CƠ SỞ DỮ LIỆU MONGO DB

NoSQL - Document Database

JSON - DOCUMENTS

- **JSON (JavaScript Object Notion)** là một định dạng dữ liệu được đưa ra bởi JavaScript.
- Bản thân JSON (*.json) là một kiểu tài liệu nên có thể đọc được bằng các ngôn ngữ, phần mềm khác.
- JSON được sử dụng để lưu trữ dữ liệu dưới dạng cặp **Key : Value**
- JSON thường được sử dụng để lưu các config của ứng dụng, hệ thống… cũng như giao tiếp giữa các ứng dụng qua **API** tương tự như XML.

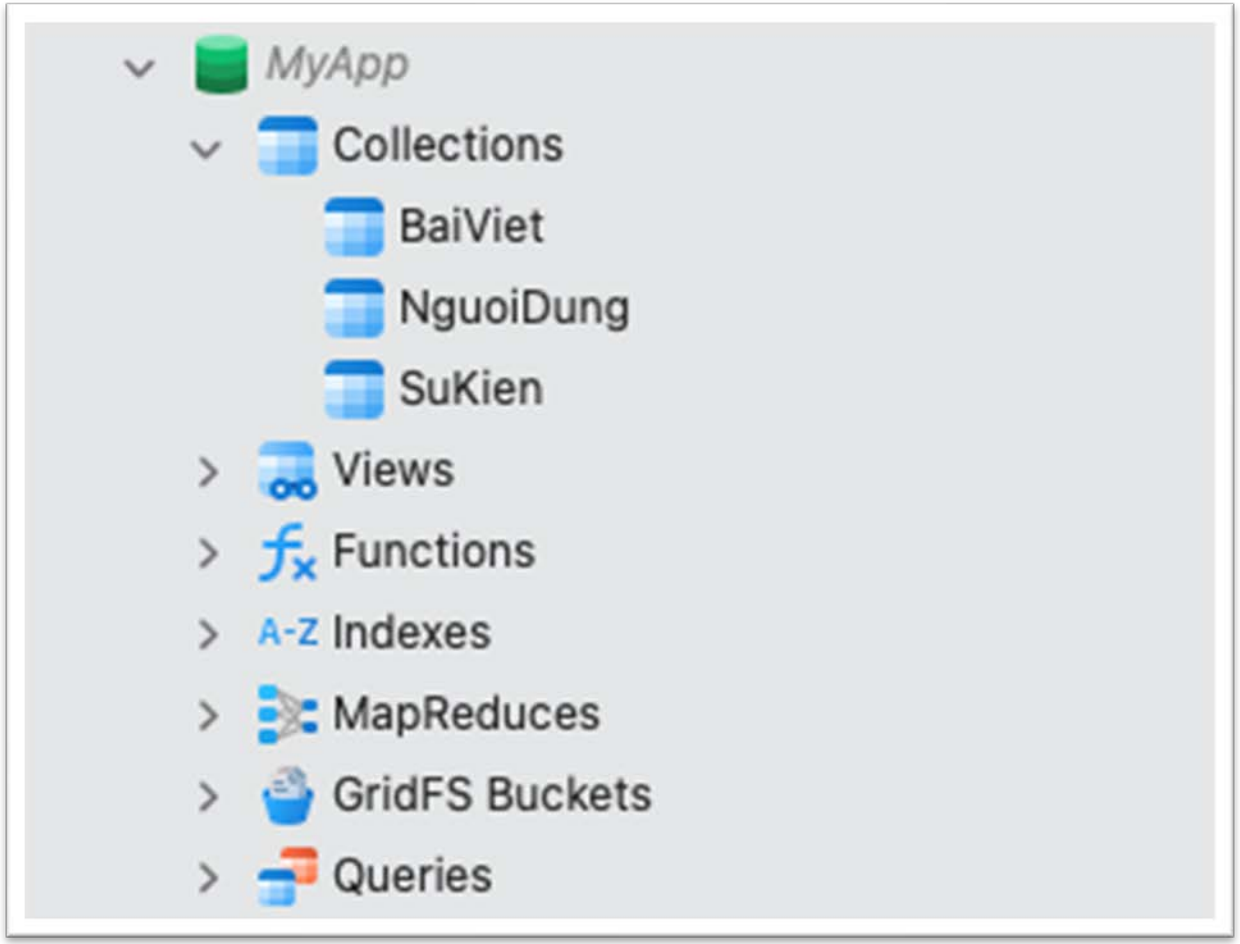
JSON – STRUCTER

- Cấu trúc của JSON là cặp Key và Value ngăn cách bằng dấu “:”
- Key: Tên đối tượng là một chuỗi ký tự đặt trong cặp dấu ”” .
- Value: Có thể là các kiểu dữ liệu trong JavaScript (Số, chuỗi, mảng, object, boolean, và **NULL**).
- Giữa các cặp **Key:Value** cách nhau bởi dấu “,”.

```json
{
    "Tên": "Nguyễn Ngọc Anh",
    "Tuổi": 21,
    "Địa Chỉ": {
        "Số Nhà": 12,
        "Đường": "Nguyễn Văn Trỗi",
        "Thành Phố": "Vinh",
        "Tỉnh": "Nghệ An"
    },
    "Sở Thích": [
        "Đọc sách",
        "Nghe nhạc",
        "Xem phim"
    ],
    "Tốt Nghiệp": false,
    "Người Yêu": null
}
```

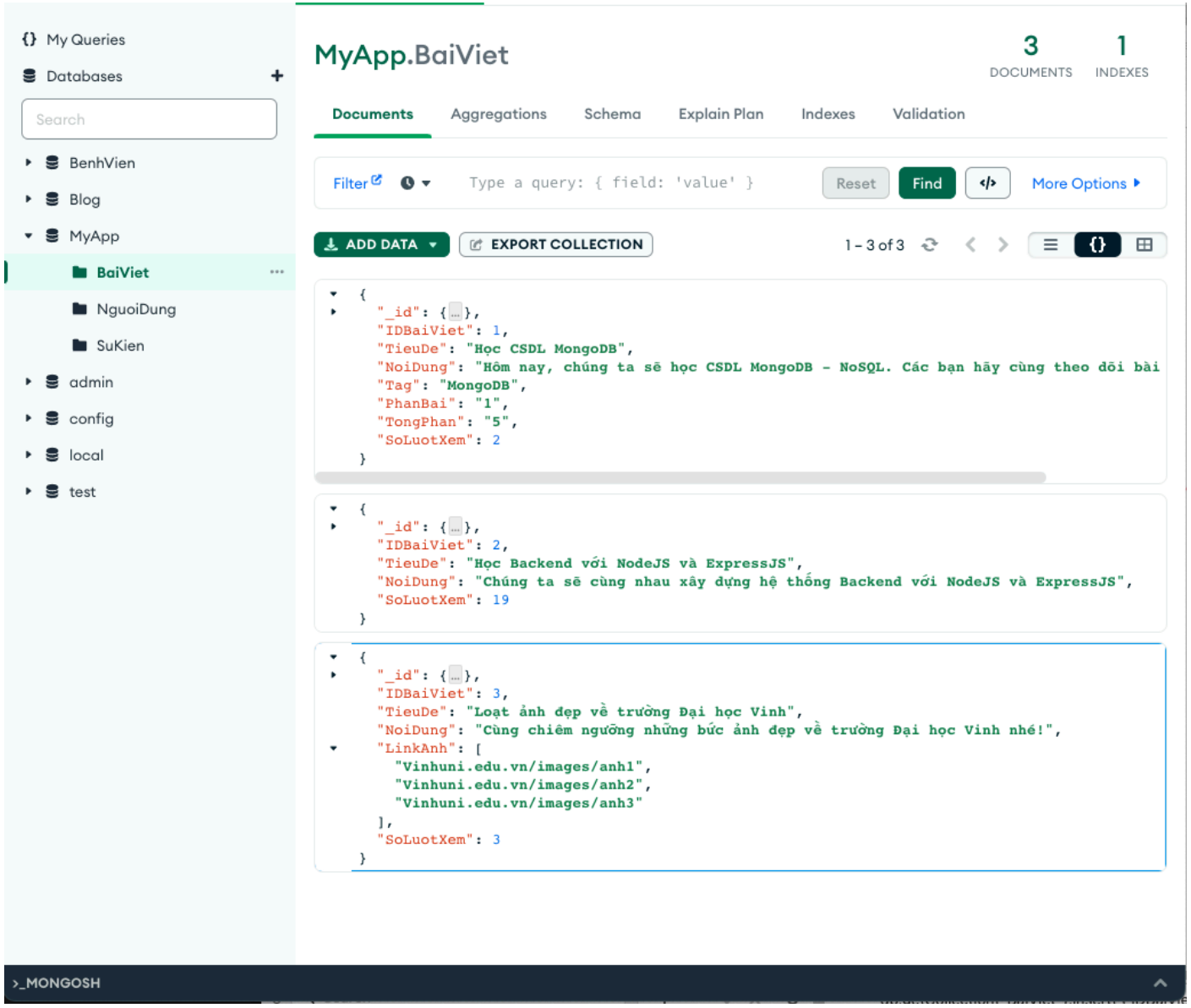
CƠ SỞ DỮ LIỆU MONGO DB

- MongoDB là một **NoSQL** dạng **Document Database**
- CSDL trong MongoDB được chia thành các **Collection** (Tương tự như Table trong CSDL Quan hệ)
- Mỗi **Collection** được biểu diễn dưới dạng **JSON Document**



MONGO DB - COLLECTIONS

- Mỗi bản ghi trong Collection là một JSON Document
- Các bản ghi không nhất thiết phải có những trường (key) giống nhau
- "_id" là một trường đặc biệt dùng để phân biệt các bản ghi với nhau, trường này mặc định do hệ thống khởi tạo



MONGO DB – BASIC QUERY

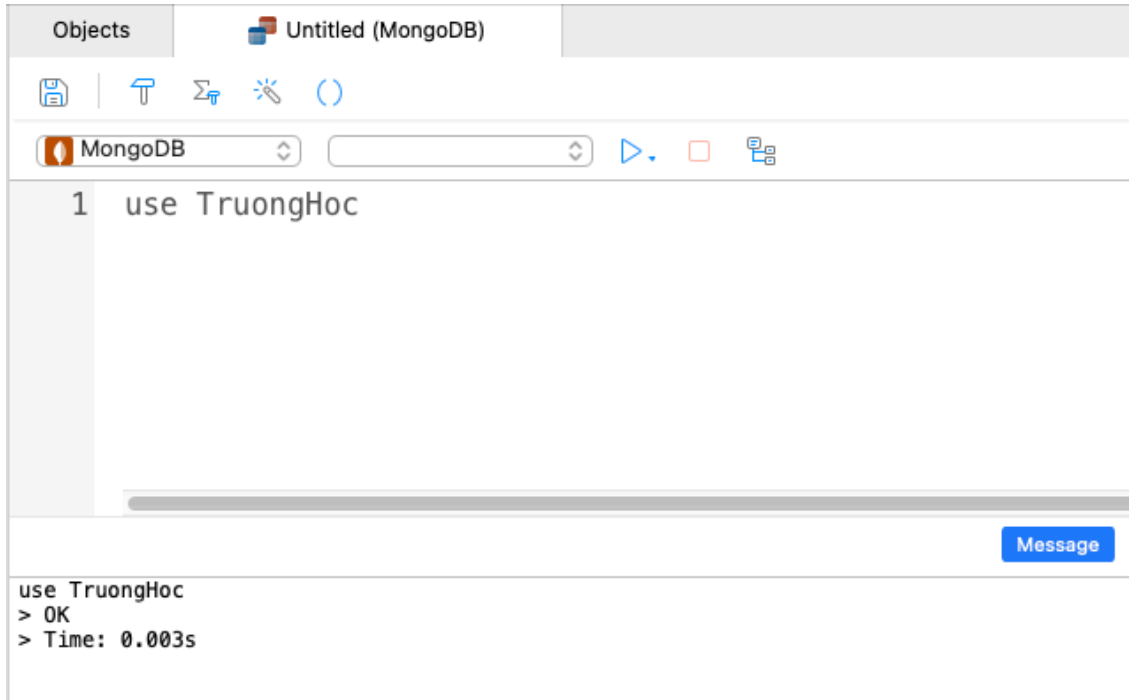
NoSQL QUERY	SQL QUERY
use MyApp Chú ý: Nếu CSDL đã tồn tại thì sẽ dùng luôn, chưa có thì sẽ tạo CSDL mới.	Create database MyApp use MyApp
db.createCollection("BaiViet")	Create table BaiViet(…)
db.BaiViet.drop()	Drop table BaiViet
db.BaiViet.find()	Select * from BaiViet
db.BaiViet.find({'IDBaiViet': 1})	Select * from BaiViet Where IDBaiViet = 1
db.BaiViet.find({$and: ['DBaiViet': 1, 'TieuDe': "Hoc MongoDB"]})	Select * from BaiViet Where IDBaiViet = 1 AND Tieude = 'Hoc MongoDB'
db.BaiViet.insertOne({ IDBaiViet: 3, Tieude: "Hoc SQL"})	Insert into BaiViet (IDBaiViet, Tieude) Values (2, 'Hoc SQL')
db.BaiViet.update({'IDBaiViet': 1} ,{$set:{'TieuDe': 'Hoc MongoDB Update'}})	Update BaiViet SET TieuDe = "Hoc MongoDB Update" Where IDBaiViet = 1
db.BaiViet.remove({'IDBaiViet': 1})	Delete From BaiViet Where IDBaiViet = 1

MONGO DB – BASIC EXAMPLE

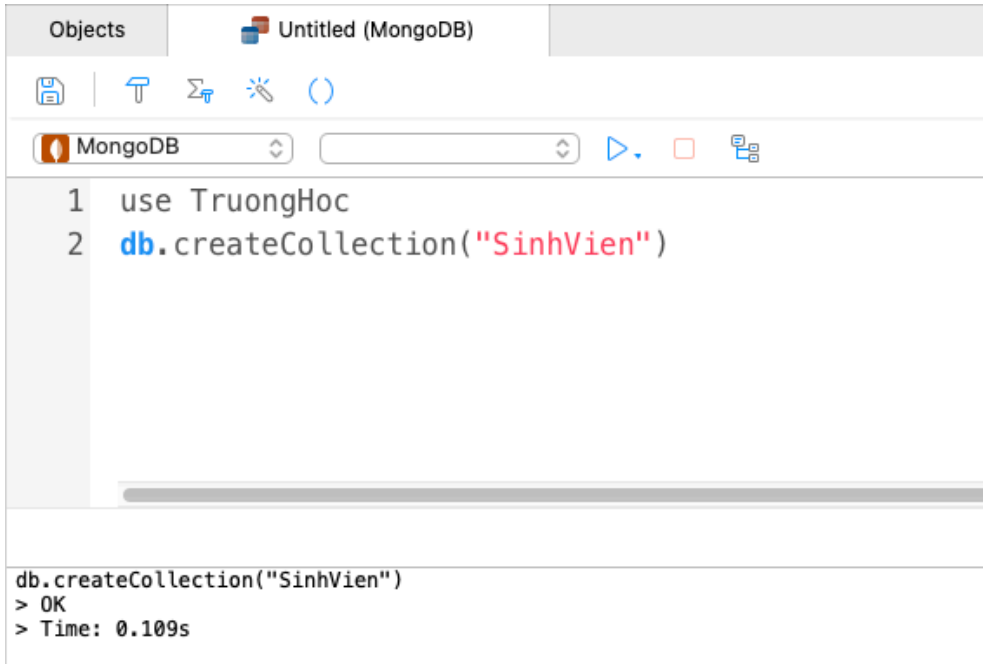
- Tạo CSDL “TruongHoc”
- Tạo Collection ”SinhVien”
- Thêm 1 đối tượng vào Collection “SinhVien” gồm các trường dữ liệu như: Mã sinh viên, Họ tên, Ngày sinh, Quê quán…
- Lấy ra thông tin tất cả sinh viên trong Collection ”SinhVien”
- Lấy ra thông tin của sinh viên qua mã sinh viên
- Xoá một sinh viên khỏi Collection “SinhVien”
- Chỉnh sửa quê quán của một sinh viên thông qua mã sinh viên

MONGO DB – BASIC EXAMPLE

- Tạo CSDL “TruongHoc”

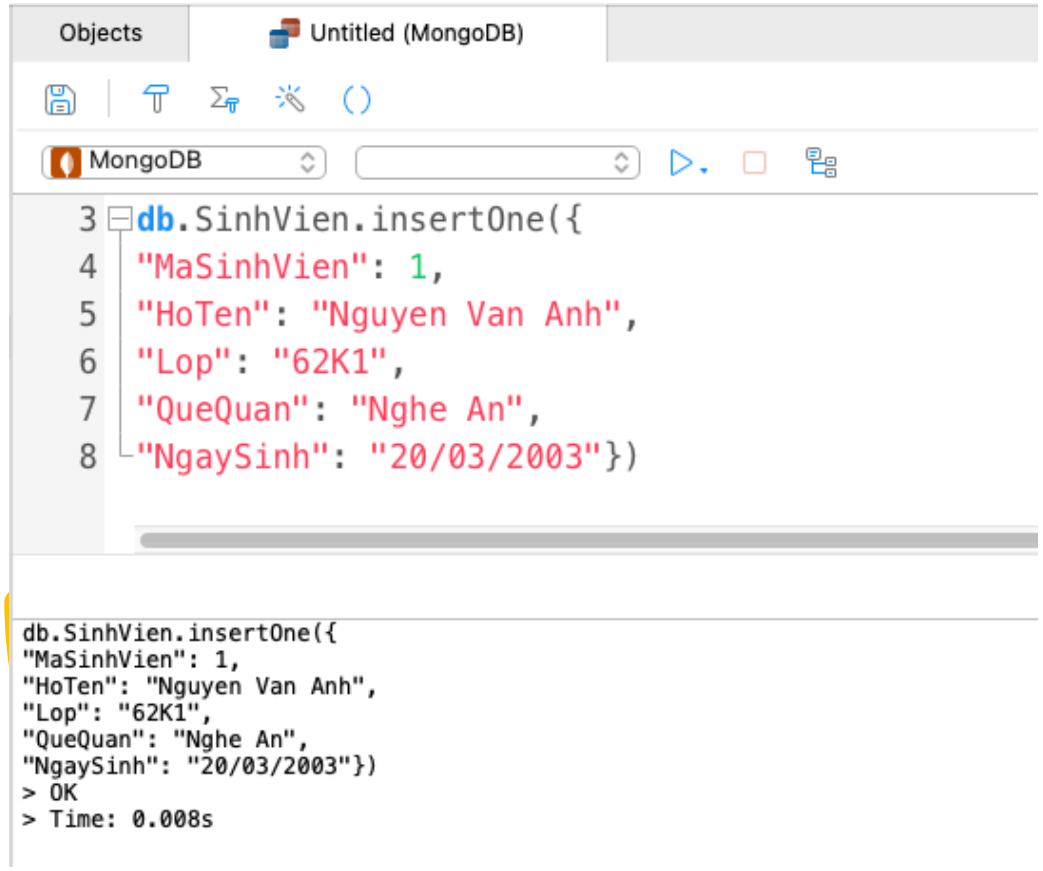


- Tạo Collection “SinhVien”

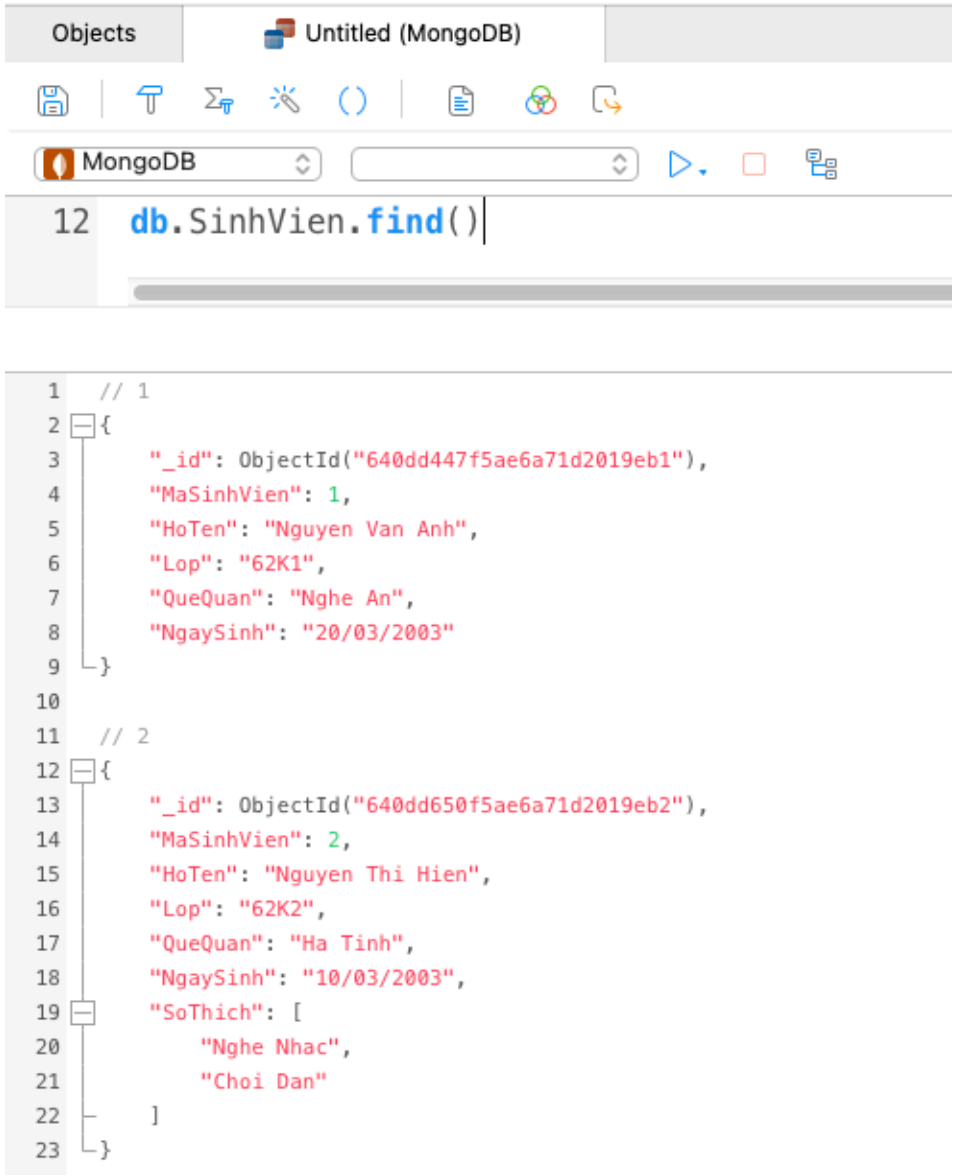


MONGO DB – BASIC EXAMPLE

- Thêm 1 sinh viên vào Collection

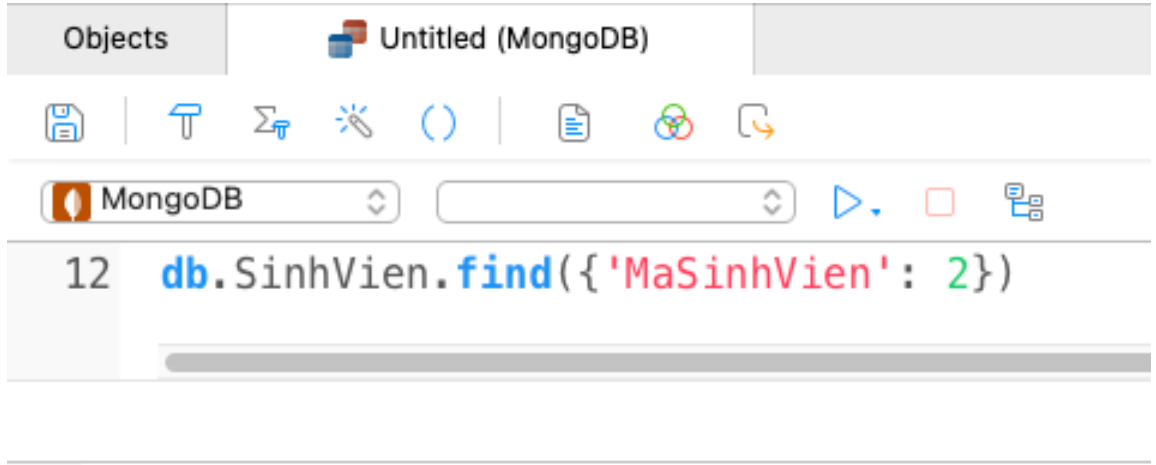


- Xem tất cả sinh viên trong Collection



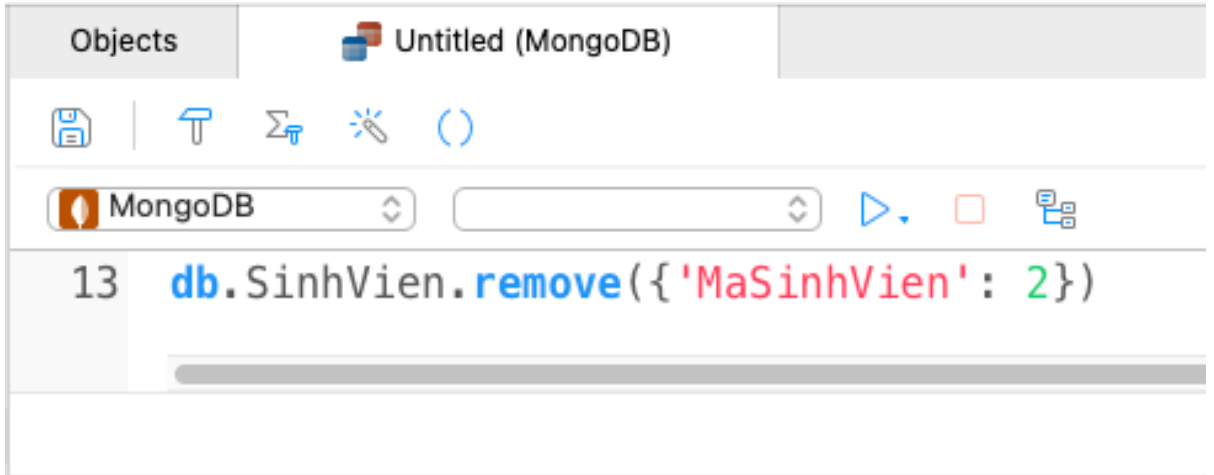
MONGO DB – BASIC EXAMPLE

- Xem thông tin sinh viên qua MSV




```
// 1
{
    "_id": ObjectId("640dd650f5ae6a71d2019eb2"),
    "MaSinhVien": 2,
    "HoTen": "Nguyen Thi Hien",
    "Lop": "62K2",
    "QueQuan": "Ha Tinh",
    "NgaySinh": "10/03/2003",
    "SoThich": [
        "Nghe Nhac",
        "Choi Dan"
    ]
}
```

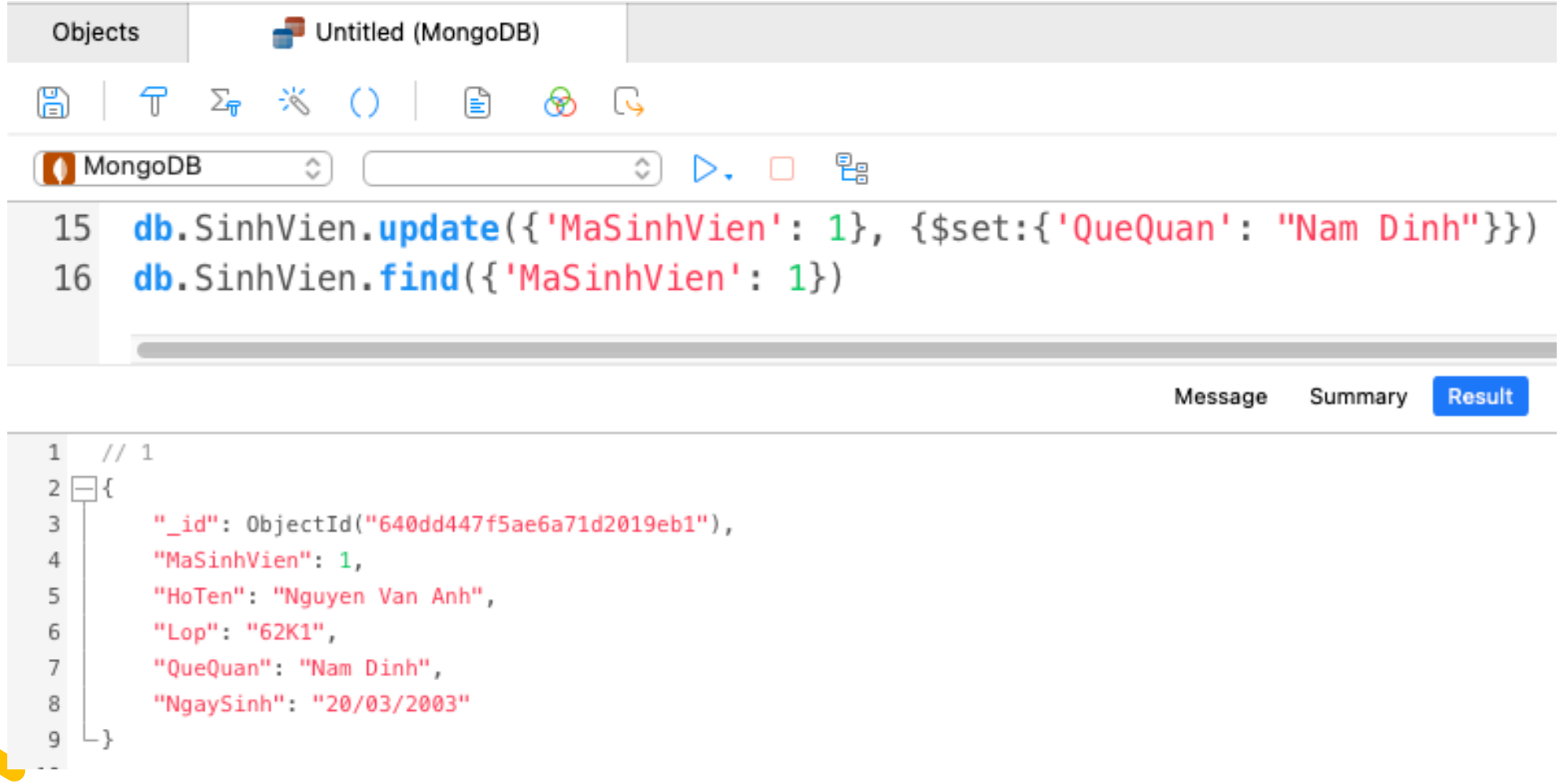
- Xoá một sinh viên qua MSV




```
db.SinhVien.remove({'MaSinhVien': 2})
> OK
> Time: 0.005s
```

MONGO DB – BASIC EXAMPLE

- Chỉnh sửa quê quán của sinh viên qua MSV



SO SÁNH GIỮA NOSQL VÀ RDBMS

Tiêu chí	NoSQL (MongoDB)	RDBMS
Kiểu dữ liệu	Dữ liệu dạng JSON (BSON)	Dữ liệu dạng bảng, hàng và cột
Tính linh hoạt	Schema động, linh hoạt, cho phép thay đổi cấu trúc dữ liệu mà không cần chỉnh sửa schema	Schema cố định, thay đổi schema khó khăn
Mối quan hệ	Mối quan hệ được biểu diễn thông qua việc nhúng tài liệu hoặc tham chiếu đến tài liệu khác	Mối quan hệ dựa trên khóa ngoại
Phương pháp lưu trữ	Denormalization và Normalization, lưu trữ dữ liệu theo cách tối ưu cho từng tình huống	Chuẩn hóa dữ liệu (Normalization)
Đọc và ghi dữ liệu	Đọc ghi dữ liệu nhanh chóng và dễ dàng do không cần join nhiều bảng	Cần thực hiện join giữa các bảng
Tính nhất quán	Tính nhất quán cuối cùng (Eventual consistency)	Tính nhất quán ngay lập tức (ACID)
Hỗ trợ ràng buộc dữ liệu	Ràng buộc dữ liệu ít hơn, có thể thêm thông qua Schema Validation	Hỗ trợ nhiều ràng buộc dữ liệu

MONGO DB – INTER. QUERY

```
use lab2
db.createCollection("posts")
db.posts.insertOne({
title: "Post Title 1",
body: "Body of post.",
category: "News",
likes: 1,
tags: ["news", "events"],
date: Date()
})
```

MONGO DB – INTER. QUERY

```
db.posts.insertMany([
{
title: "Post Title 2",
body: "Body of post.",
category: "Event",
likes: 2,
tags: ["news", "events"],
date: Date()
},
{
title: "Post Title 3",
body: "Body of post.",
category: "Technology",
likes: 3,
tags: ["news", "events"],
date: Date()
}])
```

MONGO DB – INTER. QUERY

3. Tìm kiếm

```
db.posts.find()
db.posts.findOne()
db.posts.find({})
db.posts.find( {category: "News"} )
```

$eq	So sánh bằng “=”
$ne	So sánh khác “≠”
$gt	So sánh lớn hơn
$gte	So sánh lớn hơn hoặc bằng
$lt	So sánh hỏ hơn
$lte	So sánh nhỏ hơn hoặc bằng
$in	Giá trị có trong mảng

MONGO DB – INTER. QUERY

3. Tìm kiếm

```
db.posts.find({likes:3})
db.posts.find({likes:{"$eq":3}})
db.posts.find({likes:{"$lt":3}})
db.posts.find({likes:{"$lte":3}})
db.posts.find({tags:{"$in":["news", "events"]}})
```

$eq	So sánh bằng “=”
$ne	So sánh khác “≠”
$gt	So sánh lớn hơn
$gte	So sánh lớn hơn hoặc bằng
$lt	So sánh hỏ hơn
$lte	So sánh nhỏ hơn hoặc bằng
$in	Giá trị có trong mảng

MONGO DB – INTER. QUERY

4. Phép chiếu

```
db.posts.find({}, {title: 1, date: 1})
db.posts.find({}, {_id:0, title: 1, date: 1})
```

Giá trị 0 để không hiển thị, Giá trị 1 để hiển thị thuộc tính

MONGO DB – INTER. QUERY

5. Tìm kiếm sử dụng các phép logic

```
db.posts.find( {category: "Event"} )
db.posts.find({$and:[{category: "Event"},{likes:2}]} )
Lưu ý db.posts.find({$and:[{category: "Event"},{likes:”2”}]} )
db.posts.find({$or:[{category: "Event"},{likes:2}]} )
```

$and	Đúng khi tất cả đúng
$or	Sai khi tất cả sai
$nor	Đúng khi tất cả sai
$not	Phủ định

MONGO DB – INTER. QUERY

5. Phép chiêu (lấy một số key trong documnet)

Cú pháp

db.collection.find(

{ <điều_kiện> },

{ { <trường_cần_đưa_ra>:1} }

)

db.movies.find({year:2019},{_id:0, title:1, rating:1 })

Lưu ý _id luôn hiển thị, để không hiện thì _id:0

MONGO DB – INTER. QUERY

6. Sửa nội dung đã có (updateOne)

Cú pháp

db.collection.updateOne(

{ <điều_kiện> },

{ $set: { <trường_cần_sửa>: <giá_trị_mới> } }

)

Ví dụ:

```
db.movies.updateOne(
  { title: "The Matrix" },
  { $set: { rating: 8.7 } }
)
```

MONGO DB – INTER. QUERY

Sửa nội dung đã có (update/updateMany)

Cú pháp

db.collection.update(

{ <điều_kiện> },

{ $set: { <trường_cần_sửa>: <giá_trị_mới> } }

)

db.collection.updateMany(

{ <điều_kiện> },

{ $set: { <trường_cần_sửa>: <giá_trị_mới> } }

)

MONGO DB – INTER. QUERY

Sửa nội dung đã có với các toán tử logic

Cập nhật với and/or

```
db.movies.updateOne(
  {
    $and: [
      { title: "Inception" },
      { year: 2010 }
    ]
  },
  {
    $set: { rating: 9.0 }
  }
)
```

•

MONGO DB – INTER. QUERY

7. Xóa nội dung trong collection

// cú pháp

```
db.collection.deleteOne({ <điều_kiện> })
```

//vi dụ xóa bộ phim có tiêu đề: Interstellar

```
db.movies.deleteOne({title:"Interstellar"})
```

//xóa với dk phức tạp thực hiện tương tự update

1. db.collection.deleteOne()
2. db.collection.remove() // trước đây đưa ra bây giờ ít dùng và thay băng deleteMany()
3. db.collection.deleteMany()

```
db.collection.deleteMany({ <điều_kiện> })
```

MONGO DB – INTER. QUERY

8. SẮP XẾP

Cú pháp

db.collection_name.find().sort({FieldName1: sort order 1 or -1, FieldName2: sort order})

Hiện đầy đủ thông tin của kết quả tìm được và sắp xếp

Ví dụ

```
db.movies.find().sort({title:1})
```

```
db.movies.find().sort({title:-1})
```

Hiển thiij một số thông tin và sắp xếp

```
db.movies.find({}, {"title":1, _id:0}).sort({"title":1})
```

Lưu ý: db.collection_name.find().sort()

MONGO DB – INTER. QUERY

9. ĐẾM SỐ DOCUMENT (COUNT)

Cú pháp

db.collection_name. countDocuments()

hoặc

db.collection_name. count()

Ví dụ

db.movies.countDocuments()

db.movies.countDocuments({year:{$gt:2000}})

MONGO DB – INTER. QUERY

9. THÊM KEY NẾU CHƯA CÓ

```
db.posts.updateMany(
{},
{ $set: { hobies:[]}}
)
```

```
db.employees.updateMany(
 {},
 { $set: { contact: { address: "", phone: "" } } }
)
```

Bài thực hành

Mã_nhân_viên	Họ_tên	Vai_trò	Lương	Tham_gia_dự_án
E1	Xuân	Quản lý	2000	IoT
E2	Hạ	Trưởng nhóm	1500	Blockchains
E3	Thu	Trưởng nhóm	1000	Data warehouse
E4	Đông	Quản lý	1500	OLAP
E5	Tây	Thư ký	1000	Blockchains

Viết lệnh NoSQL trong MongoDB trả lời các yêu cầu sau:

a/ Tạo một cơ sở dữ liệu và 1 Collection

b/ Nhập dữ liệu theo bảng đã cho.

c/ Bổ sung một nhân viên mới vào cơ sở dữ liệu với đủ thông tin và nhân viên đó tham gia ít nhất 2 dự án.

d/ Tăng lương cho các nhân viên có vai trò “quản lý” thành 2200.

e/ Đưa ra mã số, họ tên nhân viên giữ vai trò “Trưởng nhóm” và sắp xếp giảm dần theo lương.

Bài thực hành

Mã_nhân_viên	Họ_tên	Vai_trò	Lương	Tham_gia_dự_án
E1	Xuân	Quản lý	2000	IoT; Generative AI
E2	Hạ	Trưởng nhóm	1500	IoT; Blockchains
E3	Thu	Trưởng nhóm	1000	Data warehouse
E4	Đông	Quản lý	1500	OLAP; Graph database
E5	Tây	Thư ký	1000	Blockchains

Viết lệnh NoSQL trong MongoDB trả lời các yêu cầu sau:

a/ Tạo một cơ sở dữ liệu và 1 Collection

b/ n hập dữ liệu theo bảng đã cho.

b/ Bổ sung một nhân viên mới vào cơ sở dữ liệu với đủ thông tin và nhân viên đó tham gia ít nhất 2 dự án.

c/ Tăng lương cho các nhân viên có vai trò “quản lý” thành 2200.

d/ Đưa ra mã số, họ tên nhân viên giữ vai trò “Trưởng nhóm” và sắp xếp giảm dần theo lương.

MONGO DB – INTER. QUERY

10. CẬP NHẬT VỚI DỮ LIỆU KIỂU MẢNG

- $addToSet – Thêm một giá trị PHÂN BIỆT vào mảng (Nếu có giá trị đó rồi thì không)
- $pop – xóa 1 phần tử đầu hoặc cuối mảng [-1/1]
- $pull - xóa các phần tử trong mảng thỏa mãn điều kiện cụ thể. Cho phép sử dụng các toán tử so sánh để tạo điều kiện phức tạp
- $pullAll : Xóa tất cả các giá trị được chỉ định trong một mảng. Chỉ xóa chính xác các giá trị được liệt kê, không hỗ trợ biểu thức điều kiện.
- $push – Thêm một phần tử vào mảng.

MONGO DB – INTER. QUERY

10. CẬP NHẬT VỚI DỮ LIỆU KIỂU MẢNG

- Mảng dùng để lưu giá trị KHÔNG ĐƯỢC TRÙNG LẶP THÌ DÙNG $addToSet

- Mảng dùng để để lưu giá trị ĐƯỢC TRÙNG LẶP thì dùng $push

- Ví dụ

MONGO DB – INTER. QUERY

10. Dữ liệu minh họa

```
use lab2
db.createCollection("posts")
db.posts.insertOne({
title: "Post Title 1",
body: "Body of post.",
category: "News",
likes: 1,
tags: ["news", "events"],
date: Date()
})
```

MONGO DB – INTER. QUERY

10. 1. `$addToSet` Thêm 1/nhiều giá trị và mảng nếu GIÁ TRỊ CHƯA CÓ trong mảng

```
// Thêm tag "featured" vào mảng tags của bài post có title "Post Title 1"

// nếu tag này chưa tồn tại trong mảng

db.posts.updateOne(
{ title: "Post Title 1" },
{ $addToSet: { tags: "featured" } }
)

// Thêm nhiều tags cùng lúc (mỗi giá trị chỉ được thêm nếu chưa tồn tại)

db.posts.updateOne(
{ title: "Post Title 2" },
{ $addToSet: { tags: { $each: ["featured", "popular", "events"] } } }
)
```

MONGO DB – INTER. QUERY

10.2. $pull - Xóa tất cả phần tử khớp với điều kiện từ mảng

```
 // Xóa phần tử cuối cùng của mảng tags (giá trị 1 để xóa phần tử cuối)
db.posts.updateOne(
{ title: "Post Title 3" },
{ $pop: { tags: 1 } }
)


// Xóa phần tử đầu tiên của mảng tags (giá trị -1 để xóa phần tử đầu)
db.posts.updateOne(
{ title: "Post Title 4" },
{ $pop: { tags: -1 } }
)
```

MONGO DB – INTER. QUERY

10.3. $pop - Xóa phần tử đầu tiên hoặc cuối cùng của mảng

```
// Xóa phần tử cuối cùng của mảng tags (giá trị 1 để xóa phần tử cuối)
db.posts.updateOne(
{ title: "Post Title 3" },
{ $pop: { tags: 1 } }
)


// Xóa phần tử đầu tiên của mảng tags (giá trị -1 để xóa phần tử đầu)
db.posts.updateOne(
{ title: "Post Title 4" },
{ $pop: { tags: -1 } }
)
```

MONGO DB – INTER. QUERY

10.4. $pullAll - Xóa tất cả các phần tử được chỉ định trong mảng

```
// Xóa cả "news" và "events" khỏi mảng tags trong bài post có title "Post Title 1"

db.posts.updateOne(
{ title: "Post Title 1" },
{ $pullAll: { tags: ["news", "events"] } }
)
```

MONGO DB – INTER. QUERY

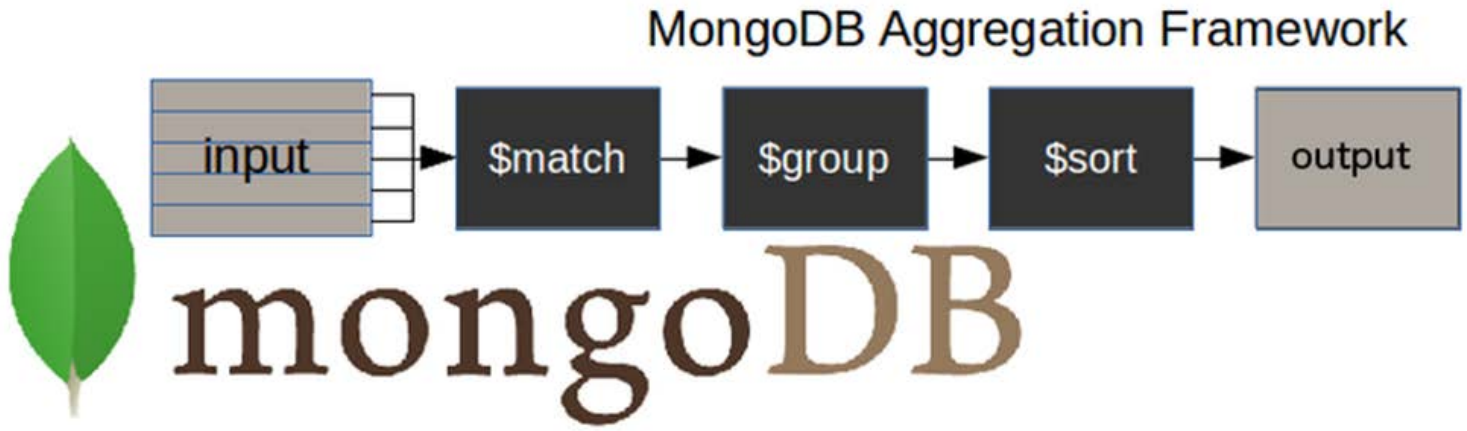
10.5. $push - Thêm phần tử vào mảng (có thể thêm trùng lặp)

```
 // Thêm tag "trending" vào mảng tags của bài post có title "Post Title 2"
db.posts.updateOne(
{ title: "Post Title 2" },
{ $push: { tags: "trending" } }
)
// Thêm nhiều tags cùng lúc vào bài post có title "Post Title 3"
db.posts.updateOne(
{ title: "Post Title 3" },
{ $push: { tags: { $each: ["popular", "recommended"] } } }
)
```

```
 // Thêm tag vào mảng và giới hạn kích thước mảng (giữ lại 3 phần tử mới nhất)
db.posts.updateOne(
{ title: "Post Title 4" },
{
$push: {
tags: {
$each: ["trending"],
$slice: -3 // Giữ lại 3 phần tử cuối cùng sau khi thêm
}
}
}
)
```

MONGO DB – INTER. QUERY

10. GOM NHÓM



- $match – lọc một hoặc các documents theo yêu cầu
- $group – thực hiện gom nhóm và áp dụng toán tử
- $sort - sắp xếp và đưa ra kết quả theo yêu cầu (tăng dần hoặc giảm dần)

Ví dụ: db.collection_name.aggregate([{$match }, {$group {biengomnhom,Toán_tử}}, {$sort}])

- **db.movies. aggregate([{$group:{_id:null, tongrating:{$sum:"$rating"}}}])**

MONGO DB – Adv. QUERY

11. 1. Đếm số lượng bài viết (đếm số lượng documnents)

```
 db.posts.countDocuments()
```

Hoặc

```
db.posts.count()
```

11.2. Đếm số lượng bài viết theo thể loại (category) – Đếm có gom nhóm

```
db.posts.aggregate([
// Nhóm theo category và đếm số lượng
{ $group: { _id: "$category", count: { $sum: 1 } } }
])
```

MONGO DB – Adv. QUERY

11. 1. ~~Đếm~~ số lượng bài viết theo category và sắp xếp giảm dần

```
 db.posts.aggregate([
// Nhóm theo category và đếm số lượng
{ $group: { _id: "$category", count: { $sum: 1 } } },
// Sắp xếp theo số lượng giảm dần
{ $sort: { count: -1 } }
])
```

MONGO DB – Adv. QUERY

11. 2. Tính tổng số lượt likes theo category

```
 db.posts.aggregate([
// Nhóm theo category và tính tổng likes
{ $group: { _id: "$category", totalLikes: { $sum: "$likes" } } },
// Sắp xếp theo tổng likes giảm dần
{ $sort: { totalLikes: -1 } }
])
```

MONGO DB – Adv. QUERY

11. 3. Thống kê số lượng likes trung bình theo category

```
 db.posts.aggregate([
// Nhóm theo category và tính trung bình likes
{ $group: {
_id: "$category",
avgLikes: { $avg: "$likes" },
}
},
// Sắp xếp theo số likes trung bình giảm dần
{ $sort: { avgLikes: -1 } }
])
```

MONGO DB – Adv. QUERY

11. 4. Tìm bài viết có nhiều lượt likes nhất

```
 db.posts.aggregate([
// Sắp xếp theo số likes giảm dần
{ $sort: { likes: -1 } },
// Giới hạn kết quả (chỉ lấy bài viết có nhiều likes nhất)
{ $limit: 1 },
// Chọn các trường muốn hiển thị
{ $project: { _id: 0, title: 1, category: 1, likes: 1 } }
])
```

MONGO DB – Adv. QUERY

12. $unwind trong MongoDB là một stage của Aggregation Framework được dùng để tách một mảng trong document thành nhiều document riêng biệt, mỗi document chứa một phần tử từ mảng gốc.

- Ví dụ:{ title: "Post Title 2",

  category: "Event",

  likes: 2,

  tags: ["news", "events"],

  date: Date()

}

db.posts.aggregate([

 { $unwind: "$tags" }

])

```
{
  title: "Post Title 2",
  category: "Event",
  likes: 2,
  tags: "news",
  date: Date()
}
{
  title: "Post Title 2",
  category: "Event",
  likes: 2,
  tags: "events",
  date: Date()
}
```

MONGO DB – Adv. QUERY

12.1 Dùng $unwind để phân tích theo từng PHẦN TỬ trong mảng

Ví dụ: Đếm số lượng bài viết cho mỗi tag

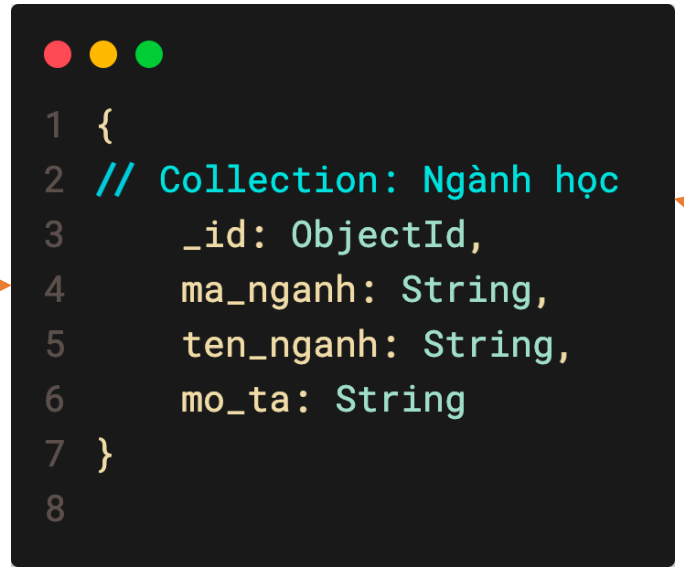
```
db.posts.aggregate([
{ $unwind: "$tags" },
{ $group: { _id: "$tags", count: { $sum: 1 } } }
])
```

```
 db.posts.aggregate([
// Tách mảng tags thành các document riêng biệt
{ $unwind: "$tags" },
// Nhóm theo tag và đếm số lượng
{ $group: { _id: "$tags", count: { $sum: 1 } } },
// Sắp xếp theo số lượng giảm dần
{ $sort: { count: -1 } }
])
```

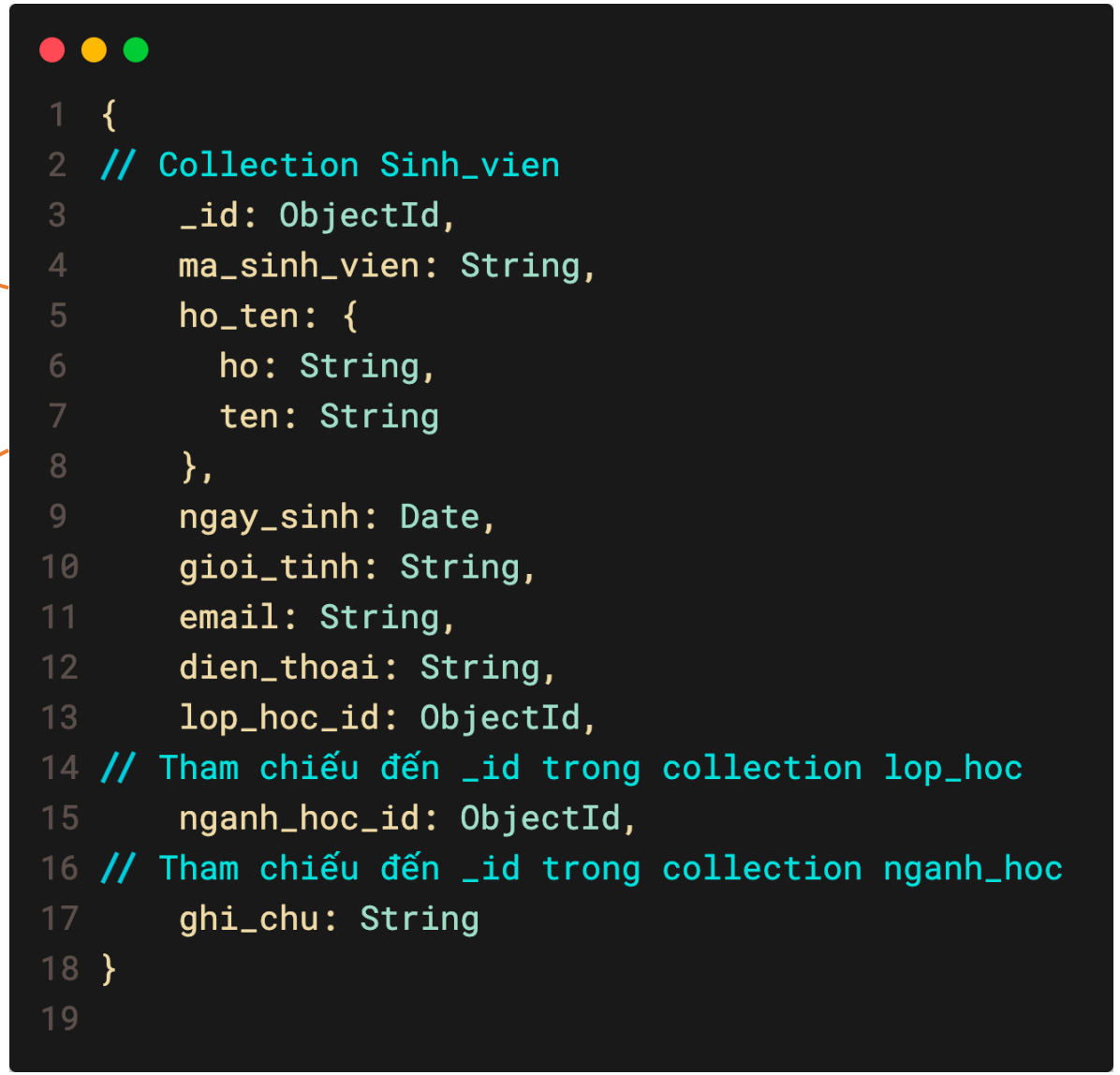
THIẾT KẾ SCHEMA TRONG MONGODB

1. Thiết kế dựa trên nhu cầu ứng dụng: Thiết kế schema phải dựa trên nhu cầu ứng dụng.
2. Sử dụng schema đơn giản
3. Tối ưu hóa cho truy vấn: Thiết kế schema phải tối ưu hóa cho các truy vấn phổ biến.
4. Sử dụng nhúng thay vì tham chiếu giúp tối ưu hóa việc truy vấn dữ liệu và giảm thiểu số lần truy vấn giữa các collection.
5. Cân bằng giữa độ linh hoạt và hiệu quả: Trong quá trình thiết kế schema, cần cân bằng giữa độ linh hoạt và hiệu quả.
6. Hạn chế sử dụng lồng nhau (nesting): Lồng nhau quá sâu có thể làm cho việc truy vấn và hiển thị dữ liệu trở nên phức tạp và chậm.
7. Sử dụng các trường kiểu array một cách hợp lý: Sử dụng các trường kiểu array giúp tối ưu hóa việc lưu trữ và truy vấn dữ liệu.

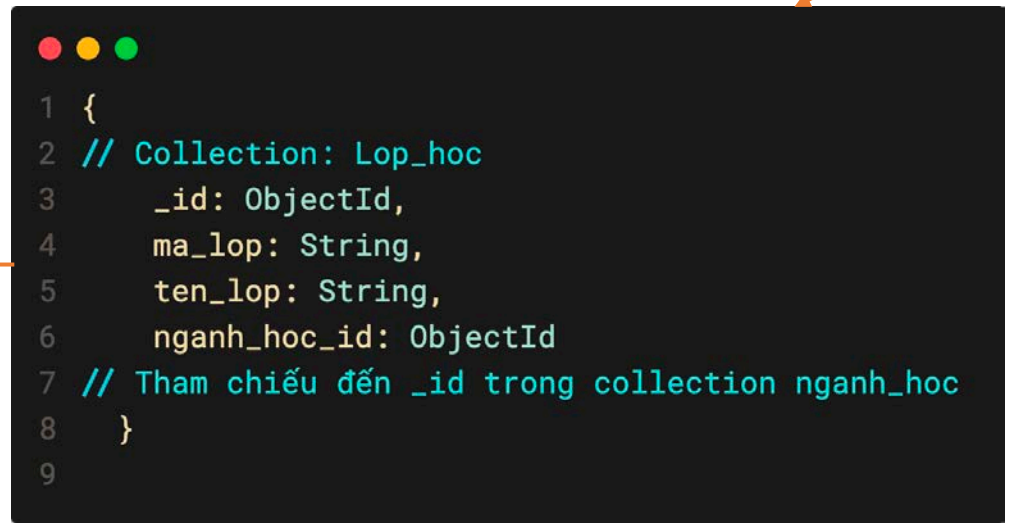
VÍ DỤ SCHEMA ĐƠN GIẢN



```
{
// Collection: Ngành học
    _id: ObjectId,
    ma_nganh: String,
    ten_nganh: String,
    mo_ta: String
}
```



```
{
// Collection Sinh_vien
    _id: ObjectId,
    ma_sinh_vien: String,
    ho_ten: {
      ho: String,
      ten: String
    },
    ngay_sinh: Date,
    gioi_tinh: String,
    email: String,
    dien_thoai: String,
    lop_hoc_id: ObjectId,
// Tham chiếu đến _id trong collection lop_hoc
    nganh_hoc_id: ObjectId,
// Tham chiếu đến _id trong collection nganh_hoc
    ghi_chu: String
}
```



```
{
// Collection: Lop_hoc
    _id: ObjectId,
    ma_lop: String,
    ten_lop: String,
    nganh_hoc_id: ObjectId
// Tham chiếu đến _id trong collection nganh_hoc
  }
```

MÔ HÌNH HOÁ DỮ LIỆU

1. LIÊN KẾT 1:1. Một GiaoVien chủ nhiệm 1 lớp và 1 lớp có 1 giaoc viên chủ nhiệm

```
// Collection: GiaoVien
{
  _id: ObjectId("661f34eaf18f2d1a2b9c9a1a"),
  ten: "Nguyen Van A",
  monDay: "Toan"
}

// Collection: Lop
{
  _id: ObjectId("662f45ab34ed3d2c3a7c9b2b"),
  tenLop: "10A1",
  giaoVienChuNhiem: ObjectId("661f34eaf18f2d1a2b9c9a1a")
}
```

MÔ HÌNH HOÁ DỮ LIỆU

1. LIÊN KẾT 1:N. Một lớp học có nhiều học sinh và mỗi học sinh chỉ thuộc 1 lớp học

```
{ _id: ObjectId("663a2c1d8c7d991d4fcabcde"),
  tenLop: "10A1",
  hocSinh: [
    { _id: ObjectId("663a2c4f8c7d991d4fcabcdf"),
      hoTen: "Tran Thi B",
      tuoi: 16    },
    { _id: ObjectId("663a2c6a8c7d991d4fcabce0"),
      hoTen: "Le Van C",
      tuoi: 15
    } ]
}
```

MÔ HÌNH HOÁ DỮ LIỆU

1. LIÊN KẾT N:M

```
{
    // Collection: SinhVien
    {
        _id: ObjectId("6170a8c4b8af1b7d2a2b3091"),
        ten_sinh_vien: "Nguyen Van A",
        ma_sinh_vien: "SV001"
    },
    {
        _id: ObjectId("6170a8c4b8af1b7d2a2b3092"),
        ten_sinh_vien: "Nguyen Van B",
        ma_sinh_vien: "SV002"
    }
}
```

```
{
// Collection: MonHoc
    {
    _id: ObjectId("6170a6d3b8af1b7d2a2b3090"),
    ten_mon_hoc: "Toan cao cap",
    ma_mon_hoc: "MATH101"
    },
    {
    _id: ObjectId("6170a6d3b8af1b7d2a2b3091"),
    ten_mon_hoc: "Lap trinh C",
    ma_mon_hoc: "C101"
    },
}
```

```
{
// Collection: DangKyHoc
    {
    _id: ObjectId("6170a8c4b8af1b7d2a2b3092"),
    sinh_vien_id: ObjectId("6170a8c4b8af1b7d2a2b3091"),
    mon_hoc_id: ObjectId("6170a6d3b8af1b7d2a2b3090")
    },
    {
    _id: ObjectId("6170a8c4b8af1b7d2a2b3093"),
    sinh_vien_id: ObjectId("6170a8c4b8af1b7d2a2b3091"),
    mon_hoc_id: ObjectId("6170a6d3b8af1b7d2a2b3091")
    }
}
```

MONGO DB – Nối các Colletions

1. Mối liên kết 1-1.

- Có 2 cách cơ bản để tổ chức dữ liệu cho mối liên kết 1-1, tùy vào nhu cầu truy xuất và kích thước dữ liệu.
- Cách 1: Nhúng (Embedded) – khi dữ liệu có kích thước nhỏ và hai phần hay được được truy xuất cùng nhau.
- Cách 2: Tham chiếu (reference) – khi dữ liệu lớn hoặc hai phần truy xuất riêng (truy xuất độc lập. Cần tách bảng rõ ràng như trong RDBMS

MONGO DB – Nối các Colletions

Ví dụ cách 1 (nhúng) để lưu trữ liên kết 1-1.

```
use KetNoi
db.createCollection("khachhang")
db.khachhang.insertMany([{
_id: 1, name: "Tran",
profile: {
age: 30,address:"123 Le Duan"}
},
{_id: 2,
name: "Le", profile: {
age: 25,address: "345 Tran Phu"}
}])
```

MONGO DB – Kết nối Colletions

Ví dụ cách 2 (tham chiếu) để lưu trữ liên kết 1-1.

```
use KetNoi
db.createCollection("users")
db.users.insertMany([{ _id: 1, name: "Tran", profile_id: 100},
{ _id: 2, name: "Le", profile_id: 200}
])

db.createCollection("profiles")
db.profiles.insertMany([
{_id: 100, age: 30, address: "123 Le Duan"},
{_id: 200, age: 25, address: "345 Tran Phu"}
])
```

MONGO DB – Nối các Colletions

2. Lấy thông tin từ 2 colletcions

Sử dụng aggregate framework và $lookup

```
db.users.aggregate([
{
$lookup: {
from: "profiles",
localField: "profile_id",
foreignField: "_id",
as: "profile"
}}
])
```

MONGO DB – Nối các Colletions

Giải thích về aggregate framework và $lookup

MongoDB sẽ tìm tất cả documents trong profiles sao cho: users.profile_id == profiles._id

Kết quả sẽ được lưu vào một mảng mới tên là **profile** trong mỗi document users.

```
db.users.aggregate([
{
  $lookup: {
    from: "profiles",              // Collection bạn muốn join
    localField: "profile_id",      // Key trong "users"
    foreignField: "_id",           // key trong "profiles" để đối chiếu
    as: "profile"                  // Tên mảng mới chứa kết quả sau join
  }
}
])
```

MONGO DB – Nối các Colletions

Lấy thông tin từ 2 colletcions sử dụng $unwind

$unwind dùng để tách mảng profile thành object đơn (flatten).

```
db.users.aggregate([
{
$lookup: {
from: "profiles",
localField: "profile_id",
foreignField: "_id",
as: "profile"
}
},
{ $unwind: "$profile" }
])
```

MONGO DB – Nối các Colletions

2. Mối liên kết 1-N.

- Có 2 cách cơ bản để tổ chức dữ liệu cho mối liên kết 1-N
- Cách 1: Nhúng (Embedded) – khi số lượng "many" nhỏ - giới hạn (dưới vài trăm mục) và dữ liệu luôn được truy xuất cùng nhau. Hiệu suất cao vì không cần join
- Cách 2: Tham chiếu (reference) tách ra và liên kết qua ID

MONGO DB – Nối các Colletions

Ví dụ về nối liên kết 1-N: Một user có nhiều địa chỉ giao hàng

- Cách 1: Nhúng (Embedded)

```
{
  _id: 1,  name: "Alice",
  addresses: [
    { city: "Hanoi", street: "123 ABC" },
    { city: "Saigon", street: "456 XYZ" }
  ]
}
```

MONGO DB – Nối các Colletions

Ví dụ về nối liên kết 1-N: Một user có nhiều đơn hàng

- Cách 2: Tách và tham chiếu theo ID
- Collection: users

  { _id: 1, name: "Alice" }
- Collection: orders

  { _id: 101, user_id: 1, product: "Laptop" }

  { _id: 102, user_id: 1, product: "Phone" }

MONGO DB – Nối các Colletions

Join 2 collections

```
db.users.aggregate([
 {
  $lookup: {
    from: "orders",
    localField: "_id",
    foreignField: "user_id",
    as: "orders"    }
 }])
```

MONGO DB – Nối các Colletions

3. Mối liên kết Many-to-Many (N-N). Dùng Collection trung gian, tương tự CSDL quan hệ

- Ví dụ: Một sinh viên có thể học nhiều lớp học phần, một lớp học phần có thể có nhiều sinh viên học
- Collection: students

  { _id: 1, name: "Nguyen Van A" }, { _id: 2, name: "Tran Thi B" }
- Collection: classes

  { _id: 101, name: "Toán" }, { _id: 102, name: "Lý" }
- Collection: enrollments (bảng liên kết M:N)

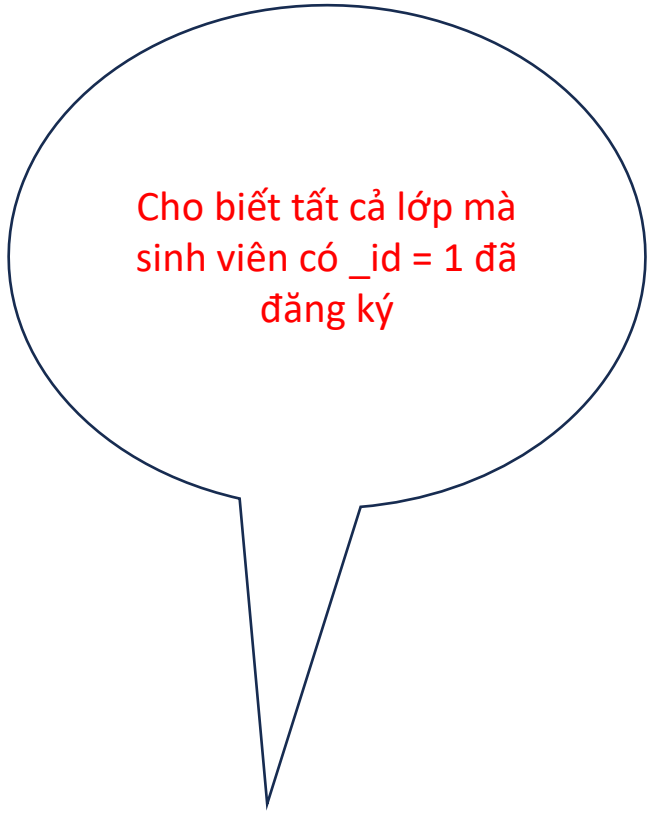
  { student_id: 1, class_id: 101 },

  { student_id: 1, class_id: 102 },

  { student_id: 2, class_id: 101 }

MONGO DB – Nối các Colletions

```
db.enrollments.aggregate([
 { $match: { student_id: 1 } },
 {   $lookup: {
    from: "classes",
    localField: "class_id",
    foreignField: "_id",
    as: "class_info"
  } },
 { $unwind: "$class_info" },
 {   $project: {
    _id: 0,
    class_id: 1,
    "class_info.name": 1
  } }
])
```



MONGO DB – Nối các Colletions

```
db.enrollments.aggregate([
 { $match: { class_id: 101 } },
 {
  $lookup: {
   from: "students",
   localField: "student_id",
   foreignField: "_id",
   as: "student_info"
  }
 },
 { $unwind: "$student_info" },
 {
  $project: {
   _id: 0,
   student_id: 1,
   "student_info.name": 1
  } }
])
```

Gợi ý khi thiết kế CSDL trong MongoDB

- MongoDB hoạt động khác với cơ sở dữ liệu quan hệ nên việc thiết kế cơ sở dữ liệu cho các ứng dụng trên MongoDB:
  - Không có chuẩn hóa
  - Không có quy tắc
- Một số gợi ý:
  - 1 - 1: Thường sử dụng cặp key-value trong document
  - 1 - vài: Thường sử dụng nhúng: Mảng…
  - 1 - nhiều: Thường sử dụng tham chiếu
  - 1 - rất nhiều: Tham chiếu
  - N-N: Tham chiếu