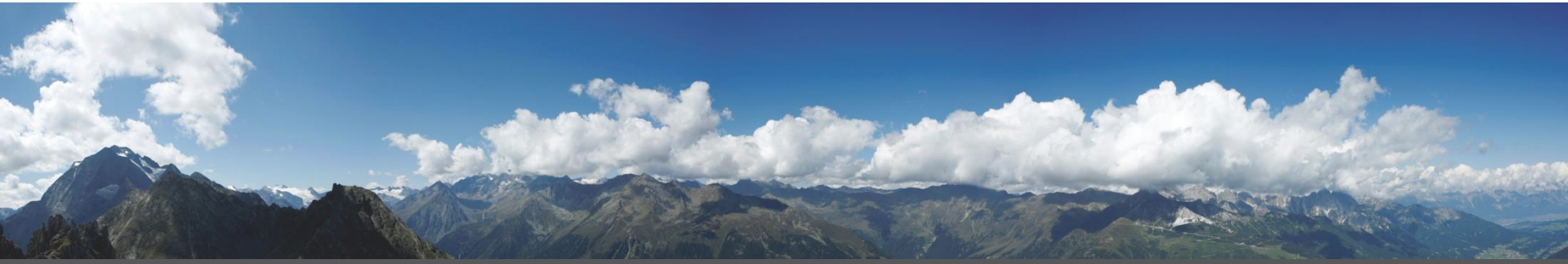
Chapitre 04 : NoSQL & mongoDB



Sommaire

- **Motivations NoSQL & Historique**
- **SQL vs NoSQL**
- **Présentation de MongoDB**
- **Réplication/Haute disponibilité**
- **Partage (Sharding) /Mise à l'échelle (Scaling)**
- **Opérations de base**

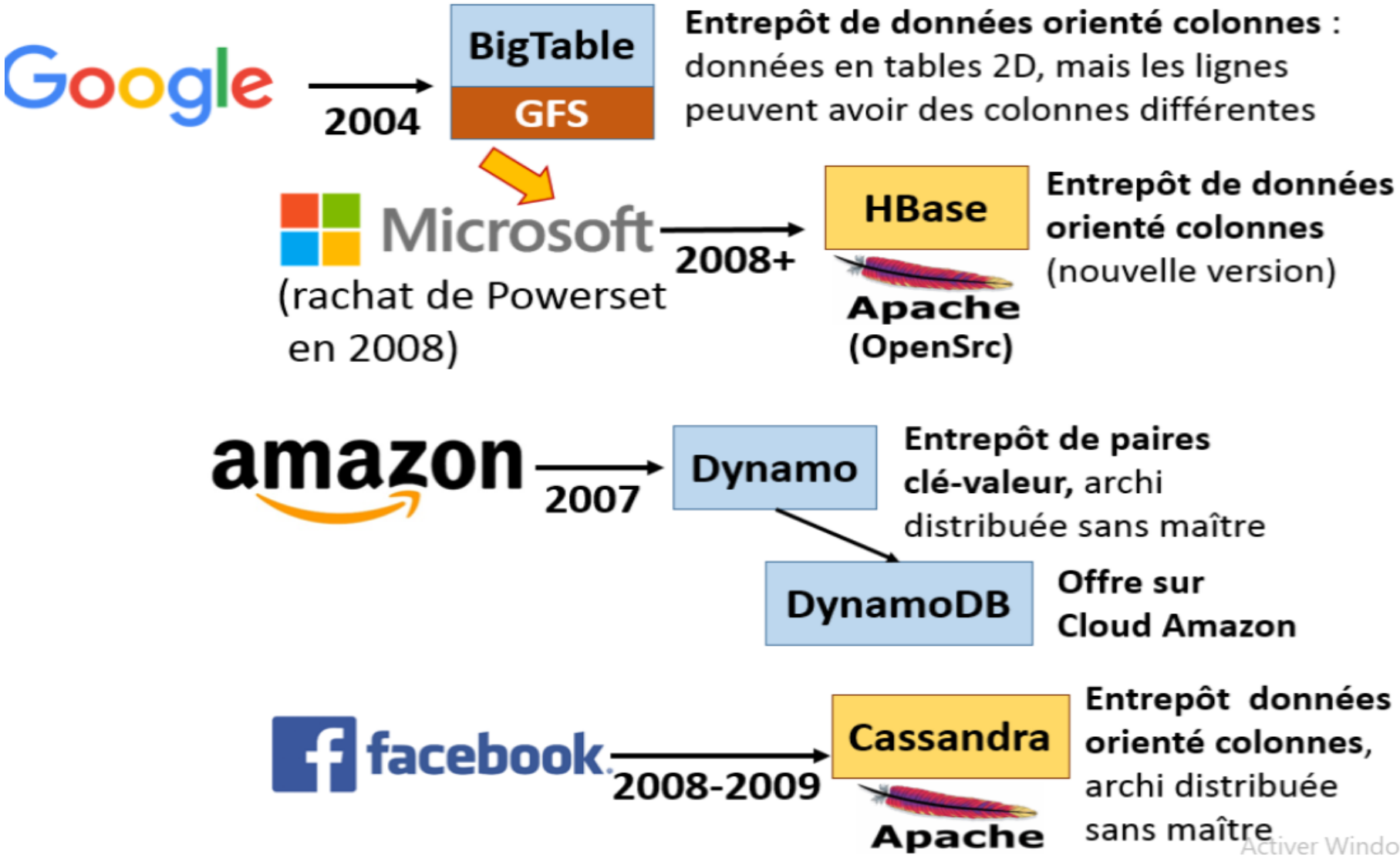
Motivations NoSQL

Contraintes du modèle relationnel (arrivée 1969-1970) :

- Toutes les **lignes** d'une Relation ont les mêmes **colonnes** (*NULL* si absence de données)
- Modifications **atomiques** pour que la **BdD** soit toujours totalement cohérente
- Interrogation par ***jointures*** de nombreuses (petites) Relations
- Toutes les données doivent respecter le **schéma initial**…
- Le langage Relationnel SQL est adapté pour **extraire** des informations selon des **conditions** sur leurs **valeurs**
- Requêtes **OLTP** (On Line Transaction Processing) : **OK** …mais n'est pas adapté pour faire des calculs **de statistiques complexes** sur ces valeurs, ni sur des données **volumineuses**!
- Requêtes **OLAP** (On Line Analytical Processing) : **problème…**

Historique:

Les premières BdD NoSQL



SQL vs NoSQL

- ✓ **Base de données *NoSQL* (souvent interprétée comme Not only SQL)**
- ✓ Elle fournit un mécanisme de **stockage** et de **récupération** de données modélisées par d'autres moyens que les **relations tabulaires** utilisées dans les bases de données relationnelles.

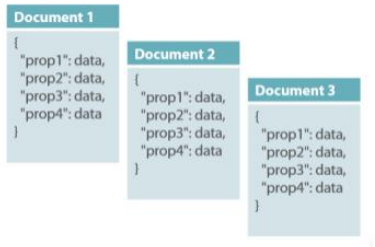
SQL	NoSQL
Système de gestion de base de données relationnelle (SGBDR)	**Système de base de données non relationnelle ou distribuée.**
Elles ont un schéma fixe, statique ou prédéfini	**Elles ont un schéma dynamique**
Elles sont les mieux adaptées aux requêtes complexes	**Elles Ne sont pas très adaptées aux requêtes complexes**
Évolutivité vertical (Vertically Scalable)	**Évolutivité horizontal (Horizontally scalable)**

Types de NoSQL

Graph database

Neo4j the graph database

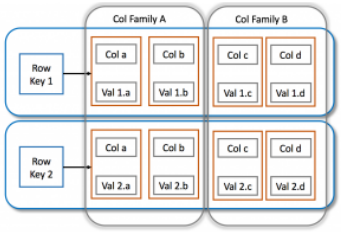
HyperGraphDB



Document-oriented

mongoDB

CouchDB relax



Column family

Cassandra

HBASE

Présentation de MongoDB

- ***MongoDB*** est une base de données open source **orientée documents**, conçue dans un souci d'**évolutivité** et d'**agilité** pour les développeurs..

- Au lieu de stocker vos données dans des **tables** et des **lignes** comme vous le feriez avec une base de données relationnelle, dans **MongoDB**, vous stockez des documents de type ***JSON*** avec des **schémas dynamiques**.

```
{
    "_id" : ObjectId("5114e0bd42…"),
    “FirstName" : "John",
    “LastName" : "Doe",
    “Age" : 39,
   “Interests" : [ "Reading",  "Mountain Biking ]
   “Favorites": {
                "color": "Blue",
                      "sport": "Soccer“
                        }
 }
```

Présentation de MongoDB

- Base de données NoSQL **orientée document**.
- **Sans schéma**.
- Basée sur ***JSON binaire*** ; **BSON**[2].
- Organisée en **groupe de documents → collections**
- **Auto-Sharding** afin de mettre à l'échelle **horizontalement**.
- Prise en charge de ***Map/Reduce***
- **Open Source** (GNU AGPL v3.0.)

MongoDB est facile à utiliser

Relational

Person:

Pers_ID	Surname	First_Name	City
0	Miller	Paul	London
1	Ortega	Alvaro	Valencia
2	Huber	Urs	Zurich
3	Blanc	Gaston	Paris
4	Bertolini	Fabrizio	Rom

— no relation

Car:

Car_ID	Model	Year	Value	Pers_ID
101	Bentley	1973	100000	0
102	Rolls Royce	1965	330000	0
103	Peugeot	1993	500	3
104	Ferrari	2005	150000	4
105	Renault	1998	2000	3
106	Renault	2001	7000	3
107	Smart	1999	2000	2

MongoDB Document

```
{
  first_name: 'Paul',
  surname: 'Miller'
  city: 'London',
  location: [45.123,47.232],
  cars: [
   { model: 'Bentley',
     year: 1973,
     value: 100000, … },
   { model: 'Rolls Royce',
     year: 1965,
     value: 330000, … }
  ]
}
```

SGBDR vs MongoDB

SGBDR		MongoDB
Database	⇒	Database
Table	⇒	Collection
Row	⇒	Document (JSON, BSON)
Column	⇒	Field
Index	⇒	Index
Join	⇒	Embedded Document
Partition	⇒	Shard

Schéma Libre

MongoDB n'a pas besoin de schéma de données **prédéfini**

Chaque **document** peut avoir des **données différentes** !

Exemple:

```
{name: “will”,
  eyes: “blue”,
  birthplace: “NY”,
  aliases: [“bill”,
“ben”],
  loc: [32.7, 63.4],
  boss: ”ben”}
```

```
{name: “jeff”,
 eyes: “blue”,
 loc: [40.7, 73.4],
 boss: “ben”}
```

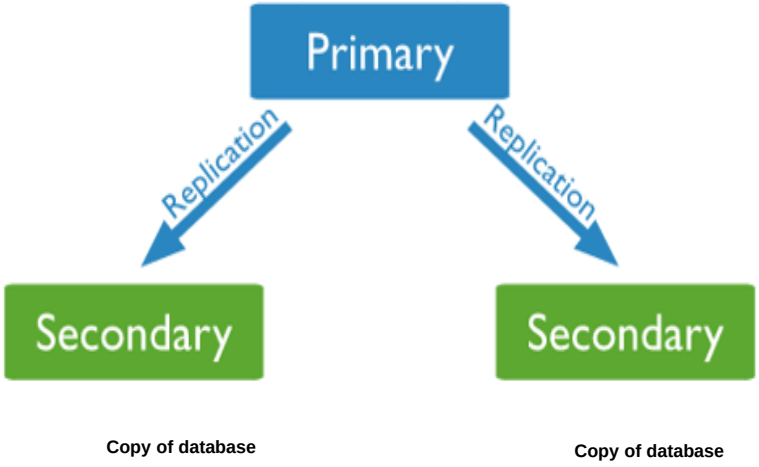
```
{name: “ben”,
  age:25}
```

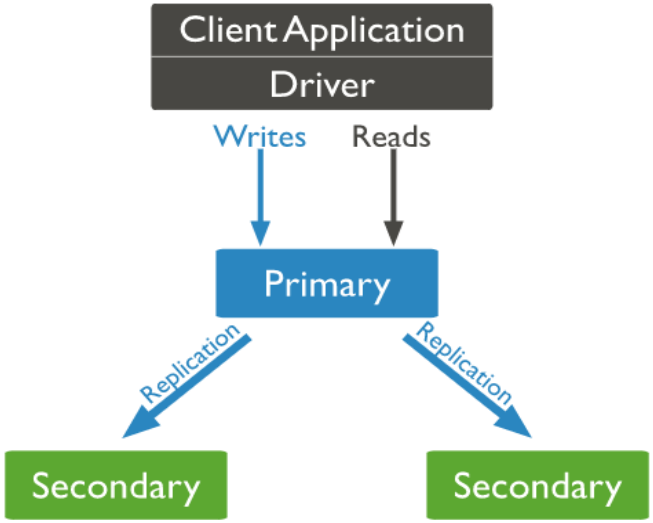
```
{name: “brendan”,
  boss: “will”}
```

```
{name: “matt”,
  weight:60,
  height: 72,
  loc: [44.6, 71.3]}
```

Replication

- La réplication offre une ***redondance*** et augmente la ***disponibilité*** des données.
- Avec plusieurs **copies** de données sur différents **serveurs** de base de données, la **réplication** offre un niveau ***de tolérance aux pannes*** contre la ***perte*** d'un seul serveur de base de données..

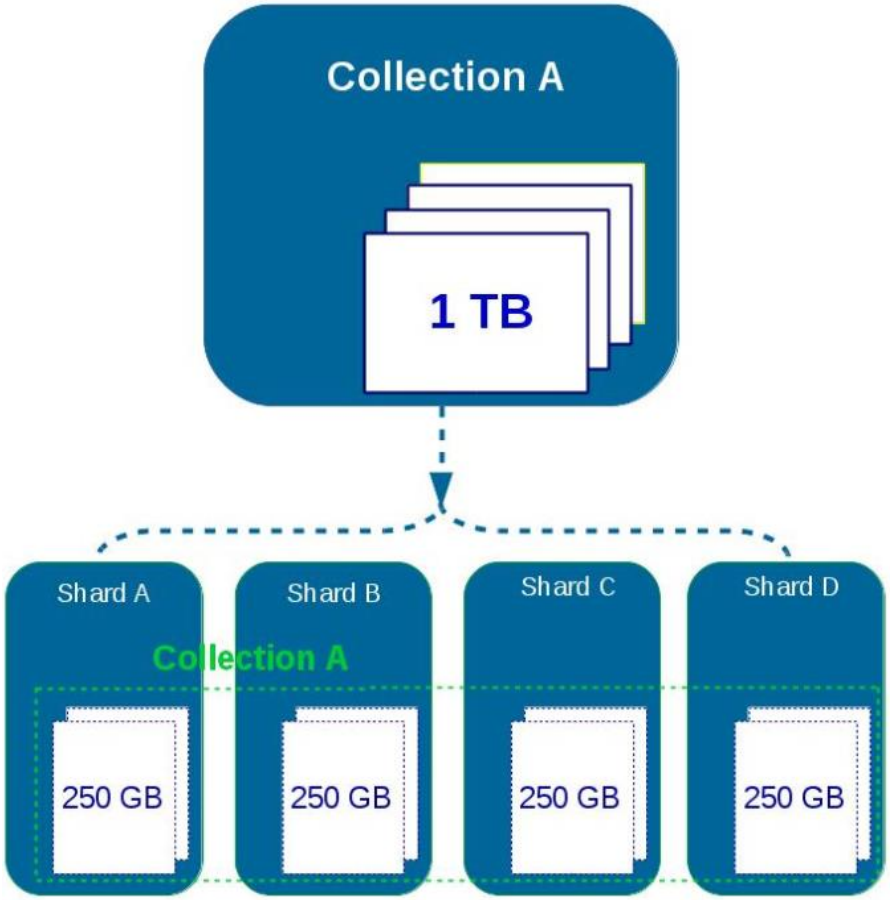




Partage (Sharding)

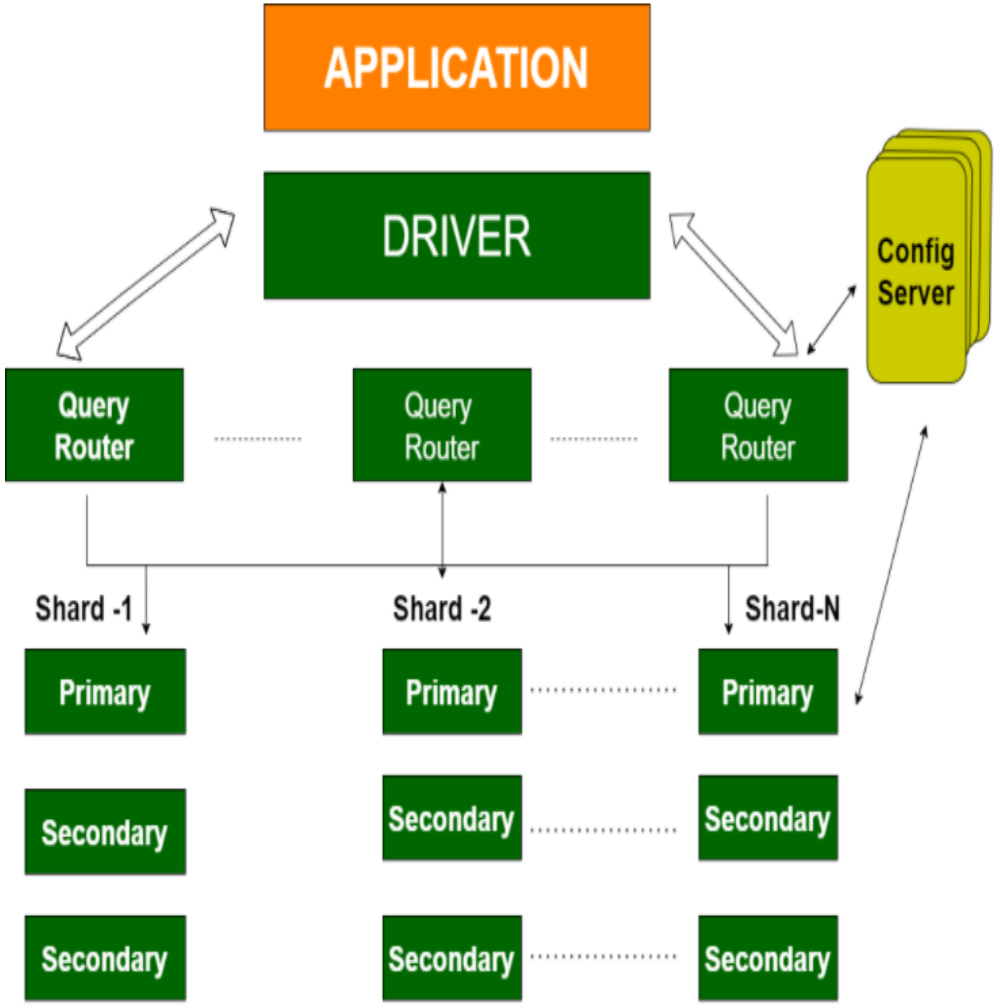
- Le **sharding** est une méthode de **distribution** de données sur **plusieurs machines**

- **MongoDB** utilise le ***sharding*** pour prendre en charge les déploiements avec de **très grands ensembles de données** et **des opérations à haut débit.**



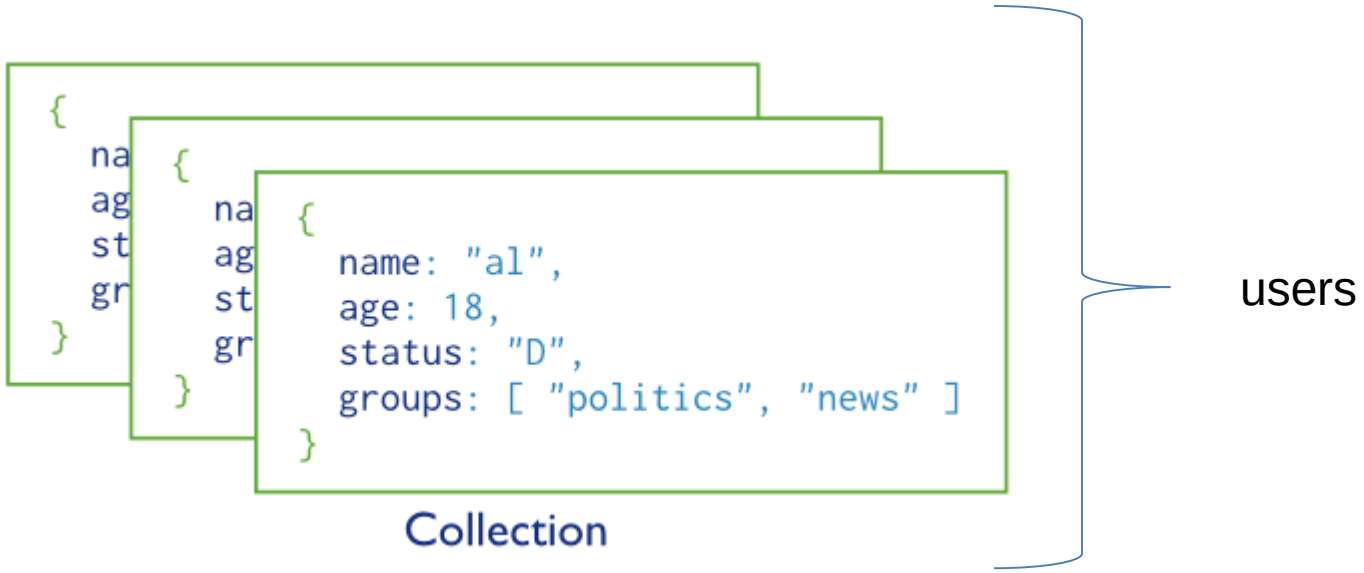
Sharding/Replication

- **Réplication**: Divisez les ensembles de données sur plusieurs nœuds de données pour une ***haute disponibilité***.
- ***Sharding :*** peut être augmentée ou réduite horizontalement lorsque cela est nécessaire pour un **débit élevé**.
- **Sharding:** est une technique utilize" pour assurer la **scalability dans MongoDB.**



Opérations de base

```
{
  name: "sue",          <-- field: value
  age: 26,              <-- field: value
  status: "A",          <-- field: value
  groups: [ "news", "sports" ]   <-- field: value
}
```



Opérations CRUD – créer (1)

Insert a new user.

SQL

```
INSERT INTO users                    <-- table
            ( name, age, status )    <-- columns
VALUES      ( "sue", 26, "A" )       <-- values/row
```

MongoDB

```
db.users.insert (                    <-- collection
   {
     name: "sue",                    <-- field: value
     age: 26,                        <-- field: value
     status: "A"                     <-- field: value
   }                                 } document
)
```

Opérations CRUD – créer (2)

Collection

Document

```
db.users.insert(
                  {
                     name: "sue",
                      age: 26,
                   status: "A",
                   groups: [ "news", "sports" ]
                  }
                )
```

Document

```
{
  name: "sue",
  age: 26,
  status: "A",
  groups: [ "news", "sports" ]
}
```

insert

Collection

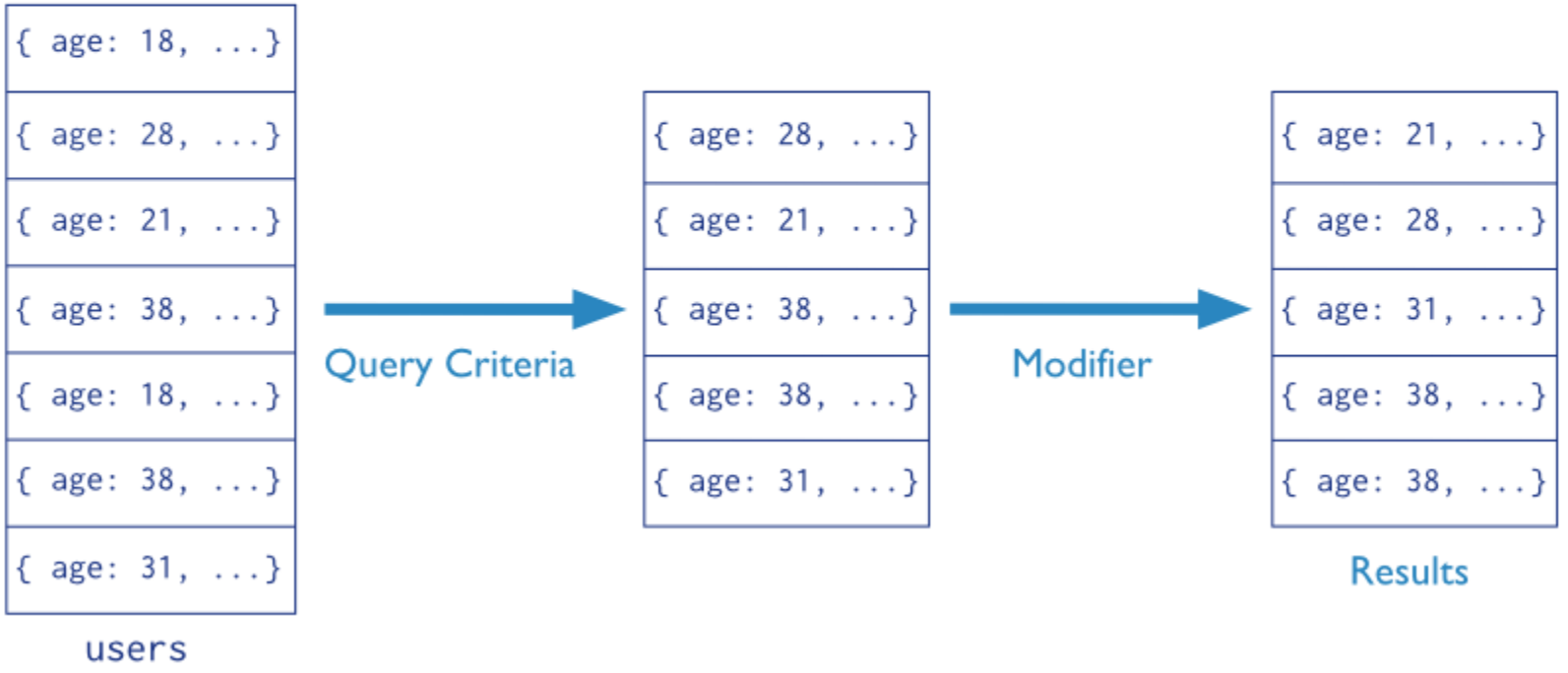
```
{ name: "al", age: 18, ... }
{ name: "lee", age: 28, ... }
{ name: "jan", age: 21, ... }
{ name: "kai", age: 38, ... }
{ name: "sam", age: 18, ... }
{ name: "mel", age: 38, ... }
{ name: "ryan", age: 31, ... }
{ name: "sue", age: 26, ... }
```

Opérations CRUD - lecture

***Find** the users of **age** greater than **18** and sort by **age**.*

Collection — Query Criteria — Modifier

```
db.users.find( { age: { $gt: 18 } } ).sort( {age: 1 } )
```



Opérations CRUD – mise à jour

*Update the **users** of **age** greater than **18** by setting the **status** field to **A**.*

SQL:

```
UPDATE users          <-- table
SET    status = 'A'   <-- update action
WHERE  age > 18       <-- update criteria
```

MongoDB:

```
db.users.update(                      <-- collection
   { age: { $gt: 18 } },              <-- update criteria
   { $set: { status: "A" } },         <-- update action
   { multi: true }                    <-- update option
)
```

Opérations CRUD - supprimer

***Delete** the **users** with **status** equal to **D**.*

SQL:

```
DELETE FROM users      <-- table
WHERE  status = 'D'    <-- delete criteria
```

MongoDB:

```
db.users.remove(       <-- collection
   { status: "D" }     <-- remove criteria
)
```

References

[1] Mikayel Vardanyan, Picking the right NoSQL Database Tool: http://blog.monitis.com/index.php/2011/05/22/picking-the-right-nosql-database-tool/

[2] BSON Specification: http://bsonspec.org/

[3] MongoDB CRUD operations: http://docs.mongodb.org/manual/crud/

[4] MongoDB Write operations: http://docs.mongodb.org/manual/core/write-operations/

[5] MongoDB Investors: http://www.mongodb.com/investors

[6] MongoDB Closes $150 Million in Funding: http://www.mongodb.com/press/mongodb-closes-150-million-funding

[7] MongoDB Aggregation introduction: http://docs.mongodb.org/manual/core/aggregation-introduction/