MỤC LỤC

1. Tổng quan Mongodb 2
a) Các khái niệm 2
b) Lợi Thế Của MongoDB so với RDBMS 4
c) Các thao tác 4
2. Mô Hình Replica Set 6
a) Cấu Trúc của Replica Set 6
b) Lợi Ích của Replica Set 7
c) Quá Trình hoạt động của Replica Set 7

TÌM HIỂU MONGODB

1. Tổng quan Mongodb

MongoDB là một loại cơ sở dữ liệu linh hoạt, cho phép người dùng lưu trữ dữ liệu theo cách đơn giản và hiệu quả. Thay vì sử dụng bảng như trong các cơ sở dữ liệu truyền thống, MongoDB sử dụng các khái niệm như Collection và Document, dùng để quản trị cơ sở dữ liệu NoSQL.

> NoSQL (Not only SQL) được sử dụng thay thế cho cơ sở dữ liệu quan hệ (Relational Database – RDB) truyền thống. Cơ sở dữ liệu NoSQL khá hữu ích trong khi làm việc với các tập dữ liệu phân tán lớn. MongoDB là một công cụ có thể quản lý thông tin hướng document cũng như lưu trữ hoặc truy xuất thông tin.
>
> Trong khi đó, ngôn ngữ truy vấn có cấu trúc (SQL) là ngôn ngữ lập trình được tiêu chuẩn hóa, dùng để quản lý cơ sở dữ liệu quan hệ. Dữ liệu được chuẩn hóa SQL dưới dạng schema và table và mọi table đều có cấu trúc cố định.

a) Các khái niệm

Khái niệm Database

Database là nơi lưu trữ các Collection. Mỗi cơ sở dữ liệu sẽ có một tập hợp các tệp riêng biệt trên máy chủ. Một máy chủ MongoDB có thể chứa nhiều cơ sở dữ liệu khác nhau.

Khái niệm Collection

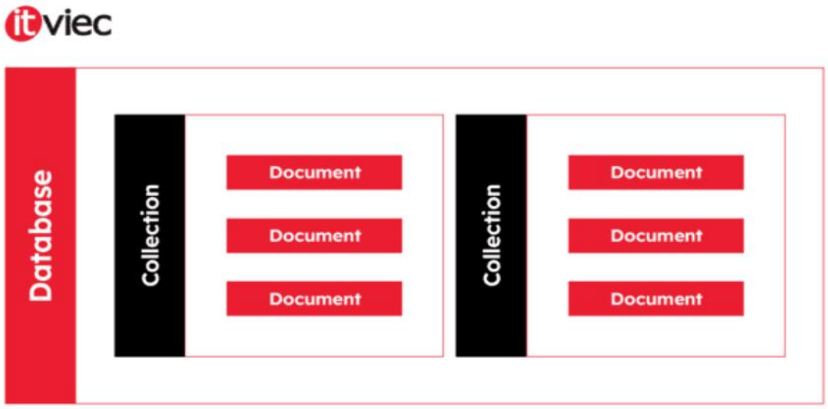
Collection là nhóm các Document. Tương tự như một bảng trong các hệ quản trị cơ sở dữ liệu khác. Một Collection thuộc về một cơ sở dữ liệu duy nhất và không có ràng buộc quan hệ như các hệ thống truyền thống, giúp việc truy xuất dữ liệu diễn ra nhanh chóng. Điều đặc biệt là trong một Collection, có thể lưu trữ nhiều loại dữ liệu khác nhau, không giống như các bảng cố định trong MySQL. Các Document trong một Collection có thể có các trường khác nhau, nhưng thường có điểm chung hoặc liên quan đến nhau.

Khái niệm Document

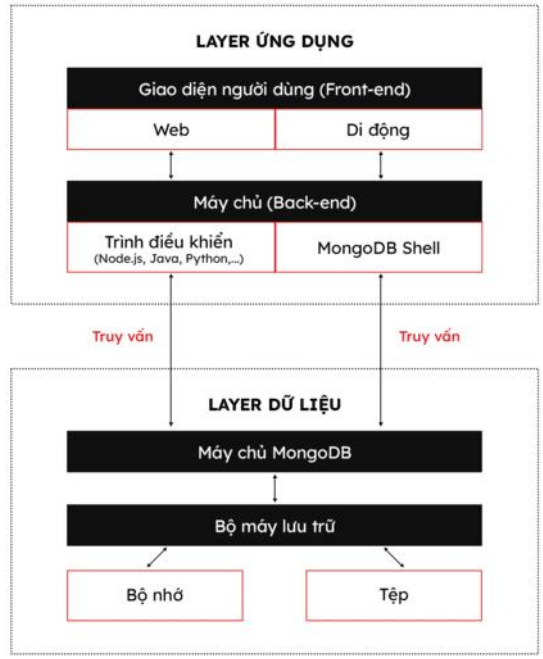
Document là đơn vị cơ bản trong MongoDB, có cấu trúc tương tự như JSON. Nó được tạo thành từ các cặp key-value. Document trong cùng một Collection không cần phải giống nhau về cấu trúc hay các trường dữ liệu. Điều này có nghĩa là mỗi Document có thể có các trường khác nhau và kiểu dữ liệu khác nhau.

RDBMS	MongoDB
Database	Database
Table	Collection
Row	Document
Column	Field

Bảng 1: Mối liên hệ giữa RDBMS và MongoDB



Hình 1: Mô hình Mongo lưu trữ dữ liệu



Hình 2: Mô hình hoạt động của MongoDB

b) Lợi Thế Của MongoDB so với RDBMS

- **Ít yêu cầu về cấu trúc (Schema):** MongoDB là cơ sở dữ liệu dựa trên Document, cho phép lưu trữ nhiều loại Document khác nhau trong cùng một Collection. Các Document có thể có số lượng trường, nội dung và kích thước khác nhau, mang lại sự linh hoạt cao.
- **Cấu trúc rõ ràng:** Mỗi Document có cấu trúc rõ ràng và dễ hiểu, giúp việc làm việc với dữ liệu trở nên đơn giản hơn.
- **Không cần Join phức tạp**: Trong MongoDB, không cần thực hiện các phép Join phức tạp như trong các hệ quản trị cơ sở dữ liệu quan hệ. Điều này giúp đơn giản hóa quy trình truy vấn dữ liệu.
- **Khả năng truy vấn mạnh mẽ:** MongoDB hỗ trợ các truy vấn linh hoạt trên các Document thông qua một ngôn ngữ truy vấn mạnh mẽ, tương tự như SQL, giúp việc tìm kiếm và lọc dữ liệu trở nên dễ dàng.
- **Dễ dàng mở rộng:** MongoDB cho phép mở rộng hệ thống một cách đơn giản, rất phù hợp cho các ứng dụng có nhu cầu phát triển nhanh chóng.
- **Không cần ánh xạ đối tượng:** Không cần chuyển đổi hoặc ánh xạ giữa các đối tượng trong ứng dụng và các đối tượng trong cơ sở dữ liệu, tiết kiệm thời gian và công sức.
- **Tối ưu hóa hiệu suất:** MongoDB sử dụng bộ nhớ trong để lưu giữ các phần công việc, giúp truy cập dữ liệu nhanh hơn và nâng cao hiệu suất của hệ thống.

c) Các thao tác

- **Tạo Database:** Để tạo một cơ sở dữ liệu mới, sử dụng lệnh use <tên_database>. Nếu cơ sở dữ liệu chưa tồn tại, MongoDB sẽ tự động tạo khi có Document đầu tiên được chèn vào.
- **Xóa Database:** Để xóa một cơ sở dữ liệu, sử dụng lệnh db.dropDatabase(). Lệnh này sẽ xóa toàn bộ dữ liệu trong cơ sở dữ liệu đó.
- **Tạo Collection:** Sử dụng lệnh db.createCollection(<tên_collection>) để tạo một Collection mới trong cơ sở dữ liệu hiện tại.

- **Xóa Collection:** Để xóa một Collection, sử dụng lệnh db.<tên_collection>.drop(), điều này sẽ xóa toàn bộ dữ liệu trong Collection đó.
- **Chèn Document:** Sử dụng lệnh db.<tên_collection>.insertOne(<document>) hoặc insertMany(<array_of_documents>) để thêm Document vào Collection.
- **Truy vấn Document:** Để truy vấn dữ liệu, sử dụng lệnh db.<tên_collection>.find(<điều_kiện>), cho phép lọc dữ liệu dựa trên các trường cụ thể.
- **Cập nhật Document:** Sử dụng lệnh db.<tên_collection>.updateOne(<điều_kiện>, <cập_nhật>) hoặc updateMany(<điều_kiện>, <cập_nhật>) để cập nhật các Document đã tồn tại.
- **Xóa Document:** Lệnh db.<tên_collection>.deleteOne(<điều_kiện>) hoặc deleteMany(<điều_kiện>) được sử dụng để xóa Document.
- **Giới hạn bản ghi:** Sử dụng limit(<số_lượng>) để giới hạn số lượng bản ghi trả về trong truy vấn, giúp quản lý hiệu suất.
- **Sắp xếp bản ghi:** Lệnh sort(<trường>: <1 hoặc -1>) cho phép sắp xếp kết quả truy vấn theo thứ tự tăng hoặc giảm.
- **Chỉ mục:** Chỉ mục giúp tăng tốc độ truy vấn bằng cách tạo cấu trúc dữ liệu đặc biệt cho các trường cụ thể. Lệnh createIndex(<trường>) được sử dụng để tạo chỉ mục.
- **Aggregation** cho phép thực hiện các phép toán phức tạp trên dữ liệu, như tính toán tổng, trung bình, và nhóm dữ liệu. Lệnh aggregate() được sử dụng cho mục đích này.

2. Mô Hình Replica Set

Replica Set là một nhóm các MongoDB server làm việc cùng nhau để đảm bảo tính sẵn có và độ tin cậy của dữ liệu. Mô hình này cung cấp khả năng tự động sao chép và phục hồi dữ liệu, giúp bảo vệ dữ liệu khỏi mất mát và tăng cường hiệu suất.

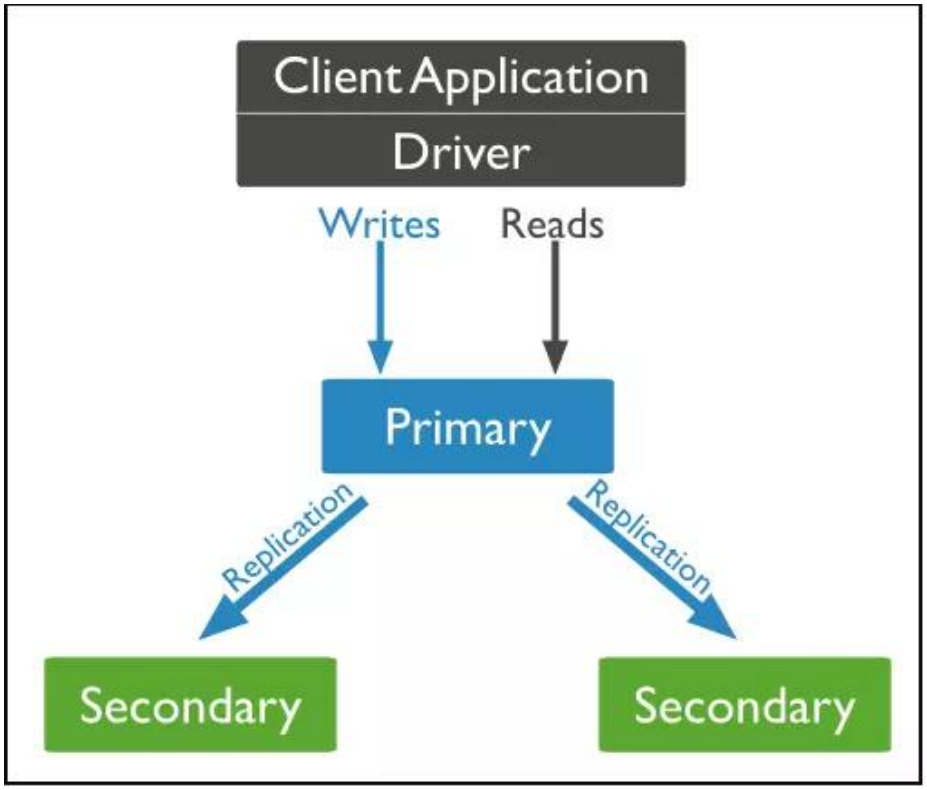
a) Cấu Trúc của Replica Set

Một Replica Set bao gồm:

Primary Node: Là node chính, nơi thực hiện các thao tác ghi dữ liệu. Tất cả các thay đổi được thực hiện trên Primary sẽ được sao chép đến các Secondary Nodes.

Secondary Nodes: Là các node phụ, sao chép dữ liệu từ Primary. Chúng có thể xử lý các truy vấn đọc, giúp giảm tải cho Primary.

Arbiter (nếu cần): Là node không chứa bản sao của dữ liệu nhưng tham gia vào quá trình bầu chọn để xác định Primary Node. Arbiter thường được sử dụng trong các Replica Set có số lượng node lẻ để duy trì tính nhất quán.



Hình 3: Mô tả mô hình Replica Set

b) Lợi Ích của Replica Set

- Tính sẵn có cao: Nếu Primary Node gặp sự cố, một Secondary Node có thể tự động trở thành Primary, đảm bảo hệ thống vẫn hoạt động liên tục.
- Khôi phục dữ liệu: Nếu có vấn đề xảy ra, dữ liệu có thể được phục hồi từ các Secondary Nodes.
- Cân bằng tải: Các truy vấn đọc có thể được phân phối giữa Primary và Secondary, giúp giảm tải cho node chính.
- Đảm bảo tính nhất quán: Replica Set cung cấp các cơ chế đảm bảo rằng tất cả các node có dữ liệu nhất quán.

c) Quá Trình hoạt động của Replica Set

Khi Primary Node không còn khả năng hoạt động, Replica Set sẽ thực hiện một quy trình bầu chọn để chọn ra một Secondary Node mới làm Primary. Quá trình này dựa trên các tiêu chí như:

- Số lượng vote từ các node.
- Tình trạng của các node (có đang hoạt động hay không).
- Thời gian hoạt động liên tục (uptime).

a. Khởi tạo: Một Replica Set bắt đầu với một node được chỉ định làm Primary, các node khác trở thành Secondary.
b. Ghi dữ liệu: Tất cả thao tác ghi diễn ra trên Primary Node. Dữ liệu được ghi vào oplog.
c. Sao chép: Các Secondary Nodes theo dõi oplog và sao chép các thay đổi để đồng bộ dữ liệu.
d. Bầu chọn: Nếu Primary gặp sự cố, các Secondary sẽ bầu chọn node mới làm Primary.
e. Truy vấn: Ghi dữ liệu chỉ trên Primary; đọc có thể từ cả Primary và Secondary để giảm tải.
f. Khôi phục: Nếu Secondary không đồng bộ, nó sẽ tự động khôi phục dữ liệu từ Primary.