Bài tập lớn 2: Cơ chế phân tán trong hệ quản trị NoSQL

Lê Minh Chánh
IS211.O11.HTCL
Trường Đại học Công nghệ thông tin
21521882@gm.uit.edu.vn

Đào Huy Hoàng
IS211.O11.HTCL
Trường Đại học Công nghệ thông tin
19521528@gm.uit.edu.vn

Lê Thị Lệ Trúc
IS211.O11.HTCL
Trường Đại học Công nghệ thông tin
21521586@gm.uit.edu.vn

Trần Thị Kim Anh
IS211.O11.HTCL
Trường Đại học Công nghệ thông tin
21520596@gm.uit.edu.vn

***Tóm tắt* — Trong bài tập lớn 2 này, nhóm chúng tôi sẽ tiến hành tìm hiểu và thực hiện cơ chế phân tán trong hệ quản trị NoSQL. Đầu tiên, chúng tôi sẽ giới thiệu về hệ quản trị CSDL NoSQL. Sau đó sẽ cài đặt trên 2 máy và thực hiện truy vấn giữa 2 máy. Rồi sẽ thực hiện thao tác dữ liệu qua lại giữa 2 máy và cuối cùng là cơ chế nhân bản trong RavenDB**

Từ khóa — Cơ sở dữ liệu phân tán, NoSQL, RavenDB

I. GIỚI THIỆU

NoSQL là một khái niệm chỉ về một lớp các hệ cơ sở dữ liệu không sử dụng mô hình quan hệ (RDBMS). RDBMS vốn tồn tại khá nhiều nhược điểm như có hiệu năng không tốt nếu kết nối dữ liệu nhiều bảng lại hay khi dữ liệu trong một bảng là rất lớn.

RavenDB là một cơ sở dữ liệu hướng tài liệu ACID hoàn toàn mã nguồn mở được viết bằng C# , được phát triển bởi Hibernating Rhinos Ltd. Nó đa nền tảng, được hỗ trợ trên Windows , Linux và Mac OS. RavenDB lưu trữ dữ liệu dưới dạng tài liệu JSON và có thể được triển khai trong các cụm phân tán với bản sao chính-chính.

Trong nghiên cứu này, chúng tôi sẽ tìm hiểu về NoSQL và RavenDB, sau đó lấy dữ liệu qua lại giữa 2 máy bằng RavenDB, thêm xóa sửa và thực hiện cơ chế nhân bản trong phân tán RavenDB.

II. TÀI NGUYÊN

A. *Lược đồ cơ sở dữ liệu*

***Hình 1:** Lược đồ CSDL*

B. *Thông tin chi tiết các cột*

- SANPHAM

Thuộc tính	Kiểu dữ liệu	Mô tả
MASP	VARCHAR	Mã sản phẩm
TENSP	VARCHAR	Tên sản phẩm
DVT	VARCHAR	Đơn vị tính
NUOCSX	VARCHAR	Nước sản xuất
GIA	VARCHAR	Giá sản phẩm

- NHANVIEN

Thuộc tính	Kiểu dữ liệu	Mô tả
MANV	VARCHAR	Mã nhân viên
HOTEN	VARCHAR	Tên nhân viên
SODT	NUMBER	Số điện thoại

- HOADON

Thuộc tính	Kiểu dữ liệu	Mô tả
SOHD	VARCHAR	Mã hóa đơn
NGHD	DATE	Ngày mua hàng
MAKH	VARCHAR	Mã khách hàng
MANV	VARCHAR	Mã nhân viên
TRIGIA	VARCHAR	Trị giá hóa đơn

- CTHD

Thuộc tính	Kiểu dữ liệu	Mô tả
SOHD	VARCHAR	Mã hóa đơn
MASP	VARCHAR	Mã sản phẩm
SL	NUMBER	Số lượng

- KHACHHANG

Thuộc tính	Kiểu dữ liệu	Mô tả
MAKH	VARCHAR	Mã khách hàng
HOTEN	VARCHAR	Tên khách hàng
DCHI	VARCHAR	Địa chỉ
SODT	VARCHAR	Số điện thoại
NGSINH	VARCHAR	Ngày sinh
DOANHSO	VARCHAR	Doanh số

III. GIỚI THIỆU VỀ HỆ QUẢN TRỊ NOSQL

A. *Giới thiệu về NoSQL*

1) Khái niệm:

NoSQL là một khái niệm chỉ về một lớp các hệ cơ sở dữ liệu không sử dụng mô hình quan hệ (RDBMS). RDBMS vốn tồn tại khá nhiều nhược điểm như có hiệu năng không tốt nếu kết nối dữ liệu nhiều bảng lại hay khi dữ liệu trong một bảng là rất lớn.

NoSQL ra đời năm 1998 bởi Carlo Strozzi khi ông lập mới một hệ cơ sở dữ liệu quan hệ mã nguồn mở nhanh và nhẹ không liên quan đến SQL 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: phân tán (distributed) + không ràng buộc (non-relational).

Đặc điểm của NoSQL là:

- NoSQL lưu trữ dữ liệu của mình theo dạng cặp giá trị "key – value". Sử dụng số lượng lớn các node để lưu trữ thông tin
- Chấp nhận dữ liệu bị trùng lặp do một số node sẽ lưu cùng thông tin giống nhau
- Phi quan hệ – không có ràng buộc nào cho việc nhất quán dữ liệu
- Có hiệu suất cao (high performance) và tính sẵn sàng cao (high availability)

2) Vì sao nên sử dụng NoSQL?

Cơ sở dữ liệu NoSQL là lựa chọn cực kỳ thích hợp cho nhiều ứng dụng hiện đại, ví dụ như di động, web và trò chơi đòi hỏi phải sử dụng cơ sở dữ liệu cực kỳ thiết thực, linh hoạt, có khả năng thay đổi quy mô và hiệu năng cao để đem đến cho người dùng trải nghiệm tuyệt vời.

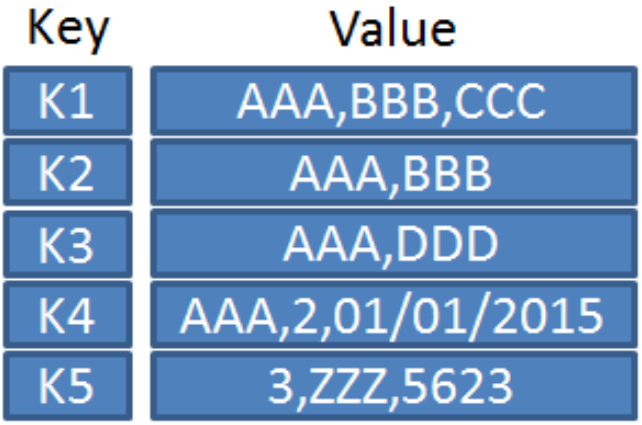
- Linh hoạt: Cơ sở dữ liệu NoSQL thường cung cấp các sơ đồ linh hoạt giúp công đoạn phát triển nhanh hơn và có khả năng lặp lại cao hơn. Mô hình dữ liệu linh hoạt biến cơ sở dữ liệu NoSQL thành lựa chọn lý tưởng cho dữ liệu không được tổ chức thành cấu trúc hoặc có cấu trúc chưa hoàn chỉnh.
- Khả năng thay đổi quy mô: Cơ sở dữ liệu NoSQL thường được thiết kế để tăng quy mô bằng cách sử dụng các cụm phần cứng được phân phối thay vì tăng quy mô bằng cách bổ sung máy chủ mạnh và tốn kém. Một số nhà cung cấp dịch vụ đám mây xử lý các hoạt động này một cách không công khai dưới dạng dịch vụ được quản lý đầy đủ.
- Hiệu năng cao: Cơ sở dữ liệu NoSQL được tối ưu hóa theo các mô hình dữ liệu cụ thể và các mẫu truy cập giúp tăng hiệu năng cao hơn so với việc cố gắng đạt được mức độ chức năng tương tự bằng cơ sở dữ liệu quan hệ.
- Cực kỳ thiết thực: Cơ sở dữ liệu NoSQL cung cấp các API và kiểu dữ liệu cực kỳ thiết thực được xây dựng riêng cho từng mô hình dữ liệu tương ứng

3) Các loại cơ sở dữ liệu NoSQL

Cơ sở dữ liệu NoSQL được phân loại thành bốn loại: Key-value pair, Column-oriented, Graph-based và Document-oriented. Mỗi loại đều có những thuộc tính và hạn chế riêng. Không có cơ sở dữ liệu nào được cho là tốt hơn để giải quyết tất cả các vấn đề. Người sử dụng nên chọn cơ sở dữ liệu dựa trên nhu cầu ứng dụng của mình

a) Key Value Pair Based NoSQL database

Với **Key Value Pair Based**, Dữ liệu được lưu trữ trong các cặp khóa / giá trị (Key/Value Pair). Nó được thiết kế theo cách để xử lý nhiều dữ liệu và tải nặng. Cơ sở dữ liệu lưu trữ cặp khóa-giá trị lưu trữ dữ liệu dưới dạng bảng băm trong đó mỗi khóa là duy nhất và giá trị có thể là JSON, BLOB (Binary Large Objects), chuỗi, v.v.



Hình 2: *Ví dụ về Key Value NoSQL database*

Giới hạn của Key Value NoSQL database:

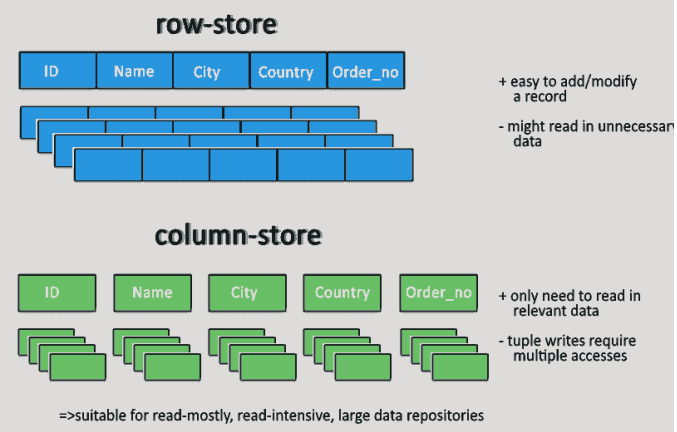
- Không có mối quan hệ giữa Multiple-Data.
- Multi-operation Transactions: Nếu bạn đang lưu trữ nhiều khóa và không thể lưu một trong các khóa, bạn không thể quay lại phần còn lại của các thao tác.
- Query Data by 'value': Tìm kiếm 'khóa' dựa trên một số thông tin được tìm thấy trong phần 'giá trị' của các cặp khóa-giá trị.
- Operation by groups: Vì các hoạt động được giới hạn trong một khóa tại một thời điểm, không có cách nào để chạy nhiều khóa đồng thời.

Key Value NoSQL database tiêu biểu:

DynamoDB
Riak
Berkeley DB
Redis

b) Column-based NoSQL database

Với **Column based database**, dữ liệu được lưu trữ trong database dưới dạng các cột. Mỗi cột được xử lý riêng biệt. Giá trị của cơ sở dữ liệu cột đơn được lưu trữ liền kề



Hình 3: *Ví dụ về Column based database*

Chúng mang lại hiệu suất cao cho các truy vấn tổng hợp như SUM, COUNT, AVG, MIN, v.v. vì dữ liệu có sẵn trong một cột. Cơ sở dữ liệu NoSQL dựa trên cột được sử dụng rộng rãi để quản lý data warehouses, business intelligence, CRM, Library card catalogs…

Giới hạn của Column based NoSQL database:

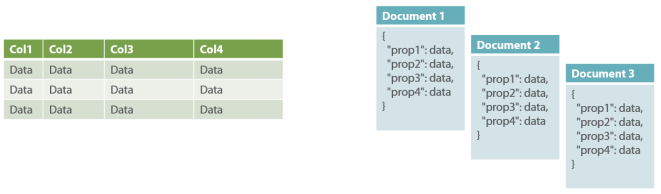
- Load dữ liệu theo kiểu incremental: Cần nhiều thời gian cho tác vụ ghi hơn tác vụ đọc. Phương thức Online Transaction Processing (OLTP) tức Xử lý giao dịch trực tuyến được sử dụng.
- Cần nhiều thời gian để đọc dữ liệu

CSDL Column Based NoSQL tiêu biểu:

Hbase
Cassandra
Hbase
Hypertable

c) Document-Oriented NoSQL

NoSQL Document Database lưu trữ và truy xuất dữ liệu dưới dạng một cặp giá trị khóa (key value pair)nhưng phần giá trị được lưu trữ dưới dạng tài liệu. Tài liệu được lưu trữ ở định dạng JSON hoặc XML. Giá trị được hiểu bởi Online Transaction Processing DB và có thể được truy vấn



Hình 4: *Ví dụ về NoSQL Document Database*

Trong sơ đồ bên trái bạn có thể thấy các hàng và cột, và ở bên phải, có một cơ sở dữ liệu tài liệu có cấu trúc tương tự như JSON. Đối với cơ sở dữ liệu quan hệ, bạn phải biết bạn có những cột nào, v.v. Tuy nhiên, đối với NoSQL document database, bạn có kho dữ liệu như đối tượng JSON. Bạn không cần phải xác định cái nào làm cho nó linh hoạt. Loại document này chủ yếu được sử dụng cho các hệ thống CMS, nền tảng blog, phân tích thời gian thực và các ứng dụng thương mại điện tử. Document database không nên sử dụng cho các giao dịch phức tạp yêu cầu nhiều hoạt động hoặc truy vấn dựa trên các cấu trúc tổng hợp khác nhau.

Giới hạn của NoSQL document database:

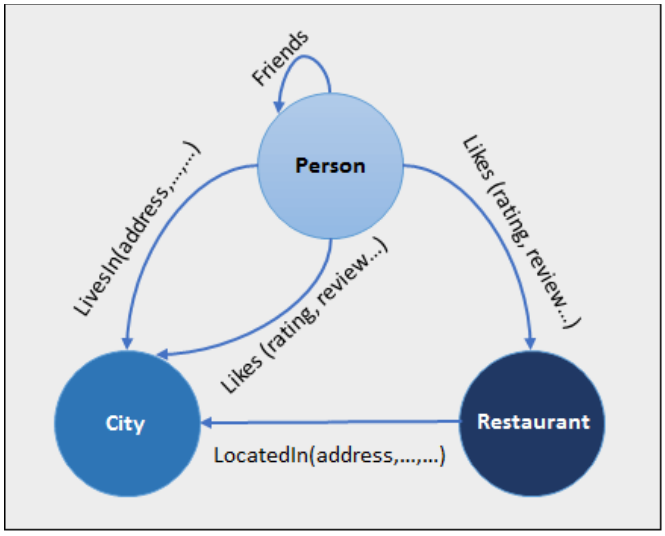
- Thông tin cơ sở trùng lặp trên nhiều tài liệu
- Thiết kế phức tạp dẫn đến không nhất quán.

Các hệ thống DBMS Document database NoSQL tiêu biểu:

Amazon SimpleDB
CouchDB
MongoDB
RavenDB

d) Graph-Based NoSQL database

Cơ sở dữ liệu kiểu đồ thị (Graph Based) lưu trữ các thực thể cũng như các mối quan hệ giữa các thực thể đó. Thực thể được lưu trữ dưới dạng một node với mối quan hệ là các cạnh. Một cạnh cho biết mối quan hệ giữa các node. Mỗi node và cạnh có một mã định danh duy nhất



Hình 5: *Ví dụ về NoSQL Graph Based Database*

So với cơ sở dữ liệu quan hệ trong đó các bảng được kết nối với nhau một cách lỏng lẻo, cơ sở dữ liệu Đồ thị có bản chất là đa quan hệ. Mối quan hệ truyền tải nhanh chóng vì chúng đã được ghi lại vào DB và không cần phải tính toán chúng. Cơ sở dữ liệu đồ thị chủ yếu được sử dụng cho mạng xã hội, hậu cần, dữ liệu không gian.

Giới hạn của NoSQL Graph database:

- Thiếu tính đồng thời hiệu suất cao (high performance concurrency): Trong nhiều trường hợp, graph database cung cấp các kiểu đọc và kiểu ghi đơn, điều này cản trở sự đồng

thời và hiệu suất, do đó phần nào hạn chế tính song song phân luồng (threaded parallelism).
- Thiếu ngôn ngữ chuẩn: Việc thiếu sự thiết lập và một ngôn ngữ khai báo chuẩn là một vấn của NoSQL graph database.
- Thiếu tính song song (parallelism): việc phân vùng một biểu đồ là một vấn đề. Hầu hết các graph database không cung cấp các truy vấn song song trên các biểu đồ lớn.

Các graph-based databases tiêu biểu:

Infinite Graph
OrientDB
FlockDB
Neo4J
Giraph

B. Giới thiệu về hệ quản trị NoSQL RavenDB

1) Khái niệm

RavenDB là một cơ sở dữ liệu hướng tài liệu ACID hoàn toàn mã nguồn mở được viết bằng C# , được phát triển bởi Hibernating Rhinos Ltd. Nó đa nền tảng, được hỗ trợ trên Windows , Linux và Mac OS. RavenDB lưu trữ dữ liệu dưới dạng tài liệu JSON và có thể được triển khai trong các cụm phân tán với bản sao chính-chính.

RavenDB là một cơ sở dữ liệu mã nguồn mở có hỗ trợ transactional được viết trên nền tảng .NET. RavenDB đưa ra mô hình dữ liệu linh hoạt nhằm đáp ứng yêu cầu của các hệ thống thế giới thực.

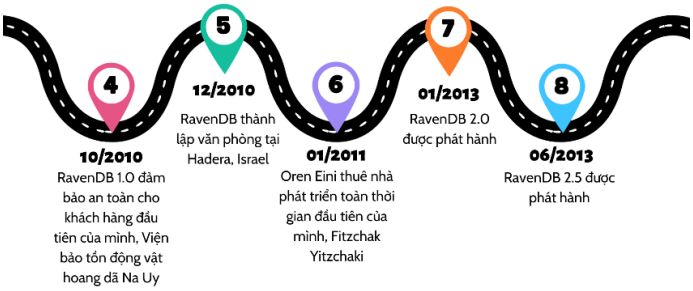
Nó cho phép xây dựng những ứng dụng có hiệu suất cao, độ trễ thấp một cách nhanh chóng và hiệu quả. Nó được thiết kế như một cơ sở dữ liệu dễ sử dụng tất cả trong một, giúp giảm thiểu nhu cầu bổ sung, công cụ của bên thứ ba hoặc hỗ trợ để tăng năng suất của nhà phát triển và đưa dự án vào sản xuất nhanh.

2) Lịch sử ra đời và nguồn gốc

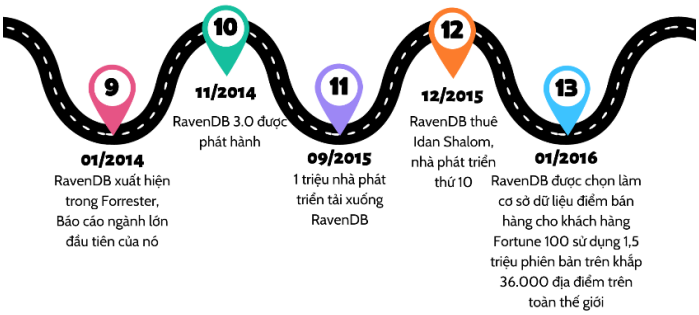
a) Lịch sử phát triển

RavenDB được phát triển vào năm 2008 bởi Oren Eini và công ty Hibernating Rhinos Ltd. Cụ thể hơn như sau:

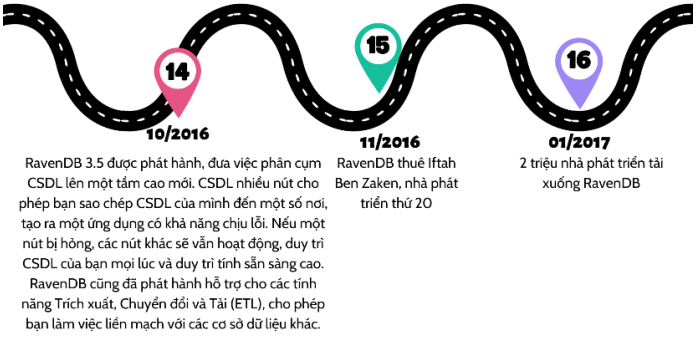
Hình 6: *Giai đoạn 1 - 3 của lịch sử phát triển RavenDB*



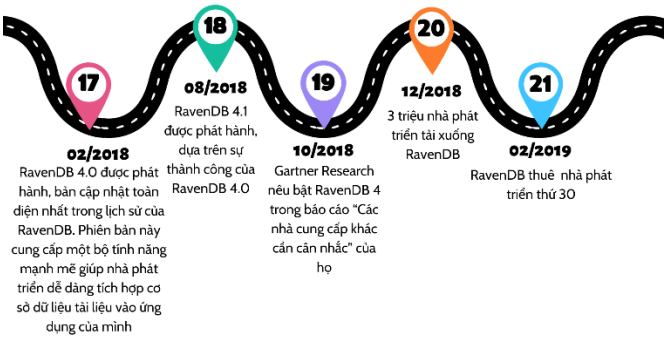
Hình 7: *Giai đoạn 4 - 8 của lịch sử phát triển RavenDB*



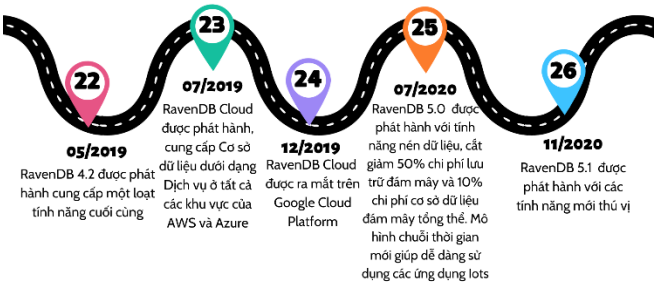
Hình 8: *Giai đoạn 9 - 13 của lịch sử phát triển RavenDB*



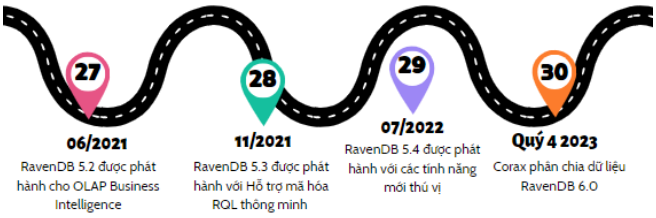
Hình 9: *Giai đoạn 14 - 16 của lịch sử phát triển RavenDB*



Hình 10: *Giai đoạn 17 - 21 của lịch sử phát triển RavenDB*



Hình 11: *Giai đoạn 22 - 26 của lịch sử phát triển RavenDB*



Hình 12: *Giai đoạn 27 - 30 của lịch sử phát triển RavenDB*

b) Nguồn gốc, ý nghĩa tên RavenDB

Tên "RavenDB" được lựa chọn với ý tưởng từ con quạ (raven) trong thiên nhiên. Con quạ là một loài chim thông minh, linh hoạt và có khả năng nhận biết và nhớ các môi trường sống của nó. Tên "RavenDB" được chọn để tạo ra một ấn tượng về tính thông minh, khả năng thích ứng và khả năng lưu trữ và truy xuất dữ liệu linh hoạt.

Con quạ cũng được coi là một biểu tượng của sự tinh ranh và sáng tạo. Tên "RavenDB" thể hiện mong muốn của nhà phát triển là tạo ra một hệ quản trị cơ sở dữ liệu thông minh, sáng tạo và linh hoạt để đáp ứng các yêu cầu phức tạp của các ứng dụng phần mềm hiện đại.

Hình 13: *Logo 1 của RavenDB*

Hình 14: *Logo 2 của RavenDB*

Logo của RavenDB chứa hình ảnh một con quạ, và điều này cũng phản ánh ý nghĩa của tên gọi này.

Con quạ trong logo thể hiện tính thông minh và sự sáng tạo. Con quạ được biết đến là một loài chim thông minh, có khả năng giải quyết các tình huống phức tạp và tìm ra cách thức để đạt được mục tiêu của mình. Nó cũng được cho là có khả năng nhận biết và nhớ các môi trường sống của mình, đại diện cho khả năng lưu trữ và truy xuất dữ liệu linh hoạt của RavenDB.

Logo cũng có thể tượng trưng cho tính linh hoạt và khả năng thích ứng của RavenDB. Con quạ có khả năng bay cao và thích ứng với nhiều môi trường sống khác nhau. Tương tự, RavenDB cũng được thiết kế để cung cấp tính linh hoạt và khả năng thích ứng cho các ứng dụng và môi trường phức tạp.

Như vậy, Logo của RavenDB không chỉ là một biểu tượng đẹp mắt, mà còn chứa đựng ý nghĩa văn vở về tính thông minh, sáng tạo, linh hoạt và khả năng lưu trữ và truy xuất dữ liệu của hệ quản trị cơ sở dữ liệu này

3) Ưu điểm và nhược điểm của RavenDB

a) Ưu điểm

RavenDB rất dễ cài đặt và bảo mật, dễ sử dụng và đưa vào sản xuất nhanh chóng. Ngôn ngữ truy vấn dễ dàng vì RQL sử dụng cú pháp SQL đến 80%.

RavenDB là một CSDL đa mô hình. Document, key/value, Counter, Graph API và Time Series cho phép sử dụng nhiều phiên bản cho các chức năng khác nhau mà không phải mua cơ sở dữ liệu khác nhau, hữu dụng cho microservice.

RavenDB dễ dàng mở rộng quy mô. Cơ sở dữ liệu phân tán cho phép thiết lập các nút (nodes) ngay lập tức. Ngoài ra, nó sử dụng phần cứng tối đa, RavenDB hoạt động trên Raspberry Pi và chip ARM vì nó tận dụng tối đa phần cứng mà nó chạy.

b) Nhược điểm

RavenDB chưa có cộng đồng đủ mạnh để hỗ trợ việc giải đáp các thắc mắc hoặc các vấn đề hay gặp phải khi sử dụng nó.

Tài liệu vẫn chưa có nhiều về các ví dụ thực hành, hoặc các trường hợp phổ biến khi sử dụng RavenDB. Ngoài ra, vẫn còn thiếu các video hướng dẫn giải quyết các vấn đề phổ biến.

Không thể kiểm tra sharding, replication, hoặc truy cập được xác thực nếu không mua giấy phép.

Giấy phép là điều bắt buộc đối với bất cứ điều gì khác ngoài sự phát triển (ngay cả UAT cũng yêu cầu giấy phép).

4) Ngôn ngữ thao tác với dữ liệu

a) Tổng quan

Câu truy vấn của RavenDB sử dụng một ngôn ngữ có cấu trúc giống SQL là RQL

RQL được thiết kế để thể hiện ra bên ngoài các thao tác xử lí truy vấn trong RavenDB một cách dễ hiểu, dễ sử dụng và không gây quá tải cho người sử dụng

Bộ tối ưu hóa câu truy vấn: Ngay khi truy vấn đến tập dữ liệu của RavenDB, tập dữ liệu gọi đến bộ tối ưu câu truy vấn để phân tích câu truy vấn và quyết định các index nào sẽ được sử dụng để trích xuất dữ liệu.

Câu truy vấn động và theo index (Dynamic and Indexed Queries)

RavenDB cung cấp 2 kiểu truy vấn:

- Câu truy vấn động (dynamic query) cho phép bộ tối ưu câu truy vấn tự do chọn index sẽ sử dụng
- Câu truy vấn theo index xác định index cụ thể mà câu truy vấn sẽ dùng

b) Sử dụng câu truy vấn theo các câu index

Trong các cơ sở dữ liệu khác, bộ tối ưu hóa câu truy vấn khi không tìm được một index thích hợp sẽ tiến hành truy vấn bằng cách quét toàn bộ dữ liệu.

RavenDB không hỗ trợ việc quét toàn bộ, Nếu một index không được tìm thấy trong một câu truy vấn thì bộ tối ưu câu truy vấn sẽ tiến hành tạo index mới cho câu truy vấn đó.

Câu truy vấn trong RavenDB sẽ sử dụng index mà đã được tạo hay tìm thấy để tối ưu hóa thời gian phản hồi và trả về kết quả với cùng tốc độ mà không phụ thuộc vào kích thước của bộ dữ liệu.

c) Danh sách các từ khóa cơ bản cho RQL

Declare: Từ khóa declare dùng để tạo các hàm javascript mà có thể tái sử dụng trong select (sau khi phép chiếu thực hiện xong)

From: từ khóa from dùng để xác định nguồn dữ liệu mà câu truy vấn sẽ thực thi. Có 2 tùy chọn cho từ khóa from

- From <collection> : dùng cho nguồn dữ liệu là một collection cụ thể hoặc dùng cho truy vấn động dành cho các index được tạo tự động
- From Index <index> :dùng cho nguồn dữ liệu là một index cụ thể

Group by: Từ khóa group by dùng để thực hiện câu truy vấn có gom nhóm

Where: Từ khóa where dùng để thực hiện lọc kết quả truy vấn theo điều kiện lọc. Các điều kiện lọc sử dụng trong where

- Các toán tử >=, <=, <>, !=, <, >, =, ==
- Toán tử between <a> and <b>: lọc giá trị trong khoản <a> và <b>
- Toán tử in : lọc giá trị nằm trong tập hợp nhất định
- Toán tử all in: Kiểm tra nếu toán tử tất cả giá trị đều xuất hiện trong mảng, toán tử này chỉ phù hợp với trường dữ liệu trong mảng
- Toán tử logic: and, or, not
- Một số hàm sử dụng để lọc khác
  - id()
  - search()
  - cmpxchg()
  - boost()
  - regex()
  - startsWith()
  - endsWith()
  - lucene()
  - exists()
  - exact()
  - intersect()
  - spatial.within()
  - spatial.contains()
  - spatial.disjoint()
  - spatial.intersects()
  - moreLikeThis()

Order by: Để sắp xếp thứ tự của các giá trị ta dùng order by. Một số kiểu sắp xếp với order by:

- ASC | ASCENDING
- DESC | DESCENDING
- AS
- string
- long
- double
- alphaNumeric
- random()
- score()
- spatial.distance()

Load: Khi muốn sử dụng data từ bên ngoài ta dùng load

Select: Phép chiếu được thể hiện thông qua select. Một số hàm đi chung với select

- DISTINCT
- key()
- sum()
- count()
- facet()

Update: để vá/cập nhật lại dữ liệu ở phía server , ta dùng lệnh update để tiến hành cập nhật những tài liệu phù hợp với tiêu chí truy vấn.

Include: dùng để hỗ trợ:

- Bao gồm thêm các tài liệu hoặc đếm số phản hồi của câu truy vấn
- Highlighting kết quả
- Thời gian truy vấn
- Giải thích

d) Giới thiệu 1 số đặc điểm khi truy vấn bằng RavenDB

Query-flow: Mỗi câu truy vấn phải phù hợp với một chỉ mục để trả về kết quả và đây là toàn bộ flow trong việc truy vấn bằng RavenDB

From index | collection: Đầu tiên câu truy vấn sẽ tìm chỉ mục phù hợp với
nó nếu không có thì nó sẽ tạo một chỉ mục tự động

Where: Khi tìm được chỉ mục thì sẽ tìm các bản ghi phù hợp với câu truy vấn

Load: Khi có một phép chiếu nào đó thì sẽ ưu tiên thực hiện phép chiếu trước khi các câu truy vấn khác

Select: Nếu truy vấn không phải là phép chiếu thì load tài liệu từ bộ nhớ ra; Nếu truy vấn là phép chiếu thì kết quả chưa được lọc ra sẽ xử lí bởi phép chiếu đó

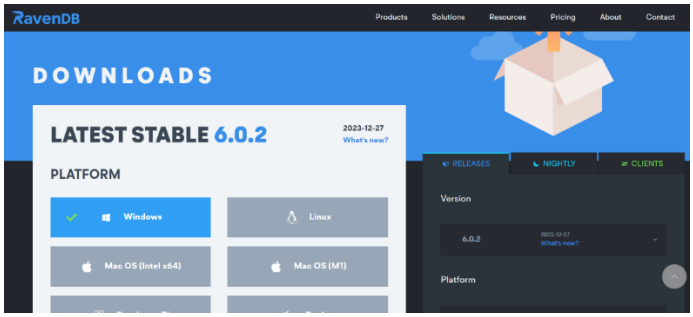
Include: Nếu include được xác định thì nó sẽ trích xuất ID của các tài liệu có tiềm năng đưa vào kết quả

Return results

C. Tạo và thêm mới dữ liệu

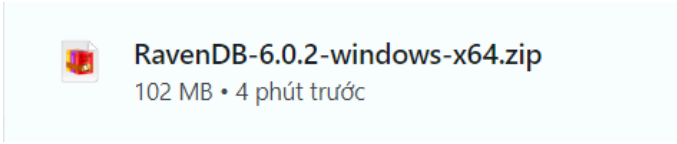
1) Cài đặt RavenDB

Bước 1: Người dùng tiến hành tải RavenDB về máy theo đường link:



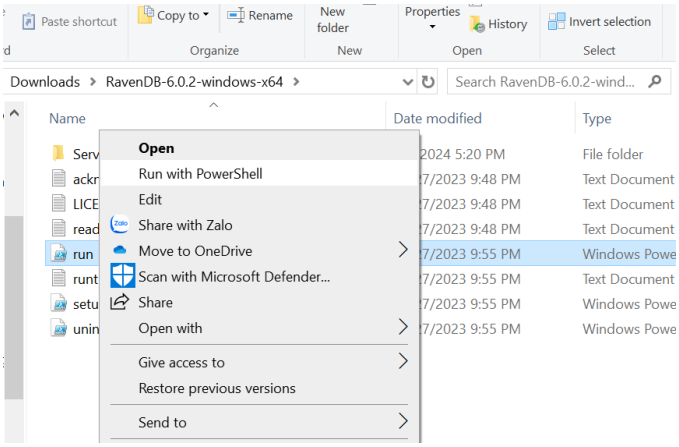
Hình 15: *Bước 1 – Cài đặt RavenDB*

Tải RavenDB thành công:



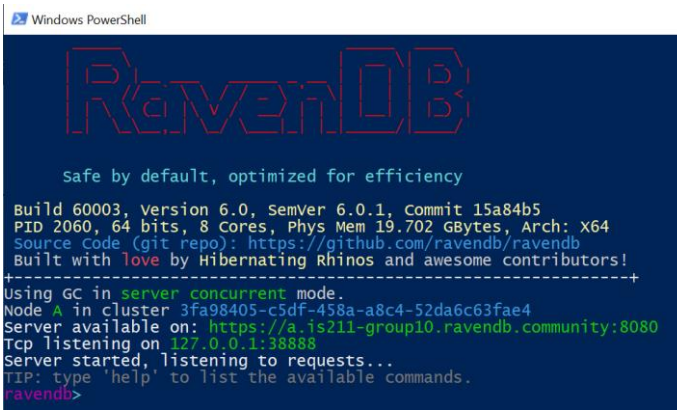
Hình 16: *RavenDB tải thành công*

Bước 2: Nhấn chuột phải "Run" rồi nhấn "Run with PowerShell" để chạy RavenDB



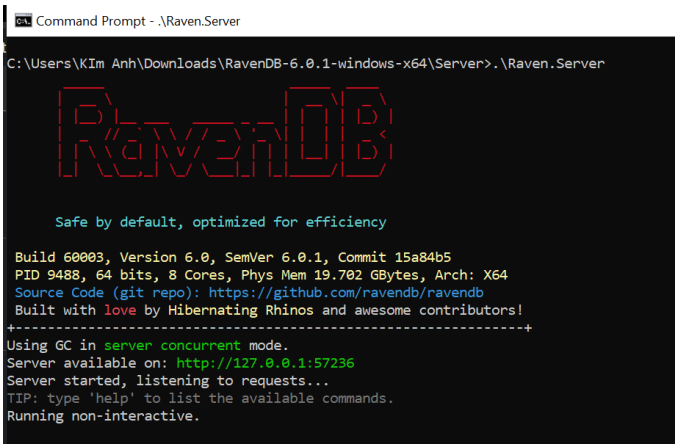
Hình 17: *Bước 2 – Cài đặt RavenDB*

Màn hình sau khi "Run"



Hình 18: *Giao diện sau khi nhấn "Run"*

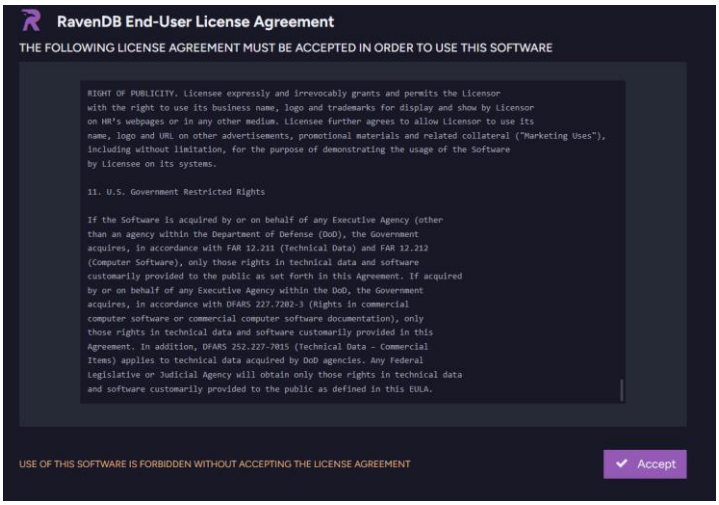
Bước 3: Sau đó mở cmd, gõ ".\Raven.Server" để chạy như hình sau



Hình 19: *Bước 3 – Cài đặt RavenDB*

Bước 4: Sau đó copy dòng "http://127.0.0.1:57236" bỏ vào trình duyệt và chạy

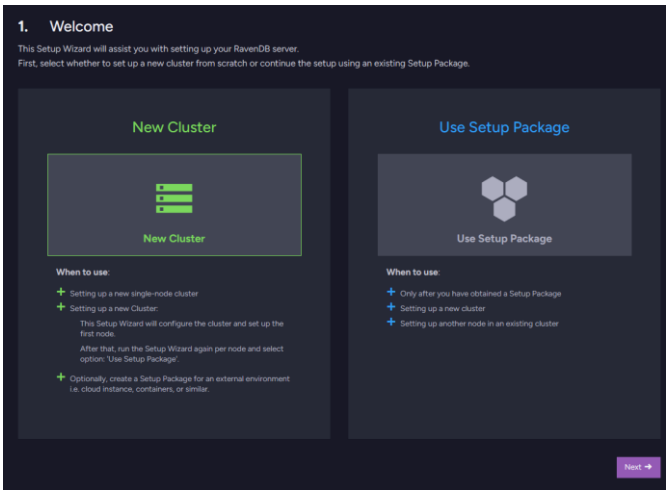
Bước 5: Thực hiện xong Bước 4, sẽ hiện ra màn hình sau. Kéo xuống dưới đọc hết chính sách và nhấn "Accept"



Hình 20: *Bước 5 – Cài đặt RavenDB*

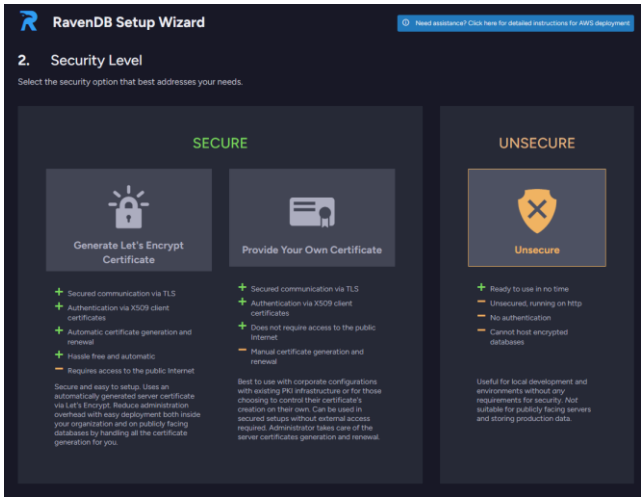
- NODE A:

Bước 6.1: Sau đó, chọn "New Cluster" và "Next"



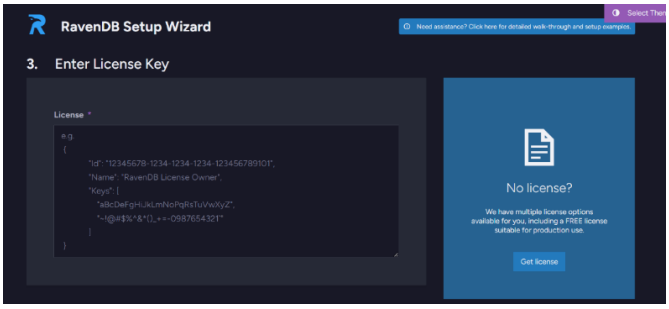
Hình 21: *Bước 6.1 – Cài đặt RavenDB*

Bước 7.1: Tiếp tục chọn " UNSECURE" và nhấn "Next"



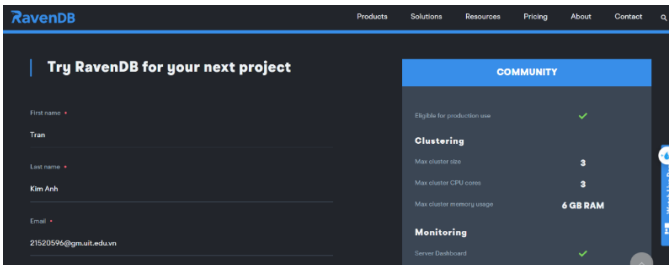
Hình 22: *Bước 7.1 – Cài đặt RavenDB*

Bước 8.1: Nhấn "Get license"



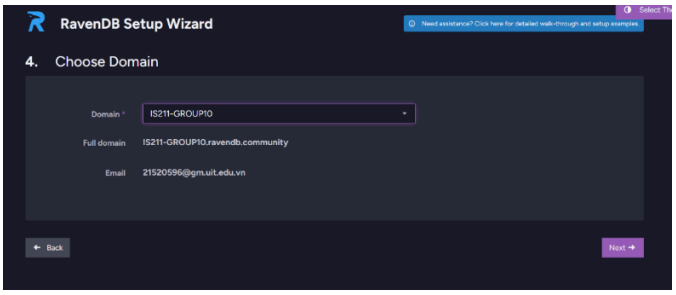
Hình 23: *Bước 8.1 – Cài đặt RavenDB*

Bước 9.1: Điền thông tin của người dùng



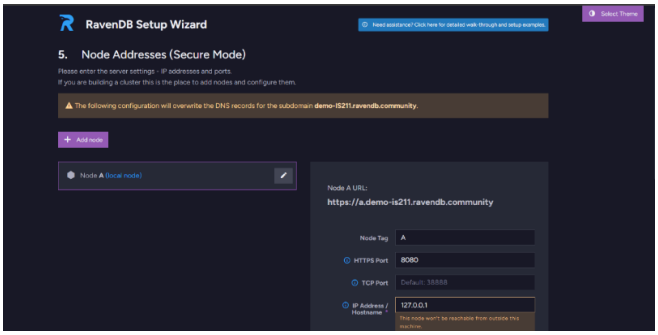
Hình 24: *Bước 9.1 – Cài đặt RavenDB*

Bước 10.1: Điền thông tin



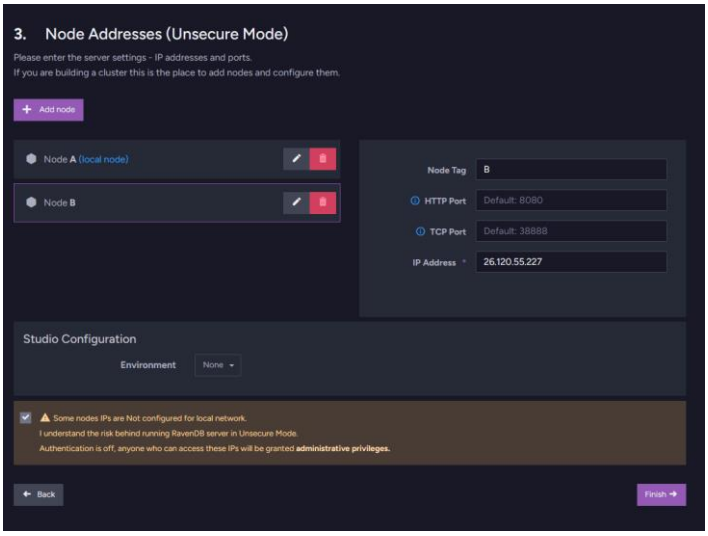
Hình 25: *Bước 10.1 – Cài đặt RavenDB*

Bước 11.1: Thêm địa chỉ IP của Máy 1 vào Node A



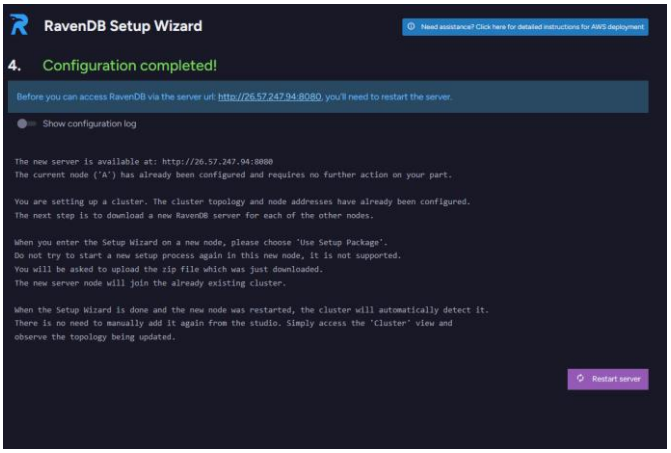
Hình 26: *Bước 11.1 – Cài đặt RavenDB*

Bước 12.1: Chọn Add Node B và thêm IPv4 của Máy B vào



Hình 27: *Bước 12.1 – Cài đặt RavenDB*

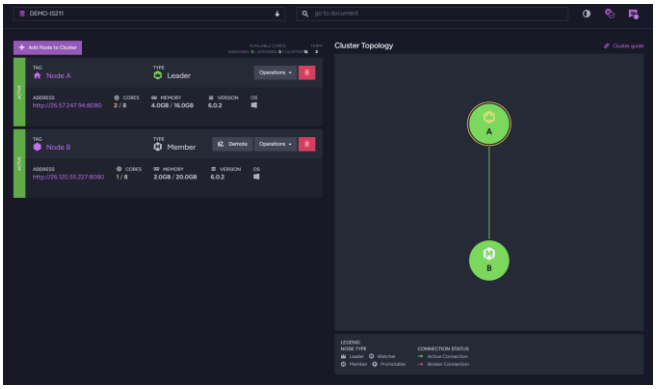
Bước 13.1: Chọn Finish, Set up thành công Node A và Node B chung 1 group. Trình duyệt tải về file zip để Node B có thể chọn Set up Package



Hình 28: *Bước 13.1 – Cài đặt RavenDB*

Bước 14.1: Restart Server. Hoàn tất set up

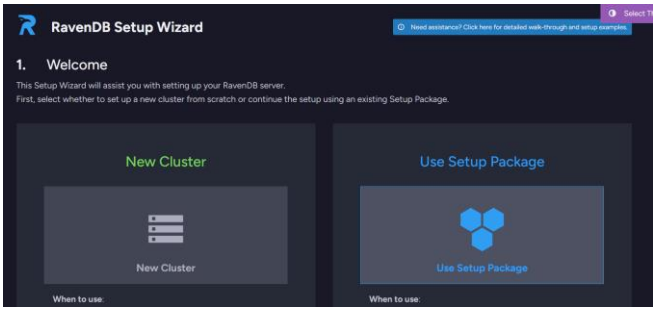
Bước 15.1: Sau khi set up Node B thành công sẽ thấy Node A và Node B ở Manage-> Cluster



Hình 29: *Bước 15.1 - Lấy note A và B*

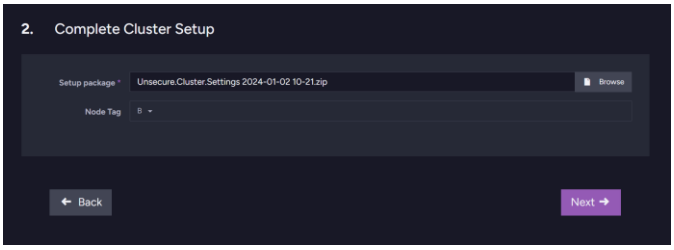
- NODE B:

Bước 6.2: Máy B sẽ sử dụng Setup Package mà máy A đã tạo. Chọn Use Setup Package, sau đó chọn Next



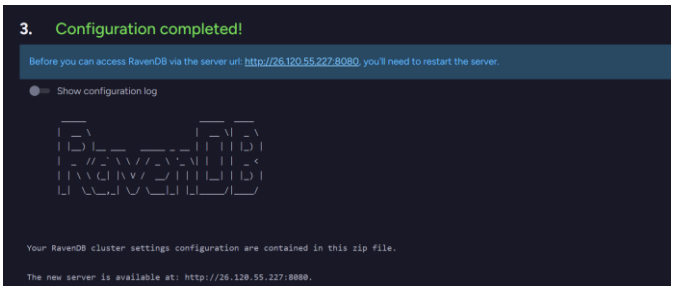
Hình 30: *Bước 6.2 – Cài đặt RavenDB*

Bước 7.2: Chọn Browse -> Chọn file zip Unsecure.Cluster.Settings. Sau đó chọn Node Tag có IP là Node B. Sau đó chọn Next



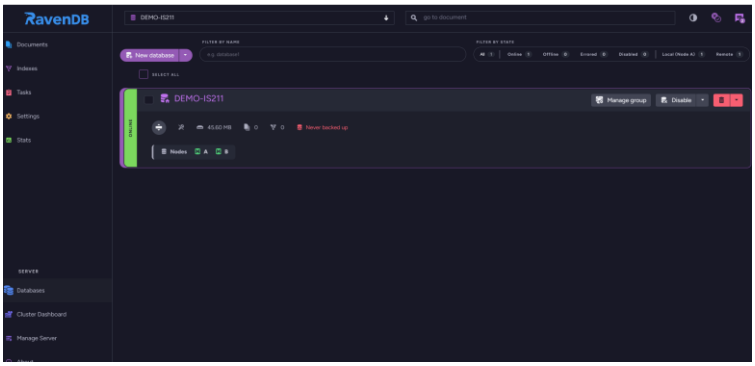
Hình 31: *Bước 7.2 – Cài đặt RavenDB*

Bước 8.2: Lúc này ta đã config xong cho server, chọn Restart Server



Hình 32: *Bước 8.2 – Cài đặt RavenDB*

Sau khi Restart Server:

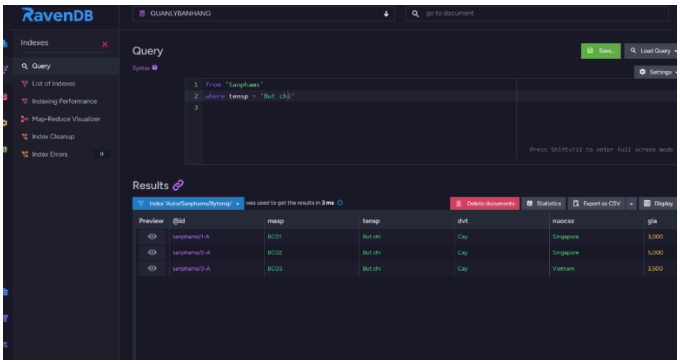


Hình 33: *Giao diện sau khi Restart Server*

IV. Cơ chế truy vấn và thao tác dữ liệu phân tán

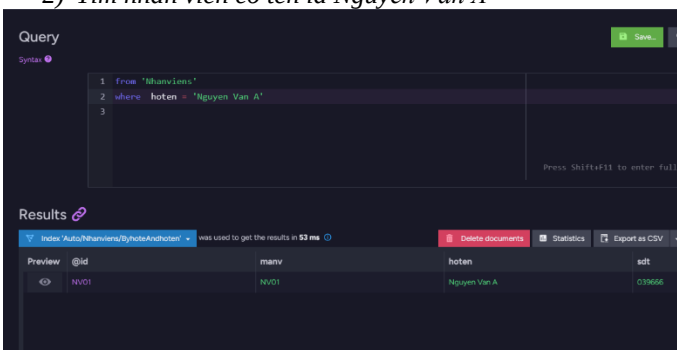
A. Truy vấn dữ liệu

1) Tìm sản phẩm tên là But chi



Hình 34: *Sản phẩm "But chi" được in ra*

2) Tìm nhân viên có tên là Nguyễn Văn A



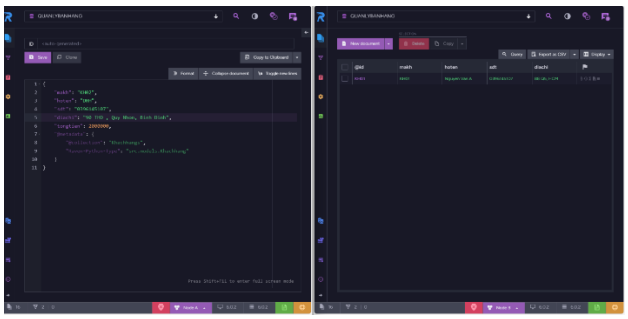
Hình 35: *Khách hàng tên "Nguyen Van A" được in ra*

3) Tìm sản phẩm trong tên có chữ "GIAY"

Hình 36: *Sản phẩm "GIAY" được in ra*

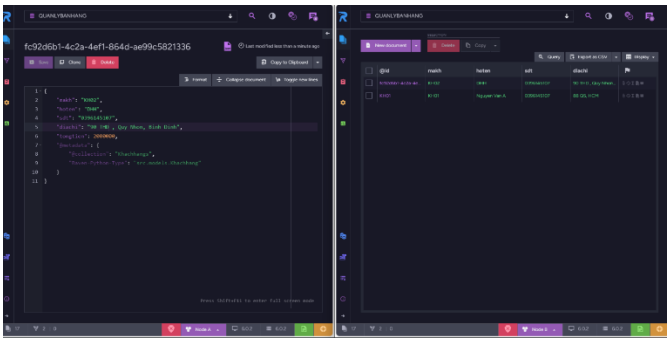
B. Thêm xóa sửa trên RavenDB

Tại Node A: Thêm một khách hàng mới với MAKH = "KH02"



Hình 37: *Thêm khách hàng mới tại Node A*

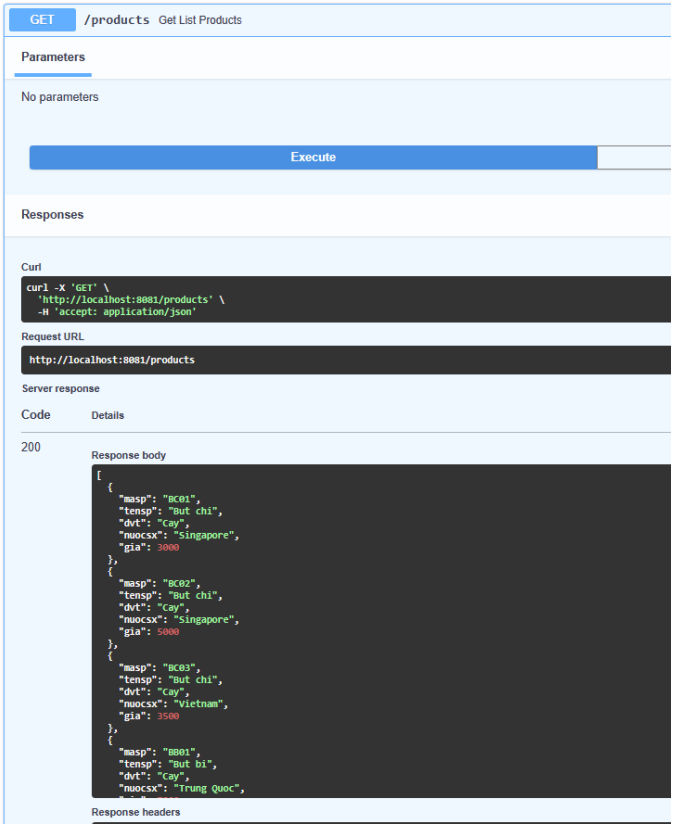
Tại Node B: Sau khi Node A tạo khách hàng mã "KH02" thành công thì lúc này Node B cũng cập nhật thành công mã khách hàng "KH02"



Hình 38: *Tại node B sau khi node A tạo thành công*

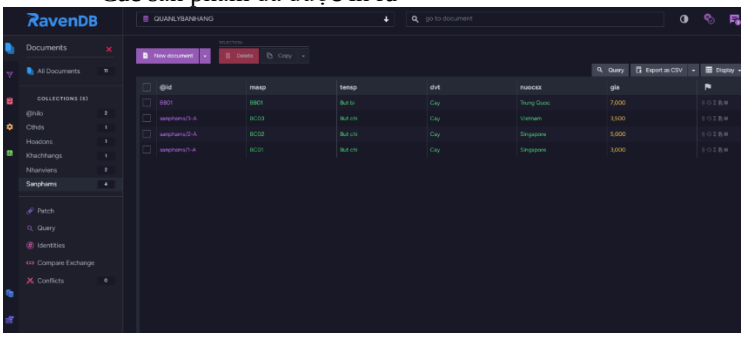
C. Lấy dữ liệu qua lại giữa 2 máy NoSQL

- Đầu tiên, ta sẽ lấy toàn bộ danh sách sản phẩm But chi



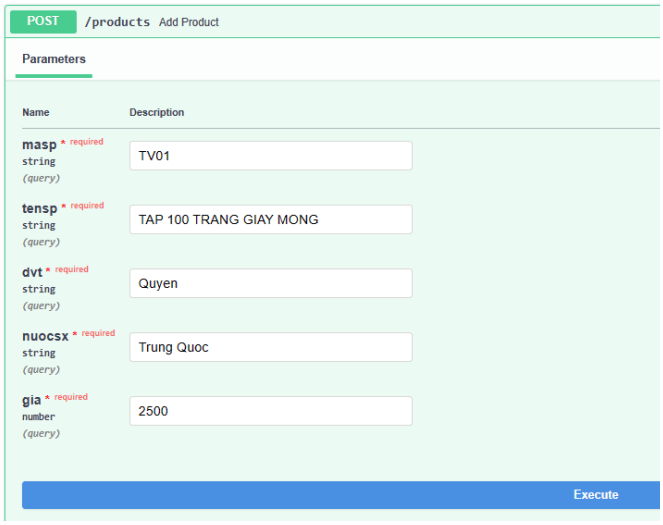
Hình 39: *Code danh sách các sản phẩm But chi*

Các sản phẩm đã được in ra



Hình 40**:** Các sản phẩm đã được in ra

- Ta sẽ thêm mới sản phẩm “TAP 100 TRANG GIAY MONG”

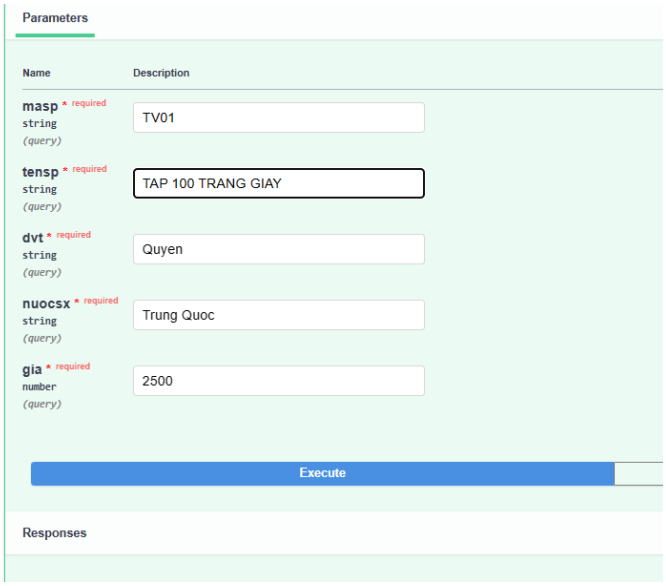


Hình 41: *Thêm sản phẩm vào*

Sản phẩm đã được thêm vào

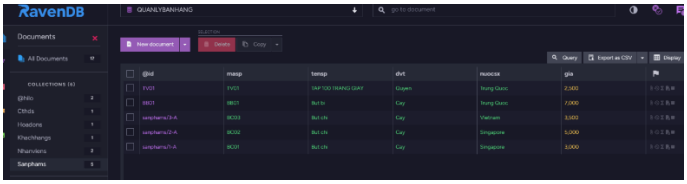
Hình 42: *Danh sách sản phẩm sau khi đã thêm*

- Ta muốn sửa dữ liệu ở trong bảng SANPHAM



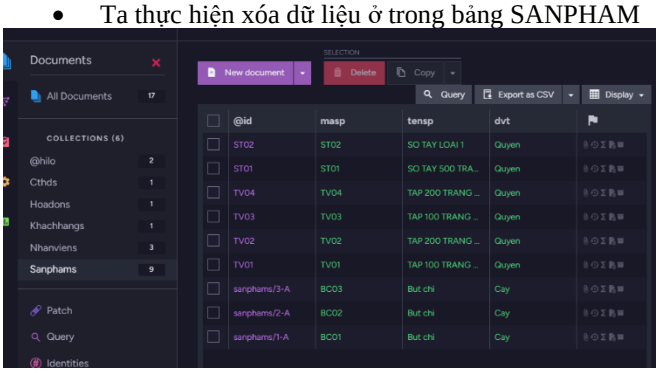
Hình 43: *Sửa thông tin sản phẩm*

Sản phẩm sau khi được sửa



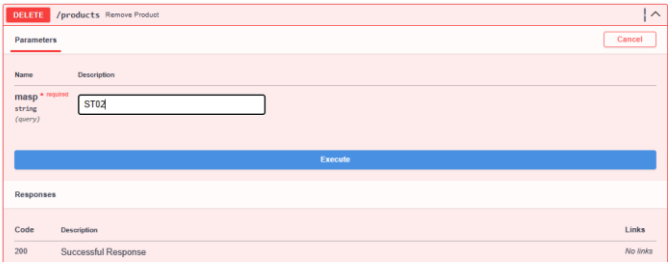
Hình 44: *Danh sách sản phẩm sau khi đã sửa*

- Ta thực hiện xóa dữ liệu ở trong bảng SANPHAM



Hình 45: *Toàn bộ danh sách sản phẩm*

Chọn MASP xóa là “ST02”



Hình 46: *Mã sản phẩm sẽ xóa*

- Code để thực hiện việc xóa sản phẩm

Hình 47: *Code để xóa sản phẩm*

Sản phẩm đã được xóa

Hình 48: *Sản phẩm đã được xóa khỏi danh sách*

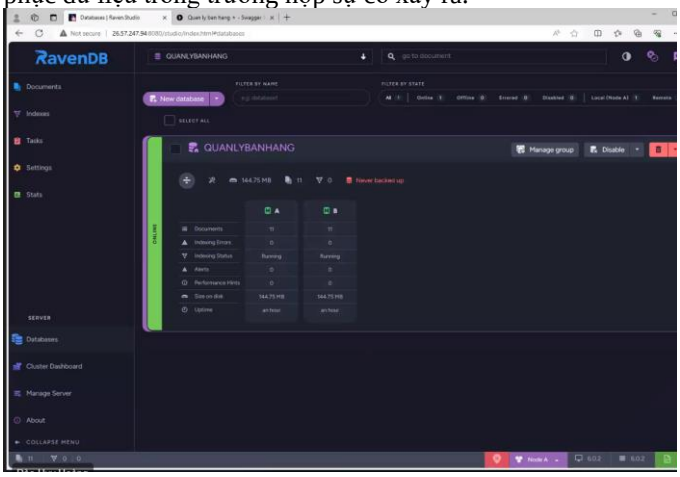
D. Cơ chế nhân bản trong phân tán RavenDB

Cơ chế nhân bản phân tán trong RavenDB dựa trên mô hình cụm (cluster) với nhiều nút (node) làm việc cùng nhau. Mỗi nút trong cụm đều lưu trữ một bản sao của dữ liệu, đồng bộ hóa và cập nhật dữ liệu với nhau để đảm bảo tính nhất quán.

Khi một yêu cầu ghi dữ liệu được gửi đến hệ thống, nó sẽ được gửi đến một nút chủ (primary node), được chọn dựa trên một giải thuật phân phối (distribution algorithm). Nút chủ sẽ tiếp nhận yêu cầu và cập nhật dữ liệu trong bản sao của nó. Sau đó, nút chủ sẽ phân phối các thay đổi đến các nút khác trong cụm thông qua cơ chế nhân bản (replication mechanism).

Cơ chế nhân bản trong RavenDB sử dụng giao thức đồng bộ hóa như giao thức REPL (Replication Protocol) hoặc giao thức CDC (Change Data Capture) để đảm bảo rằng các thay đổi dữ liệu được sao chép đến các nút khác một cách đáng tin cậy. Khi một nút chủ gặp sự cố, một nút phụ (secondary node) có thể được chọn làm nút chủ mới và tiếp tục phục vụ yêu cầu ghi dữ liệu.

Cơ chế nhân bản phân tán trong RavenDB cũng hỗ trợ khả năng sao lưu và khôi phục dữ liệu. Các bản sao dữ liệu được tạo ra có thể được sao lưu định kỳ và sử dụng để khôi phục dữ liệu trong trường hợp sự cố xảy ra.



Hình 49: *Nhân bản trong RavenDB*

V. LỜI CẢM ƠN

Đầu tiên, nhóm chúng em xin gửi lời cảm ơn và lòng biết ơn sâu sắc nhất tới giảng viên Thái Bảo Trân và giảng viên Nguyễn Minh Nhựt – những người đã giảng dạy và chia sẻ rất nhiều kiến thức cũng như các ví dụ thực tiễn trong các bài giảng. Thầy đã hướng dẫn cho chúng em làm bài tập, sửa chữa và đóng góp nhiều ý kiến quý báu giúp chúng em hoàn thành tốt báo cáo môn học của mình.

Bộ môn Cơ sở dữ liệu phân tán là môn học thú vị, vô cùng bổ ích và có tính thực tế cao. Tuy nhiên, do vốn kiến thức chuyên môn còn nhiều hạn chế và khả năng tiếp thu thực tế còn nhiều bỡ ngỡ. Mặc dù chúng em đã cố gắng hết sức nhưng chắc chắn bài báo cáo khó có thể tránh khỏi những thiếu sót và nhiều chỗ còn chưa chính xác, chúng em rất mong nhận được sự góp ý, chỉ bảo thêm của Thầy nhằm hoàn thiện những kiến thức của mình để nhóm chúng em có thể dùng làm hành trang thực hiện tiếp các đề tài khác trong tương lai cũng như là trong học tập và làm việc sau này.

Một lần nữa, nhóm xin gửi đến thầy, bạn bè lời cảm ơn đặc biệt chân thành và tốt đẹp nhất.

VI. TÀI LIỆU THAM KHẢO

[1] Các file báo cáo BTL2 của anh chị khóa trước do giảng viên Nguyễn Minh Nhựt cung cấp

[2] Tài liệu thực hành: “Database Radmin VPN Real Enviroment” do giảng viên Nguyễn Minh Nhựt cung cấp

[3] “ https://ravendb.net/ ”: tìm hiểu về RavenDB

[4] Link: “ https://db-engines.com/en/systems ” do giảng viên Nguyễn Minh Nhựt cung cấp