NOSQL

1. **Nosql**

NoSQL (Not Only SQL) là một khái niệm tổng quát cho loại cơ sở dữ liệu không dùng mô hình dữ liệu quan hệ để quản lý cơ sở dữ liệu. Nó bao gồm 2 yếu tố chính:

*Distributed (phân tán) + non-relational (không ràng buộc)*

Các mô hình dữ liệu trong NoSQL

* + Key-value
  + Tabular
  + Document
  + Graph
  + Etc

Các đặc điểm của Nosql:

* Tính linh hoạt và không cấu trúc: Dữ liệu trong NoSQL thường không yêu cầu cấu trúc cố định như trong các cơ sở dữ liệu quan hệ. NoSQL cho phép lưu trữ dữ liệu không cấu trúc hoặc có cấu trúc linh hoạt hơn.
* Khả năng mở rộng dễ dàng: Hệ thống NoSQL có khả năng mở rộng tuyến tính hoặc ngang (horizontal scaling), giúp mở rộng dung lượng lưu trữ và khả năng xử lý dữ liệu một cách dễ dàng và linh hoạt hơn so với cơ sở dữ liệu quan hệ.
* Tính phân tán cao: Các hệ thống NoSQL thường được thiết kế để hoạt động trên nhiều nút (nodes) và có khả năng chịu lỗi cao (fault-tolerant), tức là một số nút có thể gặp sự cố nhưng hệ thống vẫn tiếp tục hoạt động.
* Hiệu suất cao: Do không áp dụng các ràng buộc của mô hình quan hệ, nên NoSQL có thể cung cấp hiệu suất cao hơn đối với một số loại ứng dụng, đặc biệt là các ứng dụng có yêu cầu về tải lớn và đồng thời.
* Khả năng lưu trữ dữ liệu lớn: NoSQL thường được sử dụng cho các ứng dụng và dự án có yêu cầu về xử lý dữ liệu lớn, phân tán và có tính chất nhất quán thấp.
* Nhất quán về mặt dữ liệu sau cùng: Tính nhất quán của dữ liệu không cần phải đảm bảo ngay tức khắc sau mỗi phép ghi. Một hệ thống phân tán chấp nhận những ảnh hưởng theo phương thức lan truyền và sau một khoảng thời gian (không phải ngay tức khắc), thay đổi sẽ đi đến mọi điểm trong hệ thống để cuối cùng dữ liệu trên hệ thống sẽ trở lại trạng thái nhất quán.

1. **Các mô hình dữ liệu**
   1. **Key-value**

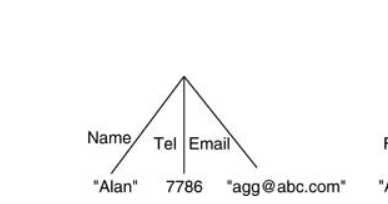
Key-Value Store là một hệ thống lưu trữ dữ liệu cơ bản, trong đó mỗi mục dữ liệu được lưu trữ dưới dạng cặp key-value. Key (khóa) là một giá trị duy nhất dùng để xác định một bản ghi cụ thể trong cơ sở dữ liệu, và value (giá trị) chứa thông tin thực sự liên quan đến key đó. Trong đó:

* Key: là 1 định danh cho duy nhất 1 mục dữ liệu, 1 key xác định duy nhất 1 value
* Value: Có thể là một chuỗi văn bản, dữ liệu nhị phân, dữ liệu có cấu trúc, và nhiều loại dữ liệu khác.

Truy Vấn Cơ Bản:

* Put (key, value):
* Get (key):
* Get\_Range (key1, key2): Trả về tất cả các value trong khoảng từ key1 đến key2.
  1. **Document**
     1. **Tổng quát**

Document databases (document stores) lưu trữ dữ liệu bán cấu trúc (hình phía dưới) và mô tả của dữ liệu trong document format. Nó cho phép lập trình viên có thể tạo và cập nhật chương trình mà không cần lược đồ tham chiếu tổng thể



* + 1. **Mô hinh dữ liệu**

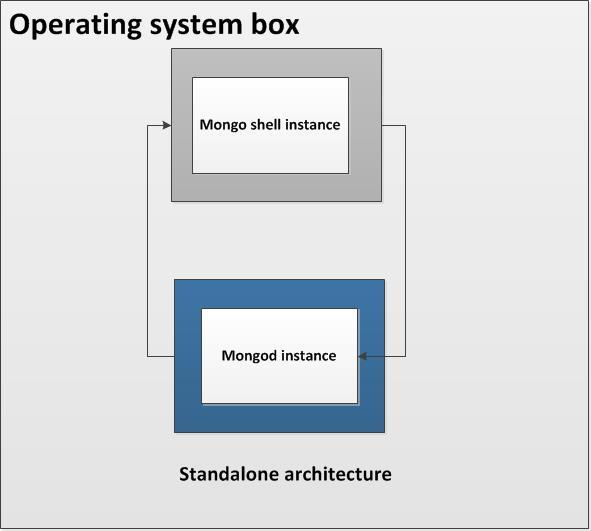
Mô hình dữ liệu chính:

* + XML (eXtensible Markup Language):
    - Tiêu Chuẩn của W3C (1998): Được phát triển như một tiêu chuẩn cho việc trao đổi dữ liệu trên web.
    - Phức Tạp và Nặng Nề: Có cấu trúc phức tạp và tải trọng lớn, thường được sử dụng cho các tài liệu có cấu trúc phức tạp.
  + JSON (JavaScript Object Notation):
    - Phát triển bởi Douglas Crockford (2005): Được tạo ra như một định dạng đơn giản và nhẹ để trao đổi dữ liệu trong JavaScript.
    - Đơn Giản và Nhẹ: Có cấu trúc đơn giản và trọng lượng nhẹ, thường được sử dụng cho việc truyền tải dữ liệu đơn giản và dễ đọc cho con người và dễ xử lý cho máy tính.

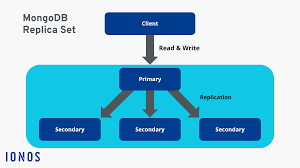
1. **Mongodb**
   1. **Giới thiệu**

MongoDB là phần mềm cơ sở dữ liệu dạng NoSQL, có mã nguồn mở hỗ trợ đa nền tảng được thiết kế theo hướng đối tượng. Các bảng (trong MongoDB gọi là collection) có cấu trúc linh hoạt cho phép tổ chức và lưu trữ dữ liệu theo một cách linh động không bắt buộc có tính cấu trúc chặt chẽ như các CSDL SQL khác.

* 1. **Kiến trúc** 
     1. **Standalone**
* Kiến trúc standalone của MongoDB thường được sử dụng trong môi trường phát triển, kiểm thử hoặc trong các ứng dụng yêu cầu một máy chủ cơ sở dữ liệu đơn giản mà không cần các tính năng như mở rộng hoặc độ tin cậy cao.
* Standalone thì DB sẽ không có tính sẵn sàng cao (High Availability) cũng như không có khả năng mở rộng (Scalable).



* + 1. **Replication**
* Với standalone mongodb, nếu server của bạn gặp sự cố thì dữ liệu của bạn sẽ bị gián đoạn, service thì không connect được còn data có thể bị mất. Vậy thì một suy nghĩ tự nhiên là sẽ phải có phương án gì để tăng tính chịu lỗi của hệ thống lên để khi 1 service chạy dịch vụ bị down thì cũng sẽ không bị mất dữ liệu. Và câu trả lời chính là sử dụng Replication.
* Ý tưởng của Replication là tạo ra một replica set, là một tập hợp của các process mongod và cùng duy trì một bộ dữ liệu. Replica set sẽ mang lại tính dư thừa (redundancy) và tính khả dụng cao (HA) cho database.



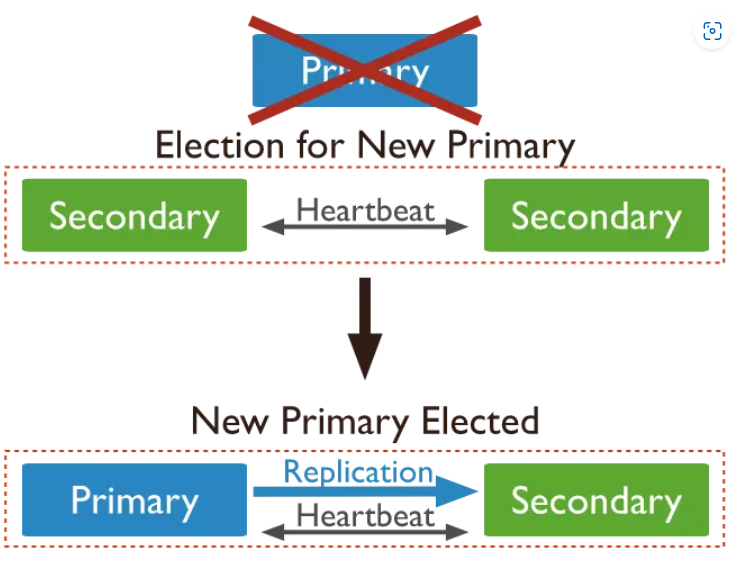
* Quá trình hoạt động thì mọi thao tác ghi dữ liệu được thực hiện vào node primary và sau đó dữ liệu được nhân bản (replicate) sang các node secondary (hay gọi là các replicas) Khi node primary bị lỗi thì trong số node secondary (replicas node) còn lại sẽ bầu ra một node mới lên làm primary để tiếp tục xử lý.

Một số lưu ý:

* Trong một replicaset chỉ có thể có 01 primary node và có thể có nhiều secondary node (là các replicas) và có thể có 01 arbiter node
* Việc ghi dữ liệu vào DB chỉ được thực hiện thông qua primary node, việc đọc dữ liệu có thể được thực hiện qua các secondary node
* Arbiter node chỉ đóng vai trò thực hiện bầu chọn khi cần chọn ra primary mới, bản thân nó không lưu dữ liệu của db
* Arbiter thì luôn là arbiter, còn primary thì có thể chuyển thành secondary và ngược lại secondary có thể chuyển thành primary
* Để đảm bảo việc bầu chọn primary thì số lượng node phải là lẻ
* Cơ chế tự động chuyển đổi

Node primary được coi là bị lỗi nên như nó không trao đổi với các node khác trong replicaset trong một khoảng thời gian nhất định, được quy định bởi tham số "**electionTimeoutMillis**" và có giá trị mặc định là 10 giây.

Một node secondary phù hợp sẽ thực hiện bầu chọn primary mới và tự đề cử nó lên làm primary mới. Cluster sẽ thực hiện bầu chọn để chọn ra primary mới và trở lại hoạt động đọc/ghi như bình thường. Các node có thể được cấu hình một trọng số nhất định để ưu tiên lên làm primary trong quá trình bầu chọn.



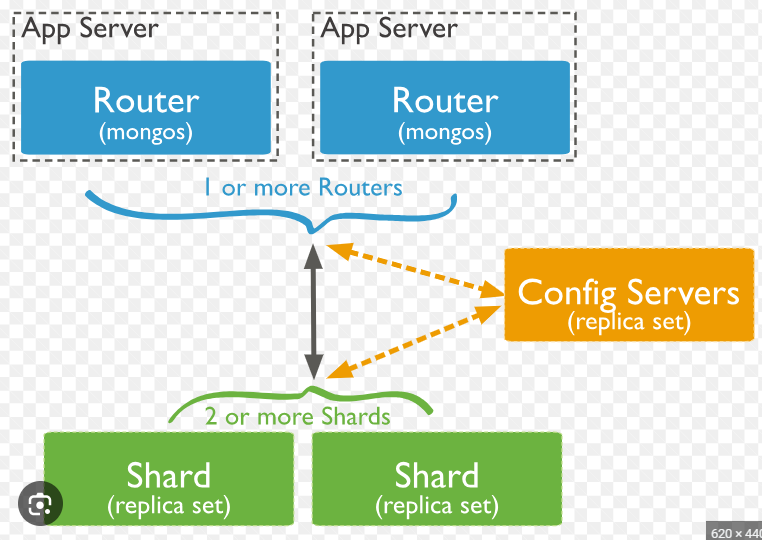
* + 1. **Sharding**

Sharding là một tiến trình lưu giữ các bản ghi dữ liệu qua nhiều thiết bị và nó là một phương pháp của MongoDB để đáp ứng yêu cầu về sự gia tăng dữ liệu. Khi kích cỡ của dữ liệu tăng lên, một thiết bị đơn không thể đủ để lưu giữ dữ liệu. Sharding giải quyết vấn đề này với việc mở rộng phạm vi theo bề ngang (horizontal scaling). Với Sharding, bạn bổ sung thêm nhiều thiết bị để hỗ trợ cho việc gia tăng dữ liệu và các yêu cầu của các hoạt động đọc và ghi.

một mongodb sử dụng sharding sẽ được gọi là một "sharded cluster”

**Một MongoDB sharded cluster bao gồm các thành phần chính sau:**

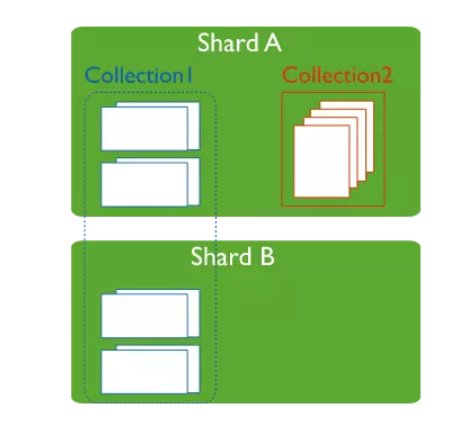
* **shard**: Mỗi shard chứa một phần của dữ liệu đã được shard. Mỗi shard này lại có thể được triển khai dưới dạng một replicaset để tăng tính dự phòng cho dữ liệu của nó quản lý.
* **mongos**: Hoạt động như một query router, là phần giao diện với các client với sharded cluster. Client sẽ chỉ cần biết kết nối tới mongos, phần còn lại là kết nối tới shard nào, replicas nào sẽ do mongos điều phối và trong suốt với client
* **config servers**: Chứa thông tin metadata và các tham số cấu hình cho cluster. Ví dụ thông tin cấu hình các shard, các replicaset.. được lưu ở config server này. Và config server cũng có thể triển khai dưới dạng replicaset.



**Shard Keys**

MongoDB sử dụng **shard key** để phân phối các bản ghi (document) của các bảng (collection) tới các shard tương ứng. Shard key có thể gồm 1 hoặc nhiều field trong bảng đó.

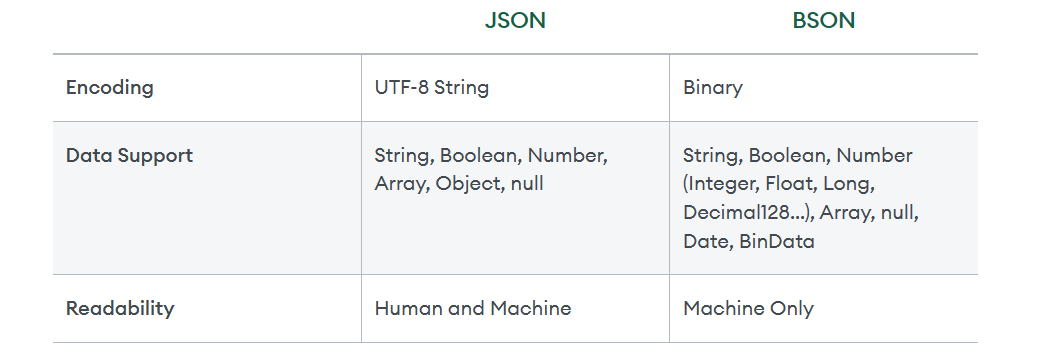
* Việc lựa chọn shard key được thực hiện ngay khi thực hiện shard một collection (bảng). Lưu ý rằng khi một collection đã được shard thì nó sẽ không thể được "unshard".
* Trong một database, ta có thể shard một số collection cụ thể chứ không nhất thiết phải shard toàn bộ collection của database đó.



* 1. **Mô hình dữ liệu**

Mô Hình Dữ Liệu và Ngôn Ngữ Truy Vấn Dựa Trên

* JSON:
* Mỗi document trong MongoDB là một tập hợp các cặp (key, giá trị) với một khóa duy nhất, được tạo ra bởi MongoDB.
* MongoDB sử dụng mô hình dữ liệu và ngôn ngữ truy vấn dựa trên JSON, giúp đơn giản hóa quá trình phát triển và tương tác với dữ liệu.
* BSON (Binary JSON):
* BSON là một phiên bản nhị phân của JSON, để lưu trữ dữ liệu.
* BSON giúp giảm kích thước của dữ liệu so với JSON, làm cho việc lưu trữ và truy xuất dữ liệu hiệu quả hơn.

****

* 1. **Trường hợp sử dụng**

**Các trường hợp sử dụng mongodb bao gồm:**

* Ứng dụng Social Networking: Các ứng dụng mạng xã hội như Facebook, Twitter, hoặc Instagram thường phải xử lý một lượng lớn dữ liệu người dùng và các hoạt động như bài đăng, bình luận, và kết bạn. MongoDB được sử dụng để lưu trữ thông tin người dùng, quản lý mối quan hệ, và tăng hiệu suất truy vấn trong các môi trường có lượng dữ liệu lớn.
* Hệ thống phân tích dữ liệu (Analytics): Các hệ thống phân tích dữ liệu thường cần phải xử lý dữ liệu lớn từ nhiều nguồn khác nhau. MongoDB cung cấp các tính năng như aggregation framework và MapReduce giúp thực hiện các phép tính và biến đổi dữ liệu phức tạp, từ đó giúp nhà phân tích dữ liệu rút ra thông tin và thông điệp quan trọng từ dữ liệu.
* Ứng dụng IoT (Internet of Things): Trong môi trường IoT, dữ liệu được tạo ra từ các thiết bị kết nối mạng và cảm biến có thể phát triển nhanh chóng và không đồng đều. MongoDB có thể được sử dụng để lưu trữ và xử lý dữ liệu từ các thiết bị IoT, bao gồm dữ liệu sensor, logs và trạng thái thiết bị, cung cấp khả năng mở rộng và linh hoạt trong việc quản lý dữ liệu.
  1. **Hạ tầng triển khai**
     1. **Cloud**
* **MongoDB Atlas** là **cloud database** của MongoDB được ra mắt vào năm 2016 chạy trên AWS, Microsoft Azure và Google Cloud Platform.
* Dữ liệu trong mỗi Cluster ở Atlas được lưu trữ theo cơ chế Replication, với 3 nodes: 1 master (primary) và 2 slaves (secondary)
  + 1. **On-premise**
* Dữ liệu sẽ được triển khai trên máy tính cá nhân hoặc doanh nghiệp