Rodney Badran

Table Des Matières

I. INTRODUCTION - NOSQL 2

II. MONGODB 5

1. Introduction……..5

2. Objectif…….....5

3. Architecture technique……6

4. Json…….....6

5. Bson……..7

6. MongoDB n'est pas……...9

III. ARCHITECTURE NEXUS 11

IV. SHARDING ..... **Error! Bookmark not defined.**

V. REPLICATION ..... **Error! Bookmark not defined.**

VI. COMPARAISON ENTRE SQL ET NOSQL ..... **Error! Bookmark not defined.**

1. Structure de Données SQL vs NOSQL……17

2. Intégrité de Données SQL vs NOSQL……17

3. La Logique de Jointure SQL vs NOSQL……...18

4. La Normalisation SQL vs La Dé-Normalisation NOSQL……18

5. Performance SQL vs NOSQL……19

6. Transaction SQL vs Transaction NOSQL……19

7. Propriétés d'ACID……20

8. Propriétés de CAP……20

9. SQL vs NOSQL dans les Nouvelles Technologies……21

VII. CONCLUSION 22

VIII. INSTALLATION DE MONGODB 24

IX. INSTALLATION DE MONGODB COMPASS ..... **Error! Bookmark not defined.**

X. COMMNADES DE MONGODB ..... **Error! Bookmark not defined.**

I. INTRODUCTION - NOSQL

Les bases relationnelles ont été inventées par Edgar Codd en 1970.
Ces dernières années, plusieurs avis nuancés ont été émis concernant le modèle relationnel :

1. Difficultés pour sauver et récupérer des grappes d'objets avec des programmes écrits en Java ou C#... (on parle de "mismatch impedance"). Base de données relationnelles (RDBMS) et l'application est écrite avec un langage oriente objet.

2. Difficultés pour satisfaire les besoins des applications Web à grande échelle (nouveaux besoins métiers, nouvelles contraintes techniques).

3. Le VOLUME des données créées double tous les 2 ans. IDC estime qu'en 2020 le volume atteindra 44 Zettabytes (1 ZB = 1 milliards de terabytes).

4. La VARIÉTÉ des types de données créées.

5. La VÉLOCITÉ avec laquelle les données changent est également très importante (Internet of Things).

D'où la naissance de NoSql. Le NoSql, qui signifie "not only Sql" (parfois on parle de "NO Sql"/pas Sql) désigne les bases de données qui ne sont pas fondées sur l'architecture classique des bases de données relationnelles.

L'utilisation de base de données NoSql a explosé depuis quelques années. Google (BigTable), Amazon (Dynamo), LinkedIn (Voldemort), Facebook (Cassandra puis HBase), SourceForge.net (MongoDB), Ubuntu One (CouchDB), Baidu (Hypertable) ont été les premiers précurseurs du modèle NoSql.

On distingue 4 types de bases de données NoSQL :

1. Bdd Entrepôts Clé-Valeurs ECV (Key Value Stores)

Les données sont stockées en clé-valeur: une clé plus un blob (dans lequel on peut mettre: nombre, date, text, XML, photo, vidéo, structure objet)

i.e.: redis, Oracle Berkeley DB

Pour	Contre
facilement scalable	mise a jour complique
temps de réponse en écriture/ lecture très bas	requêtes limitées

2. Bdd Orientées Documents (Document DB)

Ces bdd stockent des données semi-structurées: le contenu est formaté json ou XML, mais la structure n'est pas contrainte.

i.e.: CouchDB, MongoDB

Pour	Contre
Requêtage plus complet	duplication des données
Flexibilité	cohérence pas forcement assurée
évolutif au cours du temps	

3.

Bdd Orientées Colonnes (Wide-Column)

Ces bdd se rapprochent des bdd relationnelles, cat elles permettent de remplir un nombre de colonnes variable.

i.e.: Cassandra, Google Bigtable

Pour	Contre
Capacité de stockage accrue	efficace pour des données de même type et similaires
accès rapide aux données	Requêtage limite
évolutif au cours du temps	

4. Bdd Orientées Graphes (Graph DB)

Ces bdd sont gérées par nœuds, relations et propriétés.
Elles gèrent des données spatiales, sociétés ou financières.

i.e.: neo4j

Pour	Contre
adapté à la gestion de données relationnelles	architecture limitée a certains cas
architecture modelable	

II. MongoDB

1. Introduction

MongoDB est une base de données open source qui utilise un modèle de données orienté document.
L'un des nombreux types de bases de données qui se posent au milieu des années 2000 sous la bannière NOSQL.
Il est programmé en C++.

Le nom de la base de données dérive du mot « humongous » pour représenter l'idée de supporter de grandes quantités de données.
Son logo, la feuille d'arbre, représente « simple et naturel ».

Merriman et Horowitz ont form é 10Gen Inc. en 2007 pour commercialiser MongoDB. La société a été rebaptisée MongoDB Inc. en 2013 à New York.

2. Objectif

C'est une base de données généraliste. Elle est particulièrement appréciée pour:
. Assurer une Scalabilité face aux changements dynamiques et rapides des données.
. Assurer une haute performance et flexibilité pour les types de données.
. Un besoin de redondance et de continuité de service.
. Une répartition géographique des données (Distribution/partition).
. Un besoin de système de recherche, de stockage de fichier ou d'un système information géographique.

3. Architecture technique

Au lieu d'utiliser des tables et des lignes comme dans les bases de données relationnelles, MongoDB est construit sur une architecture de collections et de documents, similaire à Json, sans schéma prédéfini.

On distingue la terminologie suivante entre Sql et MongoDB :

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
Column	Field

4. Json

JavaScript Object Notation ou Json est un format de données textuelles (et pas un langage informatique) dérivé de la notation des objets du langage JavaScript. Il permet de représenter de l'information structurée comme le permet XML.

Créé par Douglas Crockford entre 2002 et 2005.

Un document JSON, format « .json », a pour fonction de représenter de l'information accompagnée d'étiquettes permettant d'en interpréter les divers éléments, sans aucune restriction sur le nombre de celles-ci.

Un document JSON ne comprend que deux types d'éléments structurels :

. Des ensembles de paires "nom" (ou clé) / "valeur" ;

. Des listes ordonnées de valeurs.

- {…} : les accolades définissent un objet.

- "cle":"valeur" : Les guillemets (double-quotes) et les double-points définissent un couple clé/valeur (on parle de membre).
- […] : Les crochets définissent un tableau (array).
  Les valeurs sont séparées par virgule.

Une valeur peut être une chaîne de caractères entre guillemets, un nombre, un tableau, un objet, ou aussi « true / false / null / (vide) ».

L'avantage de Json sur XML est sa vitesse de traitement et sa simplicité de mise en œuvre étant reconnu nativement par JavaScript.
Comme il n'a pas beaucoup de types comme XML, il est moins compliqué que le dernier.
Json a une syntaxe courte à connaitre, il est moins verbeux, donc plus léger que le XML tout en restant lisible par les machines et les ordinateurs.
Le modèle de type document réduit au maximum le nombre de relation dans la base de donnée, ce qui simplifie sa structure et augmente sa lisibilité.

5. Bson

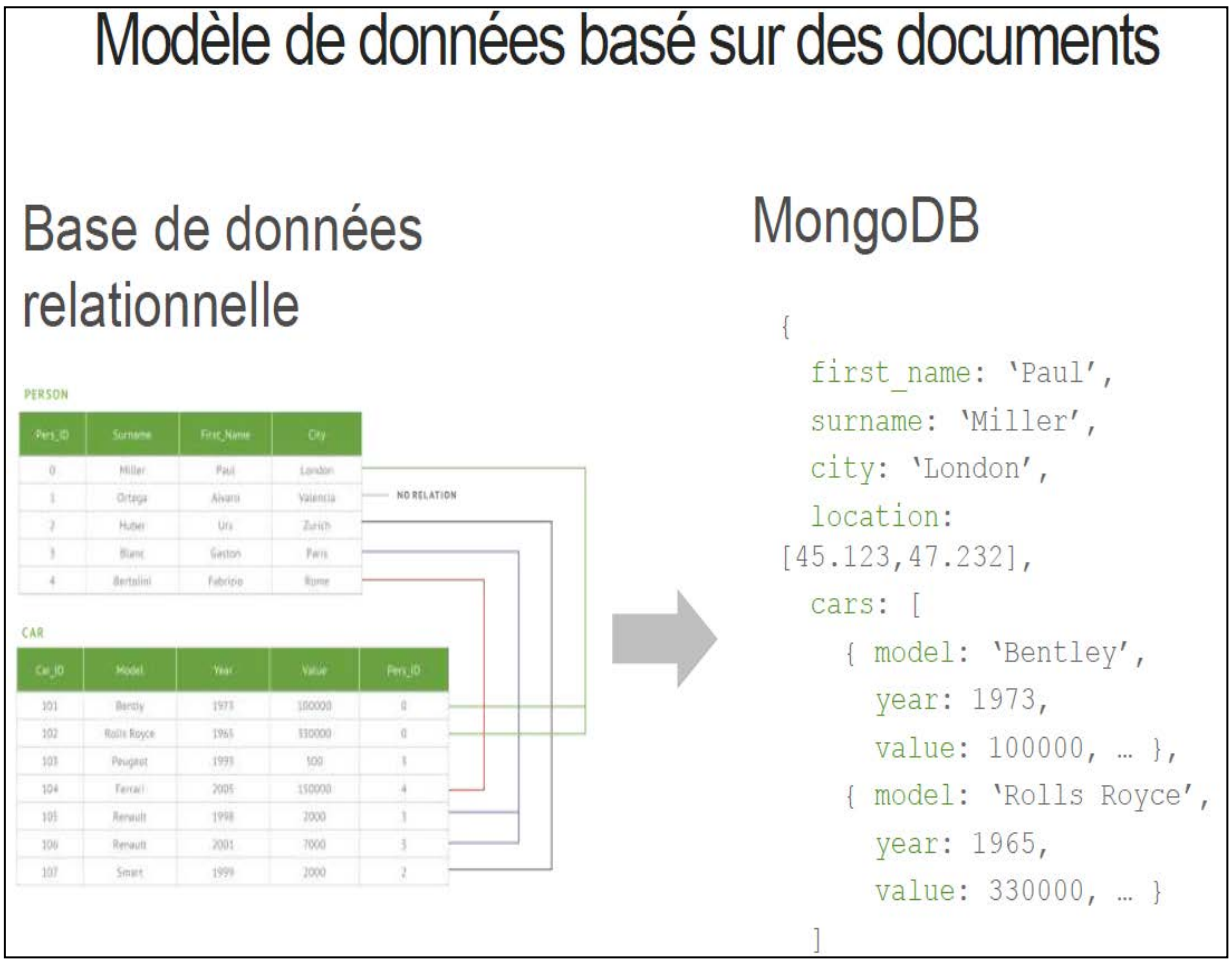
Une base de données mongo est décomposée en une série de fichiers BSON sur le disque dur, avec une taille croissante allant jusqu'à 2Go.

BSON est son propre format, construit spécifiquement pour MongoDB qui stocke les données sur le disque en tant que BSON dans votre répertoire de chemin de données, qui est habituellement / data / db.
BSON est une extension du modèle JSON pour donner d'autres types de données, et est efficace pour codage et décodage avec des différents langages.

Les types BSON sont nominalement un ensemble plus grand de types JSON (JSON n'a pas de type date par exemple), à l'exception de ne pas avoir un type « Number » universel comme le fait JSON.

(les objets Date sont stockés sous la forme d'un entier signé 64 bits représentant le nombre de millisecondes depuis l'époque Unix (1er janvier 1970)).

Comparé à JSON, BSON est conçu pour être efficace à la fois dans l'espace de stockage et la vitesse de numérisation (storage, space et scan-speed).



6. MongoDB n'est pas

1. Une suite analytique : Il n'est pas un concurrent de SAS, SPSS…

2. Une technologie d'entrepôt de données : Il n'est pas un concurrent de Teradata, Vertica…

3. Un outil de BI : Il n'est pas un concurrent de Tableau, Qlikview…

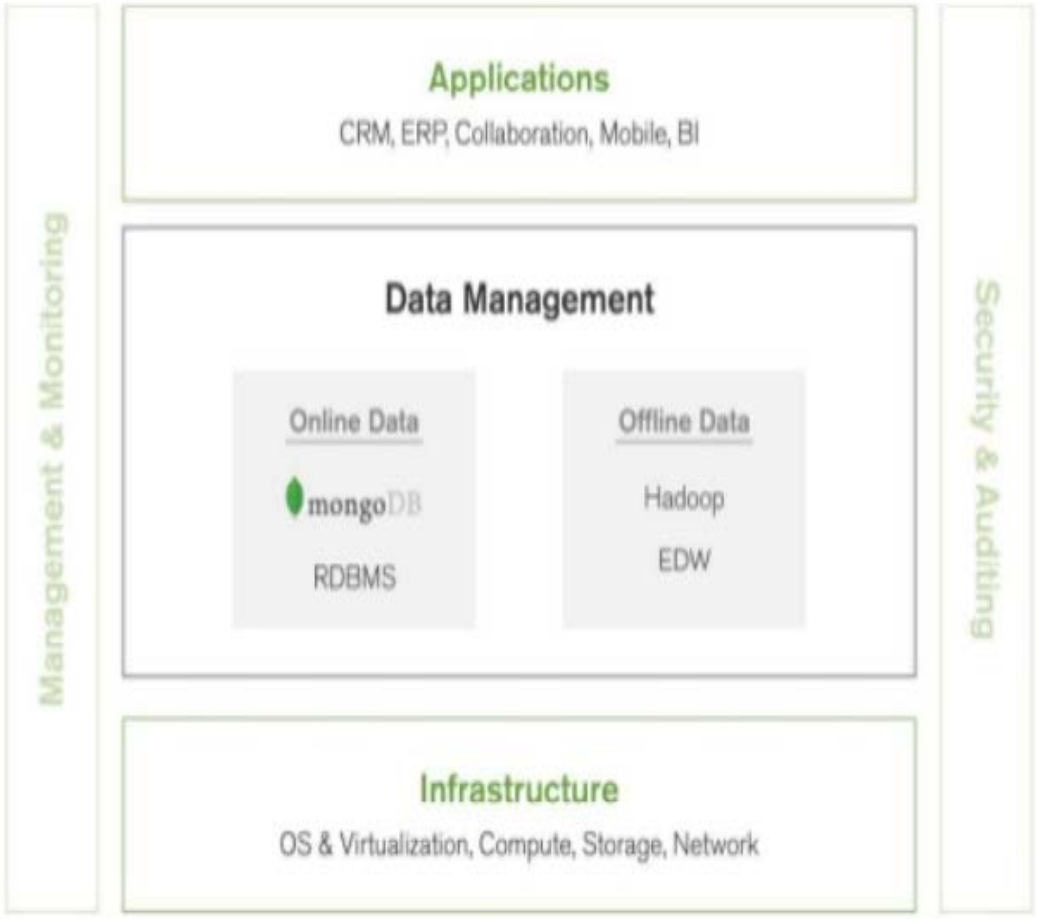
4. Traitement des transactions de service comptable : Il n'est pas un concurrent des mainframes de IBM.

5. Serveur principal d'un système de facturation ou d'un système de comptabilité générale : Il n'est pas un concurrent d'Oracle EBS.

6. Un moteur de recherche : Il n'est pas un concurrent de SOLR, ElasticSearch…

Il cohabite ces technologies mais ne les remplace pas.

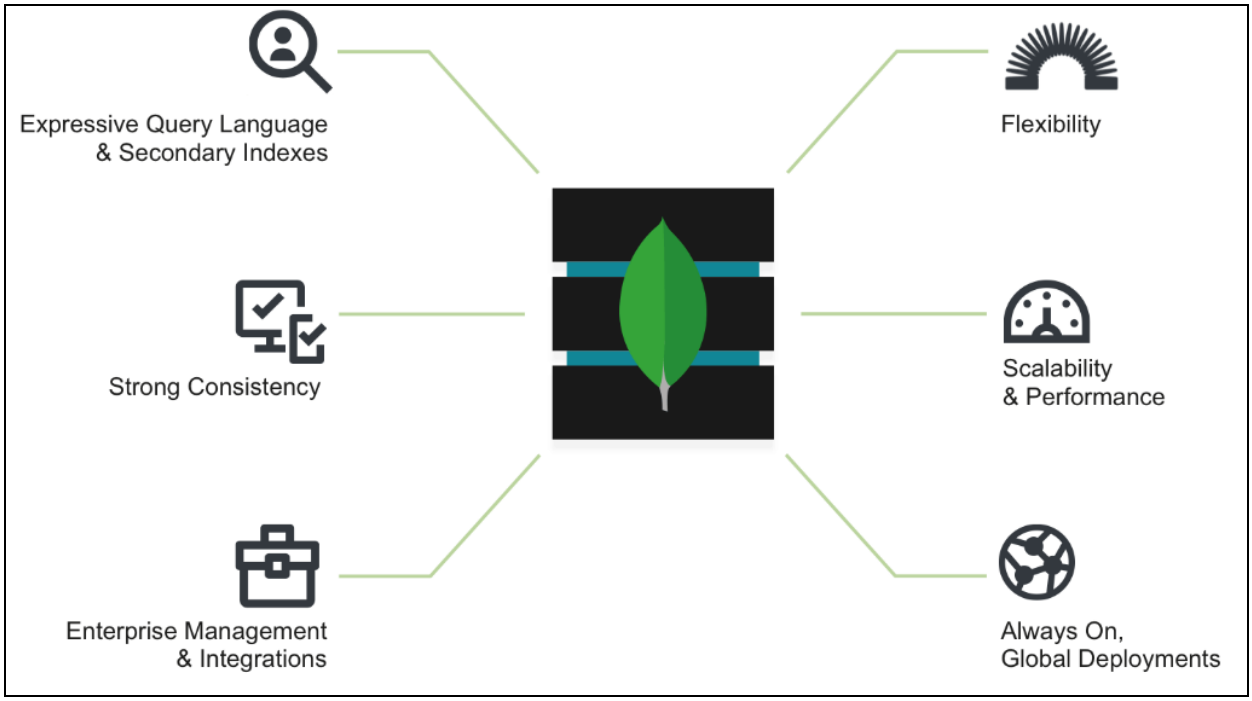
MongoDB et le stack informatique d'entreprise



mongoDB

III. ARCHITECTURE NEXUS

La philosophie de l'architecture de MongoDB est de combiner les capabilités de la bdd relationnelle avec les innovations de NOSQL.
Cette architecture s'appelle Nexus :



1. Langage de requête exclusif (Expressive query language) et Index Secondaire (Secondary indexes): Les utilisateurs doivent avoir le pouvoir d'accéder et de manipuler leurs données de manière sophistiquée pour prendre en charge les applications opérationnelles et analytiques. Et les index jouent un rôle essentiel dans la fourniture d'un accès efficace aux données, supportés nativement par la base de données plutôt que maintenus dans le code de l'application.

2. Cohérence forte (Strong consistency): Les applications doivent pouvoir lire immédiatement ce qui a été écrit dans la base de données.

3. Gestion d'entreprise et intégrations (Enterprise Management and Intégrations) : Les bases de données ne sont qu'une partie de l'infrastructure de l'application et doivent s'intégrer parfaitement dans l'organisation informatique de l'entreprise. Les entreprises ont besoin d'une base de données qui peut être sécurisée, surveillée, automatisée et intégrée à leur infrastructure technologique, à leurs processus et à leur personnel, y compris les équipes opérationnelles, les administrateurs de bdd et les analystes.

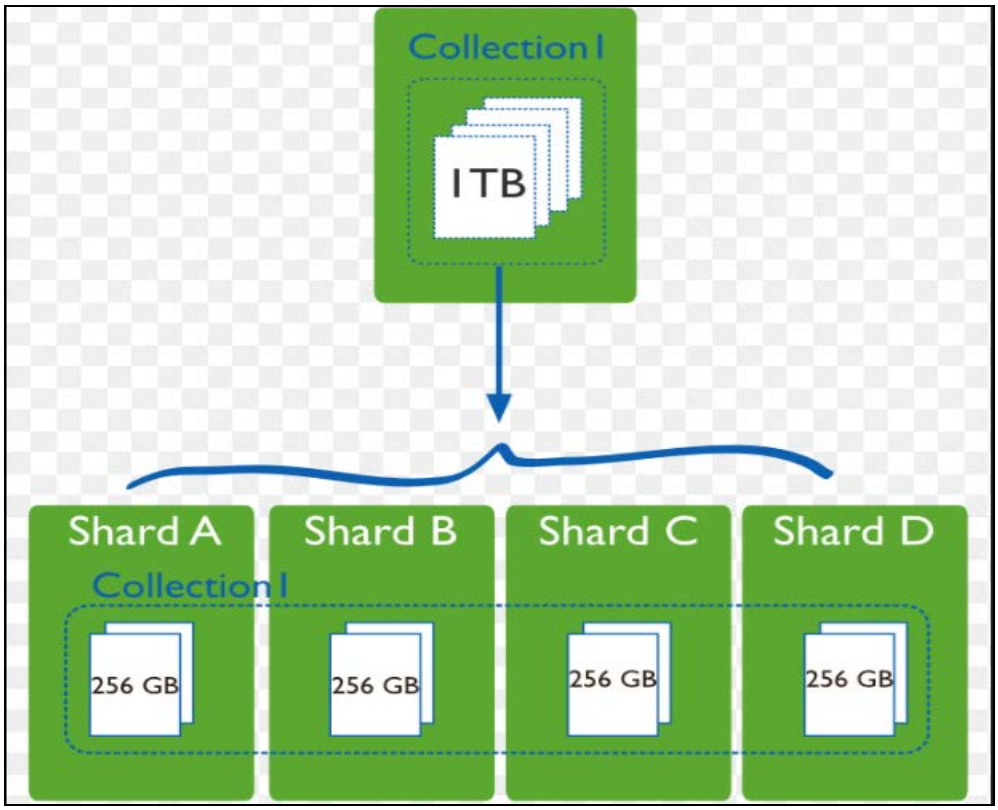
4. Flexibilité (Flexible Data Model) : Les bases de données NOSQL sont apparues pour répondre aux exigences relatives aux données des applications modernes. Un modèle de données flexible permet de stocker et de combiner facilement des données de n'importe quelle structure et de permettre une modification dynamique du schéma sans interruption.

5. Évolutivité et Performance (Scalability & Performance) : Les bases de données NoSQL ont toutes été construites en mettant l'accent sur l'évolutivité, de sorte qu'elles incluent toutes une forme de partitionnement. Cela permet à la base de données d'évoluer sur du matériel standard déployé sur site ou dans le cloud, ce qui permet une croissance quasi illimitée avec un débit plus élevé et une latence plus faible que les bases de données relationnelles

6. Des déploiements globaux et permanents: (Always-On Global Deployments) : Les bases de données NoSQL sont conçues pour les systèmes hautement disponibles qui offrent une expérience cohérente et de haute qualité aux utilisateurs du monde entier. Ils sont conçus pour s'exécuter sur plusieurs nœuds, y compris la réplication pour synchroniser automatiquement les données entre les serveurs, les racks et les centres de données.

IV. SHARDING

Sharding est le processus de stockage des enregistrements de données sur plusieurs machines et il est l'approche de MongoDB pour répondre aux exigences de la croissance des données.



C’est est un type de partitionnement de base de données qui sépare les bases de données très volumineuses en des parties plus petites, plus rapides et plus faciles à gérer, appelées partitions de données ou Shard.
Le mot shard signifie une petite partie d'un tout.

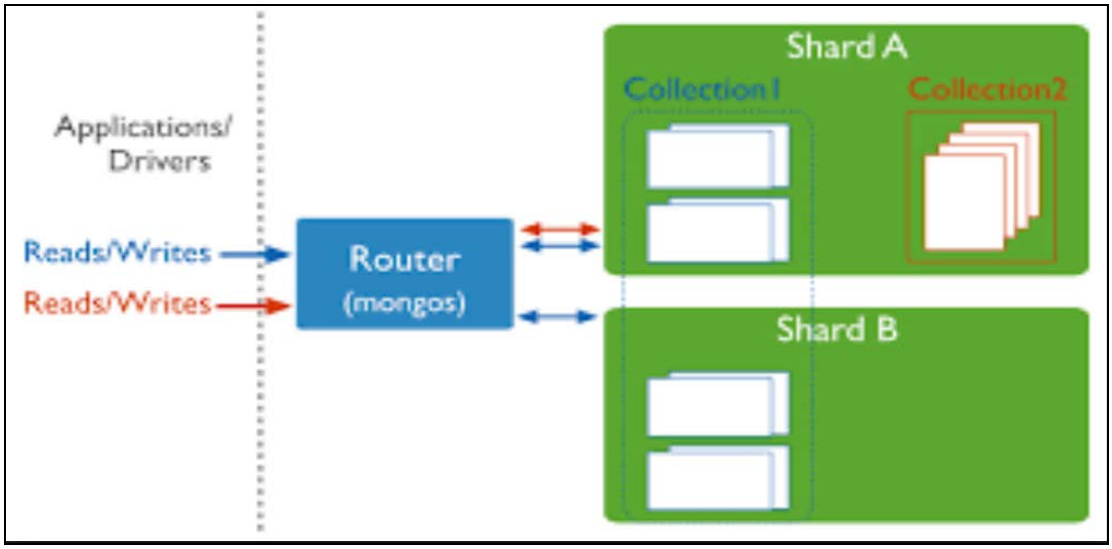
Dans le sens le plus simple, MongoDB divise la bdd en plusieurs bases de données beaucoup plus petites qui ne partagent rien et peuvent être réparties sur plusieurs serveurs.

Comme la taille des données augmente, une seule machine peut ne pas être suffisante pour stocker les données ni fournir un débit de lecture et d'écriture acceptable, alors Sharding résout le problème avec la mise à l'échelle horizontale (Horizontal Scaling) : on ajoute des machines pour soutenir la croissance des données et les exigences des opérations de lecture et d'écriture.

Le Sharding de MongoDB est synonyme de partitionnement horizontal.

(La mise à l'échelle verticale (Vertical Scaling) 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 (Hardware Scaling)).

MongoDB utilise Mongos pour son sharding.
Mongos, “MongoDB Sharded Cluster Query Router” ou MongoDB Shard, est un service de routage pour les configurations de fragmentation MongoDB qui traite les requêtes de la couche application, et détermine l'emplacement de ces données dans le cluster fragmenté, afin de compléter ces opérations.
Du point de vue de l'application, une instance de mongos se comporte de la même manière que n'importe quelle autre instance de MongoDB.



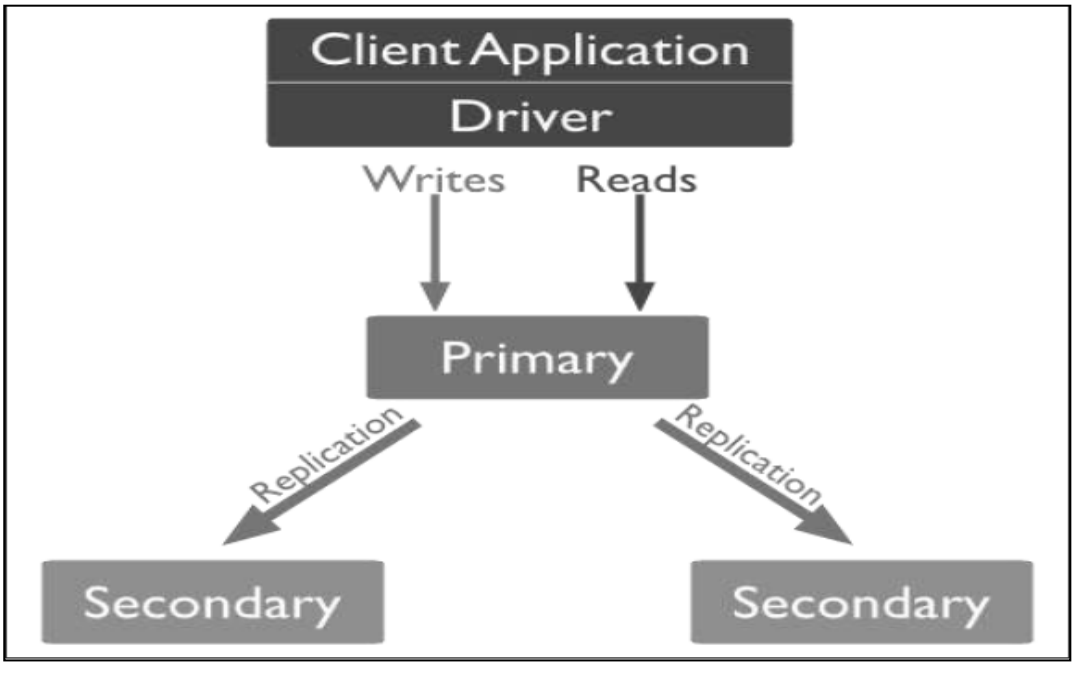
V. REPLICATION

La réplication consiste à écrire les données sur plusieurs serveurs.
La réplication dans MongoDB sert à garder les données en toute sécurité, d'avoir une haute disponibilité des données (High Availability, 24 * 7), et pour le plan de reprise d'activité (Disaster recovery).

MongoDB met à disposition un concept de réplication des données nommé Replica Set. Sa mise en place est simple. Il permet de garantir la haute disponibilité des données et dans certains cas d'augmenter les capacités en lecture.

Replica Set possède les membres suivants :

1. Primary: le seul membre autorisant l'écriture des données.
2. Secondary: Il s'agit d'une réplication du membre Primary qui accepte uniquement la lecture des données. Il peut à son tour devenir membre Primary dans le cas où ce dernier est indisponible.
3. Priority 0: Il fonctionne comme un membre Secondary mais ne peut devenir membre Primary.
4. Hidden: Il fonctionne comme un membre Secondary mais n'est pas visible aux clients. Ce membre sert généralement de backup.

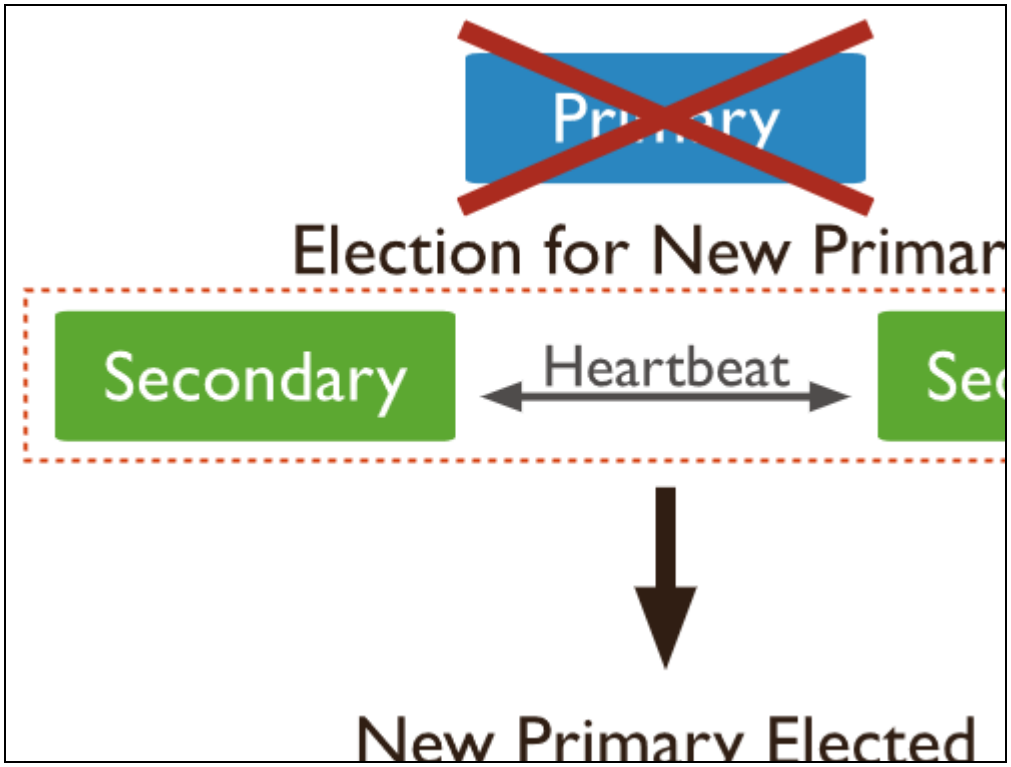


En général, un minimum de 3 nœuds sont requis :

1 primary et 2 secondary.

Dans le cas d'échec du primary, "élection" établit un nouveau primary, et un secondary est élu primaire.

Après la récupération de l'ancien primary, il rejoint le « replica set » comme un nœud secondary.



VI. COMPARAISON ENTRE SQL ET NOSQL

1. Structure de Données SQL et NOSQL

SQL organise le stockage de données sur le principe de tables reliées entre elles, dans un schéma prédéfini avant que toute logique métier (business logic) puisse être développée pour manipuler des données.
La structure et les types des données sont fixés à l'avance, et les tables SQL imposent un modèle de données strictes, donc il est difficile de faire des erreurs.

NOSQL stocke et manipule des documents qui correspondent à des collections d'objets sans schéma prédéterminé, ni une conception de document ou même une collection définie à l'avance.
NOSQL est plus flexible et pardonnable, mais la possibilité de stocker des données n'importe où peut entraîner des problèmes de cohérence.
Une base de données NOSQL peut être plus adaptée aux projets où les exigences initiales en matière de données sont difficiles à déterminer.

2. Intégrité de Données SQL et NOSQL

SQL permet d'appliquer des règles d'intégrité de données à l'aide de contraintes de clés étrangères.

Par contre, MongoDB acceptera tout ce qui est fourni.
Cela signifie que plusieurs développeurs, applications et sous-systèmes peuvent effectivement détruire les données de l'autre en réutilisant accidentellement les noms de champs ou en utilisant de mauvaises commandes.

Une des premières choses que vous devez faire avec MongoDB est d'écrire une couche de données très claire et de s'assurer que tout le monde l'utilise correctement.

3. La Logique de Jointure SQL et NOSQL

Les requêtes SQL offrent une puissante clause JOIN.

Nous pouvons obtenir des données reliées dans plusieurs tables en utilisant une seule instruction SQL.

Par exemple :

```
SELECT    e.etudiant_nom, matière_code, i.Annee
FROM      etudiants e, matieres m, inscription i
WHERE     i.matiere_id = m.matiere_id
AND       i.etudiant_id  = e.etudiant_id;
```

Cela renvoie tous les titres de livres, leurs auteurs et les noms d'éditeurs associés.

NOSQL n'a pas toujours d'équivalent de JOIN.

Par exemple avec Mongo DB, si nous avons utilisé des collections normalisées, nous devons récupérer séparément la collection de livres et d'éditeur et c'est le code qui devra les associer en implémentant une logique métier.

C'est pour cette raison que la dé-normalisation est souvent essentielle.

```
db.inscription.insert({_id: ObjectId(),Etudiant_nom: 'rodney', Code: '123',
matiere_nom: 'java', matiere_code: 'abc', annee: '2017-2018'})
```

4. La Normalisation SQL vs La Dé-Normalisation NOSQL

Normalisation revient à la façon dont les données sont ou pas dupliquées (NOSQL) ou reliées par des clés étrangères (SQL).

C'est pourtant le principe de « NOSQL dé-normalisation » !

Cela conduit à des requêtes beaucoup plus rapides, mais la mise à jour des informations (downsides) de l'éditeur dans plusieurs enregistrements sera considérablement plus lente.

5. Performance SQL vs NOSQL

NOSQL est régulièrement cité comme étant plus rapide que SQL.
Et ce n'est pas surprenant car Le principe de « dé-normalisation » induit une représentation plus simple et permet donc de récupérer toutes les informations sur un élément spécifique dans une seule requête.
Il n'y a donc pas besoin de liens JOIN ou de requêtes SQL complexes.
Mais la redondance des informations alourdie considérablement les opérations de mise à jour.

En résumé, les bases de données « orientées document » ne sont pas intrinsèquement plus rapides.
Par exemple, MongoDB est sous-performant sur les requêtes d'agrégation ou sans index.

6. Transaction SQL vs Transaction NOSQL

Dans le domaine des bases de données, une opération sur les données est appelée une transaction ou transaction informatique.

Dans les bases de données SQL, plusieurs mises à jour peuvent être exécutées au sein d'une même transaction afin de garantir le succès ou l'échec de l'exécution du code SQL.

Dans une base de données NOSQL, la modification d'un document unique est atomique.
Cependant, il n'y a pas vraiment d'équivalent de la transaction SQL pour les mises à jour de plusieurs documents.

7. Propriétés d’ACID

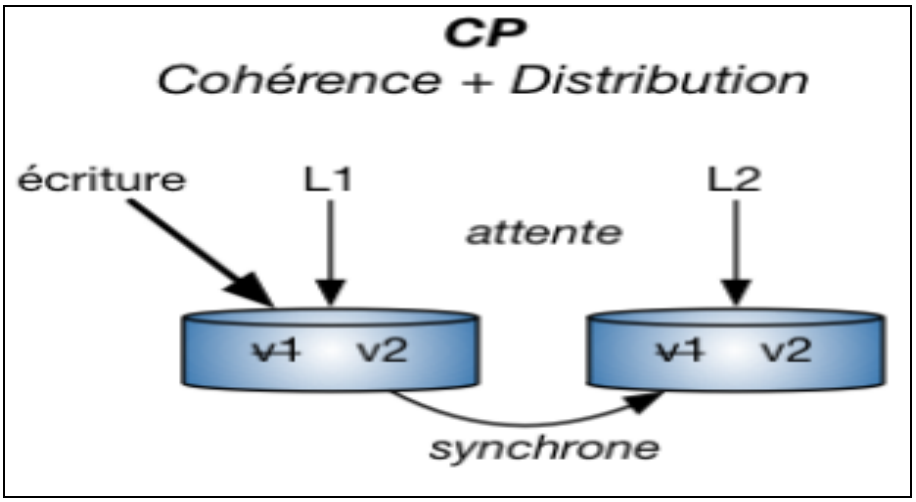
Les propriétés ACID (atomicité, cohérence, isolation et durabilité) sont un ensemble de propriétés qui garantissent qu'une transaction informatique est exécutée de façon fiable, et SQL est conforme avec ces propriétés.

MongoDB est compatible ACID, mais au niveau du document.
Ce que MongoDB ne possède pas, ce sont les transactions, c'est-à-dire les mises à jour de plusieurs documents qui peuvent être annulées et qui sont conformes à l'ACID.

8. Propriétés de CAP

Le théorème CAP (Consistency (Cohérence), Availability (Disponibilité) Partition Tolerance (Distribution)) se repose sur 3 propriétés fondamentales pour caractériser les bases de données (relationnelles, NoSQL et autres).
Dans toute base de données, vous ne pouvez respecter au plus que 2 propriétés.

MongoDB est CP : quand une crise arrive et MongoDB doit décider quoi faire, il choisira la cohérence sur la disponibilité. Il cessera d'accepter les écritures système jusqu'à ce qu'il croit qu'il peut en toute sécurité compléter ces écritures.



9. SQL vs NOSQL dans les Nouvelles Technologies

Le « load balancing » sur un serveur SQL est complexe car dû à la nature même dont les données sont stockées, organisées et reliées entre elles.
Et « clusterer » les instances SQL ont un impact direct sur les licences et le coût de ce type d'infrastructure.

Les modèles de données NOSQL peuvent rendre le processus plus facile et beaucoup d'entre eux ont été conçus nativement avec des fonctionnalités de « Scalabilité Elastique ».
L'organisation des données en documents et la « dé-normalisation » des collections permettent le partitionnement, et le déploiement dans le Cloud est plus simple et facile à gérer, ce qui conduit à une croissance pratiquement illimitée.
Cela implique le principe de base de données distribuée taillée pour la répartition de charge préférant la gestion d'une table gigantesque (i.e. Bigtable de Google) à celle de nombreuses tables interdépendantes (modèle relationnel).
A ce titre, les base de données NOSQL sont résolument orientée "Big Data".

En plus, la représentation des données en collection et le résultat en flux JSON des requêtes permet de consommer les données rapidement et facilement par les applications front (web, mobile).

VII. CONCLUSION

Pour conclure, MongoDB est une base NOSQL, qui propose un syntaxe léger, facile à comprendre, simple à installer et à utiliser, et aussi puissant que les SGBDR comme MySQL ou PostgreSQL.

Par définition de son architecture, Il est facilement scalable, et flexible, et comme il est basé sur Json, MongoDB est une bdd « web scale », qui n'utilise pas du SQL ni des « JOINs », donc très performante.

Il y a une possibilité majeure que MongoDB pourrait être la bdd la plus avantageuse parmi toutes les bases de données à l'avenir, étant déjà le numéro un dans les bdd NOSQL, 5ème parmi toutes les bdd, et reconnu par Gartner et Forester : il est devenu la norme de facto pour les bases de données de nouvelle génération.

MongoDB est une bdd adoptée et combinée de diverses manières, pour répondre à différents besoins, d'où une grande diversité de fonctionnalités, comme le « Big Data » et le « Cloud ».

Aujourd'hui, il y une vaste gamme de cas d'utilisation pour MongoDB, et les grandes entreprises ont adopté MongoDB pour leur bdd primaire, ou comme alternatif et complémentaire pour d'autres besoins.

Parmi ces entreprises,

.

.

Vue unique	Internet des objets	Mobile	Analyses en temps réel
MetLife	ENERNOC	FOURSQUARE	McAfee
CERN	BOSCH	O2	
stripe	Silver Spring	ADP	salesforce
Telefónica	Tech Mahindra	Parse	CRITTERCISM

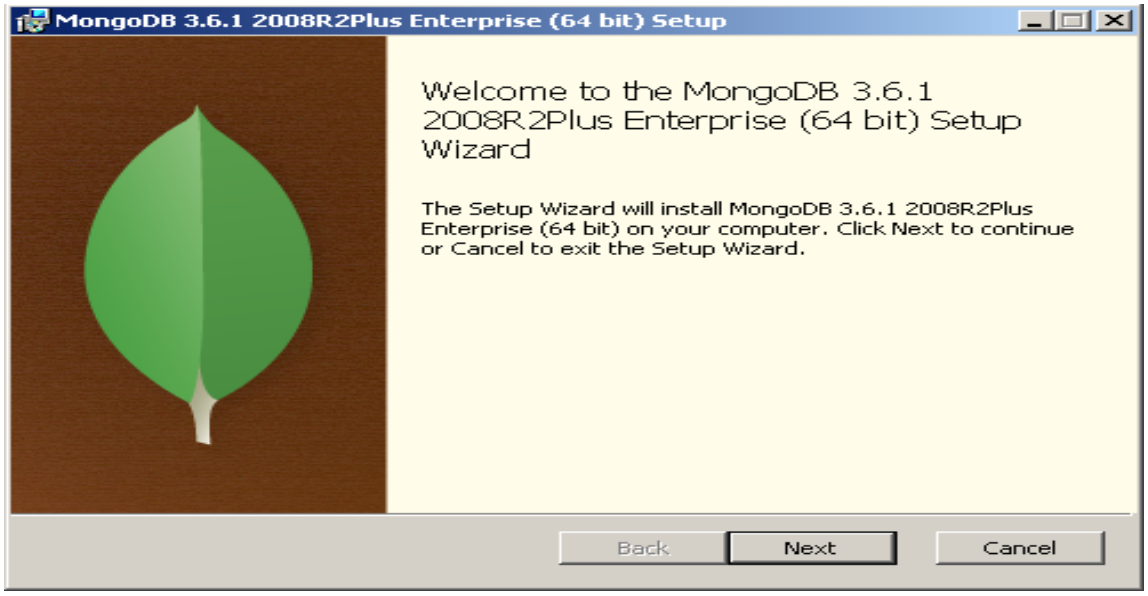
Catalogue	Personnalisation	Gestion de contenus
CARFAX	ticketmaster	VIACOM
UNDER ARMOUR	intuit	Forbes
orange	Expedia	ebay
OTTO	eHarmony	The National Archives

VIII. INSTALLATION DE MONGODB

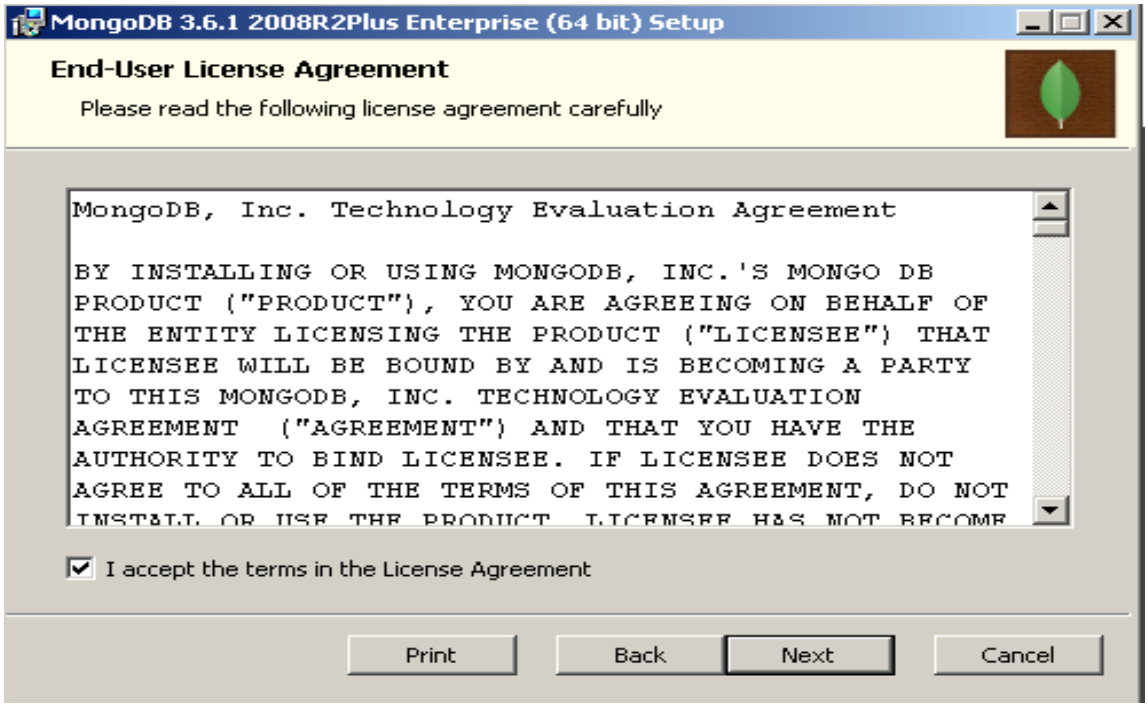
On télécharge « MongoDB Community Server » (version open source et gratuite) de https://www.mongodb.com/download-center#community

Les étapes suivantes sont pour MongoDB Windows :

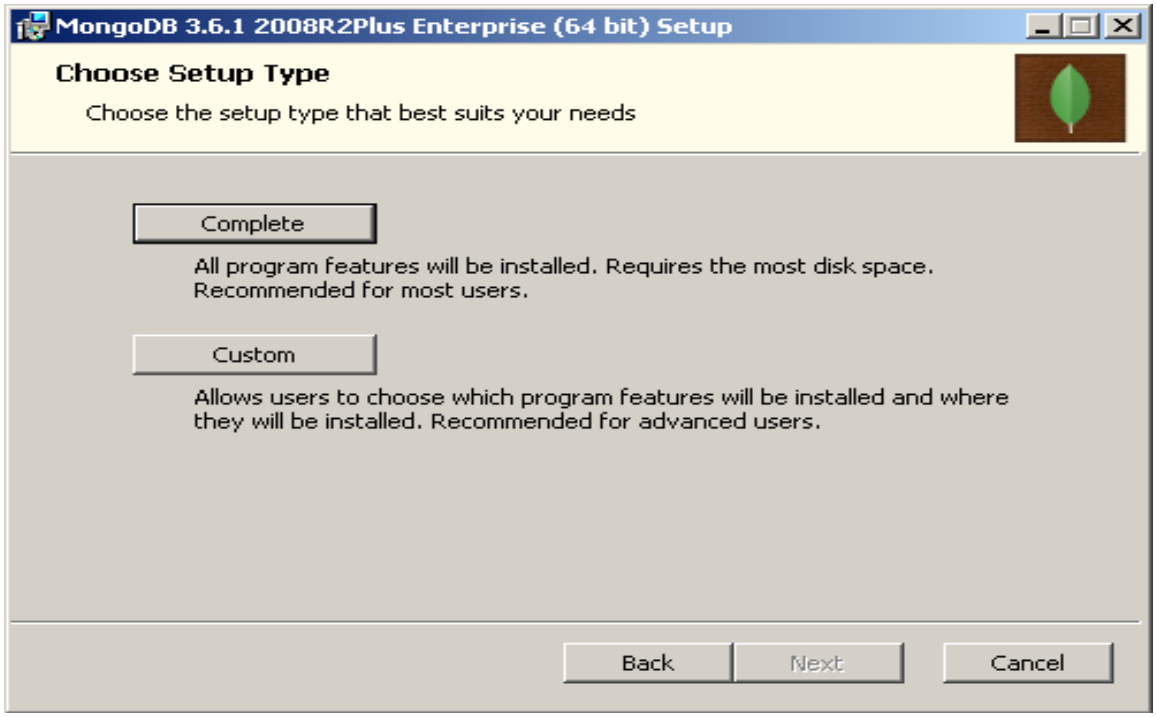
1.



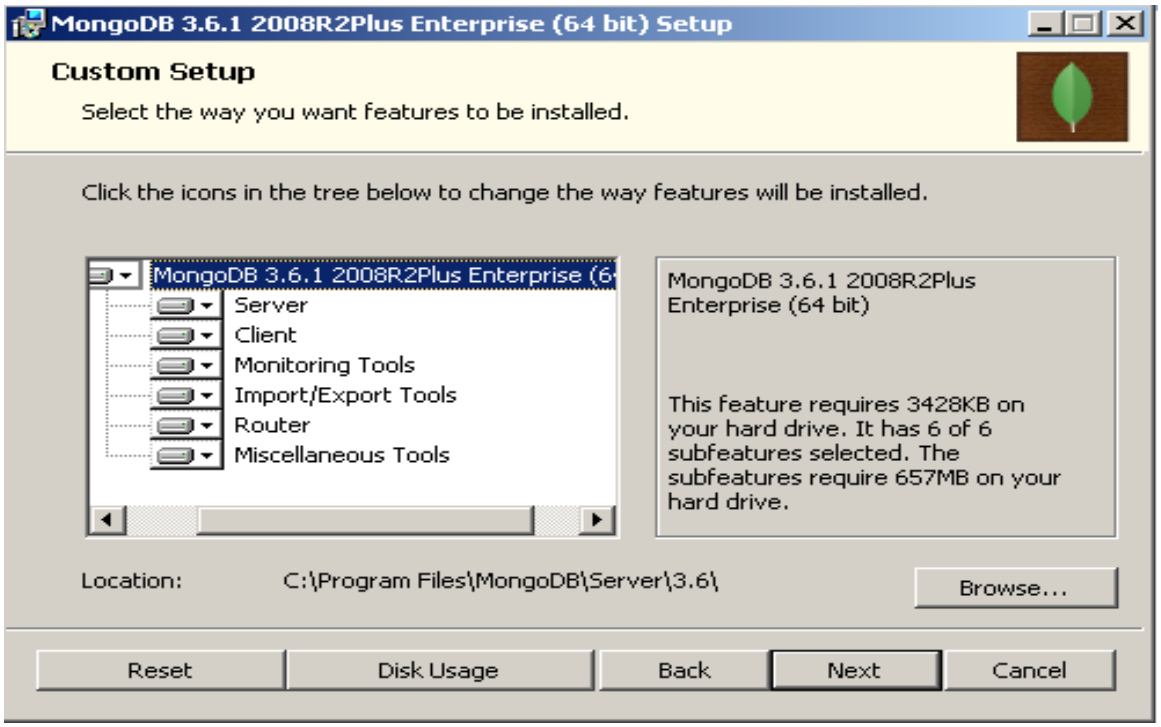
2.



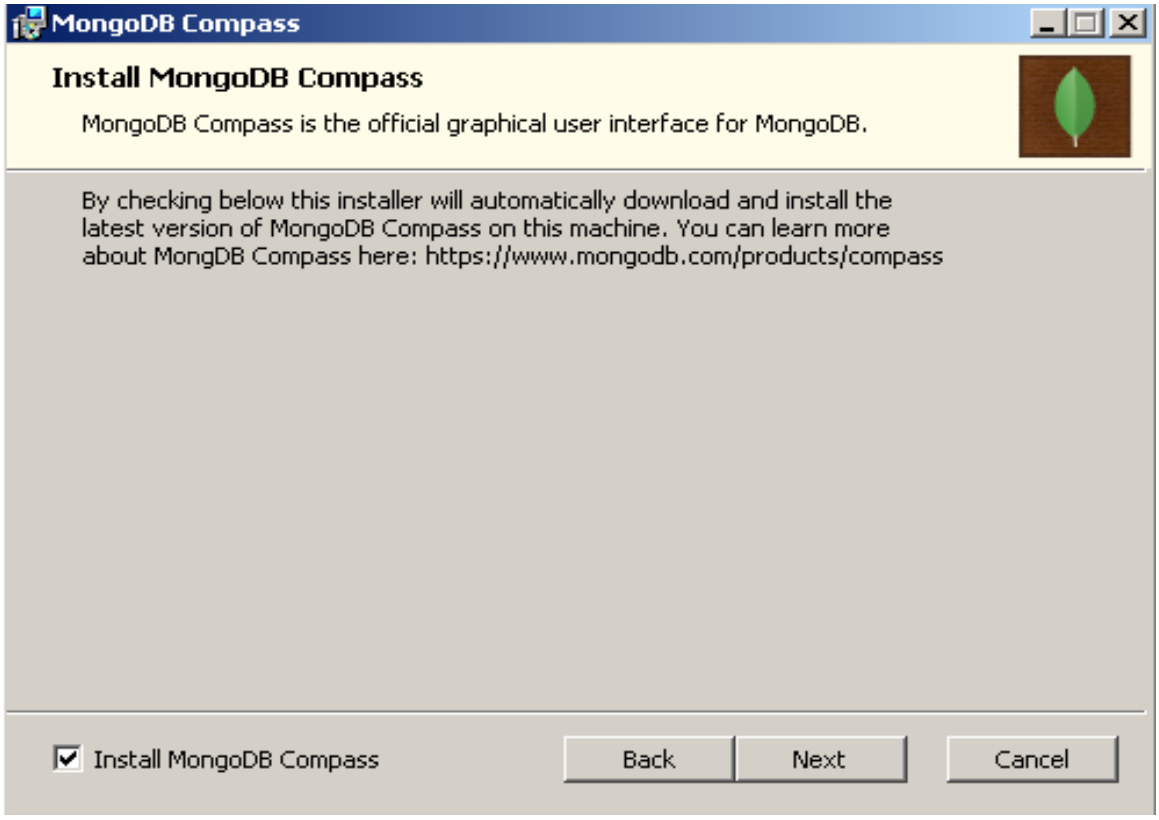
3.


MongoDB 3.6.1 2008R2Plus Enterprise (64 bit) Setup
Choose Setup Type
Choose the setup type that best suits your needs
Complete
All program features will be installed. Requires the most disk space. Recommended for most users.
Custom
Allows users to choose which program features will be installed and where they will be installed. Recommended for advanced users.
Back
Next
Cancel

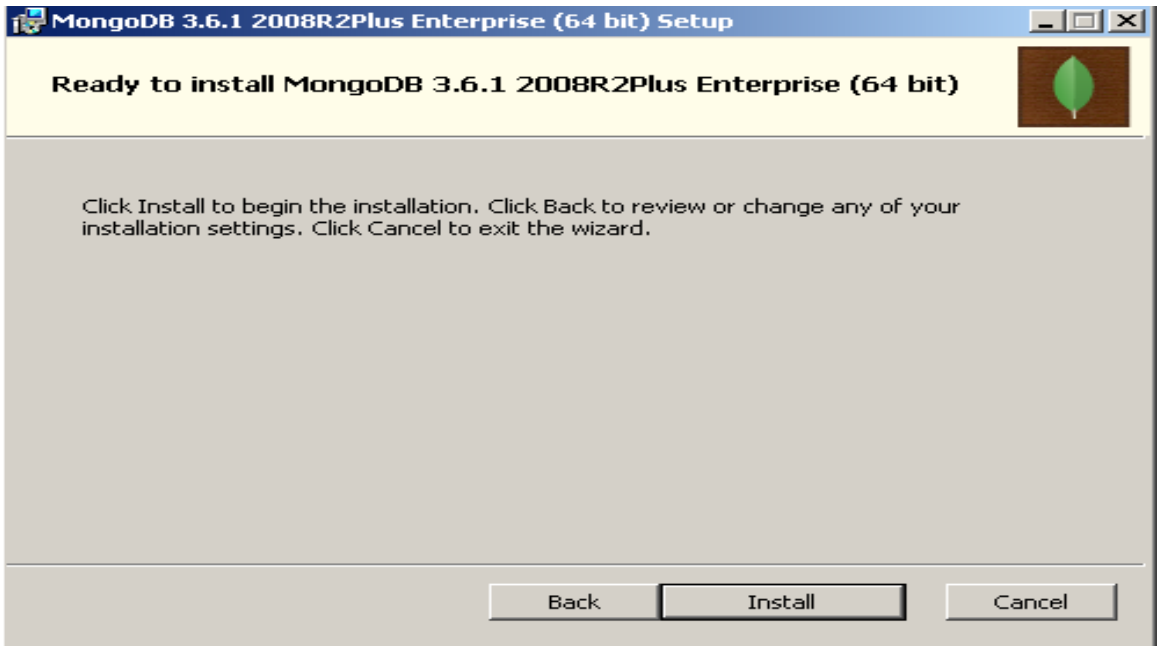
4.


MongoDB 3.6.1 2008R2Plus Enterprise (64 bit) Setup
Custom Setup
Select the way you want features to be installed.
Click the icons in the tree below to change the way features will be installed.
MongoDB 3.6.1 2008R2Plus Enterprise (6
Server
Client
Monitoring Tools
Import/Export Tools
Router
Miscellaneous Tools
MongoDB 3.6.1 2008R2Plus Enterprise (64 bit)
This feature requires 3428KB on your hard drive. It has 6 of 6 subfeatures selected. The subfeatures require 657MB on your hard drive.
Location:
C:\Program Files\MongoDB\Server\3.6\
Browse...
Reset
Disk Usage
Back
Next
Cancel

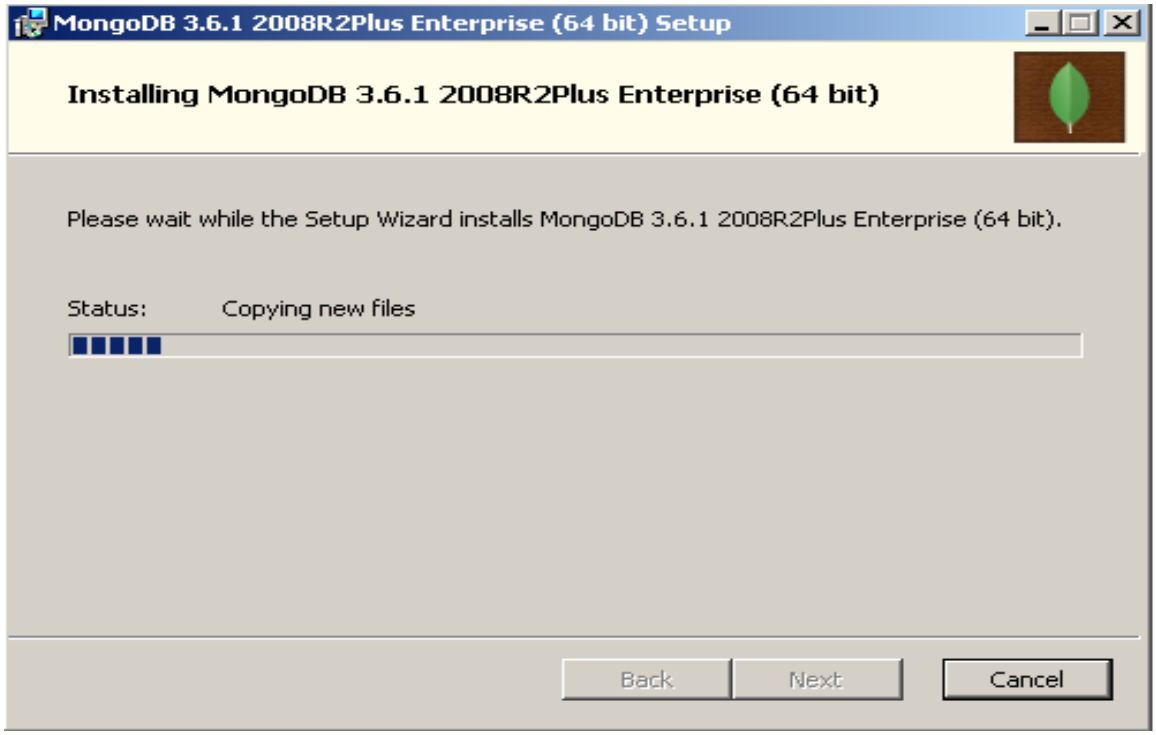
5.


MongoDB Compass
Install MongoDB Compass
MongoDB Compass is the official graphical user interface for MongoDB.
By checking below this installer will automatically download and install the latest version of MongoDB Compass on this machine. You can learn more about MongDB Compass here: https://www.mongodb.com/products/compass
Install MongoDB Compass
Back
Next
Cancel

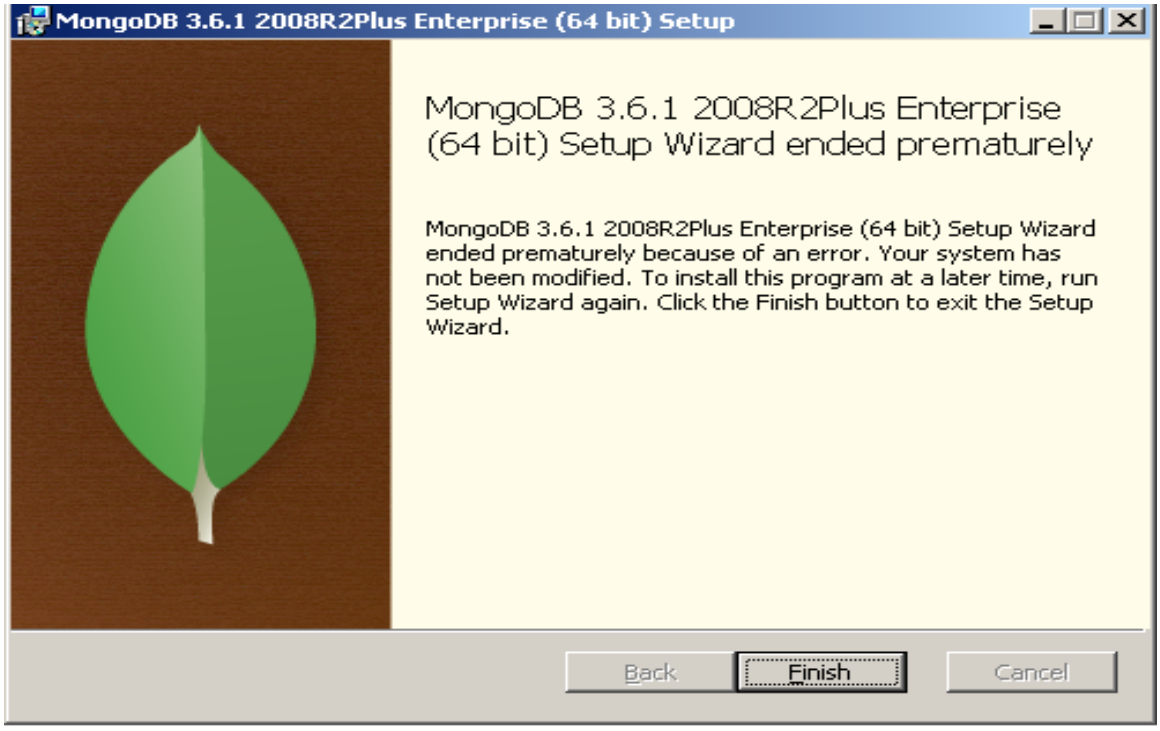
6.


MongoDB 3.6.1 2008R2Plus Enterprise (64 bit) Setup
Ready to install MongoDB 3.6.1 2008R2Plus Enterprise (64 bit)
Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.
Back
Install
Cancel

7.


MongoDB 3.6.1 2008R2Plus Enterprise (64 bit) Setup
Installing MongoDB 3.6.1 2008R2Plus Enterprise (64 bit)
Please wait while the Setup Wizard installs MongoDB 3.6.1 2008R2Plus Enterprise (64 bit).
Status: Copying new files
Back
Next
Cancel

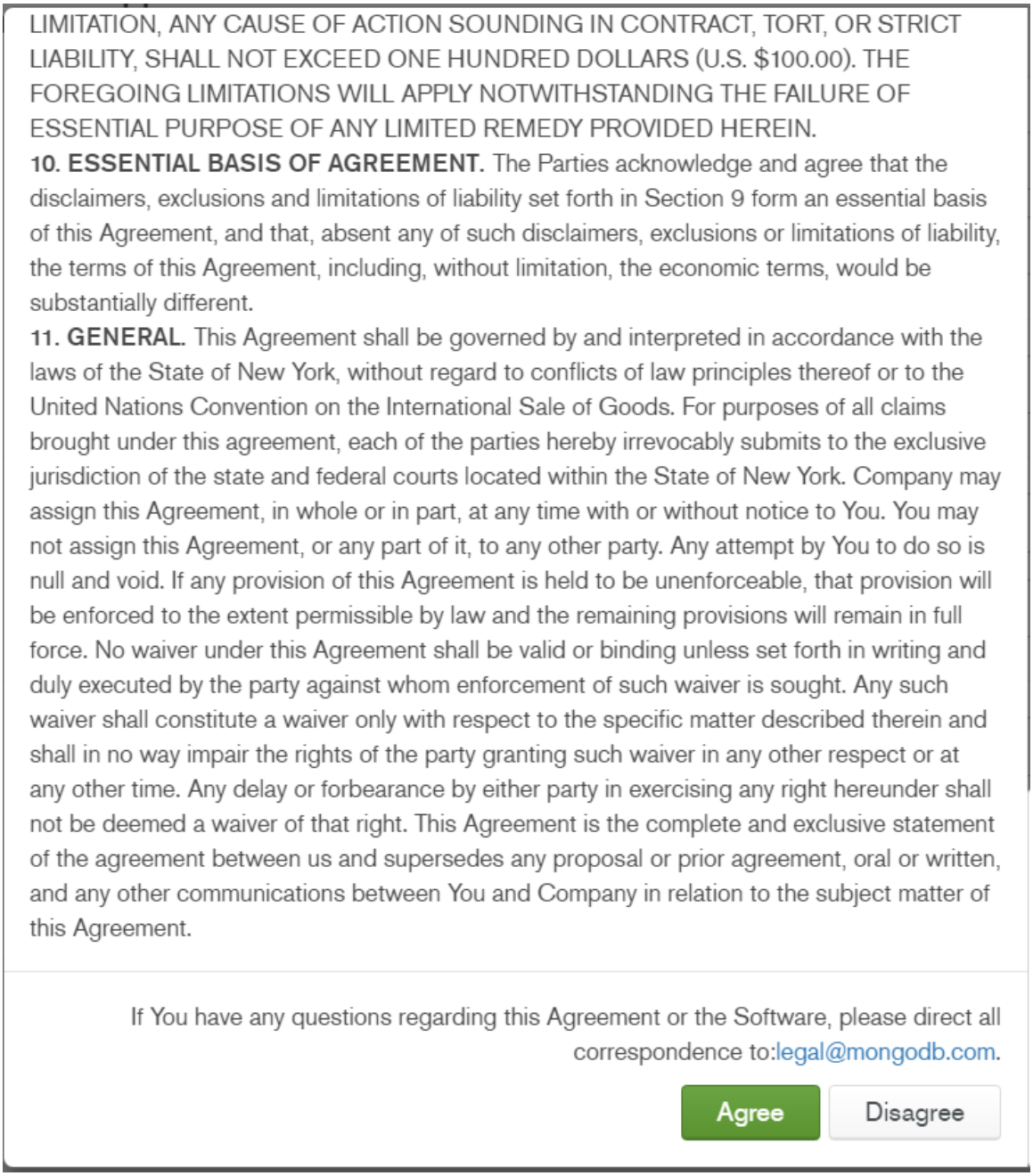
8.


MongoDB 3.6.1 2008R2Plus Enterprise (64 bit) Setup
MongoDB 3.6.1 2008R2Plus Enterprise (64 bit) Setup Wizard ended prematurely
MongoDB 3.6.1 2008R2Plus Enterprise (64 bit) Setup Wizard ended prematurely because of an error. Your system has not been modified. To install this program at a later time, run Setup Wizard again. Click the Finish button to exit the Setup Wizard.
Back
Finish
Cancel

IX. INSTALLATION DE MONGODB COMPASS

(MongoDB Compass, le GUI pour MongoDB)

1.


LIMITATION, ANY CAUSE OF ACTION SOUNDING IN CONTRACT, TORT, OR STRICT LIABILITY, SHALL NOT EXCEED ONE HUNDRED DOLLARS (U.S. $100.00). THE FOREGOING LIMITATIONS WILL APPLY NOTWITHSTANDING THE FAILURE OF ESSENTIAL PURPOSE OF ANY LIMITED REMEDY PROVIDED HEREIN.

10. ESSENTIAL BASIS OF AGREEMENT. The Parties acknowledge and agree that the disclaimers, exclusions and limitations of liability set forth in Section 9 form an essential basis of this Agreement, and that, absent any of such disclaimers, exclusions or limitations of liability, the terms of this Agreement, including, without limitation, the economic terms, would be substantially different.

11. GENERAL. This Agreement shall be governed by and interpreted in accordance with the laws of the State of New York, without regard to conflicts of law principles thereof or to the United Nations Convention on the International Sale of Goods. For purposes of all claims brought under this agreement, each of the parties hereby irrevocably submits to the exclusive jurisdiction of the state and federal courts located within the State of New York. Company may assign this Agreement, in whole or in part, at any time with or without notice to You. You may not assign this Agreement, or any part of it, to any other party. Any attempt by You to do so is null and void. If any provision of this Agreement is held to be unenforceable, that provision will be enforced to the extent permissible by law and the remaining provisions will remain in full force. No waiver under this Agreement shall be valid or binding unless set forth in writing and duly executed by the party against whom enforcement of such waiver is sought. Any such waiver shall constitute a waiver only with respect to the specific matter described therein and shall in no way impair the rights of the party granting such waiver in any other respect or at any other time. Any delay or forbearance by either party in exercising any right hereunder shall not be deemed a waiver of that right. This Agreement is the complete and exclusive statement of the agreement between us and supersedes any proposal or prior agreement, oral or written, and any other communications between You and Company in relation to the subject matter of this Agreement.

If You have any questions regarding this Agreement or the Software, please direct all correspondence to:legal@mongodb.com.

Agree Disagree

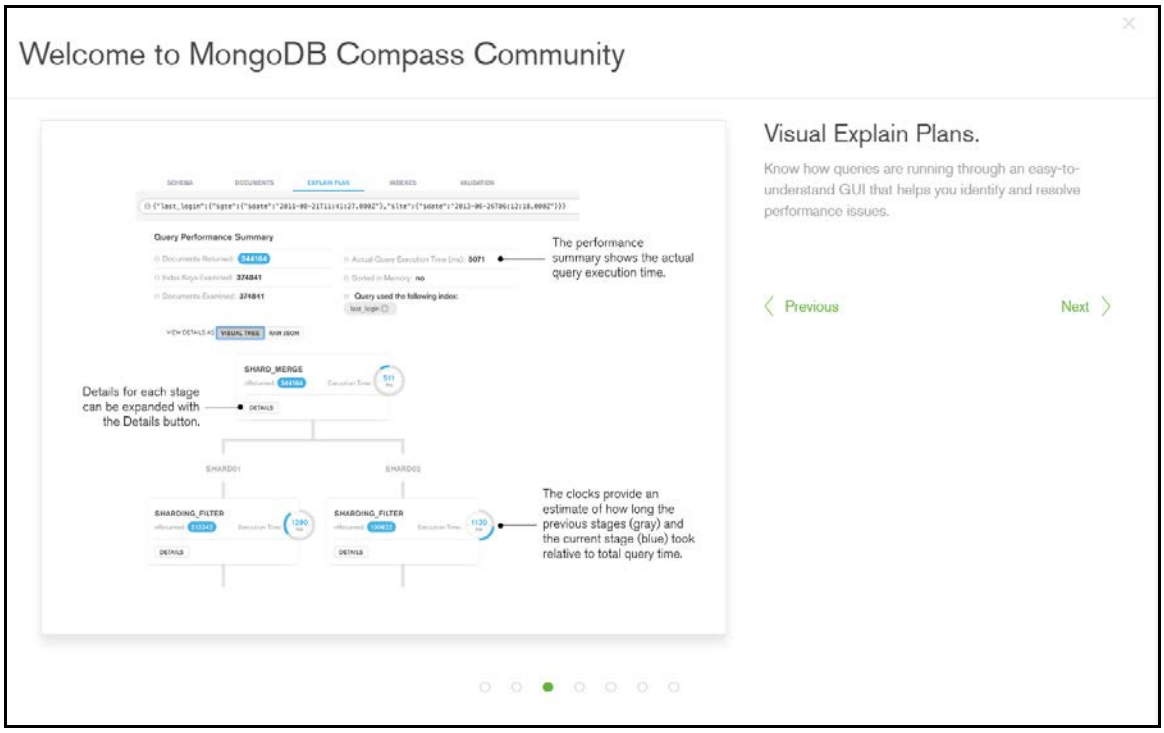
2.


Welcome to MongoDB Compass Community
Click on the instance area to get access to the list of databases and the performance charts.
localhost:27017
Community version 3.4.0-rc5
21 DBS
90 COLLECTIONS
mon|zip|^a
admin
datasets
zipcodes
mongodb
citibike
enterprise
fanclub
Use the filter box to find databases and collections. Regular expressions are supported, too.
Select a database to create and delete collections.
Select a collection to get access to Schema, Documents, Indexes, Explain Plans and Document Validation.
Sidebar. Redesigned.
See with one glance what server you are connected to. Navigate between instance, database and collection level, with powerful filtering of your namespaces.
Next

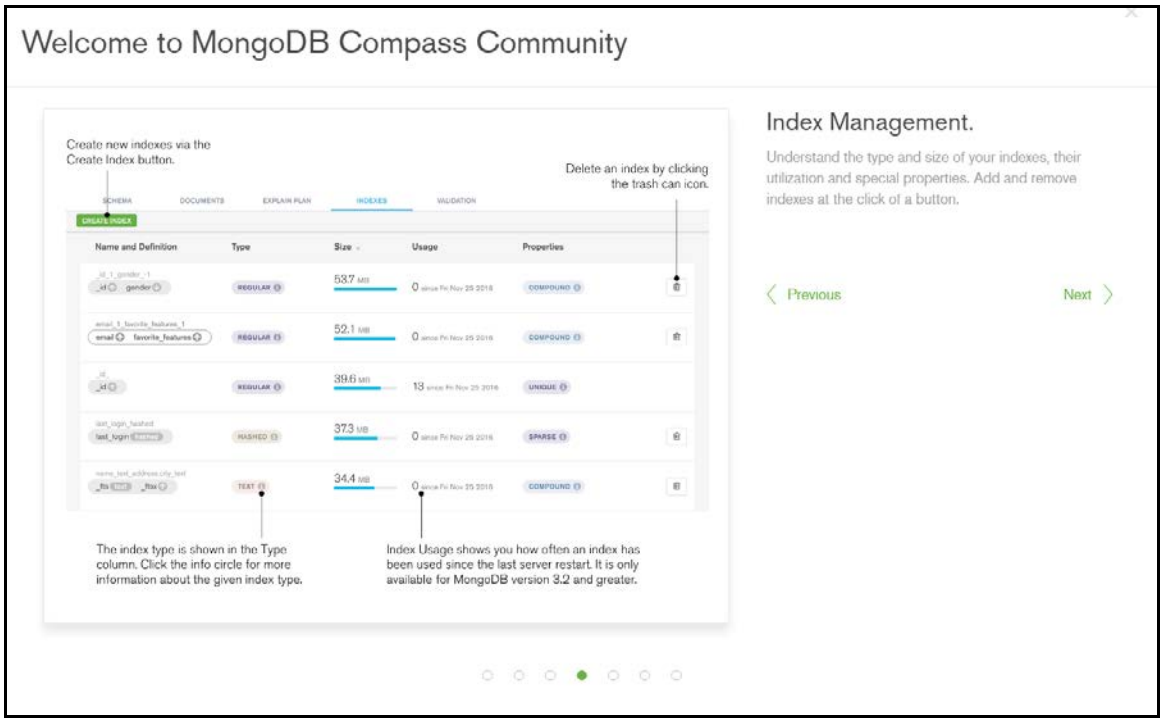
3.


Welcome to MongoDB Compass Community
Click the pen icon in the top right corner of a document to enter edit mode.
You can undo any modifications with the circular arrow icon.
new fields are green, changes are yellow, and deletions are red.
Document Modified.
CANCEL
UPDATE
Dismiss the changes with the Cancel button or send them to the server with the Update button.
Interactive Document Editor.
Modify existing documents with greater confidence using the intuitive visual editor, or insert new documents and clone or delete existing ones in just a few clicks.
Previous
Next

4.


Welcome to MongoDB Compass Community
Query Performance Summary
The performance summary shows the actual query execution time.
Details for each stage can be expanded with the Details button.
The clocks provide an estimate of how long the previous stages (gray) and the current stage (blue) took relative to total query time.
Visual Explain Plans.
Know how queries are running through an easy-to-understand GUI that helps you identify and resolve performance issues.
Previous
Next

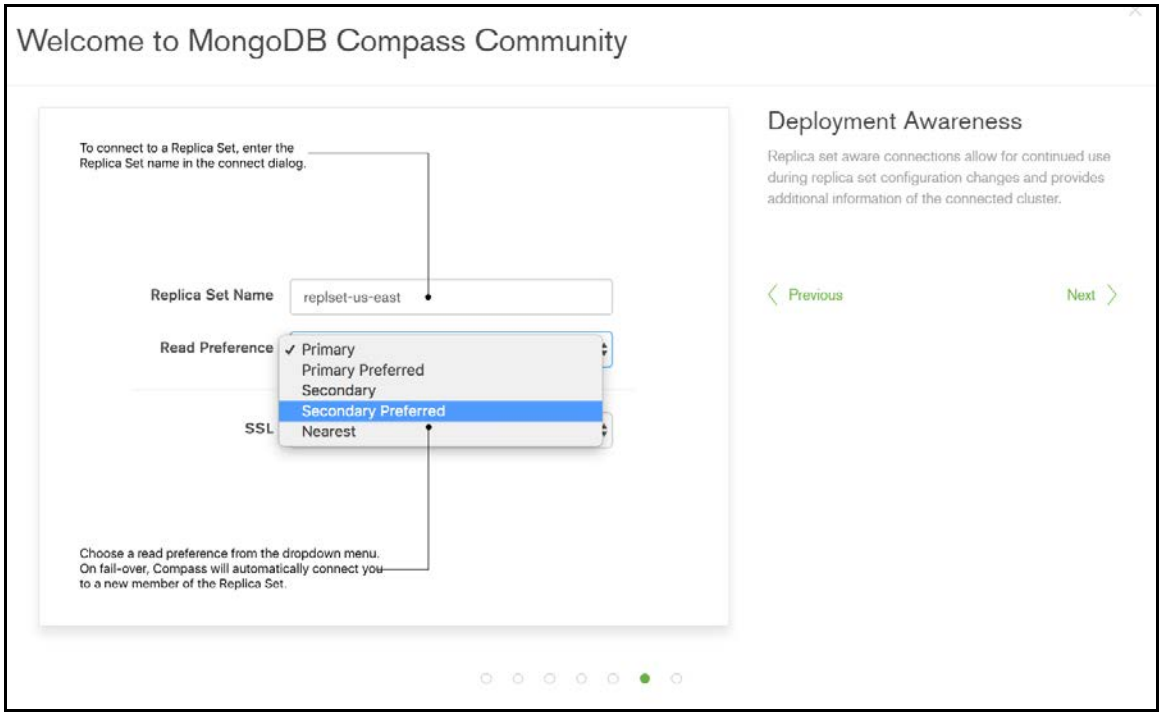
5.


Welcome to MongoDB Compass Community
Create new indexes via the Create Index button.
Delete an index by clicking the trash can icon.
The index type is shown in the Type column. Click the info circle for more information about the given index type.
Index Usage shows you how often an index has been used since the last server restart. It is only available for MongoDB version 3.2 and greater.
Index Management.
Understand the type and size of your indexes, their utilization and special properties. Add and remove indexes at the click of a button.
Previous
Next

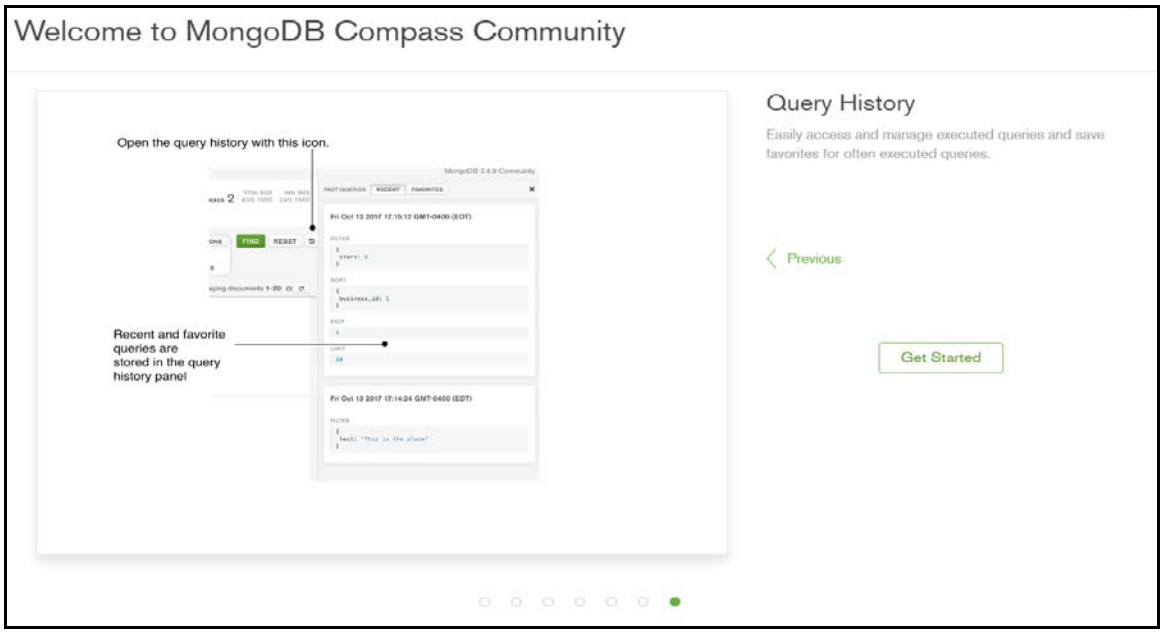
6.


Welcome to MongoDB Compass Community
Values are now validated inline based on their type.
Improved type conversion: pick any type from the dropdown to convert a value to the chosen type.
Value 9999999999 is outside the valid Int32 range.
Update not permitted while document contains errors.
CANCEL
UPDATE
Flexible date inputs: Compass tries to interpret the entered value as a date.
Improved CRUD
Better editing with validation of individual BSON types.
Previous
Next

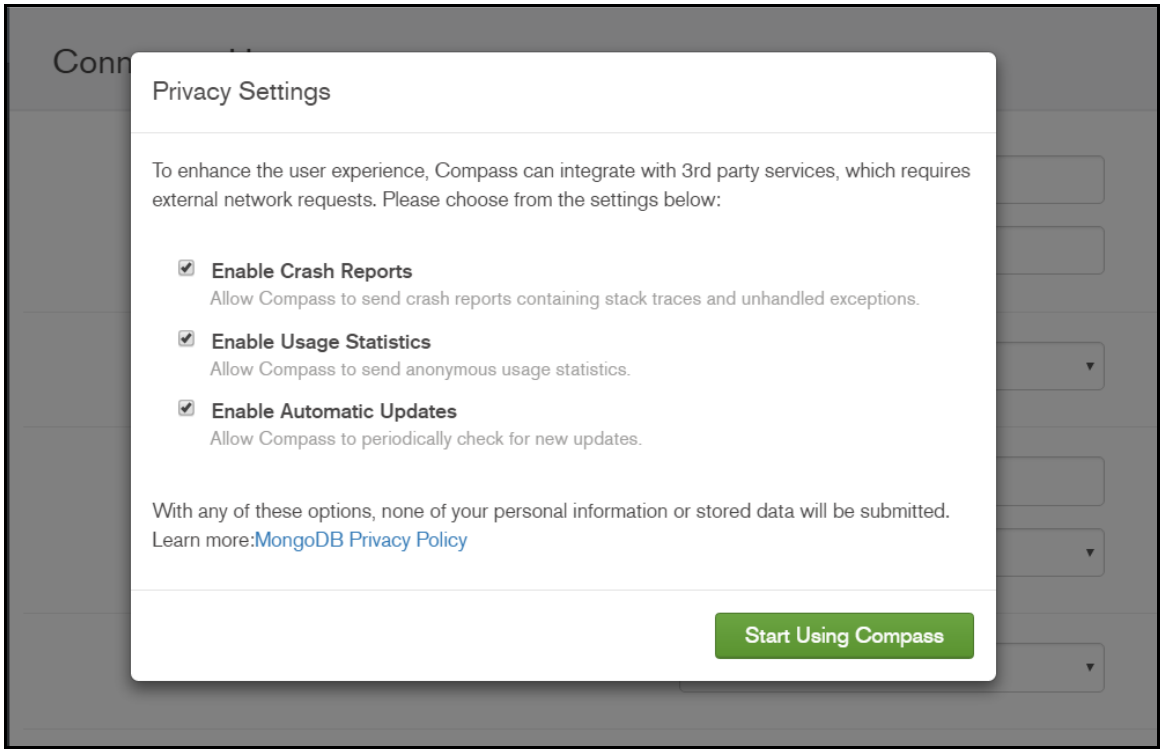
7.


Welcome to MongoDB Compass Community
To connect to a Replica Set, enter the Replica Set name in the connect dialog.
Replica Set Name
replset-us-east
Read Preference
Primary
Primary Preferred
Secondary
Secondary Preferred
Nearest
SSL
Choose a read preference from the dropdown menu. On fail-over, Compass will automatically connect you to a new member of the Replica Set.
Deployment Awareness
Replica set aware connections allow for continued use during replica set configuration changes and provides additional information of the connected cluster.
Previous
Next

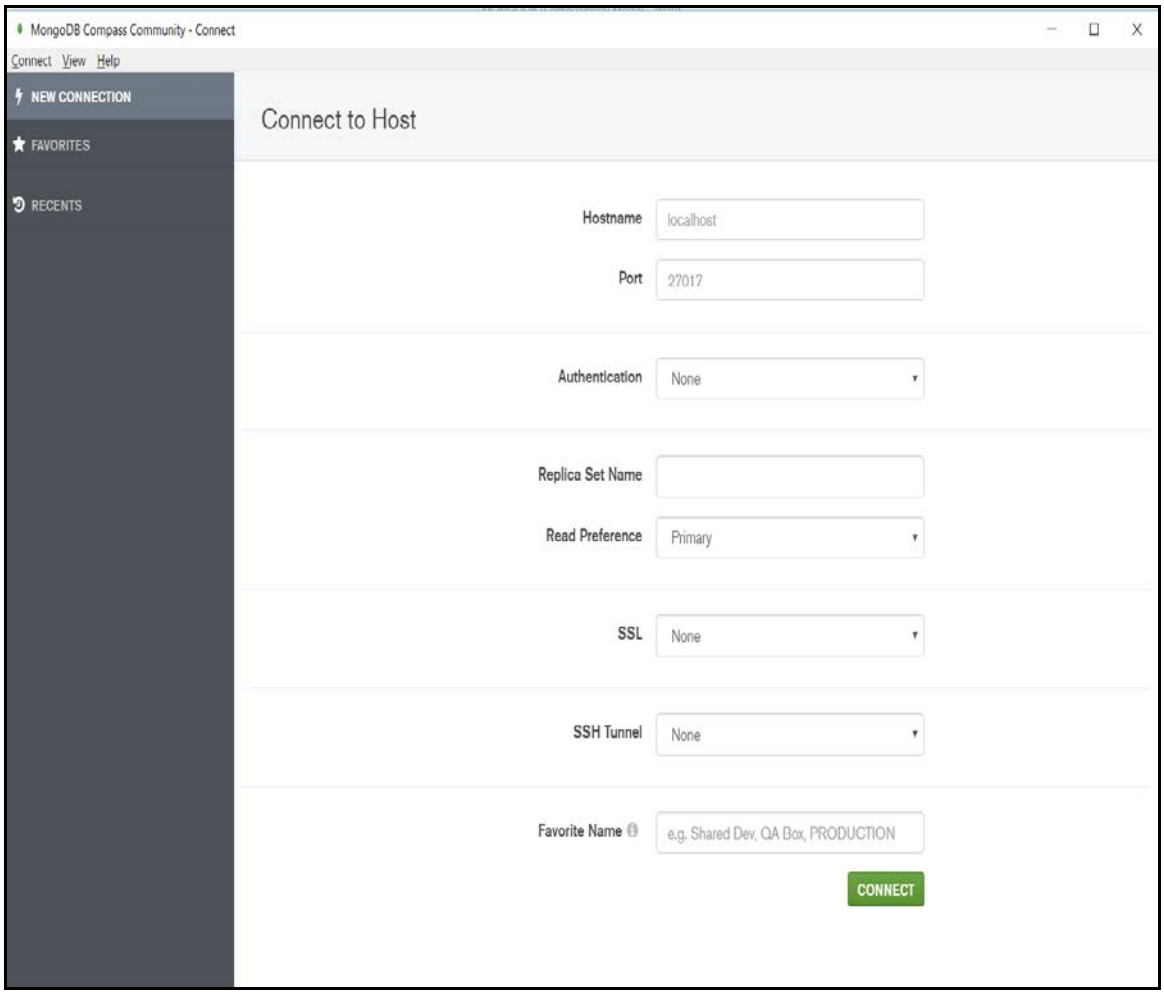
8.


Welcome to MongoDB Compass Community
Open the query history with this icon.
Recent and favorite queries are stored in the query history panel
Query History
Easily access and manage executed queries and save favorites for often executed queries.
Previous
Get Started

9.


Privacy Settings
To enhance the user experience, Compass can integrate with 3rd party services, which requires external network requests. Please choose from the settings below:
Enable Crash Reports
Allow Compass to send crash reports containing stack traces and unhandled exceptions.
Enable Usage Statistics
Allow Compass to send anonymous usage statistics.
Enable Automatic Updates
Allow Compass to periodically check for new updates.
With any of these options, none of your personal information or stored data will be submitted.
Learn more:MongoDB Privacy Policy
Start Using Compass

10.


MongoDB Compass Community - Connect
Connect View Help
NEW CONNECTION
FAVORITES
RECENTS
Connect to Host
Hostname
localhost
Port
27017
Authentication
None
Replica Set Name
Read Preference
Primary
SSL
None
SSH Tunnel
None
Favorite Name
e.g. Shared Dev, QA Box, PRODUCTION
CONNECT

X.COMMNADES DE MONGODB

1. Démarrage de MongoDB

- **Définitions :**
  - ✓ mongod, ou le "daemon Mongo", est essentiellement le processus hôte (host process) pour la base de données.
    Lorsque mongod est exécuté, le processus MongoDB est lance en arrière-plan (background).
    Mongod a plusieurs paramètres par défaut, telle que l'exécution sur le port 27017.

  - ✓ mongo est le shell de ligne de commande qui se connecte à une instance spécifique de mongod.
    Lorsque mongo est lancé sans paramètres, il se connecte par défaut à l'hôte local sur le port 27017.

- **Démarrage :**
  - ✓ Étape 1: On ouvre le cmd (command prompt), et on tape le chemin du « Mongod.exe » :
    . cd c:\Program Files\MongoDB\Server\3.6\bin
    . mongod.exe --dbpath "c:\mongodata"

  - ✓ Étape 2: On ouvre un second cmd pour démarrer le MongoDB :
    . cd c:\Program Files\MongoDB\Server\3.6\bin
    . mongo

2. Commandes Générales

- ✓ **Pour avoir de l'aide**

  db.help()

  db.collection.help()

- ✓ **Pour avoir des statistiques sur la bdd**

  db.stats()

3. Administration de la bdd

- ✓ **Créer une bdd**

  use DATABASE_NAME

  i.e : use mongo_test

- ✓ **Le nom de la bdd**

  db

- ✓ **Liste des bdd**

  show dbs

- ✓ **Jeter une bdd**

  db.dropDatabase()

4. Administration des collections

- ✓ **Créer une Collection**

  db.createcollection(name, options)

  i.e : db.createCollection("test_collection")

- ✓ **Créer une Collection avec INSERT**

  Db.COLLECTION_NAME.insert(name, options)

  i.e.: db.etudiant_collection.insert({
  "etu_code" : "etu01" ,
  "etu_nom" : "Rodney" ,
  "etu_date_nai": ISODate("1978-07-18T00:00:00Z") ,
  "etu_cycle" : 4 });

- ✓ **Le nom des Collections**

  show collections

- ✓ **Jeter une Collection**

  db.COLLECTION_NAME.drop()

  i.e.: db.test_collection.drop();

- ✓ **Retrouver les champs d'une Collection**

  Object.keys(db.rodney.findOne())

5. *Manipulation des documents*

- ✓ **Insérer un document**

  db.COLLECTION_NAME.insert(document)

  i.e.: db.etudiant_collection.insert({
  "etu_code" : "etu02" ,
  "etu_nom" : "JEAN_LUC" ,
  "etu_date_nai": ISODate("1980-01-01T00:00:00Z") ,
  "etu_cycle" : 3 });

- ✓ **Insérer un document #2**

  db.post.save(document)

  i.e.: db.etudiant_collection.insert({
  "etu_code" : "etu03" ,
  "etu_nom" : "JEAN_MARC" ,
  "etu_date_nai": ISODate("1982-01-01T00:00:00Z") ,
  "etu_cycle" : 4 });

  Si on exécute Save avec id, alors elle s'exécute comme upsert, sinon, elle s'exécute comme insert.

- ✓ **Modifier un document**

  db.collection.update(<update>,
  { upsert: <boolean>, multi: <boolean>)

  Par défaut, MongoDB modifie un seul enregistrement, alors que Multi=1 signifie la modification de tous les enregistrements qui ont cette critère.
  Upsert crée l'enregistrement s'il n'existe pas.

i.e : db.etudiant_collection.update ({
({'etu_code' : 'etu_01'}, {$set:{ 'etu_nom' : 'Rodney Badran'}})

i.e : db.etudiant_collection.update({'etu_code' : 'etu_01'}, {$set:{ 'etu_nom' : 'Rodney Badran'}} ,{multi:true})

i.e : db.etudiant_collection.update({'etu_code' : 'etu_05'}, {$set:{ ' etu_nom' : 'JEAN_JACQUES'}},{upsert:true})

- ✓ **Supprimer un document**

db.COLLECTION_NAME.remove(DELLETION_CRITTERIA)

db.COLLECTION_NAME.remove(DELETION_CRITERIA, <1 or all>)

i.e : db.etudiant_collection.remove({etu_code:'etu03'})

i.e : ***supprimer les contenus de la table***

db.etudiant_collection.remove({})

i.e : ***supprimer 1er enregistrement de la table qui vérifie le critère***

db.etudiant_collection.remove({etu_code: 'etu03'}, true)

db.etudiant_collection.remove({etu_code: 'etu03'}, 1)

Manipulation des collections

- ✓ **Requête sur une collection**

  db.COLLECTION_NAME.find()

  i.e. : ***Affichage des documents***

  db.etudiant_collection.find()

  i.e. : ***Affichage formaté de tous les documents***

  db.etudiant_collection.find().pretty()

  i.e. : ***Affichage de tous les documents % à un critère***

  db.etudiant_collection.find({etu_code:"etu_01"}).pretty()

Afficher des documents % à des critères :

LESS THAN	$lt	db.etudiant_collection.find ({"etu_cycle":{$lt:4}}).pretty()
LESS THAN OR EQUAL	$lte	db.etudiant_collection.find ({"etu_cycle ":{$lte:4}}).pretty()
GREATER THAN	$gt	db.etudiant_collection.find ({"etu_cycle ":{$gt:50}}).pretty()
GREATER THAN OR EQUAL	$gte	db.etudiant_collection.find ({"etu_cycle ":{$gte:3}}).pretty()
NOT EQUALS	$ne	db.etudiant_collection.find ({"etu_cycle ":{$ne:3}}).pretty()
AND		db.etudiant_collection.find ({"etu_cycle":4 ,"etu_nom": "JEAN_LUC"}).pretty()
OR		db.etudiant_collection.find ({$or:[{"etu_cycle":1},{"etu_nom": "JEAN_LUC"}]}).pretty()

AND + OR		db.etudiant_collection.find ({ etu_cycle: {$gt:1}, $or: [{etu_nom: 'JEAN_LUC'}, {etu_nom: 'JEAN_JACQUES'}] }).pretty()
Sort	Ordre	db.etudiant_collection.find().sort({cycle: -1})
Limit	Retourne les n 1ers documents	db.etudiant_collection.find().limit(2)
Skip	Saute les n 1ers documents	db.etudiant_collection.find().skip(2)

6. *Fonction Aggregate*

✓ ***Calculer l'agrégat pour des données dans une collection***

db.collection.aggregate(pipeline, options)

i.e : ***Group by and Calculate a Sum***

```
db.livre_collection.insert(
{_id: ObjectId(),
titre: 'MongoDB Overview',
description: 'MongoDB is no sql database',
auteur: 'tutorials point',
url: 'http://www.tutorialspoint.com',
tags: ['mongodb', 'database', 'NOSQL'],
likes: 100 })

db.livre_collection.insert(
{_id: ObjectId(),
titre: 'NOSQL Overview',
description: 'No sql database is very fast',
auteur: 'tutorials point',
url: 'http://www.tutorialspoint.com',
tags: ['mongodb', 'database', 'NOSQL'],
likes: 10 })

db.livre_collection.insert(
{_id: ObjectId(),
titre: 'Neo4j Overview',
description: 'Neo4j is nosql database',
auteur: 'Neo4j',
url: 'http://www.neo4j.com',
tags: ['neo4j', 'database', 'NOSQL'],
likes: 750 })

db.livre_collection.aggregate(
   [
     { $group : { _id : "$auteur", total: { $sum: 1 } } }
   ]
)
```

Expression	Description	Example
$sum	Sums up the defined value from all documents in the collection.	db.livre_collection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$sum : "$likes"}}}])
$avg	Calculates the average of all given values from all documents in the collection.	db.livre_collection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$avg : "$likes"}}}])
$min	Gets the minimum of the corresponding values from all documents in the collection.	db.livre_collection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$min : "$likes"}}}])
$max	Gets the maximum of the corresponding values from all documents in the collection.	db.livre_collection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$max : "$likes"}}}])
$push	Inserts the value to an array in the resulting document.	db.livre_collection.aggregate([{$group : {_id : "$by_user", url : {$push: "$url"}}}])
$addToSet	Inserts the value to an array in the resulting document but does not create duplicates.	db.livre_collection.aggregate([{$group : {_id : "$by_user", url : {$addToSet : "$url"}}}])
$first	Gets the first document from the source documents according to the grouping. Typically this makes only sense together with some previously applied “$sort”- stage.	db.livre_collection.aggregate([{$group : {_id : "$by_user", first_url : {$first : "$url"}}}])
$last	Gets the last document from the source documents according to the grouping. Typically this makes only sense together with some previously applied “$sort”- stage.	db.livre_collection.aggregate([{$group : {_id : "$by_user", last_url : {$last : "$url"}}}])

7. Curseur

✓ **Iteration des doneness d'une collection**

```
db.collection.find().forEach(<function>)
```

or

```
var cursor = db.collection.find();
cursor.forEach(function)
```

i.e.: db.etudiant_collection.find().forEach(function(cur) { print("nom: " + cur.etu_nom); });

i.e.: var myCursor = db.etudiant_collection.find();

i.e.:

```
var myCursor = db.livre_collection.find();
while (myCursor.hasNext()) {
print(tojson(myCursor.next()));
}
db.livre_collection.find().close()
```

Pas besoin de faire « close curseur » si on va itérer toutes les données. MongoDb fermera automatiquement les curseurs qui n'ont pas de résultats restants, ainsi que les curseurs inactifs depuis un certain temps.

8. Index

- ✓ **Créer un index sur une collection**

  db.collection.createIndex(keys, options)

  i.e.: db.livre_collection.createIndex({"titre":1})

  i.e.: db.livre_collection.createIndex({"titre":1,"description":-1}, { unique: false,name:"abc_index"})

  1:ascending / -1:descending

- ✓ **Créer plusieurs index sur une collection**

  db.collection.createIndexes([keyPatterns,]options)

  i.e.: db.livre_collection.createIndexes([{"titre": 1}, {"description": -1}])

- ✓ **Trouver les index d'une collection**

  db.collection.getIndexes()

- ✓ **Jeter un index**

  db.collection.dropIndex(index_name)

9. *Etapes Pour Faire La Réplication*

. On arrête le serveur de MongoDB.

. On démarre le serveur de MongoDB en spécifiant -- replSet option.

. mongod --port "PORT" --dbpath "chemin_bdd" –replSet "REPLICA_SET_INSTANCE_NAME"

	mongod --port 27017 --dbpath "D:\set up\mongodb\data" --replSet rs0
initiate a new replica set	rs.initiate()
check the replica set configuration	rs.conf()
check the status of replica	rs.status()
Add Members to Replica Set	rs.add(HOST_NAME:PORT) rs.add("mongod1.net:27017")
check whether you are connected to primary or not	db.isMaster()

10. Téléchargement (Import) des fichiers

Pour télécharger des fichiers csv, txt ou json

On ouvre le cmd (command prompt), et on tape le chemin du « mongoimport.exe » :

cd C:\Program Files\MongoDB\Server\3.6\bin

i.e : Télécharger un fichier csv où :
la table est prédéfinie,
la première ligne indique les noms des colonnes (à sauter)
mongoimport --db isae --type csv --collection etudiantimport --headerline --file C:\mongodata\importfiles\impetud.csv

i.e : Télécharger un fichier csv où :
la collection n'existe pas, et l'import va la créer du nom de l'Excel
la première ligne indique les noms des colonnes (à sauter)
mongoimport --db isae --type csv --headerline --file C:\mongodata\importfiles\impetud.csv

i.e : Télécharger un fichier json où l'import va jeter la table avant de créer la collection de nouveau et importer les données en même temps:
mongoimport --db isae --collection restaurants --drop --file C:\mongodata\importfiles\Restaurant.json

11. *MongoDB et Java*

- ✓ **Pre-requis**

  Télécharger “mongodb-driver” du site suivant:
  https://oss.sonatype.org/content/repositories/releases/org/mongodb/mongodb-driver/3.4.3/

- ✓ On ajoute la dépendance suivante dans maven :

```
<dependencies>
   <dependency>
      <groupId>org.mongodb</groupId>
      <artifactId>mongodb-driver</artifactId>
      <version>3.4.3</version>
   </dependency>
</dependencies>
```

- ✓ On ajoute les « import » suivant dans notre code java :

```
import com.mongodb.MongoClient;
import com.mongodb.MongoClientURI;
import com.mongodb.ServerAddress;

import com.mongodb.client.MongoDatabase;
import com.mongodb.client.MongoCollection;

import org.bson.Document;
import java.util.Arrays;
import com.mongodb.Block;

import com.mongodb.client.MongoCursor;
import static com.mongodb.client.model.Filters.*;
import com.mongodb.client.result.DeleteResult;
import static com.mongodb.client.model.Updates.*;
import com.mongodb.client.result.UpdateResult;
import java.util.ArrayList;
import java.util.List;
```

✓ Exemple:

```
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import org.bson.Document;
import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;

public class mongo_java_test {

   public static void main( String args[] ) {

      // Creating a Mongo client
      MongoClient mongo = new MongoClient( "localhost" , 27017 );


      // Creating Credentials
      MongoCredential credential;
      credential = MongoCredential.createCredential("sampleUser", "myDb",
        "password".toCharArray());
      System.out.println("Connected to the database successfully");


      // Accessing the database
      MongoDatabase database = mongo.getDatabase("myDb");
       System.out.println("Credentials ::"+ credential);


      // Creating a collection
     database.createCollection("sampleCollection");
     System.out.println("Collection created successfully");


      // Retieving a collection
      MongoCollection<Document> collection =
                database.getCollection("myCollection");
      System.out.println("Collection myCollection selected successfully");

     // Insert 1 document
     Document document = new Document("titre", "MongoDB")
      .append("id", 1)
      .append("description", "database")
      .append("likes", 100)
      .append("url", "http://www.tutorialspoint.com/mongodb/")
      .append("by", "tutorials point");
```

```
collection.insertOne(document);
System.out.println("Document inserted successfully");

// Insert plusieurs documents
Document doc1 = new Document("name", "Amarcord Pizzeria")
     .append("contact", new Document("phone", "264-555-0193")
     .append("email", "amarcord.pizzeria@example.net")
     .append("location",Arrays.asList(-73.88502, 40.749556)))
     .append("stars", 2)
     .append("categories", Arrays.asList("Pizzeria", "Italian", "Pasta"));


 Document doc2 = new Document("name", "Blue Coffee Bar")
     .append("contact", new Document("phone", "604-555-0102")
     .append("email", "bluecoffeebar@example.com")
     .append("location",Arrays.asList(-73.97902, 40.8479556)))
     .append("stars", 5)
     .append("categories", Arrays.asList("Coffee", "Pastries"));

  List<Document> documents = new ArrayList<Document>();
  documents.add(doc1);
  documents.add(doc2);
  collection.insertMany(documents);


// Trouver tous les documents
FindIterable<Document> iterDoc = collection.find();
int i = 1;

Iterator it = iterDoc.iterator();

while (it.hasNext()) {
   System.out.println(it.next());
 i++;
 }


// Lire avec filtre
myDoc = collection.find(eq("i", 71)).first();
System.out.println(myDoc.toJson());
```

```
// Modifier un document
collection.updateOne(Filters.eq("id", 1), Updates.set("likes", 150));
 System.out.println("Document update successfully...");


// Modifier plusieurs documents
collection.updateMany(eq("stars", 2),combine(set("stars", 0),
currentDate("lastModified")));


// Supprimer un document
collection.deleteOne(Filters.eq("id", 1));
System.out.println("Document deleted successfully...");


// Supprimer plusieurs documents
collection.deleteMany(eq("stars", 4));
System.out.println("Documents deleted successfully...");


// Jeter une collection
MongoCollection<Document> collection =
database.getCollection("sampleCollection");
collection.drop();
System.out.println("Collection dropped successfully");


// Afficher toutes les collections
for (String name : database.listCollectionNames())
System.out.println(name);


// Ecriture en lot
collection.bulkWrite(
  Arrays.asList(new InsertOneModel<>(new Document("_id", 4)),
              new InsertOneModel<>(new Document("_id", 5)),
              new InsertOneModel<>(new Document("_id", 6)),
              new UpdateOneModel<>(new Document("_id", 1),
                                  new Document("$set", new Document("x", 2))),
              new DeleteOneModel<>(new Document("_id", 2)),
              new ReplaceOneModel<>(new Document("_id", 3),
                                   new Document("_id", 3).append("x", 4))));

          }
        }
```