33333

TRƯỜNG ĐẠI HỌC SƯ PHẠM THÀNH PHỐ HỒ CHÍ MINH

KHOA CÔNG NGHỆ THÔNG TIN

**--🕮--**

**MÔN: CƠ SỞ DỮ LIỆU NÂNG CAO**

**BÁO CÁO CUỐI KÌ MONGODB**



**Giảng viên hướng dẫn:** Ths.Lương Trần Hy Hiến

**Nhóm sinh viên thực hiện:**

1. Nguyễn Công Khanh - 43.01.104.075
2. Ngô Xuân Hải - 43.01.104.041
3. Tạ Hoàng Trí - 43.01.104.189
4. Hoàng Văn Quý - 43.01.104.143
5. Nguyễn Văn Hiếu - 43.01.104.049

TP.Hồ Chí Minh, ngày 21 tháng 11 năm 2019

Mục Lục

[1. Giới thiệu về MongoDB 1](#_Toc25876061)

[2. Các thuật ngữ hay sử dụng trong MongoDB 2](#_Toc25876062)

[3. So sánh giữa RDBMS (Relational database management system) và MongoDB 3](#_Toc25876063)

[4. Các kiểu dữ liệu trong MongoDB 3](#_Toc25876064)

[6. Thiết kế Schema trong MongoDB 6](#_Toc25876065)

[7. Relationship trong MongoDB 8](#_Toc25876066)

[8. Tham chiếu Database trong MongoDB 12](#_Toc25876067)

[9. Chỉ mục (Index) trong MongoDB 13](#_Toc25876068)

[10. Phương thức ensureIndex() trong MongoDB 13](#_Toc25876069)

[11. Covered Query trong MongoDB 15](#_Toc25876070)

[12. MongoDB hoạt động như thế nào: 16](#_Toc25876071)

[13. Lợi thế của MongoDB 17](#_Toc25876072)

[14. Khi nào NÊN sử dụng MongoDB? 19](#_Toc25876073)

[15. Khi nào KHÔNG NÊN sử dụng MongoDB? 19](#_Toc25876074)

[16. Các công cụ quản trị MongoDB 19](#_Toc25876075)

[17. Cài đặt MongoDB 19](#_Toc25876076)

[18. Một số ví dụ về mongo 20](#_Toc25876077)

[19. Truy vấn dữ liệu MongoDB trên ngôn ngữ C#/.NET: 24](#_Toc25876078)

[1. Cài đặt các gói NuGet: 24](#_Toc25876079)

[2. Tạo Collection: 25](#_Toc25876080)

[3. Tạo / Chèn dữ liệu: 27](#_Toc25876081)

[4. Phương thức FIND: 28](#_Toc25876082)

[5. Biểu thức LINQ: 30](#_Toc25876083)

[6. Phương thức SKIP: 30](#_Toc25876084)

[7. Phương thức SORT: 31](#_Toc25876085)

[20. MONGODB PROJECT NODEJS Example: 34](#_Toc25876086)

[21. Tổng kết và đánh giá: 43](#_Toc25876091)

[ Tổng kết 43](#_Toc25876092)

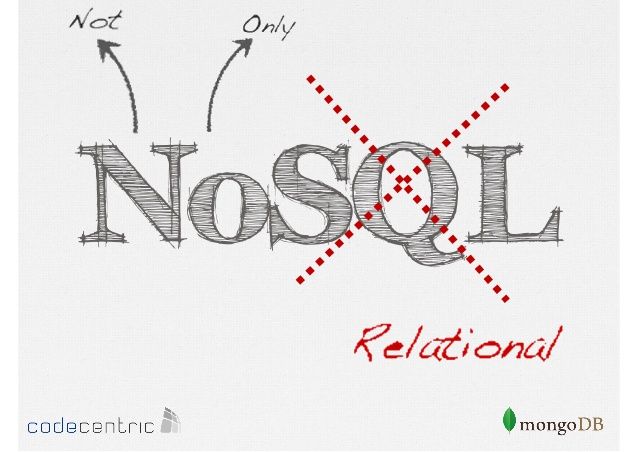
[ Đánh giá 43](#_Toc25876093)

# 1. Giới thiệu về MongoDB

MongoDB là một cơ sở dữ liệu mã nguồn mở và là cơ sở dữ liệu NoSQL hàng đầu, được hàng triệu người sử dụng. MongoDB được viết bằng C++ và được thiết kế theo kiểu hướng đối tượng trong đó các bảng được cấu trúc một cách linh hoạt cho phép các dữ liệu lưu trên bảng không cần phải tuân theo một dạng cấu trúc nhất định nào.

NoSQL là 1 dạng CSDL mã nguồn mở không sử dụng Transact-SQL để truy vấn thông tin.NoSQL viết tắt bởi: None-Relational SQL, hay có nơi thường gọi là Not-Only SQL.CSDL này được phát triển trên Javascript Framework với kiểu dữ liệu JSON. (Cú pháp của JSON là “key:value”)

Ngoài ra, MongoDB là một cơ sở dữ liệu đa nền tảng, hoạt động trên các khái niệm Collection và Document, nó cung cấp hiệu suất cao, tính khả dụng cao và khả năng mở rộng dễ dàng



# 2. Các thuật ngữ hay sử dụng trong MongoDB

**\_id** –

Là trường bắt buộc có trong mỗi document. Trường \_id đại diện cho một giá trị duy nhất trong document MongoDB. Trường \_id cũng có thể được hiểu là khóa chính trong document. Nếu bạn thêm mới một document thì MongoDB sẽ tự động sinh ra một \_id đại diện cho document đó và là duy nhất trong cơ sở dữ liệu MongoDB

**Collection** –

Là nhóm của nhiều document trong MongoDB. Collection có thể được hiểu là một bảng tương ứng trong cơ sở dữ liệu RDBMS (Relational Database Management System). Collection nằm trong một cơ sở dữ liệu duy nhất. Các collection không phải định nghĩa các cột, các hàng hay kiểu dữ liệu trước.

**Cursor** –

Đây là một con trỏ đến tập kết quả của một truy vấn. Máy khách có thể lặp qua một con trỏ để lấy kết quả.

**Database** –

Nơi chứa các Collection, giống với cơ sở dữ liệu RDMS chúng chứa các bảng. Mỗi Database có một tập tin riêng lưu trữ trên bộ nhớ vật lý. Một mấy chủ MongoDB có thể chứa nhiều Database.

**Document** –

Một bản ghi thuộc một Collection thì được gọi là một Document. Các Document lần lượt bao gồm các trường tên và giá trị.

**Field** –  
Là một cặp name – value trong một document. Một document có thể có không hoặc nhiều trường. Các trường giống các cột ở cơ sở dữ liệu quan hệ.

**JSON** –  
Viết tắt của JavaScript Object Notation. Con người có thể đọc được ở định dạng văn bản đơn giản thể hiện cho các dữ liệu có cấu trúc. Hiện tại JSON đang hỗ trợ rất nhiều ngôn ngữ lập trình.

# 3. So sánh giữa RDBMS (Relational database management system) và MongoDB

|  |  |
| --- | --- |
| RDBMS | MongoDB |
| Database | Database |
| Table | Collection |
| Tuple/Row | Document |
| Column | Field |
| Table Join | Embedded Documents |
| Primary Key | Primary Key (mặc định là \_id) |

# 

# 4. Các kiểu dữ liệu trong MongoDB

MongoDB hỗ trợ các kiểu dữ liệu sau:

|  |  |  |
| --- | --- | --- |
| Type | Number | Alias |
| Double | 1 | “double” |
| String | 2 | “String” |
| Object | 3 | “Object” |
| Array | 4 | “Array” |
| Binary data | 5 | “Bindata” |
| Undefined | 6 | “Undefined” |
| ObjectId | 7 | “ObjectId” |
| Boolean | 8 | “bool” |
| Date | 9 | “date” |
| Null | 10 | “null” |
| 32-bit integer | 16 | “int” |
| Timestamp | 17 | “Timestamp” |
| 64-bit integer | 18 | “long” |
| Min key | -1 | “minKey” |
| Max key | 123 | “maxkey” |

* **Chuỗi**: Đây là kiểu dữ liệu được sử dụng phổ biến nhất để lưu giữ dữ liệu. Chuỗi trong MongoDB phải là UTF-8 hợp lệ.
* **Số nguyên**: Kiểu dữ liệu này được sử dụng để lưu một giá trị số. Số nguyên có thể là 32 bit hoặc 64 bit phụ thuộc vào Server của bạn.
* **Boolean**: Kiểu dữ liệu này được sử dụng để lưu giữ một giá trị Boolean (true/false).
* **Double**: Kiểu dữ liệu này được sử dụng để lưu các giá trị số thực dấu chấm động.
* **Min/ Max keys**: Kiểu dữ liệu này được sử dụng để so sánh một giá trị với các phần tử BSON thấp nhất và cao nhất.
* **Mảng**: Kiểu dữ liệu này được sử dụng để lưu giữ các mảng hoặc danh sách hoặc nhiều giá trị vào trong một key.
* **Timestamp**: Giúp thuận tiện cho việc ghi chép hoặc đánh dấu thời điểm một Document được sửa đổi hoặc được thêm vào.
* **Object**: Kiểu dữ liệu này được sử dụng cho các Document được nhúng vào.
* **Null**: Kiểu dữ liệu này được sử dụng để lưu một giá trị Null.
* **Symbol**: Kiểu dữ liệu này được sử dụng giống như một chuỗi, tuy nhiên, nói chung nó được dành riêng cho các ngôn ngữ mà sử dụng kiểu symbol cụ thể.
* **Date**: Kiểu dữ liệu này được sử dụng để lưu giữ date và time hiện tại trong định dạng UNIX time. Bạn có thể xác định date time riêng cho bạn bằng việc tạo đối tượng Date và truyền ngày, tháng, năm vào trong đó.
* **Object ID**: Kiểu dữ liệu này được sử dụng để lưu giữ ID của Document.
* **Binary data**: Kiểu dữ liệu này được sử dụng để lưu giữ dữ liệu nhị phân.
* **Code**: Kiểu dữ liệu này được sử dụng để lưu giữ JavaScrip code vào trong Document.
* **Regular expression**: Kiểu dữ liệu này được sử dụng để lưu giữ Regular Expresion.

5. Một số câu lệnh dùng trong MongoDB

|  |  |
| --- | --- |
| Hiển thị database | show dbs |
| Tạo database | use database\_name |
| Xóa database | db.dropDatabase() |
| Tạo collection | db.createCollection(name, options) |
| Drop collection | db.COLLECTION\_NAME.drop() |
| Insert document | db.COLLECTION\_NAME.insert(document) |
|  | db.COLLECTION\_NAME.insertOne(document) |
|  | db.COLLECTION\_NAME.insertMany(document) |
| Truy vấn Document | db.COLLECTION\_NAME.find(query) |
|  | db.COLLECTION\_NAME.findOne(query) |
| Cập nhật Document | db.COLLECTION\_NAME.update(query, update, options) |
|  | db.COLLECTION\_NAME.updateOne(query, update, options) |
|  | db.COLLECTION\_NAME.updateMany(query, update, options) |
| Save | db.COLLECTION\_NAME.save({\_id:ObjectId(),NEW\_DATA}) |
| Xóa documnet | db.COLLECTION\_NAME.remove(query) |
| Giới hạn bản ghi | db.COLLECTION\_NAME.find().limit(NUMBER) |
| Sắp xếp bản ghi | db.COLLECTION\_NAME.find().sort({KEY:1}) |
| Tạo chỉ mục (index) | db.COLLECTION\_NAME.ensureIndex({KEY:1}) |
| …. |  |

+ query: tiêu chí để update.

+ update: document muốn update.

+ options: các document khác không bắt buộc.

Truy vấn trong MongoDB mà tương đương mệnh đề WHERE trong RDBMS

|  |  |
| --- | --- |
| **Phép toán** | **Cú pháp** |
| Bằng(Equality) | {key: value} |
| Nhỏ hơn (Less Than) | {key: {$lt: value}} |
| Nhỏ hơn bằng (Less Than Equals) | {key: {$lte: value}} |
| Lơn hơn (Greater Than) | {key: {$gt: value}} |
| Lớn hơn bằng (Greater Than Equals) | {key: {$gte: value}} |
| Khác (Not Equals) | {key: {$ne: value}} |
| Trong ( In) | {key: {$in: [value1, value2,..]}} |
| Không Thuộc (Not In) | {key: {$nin: [value1, value2,..]}} |

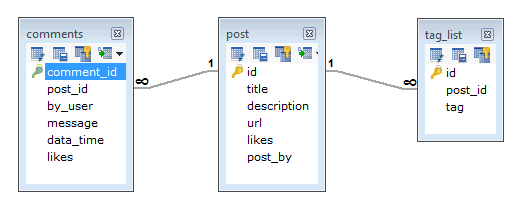
# 

# 6. Thiết kế Schema trong MongoDB

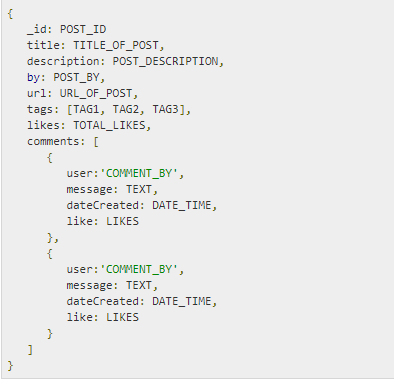
Giả sử, một khách hàng cần một thiết kế cơ sở dữ liệu cho trang blog của anh ta, và dưới đây, bạn xem các điểm khác nhau giữa thiết kế Schema của RDBMS và MongoDB. Website này có các yêu cầu sau:

* Mỗi post có tiêu đề, miêu tả và Url duy nhất.
* Mỗi post có thể có một hoặc nhiều tags.
* Mỗi post có tên người đăng và tổng số like.
* Mỗi post có các comment được cung cấp bởi người dùng cùng với tên, thông điệp, thời gian, và like của họ.
* Trên mỗi post, có thể có 0 hoặc nhiều comment.

Trong thiết kế Schema của RDBMS cho các yêu cầu trên sẽ có tối thiểu ba bảng dữ liệu:



Trong khi trong thiết kế Schema của MongoDB sẽ chỉ có một Collection Post có cấu trúc như sau:



# 7. Relationship trong MongoDB

Relationship trong MongoDB tượng trưng cho cách các Document có mối liên quan với nhau. Relationship có thể được mô hình hóa thông qua phương thức Embeded và Referenced. Những Relationship này có thể là 1:1, 1:N, N:1, hoặc N:N.

Chúng ta cùng xem xét trường hợp lưu giữ địa chỉ của người dùng. Một người dùng có thể có nhiều địa chỉ, điều này tạo ra một 1:N Relationship.

Mô hình hóa Embeded Relationships



Phương pháp này duy trì tất cả dữ liệu có liên quan trong một Document đơn, điều này giúp cho việc lấy và duy trì dữ liệu dễ dàng. Toàn bộ Document có thể được lấy trong một truy vấn đơn, giống như:

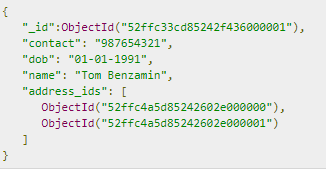
>db.users.findOne({"name":"Tom Benzamin"},{"address":1})

Ghi chú rằng, trong truy vấn trên, **db** và **users** tương ứng là Database và Collection.

Điểm hạn chế ở đây là, nếu Document được nhúng tiếp tục tăng kích cỡ quá nhiều, nó sẽ ảnh hưởng đến hiệu suất đọc/ghi.

Mô hình hóa Referenced Relationship

Đây là phương pháp thiết kế Relationship tiêu chuẩn hóa. Trong phương pháp này, cả user và address document sẽ vẫn được duy trì một cách riêng rẽ, nhưng user document sẽ chứa một trường mà sẽ tham chiếu đến trường id của address document.



Như trên, user document chứa trường **address\_ids**, mà chứa ObjectIds của địa chỉ tương ứng. Sử dụng các ObjectIds này, chúng ta có thể truy vấn address document và lấy chi tiết địa chỉ từ đó. Với hướng tiếp cận này, chúng ta sẽ cần hai truy vấn: đầu tiên lấy các trường **address\_ids** từ **user** document và sau đó là lấy các địa chỉ này từ **address** collection.



Aggregation trong MongoDB

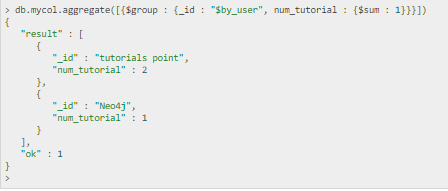
Aggregation có thể hiểu là sự tập hợp. Các **Aggregation** operation xử lý các bản ghi dữ liệu và trả về kết quả đã được tính toán. Các phép toán tập hợp nhóm các giá trị từ nhiều Document lại với nhau, và có thể thực hiện nhiều phép toán đa dạng trên dữ liệu đã được nhóm đó để trả về một kết quả duy nhất. Trong SQL, count(\*) và GROUP BY là tương đương với Aggregation trong MongoDB

Cú pháp cơ bản của phương thức aggregate() là như sau:

>db.COLLECTION\_NAME.aggregate(AGGREGATE\_OPERATION)



Sử dụng như sau



Các biểu thức



pipeline trong MongoDB

shell pipeline nghĩa là khả năng để thực thi một hoạt động trên một số input và sử dụng output như là input cho lệnh tiếp theo, và …. MongoDB cũng hỗ trợ cùng khái niệm pipeline đó trong Aggregation Framework.

Các giai đoạn có thể có trong Aggregation Framework là:

* $project: Được sử dụng để chọn một số trường cụ thể từ một Collection.
* $match: Đây là một hoạt động lọc và vì thế nó có thể giảm số Document mà được cung cấp như là input cho giai đoạn kế tiếp.
* $group: Thực hiện Aggregation thực sự, như đã trình bày ở trên.
* $sort: Sắp xếp các Document.
* $skip: Nhảy qua số Document đã cung cấp.
* $limit: Giới hạn số Document.
* $unwind: Được sử dụng để chia một Document đang sử dụng mảng thành nhiều Document. Sử dụng hoạt động này sẽ tạo một số lượng Document cho bước tiếp theo.

# 8. Tham chiếu Database trong MongoDB

**Referenced Relationship**, còn được gọi là **Manual References**, trong đó chúng ta thao tác để lưu giữ id của các Document được tham chiếu bên trong Document khác. Tuy nhiên, trong các trường hợp mà một Document chứa nhiều tham chiếu từ các Collection khác nhau, thì chúng ta sử dụng **DBRefs** trong MongODB

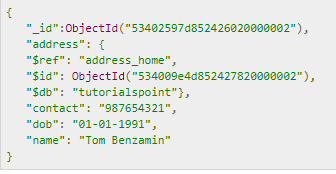
Có 3 trường trong DBRefs:

**$ref:** Trường này xác định Collection của Document được tham chiếu.

**$id:** Trường này xác định trường \_id của Document được tham chiếu.

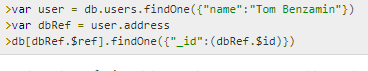
**$db:** Trường này là một trường tùy ý, chứa tên của Database mà Document được tham chiếu ở trong đó.

Giả sử một user document có trường **address** dạng DBRefs như sau

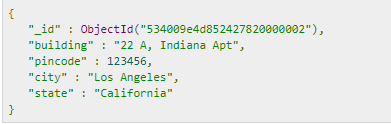


Trường address dạng DBRefs ở đây xác định rằng address document được tham chiếu ở trong address\_home collection dưới tutorialspoint database và có một id là: 534009e4d852427820000002.

Phần code sau sẽ nhìn vào Collection được xác định bởi tham số **$ref** (là address\_home trong trường hợp này) cho một Document với id được xác định bởi tham số **$id** trong DBRefs.



Code trên trả về address document sau đây, mà có mặt trong **address\_home** collection:



# 9. Chỉ mục (Index) trong MongoDB

Chỉ mục (Index) hỗ trợ việc phân giải các truy vấn hiệu quả hơn. Nếu không có chỉ mục, MongoDB phải quét qua mọi Document của một Collection để chọn các Document mà kết nối với lệnh truy vấn. Việc quét này có thể không hiệu quả và yêu cầu MongoDB xử lý một số lượng lớn dữ liệu.

Chỉ mục (Index) là các cấu trúc dữ liệu đặc biệt, lưu giữ một phần nhỏ của tập hợp dữ liệu, giúp việc "vọc" vào Collection một cách dễ dàng hơn. Chỉ mục lưu giữ giá trị của một trường cụ thể hoặc tập hợp các trường, được sắp xếp bởi giá trị của trường như đã được xác định trong chỉ mục.

# 10. Phương thức ensureIndex() trong MongoDB

Để tạo một chỉ mục, bạn cần sử dụng phương thức ensureIndex() của MongoDB.

Cú pháp

Cú pháp cơ bản của phương thức ensureIndex() là như sau:

>db.COLLECTION\_NAME.ensureIndex({KEY:1})

Ở đây, key là tên của trường mà bạn muốn tạo chỉ mục và 1 là cho thứ tự tăng dần. Để tạo chỉ mục theo thứ tự giảm dần, bạn cần sử dụng -1.

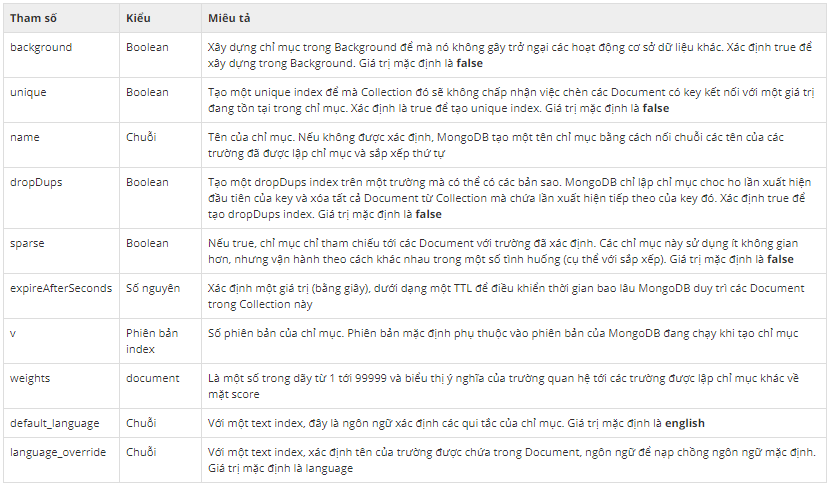
Ví dụ

>db.mycol.ensureIndex({"title":1})

Trong phương thức **ensureIndex()**, bạn có thể truyền nhiều trường, để tạo chỉ mục trên nhiều trường, bạn sử dụng:

>db.mycol.ensureIndex({"title":1,"description":-1})

Phương thức ensureIndex() cũng chấp nhận danh sách các tùy chọn tùy ý, được liệt kê dưới đây:



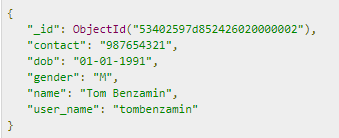
# 11. Covered Query trong MongoDB

Trong MongoDB Documentation chính thức, một Covered Query là một truy vấn mà trong đó:

* Tất cả các trường trong truy vấn là một phần của một chỉ mục và
* Tất cả các trường được trả về trong truy vấn là trong cùng chỉ mục.

Khi tất cả các trường có mặc trong truy vấn là một phần của chỉ mục, MongoDB kết nối các điều kiện truy vấn và trả về kết quả bởi sử dụng cùng chỉ mục đó mà không nhìn vào bên trong Document. Khi các chỉ mục có mặt trong RAM, việc lấy dữ liệu từ các chỉ mục là nhanh hơn khi so sánh với khi lấy dữ liệu bằng cách quét toàn bộ các Document.

Để nghiên cứu về Covered Query, bạn theo dõi Document sau trong **user** collection



Đầu tiên, chúng ta tạo một chỉ mục phức hợp cho users collection trên các trường gender và user\_name bởi sử dụng truy vấn sau:

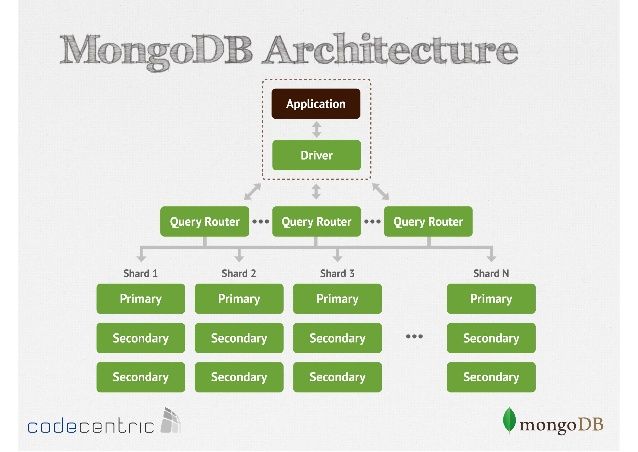


Bây giờ, chỉ mục này sẽ bao phủ truy vấn sau:



Có thể nói rằng, với truy vấn trên, MongoDB sẽ không đi vào tìm kiếm các Document trong cơ sở dữ liệu. Thay vào đó, nó sẽ lấy dữ liệu cần thiết từ dữ liệu đã lập chỉ mục, điều này sẽ giúp tiến trình xảy ra nhanh hơn rất nhiều.

# 12. MongoDB hoạt động như thế nào:

Ưu điểm

MongoDB hoạt động dưới một tiến trình ngầm service, luôn mở một cổng (Cổng mặc định là 27017) để lắng nghe các yêu cầu truy vấn, thao tác từ các ứng dụng gửi vào sau đó mới tiến hành xử lý.

Mỗi một bản ghi của MongoDB được tự động gắn thêm một field có tên “\_id” thuộc kiểu dữ liệu ObjectId mà nó quy định để xác định được tính duy nhất của bản ghi này so với bản ghi khác, cũng như phục vụ các thao tác tìm kiếm và truy vấn thông tin về sau.  
Trường dữ liệu “\_id” luôn được tự động đánh index (chỉ mục) để tốc độ truy vấn thông tin đạt hiệu suất cao nhất.

Mỗi khi có một truy vấn dữ liệu, bản ghi được cache (ghi đệm) lên bộ nhớ Ram, để phục vụ lượt truy vấn sau diễn ra nhanh hơn mà không cần phải đọc từ ổ cứng.

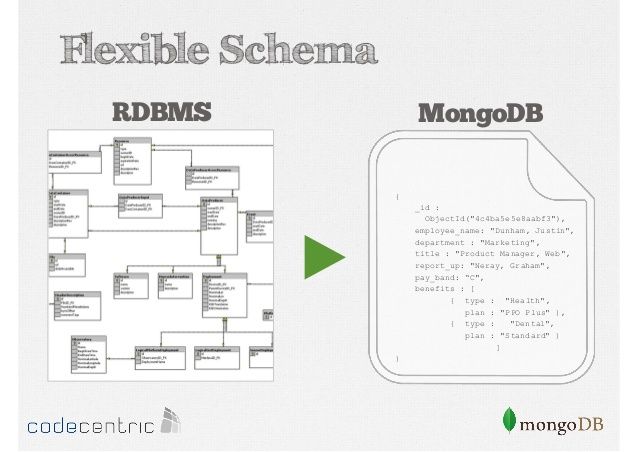
Khi có yêu cầu thêm/sửa/xóa bản ghi, để đảm bảo hiệu suất của ứng dụng mặc định MongoDB sẽ chưa cập nhật xuống ổ cứng ngay, mà sau 60 giây MongoDB mới thực hiện ghi toàn bộ dữ liệu thay đổi từ RAM xuống ổ cứng.

Nhược điểm

Dữ liệu được caching, lấy RAM làm trọng tâm hoạt động vì vậy khi hoạt động yêu cầu một bộ nhớ RAM lớn

Như đã giới thiệu ở trên, mọi thay đổi về dữ liệu mặc định đều chưa được ghi xuống ổ cứng ngay lập tức vì vậy khả năng bị mất dữ liệu từ nguyên nhân mất điện đột xuất là rất cao

# 13. Lợi thế của MongoDB

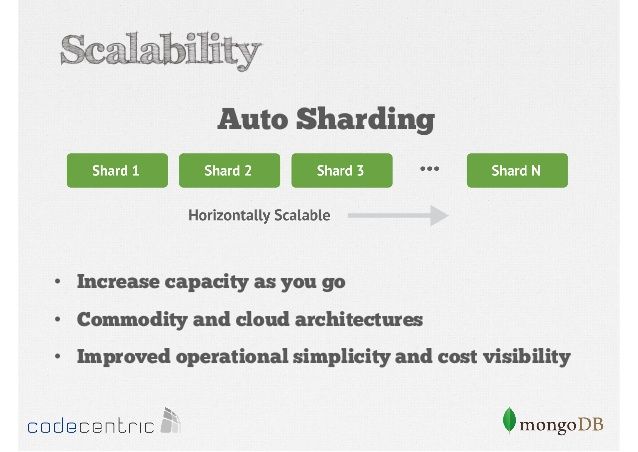


Ít schema hơn: Vì schema được sinh ra là để nhóm các đối tượng vào 1 cụm, dễ quản lý. Ví dụ như tạo 1 schema tên là Students chẳng hạn thì chỉ có những gì liên quan đến student thì mới được cho vào schema này. Trong khi đó trong mongodb thì chỉ 1 collection ta có thể chứa nhiều document khác nhau . Với mỗi document thì số trường, nội dung, kích thước lại có thể khác nhau.

Cấu trúc của một đối tượng rõ ràng.

Không có các Join phức tạp.

Khả năng mở rộng cực lớn: việc mở rộng dữ liệu mà không phải lo đến các vấn đề như khóa ngoại, khóa chính, kiểm tra ràng buộc, ... MongoDB cho phép thực hiện replication và sharding nên việc mở rộng cũng thuận lợi hơn.



Sử dụng bộ nhớ trong để lưu giữ cửa sổ làm việc cho phép truy cập dữ liệu nhanh hơn. Việc cập nhật được thực hiện nhanh gọn nhờ update tại chỗ (in-place).

# 14. Khi nào NÊN sử dụng MongoDB?

Sử dụng MongoDB trong trường hợp:

Nếu website của bạn có tính chất INSERT cao  
Bởi vì mặc định MongoDB có sẵn cơ chế ghi với tốc độ cao và an toàn.Website của bạn ở dạng thời gian thực nhiều, nghĩa là nhiều người thao tác với ứng dung. Nếu trong quá trình load bị lỗi tại một điểm nào đó thì nó sẽ bỏ qua phần đó nên sẽ an toàn.

Website bạn có nhiều dữ liệu quá  
Giả sử web bạn có đến 10 triệu records thì đó là cơn ác mộng với MYSQL. Bởi vì MongoDB có khả năng tìm kiến thông tin liên quan cũng khá nhanh nên trường hợp này nên dùng nó.

Máy chủ không có hệ quản trị CSDL  
Trường hợp này thường bạn sẽ sử dụng SQLITE hoặc là MongoDB.

# 15. Khi nào KHÔNG NÊN sử dụng MongoDB?

Các ứng dụng cần sử dụng nhiều transaction (như ngân hàng) do Mongodb không có cơ chế transaction (giao dịch) để phục vụ cho các ứng dụng ngân hàng

Các ứng dụng cần SQL (sử dụng joins).

# 16. Các công cụ quản trị MongoDB

RoboMongo

MongoExplorer

Mongodb Compass

# 17. Cài đặt MongoDB

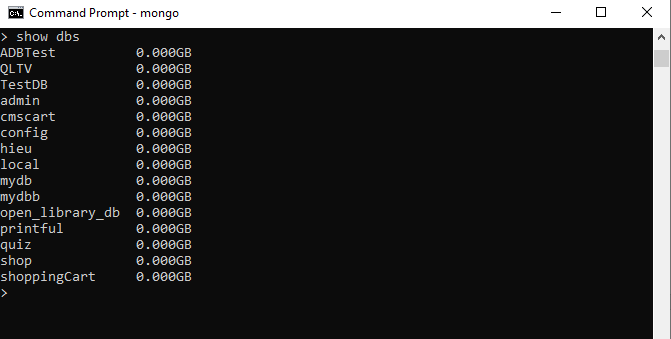
Trên Windows:

1. Tải MongoDB từ link bên dưới:  
   <https://www.mongodb.org/downloads>
2. Chạy file vừa tải về
3. Sau khi chạy xong file, vào thư mục C:\Program Files\MongoDB\Server\3.4\bin , chạy file mongod để khởi động mongoDB
4. Dùng địa chỉ 127.0.0.1:27017 để tạo connection đến mongoDB

Chi tiết tại: <https://o7planning.org/vi/10265/huong-dan-cai-dat-va-cau-hinh-database-mongodb>

# 18. Một số ví dụ về mongo

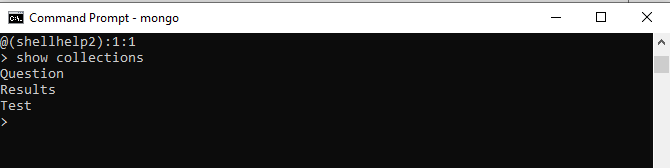
Show db



Tạo db ( lưu ý phải tạo ít nhất 1 collection thì mới hiển thị db)



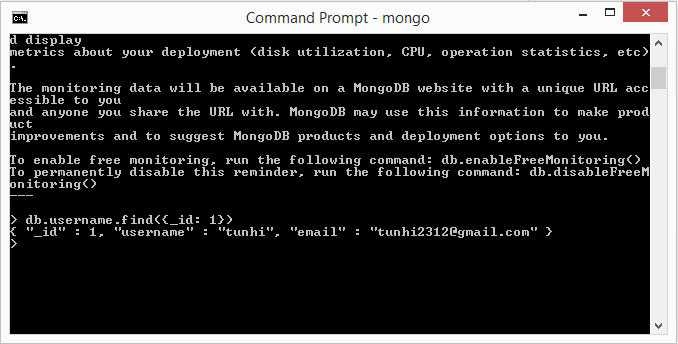
Show collection



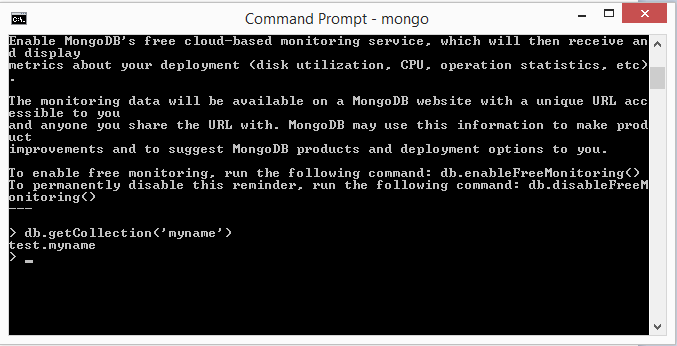
Find



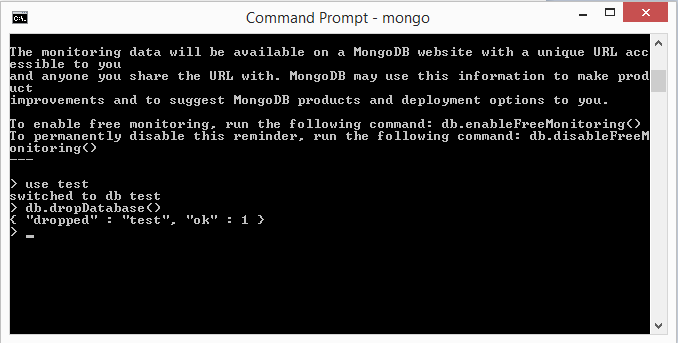
Find có điều kiện



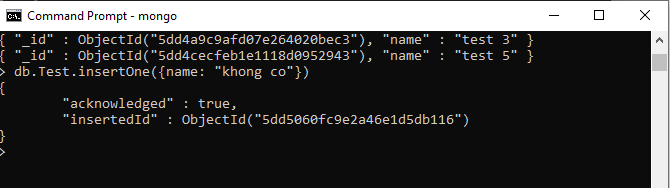
Get collection



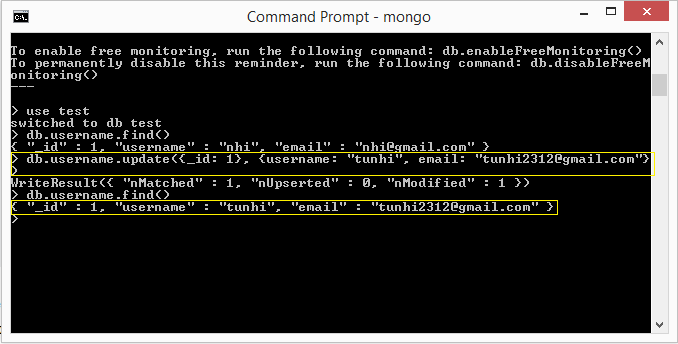
Drop database

**

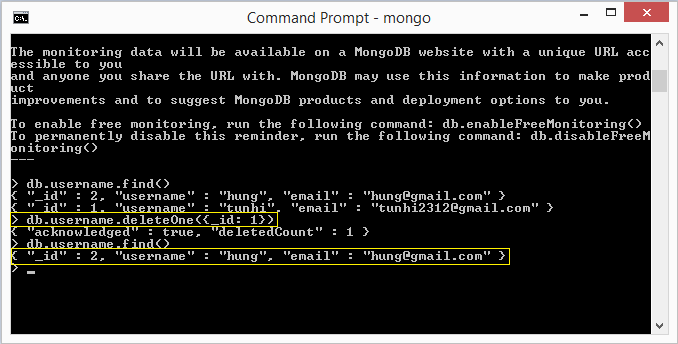
Insert



Update



Xóa



19. Truy vấn dữ liệu MongoDB trên ngôn ngữ C#/.NET:

1. Cài đặt các gói NuGet:

*Để cài đặt trình điều khiển, chúng ta cần cài đặt các gói NuGet cần thiết:*

*MongoDB.Bson* : Thư viện BSON độc lập xử lý việc chuyển đổi các POCO thành các kiểu BSON (là định dạng tệp cho MongoDB) và ngược lại.

*MongoDB.Driver.Core*: Đây là một driver của chính nó và có các thành phần cốt lõi của driver (như cách kết nối với mongoDB, kết nối tổng hợp, và các kết nối khác) để giao tiếp từ .Net đến MongoDB và ngược lại, có sự phụ thuộc vào MongoDB.Bson.

*MongoDB.Driver* : Phụ thuộc vào Driver.Core, thư viện mà phụ thuộc vào MongoDB.Bson. Nó dễ sử dụng API hơn các thành phần của driver, có các phương thức async và hỗ trợ truy vấn với LINQ.

Chạy lệnh sau để có được tất cả ba gói được cài đặt cùng một lúc:

***Install-Package******MongoDB.Driver***

Để kết nối với một cơ sở dữ liệu, chúng ta sử dụng lớp MongoClient để truy cập vào mongoDB và thông qua nó, chọn cơ sở dữ liệu chúng ta muốn sử dụng. Lớp này có bốn hàm tạo.

- Một contructor không tham số theo mặc định sẽ kết nối trên cổng 27017:

***var*** *client =* ***new*** *MongoClient();*

- Truyền tham số chuỗi kết nối:

***var*** *connectionString = "mongodb://localhost:27017";*

***var*** *client =* ***new*** *MongoClient(connectionString);*

- Sử dụng MongoUrl tương tự như sử dụng hàm tạo connectionstring có tham số. Bạn có thể tạo hàm này bằng cách sử dụng hàm tạo mặc định:

***var*** *client =* ***new*** *MongoClient(****new*** *MongoUrl("mongodb://localhost:27017"));*

-Sử dụng phương thức Create (tĩnh) từ lớp Client:

***var*** *client =* ***new*** *MongoClient(MongoUrl.Create("mongodb://localhost:27017"));*

Với MongoClient, có nhiều thứ chúng ta có thể thao tác như xóa một cơ sở dữ liệu, lấy một cơ sở dữ liệu, hoặc lấy tên một trong các cơ sở dữ liệu trên máy chủ. Không khó khăn để tạo một cơ sở dữ liệu vì một khi bạn chọn một cơ sở dữ liệu và chèn dữ liệu vào nó, nó sẽ tự động tạo ra cơ sở dữ liệu.

Phương pháp GetDatabase sẽ tự động tạo cơ sở dữ liệu theo cú pháp:

*IMongoDatabase db = client.GetDatabase("Database\_name");*

Với:Database\_name là tên cơ sở dữ liệu bạn muốn.

Các phương thức GetDatabase sẽ trả về một đối tượng là một đại diện của một cơ sở dữ liệu, từ đó chúng ta có thể truy cập vào các Collection khác nhau và thao tác với cơ sở dữ liệu. Với đối tượng trên cơ sở dữ liệu, bạn có thể tạo, đổi tên, truy xuất hoặc nhận danh sách các Collection từ cơ sở dữ liệu. Tài liệu được lưu trữ trong Collection, vì vậy bạn có thể nghĩ về một Collection dưới dạng bảng và tài liệu dưới dạng bản ghi trong bảng, nếu bạn quen sử dụng SQL.

2. Tạo Collection:

Để tạo một Collection, chúng ta sử dụng phương thức CreateCollection hoặc phương thức CreateCollectionAsync của đối tượng ImongoDatabase. Phương thức này có ba tham số (trong đó hai tham số cuối là không bắt buộc): tên của Collection, khởi tạo Collection options, CancellationToken. Cú pháp như sau:

***void******CreateCollection****(*

***string*** *name,*

*CreateCollectionOptions options = null,*

*CancellationToken cancellationToken = null*

*)*

*Task* ***CreateCollectionAsync****(*

***string*** *name,*

*CreateCollectionOptions options = null,*

*CancellationToken cancellationToken = null*

*)*

Bạn có thể tạo một Collection theo mặc định, bằng cách chỉ thiết lập tên của Collection:

***await*** *db.CreateCollectionAsync("Collection\_name");*

Với: Collection\_name là tên của Collection.

CreateCollectionOptions chỉ định các cài đặt cho một Collection, ví dụ như số lượng tối đa của dữ liệu nó chứa. Đây là một ví dụ:

***await*** *db.CreateCollectionAsync("students",* ***new*** *CreateCollectionOptions*

*{*

*AutoIndexId = false,*

*MaxDocuments = 25,*

*Capped = true*

*});*

GetCollection:

Sau khi tạo một Collection, nó sẽ yêu cầu một bước kiểm tra xem Collection có tồn tại hay không, và sau đó đổ dữ liệu vào Collection. GetCollection tự động tạo ra một Collection nếu không tồn tại và bổ sung dữ liệu vào Collection đó. Cách gọi phương thức GetCollection:

***static******async*** *Task* ***MainAsync****()*

*{*

*......*

*IMongoCollection<BsonDocument> collection = db.GetCollection<BsonDocument>("Collection\_name");*

*}*

Với Collection\_name là tên của Collection.

BsonDocument là một dạng của MongoDB.Bson package, thư viện đại diện cho một tài liệu BSON và với dạng này, chúng ta có thể làm việc với bất kỳ dạng dữ liệu nào. Gói này chứa tất cả các dạng BSON cơ bản và một vài thứ khác để làm việc với BSON.

Trong gói này, chúng ta có các lớp đại diện cho các kiểu BSON và cách ánh xạ giữa các kiểu .NET và BsonValues. Một vài trong số đó là:

- BsonDocument như trong hàm gọi GetCollection trên.

- BsonElement đại diện cho một phần tử BSON.

- BsonValue là một lớp cơ sở trừu tượng được sử dụng bởi các lớp con khác nhau như BsonString, BsonInt32 và nhiều thứ khác.

BsonDocument là từ điển của chuỗi giá trị BSON, vì vậy chúng ta có thể khởi tạo như sau:

***var*** *document =* ***new*** *BsonDocument*

*{*

*{"firstname", BsonValue.Create("Peter")},*

*{"lastname",* ***new*** *BsonString("Mbanugo")},*

*{ "subjects",* ***new*** *BsonArray(****new****[] {"English", "Mathematics", "Physics"}) },*

*{ "class", "JSS 3" },*

*{ "age",* ***int****.MaxValue }*

*};*

hoặc sử dụng phương pháp Add có một số tình trạng quá tải:

***var*** *document =* ***new*** *BsonDocument();*

*document.Add("name", "Steven Johnson");*

*document.Add("age", 23);*

*document.Add("subjects",* ***new*** *BsonArray() {"English", "Mathematics", "Physics"});*

*hoặc sử dụng indexer:*

*document["class"] = "JSS 3";*

3. Tạo / Chèn dữ liệu:

Các dữ liệu được lưu trữ trong một Collection. Mongo Collection cung cấp các phương thức để chèn một dữ liệu duy nhất tại một thời điểm hoặc nhiều dữ liệu cùng một lúc. Để làm điều này, chúng ta phải:

Tạo một đối tượng kiểu Imongocollection để đại diện cho Collection mà chúng ta muốn thao tác:

***var*** *collection = db.GetCollection<BsonDocument>("Collection\_name");*

Với Collection\_name là tên của Collection.

Tiếp theo tạo dữ liệu mà bạn muốn chèn.

Ví dụ:

***var*** *document =* ***new*** *BsonDocument*

*{*

*{"firstname", BsonValue.Create("Peter")},*

*{"lastname",* ***new*** *BsonString("Mbanugo")},*

*{ "subjects",* ***new*** *BsonArray(****new****[] {"English", "Mathematics", "Physics"}) },*

*{ "class", "JSS 3" },*

*{ "age", 45}*

*};*

Và cuối cùng chèn dữ liệu:

Sử dụng ***await*** *collection.InsertOneAsync(document);*

Hoặc *collection.InsertOne(document);*

Bạn cũng có thể chèn nhiều dữ liệu cùng một lúc bằng cách sử dụng phương thức InsertMany hoặc phương thức InsertManyAsync. Giả sử chúng ta có ba học sinh mới trong trường, chúng ta có thể chèn tất cả cùng một lúc bằng cách sử dụng phương thức này, chúng sẽ được chèn vào cùng một lúc.

***await*** *collection.InsertManyAsync(threestudent);*

4. Phương thức FIND:

Phương thức này trả về một giao diện IFindFluent. Đây là một giao diện cung cấp cho chúng ta cú pháp đơn giản cho những thao tác như như Count, Skip, Sort, và Limit. Từ IfindFluent chúng ta cũng có thể trả về một con trỏ (bằng cách gọi ToCursor hoặc ToCursorAsync) hoặc một List (bằng cách gọi ToList hoặc ToListAsync). Chúng ta có thể in tất cả các dữ liệu bằng cách sử dụng phương thức Find, Ví dụ:

*await collection.Find(FilterDefinition<BsonDocument>.Empty)*

*.ForEachAsync(doc => Console.WriteLine(doc));*

Tìm dữ liệu cụ thể:

Thay vì truy xuất tất cả dữ liệu, chúng ta có thể chỉ định truy vấn một dữ liệu phù hợp với Find. Có thể truy vấn một dữ liệu cụ thể bằng các cách sau:

- Sử dụng BsonDocument hoặc chuỗi

Chúng ta có thể định nghĩa một BsonDocument như một bộ lọc hay truy vấn, phương thức này sẽ tìm các dữ liệu khớp với các trường được định nghĩa trong chương trình. Ví dụ, tìm học sinh có tên “Peter”:

+ Sử dụng BsonDocument:

*var filter = new BsonDocument("FirstName", "Peter");*

*await collection.Find(filter)*

*.ForEachAsync(document => Console.WriteLine(document));*

+ Sử dụng chuỗi:

*var filter = "{ FirstName: 'Peter'}";*

*await collection.Find(filter)*

*.ForEachAsync(document => Console.WriteLine(document));*

- Sử dụng FilterDefinitionBuilder:

Bạn có thể sử dụng FilterDefinitionBuilder, một Builder cho FilterDefinition. Nó cung cấp một bộ các phương thức để xây dựng các truy vấn, và Lt, là một trong số các phương thức. Chúng ta có thể truy vấn bằng cách sử dụng FilterDefinitionBuilder như sau:

***var*** *filter =* ***new*** *FilterDefinitionBuilder<BsonDocument>().Lt("age", 25);*

Hoặc sử dụng phương thức overload nhận một biểu thức LINQ:

***var*** *filter =* ***new*** *FilterDefinitionBuilder<Student>().Lt( student => student.Age, 25);*

Ngoài ra, bạn có thể sử dụng một Builders lớp tĩnh để xây dựng một định nghĩa bộ lọc và lớp này cũng có các phương thức trợ giúp tĩnh để xây dựng các thứ khác, như định nghĩa tham chiếu, định nghĩa sắp xếp và một số thứ khác.

*var filter = Builders<BsonDocument>.Filter.Lt("age", 25);*

*var filter = Builders<Student>.Filter.Lt(student => student.Age, 25);*

Driver cũng quá tải 3 toán tử cho định nghĩa bộ lọc. Gồm các toán tử and (&), or (|) và not (!). Ví dụ, chúng tôi muốn học sinh có độ tuổi dưới 25 và tên là Peter, chúng ta có thể xây dựng truy vấn bằng cách sử dụng builder helper và quá tải toán tử & như sau:

var builder = Builders<BsonDocument>.Filter;

var filter = builder.Lt("Age", 40) & builder.Eq("FirstName", "Peter");

5. Biểu thức LINQ:

Các quá tải toán tử trong các phương thức trên có biểu thức LINQ và khi chúng ta có một đối tượng, chúng ta có thể xây dựng một truy vấn bằng cách sử dụng biểu thức LINQ. Ví dụ, chúng ta muốn truy xuất những sinh viên có độ tuổi dưới 25 và tên đầu tiên không phải là Peter và in ra họ và tên của họ:

***var*** *collection = db.GetCollection<Student>("students");*

***await*** *collection.Find(student => student.Age < 25 && student.FirstName != "Peter")*

*.ForEachAsync(student => Console.WriteLine(student.FirstName + " " + student.LastName));*

Với MongoDB, bạn có thể giới hạn số lượng dữ liệu bằng cách gọi Limit phương thức của IfindFluent trả về từ việc gọi Find. Ví dụ, truy sinh viên ở độ tuổi dưới 40, giới hạn kết quả của dữ liệu tối đa là hai sinh viên, chúng ta gọi hàm Limit() với giá trị là 2:

***int*** *count = 1;*

***await*** *collection.Find(x => x.Age < 40)*

*.Limit(2)*

*.ForEachAsync(*

*student =>*

*{*

*Console.WriteLine($"S/N: {count} \t Id: {student.Id}, FirstName: {student.FirstName}, LastName: {student.LastName}");*

*count++;*

*});*

6. Phương thức SKIP:

Nếu chúng ta muốn cơ sở dữ liệu bỏ qua một số tài liệu, chúng ta sử dụng phương thức Skip.Ví dụ, bỏ qua tất cả sinh viên với độ tuổi dưới 40:

***int*** *count = 1;*

***await*** *collection.Find(x => x.Age < 40)*

*.Skip(1)*

*.ForEachAsync(*

*student =>*

*{*

*Console.WriteLine($"S/N: {count} \t Id: {student.Id}, FirstName: {student.FirstName}, LastName: {student.LastName}");*

*count++;*

*});*

7. Phương thức SORT:

Các phương thức Sort của giao diện cần SortDefinition, mặc nhiên sẽ chuyển đổi từ chuỗi hoặc BsonDocument giống như FilterDefinition. Vì vậy, nếu chúng ta muốn sắp xếp LastName theo một thứ tự tăng dần bằng cách sử dụng một chuỗi như định nghĩa:

***await*** *collection.Find(FilterDefinition<Student>.Empty)*

*.Skip((currentPage - 1) \* pageSize)*

*.Limit(pageSize)*

*.Sort("{LastName: 1}")*

*.ForEachAsync(*

*student =>*

*{*

*Console.WriteLine($"S/N: {count}, \t Id: {student.Id}, FirstName: {student.FirstName}, LastName: {student.LastName}, Age: {student.Age}");*

*count++;*

*});*

Và nếu chúng ta muốn sắp xếp LastName theo thứ tự giảm dần bằng cách sử dụng một BsonDocument:

***await*** *collection.Find(FilterDefinition<Student>.Empty)*

*.Skip((currentPage - 1) \* pageSize)*

*.Limit(pageSize)*

*.Sort(****new*** *BsonDocument("LastName", -1))*

*.ForEachAsync(*

*student =>*

*{*

*Console.WriteLine($"S/N: {count}, \t Id: {student.Id}, FirstName: {student.FirstName}, LastName: {student.LastName}, Age: {student.Age}");*

*count++;*

*});*

Chúng ta cũng có thể sử dụng SortDefinitionBuilder. Vì vậy, chúng tôi có thể sử dụng builder helper để tạo hàm sắp xếp như sau:

***await*** *collection.Find(FilterDefinition<Student>.Empty)*

*.Skip((currentPage - 1) \* pageSize)*

*.Limit(pageSize)*

*.Sort(Builders<Student>.Sort.Descending("LastName"))*

*.ForEachAsync(*

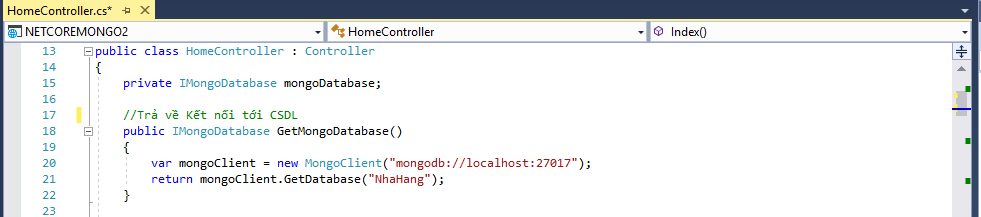
*student =>*

*{*

*Console.WriteLine($"S/N: {count}, \t Id: {student.Id}, FirstName: {student.FirstName}, LastName: {student.LastName}, Age: {student.Age}");*

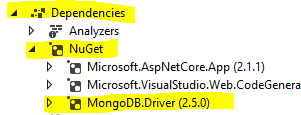
*count++;*

*});*



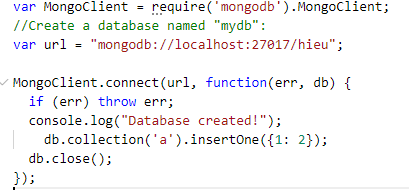
*Tạo kết nối với mongodb*

3*Hai thư viện được cài trong thư mục NuGet*



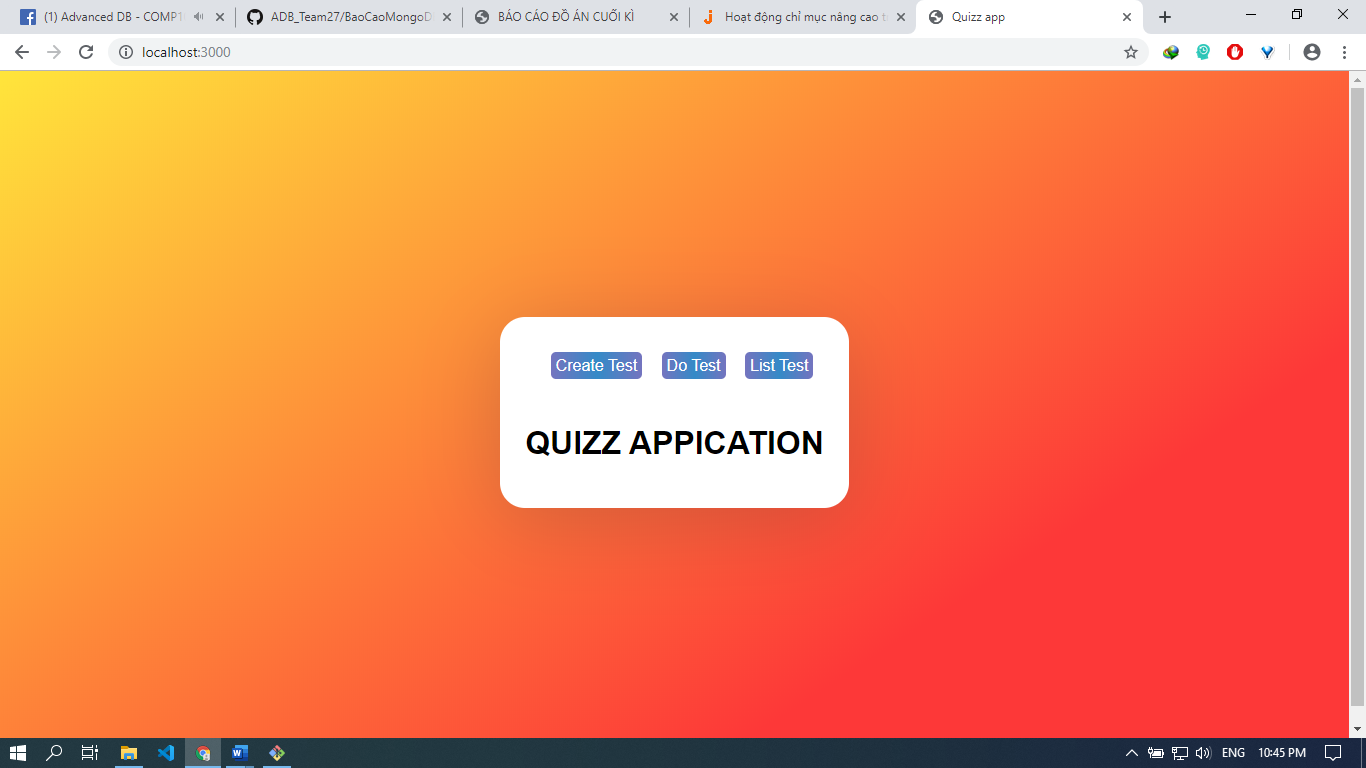
*Thư mục NuGet sau khi cài 2 thư viện trên*

Tạo kết nối với nodejs

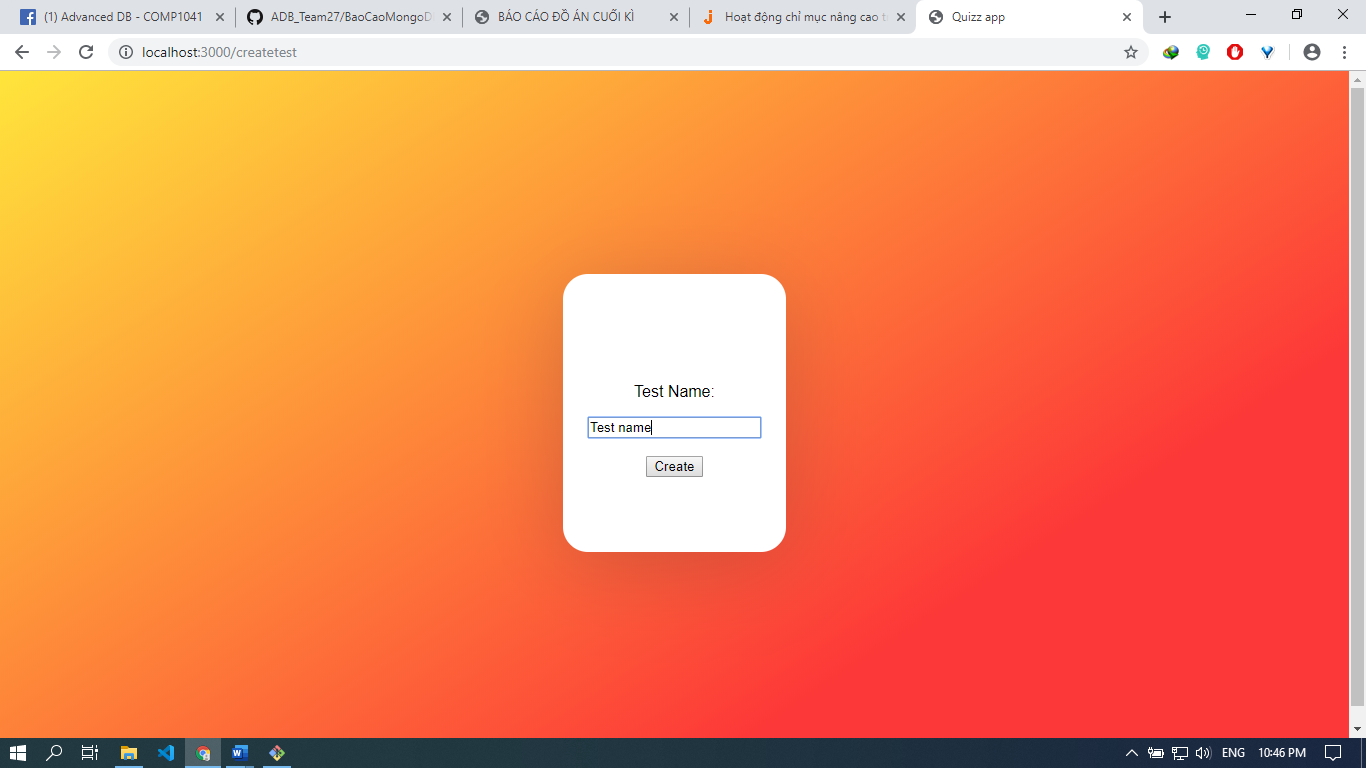


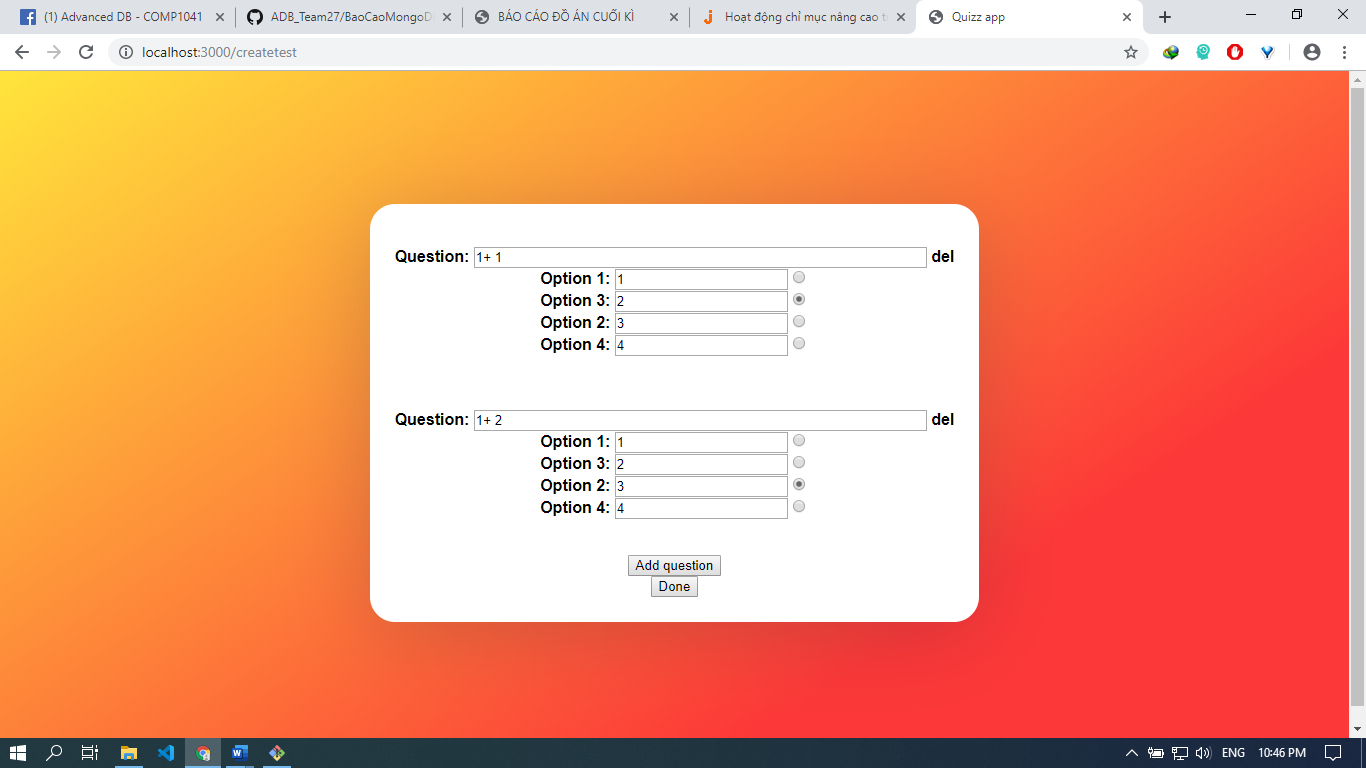
20. MONGODB PROJECT NODEJS Example:

Giao diện trang chủ

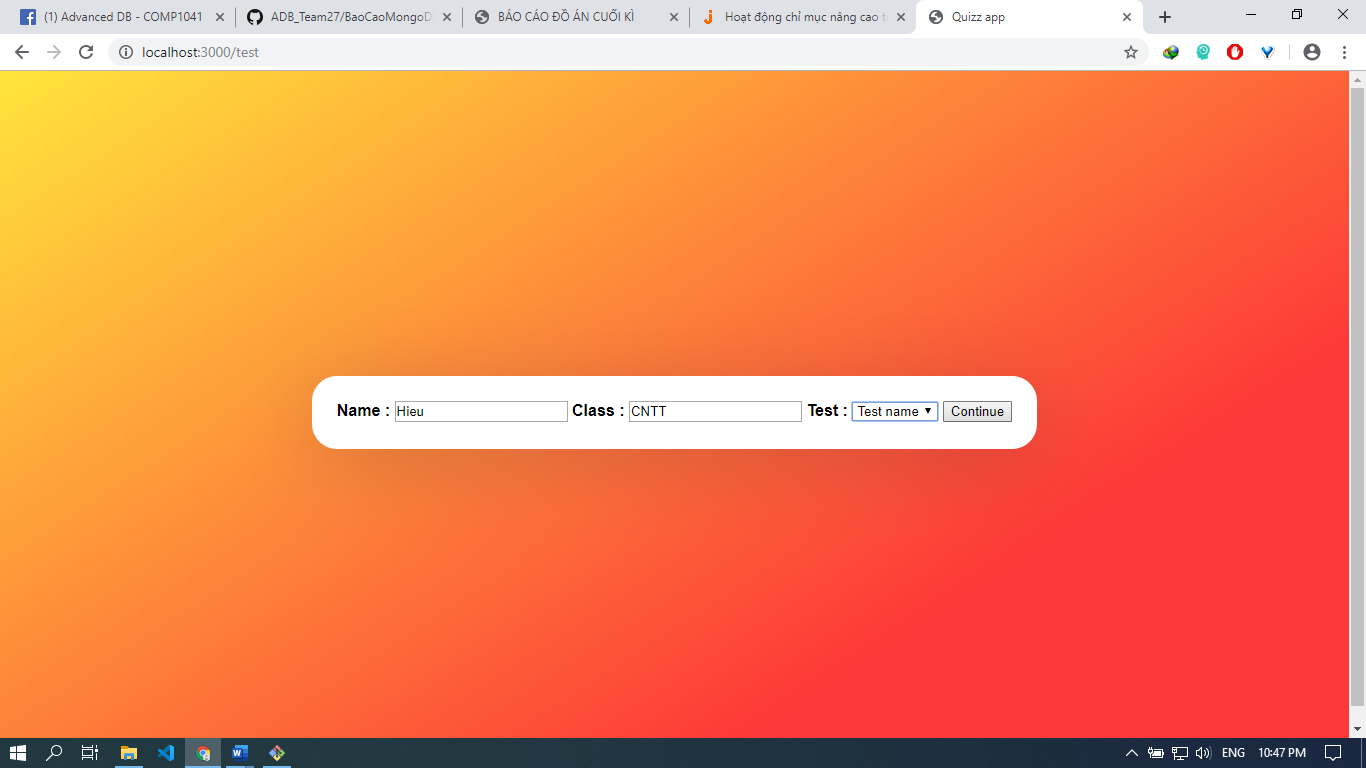


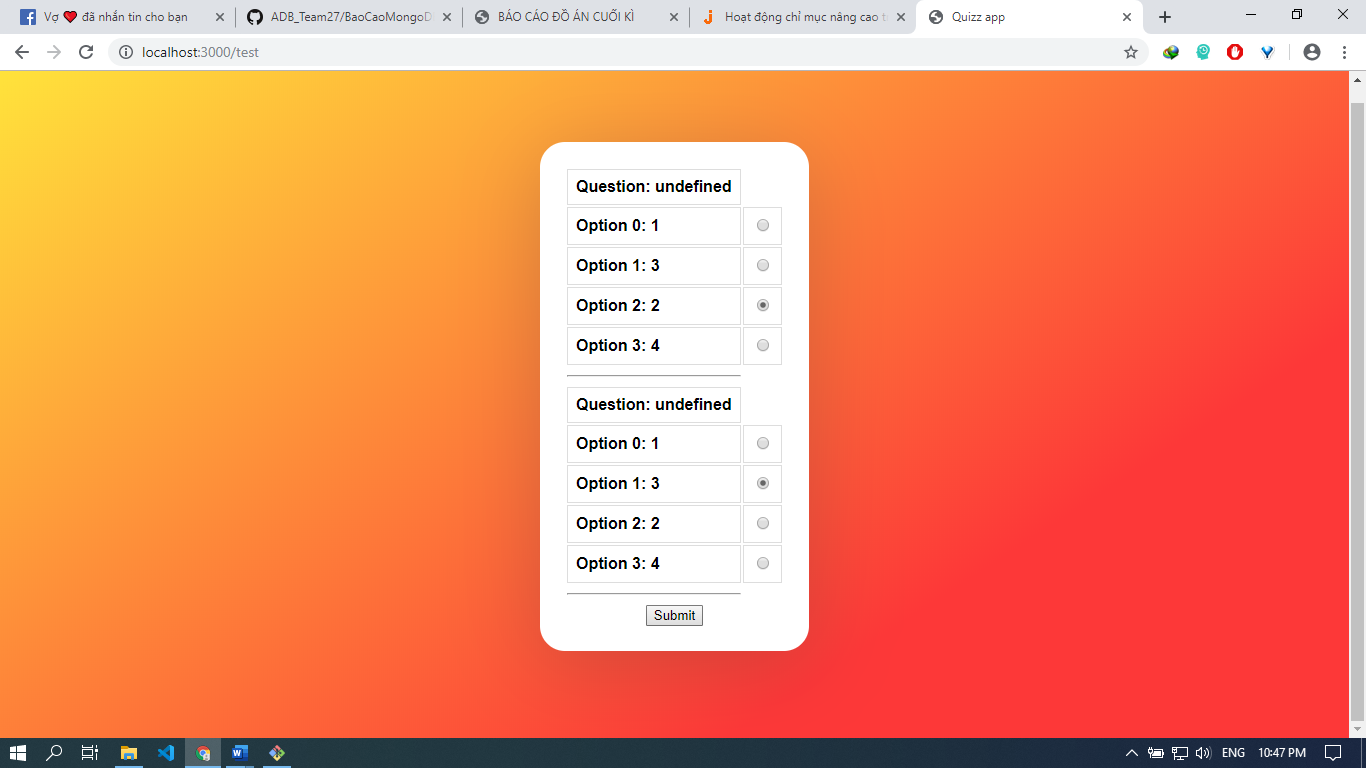
Trang create test



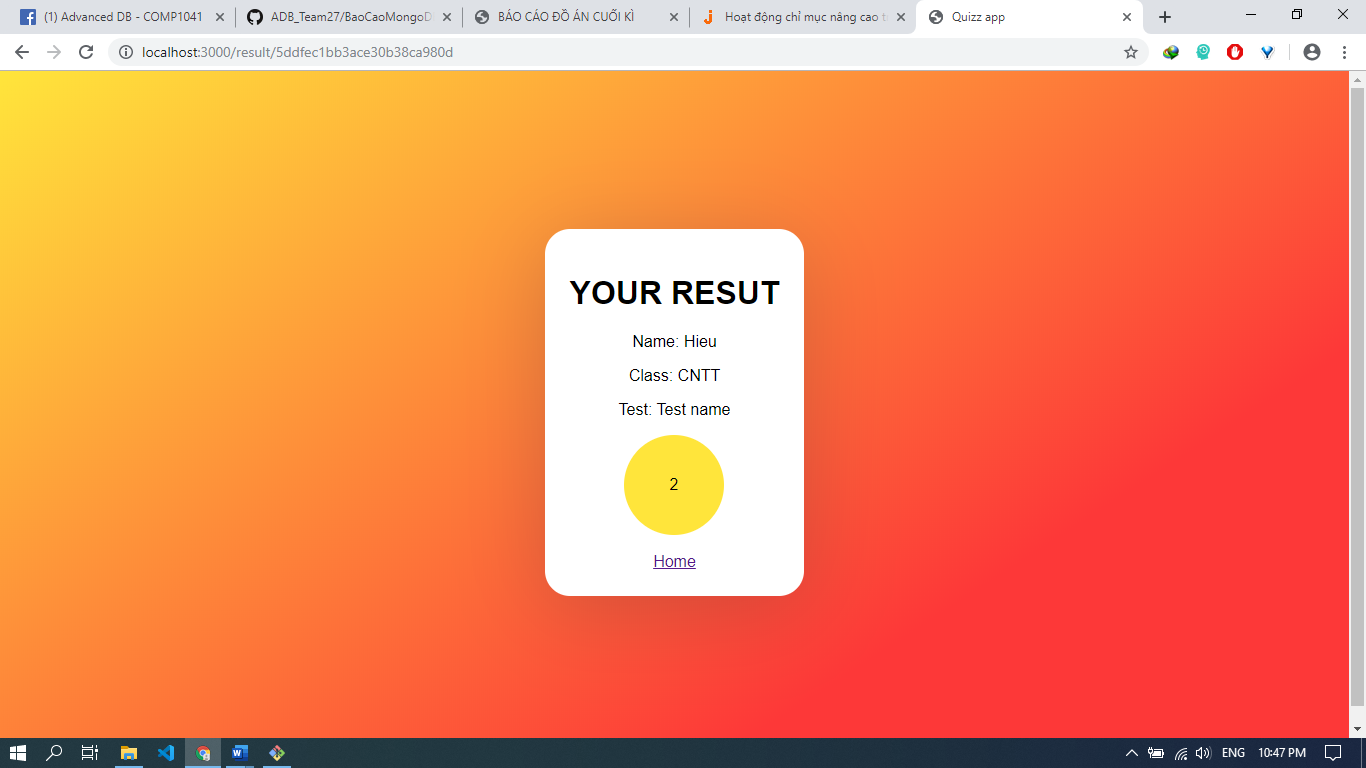


Trang do test

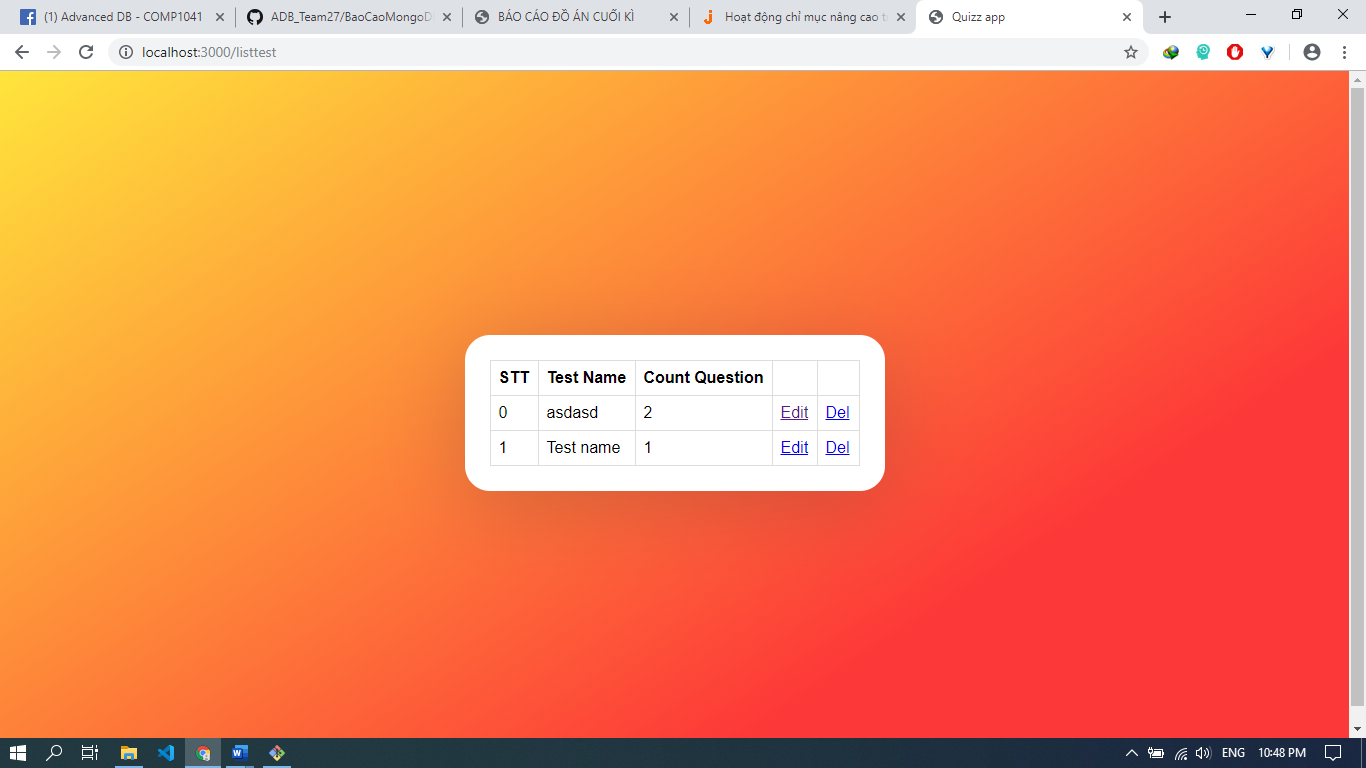




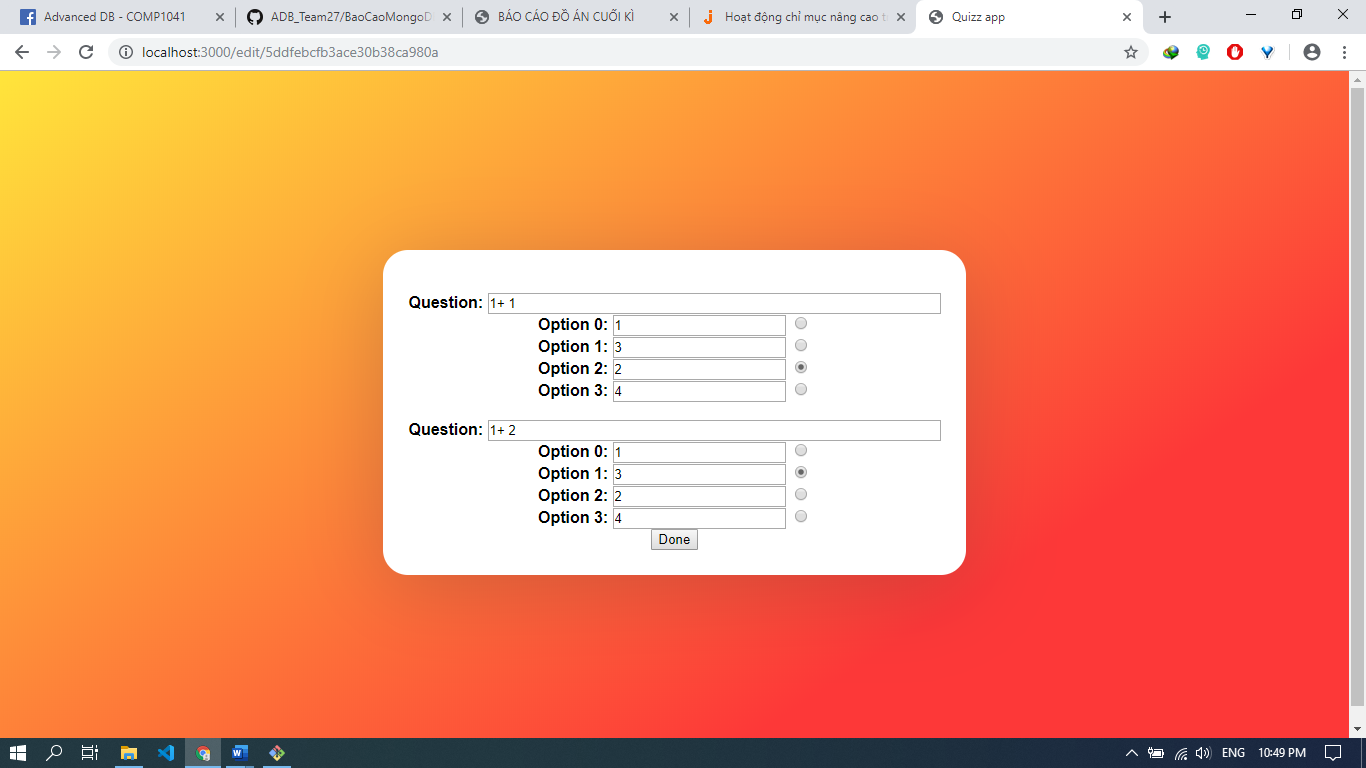
Trang kết quả



Trang quản trị test

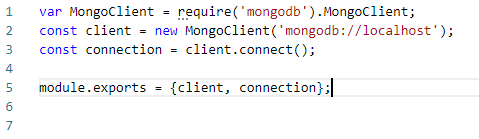


Trang edit test

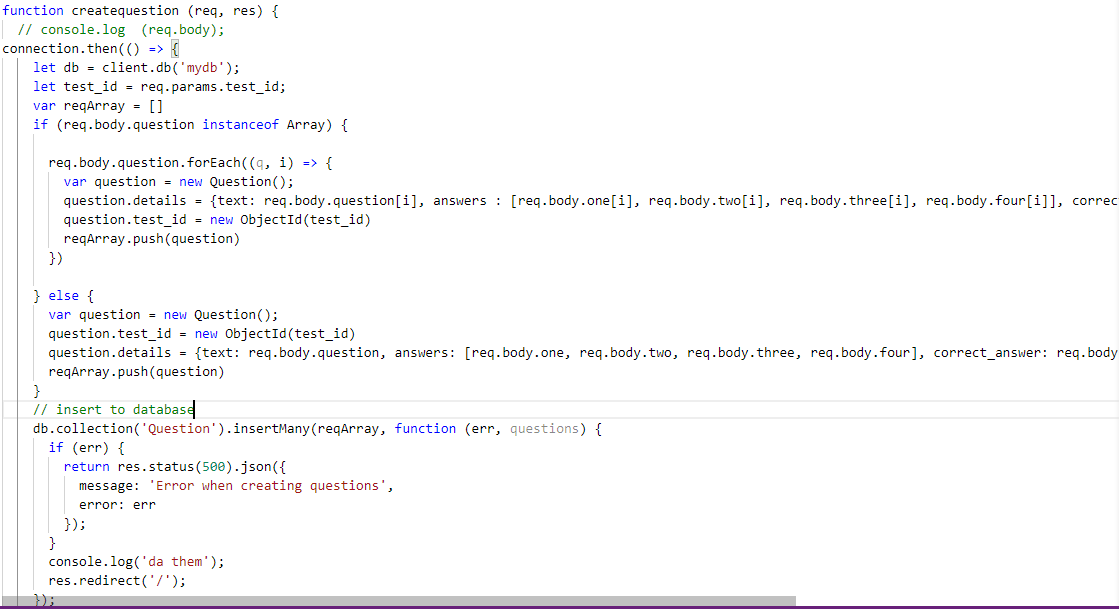


Giải thích code

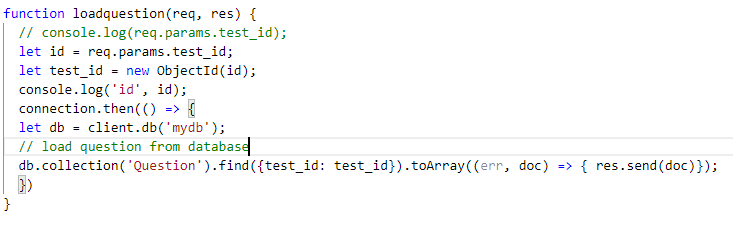
Create database connection



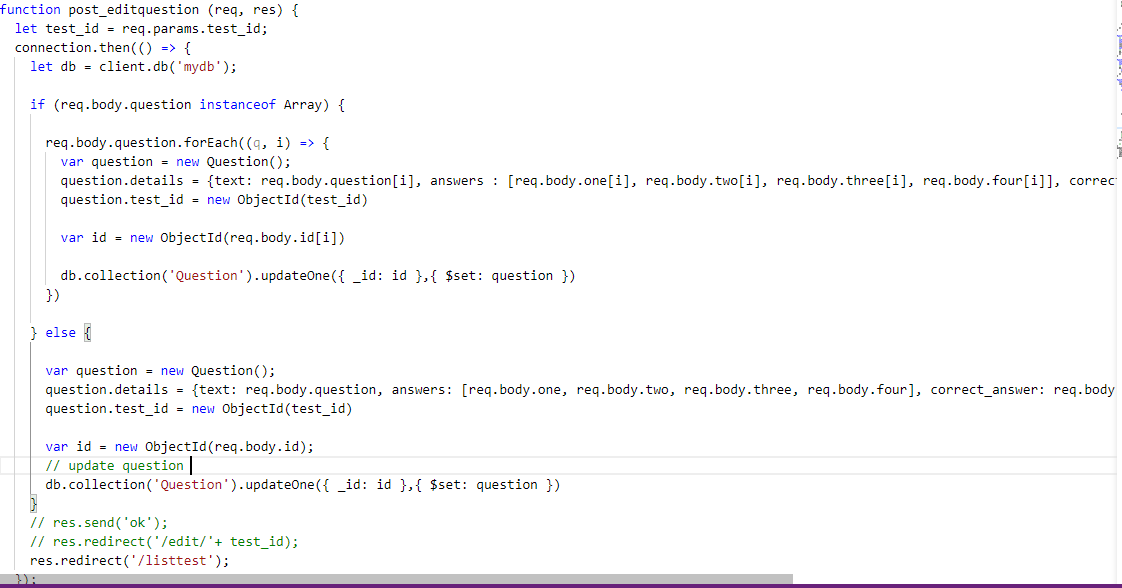
Tạo câu hỏi



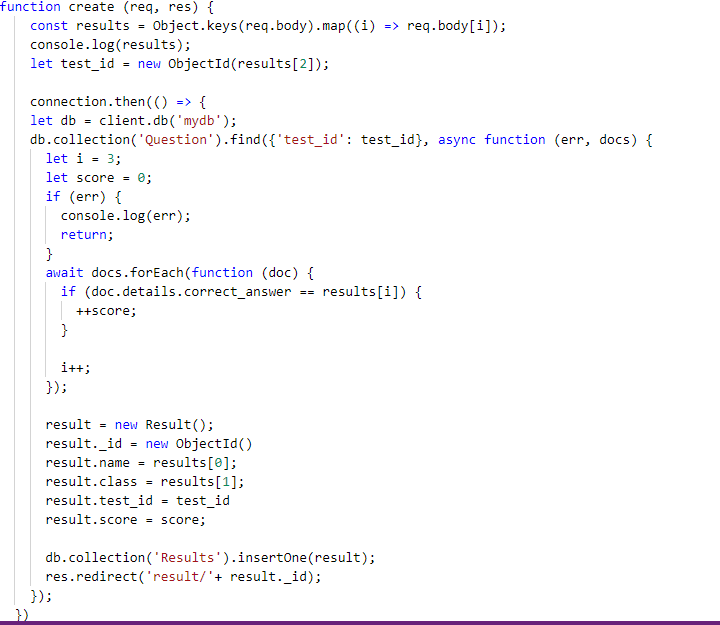
Load question



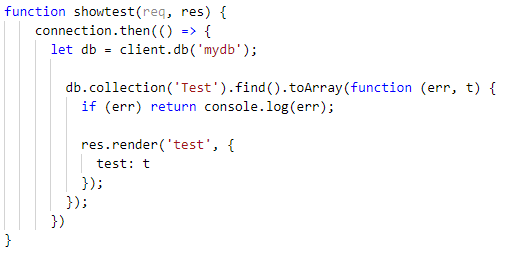
Edit question



Tạo kết quả



Load question test



Load + create test



21. Tổng kết và đánh giá:

* Tổng kết

MongoDB không sử dụng mô hình quan hệ ràng buộc(non-relational)

Mô hình Lưu trữ dạng {“key : value”}

MongoDB là 1 CSDL phân tán lưu chữ dữ liệu trên nhiều server khác nhau, đa dạng hóa dữ liệu, là dạng dữ liệu lớn( BigData).

Độ chịu tải cao ,chịu lỗi cao

Có khả năng nâng cấp theo chiều ngang, dễ dàng bảo trì, thay đổi,sửa chữa , phát triển

* Đánh giá

Chính vì những đặc điểm đó ta nên dùng nó cho các hệ thống, ứng dụng đòi hỏi lượng người dùng và dữ liệu lưu trữ đa dạng, lớn đến cực lớn.Ví dụ như mạng xã hội(facebook, Instagram, twitter…), các trang lưu trữ dữ liệu lớn như google, github, …..

TÀI LIỆU THAM KHẢO

1. <https://docs.mongodb.com/manual/tutorial/>
2. <https://en.wikipedia.org/wiki/MongoDB>
3. <https://vietjack.com/mongodb/>