## **Доклад о MongoDB**

### **Введение**

MongoDB — это документо-ориентированная NoSQL база данных, разработанная для работы с большими объемами данных и обеспечения высокой масштабируемости. Название "Mongo" происходит от слова "humongous" (огромный). Написана на языке C++. MongoDB разработана компанией 10gen (сейчас MongoDB Inc.) в 2007 году как часть платформы для облачных вычислений, но в процессе была выделена как отдельный продукт. Первая версия MongoDB была выпущена в 2009 году, и с тех пор она стала одной из самых популярных NoSQL-систем благодаря своей простоте и гибкости.

### **Архитектура и принципы работы**

MongoDB относится к документо-ориентированным NoSQL базам данных, где данные хранятся в виде документов BSON (Binary JSON), а не строк, как в реляционных СУБД. BSON представляет собой расширенную версию JSON, которая позволяет хранить дополнительные типы данных и обеспечивает более быструю обработку.

#### **Основные концепции MongoDB:**

* **Документ** — основная единица хранения данных, представляет собой набор пар "ключ-значение".
* **Коллекция** — аналог таблицы в реляционных базах, но без жесткой схемы.
* **\_id** — уникальный идентификатор документа (эквивалент первичного ключа в SQL).
* **Гибкость структуры** — нет необходимости в строгом определении схемы данных, можно легко изменять структуру без миграций.
* **Отсутствие NULL** — если у ключа нет значения, он просто не включается в документ.

### **Репликация, индексация и шардинг**

#### **Репликация**

MongoDB поддерживает механизм репликации для обеспечения отказоустойчивости. Репликационные наборы состоят из одного главного узла (**primary**) и нескольких вторичных (**secondary**). Данные автоматически реплицируются между узлами. В случае отказа главного узла один из вторичных автоматически становится новым главным. Также можно настроить чтение с вторичных узлов (**read preference**) для балансировки нагрузки.

#### **Индексация**

MongoDB поддерживает индексы для ускорения поиска данных. Индексы создаются на любом поле документа и хранятся в виде **B-деревьев** (по умолчанию). Грамотное использование индексов существенно повышает производительность запросов.

#### **Шардинг (Sharding)**

Шардинг применяется для горизонтального масштабирования MongoDB. Данные распределяются по нескольким серверам (шардам) с помощью специального ключа шардирования (**shard key**).

* **Balancer** автоматически перераспределяет данные между узлами.
* **Репликационные наборы в шардах** обеспечивают отказоустойчивость.
* **Клиенты взаимодействуют с базой через mongos**, который скрывает сложность шардирования.
* **Преимущества**: высокая масштабируемость.
* **Недостатки**: сложность настройки и управления по сравнению с SQL.

#### **CAP-теорема**

Согласно CAP-теореме, распределенные базы данных не могут одновременно удовлетворять всем трем критериям:

* **Consistency (согласованность)** — все узлы видят одни и те же данные в одно и то же время.
* **Availability (доступность)** — каждый запрос получает ответ (даже если произошел сбой узла).
* **Partition tolerance (устойчивость к разделению сети)** — система продолжает работать, даже если произошел разрыв связи между узлами.

MongoDB реализует модель **AP (Availability & Partition tolerance)**, но с возможностью выбора согласованности при настройке репликации. Это означает, что в обычном режиме при отказе узлов база остается доступной, но может некоторое время работать с устаревшими данными.

### **Функциональные возможности**

#### **CRUD-операции**

MongoDB поддерживает стандартные операции:

* **Создание** (Create)
* **Чтение** (Read)
* **Обновление** (Update)
* **Удаление** (Delete)

Запросы используют JSON-подобный синтаксис.

#### **Агрегация данных**

MongoDB поддерживает мощный механизм агрегации через **Aggregation Pipeline**, аналог SQL GROUP BY:

* $match — фильтрация.
* $group — группировка.
* $sort — сортировка.
* $project — проекция.

#### **Поиск и индексация**

* **Полнотекстовый поиск** (Text Search) позволяет выполнять запросы по текстовым данным.
* **Геопространственные запросы** используются для работы с картографическими данными.

### **Хранение больших данных**

Ограничение MongoDB на размер одного документа — **16 МБ**. Для хранения больших файлов используется **GridFS**:

* **Коллекция files** — хранит метаданные файлов (имя, размер и т. д.).
* **Коллекция chunks** — хранит фрагменты данных (обычно по 256 КБ).

### **Сравнение с SQL и другими NoSQL**

**Сравнение SQL и MongoDB**

| **Функция** | **SQL** | **MongoDB** |
| --- | --- | --- |
| Хранение данных | Структурированные таблицы | Документы BSON |
| Жесткость схемы | Обязательное определение схемы | Гибкая структура |
| Индексация | B-деревья, полнотекстовый поиск | B-деревья, полнотекстовый поиск |
| Связи между таблицами/документами | JOIN (строгие связи) | Вложенные документы, $lookup (аналог JOIN) |
| Транзакции | ACID, сложные транзакции | Поддержка multi-document транзакций (но менее мощных) |
| Масштабирование | Вертикальное (мощное оборудование) | Горизонтальное (шардинг) |
| Производительность | Высокая для сложных аналитических запросов | Оптимизирована для оперативных данных |

#### **Преимущества MongoDB перед SQL**

* **Гибкость** (нет жесткой схемы, легко изменять структуру).
* **Высокая масштабируемость** (шардинг, репликация).
* **Эффективная работа с неструктурированными данными**.

#### **Недостатки**

* **Сложности с поддержанием целостности данных**, так как нет строгих связей (JOIN).
* **Менее эффективные аналитические запросы** (в отличие от SQL).

#### **Сравнение с другими NoSQL**

* По сравнению с **ключ-значностными базами** (например, Redis) MongoDB сложнее, но предлагает больше возможностей для структурированных данных.
* В отличие от **колоночных баз данных** (например, Cassandra), MongoDB проще в использовании для приложений с динамической схемой данных.
* **Документо-ориентированная структура** делает MongoDB более удобной для хранения сложных и вложенных данных по сравнению с графовыми базами данных.

### **Интеграция с другими инструментами**

MongoDB легко интегрируется с внешними системами:

* **Драйверы**: доступны для Python (PyMongo), Java, Node.js и других языков.
* **Графические утилиты**:
  + **MongoDB Compass** — графический интерфейс для работы с базой.
  + **mongo shell** — CLI для выполнения запросов.
* **BI Connector** — позволяет выполнять SQL-запросы к MongoDB через BI-инструменты.
* **Dremio, Presto** — SQL-движки, работающие с MongoDB.
* **MongoDB Atlas** — облачное решение с API.

#### **Подключение к MongoDB**

* **Для драйверов**: установка библиотеки и указание строки подключения.
* **Для Compass**: ввод параметров подключения.
* **Для Atlas**: регистрация в облаке и настройка API-токена.

**Сравнение методов доступа (SQL vs. MongoDB)**

| **Метод** | **SQL** | **MongoDB** |
| --- | --- | --- |
| JDBC/ODBC | Да | BI Connector (аналог) |
| Прямой API-доступ | REST, GraphQL | Драйверы (PyMongo, Node.js, Java) |
| Визуальные инструменты | pgAdmin, MySQL Workbench | Compass, MongoDB Atlas |
| Облачные решения | AWS RDS, Azure SQL | MongoDB Atlas |

### **Заключение**

MongoDB — мощная NoSQL база данных, подходящая для хранения и обработки больших объемов данных. Её гибкость, масштабируемость и высокая производительность делают её популярным выбором для веб-приложений, аналитики и облачных решений. Однако сложность поддержки целостности данных и управление шардингом могут быть сложны для начинающих пользователей. В зависимости от требований проекта выбор между MongoDB и реляционными СУБД зависит от характера данных и нагрузки на систему.