

Ôn Tập Cuối Kỳ

1. Aggregation

Các lệnh căn bản:

- \$match , \$group: {_id: "", count: {\$sum: 1}}
- \$project, \$sort
- \$limit , \$skip
- \$unwind: "\$...", \$exist: true | false
- \$lookup: {
 - from: "bảng khác",
 - localField: "field muốn join ở bảng gốc",
 - foreignField: "field join của bảng khác",
 - as: "tên bảng mới "}
- \$addFields: {
 - tongDiem: { \$add: ["\$diemTB", 1] }}
- \$merge: {
 - into: "collection mà kết quả ghi vào",
 - on: "_id",
 - let: { khai báo biến },
 - pipeline: { đường dẫn để sử dụng aggregation },
 - whenMatched: "merge|replace, KeepExisting | fail"}
- merge: gộp (update các trường mới, giữ lại trường cũ).
replace: ghi đè toàn bộ document.

keepExisting: giữ nguyên document cũ, bỏ qua kết quả mới.

fail: nếu trùng thì báo lỗi

- \$out: {
 - db: "tên data base mà kết quả ghi vào",
 - coll: "tên collection mà kết quả ghi vào",

```
    granularity: "seconds" || "minutes" || "hours" // độ chi tiết dữ liệu
}
```

- Các toán tử

- \$sum, \$avg, \$max, \$min, \$push, \$addToSet, \$first, \$last

- Các toán tử biểu thức:

- \$add, \$subtract, \$multiply, \$divide

- Toán tử so sánh

- \$eq, \$gt, \$gte, \$lt, \$lte

- Điều kiện

- \$cond: [if: {}, then: "đúng thì làm gì", else: "sai thì làm gì"]

- \$switch: {\$branches: [

- {case: 1, then: "nhậu"},

- {case: 2, then: "uống nước lọc"}

-]}

- Toán tử tham chuỗi

- \$concat: ["\$ho", " ", "\$ten"]

- \$toUpperCase: "\$hoTen", \$toLowerCase: "\$hoTen"

- trích xuất chuỗi: `substrCP["$email", 0, 5]` // lấy chuỗi từ vị trí 0 -> 4
- `$set: { $strLenCP: "$hoTen" }` // lấy độ dài chuỗi
- **Toán tử mảng**
 - `$size, $arrayElemAt: ["$dsDienThoai", 0]` // lấy vị trí thứ 0 của số điện thoại
 - `$reduce: {`
 - input: “mảng cần xử lý”,
 - initialValue: “giá trị khởi tạo”,
 - in: {
 - `$add: ["$$value", "$$this"]`
 - // \$\$value : giá trị tích lũy (ví dụ ban đầu giá trị
// = 0 thì giá trị tích lũy cũng = 0)
 - // \$\$this: giá trị thêm vào sau (ví dụ tính tổng 1..n => 0 + 1 + 2
..... + n)
 - }
- **Toán tử ngày tháng**
 - `$year, $month,`
 - `$dayOfWeek: "$ngaySinh"` // 1: chủ nhật, 2 : thứ 2, 3: thứ 3
 - `$dateToString: { format: "%d/$m/%Y", date: "$ngaySinh" }`
 - // chuyển ngày về chuỗi dưới dạng day(ngày), month (tháng), Year (năm)

- Các trường hợp đặc biệt:
 - Tìm nhỏ nhất và lớn nhất trong cùng một lần xuất ra
 - \$facet: {


```
lonNhat: [ {$sort: {tongDan: -1}}, {$limit: 1} ],
nhoNhat:[ {$sort: {tongDan: 1}}, {$limit: 1} ]
```
 - }
- Phân loại theo khoảng giá trị
 - \$bucket: {


```
$groupBy: "$tongDan", // trường để phân nhóm
boundaries: [0, 1000, Infinity], // các nhóm phân [0, 1000], [1000, +vô cùng)
default: "những giá trị nằm ngoài khoảng [0, 1000, Infinity]",
output: { // dùng để thống kê cho mỗi bucket
  soDan: {$sum: 1} // ví dụ đếm số dân
}
```
 - }
 - Giải thích: số dân nằm trong [0, 1000] hoặc [1000, + vô cùng)
- So sánh giữa các trường trong cùng một document
 - \$expr: { \$lt: ["\$..", "\$.."] }

Index

1 Tạo index

```
// Single field index
db.collection.createIndex({ field: 1 })      // tăng dần
db.collection.createIndex({ field: -1 })      // giảm dần

// Compound index (nhiều trường)
db.collection.createIndex({ field1: 1, field2: -1 })

// Unique index
db.collection.createIndex({ field: 1 }, { unique: true })
db.collection.createIndex({ field1: 1, field2: 1 }, { unique: true }) // compound
unique

// Multikey index (MongoDB tự động khi field là array)
db.collection.createIndex({ arrayField: 1 })

// Text index
db.collection.createIndex({ field: "text" })
db.collection.createIndex({ field1: "text", field2: "text" }) // nhiều trường

// Hashed index (thường dùng cho sharding)
db.collection.createIndex({ field: "hashed" })

// Wildcard index (index tất cả field tự động)
db.collection.createIndex({ "$**": 1 })

// TTL index (xóa document tự động sau thời gian)
db.collection.createIndex({ expireAt: 1 }, { expireAfterSeconds: 0 })

// Index với collation (tùy locale & case/accents)
db.collection.createIndex({ field: 1 }, { collation: { locale: "vi", strength: 2 } })
```

2 Xem, xóa index

```
// Xem tất cả index của collection
db.collection.getIndexes()

// Xóa 1 index
db.collection.dropIndex("index_name")

// Xóa tất cả index
db.collection.dropIndexes()
```

3 Query với explain (xem index được sử dụng)

```
// Query đơn giản
db.collection.find({ field: value }).explain()
db.collection.find({ field: value }).explain("queryPlanner") // mặc định
db.collection.find({ field: value }).explain("executionStats") // có stats
db.collection.find({ field: value }).explain("allPlansExecution") // chi tiết tất cả
plan
```

4 Text search

```
db.collection.find({ $text: { $search: "keyword" } })
db.collection.find({ $text: { $search: "\"exact phrase\"" } })
db.collection.find({ $text: { $search: "MongoDB", $caseSensitive: true,
$diacriticSensitive: true } })
```

5 Collation (so sánh & sort theo ngôn ngữ)

```
// Case-insensitive tìm kiếm
db.collection.find({ name: "nguyen" }).collation({ locale: "vi", strength: 2 })

// Accent-insensitive tìm kiếm
db.collection.find({ name: "cafe" }).collation({ locale: "vi", strength: 1 })

// Tạo index với collation
db.collection.createIndex({ name: 1 }, { collation: { locale: "vi", strength: 2 } })
```

```
// Sort theo locale
db.collection.find().sort({ name: 1 }).collation({ locale: "vi" })
```

Lưu ý: ESR = Equality → Sort → Range

```
db.cars.createIndex({ model: 1, manufacturer: 1, price: 1 })
```

model = equality

manufacturer = sort

price = range

Nói ngắn: ESR giúp index compound dùng hiệu quả cho query kết hợp điều kiện, sắp xếp và range.

II. CẤU HÌNH REPLICA SET

1. Cấu hình cho từng node

node1.cfg

```
storage:
  dbPath: d:\db\ReplicaSet\node1\db (lưu ý chỗ này bắt buộc
  cần phải mỗi node mỗi db khác nhau)
net:
  bindIp: localhost hoặc 0.0.0.0
  port: 27011
security:
  authorization: enabled
  keyFile: d:\db\pki\keyfile
systemLog:
  destination: file
  path: d:\db\ReplicaSet\node1\mongod.log
  logAppend: true
```

```
replication:
```

```
  replSetName: rep-example
```

```
Tương tự cho node2.cfg (port 27012), node3.cfg (port 27013).
```

2. Khởi chạy các node

```
mongod -f node1.cfg
```

```
mongod -f node2.cfg
```

```
mongod -f node3.cfg
```

Mac

```
sudo chmod 600 /đường/dẫn/đến/keyfile
```

Windows

(Kiểm tra lỗi vào file log)

(Chỗ này có thể gặp lỗi keyfile are too open, nếu gặp thì cần phải sử dụng lệnh:

```
icacls d:\db\pki\keyfile /inheritance:r  
icacls d:\db\pki\keyfile /grant %USERNAME%:F
```

(Cách để biết UserName: echo %USERNAME%)

```
)
```

(Nếu bật terminal gõ lệnh này nó không nhấp nháy thì, kill các tiến trình có port giống nó.

Ví dụ, port 27011:

```
+tìm port: netstat -ano | findstr: 27011  
+kill port giống nó: taskkill /PID 27011 /F)
```

3. Kết nối đến một node

```
mongo --host localhost:27011
```

```
mongo -port 27011
```

4. Khởi tạo Replica Set

```
rs.initiate()
```

5. Tạo user quản trị

```
use admin
db.createUser({
  user: "admin",
  pwd: "admin",
  roles: [ { role: "root", db: "admin" } ]
})
```

6. Đăng nhập lại bằng tài khoản quản trị

```
// ở ngoài db
mongo --host rep-example/localhost:27011 -u admin -p admin
--authenticationDatabase admin

// nếu đã vô rồi thì có thẻ xác thực
use admin
db.auth("user name", "password")
```

7. Thêm node và xóa Replica Set

```
// Thêm
rs.add("localhost:27012")
rs.add("localhost:27013")
rs.addArb("localhost:28000") // có thẻ thêm dữ liệu
rs.add({ host: "localhost:27015", arbiterOnly: true }) // không thêm dữ liệu chỉ bâu
cử

// Xóa
rs.remove("localhost:27014")
Gặp lỗi tiếp sử dụng lệnh này để fix:
db.adminCommand({
  setDefaultRWConcern: 1,
  defaultWriteConcern: { w: "majority" },
  defaultReadConcern: { level: "majority" }
});
```

8. Kiểm tra trạng thái

```
// kiểm tra các node
rs.status()

// Kiểm tra cấu hình
cfg = rs.conf()
printjson(cfg)

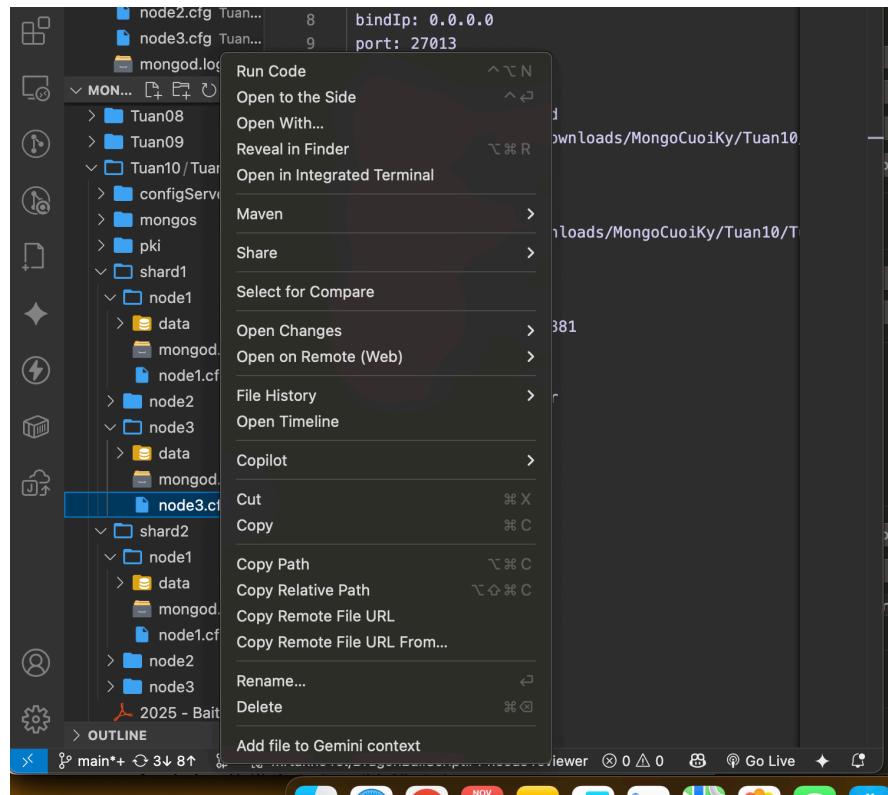
// để đọc từ secondary
db.getMongo.setReadPref("secondary")
db.collection.find()

// thay đổi độ ưu tiên để bầu cử
cfg = rs.conf()
cfg.members[1].priority = 2;
rs.reconfig(cfg)
```

III. CẤU HÌNH SHARDED CLUSTER

Sử dụng vs code để chỉnh các file config cho nhanh

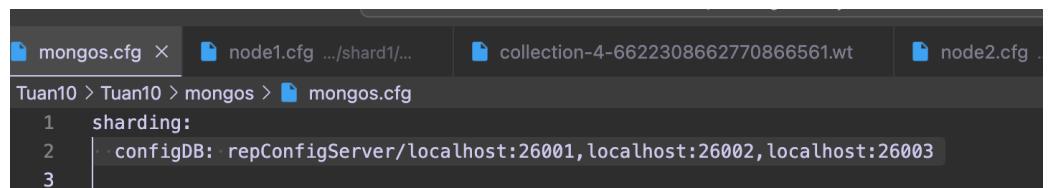
copy path, sau đó cd ctrl + v



Các lỗi gặp trong sharding (Lỗi thì nằm trong mongo.log)

- **Lỗi 1:** trước tiên phải bật các node trong thư mục configServer trước
Config Server là Trái tim của Shard Cluster
- **Lỗi 2 khi tạo user:**
MongoServerError[NotWritablePrimary]: not primary

- rep23703881 [direct: secondary] admin>
- cần [direct: primary] mới tạo được
- Cách fix: rs.status()
- xem thử primary port nào
- kết nối với port đó: mongosh —port 27013 (đây là primary tùy vào trường hợp)
- lỗi 3:
- khi kết nối với mongos=> mongos khi sử dụng mongos
- mongos -f mongos.cfg
- mongos là router sử dụng được các câu lệnh này:
 - **sh.enableSharding()** (Bật sharding cho DB)
 - **sh.shardCollection()** (Thiết lập Shard Key cho Collection)
 - **sh.addShard()** (Thêm Shard mới)
 - **sh.status()** (Kiểm tra trạng thái Cluster)
- lỗi 4: nếu muốn thêm dữ liệu thì thêm vào PRIMARY
- Cách fix: rs.status()
- xem thử primary port nào
- kết nối với port đó: mongosh —port 27013
- lỗi 5: các node trong 1 shard thì phải theo chỉ 1 replSetName

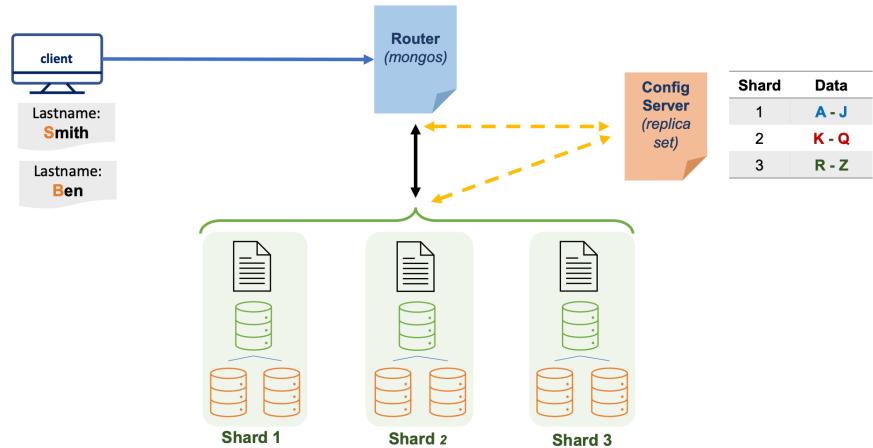


```

mongos.cfg × node1.cfg .../shard1/... collection-4-6622308662770866561.wt node2.cfg
Tuan10 > Tuan10 > mongos > mongos.cfg
1   sharding:
2   . configDB: repConfigServer/localhost:26001,localhost:26002,localhost:26003
3

```

nhớ thêm các (localhost: port) ở trong configServer vào



***** Tất cả chỉnh bindip: localhost để khi add vào shard nó dễ

CONFIG SERVERS (CSRS)

Start CSRS

```
mongod --config node1.cfg
mongod --config node2.cfg
mongod --config node3.cfg
```

(Mac book)

Nếu mở không được nữa kiểm tra đường port có bị đang sử dụng hay không:

+ tìm port 26002: sudo lsof -i :26002 // thay port trong đê
+ kill 12345 (12345: PID)
(Vì port đó đang bị chiếm hoặc có thể config sang port khác để sử dụng)

(Windows)

```
Tìm PID đang chiếm Port    `netstat -ano  
Đóng (Kill) Tiến trình      taskkill /PID <PID> /F (Ví dụ: taskkill /PID 12345 /F )
```

Mac

sudo chmod 600 /đường/dẫn/đến/keyfile

Windows

(Kiểm tra lỗi vào file log)
(Chỗ này có thể gặp lỗi keyfile are too open, nếu gặp thì cần phải sử dụng lệnh:
icacls d:\db\pki\keyfile /inheritance:r
icacls d:\db\pki\keyfile /grant %USERNAME%:F
(Cách để biết UserName: echo %USERNAME%)
)

Connect

mongosh --port 26001

Initiate

rs.initiate()

Create admin

```
use admin
db.createUser({user: "admin",pwd:"admin",roles:
[ {role:"root",db:"admin"}]})
```

Auth

```
db.auth("admin","admin")
```

Add members

```
use admin
rs.add("localhost:26002")
rs.add("localhost:26003")
```

MONGOS

Start Mongos

```
mongos --config mongos.cfg
```

Connect Mongos

```
mongosh --port 26000 -u admin -p admin --
authenticationDatabase admin
```

Lỗi:

```
MongoNetworkError: connect ECONNREFUSED 127.0.0.1:40000
```

Gặp lỗi này phải bật tất cả các mongo trong folder configServers

Check

```
sh.status()
```

SHARD1

Start shard nodes

```
mongod --config node1.cfg
mongod --config node2.cfg
mongod --config node3.cfg
```

Nếu mở không được nữa kiểm tra đường port có bị đang sử dụng hay không:

+ tìm port 26002: sudo lsof -i :26002 // thay port trong đê
+ kill 12345 (12345: PID)
(Vì port đó đang bị chiếm hoặc có thể config sang port khác để sử dụng)

(Kiểm tra lỗi vào file log)
(Chỗ này có thể gặp lỗi keyfile are too open, nếu gặp thì cần phải sử dụng lệnh:
icacls d:\db\pki\keyfile /inheritance:r
icacls d:\db\pki\keyfile /grant %USERNAME%:F
(Cách để biết UserName: echo %USERNAME%)
)

Connect & initiate

```
mongosh --port 27011
rs.initiate()
```

```
(Phải đứng ở primary mới add, remove được và use
admin )
MongoServerError[AuthenticationFailed]: Authentication failed.
tạo lại admin:
use admin
db.createUser({user: "admin", pwd: "admin", roles: [{role: "root", db:
"admin"}]})
```

Add members

```
rs.add("localhost:27012")
rs.add("localhost:27013")
```

Add Shard1 to cluster(mongos)

```
sh.addShard("repShard1/
localhost:27022,localhost:27023,localhost:27024")
ở trong mongos
```

Check

```
sh.status()
*** Nếu gặp lỗi
Vào lại chô đựng mô hình replicase
cfg = rs.conf()
cfg.members.forEach(member => {
    // Cập nhật tên host thành localhost cho tất cả các
    members
    const port = member.host.split(":")[1];
    member.host = "localhost:" + port;
});
// Áp dụng cấu hình mới
rs.reconfig(cfg);

thêm lại các node trong replicat sett đó
rs.add("")
```

```
rs.status()
```

IMPORT DATA

Copy file .js rồi add vào node primary

SHARDING

```
sh.enableSharding("climbing")
sh.shardCollection("climbing.mountains", {height:1})
sh.status()
```

USER

Create student:

```
use climbing
db.createUser({user:"student",pwd:"student",roles:[{role:"readWrite",db:"climbing"}]})
```

Login student:

```
mongosh --port 27013 -u student -p student --
authenticationDatabase climbing
```

Test insert: (Phải ở primary node)

```
use climbing
db.mountains.insert({name:"A",height:5000})
```

