**Практическая работа № 9**

**Введение**

Базы данных существуют с первых дней программирования, а появились они ещё раньше. Это – неотъемлемые части любых приложений. Хорошо спроектированная БД – это один из важнейших компонентов, влияющих на производительность программных проектов. Из-за этого множество архитекторов программных решений исследовали массу подходов к управлению данными, пытаясь выяснить то, какие из этих подходов работоспособны в определённых сценариях, а какие – нет. Выбор подходящей архитектуры БД обычно сводится к выбору между SQL и NoSQL, между реляционными и нереляционными базами данных. А иногда в одном проекте используют и то, и другое.

NoSQL – это базы данных, которые отличаются от реляционных баз данных (РСУБД) в способе хранения информации. Они предоставляют гибкие, масштабируемые и высокопроизводительные базы данных с широким набором функциональных возможностей. NoSQL-базы данных идеально подходят для современных приложений, таких как мобильные и веб-проекты, а также компьютерные игры. NoSQL-базы данных позволяют разработчикам сервисов на них предоставлять пользователям лучший опыт.

Популярность NoSQL-баз данных обусловлена необходимостью работы с данными различной структуры и размеров, включая структурированные, полуструктурированные и полиморфные данные. NoSQL предоставляет большую гибкость разработчику, особенно в условиях быстрых изменений, что особенно важно для стартапов. Эти базы данных также предлагают широкий набор функций и типов данных, специально разработанных для различных моделей данных.

NoSQL-базы данных более легкие и легко-масштабируемые, особенно горизонтально. При необходимости обработки большего объема трафика можно добавить дополнительные реплики, что упрощает масштабирование приложений. SQL-базы данных также могут поддерживать горизонтальное масштабирование, но это может быть более сложным процессом. SQL-хранилища в основном масштабируются вертикально, то есть увеличивают вычислительную мощность сервера, на котором работает база данных.

Программистам проще работать с NoSQL-базами данных, поскольку они легко адаптируются к изменениям в базе данных без необходимости в глубоком погружении. Однако за эти преимущества приходится платить. Например, для нормализации NoSQL-данных может потребоваться написание скрипта, который изменяет структуру записей. Это операция с рисками, особенно если вносится множество изменений при выпуске новых версий системы или исправлении ошибок, что может привести к появлению полиморфных данных и неудачному выполнению скрипта.

Типы NoSQL баз данных:

* Базы данных типа «ключ-значение»
* Документно-ориентированные базы данных – как раз их мы и рассмотрим в практике
* Колоночные базы данных
* Графовые базы данных

**Теория**

1. **MongoDB**

MongoDB — система управления базами данных, которая работает с документоориентированной моделью данных. В отличие от реляционных СУБД, MongoDB не требуются таблицы, схемы или отдельный язык запросов. Информация хранится в виде документов либо коллекций.

Разработчики позиционируют продукт как промежуточное звено между классическими СУБД и NoSQL. MongoDB не использует схемы, как это делают реляционные базы данных, что повышает производительность всей системы.

У MongoDB есть ряд свойств, которые выделяют ее на фоне других продуктов:

У MongoDB есть ряд свойств, которые выделяют ее на фоне других продуктов:

1. Кроссплатформенность. СУБД разработана на языке программирования С++, поэтому с легкостью интегрируется под любую операционную систему (Windows, Linux, MacOS и др.).
2. Формат данных. MongoDB использует собственный формат хранения информации — Binary JavaScript Object Notation (BSON), который построен на основе языка JavaScript.
3. Документ. Если реляционные БД используют строки, то MongoDB — документы, которые хранят значения и ключи.
4. Вместо таблиц MongoDB использует коллекции. Они содержат разные типы наборов данных
5. Репликация. Система хранения информации в СУБД представлена узлами. Существует один главный и множество вторичных. Данные реплицируются между точками. Если один первичный узел выходит из строя, то вторичный становится главным.
6. Индексация. Технология применяется к любому полю в документе на усмотрение пользователя. Проиндексированная информация обрабатывается быстрее.
7. Для сохранения данных большого размера MongoDB использует собственную технологию GridFS, состоящую из двух коллекций. В первой (files) содержатся имена файлов и метаданные по ним. Вторая (chunks) сохраняет сегменты информации, размер которых не превышает 256 Кб.
8. СУБД осуществляет поиск по специальным запросам. Например, пользователь может создать диапазонный запрос и мгновенно получить ответ.
9. Балансировщик нагрузки используется в СУБД не только для распределения нагрузки между разными базами данных, но и для горизонтального масштабирования. Сегменты БД распределяются по разным узлам, что повышает производительность. При этом базы данных, расположенные на разных узлах, синхронизированы между собой и обеспечивают целостность информации для клиента.
10. MongoDB может поставляться для конечного клиента как облачное решение.

СУБД используют для хранения событий в системе (логирование), записи информации с датчиков мониторинга на предприятии, а также в сфере электронной коммерции и мобильных приложений. Часто MongoDB применяют как хранилище в сфере машинного обучения и искусственного интеллекта.

MongoDB относится к классу NoSQL СУБД и работает с документами, а не с записями. Это кроссплатформенный продукт, который легко внедряется в любую операционную систему. Ряд уникальных особенностей позволяет использовать СУБД под определённые задачи, в которых она обеспечивает максимальную производительность и надежность.

1. **GridFS**

GridFS — это спецификация для хранения и извлечения файлов, размер которых превышает предельный размер документа BSON, равный 16 МБ.

Вместо того, чтобы хранить файл в одном документе, GridFS делит файл на части или фрагменты и сохраняет каждый фрагмент в виде отдельного документа. По умолчанию GridFS использует размер фрагмента по умолчанию 255 КБ; то есть GridFS делит файл на фрагменты по 255 КБ, за исключением последнего фрагмента. Последний кусок настолько велик, насколько это необходимо. Точно так же файлы, размер которых не превышает размер фрагмента, имеют только последний фрагмент, используя столько места, сколько необходимо, плюс некоторые дополнительные метаданные.

GridFS использует две коллекции для хранения файлов. В одной коллекции хранятся фрагменты файлов, а в другой — метаданные файлов. В разделе Коллекции GridFS подробно описывается каждая коллекция.

Когда вы запрашиваете GridFS для файла, драйвер повторно собирает фрагменты по мере необходимости. Вы можете выполнять запросы диапазона к файлам, хранящимся в GridFS. Вы также можете получить доступ к информации из произвольных разделов файлов, например, «перескочить» к середине видео- или аудиофайла.

GridFS удобна не только для хранения файлов, размер которых превышает 16 МБ, но и для хранения любых файлов, к которым вам нужен доступ, без необходимости загрузки всего файла в память.

**Задание**

Вам необходимо реализовать небольшой сервис хранения файлов, который будет хранить файлы в GridFS хранилище MongoDB.

У сервиса должно быть несколько основных REST ручек:

|  |
| --- |
| GET /files - получение списка файлов  GET /files/{id} - файла по id  GET /files/{id}/info - получение информации о файле по id  POST /files - загрузка файла  UPDATE /files/{id} - обновление файла по id  DELETE /files/{id} - удаление файла по id |

Информация о файле должна записываться в MongoDB, а сам файл должен попадать в GridFS хранилище.

Получается 3 контейнера:

1. Сервис хранилища
2. MongoDB к нему
3. И сверху Nginx, куда же без него

Обязательно соблюдать [Go Project Layout](https://github.com/golang-standards/project-layout).

Конфигурации микросервисов должны задаваться с помощью ENV файлов (порт, адрес БД и т.д.).

Не забывайте про правильное логирование.

**Полезные ссылки**

1. [Go Tour](https://go.dev/tour/welcome/1)
2. [Go Project Layout](https://github.com/golang-standards/project-layout)
3. [Курс на Stepik по Go](https://stepik.org/course/54403/info)
4. [Популярные логгеры в Go](https://blog.logrocket.com/5-structured-logging-packages-for-go/)
5. [Библиотека для работы с unit-тестами](https://github.com/stretchr/testify)
6. [Стандартная библиотека net/http](https://pkg.go.dev/net/http)
7. [Статья про стандартные методы работы с HTTP запросами](https://www.alexedwards.net/blog/an-introduction-to-handlers-and-servemuxes-in-go)
8. [Статья про HTTP Middleware](https://www.alexedwards.net/blog/making-and-using-middleware)
9. [Статья про работу с MongoDB в Go](https://blog.logrocket.com/integrating-mongodb-go-applications/)
10. [Еще статья про работу с MongoDB в Go](https://www.digitalocean.com/community/tutorials/how-to-use-go-with-mongodb-using-the-mongodb-go-driver)
11. [Официальная статья про работу с GridFS](https://www.mongodb.com/docs/drivers/go/current/fundamentals/gridfs/)
12. [Пример отправки файла в GridFS](https://stackoverflow.com/a/33976261)

**Вопросы**

1. Зачем могут пригодиться NoSQL базы данных?
2. Что такое контекст в Go?
3. Какие еще NoSQL СУБД вы знаете?
4. Когда могут пригодиться мьютексы?
5. Как лучше всего отправлять файлы между микросервисами?