MongoDB

📘 Основы работы с MongoDB и NoSQL

1. Введение в базы данных

База данных (БД) — это организованная совокупность информации, предназначенная для хранения, изменения и извлечения данных.

На практике используются два основных типа баз данных:

- **Реляционные базы данных (SQL)** — основаны на таблицах и строгой схеме.
- **Нереляционные базы данных (NoSQL)** — гибкие, масштабируемые, используют различные структуры хранения данных.

2. Что такое NoSQL?

NoSQL (Not Only SQL) — класс систем управления базами данных, не использующих традиционную табличную модель. Вместо таблиц используется один из следующих форматов:

- Документы (MongoDB)
- Ключ-значение (Redis)
- Графы (Neo4j)
- Колонки (Cassandra)

Типы NoSQL-баз данных

Тип	Описание	Примеры
Документно-ориентированные	Хранят данные в формате JSON/BSON	MongoDB, CouchDB
Колоночные	Данные представлены в виде столбцов	Apache Cassandra, HBase
Ключ-значение	Простое сопоставление "ключ — значение"	Redis, DynamoDB
Графовые	Данные организованы как графы (узлы и связи)	Neo4j, ArangoDB

3. Разница между SQL и NoSQL

Критерий	SQL (реляционные)	NoSQL (нереляционные)
Структура данных	Таблицы с фиксированной схемой	Документы, графы, колонки, ключ-значение
Гибкость	Слабая (жесткая схема)	Высокая (гибкая схема)
Масштабирование	Вертикальное (мощность сервера)	Горизонтальное (добавление узлов)
Запросы	Язык SQL, JOIN'ы	Запросы в формате JSON-подобных фильтров
Транзакции	ACID (жесткие гарантии)	BASE (гибкие гарантии)
Производительность	Высокая при сложных запросах	Высокая при большом объёме и частоте записей

4. Когда использовать SQL, а когда NoSQL?

Использовать SQL, если:

- Структура данных стабильна.
- Нужна **строгая согласованность** и транзакции.
- Часто используются сложные **JOIN-запросы**.
- Важно соответствие стандартам ACID.

Примеры:

- Банковские и финансовые системы.
- ERP, бухгалтерия.
- Медицинские базы данных.

Использовать NoSQL, если:

- Данные имеют **гибкую или меняющуюся структуру**.
- Необходимо **масштабирование и высокая скорость**.
- Данные представляют собой **вложенные структуры**.
- Не требуется жесткое соблюдение транзакционности.

Примеры:

- Веб-приложения, соцсети.
- Каталоги товаров.
- Логи, телеметрия, big data.

---

5. MongoDB: Общая информация

MongoDB — одна из самых популярных документно-ориентированных NoSQL баз данных. Основной единицей хранения данных является **документ** (формат BSON, совместим с JSON).

Преимущества MongoDB:

- Гибкая структура данных.
- Встроенная масштабируемость.
- Быстрая разработка и внедрение.
- Поддержка геолокации, индексов, агрегаций.
- Удобный графический интерфейс MongoDB Compass.

---

6. Основные сущности MongoDB

Сущность	Описание
База данных	Контейнер для коллекций
Коллекция	Аналог таблицы, содержит документы
Документ	Структурированные данные (JSON / BSON)
Поле	Пара "ключ-значение" внутри документа

Пример документа:

```
json

{
  "_id": ObjectId("665d1a..."),
  "name": "Иван",
  "age": 30,
  "email": "ivan@example.com",
  "skills": ["JavaScript", "MongoDB"],
  "address": {
    "city": "Москва",
    "zip": "101000"
  }
}
```

---

7. CRUD-операции в MongoDB

CRUD — это 4 основные операции в работе с базой данных:

🔷 C — Create (Создание)

`insertOne()` — вставка одного документа:

```
js

db.users.insertOne({
  name: "Анна",
  age: 27,
  email: "anna@example.com"
})
```

`insertMany()` — вставка массива документов:

```
js

db.users.insertMany([
  { name: "Петр", age: 35 },
  { name: "Светлана", age: 29 }
])
```

🔷 R — Read (Чтение)

`find()` — получение всех документов:

```
js

db.users.find()
```

`find()` с фильтром:

```
js

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

`findOne()` — получить только один документ:

```
js

db.users.findOne({ name: "Петр" })
```

🔷 U — Update (Обновление)

`updateOne()` — обновить одно совпадение:

```
js

db.users.updateOne(
  { name: "Анна" },
  { $set: { age: 28 } }
)
```

`updateMany()` — обновить все совпадения:

```
js

db.users.updateMany(
  { age: { $lt: 30 } },
  { $set: { status: "молодой" } }
)
```

Операторы обновления:

Оператор	Назначение
`$set`	Установить новое значение
`$inc`	Увеличить/уменьшить числовое поле
`$push`	Добавить элемент в массив
`$pull`	Удалить элемент из массива
`$unset`	Удалить поле

🔷 D — Delete (Удаление)

`deleteOne()` — удалить один документ:

```
js

db.users.deleteOne({ name: "Петр" })
```

`deleteMany()` — удалить все подходящие документы:

```
js

db.users.deleteMany({ age: { $gt: 40 } })
```

8. Операторы фильтрации MongoDB

Оператор	Назначение	Пример
`$eq`	Равно	`{ age: { $eq: 30 } }`
`$ne`	Не равно	`{ name: { $ne: "Иван" } }`
`$gt`, `$lt`	Больше / меньше	`{ age: { $gt: 18 } }`
`$in`	Совпадает с одним из значений	`{ city: { $in: ["Москва", "Питер"] } }`
`$and`, `$or`	Логические условия	`{ $and: [ { age: {$gt: 18} }, { city: "Москва" } ] }`

`$regex`	Регулярное выражение	`{ name: { $regex: "^A" } }`

9. Преимущества MongoDB Compass

- Визуальное отображение базы данных и коллекций.
- Конструктор запросов и фильтров.
- Быстрый просмотр, редактирование и вставка документов.
- Импорт/экспорт в CSV и JSON.
- Анализ индексов и производительности запросов.

10. Заключение: что выбрать — SQL или NoSQL?

Выбор между SQL и NoSQL зависит от конкретных задач, требований проекта и ограничений инфраструктуры. Оба подхода имеют свои сильные и слабые стороны, и ни один из них не является универсальным решением.

Когда стоит выбрать SQL:

- Данные чётко структурированы и подчиняются строгой схеме (schema).
- Важна целостность и согласованность данных (ACID-транзакции).
- Проект требует сложных запросов, JOIN-операций и аналитики.
- Используется традиционная реляционная модель (например, банковские системы, бухгалтерия).

Когда стоит выбрать NoSQL:

- Требуется высокая масштабируемость и быстрая обработка больших объёмов данных.
- Структура данных гибкая или может меняться со временем.
- Преобладает горизонтальное масштабирование (распределённые системы).

- Система должна обеспечивать высокую доступность и отказоустойчивость (например, социальные сети, интернет-магазины, системы рекомендаций).

В идеале, выбор должен основываться не на моде или популярности технологий, а на чётком понимании задач проекта и характера обрабатываемых данных. В некоторых случаях оправдано комбинирование обоих подходов: **Polyglot Persistence** — использование нескольких типов баз данных в одном проекте в зависимости от контекста.