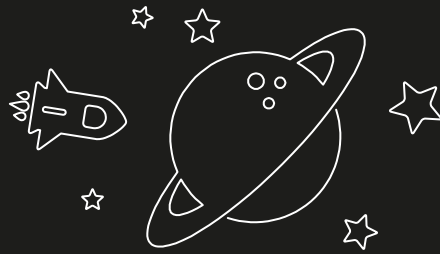


**MongoDB**



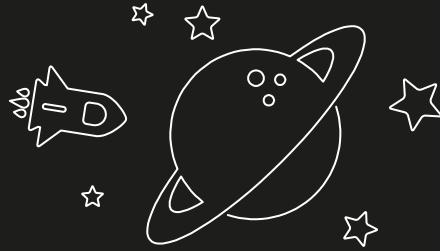


# HISTORIQUE

2

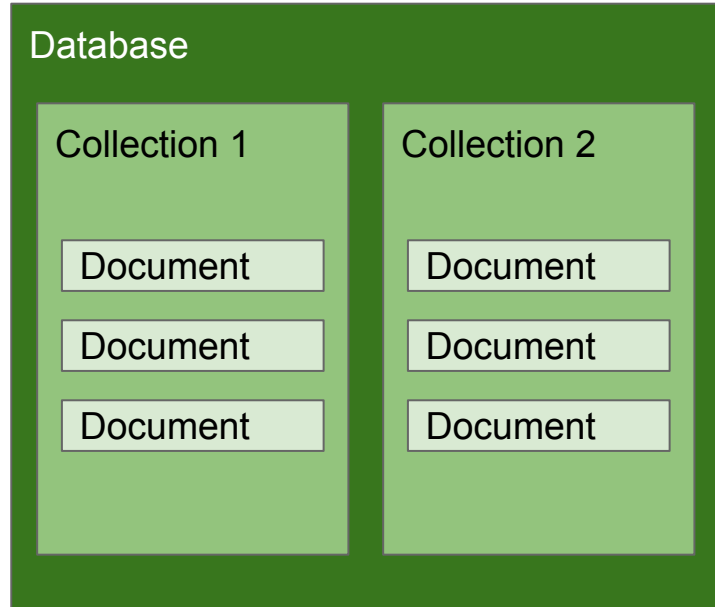
# 3

- Créateur : 10gen (MongoDB Inc en 2013)
- Date de début: 2007
- Licence: SSPL
- Dernière version: 4.4(.5)
- Type de base: NoSQL
- Type de données: Document JSON
- Language de requête: Javascript

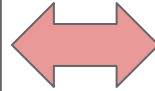
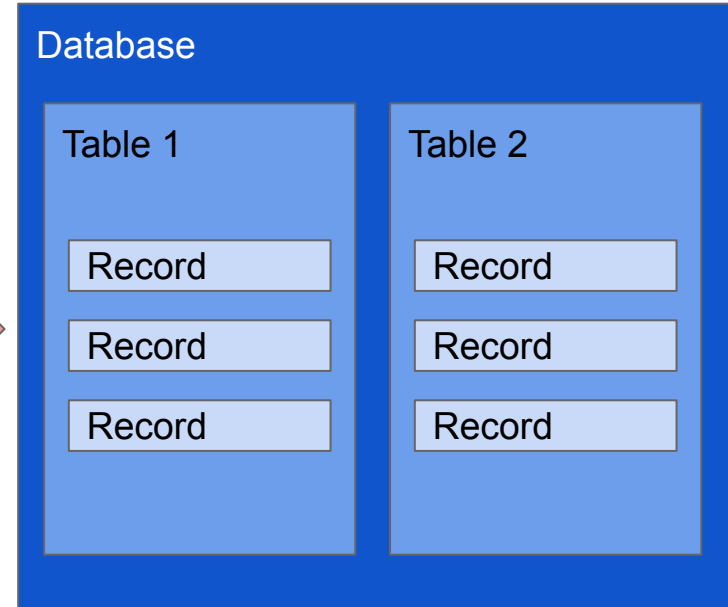


# Schématisation

## MongoDB



## SQL Database



# 6

## Principes / Avantages

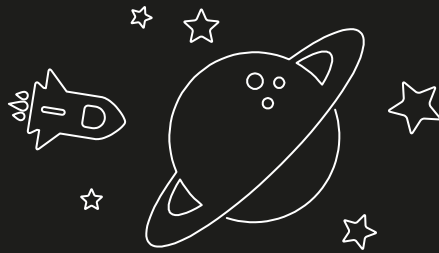
Pas de schéma

Pas de relations entre les documents

Le document correspond à la donnée applicative

Indexation FullText

Hyper-scalable



# INSTALLATION

7

**1) Télécharger le docker-compose à l'adresse suivante**

*<https://github.com/kmarques/esgi-node>*

**2) Personnaliser le docker-compose**

*Ports du service mongo*

*Credentials du service mongo*

**3) Télécharger MongoDB Compass à l'adresse suivante**

*<https://www.mongodb.com/products/compass>*

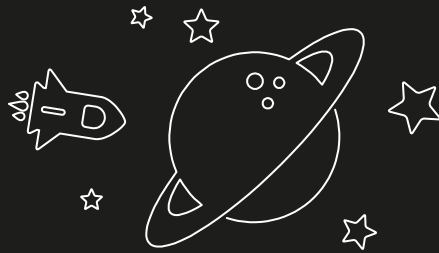
**4) Lancer les dockers**

*`docker-compose up -d`*

**5) Vérifier la connectivité**

*Lancer l'application MongoDB Compass et insérer le nouveau node*





**CRUD**

9

# 10

## Création

### Database

```
// Sélection/création de la db  
use DB_NAME;
```

### Collection

```
// Création manuelle d'une  
// collection  
db.createCollection("COLLNAME",  
    { /* OPTIONS */ }  
);
```

### Document

```
db.COLLNAME.insert(  
    { /* DOCUMENT */ }  
);
```

La collection est créée automatique si non existante. Il en sera de même pour la database.

# 11

## Suppression

### Database

```
db.dropDatabase();
```

### Collection

```
// Création manuelle d'une  
// collection  
db.COLLNAME.drop();
```

### Document

```
db.COLLNAME.deleteOne(  
    /* CRITERIA */  
);  
db.COLLNAME.deleteMany(  
    /* CRITERIA */  
);
```

**CRITERIA** : Objet de sélection

# 12

## Modification (1/2)

### Document

```
db.COLLNAME.update(  
  // CRITERIA: Object de sélection  
  {...},  
  // NEW_DATA: Nouveau document  
  {...},  
  // Upsert: Ajoute si non existant  
  boolean, // (default: false)  
  // Multi: Premier trouvé si false  
  boolean // (default: false)  
);
```

### NEW\_DATA:

- Par défaut, remplace le document trouvé
- Utiliser l'opérateur **\$set** pour mettre à jour certains champs

```
{ $set: {  
  title: "Foo"  
}}
```

# 13

## Modification (2/2)

### Document

```
db.COLLNAME.save(  
  {_id: 1, ...}  
);
```

La fonction **save** remplace entièrement un document dans la collection via l'**ID**.

Si l'**ID** n'existe pas ou n'est pas précisé, **save** va ajouter un nouveau document.

# 14

## Sélection

### Document

```
// Recherche le premier document
// matchant l'objet CRITERIA
db.COLLNAME.findOne(
    { /* CRITERIA */ }
);

// Recherche tous les documents
// matchant l'objet CRITERIA
db.COLLNAME.find(
    { /* CRITERIA */ }
);
```

**CRITERIA** : Objet de sélection

# 15

## Critère de recherches 1/2

### Combinaison

**AND** : entrée supplémentaire dans l'objet de sélection

```
db.COLLNAME.find({  
  title: "Foo",  
  description: "bar"  
});
```

**OR** : Utilisation de la clé **\$or** dont la valeur est un tableau

```
db.COLLNAME.find({$or: [  
  {title: "Foo"},  
  {title: "Bar"}  
]});
```

# 16

## Critère de recherches 1/2

### Général

#### EQUALS

```
{key: VALUE}
```

#### NOT EQUALS

```
{key: {$ne: VALUE}}
```

### Numérique/Date

#### LESS/GREATER THAN

```
{amount: {$gt/lt: NUMBER}}
```

#### LESS/GREATER THAN EQUALS

```
{amount: {$gte/lte: NUMBER}}
```



17

## Critère de recherches 2/2

### Texte

#### REGEXP

```
{key: /myregexp/}  
// ou {key: { $regexp: "myregexp" }}
```

#### TEXT SEARCH

```
{ $text: { $search: "my text" }}
```

Recherche sur tous les indexes de type  
FullText

# 18

## Pagination

### LIMIT

```
db.COLLNAME.find(  
    { /* CRITERIA */ }  
) .limit(10); // Résultat de 1 à 10
```

Limite le nombre (**X**) de résultats d'une requête

### OFFSET

```
db.COLLNAME.find(  
    { /* CRITERIA */ }  
) .limit(10) .skip(20) ;  
// Résultat de 21 à 30 <==> Page 3
```

Démarre les résultats après **X** documents

# 19

## Sort

```
db.COLLNAME.find(  
  { /* CRITERIA */  
}).sort({  
  date: 1,  
  id: -1  
});
```

Description des valeurs

1 : Asc

- 1 : Desc

**L'ordre des clés est important**

# 20

## Filtre/Projection

```
db.COLLNAME.find(  
  { /* CRITERIA */ },  
  { /* PROJECTION */  
    title: 1,  
    date: 1  
  }  
);
```

Description des valeurs

0 : Exclure

1 : Inclure

On ne peut utiliser que l'une des 2 stratégies,  
**inclusion** ou **exclusion**

- Si **DOC** à 40 valeurs et nous souhaitons en afficher 3 => Stratégie d'**inclusion**
- Si **DOC** à 40 valeurs et nous souhaitons en afficher 35 => Stratégie d'**exclusion**

## MongoDB

```
db.users.find({
  name: /jean/i,
  dob: { $gt: new Date("2001-01-01")
}
}, {
  name: 1, address: 0, dob: 1
})
.sort({
  name: 1,
  dob: -1
})
.limit(10).skip(10)
```

## SQL

```
SELECT
  name,
  dob
FROM users
WHERE
  lower(name) LIKE "%jean%"
  AND
  dob > "2001-01-01"::DATE
ORDER BY
  name ASC,
  dob DESC
LIMIT 10
OFFSET 10
```

### Exercices : Recherche "simple"

Collections: Sakila\_films

- 1) Rechercher tous les films dont la description comprend "documentary" et de catégorie "horror"
- 2) Donner le nombre de films en rating "G"

Collections: video\_movieDetails

- 3) Rechercher tous les films de 2013 ou 2012 dont la durée est entre 60 et 150 minutes
- 4) Rechercher tous les films qui ont une image certifiée sur tomato

Collections: Sakila\_films

- 5) Rechercher tous les films avec l'acteur ED CHASE
- 6) Afficher tous les ratings ("PG", "PG-13", ...) et le nombre de films pour chacun

Exercices : Agrégation

Collections: video\_movieDetails

- 1) La liste des films triés par ordre de parution qui ont eu une note imdb > 8 par Acteur par Rating avec la moyenne des notes imdb par Rating
- 2) Lister pour chaque genre de films, les différents rated avec le pire film et le meilleur film selon imdb