MỤC LỤC

BÁO CÁO KẾT QUẢ 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.

Ưu điểm của MongoDB

- Tính linh hoạt: MongoDB là một hệ thống cơ sở dữ liệu phi quan hệ, nó cung cấp khả năng lưu trữ dữ liệu bất cứ khi nào, bất cứ nơi đâu, không cần phải tuân thủ một mô hình quan hệ cụ thể.
- Khả năng mở rộng: MongoDB có khả năng mở rộng dễ dàng, nhờ tính năng sharding cho phép phân chia dữ liệu thành nhiều phần và lưu trữ trên nhiều máy chủ.
- Tốc độ truy xuất nhanh: MongoDB có thể đáp ứng các yêu cầu truy vấn dữ liệu trong thời gian ngắn hơn so với các hệ thống cơ sở dữ liệu quan hệ truyền thống.
- Tính khả dụng cao: MongoDB cung cấp tính năng sao lưu và phục hồi dữ liệu, giúp người dùng bảo vệ dữ liệu của mình khỏi những rủi ro.
- Dễ sử dụng: MongoDB cung cấp các công cụ quản lý dữ liệu trực quan và dễ sử dụng, giúp người dùng tối ưu hóa hiệu suất và quản lý cơ sở dữ liệu một cách dễ dàng.
- Dễ dàng tích hợp với Big Data Hadoop

Nhược điểm của MongoDB

- Cần sử dụng bộ nhớ cao để lưu trữ dữ liệu (data storage).
- Không được phép lưu trữ hơn 16MB data trong tài liệu.
- Data nesting trong BSON cũng bị hạn chế, bạn không được phép nest data quá 100 cấp độ.

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

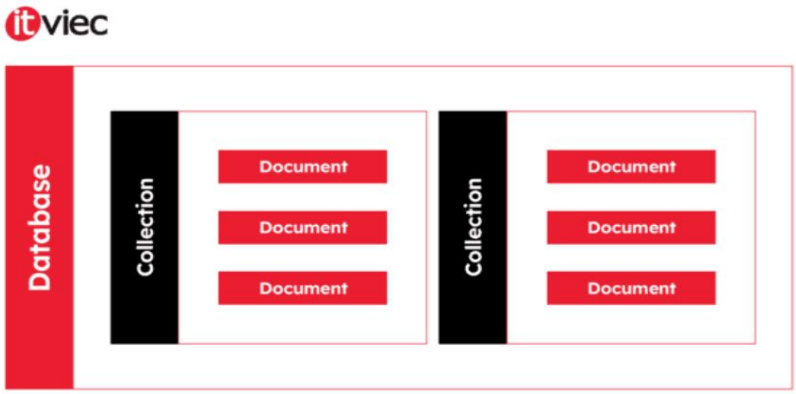
Collection là nhóm các Document. Tương tự như một bảng trong các hệ quản trị cơ sở dữ liệu khác. Một Collection thuộc về một cơ sở dữ liệu duy nhất và không có ràng buộc quan hệ như các hệ thống truyền thống, giúp việc truy xuất dữ liệu diễn ra nhanh chóng. Điều đặc biệt là trong một Collection, có thể 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

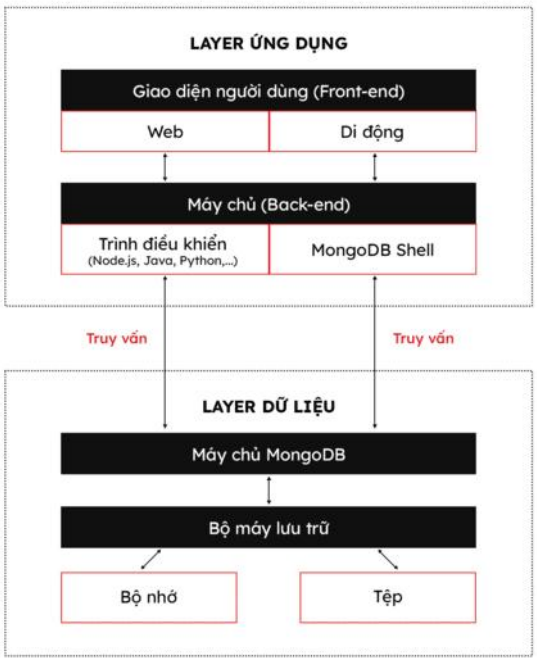
Document là đơn vị cơ bản trong MongoDB, có cấu trúc tương tự như JSON. Nó được tạo thành từ các cặp key-value. Document trong cùng một Collection không cần phải giống nhau về cấu trúc hay các trường dữ liệu. Điều này có nghĩa là mỗi Document có thể có các trường khác nhau và kiểu dữ liệu khác nhau.

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

MongoDB có nhiều ưu điểm như khả năng lưu trữ dữ liệu phân tán, linh hoạt trong cấu trúc dữ liệu, có thể mở rộng, tốc độ truy vấn nhanh và hỗ trợ các tính năng như **indexing**, **replication**, **sharding** và **map-reduce**.

Indexing

Indexing trong MongoDB giúp tăng tốc độ truy vấn dữ liệu bằng cách tạo các chỉ mục trên các trường của document. Chỉ mục trong MongoDB hoạt động tương tự như chỉ mục trong sách, giúp việc tìm kiếm thông tin diễn ra nhanh chóng mà không cần phải quét toàn bộ bộ sưu tập (collection).

Các loại chỉ mục trong MongoDB:

- Single Field Index: Chỉ mục trên một trường duy nhất. Ví dụ, chỉ mục trên trường name:

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

(1 chỉ ra rằng chỉ mục được sắp xếp theo thứ tự tăng dần.)

- Compound Index: Chỉ mục trên nhiều trường. Ví dụ, chỉ mục trên cả name và age:

```
db.collection.createIndex({ name: 1, age: -1 })
```

(Chỉ mục trên name sắp xếp tăng dần và age sắp xếp giảm dần.)

- Multikey Index: Được sử dụng khi trường là một mảng. MongoDB tạo chỉ mục trên từng phần tử trong mảng.

- Text Index: Dành cho các trường văn bản để hỗ trợ tìm kiếm văn bản toàn văn (full-text search).

- Hashed Index: Dùng để tạo chỉ mục hash trên một trường, giúp phân tán đồng đều các giá trị. Thường được sử dụng trong Sharding.

Replication

Replication là tính năng giúp sao chép dữ liệu từ một MongoDB node (Primary) sang nhiều node khác (Secondary). Mục đích chính của replication là cung cấp khả năng dự phòng, tính sẵn sàng cao và phục hồi sau thảm họa.

Các thành phần của Replication:

- Primary: Là node chính, nơi tất cả các thao tác ghi (write operations) xảy ra.

- Secondary: Sao chép dữ liệu từ Primary qua Oplog (Operation Log) và cập nhật dữ liệu theo thời gian thực. Secondary có thể xử lý các yêu cầu đọc nếu được cấu hình.

- Arbiter: Một node đặc biệt chỉ tham gia vào quá trình bầu chọn Primary khi có sự cố, nhưng không lưu trữ dữ liệu.

Lợi ích của Replication:

- Tính dự phòng (Fault tolerance): Nếu Primary gặp sự cố, một trong các Secondary sẽ được bầu làm Primary mới để tiếp tục xử lý các yêu cầu.

- Tăng tính sẵn sàng (High availability): Dữ liệu luôn có sẵn, ngay cả khi một hoặc nhiều node bị lỗi.

- Cân bằng tải cho đọc (Read scaling): Có thể cấu hình cho phép các Secondary xử lý các yêu cầu đọc, giúp giảm tải cho Primary.

Sharding

Sharding là kỹ thuật giúp chia nhỏ dữ liệu (shards) và phân tán trên nhiều máy chủ (server) để tăng cường khả năng mở rộng (scalability) của cơ sở dữ liệu. Sharding được sử dụng để mở rộng hệ thống khi dữ liệu quá lớn và không thể lưu trữ hoặc xử lý trên một máy chủ duy nhất.

Các thành phần của Sharding:

- Shards: Mỗi shard lưu trữ một phần dữ liệu của hệ thống. Một shard có thể là một Replica Set để tăng tính sẵn sàng và dự phòng.

- Mongos: Là router chịu trách nhiệm định tuyến các yêu cầu từ ứng dụng đến shard phù hợp.

- Config Servers: Lưu trữ metadata và thông tin cấu hình về sharded cluster.

Cách hoạt động của Sharding: Dữ liệu được phân mảnh dựa trên Sharding Key (một trường hoặc tập hợp các trường) và được phân phối trên nhiều shards khác nhau. Khi một truy vấn được gửi đến, Mongos sẽ định tuyến truy vấn đến shard chứa dữ liệu tương ứng.

Ví dụ:

```
db.collection.createIndex({ user_id: "hashed" })
```

Trong trường hợp này, dữ liệu sẽ được chia theo giá trị hash của user_id và phân phối trên các shards.

Map-Reduce

Map-Reduce là một mô hình lập trình được MongoDB hỗ trợ, sử dụng để xử lý và tổng hợp dữ liệu lớn bằng cách chia nhỏ các tác vụ phức tạp thành hai bước: Map và Reduce.

Cách hoạt động của Map-Reduce:

- Map: Lấy dữ liệu đầu vào và ánh xạ (map) nó thành một tập hợp các cặp khóa-giá trị.

- Hàm map xử lý từng document và trả về một cặp khóa-giá trị.

Ví dụ: Trong bài toán tính tổng doanh thu theo tháng, map sẽ nhóm doanh thu theo từng tháng.

- Reduce: Sau khi tất cả các cặp khóa-giá trị được tạo, hàm reduce được áp dụng để tổng hợp và giảm các giá trị tương ứng với cùng một khóa.

Hàm reduce tổng hợp dữ liệu theo các cặp khóa-giá trị từ bước map.

Ví dụ về Map-Reduce trong MongoDB**:**

```
db.orders.mapReduce(
   function() {
      emit(this.customerId, this.total);
   },
   function(key, values) {
      return Array.sum(values);
```

```
    },
    {
      out: "total_orders_by_customer"
    }
  )
```

Hàm trên sẽ tính tổng giá trị đơn hàng (total) của mỗi khách hàng (customerId).

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. Các mô hình triển khai và quản lý hệ thống của MongoDB

a) Mô hình Standalone

Đây là mô hình đơn giản nhất, chỉ sử dụng một instance MongoDB duy nhất.

Ưu điểm: Dễ cài đặt, quản lý.

Nhược điểm: Không có dự phòng (fault tolerance) và không thể mở rộng (scalability). Phù hợp cho các ứng dụng nhỏ hoặc trong môi trường phát triển (development).

b) Mô hình Replica Set

Một Replica Set bao gồm một Primary Node và nhiều Secondary Nodes. Primary xử lý các thao tác ghi và đồng bộ với các Secondary thông qua Oplog.

- Primary: Chịu trách nhiệm xử lý mọi yêu cầu ghi và đồng bộ với các Secondary.
- Secondary: Sao chép dữ liệu từ Primary và có thể tham gia xử lý yêu cầu đọc.
- Arbiter: Chỉ tham gia bỏ phiếu để giúp lựa chọn Primary mới khi cần, nhưng không lưu trữ dữ liệu.

Ưu điểm: Cung cấp tính sẵn sàng cao (high availability), khi Primary gặp sự cố, hệ thống sẽ tự động chuyển một Secondary thành Primary.

Nhược điểm: Tăng chi phí hạ tầng vì cần nhiều máy chủ. Khi dữ liệu quá lớn, Replica Set không đủ khả năng mở rộng lưu trữ. Giới hạn khả năng ghi trên node Primary, giảm hiệu suất ghi khi hệ thống phải xử lý nhiều thao tác đồng thời. Phù hợp cho các ứng dụng cần tính dự phòng và đảm bảo tính sẵn sàng cao.

c) Mô hình Sharding

Sharding là phương pháp chia nhỏ dữ liệu trên nhiều máy chủ (shards) để tăng cường khả năng mở rộng theo chiều ngang (horizontal scaling).

Một Sharded Cluster bao gồm:

- Shard: Lưu trữ một phần dữ liệu của hệ thống.
- Config Server: Lưu thông tin về metadata và cấu hình của cluster.
- Mongos Router: Làm nhiệm vụ định tuyến truy vấn đến đúng shard.

Ưu điểm: Cung cấp khả năng mở rộng lớn cho các hệ thống với lượng dữ liệu cực lớn và lưu trữ phân tán.

Nhược điểm: Triển khai phức tạp và yêu cầu quản lý cẩn thận.

Phù hợp với các ứng dụng có khối lượng dữ liệu lớn và yêu cầu mở rộng linh hoạt.

d) Mô hình Hybrid

Đây là mô hình kết hợp giữa Replica Set và Sharding. Mỗi shard trong Sharded Cluster có thể là một Replica Set để vừa đảm bảo tính mở rộng, vừa có dự phòng.

Ưu điểm: Cung cấp cả khả năng mở rộng và dự phòng.

Nhược điểm: Cấu hình phức tạp, đòi hỏi chi phí cao để duy trì nhiều máy chủ. Phù hợp với các ứng dụng lớn, đòi hỏi tính sẵn sàng, khả năng mở rộng cực cao.

e) So sánh các mô hình triển khai

	Standalone	**Replica Set**	**Sharded Cluster**	**Hybrid** (Replica Set + Sharded Cluster)
Số lượng máy chủ	1	3 hoặc nhiều hơn	2 hoặc nhiều shard, mỗi shard ít nhất 1 máy	2 hoặc nhiều shard, mỗi shard là một Replica Set
Tính năng	- Chỉ có một instance MongoDB	- Sao chép dữ liệu giữa các node - Đảm bảo tính sẵn sàng và dự phòng	- Phân mảnh dữ liệu trên nhiều server để mở rộng	- Phân mảnh dữ liệu và sao chép để đảm bảo tính sẵn sàng, dự phòng và mở rộng
Mở rộng theo chiều ngang	Không	Có giới hạn	Cao	Rất cao
Tính dự phòng (Failover)	Không	Có (nếu Primary gặp sự cố , một Secondary sẽ được bầu làm Primary)	Không (tuy nhiên, các shard có thể được cấu hình với Replica Set để có dự phòng)	Rất cao (Sharding kết hợp Replica Set)
Khả năng chịu lỗi	Không	Cao (do sao chép dữ liệu giữa các node)	Trung bình (nếu một shard gặp sự cố, chỉ mất dữ liệu của shard đó)	Rất cao (dữ liệu được sao chép trong từng shard)
Hiệu suất truy vấn	Cao với dữ liệu nhỏ	Trung bình đến cao (có thể dùng Secondary	Cao với dữ liệu lớn nhờ phân mảnh	Rất cao, vừa mở rộng vừa đảm bảo tính dự phòng

		để cân bằng tải cho đọc)	dữ liệu	
Sử dụng cho môi trường	Phát triển thử nghiệm	Sản xuất với yêu cầu sẵn sàng cao	Ứng dụng có lượng dữ liệu lớn và cần mở rộng theo chiều ngang	Ứng dụng có quy mô lớn, yêu cầu cao về cả mở rộng và dự phòng
Quản lý phức tạp	Đơn giản	Trung bình	Phức tạp (cần định cấu hình sharding và quản lý nhiều shard)	Rất phức tạp (kết hợp quản lý sharding và replica set)
Yêu cầu phần cứng	Thấp	Trung bình	Cao	Rất cao

Bảng 2: So sánh giữa các mô hình triển khai

3. 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.

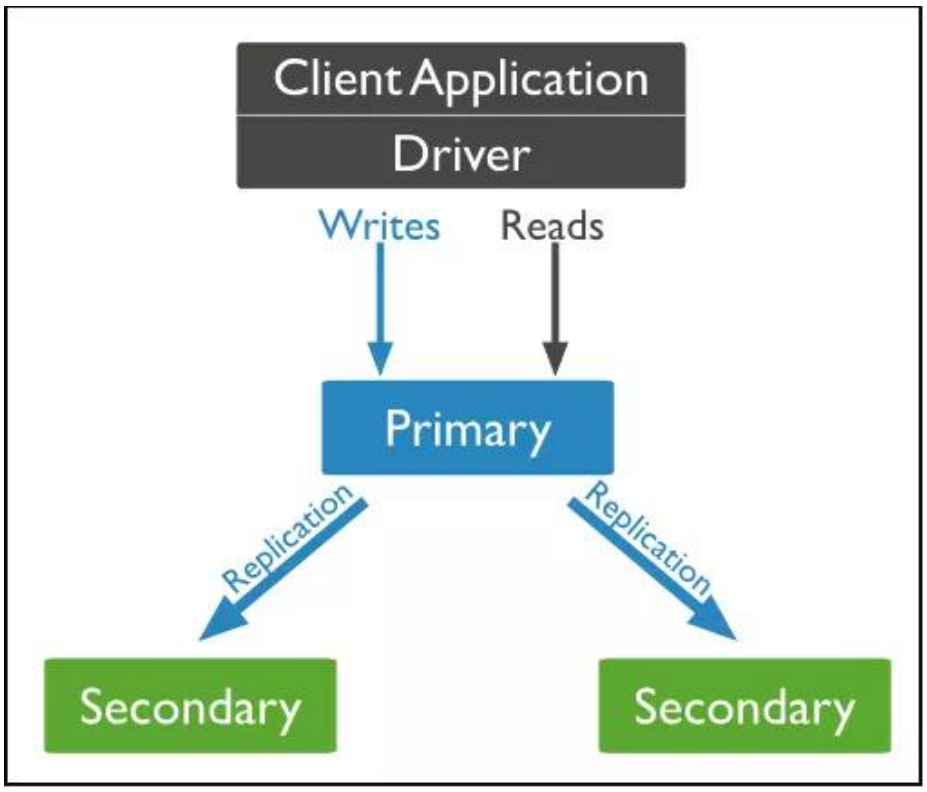
a) Cấu trúc của Replica Set

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 và tham gia vào quá trình bầu chọn nếu Primary gặp sự cố.

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

Ngoài ra, các thành viên trong Replica Set chia thành hai phần:

- Voting members: Các node tham gia bầu chọn (tối đa 7 node có quyền biểu quyết).

- Non-voting members: Các node không tham gia bầu chọn nhưng vẫn sao chép dữ liệu từ Primary dùng để dự phòng hoặc cân bằng tải.

b) Lợi ích từ tính năng của Replica Set

- Tự động failover: khi Primary gặp sự cố, Replica Set tự động bầu chọn một Secondary làm Primary mới mà không cần can thiệp thủ công.

- **Tự động khôi phục:** Khi một node gặp lỗi và quay trở lại hệ thống, nó tự động đồng bộ hóa dữ liệu với Primary mà không cần phục hồi thủ công.

- Cân bằng tải: Các truy vấn đọc có thể được phân phối giữa Primary và Secondary, giúp giảm tải cho node chính.

- Tính nhất quán dữ liệu: Replica Set hỗ trợ mô hình nhất quán Eventually Consistent, nghĩa là dữ liệu ghi vào Primary sẽ dần được sao chép sang các Secondary theo thời gian.

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).

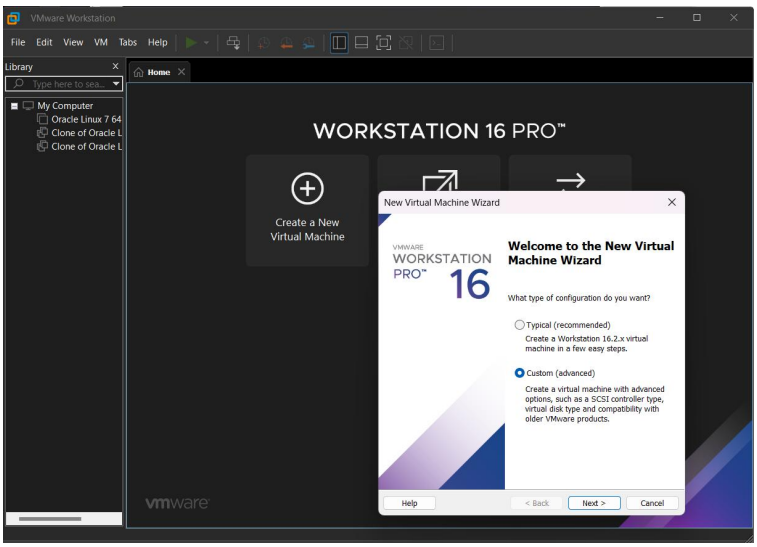
a. Khởi tạo: Một Replica Set bắt đầu với một node được chỉ định làm Primary, các node khác trở thành Secondary.
b. Ghi dữ liệu: Tất cả thao tác ghi diễn ra trên Primary Node. Dữ liệu được ghi vào oplog.
c. Sao chép: Các Secondary Nodes theo dõi oplog và sao chép các thay đổi để đồng bộ dữ liệu.
d. Bầu 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.

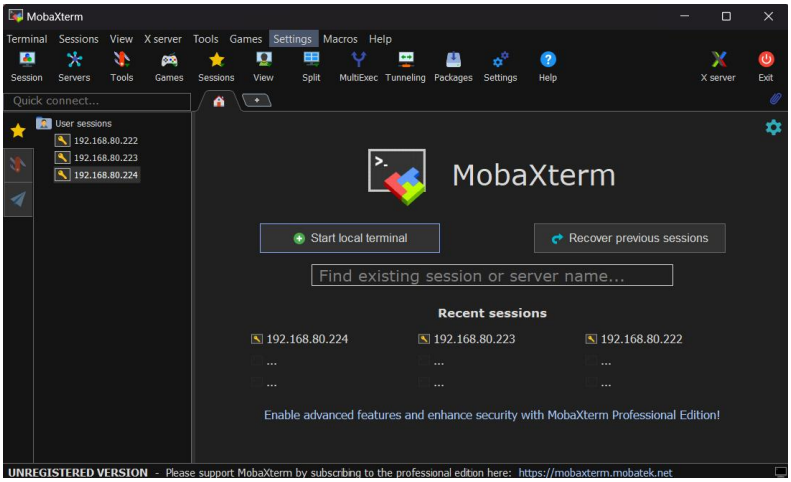
4. Cài đặt MongoDB và Replica Set

a) Môi trường cài đặt

- **Hệ điều hành: Oracle Linux 7.9**

- **Công cụ: VMware Workstation Pro, MobaXterm**





- **Phiên bản MongoDB: 7.0.14**
- **Thiết lập môi trường 3 máy ảo để cấu hình Replica Set:**
  - Máy ảo 1 (Primary): IP 192.168.80.222
  - Máy ảo 2 (Secondary): IP 192.168.80.223
  - Máy ảo 3 (Secondary): IP 192.168.80.224

Dùng tool Moba SSH để sử dụng command line của từng máy ảo.

Cấu hình ổ đĩa dữ liệu MongoDB: Đường dẫn dbPath là /data/datafile cho cả 3 máy ảo.

b) Quá trình cài đặt