Summary about MongoDB, Redis, Neo4j, MapReduce

Author: Ilya Istomin

January 14, 2025

1 Введение

This document contains notes on MongoDB, Redis, Neo4j databases, MapReduce and Casandra concept.

2 MongoDB

MongoDB — нереляционная база данных, ориентированная на документы.

2.1 CRUD commands in MongoDB

2.1.1 Insert document (Create)

Вставляет один документ в коллекцию users.
Inserts a single document into the users collection.

```
    db.users.insertOne({
        "_id": 11,
        "firstName": "Lisa",
        "lastName": "Wong",
        "email": "lisa.wong@example.com",
        "age": 30,
        "username": "lisaw",
        "lastLogin": "2024-10-01"
      })
```

2.1.2 Insert many documents (Create)

Вставляет несколько документов в коллекцию products.
Inserts multiple documents into the products collection.

```
db.products.insertMany([
  { "_id": 11, "name": "Tablet", "description": "Portable tablet with HD screen", "price": 600, "category": "Electronics" },
  { "_id": 12, "name": "Microwave", "description": "Compact microwave oven", "price": 90, "category": "Home Appliances" }
])
```

2.1.3 Find document (Read)

Ищет и возвращает один документ из users по _id.
Finds and returns a single document from users by _id.

```
db.users.findOne({ "_id": 1 })
```

2.1.4 Find many documents (Read)

Ищет всех пользователей, чей возраст больше 30 лет.
Finds all users whose age is greater than 30.

```
db.users.find({ "age": { $gt: 30 } })
```

2.1.5 Update document (Update)

Обновляет документ в коллекции users.
Updates a document in the users collection.

```
db.users.updateOne(
    { "_id": 1 },
    { $set: { "email": "john.doe.new@example.com" } }
    )
```

2.1.6 Update many documents (Update)

Увеличивает цену всех продуктов на 10%.
Increases the price of all products by 10%.

```
db.products.updateMany(
    {},
    { $mul: { "price": 1.1 } }
)
```

2.1.7 Upsert document (Update)

Обновляет документ в коллекции users или вставляет новый документ, если документ не найден.
Updates a document in the users collection or inserts a new document if the document is not found.

```
db.users.updateOne(
  { "username": "newuser" },
  {
    $set: {
      "firstName": "New",
      "lastName": "User",
      "email": "new.user@example.com",
      "age": 25,
      "lastLogin": "2024-11-01"
    }
  },
  { upsert: true }
)
```

2.1.8 Delete one document (Delete)

Удаляет один документ из коллекции products.
Deletes a single document from the products collection.

```
db.products.deleteOne({ "_id": 1 })
```

2.1.9 Delete many documents (Delete)

Удаляет всех пользователей, которые не входили в систему после 1 мая 2024 года.
Deletes all users who have not logged in after May 1, 2024.

```
db.users.deleteMany(
  { "lastLogin": { $lt: "2024-05-01" } }
)
```

2.1.10 Indexes

Индексы это структуры данных, которые ускоряют поиск документов в коллекции.

Составной индекс это индекс, который содержит несколько полей.

```
db.users.createIndex({ "firstName": 1, "lastName": 1 })
```

Текстовый индекс это индекс, который позволяет выполнять текстовый поиск в полях строкового типа.

```
db.products.createIndex({ "description": "text" })
```

Уникальный индекс это индекс, который гарантирует уникальность значений в поле или полях.

```
db.users.createIndex({ "username": 1 }, { unique: true })
```

2.1.11 Text search

Полнотекстовый поиск позволяет выполнять поиск по текстовым полям.
Full-text search allows you to search text fields.

```
db.products.find({ $text: { $search: "oven" } })
```

Поиск с условием AND.
Search with AND condition.

```
db.products.find({
    $and: [
        { "price": { $gt: 50 } },
        { $text: { $search: "oven" } }
    ]
    })
```

2.1.12 Geo queries

Создание гео индекса.
Create a geo index.

```
db.places.createIndex({ "location": "2dsphere" })
```

Поиск всех мест, находящихся в радиусе 5 км от координат
Find all places within a 5 km radius of the coordinates.

```
db.places.find({
"location": {
    $near: {
    $geometry: { type: "Point", coordinates: [-73.9857,
       40.7484] },
    $maxDistance: 5000
    }
}
})
```

Поиск всех мест, отсортированных по близости к координатам.
Find all places sorted by proximity to the coordinates.

```
db.places.find({
    "location": {
    $near: {
        $geometry: { type: "Point", coordinates: [-73.9857,
            40.7484] }
    }
    }
})
```

2.1.13 Logical operators

Поиск мест с использованием оператора OR.
Find places using the OR operator.

```
db.places.find({
    $or: [
        { "name": { $regex: "the", $options: "i" } },
        { "name": { $regex: "of", $options: "i" } }
    ]
})
```

Поиск продуктов в категориях "Electronics" или "Furniture".
Find products in the "Electronics" or "Furniture" categories.

```
db.products.find({
    "category": { $in: ["Electronics", "Furniture"] }
})
```

Поиск мест у которых обе координаты меньше 0.
Find places where both coordinates are less than 0.

```
db.places.find({
    "location.coordinates.0": { $lt: 0 },
    "location.coordinates.1": { $lt: 0 }
})
```

2.1.14 Aggregation

Все ключи аггрегации

2.1.15 Фильтрация и выборка данных

- **$match**: Фильтрация документов (аналог find())
- **$project**: Выбор полей и создание вычисляемых значений
- **$limit**: Ограничение количества документов
- **$skip**: Пропуск определенного количества документов

2.1.16 Группировка и сортировка

- **$group**: Группировка документов (аналог GROUP BY)
- **$sort**: Сортировка результатов

2.1.17 Работа с массивами

- **$unwind**: Разворачивает массив (каждый элемент создаёт отдельный документ)

2.1.18 Операции с несколькими коллекциями

- **$lookup**: Объединение данных из разных коллекций (аналог JOIN)
- **$graphLookup**: Рекурсивный поиск (например, в древовидных структурах)

2.1.19 Вывод результатов

- **$out**: Записывает результат в новую коллекцию
- **$merge**: Обновляет или добавляет данные в существующую коллекцию

2.1.20 Операторы агрегации

2.1.21 Математические операторы

- **$sum**: Суммирует значения
- **$avg**: Вычисляет среднее значение
- **$min**: Находит минимальное значение
- **$max**: Находит максимальное значение
- **$add**: Сложение чисел
- **$subtract**: Вычитание
- **$multiply**: Умножение
- **$divide**: Деление
- **$mod**: Остаток от деления

2.1.22 Операторы строк

- **$concat**: Объединяет строки
- **$substr**: Извлекает подстроку
- **$toUpper**: Преобразует в верхний регистр
- **$toLower**: Преобразует в нижний регистр

2.1.23 Операторы массивов

- **$push**: Добавляет элементы в массив
- **$addToSet**: Добавляет уникальные элементы в массив
- **$size**: Возвращает размер массива
- **$filter**: Фильтрует массив

2.1.24 Операторы условий

- **$cond**: Если-иначе (аналог if-else)
- **$ifNull**: Возвращает значение, если не null

2.1.25 Логические операторы

- **$and**: Логическое И
- **$or**: Логическое ИЛИ
- **$not**: Логическое НЕ

2.1.26 Операторы даты

- **$year**: Получает год из даты
- **$month**: Получает месяц
- **$dayOfMonth**: Получает день месяца
- **$hour**: Получает час
- **$second**: Получает секунду

3 Redis

Redis — это хранилище данных типа "ключ-значение".

3.1 Пример команды в Redis

```
SET user:1 "John Doe"
```

Listing 1: Добавление значения

4 Neo4j

Neo4j — графовая база данных.

4.1 Пример запроса в Neo4j

```
CREATE (p:Person {name: "Alice"})
```

Listing 2: Создание узла

5 MapReduce

MapReduce — модель обработки больших данных.

5.1 Пример псевдокода

```
def map(key, value):
    for word in value.split():
        emit(word, 1)

def reduce(key, values):
    emit(key, sum(values))
```

Listing 3: MapReduce

6 Cassandra

Cassandra — это распределённая NoSQL база данных, предназначенная для обработки больших объёмов данных, обеспечивающая высокую доступность и целостность данных.

6.1 Основные характеристики

- Децентрализация (высокая доступность)
- Репликация
- Отказоустойчивость
- Масштабируемость

6.2 Архитектура Cassandra

6.2.1 Основные элементы

- **Узел (Node)** – отдельный экземпляр Cassandra.
- **Раздел (Partition)** – базовая единица информации, важная для репликации и организации данных.
- **Стойка (Rack)** – логическая группа узлов.
- **Дата-центр (Data Center)** – логическая группа стоек.
- **Кластер (Cluster)** – полный набор узлов (структура полного кольца токенов).

6.2.2 Разделение данных (Partitioning)

- Partition – базовая единица хранения.
- Partition Key – ключ, определяющий расположение данных в базе.
- Partitioner – механизм распределения данных по узлам на основе ключа.
- Кольцо токенов (Token Ring) – распределение разделов между узлами.

6.2.3 Виртуальные узлы (vNodes)

- В Cassandra 2.0 узлы делятся на виртуальные подузлы.
- Основные процессы:
  - **Bootstrap** – автоматическое перераспределение сегментов токенов при добавлении нового узла.
  - **Decommissioning** – перераспределение сегментов при удалении узла.

6.2.4 Репликация (Replication)

Cassandra обеспечивает высокую доступность и отказоустойчивость за счёт репликации данных.

- **Replication Factor** – количество узлов, хранящих копии данных.
- **Replication Strategy** – стратегия распределения реплик:
  - SimpleStrategy – последовательное распределение реплик.
  - NetworkTopologyStrategy – реплики распределяются по разным дата-центрам.

6.2.5 Уровень согласованности (Consistency Level)

Определяет, сколько узлов должны ответить на запрос до того, как клиент получит подтверждение.

- **Варианты уровней согласованности**:
  - ONE, TWO, THREE – ответ от 1, 2 или 3 узлов.
  - ANY – ответ от любого узла.
  - ALL – ответ от всех узлов.
  - QUORUM – 51

6.3 Поток записи (Write Path)

Cassandra выполняет запись данных в несколько этапов:

1. **Memtable (в памяти)** – временное хранение записей.
2. **CommitLog (на диск)** – защита от потери данных.
3. **SSTable (на диск)** – окончательная запись.
4. **Компактизация** – объединение SSTables для оптимизации запросов.

6.4 Поток чтения (Read Path)

Cassandra выполняет чтение данных по следующему процессу:

1. Поиск в Memtable.
2. Поиск в SSTable (используется Partition Index и Partition Summary).
3. Использование Bloom Filter для ускоренного поиска.
4. Слияние данных (merge) и возврат клиенту.

6.5 Репликация данных и согласованность

Cassandra использует репликацию, чтобы данные были доступны даже при сбоях.

- **Hinted Handoff** – временное хранение изменений до восстановления узла.
- **Read Repair** – обновление устаревших данных при чтении.
- **Anti-Entropy Repair** – процесс фоновой синхронизации данных между узлами.