

ĐẠI HỌC SƯ PHẠM THÀNH PHỐ HỒ CHÍ MINH
Khoa Công Nghệ Thông Tin



ĐỒ ÁN MÔN CƠ SỞ DỮ LIỆU NÂNG CAO

ĐẠI HỌC SƯ PHẠM THÀNH PHỐ HỒ CHÍ MINH
Khoa Công Nghệ Thông Tin
Môn (Cơ Sở Dữ Liệu Nâng Cao)



RAVENDB

Người hướng dẫn: Ths. Lương Trần Hy Hiến
Nhóm 85:

- 1. Võ Ngọc Quỳnh Mai – 42.01.104.242**
- 2. Trần Thị Như Ý – 42.01.104.199**
- 3. Nguyễn Ngọc Yến Nhi – 42.01.104.098**
- 4. Nguyễn Thị Như Quỳnh – 42.01.104.130**

MỤC LỤC

I.	Giới thiệu	1
II.	Ưu điểm và nhược điểm	3
	1. Ưu điểm	3
	2. Nhược điểm	5
III.	Cài đặt RavenDB	6
IV.	Truy vấn và xử lý dữ liệu trên RavenDB	12
V.	Cách thức truy vấn và xử lý dữ liệu của RavenDB trên C#	16
VI.	Tài liệu tham khảo	23

Giới thiệu cơ sở dữ liệu RavenDB

RavenDB là một cơ sở dữ liệu tài liệu nguồn mở mới cho .NET. Nếu bạn chưa bao giờ làm việc với một cơ sở dữ liệu tài liệu trước đây, cách đơn giản nhất để suy nghĩ về nó là tưởng tượng việc sắp xếp các đối tượng của bạn và lưu trữ chúng trên ổ cứng nơi ứng dụng đang ở đó. Nếu bạn đã lưu nó bằng cách sử dụng khóa hoặc bất kỳ phương pháp truy cứu phổ biến nào mà bạn có thể sử dụng, sẽ dễ dàng truy xuất toàn bộ đối tượng của bạn mà không cần phải ánh xạ tới và từ các cột và hàng trong cơ sở dữ liệu SQL. Đối phó với các cách khác để tìm kiếm nó, đồng thời, vv sẽ khó khăn hơn để giải quyết, do đó tạo ra các cơ sở dữ liệu tài liệu. Các tài liệu được lưu trữ không cần thiết phải là một đối tượng được tuần tự hóa (các tài liệu tùy ý có thể được lưu trữ độc lập với các đối tượng), nhưng đó có lẽ là cách phổ biến nhất mà nó có thể được sử dụng. Chỉ cần ghi nhớ rằng nó là một cách hoàn toàn khác nhau đối phó với lưu trữ dữ liệu, do đó bạn có thể không muốn luôn luôn tiếp cận nó như bạn sẽ là một cơ sở dữ liệu SQL. Một máy khách .NET được bao gồm để liên lạc với máy chủ nhưng biểu diễn bên dưới là HTTP / JSON - vì vậy bất kỳ máy khách nào có thể giao tiếp với máy chủ qua HTTP / JSON sẽ hoạt động.

Ví dụ:

```
using (var documentStore = new DocumentStore("localhost", 8080).Initialise())
using (var session = documentStore.OpenSession())
{
    session.Store(new Company { Name = "Company 1", Region = "Asia" });
    session.Store(new Company { Name = "Company 2", Region = "Europe" });
    session.SaveChanges();

    var allCompanies = session
        .Query<Company>()
        .WaitForNonStaleResults()
        .ToArray();

    foreach (var company in allCompanies)
        Console.WriteLine(company.Name);
}
```



```
C:\Windows\system32\cmd.exe
Company 1
Company 2
```

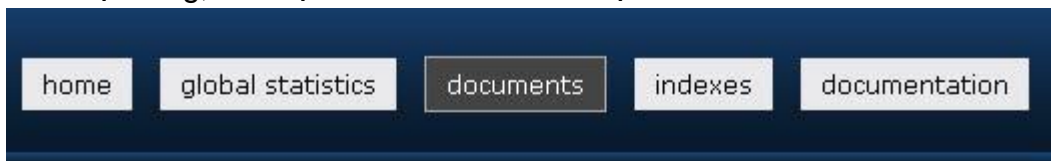
Đây là một ví dụ đơn giản về cách lưu trữ hai đối tượng POCO tùy ý trong RavenDB, truy vấn chúng và in ra một số thông tin trên màn hình. Trong ví dụ này, máy chủ RavenDB trên cùng một máy với máy khách nhưng điều đó không phải là trường hợp (thêm vào đó tiếp theo). Lưu ý rằng tôi không phải tạo một bảng, tôi không phải ánh xạ các cột và các lớp vào các bảng và tôi không phải tạo bất kỳ thủ tục được lưu trữ nào. Các Company lớp học thậm chí không được đánh dấu là Serializable - nó chỉ hoạt động.

Cũng nên nhớ rằng việc tạo một DocumentStore đối tượng nên được coi là một hoạt động tốn kém, tương tự như việc tạo một nhà máy phiên NHibernate. Hiện tại nó không phải là nhưng có kế hoạch làm việc trong tương lai có thể thay đổi điều này.

Công cụ Quản trị dựa trên trình duyệt:



Khi máy chủ Raven đang chạy, nó có thể được truy cập thông qua trình duyệt để kiểm tra nội dung, chỉ mục của nó và xem tài liệu.

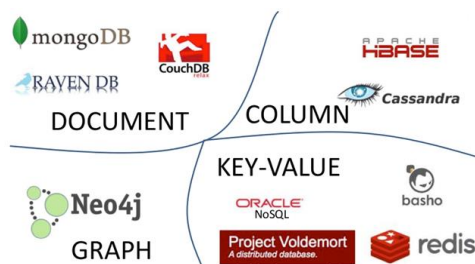


Bạn thậm chí có thể chỉnh sửa tài liệu và định nghĩa chỉ mục bằng trình duyệt của mình.

Xem thêm video giới thiệu của RavenDB với vietsub: <https://goo.gl/JbXyfG>

Ưu điểm và nhược điểm (các trường hợp sử dụng)

1. Ưu điểm



Không giống như một số công nghệ nhất định, hầu như không có bất kỳ cấu hình nào liên quan. Bạn chỉ cần đưa RavenDb một đối tượng hoặc bất kỳ hình dạng hoặc kích thước và RavenDb lưu trữ nó.

RavenDb là một cơ sở dữ liệu tài liệu. Điều đó có nghĩa là bạn không cần khai báo một lược đồ trước. Bất cứ khi nào bạn đưa RavenDb một đối tượng, nó chỉ đơn giản là sắp xếp thành định dạng JSON và lưu trữ nó trên công cụ của nó. Lợi ích có thể thấy từ điều này là giảm đi rất nhiều khi cập nhật cho lại code, vì chúng ta sẽ không cần phải lo lắng về việc thay đổi script. Chúng ta cũng có thể thêm vào các đối tượng và trong các tình huống khác, RavenDb có hỗ trợ chỉnh sửa tài liệu rất tốt.

Dễ sử dụng

RavenDb chủ yếu được viết bằng C và .NET, và do đó rất dễ hiểu cách sử dụng nó. Đây là một lợi thế tốt nhất. Dữ liệu có thể được truy vấn hiệu quả với sự trợ giúp của các truy vấn LINQ từ code .NET hoặc bằng cách sử dụng RESTful là các API chuyển trạng thái REpresentational .

Lập chỉ mục (indexes) phần lớn là tự động

Trong RavenDb, lược đồ cơ sở dữ liệu không cố định lâu; thay vào đó, dữ liệu được lưu trữ không có lược đồ dưới dạng tài liệu JSON. Do đó các tài liệu có nhiều cấu trúc tùy ý cũng như các thuộc tính được liên kết với chúng. Bên trong, RavenDb sử dụng tốt nhất các chỉ mục indexes, được tạo tự động phụ thuộc vào cách sử dụng của người dùng hoặc được tạo ra một cách rõ ràng bởi các máy khách.

Một điều nổi bật là hỗ trợ lập chỉ mục (index) tuyệt vời của RavenDb. Bất cứ khi nào bạn tạo một loại truy vấn mới, RavenDb sẽ tự động tạo một chỉ mục để bạn truy vấn. Trong các phiên bản gần đây nhất của RavenDb, nó sẽ tự động điều chỉnh các chỉ mục theo thời gian, tùy thuộc vào các mẫu truy vấn bạn thực hiện đối với cơ sở dữ liệu. Tất nhiên, bạn có thể xác định các chỉ mục phức tạp hơn một cách rõ ràng, nhưng thường bạn không bao giờ cần.

Một tình huống mà bạn sẽ cần tạo các chỉ mục là khi bạn muốn tận dụng sự hỗ trợ Map / Reduce của RavenDb . Đây là cách RavenDb hỗ trợ các truy vấn liên quan đến việc tổng hợp nhiều tài liệu, vì nó không thực hiện các nhóm câu lệnh trong truy vấn LINQ từ máy khách. Điều thú vị về cách thức thực hiện điều này là các chỉ mục Map / Reduce được duy trì giống như các chỉ mục khác, làm cho các truy vấn tổng hợp rất nhanh.

Tính mở rộng cao

RavenDb rất dễ mở rộng và có thể được xây dựng để mở rộng quy mô web. Nó cung cấp bản sao cùng với hỗ trợ sharding. Điều này quan trọng nếu bạn muốn đảm bảo rằng ứng dụng của bạn có tính khả dụng cao.

Cấu hình sao chép cũng dễ dàng như nhập chuỗi kết nối cho cơ sở dữ liệu bạn muốn RavenDb nhân rộng. Sao chép cũng giúp rất nhiều với khả năng mở rộng, vì bạn có thể cấu hình máy khách của mình để đọc từ bất kỳ bản sao nào, do đó mở rộng đường truyền, nhưng ghi vào cơ sở dữ liệu chủ, đảm bảo tính nhất quán mà không tạo xung đột.

Hỗ trợ sao lưu

RavenDb hỗ trợ sao lưu vào Amazon S3 hoặc Amazon Glacier. Bạn chỉ cần cung cấp nơi mà bạn muốn viết, các khóa API và nó sẽ tự động thực hiện.

Mức độ hỗ trợ luôn luôn là một cân nhắc quan trọng khi lựa chọn áp dụng một sản phẩm, và RavenDb có sự hỗ trợ tuyệt vời, cả từ cộng đồng RavenDb và từ Ayende (người tạo ra RavenDb) và phần còn lại của nhóm Hibernating Rhinos. RavenDb là một dự án mã nguồn mở (sử dụng giấy phép AGPL, vì vậy bạn phải mua giấy phép nếu bạn không sẵn sàng mở mã nguồn của riêng mình).

Bên cạnh đó, nếu RavenDb có một lỗi, ngoài việc tìm lỗi để sửa thì việc này đã được cải thiện rất nhiều gần đây. Chúng ta có thể dựa vào cộng đồng để được giúp đỡ.

RavenDb được xây dựng để lưu trữ dữ liệu nhanh chóng

Trong SQL Server, chúng ta dành tất cả thời gian để tìm hiểu cách “nhồi nhét dữ liệu vuông vào lỗ tròn” và sau đó dành tất cả thời gian cho việc phân chia các quyết định đó. Trong RavenDb, thay vào đó chúng ta dành thời gian để quyết định dữ liệu nào nên được giao dịch với nhau và sau đó lưu trữ nó, tuy nhiên nó có ý nghĩa.

RavenDb đi kèm với các thuộc tính hỗ trợ **ACID**:

Bộ nhớ ACID:

Tất cả các hoạt động lưu trữ được thực hiện trong RavenDb hoàn toàn là ACID (Atomicity, Consistency, Isolation, Durability), bởi vì RavenDb nội bộ đã sử dụng một công cụ lưu trữ tùy chỉnh được gọi là *Voron* (hoạt động ở tốc độ lên tới 1 triệu lần đọc mỗi giây và 150.000 ghi mỗi giây trên một nút duy nhất) , đảm bảo tất cả các thuộc tính của ACID, cho dù chúng có được thực hiện trên tài liệu hay không , chỉ mục hoặc dữ liệu lưu trữ cụm.

ACID cho thao tác trên tài liệu (ACID for document operations):

Trong RavenDb, tất cả các thao tác được thực hiện trên các tài liệu đều là ACID hoàn toàn. Mỗi thao tác tài liệu hoặc một loạt các thao tác được áp dụng cho một tập hợp các tài liệu được gửi trong một yêu cầu HTTP sẽ thực thi trong một giao dịch duy nhất. Các thuộc tính ACID của RavenDb là:

- *Atomicity* - Tất cả các hoạt động đều là atomic. Hoặc là thành công hay thất bại, không phải là thao tác giữa chừng. Đặc biệt, các hoạt động trên nhiều tài liệu tất cả sẽ xảy ra một cách *atomic*, tất cả các cách hoặc không có gì cả.

- *Tính nhất quán và cách ly (Consistency and Isolation)*- Trong một giao dịch độc lập, tất cả các hoạt động hoạt động dưới sự cô lập. Ngay cả khi bạn truy cập nhiều tài liệu, bạn sẽ nhận được tất cả trạng thái của chúng như khi bắt đầu yêu cầu.
- *Hiển thị (Visibility)* - Tất cả các giao dịch ngay lập tức được thực hiện. Do đó, nếu giao dịch được thực hiện sau khi cập nhật hai tài liệu, bạn sẽ luôn thấy các cập nhật cho hai tài liệu đó cùng một lúc. (Tức là, bạn sẽ thấy các cập nhật cho cả hai hoặc bạn không thấy cập nhật cho cả hai).
- *Độ bền (Durability)*- Nếu một thao tác đã hoàn tất, nó đã được chuyển thành đĩa. Việc đọc sẽ không bao giờ trả lại bất kỳ dữ liệu nào chưa được flushed vào đĩa.

Tất cả những ràng buộc này được đảm bảo khi bạn sử dụng một session và gọi SaveChanges .

BASE cho thao tác truy vấn

Mô hình giao dịch là khác nhau khi các chỉ mục (index) có liên quan, vì các chỉ mục là BASE (Về cơ bản có sẵn, tính nhất quán cuối cùng), không phải ACID. Sau đó, các ràng buộc sau đây được áp dụng cho các hoạt động truy vấn:

- *Cơ bản có sẵn (Basically Available)* - Kết quả truy vấn chỉ mục sẽ luôn có sẵn nhưng chúng có thể cũ.
- *Soft state* - Trạng thái của hệ thống có thể thay đổi theo thời gian vì cần một khoảng thời gian để thực hiện lập chỉ mục.
- *Sự nhất quán (Eventual consistency)* - Cơ sở dữ liệu cuối cùng sẽ trở nên nhất quán khi nó ngừng nhận tài liệu mới và thao tác lập chỉ mục kết thúc.

Tóm lại nên sử dụng RavenDb vì một trong những ưu điểm nổi bật sau:

- ❖ Dễ cài đặt và sử dụng. Không tốn chi phí.
- ❖ Dễ sử dụng: RQL truy vấn dựa trên SQL.
- ❖ Hoạt động tốt với cơ sở dữ liệu quan hệ hiện có.
- ❖ Đa nền tảng: C #, Node.js, Java, Python, Ruby, Go.
- ❖ Hệ thống đa hệ điều hành: Windows, Linux, Mac OS, Docker, Raspberry Pi.
- ❖ Sử dụng phần cứng tối đa: RavenDb hoạt động trên các chip Raspberry Pi và ARM vì nó tận dụng tối đa phần cứng mà nó chạy trên đó.
- ❖ Hoạt động hiệu quả trên các máy cũ hơn và các thiết bị nhỏ hơn.
- ❖ Được xây dựng trong tìm kiếm toàn văn bản, Map/Reduce và công cụ lưu trữ.

2. Nhược điểm

Theo xu hướng ngày nay, hầu hết người dùng thường sử dụng MongoDB cho những dự án lớn của mình. Do đó việc tìm ra những nhược điểm của RavenDb là điều vô cùng khó khăn và ít ai quan tâm. Cụ thể, chúng ta có thể tham khảo bảng dưới đây để tìm ra một vài nhược điểm của database này:

Name	MongoDb	RavenDb
Description	One of the most popular document stores	Open Source Operational and Transactional Enterprise NoSQL Document Database
DB-Engines Ranking	Score 363.19 Rank #5 <u>Overall</u> #1 <u>Document stores</u>	Score 3.56 Rank #70 <u>Overall</u> #13 <u>Document stores</u>
Supported programming languages	Actionscript , C, C#, C++, Clojure , ColdFusion, D, Dart, Delphi, Erlang, Go, Groovy, Haskell, Java, JavaScript, Lisp, Lua, MatLab, Perl, PHP, PowerShell, Prolog, Python, R, Ruby, Scala, Smalltalk	.Net, C#, Go, Java, JavaScript (Node.js), Python, Ruby
Specific characteristics	MongoDb is the fastest-growing database ecosystem, with over 40 million downloads, thousands of customers, 1M+ registrations for MongoDB University courses, and over 1,000 technology and service partners.	Mentioned in both Gartner and Forrester research, RavenDb is a pioneer in the newest database technology with over 2 million downloads and thousands of customers from Startups to Fortune 100 Large Enterprises.
Market metrics	<ul style="list-style-type: none"> 40 million downloads (growing at 30 thousand downloads per day). 6,600+ customers. Named a leader in the <i>Big Data NoSQL</i>, Q3 2016. Highest placed non-relational database in DB Engines rankings 	<ul style="list-style-type: none"> 2 million+ downloads 1000+ customers including Fortune 500 large enterprises.

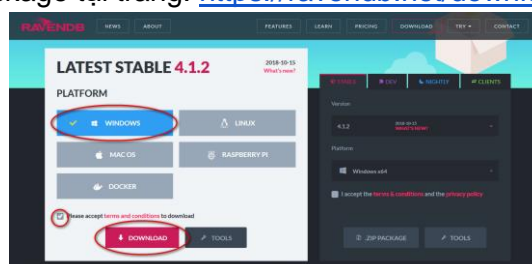
Từ bảng so sánh trên giữa MongoDB và RavenDb, ta có thể rút ra một vài nhược điểm của RavenDb như sau:

- ❖ Được người dùng đánh giá với số điểm rất thấp, đứng hạng 70 trong số các Database và đứng hạng 13 về việc lưu trữ tài liệu.
- ❖ Chỉ hỗ trợ một số ngôn ngữ lập trình (tương đối ít).
- ❖ Số lượng người downloads sử dụng khoảng hơn 2 triệu, hơn 1000 khách hàng bao gồm 500 doanh nghiệp lớn.
- ❖ Theo như người sáng lập RavenDb có đề cập trong một bài viết của mình thì RavenDb có thể làm mất dữ liệu và cơ sở dữ liệu không thể khôi phục.
- ❖ Những vấn đề về dữ liệu thường rơi vào hai loại: vấn đề quá tải về chỉ mục / dữ liệu hoặc các vấn đề về API / usage.

Cài đặt RavenDB

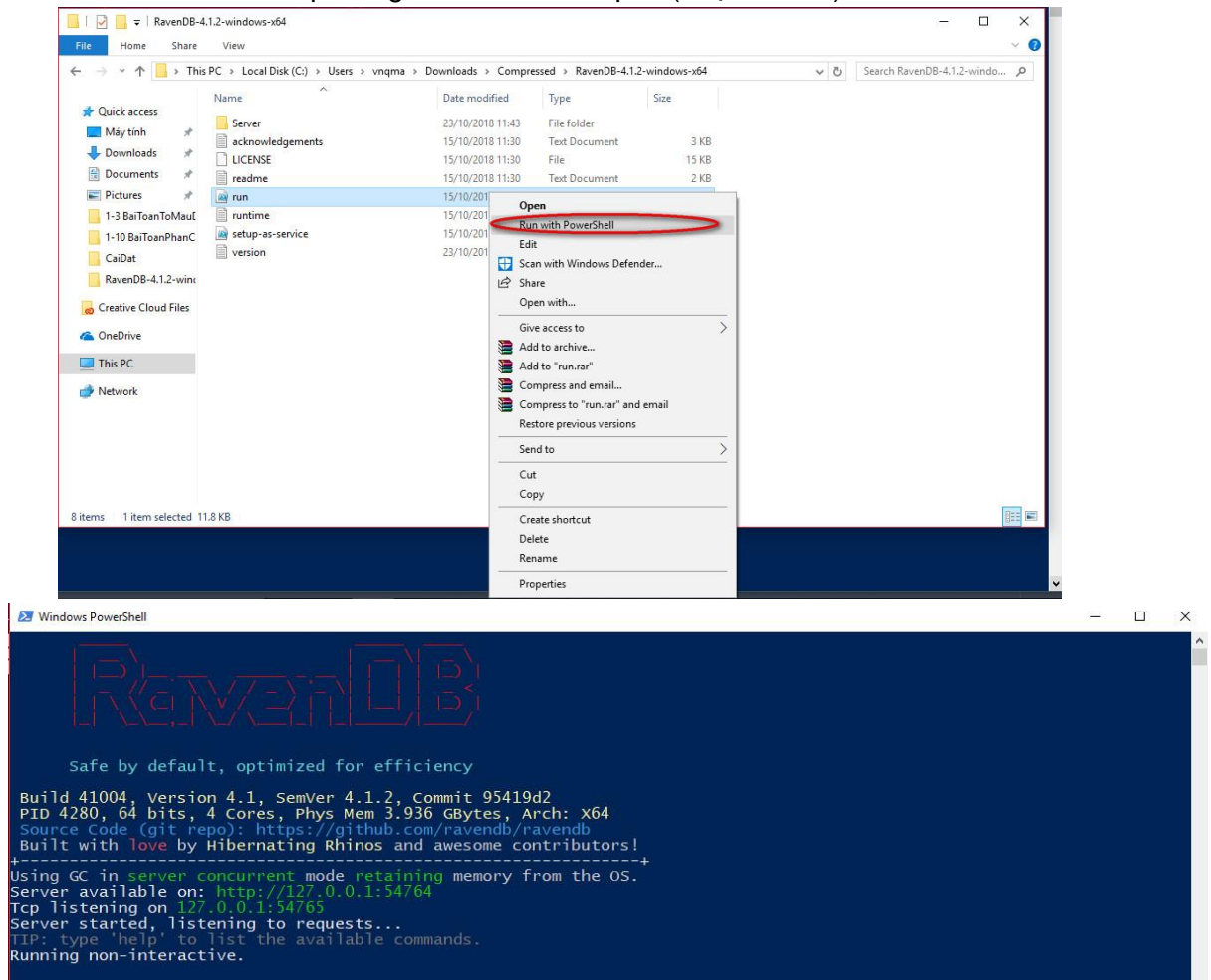
Server

Tải server package tại trang: <https://ravendb.net/downloads>



Cài đặt

Giải nén server package và mở file run.ps1 (hoặc run.sh).



Màn hình cài đặt xuất hiện trên trình duyệt. Kéo thanh cuộn xuống cuối và chọn Accept



Có 3 lựa chọn bảo mật cho máy chủ gồm:

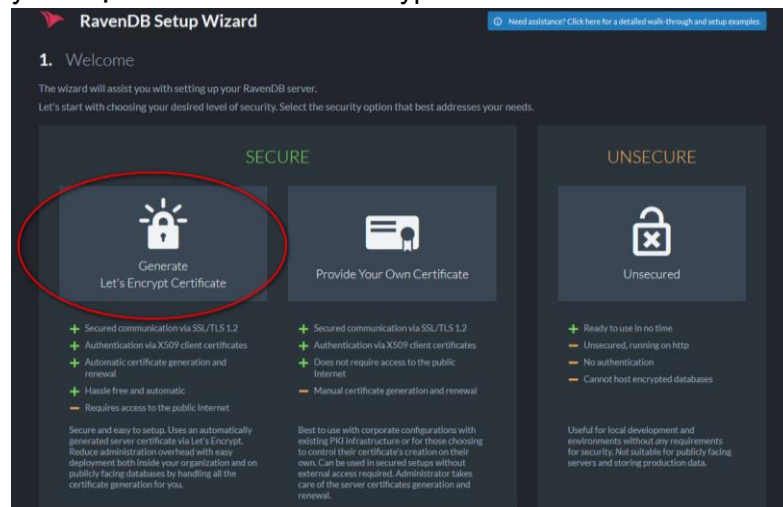
Secure:

Generate Let's Encrypt Certificate

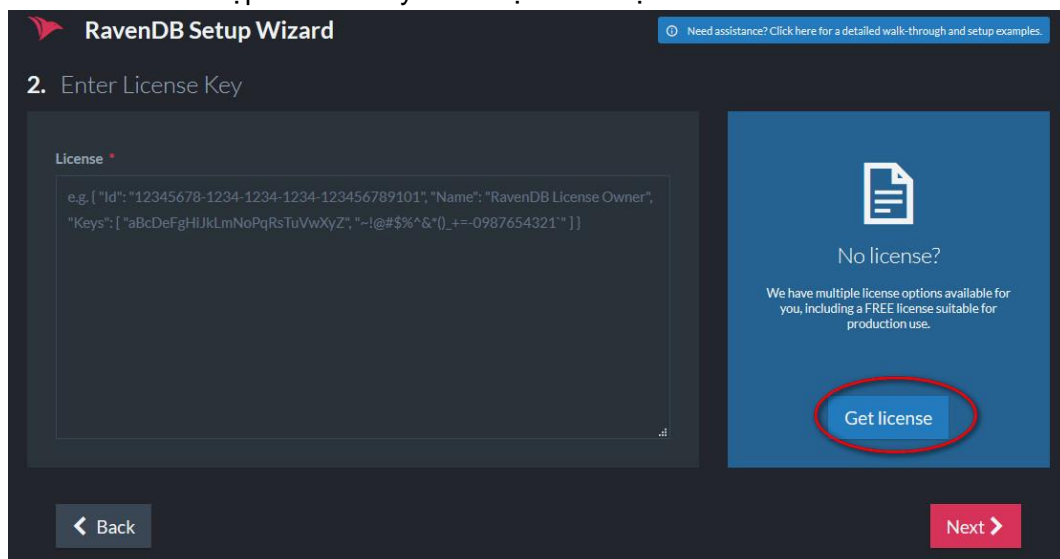
Provide Your Own Certificate

Unsecure

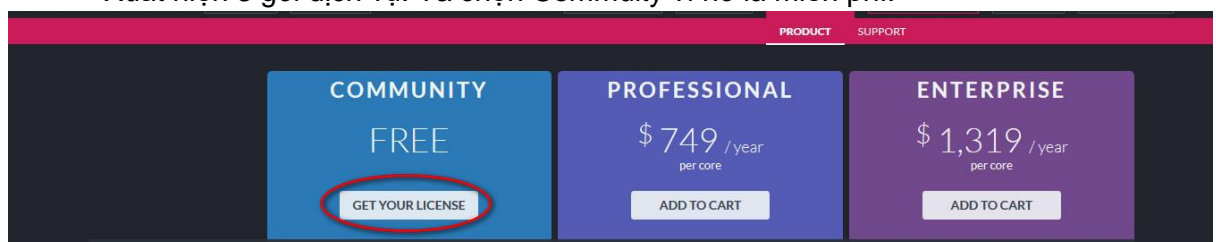
Ở đây ta chọn Generate Let's Encrypt Certificate



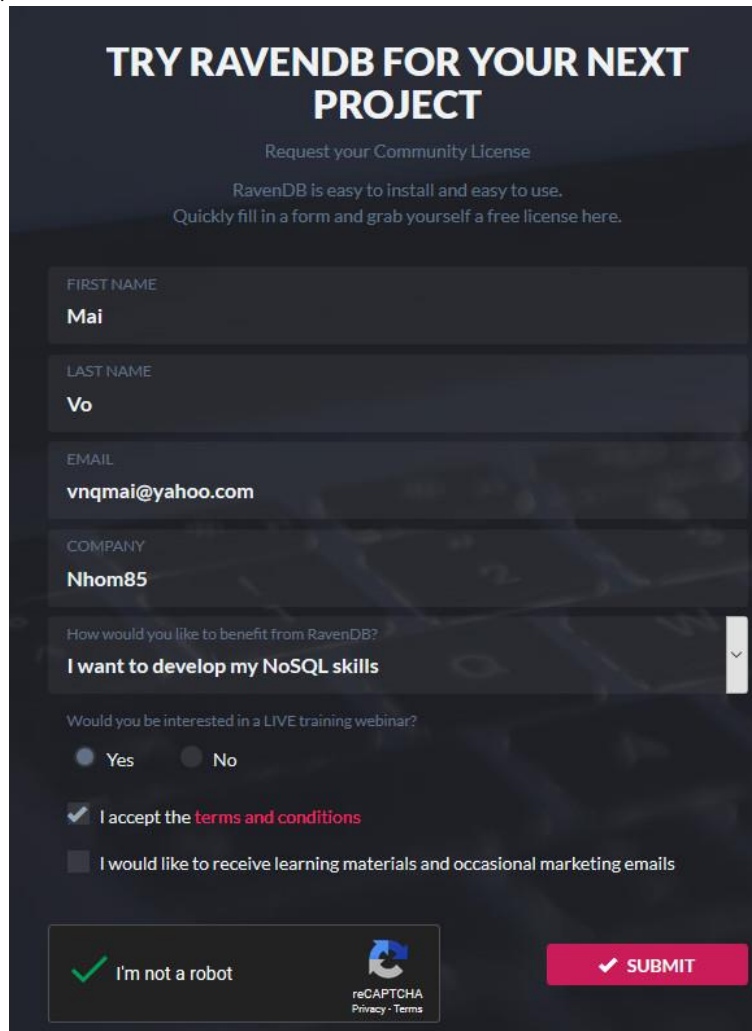
Yêu cầu nhập license key xuất hiện. Ta chọn Get license



Xuất hiện 3 gói dịch vụ. Ta chọn Community vì nó là miễn phí.



Ta điền form sau để nhận mail chứa license key. (email phải ghi chính xác để nhận key)



TRY RAVENDB FOR YOUR NEXT PROJECT

Request your Community License

RavenDB is easy to install and easy to use.
Quickly fill in a form and grab yourself a free license here.

FIRST NAME
Mai

LAST NAME
Vo

EMAIL
vnqmai@yahoo.com


COMPANY
Nhom85

How would you like to benefit from RavenDB?
I want to develop my NoSQL skills

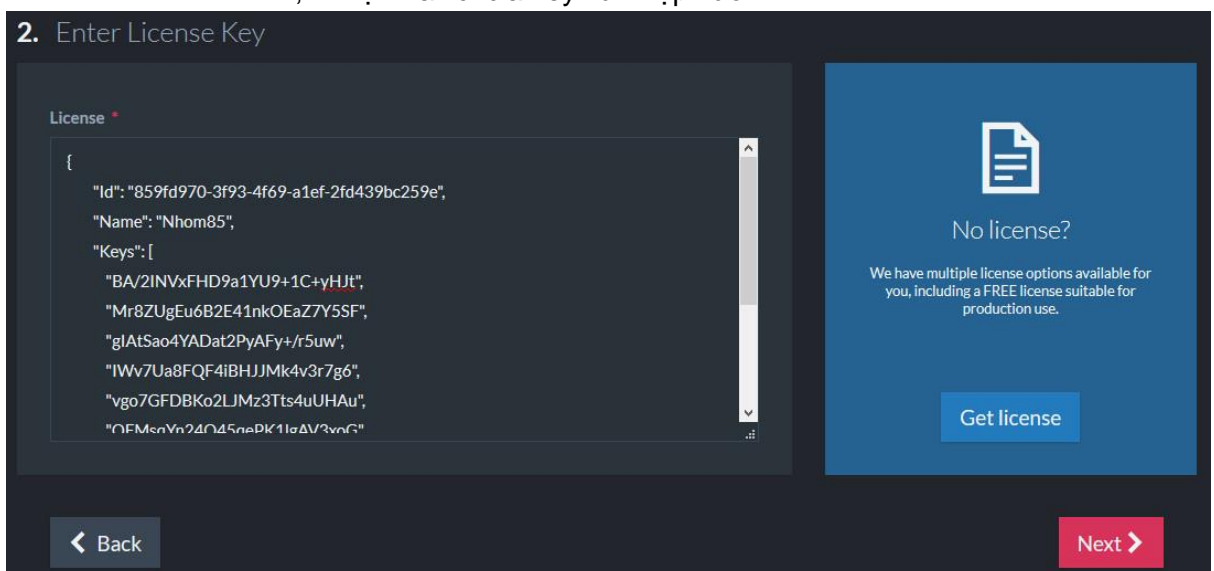
Would you be interested in a LIVE training webinar?
☐ Yes ☐ No

☒ I accept the **terms and conditions**

☐ I would like to receive learning materials and occasional marketing emails

☒ I'm not a robot  **SUBMIT**

Sau khi submit, ta đợi mail chứa key và nhập vào



2. Enter License Key

License *

```
{
  "Id": "859fd970-3f93-4f69-a1ef-2fd439bc259e",
  "Name": "Nhom85",
  "Keys": [
    "BA/2INVxFHD9a1YU9+1C+yHJt",
    "Mr8ZUgEu6B2E41nkOEaZ7Y5SF",
    "glAtSao4YADat2PyAFy+/r5uw",
    "IWv7Ua8FQF4iBHJJK4v3r7g6",
    "vgo7GFDBKo2LJMz3Tts4uUHAu",
    "QFEMsnYn24Q45naPK1laAV3ynG"
  ]
}
```

No license?

We have multiple license options available for you, including a FREE license suitable for production use.

Get license

< Back **Next >**

Nhập tên miền server và chọn Next

3. Choose Domain

Domain *

Full domain **nhom85.ravendb.community**

Email **grisha@ayende.com**

[< Back](#) [Next >](#)

Add node

RavenDB Setup Wizard [Need assistance? Click here for a detailed walk-through and setup examples.](#)

4. Node Addresses

Please enter the server settings - IP addresses and ports.
If you are building a cluster this is the place to add nodes and configure them.

▲ The following configuration will populate the DNS records for the subdomain **nhom85.ravendb.community**.

[+ Add node](#)

Node **DOAN** (local node)

Node DOAN URL:
https://doan.nhom85.ravendb.community

Node Tag: **DOAN**

HTTPS Port: Default: 443

TCP Port: Default: 38888

IP Address: **127.0.0.1**
Hostname: This node is not reachable from outside this machine.

[+ Add another IP Address](#)

RavenDB will update the DNS record for:
doan.nhom85.ravendb.community to IP Addresses: **127.0.0.1**

☒ Customize external IP and Ports

External IP Address:

External HTTPS Port:

External TCP Port:

Environment: **None**

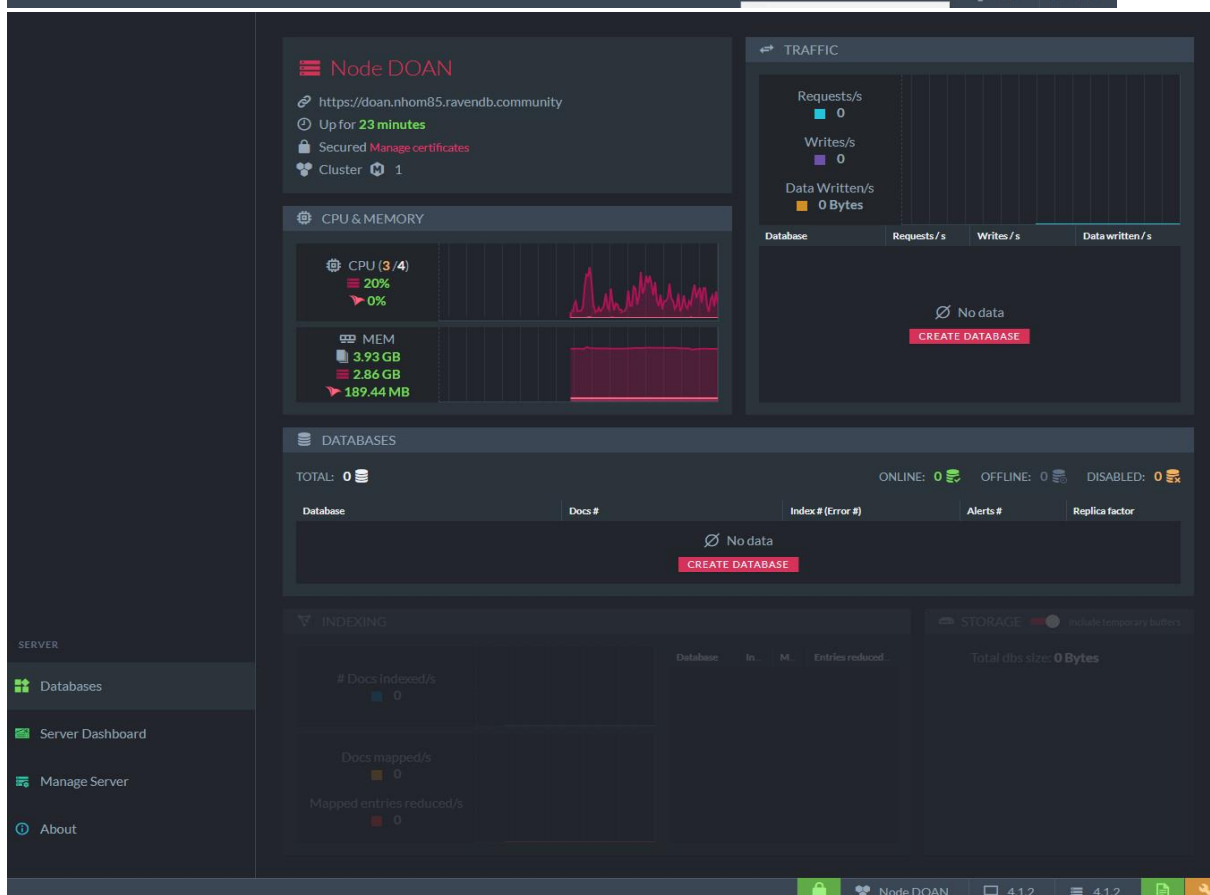
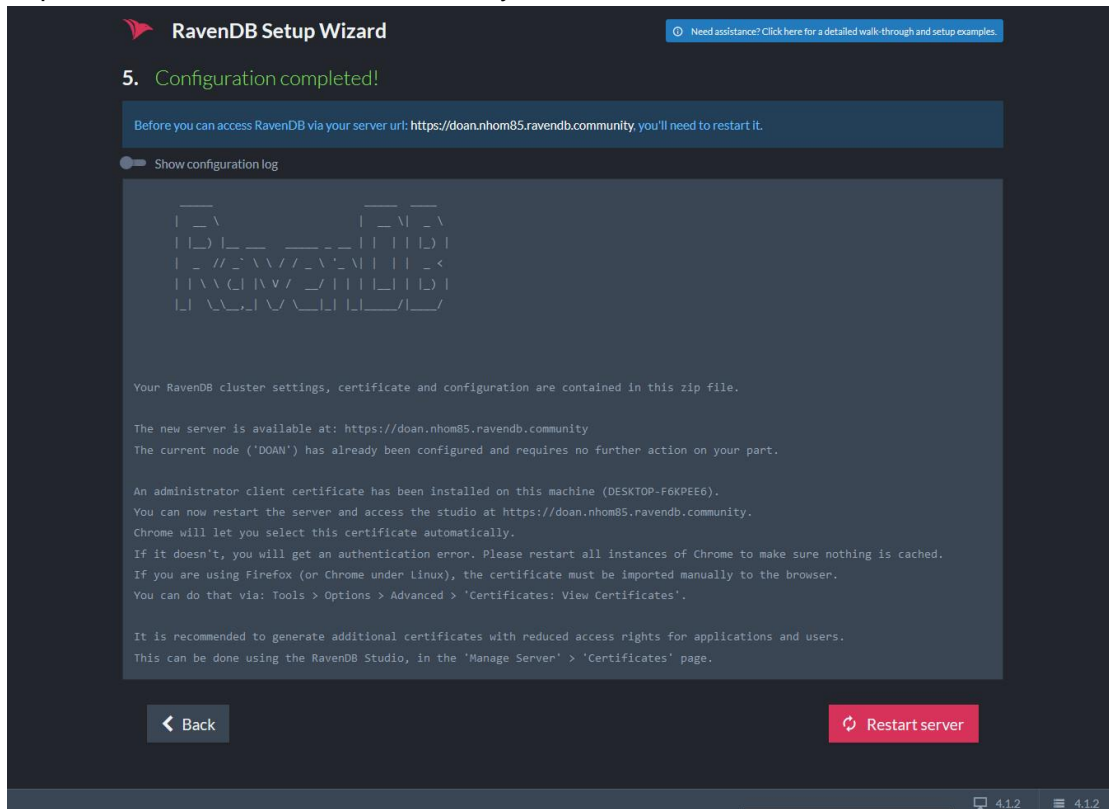
☒ Automatically register the admin client certificate in this (local) OS

☒ I accept [Let's Encrypt Subscriber Agreement](#)

[< Back](#) [Next >](#)

4.1.2 4.1.2

Hoàn thành cài đặt và cấu hình server, 1 file zip chứa cấu hình và các certificate ssl được yêu cầu tải về, ta tải chúng về và chọn Restart Sertver với tên node vừa add, ở đây là <https://doan.nhom85.ravendb.community>



Cách thức truy vấn và xử lý dữ liệu trên RavenDB

1. RQL-Ngôn ngữ truy vấn ravedb

RQL, ngôn ngữ truy vấn Raven là một ngôn ngữ SQL nhưng được sử dụng để lấy dữ liệu từ máy chủ khi truy vấn đang được thực thi.

Nó được thiết kế để truy vấn RavenDB một cách dễ hiểu, dễ sử dụng, và không quá phức tạp cho người dùng.

2. Keywords và Methods

Dưới đây là những Keywords và Method trong RQL:

- | | |
|---|---|
| <ul style="list-style-type: none">• DECLARE• FROM<ul style="list-style-type: none">◦ INDEX• GROUP BY<ul style="list-style-type: none">◦ <u>array()</u>• WHERE<ul style="list-style-type: none">◦ id()◦ <u>search()</u>◦ cmpxchg()◦ boost()◦ <u>regex()</u>◦ startsWith()◦ endsWith()◦ <u>lucene()</u>◦ exists()◦ exact()◦ <u>intersect()</u>◦ <u>spatial.within()</u>◦ <u>spatial.contains()</u>◦ <u>spatial.disjoint()</u>◦ <u>spatial.intersects()</u>◦ <u>moreLikeThis()</u> | <ul style="list-style-type: none">• ORDER BY<ul style="list-style-type: none">◦ <u>ASC ASCENDING</u>◦ <u>DESC DESCENDING</u>◦ <u>AS</u>◦ <u>string</u>◦ <u>long</u>◦ <u>double</u>◦ <u>alphaNumeric</u>◦ <u>random()</u>◦ <u>score()</u>◦ <u>spatial.distance()</u>• LOAD• SELECT<ul style="list-style-type: none">◦ DISTINCT◦ key()◦ sum()◦ count()◦ <u>facet()</u>• UPDATE• INCLUDE |
|---|---|

Cùng với những toán tử:

- | | |
|--|--|
| <ul style="list-style-type: none">• >=• <=• <> or !=• <• >• = or ==• BETWEEN | <ul style="list-style-type: none">• IN• ALL IN• OR• AND• NOT• (•) |
|--|--|

Và những giá trị sau:

- true
- false
- null
- string e.g. 'John' or "John"
- number (long and double) e.g. 17

- parameter e.g. \$param1

DECLARE

Từ khóa Declare cung cấp cho bạn khả năng tạo một JS function có thể được tái sử dụng (Khi project được hoàn thành).

FROM

Từ khóa From dùng để xác định nguồn dữ liệu khi truy vấn được thực thi.

GROUP BY

Từ khóa dùng để tạo truy vấn tập hợp.

WHERE

Từ khóa dùng để lọc dữ liệu từ kết quả cuối cùng.

LOAD

Khi cần sử dụng dữ liệu từ external document trong project. Load có thể được sử dụng.

SELECT

Từ khóa dùng để chọn Projects để thực thi..

3. Khái niệm cơ bản

Query-Flow

Mỗi truy vấn trong RavenDB phải được thể hiện qua RQL . Mỗi truy vấn phải phù hợp với một chỉ số để trở về các kết quả. Query-Flow là như sau:

1. from index | collection
2. where
3. load
4. include
5. Return results.

Các loại truy vấn

Truy vấn sử dụng LINGQ:

RavenDB Client hỗ trợ truy vấn sử dụng LINQ. Chức năng này có thể được truy cập bằng cách sử dụng các session query method, và là phương pháp phổ biến và cơ bản nhất cho truy vấn cơ sở dữ liệu.

Truy vấn cơ bản:

Để có toàn quyền kiểm soát truy vấn ravendb giới thiệu phương pháp DocumentQuery-một phương pháp có sẵn trong. Nó là một truy cập cơ bản vào các cơ chế người dùng có thể sử dụng truy vấn theo nhu cầu của mình.

4. Xử lý dữ liệu với Query

Filter

Một trong các chức năng cơ bản nhất của truy vấn là khả năng để lọc ra các dữ liệu và trả lại hồ sơ phù hợp với một điều kiện nhất định. Có vài cách để làm điều này và tất cả đều phụ thuộc vào phương pháp dùng mà bạn muốn sử dụng (Queryfrom basic session operations, DocumentQuery from advanced session operations, or directly using RQL).

Ví dụ sau đây chứng minh làm thế nào để thêm điều kiện đơn giản cho một truy vấn sử dụng tất cả các phương pháp đó.

Where

```
from index 'Employees/ByFirstAndLastName'
where FirstName = 'Robert' and LastName = 'King'
```

Where-Numeric Property

```
from index 'Products/ByUnitsInStock'
where UnitsInStock > 50
```

Where-Nested Property

```
from Orders
where ShipTo.City = 'Albuquerque'
```

Where+Any

[Any](#) rất hữu ích khi bạn có một bộ sưu tập và bạn muốn lọc ra dựa trên các giá trị từ bộ sưu tập này.

```
from index 'Order/ByOrderLinesCount'
where Lines_ProductName = 'Teatime Chocolate Biscuits'
```

Where+ConstainAny

Để kiểm tra nếu enumeration chứa bất kì các giá trị từ một bộ sưu tập được chỉ định, bạn có thể sử dụng các phương pháp ContainsAny

```
from index 'BlogPosts/ByTags'
where Tags IN ('Development', 'Research')
```

Where+ConstainAll

Để kiểm tra nếu enumeration chứa tất cả các giá trị từ một bộ sưu tập được chỉ định, bạn có thể sử dụng các phương pháp ContainsAll.

```
from index 'BlogPosts/ByTags'
where Tags ALL IN ('Development', 'Research')
```

Where+StartsWith

```

from Products
where startsWith(Name, 'ch')

Where+EndsWith
from Products
where endsWith(Name, 'ra')

```

Paging

Paging, hoặc pagination, là quá trình chia tách một tập dữ liệu vào các trang, đọc một trang một lúc. Điều này là hữu ích cho việc tối ưu hóa băng thông sử dụng lưu lượng truy cập và phần cứng hoặc chỉ đơn giản là vì không có người có thể xử lý một lượng lớn dữ liệu cùng một lúc.

Cảnh báo

Bắt đầu từ phiên bản 4.0, nếu kích thước trang là không được chỉ định **vào phía khách hàng**, các máy chủ sẽ giả **int.MaxValue** (2,147,483,647) và tất cả các kết quả sẽ được tải xuống. Đó là **đề nghị để thiết lập một kích thước trang một cách rõ ràng** để tránh thời gian dài phản ứng gây ra bằng cách gửi quá nhiều dữ liệu trên mạng hoặc bộ nhớ tiêu thụ cao do xử lý một lượng lớn tài liệu.

Bạn cũng có thể đặt `DocumentConventions.ThrowIfQueryPageSizesNotSet` thành `true` để bảo vệ bản thân từ thực hiện các truy vấn mà không có kích thước trang một cách rõ ràng thiết lập. Chúng tôi khuyên bạn nên bật công ước này, đặc biệt là trong phát triển hoặc thử nghiệm giai đoạn để phát hiện sớm các truy vấn có khả năng có thể trở lại với một kết quả khổng lồ.

Hiệu suất

Theo mặc định, nếu vượt quá số lượng các kết quả trả về **2048**, máy chủ sẽ phát hành một thông báo (có thể nhìn thấy trong Studio) với các thông tin về truy vấn thông tin chi tiết. Bạn có thể quyết định nếu hành vi này là mong muốn hay không. Ngưỡng có thể được điều chỉnh bằng cách thay đổi giá trị cấu hình `PerformanceHints.MaxNumberOfResults`.

Distinct

Phương pháp này cho phép bạn để loại bỏ trùng lặp từ kết quả.

```

from Orders
select distinct ShipTo.Country

```

Searching

```

from Users
where search(Name, 'John Adam')

```

Cách thức truy vấn và xử lý dữ liệu của RavenDB trên C#

Sau khi mở chạy server, để viết ứng dụng bạn cần có thư viện hỗ trợ mặt Client:

- .NET có [NuGet](#)
- Java có [Maven](#)
- Node.js có [NPM](#)
- Python có [PyPi](#)
- [Ruby](#)
- [Go](#)

Ở đây, chúng ta sử dụng .NET với NuGet

1. Các thành phần cấu tạo và phương thức lưu trữ, tải dữ liệu và delete *DocumentStore*

DocumentStore là đối tượng chính để tạo và quản lý kết nối giữa RavenDB server (hoặc cluster) và ứng dụng của bạn.

Document store có danh sách địa chỉ (Url) chỉ đến các node của RavenDB server.

Để tạo *DocumentStore*, bạn cần có danh sách các địa chỉ chỉ đến các node của RavenDB server, ở đây ta có địa chỉ 1 node là <https://doan.nhom85.ravendb.community>. Nội dung code như sau

```
1 using (IDocumentStore store = new DocumentStore()  
2 {  
3     Urls = new[] { "http://doan.nhom85.ravendb.community" }  
4 }.Initialize()  
5 {  
6     //tương tác với database  
7 })
```

Đoạn code này sẽ tạo một kênh kết nối giữa ứng dụng và local RavenDB server.

Ngoài ra ta có thể chỉ ra database cụ thể trên server ở đây ta có 1 database "Northwind"

```
1 using (IDocumentStore store = new DocumentStore()  
2 {  
3     Urls = new[] { "http://doan.nhom85.ravendb.community" },  
4     Database = "Northwind"  
5 }.Initialize()  
6 {  
7     //tương tác với database  
8 })
```

Cài đặt authentication (xác thực) và authorization (ủy quyền) cho phía client

Để client có thể vào được web có kết nối với RavenDB server ta cần kết nối các chứng nhận ta đã cài đặt trên server và tải về vào code

```

1 // mở certificate để kết nối
2 X509Certificate2 clientCertificate = new X509Certificate2("C:\\path_to_your_pfx_file\\cert.pfx");
3
4 using (IDocumentStore store = new DocumentStore()
5 {
6     Certificate = clientCertificate,
7     Database = "Northwind",
8     Urls = new[] { "https://doan.nhom85.ravendb.community" }
9 }.Initialize())
10 {
11     // tương tác với database
12 }

```

Session

Sau khi tạo document store, chúng ta đã có thể dùng database đã được chỉ đến. Để tương tác với RavenDB, chúng ta cần tạo đối tượng session trên document store. Đối tượng session sẽ bao gồm mọi thứ chúng ta cần để tương tác với database.

Session được sử dụng để tương tác với dữ liệu. Session khác với DocumentStore, nó là một đối tượng nhẹ được tạo thường xuyên hơn DocumentStore. Ví dụ, trong ứng dụng web, cách tương tác phổ biến (được khuyến khích) là tạo Session cho mỗi lần request.

Storing

RavenDB là một Database dạng Document. Các đối tượng lưu trữ được gọi là các documents. Mỗi document bao gồm **unique ID** để xác định document đó, **data** và **metadata**, tất cả được lưu trữ dưới dạng JSON. Metadata bao gồm thông tin mô tả về document, ví dụ như ngày sửa đổi cuối cùng (@last-modified property) hoặc các tập hợp hỗ trợ (@collection property).

```

1 using (IDocumentSession session = store.OpenSession()) // Mở 1 session trên database đã khai báo
2 {
3     Category category = new Category
4     {
5         Name = "Database Category"
6     };
7
8     session.Store(category); // Lưu một category
9
10
11     Product product = new Product
12     {
13         Name = "RavenDB Database",
14         Category = category.Id,
15         UnitsInStock = 10
16     };
17
18     session.Store(product); // Lưu một products
19
20     session.SaveChanges(); // Gửi lên Server
21                             // 1 request tiến hành trên 1 transaction
22 }

```

Dùng lệnh .Store để lưu dữ liệu vào database

Loading

Session giúp cho mọi **sự thay đổi** trên các đối tượng được **theo dõi tự động**. Lệnh SaveChanges sẽ đồng bộ (với server) **những document đã**

thay đổi với session. Tất cả sự thay đổi đó được gửi trong một request (tiết kiệm các câu lệnh truy vấn) và tiến hành trong một transaction.

```

1 using (IDocumentSession session = store.OpenSession()) // Mở 1 session trên database đã khai báo
2 {
3     Product product = session
4         .Include<Product>(x => x.Category)           // Include Category
5         .Load(productId);                           // Load Product và bắt đầu duyệt
6
7     Category category = session
8         .Load<Category>(product.Category);           // không cần gọi include,
9                                                     // Session bao gồm thực thể này từ .Include
10
11     product.Name = "RavenDB";                       // Áp dụng các thay đổi
12     category.Name = "Database";
13
14     session.SaveChanges();                           // Đồng bộ với Server
15                                                     // 1 request được tiến hành trong 1 transaction
16 }

```

Truy vấn product

Dùng lệnh .Include<Product>(x=>x.Category) để chọn tập hợp giá trị của thuộc tính Category trong bảng Product

Dùng lệnh .Load(productId) để duyệt tìm trên tập hợp đã chọn có giá trị bằng productId và load vào biến product

Truy vấn category

Không cần .Include mà chỉ dùng .Load RavenDB có thể gọi đúng thuộc tính cần xét để kiểm tra và load vào biến category

Deleting

```

1 /*Cách 1*/
2 //syntax
3 void Delete<T>(T entity);
4 //Ví dụ
5 Employee employee = session.Load<Employee>("employees/1");
6 session.Delete(employee);
7 session.SaveChanges();
8 /*=====*/
9 /*Cách 2*/
10 //Syntax
11 void Delete(string id);
12 //ví dụ
13 session.Delete("employees/1");
14 session.SaveChanges();
15

```

Cách 1:

Khởi tạo một employee mẫu để duyệt tìm

Dùng session.Delete(employee) để xóa employee giống với employee đã khởi tạo

Cách 2:

Dùng session.Delete("employees/1") để xóa employee có mã employees/1

2. Truy vấn (Querying)

Các lệnh truy vấn sẽ được biên dịch thành ngôn ngữ truy vấn Raven Query Language (RQL).

```

1 using (IDocumentSession session = store.OpenSession()) // Mở 1 session tên database đã khai báo
2 {
3     List<string> productNames = session
4         .Query<Product>() // Truy vấn danh sách sản phẩm
5         .Where(x => x.UnitsInStock > 5) // Lọc các sản phẩm có UnitsInStock > 5
6         .Skip(0).Take(10) // Số trang
7         .Select(x => x.Name) // Select
8         .ToList(); // chuyển thành list
9 }

```

Truy vấn cơ bản

RavenDB Client hỗ trợ truy vấn sử dụng LINQ. Chức năng này được cho phép sử dụng phương thức truy vấn Session Query, là phương thức cơ bản và phổ biến nhất để truy vấn cơ sở dữ liệu.

Ví dụ 1:

Hãy thực thi truy vấn và trả về tất cả employees trong Northwind database. Để làm điều đó, chúng ta cần có document store và mở session rồi chỉ ra kiểu dữ liệu trả về mà chúng ta muốn truy vấn (trong trường hợp này là Employees) bằng cách thêm tham số Employee vào phương thức truy vấn:

```

1 // load all entities from 'Employees' collection
2 IList<Employee> results = session
3     .Query<Employee>()
4     .ToList(); // send query

```

Ví dụ 2: Filtering (Chọn lọc)

Để lọc kết quả, sử dụng phương thức LINQ thích hợp, như Where:

```

1 // load all entities from 'Employees' collection
2 // where 'FirstName' is 'Robert'
3 IList<Employee> results = session
4     .Query<Employee>()
5     .Where(x => x.FirstName == "Robert")
6     .ToList(); // send query

```

```

1 // load up entity from 'Employees' collection
2 // with ID matching 'employees/1-A'
3 Employee result = session
4     .Query<Employee>()
5     .Where(x => x.Id == "employees/1-A")
6     .FirstOrDefault(); // send query

```

Ví dụ 3: Paging (Phân trang)

Phân trang rất đơn giản. Các phương thức Take và Skip có thể được sử dụng:

```

1 // load up to 10 entities from 'Products' collection
2 // where there are more than 10 units in stock
3 // skip first 5 results
4 IList<Product> results = session
5     .Query<Product>()
6     .Where(x => x.UnitsInStock > 10)
7     .Skip(5)
8     .Take(10)
9     .ToList(); // send query

```

Ví dụ 4: Querying Specified Index (Truy vấn với index cụ thể)

Trong những ví dụ trên, chúng ta không chỉ ra index mà chúng ta muốn truy vấn. RavenDB sẽ cố gắng xác định index thích hợp hoặc tạo mới.

Để chỉ ra index, chúng ta cần thêm index làm tham số thứ 2 cho phương thức Query hoặc thêm index name làm tham số.

```

1 // load all entities from 'Employees' collection
2 // where 'FirstName' is 'Robert'
3 // using 'Employees/ByFirstName' index
4 IList<Employee> results = session
5     .Query<Employee, Employees_ByFirstName>()
6     .Where(x => x.FirstName == "Robert")
7     .ToList(); // send query

```

```

1 // load all entities from 'Employees' collection
2 // where 'FirstName' is 'Robert'
3 // using 'Employees/ByFirstName' index
4 IList<Employee> results = session
5     .Query<Employee>("Employees/ByFirstName")
6     .Where(x => x.FirstName == "Robert")
7     .ToList(); // send query

```

Các loại truy vấn

Session Query

Cách đơn giản nhất để truy vấn là sử dụng phương thức Query, nó hỗ trợ chúng ta sử dụng LINQ để truy vấn. Để làm quen với các khả năng truy vấn đặc biệt trên RavenDB, API Query cung cấp những phương thức mở rộng sẽ được đề cập sau đây.

Ví dụ 1:

```

1 // load all entities from 'Employees' collection
2 List<Employee> employees = session
3     .Query<Employee>()
4     .ToList();

```

Ví dụ trên không yêu cầu bạn chỉ ra index name. RavenDB sẽ tạo index tự động nếu cần thiết.

Employee là tham số kiểu tổng quát không chỉ định nghĩa kết quả trả về, mà nó còn cho biết tập kết quả truy vấn sẽ là Employees.

Ví dụ 2: LINQ Syntax

Method syntax

```

1 // load tất cả thực thể từ tập 'Employees'
2 // where FirstName == 'Robert'
3 List<Employee> employees = session
4     .Query<Employee>()
5     .Where(x => x.FirstName == "Robert")
6     .ToList();

```

Query syntax

```

1 // load tất cả thực thể từ tập 'Employees'
2 // where FirstName == 'Robert'
3 IRavenQueryable<Employee> query = from employee in session.Query<Employee>()
4                                     where employee.FirstName == "Robert"
5                                     select employee;
6
7 List<Employee> employees = query.ToList();

```

Raw Query

Truy vấn trong RavenDB sử dụng ngôn ngữ như SQL được gọi là RavenDB Query Language ([RQL](#)). Tất cả các truy vấn trên sinh ra RQL gửi đến server. Session cũng cho phép ta tăng tốc độ truy vấn bằng đến RQL bằng cách sử dụng phương thức RawQuery.

```

1 // load tất cả thực thể từ tập 'Employees'
2 // where FirstName == 'Robert'
3 List<Employee> employees = session
4     .Advanced.RawQuery<Employee>("from Employees where FirstName = 'Robert'")
5     .ToList();

```


Một số phương thức mở rộng dùng LINQ

Phương thức	Ý nghĩa
ofType<T>	Nhận kết quả trả về từ server và map chúng vào kiểu dữ liệu T
Group By	Gom nhóm
Include	Xử lý quan hệ giữa các documents
Select	Có thể chọn các fields từ document bạn muốn truy vấn hoặc các truy vấn phức tạp
MoreLikeThis	Trả về documents tương tự theo các tiêu chí và option đã cung cấp.

Tài liệu tham khảo

Tiếng Anh

[1] RavenDB documents

<https://ravendb.net/docs/article-page/4.1/csharp>

[2] Why I chose RavenDB my startup

<https://dzone.com/articles/why-i-chose-RavenDb-my-startup-0>

[3] RavenDB unrecoverable database and data loss

<https://ayende.com/blog/168420/RavenDb-unrecoverable-database-and-data-loss>

[4] MongoDB vs RavenDB

<https://db-engines.com/en/system/MongoDb;RavenDb>