#HCMUS
#MongoDB
#NoSQL

Thông tin kì thi

Phòng thi: E404 (NVC)
Thời gian: 10/07/2024 - 13h30

```
--------------------
- Các câu hỏi tình huống (Cho data => truy xuất)
- Thi 90' (Không mang tài liệu)
- Các câu hỏi lý thuyết (trong bài học)

MongoDB
- Học kỳ từ chương 1 đến chương 5 trên sách
- Cho collections => query
- Các câu hỏi lý thuyết liên quan
- Sự khác nhau giữa hai hệ quản trị cơ sở dữ liệu

SQL: Tập trung 9 Chapters đầu tiên
- Ôn kỹ chương 5: so sánh sự khác nhau giữa các loại JOIN
- Chương 10 - 11 coi thêm
- Các câu hỏi lý thuyết (vd: sự khác nhau giữa primary key - foreign
key)
```

PostgreSQL vs MongoDB

#PostgreSQL

Sự khác nhau giữa MongoDB và PostgreSQL thể hiện qua nhiều khía cạnh như cấu trúc dữ liệu, mô hình lưu trữ, ngôn ngữ truy vấn, khả năng mở rộng, và các tính năng đặc biệt khác.

Dưới đây là bảng so sánh chi tiết giữa MongoDB và PostgreSQL:

Đặc điểm	MongoDB	PostgreSQL
Loại cơ sở dữ liệu	Cơ sở dữ liệu NoSQL, tài liệu (document-based)	Cơ sở dữ liệu SQL, quan hệ (relational)
Mô hình dữ liệu	JSON-like documents (BSON)	Bảng với hàng và cột

Đặc điểm	MongoDB	PostgreSQL
Schema	Schema-less (linh hoạt, không có cấu trúc cố định)	Schema-based (cấu trúc cố định, bắt buộc phải định nghĩa schema)
Ngôn ngữ truy vấn	MongoDB Query Language (MQL)	SQL (Structured Query Language)
Hỗ trợ JOIN	Không hỗ trợ trực tiếp JOIN phức tạp, có thể dùng lookup	Hỗ trợ đầy đủ các loại JOIN (INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN)
Transactions	Hỗ trợ transactions đa tài liệu từ phiên bản 4.0	Hỗ trợ transactions đầy đủ, ACID-compliant
Khả năng mở rộng	Thiết kế cho horizontal scaling, sharding	Chủ yếu là vertical scaling, hỗ trợ một số tính năng horizontal scaling (như table partitioning)
Indexing	Hỗ trợ nhiều loại index như single field, compound, multikey, text, geoSpatial	Hỗ trợ nhiều loại index như B-tree, Hash, GiST, GIN, SP-GiST, BRIN
Replication	Hỗ trợ replica sets và sharding	Hỗ trợ master-slave replication và streaming replication (logical replication)
Consistency	Eventually consistent by default, có thể cấu hình strong consistency	Strong consistency by default (ACID properties)
Use Cases	Phù hợp cho các ứng dụng với dữ liệu phi cấu trúc, hệ thống phân tán, cần mở rộng dễ dàng	Phù hợp cho các ứng dụng cần tính toàn vẹn dữ liệu, giao dịch phức tạp, phân tích dữ liệu
Hỗ trợ JSON	Hỗ trợ lưu trữ và truy vấn tài liệu JSON natively	Hỗ trợ lưu trữ và truy vấn tài liệu JSON với các kiểu dữ liệu JSON và JSONB
Full-Text Search	Hỗ trợ tích hợp full-text search	Hỗ trợ tích hợp full-text search, mặc dù không mạnh bằng các công cụ chuyên dụng như Elasticsearch

1. Introduce

MongoDB là một hệ quản trị cơ sở dữ liệu NoSQL, mã nguồn mở. Nó được thiết kế để xử lý các dữ liệu lớn và không có cấu trúc hoặc bán cấu trúc

1.1. Thao tác với database

Tạo, xóa database

```
use mydb  // kết nối đến mydb
show dbs  // hiển thị các db
db.dropDatabase() // xoá db hiện tại
```

Tạo user: `db.createUser()`

```
// tạo người dùng có tên user1 có quyền đọc ghi trên database mydb
db.createUser(
        {
                user: "user1",
                pwd: "xxx",
                roles: [{ role: "readWrite", db: "mydb" }]
        }
)
```

Tạo collection: `db.createCollection('collection_name')`

```
db.createCollection('people') // tạo collection 'people'
show collections // hiển thị các collections
```

Xóa collection: `db.collection_name.drop()`

```
db.people.drop() // xoá collection 'people'
```

1.2. Import dữ liệu

Import JSON

```
# mongoimport --db <db_name> --collection <collection_name> --file
<filename.json>

mongoimport --db=mydb --collection=students --file=students.json
```

Import CSV

```
# mongoimport --db <db_name> --collection <collection_name> --file
<file_name.csv> --type csv --headerline

mongoimport --db mydb --collection tracks --file track_raw.csv --
type csv --fields="title,artist,album,len,rating,count"
```

1.3. Làm việc với Docker

```
# Pull image
docker pull mongo

# Create container whose name is mongodb
docker run -d -p 27017:27017 --name mongo mongodb

# Execute container and work inside it with bash
docker exec -it mongodb bash

# Execute mongosh
mongosh
```

2. CRUD

#CRUD

CRUD (Create, Read, Update, Delete) là bốn hoạt động cơ bản mà bạn có thể thực hiện trên dữ liệu trong MongoDB. Dưới đây là chi tiết về từng hoạt động và cách thực hiện chúng trong MongoDB:

2.1. Create (Tạo)

Để tạo một document mới trong MongoDB, bạn sử dụng lệnh `insertOne()` hoặc `insertMany()`.

Ví dụ:

```
// Kết nối đến database và collection
use myDatabase
db.myCollection.insertOne({
    name: "John Doe",
    age: 30,
```

```
    email: "john.doe@example.com"
})
```

Lệnh này sẽ chèn một document mới vào collection `myCollection`.

2.2. Read (Đọc)

Để đọc hoặc truy xuất dữ liệu từ MongoDB, bạn sử dụng lệnh `find()`.

Ví dụ:

```
// Lấy tất cả các document trong collection
db.myCollection.find()

// Lấy các document thoả mãn điều kiện
db.myCollection.find({ age: { $gte: 25 } })
```

Lệnh `find()` có thể chấp nhận một tiêu chí tìm kiếm để lọc kết quả.

2.3. Update (Cập nhật)

Để cập nhật các document trong MongoDB, bạn sử dụng lệnh `updateOne()`, `updateMany()` hoặc `replaceOne()`.

2.3.1. `$set`

Thay thế giá trị nếu trường cần cập nhật đã có và thêm mới nếu chưa có

```
// Cập nhật một document
db.myCollection.updateOne(
    { name: "John Doe" }, // Điều kiện
    { $set: { age: 31 } } // Cập nhật
)

// Cập nhật nhiều document
db.myCollection.updateMany(
    { age: { $gte: 25 } }, // Điều kiện
    { $set: { status: "active" } } // Cập nhật
)
```

Lệnh `updateOne()` sẽ cập nhật document đầu tiên thoả mãn điều kiện, trong khi `updateMany()` sẽ cập nhật tất cả các document thoả mãn điều kiện.

2.3.1. `$unset`

Loại bỏ trường ra khỏi document cần cập nhật

```
db.student.updateOne(
        { first_name: "Lenny" },
        { $unset: { subject: 1 } }
)
```

Trường hợp trên là loại bỏ trường tên là `subject`

2.3.2. `$push, $addToSet`

Thêm 1 giá trị vào cuối 1 mảng nếu mảng đó tồn tại và thêm mảng mới nếu chưa tồn tại

```
db.students.updateOne(
        {full_name: "Paul Pan"},
        {$push: {subject: "science"}});


db.students.updateOne(
        {full_name: "Paul Pan"},
        {$addToSet: {subject: "chemistry"}});
```

2.3.3. `$pull`

Remove 1 phần tử ra khỏi mảng

```
db.students.updateOne(
        {full_name: "Arya Stark"},
        {$pull: {subject: "science"}});
```

2.3.4. `$pop`

Remove phần tử đầu tiên hoặc cuối cùng của mảng

```
db.students.findOneAndUpdate(
        {full_name: "Arya Stark"},
        {$pop: {subject: -1}},
```

```
        {returnNewDocument : true}
); //1: phần tử cuối cùng; -1: phần tử đầu tiên
```

2.3.5. `$inc`

Tăng hoặc giảm giá trị

```
db.students.updateOne(
        {first_name: "Paula"},
        {$inc: {age: 3}} // tăng tuổi thêm 3 đơn vị
);
```

2.4. Delete (Xoá)

Để xoá các document trong MongoDB, bạn sử dụng lệnh `deleteOne()` hoặc `deleteMany()`.

```
// Xoá một document
db.myCollection.deleteOne({ name: "John Doe" })

// Xoá nhiều document
db.myCollection.deleteMany({ status: "inactive" })
```

Lệnh `deleteOne()` sẽ xoá document đầu tiên thoả mãn điều kiện, trong khi `deleteMany()` sẽ xoá tất cả các document thoả mãn điều kiện.

2.5. Các lệnh khác hữu ích trong CRUD:

`findOne()`: Trả về document đầu tiên thoả mãn điều kiện.

```
db.myCollection.findOne({ name: "John Doe" })
```

`replaceOne()`: Thay thế một document bằng document mới.

```
db.myCollection.replaceOne(
  { name: "John Doe" },
  { name: "John Doe", age: 31, email: "john.doe@newdomain.com" },
  { upset: true } // trường hợp không tìm được thì thêm mới
)
```

3. Regex

```
// name có chứa "Charger"
db.products.find({name: {$regex: "Charger"}});

// name bắt đầu bằng "AC3 Case"
db.products.find({name: {$regex: "^AC3 Case"}});

// name kết thúc bằng "Warranty"
db.products.find({name: {$regex: "Warranty$"}});

// i: case-insensitive
db.products.find({name: {$regex: "phone service", $options:
'i'}});

db.products.find({name: /^AC3 Case/}).sort({price: -1, rating:
1});

// name chứa AC3 hoặc ac7, i: case-insensitive
db.products.find({name: /AC3|ac7/i});
```

4. Aggregate

#Query

SQL	Aggregate operator
SELECT	`$project`
FROM	`db.collection.aggregate(...)`
JOIN	`$unwind`
WHERE	`$match`
GROUP BY	`$group`
HAVING	`$match`
ORDER BY	`$sort`
LIMIT, OFFSET	`$skip, $limit`

5. Cursor

Trong MongoDB, một **cursor** là một đối tượng cho phép bạn duyệt qua kết quả của một truy vấn.

Khi bạn thực hiện một truy vấn sử dụng phương thức `find()`, MongoDB trả về một con trỏ tới kết quả truy vấn đó.

Con trỏ này có thể được sử dụng để duyệt qua từng document trong tập kết quả một cách tuần tự.

Các khái niệm chính về Cursor:

1. **Khởi tạo Cursor**:

Khi bạn gọi `db.collection.find()`, MongoDB trả về một cursor trỏ tới các document phù hợp với tiêu chí truy vấn.

```
var cursor = db.myCollection.find({ age: { $gte: 25 } });
```

2. **Duyệt qua Cursor**:

Bạn có thể sử dụng các phương thức như `next()` hoặc `hasNext()` để duyệt qua các document.

```
while (cursor.hasNext()) {
   printjson(cursor.next());
}
```

3. **Phương thức Cursor hữu ích**:

`forEach(callback)`: Lặp qua tất cả các document trong cursor và áp dụng hàm callback cho mỗi document.

```
cursor.forEach(function(doc) {
 printjson(doc);
});
```

`toArray()`: Chuyển đổi cursor thành một mảng các document.

```
var documents = cursor.toArray();
printjson(documents);
```

`count()`: Trả về số lượng document trong cursor.

```
var count = cursor.count();
print("Number of documents: " + count);
```

`limit(n)` : Giới hạn số lượng document được trả về bởi cursor.

```
var limitedCursor = cursor.limit(10);
```

`skip(n)` : Bỏ qua n document đầu tiên trong cursor.

```
var skippedCursor = cursor.skip(5);
```

`sort(order)` : Sắp xếp các document trong cursor theo thứ tự xác định.

```
var sortedCursor = cursor.sort({ age: -1 });
```

4. **Cursor Timeout**:

Mặc định, MongoDB đóng cursor sau 10 phút không hoạt động. Bạn có thể thay đổi thời gian này hoặc tắt timeout bằng cách sử dụng tùy chọn `noCursorTimeout`.

```
var cursor = db.myCollection.find().noCursorTimeout();
```

5. **Batch Size**:

Cursor đọc các document theo lô (batch). Bạn có thể điều chỉnh kích thước của lô bằng phương thức `batchSize()`.

```
var cursor = db.myCollection.find().batchSize(50);
```

Ví dụ cụ thể về sử dụng Cursor:

```
// Khởi tạo một cursor cho truy vấn
var cursor = db.myCollection.find({ age: { $gte: 25 } });

// Duyệt qua từng document và in ra
cursor.forEach(function(doc) {
    printjson(doc);
});
```

```
// Giới hạn kết quả truy vấn
var limitedCursor = db.myCollection.find().limit(10);
limitedCursor.forEach(function(doc) {
    printjson(doc);
});

// Sắp xếp kết quả truy vấn
var sortedCursor = db.myCollection.find().sort({ age: -1 });
sortedCursor.forEach(function(doc) {
    printjson(doc);
});
```

6. Indexes

#Indexing

Indexes (chỉ mục) trong MongoDB là cấu trúc dữ liệu đặc biệt giúp tăng tốc độ truy vấn trên các collection. Bằng cách tạo index trên một hoặc nhiều field của document, bạn có thể cải thiện hiệu suất của các truy vấn tìm kiếm, sắp xếp và nhóm.

Các loại index trong MongoDB:

1. **Single Field Index** (Chỉ mục một trường)
   - Tạo index trên một field duy nhất.
   - **Ví dụ**:
     `javascript db.collection.createIndex({ name: 1 }); // 1: tăng dần, -1: giảm dần`
2. **Compound Index** (Chỉ mục hợp chất)
   - Tạo index trên nhiều field.
   - **Ví dụ**:
     `javascript db.collection.createIndex({ name: 1, age: -1 });`
3. **Multikey Index** (Chỉ mục đa khóa)
   - Tạo index trên các field chứa mảng. MongoDB tạo các entry index riêng lẻ cho mỗi giá trị trong mảng.
   - **Ví dụ**:
     `javascript db.collection.createIndex({ tags: 1 });`
4. **Text Index** (Chỉ mục văn bản)

- Được sử dụng để hỗ trợ tìm kiếm văn bản, bao gồm tìm kiếm từ khóa và tìm kiếm toàn văn.
- **Ví dụ**:
  `javascript db.collection.createIndex({ description: "text" });`

5. **Hashed Index** (Chỉ mục băm)
   - Được sử dụng cho sharding bằng cách băm giá trị của một field.
   - **Ví dụ**:
     `javascript db.collection.createIndex({ user_id: "hashed" });`
6. **Geospatial Indexes** (Chỉ mục không gian địa lý)
   - Được sử dụng cho các truy vấn không gian địa lý như tìm kiếm gần nhất.
   - **2dsphere Index**:
     `javascript db.collection.createIndex({ location: "2dsphere" });`
   - **2d Index**:
     `javascript db.collection.createIndex({ location: "2d" });`
7. **TTL Index** (Chỉ mục TTL)
   - Tự động xoá các document sau một khoảng thời gian nhất định.
   - **Ví dụ**:
     `javascript db.collection.createIndex( { createdAt: 1 }, { expireAfterSeconds: 3600 } );`

Các thao tác với index:

1. **Tạo Index**
   - **Ví dụ**:
     `javascript db.collection.createIndex({ name: 1 });`
2. **Xem các Index hiện có**
   - **Ví dụ**:
     `javascript db.collection.getIndexes();`
3. **Xoá Index**
   - **Ví dụ**:
     `javascript db.collection.dropIndex("name_1");`
4. **Xoá tất cả các Index**
   - **Ví dụ**:
     `javascript db.collection.dropIndexes();`

Index Tuning:

1. **Explain**:
   - Để phân tích hiệu suất truy vấn, bạn có thể sử dụng `explain()` để xem cách MongoDB sử dụng các index.
   - **Ví dụ**:
     ```
     javascript db.collection.find({ name: "Alice"
     }).explain("executionStats");
     ```
2. **Index Cardinality**:
   - Độ đặc biệt (cardinality) của các giá trị trong field được index ảnh hưởng đến hiệu quả của index. Các field có độ đặc biệt cao thường mang lại hiệu quả index tốt hơn.
3. **Index Size**:
   - Index tiêu tốn bộ nhớ và tài nguyên CPU. Bạn nên cân nhắc khi tạo quá nhiều index, vì điều này có thể ảnh hưởng đến hiệu suất ghi dữ liệu.

Lợi ích và hạn chế của Index:

Lợi ích:

- Tăng tốc độ truy vấn.
- Cải thiện hiệu suất sắp xếp và nhóm dữ liệu.
- Hỗ trợ tìm kiếm văn bản và truy vấn không gian địa lý.

Hạn chế:

- Tiêu tốn bộ nhớ.
- Làm chậm hiệu suất ghi dữ liệu do cần cập nhật index khi thêm, sửa hoặc xoá document.
- Quản lý phức tạp khi có nhiều index.