**CRUD-операции в MongoDB**

**1. Основные CRUD-операции**

CRUD — это аббревиатура, представляющая основные операции: Create (создание), Read (чтение), Update (обновление) и Delete (удаление). Рассмотрим их применение в MongoDB.

**2. Вставка данных (Create)**

**insertOne**: добавляет один документ в коллекцию.

db.products.insertOne({

  "name": "Смартфон",

  "brand": "XYZ",

  "price": 25000,

  "stock": 50

})

**insertMany**: добавляет несколько документов.

db.products.insertMany([

  { "name": "Ноутбук", "brand": "ABC", "price": 60000, "stock": 20 },

  { "name": "Наушники", "brand": "XYZ", "price": 5000, "stock": 100 }

])

**3. Чтение данных (Read)**

**find**: возвращает документы, соответствующие фильтру.

Получение всех документов:

db.products.find()

Получение документов с фильтром:

db.products.find({ "price": { $gt: 10000 } })

**findOne**: возвращает первый найденный документ.

db.products.findOne({ "brand": "XYZ" })

**4. Обновление данных (Update)**

**updateOne**: обновляет первый документ, соответствующий фильтру.

db.products.updateOne(

  { "name": "Смартфон" },

  { $set: { "price": 24000 } }

)

**updateMany**: обновляет все документы, соответствующие фильтру.

db.products.updateMany(

  { "brand": "XYZ" },

  { $inc: { "stock": 10 } }

)

Операторы обновления:

* + $set: задаёт новое значение поля.
  + $inc: увеличивает или уменьшает числовое значение.
  + $unset: удаляет указанное поле.
  + $rename: переименовывает поле.

**5. Удаление данных (Delete)**

**deleteOne**: удаляет первый документ, соответствующий фильтру.

db.products.deleteOne({ "name": "Наушники" })

**deleteMany**: удаляет все документы, соответствующие фильтру.

db.products.deleteMany({ "brand": "XYZ" })

**6. Фильтры и операторы поиска**

Сравнительные операторы:

* + $eq: равно.
  + $ne: не равно.
  + $gt, $gte: больше, больше или равно.
  + $lt, $lte: меньше, меньше или равно.

Пример:

db.products.find({ "price": { $gte: 20000, $lte: 50000 } })

Логические операторы:

* + $and: логическое "И".
  + $or: логическое "ИЛИ".
  + $not: отрицание условия.

Пример:

* db.products.find({
* $or: [
* { "price": { $lt: 10000 } },
* { "stock": { $gt: 50 } }
* ]
* })

Поиск в массивах:

db.orders.find({ "items": { $in: ["Смартфон", "Наушники"] } })

**7. Индексация и её влияние на производительность**

Индексы ускоряют выполнение операций поиска, но увеличивают время вставки и обновления данных.

Создание индекса:

db.products.createIndex({ "price": 1 })

Индекс упорядочивает данные по полю price в порядке возрастания.

Проверка индексов:

db.products.getIndexes()

Удаление индекса:

db.products.dropIndex("price\_1")

**Преимущества индексации**:

* + Значительное ускорение поиска.
  + Улучшение производительности при использовании сложных запросов.

**Недостатки индексации**:

* + Увеличение объёма хранимых данных.
  + Замедление операций вставки и обновления из-за необходимости обновлять индекс.

**Примеры работы с индексами**

Поиск с использованием индекса:

db.products.find({ "price": { $gt: 30000 } }).explain("executionStats")