**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**



**TIỂU LUẬN NỘI DUNG:**

**TRÌNH BÀY VỀ VIỆC LƯU TRỮ VÀ PHÂN TÁN CỦA HQTCSDL NO SQL**

**Sinh viên thực hiện:** Lê Văn Long-19521783

Phạm Hoàng Phúc- 19520854

**Lớp:** IS211.M12.HTCL

**Giảng viên:**ThS. Nguyễn Hồ Duy Tri

Mục lục

[PHẦN 1: GIỚI THIỆU VỀ HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU ARANGODB: 4](#_Toc89709639)

[Ngôn ngữ thao tác dữ liệu: 12](#_Toc89709640)

[5) Cơ chế phân tán của ArangoDB: 17](#_Toc89709641)

[a) Giới thiệu chung 17](#_Toc89709642)

[b) Shard Keys 19](#_Toc89709643)

[c) Configuring Shards 21](#_Toc89709644)

[PHẦN 2: MÔ PHỎNG CÀI ĐẶT VÀ PHÂN TÁN TRONG ARANGODB 22](#_Toc89709645)

[1. HƯỚNG DẪN CÀI ĐẶT: 23](#_Toc89709646)

[a) Cài đặt trên 1 máy 23](#_Toc89709647)

[b. Cài đặt trên cụm máy phân tán: 25](#_Toc89709648)

[2. Thực nghiệm mô phỏng phân tán: 30](#_Toc89709649)

[a) Tạo database và thêm dữ liệu: 30](#_Toc89709650)

[b) Phân quyền: 38](#_Toc89709651)

[c) Tạo Index: 43](#_Toc89709652)

[b) Demo query trong bài toán của nhóm: 46](#_Toc89709653)

## PHẦN 1: GIỚI THIỆU VỀ HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU ARANGODB:

**1) Đặt vấn đề**

Với sự phát triển không ngừng của ngành công nghệ thông tin. Khối dữ liệu cần xử lý trong các ứng dụng là rất lớn. Nơi các mạng dịch vụ dữ liệu cộng đồng cho phép người dùng tự do tạo nội dung trên web, dẫn đến dữ liệu tăng lên rất nhanh, vượt qua giới hạn xử lý của các Hệ quản trị cơ sở dữ liệu quan hệ truyền thống. Để đáp ứng được nhu cầu phát triển của xã hội, đòi hỏi một cơ sở dữ liệu (CSDL) có thể lưu trữ, xử lý được một lượng dữ liệu lớn một cách nhanh chóng và hiệu quả. NoSQL đã ra đời, thay thế hệ quản trị CSDL quan hệ, giải quyết bài toán trên.

**2) Giới thiệu về Nosql**

Với hầu hết các thời kỳ web, Hệ quản trị cơ sở dữ liệu quan hệ dựa trên SQL đã thống trị hầu hết các hệ Quản trị Cơ sở dữ liệu. Tuy nhiên, thời gian gần đây, một cách tiếp cận mới đã bắt đầu biết đến là NoSQL, tạo ra sự thay thế cho các hệ quản trị cơ sở dữ liệu quan hệ truyền thống.

NoSQL còn có nghĩa là Non-Relational - không ràng buộc. Tuy nhiên, thuật ngữ đó ít phổ dụng hơn và ngày nay người ta thường dịch NoSQL thành Not Only SQL- Không chỉ SQL. NoSQL ám chỉ đến những cơ sở dữ liệu không dùng mô hình dữ liệu quan hệ để quản lý dữ liệu trong lĩnh vực phần mềm

Thuật ngữ NoSQL được giới thiệu lần đầu vào năm 1998 sử dụng làm tên gọi chung cho các cơ sở dữ liệu quan hệ nguồn mở nhỏ nhưng không sử dụng SQL cho truy vấn.

Vào năm 2009, Eric Evans, nhân viên của Rackspace giới thiệu lại thuật ngữ NoSQL khi Johan Oskarsson của Last.fm muốn tổ chức một hội thảo về cơ sở dữ liệu nguồn mở phân tán. Thuật ngữ NoSQL đánh dấu bước phát triển của thế hệ CSDL mới: một thế hệ CSDL không ràng buộc, phân tán, nguồn mở, khả năng mở rộng theo chiều ngang, có thể lưu trữ, xử lý từ một lượng rất nhỏ cho tới hàng petabytes dữ liệu trong hệ thống có độ chịu tải, chịu lỗi cao với những đòi hỏi về tài nguyên phần cứng thấp.

Một số đặc điểm nhận dạng cho thế hệ CSDL mới này bao gồm: schema-free, hỗ trợ mở rộng dễ dàng, API đơn giản, nhất quán cuối (eventual consistency), không giới hạn không gian dữ liệu,...

Có nhiều cách phân loại các cơ sở dữ liệu NoSQL khác nhau, mỗi loại với các loại và loại con khác nhau, một số trong số đó có thể chồng chéo lên nhau. Một phân loại cơ bản dựa trên mô hình dữ liệu, với các ví dụ:

Column: Accumulo, Cassandra, Druid, HBase, Vertica

Document: Apache CouchDB, Clusterpoint, Couchbase, DocumentDB, HyperDex, Lotus Notes, MarkLogic, MongoDB, OrientDB, Qizx, RethinkDB

Key-value: Aerospike, CouchDB, Dynamo, FairCom c-treeACE, FoundationDB, HyperDex, MemcacheDB, MUMPS, Oracle NoSQL Database, OrientDB, Redis, Riak, Berkeley DB

Graph: AllegroGraph, InfiniteGraph, MarkLogic, Neo4J, OrientDB, Virtuoso, Stardog

Multi-model: Alchemy Database, ArangoDB, CortexDB, FoundationDB, MarkLogic, OrientDB

Tuy cùng mang những đặc điểm chung của NoSQL nhưng mỗi CSDL NoSQL cũng có những đặc điểm riêng, và vì thế thường được dùng cho những dự án khác nhau.

Ví dụ

MongoDB và Redis là những lựa chọn tốt cho việc lưu trữ các dữ liệu thống kê ít được đọc mà lại được viết thường xuyên.

Hadoop, một CSDL dạng tự do, phân tán làm tốt công việc lưu trữ các dữ liệu lớn như các con số thống kê thời tiết hoặc công việc phân tích nghiệp vụ.

Memcachedb, một CSDL nhất thời chóng tàn, tuyệt vời trong lưu trữ các phiên làm việc web, các khóa, và các con số thống kê ngắn hạn.

Cassandra và Riak (các lưu trữ dư thừa, tự động tạo bó cluster) làm tốt trong các môi trường với các ứng dụng có tính sẵn sàng cao, khi thời gian sống tối đa là sống còn.

Để tìm hiểu sâu hơn về các CSDL hiện đại NoSQL, chúng ta đi nghiên cứu chi tiết CSDL đặc trưng là MongoDB.

\*Một số đặc điểm chung:

High Scalability: Gần như không có một giới hạn cho dữ liệu và người dùng trên hệ thống.

High Availability: Do chấp nhận sự trùng lặp trong lưu trữ nên nếu một node

(commodity machine) nào đó bị chết cũng không ảnh hưởng tới toàn bộ hệ thống.

Atomicity: Độc lập data state trong các operation.

Consistency: chấp nhận tính nhất quán yếu, có thể không thấy ngay được sự thay

đổi mặc dù đã cập nhật dữ liệu.

Durability: dữ liệu có thể tồn tại trong bộ nhớ máy tính nhưng đồng thời cũng

được lưu trữ lại đĩa cứng.

Deployment Flexibility: việc bổ sung thêm/loại bỏ các node, hệ thống sẽ tự động

nhận biết để lưu trữ mà không cần phải can thiệp bằng tay. Hệ thống cũng không

đòi hỏi cấu hình phần cứng mạnh, đồng nhất.

Modeling flexibility: Key-Value pairs, Hierarchical data (dữ liệu cấu trúc),

Graphs.

Query Flexibility: Multi-Gets, Range queries (load một tập giá trị dựa vào một

dãy các khóa).

Phân loại NoSQL database. Có bốn loại chung (loại phổ biến nhất) của cơ sở dữ liệu NoSQL. Mỗi loại đều có các thuộc tính và giới hạn riêng. Không có một giải pháp duy nhất nào tốt hơn tất cả các giải pháp khác, tuy nhiên có một số cơ sở dữ liệu tốt hơn để giải quyết các vấn đề cụ thể.

**3) Giới thiệu về ArangoDB:**

ArangoDB là một hệ quản trị cơ sở dữ liệu NoSQL mã nguồn mở được phát triển bởi ArangoDB GmbH (Gesellschaft mit beschränkter Haftung- Công ty trách nhiệm hữu hạn). Được ra mắt vào năm 2011 dưới cái tên AvocadoDB, sau đó được đổi tên thành ArangoDB như ngày nay. Arango có nghĩa là một loại tiền tệ được làm từ đá carnelian trong thời kì buôn bán nô lệ ở Châu Phi.’

Hiện nay, công ty TNHH ArangoDB đã được phát triển thành quy mô tập đoàn vào năm 2014 và có trụ sở chính tại San Francisco (USA) và Cologne (Germany).

Arango hỗ trợ 3 mô hình lưu trữ key/value, documents, graphs:

**Key-value**:

Mô hình lưu trữ key-value là mô hình đơn giản nhất để scale. Trong ArangoDB, mỗi document trong một collection đều có khóa chính \_key ở attribute đầu tiên. Ở các attribute tiếp theo trong document, việc thiếu đi Index sẽ khiến nó hoạt động như một key-value đơn giản.

Thao tác duy nhất ở trường hợp này là việc truy vấn khóa đơn và các truy vất kết hoặc update. Nếu \_key là attribute sharding duy nhất thì quá trình sharding được thực hiện đối với khóa chính và tất cả các hoạt động này sẽ chia theo tỷ lệ tuyến tính.

Nếu việc sharding được thực hiện bằng cách sử dụng các khóa phân đoạn khác nhau, thì việc tra cứu một khóa đơn cần phải yêu cầu tất cả các phân đoạn và do đó không chia tỷ lệ tuyến tính.

**Document**:.

Đối với mô hình lưu trữ dữ liệu Document, kể cả khi có các Index phụ thì những đối số cũng được áp dụng giống nhau. Vì Index cho một sharded collectio vẫn scale tuyến tính giống như các local index trên mỗi shard. Do đó, mỗi document vẫn scale tuyến tính theo kích cỡ của cluster, trừ khi người dùng thiết lập sharding kiểu khác làm cho việc tra cứu tốn nhiều tài nguyên hơn

**Graph:**

Cơ sở dữ liệu dạng graph đặc biết tốt trong trường hợp truy vấn đồ thị liên thông đến các đường dẫn có độ dài chưa biết trước, ví dụ như tìm đường đi ngắn nhất giữa 2 phương tiện bằng đồ thị, hoặc tìm đường đi trùng khớp với mẫu nhất định được quy định trước

**4) Lý thuyết về NoSQL trong ArangoDB**

ArangoDB cung cấp cho người dùng 3 cách để tương tác với CSDL:

* AQL (Arango query Language)-ở Web UI hoặc Terminal
* http/API-ở Web UI hoặc Terminal
* Shell script- ở Terminal

So sánh thuật ngữ trong Cơ sở dữ liệu quan hệ và ArangoDB:

|  |  |
| --- | --- |
| Cơ sở dữ liệu quan hệ | ArangoDB |
| Table | Collection |
| Row | Document |
| Column | Attribute |
| primary key | primary key (automatically present on \_key attribute |
| Index | Index |

Cấu trúc Document và Collection:

Để tìm hiểu sâu cơ chế phân tán trong hệ quản trị ArangoDB đầu tiên ta tìm hiểu về các khái niệm của hệ quản trị này:

Document là một đơn vị dòng trong hệ quản trị ArangoDB, trong hệ quản trị CSDL quan hệ còn được gọi là Rows. Ví dụ

**SINHVIEN**

|  |  |  |
| --- | --- | --- |
| **MaSV** | **HoTen** | **MaLop** |
| 19521783 | Lê Văn Long | IS211.M12.HTCL |
| 19520854 | Phạm Hoàng Phúc | IS211.M12.HTCL |

Trong CSDL quan hệ thì (19521783, ‘Lê Văn Long’, ‘IS211.M12.HTCL’) là một dòng dữ liệu, còn trong ArangoDB gọi đây là một Document.

Collection là một đơn vị chứa tất cả các Document thuộc Collection đó, tham chiếu sang hệ quản trị CSDL tập trung thì Collection là một table. Trong trường hợp trên Collection là SINHVIEN.

Attribute là đơn vị thay cho column trong RDBMS ví dụ MaSV là một Attribute.

Mỗi Document có một khóa đặc biệt “\_id” (được ArangoDB tạo sẵn) và nó là duy nhất trong Collection. Chúng ta có thể cung cấp cho “\_id” cho ArangoDB nếu không cung cấp nó sẽ trả về mặc định ArangoDB tự động tạo.

Collection không bắt buộc các Document phải cùng một cấu trúc hay còn gọi là Schema-free.

Ví dụ: {“HoTen”: “Phạm Hoàng Phúc”} và {“loaikh”: “VIP”} là hai Document khác cấu trúc nhưng có thể cùng một Collection.

**6) So sánh ArangoDB với MongoDB**

Đa mô hình: MongoDB chỉ hỗ trợ mô hình document còn Arango hỗ trợ đa mô hình. Điều này giúp người dùng truy xuất dữ liệu được lưu trữ ở nhiều dạng mô hình khác nhau.

Joins: Ở MongoDB, người dùng không thể kết và mở rộng các collection. Tuy nhiên, ArangoDB có hỗ trợ các tính năng này.

Ngôn ngữ truy vấn khai báo: MongoDB sử dụng cú pháp JSON để truy vấn, nó không hỗ trợ ngôn ngữ truy vấn khai báo. Mặc khác, ArangoDB có phát triển một ngôn ngữ truy vấn riêng gần giống với SQL, cho phép người dùng truy xuất dữ liệu phức tạp trong 1 câu query.

Giao tác phức tạp: ArangoDB sử dụng các giao tác phức tạp để nối giữa các document và giữa các collection. Đảm bảo tính cô lập ở các cluster có sẵn. Tính năng này không có ở MongoDB

Ngoài ra, ArangoDB còn cho phép người dùng sử dụng một data-centric microservices trên nền tản Foxx (một javascript framework).

## Ngôn ngữ thao tác dữ liệu:

ArangoDB là một cơ sở dữ liệu cung cấp documents cho người dùng. Các document được vận chuyển dưới dạng JSON bằng kết nối TCP, theo phương thức HTTP. Các API được cung cấp để tương tác với cơ sở dữ liệu.

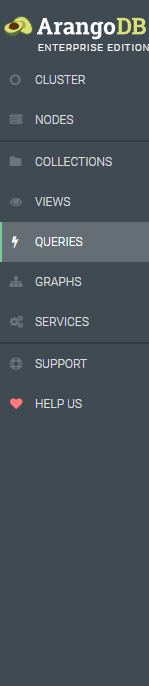
Trang Web của ArangoDB được gọi là Aadvark, cung cấp giao diện cho người dùng dễ sử dụng. Shell tương tác được gọi là arangosh, là một hệ thống đa môi trường và 1 ngôn ngữ lập trình. Tất các công cụ trên sử dụng giao diện HTTP của server và loại bỏ việc lập trình sử dụng ngôn ngữ bậc thấp trong đa số trường hợp

Arango cung cấp các cách sau để tương tác với dữ liệu

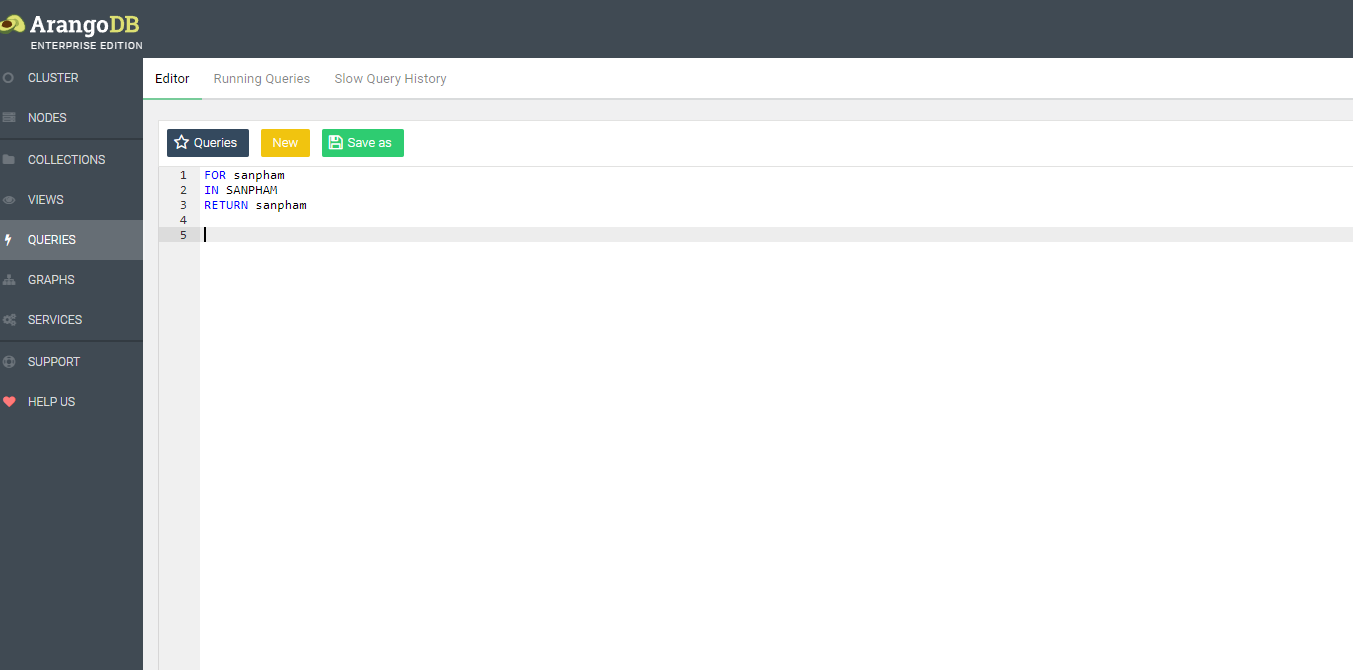
* **AQL (Arango query Language)-ở Web UI hoặc Terminal**

Cách chạy query trên WEB UI:

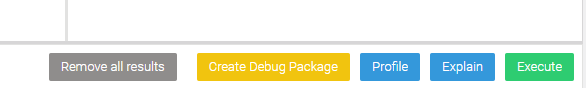
1. Chọn tab query:



1. Thêm câu truy vấn vào typebox

****

Ở góc dưới bên phải sẽ có các button sau**:**

****

Remove all results: Clear kết quả query.

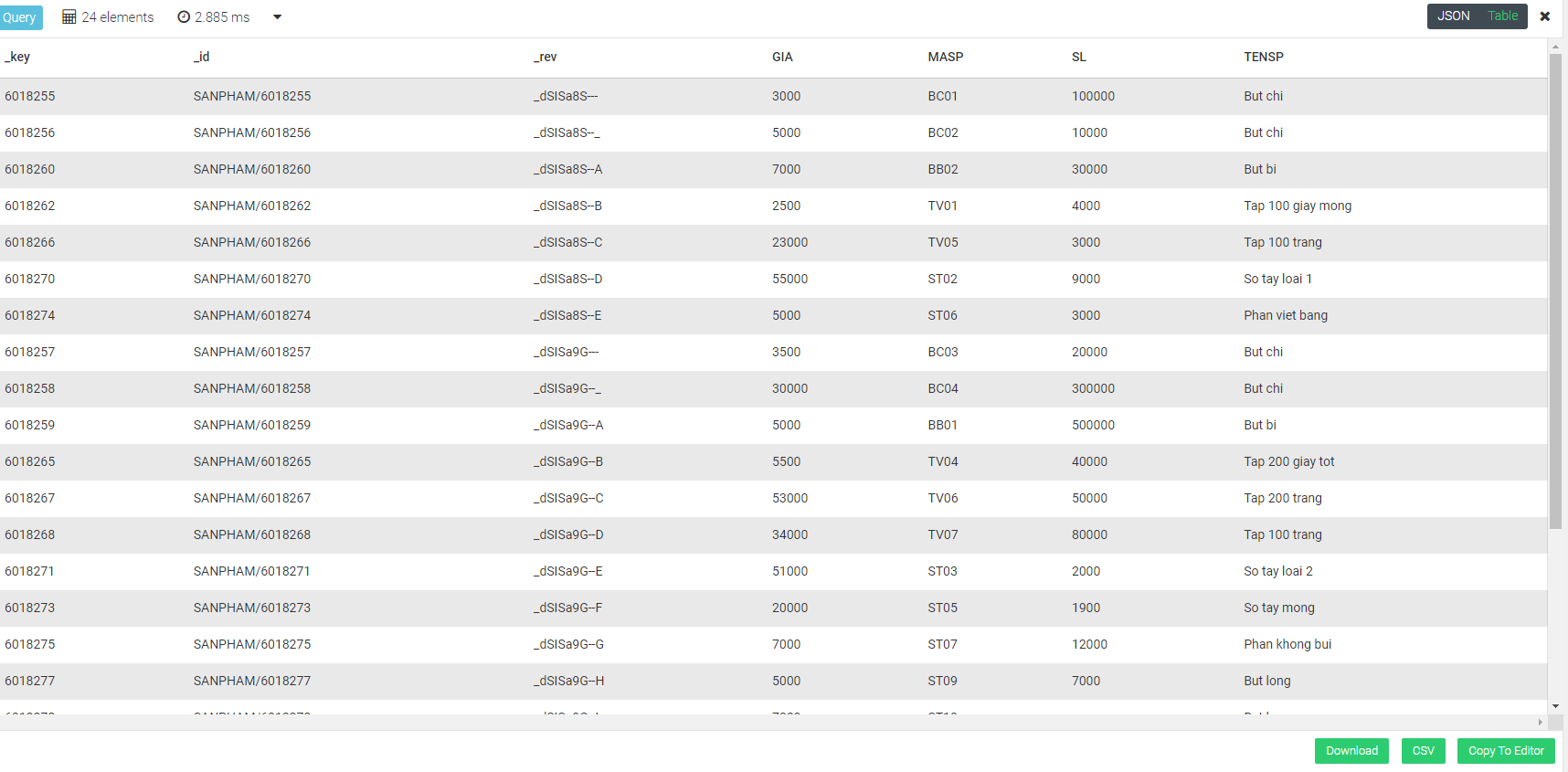
Create Debug Package: Tải query debug package.

Profile: Đo đạc performance của query.

Explain: Giải thích cách query thực hiện.

Execute: chạy câu query.

1. Kết quả sau khi bấm execute:

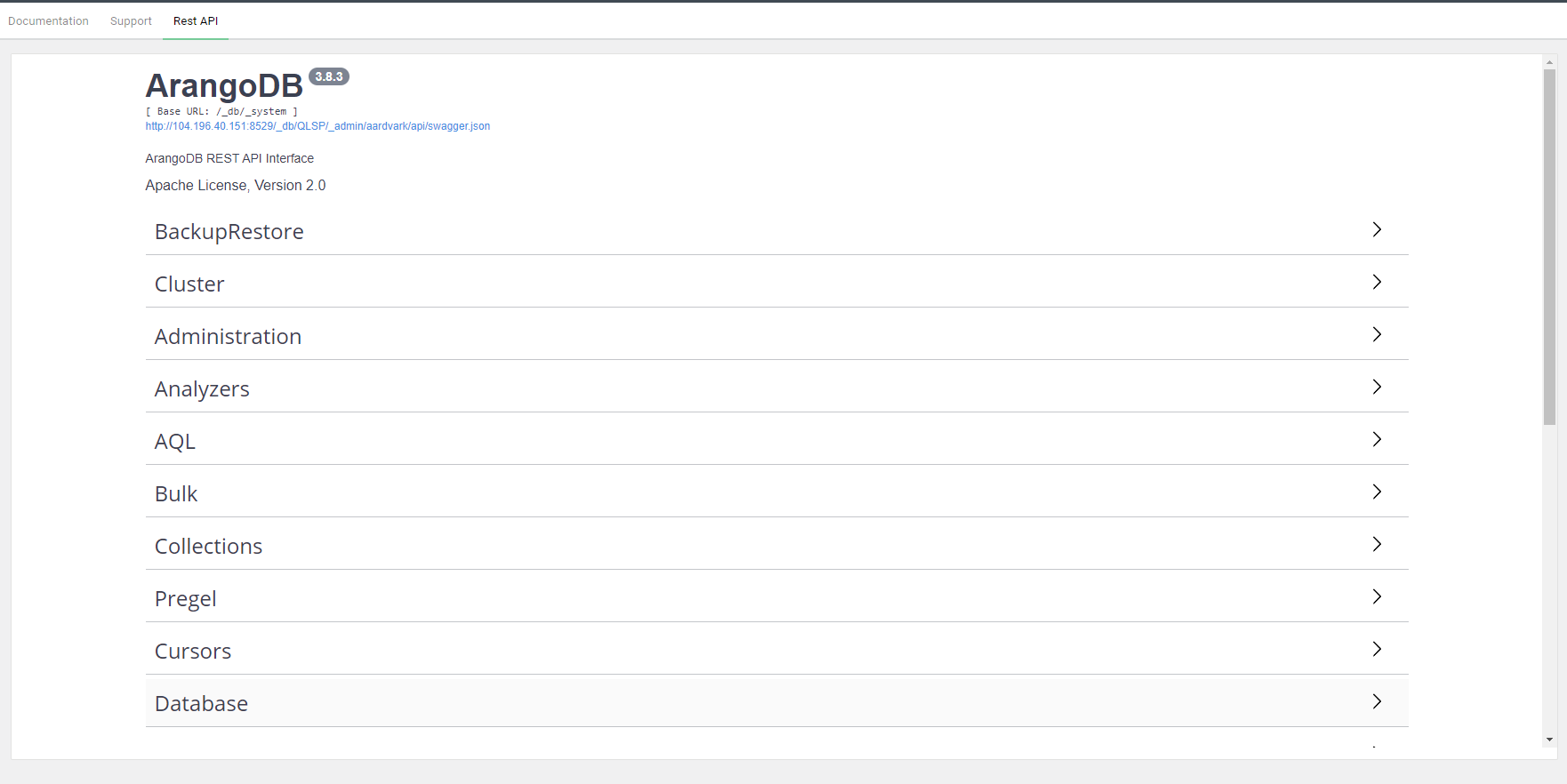
****

1. So sánh câu truy vấn với SQL

|  |  |  |
| --- | --- | --- |
|  | **SQL** | **AQL** |
| **Thêm 1**  **row/document:** | INSERT INTO users (name, gender)  VALUES ("John Doe", "m") | INSERT { name: "John Doe", gender: "m" }  INTO users |
| **Thêm nhiều row/document:** | INSERT INTO users (name, gender)  VALUES ("John Doe", "m"),  ("Jane Smith", "f"); | FOR user IN [  { name: "John Doe", gender: "m" },  { name: "Jane Smith", gender: "f" }  ]  INSERT user INTO users |
| **Update 1**  **row/Document** | UPDATE users  SET name = "John Smith"  WHERE id = 1; | UPDATE { \_key: "1" }  WITH { name: "John Smith" }  IN users |
| **Thêm 1**  **column/attribute với kiểu dữ liệu mặc định** | ALTER TABLE users ADD COLUMN numberOfLogins  INTEGER NOT NULL default | FOR user IN users  UPDATE user  WITH { numberOfLogins: 0 } IN users |
| **Xóa 1**  **row/Document** | DELETE FROM users  WHERE id = 1 | REMOVE { \_key:"1" } IN users |
| **Xóa nhiều**  **row/Document** | DELETE FROM users  WHERE active = 1; | FOR user IN users  FILTER user.active == 1  REMOVE user IN users |
| **Truy xuất dữ liệu từ 1**  **bảng/collection** | SELECT \* FROM users | FOR user  IN users  RETURN user |
| **Sắp xếp các row/document trong 1**  **table/collection** | SELECT \* FROM users  WHERE active = 1  ORDER BY name, gender; | FOR user IN users  FILTER user.active == 1  SORT user.name, user.gender  RETURN user |

* **http/API-ở Web UI hoặc Terminal**

Arango cung cấp cho người dùng Swagger Document API, chỉ việc gọi API tương tứng



Ví dụ: Returns all registered AQL user functions

|  |
| --- |
| curl -X GET "http://104.196.40.151:8529/\_db/\_system/\_api/aqlfunction" -H "accept: application/json" -H "Authorization: bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJwcmVmZXJyZWRfdXNlcm5hbWUiOiJyb290IiwiaXNzIjoiYXJhbmdvZGIiLCJpYXQiOjE2Mzc0OTg3MDYsImV4cCI6MTY0MDA5MDcwNn0.1hto8i4UFAIK8bX-lLm7uvDv\_q8iVrlwAHcSC5HDCtc" |

* **Shell script- ở Terminal**

Tạo database:

|  |
| --- |
| db.\_createDatabase(database-name) |

Tạo collection:

|  |
| --- |
| db.\_create(collection-name, properties) |

Xóa collection”

|  |
| --- |
| collection.drop() |

## 5) Cơ chế phân tán của ArangoDB:

### a) Giới thiệu chung

ArangoDB tổ chức các collection của nó trên các shards (là một mẫu phân vùng dữ liệu để đặt vào các server riêng biệt, có thể trải rộng khắp thế giới). Sharding cho phép nhiều máy tính chạy các cluster của ArangoDB để cùng nhau tạo thành một cơ sở dữ liệu thống nhất.

Sharding được sử dụng để phân tán dữ liệu qua các máy tính trong 1 ArangoDB Cluster. Nó cũng là cách để xác định vị trí tối ưu của các document trên từng server riêng lẻ

Sharding cho phép người dùng lưu trữ được nhiều dữ liệu hơn vì ArangoDB phân tán dữ liệu một cách tự động qua nhiều server khác nhau. Trong nhiều trường hợp, người dùng có thể nhận được nhiều lợi ích từ thông lượng dữ liệu vì việc load dữ liệu cũng có thể được phân tán qua các server khác.

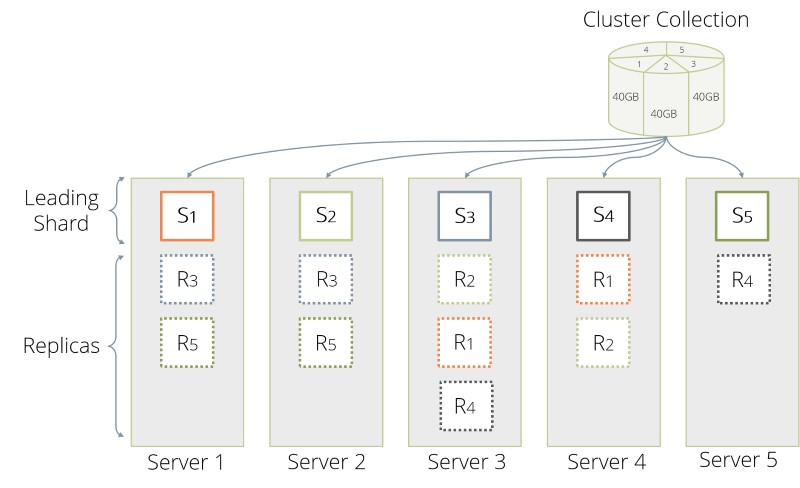
Sharding cho phép Arango hỗ trợ người dùng triển khai một khối lượng dữ liệu rất lớn và hầu như không thể chứa đủ trong một máy đơn lẻ. Việc đọc ghi của AQL cũng có thể làm tràn RAM hoặc đầy bộ nhớ.

.Có 2 cách chính để phân mảnh dữ liệu trong một hệ quản trị:

* Phân mảnh dọc.
* Phân mảnh ngang.

Phân mảnh dọc đòi hỏi máy tính phải được nâng cấp tối đa phần cứng. Mặc dù tốt kém nhưng đây vẫn là một cách phân mảnh hiểu quả vì việc quản lý có thể diễn ra một cách dễ dàng và hiệu suất không thay đổi nhiều. Suy luận về cách hoạt động của hệ thống máy đơn cũng dễ dàng hơn nhiều so với hệ thống đa máy. Tuy nhiên, tại một thời điểm nào đó, phần cứng có thể trở nên quá tải và tốn kém thêm chi phí nâng cấp.

Phân mảnh ngang chỉ đơn giản là tăng số lượng server. Server thường được host trên các máy tính có chất lượng phần cứng vừa phải và hoàn toàn có thể được thay thế bởi các dịch vụ cloud hiện nay. Mỗi máy có thể không chứa nhiều dữ liệu nhưng một hệ thống nhiều máy sẽ chứa được một khối lượng lớn dữ liệu. Việc triển khai thêm máy sẽ ít tốn kém hơn việc nâng cấp một máy đơn lẻ. Sự phức tạp ngày càng tăng trong cơ sở hạ tầng có thể được quản lý bằng cách sử dụng các công cụ điều phối theo cụm và container hiện đại như Kubernetes.



Để thực hiện những điều trên, ArangoDB phân mảnh các set dữ liệu của người dùng vào các shards. Số lượng các Shards được thiết lập dựa trên nhu cầu của người dùng. Việc Sharding phù hợp là cần thiết để tối ưu hóa hiệu suất hệ thống. Từ bên ngoài, quy trình tách nhập dữ liệu diễn tra một cách hoàn toàn trong suốt, giúp mô hình phân tán của ArangoDB đạt được mục tiêu “Master-Master Replication”.

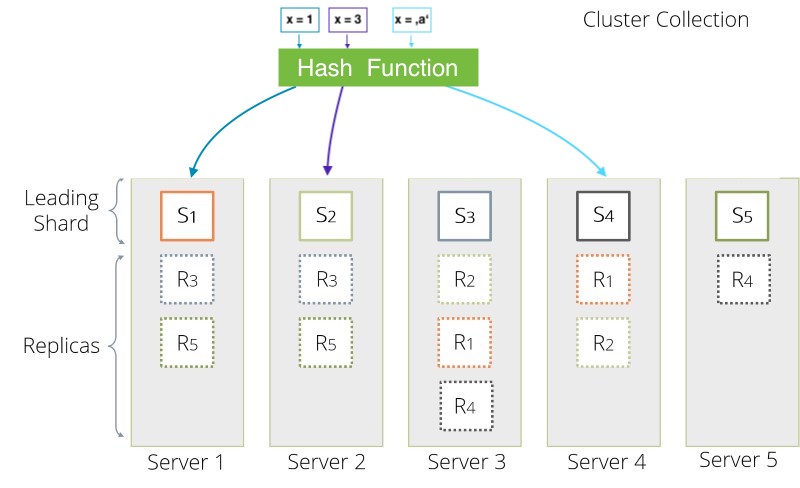
Một ứng dụng có thể tương tác với bất kì Coodinator nào và nó sẽ tự động tìm tới nơi mà dữ liệu đang được lưu trữ (trường hợp đọc) hoặc nơi nó sẽ được lưu trữ (trường hợp ghi). Thông tin của các Shards được chia sẽ giữa các Coodinator bằng Agency.

Shards được cấu hình trên mỗi collection, do đó nhiều shards tập hợp các collection thành một thể đồng nhất. Để xác định chứa data trên shard nào, ArangoDB thực hiện băm trên các giá trị. Theo mặc định, hàm băm này được khởi tạo từ thuộc tính \_key.

Không phải trường hợp nào cũng yêu cần phân mảnh ngang. Trong những trường hợp như vậy, hãy xem xét tính năng OneShard thay thế cho tính năng sharding linh hoạt.

### b) Shard Keys

ArangoDB sử dụng các thuộc tính shard key được chỉ định để xác định shard nào mà một tài liệu nhất định sẽ được lưu trữ. Việc chọn đúng shard key có thể có tác động đáng kể đến hiệu suất của hệ thống có thể làm giảm lưu lượng mạng và tăng hiệu suất .



ArangoDB sử dụng phép băm nhất quán để tính shard mục tiêu từ các giá trị đã cho (như được chỉ định bởi thuộc tính bộ sưu tập shardKeys). Bộ shard key lý tưởng cho phép ArangoDB phân phối tài liệu đồng đều trên các shard và DB server của người dùng. Theo mặc định, ArangoDB sử dụng trường \_key làm khóa phân đoạn. Đối với shard key tùy chỉnh, người dùng nên xem xét một số thuộc tính khác nhau như sau :

Theo bản chất: bẩn chất của một tập hợp là số lượng các giá trị khác nhau cảu nó. Một shard key với 10 giá trị khác nhau không thể được băm thành các mảnh có số lượng lớn hơn N. Người dùng nên sử dụng nhiều shard trong trường hợp shard key có ít dữ liệu khách nhau.

Theo tần suất: Xem xét tần suất một giá trị shard key nhất định xuất hiện trong dữ liệu của người dùng.Hệ thống có nhiều Document cùng giá trị shard key có thể dẫn tới phân tán không đều.

Theo các ý trên, một shard có thể trở thành nút cổ chai trong mỗi cluster. Việc phân mảnh ngang sẽ ngăn chặn tình huống nhiều document tập hợp trong 1 shard. Tại thời điểm này, các shard không thể được chia nhỏ, do đó người triển khai cần phải đặc biệt để tâm tới kích cỡ các shard.

Dựa trên tần suất và bản chất để chọn shard key phù hợp, trong nhiều tình huống, ta có thể chọn nhiều shard keys cùng lúc.

### c) Configuring Shards

Số lượn shards có thể được cấu hình lúc tạo collection (trong web UI hoặc trong arangosh:

|  |
| --- |
| b.\_create("sharded\_collection", {"numberOfShards": 4, "shardKeys": ["country"]}); |

Trong ví dụ trên, country được sử dụng làm shardkey để lưu trữ dự liệu của một quốc gia trong một shard, giúp người dùng tra cứu dữ liệu dễ dàng trong một hệ thống có nền tảng đa quốc gia.

Ta cũng có thể chọn nhiều shard keys cùng lúc.

Tuy nhiên, nếu ta đổi shard key thành thuộc tính khác mặc định (\_key) thì việc tìm các document trong một collection qua shard key sẽ truy xuất đến từng shard một. Tuy nhiên việc này có thể được giảm tải: tất của CRUD APIs và AQLs đều hỗ trợ sử dụng giá trị shard key để gợi ý tra cứu. Ta chỉ cần gõ một phần của câu query update, replace hoặc hoặc remove với các giá trị shard key trong đó.

UPDATE { \_key: "123", country: "…" } WITH { … } IN sharded\_collection

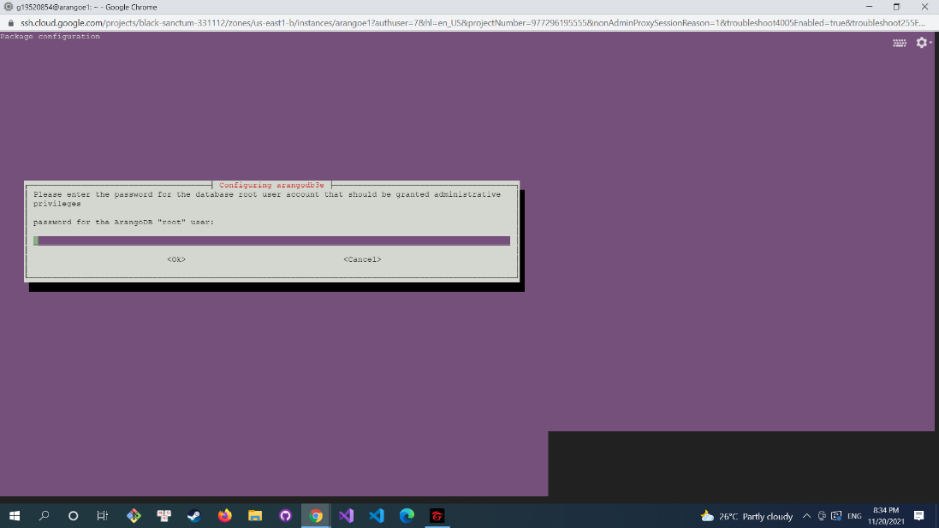
Trong trường hợp các shard key tùy chỉnh được sử dụng, ta không thể chỉ định giá trị khóa chính của document mới nữa mà phải sử dụng giá trị được tạo tự động. Hạn chế thứ hai này xuất phát từ thực tế là việc đảm bảo tính duy nhất của khóa chính sẽ rất kém hiệu quả nếu người dùng có thể chỉ định khóa chính.

## PHẦN 2: MÔ PHỎNG CÀI ĐẶT VÀ PHÂN TÁN TRONG ARANGODB

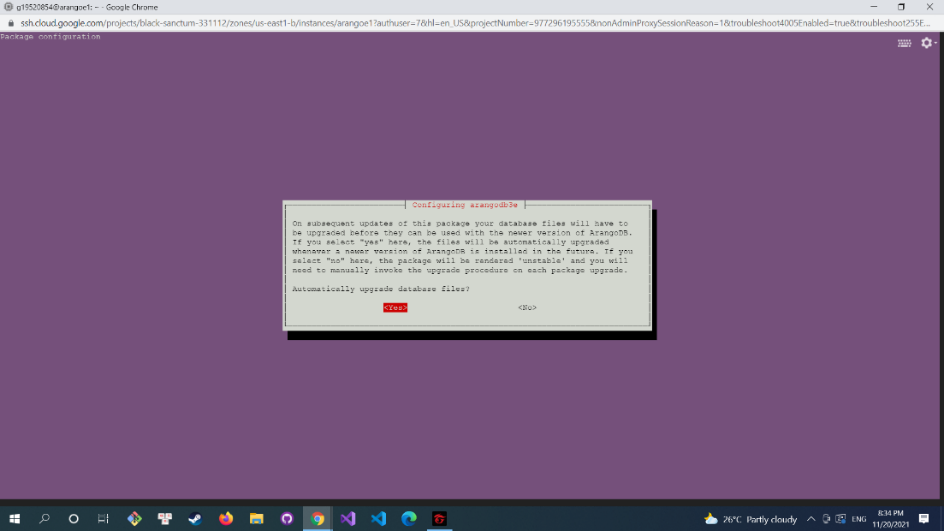
## 1. HƯỚNG DẪN CÀI ĐẶT:

### a) Cài đặt trên 1 máy





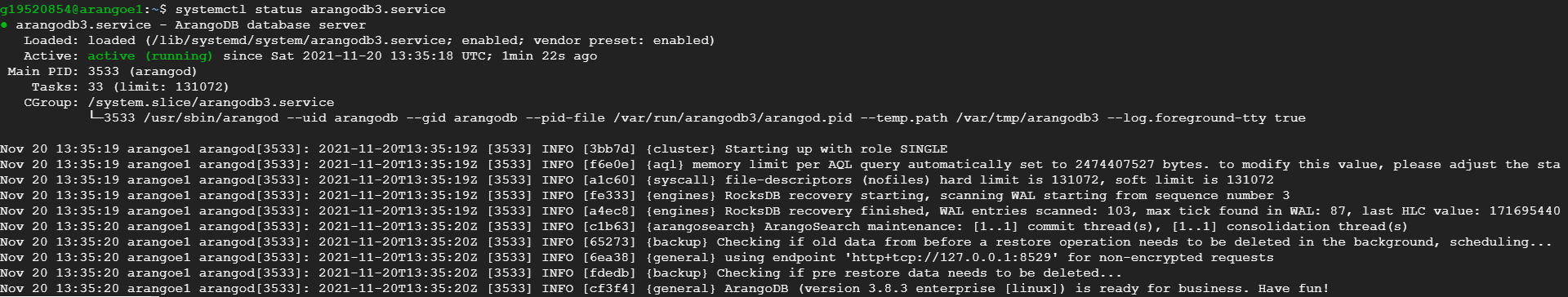
Set pass word cho root user



Lựa chọn upgrade database file khi upgrade ArangoDB hay không.



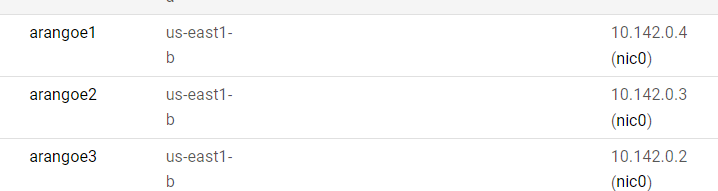
Lựa chọn backup database file trước khi upgrade ArangoDB hay không.



Kiểm tra status xem cài đặt ArangoDB thành công trên máy hiện tại hay chưa

### b. Cài đặt trên cụm máy phân tán:

Để cài đặt cụm máy phân tán trên ArangoDB, ta thiết lập cluster, một cluster gồm các node(Máy tính trong cụm phân tán). ArangoDB quy định, để start được một cluster thì cần tối thiểu 3 máy. Nếu có quá nhiều máy thì ArangoDB khuyến nghị nên phân tán các máy trên nhiều cluster thay vì nhiều máy trên 1 cluster.



Ảnh trên: nhóm thiết lập 3 máy trên Google Cloud Platform (GCP) và các IP trên thuộc cùng NAT network nên mới có thể connect với nhau.

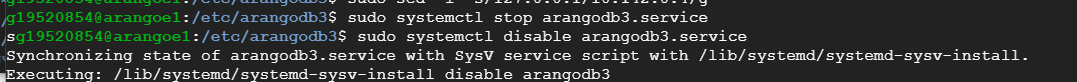
3 máy trên đã cài đặt xong ArangoDB

Giả sử ta đang đứng ở máy 1, ta sẽ thực hiện các lệnh sau

cd /etc/arangodb3 --- Di chuyển vào folder của ArangoDB

sudo sed -i 's/127.0.0.1/10.142.0.4/g' \* ----Do các máy sẽ liên kết với nhau thông qua ip nên ta sẽ đổi 127.0.0.1 (localhost) thành IP của máy hiện tại (máy 1)

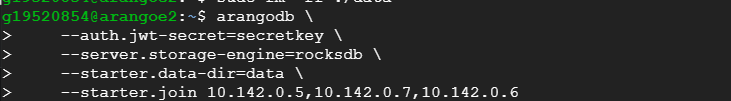
thực hiện tương tự cho 2 máy còn lại



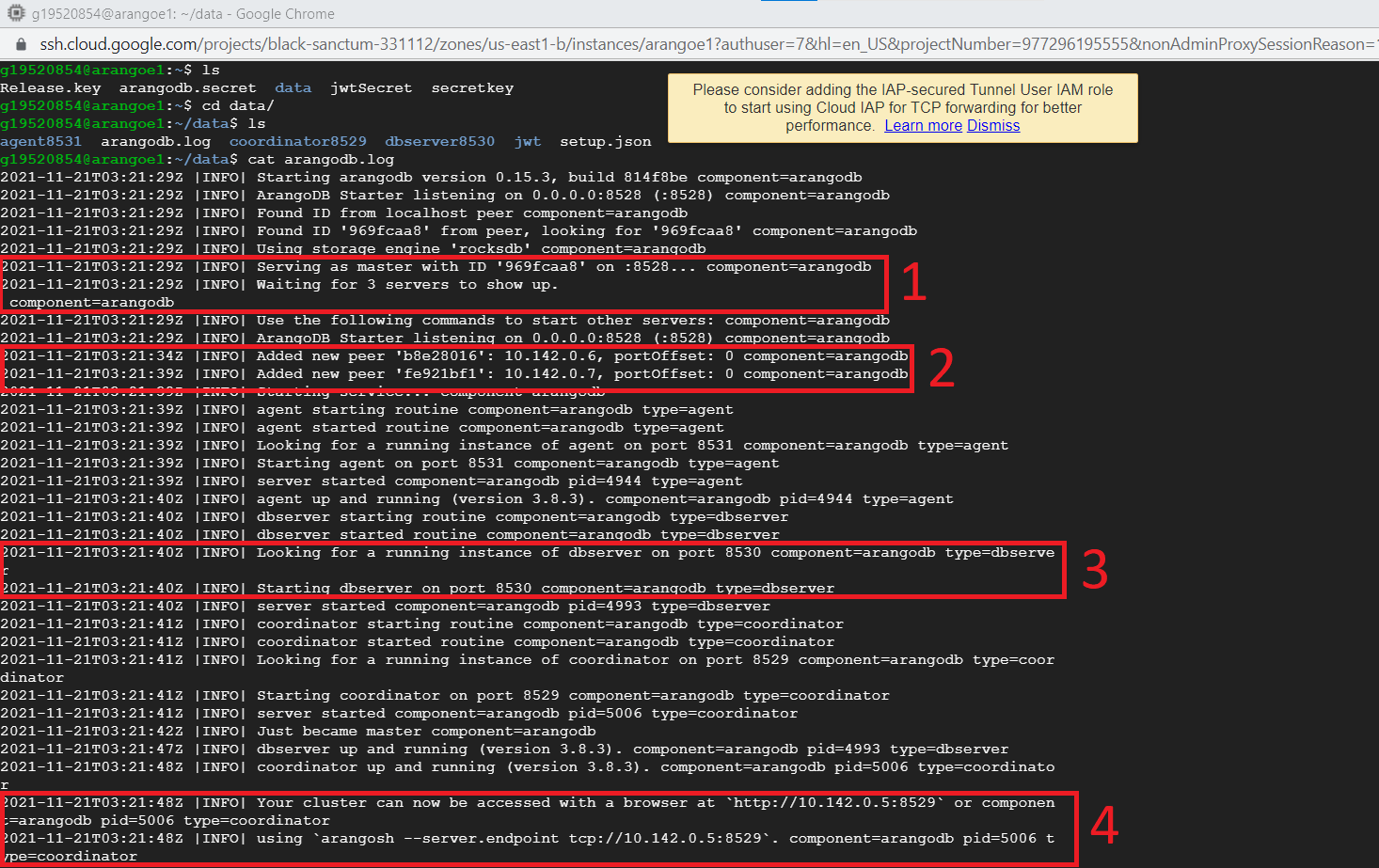
Stop và disable services của ArangoDB ở cả 3 máy vì ta sẽ chạy script start service của cluster.

Do cụm phân tán có bảo mật nên cần phải thêm authentication, theo document của ArangoDB thì cần phải tạo secret key. Secret key phải giống nhau ở các máy trong cụm phân tán.

Không có mô tả.



Script để start cluster, chạy script này ở tất cả các máy trong cluster



1: Ở đây ta đang chạy script ở máy 1 đầu tiên nên máy 1 sẽ được serving as master ở port 8528 và đợi 2 server còn lại khởi động thì sẽ set up cluster thành công.

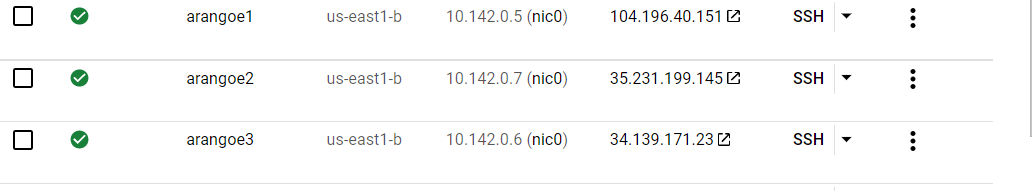
2: Khi ta chạy script trên ở 2 máy còn lại thì nó sẽ lần lượt được thêm vào cluster theo ip máy.

3: Tạo DB Server chạy ở port 8530, đây sẽ là nơi lưu trữ data, shards.

4: Thông báo cluster đã khởi động thành công và có thể truy cập qua ip như hình. Tuy nhiên đây chỉ là những IP trong NAT Network, muốn truy cập từ bên ngoài, ta phải truy cập qua public IP.

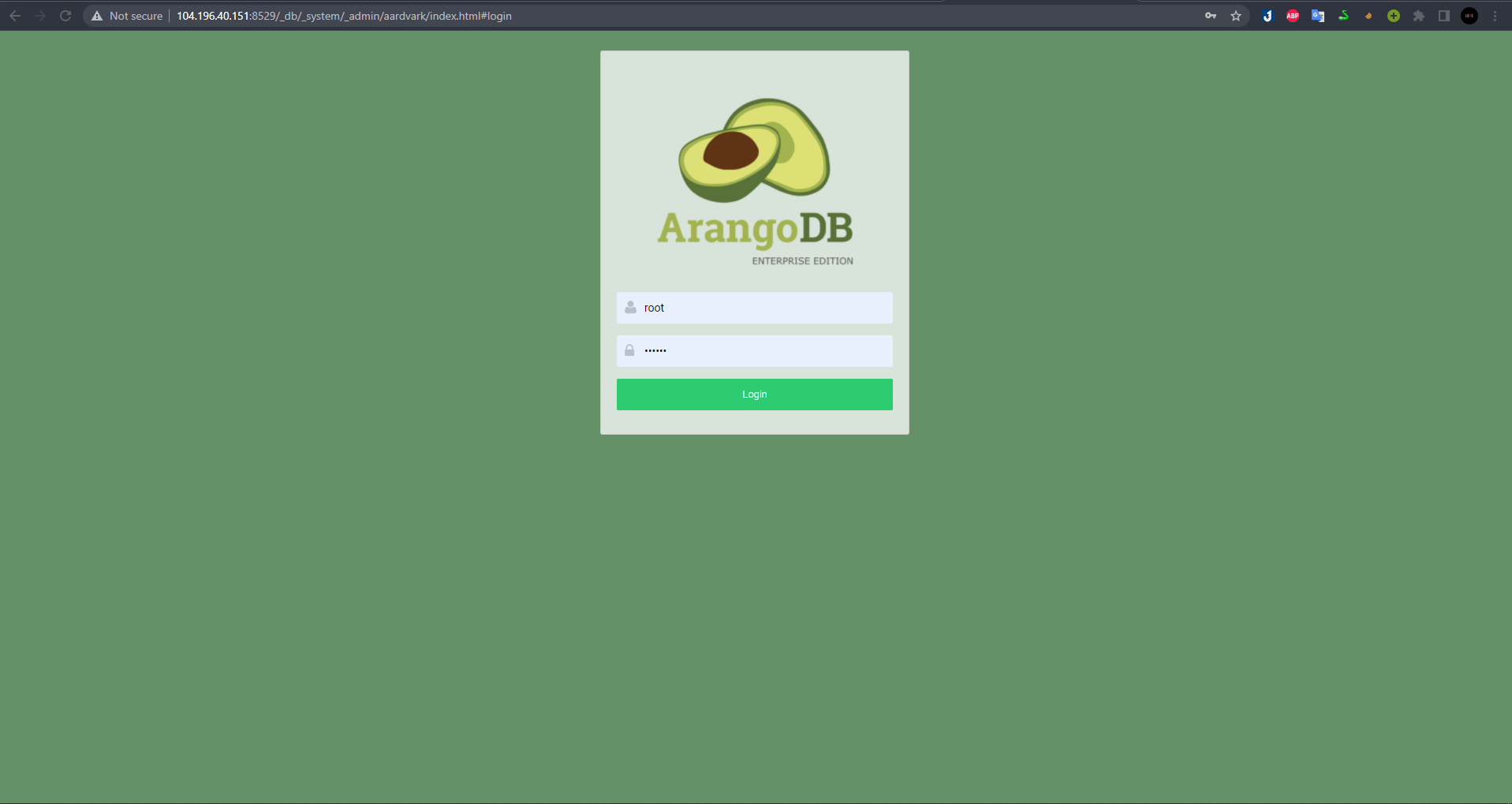
Danh sách public IP:

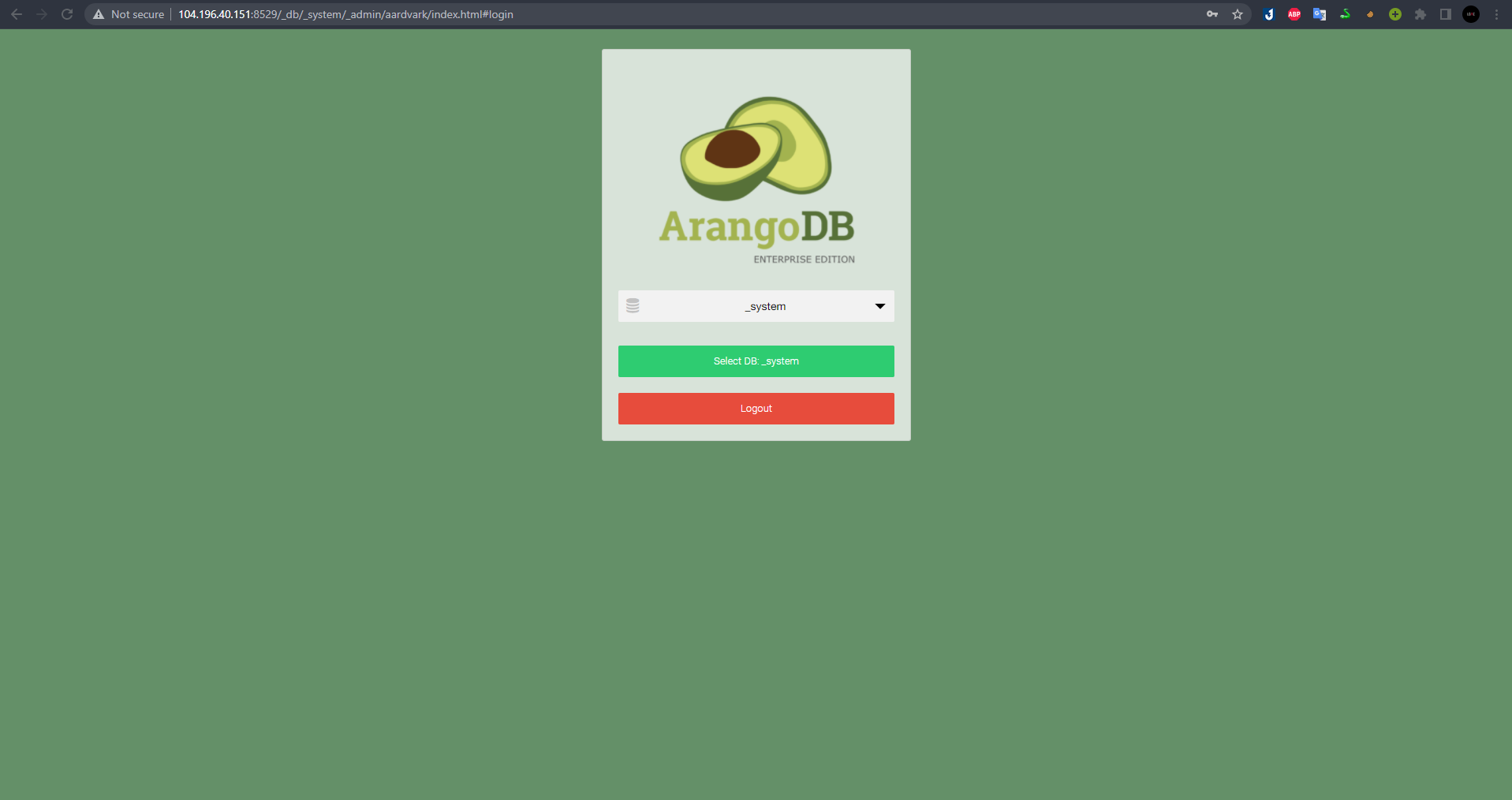
Ví dụ ở đây, máy 1 như trên hình có NAT IP là 10.142.0.5 sẽ được truy cập qua public IP <http://104.196.40.1551:8529> (Có thể config thành https nếu ta config SSL)



Sau truy cập link trên thì ta sẽ thấy giao diện sau:

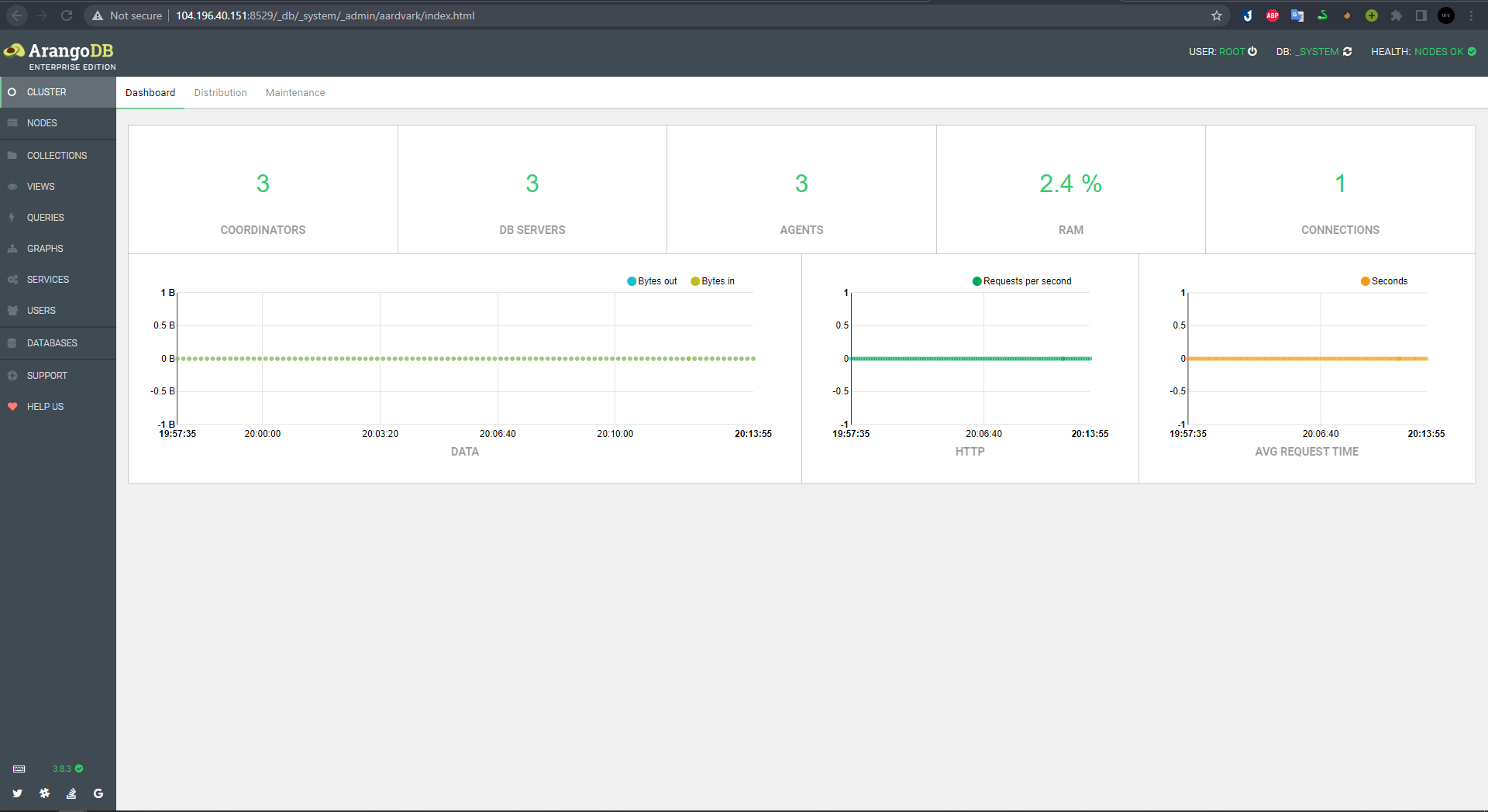
Ở đây ban đầu khi truy cập, ta đăng nhập bằng user root và password để trống. Đăng nhập xong ta có thể chọn tab user để sửa password cho root.



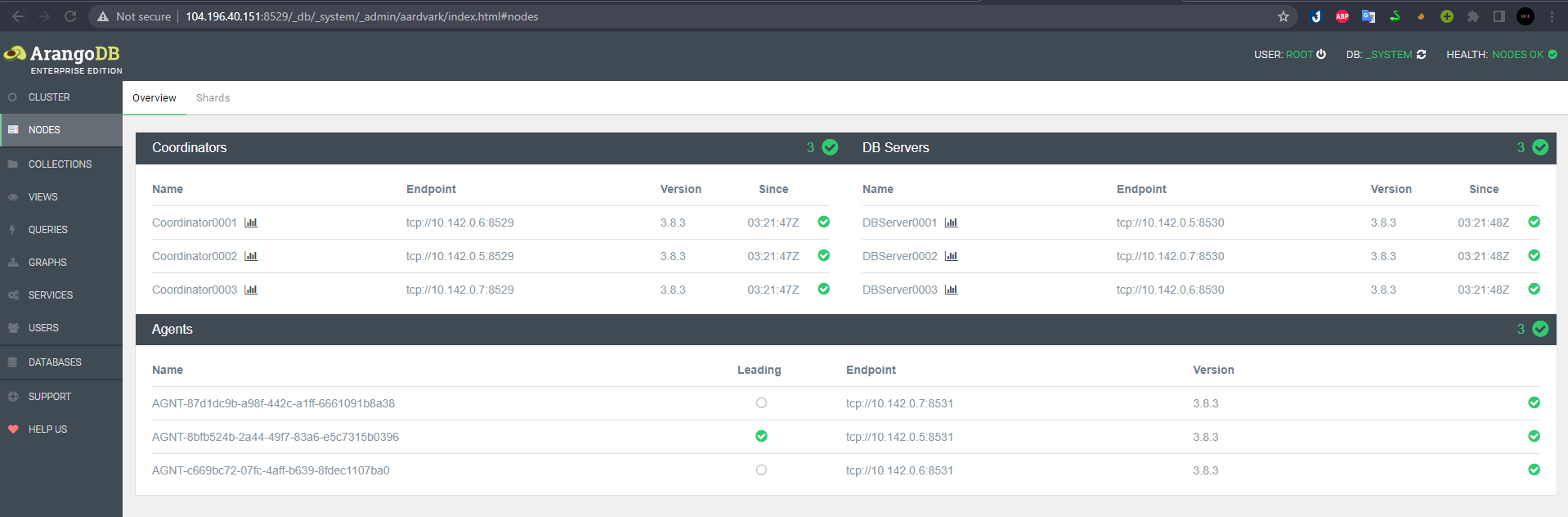


Chọn database để truy cập, ở đây user root sẽ có quyền truy cập vào tất cả database. Các user còn lại chỉ được truy cập vào các database được phân quyền truy cập.

Sau khi select database thì ta sẽ có giao diện sau:



Vào node để kiểm tra tình trạng database trong cluster:



Ở đây ta thấy cả 3 máy đều đã được khởi động.

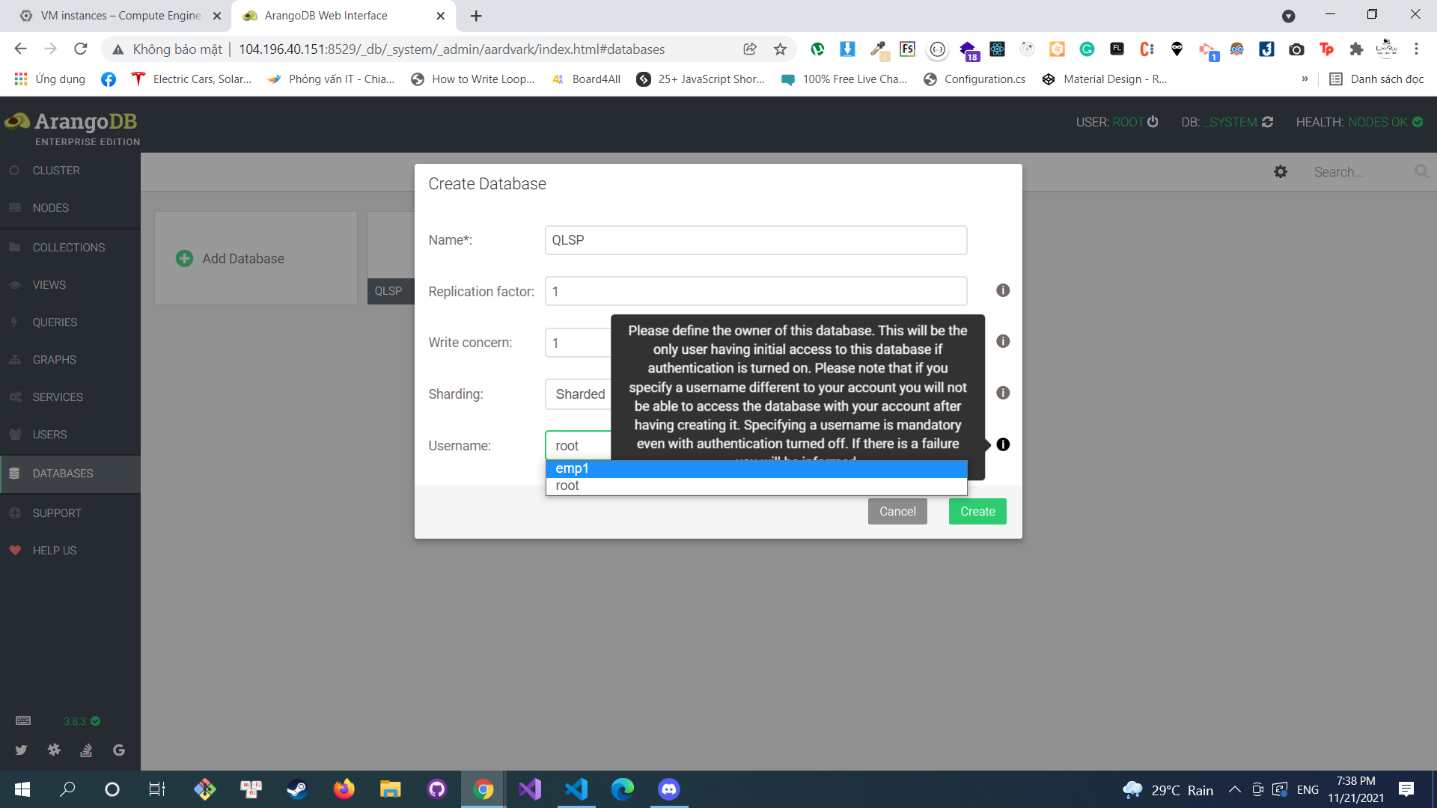
## 2. Thực nghiệm mô phỏng phân tán:

### a) Tạo database và thêm dữ liệu:

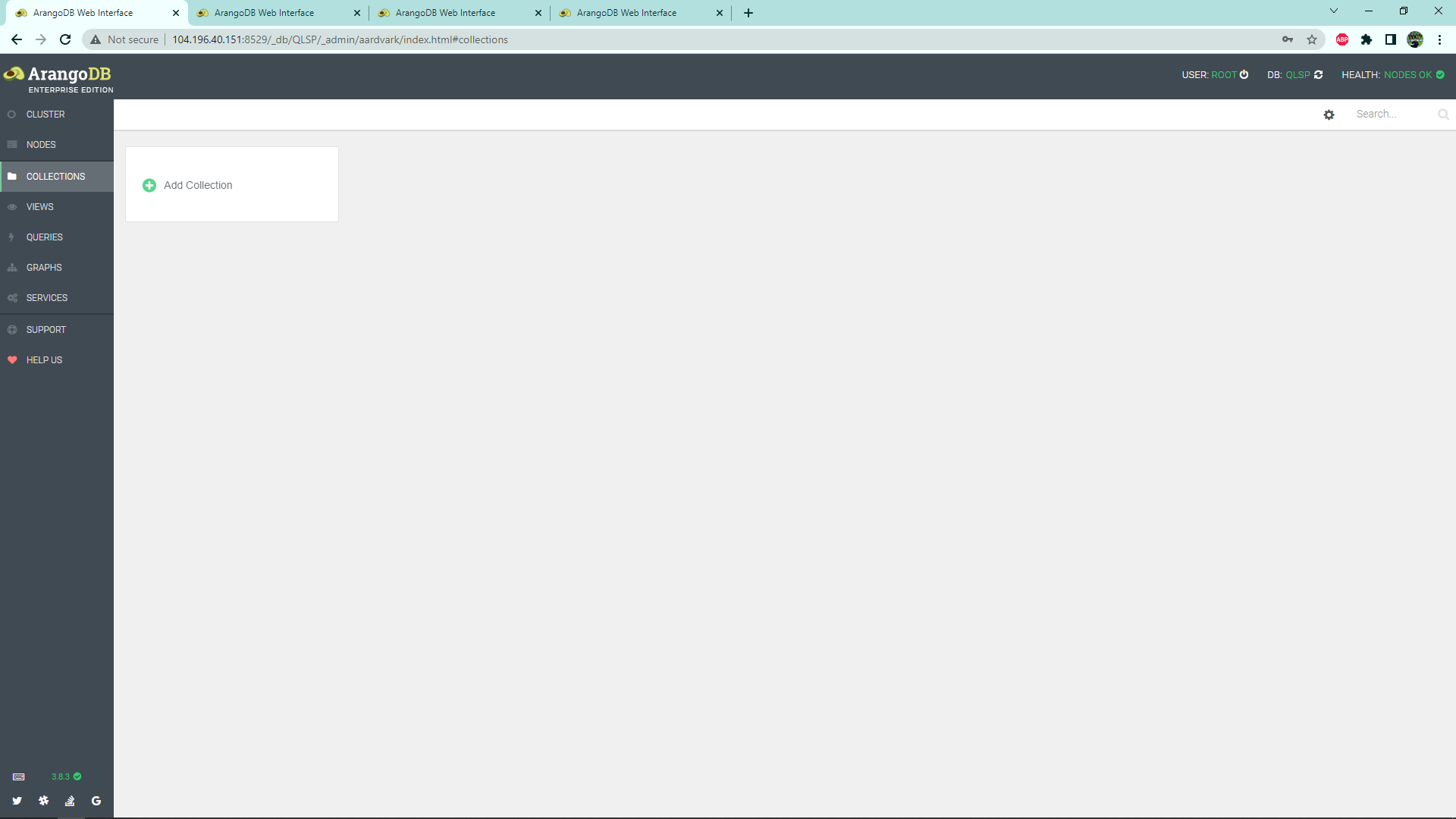
Bài toán quản lý văn phòng phẩm với các collection SANPHAM, NHANVIEN, HOADON, CTHD.

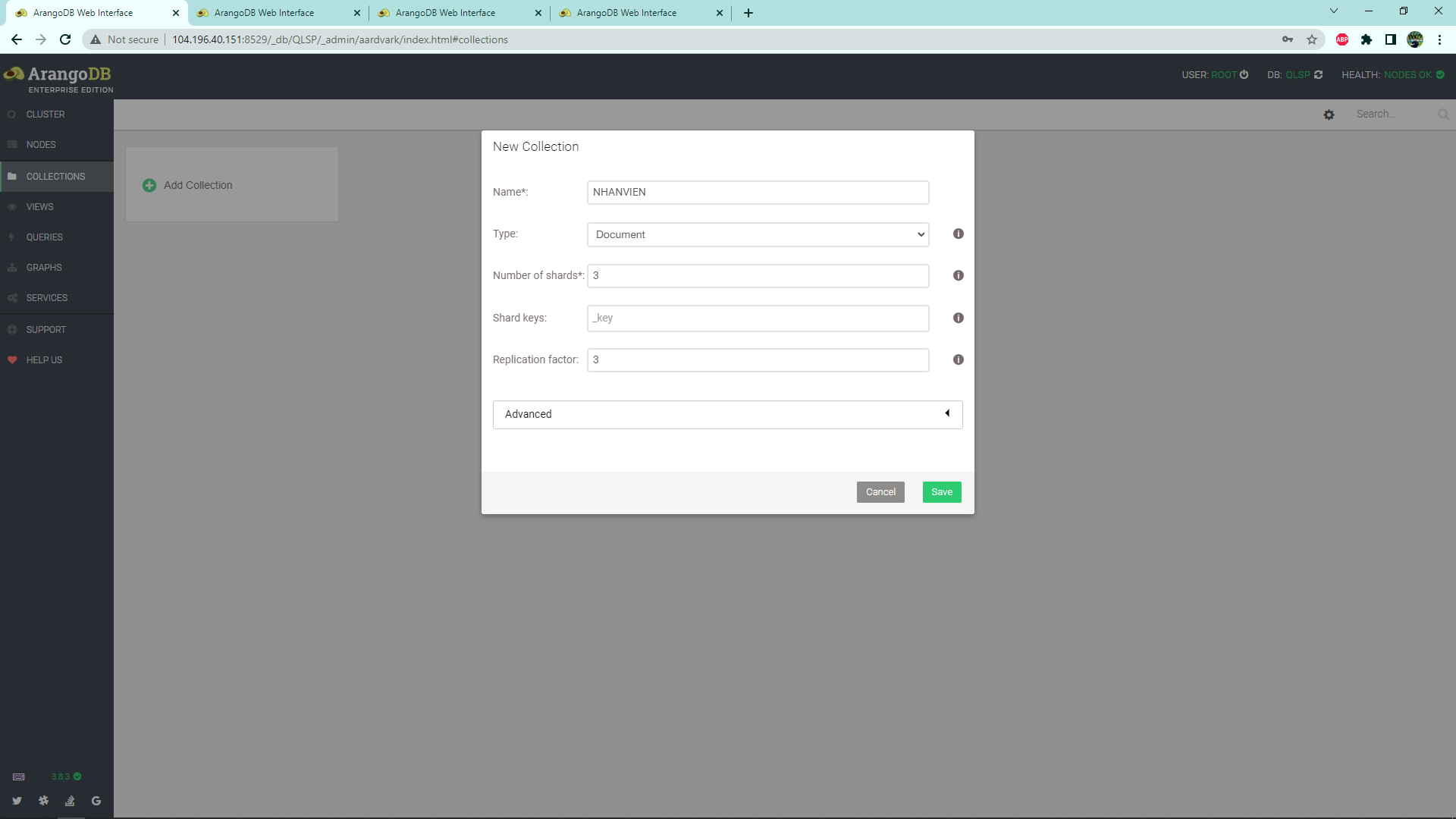
|  |  |  |
| --- | --- | --- |
| Tên collection | Tên attribute | Chú thích |
| SANPHAM | MASP (index) | Mã sản phẩm |
| TENSP | Tên sản phẩm |
| SL | Số lượng |
| GIA | Giá |
| NHANVIEN | MANV (index) | Mã nhân viên |
| HOTEN | Họ và tên nhân viên |
| CHUCVU | Chức vụ |
| HOADON | MAHD (index) | Mã hóa đơn |
| KHACHHANG | Tên khách hàng |
| MANV | Mã nhân viên ghi hóa đơn |
| CTHD | MAHD (index) | Mã hóa đơn |
| MASP (index) | Mã sản phẩm |
| SOLUONG | Số lượng sản phẩm |
| THANHTIEN | Thành tiền |

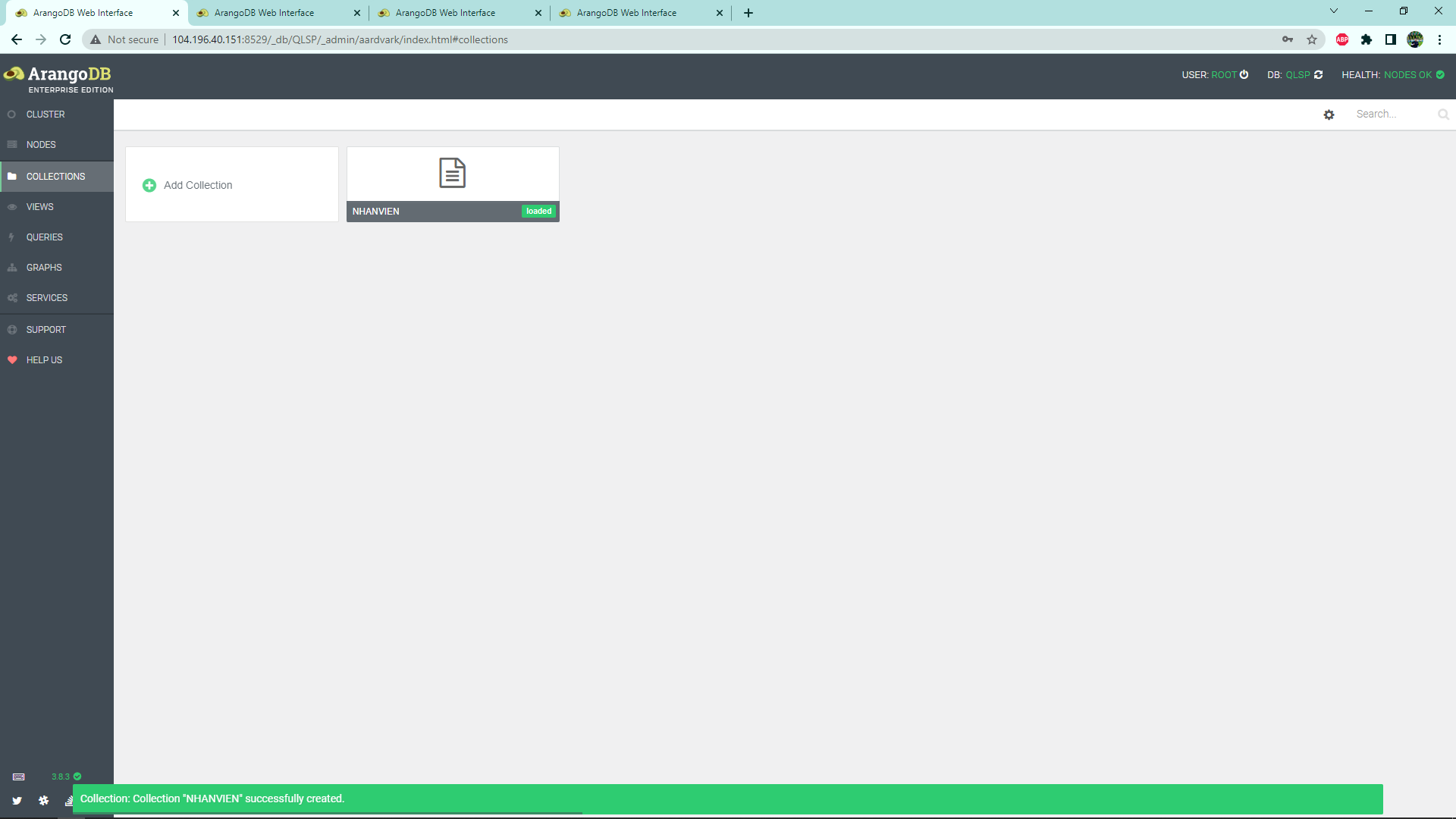
**Tạo Database trong ArangoDB**



**Tạo collection:**







**Thêm dữ liệu vào collection NHANVIEN:**

AQL:

for nhanvien in[

{MANV: 'NV01',HOTEN:'Nguyen Nhu Nhut',CHUCVU: 'Quan ly'},

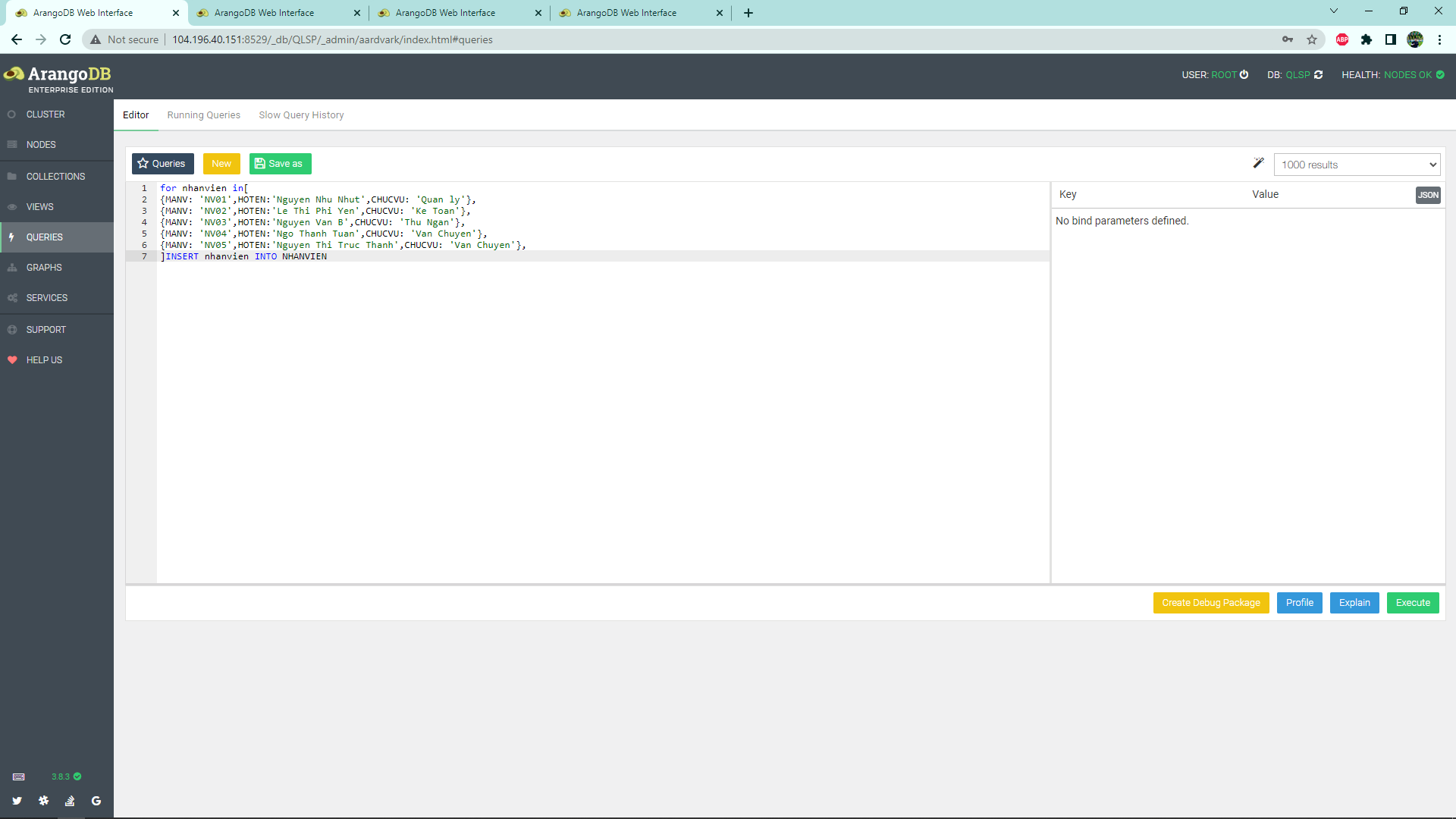
{MANV: 'NV02',HOTEN:'Le Thi Phi Yen',CHUCVU: 'Ke Toan'},

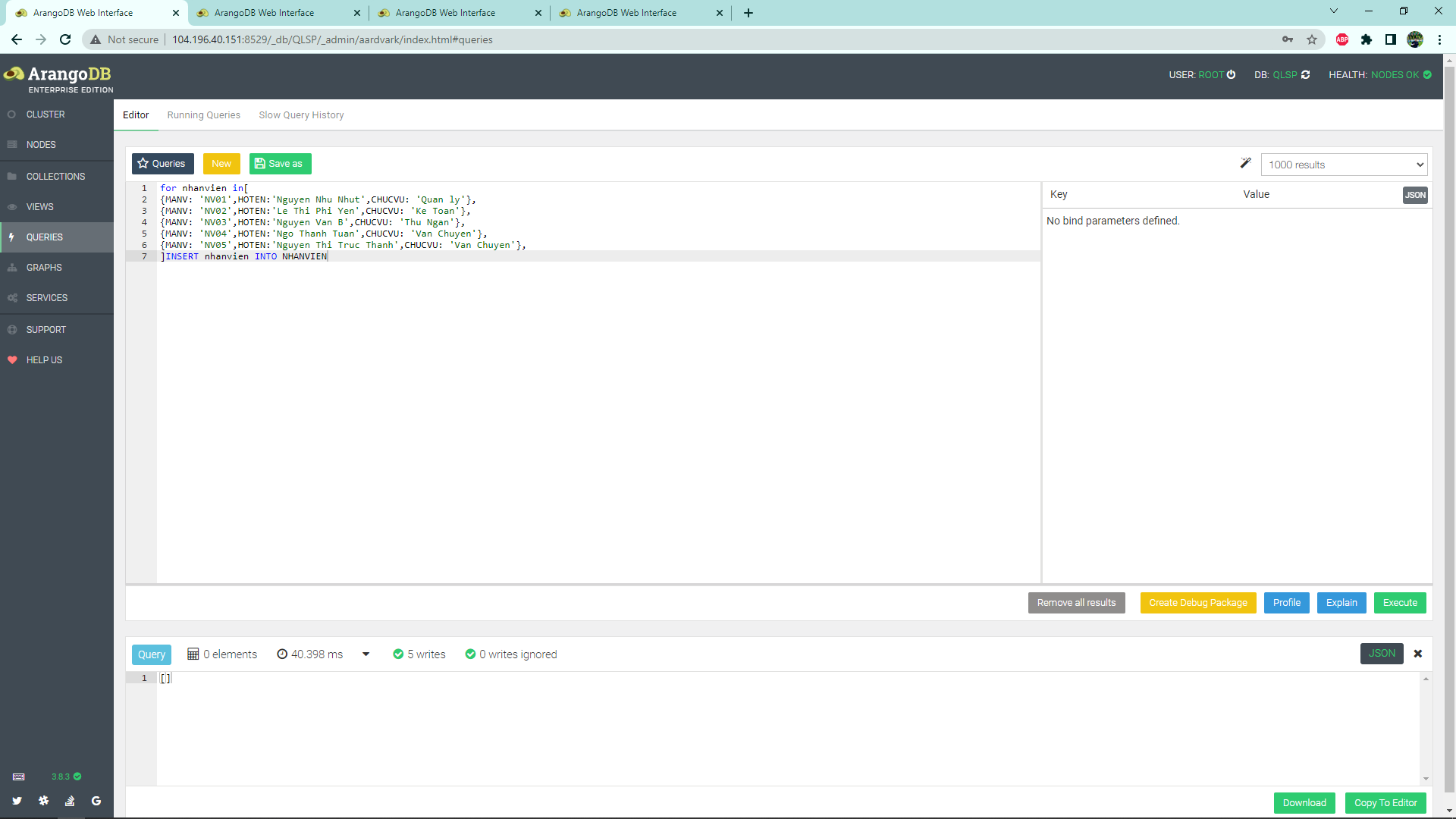
{MANV: 'NV03',HOTEN:'Nguyen Van B',CHUCVU: 'Thu Ngan'},

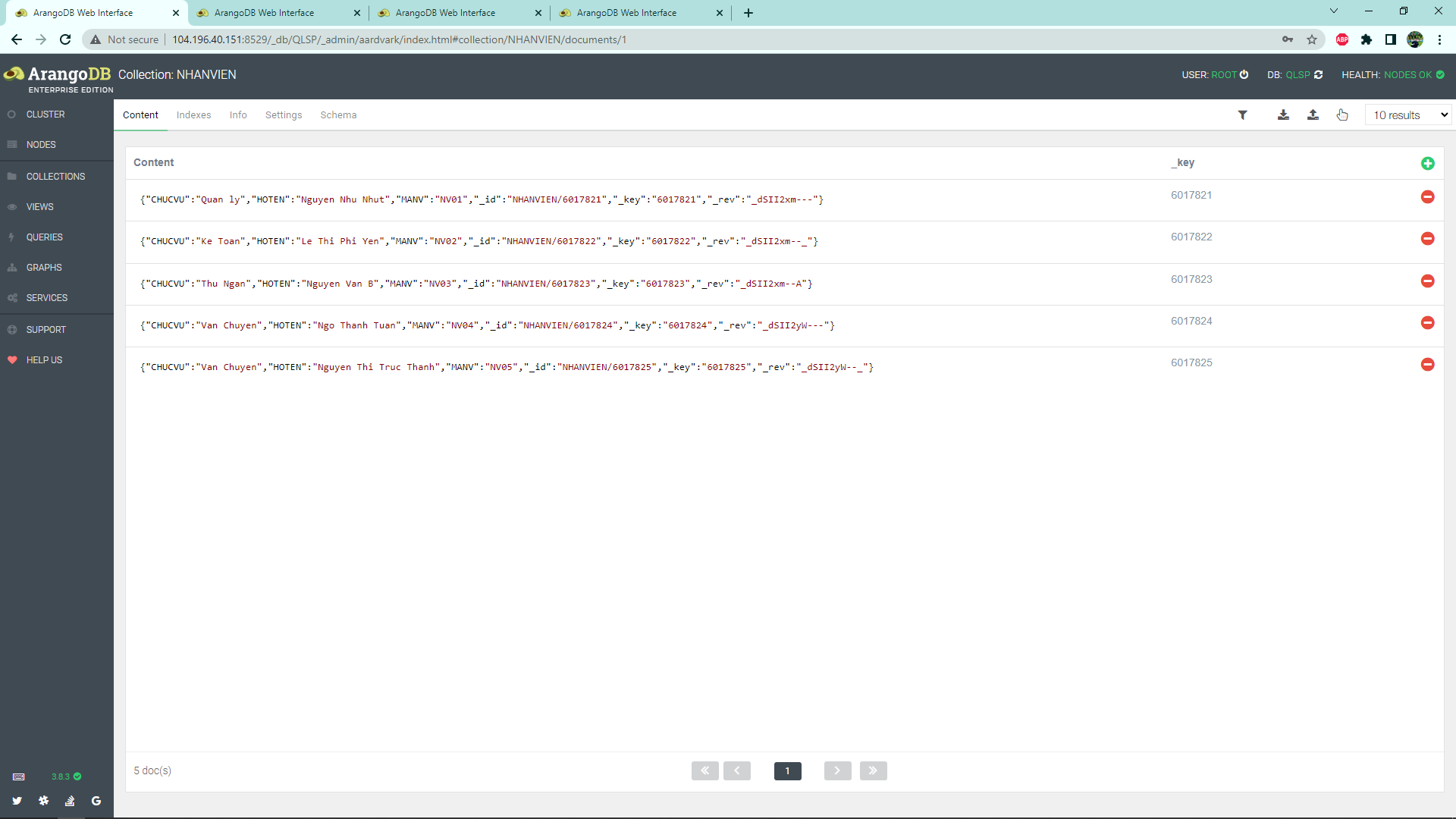
{MANV: 'NV04',HOTEN:'Ngo Thanh Tuan',CHUCVU: 'Van Chuyen'},

{MANV: 'NV05',HOTEN:'Nguyen Thi Truc Thanh',CHUCVU: 'Van Chuyen'},

]INSERT nhanvien INTO NHANVIEN







Làm tương tự để thêm data vào các collection SANPHAM, HOADON,CTHD với script sau:

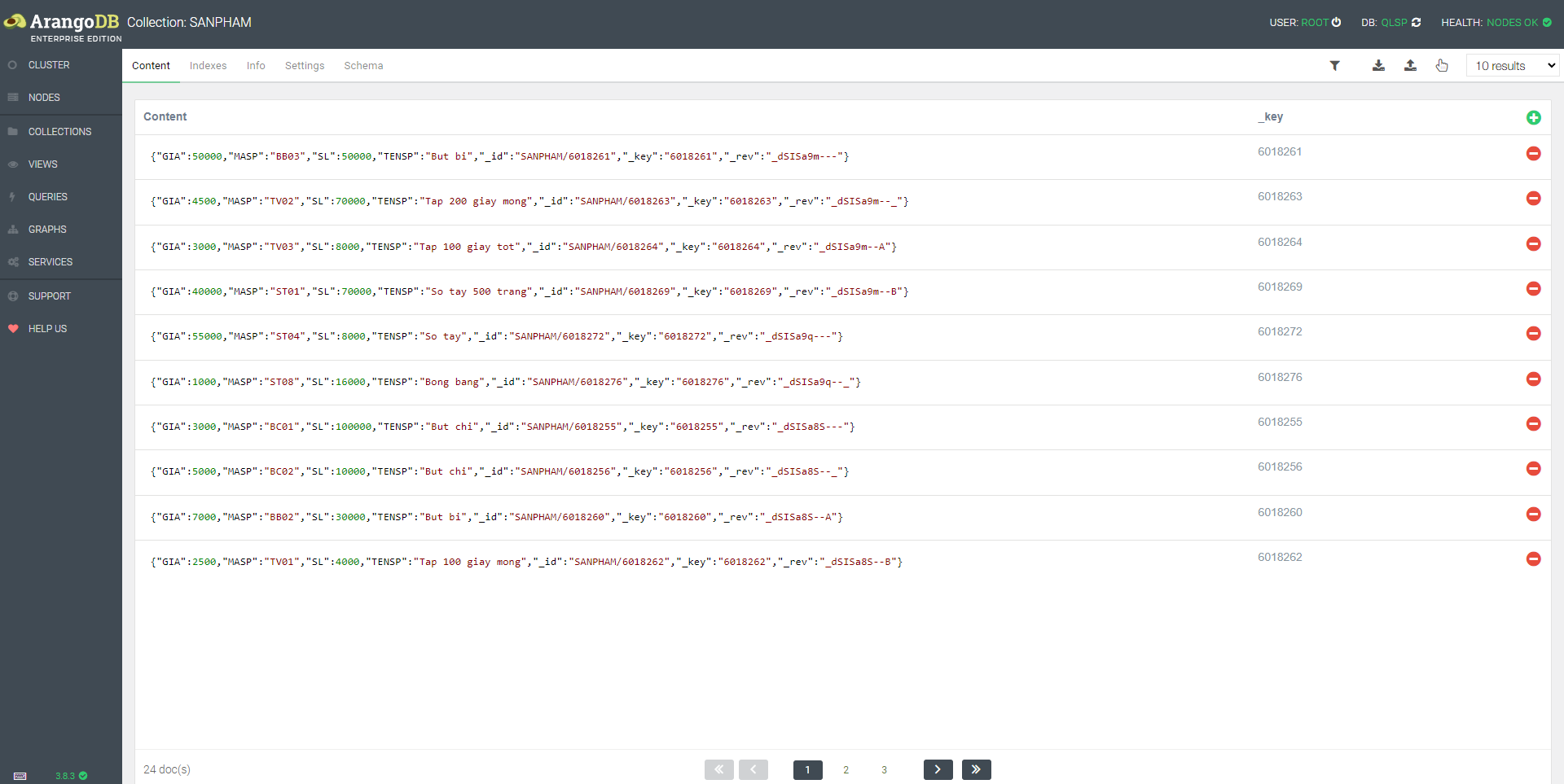
|  |
| --- |
| for sanpham in[  {MASP: 'BC01',TENSP: 'But chi',GIA: 3000,SL: 100000},  {MASP: 'BC02',TENSP: 'But chi',GIA: 5000,SL: 10000},  {MASP: 'BC03',TENSP: ' But chi',GIA: 3500,SL: 20000},  {MASP: 'BC04',TENSP: 'But chi',GIA: 30000,SL: 300000},  {MASP: 'BB01',TENSP: 'But bi',GIA: 5000,SL: 500000},  {MASP: 'BB02',TENSP: 'But bi',GIA: 7000,SL: 30000},  {MASP: 'BB03',TENSP: 'But bi',GIA: 50000,SL: 50000},  {MASP: 'TV01',TENSP: 'Tap 100 giay mong',GIA: 2500,SL: 4000},  {MASP: 'TV02',TENSP: 'Tap 200 giay mong',GIA: 4500,SL: 70000},  {MASP: 'TV03',TENSP: 'Tap 100 giay tot',GIA: 3000,SL: 8000},  {MASP: 'TV04',TENSP: 'Tap 200 giay tot',GIA: 5500,SL: 40000},  {MASP: 'TV05',TENSP: 'Tap 100 trang',GIA: 23000,SL: 3000},  {MASP: 'TV06',TENSP: 'Tap 200 trang',GIA: 53000,SL: 50000},  {MASP: 'TV07',TENSP: 'Tap 100 trang',GIA: 34000,SL: 80000},  {MASP: 'ST01',TENSP: 'So tay 500 trang',GIA: 40000,SL: 70000},  {MASP: 'ST02',TENSP: 'So tay loai 1',GIA: 55000,SL: 9000},  {MASP: 'ST03',TENSP: 'So tay loai 2',GIA: 51000,SL: 2000},  {MASP: 'ST04',TENSP: 'So tay',GIA: 55000,SL: 8000},  {MASP: 'ST05',TENSP: 'So tay mong',GIA:20000,SL: 1900},  {MASP: 'ST06',TENSP: 'Phan viet bang',GIA: 5000,SL: 3000},  {MASP: 'ST07',TENSP: 'Phan khong bui',GIA: 7000,SL: 12000},  {MASP: 'ST08',TENSP: 'Bong bang',GIA: 1000,SL: 16000},  {MASP: 'ST09',TENSP: 'But long',GIA: 5000,SL: 7000},  {MASP: 'ST10',TENSP: 'But long',GIA: 7000},  ] INSERT sanpham INTO SANPHAM |

|  |
| --- |
| For hoadon in[  {MAHD: 'HD01',KHACHANG: 'Le Ha Vinh',MANV: 'NV01'},  {MAHD: 'HD02',KHACHANG: 'Tran Ngoc Linh' ,MANV: 'NV02'},  {MAHD: 'HD03',KHACHANG: 'Le Hoai Thuong',MANV: 'NV01'},  {MAHD: 'HD04',KHACHANG: 'Nguyen Van Tam',MANV: 'NV01'},  ] INSERT hoadon INTO HOADON |

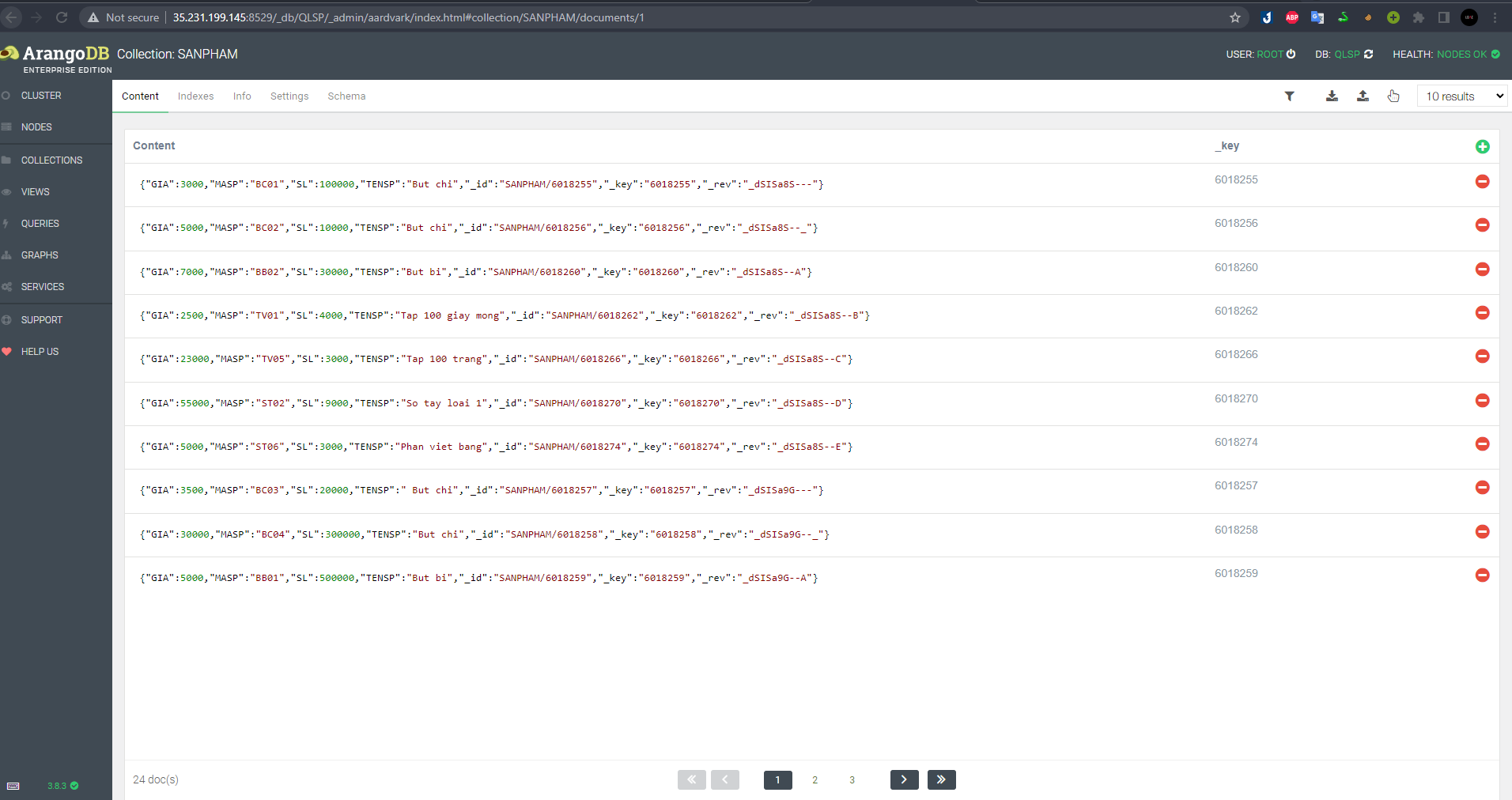
|  |
| --- |
| for cthd in[  {MAHD: 'HD01',MASP: 'TV02',SL: 10, THANHTIEN: 45000},  {MAHD: 'HD01',MASP: 'ST01',SL: 5, THANHTIEN:200000},  {MAHD: 'HD01',MASP: 'BC01',SL: 5, THANHTIEN:15000},  {MAHD: 'HD01',MASP: 'BC02',SL: 10, THANHTIEN:100000},  {MAHD: 'HD01',MASP: 'ST08',SL: 10, THANHTIEN:10000},  {MAHD: 'HD02',MASP: 'BC04',SL: 20, THANHTIEN:600000},  {MAHD: 'HD02',MASP: 'BB01',SL: 20, THANHTIEN:100000},  {MAHD: 'HD02',MASP: 'BB02',SL: 20, THANHTIEN:140000},  {MAHD: 'HD03',MASP: 'BB03',SL: 10, THANHTIEN:500000},  {MAHD: 'HD04',MASP: 'TV01',SL: 20, THANHTIEN:50000},  {MAHD: 'HD04',MASP: 'TV02',SL: 10, THANHTIEN:45000},  {MAHD: 'HD04',MASP: 'TV03',SL: 10, THANHTIEN:30000},  {MAHD: 'HD04',MASP: 'TV04',SL: 10, THANHTIEN:55000},  ] INSERT cthd INTO CTHD |

**Kiểm tra data ở các server sau khi Insert data:**

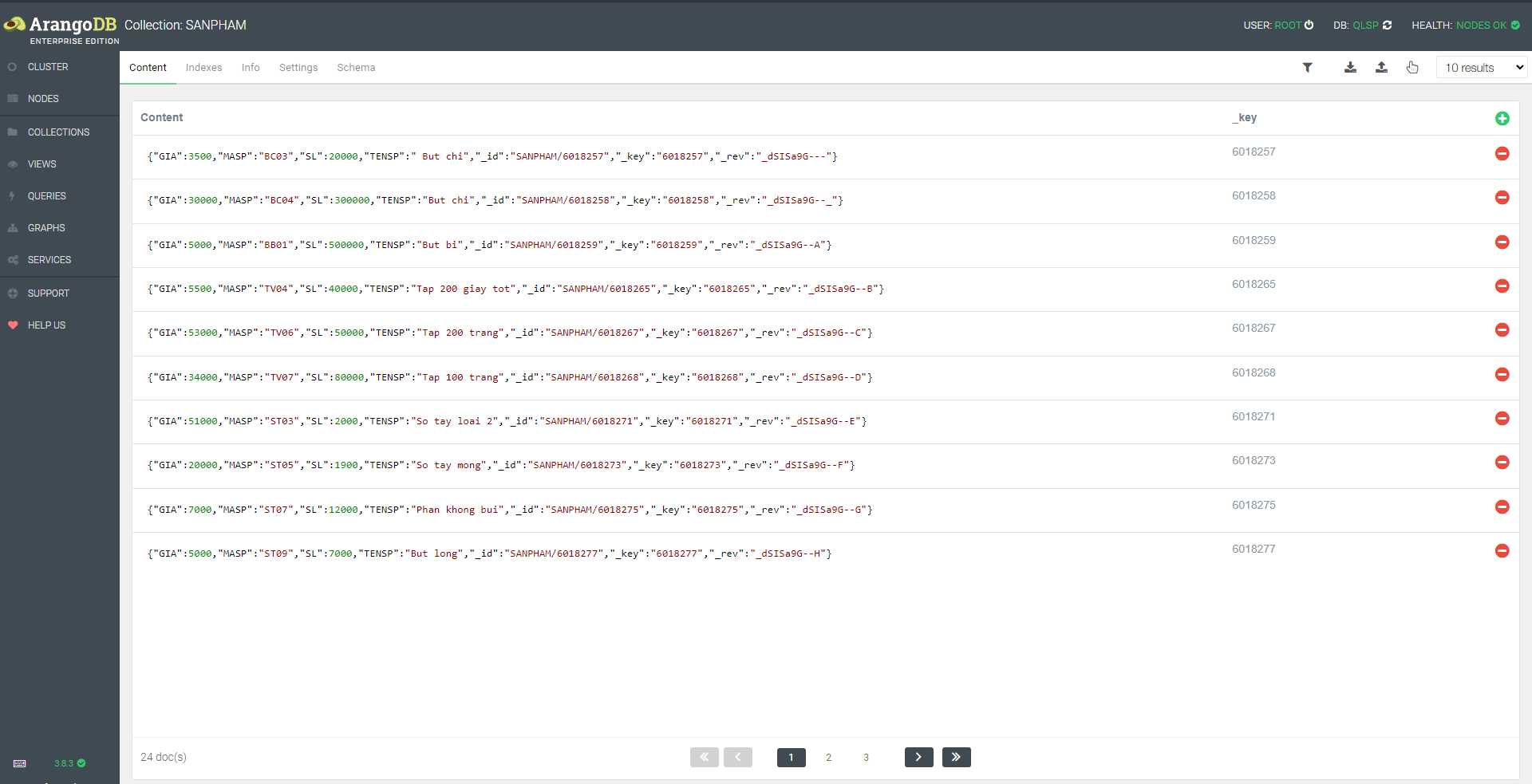
**Server 1:**



**Server 2:**

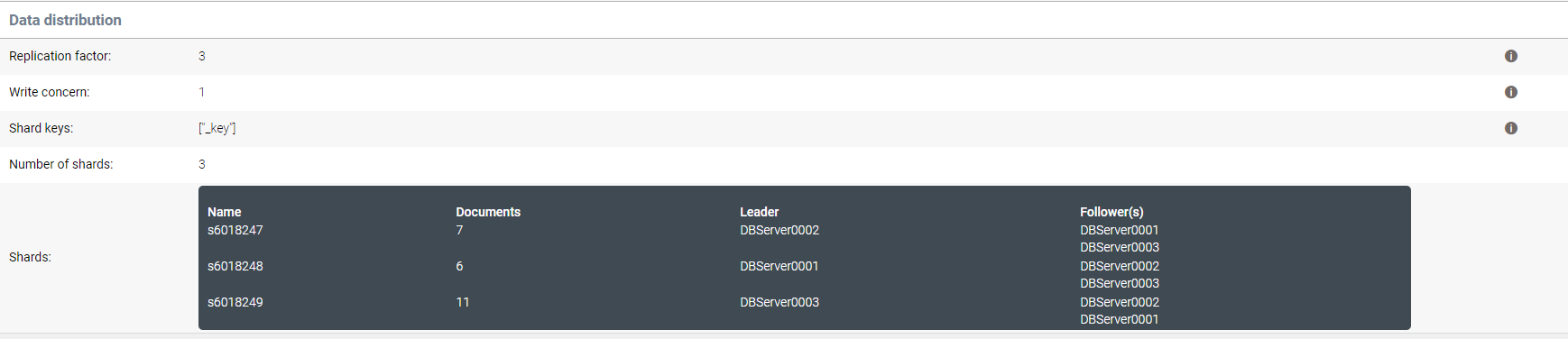


**Server 3:**



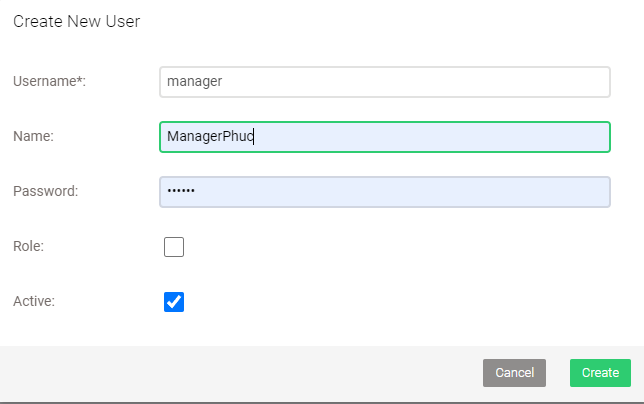
Ta thấy data của collection SANPHAM đã được phân tán và xuất hiện ở cả 3 servers.

Phân bổ Shards ở collection SANPHAM:



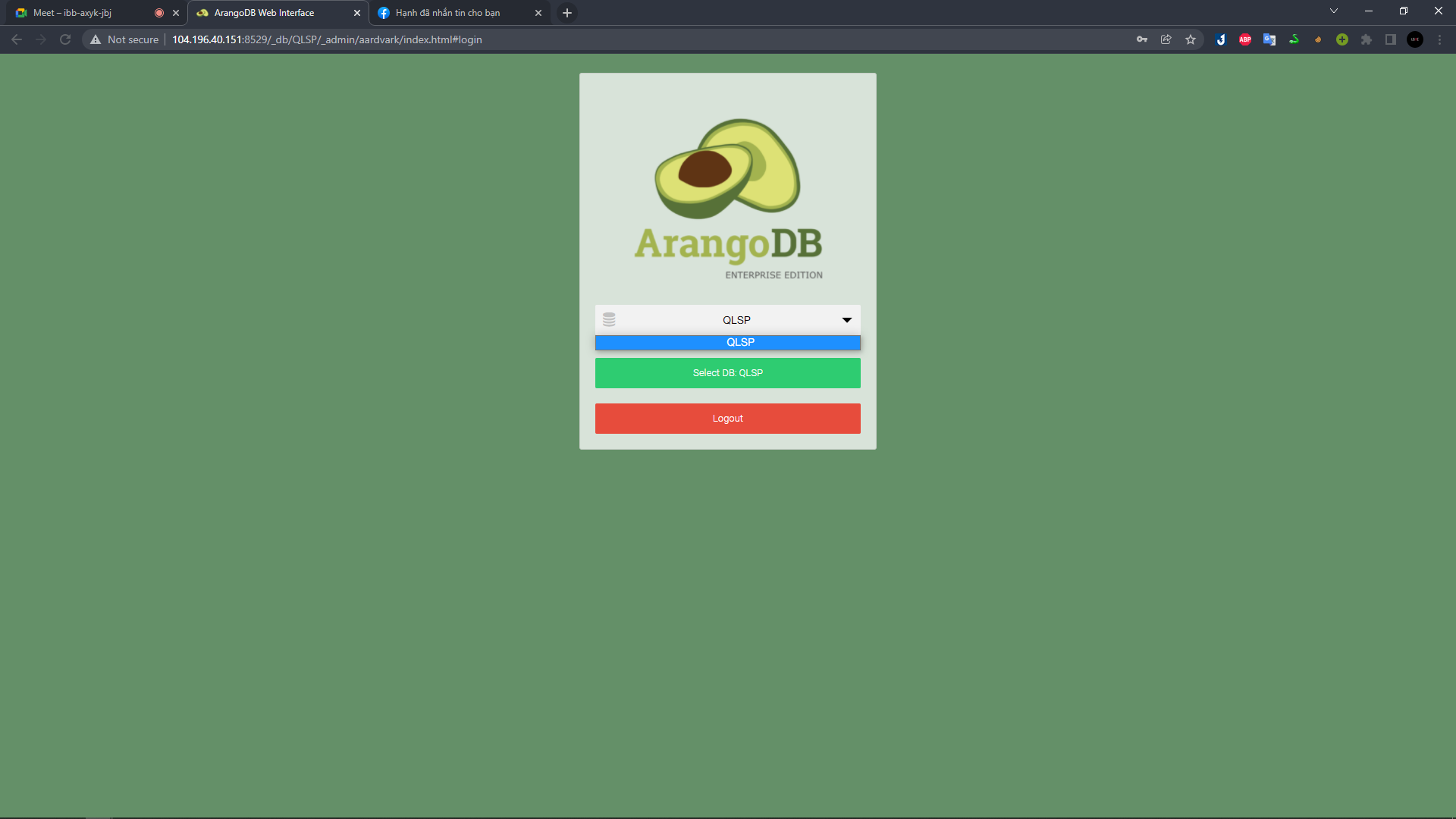
### b) Phân quyền:

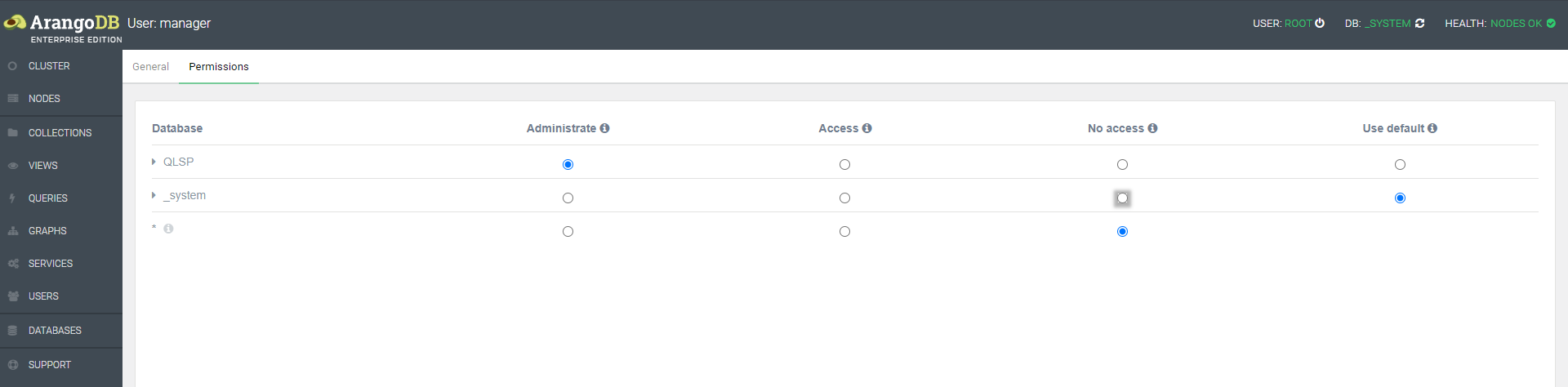
**Tạo user manager có full quyền trên database QLSP:**



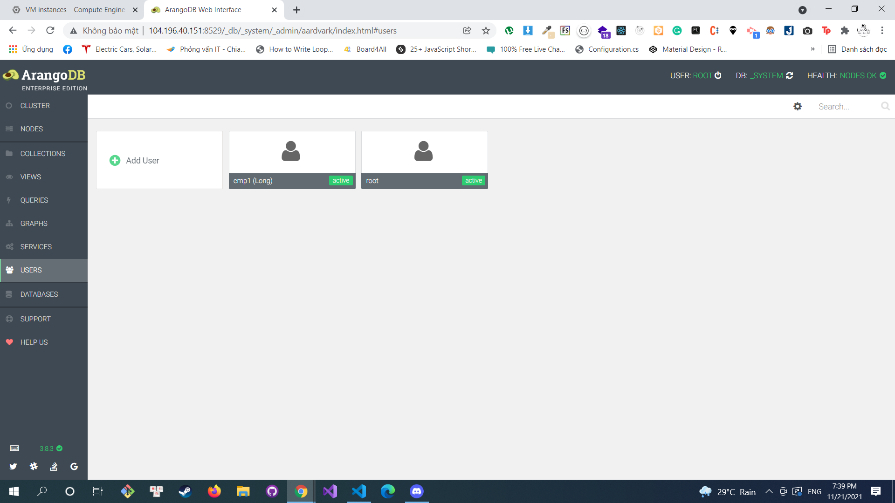
Cấp quyền quản trị cho user manager:

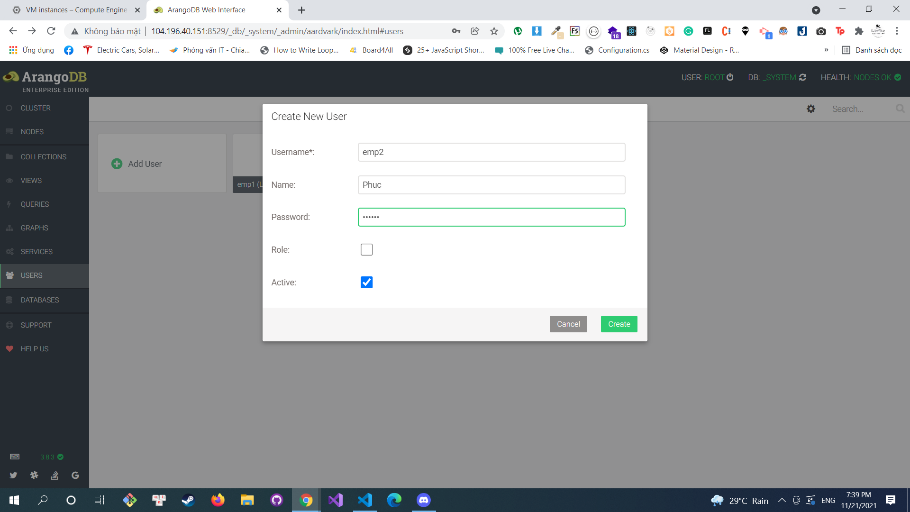
Với quyền admin, manager có thể tạo, xóa và sử dụng tất cả các function trên các collection nhưng chỉ có quyền trên mỗi database QLSP:

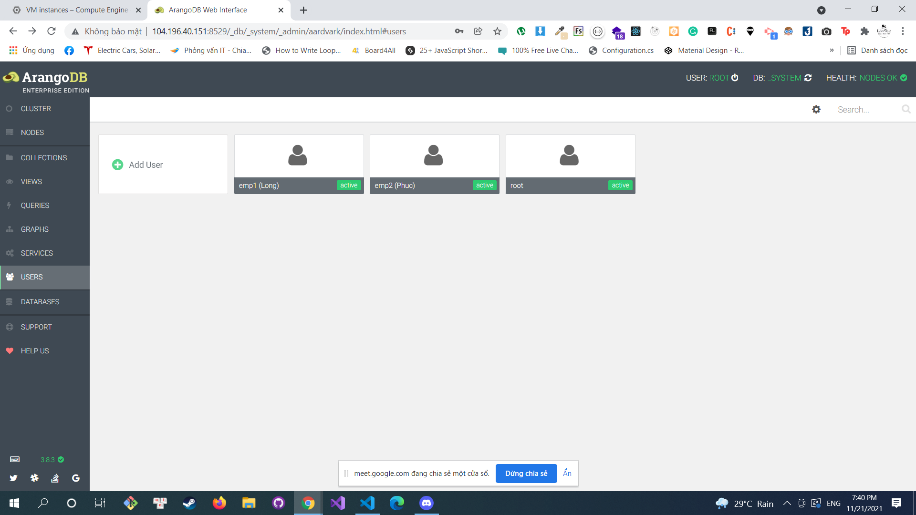




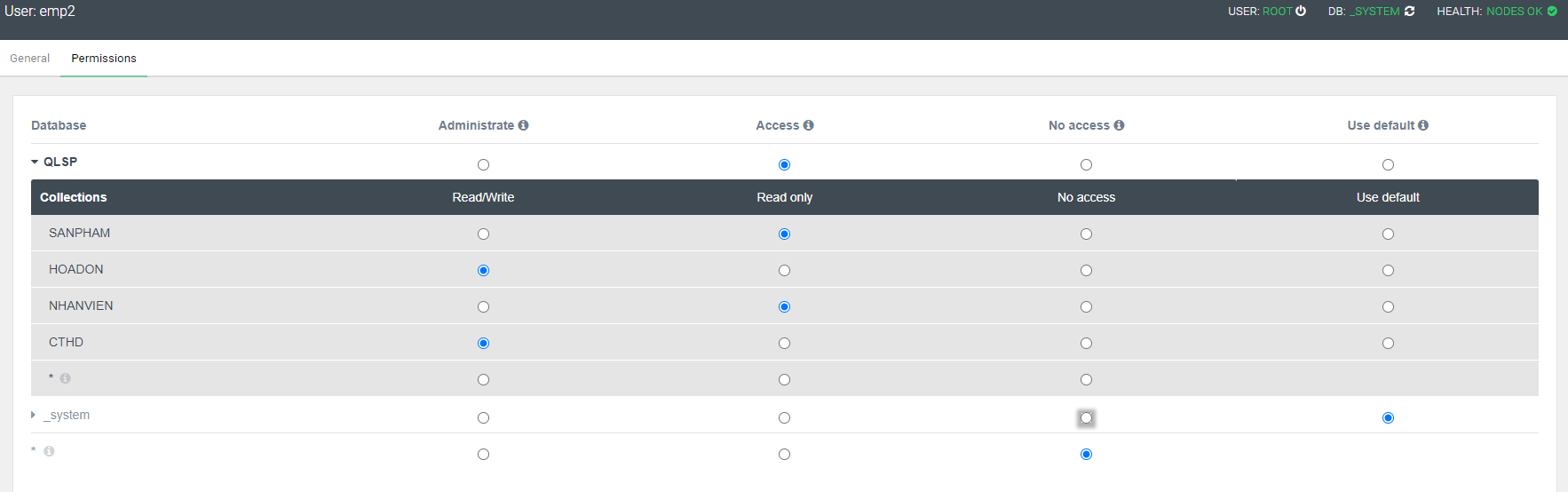
**Tạo user emp2**





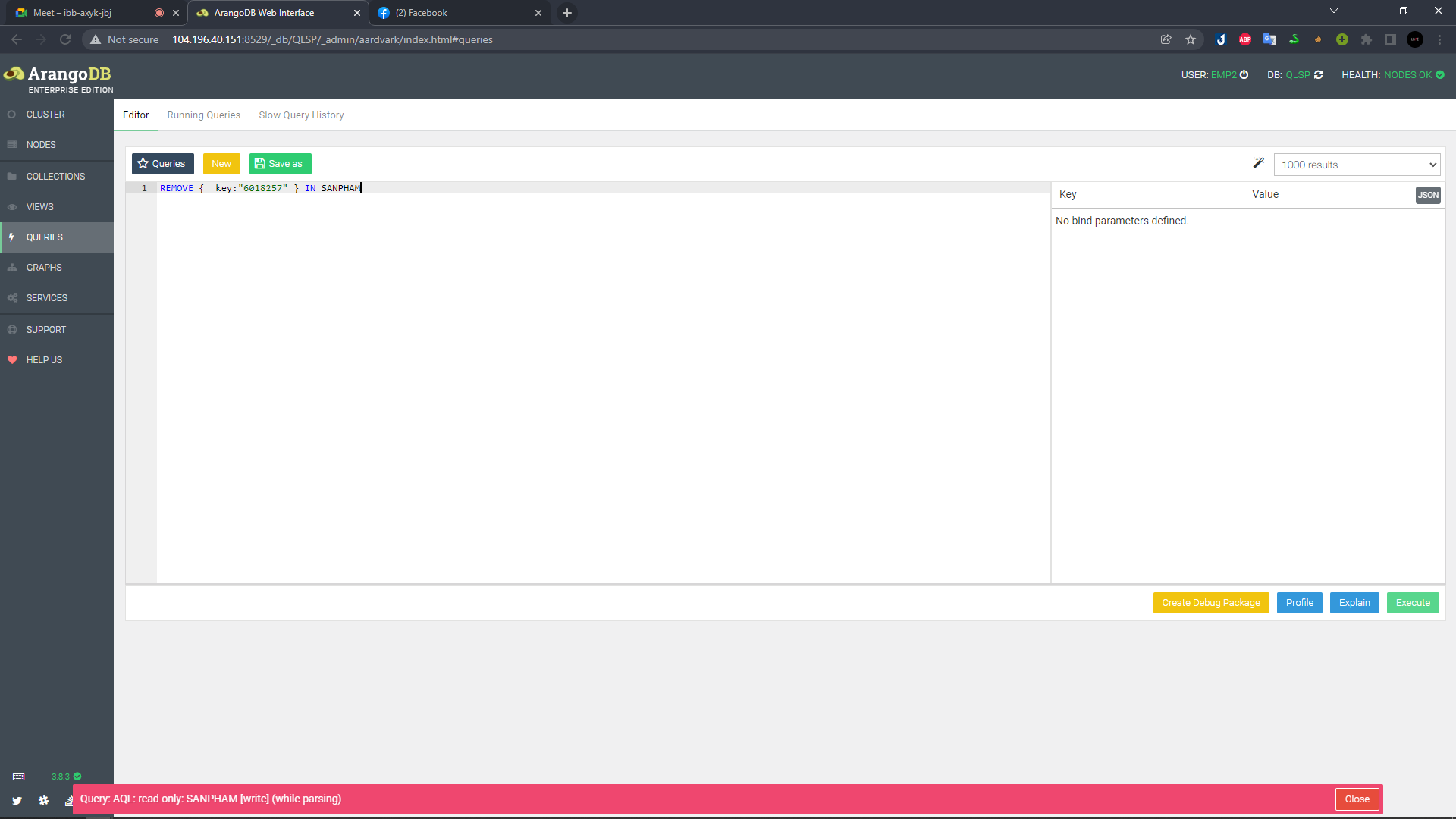


**Cấp quyền read/write trên một số collection nhất định cho user emp2:**



**Kiểm tra quyền của user emp2:**

Thử xóa một document trên collection SANPHAM:



Ta thấy hiển thị lỗi “Query: AQL: read only: SANPHAM [write](while parsing)” do user emp2 không có quyền write trên collection SANPHAM.

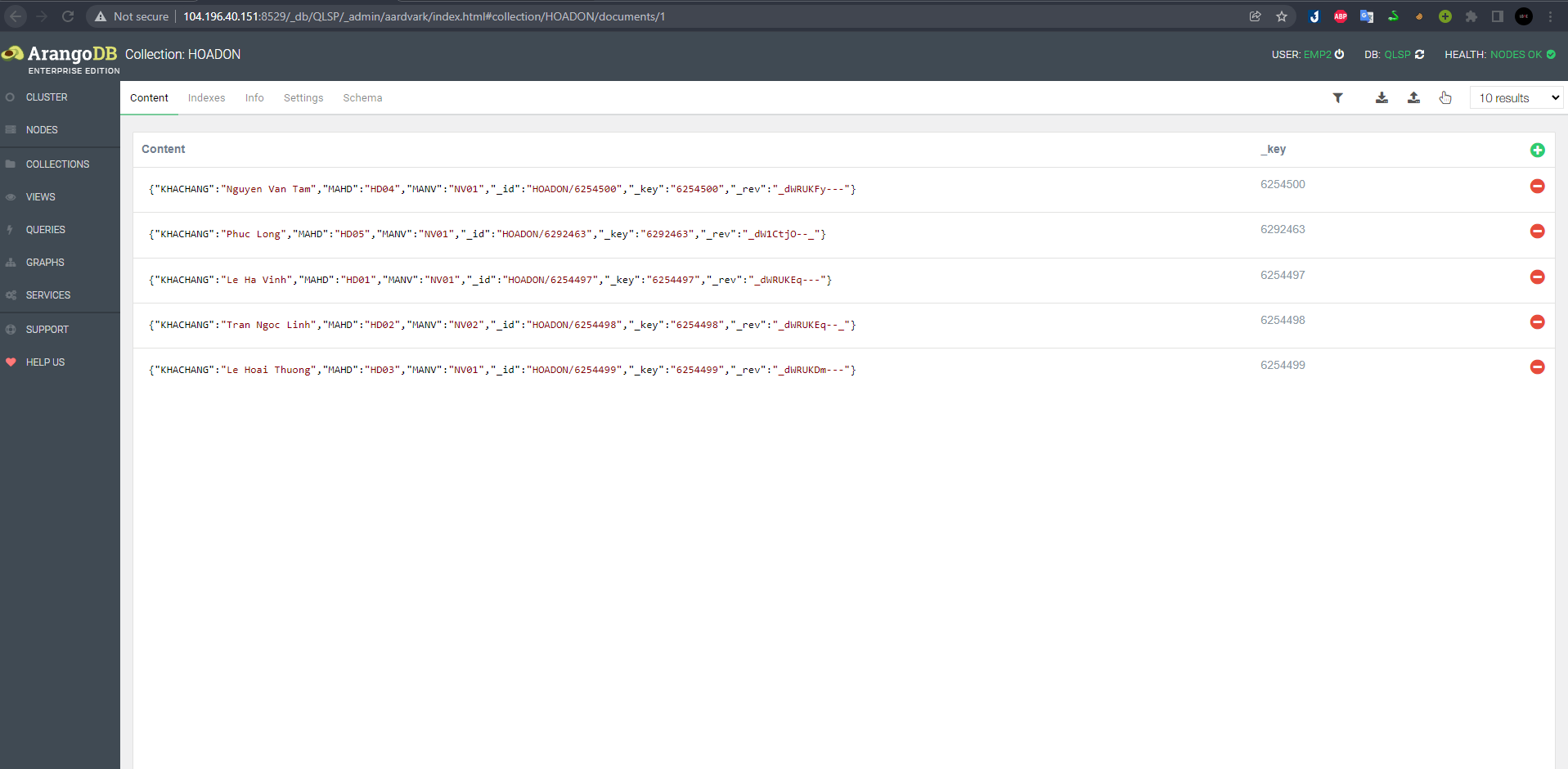
**Thử thêm 1 document trên collection HOADON:**



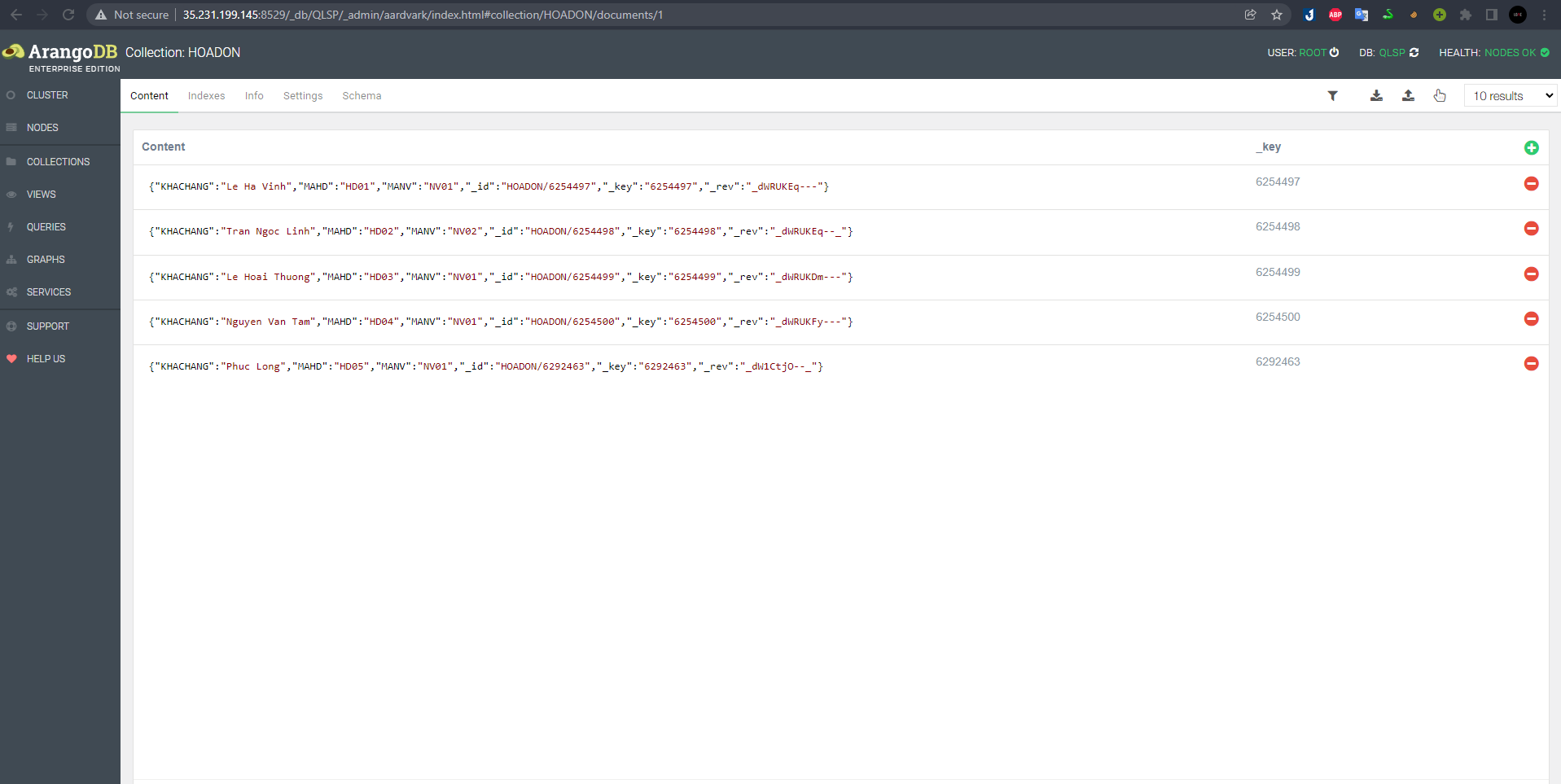
Ta thấy kết quả hiển thị là 1 writes, có nghĩa là việc thêm đã thành công.

Tiếp theo, ta kiểm tra xem document trên đã được thêm vào database ở các server hay chưa:

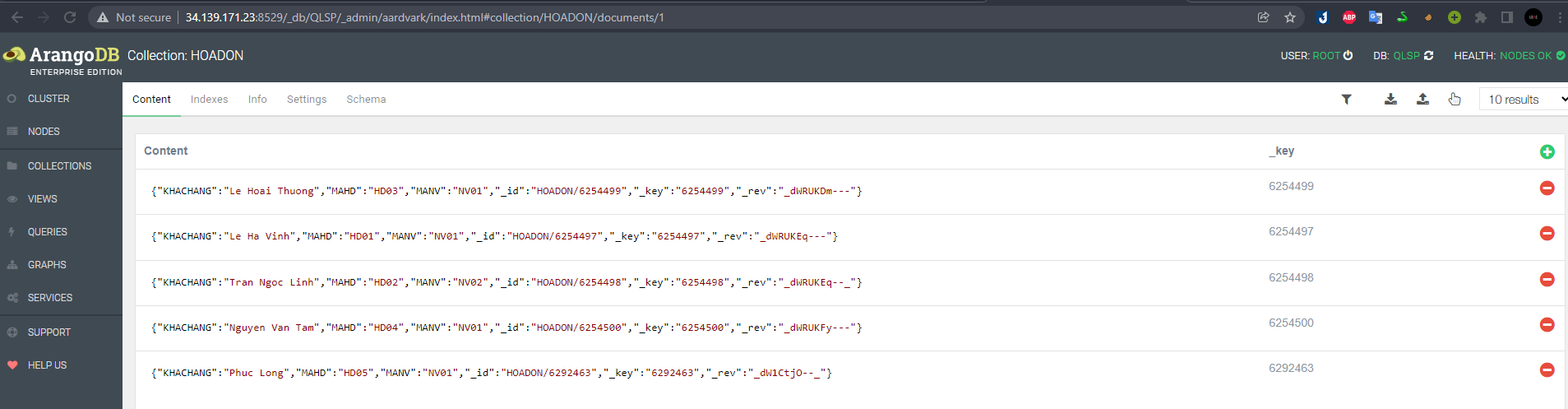
**Server 1:**



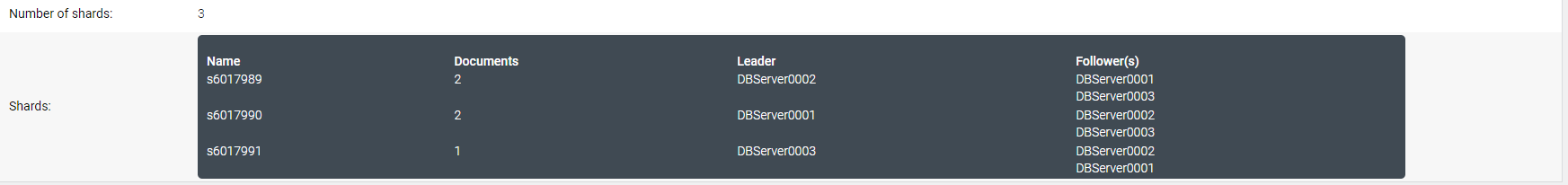
**Server 2:**



**Server 3:**



**Ta kiểm tra xem server có phân bổ shard (tự phân tán) lại hay không:**



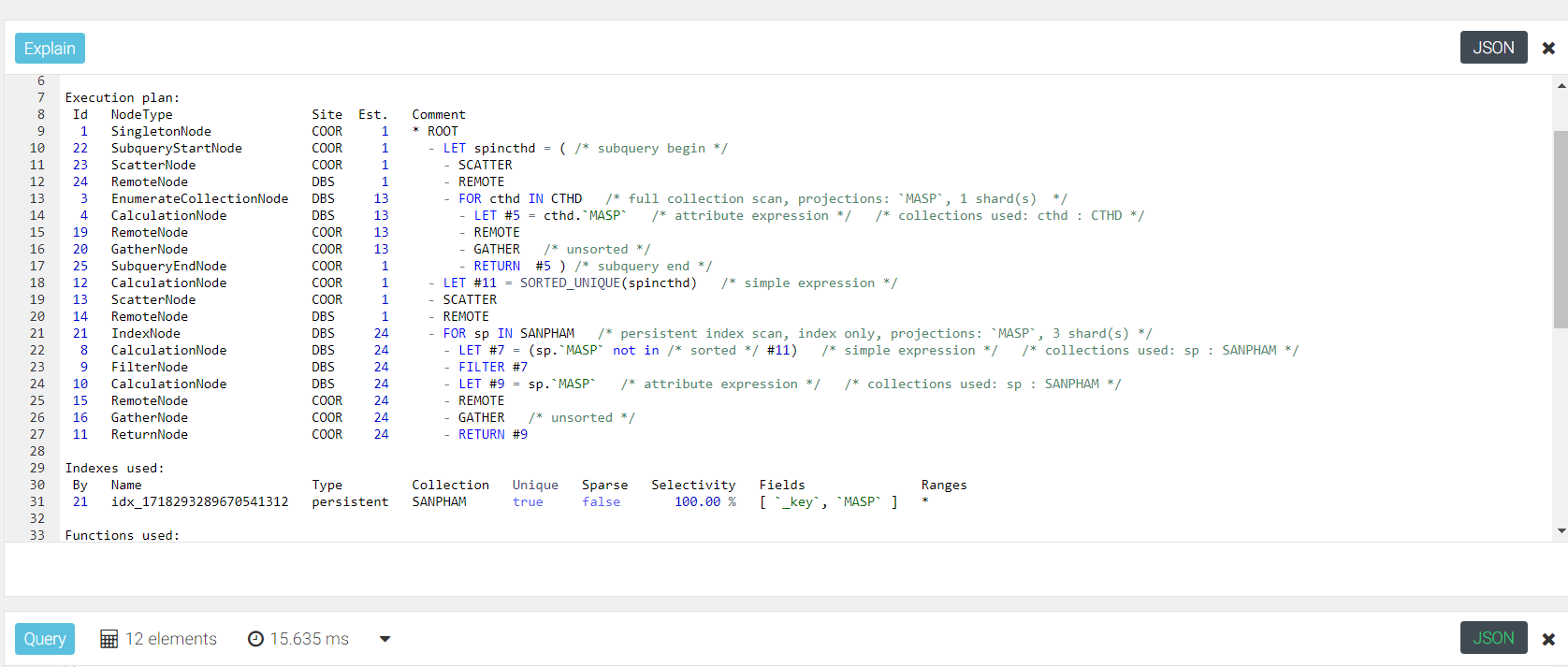
Ta thấy số lượng document là 5 thay vì 4 như ban đầu, chứng tỏ server đã tự phân bổ shard lại.

### c) Tạo Index:

Để tránh việc truy xuất dữ liệu phải quét hết collection (tốn thời gian và tài nguyên), ta tạo index trên các attribute. Để truy xuất dữ liệu chính xác trên các document, lựa chọn tối ưu nhất là hash index. Primary Index của Key attribute cũng là một hash index.

Vấn đề quan trọng nhất là performance. Dù là RDBMS hay NoSQL, nếu không đánh index hoặc đánh index sai thì câu query sẽ không được lên execution plan và optimize, index bị bỏ qua và worst case sẽ là full table (collection) scan.

Ta xét câu query sau:



let spincthd = ( FOR cthd IN CTHD return cthd.MASP )

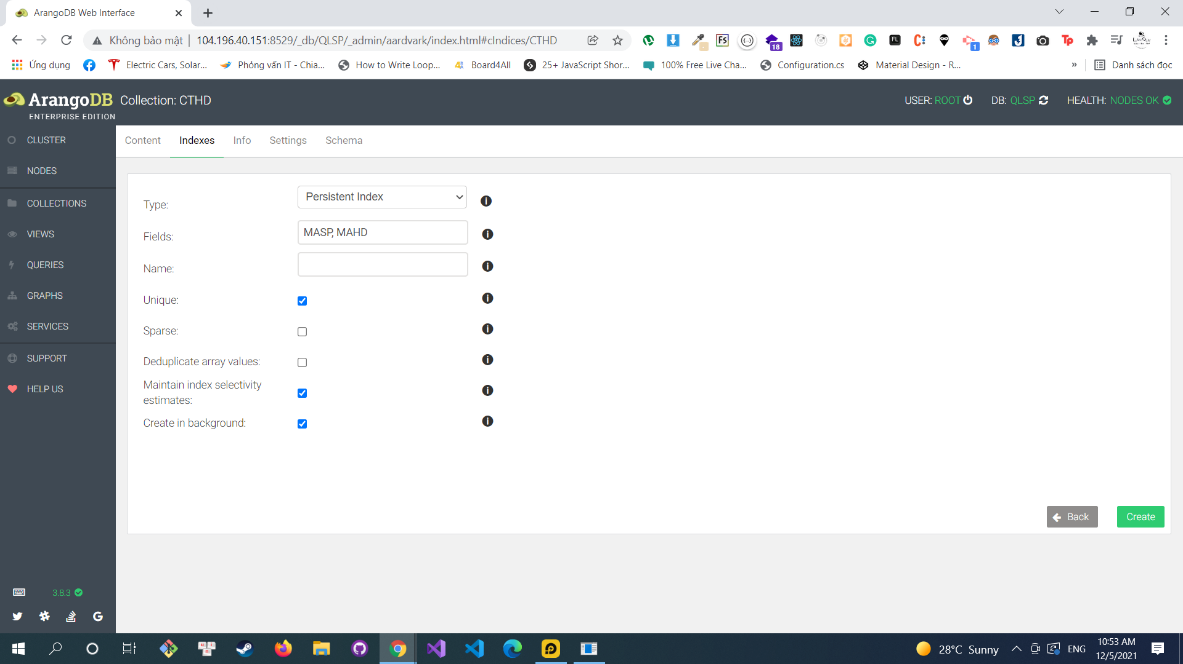
FOR sp IN SANPHAM

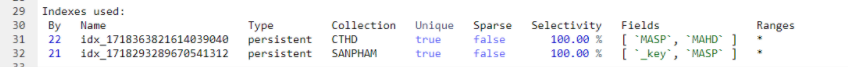
FILTER sp.MASP NOT IN spincthd

RETURN sp.MASP

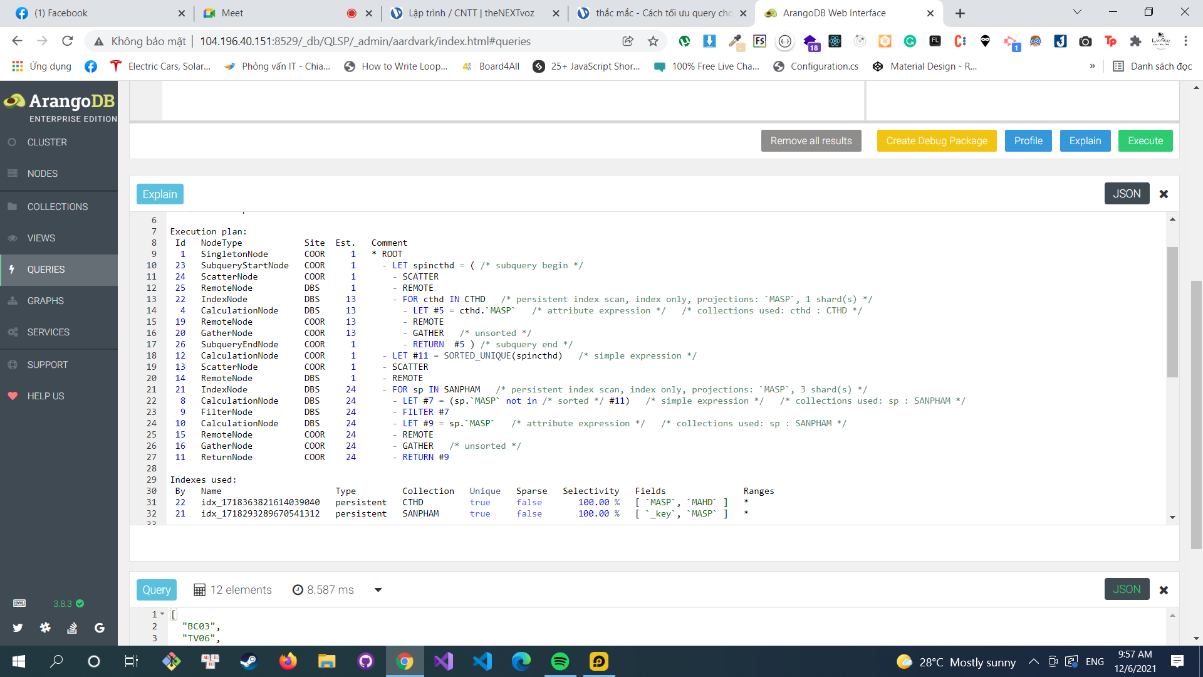
Ở hình trên, ta thấy dòng 13 EnumerateCollectionode hiện full collection scan và thời gian query là 15.635ms.

Tiếp theo ta tiến hành tạo index trên collection CTHD: Ở đây cặp khóa chính là (MASP, MAHD)

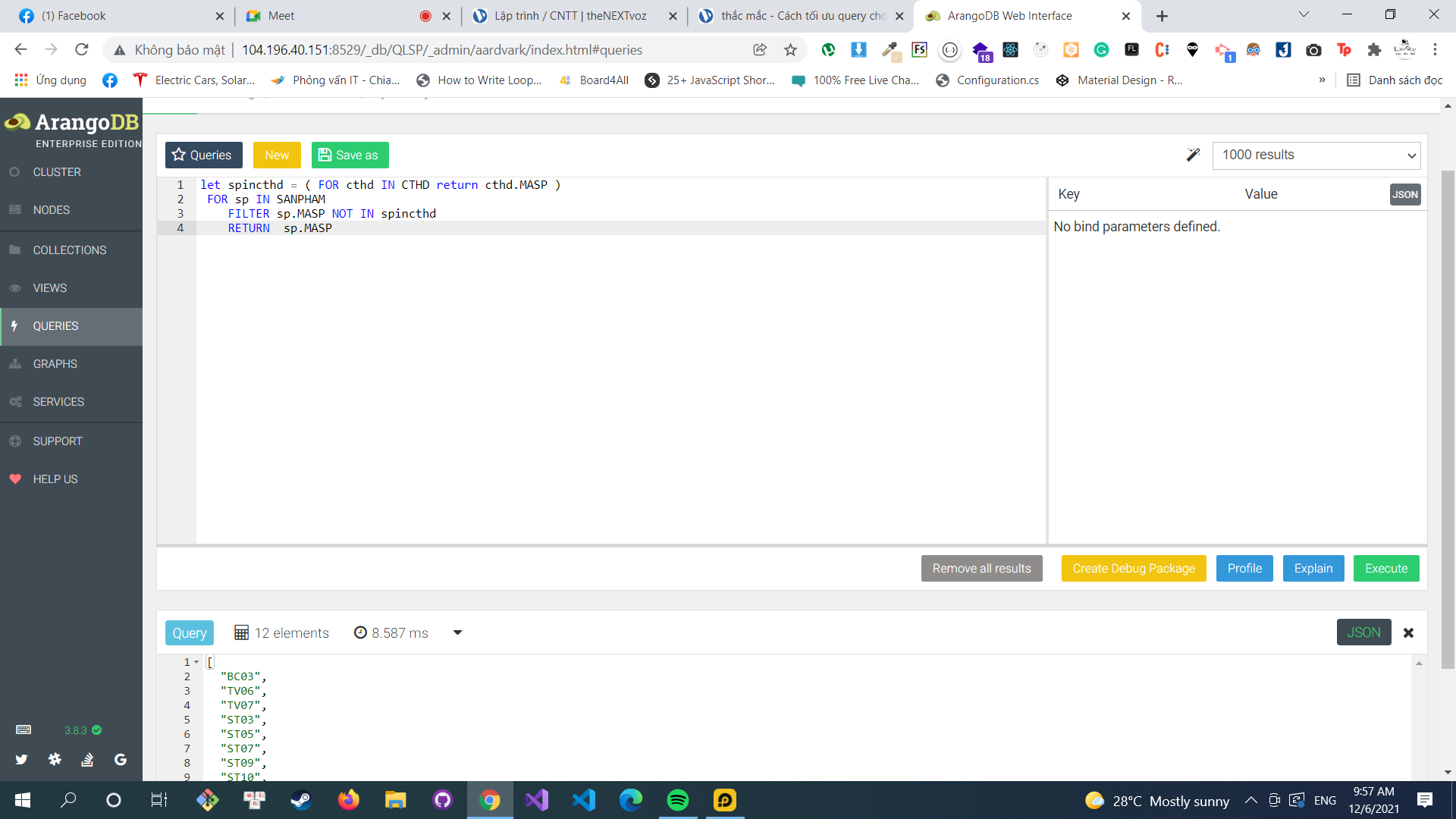




Ở ảnh trên, ta thấy có 2 Indexes used là [‘\_key’,'MASP’] (\_key là khóa chính do ArangoDB tự đặt khi ta insert document vào collection) và [‘MASP’,'MAHD’].



Ở hình trên ta thấy, dòng 13 hiển thị persistent index scan thay vì full table scan và thời gian query được rút ngắn còn 8.587ms.



### b) Demo query trong bài toán của nhóm:

|  |  |
| --- | --- |
| **Truy vấn:** | **AQL** |
| **Tính tổng giá tiền của hóa đơn:** | FOR doc IN CTHD  COLLECT group = doc.MAHD  AGGREGATE s= SUM(doc.THANHTIEN)  RETURN {group, s} |
| **In danh sách sản phẩm chưa bán được** | let spincthd = ( FOR cthd IN CTHD  return cthd.MASP )  FOR sp IN SANPHAM  FILTER sp.MASP NOT IN spincthd  RETURN sp.MASP |
| **In danh sách sản phẩm đã mua của khách hàng có tên ‘Le Ha Vinh’** | FOR hoadon IN HOADON  FOR cthd IN CTHD  FOR sp IN SANPHAM  FILTER hoadon.MAHD == cthd.MAHD && hoadon.KHACHANG== "Le Ha Vinh" && sp.MASP== cthd.MASP  COLLECT TENSP= sp.TENSP, MASP= sp.MASP  RETURN {TENSP, MASP} |
| **In danh sách và số lượng của các sản phẩm đã bán** | FOR cthd IN CTHD  FOR sp IN SANPHAM  FILTER cthd.MASP==sp.MASP  COLLECT TENSP = sp.TENSP  WITH COUNT INTO SLDB  RETURN {  TENSP: TENSP,  SLDB: SLDB  } |
| **In top 3 sản phẩm bán chạy nhất** | FOR cthd IN CTHD  FOR sp IN SANPHAM  FILTER cthd.MASP==sp.MASP  COLLECT TENSP = sp.TENSP  WITH COUNT INTO SLDB  SORT SLDB DESC  LIMIT 3  RETURN {  TENSP: TENSP,  SLDB: SLDB  } |