
# LES RELATIONS DANS MONGO (NO SQL)

## RELATION ONE-TO-ONE

Prenons l'exemple suivant qui met en correspondance les relations entre une personne et son adresse. Cet exemple illustre l'avantage de l'incorporation par rapport à la référence si vous devez visualiser une entité de données dans le contexte de l'autre. Dans cette relation biunivoque entre les données personne et d'adresse, l'adresse appartient à la personne.

Dans le modèle de données normalisé , le document adresse contient une référence au document patron.

ça pourrait être modélisé ainsi dans mongodb (à l'instar d'une SGBD relationnelle):

```
collection personne (table)
{
       _id: "joe",
      name: "Joe Strummer"
}

collection adresse (table)
{
      id_: …
      patron_id: "joe",
      rue: "des lilas",
      cp: 29200,
      ville: "Brest"
}
```

Si les données d'adresse sont fréquemment récupérées avec les informations de nom, alors avec le référencement, votre application doit émettre plusieurs requêtes pour résoudre la référence.

Le meilleur modèle de données serait d'intégrer les données d'adresse dans les données de la personne, comme dans le document suivant :

```
collection personne
{
       _id: "joe",
      name: "Joe Strummer"
      adresse: {
            rue: "des lilas",
            cp: 29200,
            ville: "Brest"
      }
}
```

De plus cette façon de faire se rapproche bien plus de la réalité, NOSql est plus représentatif du réel qu'une base de données relationnelle.

## RELATION ONE-TO-MANY

# Embedded

Si la personne a plusieurs adresses il est encore ici plus efficace d'intégrer les adresses dans le document personne  (une seule requête)

```
collection personne
{
       _id: "joe",
      name: "Joe Strummer"
      adresse: [
            {
            rue: "des lilas",
            cp: 29200,
            ville: "Brest"
            },
            {
            rue: "kerdiles",
            cp: 29800,
            ville: "Saint-Thonan"
            }
      ]
}
```

# Avec références sur les documents

Soit un éditeur et des livres.

L'incorporation du document de l'éditeur dans le document du livre entraînerait la répétition des données de l'éditeur, comme le montrent les documents suivants :

```
 {
   title: "MongoDB: The Definitive Guide",
   author: [ "Kristina Chodorow", "Mike Dirolf" ],
   published_date: ISODate("2010-09-24"),
   pages: 216,
   language: "English",
   publisher: {
              name: "O'Reilly Media",
              founded: 1980,
              location: "CA"
            }
}

{
   title: "50 Tips and Tricks for MongoDB Developer",
   author: "Kristina Chodorow",
   published_date: ISODate("2011-05-06"),
   pages: 68,
   language: "English",
   publisher: {
              name: "O'Reilly Media",
              founded: 1980,
              location: "CA"
            }
}
```

C’est relou !

Pour éviter la répétition des données sur l'éditeur, utilisez des références et conservez les informations sur l'éditeur dans une collection distincte de celle des livres.

Collection éditeur avec comme donnée liste de référence sur les livres

collection editor

```
{
   name: "O'Reilly Media",
   founded: 1980,
   location: "CA",
   books: [12346789, 234567890]
}
```

```
collection livre

{
```

```
    _id: 123456789,
    title: "MongoDB: The Definitive Guide",
    author: [ "Kristina Chodorow", "Mike Dirolf" ],
    published_date: ISODate("2010-09-24"),
    pages: 216,
    language: "English"
}



{
   _id: 234567890,
   title: "50 Tips and Tricks for MongoDB Developer",
   author: "Kristina Chodorow",
   published_date: ISODate("2011-05-06"),
   pages: 68,
   language: "English"
}
```

Si il y a beaucoup de livres , pour éviter les tableaux mutables et croissants, stockez la référence de l'éditeur à l'intérieur du document du livre

```
collection publisher
{
   _id: "oreilly",
   name: "O'Reilly Media",
   founded: 1980,
   location: "CA"
}


collection livres

{
   _id: 123456789,
   title: "MongoDB: The Definitive Guide",
   author: [ "Kristina Chodorow", "Mike Dirolf" ],
   published_date: ISODate("2010-09-24"),
   pages: 216,
   language: "English",
   publisher_id: "oreilly"
}



{
   _id: 234567890,
   title: "50 Tips and Tricks for MongoDB Developer",
   author: "Kristina Chodorow",
   published_date: ISODate("2011-05-06"),
   pages: 68,
   language: "English",
   publisher_id: "oreilly"
}
```

# RELATION MANY-TO-MANY

La mise en œuvre d'une relation de plusieurs à plusieurs dans une base de données relationnelle n'est pas aussi simple qu'une relation de un à plusieurs, car il n'existe pas de commande unique pour l'accomplir. Il en va de même pour leur mise en œuvre dans mongoDB. En fait, vous ne pouvez mettre en œuvre aucun type de relation dans mongoDB via une commande. Cependant, la possibilité de stocker des tableaux dans un document vous permet de stocker les données d'une manière qui est rapide à récupérer et facile à maintenir et vous fournit les informations pour relier deux documents dans votre code.

Le plus simple est de tester les exemples suivants dans le shell mongo

Dans une collection personne

```
db.personne.insert({
  "_id": "joe_mongo",
  "firstName": "Joe",
  "lastName": "Mongo",
  "groups": [
     "g1",
     "g2"
  ]
});

db.personne.insert({
  "_id": "sally_mongo",
  "firstName": "Sally",
  "lastName": "Mongo",
  "groups": [
     "g1"
  ]
});

Dans une collection groupes

db.groupes.insert({
  "_id": "g1",
  "groupName": "mongoDB User",
```

```
  "persons": [
    "joe_mongo",
    "sally_mongo"
  ]
});
```

```
db.groupes.insert({
  "_id": "g2",
  "groupName": "mongoDB Administrator",
  "persons": [
   "joe_mongo"
  ]
});
```

vous pouvez simplement créer le tableau approprié soit sur les documents de la personne, soit sur les documents du groupe, mais cela compliquerait quelque peu vos requêtes.

Les requêtes suivantes vous montrent comment interroger les données sans avoir à utiliser les jointures comme dans une base de données relationnelle

```
// toutes les personnes du groupe "mongoDB User"
db.personne.find({"groups": "g1" });

// toutes les personnes du groupe"mongoDB Administrator"
db.personne.find({"groups": "g2" });

// Groupes de "Joe Mongo"
db.groupes.find({"persons":"joe_mongo" });

// Groupes de "Sally Mongo"
db.groupes.find({"persons":"sally_mongo" });
```

Ajouter une personne à un groupe

```
db.groupes.update({ "_id" : "g2"},{ $addToSet : { "persons":"jam_mongo" }})
```

Afin d'améliorer les performances des requêtes ci-dessus, vous devez créer des index sur le champ person.groups et le champ groups.persons. Ceci peut être réalisé en utilisant les commandes suivantes.

pour des meilleures performances penser à indexer vos collections

```
db.person.ensureIndex({"groups": 1});
db.groups.ensureIndex({"persons": 1});
```

exemple pour ajouter un élément dans un tableau de tableau

```
db.demo.insertOne(
   {
       "_id": Object_id("…")
      "UserDetails":[
         {
            "UserId" :"user121",
            "userGroupMessage":[]
         },
         {
            "UserId" :"user221",
            "userGroupMessage":["Cool","Good Morning"]
         }
      ]
   }
);
```

```
db.demo.update({"UserDetails.UserId":"user121"},
   {"$addToSet":{"UserDetails.$.userGroupMessage":"Hello"}});
```

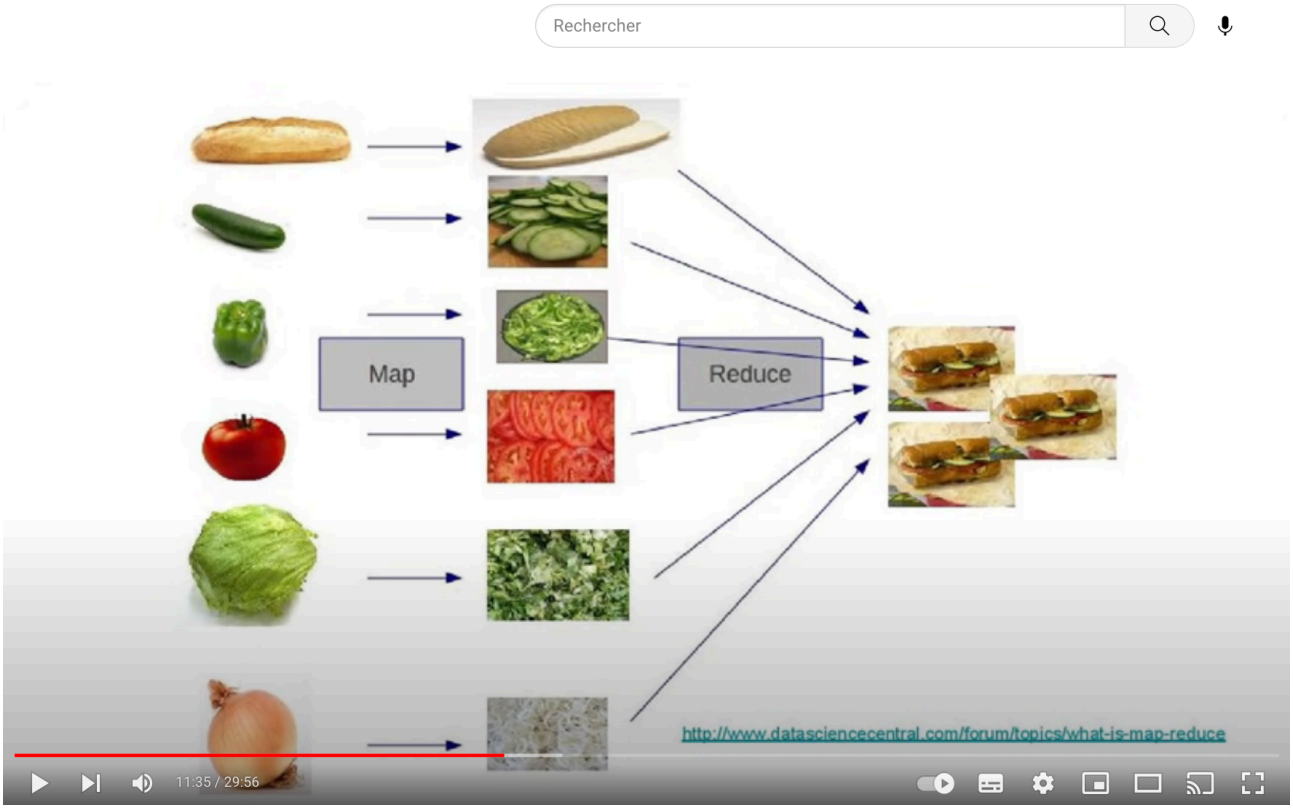
# Récupération lastInsertId en nodejs

Exemple de récupération  du dernier id créé par mongo lors d'une insertion

```
 db.collection('personne').insertOne(objetPerson, function(err, result)
{
                    console.log("Record added as "+result.insertedId);
                            callback(result);
                      })
```

# AGGREGATION

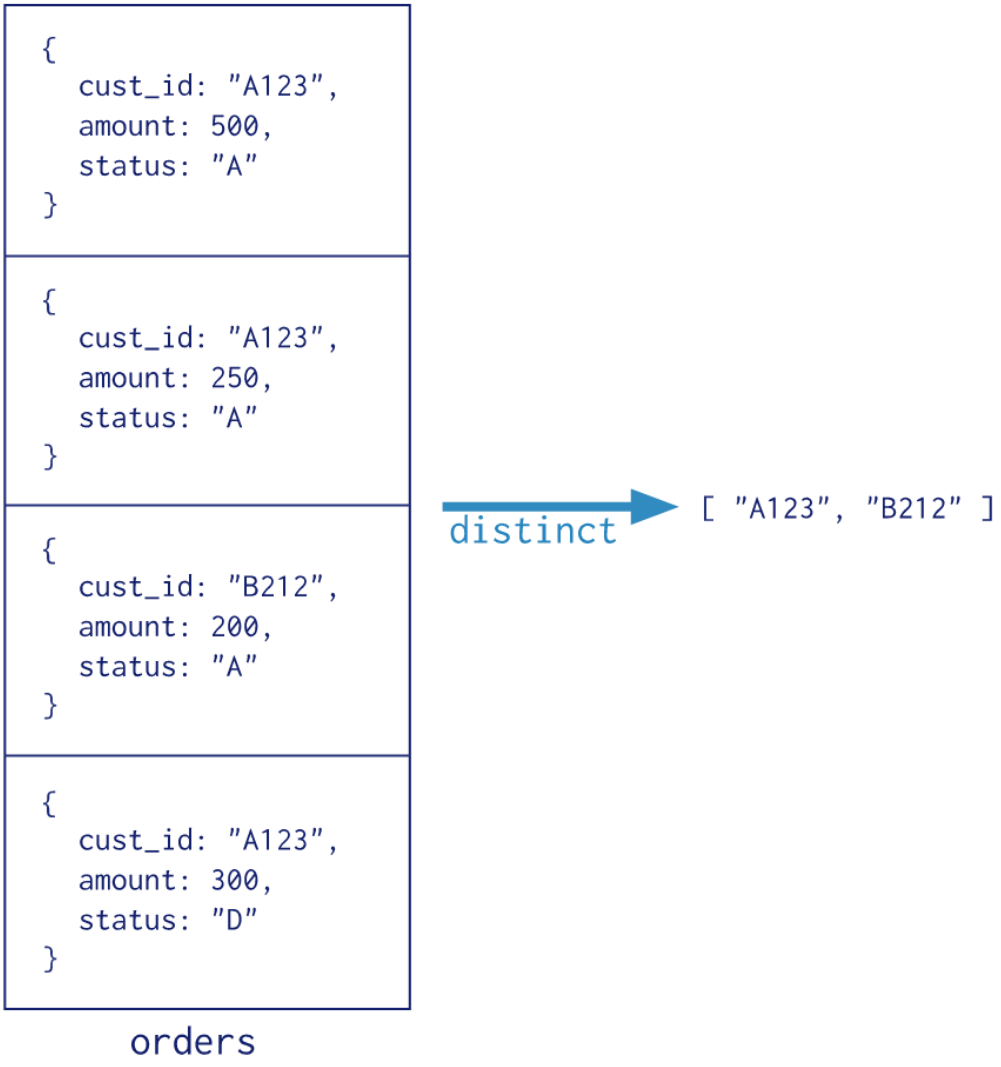
L'aggregation utilise le paradigme MapReduce pour obtenir l'équivalent de certaines opérations possibles en SQL ( count,  sum , max , min ,avg , group by, distinct…etc…)

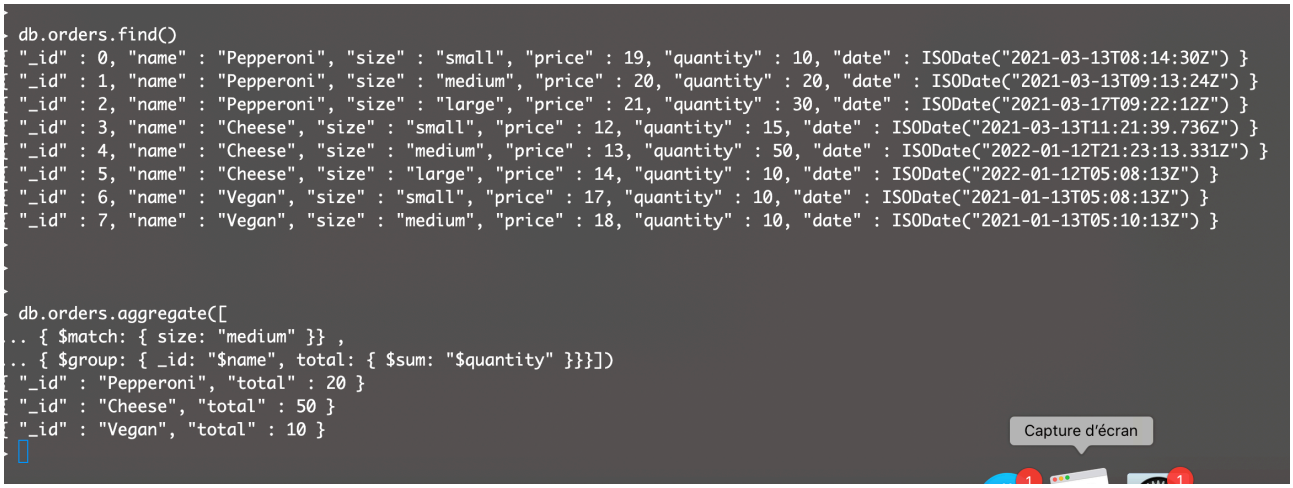



```
db.x.count({"pizza_type_id" : "veggie_veg"   })

db.x.aggregate( { $match : {"pizza_type_id" : "veggie_veg"}} ,
{ $count: "nb" } )


db.x.aggregate([ { $group: { _id: "$pizza_type_id", nb_type:
{ $sum: 1 }}}])

db.x.aggregate([ { $group: { _id: null, nb: { $sum: 1 }}}])
```

Collection

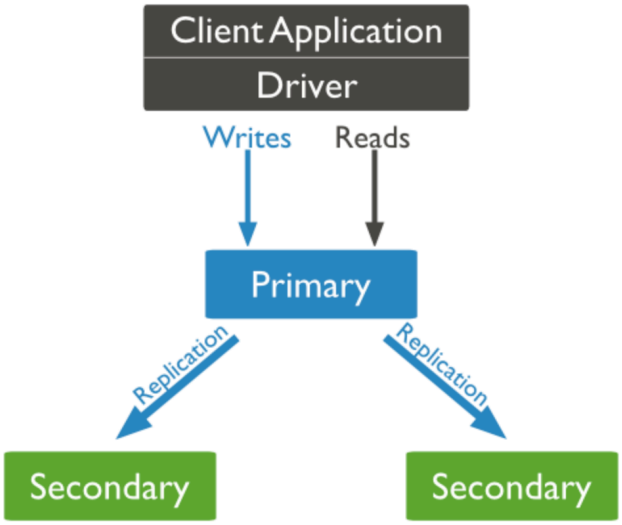
```
db.orders.distinct( "cust_id" )
```






```
db.orders.aggregate( [
...     // Stage 1: filtre date debut - date fin
...     {
...        $match:
...        {
...           "date": { $gte: new ISODate( "2020-01-30" ), $lt: new
ISODate( "2022-01-30" ) }
...        }
...     },
...     // Stage 2: regroupement par date puis calculs (total de
commandes et quantités moyennes
...     {
...        $group:
...        {
...           _id: { $dateToString: { format: "%Y-%m-%d", date:
"$date" } },
...           totalOrderValue: { $sum: { $multiply: [ "$price",
"$quantity" ] } },
...           averageOrderQuantity: { $avg: "$quantity" }
...        }
...     },
...     // Stage 3: tri descendant sur le total prix des commande
...     {
...        $sort: { totalOrderValue: -1 }
...     }
...   ] )
```

# Réplication




Les réplications permettent 1) supporter la montée en charge 2) toléré les pannes. En pratique les différents répliquants sont disposé sur différents serveurs pour une meilleure sécurité, ici nous pouvons effectuer des tests en local en exécutant plusieurs services mongod sur des ports différents.

Un serveur sera primaire et en cas de déficience de ce serveur le système procède à une élection pour décider qui deviendra le serveur primaire.

Un système de "heartbeat" vérifie en permanence la visibilité des serveurs.

Pratique:

il faut un minimum de 3 serveurs mongod pour 1 replicaSet (possibilité d'avoir 50 répliquants)

créer répertoires

**data/RS01**
**data/RS02**
**data/RS03**

Lancer dans 3 terminaux différents

```
 /usr/bin/mongod --port 27017 --dbpath "RS01" --replSet "monreplika"
ou C:\chemin-vers\mongosh.exe
```

```
/usr/bin/mongod --port 27018 --dbpath ".RS02" --replSet
"monreplika"
/usr/bin/mongod --port 27019 --dbpath "./RS03" --replSet
"monreplika"
```

Se connecter au mongoshell sur un des port

```
/usr/bin/mongo –port 27017
ou C:\chemin-vers\mongosh.exe

> rs.initiate();
> rs.conf();
> rs.add("localhost:27018");
> rs.add("localhost:27019");
> rs.status();


// si ce n’est pas sur le même serveur créer un
répliquant arbitre.

bin/mongod --port 30000 --dbpath "./data/arb" --
replSet "monreplika"
```