mongoDB
FOR GIANT IDEAS

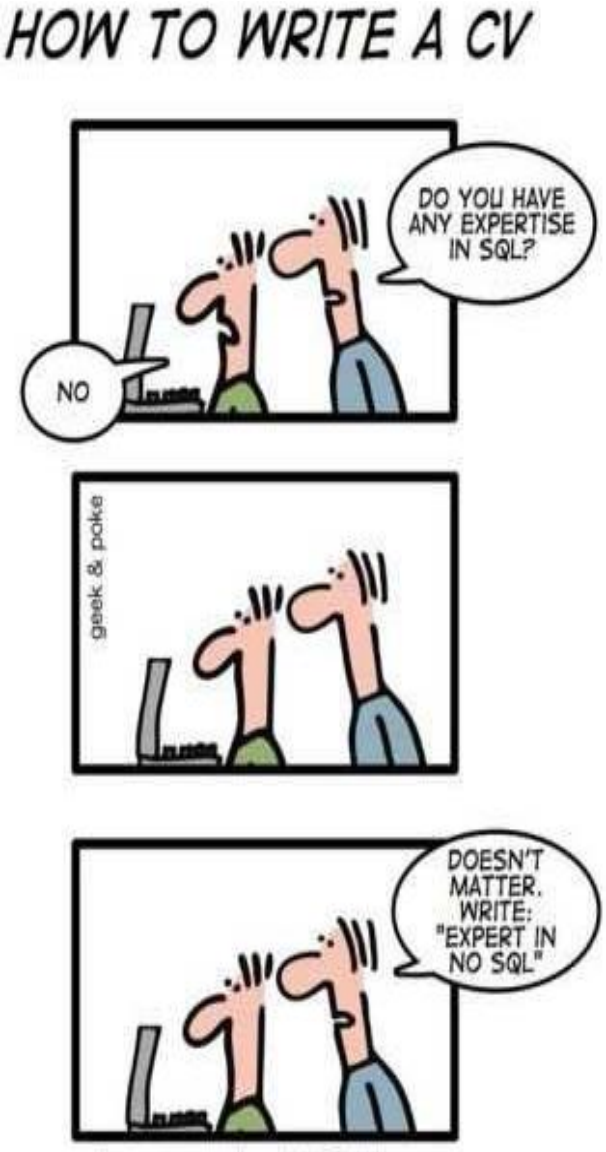
mongoDB
{ name: mongo, type: DB }

Cours de Bases de données NOSQL (MongoDB)

madaniabdellah@gmail.com

Plan du cours

- Rappels :
  - JSON
  - Bases de Données Relationnelles
- Introduction : Mouvement NoSQL
- MongoDB
  - Caractéristiques de MongoDB
  - Requêtes avec MongoDB
  - MongoDB et Java
  - MongoDB et PHP
  - Réplication et reprise sur panne dans MongoDB
  - **Sharding dans MongoDB**



Sharding dans MongoDB

madaniabdellah@gmail.com

Rappel

- Une partition d'un ensemble S est un ensemble {F1, F2, … Fn} de parties de S telles que Fi ∩Fj= {∅} et ∪Fi = S.
- Dans notre cas :
  - Les ensembles sont des collections ;
  - les éléments de l'ensemble sont les documents ;
  - les parties sont nommées fragments (shard en anglais) ;

Introduction

- Sharding est une méthode de distribution de données sur plusieurs machines.
- Les SGBD NOSQL utilisent le sharding pour prendre en charge des ensembles de données très volumineux présentant plusieurs contraintes, par exemple :
  - Les débits élevés de requête peuvent épuiser la capacité du processeur du serveur.
  - Les tailles plus grandes que la RAM du système dégradent la capacité d'E / S des unités de disque.

Introduction

- Il existe deux méthodes pour gérer la montée en charge : la mise à l'échelle verticale et horizontale.
- La mise à l'échelle verticale consiste à augmenter la capacité d'un seul serveur, par exemple en utilisant un processeur plus puissant, en ajoutant plus de RAM ou en augmentant la quantité d'espace de stockage.
- Les limites de la technologie disponible peuvent empêcher une seule machine d'être suffisamment puissante pour une charge de travail donnée.
- De plus, les fournisseurs basés sur le cloud ont des plafonds (seuils) basés sur les configurations matérielles disponibles.

Introduction

- La mise à l'échelle horizontale consiste à diviser (partitionner) les données et à les charger sur plusieurs serveurs, en ajoutant des serveurs supplémentaires pour augmenter la capacité selon les besoins.
- Alors que la vitesse ou la capacité globale d'une seule machine peut ne pas être élevée, chaque machine gère un sous-ensemble de la charge du travail globale, offrant potentiellement une meilleure efficacité qu'un seul serveur grande capacité haute vitesse.
- L'extension de la capacité du déploiement nécessite uniquement l'ajout de serveurs supplémentaires, ce qui peut représenter un coût global inférieur à celui du matériel haut de gamme pour une seule machine.
- L'inconvénient est que cette solution est plus complexe à mettre en œuvre et à déployer.
- MongoDB prend en charge la mise à l'échelle horizontale par sharding.

Le principe du sharding

- Le sharding consiste à :
  - découper une (grande) collection en fragments en fonction d'une clé ;
  - placer chaque fragment sur un serveur ;
  - Maintenir un répertoire indiquant que telle clé se trouve dans tel fragment sur tel serveur
- Le partitionnement apporte la répartition de charge (load balancing)
- S'applique à des collections de paires (clé, valeur), où valeur est n'importe quelle information structurée.
- La clé indique à quel fragment appartient un document.

Le principe du sharding

- Un système partitionné avec MongoDB comprend les composants suivants:
  - **Shard** (fragment) : chaque shard contient un sous-ensemble des données partagées. Chaque shard peut être déployé en tant que réplicaset.
  - **Mongos** : Le mongos agit comme un routeur de requête, fournissant une interface entre les applications cliente et les shards.
  - **Serveurs de configuration**: les serveurs de configuration stockent les métadonnées et les paramètres de configuration du cluster.

Le principe du sharding

- MongoDB partitionne les données en morceaux, appelés Chunks.
- Chaque chunk est défini par deux bornes : borne inférieure inluse et une borne supérieure exclue. Ces deux bornes sont basées sur la clé de partitionnement
- La taille par défaut d'un chunk dans MongoDB est de 64 mégaoctets. Vous pouvez augmenter ou réduire la taille du chunk.

Avantages de sharding

- Lectures/Ecritures

  MongoDB distribue la charge de travail en lecture et en écriture sur les fragments du cluster partitionné, permettant à chaque fragment de traiter un sous-ensemble d'opérations de cluster. Les charges de travail en lecture et en écriture peuvent être réduites en ajoutant plus de fragments.

- Capacité de stockage

  Sharding répartit les données entre les fragments du cluster, permettant à chaque fragment de contenir un sous-ensemble des données de cluster totales. À mesure que l'ensemble de données augmente, des fragments supplémentaires augmentent la capacité de stockage du cluster.

Avantages de sharding

- La haute disponibilité
  - Un cluster fragmenté peut continuer à effectuer des opérations de lecture / écriture partielles même si un ou plusieurs fragments sont indisponibles. Bien que le sous-ensemble de données sur les fragments indisponibles ne soit pas accessible pendant le temps d'arrêt, les lectures ou les écritures dirigées vers les fragments disponibles peuvent toujours réussir.
  - Dans les environnements de production, les fragments individuels doivent être déployés en tant que réplicaSet, ce qui augmente la redondance et la disponibilité.

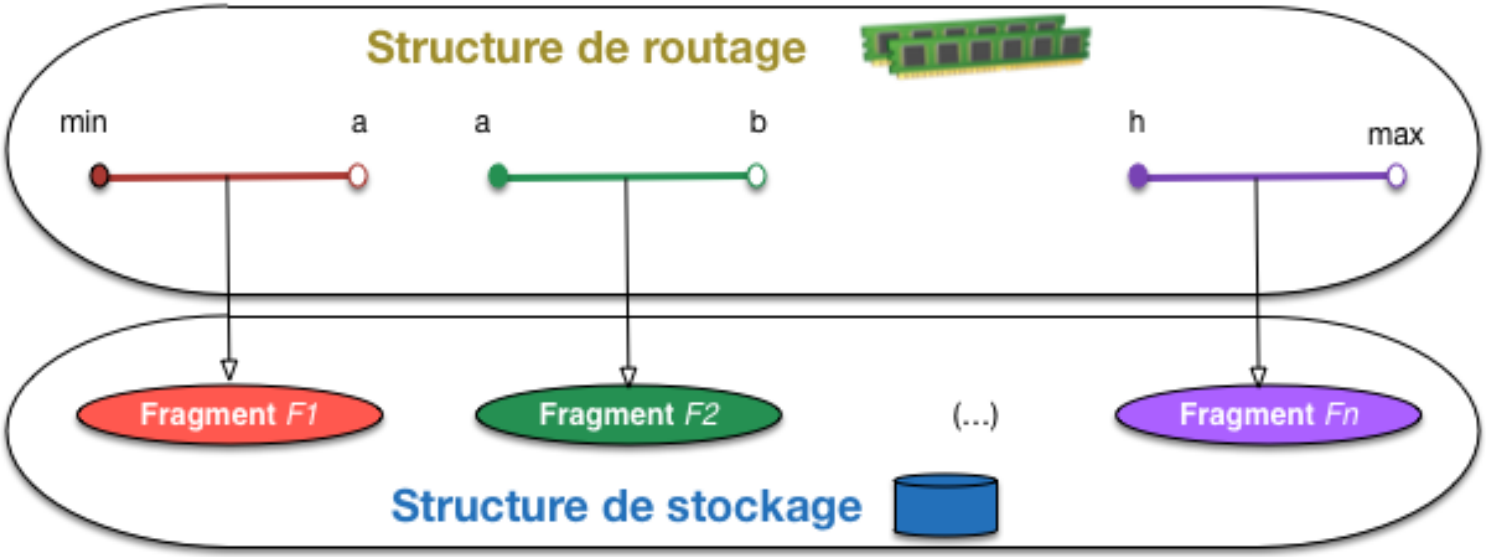
Opérations

Les opérations sont de type “dictionnaire”, et peuvent être transmises à un seul serveur :

- get(k) : v, recherche par clé
- put(k, v), insertion
- delete(k), suppression
- range(k1, k2), recherche par intervalle (peut impliquer plusieurs serveurs, ne marche pas avec le hachage)

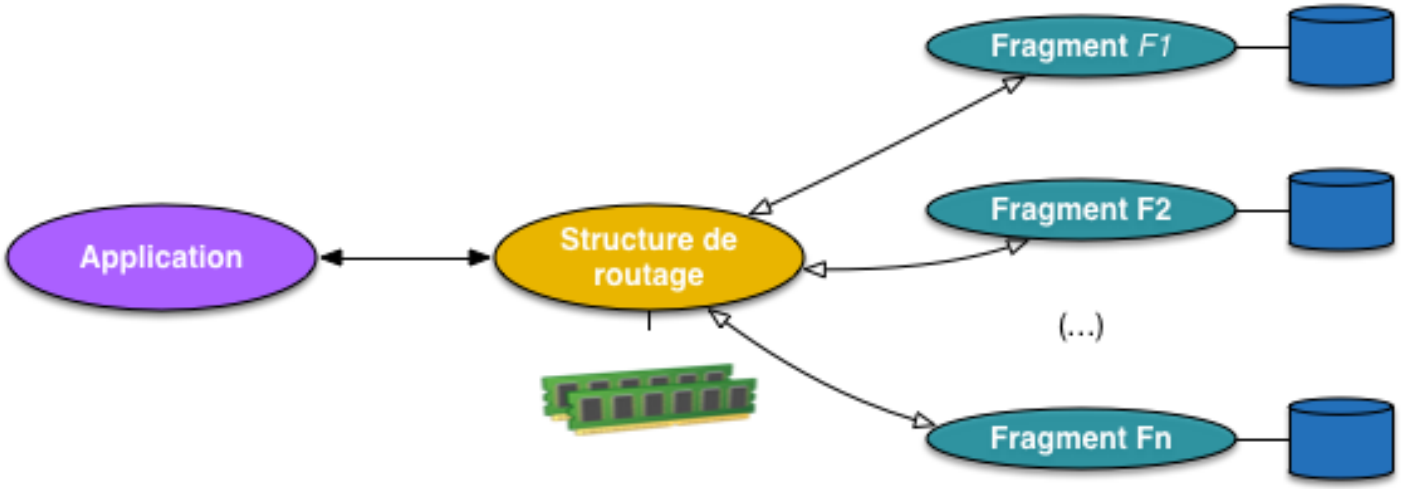
Techniques de sharding

- La technique de sharding utilisée par MongoDb est une technique par Intervalle
- Dans cette technique, les documents sont triés sur la clé, puis groupés par intervalles.
- Hbase (Big Table) utilise aussi cette technique

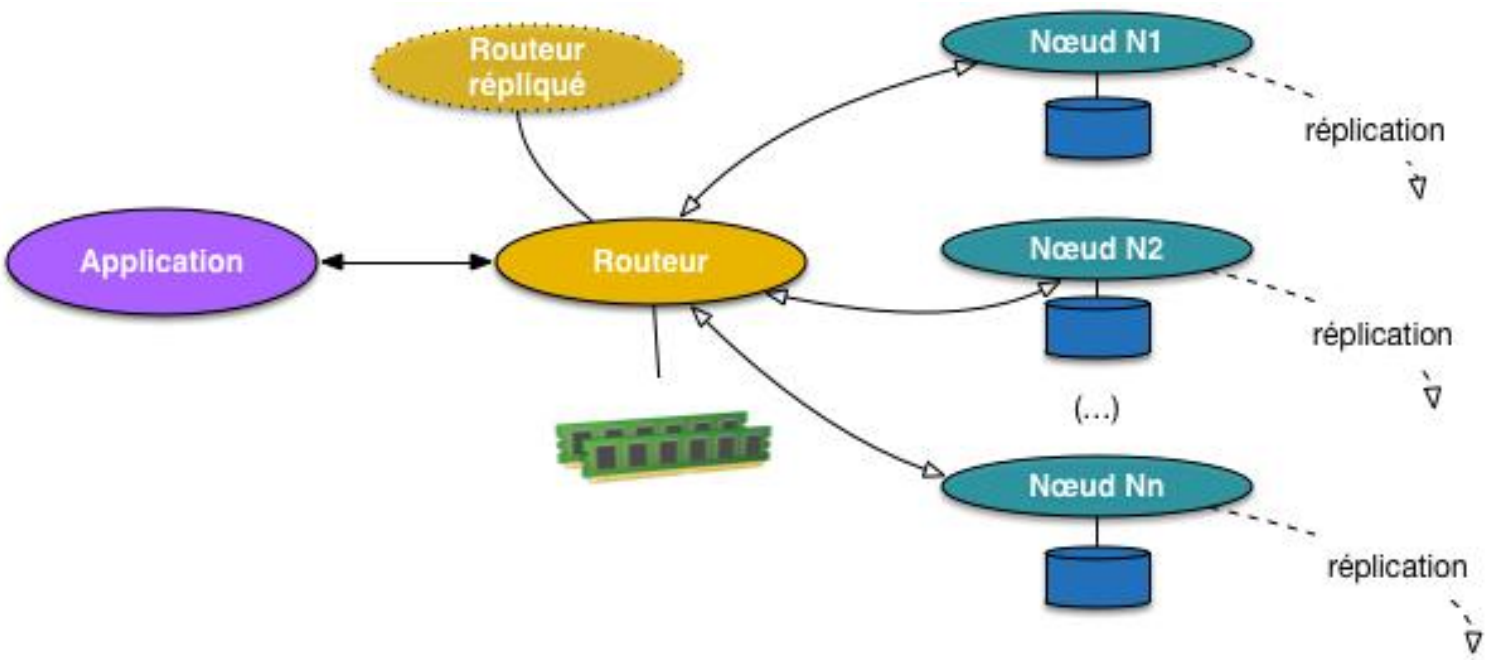


Structure d'un système shardé

- Routeur = Master
- Config server = Balancer
- Fragment = chunk, tablet, region, bucket, …



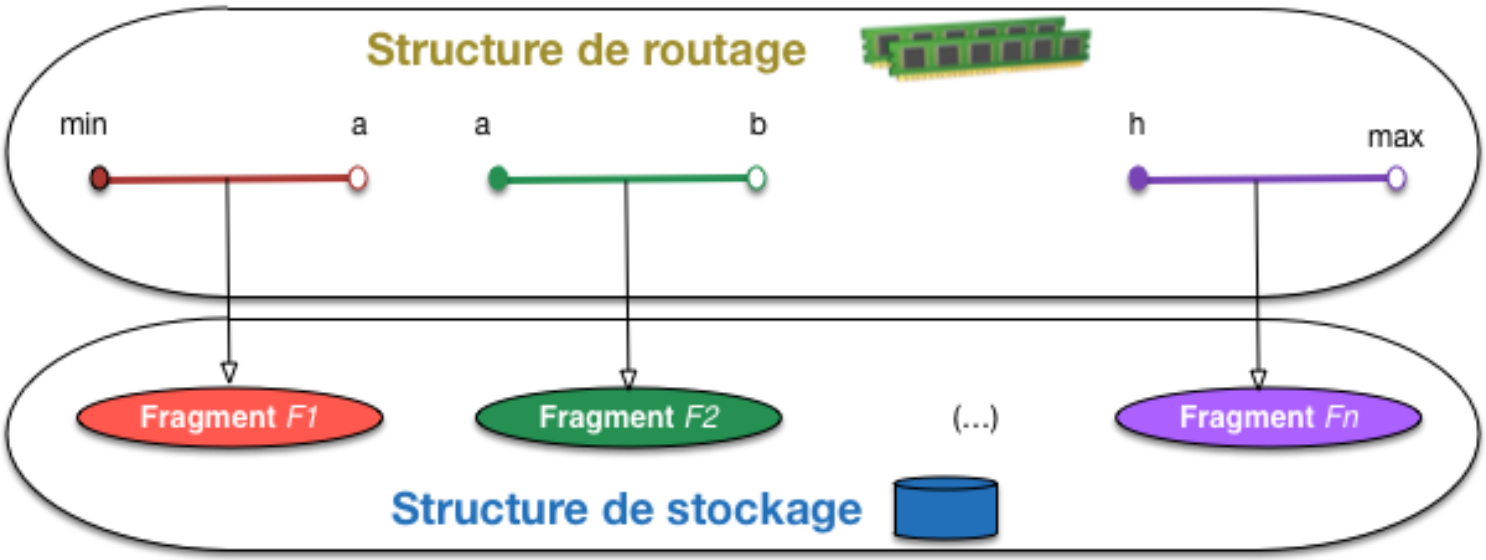
Structure d'un système shardé et répliqué



- Chaque serveur gère une réplication locale pour tolérance aux pannes.

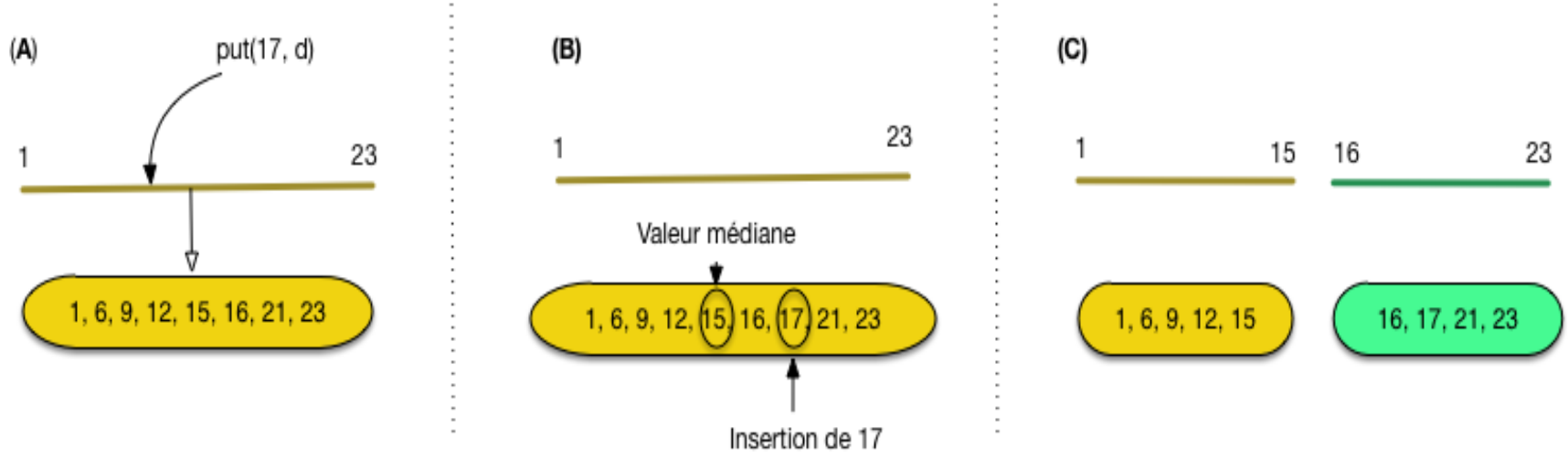
Sharding par intervalle : principe

- La collection est triée sur la clé et on la découpe en fragments d'une taille maximale pré-déterminée (de 64MO à quelques GO)
- Chaque fragment couvre donc un intervalle [minf ;maxf [.



Sharding par intervalle : dynamicité

- Le système évolue en permanence
- La division d'un gros fragment en deux petits se fait par le split.



Sharding par intervalle

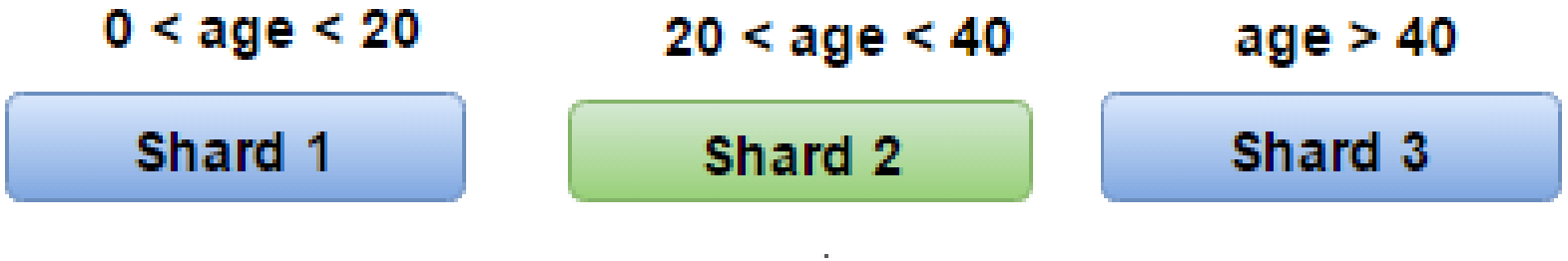
- Pour tirer pleinement partie de la répartition de charges, il faut que les données soient dispatchées uniformément entre les différents shards.
- C'est le rôle du routeur de répartir les données.
- Ce partitionnement se fait au niveau des collections et s'appuie sur une clé de sharding.
- MongoDB va définir un intervalle de valeur sur cette clé pour chaque shard

Sharding par intervalle

- On définit comme clé un champ présent dans tous les documents.
- Si vous êtes habitué au SQL cela doit vous sembler trivial, mais il faut se rappeler qu'avec MongoDB il n'y a pas de schémas imposés.
- Deux documents peuvent très bien appartenir à la même collection et avoir des champs différents, alors qu'en SQL il n'est pas possible pour deux lignes d'une même table d'avoir des colonnes différentes.
- Vous devez donc vous assurer que le champ que vous choisirez sera bien présent dans tous les documents de la collection que vous voulez partitionner.

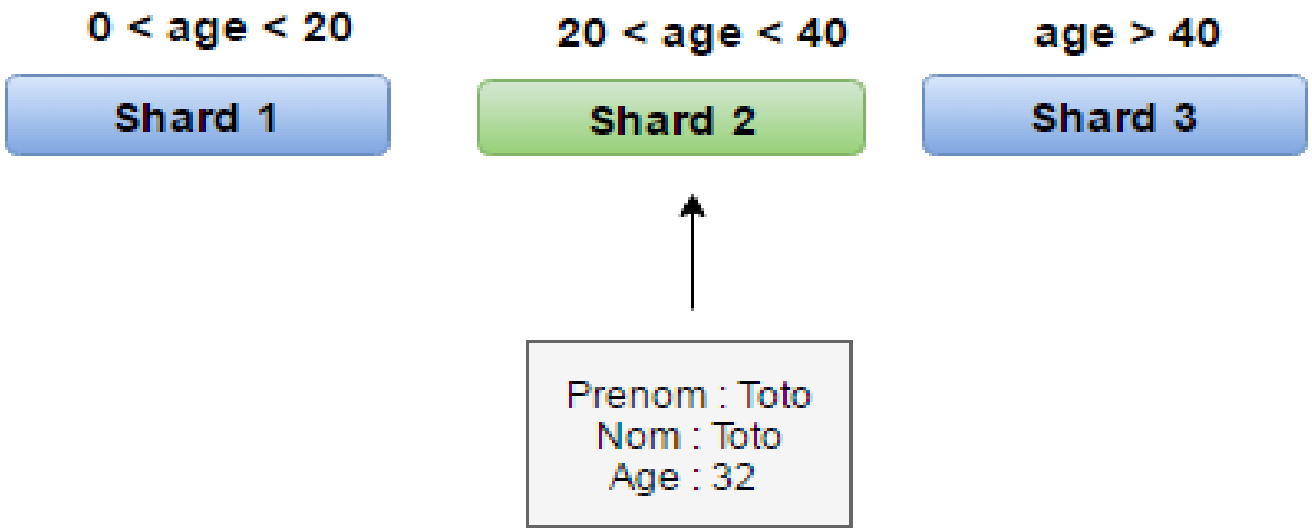
Sharding par intervalle : Exemple

- Prenons un document Personne qui contient le nom, le prénom et l'âge d'une personne.
- Si on choisit l'âge comme clé. MongoDB va définir automatiquement des intervalles d'âges. Le premier shard contiendra (par exemple) toutes les personnes de moins de 20 ans, le shard 2 toutes celles qui ont entre 20 et 40 ans, et le dernier shard les personnes de plus de 40 ans.



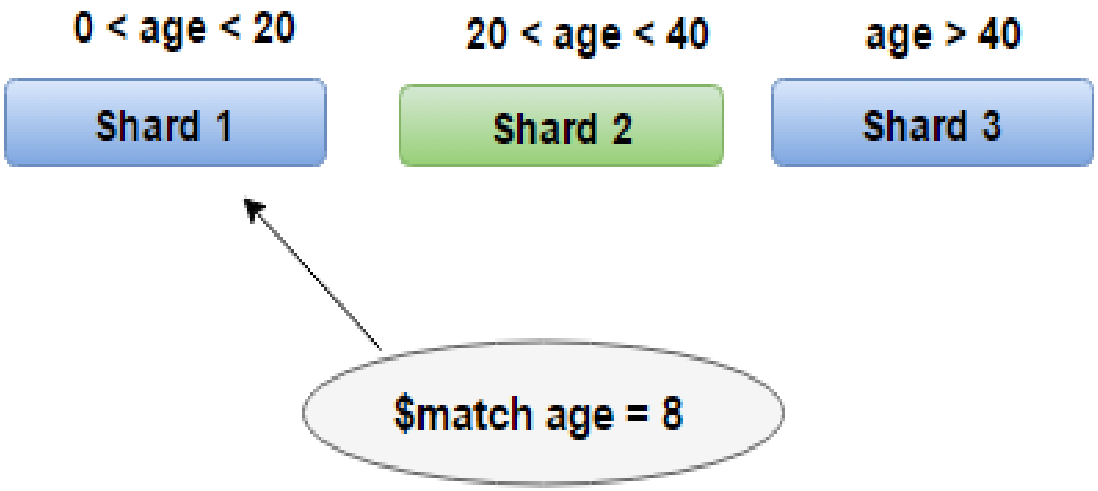
Sharding par intervalle : Exemple

- Quand un nouveau document sera ajouté dans la base, il sera directement dirigé vers le shard qui lui correspond, ici la personne a 32 ans, elle sera donc stockée sur le shard 2. C'est le routeur qui est chargé d'orienté le document.



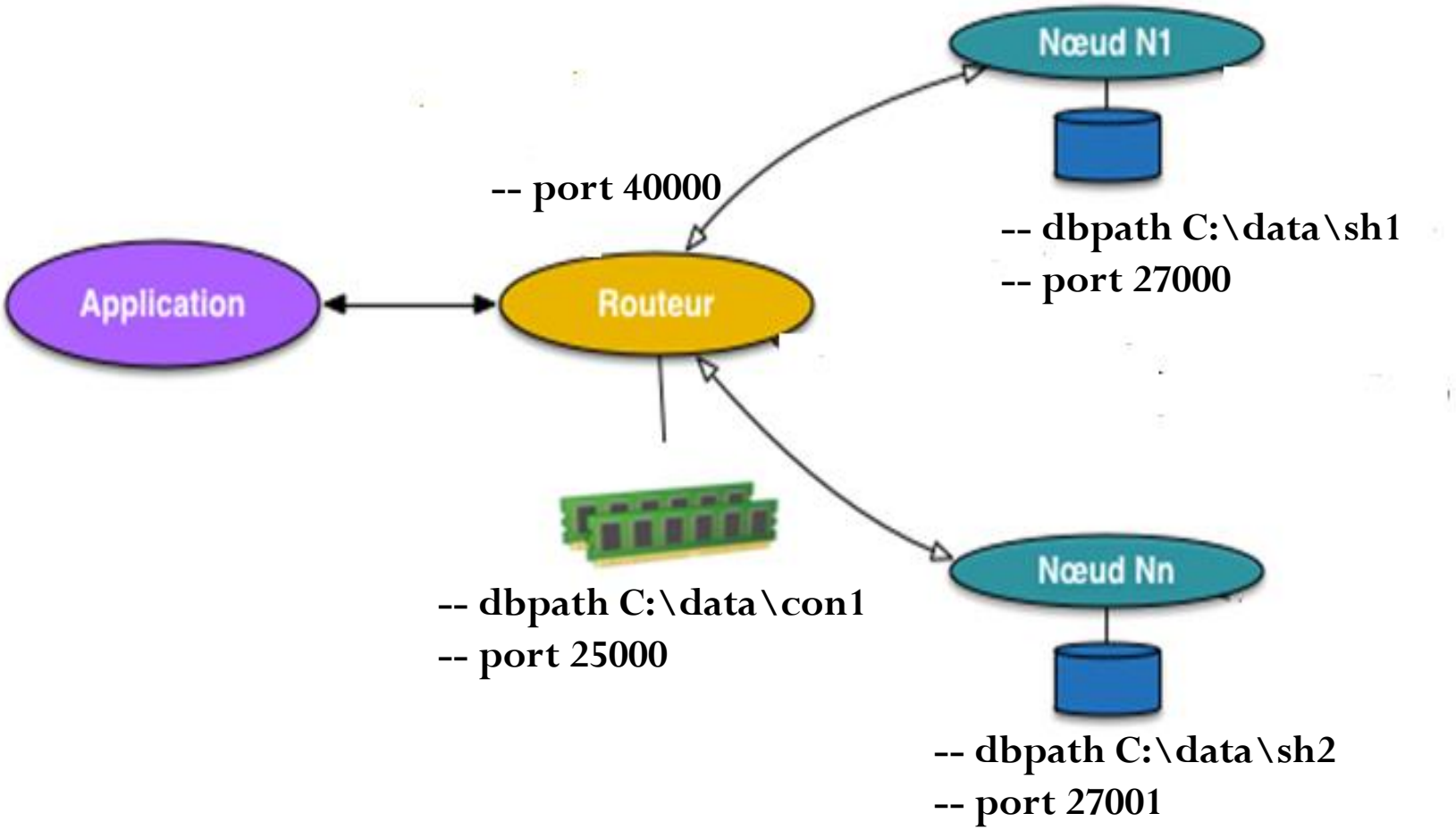
Sharding par intervalle : Exemple

- Une requête portant sur la clé de sharding sera très efficace car elle ne portera pas sur l'ensemble des données, mais seulement sur le shard correspondant à la requête, c'est à dire sur un sous ensemble de données.



Et si la requête s'applique sur plusieurs shards ?
elle s'effectue en parallèle sur les différents shards.

Implémentation



1/ Créer des répertoires (/data/sh1, /data/sh2, /data/con1)

2/ Lancez les deux serveurs de stockage

$ mongod --shardsvr --dbpath /data/sh1 --port 27000

$ mongod --shardsvr --dbpath /data/sh2 --port 27001

3/ Lancez le serveur de configuration

$ mongod --configsvr --dbpath /data/con1 --port 25000

4/ On continue avec le routeur, un processus mongos qui doit communiquer avec le serveur de configuration. Notez que le routeur n'utilise aucune donnée persistante et n'a donc pas besoin d'être associé à un répertoire.

$ mongos --port 40000 --configdb localhost:25000

6/ Lancez le client

$ mongo --port 40000

5/ Ajouter les shards (declarer les deux shard servers dans le configuration server)

mongos> sh.addShard("localhost:27000")

mongos> sh.addShard("localhost:27001")

6/ Tester le sharding sur la base shardDB

mongos> use shardDB

mongos> sh.enableSharding("shardDB")

mongos>sh.shardCollection("shardDB.person", {name: "hashed"})

mongos> charger des données dans la collection person de la base shardDB

mongos> db.person.getShardDistribution()

3/ Lancez le serveur de configuration

$ mongod --configsvr –replSet rs_sh --dbpath /data/con1 --port 25000

$ mongod --configsvr --replSet rs_sh --dbpath /data/con2 --port 25001

$mongo --port 25000

Rs.intiate()

Rs.add(« localhost:25001 »)

4/ On continue avec le routeur, un processus mongos qui doit communiquer avec le serveur de configuration. Notez que le routeur n'utilise aucune donnée persistante et n'a donc pas besoin d'être associé à un répertoire.

$ mongos --port 40000 --configdb rs_sh/localhost:25000