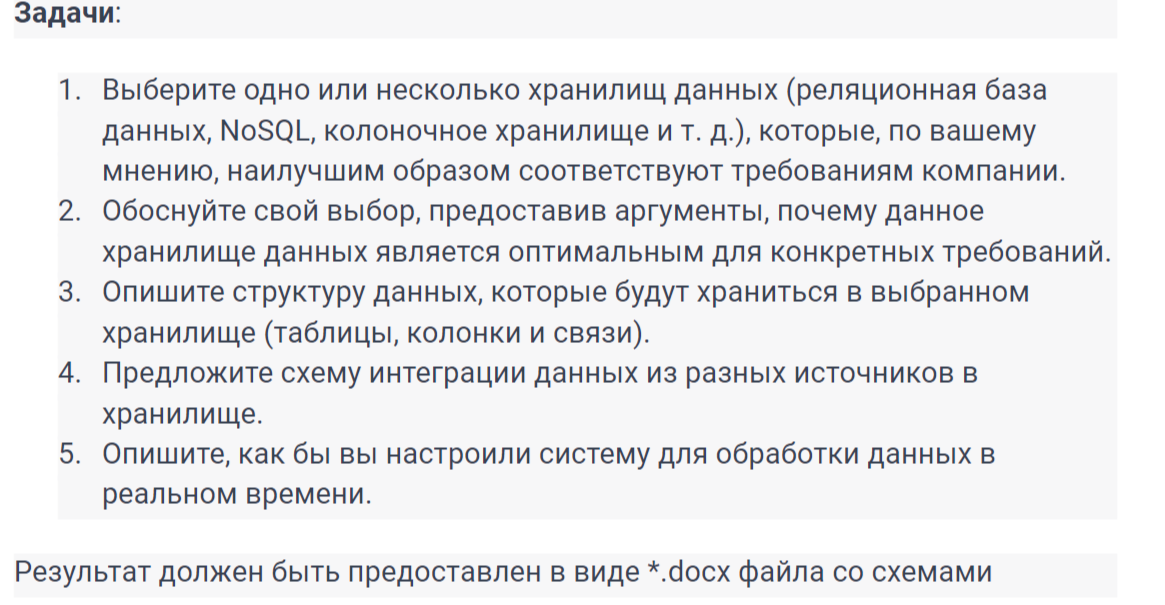
**Лабораторная работа по теме “Выбор хранилища данных”**





1. Для данной задачи выбрано колоночное хранилище данных NoSQL.
2. Обоснование

В колоночных NoSQL базах данных данные хранятся в ячейках, сгруппированных в колонки, а не в строки данных. Колонки логически группируются в колоночные семейства. Колоночные семейства могут состоять из практически неограниченного количества колонок, которые могут создаваться во время работы программы или во время определения схемы. Чтение и запись происходит с использованием колонок, а не строк.

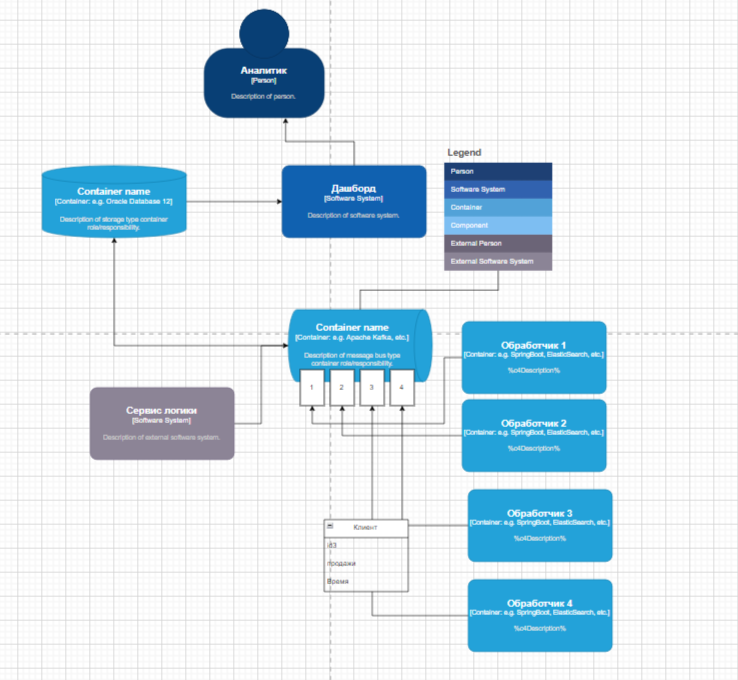
В сравнении с хранением данных в строках, как в большинстве реляционных баз данных, преимущества хранения в колонках заключаются в быстром поиске/доступе и агрегации данных. Реляционные базы данных хранят каждую строку как непрерывную запись на диске. Разные строки хранятся в разных местах на диске, в то время как колоночные базы данных хранят все ячейки, относящиеся к колонке, как непрерывную запись, что делает операции поиска/доступа быстрее.

1. Описание структуры

Модель данных

* Колоночное семейство — структура, которая может легко группировать колонки и суперколонки;
* Ключ — постоянное имя записи. У ключей может быть разное количество колонок, поэтому база данных может расширяться неравномерно;
* Пространство ключей — определяет самый внешний уровень организации, как правило, имя приложения/базы данных.
* Колонка — имеет упорядоченный список элементов — кортежей с именами и значениями.

1. Схема интеграции данных



Данные о продажах обрабатываются обработчиком, по ключу ID сортируются по колонкам, виды продаж, дата продаж, продавец и т.д.

Данные отгружаются в базу данных. Из базы данных формируются запросы пользователей дашборды и проводится аналитика данных.

1. Настройка системы обработки данных в онлайн режиме

Мы получаем аналитику дашборда продавца, анализируя заказы электронной коммерции. В настоящее время заказы хранятся в базе данных MySQL, что затрудняет выполнение аналитических запросов. OLAP-запросы требуют агрегирования и фильтрации большого количества записей для получения метрик, что приводит к снижению производительности MySQL, которая не предназначена для обработки таких запросов.

Поэтому нам нужно извлечь заказы из MySQL и переместить их в Apache Pinot для дальнейшего анализа. Для этого мы используем Airbyte.

для MySQL от Airbyte извлекает заказы из MySQL с помощью захвата измененных данных CDC (Change Data Capture).

Реляционная база данных, такая как MySQL, ведет лог транзакций для записи каждой операции, изменяющей состояние, такой как вставка, обновление и удаление. Механизм CDC обрабатывает этот лог для обнаружения объектов базы данных, которые были изменены, и передает их в виде событий изменения. Таким образом, мы можем получить инкрементные обновления, чтобы избежать дорогостоящего извлечения всей таблицы.

Airbyte запускает такое извлечение через запланированный интервал времени, например, каждый час, день, неделю или около того. Давайте зададим его через каждые пять минут, чтобы получить больше свежих данных. Извлеченные заказы записываются в тему Kafka (orders) как события в формате JSON, а оттуда передаются в Pinot.

**Apache Pinot для быстрого выполнения OLAP-запросов на потоковых данных**

Пайплайн Airbyte ELT, запускаемый каждые пять минут, генерирует огромное количество необработанных данных, которые необходимо проанализировать. Кроме того, проанализированные данные будут доступны для всех продавцов платформы, что потребует от нас высокой пропускной способности запросов и задержки в миллисекундах. Следовательно, движок аналитики должен быть способен выполнять аналитические запросы и возвращать результаты в режиме реального времени, чтобы обеспечить хороший пользовательский опыт. Поэтому мы будем использовать Apache Pinot в качестве движка аналитики для удовлетворения вышеперечисленных потребностей.

