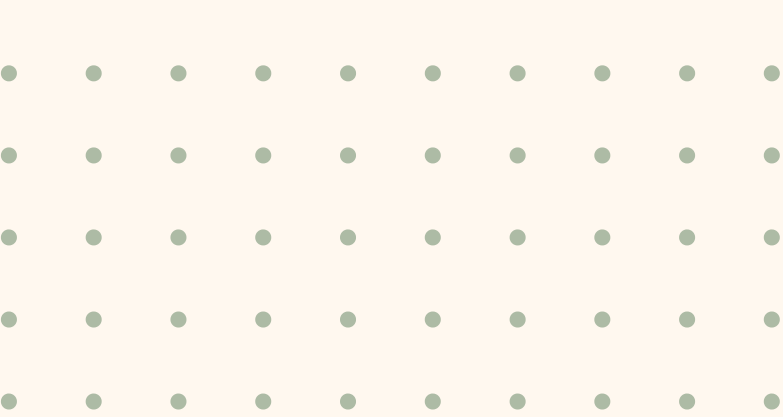


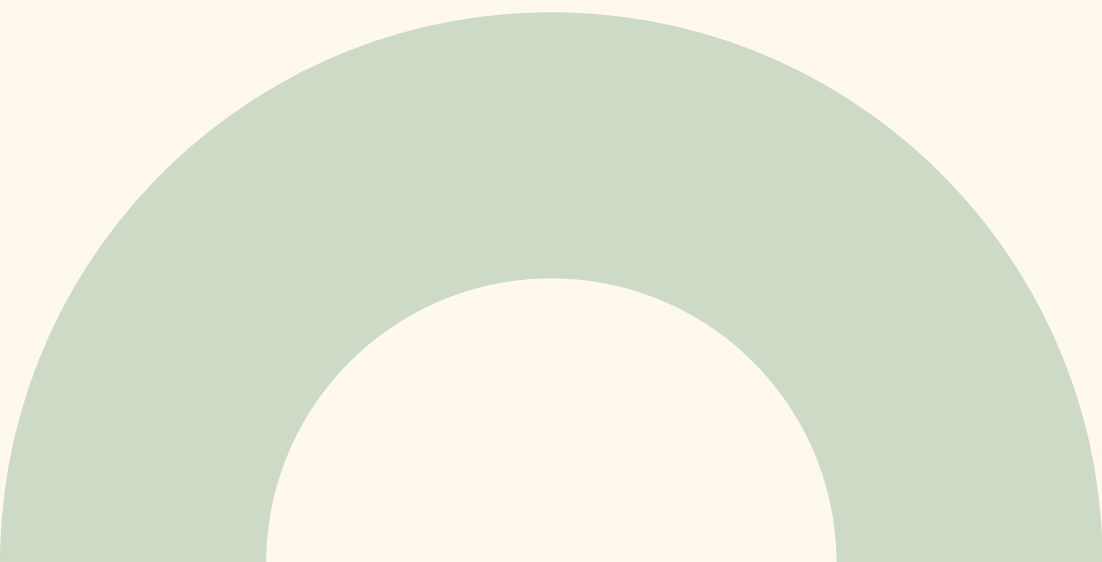
MONGODB – NHÓM 3

Nguyễn Hà Nam
Lê Quốc Lâm
Lê Thị Phương
Lê Quang Đạt
Bùi Nguyên Phong





NỘI DUNG



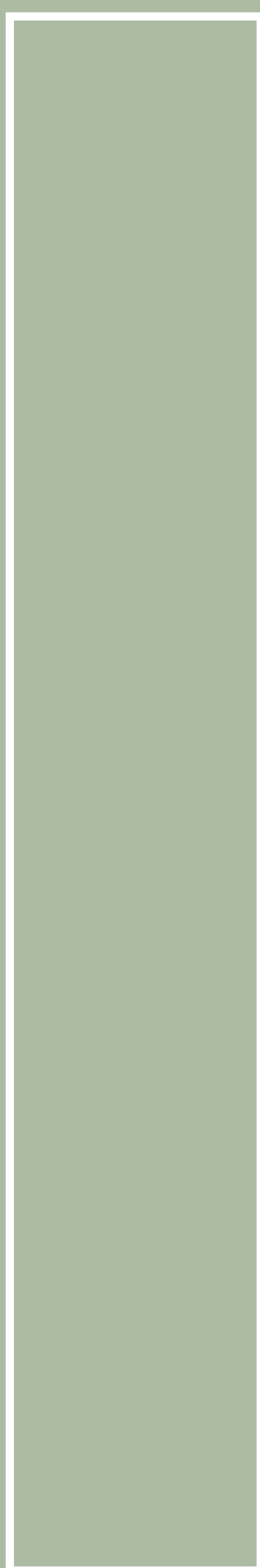
1. Giới thiệu

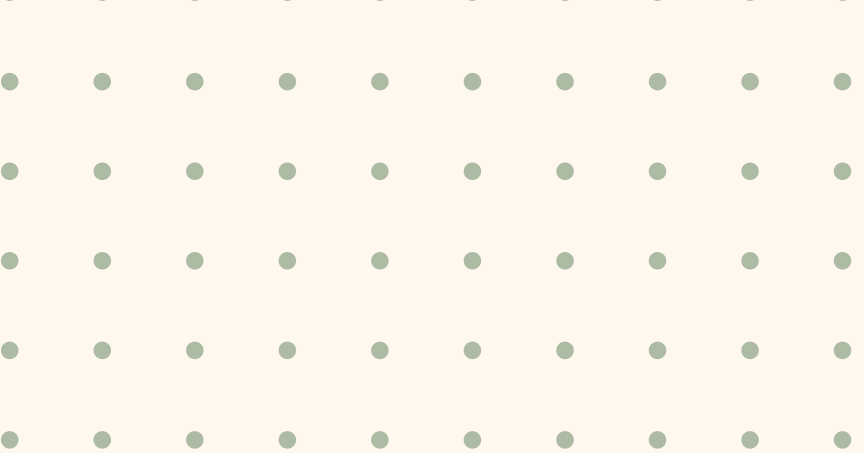
2. Truy vấn

3. INDEX

4. Hiệu năng

5. Kết luận

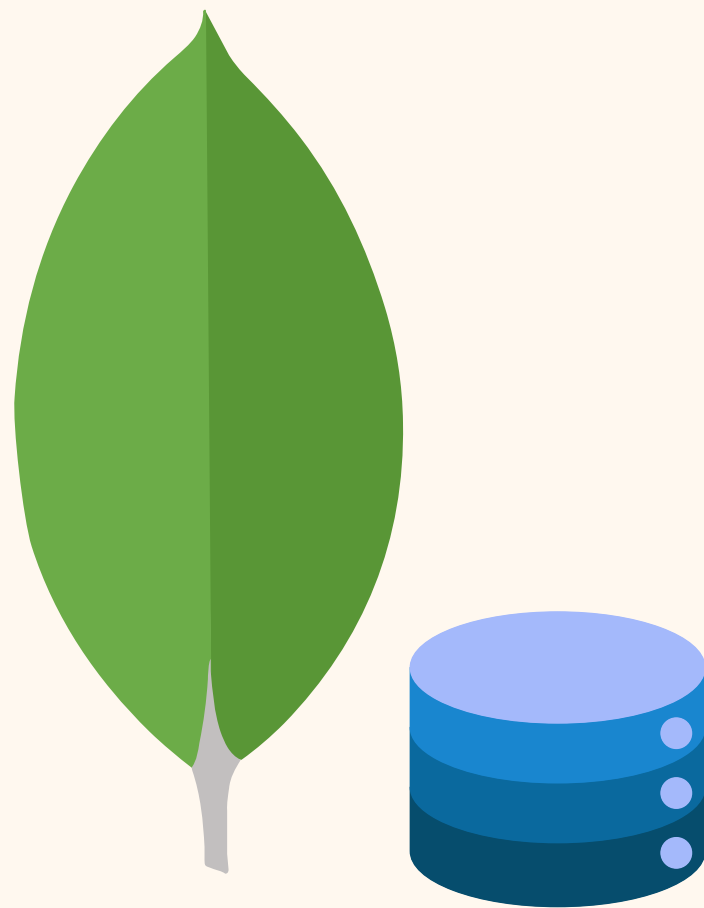




1. GIỚI THIỆU



GIỚI THIỆU - MONGODB LÀ GÌ ?



MongoDB lần đầu ra đời bởi MongoDB Inc., tại thời điểm đó là thế hệ 10, vào tháng Mười năm 2007, nó là một phần của sản phẩm PaaS (Platform as a Service) tương tự như Windows Azure và Google App Engine. Sau đó nó đã được chuyển thành nguồn mở từ năm 2009.

CƠ SỞ LÝ THUYẾT

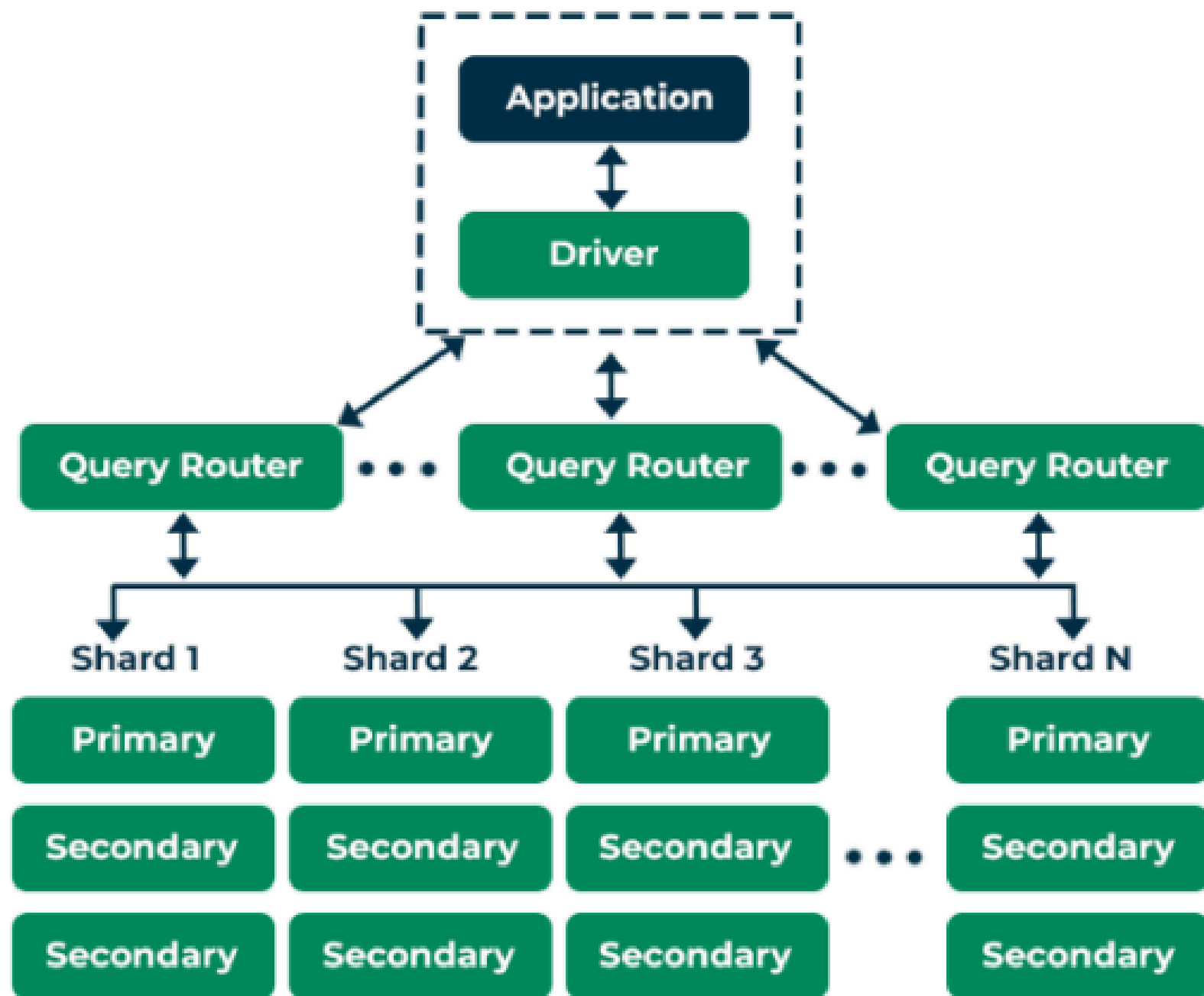
NoSQL

MongoDB là hệ quản trị cơ sở dữ liệu NoSQL mã nguồn mở, phát triển bởi MongoDB Inc. Trong hệ sinh thái NoSQL, có một số nhóm phổ biến như: Key-Value pair storage, Column Store, **Document Store**, Graph databases

Lưu trữ

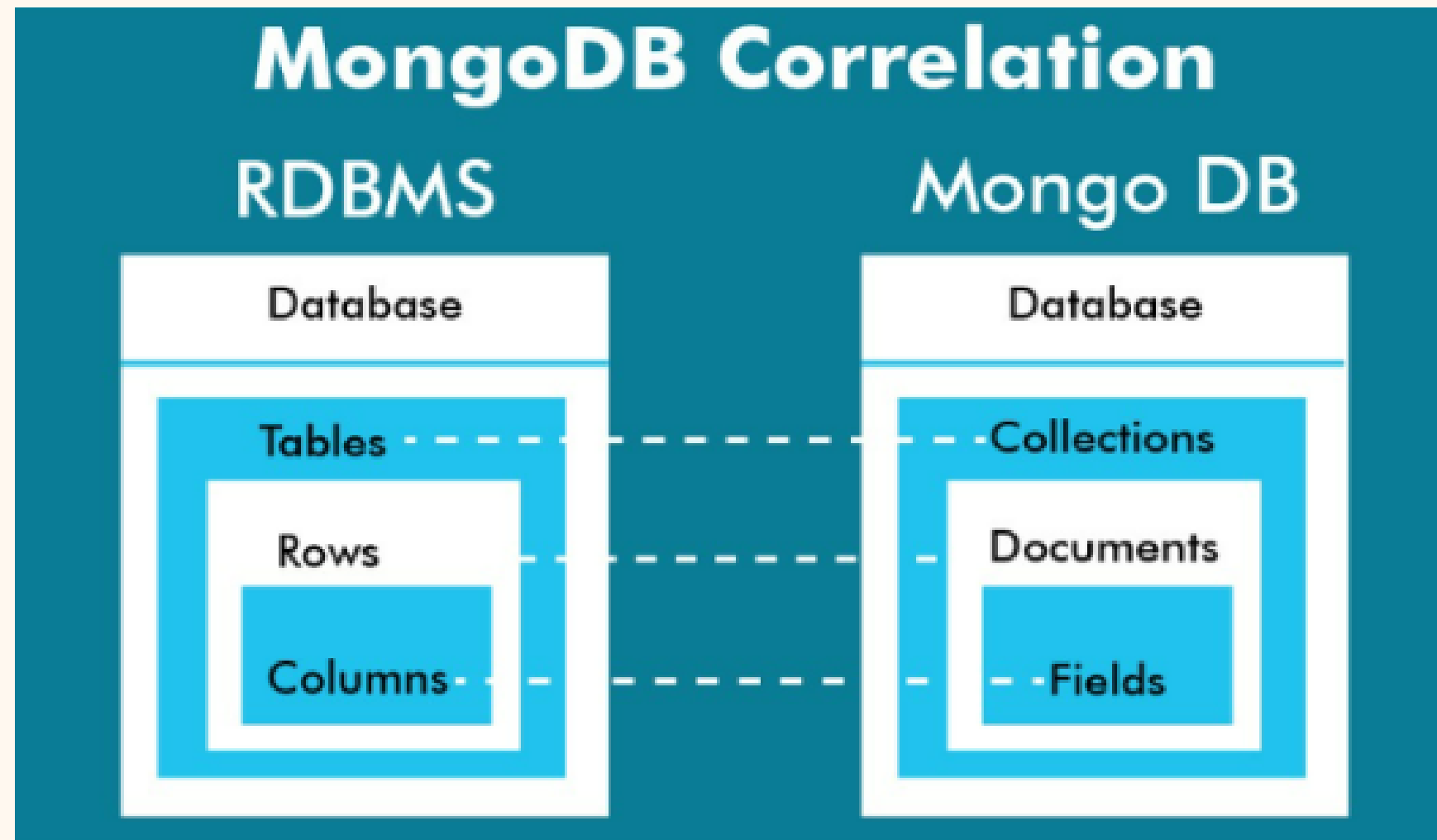
Dữ liệu trong MongoDB được lưu trữ dưới dạng Document với định dạng BSON, giúp linh hoạt trong việc lưu trữ dữ liệu phi cấu trúc hoặc bán cấu trúc mà không cần định nghĩa trước schema. MongoDB nổi bật với khả năng mở rộng và hiệu năng cao.

KIẾN TRÚC HOẠT ĐỘNG



- Ứng dụng gửi truy vấn đến Query Router (mongos).
- Mongos sử dụng metadata về phân vùng dữ liệu (sharding metadata) — được tải từ Config Servers — để xác định shard key và xác định shard nào chứa dữ liệu phù hợp.
- Sau đó chuyển tiếp truy vấn đến shard phù hợp.


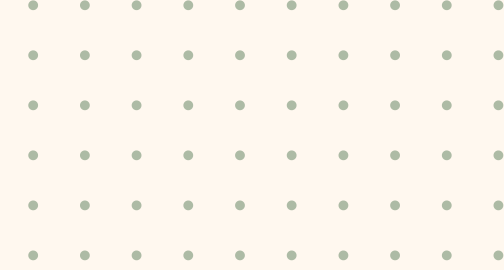
CÁC THÀNH PHẦN



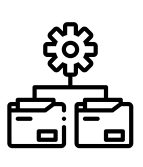
- MongoDB được xây dựng từ collections và documents.
- Tài liệu được tạo thành từ các cặp key-value pairs
- Các Collections , tương đương với các bảng trong SQL, chứa các tập hợp tài liệu.
- Một Document trong MongoDB tương ứng với một bản ghi (record) trong cơ sở dữ liệu quan hệ

Document linh hoạt hơn vì nó có thể chứa các trường (field) hoặc giá trị (value) có cấu trúc khác nhau.

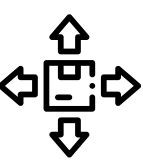
ĐẶC ĐIỂM CỦA MONGODB



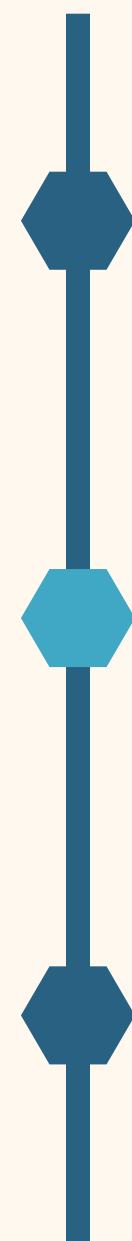
Lưu trữ dữ liệu dạng tài liệu



Không cần cấu trúc dữ liệu cố định




Khả năng mở rộng



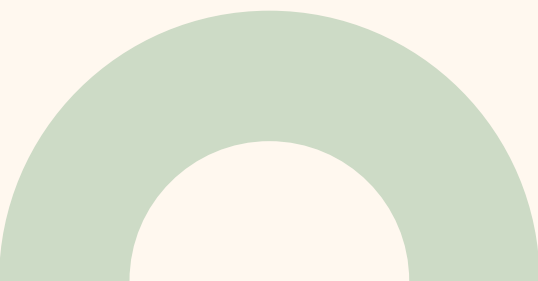
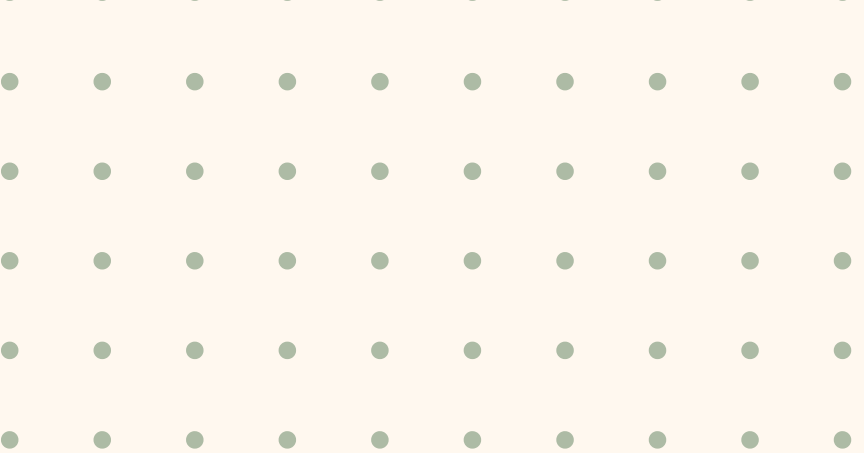
Truy vấn linh hoạt



Index



Sao lưu và nhân bản dữ liệu



2. TRUY VẤN



MỘT SỐ TRUY VẤN CƠ BẢN

Chèn dữ liệu

- **insertOne():** Chèn một tài liệu duy nhất vào collection.
- **insertMany():** Chèn nhiều tài liệu cùng một lúc.

```
db.students.insertOne({  
  name: "John Doe", age: 28,  
  subject: "Mathematics",  
  email: "john.doe@example.com"})
```



```
INSERT INTO students (name, age, subject, email)  
VALUES ('John Doe', 28, 'Mathematics',  
'john.doe@example.com');
```

MỘT SỐ TRUY VẤN CƠ BẢN

Truy vấn dữ liệu

Phương thức find

- Trả về tất cả bản ghi trong Collection.
- Trả về tài liệu với điều kiện cụ thể.
- Kết hợp nhiều điều kiện với nhau.

```
db.students.find({  
  name: "John Doe",  
  age: 28})
```



```
SELECT * FROM students  
WHERE name = 'John Doe' AND age = 28;
```

MỘT SỐ TRUY VẤN CƠ BẢN

Truy vấn dữ liệu

Xác định Keys cần trả về

Có thể thay đổi đối số thứ 2 truyền vào hàm find() để lấy ra Keys:

- 1 hoặc true để trả về Key đó.
- 0 hoặc false để loại Key đó.
- Toán tử projection khác.

```
db.students .find({},  
{name : 1, email : 1})
```



```
SELECT name, email FROM students;
```

MỘT SỐ TRUY VẤN CƠ BẢN

Truy vấn dữ liệu

Toán tử so sánh

- \$lt (less than).
- \$lte (less than or equal).
- \$gt (greater than).
- \$gte (greater than or equal).
- \$ne (not equal).

```
db.students.find({age:  
  {$gte: 18,  
  $lte: 30 } })
```



```
SELECT * FROM students  
WHERE age BETWEEN 18 AND 30;
```

MỘT SỐ TRUY VẤN CƠ BẢN

Truy vấn dữ liệu

Toán tử logic

- `$and`: yêu cầu tất cả các điều kiện phải thỏa mãn.
- `$or`: ít nhất 1 trong các điều kiện phải thỏa mãn.
- `$nor`: phủ định của toán tử `$or`.
- `$not`: phủ định kết quả của một điều kiện.

```
db.students.find({$and: [{age: {$gte: 18}}, {age: {$lte: 30}}]})
```

```
db.students.find({$or: [{age: {$gte: 18}}, {age: {$lte: 30}}]})
```

```
db.students.find({$nor: [{age: {$gte: 18}}, {age: {$lte: 30}}]})
```

```
db.students.find({age: {$not: {$gte: 30}}})
```

MỘT SỐ TRUY VẤN CƠ BẢN

Truy vấn dữ liệu

Toán tử tìm kiếm chuỗi (\$regex)

Sử dụng để tìm kiếm chuỗi theo mẫu, cho phép tìm kiếm các giá trị chuỗi có thể khớp với một mẫu nhất định.

Cú pháp

```
db.collection.find(  
  { "field": { "$regex": "pattern" } })
```

```
db.students.find({name: {$regex: /Jo/}})
```

```
db.students.find({name: {$regex: /oe$/}})
```

```
db.students.find({name: {$regex: "john",  
  $options: "i"}})
```

MỘT SỐ TRUY VẤN CƠ BẢN

Truy vấn dữ liệu

Truy vấn với mảng

- Toán tử \$in.
- Toán tử \$all.

```
db.students.find({ age: { "$in": [25, 30, 35] } })
```

```
db.students.find({subject:  
{$all: ["Mathematics", "Physics"]}})
```



```
SELECT * FROM students WHERE age IN (25, 30, 35);
```

```
SELECT * FROM students  
WHERE subject LIKE '%Mathematics%' AND subject LIKE  
'%Physics%';
```


MỘT SỐ TRUY VẤN CƠ BẢN

Cập nhật dữ liệu

- **updateOne():** Cập nhật một tài liệu duy nhất vào collection.
- **updateMany():** Cập nhật nhiều tài liệu cùng một lúc.
- **replaceOne():** thay thế toàn bộ tài liệu khớp với điều kiện.

```
db.students.updateOne({name: "Jane Smith"},  
{$set: {age: 33}})
```

```
db.students.replaceOne({ name: "Jane Smith" },  
{name: "Jane Smith", age: 33,  
  subject: ["Physics", "Mathematics"],  
  email: "jane.smith.new@example.com"},  
{ upsert: true })
```

MỘT SỐ TRUY VẤN CƠ BẢN

Xóa dữ liệu

- **deleteOne():** Xóa 1 tài liệu đầu tiên.
- **deleteMany():** Xóa tất cả tài liệu.

```
db.students.delete_one({"name": "Jane Smith"})
```

```
db.students.deleteMany({age: {$lt: 28}})
```



```
DELETE FROM students WHERE name = 'Jane Smith'  
LIMIT 1;
```

```
DELETE FROM students WHERE age < 28;
```

TRUY VẤN

Documents 8

Aggregations

Schema

Indexes 2

Validation

🕒 ▼

{ }

Generate query ✨

Explain

Reset

Find

</>

Options ▶

➕ ADD DATA ▼

📄 EXPORT DATA ▼

✎ UPDATE

🗑 DELETE

25 ▼

1 – 8 of 8

↺

⏪

⏩

▼

☰

{ }

|

📊

[Documents](#)

5

[Aggregations](#)[Schema](#)[Indexes](#)

2

[Validation](#)

▼ Stage 1 \$match



```
1  /**
2    * query: The query in MQL.
3    */
4  {
5    age: { "$gte": 20 }
6  }
```

Output after [\\$match](#) stage (Sample of 5 documents)

```
_id: ObjectId('674181b0228ba3bcf6f6a02d')
student_id: "A001"
name: "Le Quoc Lam"
age: 20
subjects: Array (2)
```

```
_id: ObjectId('67418204228ba3bcf6f6a032')
student_id: "A002"
name: "Le Quang Dat"
age: 20
subjects: Array (2)
```



▼ Stage 2 \$sort



```
1  /**
2    * Provide any number of field/order pair
3    */
4  {
5    name: 1
6  }
```

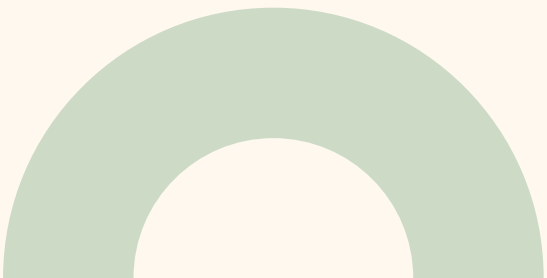
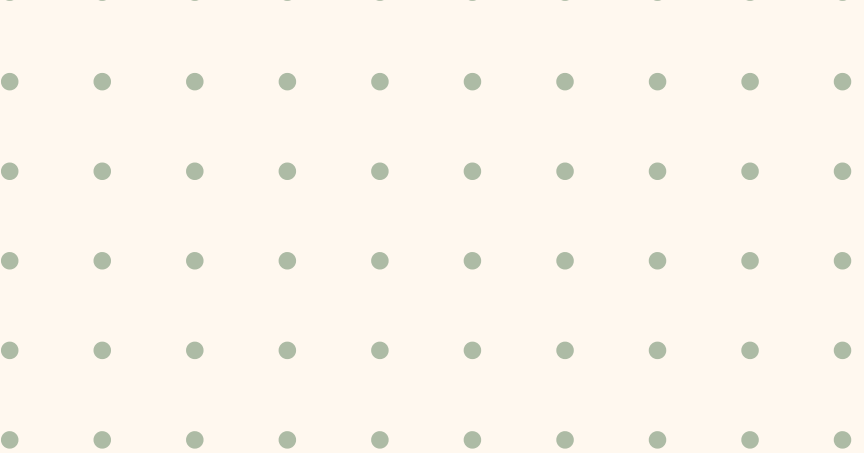
Output after [\\$sort](#) stage (Sample of 8 documents)

```
_id: ObjectId('67fbba64124dcb909105784b')
name: "Alice"
age: 25
city: "Hanoi"
```

```
_id: ObjectId('67fbba64124dcb909105784c')
name: "Bob"
age: 30
city: "Ho Chi Minh"
```

AGGREGATION FRAMEWORK

\$match	WHERE	Lọc các tài liệu theo điều kiện
\$project	SELECT	Chọn trường cần hiển thị, có thể đổi tên, tính toán trường mới
\$group	GROUP BY , SUM, AVG	Gom nhóm và tính toán các chỉ số tổng hợp
\$sort	ORDER BY	Sắp xếp kết quả tăng/giảm dần
\$limit	LIMIT	Giới hạn số lượng kết quả
\$skip	OFFSET	Bỏ qua một số dòng đầu (thường dùng phân trang)
\$unwind	Không có tương đương trực tiếp	Tách từng phần tử trong mảng thành tài liệu riêng
\$lookup	JOIN	Kết nối hai collection (giống JOIN giữa hai bảng trong SQL)



3. INDEX

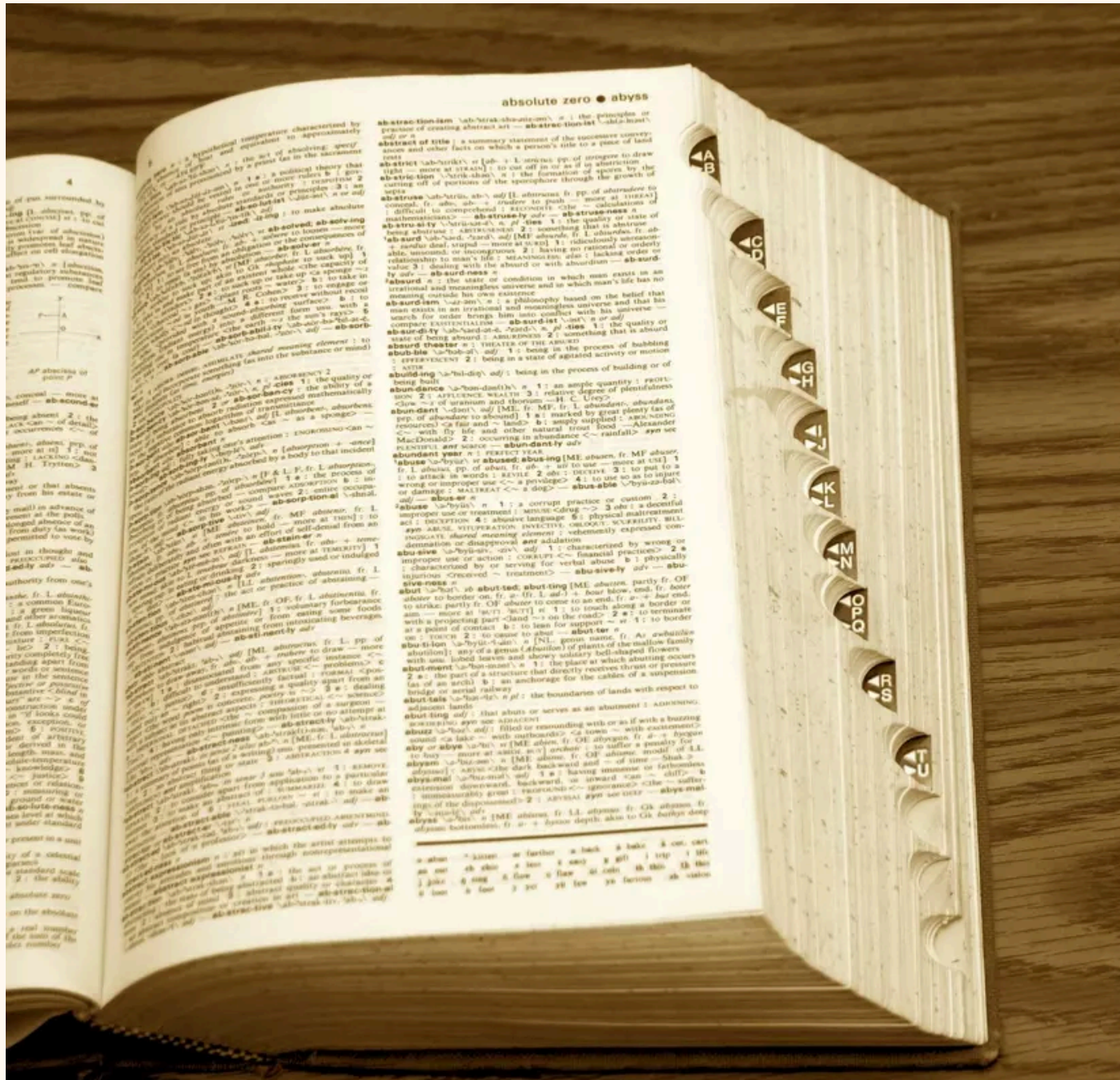


INDEX

🔍 1.Khái niệm



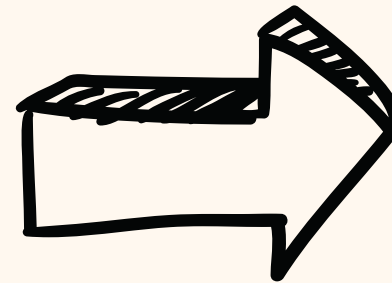
Index trong Database cũng
giống như mục lục
của một cuốn sách



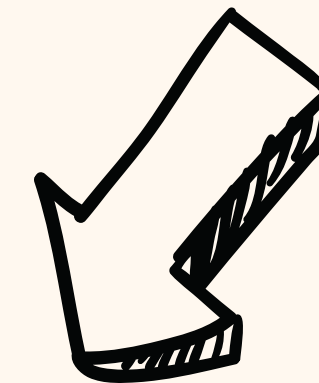
INDEX

🔍 2. Cách thức hoạt động ✕

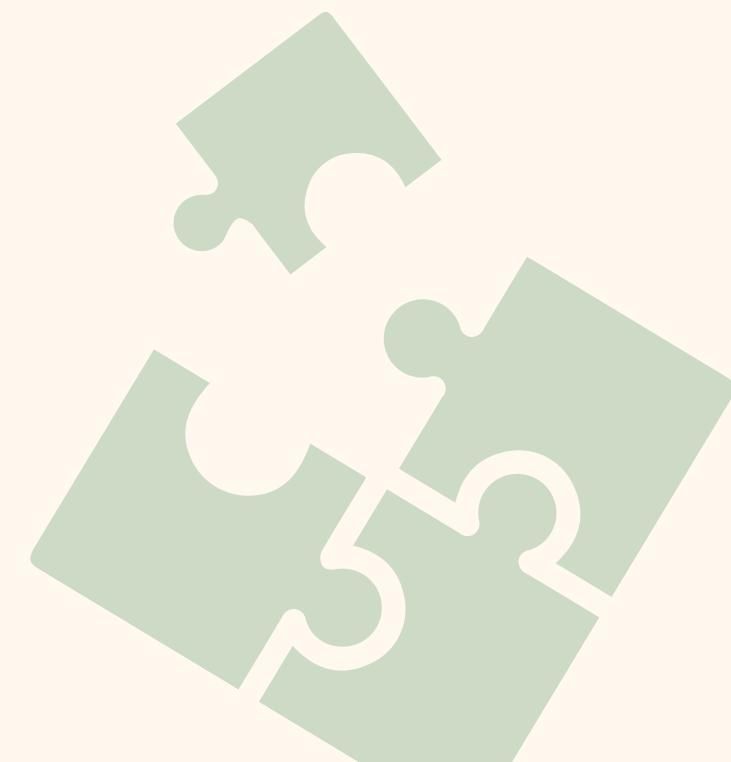
Tạo cấu trúc dữ liệu chỉ mục



Sử dụng chỉ mục trong truy vấn



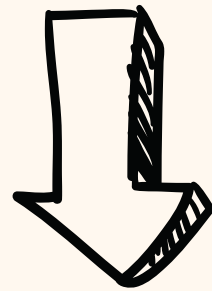
Ánh xạ giá trị đến document



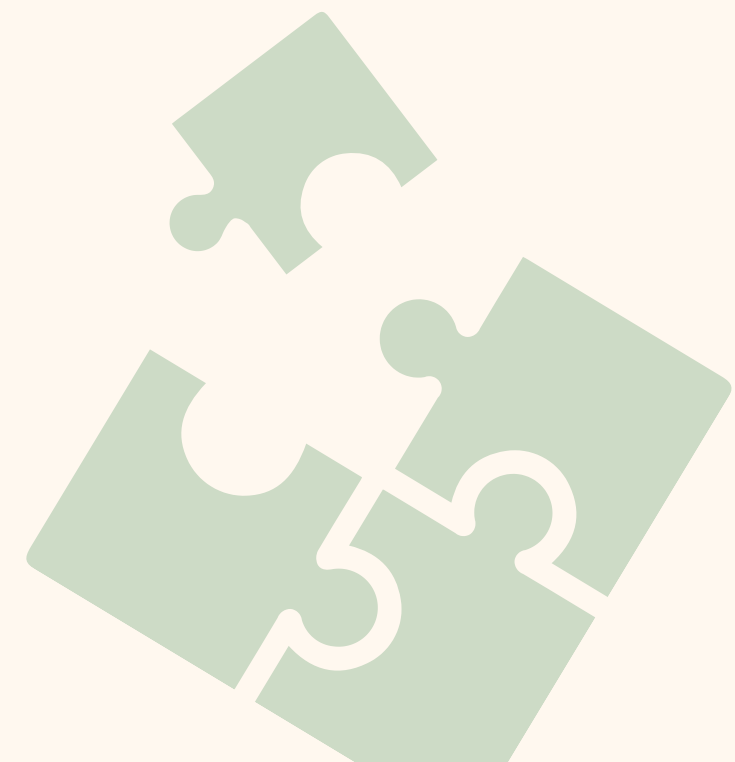
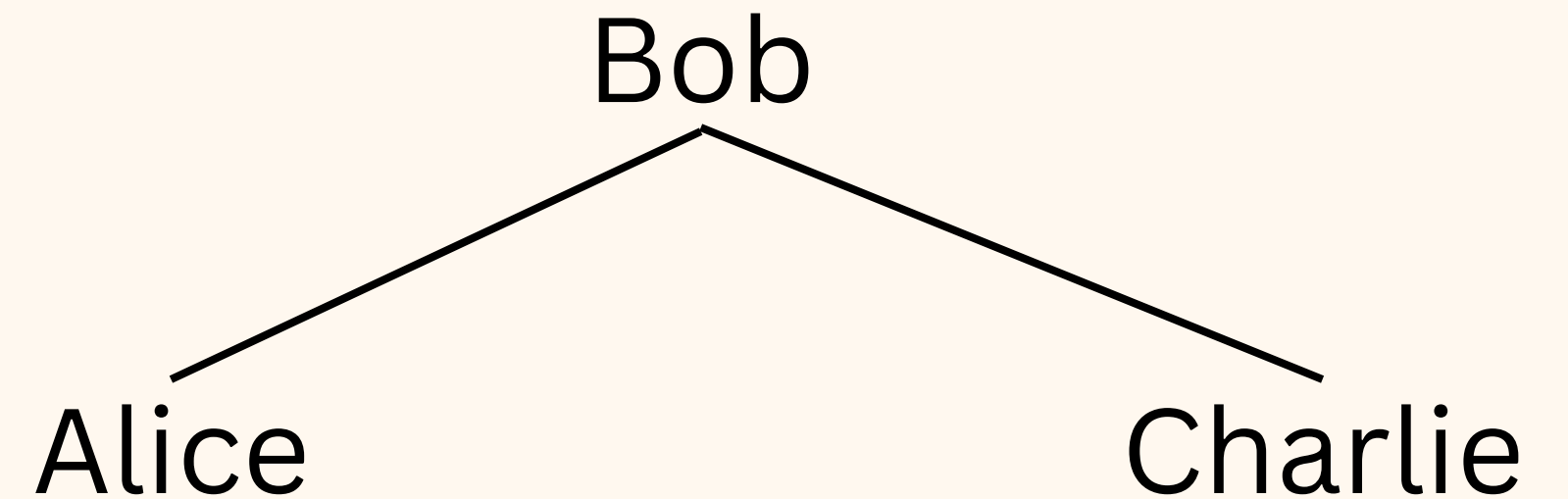
INDEX

🔍 2. Cách thức hoạt động ✕

```
db.students.insertMany([  
  { _id: 1, name: "Alice", age: 24 },  
  { _id: 2, name: "Bob", age: 27 },  
  { _id: 3, name: "Charlie", age: 22 } ])
```



```
db.students.createIndex({ name: 1 })
```



INDEX

Ưu điểm: Chỉ mục giúp tăng tốc độ truy vấn và hiệu năng.

Nhược điểm: Mỗi chỉ mục không chỉ chiếm dụng một phần không gian bộ nhớ mà còn tốn bộ nhớ mỗi khi thực hiện thêm, sửa và xóa dữ liệu trong database.



3. Đặc điểm



Sử dụng chỉ mục trong các trường hợp sau:

- Truy vấn thường xuyên trên một hoặc nhiều trường cụ thể.
- Dữ liệu lớn và việc quét toàn bộ collection trở nên chậm chạp.
- Truy vấn cần sắp xếp hoặc lọc nhiều dữ liệu.

INDEX



3. Đặc điểm



Tạo chỉ mục

```
db.COLLECTION_NAME.createIndex({KEY:1})
```

Xem chỉ mục hiện có

```
db.COLLECTION_NAME.getIndexes()
```

Xoá chỉ mục

```
db.COLLECTION_NAME.dropIndex({KEY:1})
```

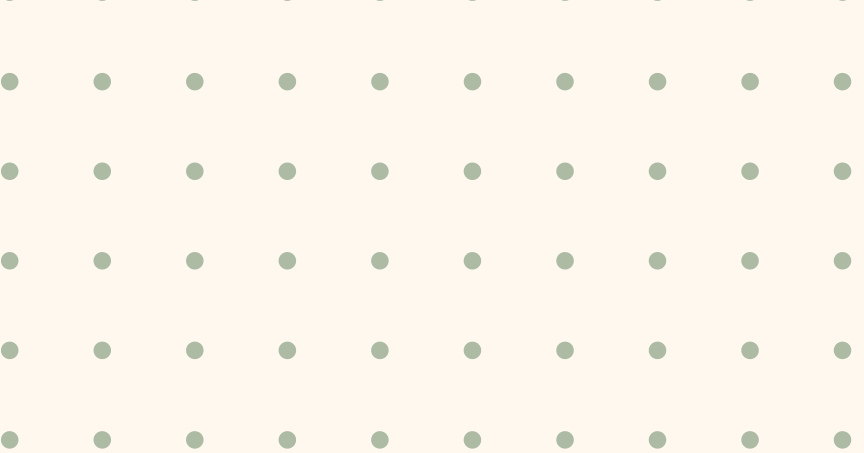
```
db.COLLECTION_NAME.dropIndex()
```

Truy vấn bằng index

```
db.users.find({username: 'user112'})
```

```
db.users.find({gender: 'female', age: 29})
```





4. HIỆU NĂNG



HIỆU NĂNG TRUY VẤN

MongoDB:

Hỗ trợ truy vấn nhanh hơn khi làm việc với dữ liệu phi cấu trúc hoặc dữ liệu lớn nhờ sharding và caching tích hợp nên MongoDB nhanh hơn khi truy vấn các tập dữ liệu lớn hoặc không đồng nhất.

MySQL:

Hiệu năng cao nếu dữ liệu đã được tối ưu hóa và đánh chỉ mục tốt. Tuy nhiên, có thể bị chậm khi xử lý dữ liệu phức tạp hoặc phi cấu trúc. MySQL nhanh hơn với các truy vấn phức tạp đòi hỏi mối quan hệ giữa các bảng.

HIỆU NĂNG GHI XUẤT DỮ LIỆU

MongoDB:

Xử lý ghi dữ liệu nhanh hơn nhờ mô hình không có giao dịch chặt chẽ trong các trường hợp không cần đảm bảo tính nhất quán cao và phù hợp với các hệ thống ghi dữ liệu theo thời gian thực

MySQL:

Hiệu quả hơn khi cần quản lý giao dịch tài chính hoặc dữ liệu quan trọng. Nhược điểm là ghi dữ liệu chậm hơn do phải đảm bảo tính toàn vẹn và nhất quán ACID trong các giao dịch.

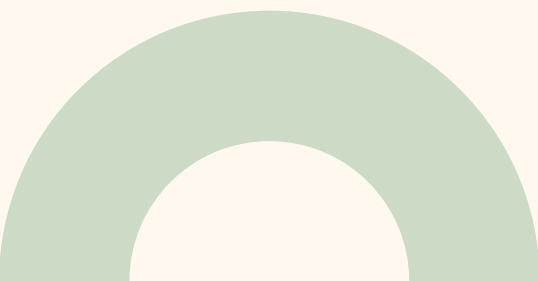
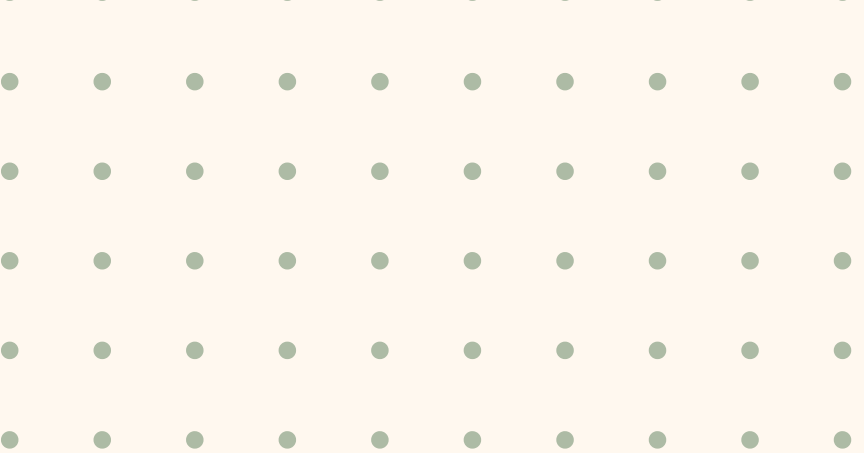
HIỆU NĂNG ĐỌC DỮ LIỆU

MongoDB:

Đọc nhanh hơn với các trường hợp không cần xử lý mối quan hệ phức tạp, đặc biệt khi dữ liệu đã được tối ưu hóa với các index.

MySQL:

Đọc tốt hơn với các truy vấn phức tạp liên quan đến nhiều bảng (JOIN, GROUP BY, ORDER BY). Với hiệu năng cao nếu dữ liệu được chuẩn hóa và đánh chỉ mục hợp lý.



5. KẾT LUẬN

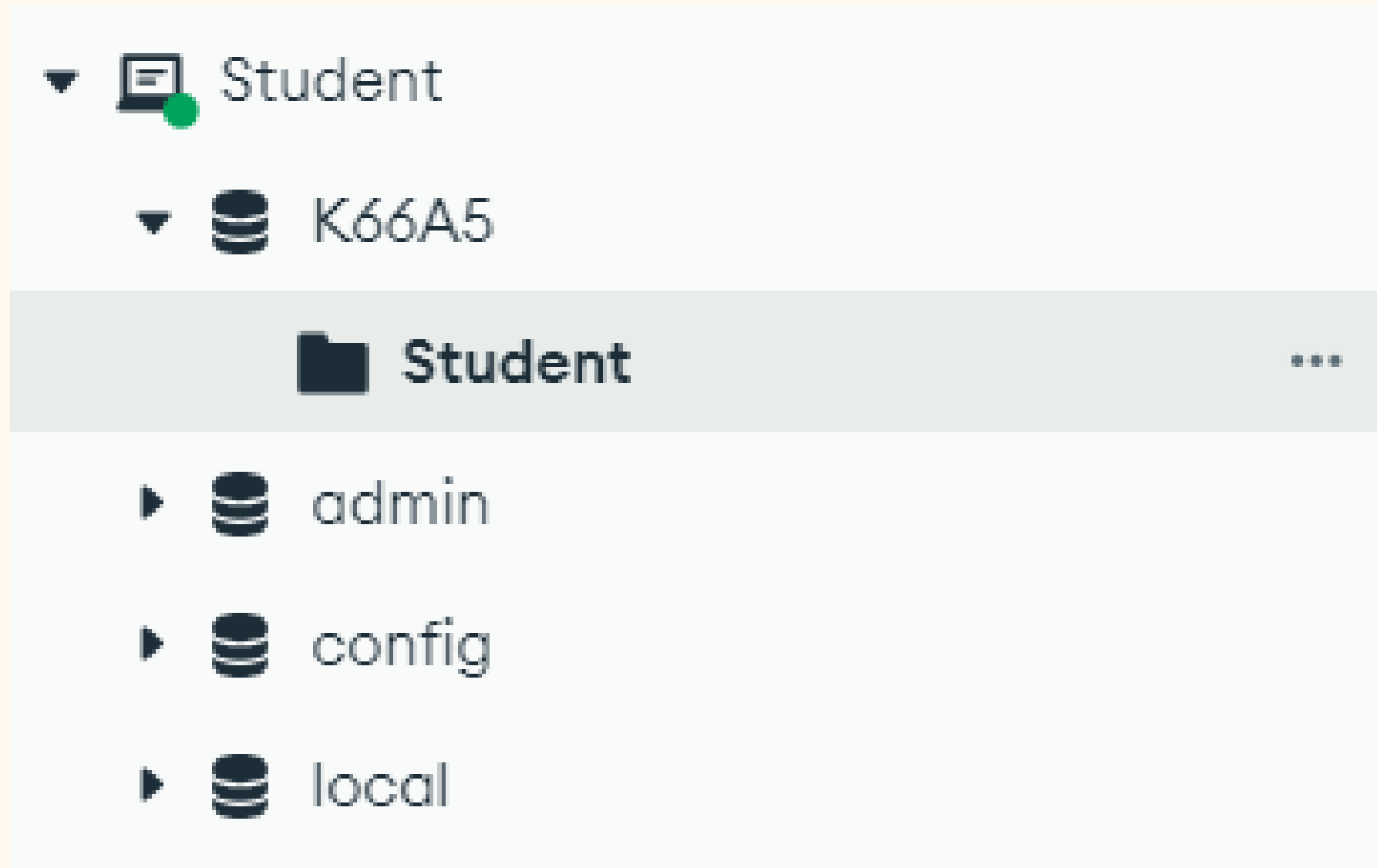


KẾT LUẬN

- Nhờ thiết kế tối ưu, MongoDB mang đến hiệu suất cao trong việc truy xuất dữ liệu và thao tác nhanh chóng. Bên cạnh đó, giao diện thân thiện cùng cấu trúc câu lệnh đơn giản, dễ hiểu, giúp người dùng dễ dàng tiếp cận và sử dụng
- Việc lựa chọn hệ quản trị nào phụ thuộc vào tính chất của ứng dụng: liệu chúng ta cần hiệu suất tốt hơn cho dữ liệu phi cấu trúc và tốc độ ghi, hay cần khả năng xử lý các giao dịch phức tạp với yêu cầu tính nhất quán cao.



VÍ DỤ MINH HOẠ - MONGODB



Tạo Collections có tên là Student



Tạo các Document

Insert Document

To collection K66A5.Student

```
1  ▼ {  
2      "_id": 1,  
3      "name": "John Doe",  
4      "age": 30,  
5      "email": "john.doe@example.com"  
6  }
```

Tạo các Document mà không cùng
cấu trúc với các Documents có sẵn

VÍ DỤ MINH HOẠ - MONGODB

Document 1

```
{  
  "_id": 1,  
  "name": "John Doe",  
  "age": 30,  
  "email": "john.doe@example.com"  
}
```

Document 2

```
{  
  "_id": 2,  
  "name": "Jane Smith",  
  "is_active": true  
}
```

Các document khác nhau trong cùng một collection có thể chứa các trường (fields) khác nhau với các kiểu dữ liệu khác nhau

Options ►



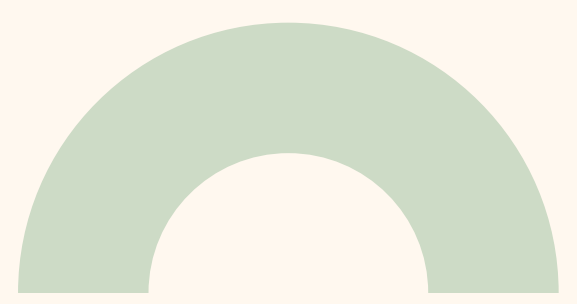
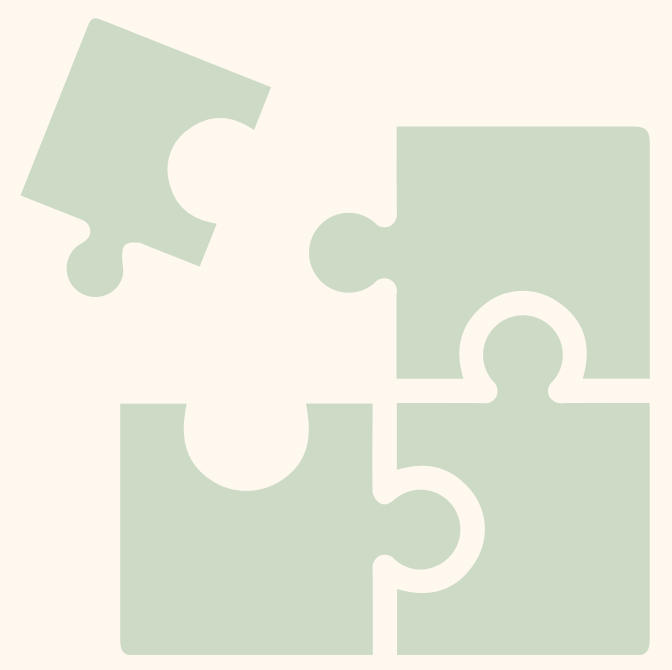
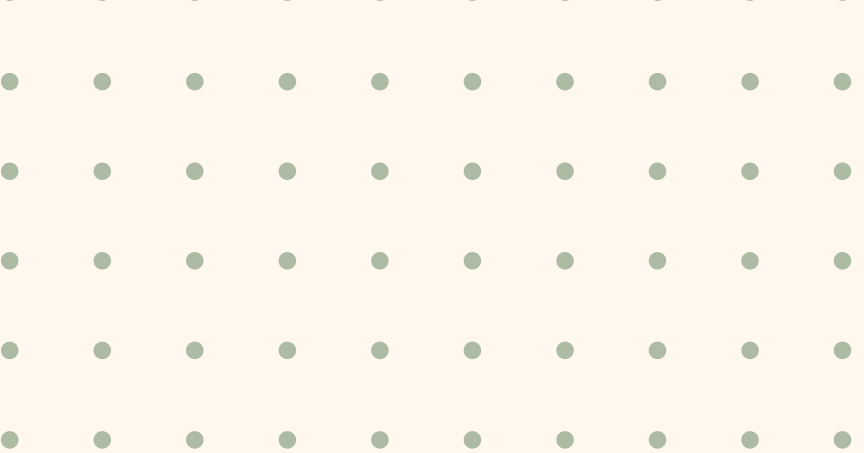
```
_id: 1
name : "John Doe"
age : 30
email : "john.doe@example.com"
```

VÍ DỤ MINH HOẠ - SQL

```
CREATE TABLE students (  
  student_id VARCHAR(10),  
  name VARCHAR(50),  
  age INT,  
  subjects JSON,  
  graduated BOOLEAN  
);
```

```
INSERT INTO students (student_id, name, age, subjects, graduated)  
VALUES ('A001', 'Nguyen Van A', 20, '["Math", "Physics"]', false);
```

Phải tạo một bảng với cấu trúc cố định trước, và các bản ghi cần tuân thủ theo cấu trúc đó



THANK YOU

