



CHƯƠNG 2

HỆ QUẢN TRỊ CSDL MONGODB

NỘI DUNG

1. Tổng quan về MongoDB

2. Phiên bản và các công cụ

3. Mô hình dữ liệu

4. Các lệnh thao tác cơ bản

5. Sao lưu và phục hồi dữ liệu

6. Truy vấn dữ liệu

TỔNG QUAN MONGODB

- ❖ Là một cơ sở dữ liệu mã nguồn mở
- ❖ Thuộc loại cơ sở dữ liệu NoSQL hàng đầu được hàng triệu người sử dụng.
- ❖ Là một cơ sở dữ liệu đa nền tảng, hoạt động trên các khái niệm Collection và Document.
- ❖ Được phát triển bởi 10gen năm 2007. Đến năm 2009 phát hành bản mã nguồn mở.

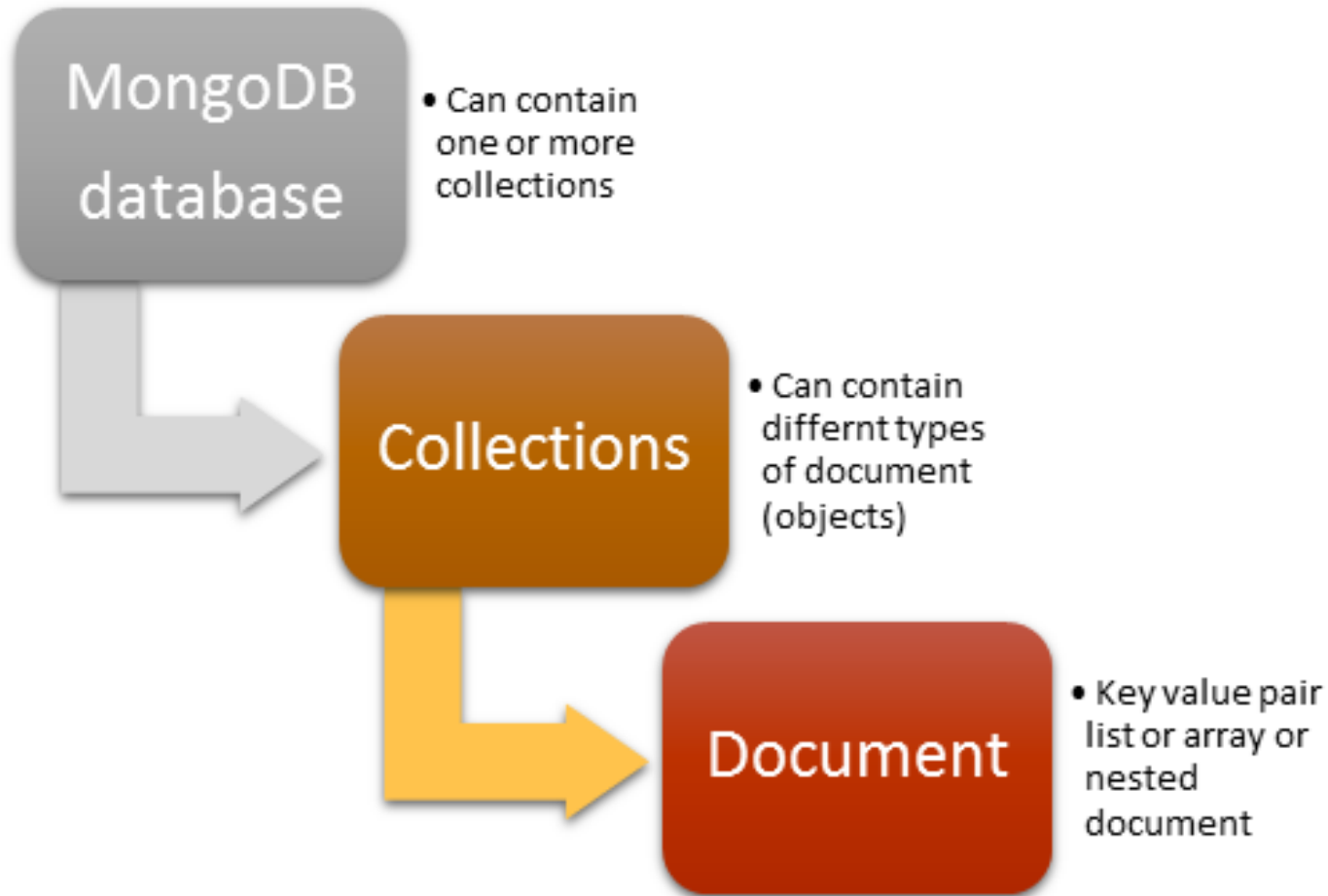
TỔNG QUAN MONGODB (tt)

- ❖ Được viết bằng C++ nên có khả năng tính toán với tốc độ cao.
- ❖ Được sử dụng bởi nhiều hãng lớn như MTV Networks, Adobe, Google, Cisco, Ebay, Facebook,...
- ❖ Khả năng tương thích: Windows, Linux, OS X, Solaris. Hỗ trợ ngôn ngữ: C, C++, C#/.Net, Java, Node.JS, PHP, Python,...

Sử dụng MongoDB ở đâu?

- ❖ Dữ liệu lớn.
- ❖ Quản lý và phân phối nội dung.
- ❖ Cơ sở hạ tầng di động và xã hội.
- ❖ Quản lý dữ liệu người dùng.
- ❖ Trung tâm dữ liệu.

Một số thuật ngữ trong MongoDB



Một số thuật ngữ trong MongoDB (tt)

❖ Database:

- Nơi chứa các Collection, giống với cơ sở dữ liệu RDBMS chúng chứa các bảng.

❖ Collection:

- Là nhóm nhiều document trong MongoDB. Có thể hiểu là một bảng trong cơ sở dữ liệu RDBMS
- 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.

Một số thuật ngữ trong MongoDB (tt)

❖ 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.

Collection

Document 1

```
{
  "id": "1",
  "name": "John Smith",
  "isActive": "true",
  "dob": "1977-03-20"
}
```

Document 2

```
{
  "id": "2",
  "fullName": "Sarah Jones",
  "isActive": "false",
  "dob": "1982-10-03"
}
```

Document 3

```
{
  "id": "3",
  "fullName": {
    "first": "Adam",
    "last": "Stark"
  },
  "isActive": "true",
  "dob": "2000-07-05"
}
```

Table

id	name	isActive	dob
1	John Smith	true	1977-03-20
2	Sarah Jones	false	1982-10-03
3	Adam Stark	true	2000-07-05

Một số thuật ngữ trong MongoDB (tt)

❖ _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.

```
{
  _id: ObjectId("5099803df3f4948bd2f98391"),
  name: { first: "Alan", last: "Turing" },
  birth: new Date('Jun 23, 1912'),
  death: new Date('Jun 07, 1954'),
  contribs: [ "Turing machine", "Turing test", "Turingery" ],
  views : NumberLong(1250000)
}
```

Một số thuật ngữ trong MongoDB (tt)

❖ 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.

```
{  
  _id: ObjectId("5099803df3f4948bd2f98391"),  
  name: { first: "Alan", last: "Turing" },  
  birth: new Date('Jun 23, 1912'),  
  death: new Date('Jun 07, 1954'),  
  contribs: [ "Turing machine", "Turing test", "Turingery" ],  
  views : NumberLong(1250000)  
}
```

Một số thuật ngữ trong MongoDB (tt)

❖ 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.

❖ BSON:

- Binary JSON dựa trên cấu trúc của JSON nhưng bổ sung thêm một số kiểu dữ liệu như ngày tháng và kiểu Binary

Kiểu dữ liệu trong MongoDB

- **String:** Là kiểu sử dụng phổ biến nhất để lưu trữ dữ liệu. String trong MongoDB phải là UTF-8.
- **Integer:** Được sử dụng để lưu các giá trị số. Integer có thể là 32 bit hay 64 bit phụ thuộc vào server của bạn.
- **Boolean:** Được dùng để lưu giá trị boolean.
- **Double:** Được sử dụng để lưu trữ các giá trị dấu phẩy động.
- **Min/ Max keys:** Được sử dụng để so sánh một giá trị với các phần tử BSON thấp nhất và cao nhất.

Kiểu dữ liệu trong MongoDB (tt)

- **Arrays:** Được dùng để lưu các mảng hoặc danh sách hoặc nhiều giá trị trong một key.
- **Timestamp:** Dùng để ghi lại khi một tài liệu đã được sửa đổi hoặc thêm vào.
- **Object:** Đây là kiểu dữ liệu được dùng cho embedded documents.
- **Null:** Dùng để lưu các giá trị là Null
- **Symbol:** Kiểu dữ liệu này được sử dụng giống như string; tuy nhiên, nó thường dành riêng cho các ngôn ngữ sử dụng một loại ký hiệu cụ thể.

Kiểu dữ liệu trong MongoDB (tt)

- **Date:** Được sử dụng để lưu trữ date và time hiện tại trong định dạng UNIX time.
- **Object ID:** Được sử dụng để lưu giữ ID của Document.
- **Binary data:** Được sử dụng để lưu giữ dữ liệu nhị phân.
- **Code:** Được sử dụng để lưu trữ JavaScript code vào trong Document.
- **Regular expression:** Được dùng để lưu trữ regular expression.

Phiên bản và công cụ thao tác

❖ Các phiên bản MongoDB

- MongoDB Community server (miễn phí)
- Enterprise Server (thương mại).

❖ Cài đặt MongoDB

- Tham khảo tài liệu thực hành.

❖ Các công cụ thao tác trên MongoDB

- MongoDB Compass
- Studio 3T
- Mongo Management Studio
- ...

Mô hình dữ liệu trong MongoDB

- ❖ Dữ liệu trong MongoDB có một Schema linh động.
- ❖ Các Document trong cùng Collection không cần thiết phải có cùng tập hợp các trường hoặc cấu trúc.
- ❖ Các trường chung trong các Document của Collection có thể giữ các kiểu dữ liệu khác nhau.

Mô hình dữ liệu trong MongoDB (tt)

- ❖ Hai mô hình dữ liệu thông dụng được sử dụng trong MongoDB
 1. Embedded Data Models
 2. References Data Models

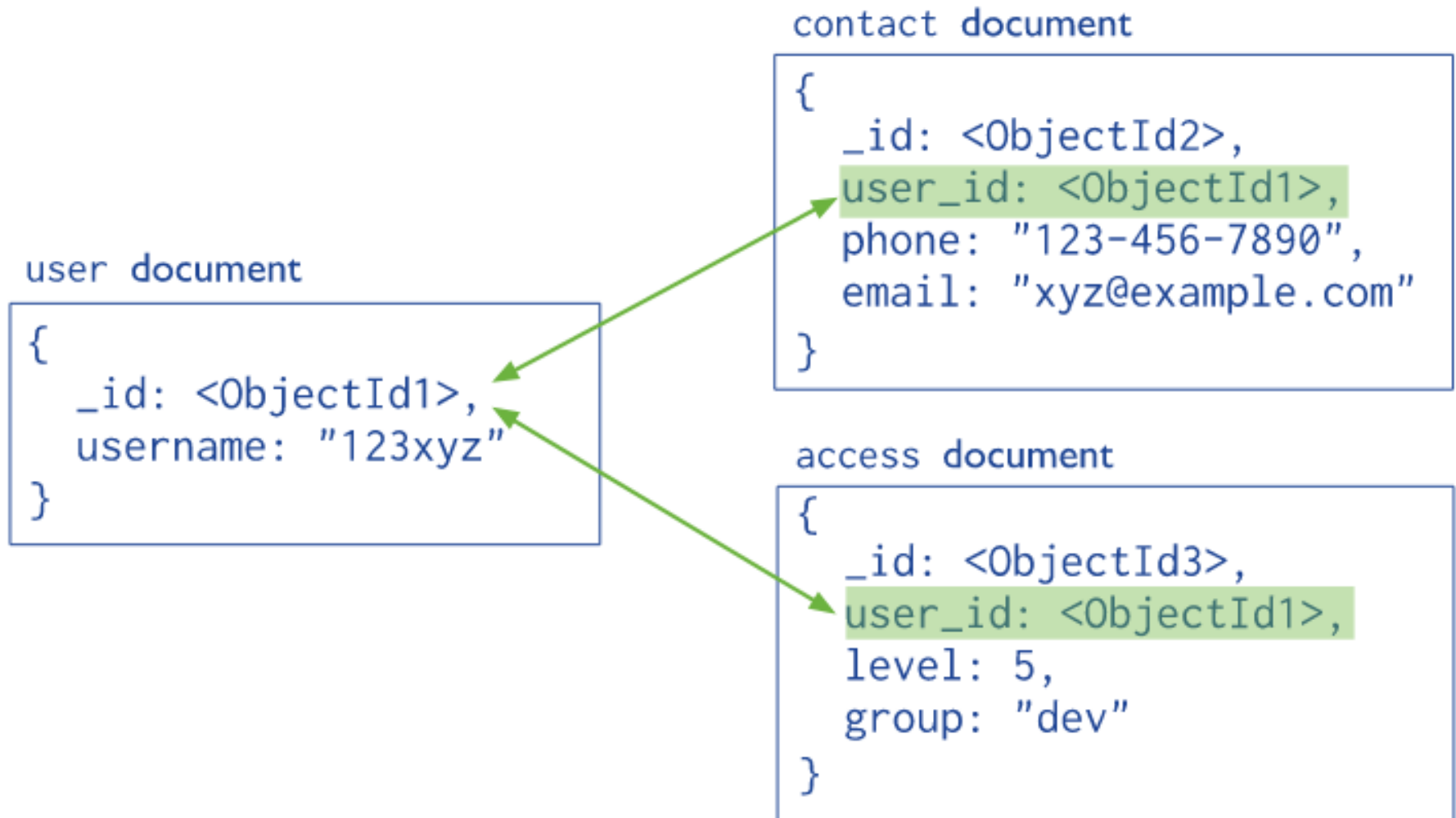
Embedded Data Models

```
{  
  _id: <ObjectId1>,  
  username: "123xyz",  
  contact: {  
    phone: "123-456-7890",  
    email: "xyz@example.com"  
  },  
  access: {  
    level: 5,  
    group: "dev"  
  }  
}
```

Embedded sub-document

Embedded sub-document

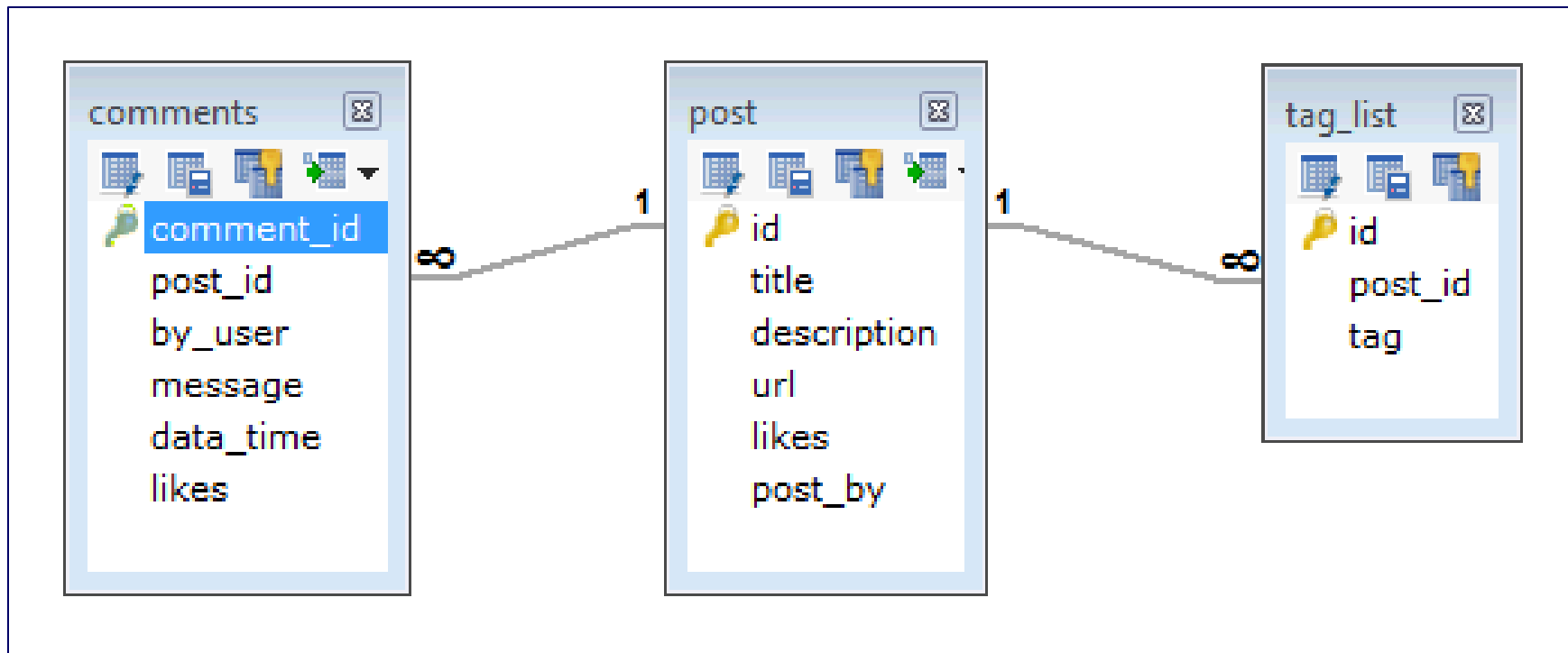
References Data Models



Mô hình dữ liệu trong MongoDB (tt)

- ❖ Ví dụ: Thiết kế mô hình dữ liệu trên MongoDB để quản lý thông tin các bài viết trên một blog theo 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.

Thiết kế trên mô hình dữ liệu quan hệ



Thiết kế trên MongoDB

```
{
  _id: POST_ID
  title: TITLE_OF_POST,
  description: POST_DESCRIPTION,
  by: POST_BY,
  url: URL_OF_POST,
  tags: [TAG1, TAG2, TAG3],
  likes: TOTAL_LIKES,
  comments: [
    {
      user: 'COMMENT_BY',
      message: TEXT,
      dateCreated: DATE_TIME,
      like: LIKES
    },
    {
      user: 'COMMENT_BY',
      message: TEXT,
      dateCreated: DATE_TIME,
      like: LIKES
    }
  ]
}
```

Mô hình dữ liệu trong MongoDB (tt)

- ❖ Chú ý: Trong RDBMS cần kết hợp ba bảng và trong MongoDB sẽ chỉ cần một Collection
- ❖ Bài tập:
 - Thiết kế cấu trúc collection để lưu các thông tin gồm: sinh viên, lớp, môn học.
 - Thiết kế cấu trúc collection để lưu thông tin bán sản phẩm tại một cửa hàng.
 - Mỗi cấu trúc collection viết 3 document tương ứng.

Thiết kế collection sinhvien (sv)

```
1. {  
2.     Masv: Mã số sinh viên,  
3.     Hoten: Họ và tên sinh viên,  
4.     Ngaysinh: ngày sinh (dữ liệu date)  
5.     Phai: Giới tính  
6.     Email: [Mảng các email]  
7.     Lop: {  
8.         Malop: Mã lớp,  
9.         Tenlop: Tên lớp  
10.    },  
11.    Monhoc:  
12.    [ {  
13.        Mamh: Mã môn học,  
14.        Tenmh: Tên môn học,  
15.        Sotc: Số tín chỉ,  
16.        Diem: Điểm thi  
17.    }, {...}  
18.    ]  
19. }
```


Thiết kế collection sinhvien (sv)

```
1. {
2.     Masv: Mã số sinh viên,
3.     Hoten: Họ và tên sinh viên,
4.     Tuoi: Tuổi sinh viên
5.     Ngaynhaphoc: ngày nhập học
6.     Phai: Giới tính
7.     Quoctich: Quốc tịch
8.     Ngoaingu: [Mảng các C.Chữ]
9.     Lop: {
10.         Malop: Mã lớp,
11.         Tenlop: Tên lớp
12.     },
13.     Monhoc:
14.     [ {
15.         Mamh: Mã môn học,
16.         Tenmh: Tên môn học,
17.         Sotc: Số tín chỉ,
18.         Diem: Điểm thi
19.     }, {...}
20. ]
21. }
```

```
{
    Masv:"sv001",
    Hoten:"Trần Minh Nghĩa",
    Tuoi:22,
    Ngaynhaphoc: new Date("2018-04-14"),
    Phai:"Nam",
    Quoctich:"Việt Nam",
    Ngoaingu:["Tiếng Anh","Tiếng Pháp"],
    Lop:{Malop:"101",Tenlop:"08DHTH"},
    Monhoc:
    [
        {Mamh:"m001",Tenmh:"Cơ sở dữ liệu",Sotc:3,Diem:7.5},
        {Mamh:"m002",Tenmh:"Toán cao cấp",Sotc:2,Diem:9},
        {Mamh:"m003",Tenmh:"Lập trình C",Sotc:3,Diem:8.5},
    ]
},
{
    Masv:"sv002",
    Hoten:"Đỗ Thị Bình",
    Tuoi: 22,
    Ngaynhaphoc: new Date("2019-07-20"),
    Phai:"Nữ",
    Quoctich:"Việt Nam",
    Ngoaingu:["Tiếng Anh","Tiếng Nga"],
    Lop:{Malop:"101",Tenlop:"08DHTH"},
    Monhoc:
    [
        {Mamh:"m001",Tenmh:"Cơ sở dữ liệu",Sotc:3,Diem:7.8},
        {Mamh:"m002",Tenmh:"Toán cao cấp",Sotc:2,Diem:8},
        {Mamh:"m004",Tenmh:"Mạng máy tính",Sotc:2,Diem:6.5},
    ]
},
}
```

Các thao tác trên Database

- ❖ Thiết lập kết nối ban đầu
- ❖ Thao tác trên Database
- ❖ Thao tác trên Collection, Document
- ❖ *Hướng dẫn tìm kiếm tài liệu MongoDB trên Google*
 - *Từ khóa tìm kiếm: getting start mongodb, mongodb document, mongodb insert document,...*

Thiết lập kết nối ban đầu

- ❖ Khởi động dịch vụ MongoDB Server
- ❖ Kết nối mongoDB từ cửa sổ lệnh cmd, thiết lập biến môi trường
- ❖ Kết nối mongoDB từ chương trình MongoDB Compass

Các lệnh thao tác cơ bản

- > mongo *//Khởi động mongodb shell*
- > show dbs *//Hiển thị danh sách database*
- > show collections *//Hiển thị danh sách Collection*
- > db *//Hiển thị database hiện tại*
- > use database_name *//Chuyển đến database chỉ định hoặc tạo database mới*
- > db.dropDatabase() *//Xóa database hiện hành*
- > db.createCollection("collection name") *//Tạo mới một collection*
- > db.collection_name.drop() *//Xóa Collection*

Thao tác cơ bản trên MongoDB Compass

- Kết nối MongoDB từ MongoDB Compass
- Xem danh sách database, collection, document
- Tạo/Xóa database
- Tạo/Xóa collection
- Sử dụng cửa sổ lệnh trên MongoDB Compass

Import và Export Collection

❖ Export Collection

- Mở cửa sổ *cmd.exe* tại vị trí thư mục bin có chứa các tập tin sau:

Ví dụ: C:\Program Files\MongoDB\Server\4.2\bin

bsondump.exe	24/03/2020 04:54	Aj
InstallCompass.ps1	24/03/2020 05:22	Ps
mongo.exe	24/03/2020 05:10	Aj
mongod.cfg	02/12/2020 15:28	Cl
mongod.exe	24/03/2020 05:26	Aj
mongod.pdb	24/03/2020 05:26	Pr
mongodump.exe	24/03/2020 04:54	Aj
mongoexport.exe	24/03/2020 04:54	Aj
mongofiles.exe	24/03/2020 04:54	Aj
mongoimport.exe	24/03/2020 04:54	Aj
mongorestore.exe	24/03/2020 04:54	Aj
mongos.exe	24/03/2020 05:24	Aj
mongos.pdb	24/03/2020 05:24	Pr
mongostat.exe	24/03/2020 04:54	Aj
mongotop.exe	24/03/2020 04:54	Aj
qlsv.json	19/01/2021 15:35	JS

Import và Export Collection (tt)

- Sử dụng lệnh:

```
mongoexport --db database_name --collection  
collection_name --out "file_name.json"
```

Có thể thay thế "--db" bằng "-d"; "--collection" bằng "-c"; "--out" bằng "-o"

- Ví dụ:

```
> mongoexport -d qlsv -c sv -o  
"D:\MONGO\EXPORT\qlsv\sv.json"
```

Import và Export Collection (tt)

❖ Import Collection

- Mở cửa sổ lệnh cmd tương tự như khi Export
- Sử dụng lệnh:

```
> mongoimport --db database_name --  
collection collection_name --file  
"file_name.json"
```

```
> Có thể thay thế "--db" bằng "-d"; "--  
collection" bằng "-c". Lưu ý "--file"  
không thể thay thế.
```


Sao lưu và phục hồi CSDL

❖ Sao lưu database

- Mở cửa sổ lệnh cmd tương tự như khi Export
- Sử dụng cấu trúc lệnh:

```
> mongodump -d <database name> -o  
    <folder_database_backup>
```

```
> Ví dụ: mongodump -d qlsv -o  
    "D:\MONGO\EXPORT\fulldb"
```

Sao lưu và phục hồi CSDL (tt)

❖ Phục hồi database

- Mở cửa sổ lệnh cmd tương tự như khi sao lưu.
- Sử dụng cấu trúc lệnh:

```
> mongorestore -d <database_name>  
    <folder_database_backup_before>
```

```
> Ví dụ: mongorestore -d qlsv  
    "D:\MONGO\EXPORT\fulldb\qlsv"
```

Thao tác trên Document

- ❖ Thêm một Document mới
- ❖ Xóa Document
- ❖ Cập nhật Document

Thêm mới một Document

- Cú pháp lệnh:

`db.collection_name.insert(document)`

- Ví dụ: Thêm một document vào collection sv

```
db.sv.insert({Masv:"sv001",Hoten:"Đỗ Thị Lan"})
```

```
db.sv.insert([{"Masv":"sv002",Hoten:"Trần Minh"}, {"Masv":"sv003",Hoten:"Phạm Tuấn"}])
```

```
db.sv.insert(
```

```
{  
    Masv:"sv004",  
    Hoten:"Trần Minh Nghĩa",  
    Ngaysinh: new Date("2001-04-14"),  
    Lop:{Malop:"l01",Tenlop:"08DHTH"},  
    Monhoc:  
    [  
        {Mamh:"m001",Tenmh:"Cơ sở dữ liệu",Sotc:3,Diem:7.5},  
        {Mamh:"m002",Tenmh:"Toán cao cấp",Sotc:2,Diem:9},  
        {Mamh:"m003",Tenmh:"Công nghệ phần mềm",Sotc:3,Diem:8.5},  
    ]  
})
```

Thêm mới một Document (tt)

❖ Thao tác trên MongoDB Compass

- Insert Document sử dụng các chế độ hiển thị khác nhau

Insert to Collection qlsinhvien.sinhvien

VIEW

{}

≡

```
1 ▾ /**
2   * Paste one or more documents here
3   */
4
```

Xóa Document

❖ Cú pháp lệnh:

- `db.collection.remove({criteria})` //Xóa Document thỏa điều kiện

Ví dụ:

```
> db.sv.remove({masv:"sv001"})  
> db.sv.deleteOne({Masv:"sv001"})  
> db.sv.deleteMany({Masv:"sv001"})
```

- `db.collection.remove({})` //Xóa tất cả các Document trong collection

```
db.collection.deleteMany({})
```

❖ Xóa Document bằng thao tác

Sửa Document

❖ Cập nhật sửa giá trị của một field

- Sử dụng toán tử **\$set**
- `db.collection.update({criteria},{ $set:{key:"new value"}})`

```
> db.sv.update ( {Masv: "sv002"} , { $set: {Tuoi: 24} } )
```

```
Masv: 'sv002',  
Hoten: 'Đỗ Thị Bình',  
Tuoi: 24,
```

- Sửa Tên thành “Đỗ Thị Minh” và quốc tịch thành “Mỹ” của sinh viên có mã là sv002?

Sửa Document (tt)

❖ Cập nhật sửa giá trị trong một mảng

- `db.collection.update({criteria},{ $set: {"array.position": "new value"}}`)

> `db.sv.update({Masv: "sv003"}, { $set: { "Ngoaingu.1": "Tiếng Nhật" }})` //sửa giá trị của mảng Ngoaingu tại vị trí 1 có giá trị mới là "Tiếng Nhật"

```
Phai: 'Nam',
Quoctich: 'Việt Nam',
Ngoaingu: [ 'Tiếng Anh', 'Tiếng Nhật' ],
Lop: { Malop: '103', Tenlop: '09DHTH' },
Monhoc:
```


Sửa Document (tt)

❖ Cập nhật sửa giá trị của field trong document nhúng

- Cấu trúc: `field:{subfield:subvalue,...}`
- `db.collection.update({criteria},{ $set:{"field.subfield": "new value"}}`)

```
> db.sv.update ({Masv: "sv003"}, { $set: { "Lop.Malop":  
"103", "Lop.Tenlop": "09DHTH" } }) //Sửa giá trị mã lớp  
và tên lớp của SV003 thành 103 và 09DHTH
```

```
Phai: 'Nam',  
Quoctich: 'Việt Nam',  
Ngoaingu: [ 'Tiếng Anh', 'Tiếng Nhật' ],  
Lop: { Malop: '103', Tenlop: '09DHTH' },  
Monhoc:
```

Sửa Document (tt)

- ❖ Cập nhật giá trị trong mảng document con
 - `field:[{subfield1:value1},{subfield2:value2},...]`
 - `db.collection.update({criteria},{ $set:{ "array.position.subfield": "new value" } })`

Ví dụ:

```
> db.sv.update ( {Masv: "sv003" }, { $set: { "Monhoc  
.1.Diem": 8.5 } })
```

```
Ngoinhu: [ 'Tieng Anh', 'Tieng Nhac' ],  
Lop: { Malop: '103', Tenlop: '09DHTH' },  
Monhoc:  
  [ { Mamh: 'm005', Tenmh: 'Lich sử đảng', Sotc: 3, Diem: 9 },  
    { Mamh: 'm006', Tenmh: 'Toán rời rạc', Sotc: 3, Diem: 8.5 },  
    { Mamh: 'm001', Tenmh: 'Cơ sở dữ liệu', Sotc: 3, Diem: 6 } ],  
Malop: '103'
```

Sửa Document (tt)

❖ Cập nhật sửa nhiều giá trị

- Sử dụng thuộc tính **{multi:true}**

```
> db.sv.update({"Lop.MaLop":"101"},{$set:{"Lop.Tenlop":"08DHTH2"}},{multi:true}) //Sửa tên của những  
lớp có mã 101 thành 08DHTH2
```

Sửa Document (tt)

❖ Cập nhật thêm giá trị vào mảng:

- `db.collection.update({criteria},{<toán_tử_thêm>:{key_array:"new value"}})`

- Toán tử **\$push**: thêm không kiểm tra trùng

```
> db.sv.update({Masv:"sv001"},{$push:{Ngoaingu:"Tiếng Nga"}})
```

```
> db.sv.update({Masv:"sv003"},{$push:{Monhoc:{Mamh:"m001",Tenmh:"Cơ sở dữ liệu",Sotc:3,Diem:6}}})
```

- Toán tử **\$addToSet**: thêm có kiểm tra trùng

```
> db.sv.update({Masv:"sv001"},{$addToSet:{Ngoaingu:"Tiếng Nga"}})
```

Sửa Document (tt)

❖ Cập nhật loại bỏ giá trị ra khỏi mảng:

- Sử dụng toán tử **\$pull**
- `db.collection.update({criteria},{ $pull:{key_array: "value"}}`)

Ví dụ:

```
> db.sv.update( {Masv:"sv001"}, { $pull:{Ngoaingu: "Tiếng Nga"}} )
```

```
> db.sv.update( {Masv:"sv003"}, { $pull:{Monhoc: {Mamh:"m001", Tenmh:"Cơ sở dữ liệu", Sotc:3, Diem:6}}}} )
```

Thay thế document

❖ Cú pháp lệnh:

- `db.collection_name.save({"_id":ObjectId(...), key1:value1, key2: value2})`

- Ví dụ:

```
> db.sv.save({_id: ObjectId("60165a20254da01228f2f015"),
masv: "sv018",
hoten: "Trần Văn Minh",
tuoi: 20,
malop: "L44",
ngaysinh: "2020-03-25"})
```

Bài tập

1) Sử dụng cơ sở dữ liệu lưu trữ thông tin sinh viên, lớp, môn học được mô hình hóa bởi collection `sinhvien`. Thực hiện các yêu cầu sau:

- a/ Viết lệnh thêm vào collection `sinhvien` trong 2 trường hợp: thêm một và nhiều document.
- b/ Viết lệnh xóa document với điều kiện mã sinh viên là “sv005”
- c/ Viết lệnh xóa những sinh viên học lớp có mã lớp là l03
- d/ Viết lệnh sửa Họ tên của sinh viên có mã sv001 thành Đỗ Nhật Lâm
- e/ Sửa tuổi thành 25, Phái thành Nữ, Họ tên thành Trần Thị Lan cho sinh viên có mã là sv003
- f/ Sửa ngoại ngữ thứ 2 của sinh viên có mã sv003 thành Tiếng Hàn
- g/ Sửa điểm của sinh viên có mã sv003 học môn thứ 1 thành 9
- h/ Viết lệnh thay thế một document với `_id` được chỉ định.

Bài tập (tt)

2) Sử dụng cơ sở dữ liệu quản lý bán sản phẩm với collection *hoadon* có cấu trúc như sau:

Collection: Hóa đơn

```
{  
    Mã hóa đơn: Chuỗi lưu mã số hóa đơn,  
    Ngày lập: Ngày lập hóa đơn,  
    Khách hàng: {Mã khách hàng, Tên khách hàng, Địa chỉ}  
    Sản phẩm bán: [  
        {Mã sp1, Tên sp1, số lượng, giá bán, thành tiền},  
        {Mã sp2, Tên sp2, số lượng, giá bán, thành tiền}  
        ...  
    ]  
}
```

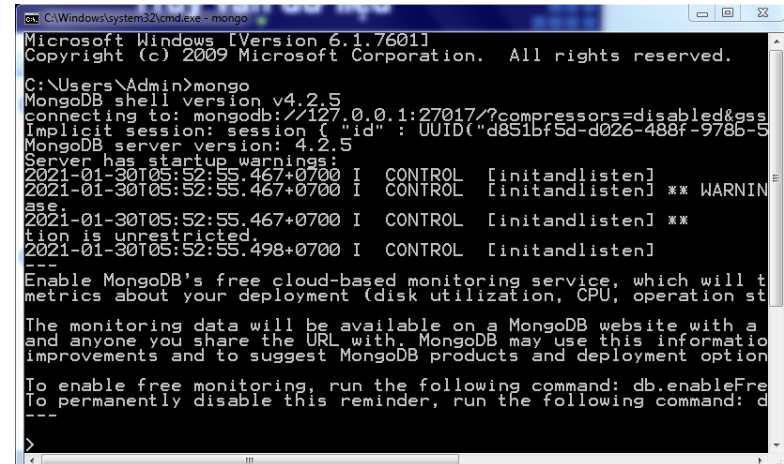

Bài tập (tt)

- a/ Viết lệnh thêm vào 2 document vào collection hoaddon.
- b/ Viết lệnh xóa những hóa đơn có ngày lập “2020-03-25”
- c/ Viết lệnh xóa những hóa đơn của khách hàng có mã là “kh001”
- d/ Viết lệnh sửa ngày lập của hóa đơn có mã h001 thành “2021-02-25”
- e/ Sửa thông tin khách hàng có mã là kh001 với Họ tên khách hàng thành Trần Thị Lan, địa chỉ thành TPHCM
- f/ Sửa tên khách hàng có mã kh003 thành Đỗ Thanh Bình
- g/ Thêm một sản phẩm vào hóa đơn có mã hd003

Truy vấn dữ liệu

❖ Sử dụng cửa sổ lệnh:

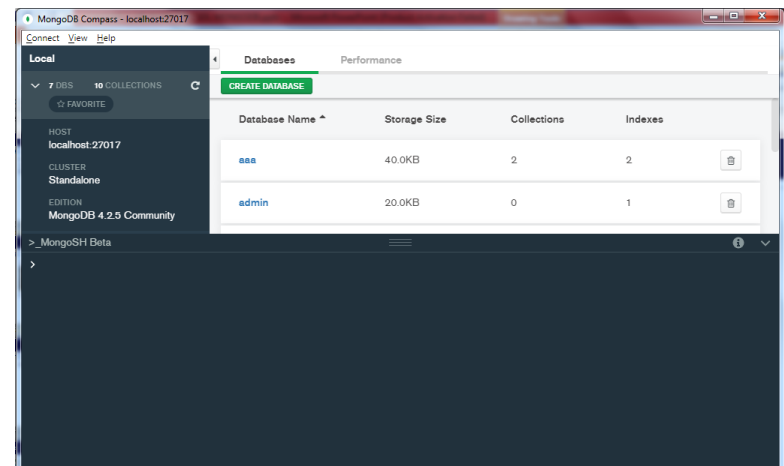
- Cửa sổ lệnh cmd hoặc



```
C:\Windows\system32\cmd.exe - mongo
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Admin>mongo
MongoDB shell version v4.2.5
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gss
implicit session: session { "id" : UUID("d851bf5d-d026-488f-9786-5
MongoDB server version: 4.2.5
Server has startup warnings:
2021-01-30T05:52:55.467+0700 I CONTROL [initandlisten] ** WARNIN
ase.
2021-01-30T05:52:55.467+0700 I CONTROL [initandlisten] **
tion is unrestricted.
2021-01-30T05:52:55.498+0700 I CONTROL [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will t
metrics about your deployment (disk utilization, CPU, operation st
The monitoring data will be available on a MongoDB website with a
and anyone you share the URL with. MongoDB may use this informati
improvements and to suggest MongoDB products and deployment option
To enable free monitoring, run the following command: db.enableFre
To permanently disable this reminder, run the following command: d
>
```

- Cửa sổ lệnh trên MongoDB Compass



Truy vấn không điều kiện

❖ Phương thức find() không tham số:

- > `db.collection.find()` *//Tìm tất cả các document*
- > `db.collection.findOne()` *//Tìm 1 document*
- > `db.collection.find().pretty()` *//Hiển thị kết quả theo định dạng JSON*

Truy vấn có điều kiện

❖ Điều kiện bằng (Equality condition)

- Sử dụng biểu thức {<field1>:<value1>,...}
- Ví dụ: Tìm các sinh viên có mã số là sv001

```
> db.sv.find( {Masv: "sv001"} )
```

- Tìm những sinh viên phái *Nam* có quốc tịch Việt Nam:

```
> db.sv.find( {Phai: "Nam", Quoctich: "Việt  
Nam"} )
```

Lọc các trường trong truy vấn

❖ Cú pháp:

- `db.collection.find({criteria},{field:value,...})`

Trong đó: `value=1` là hiển thị, `value=0` là ẩn

Trường `_id` sẽ tự động hiển thị. Nếu muốn ẩn thì ghi `_id:0`

Ví dụ: Tìm những sinh viên phái Nữ, hiển thị Mã sv và Họ tên.

```
> db.sv.find({Phai:"Nữ"},{Masv:1,Hoten:1,_id:0})
```

Toán tử \$in

❖ Cú pháp:

- `db.collection.find({field:{$in:["value1","value2",...]}})`
- Ví dụ: Tìm những sinh viên có quốc tịch thuộc tập các quốc tịch gồm: Việt Nam, Singapore

```
> db.sv.find({quoctich:{$in:["Việt  
Nam","Singapore"]}})
```

Tìm kiếm tương đối

{field:/^s/}: field có giá trị bắt đầu bằng chuỗi s

{field:/s\$/}: field có giá trị kết thúc bằng chuỗi s

{field:/s/}: field có giá trị chứa chuỗi s. Lưu ý: Chuỗi s không đặt trong cặp dấu nháy ""

- Ví dụ:

- Tìm những sinh viên có họ *Trần*

```
> db.sv.find({Hoten: /^Trần/})
```

- Tìm những sinh viên có tên *Bình*

```
> db.sv.find({Hoten: /Bình$/})
```

- Tìm những sinh viên có tên lót là *Thị*

```
> db.sv.find({Hoten: /Thị/})
```

Các toán tử so sánh

❖ Cú pháp:

- `{field:{<operator>:<value>}}`
- Trong đó các toán tử (operator):
 - > `$eq`: equal to (`=`)
 - > `$lt`: less than (`<`),
 - > `$lte`: less than or equal to (`≤`),
 - > `$gt`: greater than (`>`),
 - > `$gte`: greater than or equal to (`≥`)
 - > `$ne`: not equal (`!=`)
- Ví dụ: Tìm những sinh viên có tuổi < 23
 - > `db.sv.find({Tuoi:{$lt:23}})`

Điều kiện kết hợp

❖ Toán tử \$and

- Cú pháp: {\$and:[{criteria1}, {criteria2}]}
- Ví dụ1: Tìm những sinh viên quốc tịch Việt Nam có tuổi >22

```
> db.sv.find({$and:[{Quoctich:"ViệtNam"}, {Tuoi:{$gt:22}}]})
```

```
> db.sv.find({Quoctich:"ViệtNam",Tuoi:{$gt:22}})
```

> Ví dụ 2: Tìm những sinh viên có tuổi từ 18 đến 23

```
> db.sv.find({$and:[{Tuoi:{$gte:18,$lte:22}}]})
```

Điều kiện kết hợp (tt)

❖ Toán tử \$or

- Cú pháp: {\$or:[{criteria1}, {criteria2}]}
- Ví dụ1: Tìm những sinh viên có tuổi <22 hoặc >25

```
> db.sv.find({$or:[{Tuoi:{$lt:22}}, {Tuoi:{$gt:25}}]})
```

Thao tác trên mảng và document con

- ❖ Mảng
- ❖ Document con
- ❖ Mảng các document con

Điều kiện từ mảng các giá trị

- ❖ Document: {field:[value1,value2,...]}
- ❖ Cú pháp: {field:value}
- ❖ Ví dụ: Tìm những sinh viên có ngoại ngữ là “Tiếng Nga”

```
> db.sv.find({Ngoaingu: "Tiếng Nga"})
```

Điều kiện từ document con

- ❖ Document: {field:{subfield:value}}
- ❖ Cú pháp: {"field.subfield":value1}
- ❖ Ví dụ: Tìm những sinh viên học lớp có mã lớp là 101

```
> db.sv.find( {"Lop.Malop": "101" } )
```

Điều kiện từ mảng các document con

❖ Document:

- {field:[{field1:value1},...]}

❖ Cú pháp:

- {field:{\$elemMatch:{field1:value1}}}

❖ Ví dụ: Tìm những sinh viên học môn học có tên là *Cơ sở dữ liệu*

```
> db.sv.find({Monhoc:{$elemMatch:{Tenmh:"  
Cơ sở dữ liệu"}}})
```

Lệnh limit và skip

- Lệnh *limit(n)*: trả về n document đầu tiên trong kết quả truy vấn.
- Lệnh *skip(m)*: bỏ qua m document đầu tiên trong kết quả truy vấn
- Ví dụ:
 - > `db.sv.find().limit(2)` //trả về 2 document đầu tiên
 - > `db.sv.find().skip(2)` //bỏ qua 2 document đầu tiên
 - > `db.sv.find().limit(2).skip(2)` //bỏ qua 2 document đầu, lấy 2 document tiếp theo.

Sắp xếp dữ liệu

❖ Cú pháp:

- `db.collection.find().sort({field1:value1,field2:value2,...})`
- Value=1: sắp tăng, value=-1: sắp giảm
- Ví dụ:
 - > `db.sv.find().sort({Tuoi:1})`
 - > `db.sv.find().sort({"Lop.Malop":1,Tuoi:-1})`

Bài tập

Câu 1) Sử dụng cơ sở dữ liệu lưu trữ thông tin sinh viên, lớp, môn học được mô hình hóa bởi collection `sinhvien`. Viết các câu truy vấn sau:

- 1/ Cho biết những sinh viên phái nữ có ngoại ngữ là Tiếng Anh
- 2/ Liệt kê những sinh viên phái nam trên 22 tuổi
- 3/ Liệt kê những sinh viên có họ tên bắt đầu bằng chữ T
- 4/ Liệt kê những sinh viên có tên là Lan, chỉ hiển thị Mã sinh viên, Họ tên và Phái.
- 5/ Tìm những sinh viên học các ngoại ngữ thuộc tập gồm: Tiếng Pháp, Tiếng Nhật
- 6/ Liệt kê các sinh viên của 2 lớp có tên là 08DHTH và 09DHTH, hiển thị mã sinh viên và họ tên.
- 7/ Liệt kê những sinh viên học lớp 09DHTH có tuổi < 23 hoặc > 25
- 8/ Tìm những sinh viên lớp 09DHTH có ngoại ngữ Tiếng Pháp hoặc Tiếng Nhật
- 9/ Những sinh viên học môn cơ sở dữ liệu có điểm > 7.5

Bài tập (tt)

Câu 2) Sử dụng cơ sở dữ liệu quản lý bán sản phẩm viết các câu truy vấn sau:

- 1/ Cho biết thông tin những hóa đơn được lập ngày 12/03/2021
- 2/ Liệt kê Mã hóa đơn, ngày lập của khách hàng có mã số là kh001
- 3/ Liệt kê những hóa đơn có bán sản phẩm có mã là sp012 với số lượng >30
- 4/ Những hóa đơn nào được lập trong thời gian từ ngày 01/03/2020 đến 30/05/2021
- 5/ Liệt kê thông tin những hóa đơn không lập trong ngày 5/7/2020. Thông tin liệt kê gồm: Mã hóa đơn, ngày lập

Truy vấn mở rộng với Aggregation

- ❖ Aggregation framework hỗ trợ các truy vấn phức tạp.
- ❖ Có thể thực hiện các truy vấn với việc nhóm dữ liệu như trong SQL như: Group by, Count, sum,...

Truy vấn mở rộng với Aggregation

❖ Cú pháp:

- `db.collection.aggregate(options)`

Collection

```
db.orders.aggregate( [  
  $match stage → { $match: { status: "A" } },  
  $group stage → { $group: { _id: "$cust_id", total: { $sum: "$amount" } } }  
)
```

```
{  
  cust_id: "A123",  
  amount: 500,  
  status: "A"  
}  
{  
  cust_id: "A123",  
  amount: 250,  
  status: "A"  
}  
{  
  cust_id: "B212",  
  amount: 200,  
  status: "A"  
}  
{  
  cust_id: "A123",  
  amount: 300,  
  status: "D"  
}
```

orders

\$match

```
{  
  cust_id: "A123",  
  amount: 500,  
  status: "A"  
}  
{  
  cust_id: "A123",  
  amount: 250,  
  status: "A"  
}  
{  
  cust_id: "B212",  
  amount: 200,  
  status: "A"  
}
```

\$group

```
{  
  _id: "A123",  
  total: 750  
}  
{  
  _id: "B212",  
  total: 200  
}
```

\Leftrightarrow `SELECT cust_id, sum(amount)
FROM orders
WHERE status='A'
GROUP BY cust_id`

Các biểu thức sử dụng trong gom nhóm

Biểu thức	Mô tả
\$sum	Tổng giá trị được xác định từ tất cả Document trong Collection đó
\$avg	Tính trung bình của tất cả giá trị đã cho từ tất cả Document trong Collection đó
\$min	Lấy giá trị nhỏ nhất của các giá trị từ tất cả Document trong Collection đó
\$max	Lấy giá trị lớn nhất của các giá trị từ tất cả Document trong Collection đó
\$push	Chèn giá trị vào trong một mảng trong Document kết quả
\$addToSet	Chèn giá trị tới một mảng trong Document kết quả, nhưng không tạo các bản sao
\$first	Lấy Document đầu tiên từ Source Document theo nhóm
\$last	Lấy Document cuối cùng từ Source Document theo nhóm

Một số ví dụ

```
{ _id: ObjectId("602cd3b0ac809405bc9b8609"),  
  Manv: 'nv001',  
  Hoten: 'Trần Văn Bình',  
  Phai: 'Nam',  
  Luong: 4000,  
  Phongban: 'Kinh doanh',  
  Ngoaingu: [ 'Tiếng Anh', 'Tiếng Trung' ] }  
{ _id: ObjectId("602cd3fbac809405bc9b860a"),  
  Manv: 'nv002',  
  Hoten: 'Đỗ Thị Châu',  
  Phai: 'Nữ',  
  Luong: 5000,  
  Phongban: 'Kinh doanh',  
  Ngoaingu: [ 'Tiếng Anh', 'Tiếng Pháp' ] }  
{ _id: ObjectId("602cd430ac809405bc9b860b"),  
  Manv: 'nv003',  
  Hoten: 'Bùi Minh Tuấn',  
  Phai: 'Nam',  
  Luong: 5000,  
  Phongban: 'Quản trị',  
  Ngoaingu: [ 'Tiếng Anh', 'Tiếng Nga' ] }
```

■ Viết các câu truy vấn sau:

- > Liệt kê tên phòng và tổng lương của các nhân viên phái nam trong từng phòng.
- > Liệt kê tên phòng và số nhân viên trong từng phòng
- > Cho biết lương trung bình theo phái
- > Cho biết tên phòng và lương cao nhất trong từng phòng

Một số ví dụ (tt)

```
{ _id: ObjectId("60165a20254da01228f2f015"),
  Masv: 'sv001',
  Hoten: 'Trần Minh Nghĩa',
  Tuổi: 21,
  Ngaynhaphoc: 2018-04-14T00:00:00.000Z,
  Phai: 'Nam',
  Quoctich: 'Việt Nam',
  Ngoaingu: [ 'Tiếng Anh', 'Tiếng Pháp', 'Tiếng Nga' ],
  Lop: { Malop: '101', Tenlop: '08DHTH2' },
  Monhoc:
    [ { Mamh: 'm001', Tenmh: 'Cơ sở dữ liệu', Sotc: 3, Diem: 7.5 },
      { Mamh: 'm002', Tenmh: 'Toán cao cấp', Sotc: 2, Diem: 9 },
      { Mamh: 'm003', Tenmh: 'Lập trình C', Sotc: 3, Diem: 8.5 } ] }

{ _id: ObjectId("60165a20254da01228f2f016"),
  Masv: 'sv002',
  Hoten: 'Đỗ Thị Bình',
  Tuổi: 24,
  Ngaynhaphoc: 2019-07-20T00:00:00.000Z,
  Phai: 'Nữ',
  Quoctich: 'Việt Nam',
  Ngoaingu: [ 'Tiếng Anh', 'Tiếng Nga' ],
  Lop: { Malop: '101', Tenlop: '08DHTH2' },
  Monhoc:
    [ { Mamh: 'm001', Tenmh: 'Cơ sở dữ liệu', Sotc: 3, Diem: 7.8 },
      { Mamh: 'm002', Tenmh: 'Toán cao cấp', Sotc: 2, Diem: 8 },
      { Mamh: 'm004', Tenmh: 'Mạng máy tính', Sotc: 2, Diem: 6.5 } ] }
```

- Viết các câu truy vấn sau:
- 1) Cho biết mã lớp, tên lớp và số sinh viên trong từng lớp
- 2) Cho biết tên lớp và số sinh viên trong từng lớp
- 3) Cho biết Mã sinh viên, họ tên và số môn học của từng sinh viên
- 4) Cho biết Mã sinh viên, họ tên và điểm trung bình của từng sinh viên
- 5) Cho biết Mã và tên lớp có số sinh viên ≥ 2 .
- 6) Cho biết mã và họ tên những sinh viên học trên 2 môn học

Một số ví dụ (tt)

- Viết các câu truy vấn sau:

1) Cho biết mã lớp, tên lớp và số sinh viên trong từng lớp

```
> db.sv.aggregate({$group:{_id:"$Lop", Sosv:{$sum:1}}})
```

2) Cho biết tên lớp và số sinh viên trong từng lớp

```
> db.sv.aggregate({$group:{_id:"$Lop.Tenlop", Sosv:{$sum:1}}})
```

Lưu ý: Trường hợp lấy nhiều thuộc tính trong 1 document:
_id:{Malop:"\$Lop.Malop", Tenlop:"\$Lop.Tenlop"}

Một số ví dụ (tt)

3) Cho biết Mã sinh viên, họ tên và số môn học của từng sinh viên

```
> db.sv.aggregate({$project:{_id:0,Masv:1,Hoten:1,Somh:{$size:"$Monhoc"}}})
```

Lưu ý: Toán tử \$size không dùng trong \$group mà dùng trong \$project

4) Cho biết Mã sinh viên, họ tên và điểm trung bình của từng sinh viên

```
> db.sv.aggregate({$project:{_id:0,Masv:1,Hoten:1,DiemTB:{$avg:"$Monhoc.Diem"}}})
```

Một số ví dụ (tt)

5) Cho biết Mã lớp, tên lớp và số sinh viên của những lớp có số sinh viên ≥ 2

```
> db.sv.aggregate([{$group: {_id: "$Lop", Sosv: {$sum: 1}}}, {$match: {Sosv: {$gte: 2}}}] )
```

6) Cho biết mã sv, họ tên và số môn học của những sinh viên học từ 2 môn trở lên

```
> db.sv.aggregate([{$project: {_id: 0, Masv: 1, Hoten: 1, Somh: {$size: "$Monhoc"}}}, {$match: {Somh: {$gte: 3}}}] )
```

Aggregate với \$addToSet và \$push

- Cho cơ sở dữ liệu với collection products như sau

_id	ObjectId	name String	manufacturer String	category String	price Mi
1	602cf95dac809405bc9b860c	"iPad 16GB Wifi"	"Apple"	"Tablets"	499
2	602cf95dac809405bc9b860d	"iPad 32GB Wifi"	"Apple"	"Tablets"	599
3	602cf95dac809405bc9b860e	"iPad 64GB Wifi"	"Apple"	"Tablets"	699
4	602cf95dac809405bc9b860f	"Galaxy S3"	"Samsung"	"Cell Phones"	563.99
5	602cf95dac809405bc9b8610	"Galaxy Tab 10"	"Samsung"	"Tablets"	450.99
6	602cf95dac809405bc9b8611	"Vaio"	"Sony"	"Laptops"	499
7	602cf95dac809405bc9b8612	"Macbook Air 13inch"	"Apple"	"Laptops"	499
8	602cf95dac809405bc9b8613	"Nexus 7"	"Google"	"Tablets"	199
9	602cf95dac809405bc9b8614	"Kindle Paper White"	"Amazon"	"Tablets"	129
10	602cf95dac809405bc9b8615	"Kindle Fire"	"Amazon"	"Tablets"	199

- Viết truy vấn:

Lọc dữ liệu group theo manufacturer, đưa category vào mảng và tính giá trung bình của sản phẩm theo manufacturer.

Aggregate với \$addToSet

- Lọc dữ liệu group theo manufacturer, đưa category vào mảng (có loại bỏ giá trị trùng) và tính giá trung bình của sản phẩm theo manufacturer.

```
> db.products.aggregate({$group:{  
  _id:"$manufacturer",  
  categories:{$addToSet:"$category"},  
  avg_price:{$avg:"$price"}}})
```

```
{ _id: 'Apple',  
  categories: [ 'Laptops', 'Tablets' ],  
  avg_price: 574 },  
{ _id: 'Samsung',  
  categories: [ 'Cell Phones', 'Tablets' ],  
  avg_price: 507.49 },  
{ _id: 'Google', categories: [ 'Tablets' ], avg_price: 199 },  
{ _id: 'Amazon', categories: [ 'Tablets' ], avg_price: 164 },  
{ _id: 'Sony', categories: [ 'Laptops' ], avg_price: 499 } ]
```

Aggregate với \$push

- Lọc dữ liệu group theo manufacturer, đưa category vào mảng (không loại bỏ giá trị trùng), hiển thị giá cao nhất, thấp nhất của sản phẩm theo manufacturer.

```
> db.products.aggregate({$group:{  
  _id:"$manufacturer",  
  categories:{$push:"$category"},  
  max_price:{$max:"$price"},  
  min_price:{$min:"$price"}}})
```

```
[ { _id: 'Google',  
  categories: [ 'Tablets' ],  
  max_price: 199,  
  min_price: 199 },  
  { _id: 'Apple',  
  categories: [ 'Tablets', 'Tablets', 'Tablets', 'Laptops' ],  
  max_price: 699,  
  min_price: 499 },  
  { _id: 'Sony',
```

Sử dụng \$toLower, \$toUpper, \$multiply

- \$toLower: hiển thị chữ in thường
- \$toUpper: hiển thị chữ in hoa
- \$multiply: Phép nhân
- Ví dụ: Hiển thị dữ liệu với tên các manufacture phải là ký tự in hoa hay ký tự không in hoa có đơn giá nhân lên 10.

Sử dụng \$toLower, \$toUpper, \$multiply (tt)

```
db.products.aggregate([
  {$project:{
    _id:0,
    makerLower: {$toLower:"$manufacturer"},
    makerUpper: {$toUpper:"$manufacturer"},
    details: {
      category: "$category",
      price : {"$multiply":["$price",10]}
    },
    item:"$name"
  }}])
```

```
[ { makerLower: 'apple',
  makerUpper: 'APPLE',
  details: { category: 'Tablets', price: 4990 },
  item: 'iPad 16GB Wifi' },
{ makerLower: 'apple',
  makerUpper: 'APPLE',
  details: { category: 'Tablets', price: 5990 },
  item: 'iPad 32GB Wifi' },
{ makerLower: 'apple',
```

Sử dụng \$sort trong Aggregation

❖ Ví dụ: sắp xếp document theo giá tăng dần

```
> db.products.aggregate({$sort:{price:1}})
```

```
[ { _id: ObjectId("602cf95dac809405bc9b8614"),  
  name: 'Kindle Paper White',  
  category: 'Tablets',  
  manufacturer: 'Amazon',  
  price: 129 },  
  { _id: ObjectId("602cf95dac809405bc9b8613"),  
    name: 'Nexus 7',  
    category: 'Tablets',  
    manufacturer: 'Google',  
    price: 199 },  
  { _id: ObjectId("602cf95dac809405bc9b8615")
```


\$skip và \$limit trong Aggregation

❖ Ví dụ: Bỏ qua 2 document đầu, lấy 3 document tiếp theo

```
> db.products.aggregate([  
  {$skip:2}, {$limit: 3}])
```

\$first và \$last trong Aggregation

- Xét collection sales được insert dữ liệu như sau:

```
db.sales.insert({ "_id" : 1, "item" : "abc", "price" : 10, "quantity" : 2, "date" : ISODate("2014-01-01T08:00:00Z") })
db.sales.insert({ "_id" : 2, "item" : "jkl", "price" : 20, "quantity" : 1, "date" : ISODate("2014-02-03T09:00:00Z") })
db.sales.insert({ "_id" : 3, "item" : "xyz", "price" : 5, "quantity" : 5, "date" : ISODate("2014-02-03T09:05:00Z") })
db.sales.insert({ "_id" : 4, "item" : "abc", "price" : 10, "quantity" : 10, "date" : ISODate("2014-02-15T08:00:00Z") })
db.sales.insert({ "_id" : 5, "item" : "xyz", "price" : 5, "quantity" : 10, "date" : ISODate("2014-02-15T09:05:00Z") })
db.sales.insert({ "_id" : 6, "item" : "xyz", "price" : 5, "quantity" : 5, "date" : ISODate("2014-02-15T12:05:10Z") })
db.sales.insert({ "_id" : 7, "item" : "xyz", "price" : 5, "quantity" : 10, "date" : ISODate("2014-02-15T14:12:12Z") })
```

- Để lấy ra những sản phẩm được bán đầu tiên, thực hiện như sau:

```
> db.sales.aggregate({ $group:{_id: "$item",
    firstSalesDate: { $first: "$date" } } })
```

```
[ { _id: 'xyz', firstSalesDate: 2014-02-03T09:05:00.000Z },
  { _id: 'jkl', firstSalesDate: 2014-02-03T09:00:00.000Z },
  { _id: 'abc', firstSalesDate: 2014-01-01T08:00:00.000Z } ]
```

Sử dụng \$unwind

- **\$unwind:** Được sử dụng để loại bỏ các phần tử mảng ra và đặt từng phần tử vào Document.
- **Ví dụ:** Xét collection inventory có nội dung như sau:

```
{ "_id" : 1, "item" : "ABC1", "sizes" : [ "S", "M", "L" ] }
```

Để hiển thị ra các item theo từng sizes một (tách mảng sizes ra từng bản ghi riêng rẽ), sẽ dùng \$unwind

```
db.inventory.aggregate( [ { $unwind : "$sizes" } ] )
```

Kết quả sẽ ra 3 item cho từng size như sau:

```
{ "_id" : 1, "item" : "ABC1", "sizes" : "S" }  
{ "_id" : 1, "item" : "ABC1", "sizes" : "M" }  
{ "_id" : 1, "item" : "ABC1", "sizes" : "L" }
```

Sử dụng \$unwind (tt)

```
{ _id: ObjectId("60165a20254da01228f2f015"),  
  Masv: 'sv001',  
  Hoten: 'Trần Minh Nghĩa',  
  Tuổi: 21,  
  Ngaynhaphoc: 2018-04-14T00:00:00.000Z,  
  Phai: 'Nam',  
  Quoctich: 'Việt Nam',  
  Ngoaingu: [ 'Tiếng Anh', 'Tiếng Pháp', 'Tiếng Nga' ],  
  Lop: { Malop: '101', Tenlop: '08DHTH2' } }
```

Sử dụng csdl sinh viên,
cho biết có bao nhiêu
sinh viên của từng
ngoại ngữ.

```
> db.sv.aggregate([{$unwind:"$Ngoaingu"},  
  {$group:{_id:"$Ngoaingu", Sosv:{$sum:1}}}] )
```

```
[ { _id: 'Tiếng Nga', Sosv: 2 },  
  { _id: 'Tiếng Nhật', Sosv: 1 },  
  { _id: 'Tiếng Pháp', Sosv: 1 },  
  { _id: 'Tiếng Anh', Sosv: 3 } ]
```

Bài tập

❖ Sử dụng csdl quản lý sản phẩm, viết các câu truy vấn sau:

Bài tập

Cho Database gồm các Collection và Document như sau:

```
SANPHAM (_id:5ed6e9aee555191b4c2afdeb1  
THELOAI: : "MAINBOARD"  
NHASX: "IBM"  
DONGIA: 150 )
```

```
BINHLUAN(_id:5ed6e9aee555191b4c2afdec1  
TIEUDE: : ""  
MOTA: ""  
TACGIA: ""  
SOLIKE: 100 )
```

Yêu cầu: Hãy thực hiện truy vấn với Aggregation framework.

- a/ Từ Collection BINHLUAN hiển thị danh sách các bài viết bởi TACGIA là “DANY” có số LIKE lớn hơn 100.
- b/ Sắp xếp các bình luận theo số like giảm dần.
- c/ Hãy cho biết số lượng sản phẩm của thể loại MAINBOARD thuộc mỗi nhà sản xuất “IBM” là bao nhiêu.
- d/ Với mỗi sản phẩm hiển thị tên của nhà sản xuất theo ký tự in HOA, thể loại theo ký tự in thường và Đơn giá tăng lên 2 lần so với đơn giá gốc.

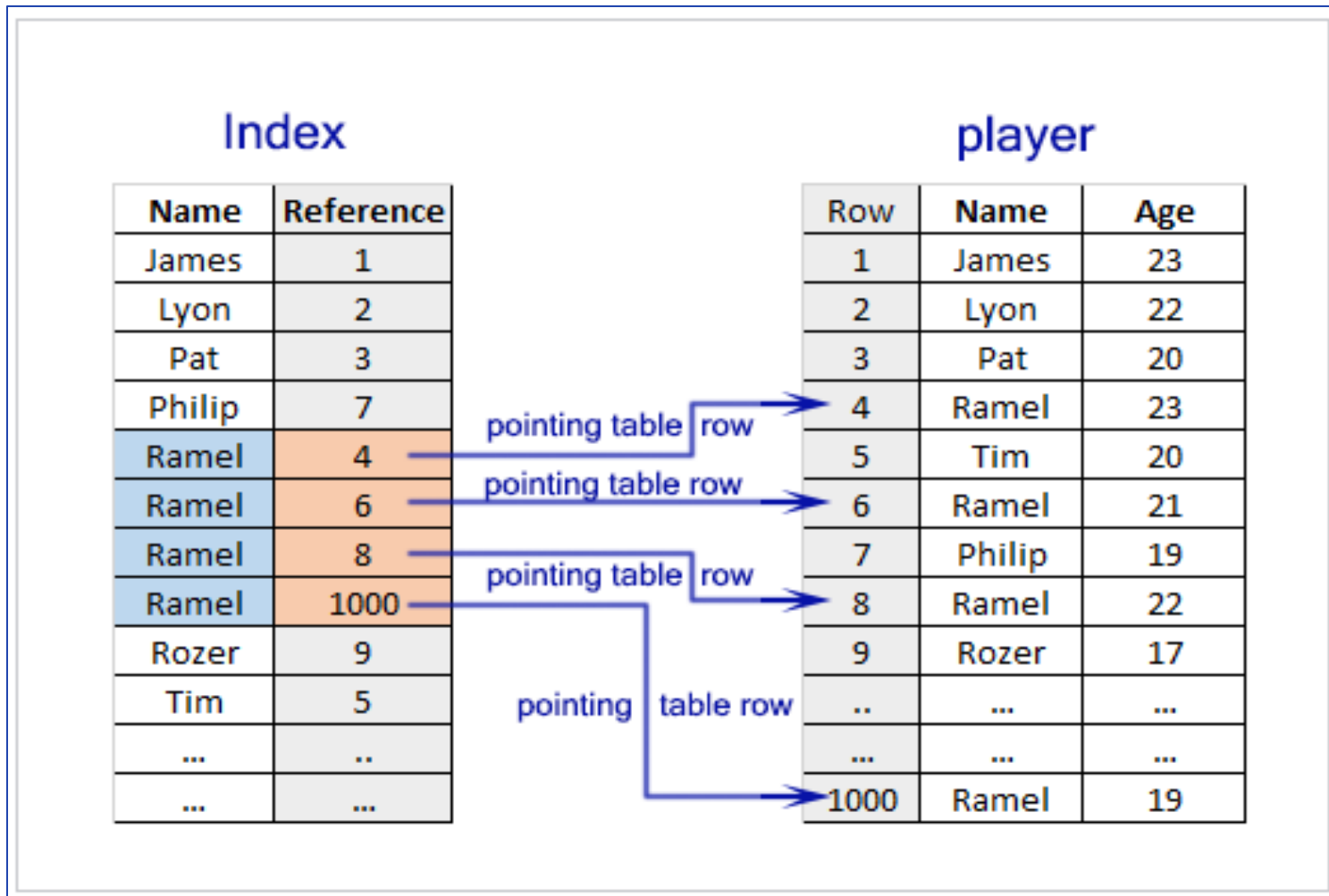
Aggregation vs SQL

STT	Trong Aggregation	Trong SQL
1	\$match	WHERE, HAVING
2	\$group	GROUP BY
3	\$project	SELECT
4	\$sort	ORDER BY
5	\$limit	LIMIT
6	\$sum	SUM, COUNT
7	\$avg, \$max, \$min	AVG, MAX, MIN
8	\$unwind	JOIN

Chỉ mục (Index)

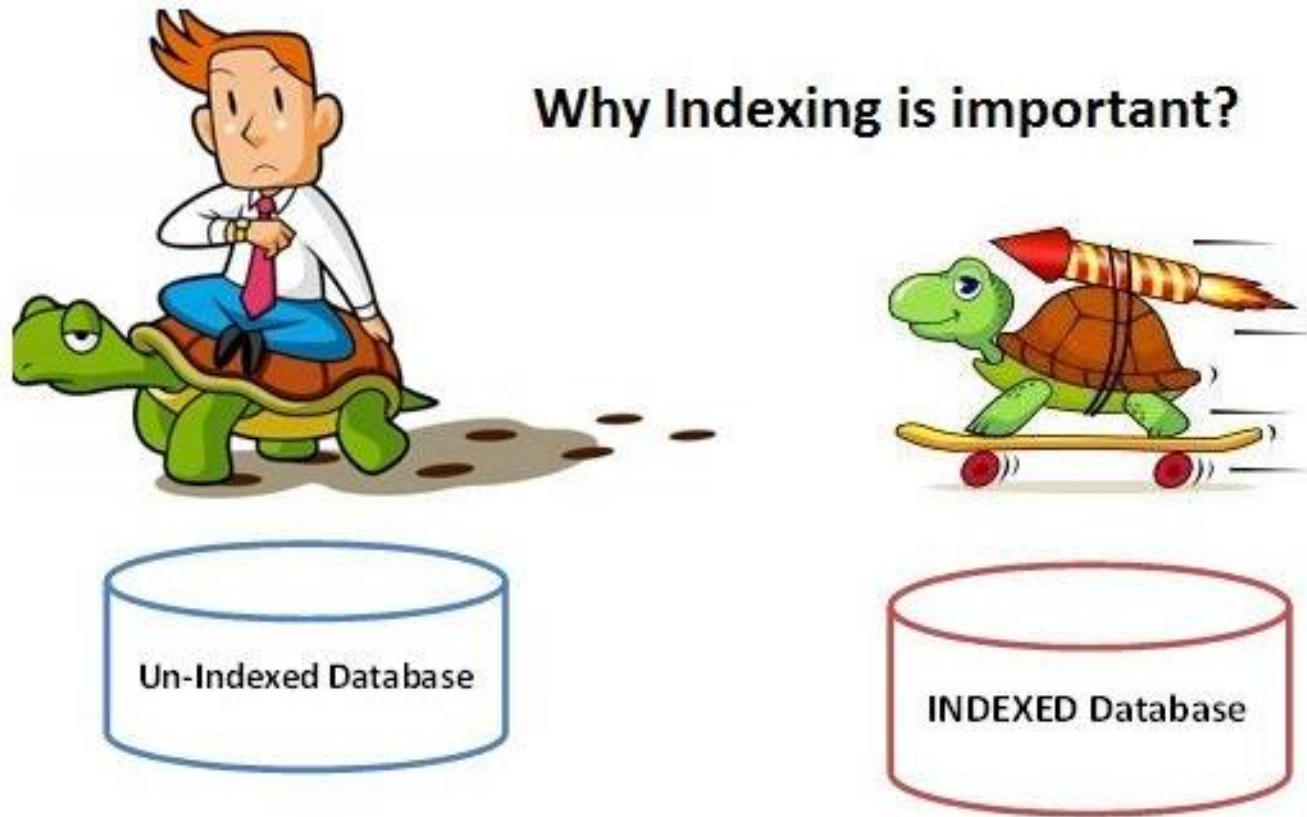
- ❖ Là một cấu trúc dữ liệu được lưu trữ trên ổ cứng ứng với một collection để tăng tốc việc truy xuất dữ liệu từ collection.
- ❖ Một chỉ mục được xây dựng từ một hoặc nhiều field trong collection.
- ❖ Tạo chỉ mục trên một field của một collection là việc tạo ra một cấu trúc dữ liệu lưu trữ field value, và trở tới document mà nó liên quan.

Ví dụ chỉ mục



Ưu điểm của chỉ mục

❖ Tăng tốc độ câu truy vấn

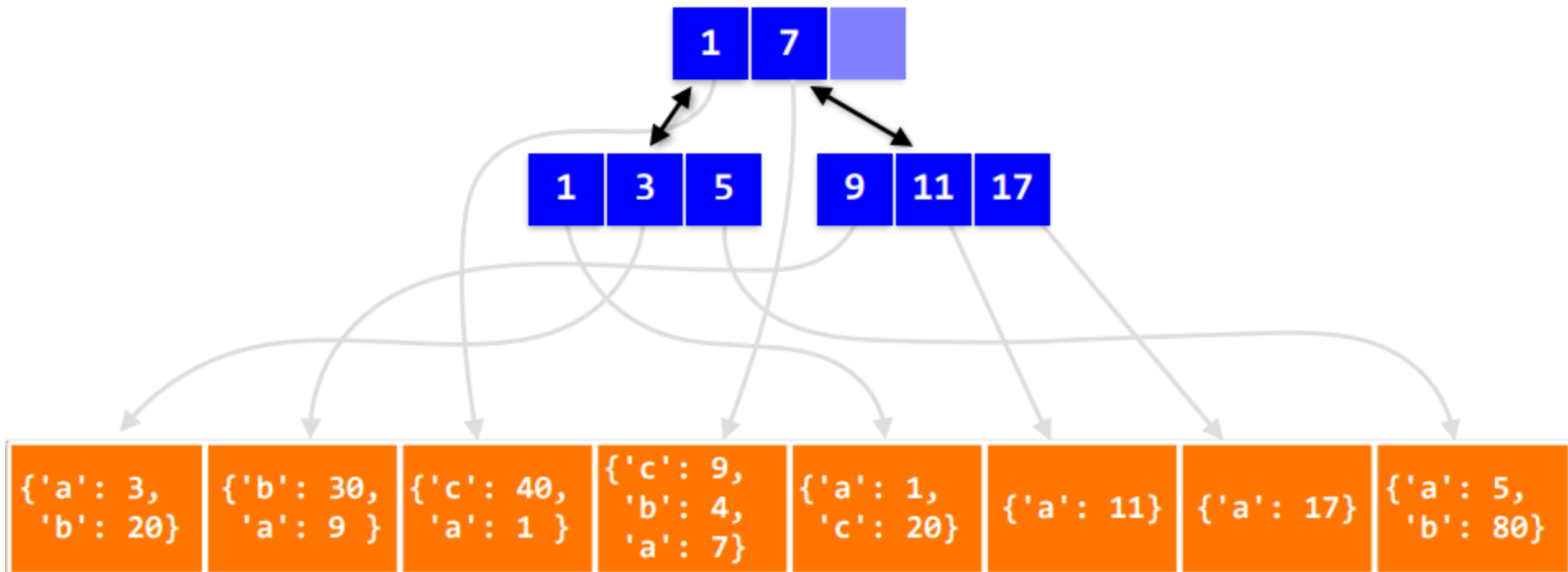


Nhược điểm của chỉ mục

- ❖ Tốn bộ nhớ trong Database để chứa các bảng Index
 - ❖ Giảm tốc độ các câu lệnh thêm, sửa, xóa vì mỗi lần dữ liệu thay đổi sẽ phải đánh lại toàn bộ Index, để đảm bảo dữ liệu truy vấn chính xác.
- => Cần cân nhắc khi tạo chỉ mục

Cấu trúc dữ liệu Index: B-Tree

Index on 'a'



Các loại chỉ mục

- ❖ Chỉ mục đơn
- ❖ Chỉ mục hỗn hợp các khóa
- ❖ Chỉ mục thừa thớt
- ❖ Chỉ mục duy nhất

Chỉ mục đơn (Single Field Index)

❖ Chỉ mục đơn

- Là loại chỉ mục mặc định
- Chỉ mục mặc định luôn luôn được tạo ra là `_id`. Chỉ mục này là đặc biệt và không thể bị xóa trong một Collection.
- Có thể tạo chỉ mục trên các khóa bên trong document nhúng.

❖ Ví dụ:

```
{  _id: ObjectId(...),  
  Hoten: "Trần Văn Nam",  
  Diachi: { Quan: "Tan Phu", TP: "TP.HCM" }
```

Chỉ mục hỗn hợp (Compound Index)

❖ Là chỉ mục đánh trên nhiều trường dữ liệu

❖ Ví dụ:

```
{  _id: ObjectId(...),  
    Hoten: "Trần Văn Nam",  
    Diachi: { Quan: "Tan Phu", TP: "TP.HCM" }  
}
```

❖ Ta có thể đánh chỉ mục hỗn hợp trên 2 trường Hoten và Quan


Chỉ mục thưa thớt (Sparse Index)

- ❖ Là chỉ mục mà chỉ bao gồm các document có trường được đánh chỉ mục.
- ❖ Bất kỳ document nào bị thiếu trường đánh chỉ mục thưa thớt đều không được lưu vào trong chỉ mục.

Ví dụ chỉ mục thưa thớt

- ❖ Giả sử đánh chỉ mục trên trường Diachi, chỉ mục lúc này không lưu trữ document số 4

Diachi String		_id ObjectId	Hoten String	Diachi String
"Khanh Hoa"		1 5e0eec51286e7598765f30da	"Thanh Thuy"	"TP.HCM"
"TP.HCM"		2 5e0eed0e286e7598765f30db	"Tran Van An"	"Vung Tau"
"TP.HCM"		3 5e0eef6c286e7598765f30dc	"Manh Thien Ly"	"TP.HCM"
"Vung Tau"		4 5e0ef7f634261a053cc602a4	"Lam Thi Hoa Mi"	No field
		5 5e181b4fc78dc61a30ce62d5	"Nguyen Thi Thanh Thuy"	"Khanh Hoa"



Chỉ mục duy nhất (Unique Index)

- ❖ Để đảm bảo rằng không có document nào được chèn có giá trị cho các khóa được lập chỉ mục khớp với các document hiện có.
- ❖ Ví dụ: Cần tạo ràng buộc tên môn học là duy nhất trên collection Môn học, ta đánh chỉ mục duy nhất trên trường tên môn học.

Các thao tác trên chỉ mục

- ❖ Xem các chỉ mục hiện có
- ❖ Tạo chỉ mục
- ❖ Xóa chỉ mục
- ❖ Tạo lại chỉ mục

Xem danh sách chỉ mục

❖ Cú pháp:

- `db.collection.getIndexes()`

❖ Ví dụ:

`> db.sv.getIndexes()`

```
[ { v: 2, key: { _id: 1 }, name: '_id_', ns: 'qlsinhvien.sv' },  
  { v: 2,  
    unique: true,  
    key: { Masv: 1 },  
    name: 'Masv_1',  
    ns: 'qlsinhvien.sv',  
    sparse: true },  
  { v: 2,  
    unique: true,  
    key: { Hoten: -1 },  
    name: 'Hoten_-1',  
    ns: 'qlsinhvien.sv',  
    sparse: true } ]
```

Tạo chỉ mục

❖ Cú pháp:

`db.collection.createIndex(keys,options)` // Tạo một chỉ mục

❖ Ví dụ:

```
> db.sv.createIndex({Masv:1})  
> db.sv.createIndex({Masv:1,"Lop.MaLop":-1})  
> db.sv.createIndex({Hoten:1},{name:"my_index"})  
> db.sv.createIndex({Masv:1},{unique:true,sparse:  
  :true})
```

❖ Chú ý:

Khi khởi tạo một chỉ mục, số đi cùng với khóa là hướng của chỉ mục, 1: tăng dần, -1: giảm dần

Tạo chỉ mục (tt)

❖ Cú pháp:

```
db.collection.createIndexes([keys1,keys2,...],options)
```

//Tạo nhiều chỉ mục cùng lúc

❖ Ví dụ:

```
> db.sv.createIndexes ([ {Masv:1} , {Hoten:-1} ] )
```

```
> db.sv.createIndexes ([ {Masv:1} , {Hoten:-  
1} ] , {unique:true, sparse:true} )
```

Xóa chỉ mục

- `db.collection.dropIndex(index)` // Xóa index được chỉ định.
index có thể là keys hoặc name
- `db.collection.dropIndexes([index1,index2,...])`
// index1, index2,.. phải là name

```
[ { v: 2, key: { _id: 1 }, name: '_id_', ns: 'qlsinhvien.sv' },  
  { v: 2,  
    unique: true,  
    key: { Masv: 1 },  
    name: 'Masv_1',  
    ns: 'qlsinhvien.sv',  
    sparse: true },  
  { v: 2,  
    unique: true,  
    key: { Hoten: -1 },  
    name: 'Hoten_-1',  
    ns: 'qlsinhvien.sv',  
    sparse: true } ]
```

- Ví dụ:

```
> db.sv.dropIndex ( {Masv:1} )
```

```
> db.sv.dropIndexes ( [ "Masv_1", "Hoten_-1" ] )
```

Tạo lại chỉ mục

❖ Cú pháp:

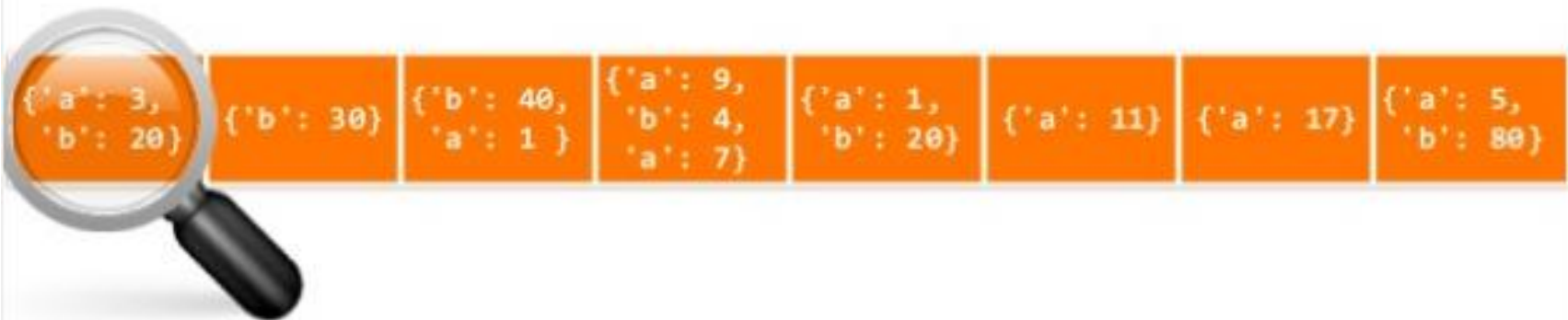
- `db.collection.reIndex()`

❖ Lưu ý:

- Lệnh này xóa tất cả các chỉ mục trên một collection và tạo lại chúng. Hoạt động này gây tốn kém cho các collection số lượng dữ liệu lớn hoặc có một số lượng lớn các chỉ mục.
- Nên cân nhắc và nên tránh sử dụng thao tác này.

Ví dụ

Full collection scan



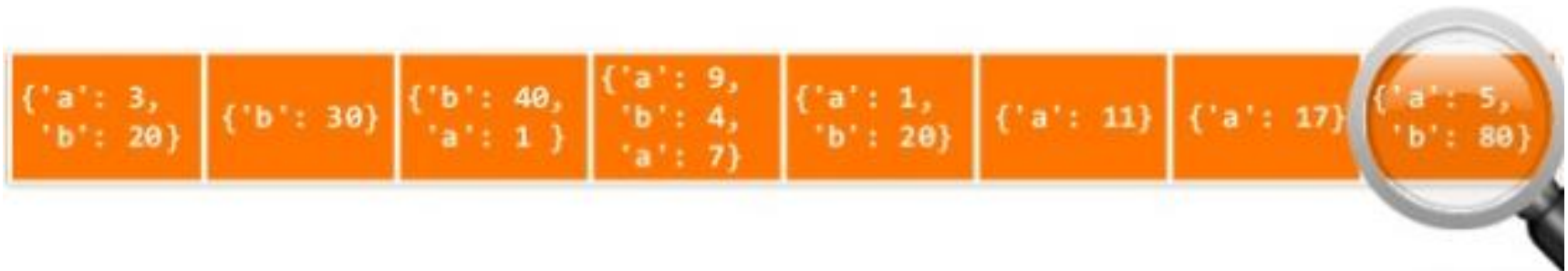
```
db.test.find({a:5})
```

documents scanned:

1

Ví dụ

Full collection scan



Time Complexity for searching in a list with n entries: **$O(n)$**

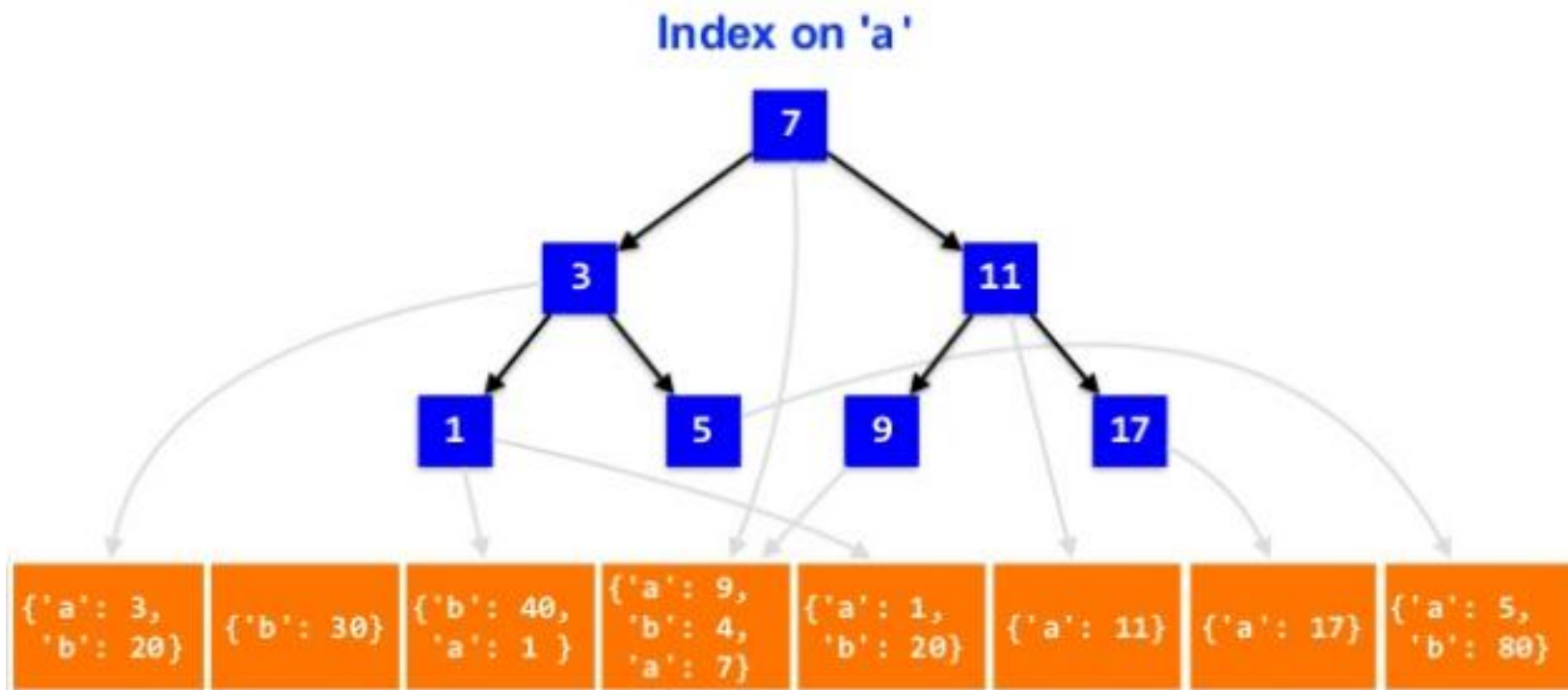
```
db.test.find({a:5})
```

documents scanned:

8

Ví dụ

Index scan



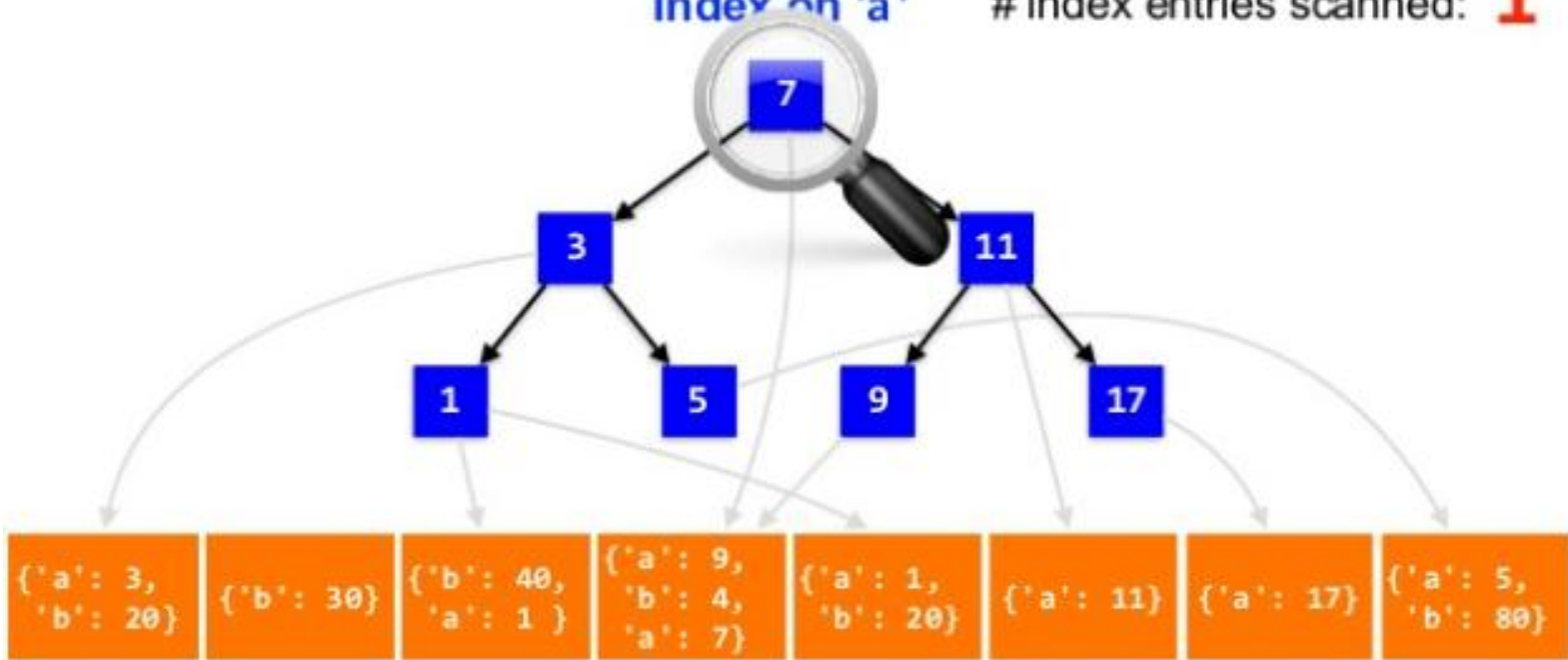
Ví dụ

Index scan

```
db.test.find({a:5})
```

Index on 'a'

index entries scanned: **1**



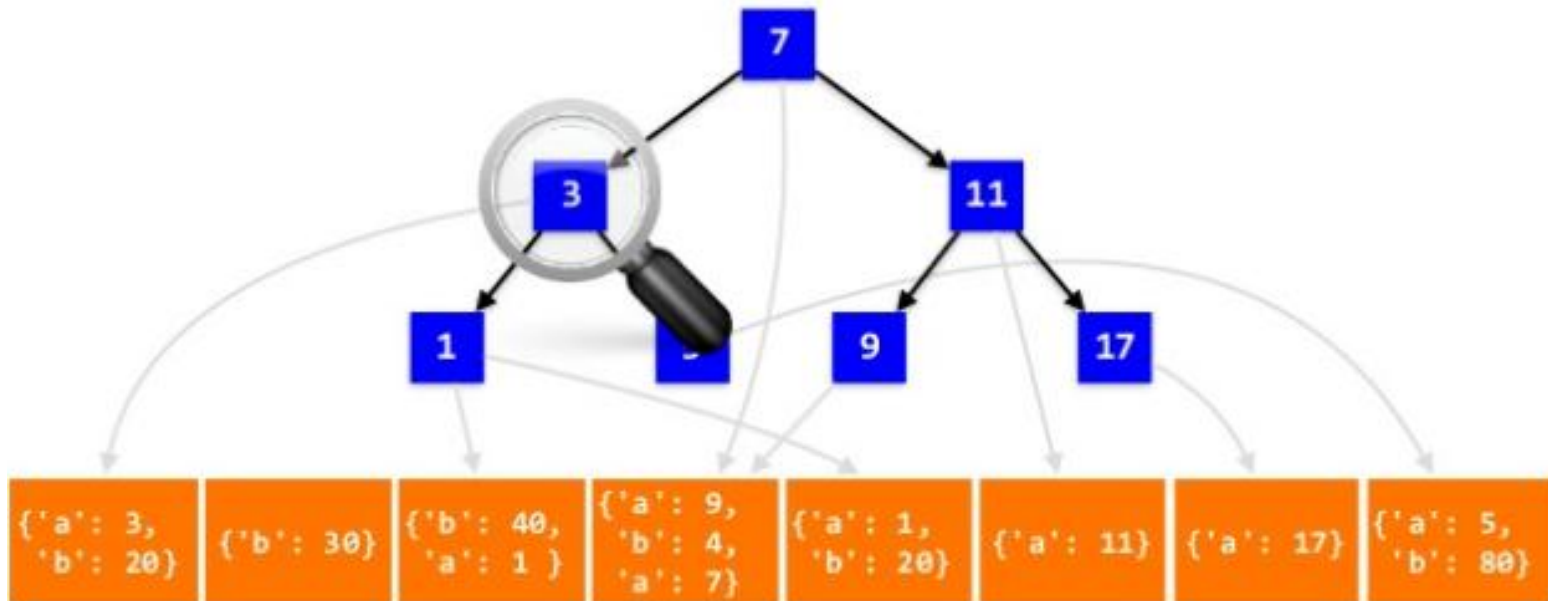
Ví dụ

Index scan

```
db.test.find({a:5})
```

Index on 'a'

index entries scanned: **2**



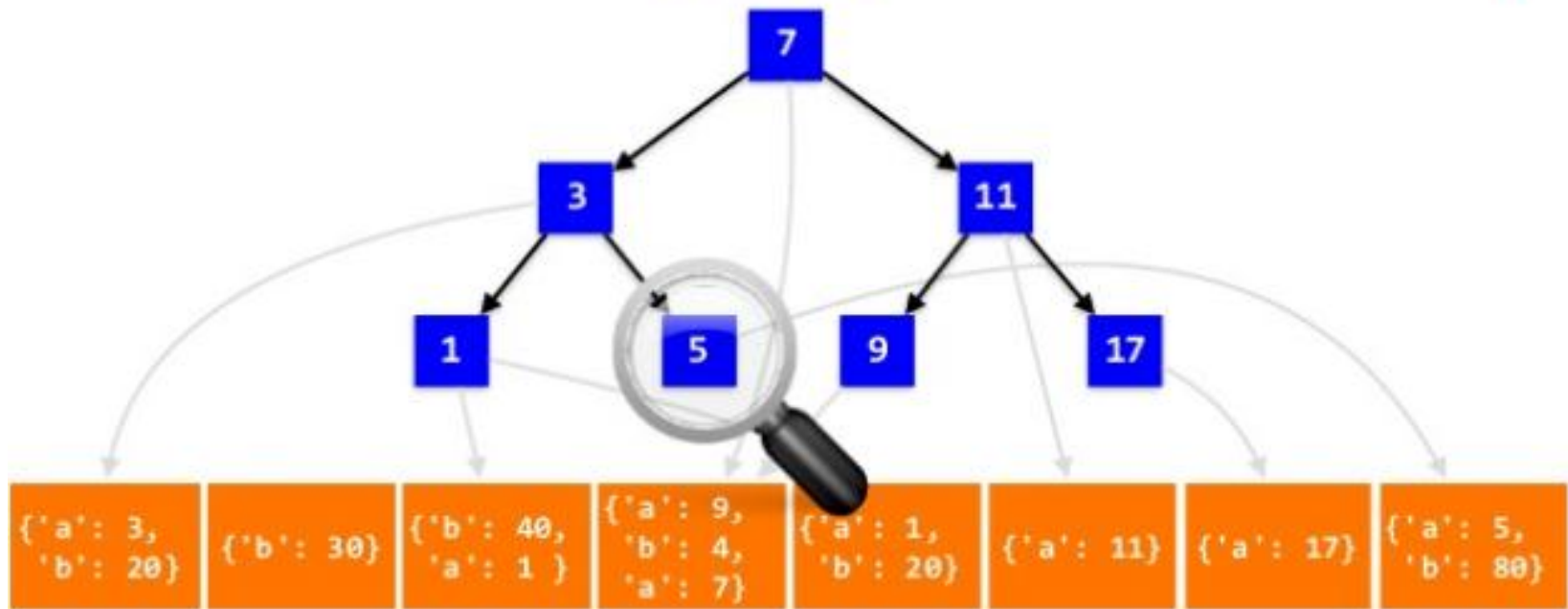
Ví dụ

Index scan

```
db.test.find({a:5})
```

Index on 'a'

index entries scanned: **3**



Ví dụ

Index scan

```
db.test.find({a:5})
```

Time Complexity for
searching in a binary
search tree with n nodes:

$$O(\log_2 n)$$

$$\log_2 10.000 = 13$$

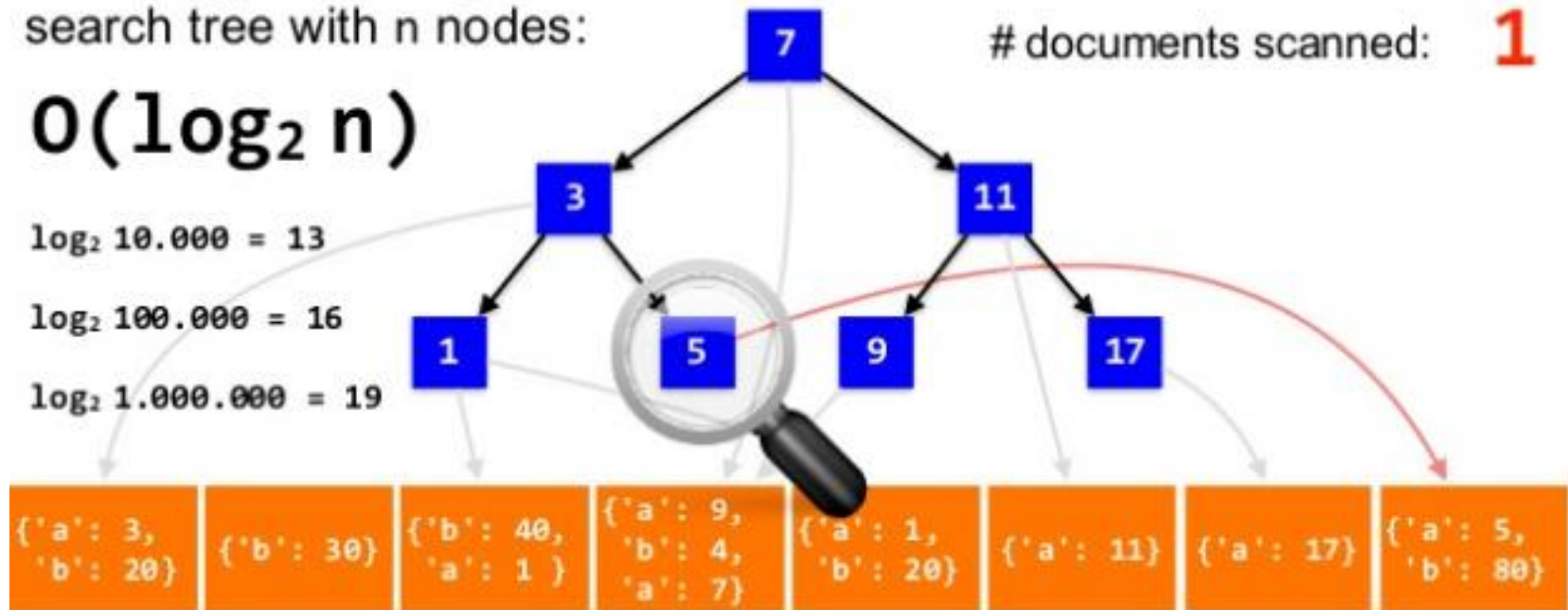
$$\log_2 100.000 = 16$$

$$\log_2 1.000.000 = 19$$

Index on 'a'

index entries scanned: **3**

documents scanned: **1**

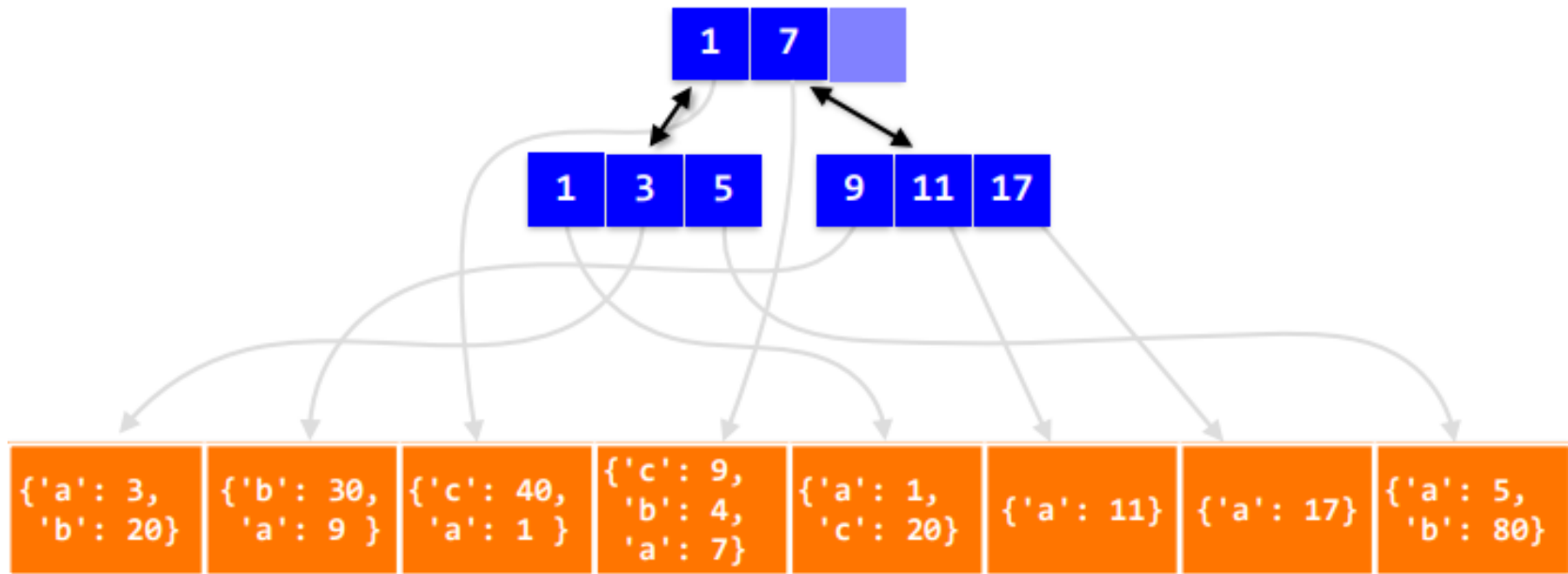


Ví dụ

MongoDB Index Data Structure: B-Tree

```
db.test.createIndex({a:1})
```

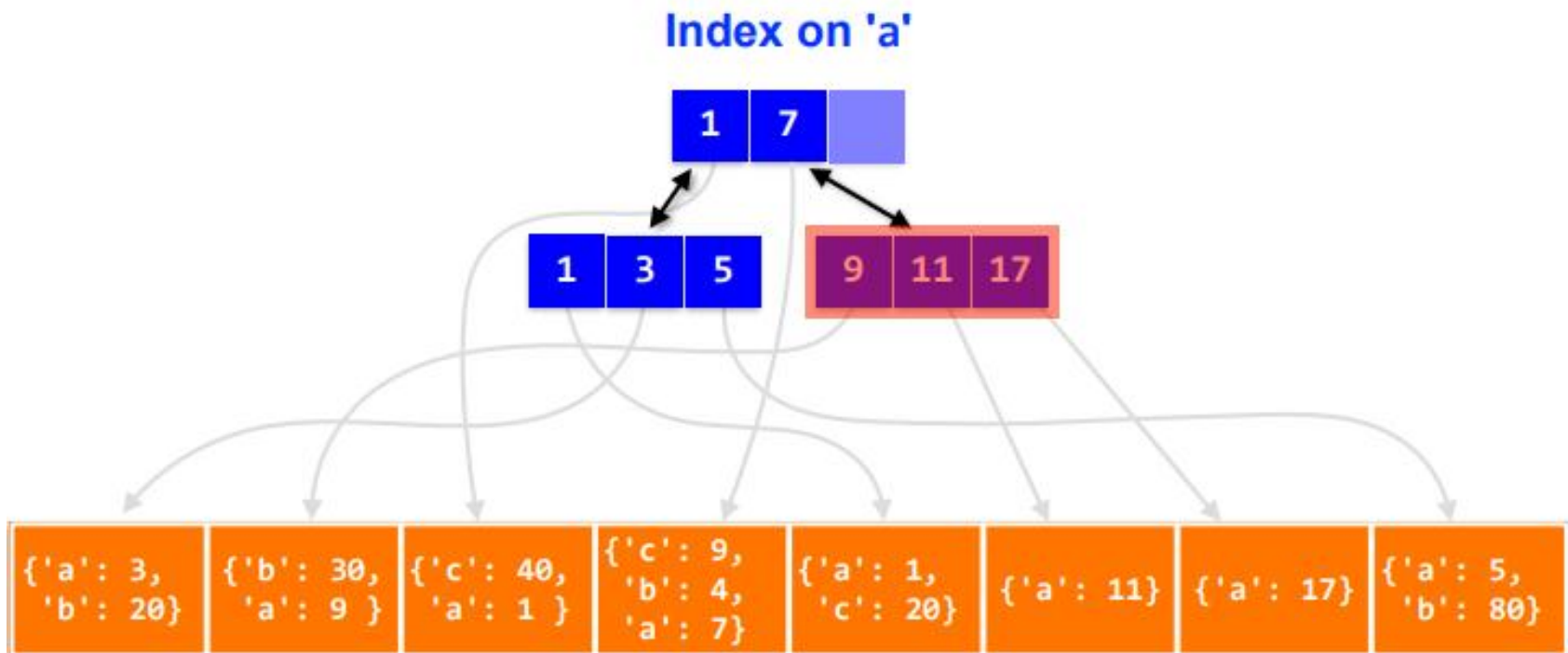
Index on 'a'



Ví dụ

MongoDB Index Data Structure: B-Tree

```
db.test.find({a:{$gte:9, $lte:17}})
```

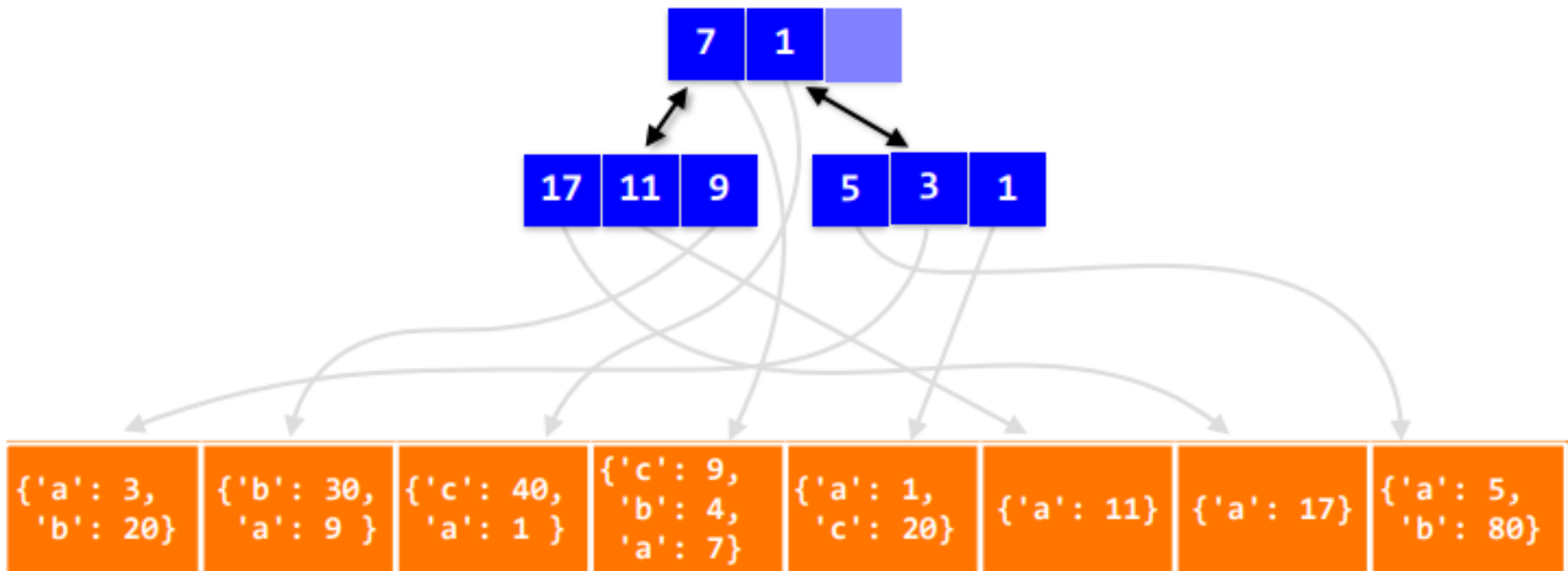


Ví dụ

MongoDB Index Data Structure: B-Tree

```
db.test.createIndex({a:-1})
```

Index on 'a'



4. Hiệu suất khi sử dụng chỉ mục

- ❑ Download file dữ liệu zips.json trên trang chủ MongoDB và import vào cơ sở dữ liệu

	_id String	city String	loc Array	pop Int32	state
1	"35004"	"ACMAR"	[] 2 elements	6055	"AL"
2	"35005"	"ADAMSVILLE"	[] 2 elements	10616	"AL"
3	"35006"	"ADGER"	[] 2 elements	3205	"AL"
4	"35007"	"KEYSTONE"	[] 2 elements	14218	"AL"
5	"35010"	"NEW SITE"	[] 2 elements	19942	"AL"
6	"35014"	"ALPINE"	[] 2 elements	3062	"AL"
7	"35016"	"ARAB"	[] 2 elements	13650	"AL"
8	"35019"	"BAILEYTON"	[] 2 elements	1781	"AL"
9	"35020"	"BESSEMER"	[] 2 elements	40549	"AL"

Hiệu suất khi sử dụng chỉ mục

❖ Thực hiện truy vấn

```
> db.zips.find({state:"NY",city:"NEW YORK",  
pop:{ $gt:100000}})
```

```
{ _id: '10021',  
  city: 'NEW YORK',  
  loc: [ -73.958805, 40.768476 ],  
  pop: 106564,  
  state: 'NY' }  
{ _id: '10025',  
  city: 'NEW YORK',  
  loc: [ -73.968312, 40.797466 ],  
  pop: 100027,  
  state: 'NY' }
```

Đánh giá hiệu suất khi chưa tạo chỉ mục

```
> db.zips.find({state:"NY",city:"NEW  
YORK",pop:{ $gt:100000}}).explain("executio  
nStats")
```

```
executionStats:  
  { executionSuccess: true,  
    nReturned: 2,  
    executionTimeMillis: 31,  
    totalKeysExamined: 0,  
    totalDocsExamined: 27555,  
    executionStages:  
      { stage: 'COLLSCAN',  
        filter:
```

Tạo index trên trường state

```
> db.zips.createIndex({state:1})  
> db.zips.find({state:"NY",city:"NEW YORK",  
  pop:{$gt:100000}}).explain("executionStats")
```

```
executionStats:  
  { executionSuccess: true,  
    nReturned: 2,  
    executionTimeMillis: 9,  
    totalKeysExamined: 1596,  
    totalDocsExamined: 1596,  
    executionStages:  
      { stage: 'FETCH',
```

Tạo index trên 2 trường state và city

```
> db.zips.createIndex({state:1,city:1})  
> db.zips.find({state:"NY",city:"NEW YORK",  
  pop:{$gt:100000}}).explain("executionStats")
```

```
executionStats:  
  { executionSuccess: true,  
    nReturned: 2,  
    executionTimeMillis: 2,  
    totalKeysExamined: 40,  
    totalDocsExamined: 40,  
    executionStages:  
      { stage: 'FETCH',
```

Tạo index trên 3 trường state, city và pop

```
> db.zips.createIndex({state:1,city:1,pop:1})  
> db.zips.find({state:"NY",city:"NEW YORK",  
  pop:{$gt:100000}}).explain("executionStats")
```

```
executionStats:  
  { executionSuccess: true,  
    nReturned: 2,  
    executionTimeMillis: 0,  
    totalKeysExamined: 2,  
    totalDocsExamined: 2,  
    executionStages:  
      { stage: 'FETCH'
```


Lưu ý khi sử dụng INDEX

- ❖ Nên sử dụng trong các cơ sở dữ liệu cần đọc nhiều, ghi ít.
- ❖ Không nên sử dụng trong các bảng nhỏ, ít bản ghi.
- ❖ Không nên sử dụng Index trong bảng mà các hoạt động UPDATE, INSERT xảy ra thường xuyên với tần suất lớn.
- ❖ Không nên sử dụng cho các cột mà chứa một số lượng lớn giá trị NULL.



Hết chương 2