MongoDB - https://docs.mongodb.com/manual/

MongoDB organizes data on the database, collection, and document levels.

Remember, MongoDB itself doesn't enforce a schema, but every application needs some basic internal standards about how its data is stored.

1. Một số thao tác cơ bản

Tài liệu mẫu

Sinhvien(_id, mssv, hodem, ten, ngaysinh, gioitinh, Lophoc, dsDienthoai, diemTB)

```
[{
	"_id" : ObjectId("5f5e134b3a1929aaad36efcc"),
	"mssv" : "18366961",
	"hodem" : "Nguyễn Hoàng",
	"ten" : "Việt",
	"ngaysinh" : {
		"year" : 2000,
		"month" : 1,
		"day" : 4
	},
	"gioitinh" : "Nam",
	"malop" : "DHKTPM14BTT",
	"dsDienthoai" : [
		"0903687563",
		"0913838635"
	],
	"diemTB" : 2.96
}]
```

Lophoc(_id, malop, tenlop, ghichu)

```
[{
	"_id" : ObjectId("5f5e134a007a0bd88ff76da7"),
	"malop" : "DHKTPM14BTT",
	"tenlop" : "Đại học Kỹ thuật Phần mềm 14 Tiên tiến",
	"ghichu" : "Công nghệ thông tin"
}]
```

- Khởi động MongoDB Server, sử dụng command line:
  `mongo host:port`
  hoặc với default port (27017) `mongo`
- Liệt kê danh sách các database:
  `show dbs`
- Chuyển quyền db:

```
use db_name
```

- Liệt kê danh sách các collection có trong một database

```
show collections
```

- Sao chép database

```
db.copyDatabase(frmDBName, toDBName, [frmHostName], [user], [pwd])
```

- Chèn 1 *(insertOne)* hoặc n *(insertMany)* document:

```
db.collection.insert(<document or array of documents>)
```

```
> db. dsLophoc.insertOne({
        "malop": "DHKHMT13C",
        "tenlop": "Đại học Khoa học Máy tính 13C",
        "ghichu": "Công nghệ thông tin"
})
```

Note: *Thuộc tính _id được thêm vào document, có thể được xem như là khóa chính của document. Mỗi document trong MongoDB đều phải có _id, nếu ta không chỉ định _id thì ObjectID sẽ tự động phát sinh và thêm vào document ở thời điểm runtime.*

- Import dữ liệu file json (lophocs.json, sinhviens.json) vào mongo

```
mongoimport --host localhost --port 27017 --db qlsvdb --collection dsLophoc --file lophocs.json --jsonArray
mongoimport --host localhost --port 27017 --db qlsvdb --collection dsSinhvien --file sinhviens.json --jsonArray
```

- Cập nhật 1 *(updateOne)* hoặc n *(updateMany)* document:

```
db.collection.update(<query>,<update>)
```

```
db. dsLophoc.updateMany({},{$set:{"ghichu":"Công nghệ thông tin - Đại học công Nghiệp TPHCM"}})
```

- Xóa 1 *(deleteOne)* hoặc n *(deleteMany)* document:

```
db.collection. delete(<filter>)
db.collection. remove(<filter>)
```

```
db. dsLophoc.deleteOne({malop:"DHKHMT13C"})
```

Note: *Trên thực tế việc tạo database là không bắt buộc. Database và collection chỉ được tạo ra khi có document được thêm vào lần đầu tiên, cấu trúc của document cũng không cần được xác định trước.*

2. Constructing queries

Operators

Operator	Comparison Query Operators
$eq	So sánh bằng **Syntax:** {<field>: {$eq: <value>}} **Example:** Tìm lớp học có mã lớp học DHKTPM14BTT db. dsLophoc.find({malop:{$eq:"DHKTPM14BTT"}}) db. dsLophoc.find({malop:"DHKTPM14BTT"})
$ne	Không bằng **Syntax:** {field: {$ne: value}}
$gt	Lớn hơn
$gte	Lớn hơn hoặc bằng **Example:** Số sinh viên có điểm trung bình từ 5 trở lên db.dsSinhvien.find({diemTB:{$gte:5}}).count()
$lt	Nhỏ hơn
$lte	Nhỏ hơn hoặc bằng
$lt	Nhỏ hơn
$in	Bằng bất kỳ giá trị nào trong mảng **Syntax:** {field: {$in: [<value1>, <value2>, ... <valueN>]}} **Example:** Danh sách sinh viên có điểm trung bình là 9.85 hoặc 9.95 db.dsSinhvien.find({diemTB:{$in:[9.85, 9.95]}}).pretty()
$nin	Không nằm trong các giá trị trong mảng **Syntax:** {field: {$nin: [<value1>, <value2> ... <valueN>]}}
Operator	**Logical Query Operators¶**
$not	**Syntax:** {field: {$not: {<operator-expression>}}}
$or	**Syntax:** {$or: [{ <expression1>}, { <expression2>}, ... , { <expressionN>}]}
$and	**Syntax:** {$and: [{<expression1>}, {<expression2>} , ... , { <expressionN>}]} **Example:** Số sinh viên có điểm trung bình từ 7.0 đến nhỏ hơn 8.5 db.dsSinhvien.find({$and:[{diemTB:{$gte:7.0}},{diemTB:{$lt:8.5}}]}).count()
$nor	false tất cả các biểu thức truy vấn trong mảng **Syntax:** {$nor:[{<expression1>}, {<expression2>}, ... {<expressionN>}]}
Operator	**Element Query Operators**

$exists	exists → tồn tại
	Syntax: {field:{$exists: <boolean>}}
	Example: Giả sử ta chèn document sau, và yêu cầu tìm danh sách sinh viên chưa có điểm trung bình
	db.dsSinhvien.insert({
	"mssv" : "18745491",
	"hodem" : "Trương Đức",
	"ten" : "Hoàng",
	"ngaysinh" : {

<table>
<tr><td></td><td>
"year" : 2000,
"month" : 11,
"day" : 19
},
"gioitinh" : "Nam",
"malop" : "DHKTPM14BTT",
"dsDienthoai" : [
"0903933485",
"0913896940"
]
})

db.dsSinhvien.find({diemTB:{$exists:false}})
</td></tr>
<tr><td>$type</td><td>Lọc theo kiểu dữ liệu (Available Types → Bảng bên dưới)
Syntax: {field:{$type:<BSON type>}}
Example: Giới tính có kiểu dữ liệu là boolean.
{"gioitinh":{$type:"bool"}}
{"gioitinh":{$type:8}}</td></tr>
<tr><td>Operator</td><td>Evaluation Query Operators</td></tr>
<tr><td>$expr</td><td>Cho phép sử dụng các biểu thức tổng hợp trong ngôn ngữ truy vấn
Syntax: { $expr: { <expression> } }
Ví dụ:
<table>
<tr><td>{ "_id" : 1, "category" : "food", "budget": 400, "spent": 450 }
{ "_id" : 2, "category" : "drinks", "budget": 100, "spent": 150 }
{ "_id" : 3, "category" : "clothes", "budget": 100, "spent": 50 }
{ "_id" : 4, "category" : "misc", "budget": 500, "spent": 300 }
{ "_id" : 5, "category" : "travel", "budget": 200, "spent": 650 }</td></tr>
<tr><td>Những document có spent lớn hơn budget
{ $expr: { $gt: ["$spent" , "$budget"] } }</td></tr>
</table>
</td></tr>
<tr><td>$regex</td><td>So khớp chuỗi với biểu thức chính quy
Syntax:
{ <field>: { $regex: /pattern/, $options: '<options>' } }
{ <field>: { $regex: 'pattern', $options: '<options>' } }
{ <field>: { $regex: /pattern/<options> } }
{ <field>: /pattern/<options> }
<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>i</td><td>Case insensitivity: Không phân biệt chữ thường chữ hoa</td></tr>
<tr><td>^</td><td>Bắt đầu</td></tr>
<tr><td>$</td><td>Kết thúc</td></tr>
</table>
</td></tr>
</table>

	Example: a. Tên những sinh viên bắt đầu là ký tự "Tr", không phân biệt chữ hoa chữ thường. db.dsSinhvien.find({ten:{$regex:/^tr/i}}) db.dsSinhvien.find({ten:/^tr/i}) b. Những sinh viên kết thúc có lót chữ "Van", không phân biệt chữ hoa chữ thường. db.dsSinhvien.find({hodem:{$regex:/văn$/,$options:"i"}}) db.dsSinhvien.find({hodem:/văn$/i})

$text	Text search
	Thực hiện tìm chuỗi xuất hiện trong nội dung của các fields đã tạo text index
	Syntax:
	{
	$text:
	{
	$search: <string>,
	$language: <string>,
	$caseSensitive: <boolean>,
	$diacriticSensitive: <boolean>
	}
	}
	Note:
	Trước khi thực hiện "text search" → Nhớ phải tạo "text index"
	Example:
	a. Ta có text index sau:
	db.dsSinhvien.createIndex({ten:"text"})
	Khi thực hiện câu truy vấn sau:
	db.dsSinhvien.find({$text:{$search:"luân",$caseSensitive:false}})
	→Những sinh viên có tên chứa chữ "luân" được in ra.
	b. Ta có text index sau:
	db. dsSinhvien.createIndex({"$**":"text"})
	Khi thực hiện câu truy vấn sau:
	db.dsSinhvien.find({$text:{$search:"trường",$caseSensitive:false}})
	→Tất cả các field chứa giá trị chuỗi, mà chứa chữ "Trường" được in ra.
	db. dsSinhvien.find({$text:{$search:"trường",$caseSensitive:true}})
	→Phân biệt chữ hoa chữ thường, default là false

$mod	**Syntax:** {field:{$mod:[divisor, remainder]}} **Example:** Những số là bội số của 50 db.numbers.find({num:{$mod:[50,0]}})
Operator	**Array Query Operators**

$elemMatch	Lọc điều kiện trong subdocument trong mảng, *chỉ nên dùng khi muốn so trùng từ 2 điều kiện trong subdocument.*
	Syntax: { <field>: { $elemMatch: { <query1>, <query2>, ... } } }
	Example: Các document có school là 102 và age là 11
	{
	"_id": 1,
	"zipcode": *"63109"*,
	"students": [
	{"name": *"john"*, "school": 102, "age": 10},
	{"name": *"jess"*, "school": 102, "age": 11},
	{"name": *"jeff"*, "school": 108, "age": 15}
	]
	}
	db.svs.find({students:{$elemMatch:{school:102,age:11}}})

$all	Trùng với tất cả các phần tử trong mảng. **Syntax:** { <field>: { $all: [<value1> , <value2> ...] } } **Example:** Danh sách các sinh viên có cả số điện thoại có dầu số là 0913 và 0914 db.dsSinhvien.find({"dsDienthoai":{$all:[/^0903/,/^0913/]}})
$size	Số phần tử trong mảng **Syntax:** { field: { $size: int_number } }) **Example:** db.collection.find({ field: { $size: 2 } })

3. Phương thức explain()

Phương thức explain(): Trả về một document mô tả quá trình cũng như các index sử dụng trong việc xử lý câu truy vấn, nó cung cấp thông tin chi tiết hữu ích khi ta muốn cố gắng tối ưu hóa truy vấn.

Syntax: db.collection.explain()

Hoặc cursor.explain()

Starting in v3.2, the $explain operator is deprecated in the mongo shell

Example:

1/ Giả sử ta có một collection gồm 50000 document như sau:

```
> for(i = 0; i < 50000; i++){
        db.numbers.save({num:i});
}
WriteResult({ “nInserted” : 1 })
```

2/ Yêu cầu tìm các số có giá trị lớn hơn 49995 trở lên *(xem thông tin quá trình xử lý câu truy vấn)*

```
> db.numbers.find({num:{$gt:49995}}).explain("executionStats")
{
```

```
    "queryPlanner" : {
        "plannerVersion" : 1,
        "namespace" : "sinhviendb.numbers",
        "indexFilterSet" : false,
        "parsedQuery" : {
            "num" : {
                "$gt" : 49995
            }
        },
        "winningPlan" : {
            "stage" : "COLLSCAN",
            "filter" : {
                "num" : {
                    "$gt" : 49995
                }
            },
            "direction" : "forward"
        },
        "rejectedPlans" : [ ]
    },
    "executionStats" : {
        "executionSuccess" : true,
        "nReturned" : 4,
        "executionTimeMillis" : 30,
        "totalKeysExamined" : 0,
        "totalDocsExamined" : 50000,
        "executionStages" : {
            "stage" : "COLLSCAN",
            "filter" : {
                "num" : {
                    "$gt" : 49995
                }
            },
            "nReturned" : 4,
            "executionTimeMillisEstimate" : 0,
            "works" : 50002,
            "advanced" : 4,
            "needTime" : 49997,
            "needYield" : 0,
            "saveState" : 50,
            "restoreState" : 50,
            "isEOF" : 1,
            "direction" : "forward",
```

```
                "docsExamined" : 50000
            }
        },
        "serverInfo" : {
            "host" : "DESKTOP-7DCNV58",
            "port" : 27017,
            "version" : "4.4.1",
            "gitVersion" : "ad91a93a5a31e175f5cbf8c69561e788bbc55ce1"
        },
        "ok" : 1
}
```

3/ Tạo một index trên field num

```
> db.numbers.createIndex({num:1})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
```

4/ Thực hiện lại câu truy vấn ở bước 2/ và so sánh kết quả

```
> db.numbers.find({num:{$gt:49995}}).explain("executionStats")
{
        "queryPlanner" : {
            "plannerVersion" : 1,
            "namespace" : "sinhviendb.numbers",
            "indexFilterSet" : false,
            "parsedQuery" : {
                "num" : {
                    "$gt" : 49995
                }
            },
            "winningPlan" : {
                "stage" : "FETCH",
                "inputStage" : {
                    "stage" : "IXSCAN",
                    "keyPattern" : {
                        "num" : 1
                    },
                    "indexName" : "num_1",
                    "isMultiKey" : false,
                    "multiKeyPaths" : {
                        "num" : [ ]
                    },
```

```
                "isUnique" : false,
                "isSparse" : false,
                "isPartial" : false,
                "indexVersion" : 2,
                "direction" : "forward",
                "indexBounds" : {
                        "num" : [
                                "(49995.0, inf.0]"
                        ]
                }
        }
        },
        "rejectedPlans" : [ ]
},
"executionStats" : {
        "executionSuccess" : true,
        "nReturned" : 4,
        "executionTimeMillis" : 4,
        "totalKeysExamined" : 4,
        "totalDocsExamined" : 4,
        "executionStages" : {
                "stage" : "FETCH",
                "nReturned" : 4,
                "executionTimeMillisEstimate" : 0,
                "works" : 5,
                "advanced" : 4,
                "needTime" : 0,
                "needYield" : 0,
                "saveState" : 0,
                "restoreState" : 0,
                "isEOF" : 1,
                "docsExamined" : 4,
                "alreadyHasObj" : 0,
                "inputStage" : {
                        "stage" : "IXSCAN",
                        "nReturned" : 4,
                        "executionTimeMillisEstimate" : 0,
                        "works" : 5,
                        "advanced" : 4,
                        "needTime" : 0,
                        "needYield" : 0,
                        "saveState" : 0,
                        "restoreState" : 0,
```

```
                "isEOF" : 1,
                "keyPattern" : {
                        "num" : 1
                },
                "indexName" : "num_1",
                "isMultiKey" : false,
                "multiKeyPaths" : {
                        "num" : [ ]
                },
                "isUnique" : false,
                "isSparse" : false,
                "isPartial" : false,
                "indexVersion" : 2,
                "direction" : "forward",
                "indexBounds" : {
                        "num" : [
                                "(49995.0, inf.0]"
                        ]
                },
                "keysExamined" : 4,
                "seeks" : 1,
                "dupsTested" : 0,
                "dupsDropped" : 0
        }
    }
    },
    "serverInfo" : {
        "host" : "DESKTOP-7DCNV58",
        "port" : 27017,
        "version" : "4.4.1",
        "gitVersion" : "ad91a93a5a31e175f5cbf8c69561e788bbc55ce1"
    },
    "ok" : 1
}
```


4. Available Types

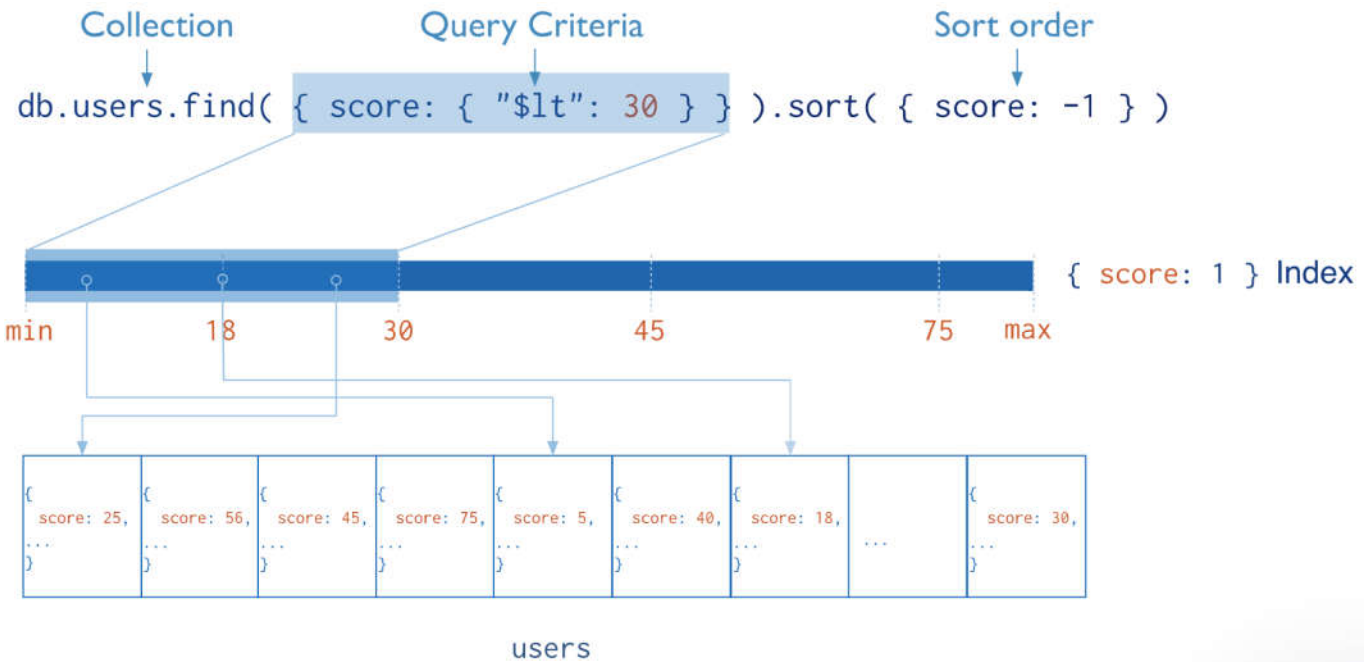
Type	Number	Alias	Notes
Double	1	“double”	
String	2	“string”	
Object	3	“object”	
Array	4	“array”	
Binary data	5	“binData”	
Undefined	6	“undefined”	Deprecated.

ObjectId	7	“objectId”	
Boolean	8	“bool”	
Date	9	“date”	
Null	10	“null”	
Regular Expression	11	“regex”	
DBPointer	12	“dbPointer”	Deprecated.
JavaScript	13	“javascript”	
Symbol	14	“symbol”	Deprecated.
JavaScript (with scope)	15	“javascriptWithScope”	
32-bit integer	16	“int”	
Timestamp	17	“timestamp”	
64-bit integer	18	“long”	
Decimal128	19	“decimal”	New in version 3.4.
Min key	-1	“minKey”	
Max key	127	“maxKey”	

5. Indexes

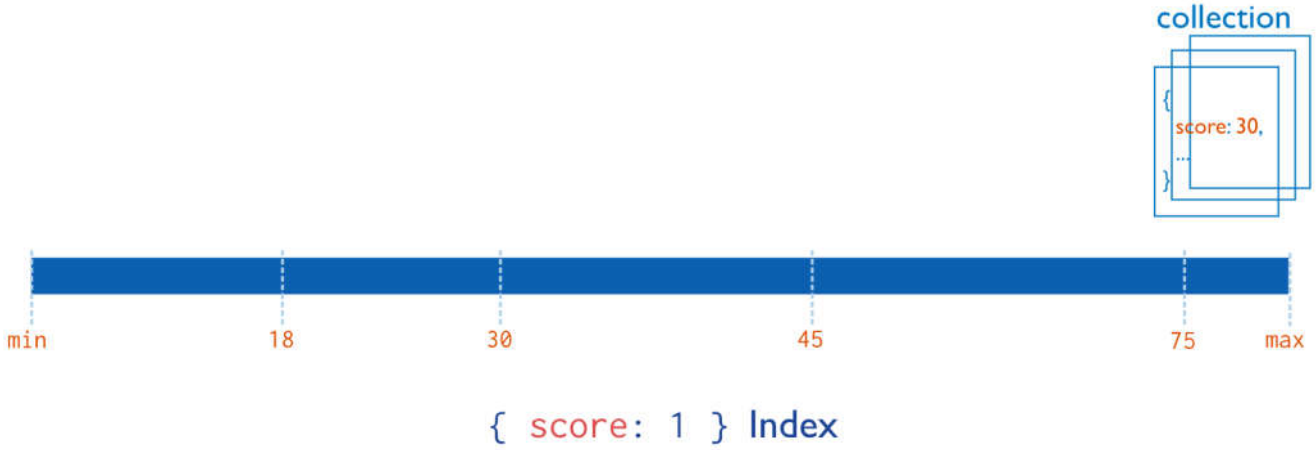
Indexes tăng hiệu quả của các truy vấn trong MongoDB.

- Nếu không có các index, MongoDB phải quét qua từng document trong toàn bộ collection để tìm document thỏa trong câu truy vấn.
- Ngược lại, nếu có index thích hợp tồn tại, MongoDB có thể sử dụng chỉ mục để giới hạn số lượng document mà nó cần phải tìm.
- Tạo một chỉ mục không thay đổi tập dữ liệu ban đầu; nó chỉ tạo một cấu trúc dữ liệu mới và nó trỏ đến đối tượng ban đầu.

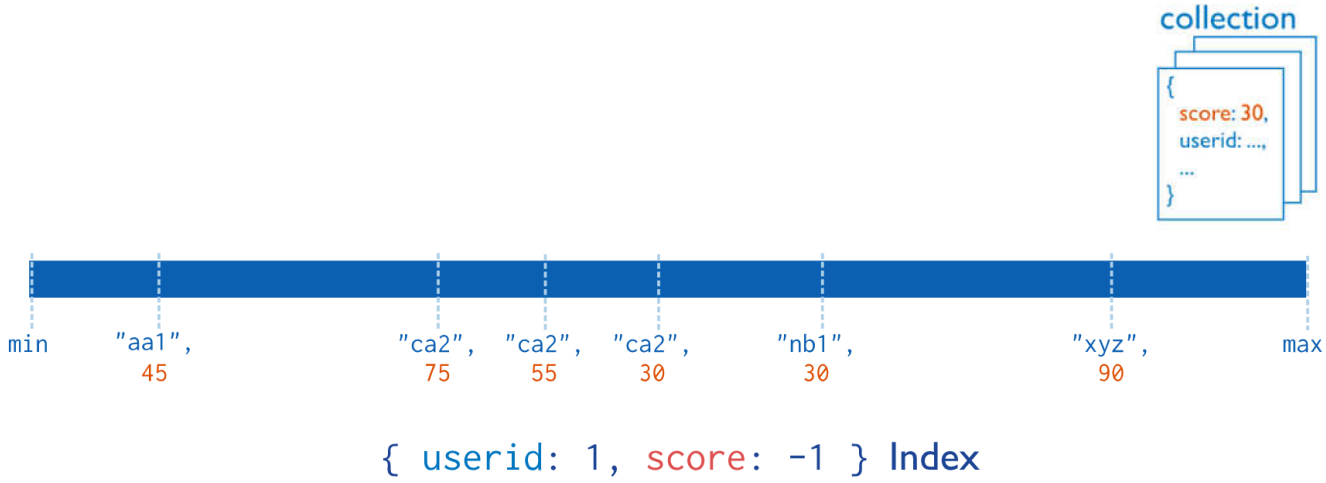


Index Types

Single Field



Compound Index



STT	Collection Methods
1	**Tạo index** **Syntax: db.collection.createIndex(keys, options)¶**
	1.1. Tạo các index trên collection - an ascending index → 1 - a descending index → -1 **Example:** a. Tạo index cho tập các số nguyên db.numbers.createIndex({num:1}) b. Tạo một unique index tăng dần trên thuộc tính mã lớp db.dsLophoc.createIndex({malop:1},{unique:1}) c. Tạo một unique index tăng dần trên field mã số sinh viên db. dsSinhvien.createIndex({"mssv":1},{unique:true}) d. Tạo compound index trên field họ và tên sinh viên db. dsSinhvien.createIndex({"ten":1,"hodem":1})
	1.2. Tạo các text index trên collection

	Note: Một collection có thể có nhiều nhất một text index. **Example:** Tạo compound text index trên field họ và tên sinh viên. db. dsSinhvien.createIndex({"ten":"text","hodem":"text"})
	1.3. Wildcard Text Indexes - Khi muốn tạo text index trên nhiều field, ta có thể sử dụng ký tự đại diện ($**). Với text index sử dụng ký tự đại diện này, MongoDB lập index mọi field có chứa dữ liệu chuỗi cho mỗi document trong collection. - Index này cho phép tìm kiếm văn bản trên tất cả các field có nội dung là chuỗi. - Với Wildcard Text Indexes, cũng như với tất cả các text index khác, có thể là một phần của compound index. **Example:** Lập index mọi field có chứa dữ liệu chuỗi db. dsSinhvien.createIndex({"$**":"text"}) →Text search db. dsSinhvien.find({$text:{$search: "Long", $caseSensitive:false}})
	1.4. Unique index **Syntax:** db.collection.createIndex(keys, { unique: true }) Tạo unique index, có thể tạo unique index trên nhiều field **Example:** Tạo unique index trên field "mã lớp" db. dsLophoc.createIndex({malop:1},{unique:true})
	1.5. Partial Index **Syntax:** db.collection.createIndex(keys,{partialFilterExpression: {filter_condition}}) Filter_condition sử dụng: - $eq expressions, - $exists: true expression, - $gt, $gte, $lt, $lte expressions, - $type expressions, - $and operator at the top-level only **Example:** Tạo index trên field "mã số sinh viên" cho những document có điểm TB từ 8 trở lên. db.dsSinhvien.createIndex({"mssv":1},{partialFilterExpression:{"diemtb":{ $gte:8}}})
2	**db.collection.ensureIndex(keys, options)¶**
	Note: Deprecated since version 3.0.0
3	**Xóa index**
	3.1 Xóa index chỉ định có trong một collection **Syntax:** db.collection.dropIndex(index)¶ Tham số index có thể là string hoặc document. **Example:** Xóa index trên field mã số sinh viên đã tạo trên collection dsSinhvien db.dsSinhvien.dropIndex("mssv_1")

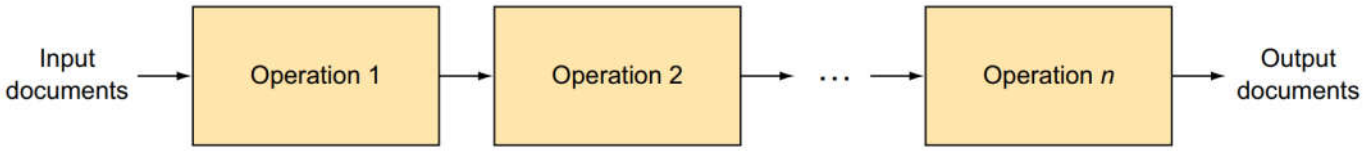
	Xóa unique index trên mã lớp đã tạo trong collection dsLophoc db.dsLophoc.dropIndex({"malop" : 1})
	3.2 Xóa mọi index có trong collection **Syntax:** db.collection.dropIndexes() Xóa index chỉ định hoặc toàn bộ các index có trong 1 collection *(ngoại trừ index trên trường _id)* **Example:** Xóa tất cả các index có trong collection dsLophoc *(ngoại trừ index trên trường _id)* db.dsLophoc.dropIndexes()
4	**Index Names**
	Mặc định tên của index là ghép giữa khóa của index và hướng của index, cách nhau bằng dấu _. Ngoài ra, ta có thể tạo tên cho index để dễ đọc, dễ nhớ. **Example:** db.dsLophoc.createIndex({malop:1}) → Tên index tự động tạo ra là malop_1 db.dsSinhvien.createIndex({"ten":1,"hodem":1},{name:"ho_ten_text"}) →Tên do người dùng đặt
5	**Liệt kê danh sách các index có trong collection**
	Syntax: db.collection.getIndexes() Trả về một mảng các document mô tả các index có trong collection. **Example:** Liệt kê tập các index trong collection có tên: dsLophoc db.dsLophoc.getIndexes()

6. Aggregation framework

Aggregation operations – Các toán tử xử lý tập các dữ liệu và trả về kết quả sau khi tính toán, nhóm các giá trị dữ liệu từ nhiều document lại với nhau và có thể thực hiện nhiều thao tác trên nhóm và trả về một kết quả.

6.1 Aggregation Pipeline

MongoDB's aggregation framework - Được mô hình hóa dựa trên khái niệm đường ống xử lý dữ liệu *(pipeline)*. Các document được đưa vào pipeline nhiều giai đoạn để thống kê tính toán ra được kết quả tổng hợp.



Đầu ra của mỗi giai đoạn là đầu vào của giai đoạn kết tiếp. Mỗi giai đoạn chỉ thực hiện duy nhất một thao tác *(single operation)* dựa trên tập document đầu vào.

Syntax: db.collection.aggregate([{ <stage> }, ...])

Aggregation pipeline operations

Stage	Aggregation Pipeline Stages - Description
$project	Chọn các field mong muốn **Example:** Danh sách sinh viên gồm: Mã số, tên và điểm trung bình `db.dsSinhvien.aggregate([{$project:{_id:0, mssv:1, ten:1, diemTB:1}}])` **Output:** { "mssv" : "18160161", "ten" : "Trường", "diemTB" : 4.91 } { "mssv" : "18233981", "ten" : "Phát", "diemTB" : 7.57 }

$match	Lọc các document
	Example: Danh sách sinh viên gồm: Mã số, tên và điểm trung bình; có điểm từ 9.5 trở lên.
	`db.dsSinhvien.explain("executionStats").aggregate([`
	`{$match:{diemTB:{$gte:9.5}}},`
	`{$project:{_id:0, mssv:1, ten:1, diemTB:1}}])`
	Output:
	{ "mssv" : "18801441", "ten" : "Nghĩa", "diemTB" : 9.53 }
	{ "mssv" : "18427421", "ten" : "Thiện", "diemTB" : 9.63 }

$limit	Giới hạn số document
	Example: Danh sách sinh viên có điểm trung bình từ 9.5 trở lên, trong 50 sinh viên đầu tiên trong danh sách.
	`db.dsSinhvien.aggregate([`
	`{$limit:50},`
	`{$match:{diemTB:{$gte:9.5}}},`
	`{$project:{_id:0, mssv:1, ten:1, diemTB:1}}])`

$skip	Bỏ qua bao nhiêu document
	Example: Bỏ qua 5 document trong tập kết quả
	`db.dsSinhvien.aggregate([`
	`{$match:{diemTB:{$gte:9.5}}},`
	`{$project:{_id:0, mssv:1, ten:1, diemTB:1}},`
	`{$skip:5}])`

$unwind	Phân rã mỗi document riêng biệt cho từng phần tử trong mảng.
	Example:
	1/ Sau khi thực thi câu truy vấn sau:
	`db.dsSinhvien.aggregate([`
	`{$project:{_id:0, mssv:1, ten:1, diemTB:1, dsDienthoai:1}}])`
	2/ Mỗi document trong tập kết quả có cấu trúc như sau:
	`{`
	`"mssv" : "18655681",`
	`"ten" : "Thắng",`
	`"dsDienthoai" : [`

	"0903851555",
	"0913520111"
	],
	"diemTB" : 8.28
	}
	3/ Thực hiện stage kế tiếp là $unwind trên field dsDienthoai
	db.dsSinhvien.aggregate([
	{$project:{_id:0, mssv:1, ten:1, diemTB:1, dsDienthoai:1}},
	{$unwind:"$dsDienthoai"}])
	4/ Mỗi document trong tập kết quả lúc này có cấu trúc như sau:
	{
	"mssv" : "18655681",
	"ten" : "Thắng",
	"dsDienthoai" : "0903851555",
	"diemTB" : 8.28
	}
	{
	"mssv" : "18655681",
	"ten" : "Thắng",
	"dsDienthoai" : "0913520111",
	"diemTB" : 8.28
	}

$group	Nhóm các document theo tiêu chí, tính toán theo nhóm.
	Syntax:
	{
	$group:{
	_id: <expression>, // Group By Expression
	<field1>: { <accumulator1> : <expression1> },
	}
	}
	Example: 1/ Đếm số sinh viên có điểm trung bình từ 9.0 trở lên.
	db.dsSinhvien.aggregate([
	{$match:{diemTB:{$gte:9.0}}},
	{$group:{_id:null,tongSV:{$sum:1}}},
	{$project:{_id:0, tongSV:1}}])
	2/ Đếm số sinh viên có điểm trung bình từ 9.0 trở lên theo từng lớp. Thông tin bao gồm: Mã lớp và tổng số sinh viên.
	db.dsSinhvien.aggregate([
	{$match:{diemTB:{$gte:9.0}}},
	{$group:{_id:{malop:"$malop"},tongSV:{$sum:1}}}])

<table>
<tr><td>$sort</td><td>Sắp xếp tập document
Example: Đếm số sinh viên có điểm trung bình từ 9.0 trở lên theo từng lớp. Thông tin bao gồm: Mã lớp và tổng số sinh viên. Sắp xếp giảm dần theo số lượng sinh viên.
db.dsSinhvien.aggregate([
{$match:{diemTB:{$gte:9.0}}},
{$group:{_id:{malop:"$malop"},tongSV:{$sum:1}}},
{$sort:{tongSV:-1}}])</td></tr>
<tr><td>$out</td><td>Ghi tập kết quả ra thành một collection
Syntax: { $out: { db: "<output-db>", coll: "<output-collection>" } }
Example: Xuất danh sách sinh viên có điểm từ 8.5 trở lên ra thành collection riêng biệt
db.dsSinhvien.aggregate([
{$match:{diemTB:{$gte:8.5}}},
{$out:"dsSinhvienGioi"}])
Xem kết quả đầu ra:
db.dsSinhvienGioi.find()</td></tr>
<tr><td>$replaceRoot</td><td>Syntax: { $replaceRoot: { newRoot: <replacementDocument> } }
Example:
db.dsSinhvien.aggregate([{$replaceRoot:{newRoot:"$ngaysinh"}}])
Output:
{ "year" : 2000, "month" : 1, "day" : 28 }
{ "year" : 2000, "month" : 11, "day" : 2 }</td></tr>
<tr><td>$replaceWith</td><td>Syntax: { $replaceWith: <replacementDocument> }
Example: Danh sách sinh viên gồm: Mã số, họ và tên, year, month, day
db.dsSinhvien.aggregate([
{$replaceWith:
{$mergeObjects:[{
mssv:"$mssv",
hoten:{$concat:["$hodem"," ","$ten"]}},
"$ngaysinh"]}}])
Output:
{ "mssv": "17861241", "hoten": "Trần Xuân Quang", "year" : 1999, "month" : 7, "day" : 22 }
{"mssv": "18160161", "hoten": "Nguyễn Ngọc Trường", "year" : 2000, "month" : 10, "day" : 3 }</td></tr>
<tr><td>$lookup</td><td>Thực hiện phép kết bằng
Syntax:
{
$lookup:{
from: <collection to join>,</td></tr>
</table>

	localField: <field from the input documents>,
	foreignField: <field from the documents of the "from" collection>,
	as: <output array field>
	}
	}
	Example: 1/ Danh sách sinh viên của lớp có tên lớp "Đại học Kỹ thuật Phần mềm 14 Tiên tiến"
	db.dsSinhvien.aggregate([
	{
	$lookup:{
	from:"dsLophoc",
	localField:"malop",
	foreignField:"malop",
	as:"result"}
	},
	{
	$match:{
	"result.tenlop":"Đại học Kỹ thuật Phần mềm 14 Tiên tiến"}
	}])
	2/ Tính sĩ số thực trên từng lớp. Thông tin gồm: Mã lớp, tên lớp và sĩ số.
	db.dsLophoc.aggregate([
	{
	$lookup:{
	from:"dsSinhvien",
	localField:"malop",
	foreignField:"malop",
	as: "dssv"
	}
	},
	{
	$project:{
	malop:1, tenlop:1, _id:0, siso:{$size:"$dssv"}
	}
	}])

$addFields	Thêm các field vào document có trước. **Syntax:** { $addFields: { <newField>: <expression>, ... } } **Example:** Tính tuổi của sinh viên. Thông tin gồm thông tin của sinh viên và tuổi. db.dsSinhvien.aggregate([{$addFields: {"tuoi":

	{$subtract:[{$year:{date:new Date()}}, "$ngaysinh.year"] } } }])
$count	Số lượng document có trong stage **Syntax:** { $count: <string> } **Example:** Đếm tổng số document có trong có trong collection dsLophoc db.dsLophoc.aggregate({$count:"tongsoLophoc"})
$merge	Ghi kết quả tính toán thành một collection riêng biệt. Nên là stage cuối cùng trong aggregation pipeline. **Syntax:** { $merge: { into: <collection> -or- { db: <db>, coll: <collection> }, on: <identifier field> -or- [<identifier field1>, ...], // Optional let: <variables>, // Optional whenMatched: <replace|keepExisting|merge|fail|pipeline>, // Optional whenNotMatched: <insert|discard|fail> // Optional } } *Note: $merge requires a unique, index with keys that correspond to the on identifier fields* **Example:** Tạo một collection tên "dssv" chứa danh sách các sinh viên. Thông tin gồm: Mã số, họ, tên, giới tính và mã lớp. Sắp xếp theo mã lớp, trùng mã lớp sắp xếo tên sinh viên tăng dần. db.dssv.createIndex({mssv:1},{unique:1}) db.dsSinhvien.aggregate([{$project:{_id:0,mssv:1,hodem:1,ten:1,gioitinh:1,malop:1}}, {$sort:{malop:1, ten:1}}, {$merge:{into:"dssv", on:"mssv"}}])
$sortByCount	Gom nhóm và đếm số lượng trong nhóm, kết quả sắp xếp giảm dần theo số lượng. **Syntax:** { $sortByCount: <expression> } **Example:** Tính sĩ số trên từng lớp, sắp xếp giảm dần theo cột sĩ số. db.dsSinhvien.aggregate([{$sortByCount:"$malop"}, {$project:{siso:"$count"}}])
Operator	**Aggregation Pipeline Operators - Accumulators ($group)**
$addToSet	**Syntax:** { $addToSet: <expression> }
$avg	**Syntax:** { $avg: <expression> } **Example:** Tính tuổi trung bình của các sinh viên. db.dsSinhvien.aggregate([

	{$group: {_id:null, tuoiTB: {$avg: {$subtract:[{$year:{date:new Date()}}, "$ngaysinh.year"] } } } }, {$project:{_id:0,tuoiTB:1}}]) **Output:** { "tuoiTB" : 20.36082474226804 }
$max	**Syntax:** { $max: <expression> } **Example:** Tìm điểm TB lớn nhất db.dsSinhvien.aggregate([{$group:{_id:null,diemTBMax:{$max:"$diemTB"}}}, {$project:{_id:0}}])
$min	**Syntax**: { $min: <expression> }
$push	**Syntax**: { $push: <expression> } **Example:** Lọc các sinh viên theo từng lớp db.dsSinhvien.aggregate([{$group:{ _id:"$malop", dssv:{$push:{mssv:"$mssv",ten:"$ten"}} }}])
$sum	**Syntax**: { $sum: <expression> } **Example:** Số lượng sinh viên có điểm TB từ 8.5 trở lên của từng lớp. Thông tin bao gồm: Mã lớp, tên lớp và số lượng sinh viên. db.dsLophoc.aggregate([{$lookup: {from:"dsSinhvien", localField:"malop", foreignField:"malop", as:"result"} }, {$unwind:"$result"}, {$match:{"result.diemTB":{$gte:8.5}}}, {$group:{_id:{malop:"$result.malop", tenlop:"$tenlop"}, soluong:{$sum:1}}}])

SQL to Aggregation Mapping Chart	
SQL Terms, Functions, and Concepts	**MongoDB Aggregation Operators**
WHERE	$match

SQL to Aggregation Mapping Chart	
SQL Terms, Functions, and Concepts	**MongoDB Aggregation Operators**
GROUP BY	$group
HAVING	$match
SELECT	$project
ORDER BY	$sort
LIMIT	$limit
SUM()	$sum
COUNT()	$sum $sortByCount
join	$lookup
SELECT INTO NEW_TABLE	$out
MERGE INTO TABLE	$merge *(Available starting in MongoDB 4.2)*
UNION ALL	$unionWith *(Available starting in MongoDB 4.4)*

Bài tập

Bài 1

Viết các câu truy vấn trên tài liệu hướng dẫn với dữ liệu mẫu sau:

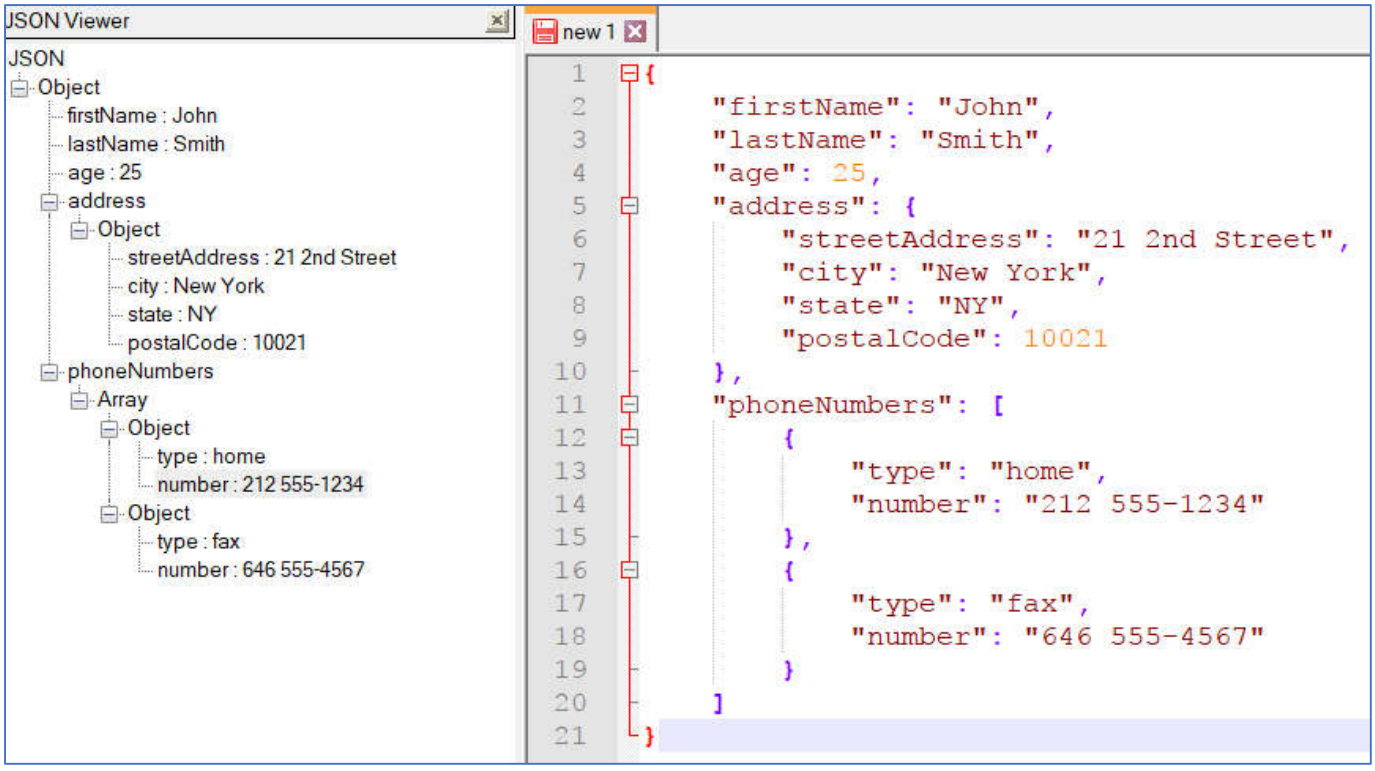
Tài liệu json dạng:

```
[
        {
                "malop": "DHKHMT13A",
                "tenlop": "Đại học Khoa học Máy tính 13A",
                "ghichu": "Công nghệ thông tin",
                "dssv": [
                        {
                                "mssv": "17640461",
                                "hodem": "Đặng Lê Nguyên",
                                "ten": "Lương",
                                "ngaysinh": {
                                        "year": 1999,
                                        "month": 4,
                                        "day": 10
                                },
                                "gioitinh": "Nam",
                                "dsDienthoai": [
                                        "0903603220",
                                        "0913612261"
                                ],
                                "diemTB": 8.21
                        },
                        {
```

```
                        "mssv": "17659941",
                        "hodem": "Trần Trọng",
                        "ten": "Thành",
                        "ngaysinh": {
                                "year": 1999,
                                "month": 12,
                                "day": 11
                        },
                        "gioitinh": "Nam",
                        "dsDienthoai": [
                                "0903521617",
                                "0913116610"
                        ],
                        "diemTB": 6.83
                }
        ]
    }
]
```


Bài 2

Cho tại liệu JSON có cấu trúc như hình sau



Hãy viết các câu truy vấn sau:

1. Tìm những person đang sống tại 1 bang cho trước
2. Tìm số điện thoại của 1 person cho trước khi biết firstName và lastName
3. Tìm những person có tuổi trên 50
4. Tính độ tuổi trung bình cho mỗi bang
5. Tính tổng số person cho mỗi bang