Université Hassan II Mohammedia Casablanca

Faculté des sciences Ben Msik



Atelier: Indexation

Master DSBD

Année universitaire: 2024-2025

Encadré par : Pr. Hanoune Mostapha

Réalisé par :

Bahafid Anouar , Mossati Ayoub , Bounit Meryem , Misbah Abdelhakim , Mohammed Anouar , Aboussad Mohammed Khalil , Chahid Achraf



Casablanca





Génération des données en utilisons python :

Installer les package nécessaires en exécutons cette commandes sur le terminale

pip install pymongo

Créer un fichier nommer data.py et écrire le code suivant :

this is the begining of the file

from pymongo import MongoClient

import random

from datetime import datetime, timedelta

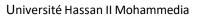
```
categories = ['Electronics', 'Clothing', 'Home', 'Books', 'Sports']
first_names = ['James', 'Mary', 'John', 'Patricia', 'Robert']
last_names = ['Smith', 'Johnson', 'Williams', 'Brown', 'Jones']

client = MongoClient("mongodb://localhost:27017")
db = client.ecommerce_workshop

# Generate products
products = [{
    "name": f"Product {i:05d}",
    "price": round(random.uniform(5.99, 199.99), 2),
    "category": random.choice(categories),
    "stock": random.randint(0, 500),
    "createdAt": datetime.now() - timedelta(days=random.randint(1, 365))
} for i in range(1, 50001)]

db.products.insert_many(products)
```

Generate users





Casablanca



Faculté des sciences Ben Msik

```
users = [{
  "firstName": random.choice(first_names),
  "lastName": random.choice(last_names),
  "email": f"user{i}@example.com",
  "createdAt": datetime.now() - timedelta(days=random.randint(1, 365))
} for i in range(1, 10001)]
db.users.insert_many(users)
# Generate orders (after products and users exist)
for _ in range(200000):
  user = random.choice(users)
  items = [{
    "productId": random.choice(products)["_id"],
    "quantity": random.randint(1, 5),
    "priceAtPurchase": round(random.uniform(5.99, 199.99), 2)
  } for _ in range(random.randint(1, 5))]
  db.orders.insert_one({
    "userId": user["_id"],
    "items": items,
    "status": random.choice(["pending", "shipped", "delivered", "cancelled"]),
    "totalAmount": round(sum(item["priceAtPurchase"] * item["quantity"] for item in items), 2),
    "createdAt": datetime.now()
  })
# this is the end of the file
```

Exécuter la commande suivante sur le terminal (nb : vous êtes dans le bon répertoire du fichier) : python data.py

Maintenant que nous avons généré notre dataset e-commerce (50k produits, 10k utilisateurs, 200k commandes)

On va créer une version améliorée et filtre de explain pour qu'il ne renvoie pas beaucoup d'informations non nécessaires



Casablanca





Entrer sur MongoShell et exécuter toute la commandes ci-dessous :

```
function explainFilter(cmd, options = {}) {
 const explain = cmd.explain("executionStats");
 const DEFAULT_FILTER = {
  stats: true,
  stages: true,
  indexes: true
 };
 const config = Object.assign(DEFAULT_FILTER, options);
 const result = {};
 if (config.stats) {
  result.stats = {
   execution Time Millis: explain. execution Stats. execution Time Millis,\\
   nReturned: explain.executionStats.nReturned,
   totalKeysExamined: explain.executionStats.totalKeysExamined,
   totalDocsExamined: explain.executionStats.totalDocsExamined
  };
 }
 if (config.stages) {
  result.stages = parseStages(explain.executionStats.executionStages);
 }
 if (config.indexes) {
  result.indexes = findUsedIndexes(explain.executionStats.executionStages);
 }
 return result;
}
function parseStages(stage, depth = 0) {
```



Casablanca



Faculté des sciences Ben Msik

```
const simplified = {type: stage.stage, docsExamined: stage.docsExamined, keysExamined:
stage.keysExamined, nReturned: stage.nReturned };
 if (stage.inputStage) {
  simplified.input = parseStages(stage.inputStage, depth + 1);
 }
 if (stage.executionStages) {
  simplified.shards = parseStages(stage.executionStages, depth + 1);
 }
 return simplified;
}
function findUsedIndexes(stage) {
 if (stage.indexName) return [stage.indexName];
 if (stage.inputStage) return findUsedIndexes(stage.inputStage);
 if (stage.executionStages) return findUsedIndexes(stage.executionStages);
 return ["COLLSCAN"];
}
```

Analyse des Requêtes Sans Index

Sur Mongoshell utiliser la bdd générer avec la commande suivante :

Use ecommerce_workshop

Après s'assurer que tous les index sont supprimés pour examiner les performances

```
db.products.dropIndexes();
db.orders.dropIndexes();
db.users.dropIndexes();
```

Test de Performance de Base

Requête 1: Trouver les produits d'une catégorie :



Casablanca



Faculté des sciences Ben Msik

```
const slowQuery = db.products.find({ category: "Electronics",price: { $gt: 100 }});
printjson(explainFilter(slowQuery, {verbose: true}));
explainFilter(db.products.find({ category: "Books" }});
Requête 2: Commandes récentes
const slowQuery2 = db.products.find({ category: "Electronics", price: { $lt: 1000 }, stock: { $gt: 10 }}).sort({ price: -1 });
printjson(explainFilter(slowQuery2, {verbose: true}));
```

Création d'Index Stratégiques

Index Simple

Création

db.products.createIndex({ category: 1 });

1 = ordre croissant

explainFilter(db.products.find({ category: "Books" }));

explainFilter(db.products.find({ category: "Electronics" }).limit(10));

Comparaison avec une requête utilisant à la fois category et price

explainFilter(db.products.find({ category: "Electronics", price: { \$gt: 100 } }).limit(10));

Analyse

const simpleIndexQuery = db.products.find({ category: "Electronics" });

printjson(explainFilter(simpleIndexQuery, {verbose: true}));

Index Compose

Création

db.products.createIndex({ category: 1, price: 1 });

Test d'optimisation







Faculté des sciences Ben Msik

```
explainFilter(db.products.find({ category: "Electronics", price: { $gt: 100 } }).limit(10));
Test d'ordre des champs dans l'index
explainFilter(db.products.find({ price: { $gt: 100 }, category: "Electronics" }).limit(10));
Test avec seulement une partie des champs indexés
explainFilter(db.products.find({ category: "Electronics" }).limit(10));
explainFilter(db.products.find({ price: { $gt: 100 } }).limit(10));
Index Multikey
Création
db.products.updateMany({}, { $set: { tags: ["promo", "nouveau"] } });
db.products.createIndex({ tags: 1 });
Analyse
explainFilter(db.products.find({ tags: "promo" }));
Index Textuel
db.products.updateMany({}, [{ $set: { name: { $concat: ["$name", " ", { $arrayElemAt: ["$tags", 0] }
db.products.createIndex({ name: "text" });
explainFilter(db.products.find({ $text: { $search: "electronics" } }).limit(10))
Index Unique
Application aux emails utilisateurs
db.users.createIndex({ email: 1 }, { unique: true });
Test de violation
try {
  db.users.insertOne({ email: "existant@example.com" });
```



Casablanca

Faculté des sciences Ben Msik





db.users.insertOne({ email: "existant@example.com" });

} catch (e) {

```
print("Erreur d'unicité: " + e.errmsg);
```

}

Un benchmark sert principalement à :

- Mesurer les performances : Temps d'exécution des requêtes, débit des opérations, etc.
- Comparer des configurations : Avant/après l'ajout d'un index, différentes structures de requêtes, etc.
- Identifier les goulots d'étranglement : Comprendre quelles opérations ralentissent le système

Préparation

```
function benchmarkQuery(queryFunc, iterations = 100) {
  let totalTime = 0;
  for (let i = 0; i < iterations; i++) {
    const start = new Date();
    queryFunc();
    totalTime += new Date() - start;
  }
  return totalTime / iterations;
}</pre>
```

Benchmark sans index

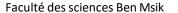
```
const noIndexTime = benchmarkQuery(() => {
  db.products.find({ category: "Electronics", price: { $gt: 100 } }).limit(10).toArray();
});
```

Benchmark avec index simple

```
const simpleIndexTime = benchmarkQuery(() => {
  db.products.find({ category: "Electronics" }).limit(10).toArray();
```



Casablanca





});

Benchmark avec index compose

```
const compoundIndexTime = benchmarkQuery(() => {
  db.products.find({ category: "Electronics", price: { $gt: 100 } }).limit(10).toArray();
});
```

Affichage des résultats

```
print("Temps moyen d'exécution:");
print(`- Sans index: ${noIndexTime} ms`);
print(`- Avec index simple: ${simpleIndexTime} ms`);
print(`- Avec index composé: ${compoundIndexTime} ms`);
```