Введение

Перед тем как рассматривать MongoDB, давайте разберемся: что такое вообще базы данных NoSQL, к которым относится MongoDB?

NoSQL, также известные как "not only SQL", не являются табличными БД и хранят данные иначе, чем реляционные базы вроде Postgresql.

Базы данных NoSQL бывают разных типов в зависимости от их модели данных. Основные типы:

- документоориентированная
- key-value
- колоночная
- графовая

Создавались они для обработки больших объемов данных и более легкого масштабирования под высокие нагрузки.

Документоориентированные БД хранят данные в документах, подобных объектам JSON (JavaScript Object Notation). Каждый документ содержит пары полей и значений. Значения могут быть различных типов, включая такие типы, как строки, числа, bool, массивы или объекты.

Базы данных «ключ-значение» — более простой тип БД, где каждый элемент содержит ключи и значения.

Колоночные базы данных похожи на реляционные, за тем исключением, что данные хранятся не по строкам, а в колонках. Такие базы применяются для построения аналитики в реальном времени. На основе такой БД построена «Яндекс.Метрика».

Графовые базы данных хранят данные в узлах и ребрах в виде графов. Узлы обычно хранят информацию о людях, местах и вещах, а ребра хранят информацию о взаимосвязях между узлами.

Примеры:

- документоориентированная: https://www.couchbase.com/, https://www.mongodb.com/
- key-value: https://redis.io/, https://www.memcached.org/
- колоночная: https://clickhouse.com/
- графовая: https://neo4j.com/

Отличие реляционной схемы хранения данных от NoSQL

Несмотря на то, что между системами управления реляционными базами данных (СУБД) и базами данных NoSQL существует множество различий, одним из ключевых различий является способ хранения данных в БД.

Чтобы разобраться в отличиях между реляционными БД и NoSQL-БД, рассмотрим пример хранения информации о пользователе и его увлечениях. Нам нужно сохранить имя, фамилию, номер мобильного телефона, город и увлечения пользователя.

В реляционной базе данных мы создадим две таблицы: пользователи и хобби.

Users

ID	first_name	last_name	cell	city
1	Leslie	Yepp	8125552344	Pawnee

Hobbies

ID	user_id	hobby
10	1	scrapbooking
11	1	eating waffles
12	1	working

Чтобы получить информацию о пользователе и о его хобби одним запросом, нужно соединить 2 таблицы и написать примерно такой запрос:

```
SELECT u.*, h.hobby FROM Users AS u LEFT JOIN Hobbies AS h ON (h.user_id = u.user_id)
WHERE u.id = 1
```

В случае с такой документоориентированной БД, как MongoDB, можно сложить данные в один документ. Тогда схема данных будет выглядить следующим образом:

```
{
   "_id": "uuid",
   "first_name": "Leslie",
   "last_name": "Yepp",
   "cell": "8125552344",
   "city": "Pawnee",
   "hobbies": ["scrapbooking", "eating waffles", "working"]
}
```

Вся информация хранится внутри одного «документа», а данные можно получить за один запрос без соединений таблиц.

Преимущества NoSQL-подхода

Гибкие модели данных. Базы данных NoSQL обычно имеют гибкие схемы, то есть записи в одной таблице (в терминах реляционных БД) могут иметь разный формат. Гибкая схема позволяет легко вносить изменения в базу данных по мере изменения требований.

Но в таком подходе есть существенный минус: нет фиксированной структуры, нет схемы данных, а значит, нужно быть готовым к сюрпризам: например, отстутствию полей, недопустимым типам данных и т.д.

Между реляционными и NoSQL-базами можно провести аналогию с языками программирования с динамической и статической типизацией. К примеру, Python и Golang или C++. В Python переменной можно присвоить любой тип, и это дает свои преимущества: код писать быстрее, его проще тестировать и т.д. Но в то же время программист не всегда знает, какой тип принимает или возвращает функция. Иногда сложно догадаться, какая вообще будет вызвана функция, так как она была добавлена «черной магией» Питона.

Горизонтальное масштабирование. Для большинства БД SQL требуется вертикальное масштабирование — переход на более крупный и более дорогой сервер, когда нагрузка превышает возможности текущего. И наоборот, большинство баз данных NoSQL позволяют масштабироваться горизонтально: вы можете добавлять более дешевые стандартные серверы, когда нужно.

Быстрые запросы. Запросы в базах данных NoSQL могут быть быстрее, чем в базах данных SQL. Почему? Данные в SQL-БД обычно нормализованы, поэтому запросы для одного объекта или сущности требуют объединения данных из нескольких таблиц. По мере роста ваших таблиц соединения могут стать дорогостоящими. Однако данные в базах данных NoSQL обычно хранятся так, чтобы оптимизировать их для запросов. Эмпирическое правило при использовании MongoDB — данные, которые доступны вместе, должны храниться вместе. Запросы обычно не требуют объединений, поэтому могут выполняться быстрее.

В этой главе мы не будем рассматривать все возможности MongoDB, это тема для отдельного курса.

Мы научимся работать с асинхронным коннектором для Mongo и попрактикуемся писать простые запросы CRUD (create read update delete). Если вы не сталкивались с этой базой данных, то рекомендую изучить материалы MongoDB-университета:

https://university.mongodb.com/courses/M001/about

https://university.mongodb.com/courses/M220P/about

Полезные ссылки

https://www.mongodb.com/nosql-explained

https://www.mongodb.com/nosql-explained/nosql-vs-sql

https://www.mongodb.com/nosql-explained/when-to-use-nosql