**Структура данных в MongoDB**

**1. Документно-ориентированная модель данных**

В MongoDB данные хранятся в виде документов, которые представляют собой JSON-подобные структуры. Каждый документ — это запись, содержащая пары "ключ-значение".

**Особенности модели данных**:

Гибкость структуры: Документы одной коллекции могут иметь разное количество полей и разные типы данных.

Иерархическая организация: Поля документа могут содержать вложенные документы или массивы.

Отсутствие строгой схемы: Не требуется предварительное определение структуры данных, как в реляционных БД.

Пример документа:

{

  "name": "Иван Иванов",

  "age": 30,

  "contacts": {

    "email": "ivan.ivanov@example.com",

    "phone": "+7-999-123-45-67"

  },

  "skills": ["Python", "MongoDB", "JavaScript"]

}

Здесь поле contacts является вложенным документом, а skills — массивом.

**2. Формат BSON: особенности и преимущества**

BSON (Binary JSON) — это двоичный формат сериализации, используемый MongoDB для хранения и передачи данных.

**Особенности BSON**:

Поддержка сложных типов данных: Включает такие типы, как Date, Binary, Decimal128, которые отсутствуют в обычном JSON.

Быстродействие: Формат оптимизирован для быстрого чтения и записи.

Компактность: Занимает меньше места, чем JSON, благодаря двоичной структуре.

Преимущества BSON перед JSON:

* 1. Эффективность для хранения больших объёмов данных.
  2. Поддержка дополнительных типов данных.
  3. Быстрое выполнение операций благодаря бинарному формату.

**3. Коллекции и документы**

В MongoDB данные организованы в коллекции, которые аналогичны таблицам в реляционных БД, а документы внутри коллекции — это строки (записи).

**Создание коллекций**

В MongoDB коллекции создаются автоматически при добавлении первого документа, а также можно создать коллекцию вручную:

db.createCollection("users")

**Добавление документов**

Пример вставки одного документа:

db.users.insertOne({

  "name": "Иван Иванов",

  "age": 30,

  "email": "ivan.ivanov@example.com"

})

Пример вставки нескольких документов:

db.users.insertMany([

  { "name": "Пётр Петров", "age": 25, "email": "petr.petrov@example.com" },

  { "name": "Мария Смирнова", "age": 28, "email": "maria.smirnova@example.com" }

])

**Чтение данных**

Получить все документы коллекции:

db.users.find()

Результат:

[

  { "\_id": ObjectId("64e23f22..."), "name": "Иван Иванов", "age": 30, "email": "ivan.ivanov@example.com" },

  { "\_id": ObjectId("64e23f23..."), "name": "Пётр Петров", "age": 25, "email": "petr.petrov@example.com" }

]

**Фильтрация документов**:

db.users.find({ "age": { $gte: 28 } })

**Обновление документов**

Обновление одного документа:

db.users.updateOne(

  { "name": "Иван Иванов" },

  { $set: { "age": 31 } }

)

Обновление нескольких документов:

db.users.updateMany(

  { "age": { $lt: 30 } },

  { $set: { "status": "молодой специалист" } }

)

**Удаление документов**

Удаление одного документа:

db.users.deleteOne({ "name": "Пётр Петров" })

Удаление всех документов, соответствующих фильтру:

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