



Choix de la Structure: Normalisé vs Dénormalisé

Normalisation : La normalisation de la base de données vise à minimiser la redondance et assurer l'intégrité des données. Cela implique de structurer les données de manière à éviter les duplications en séparant les informations dans différentes collections (tables).

Dénormalisation : La dénormalisation, en revanche, implique de regrouper les données redondantes dans une seule collection pour des raisons de performance, surtout lors des requêtes fréquentes et complexes.

Justification : Pour cette base de données de magasin de sport, nous avons opté pour une structure partiellement normalisée. Nous avons séparé les articles et les rayons en deux collections distinctes :

- **Articles**: Contient les informations détaillées sur chaque article.
- **Rayons**: Contient les informations sur chaque rayon du magasin.

La relation entre les deux collections est établie par l'attribut `rayon_id` dans la collection `articles`, qui fait référence à l'identifiant de la collection `rayons`. Ce choix permet une certaine flexibilité dans les mises à jour tout en évitant des duplications excessives. Cependant, certaines informations comme les fournisseurs sont incluses directement dans les articles pour simplifier les requêtes fréquentes.

Schémas de Bases de Données MongoDB avec Explications

Schéma de la Collection Articles

```
{
  bsonType: "object",
  required: ["marque", "prix", "nom", "reference", "fournisseur", "rayon_id"],
  properties: {
    marque: { bsonType: "string" },
    prix: { bsonType: "decimal" },
    nom: { bsonType: "string" },
    reference: { bsonType: "string" },
    categorie: { enum: ["enfant", "junior", "senior"] },
    tailles: {
      bsonType: "array",
      items: { bsonType: "int" }
    },
    fournisseur: {
      bsonType: "object",
      required: ["nom", "adresse"],
      properties: {
        nom: { bsonType: "string" },
        adresse: { bsonType: "string" }
      }
    },
    rayon_id: { bsonType: "objectId" }
  }
}
```

Explication :

- **marque** : Le type est une chaîne de caractères (`string`). Ce champ est requis et contient le nom de la marque de l'article.
- **prix** : Le type est un nombre décimal (`decimal`). Ce champ est requis et stocke le prix de l'article.
- **nom** : Le type est une chaîne de caractères (`string`). Ce champ est requis et contient le nom de l'article.
- **reference** : Le type est une chaîne de caractères (`string`). Ce champ est requis et contient une référence unique pour l'article.
- **categorie** : Le type est une chaîne de caractères avec une valeur énumérée (`enum`). Il peut être "enfant", "junior" ou "senior". Ce champ est optionnel et indique la catégorie de l'article.
- **tailles** : Le type est un tableau d'entiers (`array of int`). Ce champ est optionnel et contient les tailles disponibles pour l'article.
- **fournisseur** : Le type est un objet (`object`). Ce champ est requis et contient des sous-champs :
 - **nom** : Une chaîne de caractères représentant le nom du fournisseur.
 - **adresse** : Une chaîne de caractères représentant l'adresse du fournisseur.
- **rayon_id** : Le type est un ObjectId (`objectId`). Ce champ est requis et stocke l'identifiant du rayon auquel l'article appartient.

Schéma de la Collection Rayons

```
{
  bsonType: "object",
  required: ["description", "employe_responsable"],
  properties: {
    description: { bsonType: "string" },
    employe_responsable: { bsonType: "string" }
  }
}
```

Explication :

- **description** : Le type est une chaîne de caractères (`string`). Ce champ est requis et contient une description du rayon.
- **employe_responsable** : Le type est une chaîne de caractères (`string`). Ce champ est requis et contient le nom de l'employé responsable du rayon.

Requêtes Permettant de Tester les Schémas

Insertion dans la collection rayons :

```
db.rayons.insertOne({
  description: "Rayon Vêtements",
  employe_responsable: "Jean Dupont"
});
```

Cette requête récupère et affiche tous les articles de la collection `articles`.

Insertion dans la collection articles :

```
db.articles.insertOne({
  marque: "Nike",
  prix: NumberDecimal("59.99"),
  nom: "T-shirt Sport",
  reference: "TS123",
  categorie: "senior",
  tailles: [38, 40, 42],
  fournisseur: { nom: "Nike Inc", adresse: "Beaverton, OR, USA" },
  rayon_id: ObjectId("66589ac6d4b212f0082202d8")
});
```

Cette requête insère un document dans la collection `articles` avec toutes les propriétés requises et une référence au rayon correspondant.

Jeu de 5 Requêtes Simples avec Explications et Résultats

1) Tous les articles :

```
db.articles.find().pretty();
```

Cette requête récupère et affiche tous les articles de la collection articles.

```
[
  {
    _id: ObjectId('66589afcd4b212f0082202d9'),
    marque: 'Nike',
    prix: Decimal128('59.99'),
    nom: 'T-shirt Sport',
    reference: 'TS123',
    categorie: 'senior',
    tailles: [ 38, 40, 42 ],
    fournisseur: { nom: 'Nike Inc', adresse: 'Beaverton, OR, USA' },
    rayon_id: ObjectId('66589ac6d4b212f0082202d8')
  }
]
```

2) Articles par marque :

```
db.articles.find({ marque: "Nike" }).pretty();
```

Cette requête récupère tous les articles dont la marque est "Nike".

```
[
  {
    _id: ObjectId('66589afcd4b212f0082202d9'),
    marque: 'Nike',
    prix: Decimal128('59.99'),
    nom: 'T-shirt Sport',
    reference: 'TS123',
    categorie: 'senior',
    tailles: [ 38, 40, 42 ],
    fournisseur: { nom: 'Nike Inc', adresse: 'Beaverton, OR, USA' },
    rayon_id: ObjectId('66589ac6d4b212f0082202d8')
  }
]
```

3) Articles par catégorie :

```
db.articles.find({ categorie: "senior" }).pretty();
```

Cette requête récupère tous les articles de la catégorie "senior".

```
[
  {
    _id: ObjectId('66589afcd4b212f0082202d9'),
    marque: 'Nike',
    prix: Decimal128('59.99'),
    nom: 'T-shirt Sport',
    reference: 'TS123',
    categorie: 'senior',
    tailles: [ 38, 40, 42 ],
    fournisseur: { nom: 'Nike Inc', adresse: 'Beaverton, OR, USA' },
    rayon_id: ObjectId('66589ac6d4b212f0082202d8')
  }
]
```

4) Articles en fonction du prix (moins de 50€) :

```
db.articles.find({ prix: { $lt: NumberDecimal("100.00") } }).pretty();
```

Cette requête récupère tous les articles dont le prix est inférieur à 50 euros.

```
[
  {
    _id: ObjectId('66589afcd4b212f0082202d9'),
    marque: 'Nike',
    prix: Decimal128('59.99'),
    nom: 'T-shirt Sport',
    reference: 'TS123',
    categorie: 'senior',
    tailles: [ 38, 40, 42 ],
    fournisseur: { nom: 'Nike Inc', adresse: 'Beaverton, OR, USA' },
    rayon_id: ObjectId('66589ac6d4b212f0082202d8')
  }
]
```

5) Rayons par description :

```
db.rayons.find({ description: "Rayon Vêtements" }).pretty();
```

Cette requête récupère tous les rayons ayant la description "Rayon Vêtements".

```
[
  {
    _id: ObjectId('66589ac6d4b212f0082202d8'),
    description: 'Rayon Vêtements',
    employe_responsable: 'Jean Dupont'
  }
]
```

Jeu de 5 Requêtes Recherchées (Modifications et Suppressions) avec Explications et Résultats

1) Mettre à jour le prix d'un article :

```
db.articles.updateOne({ reference: "TS123" }, { $set: { prix: NumberDecimal("49.99") } });
```

Cette requête met à jour le prix de l'article ayant la référence "TS123" à 49.99 euros.

```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

2) Ajouter une taille à un article :

```
db.articles.updateOne({ reference: "TS123" }, { $push: { tailles: 44 } });
```

Cette requête ajoute la taille 44 à la liste des tailles disponibles pour l'article "TS123".

```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

3) Modifier le fournisseur d'un article :

```
db.articles.updateOne({ reference: "TS123" }, { $set: { "fournisseur.nom": "Adidas Inc" } });
```

Cette requête modifie le nom du fournisseur de l'article "TS123" à "Adidas Inc".

```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

4) Supprimer un article par référence :

```
db.articles.deleteOne({ reference: "TS123" });
```

Cette requête supprime l'article ayant la référence "TS123" de la collection.

```
{ acknowledged: true, deletedCount: 1 }
```

5) Mettre à jour la description d'un rayon :

```
db.rayons.updateOne({ description: "Rayon Vêtements" }, { $set: { description: "Rayon Sportswear" } });
```

Cette requête met à jour la description du rayon de "Rayon Vêtements" à "Rayon Sportswear".

```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Jeu de 5 Requêtes Complexes (Aggregate entre autres) avec Explications et Résultats

1) Articles avec leur rayon et prix total par catégorie :

```
db.articles.aggregate([
  {
    $lookup: {
      from: 'rayons',
      localField: 'rayon_id',
      foreignField: '_id',
      as: 'rayon'
    }
  },
  {
    $unwind: '$rayon'
  },
  {
    $group: {
      _id: "$categorie",
      totalPrix: { $sum: "$prix" }
    }
  }
]);
```

[{ _id: 'senior', totalPrix: Decimal128('59.99') }]

Cette requête récupère les articles avec leur rayon correspondant et calcule le prix total des articles par catégorie.

```
[ { _id: 'senior', totalPrix: Decimal128('59.99') } ]
```

2) Moyenne des prix des articles par catégorie :

```
db.articles.aggregate([
  {
    $group: {
      _id: "$categorie",
      moyennePrix: { $avg: "$prix" }
    }
  }
]);
```

Cette requête calcule la moyenne des prix des articles pour chaque catégorie.

```
[ { _id: 'senior', moyennePrix: Decimal128('59.99') } ]
```


3) Articles par fournisseur :

```
db.articles.aggregate([
  {
    $group: {
      _id: "$fournisseur.nom",
      articles: { $push: "$$ROOT" }
    }
  }
]);
```

Cette requête regroupe les articles par fournisseur.

```
[
  {
    _id: 'Nike Inc',
    articles: [
      {
        _id: ObjectId('66589fa7d4b212f0082202da'),
        marque: 'Nike',
        prix: Decimal128('59.99'),
        nom: 'T-shirt Sport',
        reference: 'TS123',
        categorie: 'senior',
        tailles: [ 38, 40, 42 ],
        fournisseur: { nom: 'Nike Inc', adresse: 'Beaverton, OR, USA' },
        rayon_id: ObjectId('66589ac6d4b212f0082202d8')
      }
    ]
  }
]
```

4) Nombre d'articles par rayon :

```
db.articles.aggregate([
  {
    $group: {
      _id: "$rayon_id",
      count: { $sum: 1 }
    }
  }
]);
```

Cette requête compte le nombre d'articles dans chaque rayon.

```
[ { _id: ObjectId('66589ac6d4b212f0082202d8'), count: 1 } ]
```

5) Articles avec tailles disponibles et rayon :

```
db.articles.aggregate([
  {
    $lookup: {
      from: 'rayons',
      localField: 'rayon_id',
      foreignField: '_id',
      as: 'rayon'
    }
  },
  {
    $unwind: '$rayon'
  },
  {
    $project: {
      nom: 1,
      tailles: 1,
      'rayon.description': 1
    }
  }
]);
```

Cette requête récupère les articles avec les tailles disponibles et la description de leur rayon.

```
[
  {
    _id: ObjectId('66589fa7d4b212f0082202da'),
    nom: 'T-shirt Sport',
    tailles: [ 38, 40, 42 ],
    rayon: { description: 'Rayon Sportswear' }
  }
]
```

Monté en charge

Tests faits avec <https://locust.io/>

MongoDB :

Type	Name	# reqs	# fails	Avg	Min	Max	Med	req/s	failures/s				
GET	/articles	380	0(0.00%)	57	32	172	57	6.37	0.00				
POST	/articles	389	0(0.00%)	8	3	158	5	6.52	0.00				
DELETE	/articles/66589fa7d4b212f0082202da	375	0(0.00%)	8	2	160	5	6.29	0.00				
PUT	/articles/66589fa7d4b212f0082202da	395	0(0.00%)	7	3	126	5	6.62	0.00				
GET	/rayons	339	0(0.00%)	7	3	131	5	5.69	0.00				
Aggregated		1878	0(0.00%)	17	2	172	6	31.50	0.00				
Response time percentiles (approximated)													
Type	Name	50%	66%	75%	80%	90%	95%	98%	99%	99.9%	99.99%	100%	# reqs
GET	/articles	57	64	68	70	76	86	99	110	170	170	170	380
POST	/articles	5	6	7	7	10	24	46	63	160	160	160	389
DELETE	/articles/66589fa7d4b212f0082202da	5	6	6	7	9	23	50	120	160	160	160	375
PUT	/articles/66589fa7d4b212f0082202da	5	6	6	7	9	20	45	70	130	130	130	395
GET	/rayons	5	6	6	7	9	22	40	48	130	130	130	339
Aggregated		6	7	20	37	59	69	79	93	160	170	170	1878

Mongoose :

Type	Name	# reqs	# fails	Avg	Min	Max	Med	req/s	failures/s				
GET	/articles	355	0(0.00%)	92	61	224	90	5.95	0.00				
POST	/articles	364	0(0.00%)	9	5	58	8	6.10	0.00				
DELETE	/articles/66589fa7d4b212f0082202da	391	0(0.00%)	7	3	63	6	6.55	0.00				
PUT	/articles/66589fa7d4b212f0082202da	390	1(0.26%)	8	2	73	6	6.54	0.02				
GET	/rayons	366	0(0.00%)	8	3	50	6	6.13	0.00				
Aggregated		1866	1(0.05%)	24	2	224	7	31.27	0.02				
Response time percentiles (approximated)													
Type	Name	50%	66%	75%	80%	90%	95%	98%	99%	99.9%	99.99%	100%	# reqs
GET	/articles	90	94	97	99	110	130	140	200	220	220	220	355
POST	/articles	8	8	9	9	11	16	38	49	58	58	58	364
DELETE	/articles/66589fa7d4b212f0082202da	6	6	7	7	9	20	36	46	63	63	63	391
PUT	/articles/66589fa7d4b212f0082202da	6	7	8	8	12	26	44	53	74	74	74	390
GET	/rayons	6	6	7	7	10	27	37	44	51	51	51	366
Aggregated		7	8	13	44	89	96	110	120	220	220	220	1866

Temps de Réponse:

- MongoDB a des temps de réponse plus rapides pour les requêtes GET /articles comparé à Mongoose. Cela pourrait être dû à une surcharge supplémentaire de Mongoose, qui est une couche ODM (Object Document Mapper) et ajoute des abstractions supplémentaires.

- Pour les autres types de requêtes (POST, DELETE, PUT, GET /rayons), les temps de réponse sont comparables entre MongoDB et Mongoose, avec une légère avance pour MongoDB.

Débit (req/s):

Le débit est presque identique, indiquant que les deux systèmes peuvent gérer un nombre similaire de requêtes par seconde.

Screenshots de l'interface React

Liste des articles :



Gestion du Magasin de Sport

Articles

Rayons

Créer un Article

Marque

Prix

Nom

Référence

Catégorie

Sélectionnez une catégorie



Tailles (séparées par des virgules)

Nom du Fournisseur

Adresse du Fournisseur

Rayon

Sélectionnez un rayon



Retour

Créer

Gestion du Magasin de Sport

Articles

Rayons

Modifier un Article

Marque

Nike

Prix

59.99

Nom

T-shirt Sport

Référence

TS123

Catégorie

Senior



Tailles (séparées par des virgules)

38,40,42

Nom du Fournisseur

Nike Inc

Adresse du Fournisseur

Beaverton, OR, USA

Rayon

Rayon Sportswear



Retour

Modifier

Liste des rayons :

Gestion du Magasin de Sport

Articles Rayons

Rayons

Créer un nouveau rayon

Description: test3
Employé Responsable: test

Supprimer Modifier

Description: Rayon Sportswear
Employé Responsable: Jean Dupont

Supprimer Modifier

Modification d'un rayon :

Gestion du Magasin de Sport

Articles Rayons

Modifier un Rayon

Description

Rayon Sportswear

Employé Responsable

Jean Dupont

Retour Modifier

Création d'un rayon :

Gestion du Magasin de Sport

Articles

Rayons

Créer un Rayon

Retour

Créer