TD 4 : MongoDB - Opérations CRUD et MapReduce

Alexandre Clenet - Florian Tran
Année 3 - Groupe 1-2

Exercice 1 : opérations de mise à jour

1- Insérer la collection produit dans mabase

```
db.produit.insertMany([
    { '_id': 1, 'libelle': 'chou', 'prixU': 5, 'qte': 7, 'dateFab': new
Date('2022-03-01T08:00:00Z') },
    { '_id': 2, 'libelle': 'tomate', 'prixU': 10, 'qte': 2, 'dateFab': new
Date('2021-03-01T08:00:00Z') },
    { '_id': 3, 'libelle': 'oignon', 'prixU': 20, 'qte': 1, 'dateFab': new
Date('2021-03-01T09:00:00Z') },
    { '_id': 4, 'libelle': 'salade', 'prixU': 5, 'qte': 10, 'dateFab': new
Date('2021-03-15T09:00:00Z') },
    { '_id': 5, 'libelle': 'tomate', 'prixU': 5, 'qte': 20, 'dateFab': new
Date('2021-04-04T11:21:39.736Z') },
    { '_id': 6, 'libelle': 'chou', 'prixU': 10, 'qte': 10, 'dateFab': new
Date('2021-04-04T21:23:13.331Z') },
    { '_id': 7, 'libelle': 'oignon', 'prixU': 7.5, 'qte': 5, 'dateFab': new
Date('2022-06-04T05:08:13Z') },
    { '_id': 8, 'libelle': 'tomate', 'prixU': 7.5, 'qte': 10,'dateFab': new
Date('2022-09-10T08:43:00Z')},
    { '_id': 9, 'libelle': 'tomate', 'prixU': 10, 'qte': 5, 'dateFab':new
Date('2023-02-06T20:20:13Z') }
]);
```

2- Insérer la collection vendeur dans mabase

```
db.vendeur.insertMany([
    {'_id':'f1','nom':'Alfred','statut':20,'ville':'Londres','ventes':[{ '_id': 1,
'qte': 7, 'date': new Date('2022-03-01T08:00:00Z') },{ '_id': 2, 'qte': 2, 'date':
new Date('2021-03-01T08:00:00Z')}]},
    {'_id':'f2','nom':'Robert','statut':10,'ville':'Paris','ventes':[{ '_id': 3,
'qte': 1, 'date': new Date('2021-03-01T09:00:00Z') },{ '_id': 4, 'qte': 10,
'date': new Date('2021-03-15T09:00:00Z')},
    { '_id': 1, 'qte': 2, 'date': new Date('2022-05-16T10:00:00Z')},{ '_id': 7,
'qte': 4, 'date': new Date('2022-03-01T09:00:00Z') }]},
    {'_id':'f3','nom':'Raymonde','statut':30,'ville':'Paris','ventes':[
    { '_id': 2, 'qte': 20, 'date': new Date('2021-04-04T11:21:39.736Z') },
    { '_id': 1, 'qte': 10, 'date': new Date('2021-04-04T21:23:13.331Z') },
    { '_id': 4, 'qte': 7, 'date': new Date('2022-010-23T21:23:13.331Z') },
    { '_id': 5, 'qte': 12, 'date': new Date('2022-09-21T21:23:13.331Z') },
    ]},
    {'_id':'f4','nom':'Gaston','statut':20,'ville':'Londres','ventes':[
```

```
    { '_id': 3, 'qte': 5, 'date': new Date('2022-06-04T05:08:13Z') },
    { '_id': 2, 'qte': 10, 'date': new Date('2022-09-10T08:43:00Z') }]},
    {'_id':'f5','nom':'Hector','statut':30,'ville':'Nantes',
    'ventes':[{ '_id': 2, 'qte': 5, 'date': new Date('2023-02-06T20:20:13Z') }]}
]);
```

3- Insérer dans la collection produit le document suivant (opérateur insert) : <_id: 10, libelle:orange, prixU: 20, dateFab: 2023-02-06T20:20:13Z>

```
db.produit.insertOne({ '_id': 1, 'libelle': 'orange', 'prixU': 20, 'dateFab': new
Date('2023-02-06T20:20:13Z') });
```

4- Ajouter le champs Etat au produit numéro 1. L'état de ce produit doit être initialisé à Frais. (opération update avec l'option set)

```
db.produit.updateOne({ '_id': 1 }, { $set: { 'etat': 'Frais' } });
```

5- Supprimer le champs état du produit numéro 1 (opérateur updateOne avec l'option unset).

```
db.produit.updateOne({ '_id': 1 }, { $unset: { 'etat': 1 } });
```

6- Supprimer le produit numéro 1 (opération delete)

```
db.produit.deleteOne({ '_id': 1 });
```

7- Supprimer la collection produit (opération drop)

```
db.produit.drop();
```

8- Insérer de nouveau le produit numéro 1 dans la collection produit

```
db.produit.insertOne({ '_id': 1, 'libelle': 'chou', 'prixU': 5, 'qte': 7,
'dateFab': new Date('2022-03-01T08:00:00Z') });
```

9- Remplacer le libellé du produit. Le nouveau libellé du produit est Orange (opérateur replace ou replaceOne)

```

```

```
db.produit.replaceOne({ '_id': 1 }, { 'libelle': 'Orange', 'prixU': 5, 'qte': 7,
'dateFab': new Date('2022-03-01T08:00:00Z') });
```

Exercice 2 : opérations de consultation

1- Afficher tous les documents dans produit

```
db.produit.find();
```

2- Afficher le nombre de documents dans produit

```
db.produit.count();
```

3- Afficher les produits qui sont fabriqués entre le 03/04/2021 et le 05/04/2023

```
db.produit.find({ 'dateFab': { $gte: new Date('2021-04-03'), $lte: new Date('2023-
04-05') }
```

4- Afficher tous les produits autres que Tomate

```
db.produit.find({ 'libelle': { $ne: 'tomate' } });
```

5- Afficher les prix des produits dont le libellé est composé de moins de 5 lettres (opérateur strLenCP)

```
db.produit.find({ $expr: { $lt: [{ $strLenCP: '$libelle' }, 5] } }, { 'prixU': 1,
'_id': 0 });
```

6- Afficher le prix et la date de fabrication des choux (opérateur aggregate avec match)

```
db.produit.aggregate([
  { $match: { 'libelle': 'chou' } },
  { $project: { 'prixU': 1, 'dateFab': 1, '_id': 0 } }
]);
```

7- Afficher la somme des prix de tous les produits (qte*prix)

```
db.produit.aggregate([
  { $project: { 'sommePrix': { $multiply: ['$qte', '$prixU'] } } },
  { $group: { '_id': null, 'total': { $sum: '$sommePrix' } } }
]);
```

8- Afficher le prix moyen de chacun des produits

```
db.produit.aggregate([
  { $group: { '_id': '$libelle', 'prixMoyen': { $avg: '$prixU' } } }
]);
```

9- Afficher tous les documents dans vendeur

```
db.vendeur.find();
```

10- Trier les vendeurs par ordre croissant de leur nom et décroissant de leur numéro d'identifiant

```
db.vendeur.find().sort({ 'nom': 1, '_id': -1 });
```

11- Liste des vendeurs localisés à Paris et à Londres

```
db.vendeur.find({ 'ville': { $in: ['Paris', 'Londres'] } });
```

12- Afficher le total des ventes (nombre de ventes)

```
db.vendeur.aggregate([
  { $unwind: '$ventes' },
  { $group: { '_id': null, 'totalVentes': { $sum: '$ventes.qte' } } }
]);
```

13- Afficher le total de ventes pour chacun des vendeurs

```
db.vendeur.aggregate([
  { $unwind: '$ventes' },
  { $group: { '_id': '$_id', 'totalVentes': { $sum: 1 } } }
]);
```

14- Afficher le total de ventes du produit numéro 3

```
db.vendeur.aggregate([
  { $unwind: '$ventes' },
  { $match: { 'ventes._id': 3 } },
  { $group: { '_id': '$_id', 'totalVentesProduit3': { $sum: '$ventes.qte' } } }
]);
```

15- Afficher le nombre de vendeurs qui ont vendu à la fois les produits 2 et 3

```
db.vendeur.aggregate([
  { $unwind: '$ventes' },
  { $match: { 'ventes._id': { $in: [2, 3] } } },
  { $group: { '_id': '$_id', 'count': { $sum: 1 } } },
  { $match: { 'count': { $eq: 2 } } }
]);
```

16- Afficher le nombre de vendeurs qui ont vendu seulement deux produits

```
db.vendeur.aggregate([
  { $unwind: '$ventes' },
  { $group: { '_id': '$_id', 'distinctProducts': { $addToSet: '$ventes._id' } } },
  { $match: { 'distinctProducts': { $size: 2 } } },
  { $count: 'totalVendeurs' }
]);
```

17- Afficher le nombre de vendeurs dans chaque ville

```
db.vendeur.aggregate([
  { $group: { '_id': '$ville', 'nombreVendeurs': { $sum: 1 } } }
]);
```

18- Afficher le Nom des vendeurs qui vendent des choux (opérateur aggregate avec unwind et lookup)

```
db.vendeur.aggregate([
  { $unwind: '$ventes' },
  { $lookup: { from: 'produit', localField: 'ventes._id', foreignField: '_id', as:
'produitInfo' } },
  { $match: { 'produitInfo.libelle': 'chou' } },
  { $project: { 'nom': 1, '_id': 0 } }
]);
```

Exercice 3 : Map Reduce

1- Insérer la collection facture dans mabase

```
db.facture.insertMany([
  { _idf: 1, idc: "Dupond", date: new Date("2023-02-01"), montant: 95, ventes: [{
idp: 1,
qte: 5 }, { idp: 4, qte: 2 }, { idp: 3, qte: 3 }], etat: "A" },
{ _idf: 2, idc: "Dupond", date: new Date("2023-03-05"), montant: 62, ventes: [{
idp:
2, qte: 4 }, { idp: 8, qte: 3 }], etat: "O" },
{ _idf: 3, idc: "Dupond", date: new Date("2023-07-10"), montant: 100, ventes: [{
idp:
7, qte: 2 }, { idp: 2, qte: 6 }, { idp: 1, qte: 5 }], etat: "O" },
{ _idf: 4, idc: "Lucas", date: new Date("2023-08-17"), montant: 52, ventes: [{
idp:
7, qte: 7 }], etat: "A" },
{ _idf: 5, idc: "Lucas", date: new Date("2023-03-20"), montant: 75, ventes: [{
idp:
1, qte: 1 }, { idp: 2, qte: 7 }], etat: "A" },
{ _idf: 6, idc: "Sophie", date: new Date("2023-03-21"), montant: 40, ventes: [{
idp:
3, qte: 2 }], etat: "O" },
{ _idf: 7, idc: "Eliot", date: new Date("2023-11-05"), montant: 45, ventes: [{
idp:
4, qte: 3 }, { idp: 2, qte: 3 }], etat: "A" },
{ _idf: 8, idc: "Eliot", date: new Date("2023-12-20"), montant: 25, ventes: [{
idp:
2, qte: 2 }, { idp: 1, qte: 1 }], etat: "O" },
{ _idf: 9, idc: "Martin", date: new Date("2023-05-10"), montant: 20, ventes: [{
idp:
5, qte: 4 }], etat: "O" },
{ _idf: 10, idc: "Martin", date: new Date("2023-11-04"), montant: 32, ventes: [{
idp:
4, qte: 2 }, { idp: 8, qte: 3 }], etat: "A" }
]);
```

2- Afficher le montant total payé par chaque client

```
var mapFunction = function () {
  emit(this.idc, this.montant);
};

var reduceFunction = function (key, values) {
  return Array.sum(values);
};

db.facture.mapReduce(
  mapFunction,
  reduceFunction,
  { out: { inline: 1 } }
);
```

3- Même question que 2 en stockant le résultat dans une nouvelle collection temporaire Temp1

```
db.facture.mapReduce(
  mapFunction,
  reduceFunction,
  { out: "Temp1" }
);
```

4- Afficher le contenu de Temp1 en triant les clients par ordre croissant de leur nom

```
db.Temp1.find().sort({ '_id': 1 });
```

5- Ecrire une autre version de la question 3 en remplaçant MapReduce par une agrégation

```
db.facture.aggregate([
  { $group: { '_id': '$idc', 'totalMontant': { $sum: '$montant' } } }
]);
```

6- Calculer le nombre de factures pour chacun des clients

```
var mapFunction6 = function () {
  emit(this.idc, 1);
};

var reduceFunction6 = function (key, values) {
  return Array.sum(values);
};

db.facture.mapReduce(
  mapFunction6,
  reduceFunction6,
  { out: { inline: 1 } }
);
```

7- Ecrire une autre version de la question 6 en remplaçant MapReduce par une agrégation

```
db.facture.aggregate([
  { $group: { '_id': '$idc', 'count': { $sum: 1 } } }
]);
```

8- Calculer le nombre d'apparition de chaque produit dans les factures

```
var mapFunction8 = function () {
  this.ventes.forEach(function (vente) {
    emit(vente.idp, 1);
  });
};

var reduceFunction8 = function (key, values) {
  return Array.sum(values);
};

db.facture.mapReduce(
  mapFunction8,
  reduceFunction8,
  { out: { inline: 1 } }
);
```

9- Calculer le nombre d'apparition des produits commandés par Eliot

```
var mapFunction9 = function () {
  var client = this.idc;
  this.ventes.forEach(function (vente) {
    emit({ idp: vente.idp, client: client }, 1);
  });
};

var reduceFunction9 = function (key, values) {
  return Array.sum(values);
};

db.facture.mapReduce(
  mapFunction9,
  reduceFunction9,
  { out: { inline: 1 }, query: { 'idc': 'Eliot' } }
);
```

10- Ecrire une autre version de la question 9 en remplaçant MapReduce par une agrégation

```
db.facture.aggregate([
  { $match: { 'idc': 'Eliot' } },
  { $unwind: '$ventes' },
  { $group: { '_id': { idp: '$ventes.idp', client: '$idc' }, 'count': { $sum: 1 }
} }
]);
```

11- Calculer la quantité totale de chaque produit commandé par les clients

```
var mapFunction11 = function () {
  this.ventes.forEach(function (vente) {
    emit(vente.idp, vente.qte);
  });
};

var reduceFunction11 = function (key, values) {
  return Array.sum(values);
};

db.facture.mapReduce(
  mapFunction11,
  reduceFunction11,
  { out: { inline: 1 } }
);
```

12- Même question que 11 en se limitant aux produits facturés avant le 10 mars 2023

```
var mapFunction12 = function () {
  var dateLimite = new Date('2023-03-10T00:00:00Z');
  this.ventes.forEach(function (vente) {
    if (this.date <= dateLimite) {
      emit(vente.idp, vente.qte);
    }
  });
};

db.facture.mapReduce(
  mapFunction12,
  reduceFunction11,
  { out: { inline: 1 } }
);
```

13- Calculer la quantité moyenne des produits facturés avant le 10 mars 2023

```
var mapFunction13 = function () {
  var dateLimite = new Date('2023-03-10T00:00:00Z');
  this.ventes.forEach(function (vente) {
    if (this.date <= dateLimite) {
      emit(vente.idp, { total: vente.qte, count: 1 });
    }
  });
};

var reduceFunction13 = function (key, values) {
  var reduced = { total: 0, count: 0 };
  values.forEach(function (value) {
    reduced.total += value.total;
    reduced.count += value.count;
```

```
  });
  return reduced;
};

var finalizeFunction13 = function (key, reduced) {
  return reduced.total / reduced.count;
};

db.facture.mapReduce(
  mapFunction13,
  reduceFunction13,
  { out: { inline: 1 }, finalize: finalizeFunction13 }
);
```