăng nhập vào database hệ thống (\_system)

Tất cả các thao tác quản lý user chỉ có thể thực hiện được khi đang ở trong database \_system.

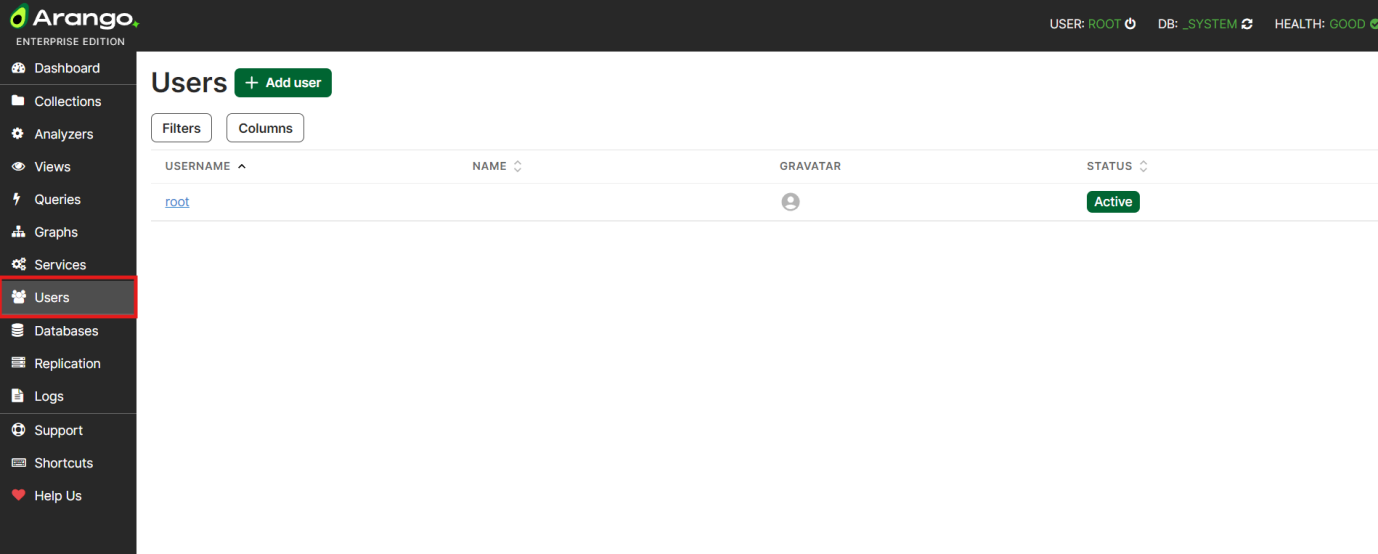
Truy cập: http://localhost:8529/\_db/\_system/\_admin/aardvark/index.html

Đăng nhập với tài khoản root và mật khẩu quản trị (ví dụ: admin123).



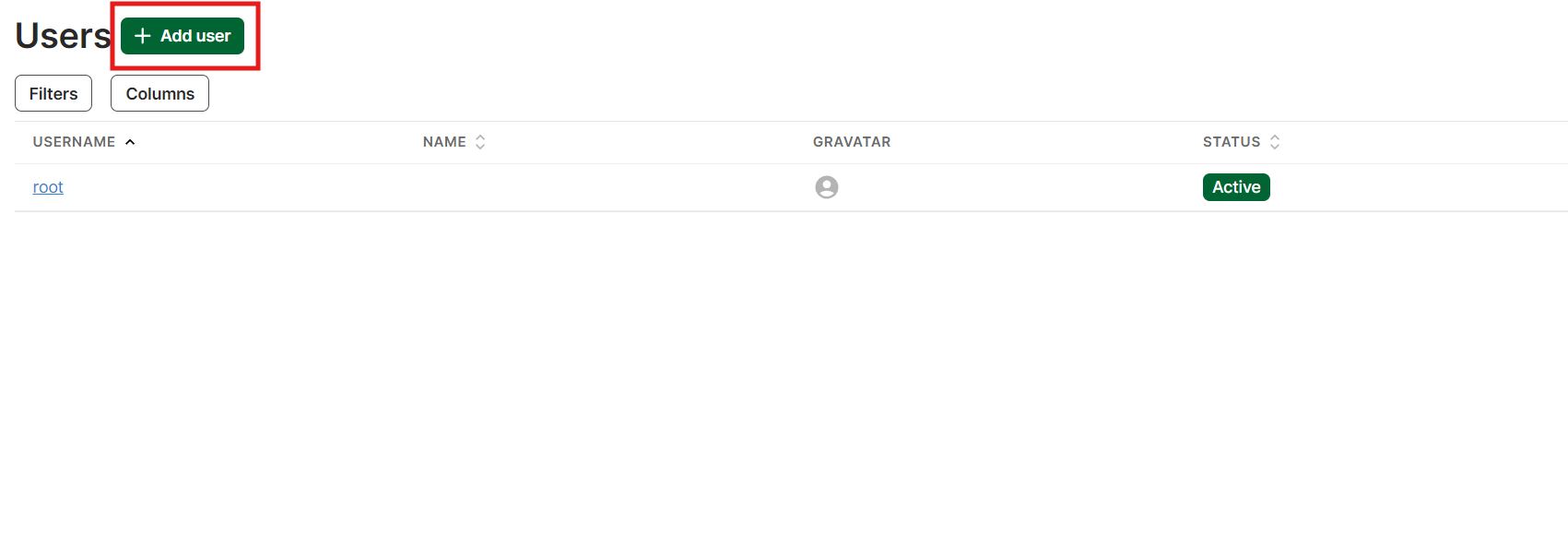
**Bước 2**: Truy cập mục Users

Trong menu điều hướng bên trái, chọn tab USERS. Tại đây sẽ hiển thị danh sách tất cả người dùng hiện có trong hệ thống.



**Bước 3**: Tạo người dùng mới

Nhấn nút Add User ở góc trên bên phải. Một hộp thoại sẽ xuất hiện.



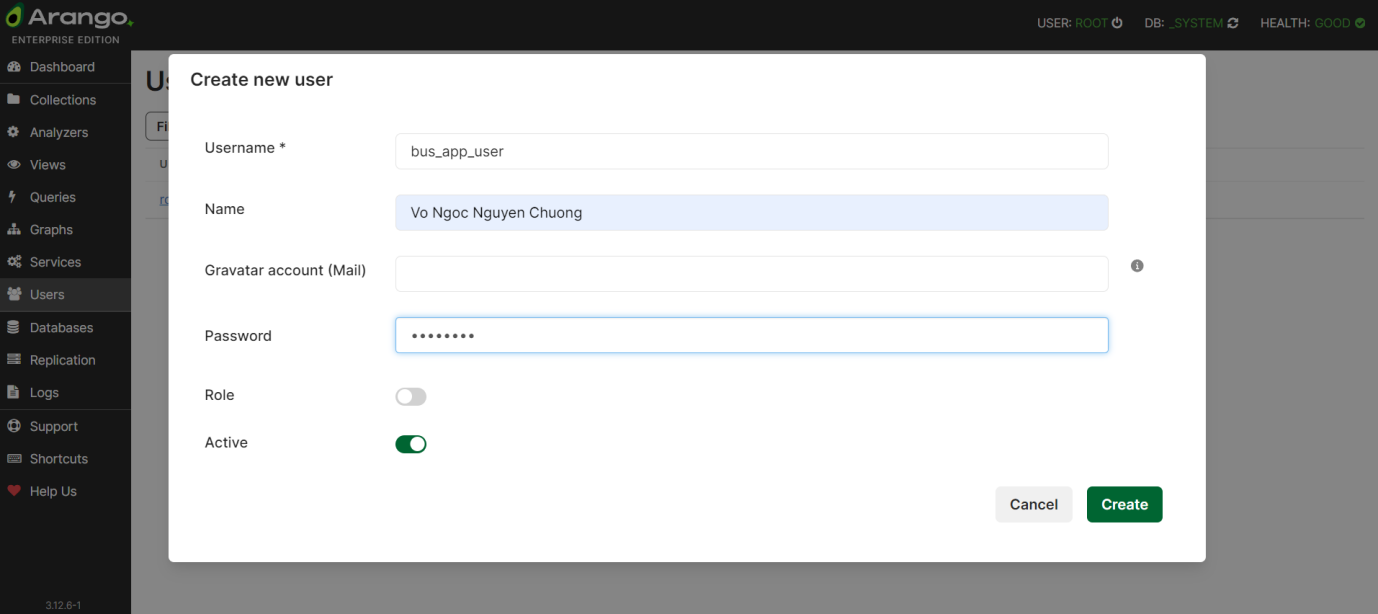
Điền thông tin:

Username: bus\_app\_user (ví dụ)

Password: Nhập mật khẩu mong muốn.

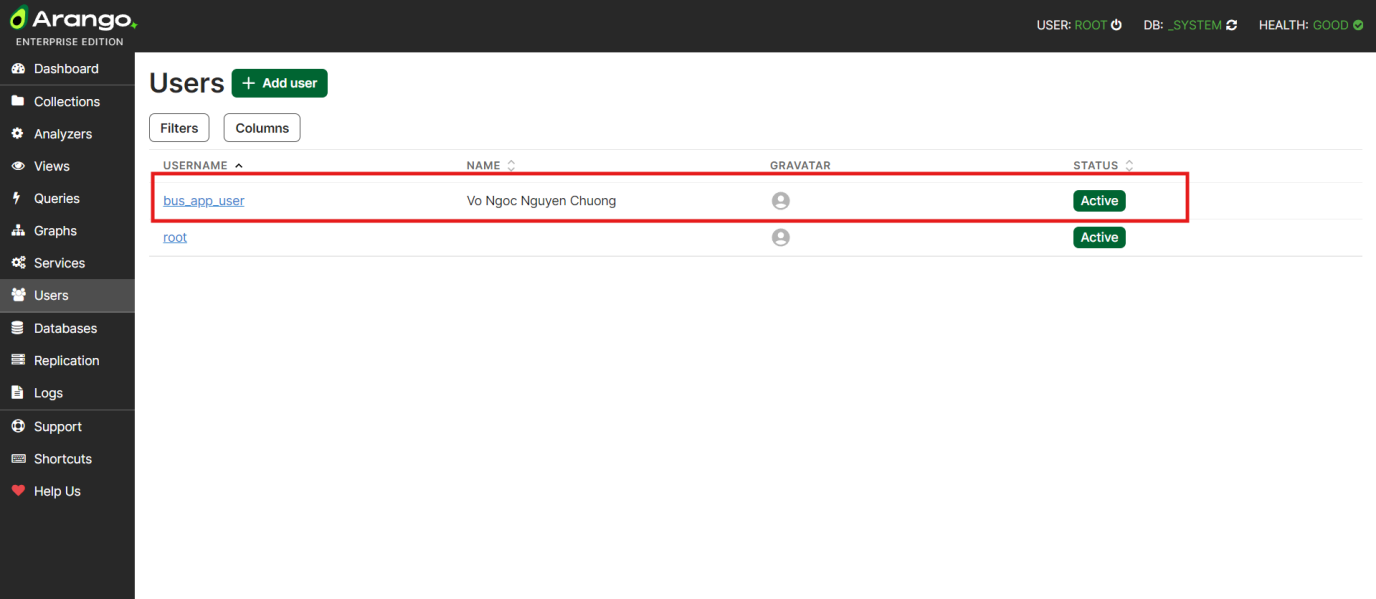
Active: Đảm bảo ô này được tích chọn để tài khoản hoạt động.

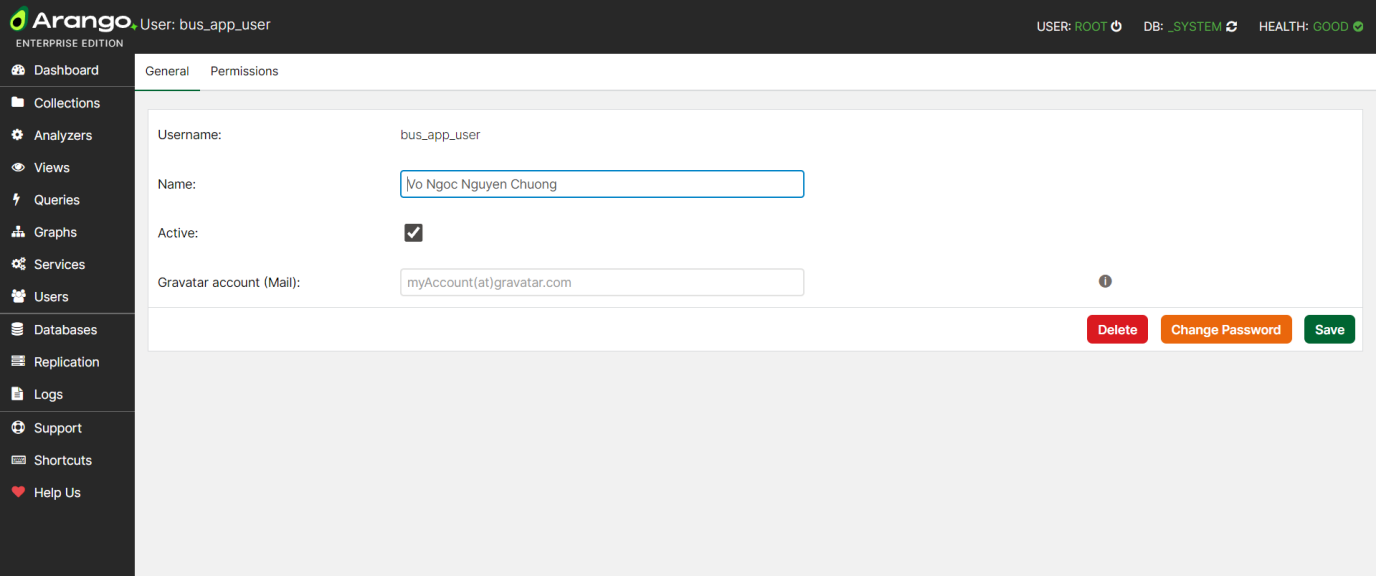
Nhấn Create để tạo user.



Bước 4: Phân quyền trên Database

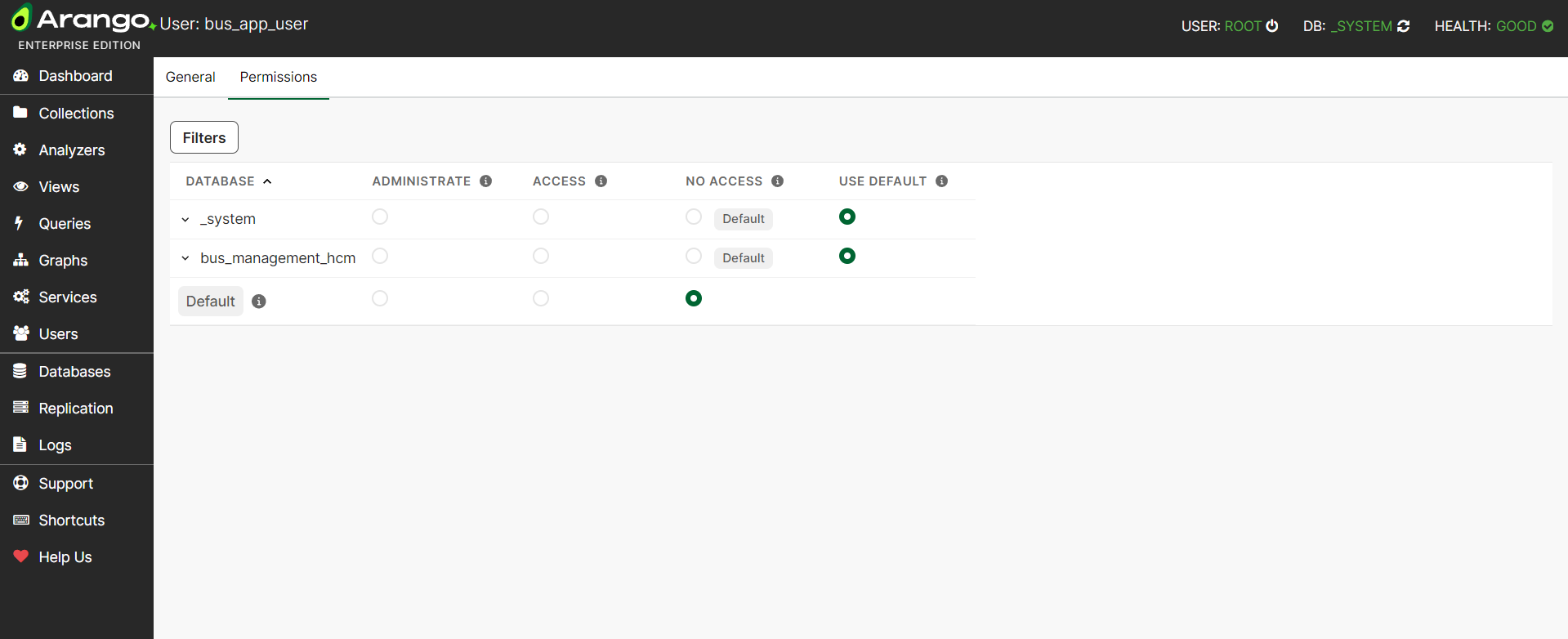
Sau khi tạo, click vào tên người dùng bus\_app\_user trong danh sách để vào trang chi tiết.





Chuyển sang tab Permissions. Tại đây, bạn sẽ thấy danh sách các database hiện có.

Tìm database bus\_management\_hcm và chọn quyền hạn mong muốn:

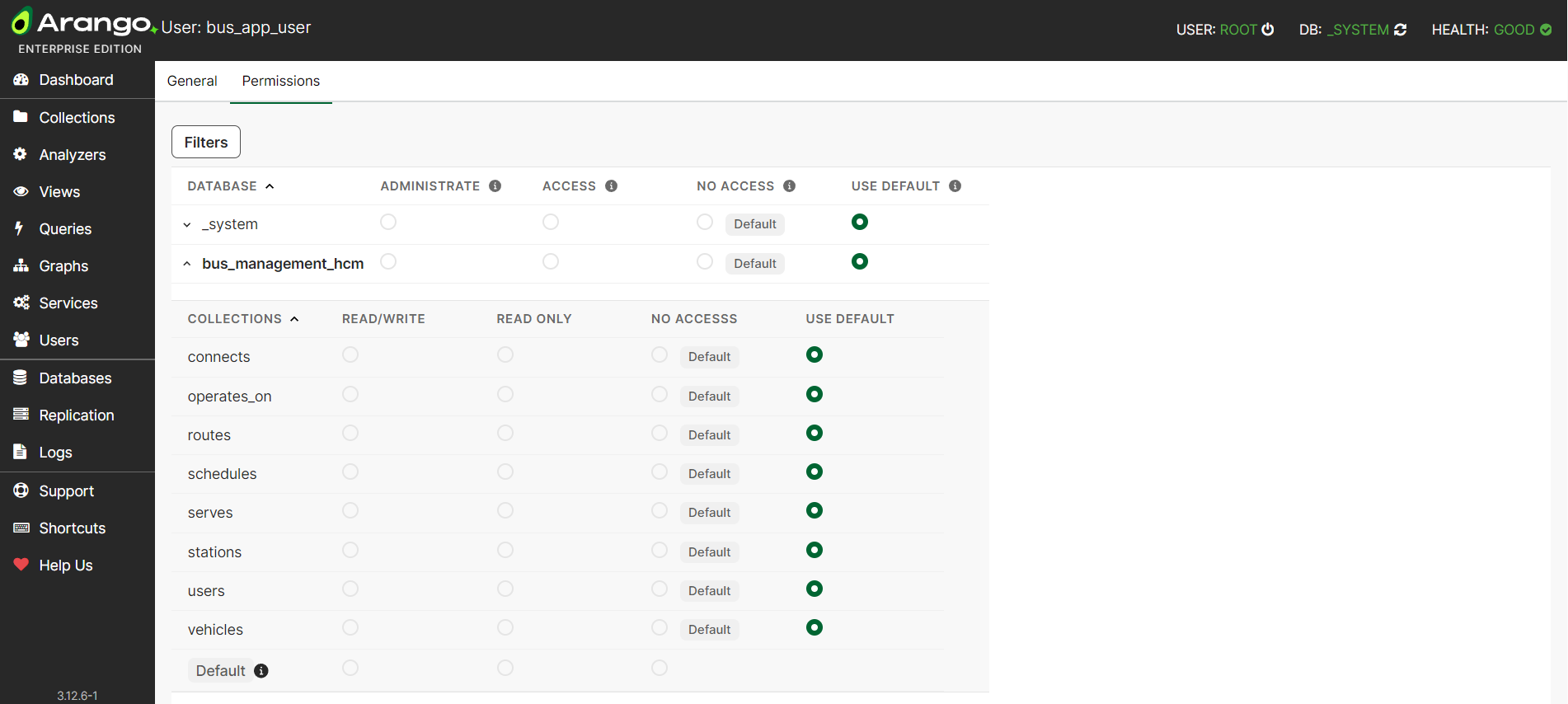


Administrator (rw): Toàn quyền quản trị trên database đó (tạo/xóa collection, index...).

Access (ro/rw): Có quyền truy cập đọc/ghi dữ liệu. Đây là quyền phù hợp nhất cho các ứng dụng kết nối vào CSDL.

No Access (none): Không có quyền truy cập database này.

Chọn quyền Access (hoặc Administrator tùy nhu cầu) cho bus\_management\_hcm và nhấn Save.



**Cách 2: Sử dụng Dòng lệnh (CMD / Arangosh)**

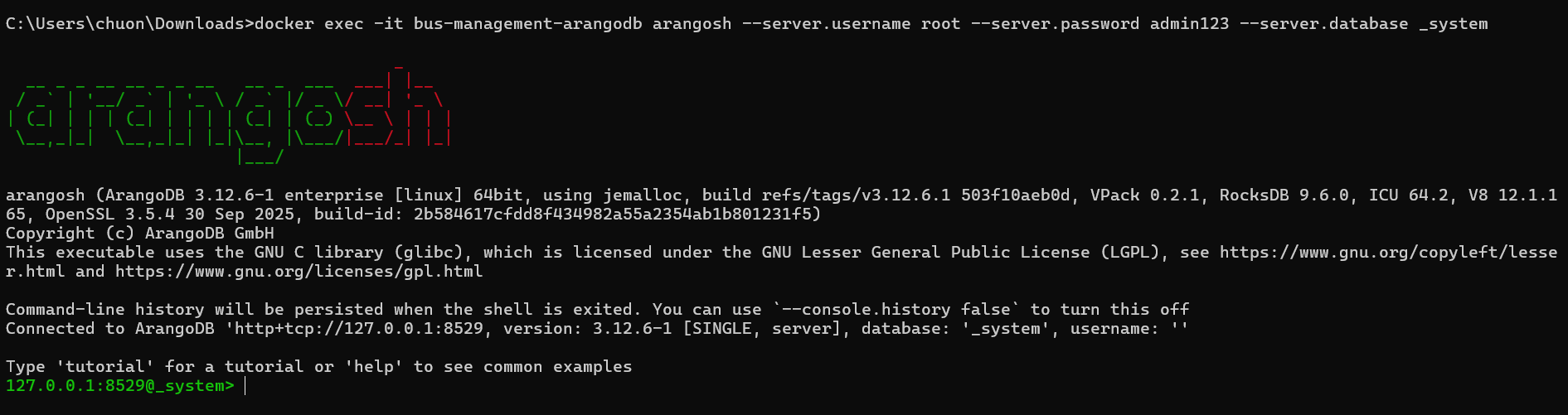
Đây là phương pháp chuyên nghiệp, nhanh chóng và có thể tự động hóa được (scripting), rất hữu ích khi triển khai hệ thống trên môi trường server hoặc Docker.

**Bước 1**: Kết nối vào ArangoDB Shell (arangosh)

Mở Command Prompt (CMD) hoặc Terminal. Nếu sử dụng Docker, ta cần thực thi lệnh bên trong container.

Chạy lệnh sau để đăng nhập vào arangosh với quyền root trên database \_system:

“*docker exec -it bus-management-arangodb arangosh --server.username root --server.password admin123 --server.database \_system*”



**Bước 2**: Tạo người dùng mới

Tại dấu nhắc lệnh của arangosh (thường là 127.0.0.1:8529@\_system>), nhập lệnh sau để tạo user bus\_app\_user với mật khẩu bus\_app\_pass:

*“require("@arangodb/users").save("bus\_app\_user", "bus\_app\_pass");”*

Nếu thành công, hệ thống sẽ trả về một object chứa thông tin user vừa tạo.



**Bước 3**: Phân quyền cho Database

Cấp quyền truy cập cho user vừa tạo trên database bus\_management\_hcm.

Để cấp quyền Access (đọc/ghi dữ liệu - rw):

*“require("@arangodb/users").grantDatabase("bus\_app\_user", "bus\_management\_hcm", "rw");”*

Để cấp quyền Administrator (quản trị đầy đủ):

*“db.\_useDatabase("bus\_management\_hcm");*

*const users = require("@arangodb/users");*

*let cols = db.\_collections();*

*cols.forEach(c => {*

*let name = c.name();*

*if (!name.startsWith("\_")) {*

*users.grantCollection("bus\_app\_user", "bus\_management\_hcm", name, "rw");*

*}*

*});*

*”*

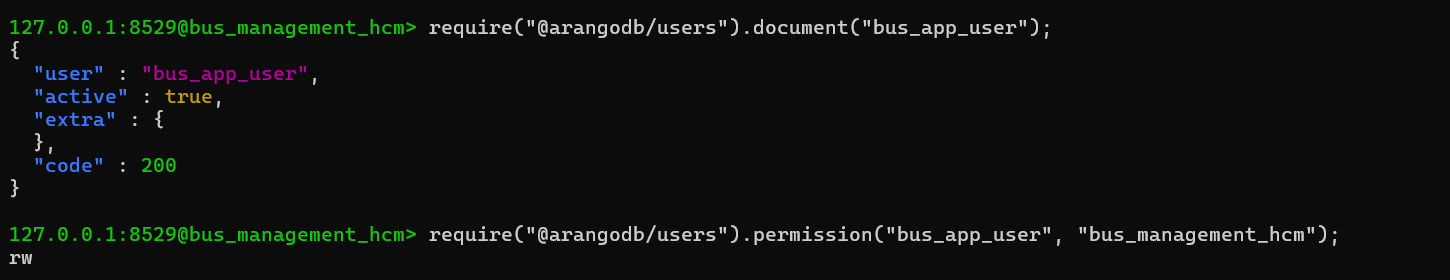
Bước 4: Kiểm tra lại thông tin

Để xác minh user đã được tạo và phân quyền đúng hay chưa, bạn có thể dùng các lệnh sau:

Kiểm tra thông tin user:

“*require("@arangodb/users").document("bus\_app\_user");*

*require("@arangodb/users").permission("bus\_app\_user", "bus\_management\_hcm");*”

HÌNH: Màn hình terminal thực hiện các lệnh arangosh để tạo và phân quyền user

Đối với ứng dụng quản lý xe buýt, chúng ta có thể tạo một user là bus\_app\_user và cấp quyền Access cho database bus\_management\_hcm.  
**4.8. Các tính năng nâng cao**

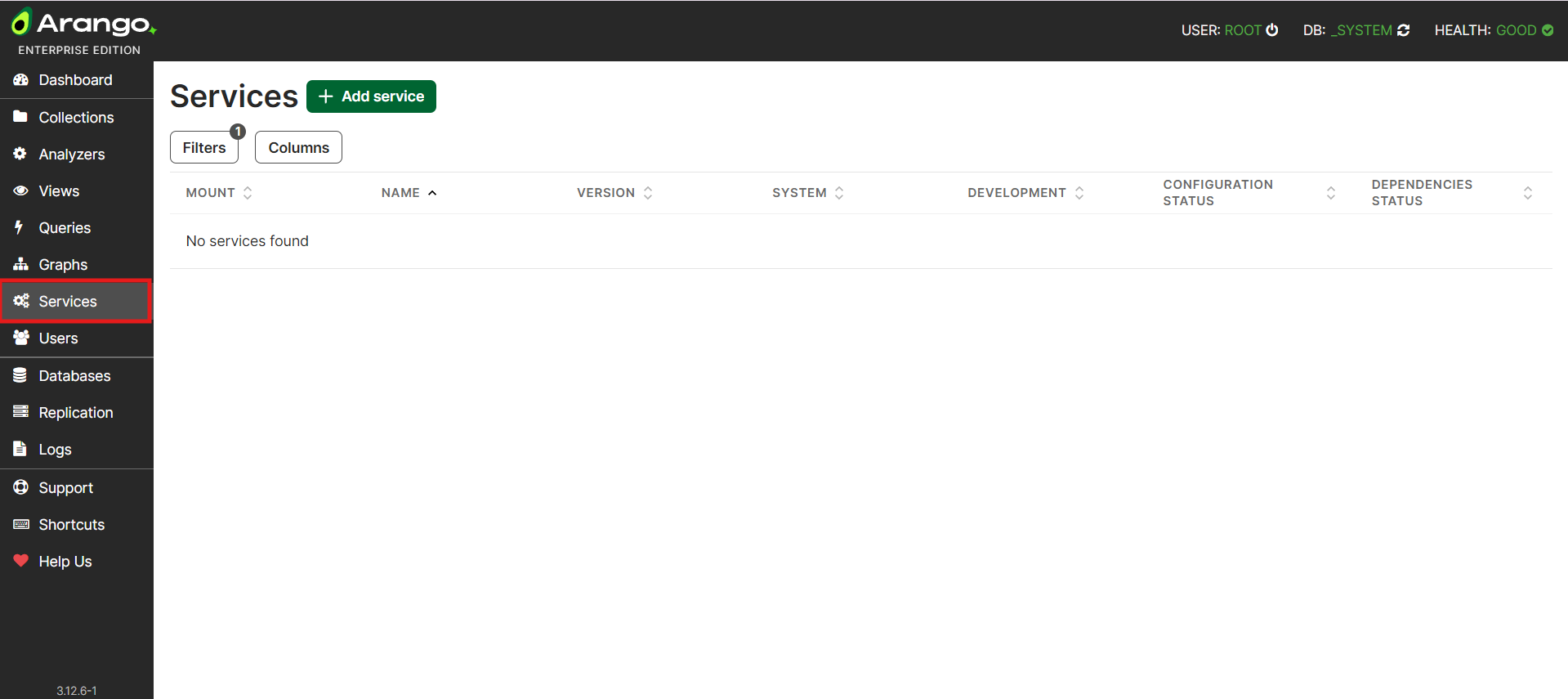
* Foxx Services

Khái niệm: Foxx là một framework microservice được tích hợp sẵn trong ArangoDB, cho phép bạn viết các API RESTful bằng JavaScript và chạy chúng trực tiếp bên trong CSDL.

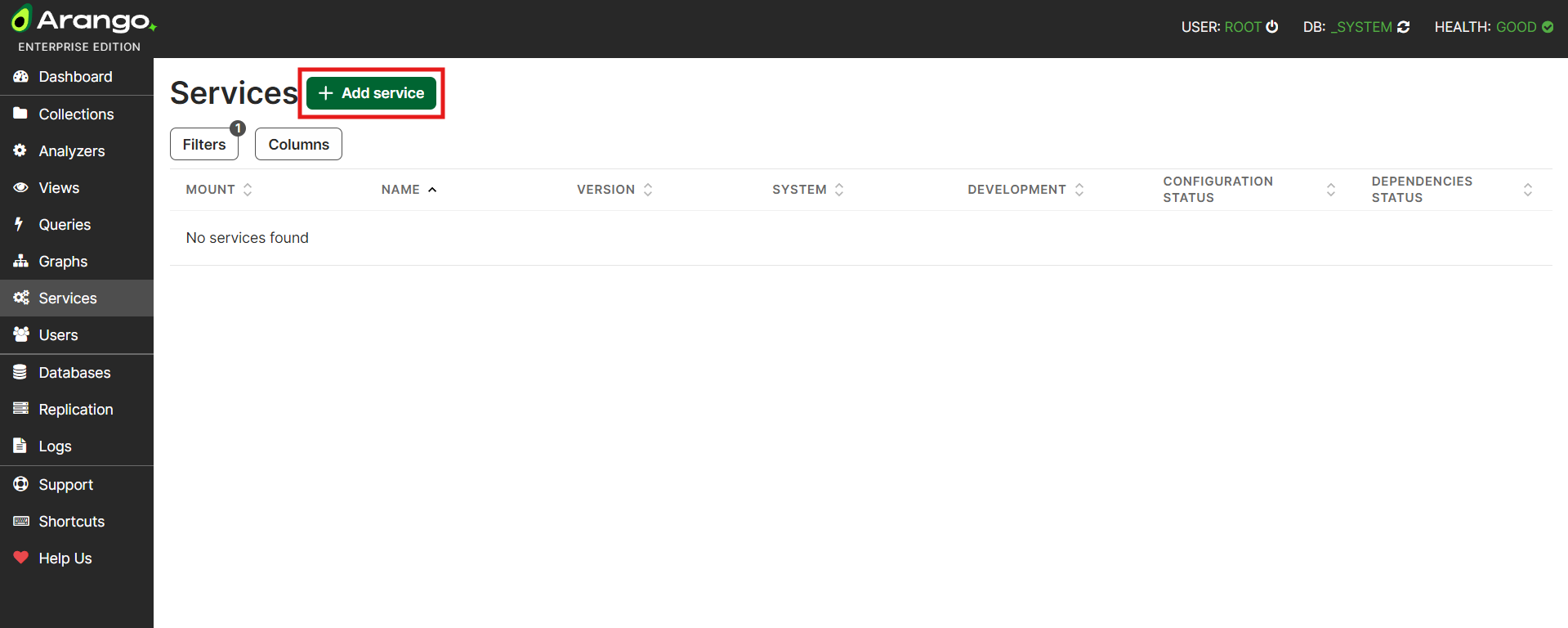
Lợi ích: Loại bỏ độ trễ mạng giữa lớp ứng dụng và CSDL, lý tưởng cho các tác vụ cần xử lý dữ liệu nặng.

Cách sử dụng:

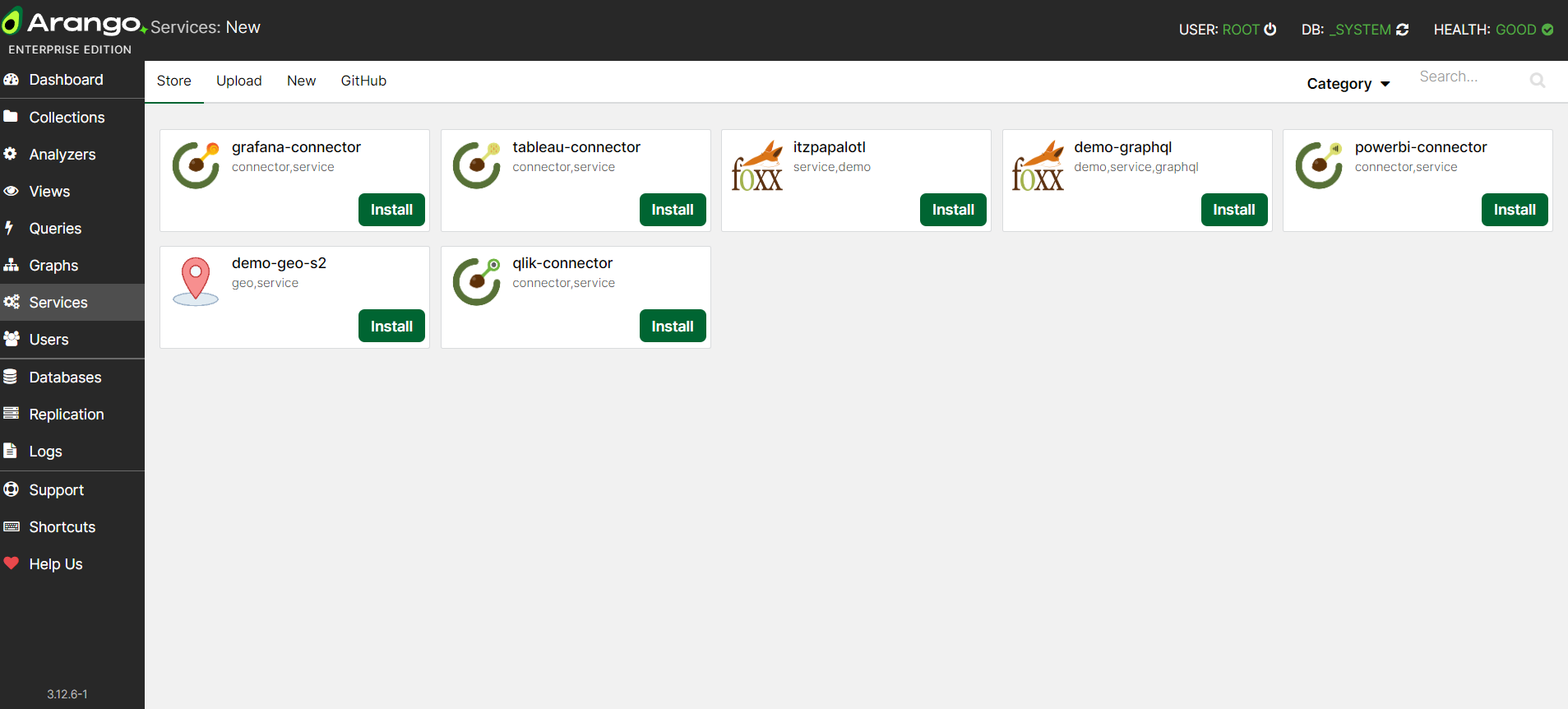
* Truy cập tab SERVICES trong Web Interface.



* Nhấn Add Service.



* Có thể cài đặt từ ArangoDB Store, upload một file zip chứa code, hoặc kết nối với GitHub.



* Index Management

Khái niệm: ArangoDB Web Interface cung cấp các công cụ trực quan để giám sát sức khỏe và hiệu năng của hệ thống.

Các mục giám sát chính:

* Dashboard (tab STATS): Cung cấp cái nhìn tổng quan về tài nguyên hệ thống (CPU, bộ nhớ), số lượng yêu cầu, thông tin phiên bản.
* Thống kê Collection (Collection -> Info): Hiển thị số lượng document, kích thước dữ liệu trên đĩa của từng collection.

Truy vấn (tab QUERIES):

* Running Queries: Xem các truy vấn đang được thực thi.
* Slow Queries: Liệt kê các truy vấn chạy chậm, giúp xác định và tối ưu hóa các AQL kém hiệu quả. Đây là công cụ cực kỳ hữu ích cho DBA.
* System Monitoring

Khái niệm: ArangoDB Web Interface cung cấp các công cụ trực quan để giám sát sức khỏe và hiệu năng của hệ thống.

Các mục giám sát chính:

* Dashboard (tab STATS): Cung cấp cái nhìn tổng quan về tài nguyên hệ thống (CPU, bộ nhớ), số lượng yêu cầu, thông tin phiên bản.
* Thống kê Collection (Collection -> Info): Hiển thị số lượng document, kích thước dữ liệu trên đĩa của từng collection.

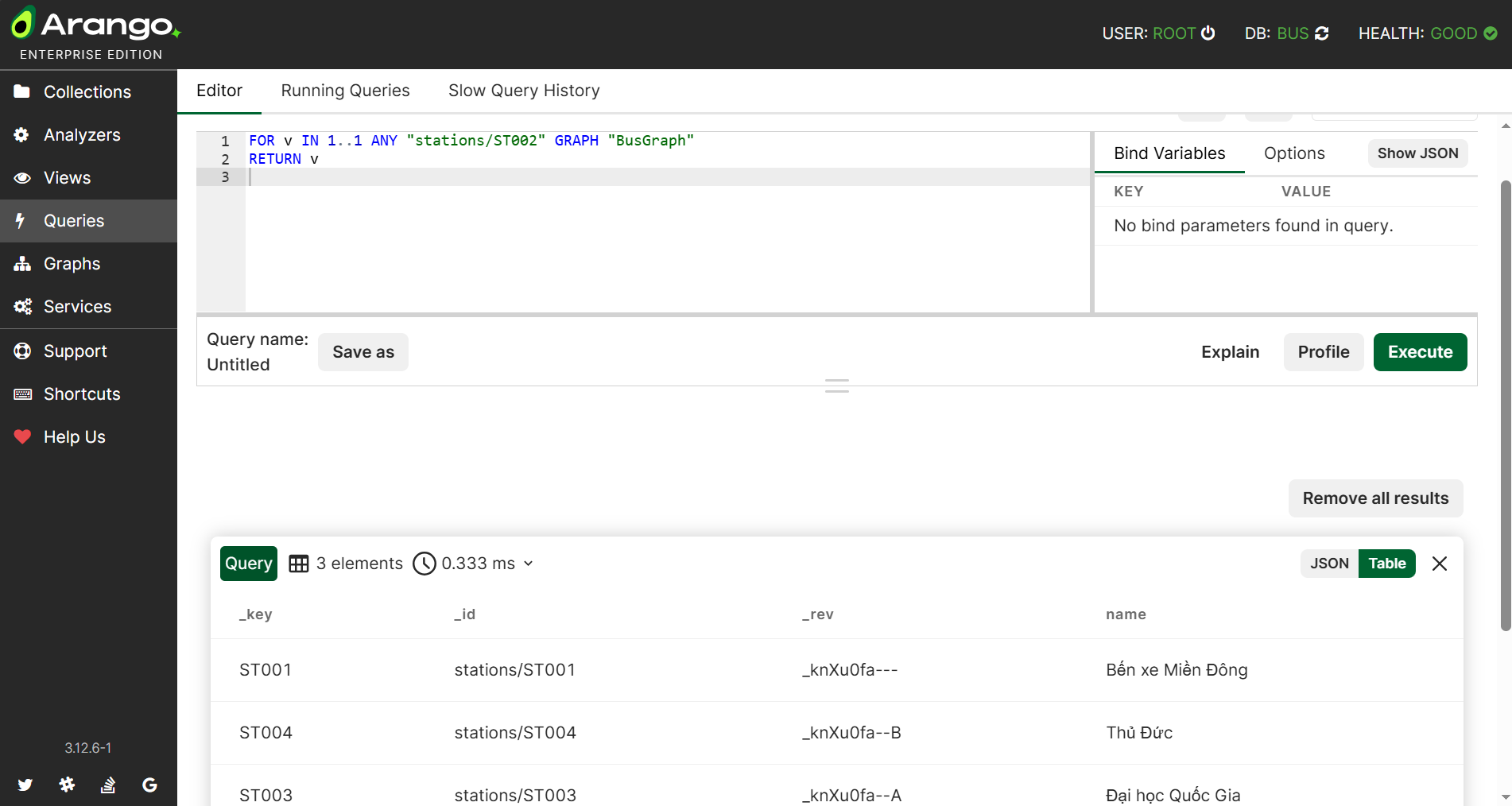
Truy vấn (tab QUERIES):

* Running Queries: Xem các truy vấn đang được thực thi.
* Slow Queries: Liệt kê các truy vấn chạy chậm, giúp xác định và tối ưu hóa các AQL kém hiệu quả. Đây là công cụ cực kỳ hữu ích cho DBA.

# CHƯƠNG 5. SO SÁNH ARANGODB WEB UI VỚI NEO4J BROWSER

# 5.1. Tiêu chí so sánh

Ví dụ : Tìm các trạm lân cận (1 hop neighbors)



Hình : Kết quả thực thi truy vấn của ArangoDB



Hình : Kết quả thực thi truy vấn của Neo4j

|  |  |  |  |
| --- | --- | --- | --- |
| Tiêu chí | ArangoDB Web UI | Neo4j Browser | Đánh giá |
| Thời gian thực thi (ví dụ so sánh ở trên) | 0.333 ms | ~2 ms | ArangoDB nhanh hơn ở dataset nhỏ do caching. |
| Mức độ dễ sử dụng | UI kỹ thuật, nhiều mục (Collections, Graphs, Queries…) | Tập trung vào graph, dễ hiểu với người mới | Neo4j thân thiện hơn |
| Dễ mở rộng (scaling) | Có clustering + sharding rất mạnh | Có clustering nhưng theo mô hình riêng | Tùy bài toán; ArangoDB linh hoạt multi-model. |
| Phù hợp cho loại dữ liệu | Kết hợp document + graph (vd: hệ thống xe buýt) | Quan hệ đồ thị thuần (social, shortest path) | Tùy yêu cầu ứng dụng. |

Bảng : So sánh tổng hợp các tiêu chí

# 5.2. Điểm mạnh của ArangoDB Web Interface

ArangoDB Web Interface (Web UI) là công cụ quản trị trực quan được tích hợp sẵn trong ArangoDB, hỗ trợ đầy đủ các tác vụ quản lý, truy vấn, trực quan hóa và phân tích dữ liệu. Giao diện này là một trong những ưu điểm lớn của ArangoDB, đặc biệt trong các hệ thống cần xử lý mô hình đa dữ liệu như tài liệu (document), quan hệ đồ thị (graph), và key–value. Những điểm mạnh nổi bật bao gồm:

* Quản lý đa mô hình (Multi-Model) trong một giao diện duy nhất
  + ArangoDB Web UI cho phép người dùng quản lý Document Collections, Edge Collections, và Graphs ngay trong cùng một công cụ, giúp việc thao tác giữa các mô hình trở nên liền mạch và thuận tiện.
  + Người dùng có thể xem, chỉnh sửa và thêm tài liệu JSON trực tiếp.
  + Có thể quan sát cấu trúc đồ thị, xem edge, node và các thuộc tính chi tiết.
  + Chuyển đổi linh hoạt giữa document view và graph view.

Điều này giúp ArangoDB Web UI mạnh hơn nhiều so với Neo4j Browser, vốn chỉ tập trung vào đồ thị và không hỗ trợ quản lý document.(5.2. 1\*)

* Hiển thị dữ liệu dạng JSON trực quan và đầy đủ
  + ArangoDB Web UI hỗ trợ xem dữ liệu dạng JSON hoàn chỉnh, có thể mở rộng, thu gọn, và chỉnh sửa trực tiếp.
  + Ưu điểm: Theo dõi dữ liệu raw rất rõ ràng, dễ dàng debug dữ liệu khi làm việc với API, hiệu quả khi quản lý dữ liệu phức tạp như thông tin trạm xe buýt (địa chỉ, vị trí GPS, tiện ích…).

Đây là lợi thế lớn so với Neo4j Browser, vốn chỉ hiển thị property theo dạng bảng và không hỗ trợ cấu trúc JSON lồng nhau.(5.2. 2\*)

* Công cụ truy vấn mạnh mẽ (AQL Query Editor)
  + ArangoDB Web UI cung cấp AQL Editor với các tính năng: Gợi ý cú pháp, chạy truy vấn trực tiếp, chế độ Explain & Profile để tối ưu query, xuất kết quả JSON hoặc bảng, chạy query trên cả document và graph bằng một ngôn ngữ duy nhất (AQL).

AQL Editor cho phép kiểm tra hiệu năng truy vấn, phù hợp khi cần đánh giá độ trễ trong bài toán mạng lưới xe buýt.(5.2.3\*)

* Trình quản lý Graph trực quan
  + Trong phần Graphs, ArangoDB Web UI cho phép: Tạo graph nhanh chóng, định nghĩa edge direction, edge collection, xem cấu trúc graph cơ bản, quan sát mối quan hệ giữa các trạm xe buýt.

Mặc dù không mạnh bằng Neo4j về visualization, ArangoDB vẫn cung cấp đủ công cụ để hiểu cấu trúc mạng lưới và hỗ trợ phát triển ứng dụng.(5.2.4\*)

* Hỗ trợ tốt cho hệ thống có dữ liệu phức tạp như mạng lưới xe buýt
  + ArangoDB Web UI hỗ trợ rất tốt: Lưu trữ thông tin trạm bằng Document, lưu quan hệ trạm–trạm bằng Edge, truy vấn đường đi, lân cận bằng Graph Traversal.
  + Kết hợp Document + Graph trong một query AQL → rất phù hợp.

# 5.3. Hạn chế của ArangoDB Web Interface

Mặc dù ArangoDB Web Interface (Web UI) cung cấp nhiều tính năng mạnh mẽ cho việc quản lý cơ sở dữ liệu đa mô hình, giao diện này vẫn tồn tại một số hạn chế nhất định, đặc biệt khi so sánh với các công cụ chuyên về đồ thị như Neo4j Browser. Một số hạn chế chính bao gồm:

* Khả năng trực quan hóa đồ thị còn hạn chế
  + ArangoDB Web UI chỉ cung cấp mức trực quan hóa cơ bản cho mô hình đồ thị:
* Không hỗ trợ kéo–thả node để điều chỉnh vị trí.
* Không có auto-layout động như Neo4j.
* Không nổi bật rõ degree, relationship type, hoặc path như Cypher visualization.
* Không hỗ trợ highlight path, expand neighbors, hoặc mở rộng depth 1-click.
* Cú pháp AQL cho đồ thị phức tạp khó viết và khó học hơn Cypher
  + AQL được thiết kế để thao tác cả document và graph, dẫn đến:
* Truy vấn đồ thị dài hơn.
* Tập trung vào tính biểu thức hơn tính trực quan.
* Không thân thiện với người mới như Cypher của Neo4j.
* Không chuyên dụng cho đồ thị như Neo4j Browser
  + ArangoDB Web UI phục vụ multi-model nên:
* Giao diện phân tán nhiều mục (Documents, Collections, Graphs, Views, Services…)
* Không tập trung hoàn toàn vào graph analysis
* Không có plugin graph analytic như Neo4j Bloom
* Không có khả năng mở rộng graph trực quan theo chiều sâu (Expand)

Trong Neo4j Browser:

* + Khi click node → tự động gợi ý mở rộng neighbors.
  + Hỗ trợ xem cấp độ liên kết theo mong muốn.
  + Hiển thị quan hệ 1-click.

Trong ArangoDB Web UI:

* + Visualization graph mang tính “snapshot” → không expandable.
  + Không hỗ trợ multi-hop exploration theo cách tương tác.
  + Điều này gây hạn chế trong việc khám phá cấu trúc mạng lưới lớn.

# 5.4. ArangoDB phù hợp với những bài toán

ArangoDB là cơ sở dữ liệu đa mô hình (multi-model), cho phép lưu trữ tài liệu (document), quan hệ đồ thị (graph) và dữ liệu key–value trong cùng một hệ thống. Nhờ sự linh hoạt này, ArangoDB phù hợp với nhiều bài toán thực tế đòi hỏi vừa quản lý dữ liệu chi tiết, vừa phân tích mối quan hệ phức tạp giữa các đối tượng. Một số nhóm bài toán tiêu biểu bao gồm:

1. Bài toán mô phỏng mạng lưới (Network Modeling)

*Ví dụ thực tế:*

* Mạng lưới giao thông.
* Mạng lưới vận chuyển hàng hóa.
* Hệ thống đường dây điện, đường ống.
* Mạng viễn thông.

*Lý do phù hợp:*

* Lưu thông tin nút (điểm / trạm / thiết bị) dưới dạng document, rất linh hoạt.
* Lưu kết nối giữa các điểm bằng edge collection.
* AQL hỗ trợ các truy vấn đồ thị như traverse, shortest path, neighbors,…  
   → Giúp phân tích kết nối, mô phỏng lộ trình, và đánh giá mạng lưới dễ dàng.

1. Bài toán quản lý quan hệ trong tổ chức (Organization & Relationship Management)

*Ví dụ thực tế:*

* Quan hệ nhân sự trong công ty.
* Hệ thống hồ sơ khách hàng (CRM).
* Quan hệ giữa phòng ban - dự án - nhân viên.
* Mạng lưới đối tác, nhà cung cấp.

*Lý do phù hợp:*

* Các thực thể có nhiều thông tin chi tiết → phù hợp với document.
* Các quan hệ giữa thực thể thường phức tạp → phù hợp với graph.

1. Bài toán IoT và hệ thống giám sát (IoT & Device Monitoring)

*Ví dụ thực tế:*

* Mạng lưới thiết bị cảm biến.
* Hệ thống giám sát nhà xưởng.
* Smart buildings, smart city.

*Lý do phù hợp:*

* Sensor gửi dữ liệu JSON linh hoạt → phù hợp document.
* Quan hệ giữa thiết bị – gateway – khu vực → graph.
* Truy vấn theo quan hệ thiết bị rất hiệu quả (neighbors, multi-hop)

# CHƯƠNG 6. PHÂN TÍCH BÀI TOÁN VÀ TRIỂN KHAI ỨNG DỤNG

**6.1. Phân tích và thiết kế hệ thống quản lý xe bus**

**6.1.1. Quy trình hoạt động của hệ thống xe buýt**

Hệ thống quản lý xe buýt là một bài toán phức tạp với nhiều thực thể và mối quan hệ đan xen. Để xây dựng một ứng dụng hiệu quả, nhóm đã phân tích quy trình hoạt động thực tế và xác định các vai trò người dùng chính:

Người Quản trị Hệ thống (Administrator):

* Quản lý tài khoản người dùng (tạo, sửa, xóa, phân quyền).
* Giám sát hoạt động tổng thể của hệ thống.
* Thực hiện các tác vụ bảo trì hệ thống như backup, restore dữ liệu.

Người Quản lý Vận hành (Manager):

* Quản lý cơ sở hạ tầng: Thêm, sửa, xóa thông tin các trạm xe buýt (stations).
* Quản lý tuyến: Thiết kế và quản lý các tuyến xe buýt (routes), bao gồm việc định nghĩa lộ trình (các trạm mà tuyến đi qua - serves), giá vé, tần suất, giờ hoạt động.
* Quản lý phương tiện: Quản lý thông tin đội xe (vehicles), theo dõi tình trạng, lịch bảo trì.

Phân công vận hành: Gán xe cụ thể vào các tuyến (operates\_on) và tạo lịch trình chi tiết (schedules) cho tài xế và xe.

Theo dõi và báo cáo: Xem các thống kê, báo cáo về hiệu suất hoạt động của các tuyến, mức độ sử dụng xe.

Người dùng cuối (End User - Public):

* Sử dụng các tính năng công khai của hệ thống (thông qua ứng dụng di động hoặc website).
* Tìm kiếm lộ trình: Tìm đường đi tối ưu giữa hai điểm dừng, xem các tuyến xe có thể đi, thời gian và chi phí dự kiến.
* Xem thông tin chi tiết về một tuyến xe hoặc một trạm xe.

**6.1.2. Phân tích yêu cầu chức năng và phi chức năng**

A. Yêu cầu chức năng

Hệ thống cần cung cấp các chức năng chính sau:

*F1: Quản lý Trạm (Stations)*: Cung cấp các thao tác CRUD (Tạo, Đọc, Cập nhật, Xóa) cho thông tin các trạm xe buýt, bao gồm tên, địa chỉ, tọa độ, loại trạm, trạng thái, và các tiện ích.

*F2: Quản lý Tuyến (Routes)*: Cung cấp các thao tác CRUD cho các tuyến xe buýt, bao gồm mã tuyến, tên tuyến, giá vé, giờ hoạt động, tần suất. Đặc biệt, cho phép quản lý danh sách các trạm mà tuyến đi qua (quản lý cạnh serves).

*F3: Quản lý Xe (Vehicles)*: Cung cấp các thao tác CRUD cho các phương tiện, bao gồm biển số, loại xe, sức chứa, tình trạng, lịch bảo trì và các tiện nghi trên xe. Cho phép gán xe vào một tuyến (quản lý cạnh operates\_on).

*F4: Quản lý Lịch trình (Schedules)*: Cho phép tạo và quản lý lịch trình chi tiết cho từng chuyến xe, bao gồm giờ xuất phát, tài xế, và ngày hoạt động.

*F5: Quản lý Người dùng và Phân quyền*:

Hệ thống xác thực người dùng qua tên đăng nhập và mật khẩu.

Hệ thống phân quyền truy cập chức năng dựa trên vai trò (Admin, Manager, User).

*F6: Tìm kiếm Lộ trình (Graph Traversal)*: Tính năng cốt lõi, cho phép tìm đường đi ngắn nhất (về khoảng cách hoặc thời gian) giữa hai trạm bất kỳ.

*F7: Thống kê và Báo cáo*: Cung cấp dashboard trực quan với các biểu đồ thống kê về:

* Tổng quan hệ thống (số trạm, tuyến, xe).
* Phân bố trạm theo khu vực.
* Hiệu suất các tuyến (tuyến đông khách, tuyến có độ phủ rộng).
* Tỷ lệ sử dụng phương tiện.

B. Yêu cầu phi chức năng

*NF1: Hiệu năng (Performance)*: Các truy vấn tìm đường, đặc biệt là tìm đường đi ngắn nhất, phải được thực thi nhanh chóng (dưới 2 giây) để đảm bảo trải nghiệm người dùng tốt.

*NF2: Bảo mật (Security)*:

* Mật khẩu người dùng phải được mã hóa (sử dụng bcrypt).
* Truy cập API phải được bảo vệ bằng JWT (JSON Web Tokens).
* Dữ liệu nhạy cảm không được trả về cho các vai trò không có quyền.

*NF3: Dễ sử dụng (Usability)*: Giao diện quản trị phải trực quan, dễ thao tác, có tính năng tìm kiếm, lọc và phân trang để xử lý lượng dữ liệu lớn.

*NF4: Tính sẵn sàng và Bảo trì (Availability & Maintainability)*:

Hệ thống có thể được triển khai dễ dàng thông qua Docker.

Cung cấp cơ chế sao lưu (backup) và phục hồi (restore) dữ liệu.

*NF5: Khả năng mở rộng (Scalability)*: Kiến trúc hệ thống và CSDL phải có khả năng mở rộng để xử lý số lượng trạm, tuyến và yêu cầu truy vấn tăng lên trong tương lai.

**6.1.3. Thiết kế CSDL trên ArangoDB**

Cấu trúc CSDL được thiết kế như sau:

A. Document Collections (Các đỉnh của đồ thị)

stations: Lưu trữ thông tin về các trạm xe buýt.

*{*

*"station\_id": "ST001",*

*"name": "Bến Xe Miền Đông",*

*"address": { "street": "292 Đinh Bộ Lĩnh", "ward": "Phường 26", "city": "TP.HCM" },*

*"location": { "latitude": 10.8142, "longitude": 106.7115 },*

*"type": "terminal", // terminal, intermediate, stop*

*"status": "active" // active, maintenance, inactive*

*}*

routes: Lưu trữ thông tin về các tuyến xe buýt.

*{*

*"route\_id": "R001",*

*"route\_code": "01",*

*"route\_name": "Bến Xe Miền Đông - Bến Xe Chợ Lớn",*

*"type": "normal", // normal, express, rapid*

*"fare": { "adult": 7000, "student": 3500 },*

*"status": "active"*

*}*

vehicles: Lưu trữ thông tin về các xe buýt.

*{*

*"vehicle\_id": "V001",*

*"license\_plate": "51B-10001",*

*"type": "bus\_40", // bus\_16, bus\_40, bus\_60*

*"capacity": 40,*

*"status": "active" // active, maintenance*

*}*

users: Lưu trữ thông tin tài khoản người dùng hệ thống.

*{*

*"username": "admin",*

*"password\_hash": "$2b$...",*

*"role": "admin", // admin, manager, user*

*"full\_name": "Administrator"*

*}*

schedules: Lưu trữ lịch trình chi tiết

*{*

*"route\_id": "R001",*

*"vehicle\_id": "V001",*

*"departure\_time": "05:00",*

*"shift": "morning",*

*"day\_of\_week": ["monday", "tuesday", ...]*

*}*

B. Edge Collections (Các cạnh của đồ thị)

connects: Biểu diễn kết nối vật lý (đoạn đường) giữa hai trạm xe buýt.

*\_from: stations/{id}*

*\_to: stations/{id}*

* Thuộc tính: distance (mét), duration (phút).

serves: Biểu diễn việc một tuyến xe (route) phục vụ một trạm (station).

*\_from: routes/{id}*

*\_to: stations/{id}*

* Thuộc tính: stop\_order (thứ tự dừng), arrival\_offset (thời gian đến tính từ trạm đầu).

operates\_on: Biểu diễn việc một xe (vehicle) được phân công hoạt động trên một tuyến (route).

*\_from: vehicles/{id}*

*\_to: routes/{id}*

* Thuộc tính: shift (ca làm việc), assignment\_date.

C. Graph Definition

Tất cả các collection trên được gom lại thành một đồ thị có tên là *bus\_network* với các định nghĩa cạnh như sau:

*connects: from [stations] to [stations]*

*serves: from [routes] to [stations]*

*operates\_on: from [vehicles] to [routes]*

Hai collection users và schedules được định nghĩa là Orphan Collections (các collection mồ côi), vì chúng không trực tiếp tham gia vào các mối quan hệ đồ thị chính nhưng vẫn thuộc về hệ thống.

6.1.4. Thiết kế kiến trúc ứng dụng

Hệ thống được thiết kế theo kiến trúc 3 lớp (3-Tier Architecture) hiện đại và phổ biến:

1. *Presentation Layer (Frontend)*:

* Công nghệ: Next.js (React Framework), TypeScript, Tailwind CSS, và thư viện component Shadcn/UI.
* Vai trò: Xây dựng giao diện người dùng (UI) cho trang quản trị, xử lý tương tác người dùng, gọi API đến Backend để lấy và gửi dữ liệu, và hiển thị dữ liệu một cách trực quan. Kiến trúc component-based giúp tái sử dụng và bảo trì code dễ dàng.

1. *Business Logic Layer (Backend):*

* Công nghệ: Python với framework Flask, pyArango (ArangoDB driver).
* Vai trò: Cung cấp các RESTful API để Frontend có thể tương tác với CSDL. Lớp này chịu trách nhiệm xử lý logic nghiệp vụ, xác thực và phân quyền người dùng (JWT), và thực thi các truy vấn AQL phức tạp trên ArangoDB.

1. *Data Layer (Database)*:

* Công nghệ: ArangoDB.
* Vai trò: Lưu trữ và quản lý toàn bộ dữ liệu của hệ thống dưới dạng multi-model (document và graph). Chịu trách nhiệm thực thi các truy vấn, đặc biệt là các truy vấn đồ thị hiệu năng cao.

**6.2. Triển khai xây dựng hệ thống**

**6.2.1. Xây dựng và Chuẩn bị Dữ liệu**

Việc chuẩn bị dữ liệu đóng vai trò nền tảng cho toàn bộ hệ thống. Thay vì nhập liệu thủ công dễ sai sót, nhóm đã xây dựng một quy trình tự động hóa bằng Python script để tạo ra bộ dữ liệu mẫu phong phú và thực tế.

**A. Cấu trúc dữ liệu chuẩn bị**

Dữ liệu được thiết kế bám sát mô hình thực tế của hệ thống xe buýt TP.HCM, bao gồm:

* 30 Trạm xe buýt (stations): Đa dạng loại hình (bến xe đầu mối, trạm trung gian, điểm dừng), phân bố rộng khắp các quận huyện, với thông tin chi tiết về địa chỉ (cập nhật theo quy hoạch mới), tọa độ GPS và tiện ích.
* 15 Tuyến xe buýt (routes): Bao gồm các tuyến trục chính, tuyến vành đai và tuyến gom, với thông tin về lộ trình, tần suất, giá vé và giờ hoạt động.
* 40 Xe buýt (vehicles): Đội xe đa dạng về chủng loại (16-60 chỗ), hãng sản xuất và trạng thái hoạt động.
* Người dùng (users): Tài khoản mẫu cho các vai trò Admin, Manager và User để phục vụ kiểm thử phân quyền.

**B. Tạo quan hệ đồ thị (Edges)**

Sức mạnh của ArangoDB nằm ở khả năng xử lý đồ thị. Nhóm đã tạo ra hàng trăm mối quan hệ để mô phỏng mạng lưới giao thông:

connects (100+ cạnh): Mô phỏng mạng lưới đường đi vật lý giữa các trạm, bao gồm cả khoảng cách và thời gian di chuyển ước tính. Đặc biệt chú trọng tạo các cạnh hai chiều để phục vụ thuật toán tìm đường chính xác.

serves (70+ cạnh): Định nghĩa chi tiết lộ trình của từng tuyến xe đi qua các trạm nào, theo thứ tự ra sao (stop\_order).

operates\_on (35+ cạnh): Gán xe cụ thể vào các tuyến hoạt động theo ca, tạo cơ sở dữ liệu cho việc quản lý lịch trình.

**C. Thực thi Script**

Script insert\_full\_data.py được viết bằng Python sử dụng thư viện pyArango. Script này thực hiện các bước:

* Kết nối đến database bus\_management\_hcm.
* Xóa sạch dữ liệu cũ (nếu có) để đảm bảo tính nhất quán (Clean Slate).
* Lần lượt insert dữ liệu vào các Document Collections và Edge Collections.
* Sinh dữ liệu schedules tự động dựa trên phân công xe.

**6.2.2. Import/Export và Backup/Restore Dữ liệu**

Để đảm bảo an toàn dữ liệu và khả năng di chuyển hệ thống, các quy trình sao lưu và phục hồi được thực hiện thông qua Docker.

Lệnh thực hiện:

“docker exec bus-management-arangodb arangodump ^

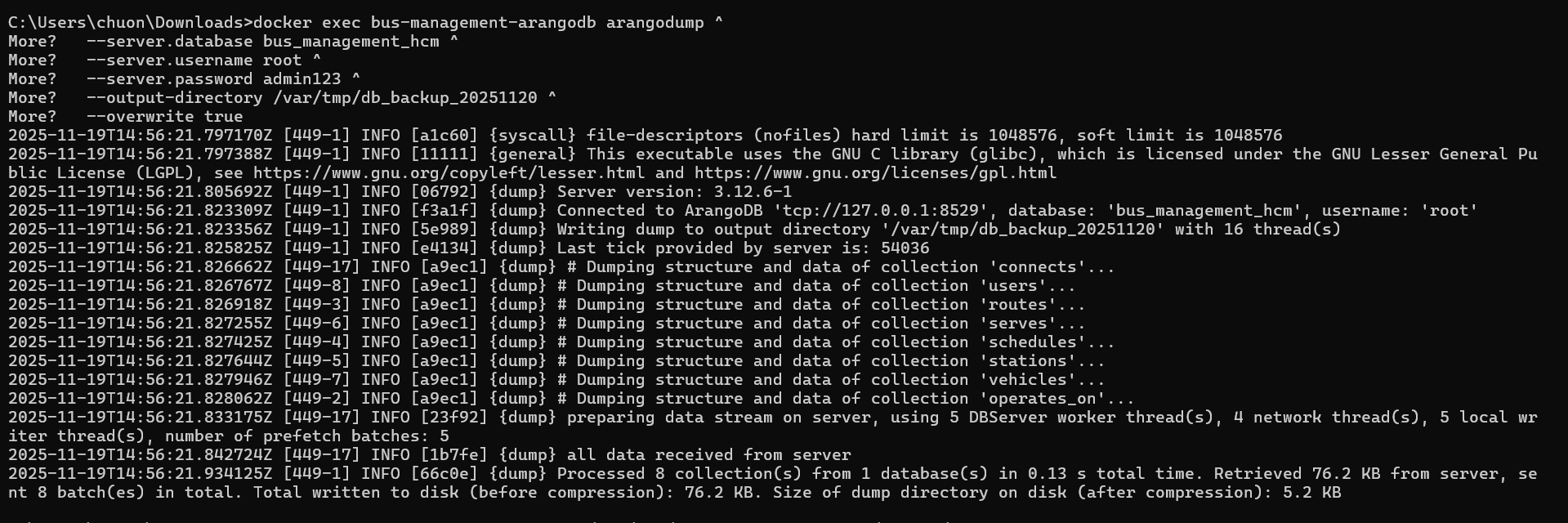
--server.database bus\_management\_hcm ^

--server.username root ^

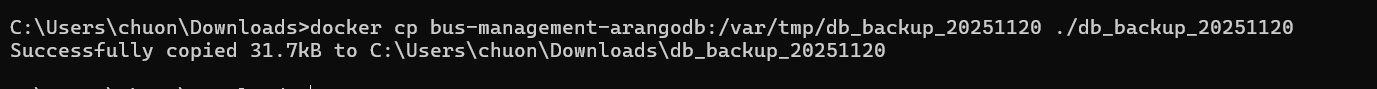
--server.password admin123 ^

--output-directory /var/tmp/db\_backup\_20251120 ^

--overwrite true” (1)



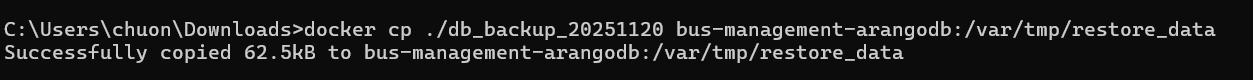
“docker cp bus-management-arangodb:/var/tmp/db\_backup\_20251120 ./db\_backup\_20251120” (2)



1. **Restore dữ liệu**

Sử dụng công cụ arangorestore để khôi phục dữ liệu từ bản sao lưu khi cần thiết.

“docker cp ./db\_backup\_20251120 bus-management-arangodb:/var/tmp/restore\_data” (1)



“docker exec -it bus-management-arangodb arangorestore ^

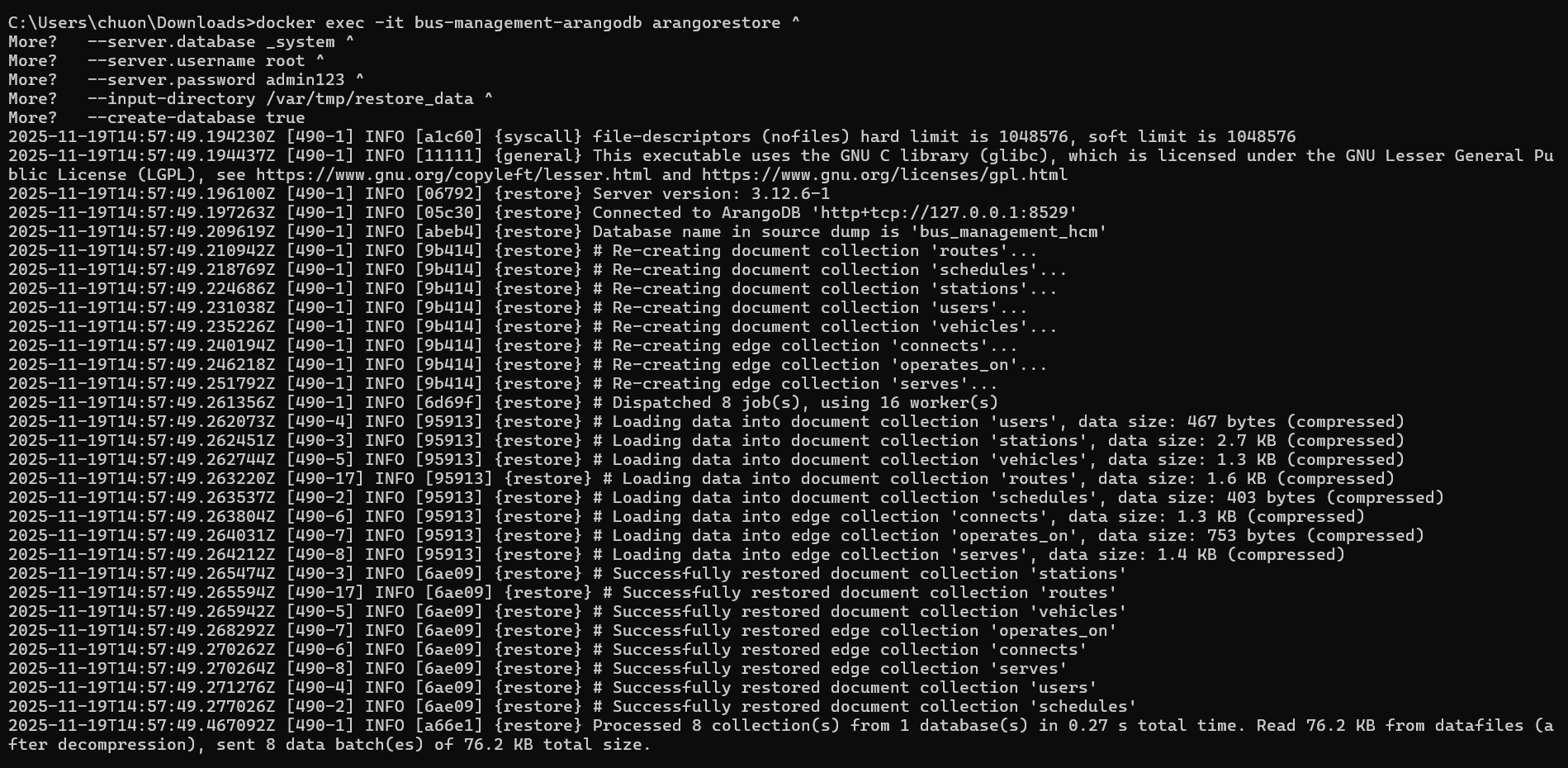
--server.database \_system ^

--server.username root ^

--server.password admin123 ^

--input-directory /var/tmp/restore\_data ^

--create-database true” (2)



**6.2.3. Triển khai các Truy vấn trên ArangoDB Web Interface**

6.2.3. Triển khai các Truy vấn trên ArangoDB Web Interface

ArangoDB Web Interface cung cấp công cụ AQL Editor mạnh mẽ, cho phép phát triển và kiểm thử các câu truy vấn trước khi tích hợp vào mã nguồn ứng dụng. Để giải quyết bài toán quản lý xe buýt phức tạp, nhóm đã xây dựng và triển khai hệ thống truy vấn theo ba cấp độ: Phức tạp, Nâng cao và Chuyên sâu về Đồ thị.

Dưới đây là chi tiết các kịch bản truy vấn đã được triển khai và kiểm thử thành công:

**A. Cấp độ 1: Truy vấn Phức tạp (Complex Queries)**

Mục tiêu: Kết hợp dữ liệu từ nhiều collection khác nhau (Joins), lọc và sắp xếp dữ liệu đa điều kiện.

Kịch bản 1: Lấy chi tiết lịch trình hoạt động của một tuyến xe

Truy vấn này cần kết hợp thông tin từ 3 collection: schedules (lịch trình), routes (thông tin tuyến) và vehicles (thông tin xe) để hiển thị đầy đủ cho người quản lý.

*“FOR schedule IN schedules*

*FILTER schedule.route\_id == "R001" // Lọc theo tuyến số 01*

*// Join với collection routes để lấy tên tuyến*

*LET route = FIRST(*

*FOR r IN routes*

*FILTER r.route\_id == schedule.route\_id*

*RETURN r*

*)*

*// Join với collection vehicles để lấy biển số xe*

*LET vehicle = FIRST(*

*FOR v IN vehicles*

*FILTER v.vehicle\_id == schedule.vehicle\_id*

*RETURN v*

*)*

*SORT schedule.departure\_time ASC // Sắp xếp theo giờ xuất bến*

*RETURN {*

*route\_name: route.route\_name,*

*license\_plate: vehicle.license\_plate,*

*departure\_time: schedule.departure\_time,*

*driver: schedule.driver.name,*

*shift: schedule.shift*

}”

→ Kết quả: Danh sách các chuyến xe trong ngày của tuyến R001, kèm theo thông tin tài xế và xe cụ thể.

**B. Cấp độ 2: Truy vấn Nâng cao (Advanced Queries)**

Mục tiêu: Sử dụng các hàm tổng hợp (Aggregation), gom nhóm (Grouping) và phân tích dữ liệu thống kê.

Kịch bản 2: Thống kê mức độ bao phủ của các tuyến xe theo quận/huyện

Truy vấn này giúp nhà quản lý biết được mỗi tuyến xe đi qua bao nhiêu quận và tổng số trạm dừng, từ đó đánh giá độ phủ sóng của tuyến.

“FOR route IN routes

// Duyệt qua các cạnh 'serves' để tìm tất cả các trạm thuộc tuyến này

LET stations = (

FOR v, e IN OUTBOUND route serves

RETURN v

)

// Lấy danh sách các quận duy nhất mà tuyến đi qua

LET districts = UNIQUE(stations[\*].address.district)

SORT LENGTH(stations) DESC // Sắp xếp theo số lượng trạm giảm dần

RETURN {

route\_code: route.route\_code,

route\_name: route.route\_name,

total\_stops: LENGTH(stations), // Tổng số trạm dừng

districts\_count: LENGTH(districts), // Số lượng quận đi qua

districts\_list: districts // Danh sách tên các quận

}”

Kết quả: Bảng thống kê cho thấy tuyến nào dài nhất, đi qua nhiều khu vực hành chính nhất.

Kịch bản 3: Tìm Top 5 trạm xe buýt "bận rộn" nhất

Xác định các trạm trung chuyển quan trọng bằng cách đếm số lượng tuyến xe đi qua trạm đó.

“FOR station IN stations

// Đếm số lượng cạnh 'serves' hướng vào trạm này (Inbound edges)

LET route\_count = LENGTH(

FOR route, edge IN INBOUND station serves

RETURN 1

)

SORT route\_count DESC

LIMIT 5

RETURN {

station\_name: station.name,

address: station.address.street,

total\_routes: route\_count

}”

**C. Cấp độ 3: Truy vấn Cực Nâng cao - Graph Traversal (Deep Graph Queries)**

Mục tiêu: Khai thác sức mạnh cốt lõi của Graph Database để giải quyết các bài toán tìm đường, tối ưu lộ trình mà SQL truyền thống rất khó thực hiện.

Kịch bản 4: Tìm lộ trình ngắn nhất giữa hai địa điểm bất kỳ (Shortest Path)

Đây là tính năng quan trọng nhất của ứng dụng cho người dùng cuối. Hệ thống tìm đường đi ngắn nhất dựa trên độ dài quãng đường (distance) giữa hai trạm.

“// Định nghĩa điểm đầu và điểm cuối

LET start\_station = (FOR s IN stations FILTER s.station\_id == "ST001" RETURN s.\_id)[0]

LET end\_station = (FOR s IN stations FILTER s.station\_id == "ST010" RETURN s.\_id)[0]

// Sử dụng thuật toán Shortest Path có sẵn của ArangoDB

FOR v, e IN OUTBOUND SHORTEST\_PATH start\_station TO end\_station connects

RETURN {

station: v.name,

stop\_address: v.address.street,

distance\_to\_next: e.distance // Khoảng cách đến trạm kế tiếp

}”

Kịch bản 5: Tìm tất cả các lộ trình thay thế (K-Shortest Paths) với thông tin chi tiết

Không chỉ tìm một đường, hệ thống cung cấp 3 lựa chọn lộ trình tốt nhất để người dùng so sánh. Truy vấn này cực kỳ phức tạp vì phải tính tổng khoảng cách và thời gian cho toàn bộ hành trình.

“LET start\_id = "stations/ST001" // Bến xe Miền Đông

LET end\_id = "stations/ST003" // Chợ Bến Thành

FOR path IN OUTBOUND K\_SHORTEST\_PATHS start\_id TO end\_id connects

LIMIT 3 // Lấy 3 lộ trình tốt nhất

// Tính toán tổng hợp cho mỗi lộ trình

LET total\_distance = SUM(path.edges[\*].distance)

LET total\_duration = SUM(path.edges[\*].duration)

RETURN {

path\_index: 1, // Đánh số lộ trình

stops: path.vertices[\*].name, // Danh sách tên các trạm đi qua

total\_distance\_km: total\_distance / 1000, // Quy đổi ra km

estimated\_time\_minutes: total\_duration,

number\_of\_stops: LENGTH(path.vertices) - 1 // Số trạm phải dừng

}”

Kịch bản 6: Truy vấn mạng lưới - Tìm các trạm có thể tiếp cận trong vòng N bước

Giúp quản lý phân tích vùng phục vụ: Từ một trạm trung tâm, xe buýt có thể đi đến những đâu trong vòng 2 chặng dừng mà không cần đổi tuyến?

“FOR v, e, p IN 1..2 OUTBOUND "stations/ST003" connects

OPTIONS {bfs: true, uniqueVertices: 'global'} // Duyệt theo chiều rộng, không lặp lại đỉnh

RETURN {

destination: v.name,

steps: LENGTH(p.edges), // Số chặng

path: p.vertices[\*].name // Lộ trình cụ thể

}”

**6.2.4. Triển khai Backend và Kết nối CSDL**

Backend của hệ thống được xây dựng bằng ngôn ngữ Python trên nền tảng framework Flask. Đây là lớp trung gian quan trọng, chịu trách nhiệm xử lý các yêu cầu từ phía người dùng (Frontend), thực thi logic nghiệp vụ, xác thực bảo mật và tương tác trực tiếp với cơ sở dữ liệu ArangoDB.

**A. Cài đặt môi trường phát triển**

Nhóm sử dụng môi trường ảo (Virtual Environment) để quản lý các thư viện phụ thuộc, đảm bảo tính nhất quán và tránh xung đột phiên bản.

**Bước 1**: Khởi tạo môi trường ảo

python -m venv venv

source venv/bin/activate # Trên Linux/Mac

# hoặc venv\Scripts\activate trên Windows

**Bước 2**: Cài đặt các thư viện cần thiết

Các thư viện chính được sử dụng bao gồm:

flask: Framework web nhẹ và linh hoạt.

pyArango: Driver chính thức để kết nối và thao tác với ArangoDB từ Python.

flask-jwt-extended: Thư viện hỗ trợ xác thực người dùng bằng JSON Web Token (JWT).

python-dotenv: Quản lý biến môi trường.

**B. Cấu trúc Project (Backend):**

backend/

├── app/

│ ├── models/ # Định nghĩa cấu trúc dữ liệu (Station, Route...)

│ ├── routes/ # Định nghĩa API Endpoints (GET, POST, PUT, DELETE)

│ ├── utils/ # Tiện ích kết nối DB (db\_connection.py)

│ └── \_\_init\_\_.py # Khởi tạo Flask App

├── scripts/ # Các script tạo dữ liệu (insert\_full\_data.py)

├── .env # Biến môi trường (DB Host, Password, Secret Key)

└── run.py # File khởi chạy server

1. **Thiết lập kết nối CSDL:**

Sử dụng mẫu thiết kế Singleton để quản lý kết nối ArangoDB, đảm bảo chỉ có một instance kết nối được tạo ra và tái sử dụng cho toàn bộ ứng dụng, giúp tiết kiệm tài nguyên.

1. **Triển khai API:**

Các API được tổ chức theo kiến trúc RESTful, trả về dữ liệu định dạng JSON. Ví dụ API lấy danh sách trạm có hỗ trợ phân trang: GET /api/stations?page=1&limit=10

1. **Khởi chạy Server:**

Trong CMD ta dùng lênh “*python run.py*” để khởi chạy API BackEnd

# Server sẽ lắng nghe tại http://localhost:5000

**6.2.5. Triển khai Frontend**

Frontend là giao diện tương tác người dùng, được xây dựng để đảm bảo tính thẩm mỹ và trải nghiệm mượt mà.

**A. Cài đặt môi trường:**

Framework: Next.js 14 (App Router)

Thư viện UI: Shadcn/UI, Tailwind CSS, Lucide React (Icons).

Cài đặt: npx create-next-app@latest và npm install các thư viện cần thiết.

1. **Cấu trúc Project (Frontend):**

frontend/

├── src/

│ ├── app/ # Chứa các Page (Dashboard, Stations, Routes...)

│ ├── components/ # Các UI Component tái sử dụng (Card, Table, Modal...)

│ ├── lib/ # API Client (api.ts) cấu hình Axios

│ ├── types/ # Định nghĩa TypeScript Interfaces

│ └── hooks/ # Custom Hooks (useAuth, useStations...)

└── .env.local # Cấu hình API Endpoint (NEXT\_PUBLIC\_API\_URL)

**C. Triển khai các tính năng nâng cao:**

Bản đồ số: Tích hợp thư viện Vietmap API để hiển thị trực quan vị trí các trạm xe buýt trên bản đồ thực tế.

Biểu đồ: Sử dụng Recharts để vẽ các biểu đồ thống kê động tại trang Dashboard.

**D. Build và Chạy ứng dụng:**

Trong CMD chạy lệnh “*npm run dev*” (chạy tại http://localhost:3000)

# CHƯƠNG 7. KẾT QUẢ TRIỂN KHAI VÀ MINH HỌA ỨNG DỤNG

**7.1. Giới thiệu giao diện và các chức năng chính**

Giao diện ứng dụng được thiết kế theo phong cách hiện đại, tối giản và tập trung vào trải nghiệm người dùng (User Centric), sử dụng bộ thư viện Shadcn/UI.

**A. Trang Đăng nhập (Login)**

Đây là cổng vào duy nhất của hệ thống quản trị. Giao diện yêu cầu xác thực người dùng để đảm bảo an ninh.

Mô tả: Form đăng nhập đơn giản gồm tên đăng nhập và mật khẩu. Hệ thống hỗ trợ thông báo lỗi tức thì nếu thông tin sai.

[CHÈN HÌNH 7.1.1: Giao diện trang Đăng nhập]

**B. Bảng điều khiển trung tâm (Dashboard)**

Sau khi đăng nhập thành công, người dùng được chuyển đến Dashboard. Đây là nơi cung cấp cái nhìn tổng quan nhất về tình trạng hệ thống.

Mô tả:

Thẻ thống kê (Stat Cards): Hiển thị các con số quan trọng như Tổng số trạm, Số tuyến đang hoạt động, Số lượng xe, và Tổng người dùng.

Biểu đồ trực quan: Biểu đồ tròn thể hiện tỷ lệ các loại trạm (Terminal/Stop) và biểu đồ cột thể hiện phân bố trạm theo quận/huyện.

[CHÈN HÌNH 7.1.2: Giao diện Dashboard với các biểu đồ thống kê]

**C. Thanh điều hướng (Sidebar)**

Thanh bên trái chứa menu điều hướng đến các phân hệ chức năng. Menu này hiển thị động dựa trên quyền hạn của người dùng (Admin thấy tất cả, User chỉ thấy các mục cơ bản).

[CHÈN HÌNH 7.1.3: Thanh Sidebar với đầy đủ các mục menu chức năng]

**7.2. Demo chức năng quản trị người dùng**

Chức năng này chỉ dành riêng cho tài khoản có quyền Administrator.

**A. Danh sách người dùng**

Thao tác: Truy cập menu "Quản lý User".

Kết quả: Hiển thị bảng danh sách tất cả tài khoản trong hệ thống với các cột: Tên đăng nhập, Họ tên, Email, Vai trò (Role), Trạng thái.

[CHÈN HÌNH 7.2.1: Bảng danh sách người dùng]

**B. Tạo người dùng mới và Phân quyền**

Thao tác: Nhấn nút "Thêm người dùng", điền thông tin và chọn vai trò (Admin/Manager/User).

Kết quả: Người dùng mới được tạo thành công và xuất hiện ngay lập tức trên danh sách. Hệ thống backend tự động mã hóa mật khẩu trước khi lưu xuống ArangoDB.

[CHÈN HÌNH 7.2.2: Form thêm mới người dùng với dropdown chọn Vai trò]

**7.3. Demo chức năng thao tác CSDL (Quản lý Trạm & Tuyến)**

Đây là nhóm chức năng cốt lõi thể hiện khả năng tương tác CRUD với cơ sở dữ liệu đồ thị.

**A. Quản lý Trạm xe buýt (Stations)**

Xem danh sách: Hiển thị danh sách trạm dưới dạng lưới (Grid) các thẻ (Card) thông tin. Hỗ trợ tìm kiếm trạm theo tên hoặc địa chỉ.

[CHÈN HÌNH 7.3.1: Danh sách các trạm xe buýt dạng Card]

Bản đồ số: Tích hợp bản đồ Leaflet hiển thị vị trí trực quan của các trạm dựa trên tọa độ GPS lưu trong CSDL.

[CHÈN HÌNH 7.3.2: Bản đồ hiển thị các marker vị trí trạm]

Thêm/Sửa trạm: Form nhập liệu chi tiết bao gồm thông tin hành chính (Tỉnh/Phường) và các tiện ích đi kèm (Wifi, WC...). Dữ liệu địa chỉ được chuẩn hóa theo quy định mới (không còn Quận).

[CHÈN HÌNH 7.3.3: Giao diện Form thêm mới trạm xe buýt]

**B. Quản lý Tuyến xe (Routes)**

Chi tiết tuyến: Trang xem chi tiết một tuyến xe không chỉ hiện thông tin cơ bản mà còn liệt kê danh sách các trạm dừng theo thứ tự (stop\_order). Đây là kết quả của truy vấn đồ thị duyệt qua cạnh serves.

[CHÈN HÌNH 7.3.4: Trang chi tiết tuyến xe buýt với danh sách trạm dừng]

Quản lý lộ trình: Giao diện cho phép kéo thả để sắp xếp thứ tự các trạm trong lộ trình, thêm hoặc xóa trạm khỏi tuyến.

[CHÈN HÌNH 7.3.5: Giao diện chỉnh sửa lộ trình (Drag & Drop)]

**7.4. Demo chức năng quản lý theo mô tả bài toán**

Phần này minh họa các chức năng đặc thù của bài toán quản lý vận tải hành khách công cộng.

**A. Quản lý Phương tiện (Vehicles) và Phân công**

Danh sách xe: Quản lý đội xe với thông tin chi tiết về biển số, loại xe, sức chứa và tình trạng bảo trì.

Phân công vận hành: Chức năng gán một xe cụ thể vào một tuyến đường (operates\_on). Hệ thống đảm bảo một xe không thể hoạt động trên hai tuyến cùng lúc trong cùng một ca.

[CHÈN HÌNH 7.4.1: Giao diện danh sách xe và trạng thái hoạt động]

[CHÈN HÌNH 7.4.2: Giao diện phân công xe vào tuyến]

**B. Tìm kiếm Lộ trình (Journey Planner) - Ứng dụng Graph Traversal**

Đây là tính năng minh họa rõ nét nhất sức mạnh của ArangoDB.

Kịch bản: Người dùng chọn điểm đi là "Bến Xe Miền Đông" và điểm đến là "Chợ Bến Thành".

Kết quả: Hệ thống sử dụng thuật toán K\_SHORTEST\_PATHS để tìm và hiển thị 3 lộ trình tối ưu nhất. Kết quả hiển thị rõ ràng bao gồm: Tổng quãng đường, Thời gian dự kiến, Danh sách các trạm phải đi qua và Tuyến xe cần bắt.

[CHÈN HÌNH 7.4.3: Giao diện nhập điểm đi và điểm đến]

[CHÈN HÌNH 7.4.4: Kết quả tìm kiếm lộ trình hiển thị trực quan trên giao diện]

**7.5. Demo các chức năng hỗ trợ khác**

**A. AQL Query Runner (Trang Truy vấn)**

Một công cụ dành cho người quản trị kỹ thuật để chạy trực tiếp các câu lệnh AQL ngay trên giao diện ứng dụng mà không cần vào ArangoDB Web Interface.

Chức năng: Cung cấp sẵn các mẫu truy vấn (Templates) và khu vực hiển thị kết quả dạng bảng hoặc JSON.

[CHÈN HÌNH 7.5.1: Giao diện trang thực thi truy vấn AQL]

**B. Thống kê và Phân tích (Analytics)**Trang báo cáo chuyên sâu cung cấp các số liệu quan trọng để ra quyết định.  
Biểu đồ hiệu suất: Thống kê Top 10 trạm bận rộn nhất (có nhiều tuyến đi qua).  
Biểu đồ độ phủ: Phân tích mức độ bao phủ của các tuyến xe theo khu vực địa lý (Phường/Xã).

# [CHÈN HÌNH 7.5.2: Trang Analytics với các biểu đồ phân tích chuyên sâu]

# CHƯƠNG 8. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

**8.1. Đánh giá so với mục tiêu ban đầu**

Đồ án "Xây dựng ứng dụng Quản lý lộ trình xe buýt công cộng tại TP.HCM sử dụng ArangoDB" được khởi động với mục tiêu nghiên cứu ứng dụng công nghệ CSDL đồ thị (Graph Database) vào bài toán thực tế về giao thông công cộng. Sau quá trình triển khai, nhóm thực hiện đã đạt được những kết quả cụ thể sau:

**1. Về mặt nghiên cứu công nghệ:**

Mục tiêu: Tìm hiểu sâu về ArangoDB, mô hình dữ liệu đa mô hình (Multi-model) và ngôn ngữ truy vấn AQL.

Kết quả: Nhóm đã nắm vững kiến trúc của ArangoDB, cách thiết kế schema cho bài toán đồ thị phức tạp. Đã thành thạo việc sử dụng AQL để thực hiện từ các truy vấn CRUD cơ bản đến các thuật toán đồ thị nâng cao như Shortest Path (đường đi ngắn nhất) và Graph Traversal (duyệt đồ thị). Việc so sánh hiệu năng và tính năng với các CSDL quan hệ truyền thống cũng đã được thực hiện thông qua thực nghiệm.

**2. Về mặt ứng dụng thực tiễn:**

Mục tiêu: Xây dựng một ứng dụng Web hoàn chỉnh cho phép quản lý và khai thác dữ liệu xe buýt.

Kết quả: Hệ thống đã được xây dựng hoàn thiện với kiến trúc hiện đại (Next.js + Flask + ArangoDB), bao gồm đầy đủ các phân hệ:

Quản trị hệ thống: Quản lý trạm, tuyến, phương tiện, lịch trình và người dùng với phân quyền chặt chẽ (Admin/Manager/User).

Tìm kiếm thông minh: Tính năng "Tìm lộ trình" hoạt động hiệu quả, giúp người dùng tìm đường đi tối ưu giữa hai điểm bất kỳ dựa trên dữ liệu đồ thị thực tế.

Báo cáo thống kê: Dashboard trực quan cung cấp cái nhìn tổng quan về hoạt động của hệ thống vận tải.

**3. Về mặt dữ liệu:**

Mục tiêu: Xây dựng bộ dữ liệu mẫu sát với thực tế.

Kết quả: Đã xây dựng thành công bộ dữ liệu gồm hơn 30 trạm, 15 tuyến và hàng trăm mối quan hệ kết nối, mô phỏng chính xác mạng lưới giao thông tại các khu vực trọng điểm của TP.HCM. Dữ liệu địa chỉ đã được cập nhật theo quy hoạch hành chính mới nhất (loại bỏ cấp Quận).

Đánh giá chung: Đồ án đã hoàn thành 100% các yêu cầu cơ bản và khoảng 90% các yêu cầu nâng cao đặt ra ban đầu. Hệ thống vận hành ổn định, đáp ứng tốt các kịch bản sử dụng thực tế.

**8.2. Những hạn chế của hệ thống**

Mặc dù đã đạt được những kết quả tích cực, hệ thống vẫn còn tồn tại một số hạn chế do giới hạn về thời gian thực hiện và nguồn lực:

1. Hạn chế về dữ liệu bản đồ:

Hiện tại, khoảng cách và thời gian di chuyển giữa các trạm (distance, duration) trong CSDL là dữ liệu tĩnh, được ước lượng hoặc nhập liệu cố định. Hệ thống chưa tích hợp được với các API bản đồ thực tế (như Google Maps API hay OpenStreetMap) để tính toán khoảng cách thực tế theo tình hình giao thông thời gian thực (Real-time traffic).

2. Hạn chế về thuật toán tìm đường:

Thuật toán tìm đường hiện tại chủ yếu dựa trên khoảng cách ngắn nhất (distance). Trong thực tế, hành khách đi xe buýt quan tâm nhiều hơn đến tổng thời gian di chuyển (bao gồm thời gian chờ xe, thời gian chuyển tuyến) và chi phí. Hệ thống chưa hỗ trợ tìm đường đa tiêu chí (Multi-criteria routing) một cách hoàn hảo.

Chưa hỗ trợ tìm lộ trình kết hợp đa phương thức (ví dụ: đi bộ đến trạm A -> đi xe buýt đến trạm B -> đi Grab đến đích).

**3. Hạn chế về giao diện người dùng cuối (End-user App):**

Đồ án hiện tại tập trung nhiều vào giao diện quản trị (Admin Dashboard). Phân hệ dành cho người đi xe buýt (tìm đường, xem lịch trình) tuy đã có nhưng chưa được phát triển thành một ứng dụng di động (Mobile App) chuyên biệt, làm giảm tính tiện dụng khi di chuyển.

**4. Hạn chế về khả năng chịu tải:**

Hệ thống chưa được kiểm thử chịu tải (Load Testing) với lượng dữ liệu lớn (Big Data) quy mô toàn thành phố (hàng ngàn trạm, hàng triệu người dùng) để đánh giá chính xác khả năng mở rộng (Scalability) của ArangoDB trong môi trường Cluster.

**8.3. Đề xuất cải tiến và mở rộng**

Dựa trên những hạn chế đã phân tích, nhóm đề xuất các hướng phát triển tiếp theo để hoàn thiện hệ thống:

* 1. Tích hợp dữ liệu thời gian thực (Real-time Data):

Tích hợp thiết bị GPS trên xe buýt để cập nhật vị trí xe theo thời gian thực về hệ thống.

Sử dụng dữ liệu này để cung cấp tính năng "Xe sắp đến" (ETA - Estimated Time of Arrival) chính xác cho người dùng, giống như các ứng dụng gọi xe công nghệ.

* 2. Nâng cấp thuật toán tìm đường thông minh:

Phát triển thuật toán tìm đường đa tiêu chí, cho phép người dùng tùy chọn ưu tiên: "Rẻ nhất", "Nhanh nhất", "Ít phải đi bộ nhất" hoặc "Ít lần chuyển tuyến nhất".

Sử dụng tính năng SmartGraphs hoặc SatelliteGraphs của ArangoDB Enterprise để tối ưu hóa hiệu năng truy vấn khi dữ liệu mở rộng ra quy mô lớn.

* 3. Phát triển ứng dụng di động (Mobile App):

Xây dựng ứng dụng native trên Android/iOS sử dụng React Native hoặc Flutter, giúp người dùng dễ dàng tra cứu lộ trình ngay trên điện thoại khi đang di chuyển. Tích hợp tính năng thông báo (Push Notification) khi xe sắp đến trạm.

* 4. Ứng dụng AI và Học máy (Machine Learning):

Sử dụng lịch sử di chuyển để gợi ý lộ trình cá nhân hóa cho người dùng.

Phân tích dữ liệu lưu lượng hành khách để tối ưu hóa lịch trình chạy xe, tần suất xuất bến, giúp đơn vị vận hành tiết kiệm chi phí và nâng cao hiệu quả phục vụ.

**8.4. Kết luận chung**

Đồ án này là một minh chứng rõ ràng cho thấy sự phù hợp và sức mạnh của ArangoDB và mô hình dữ liệu đồ thị trong việc giải quyết các bài toán thực tế có tính kết nối cao như hệ thống giao thông vận tải.

Việc chuyển đổi từ tư duy quản lý dữ liệu dạng bảng (Relational) sang dạng đồ thị (Graph) không chỉ giúp việc mô hình hóa dữ liệu trở nên tự nhiên, trực quan hơn mà còn mang lại hiệu năng vượt trội cho các truy vấn phức tạp về tìm đường và phân tích mạng lưới.

Kiến trúc hệ thống hiện đại kết hợp giữa Next.js, Python Flask và ArangoDB đã tạo ra một nền tảng vững chắc, linh hoạt, dễ dàng mở rộng và bảo trì. Đây là tiền đề quan trọng để phát triển các ứng dụng giao thông thông minh (Smart Mobility) trong tương lai, góp phần vào công cuộc chuyển đổi số và xây dựng thành phố thông minh tại Việt Nam.

Thông qua đồ án, nhóm thực hiện không chỉ nâng cao được kiến thức chuyên môn về NoSQL và công nghệ phần mềm mà còn rèn luyện được kỹ năng phân tích, giải quyết vấn đề và làm việc nhóm hiệu quả.

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | N. Antonio, A. de Almeida and L. Nunes, "Hotel booking demand datasets," ScienceDirect, 11 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352340918315191. [Accessed 02 11 2025]. |
| [2] | ResearchGate, "Hotel booking demand datasets (open access)," ResearchGate, 2019. [Online]. Available: https://www.researchgate.net/publication/329286343\_Hotel\_booking\_demand\_datasets. [Accessed 02 11 2025]. |
| [3] | arXiv.org, "Hotel Booking Cancellation Prediction Using Applied Bayesian Models," arXiv, 10 2024. [Online]. Available: https://arxiv.org/html/2410.16406v2. [Accessed 02 11 2025]. |
| [4] | A. Ramesh, "Exploratory Data Analysis of the Hotel Booking Demand with Python," Medium, 7 2022. [Online]. Available: https://medium.com/analytics-vidhya/exploratory-data-analysis-of-the-hotel-booking-demand-with-python-200925230106. [Accessed 02 11 2025]. |
| [5] | N. D. S. Academy, "Hotel Bookings Data Analysis Project," NYC Data Science, 3 2021. [Online]. Available: https://nycdatascience.com/blog/r/hotel-bookings-data-analysis. [Accessed 02 11 2025]. |
| [6] | IJARSCT, "Real Time Hotel Booking Demand Optimization," International Journal of Advanced Research in Science, Communication and Technology, 6 2023. [Online]. Available: https://ijarsct.co.in/Paper17978.pdf. [Accessed 02 11 2025]. |
| [7] | D. S. Journal, "An Automated Machine Learning Based Decision Support System to predict hotel booking cancellations," CODATA, 2019. [Online]. Available: https://datascience.codata.org/articles/10.5334/dsj-2019-032. [Accessed 02 11 2025]. |
| [8] | A. S. Foundation, "Apache Spark – Unified analytics engine for large-scale data processing," Apache Software Foundation, 12 2024. [Online]. Available: https://spark.apache.org/. [Accessed 02 11 2025]. |
| [9] | A. Bhardwaj, "A Comprehensive Guide to K-Means Clustering," Analytics Vidhya, 8 2019. [Online]. Available: https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/. [Accessed 02 11 2025]. |