MongoDB - Aide-mémoire

À propos de cette aide-mémoire L'idée derrière

cela est d'avoir toutes (enfin, la plupart) les informations du didacticiel mentionné ci-dessus immédiatement disponibles dans un format très compact. Toutes les commandes peuvent être utilisées sur une petite base de données créée dans la section d'insertion. Toutes les informations contenues dans cette fiche sont fournies sans la moindre garantie d'exactitude. À utiliser à vos risques et périls. Amusez-vous !

Informations de base

Téléchargez MongoDB http://www.mongodb.org/downloads http://www.json.org/

Spécification JSON http://

Spécification BSON http://bsonspec.org/

Tutoriel Java www.mongodb.org/display/DOCS/Java+Tutorial

Insérer des documents

```
db.ships.insert({nom : 'USS Enterprise-D', opérateur : 'Starfleet', type : 'Explorer', classe : 'Galaxy', équipage : 750, codes :[10,11,12]})
db.ships.insert({nom : 'USS Prometheus', opérateur : 'Starfleet', classe : 'Prometheus', équipage : 4, codes :[1,14,17]})
db.ships.insert({nom : 'USS Defiant', opérateur : 'Starfleet', classe : 'Defiant', équipage : 50, codes :[10,17,19]})
db.ships.insert({nom : 'IKS Buruk', opérateur : 'Empire Klingon', classe : 'Warship', équipage : 40, codes :[100,110,120]})
db.ships.insert({nom : 'IKS Somraw', opérateur : 'Empire Klingon', classe : 'Raptor', équipage : 50, codes :[101,111,120]})
db.ships.insert({nom : 'Scimitar', opérateur : 'Romulan Star Empire', type : 'Warbird', classe : 'Warbird', équipage : 25, codes :[201,211,220]})
db.ships.insert({nom : 'Narada', opérateur : 'Romulan Star Empire', type : 'Warbird', classe : 'Warbird', équipage : 65, codes :[251,251,220]})
```

Trouver des documents

db.ships.findOne()	Trouve un document arbitraire
db.ships.find().prettyPrint()	Trouve tous les documents et utilise un formatage agréable
db.ships.find({}, {name:true, _id:false})	Affiche uniquement les noms des navires
db.ships.findOne({'name':'USS Defiant'})	Recherche un document par attribut

Concepts de base et commandes Shell

db.ships.<command>	db – handle implicite de la base de données utilisée
	navires – nom de la collection utilisée
utiliser <base de données>	Passer à une autre base de données
afficher les collections	Répertorie les collections disponibles
aide	Imprime les commandes disponibles et l'aide

Recherche de documents à l'aide des opérateurs $gt / $gte supérieur à /

supérieur à égal à	db.ships.find({class:{$gt:'P'}	$lt / $lte inférieur à / inférieur à égal
		db.ships.find({class:{$lte:'P'} db.ships.find({type:
$existe	est-ce qu'un attribut existe ou non	{$exists:true}}) db.ships.find({name:{$regex:'^USS \
$regex	Recherche de correspondance de	\sE'}}) db.ships.find({name : {$type:2}})
$type	modèle de style Perl par type d'élément	

Types BSON

Chaîne	2
Tableau	4
Données binaires	5
Date	9

http://www.w3resource.com/mongodb/mongodb-type-operators.php

Mise à jour des documents

db.ships.update({name : 'USS Prometheus'}, {name : 'USS Something'}) Remplace l'ensemble des ensembles de documents / modifie certains attributs	
d'un le document donné {$set : {operator : 'Starfleet', class : 'Prometheus'}}) supprime un attribut d'un document donné {$unset : {operator : 1}})	db.ships.update({name : 'USS Something'}, db.ships.update({name : 'USS Something'}, le

Suppression de documents

db.ships.remove({nom : 'USS Prometheus'}) db.ships.remove({nom :	supprime le document
{$regex:'^USS\\sE'}})	supprime à l'aide de l'opérateur

Chaque suppression de document individuel est atomique par rapport à un lecteur ou un écrivain simultané. Aucun client ne verra un document à moitié supprimé.

Page de la communauté G+ :
https://plus.google.com/u/0/communities/
115421122548465808444

Travailler avec des index

Création d'un index	db.ships.ensureIndex({nom : 1}) db.ships.dropIndex({nom :
Supprimer un index	1}) db.ships.ensureIndex({nom : 1, opérateur : 1,
Création d'un index composé	classe : 0}) db.ships.dropIndex({nom : 1, opérateur : 1, classe : 0})
Suppression d'un index composé	
Création d'un index composé unique	db.ships.ensureIndex({nom : 1, opérateur : 1, classe : 0}, {unique : true})

Index – Conseils et statistiques

db.ships.find ({'name':'USS Defiant'}).explain() Explique l'utilisation de l'index	
db.ships.stats()	Statistiques de l'index
db.ships.totalIndexSize()	Taille de l'index

Commandes système Top et statistiques

./mongotop Affiche le temps passé par opérations par collection ./	
mongostat Affiche un instantané sur le système MongoDB	

∑

UN g g R. ET g UN T je Ô N F R. UN M ET DANS Ô R. K

Étapes du pipeline

$project	Changez l'ensemble de documents en modifiant les clés et les valeurs. Il s'agit d'une cartographie 1:1.
$match	Il s'agit d'une opération de filtrage qui peut donc réduire la quantité de documents fournis en entrée au étape suivante. Cela peut être utilisé par exemple si l'agrégation ne doit se produire que sur un sous-ensemble de données.
$group	Ceci effectue l'agrégation réelle et comme nous regroupons par une ou plusieurs clés, cela peut avoir un effet réducteur sur la quantité de documents.
$trier	Trier les documents dans un sens ou dans l'autre pour l'étape suivante. Il convient de noter que cela peut nécessiter beaucoup de mémoire. Ainsi, si possible, il faut toujours essayer de réduire le nombre de documents en premier.
$ sauter	Avec cela, il est possible d'avancer dans la liste des documents pour un nombre donné de documents. Cela permet par exemple de partir uniquement du 10ème document. Généralement, cela sera utilisé avec « $sort » et surtout avec « $limit ».
$limit	Cela limite le nombre de documents à consulter selon le nombre donné à partir de la position actuelle.
$unwind	Ceci est utilisé pour dérouler un document utilisant des tableaux. Lorsque vous utilisez un tableau, les données sont en quelque sorte pré-jointes et cette opération sera annulée avec ceci pour avoir à nouveau des documents individuels. Ainsi, avec cette étape, nous augmenterons la quantité de documents pour l'étape suivante.

Comparaison avec SQL

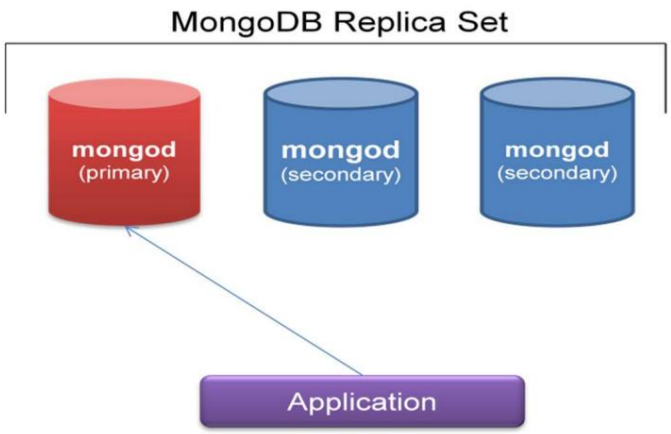
OÙ	$correspondance
PAR GROUPE	$groupe
AYANT	$correspondance
SÉLECTIONNER	$projet
COMMANDÉ PAR	$trier
LIMITE	$limite
SOMME	somme $
COMPTER	somme $
REJOINDRE	$se détendre

Exemples d'agrégation

db.ships.aggregate([{$group : {_id : "$operator", num_ships : {$somme : 1}}}])	Compte le nombre de navires par opérateur, serait en SQL : Opérateur SELECT, count(*) FROM navires Opérateur GROUP BY ;
db.ships.aggregate([{$project : {_id : 0, opérateur : {$toLower : "$operator"}, équipage : {"$multiply" : ["$crew",10]}}}])	Combinaison de $project-stage et $group-stage.

Expressions d'agrégation

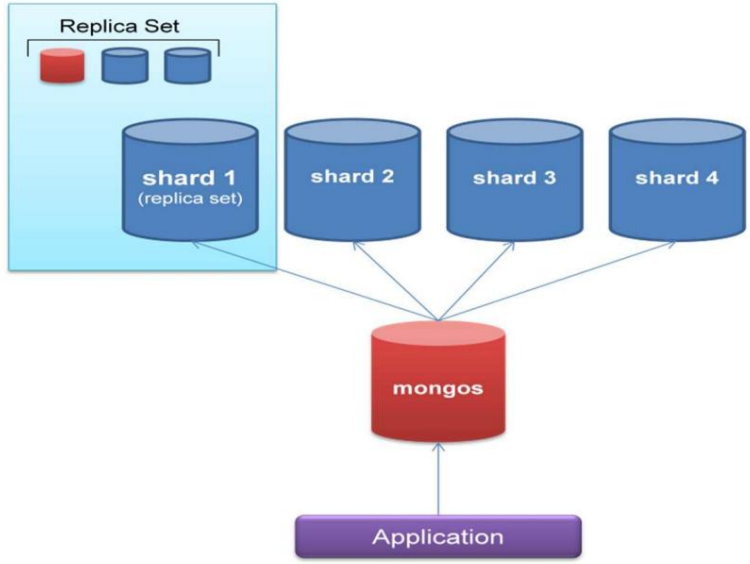
somme $	Résumer les valeurs $avg	db.ships.aggregate([{$group : {_id : "$operator", num_ships : {$sum : "$crew"}}}]) db.ships.aggregate([{$group : {_id : "$ Operator",
	Calcul des valeurs moyennes $min /	num_ships : {$avg : "$crew"}}}]) db.ships.aggregate([{$group : {_id : "$operator", num_ships : {$min : "$crew"}}}]) db.ships.aggregate([{$group :
$max	Trouver les valeurs min/max	{_id : "$operator", classes : {$push: "$class"}}}])
$push	Pousser les valeurs vers un résultat tableau	
$ajouterToSet	Pousser les valeurs vers un tableau de résultats sans doublons	db.ships.aggregate([{$group : {_id : "$operator", classes : {$addToSet : "$class"}}}])
$premier / $dernier	Obtenir le premier/dernier document	db.ships.aggregate([{$group : {_id : "$operator", last_class : {$last : "$class"}}}])



Ensembles de répliques			
Taper	Autorisé à voter ?	Peut-on devenir Primaire ?	Description
Régulier Oui		Oui	C'est le type de nœud le plus courant. Il peut agir comme un nœud principal ou secondaire. Les
Arbitre	Oui	Non	nœuds arbitres ne sont là qu'à des fins de vote. Ils peuvent être utilisés pour garantir qu'il existe un certain nombre de nœuds dans un jeu de réplicas même s'il n'y a pas beaucoup de serveurs physiques.
Retardé Oui		Non	Souvent utilisé comme nœud de reprise après sinistre. Les données stockées ici ont généralement quelques heures de retard sur les données de travail réelles.
Masqué Non		Non	Souvent utilisé pour l'analyse dans le jeu de réplicas.

Sharding

Chaque document doit définir une clé de partition. La valeur de la clé de fragmentation est immuable. La clé de partition doit faire partie d'un index et doit être le premier champ de cet index. Il ne peut y avoir d'index unique à moins que la clé de partition en fasse partie et soit alors le premier champ. Les lectures effectuées sans spécifier la clé de partition entraîneront des requêtes vers toutes les différentes partitions. La clé de partition doit offrir une cardinalité suffisante pour pouvoir utiliser toutes les partitions.



Durabilité des écritures w – Ceci

indique au pilote d'attendre que l'écriture soit reconnue. Cela garantit également qu'aucun index n'est violé. Néanmoins, les données peuvent toujours être perdues car elles ne sont pas nécessairement déjà conservées sur le disque. j – Cela signifie mode journal. Il indique au pilote d'attendre que le journal soit validé sur le disque. Une fois que cela s'est produit, il est certain que l'écriture sera persistante, sauf en cas de panne de disque. w=0 j=0 C'est « tirer et oublier ».

w=1 j=0	Attend un accusé de réception indiquant que l'écriture a été reçue et qu'aucun index n'a été violé. Les données peuvent toujours être perdues.
w=1 j=1	La configuration la plus économe en attendant que l'écriture dans le journal soit terminée.
w=0 j=1	Fondamentalement, c'est la même chose que ci-dessus.

Dans le contexte des jeux de réplicas, la valeur du paramètre w signifie désormais le nombre de nœuds qui ont accusé réception d'une écriture. Il existe une courte notation utile pour garantir que l'écriture a été effectuée sur une majorité de nœuds en utilisant w='majority'. Pour le paramètre de journal, la valeur un reste la meilleure qui puisse être obtenue. Cela signifie que les données sont écrites dans le journal du nœud principal.