МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

СӘТБАЕВ УНИВЕРСИТЕТІ

Институт автоматики и информационных технологий
Кафедра «Программной инженерии»

Родионов Николай Петрович

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

На соискание академической степени магистра

Название диссертации	Исследование возможностей NoSQL колоночной базы данных для оперативной-аналитической обработки данных
Направление подготовки	7M06101– SoftwareEngineering

Научный руководитель
PhD, ассоциированный профессор
_____Мукажанов Н.К.
«__» _____ 2023 г.

Оппонент
PhD, ассоциированный профессор
_____ Чинибаева Т.Т.
«__» _____ 2023 г.

Нормоконтроль
_____Ахмедиярова А.Т.
«__» _____ 2023 г.

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой ПИ
Кандидат физико-математических наук, профессор
_____ Молдагулова А. Н.
«__» _____ 2023 г.

Алматы 2023

Институт автоматики и информационных технологий

Кафедра «Программной инженерии»

Специальность: 7М06101– Software Engineering

УТВЕРЖДАЮ
Заведующий кафедрой ПИ
Кандидат физико-математических наук, профессор
_____ Молдагулова А. Н.
«__» _____ 2023 г.

ЗАДАНИЕ
на выполнение магистерской диссертации

Магистранту:РодионовуНиколаюПетровичу
Тема: «Исследование возможностей NoSQL колоночной базы данных для оперативной-аналитической обработки данных»
Срок сдачи законченной диссертации «__» _____

Исходные данные к магистерской диссертации:дан анализ структуры хранения данных, многомерной модели данных и оперативно аналитическая обработки данных. Поставлены цели и задачи диссертационной работы по исследованию применения колоночной базы данных для создания оперативно аналитической системы.

Перечень подлежащих разработке в магистерской диссертации вопросов или краткое содержание магистерской диссертации:а) обзор по применению NoSQL для аналитических систем; б) подходы создания аналитической системы на основе колоночных баз данных; в) разработкаразвертывания OLAP-куба.

Рекомендуемаяосновнаялитература:

1) Kimball R., Ross M. "The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling" (2013)
2) Pedersen T.B., et al. "Multidimensional Databases and Data Warehousing" (1999).
3) Sadalage P. J., Fowler M. "NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence" (2012)

ГРАФИК

подготовки магистерской диссертации

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Раздел 1. Обзор по применению NoSQL для аналитических систем	30.10.2022	Выполнено
Раздел 2. Подходы создания аналитической системы на основе колоночных баз данных	03.03.2023	Выполнено
Раздел 3. Практическая реализация	24.04.2023	Выполнено

Консультации по проекту с указанием относящихся к ним разделов проекта

Раздел	Консультант, (уч. степень, звание)	Сроки	Подпись
Нормоконтроль	Ахмедиярова Айнур Танатаровна, PhD, ассоциированный профессор		
Программное обеспечение	Мукажанов Нуржан Какенович, PhD, ассоциированный профессор		
Антиплагиат	Аубакиров Бакдаулет, Ассистент		

Дата выдачи задания "____"_____ 2023 г.

Заведующий кафедрой _____Молдагулова А. Н.

Научный руководитель _____ МукажановН. К.

Задание принял к исполнению магистрант _____ Родионов Н. П.

Дата "_____"_____ 2023 г.

АННОТАЦИЯ

В настоящее время объемы аналитических данных стали достигать критических размеров, бросая вызов традиционным подходам к хранению данных, текущие решения основаны на реляционных базах данных, которые больше не адаптированы к таким объемам данных. Решения NoSQL позволяют нам рассматривать новые подходы эффективной обработки аналитических данных, особенно с точки зрения пользования многомерной моделью. Это и есть предмет данной статьи.

Исследовательская часть данной работы посвящена изучению вариантов применения NoSQL для аналитических систем. В качестве СУБД был выбраны ClickHouse. Исследование представляет собой краткий сравнительный анализ преимуществ программы.

В практической части будет обзор применения подходов создания многомерной модели данных из нереляционных БД. В частности, пример создания OLAP в ClickHouse и сравнение с СУБД PostgreSQL.

В заключительной части разобраны архитектура и компоненты рабочего контура OLAP системы на примере СУБД ClickHouse.

АҢДАТПА

Қазіргі уақытта кез келген ақпараттық жүйеге қызмет көрсетудің құрамдас бөлігі оның жұмысын бақылау және талдау болып табылады. Бұл процестің маңызды бөлігі алынған мәліметтерді автоматтандыру және визуализациялау болып табылады.

Бұл жұмыстың зерттеу бөлімі ақпараттық жүйенің жұмыс жүктемесін бақылау және талдау үшін жиналған метрикалық деректерді визуализациялау нұсқаларын зерттеуге арналған. Визуализация бағдарламалары ретінде Zabbix, PRTG және Grafana таңдалды. Зерттеу таңдалған бағдарламалардың қысқаша салыстырмалы талдауы болып табылады.

Практикалық бөлігі - бұл визуализация үшін метрика түрінде өнімділікті бақылау деректерін қамтамасыз ететін бағдарламалық жасақтаманың өзегін тікелей әзірлеу. Негізгі жаңалық – бір график ақпараттық жүйенің жұмысына пайдаланушының қанағаттануы туралы да, ақпараттық жүйенің жұмыс көлемі туралы да мәліметтер береді.

Қорытынды бөлімде әзірленген құралдың нәтижесінде алынған деректерді бақылау және талдау арқасында өнімді тізбектегі өнімділікті жақсарту үшін бірнеше «жағдайлар» талданады

SUMMARY

At present, the volumes of analytical data have become critical, challenging traditional approaches to data storage, current solutions are based on relational databases, which are no longer adapted to such volumes of data. NoSQL solutions allow us to consider new approaches to efficient processing of analytical data, especially from the point of view of using a multidimensional model. This is the subject of this article.

The research part of this work is devoted to the study of NoSQL application options for analytical systems. ClickHouse was chosen as the DBMS. The study is a brief comparative analysis of the benefits of the program.

In the practical part, there will be an overview of the application of approaches for creating a multidimensional data model from non-relational databases. In particular, an example of creating OLAP in ClickHouse and a comparison with the PostgreSQL DBMS.

In the final part, the architecture and components of the OLAP system workflow are analyzed using the ClickHouse DBMS as an example.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современной организации хранение и обработка данных являются неотъемлемой частью ее деятельности. Для повышения производительности информационных систем важна скорость обработки данных. Решение этой проблемы предлагает СУБД NoSQL, способная выполнять аналитические запросы в реальном времени на больших структурированных данных. Эта СУБД специально разработана для аналитики и предоставляет возможность эффективно работать с данными.

Решение проблемы производительности осуществляется с помощью СУБД NoSQL типа, которая способна осуществлять аналитические запросы в режиме реального времени на структурированных больших данных. Эти базы данных разработаны специально для аналитики и обладают высокой скоростью обработки данных. Они предоставляют возможность эффективно работать с большими объемами данных и осуществлять операции агрегации и анализа. Таким образом, использование NoSQL, на примере СУБД ClickHouse [1], позволяет организациям значительно повысить производительность своих информационных систем и обеспечить оперативную аналитику в реальном времени.

Целью данной магистерской диссертации является исследование применение колоночной базы данных для создания оперативно аналитической системы на примере СУБД ClickHouse.

В ходе реализации данной цели были поставлены и реализованы следующие задачи:

- обзор аналитических систем
- обзор по возможностям баз данных для хранения аналитических данных
- изучение СУБД ClickHouse для хранения аналитической системы
- повысить скорость выполнения обработки данных в режиме реального времени в СУБД ClickHouse.

Для практической наглядности объектом исследования будет реальное предприятие – АО «KaspiBank», а предметом исследования – биллинг система Контакт Центра.

[1]https://clickhouse.com/

Научный аппарат диссертации

Тема данной магистерской диссертации: «Исследование возможностей NoSQL колоночной базы данных для оперативной-аналитической обработки данных».

Объектом исследования данной магистерской диссертации является работа биллинг системы Контакт Центра на базе СУБД ClickHouse АО «KaspiBank»

Предметом исследования данной магистерской диссертации является производительность работы биллинг системы Контакт Центра на базе СУБД ClickHouse АО «KaspiBank»

Цель -исследование применения колоночной базы данных для создания оперативно аналитической системы на примере СУБД ClickHouse.

Задачи, которые необходимо решить в рамках данной магистерской диссертации:

- обзор аналитических систем
- обзор по возможностям баз данных для хранения аналитических данных
- изучение СУБД ClickHouse для хранения аналитической системы
- повысить скорость выполнения обработки данных в режиме реального времени в СУБД ClickHouse.

Гипотеза:в любой информационной системе имеется узкое место (одно или несколько), которое является основным источником сильного снижения производительности. СУБД ClickHouse на постоянной основе позволят устранить узкие места, такие, как скорость выполнения запросов, размер данных, занимающих дисковое пространство, тем самым снизив нагрузку на систему. В результате чего, появится возможность получения запрашиваемых данных для дальнейшей аналитики в короткой срок и в реальном времени

Новизна- исследование возможностей NoSQL колоночной базы данных для оперативной-аналитической обработки данных показало, что СУБД ClickHouse становится все популярнее и известнее. Ее основное преимущество заключается в том, что позволяет выполнять аналитические SQL-запросы в режиме реального времени на структурированных больших данных, а так же ClickHouse использует собственный диалект SQL близкий к стандартному, но содержащий различные расширения: массивы и вложенные структуры данных, специализированные агрегатные функции, функции для работы с URL и.т.д

Методы:структура хранения данных, многомерный анализ данных, оперативно аналитическая обработка данных.

Теоретическая значимость.Исследование возможностей NoSQL колоночной базы данных для оперативной-аналитической обработки данных на примере СУБД ClickHouse позволит улучшить понимание методов и способов аналитической обработки данных в режиме реального времени на структурированных больших данных со скоростью около миллиона записей в секунду на относительно скромных мощностях.

Практическая значимость.Исследование возможностей NoSQL колоночной базы данных для оперативной-аналитической обработки данных на примере СУБД ClickHouse позволит улучшить производительность работы биллинг системы Контакт Центра на базе СУБД ClickHouse, а значит повысит скорость работы биллинг системы Контакт Центра на базе СУБД ClickHouse АО «KaspiBank»

1. Обзор по применению NoSQL для аналитических систем.

1.1 Аналитическая система

Аналитическая система — это программный комплекс, используемый для анализа больших объемов данных с целью получения важных выводов и инсайтов. Она может включать в себя инструменты для визуализации, отчетности и прогнозирования, а также использовать машинное обучение и другие технологии анализа данных. Существует множество различных типов аналитических систем, включая [1,2]:

1. Системы оперативного анализа (OLAP)
2. Системы бизнес-интеллекта (BI)
3. Системы анализа данных (Data Analytics)
4. Системы анализа временных рядов (Time-series Analytics)
5. Системы анализа гео-данных (Geospatial Analytics)
6. Системы машинного обучения (Machine Learning)

1.1.1Системы оперативного анализа

Системы оперативного анализа (OLAP) — это специальные программные инструменты, которые предназначены для анализа больших объемов транзакционных данных. OLAP-системы используются в бизнесе и в других областях, где необходимо быстро и удобно анализировать данные, собранные из различных источников.

OLAP-системы работают с многомерными данными, что позволяет пользователю просматривать информацию в различных измерениях и точках зрения. Например, OLAP-система может показывать продажи по продукту, по региону, по каналу продаж и по времени, что позволяет бизнесу лучше понимать, какие продукты продаются лучше и в каких регионах. OLAP-системы позволяют анализировать данные в реальном времени, а также проводить прогнозирование и моделирование будущих событий. OLAP-системы часто используются вместе с системами бизнес-интеллекта, которые помогают организациям собирать, обрабатывать и анализировать данные из различных источников. Преимущества OLAP-систем включают в себя:

- Быстрый доступ к данным и возможность проводить анализ в реальном времени;
- Возможность анализировать данные в различных измерениях;
- Возможность проводить прогнозирование и моделирование;
- Возможность быстро отвечать на запросы пользователя;
- Удобный и интуитивно понятный интерфейс для работы с данными.

OLAP-системы могут быть реализованы как в виде программных продуктов, которые устанавливаются на компьютере пользователя, так и в виде веб-приложений, которые работают через браузер. OLAP-системы могут быть

настроены для работы с различными источниками данных, такими как реляционные базы данных, многомерные кубы данных, файлы Excel и другие[3,4].

1.1.2 Системы бизнес-интеллекта

Системы бизнес-интеллекта (BI) — это программные инструменты, которые позволяют организациям собирать, хранить, обрабатывать и анализировать большие объемы данных из различных источников. BI-системы позволяют пользователям принимать более обоснованные и точные решения, основанные на фактах и анализе данных.BI-системы обычно состоят из нескольких компонентов, таких как:

- ETL (extract, transform, load) - компонент, который позволяет собирать и переносить данные из различных источников в централизованное хранилище данных;
- Хранилище данных (data warehouse) - централизованное хранилище данных, которое позволяет пользователям быстро и удобно получать доступ к данным из различных источников;
- Отчетность и аналитика - компонент, который позволяет пользователям создавать отчеты и анализировать данные;
- Интерфейс для пользователя - пользовательский интерфейс, который позволяет пользователям получать доступ к данным и создавать отчеты и аналитику.

BI-системы могут использоваться для различных целей, таких как анализ продаж, управление финансами, анализ рынка, мониторинг производственных процессов и других. BI-системы могут использоваться как в крупных корпорациях, так и в небольших компаниях. Преимущества BI-систем включают в себя:

- Возможность собирать данные из различных источников в едином хранилище данных;
- Быстрый доступ к данным и возможность проводить анализ в реальном времени;
- Возможность анализировать данные в различных измерениях;
- Возможность создавать отчеты и аналитику на основе данных;
- Удобный и интуитивно понятный интерфейс для работы с данными.

BI-системы могут быть реализованы как в виде программных продуктов, которые устанавливаются на компьютере пользователя, так и в виде веб-приложений, которые работают через браузер. BI-системы могут быть настроены для работы с различными источниками данных, такими как реляционные базы данных, многомерные кубы данных, файлы Excel и другие [5,6].

1.1.3 Системы анализа данных

Системы анализа данных (Data Analytics) — это программные инструменты, которые используются для анализа больших объемов данных и извлечения полезной информации для принятия бизнес-решений, которые включают в себя различные методы и техники, такие как статистический анализ, машинное обучение, обработку естественного языка и другие. Системы анализа данных могут использоваться для различных целей, таких как анализ рынка, управление продажами, прогнозирование спроса, анализ поведения потребителей, оптимизация производственных процессов и другие. Системы анализа данных обычно состоят из нескольких компонентов, таких как:

- Сбор данных - компонент, который позволяет собирать данные из различных источников, таких как базы данных, веб-страницы, социальные сети, датчики и другие;
- Хранение данных - компонент, который позволяет хранить и управлять большими объемами данных;
- Анализ данных - компонент, который позволяет анализировать данные и извлекать полезную информацию;
- Визуализация данных - компонент, который позволяет представлять данные в удобной и понятной форме.

Преимущества систем анализа данных включают в себя:

- Возможность анализировать большие объемы данных;
- Возможность использовать различные методы и техники для анализа данных;
- Возможность принимать более обоснованные и точные бизнес-решения;
- Возможность увеличить эффективность и производительность бизнес-процессов.

Системы анализа данных могут быть реализованы как в виде программных продуктов, которые устанавливаются на компьютере пользователя, так и в виде веб-приложений, которые работают через браузер. Системы анализа данных могут использоваться как в крупных корпорациях, так и в небольших компаниях. Различные отрасли, такие как финансы, медицина, телекоммуникации, транспорт и другие, могут использовать системы анализа данных для решения различных задач и проблем [7,8].

1.1.4 Системы анализа временных рядов

Системы анализа временных рядов (Time-series Analytics) — это инструменты, которые используются для анализа данных, упорядоченных во времени. Такие данные могут быть собраны из различных источников, таких как датчики, метеостанции, торговые площадки и другие. Системы анализа временных рядов помогают описать и понять изменения в данных, а также

прогнозировать будущие значения. Системы анализа временных рядов используют различные методы и техники, такие как:

- Анализ статистики временных рядов - это метод, который позволяет определить основные характеристики временных рядов, такие как среднее значение, стандартное отклонение, корреляция и другие;
- Моделирование временных рядов - это метод, который позволяет создать математическую модель для временного ряда и прогнозировать его будущие значения;
- Машинное обучение - это метод, который позволяет автоматически определить связи и закономерности в данных временных рядов и создавать модели на основе этих связей.

Преимущества систем анализа временных рядов включают в себя:

- Возможность анализировать данные, упорядоченные во времени, и определять тренды и циклы в данных;
- Возможность использовать прогнозирование для предсказания будущих значений и принятия соответствующих решений;
- Возможность улучшения эффективности и производительности бизнес-процессов.

Системы анализа временных рядов могут быть использованы в различных отраслях, таких как финансы, производство, энергетика и другие, для решения различных задач, таких как прогнозирование спроса, оптимизация производственных процессов, анализ рынка и другие. Системы анализа временных рядов могут быть реализованы как в виде программных продуктов, которые устанавливаются на компьютере пользователя, так и в виде веб-приложений, которые работают через браузер [9,10].

1.1.5 Системы анализа гео-данных

Системы анализа гео-данных (Geospatial Analytics) — это инструменты анализа данных, которые используются для обработки, анализа и визуализации гео-данных, таких как карты, изображения, аэрофотоснимки, спутниковые снимки и другие географические данные. Системы анализа гео-данных могут использоваться в различных областях, таких как география, геология, экология, транспорт, землеустройство, градостроительство, а также в бизнесе для принятия решений, связанных с местоположением и локализацией. Системы анализа гео-данных включают в себя множество методов и техник, таких как пространственный анализ, картография, геокодирование, геоматика, моделирование и прогнозирование. Они позволяют производить анализ данных на основе географических координат, а также интегрировать данные из различных источников для получения более полной картины. Преимущества систем анализа гео-данных включают в себя:

- Возможность анализировать и визуализировать гео-данные в реальном времени;

- Возможность использования различных методов и техник для анализа гео-данных;
- Возможность принимать более обоснованные и точные бизнес-решения, связанные с местоположением и локализацией;
- Возможность увеличить эффективность и производительность бизнес-процессов, связанных с гео-данными.

Системы анализа гео-данных могут быть реализованы как в виде программных продуктов, которые устанавливаются на компьютере пользователя, так и в виде веб-приложений, которые работают через браузер. Системы анализа гео-данных могут использоваться как в крупных корпорациях, так и в небольших компаниях. Они могут быть использованы для решения различных задач, таких как анализ рисков, мониторинг объектов, анализ рынков, прогнозирование спроса, планирование маршрутов и другие [11,12].

1.1.6 Системы машинного обучения

Системы машинного обучения (Machine Learning) — это подход к анализу данных, который позволяет компьютерам обучаться на основе данных, вместо явного программирования. Системы машинного обучения используют алгоритмы, которые позволяют компьютерам извлекать знания из данных и использовать их для решения задач. Системы машинного обучения используются во многих областях, таких как медицина, финансы, наука о материалах, маркетинг, транспорт, реклама и многих других. Они используются для решения различных задач, таких как классификация, регрессия, кластеризация, обнаружение аномалий и прогнозирование. Системы машинного обучения могут быть реализованы как в виде программных продуктов, которые устанавливаются на компьютере пользователя, так и в виде веб-приложений, которые работают через браузер. Они могут использоваться как в крупных корпорациях, так и в небольших компаниях. Преимущества систем машинного обучения включают в себя:

- Возможность решать задачи, которые ранее были трудно или невозможно решить с помощью традиционных методов анализа данных;
- Возможность использования больших объемов данных;
- Возможность автоматической адаптации к изменяющимся условиям и данным;
- Возможность получения точных и надежных прогнозов;
- Возможность повышения эффективности и производительности бизнес-процессов.

Одним из основных элементов систем машинного обучения является выборка данных. Данные должны быть правильно отобраны и обработаны, чтобы обучение модели было эффективным. После обучения модель может быть использована для анализа новых данных и для прогнозирования результатов. Однако, как и все другие методы анализа данных, системы

машинного обучения не являются универсальным решением для всех задач, и их применение должно быть оценено в контексте конкретной задачи и с учетом ее особенностей [13,14]

1.2 NoSQL

NoSQL (нереляционные системы управления базами данных) — это категория баз данных, которые отличаются от традиционных реляционных баз данных, таких как MySQL или PostgreSQL. NoSQL-базы данных не используют таблицы и связи между ними, что позволяет им гораздо легче масштабироваться и обрабатывать большие объемы данных в режиме реального времени. Существует несколько типов NoSQL баз данных, включая [15]:

1. Документоориентированные базы данных
2. Ключевые-значение базы данных
3. Графовые базы данных
4. Колоночныебазыданных
5. Коллективныебазыданных
6. Коллективные базы данных с продуктивными картами

1.2.1 Документоориентированные базы данных

Документоориентированные базы данных (Document-oriented databases) — это базы данных, которые хранят данные в формате документов, обычно в формате JSON или BSON. Каждый документ представляет собой набор полей, содержащих данные, связанные с конкретным объектом или сущностью. Основным преимуществом документоориентированных баз данных является их гибкость. Поскольку данные хранятся в формате документов, то структура этих документов может быть очень гибкой и изменяться в зависимости от требований приложения. Это позволяет быстро и легко адаптироваться к новым требованиям и изменениям в данных. Кроме того, документоориентированные базы данных могут быть очень масштабируемыми и предоставлять высокую производительность при работе с большими объемами данных. Они могут использоваться для хранения и обработки различных типов данных, таких как текст, изображения, видео, аудио и т.д. Документоориентированные базы данных также обладают рядом других преимуществ, таких как:

- Легкость в использовании и разработке, так как они используют привычный формат документов;
- Поддержка гибкого индексирования, позволяющего быстро и эффективно выполнять поиск данных;
- Возможность использования множества языков программирования и инструментов для работы с данными.

Однако документоориентированные базы данных имеют и некоторые ограничения, например, они могут быть менее эффективными для выполнения

сложных запросов, которые включают объединение данных из разных коллекций или документов. Примерами документоориентированных баз данных являются MongoDB, Couchbase, Amazon DocumentDB, Cassandra, и многие другие. Они широко используются в различных сферах, включая финансы, медицину, маркетинг, образование, науку о материалах и многие другие[16,17].

1.2.2 Ключевые-значение базы данных

Ключевые-значение базы данных представляют собой простую структуру данных, где каждому ключу соответствует определенное значение. Эти базы данных применяются в системах, где требуется быстрый доступ к данным и высокая производительность. Они подходят для хранения малых и средних объемов данных. Redis и Riak — это примеры ключ-значение баз данных.

Redis - это открытая ключ-значение база данных с открытым исходным кодом, которая широко используется для кэширования, сообщений и сессий, а также для работы с графиками, множествами и списками. Redis хранит данные в памяти, что делает его одним из самых быстрых инструментов для хранения данных. Одной из ключевых особенностей Redis является возможность кэширования данных, что позволяет уменьшить нагрузку на базу данных и ускорить обработку запросов. Redis также поддерживает различные структуры данных, такие как строки, списки, множества, хэши и сортированные множества.

Riak - это распределенная база данных ключ-значение, которая предназначена для хранения больших объемов данных и обеспечения высокой доступности. Riak использует горизонтальное масштабирование, что позволяет добавлять новые узлы в систему для обработки больших объемов данных. Он также предоставляет механизмы для репликации и обеспечения отказоустойчивости данных. Одной из особенностей Riak является возможность работы в режиме согласованного разделения, что позволяет обеспечить высокую доступность данных и уменьшить время восстановления после сбоя. Riak также поддерживает множество клиентских библиотек, что делает его удобным инструментом для работы с различными языками программирования.

Ключ-значение базы данных Redis и Riak нашли применение в различных областях, таких как интернет-магазины, мессенджеры, социальные сети и другие [18,19].

1.2.3 Графовые базы данных

Графовые базы данных представляют собой тип баз данных, в которых данные хранятся в виде графов, используются для хранения связей между данными, таких как социальных сетей, географических карт и многое другое. Они также могут быть использованы для моделирования и анализа сложных

систем, таких как транспортные сети и телекоммуникационные системы. Neo4j и OrientDB — это примеры графовых баз данных.

Neo4j - это открытая графовая база данных с открытым исходным кодом, которая обеспечивает высокую производительность и масштабируемость при работе с данными. Он использует язык запросов Cypher для выполнения запросов к данным, который позволяет легко извлекать и анализировать связи между данными.Neo4j также поддерживает механизмы для работы с транзакциями и обеспечения целостности данных, что делает его подходящим инструментом для работы с критически важными приложениями. Он также предоставляет широкий спектр дополнительных инструментов и библиотек для работы с данными, таких как библиотека для анализа данных и библиотека для визуализации данных.

OrientDB - это графовая база данных с открытым исходным кодом, которая поддерживает графы, документы и объектно-ориентированные базы данных. Он обеспечивает высокую производительность при работе с данными, используя различные механизмы кэширования данных и оптимизации запросов. Одной из ключевых особенностей OrientDB является поддержка транзакционной модели, что позволяет обеспечить целостность и согласованность данных при работе с графами. Он также поддерживает различные языки запросов, включая SQL, графовый язык запросов и язык запросов для документов.

Графовые базы данных Neo4j и OrientDB нашли применение в различных областях, таких как социальные сети, рекомендательные системы, биоинформатика, геология и другие [20,21].

1.2.4 Колоночные базы данных

Колоночные базы данных — это тип NoSQL баз данных, в которых данные хранятся в виде колонок, в отличие от реляционных баз данных, где данные хранятся в виде строк. Колоночные базы данных предоставляют высокую производительность при чтении больших объемов данных и подходят для хранения данных, которые не меняются часто или вовсе не изменяются. ApacheCassandra и ApacheHBase - это примеры колоночных баз данных.

ApacheCassandra - это распределенная колоночная база данных с открытым исходным кодом, разработанная для обработки больших объемов данных в режиме реального времени. Она обеспечивает высокую доступность и масштабируемость благодаря архитектуре, основанной на модели "shared-nothing". Cassandra использует распределенный механизм хранения данных, который разбивает данные на различные узлы и реплицирует данные на другие узлы в кластере для обеспечения отказоустойчивости. Cassandra также поддерживает механизмы для работы с транзакциями и обеспечения целостности данных.

ApacheHBase - это распределенная колоночная база данных, построенная поверх ApacheHadoop. HBase обеспечивает высокую доступность и

масштабируемость благодаря архитектуре, основанной на модели "shared-nothing". HBase использует распределенный механизм хранения данных, который разбивает данные на различные узлы и реплицирует данные на другие узлы в кластере для обеспечения отказоустойчивости. Он также поддерживает механизмы для работы с транзакциями и обеспечения целостности данных.

Колоночные базы данных ApacheCassandra и ApacheHBase широко используются для хранения и обработки больших объемов данных в различных областях, таких как телекоммуникации, финансы, интернет-магазины и многое другое. Они предоставляют высокую производительность и масштабируемость, что делает их подходящими инструментами для работы с крупными наборами данных [22,23].

1.2.5 Коллективные базы данных

Коллективные базы данных — это тип NoSQL баз данных, который обеспечивает горизонтальное масштабирование и распределение данных. Они используют распределенный механизм хранения данных, который позволяет разбивать данные на различные узлы и реплицировать данные на другие узлы в кластере для обеспечения отказоустойчивости. Коллективные базы данных предоставляют возможность хранения и обработки больших объемов данных с высокой производительностью и отказоустойчивостью. Они используются для приложений, где необходимо масштабировать приложение и обрабатывать большие объемы данных в режиме реального времени. Пример коллективной базы данных:

Amazon DynamoDB — это управляемая коллективная база данных, которую предоставляет Amazon Web Services (AWS). Он предназначен для обработки любых приложений, которые требуют миллисекундных задержек и высокой доступности. DynamoDB использует модель данных "ключ-значение" и обеспечивает автоматическое масштабирование приложений, что позволяет уменьшить время настройки и управления кластером баз данных.

Коллективные базы данных предоставляют ряд преимуществ, таких как горизонтальное масштабирование, высокую доступность и отказоустойчивость. Они также позволяют легко масштабировать приложения при росте объемов данных, что делает их популярным выбором [24,25]

1.2.6 Коллективные базы данных с продуктивными картами

Коллективные базы данных с продуктивными картами, которые предназначены для обработки больших объемов данных, распределенных на несколько узлов. Они используются для анализа больших объемов данных, создания и обучения моделей машинного обучения, а также для выполнения вычислений в реальном времени. Примерами таких баз являются, Apache Hadoop и Apache Spark.

Apache Hadoop - это фреймворк, который позволяет распределенно хранить и обрабатывать большие объемы данных. Hadoop состоит из нескольких компонентов, включая HDFS (Hadoop Distributed File System) для хранения данных, MapReduce для обработки данных и YARN (Yet Another Resource Negotiator) для управления ресурсами кластера. Hadoop также поддерживает множество инструментов для работы с данными, включая Apache Hive, Apache Pig и Apache Spark.

Apache Spark - это более новый фреймворк для обработки данных, который работает быстрее, чем Hadoop. Он также предоставляет распределенное хранилище данных и возможности параллельной обработки, но включает в себя более широкий набор инструментов для работы с данными, таких как машинное обучение, графовые вычисления и обработка потоков данных в реальном времени.

Основное преимущество Hadoop и Spark - это способность масштабироваться горизонтально, т.е. добавлять новые узлы в кластер, чтобы увеличить его мощность и производительность. Это позволяет легко обрабатывать большие объемы данных и обеспечивать высокую отказоустойчивость. Hadoop и Spark также являются открытыми и бесплатными для использования, что делает их популярными среди разработчиков и компаний, работающих с большими объемами данных.

Каждый тип NoSQL-базы данных предназначен для решения разных задач и имеет свои особенности и преимущества. Выбор типа NoSQL-базы данных зависит от функциональных требований проекта и особенностей данных, которые нужно хранить и обрабатывать [26,27].

1.3 NoSQL базы данных могут использоваться для аналитической обработки данных

NoSQL базы данных могут использоваться для аналитической обработки данных, поскольку они могут эффективно работать с большими объемами данных и высокой плотностью данных. Некоторые NoSQL-базы данных, такие как ApacheCassandra, ApacheHbase, ClickHouse, имеют колоночную архитектуру, которая оптимизирована для аналитических операций. Они могут также использоваться совместно с инструментами аналитической обработки, такими как Apache Spark, для дополнительной обработки и анализа данных [28,29].

1.4 OLAP (Online Analytical Processing)

OLAP (Online Analytical Processing) — это метод анализа данных, который позволяет выполнять быстрые и эффективные анализы больших многомерных наборов данных. Он использует многомерные модели данных, которые хранятся в специальных структурах данных, известных как кубы. Это позволяет выполнять анализы данных и извлекать информацию в режиме

реального времени. OLAP может использоваться в различных отраслях, таких как бизнес, финансы, маркетинг, здравоохранение и другие. Он помогает делать более обоснованные и эффективные решения, основанные на данных. OLAP (On-Line Analytical Processing) включает в себя ряд методов анализа данных, которые позволяют быстро и эффективно искать информацию в больших многомерных данных. Некоторые из самых распространенных методов OLAP:

1. Развертывание (Drill-down) – метод, позволяющий исследовать данные по уровням детализации.
2. Развернуть (Roll-up) – метод, позволяющий сокращать многомерные данные до высшего уровня агрегации.
3. Свертывание (Slice-and-dice) – метод, позволяющий разрезать многомерные данные по одной из измерений.
4. Ротация (Pivot) – метод, позволяющий изменять положение измерений в многомерном представлении данных.
5. Дополнительные аналитические операции – включают в себя такие операции, как расчет средних значений, агрегация, ранжирование.

OLAP (Online Analytical Processing) обычно реализуется с использованием реляционных СУБД. Однако, в некоторых случаях, NoSQL базы данных могут быть использованы для аналитической обработки данных, если необходимы высокая масштабируемость и производительность. Использование NoSQL вместо реляционных СУБД для OLAP может быть сложным и требует специальных знаний, так как аналитические операции в NoSQL могут выполняться по-другому, чем в реляционных СУБД. NoSQL может решать проблему скалярности и оперативности для аналитической обработки данных, когда объем данных становится слишком велик для обработки с помощью реляционных баз данных. Также NoSQL базы данных могут улучшить гибкость хранения данных, обеспечить поддержку высоконагруженных приложений и повысить отказоустойчивость системы в сравнении с реляционными базами данных. Это достигается благодаря горизонтальной масштабируемости, гибкой схеме данных, асинхронной репликации и распределенной архитектуре NoSQL баз данных. [30,31,32,33,34].

Задачи, которые необходимо решить в рамках данной магистерской диссертации:

1. провести обзор аналитических систем
2. исследовать возможности баз данных для хранения аналитических данных
3. изучить СУБД ClickHouse для хранения аналитической системы
4. повысить скорость выполнения обработки данных в режиме реального времени в СУБД ClickHouse.

2. Подходы создания аналитической системы на основе колоночных баз данных.

Создание многомерной аналитической системы на основе колоночной базы данных имеет целью обеспечить эффективное хранение и анализ больших объемов данных. В такой системе используется колоночная модель для организации данных, в отличие от традиционной строковой модели.
В колоночной базе данных данные хранятся в виде столбцов, где каждый столбец содержит значения определенного типа данных для всех записей. Это позволяет значительно улучшить производительность операций анализа и запросов к данным, особенно при работе с большими объемами информации. Многомерная аналитическая система строится на основе колоночной базы данных путем добавления дополнительных функций и инструментов, специально предназначенных для анализа данных. Например, система может включать в себя OLAP (Online Analytical Processing) движок для многомерного анализа данных, инструменты для построения отчетов и дашбордов, а также возможности для проведения сложных аналитических запросов.

2.1 Многомерная модель

Многомерная модель описывает систему данных, в которой данные хранятся в нескольких мерных контекстах, таких как временной, географический, продуктовый или клиентский. Это означает, что данные представлены в виде матрицы, где каждая колонка является атрибутом, а каждая строка является наблюдением. Многомерная модель обычно используется в бизнес-аналитике и обработке больших объемов данных, так как она позволяет легко анализировать данные по разным аспектам и выявлять закономерности и тренды. Многомерные модели также позволяют выполнять различные виды аналитики, такие как кумулятивный анализ, анализ по регионам и сегментам, анализ частоты и т.д. [16]

В основе технологии OLAP, представляющей информацию в виде куба, лежит идея многомерной модели данных. Многомерная модель данных состоит из следующих основных понятий: гиперкуб, измерения, элементы, ячейки, мера или значения, рисунок 1. С математической точки зрения количество членов обычного гиперкуба должно быть равным, но OLAP-куб не имеет таких ограничений. Несмотря на отмеченные отличия, термин гиперкуб получил широкое распространение.

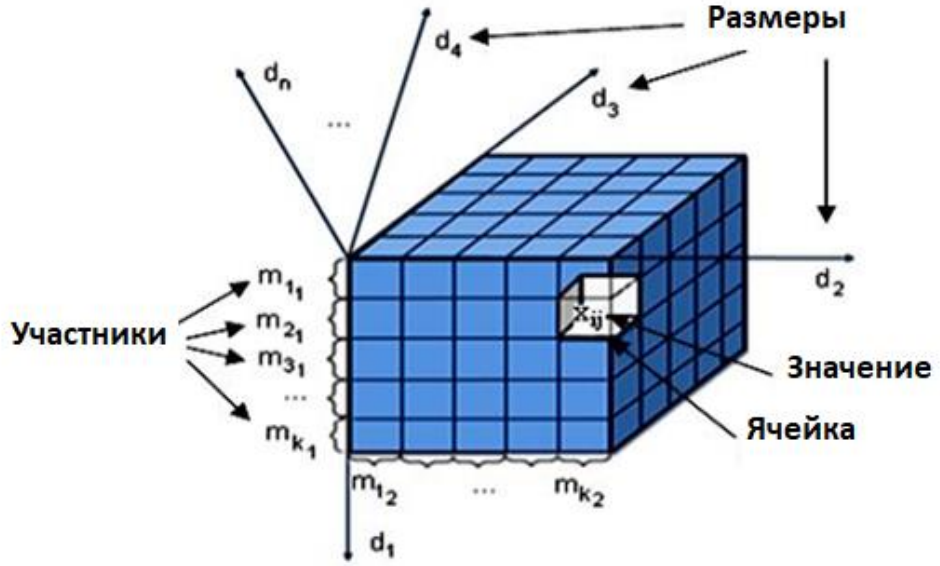

Рисунок 1 - Многомерный OLAPкуб

Для измерения используемых при реализации гиперкуба, и для описания элементовприменяется «Теория последовательности». В соответствии с теорией последовательности принимаются следующие значения:

D - сочетание размеров, но если учесть d_iкак участники размерных комбинаций, то имеем $d_i \in D$то есть $D=\{d_1,d_2,d_3,...,d_n\}$. Отсюда n – количестворазмерности, $d_1,d_2,d_3,...,d_n$- участники комбинации измерений размещаются на оси гиперкуба.

Каждый член размерной комбинации состоит из внутренних элементов, размещенных в соответствии с координационными осями гиперкуба. Таким образом, в соответствии с внутренними элементами измерения гиперкуба мы реализуем следующие значения:

$M_d=\{m_{1i},m_{2i},...,m_{ki}\}$–комбинация элементов измерения или внутренняя комбинация размеров,$i=1...n$ - этосочетание значений в измерениях.

Соответственно каждое измерение$d_1=\{m_{1_2},m_{2_1},...,m_{k_1}\}$

$d_1=\{m_{1_2},m_{2_1},...,m_{k_2}\}$,…,$d_1=\{m_{1_n},m_{2_n},...,m_{k_n}\}$состоит из комбинации внутренних элементов. Здесь,k_i - является значением внутренних элементов каждого измерения. Взяв M – в качестве участника измерений куба, через связь можно показать все внутренние члены измерений ($\cup$):$M=M_{d_1}\cup M_{d_2}\cup...\cup M_{d_n}$

Значение членов каждого измерения:d_1– комбинация элементов измеренияM_{d_1},значение членов d_2 – комбинация элементов размерности, M_{d_2}, в значении членов

$k_2=|M_{d_2}|$, d_3 - участник размерности M_{d_3}, значение членов $k_3=|M_{d_3}|$, ... , d_n элемент размерности M_{d_n}, значение членов$k_n=|M_{d_n}|$

Агрегированные значения соответствуют точке пересечения элементов гиперкуба. Однако возможное агрегированное значение равно всем точкам пересечения,выполненным в соответствии с гиперкубом. Это означает, что значение агрегированного значения напрямую зависит от значения членов. При оперативном аналитическом анализе, в связи с тем, что все агрегированные данные сохраняются в оперативной памяти, превышение количества агрегированных значений требует дополнительных ресурсов оперативной памяти.

При разработке структуры для формирования агрегационных значений гиперкуба учитываются сумма агрегированных значений гиперкуба (sum), среднее значение (avg), минимальное значение (min), максимальное значение (max), дисперсия, отклонение (variation) и т.д. Агрегация — это предварительно вычисленная структура данных, хранящаяся на основе значений данных. При разработке гиперкуба необходимо применять эффективное формирование агрегированных значений и структуры данных для организации оптимального пересечения в ходе оперативного аналитического анализа по набору показателей. Применяемая структура данных должна обеспечивать высокую производительность труда [M5]. В целях обеспечения указанных требований выполняются следующие этапы реализации эффективной разработки гиперкуба, рисунок 2:

1. Разработка структуры многомерного индекса по первичным ключам участников. Гиперкуб реализуется при первом его использовании для формирования структуры индексации. Если куб был построен ранее и требуется только обновление, то определяются вновь добавленные измерения оси элементов и координат, которые записываются, как новый индекс в многомерную индексирующую структуру. При условии, что новые элементы не добавляются, применяется предварительно сформированная структура индекса. Согласно заключению хранилища данных, если мы примем во внимание, что таблица измерений состоит из неизменяемых или редко изменяемых данных, то эти индексы в многомерной структуре индексации соответственно не изменяются или редко встречаются [17]. Конечно, изменение таблицы размеров возможно и в модели. В данном случае, как ужеупоминалось выше, обновление структуры индексации.

2. Поиск количественных значений из исходных данных с соответствующими индексами в соответствии с многомерным индексированием объединением найденных при поиске значений в массивы. Рис. M2 соответствующие начальные значения в соответствии с индексами считаются преобразованными в массивы x[1]][1][1]...[1],x[1][1][1]...[2], x[1]][1][1]...3],...,$x[k_1][k_2][k_3]...[k_n]$. Если соответствующие каким-либо индексам начальные количественные значения не найдены, (NULL) – в этот индекс помещается пустой указатель. Это означает, что ячейка в пересечении координации любогозначение члена равно NULL.

3. Размещение указателей в соответствии со значениями, полученными в результате получения суммы элементов массива и процесса

суммирования по многомерным индексам в уравнениях. (1) ~ (4). На этом этапе при загрузке данных выполняется предварительная агрегация данных, то есть через многомерный индекс в соответствии с объединенными элементами массива таблицы фактов $x[1][1][1]...[1]=1,2,3,...,j_{111...1}$,

$x[1][1][1]...[2]=1,2,3,...,j_{11...2}$,

$x[1][1][1]...[3]=1,2,3,...,j_{11...3}$,…,

$x[k_1]\ [k_2]\ [k_3]\ ...[k_n]=1,2,3,...,j_{kk_2k_3...k_n}$вычисляются значения суммы, берутся другие агрегированные значения

$$S_{all_{111...1}} = \sum_{i=1}^{j_{111...1}} x_i[1][1][1]...[1], \quad (1)$$

$$S_{all_{111...2}} = \sum_{i=1}^{j_{111...2}} x_i[1][1][1]...[2], \quad (2)$$

$$S_{all_{111...3}} = \sum_{i=1}^{j_{111...3}} x_i[1][1][1]...[3], \quad (3)$$

...

$$S_{all_{k_1k_2k_3...k_n}} = \sum_{i=1}^{j_{k_1k_2k_3...k_n}} x_i[k_1][k_2][k_3]...[k_n] \quad (4)$$

Из данных $S_{all_{111...1}}$, $S_{all_{111...2}}$,$S_{all_{111...3}}$,..., $S_{all_{k_1k_2k_3...k_n}}$– начальные агрегированныезначения взяты путем суммирования, но $j_{111...1}$,$j_{111...2}$,$j_{111...3}$,...,$j_{k_1k_2k_3...k_n}$– это значения, объединенные в один массив.

Вычисленные значения суммы агрегации сохраняются в отдельной структуре, а указатель помещается на адрес значения суммы агрегации в структуре многомерного индексирования. Это означает, что только одно значение равно одному многомерному индексу. В гиперкубе измерения — это оси, а участники — это координаты соответствующих осей. Если предварительно не вычислять агрегированные значения, берущие пересечение координат элементов, то параллельно незаполненному согласованию значения пересечения в кубе появляется согласование. Если мы рассмотрим координацию пересечения куба как матрицу, то там есть пустые ячейки, вы обнаружите, что матрица, содержащая количественные значения, встречается с разрежением, с одной стороны.

В процессе вычисления агрегированного значения все участники, равные 0, удаляются. Только при согласовании членов выбирается пересечение, равное определенному значению, и объединяется в одно значение суммы, если они совпадают с индексами. Уменьшить емкость значений за счет агрегирования по предварительному расчету значений фактов и обеспечения в дальнейшем быстрой работы системы и снижения требований к оперативной памяти

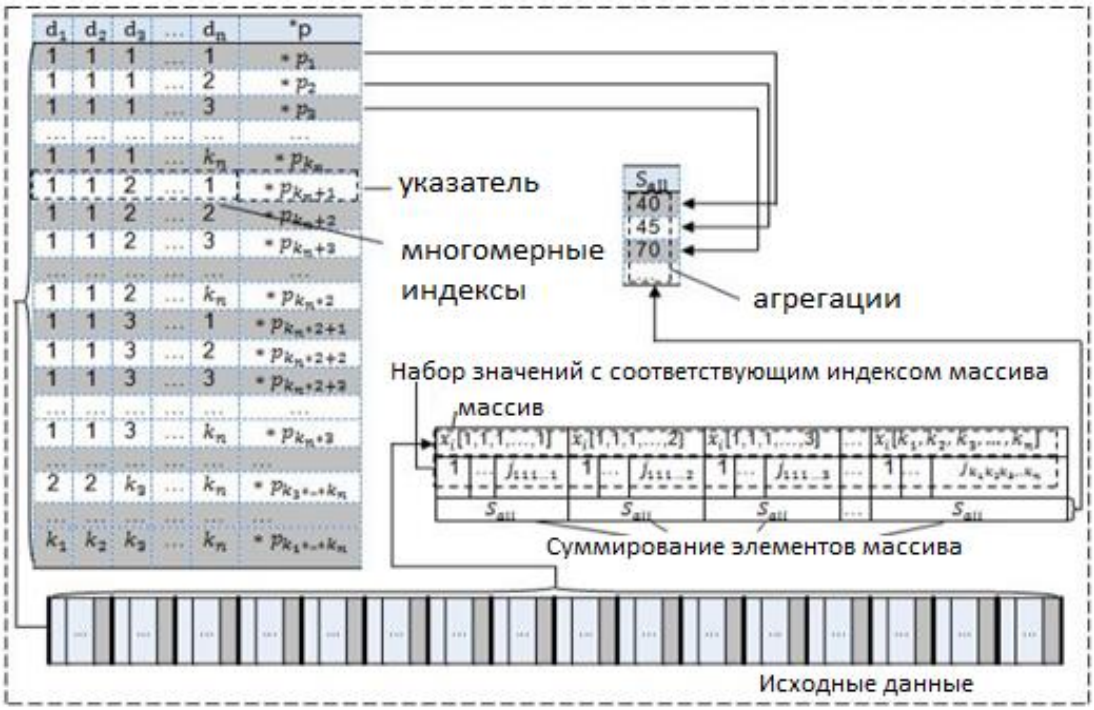

Рисунок 2 - Структура многомерного индекса и схема формирования агрегированных значений за счет предварительной агрегации исходных фактических данных

Символы на рисунке 2:

p_i – указатели на агрегированные значения;

$x_i[1]...[k_n]$–массивы;

$j_1...k_n$ – количество участников массива;

k_i – ключевое значение соответствующего измерения;

S_{all} – сумма элементов массива (агрегатное значение);

$\Rightarrow$ взаимосвязь компонентов схемы.

4. После вычисления начальных значений (предварительная агрегация) общее конечное значение, в соответствии с одним измерением, из пересечений двух измерений, и вычисление значений агрегации проводится до пересечения n-1 измерения. В соответствии с пересечением измерений формируются отдельные структуры и формируются элементы в отдельных структурах в соответствии с многомерным индексом первичных ключей. Структуры показаны в разделах n-3, n-2, n-1, n на рисунке 3.

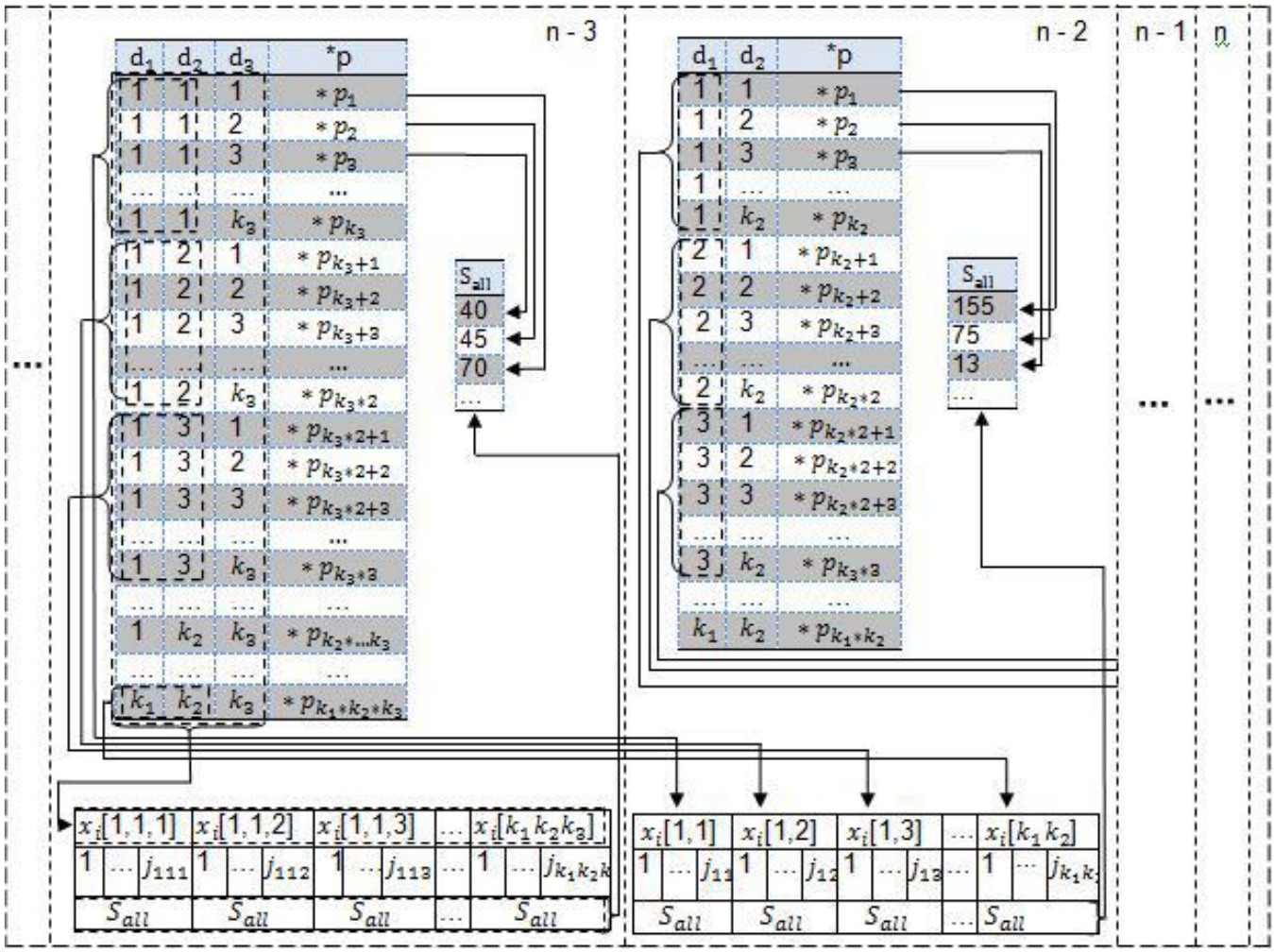

Рисунок 3 - Схема расчета и формирования агрегированных показателей гиперкуба из предварительно агрегированных данных

Расчет производится следующим образом: в соответствии с сформированной отдельной структурой производится сравнение для предварительно рассчитанной агрегированной структуры. Во время сравнения каждый многомерный индекс сравнивается с отдельными структурными индексами. Когда индексы соответствуют, значения агрегации записываются в один массив и подсчитывается сумма элементов массива. Значения агрегации, полученные в результате суммирования элементов массива, сохраняются в отдельной структуре, а указатели размещаются, насколько это возможно, на сохраненные значения агрегации в структуре индексации.

В процессе получения кубических срезов формируется маска индексации из используемых ключей измерений, и в соответствии этой с маской берутся заранее рассчитанные агрегированные значения. Агрегирование рассчитанных значений в разрезы гиперкуба осуществляется путем объединения и суммирования начальных значений в массивы от минимальной степени до окончательных определяющих значений.

Например, расчет агрегированных значений, который берется в соответствии с двумя измерениями (уравнения (5) ~ (8), рисунок 3 n - 2):

$$S_{all_{11}} = \sum_{i=1}^{j_{11}} x_i[1][1], \quad (5)$$

$$S_{all_{12}} = \sum_{i=1}^{j_{12}} x_i[1][2], \quad (6)$$

$$S_{all_{13}} = \sum_{i=1}^{j_{13}} x_i[1][3], \quad (7)$$

...

$$S_{all_{k_1k_2}} = \sum_{i=1}^{j_{k_1k_2}} x_i[k_1][k_2]. \quad (8)$$

Здесь, $S_{all_1}, S_{all_2}, S_{all_3}, \ldots, S_{all_{k_1k_2}}$ – агрегированные расчетные значения по двум размерностям, $j_{11}, j_{12}, j_{13}, \ldots, j_{k_1k_2}$ - значения объединены в один массив.

Далее в соответствии с отдельной размерностью осуществляется расчет значений. Если мы рассматриваем данные гиперкуба как состоящие из различных измерений, то агрегированные значения, взятые в соответствии с одним измерением, равны окончательным заключительным значениям в виде столбца и разрешаемой строки. В конце, взяв сумму всех значенийподсчитывается итоговое значение.

В этомисследовании даны основные понятия модели многомерных данных в виде гиперкуба, а последовательность теории применена для описания структуры гиперкуба. Для формирования многомерного гиперкуба разрабатываются установление многомерной индексной структуры эффективного пересечения и установленных значений. По многомерной структуре производится предварительная агрегация исходных данных, а в ходе анализа предлагаются способы уменьшения объема данных, сохраняемых в оперативной памяти.

Развитая структура OLAP дает возможность повысить производительность работы технологии аналитической обработки данных. Можно пересекаться с разрезами данных через индексирующую структуру. Значение, помещенное в многомерный индекс, позволяет напрямую пересекаться с физическим адресом агрегированных значений[35,36,37,38,39].

2.2 Колоночная схема NoSQL БД

Колоночная схема NoSQL базы данных отличается от традиционных реляционных баз данных и использует особую модель и структуру для организации данных. Основные концепции, связанные с колоночной схемой NoSQL баз данных, включают модель данных, структуры данных, семейства столбцов (column families) и организационную структуру:

Семейство столбцов (column family) является структурой для организации и группировки связанных столбцов внутри базы данных. Семейство столбцов содержит набор столбцов, которые имеют общую характеристику или связь между собой. Оно может быть представлено как аналог таблицы в реляционной

модели баз данных, но с более гибкой структурой. Каждое семейство столбцов может иметь свою схему, определяющую типы данных для каждого столбца и правила доступа к данным.

Например, в семействе столбцов "Users" может быть столбец "Name" с типом данных "строка" и столбец "Age" с типом данных "число",рисунок 4.

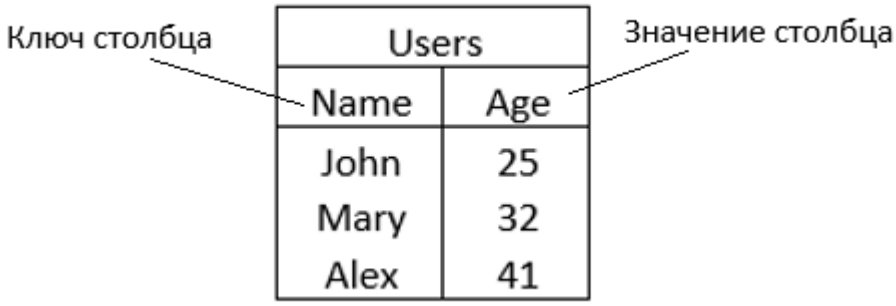

Рисунок 4 -Семейство столбцов

В целом, семейства столбцов в NoSQL базах данных предоставляют гибкую структуру для организации и управления данными, позволяя эффективно работать с различными типами данных и обеспечивая высокую производительность при выполнении операций с базой данных.

Запись (Row) в NoSQL базах данных представляет собой набор данных, связанных между собой и образующих единицу информации. В отличие от реляционных баз данных, где запись представляет собой строку с набором полей, в NoSQL базах данных запись может иметь более гибкую структуру, таблица1.

Таблица 1.

Запись (Row)в NoSQL.

Key	Name	Age
1	John	25

В данном примере, запись содержит три поля: "Key", "Name" и "Age". Поле "Key" представляет уникальный идентификатор или ключ записи, который помогает идентифицировать и извлекать конкретную запись из базы данных. Поле "Name" содержит имя пользователя, а поле "Age" содержит возраст.Графическое представление состоит из таблицы, где каждая строка соответствует одной записи, а каждый столбец представляет отдельное поле данных. Значения ячеек таблицы заполняются конкретными данными для каждой записи.

Колоночная база данных (Columnar Database) — это тип NoSQL базы данных, где данные организованы и хранятся по колоночной схеме. В отличие от реляционных баз данных, где данные организованы и хранятся по строковой схеме, в колоночной базе данных данные группируются по столбцам, а не по

строкам. Каждый столбец содержит отдельное поле данных, и все значения этого поля хранятся последовательно. В данном примере, каждый столбец представляет отдельное поле данных: "ID", "Name", "Age" и "Salary". Каждая запись представляет отдельную строку, где значения каждого поля данных располагаются последовательно, рисунок 5, 6, 7.

```
Колонка ID:       1, 2, 3
Колонка Name:     John, Mary, Alex
Колонка Age:      25, 32, 41
Колонка Salary:   5000, 6000, 7000
```

Рисунок 5 - Структура колоночной базы данных

ID
1
2
3

Name
John
Mary
Alex

Age
25
32
41

Salary
5000
6000
7000

ID
1
2
3

Name
John
Mary
Alex

Age
25
32
41

Salary
5000
6000
7000

Рисунок 6 - Структура колоночной базы данных

```
+-----+-------+-------+--------+
|  ID | Name  |  Age  | Salary |
+-----+-------+-------+--------+
|  1  | John  |  25   |  5000  |
|  2  | Mary  |  32   |  6000  |
|  3  | Alex  |  41   |  7000  |
+-----+-------+-------+--------+
```

Рисунок 7 - Структура колоночной базы данных

Форматсжатиястолбцов (ColumnCompressionFormat): эта модель использует различные алгоритмы сжатия данных для эффективного хранения значений столбцов. Вместо хранения каждого значения в отдельной ячейке, данные сжимаются и хранятся в оптимизированном формате. Это позволяет сократить объем хранимых данных и улучшить производительность при выполнении запросов. Пример представления модели сжатия столбцов, рисунок 8.

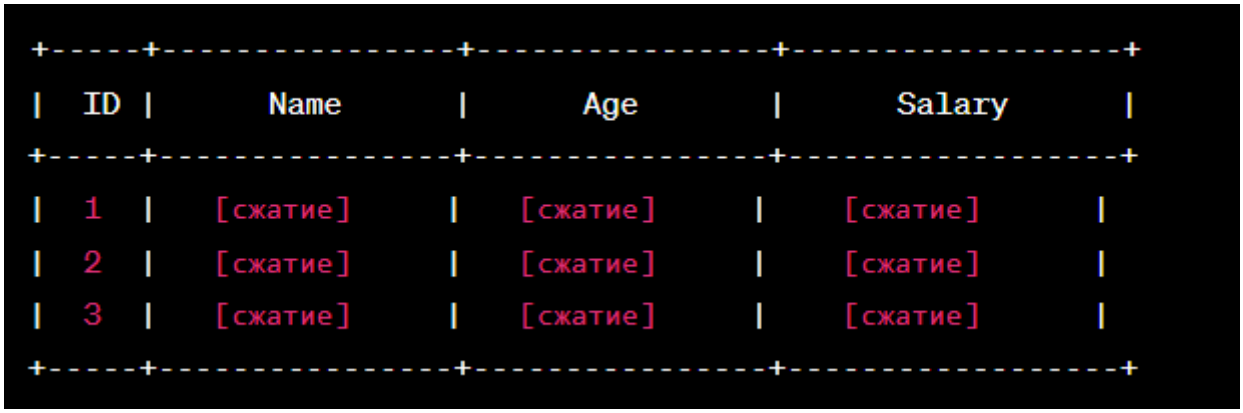

Рисунок 8- Модель сжатия столбцов

Колоночная схема NoSQL базы данных, такая как ClickHouse, представляет собой модель хранения данных, где данные организуются и хранятся в виде столбцов. В отличие от традиционных реляционных баз данных, где данные хранятся по строкам, колоночная модель обладает рядом преимуществ, которые делают ее привлекательным выбором в некоторых сценариях. Преимущества колоночной схемы NoSQL базы данных, такой как ClickHouse, включают:

- эффективность сжатия
- высокая производительность аналитических запросов
- горизонтальное масштабирование
- поддержка аналитических операций

2.3 ClickHouse

ClickHouse — это коллективный аналитический СУБД, разработанный компанией Yandex, является проектом с открытым исходным кодом, что позволяет разработчикам и администраторам легко модифицировать или интегрировать его в свои системы. Он оптимизирован для обработки больших объемов данных в реальном времени и использует колоночную архитектуру для эффективной работы с многокритериальными индексами. Он поддерживает SQL и имеет широкий набор функций аналитики данных, включая группировку, агрегацию, аналитику временных рядов и географическую аналитику. Также он может работать как с журнальными данными, так и с историческими.

ClickHouse использует колоночную архитектуру для хранения данных, которая отличается от традиционной строковой архитектуры RDBMS. Это делает его более эффективным при работе с большими объемами данных и многокритериальными индексами. Так же ClickHouse поддерживает запросы на математические операции над колонками без необходимости считывать всю таблицу, и не поддерживает как транзакции и операции, связанные с изменением данных. В целом, можно смело утверждать, что ClickHouse это NoSQL СУБД, но при этом оставляет доступ к SQL запросам и индексацию данных. В строковой СУБД, данные хранятся в таком виде, таблица 2:

Таблица 2

Пример расположения данных в строковой СУБД.

Строка	ID	Enable	Name	Time	EventTime
#0	88354350772	1	John	1	18.05.2023 5:19
#1	92329507758	1	Mary	1	18.05.2023 8:10
#2	69953706324	1	Alex	1	18.05.2023 7:38
#N	...	...	...	...	...

В столбцовых СУБДзначения из разных столбцов хранятся отдельно, а данные из одного столбца вместе иимеют такой вид, таблица 3:

Таблица 3

Пример расположения данных в столбцовой СУБД.

Строка:	#0	#1	#2	#N
ID:	88354350772	92329507758	69953706324	...
Enable:	1	0	1	...
Name:	John	Mary	Alex	...
Time:	1	1	1	...
EventTime:	18.05.2023 5:19	18.05.2023 8:10	18.05.2023 7:38	...

Различные варианты хранения данных оптимальны для разнообразных рабочих сценариев. Рабочий сценарий описывает типы запросов, их частоту и соотношение между ними, объем данных, считываемых при каждом запросе (количество строк, столбцов или байтов), соотношение между чтением и обновлением данных, размер рабочего набора данных и его локальное использование, использование транзакций и уровень их изоляции, требования к дублированию данных и обеспечению логической целостности, а также требования к задержкам выполнения и пропускной способности для каждого типа запросов и т.д.Чем больше система нагружена, тем важнее становится специализация для определенного рабочего сценария, и тем более конкретными должны быть такие специализации. Нет универсальной системы, которая идеально подходила бы для значительно различных рабочих сценариев. Если система способна обслуживать широкий спектр рабочих сценариев, то при высокой нагрузке она будет плохо справляться со всеми сценариями работы или хорошо работать только с одним из них.

Описание схемы базы данных ClickHouse может быть представлено в формате JSON-подобной структуры, где каждый элемент описывает таблицу и ее столбцы. Вот пример графического представления схемы базы данных ClickHouse в стиле JSON, рисунок 9:

Рисунок 9- Пример схемы базы данных ClickHouse в JSON формате

В приведенном примере есть две таблицы: "users" и "orders". Каждая таблица имеет свои столбцы с указанными их типами данных. Например, в таблице "users" есть столбцы "id" (тип UInt32), "name" (тип String), "age" (тип UInt8) и "email" (тип String). В таблице "orders" есть столбцы "order_id" (тип UInt64), "user_id" (тип UInt32), "product" (тип String), "quantity" (тип UInt16) и "price" (тип Float32).

Основные характеристики рабочего сценария OLAP:

1. Преобладают запросы на чтение.

2. Обновление данных происходит пакетами, которые включают более 1000 строк, а не по одной строке, либо данные не обновляются вообще.

3. Данные добавляются в базу данных, но не изменяются после этого.

4. При чтении извлекается значительное количество строк из базы данных, но только небольшое подмножество столбцов.

5. Таблицы являются "широкими", то есть содержат много столбцов.

6. Запросы поступают сравнительно редко (обычно не более сотни запросов в секунду на сервер).

7. При выполнении простых запросов допустимы задержки около 50 мс.

8. Значения в столбцах относительно небольшие - числа и короткие строки (например, 60 байт для URL).

9. Требуется высокая пропускная способность для обработки одного запроса (до миллиарда строк в секунду на одном сервере).

10. Транзакции не используются.

11. Низкие требования к согласованности данных.

12. В запросе присутствует одна большая таблица, в то время как все остальные таблицы являются небольшими.

13. Результат выполнения запроса существенно меньше исходных данных, то есть данные фильтруются или агрегируются, а результат помещается в оперативную память на одном сервере.

Очевидно, что OLAP-сценарий работы значительно отличается от других распространенных сценариев, таких как OLTP или сценарии работы с ключ-значение. Следовательно, нет смысла пытаться использовать базы данных OLTP или базы данных ключ-значение для обработки аналитических запросов, если необходимо достичь хорошей производительности. Основные причины, по которым СУБД ClickHouseподходит для сценария OLAP:

1. Для выполнения аналитических запросов требуется чтение небольшого количества столбцов из таблицы. В столбцовых базах данных возможно чтение только необходимых данных. Например, если необходимо только 5-6 столбцов из 100, то можно ожидать сокращения ввода-вывода в 20 раз.

2. Так как данные читаются пакетами, их легче сжимать. Данные, хранящиеся по столбцам, также сжимаются лучше. Это приводит к дополнительному сокращению объема ввода-вывода.

3. В результате сокращения ввода-вывода в системный кэш может быть загружено больше данных.

Пример создания OLAP-кубов в ClickHouse:

1. Использование встроенных в ClickHouse функций агрегации для создания многомерного куба путем агрегирования данных из таблицы.

2. Использование предложения GroupBy в ClickHouse для агрегирования данных на основе определенных измерений.

3. Использование внешних инструментов, таких как Apache Superset или Metabase, для подключения к ClickHouse и создания кубов OLAP с графическим пользовательским интерфейсом.

Стоит отметить, что ClickHouse предназначен для крупномасштабной обработки данных и может быть эффективным выбором для рабочих нагрузок OLAP [40,41,42].

2.4 Модель формирования OLAP куба из ClickHouse

В ClickHouse, модель формирования OLAP-куба может быть основана на структурах данных, оптимизированных для аналитических операций, таких как агрегации и фильтрация данных. Одна из наиболее распространенных моделей формирования OLAP-куба в колоночных NoSQL базах данных — это модель столбцов (columnar model). В данной модели, данные хранятся по столбцам, а не по строкам, что позволяет эффективно хранить и обрабатывать данные при агрегации и аналитических операциях. Вот общий подход к формированию OLAP-куба в колоночных NoSQL базах данных:

Определение измерений(Dimensions): Измерения представляют собой атрибуты или характеристики данных, которые используются для организации и агрегации данных в OLAP-кубе. Каждое измерение может иметь связанные с ним уровни детализации.

Определение фактов(Facts): Факты представляют числовые значения, которые агрегируются и анализируются в OLAP-кубе. Они представляют собой количественные метрики.

Иерархии измерений (Dimension Hierarchies): Каждое измерение в OLAP-кубе имеет иерархию, которая описывает уровни детализации этого измерения.

Создание колоночных структур данных: для каждого измерения и факта создаются отдельные колоночные структуры данных, которые оптимизированы для хранения и обработки соответствующих данных. Колоночные структуры данных позволяют эффективно сжимать и фильтровать данные, а также выполнять агрегации по столбцам.

OLAP-куб в БД ClickHouse формируется на основе данных, которые хранятся в таблицах. Он представляет собой многомерную структуру, которая облегчает анализ данных и их интерактивное исследование. Модель формирования OLAP-куба для БД ClickHouse включает несколько этапов:

1. Подготовка исходных данных: данные, которые будут использоваться для формирования OLAP-куба, должны быть подготовлены заранее. Это может включать в себя очистку данных, преобразование их в нужный формат и т.д.
2. Определение структуры OLAP-куба: структура OLAP-куба определяется на основе анализа бизнес-процессов и требований к анализу данных. Она может включать в себя несколько измерений, факты, иерархии и прочие параметры.
3. Создание таблицы для OLAP-куба: после определения структуры OLAP-куба необходимо создать таблицу для его хранения. Для этого используется команда CREATE TABLE, в которой определяются поля таблицы и их типы данных.

4. Загрузка данных в OLAP-куб: данные загружаются в OLAP-куб из исходных таблиц в БД ClickHouse с помощью операции INSERT INTO. Для загрузки данных в OLAP-куб может использоваться синтаксис SELECT INTO, который позволяет выбирать и загружать только определенные поля из исходных таблиц.

5. Создание индексов и материализованных представлений: для оптимизации работы OLAP-куба и ускорения запросов могут быть созданы индексы и материализованные представления. Индексы позволяют быстро находить нужные данные в таблице, а материализованные представления - хранить результаты запросов, что позволяет ускорить их выполнение.

6. Запросы к OLAP-кубу: OLAP-куб можно анализировать с помощью запросов на выборку данных, которые могут включать в себя различные операции, такие как группировка, фильтрация, агрегирование и т.д.

Таким образом, модель формирования OLAP-куба для БД ClickHouse включает в себя несколько этапов, начиная от подготовки данных и определения структуры OLAP-куба и заканчивая анализом данных с помощью запросов на выборку данных [43,44,45], рисунок 10.

Рисунок 10 – Пошаговое формирования OLAP-куба в ClickHouse

2.5 Традиционные подходы к формированию агрегированных

Традиционные подходы к формированию агрегированных показателей включают использование OLAP-систем, таких как Microsoft Excel, Oracle Essbase, IBM Cognos и других. Эти системы предоставляют возможность агрегирования и анализа данных, включая данные из различных источников и в разных форматах, обеспечивая широкий спектр функциональных возможностей для аналитики и отчетности.

В системах OLAP формирование агрегированных показателей может происходить с использованием OLAP-кубов, содержащих предварительно вычисленные значения показателей для всех возможных комбинаций измерений. Создание таких кубов включает определение данных, используемых для формирования показателей, и применение операций агрегации.

В ClickHouse формирование агрегированных показателей осуществляется через выполнение запросов, работающих напрямую с данными в таблицах. Благодаря компактному хранению данных и оптимизированной обработке запросов, ClickHouse обеспечивает высокую скорость выполнения запросов, что позволяет эффективно формировать агрегированные показатели и проводить аналитику данных. Это позволяет быстро получать необходимую информацию для принятия решений и анализа данных.Для формирования агрегированных показателей в ClickHouse могут использоваться различные функции агрегирования, такие как SUM, AVG, COUNT и др., которые позволяют выполнять агрегирование данных по определенным полям и условиям. Также в ClickHouse доступны инструменты для создания материализованных представлений, которые предварительно вычисляют агрегированные показатели для определенных комбинаций измерений и значений. Это позволяет ускорить выполнение запросов на анализ данных.

Таким образом, формирование агрегированных показателей в традиционных подходах и в ClickHouse имеет свои особенности. В ClickHouse это может включать использование OLAP-систем, кубов, функций агрегирования и материализованных представлений. Эти инструменты позволяют анализировать данные и получать необходимые показатели, необходимые для принятия решений. [46,47].

2.6 СУБД для OLAP: ClickHouse или PostgreSQL

Были проведены различные тесты для оценки производительности ClickHouse. Эти тесты сравнивают ClickHouse с другими системами баз данных, такими как Apache Hive, Apache Impala и Apache Spark, по таким показателям, как производительность запросов, масштабируемость и использование ресурсов.

Результаты этих тестов различаются в зависимости от конкретного варианта использования, размера данных и сложности запроса. Однако во

многих случаях ClickHouse демонстрирует отличную производительность по сравнению с другими системами, особенно когда речь идет об обработке больших объемов данных и аналитике в реальном времени. ClickHouse способен эффективно обрабатывать запросы на миллионы строк данных за секунды благодаря своей колоночной структуре хранения данных, которая ускоряет агрегацию и анализ больших объемов данных.

Важно помнить, что эталонные тесты и тесты производительности — это лишь один из аспектов оценки системы баз данных. При выборе СУБД для OLAP или любого другого варианта использования следует также учитывать другие факторы, такие как простота использования, совместимость с другими инструментами и системами, а также поддержка конкретных бизнес-требований.

PostgreSQL является реляционной базой данных с открытым исходным кодом, которая поддерживает множество типов данных и обладает широким функционалом для обработки транзакций, репликации и масштабирования. Она часто используется в крупных проектах, где требуется хранение и обработка больших объемов структурированных данных. Одним из основных отличий между PostgreSQL и ClickHouse является подход к обработке данных. PostgreSQL подходит для работы с транзакционными данными и поддерживает ACID-транзакции (Atomicity, Consistency, Isolation, Durability), что обеспечивает целостность данных. ClickHouse, с другой стороны, более ориентирован на аналитические запросы и поддерживает более ограниченный набор функций транзакций.

Еще одним важным отличием является производительность. ClickHouse, благодаря своей колоночной структуре хранения данных, может обрабатывать запросы на миллионы строк данных за секунды, что делает его привлекательным выбором для задач аналитики. PostgreSQL, хотя и обладает более общим назначением и может обрабатывать как транзакционные, так и аналитические запросы, может быть менее производительным при работе с большими объемами данных. Он обладает богатым функционалом и поддерживает сложные операции с данными, но его производительность может быть ниже по сравнению с ClickHouse при выполнении аналитических запросов.[48,49,50].

3. Практические реализация и тесты

В данном исследовании предполагается проведение нагрузочного тестирования двух систем управления базами данных (СУБД): ClickHouse и PostgreSQL. Обе СУБД имеют разные цели и специфику использования, как описано выше. Выбор между PostgreSQL и ClickHouse зависит от конкретных потребностей проекта. Если требуется хранение и обработка структурированных данных, включая транзакционные данные, то PostgreSQL является предпочтительным выбором. Если же требуется работа с большими объемами данных, требующими аналитической обработки, то ClickHouse может быть более подходящим вариантом.

В данном исследовании рассматривается пример хранения данных по звонкам CDR (Call Detail Record) с телекоммуникационных устройств Cisco Unified Border Element[2]. Эти данные используются для совершения и принятия звонков в Контакт-центре, базирующемся на АО «Kaspi Bank».

Первоначально требуется подготовить данные для дальнейшей обработки. CDR (Call Detail Record) на роутерах Cisco Border Element (CUBE) представляют собой записи, содержащие информацию о телефонных вызовах, проходящих через данный элемент сети. CDR служит для регистрации сведений о вызовах, включая информацию о времени, длительности, номере вызывающего и вызываемого абонентов, а также другую полезную информацию.

Формирование CDR на роутерах Cisco Border Element осуществляется путем конфигурации соответствующих параметров и функций. Сначала активируется функция CDR, затем определяется формат, в котором CDR будут сохраняться. Обычно используются стандартные форматы, такие как CSV (Comma-Separated Values) для представления записей в формате значений, разделенных запятымии протокол SFTP (Secure File Transfer Protocol) для передачи данных.

Затем определяются поля и информация, которые должны быть включены в CDR. Это может включать номера телефонов, время начала и окончания вызова, длительность, протокол связи, качество связи и другие параметры.В представленном примере, данные по звонку cdrфайле с устройства CUBEipcc-gw2, выглядит следующим образом:

1663660800,13329677,1,2,"F554AB8B 37F011ED BD17F337 D1F97635","77073991935","","13:59:16.616 ALA Tue Sep 20 2022","13:59:20.386 ALA Tue Sep 20 2022","14:00:00.016 ALA Tue Sep 20 2022","14:00:00.016 ALA Tue Sep 20 2022","10 ","normal call clearing (16)","originate",0,"speech",0,0,0,0,"90046053","552205040444","77073991935","T

[2] https://www.cisco.com/c/en/us/products/unified-communications/unified-border-element/index.html

WC","09/20/2022
13:59:16.601","552205040444","77073991935",0,12982848,F554AB8B 37F011ED
BD17F337 D1F97635,CB650D,"","","",""

Далее, со всех устройств CUBE, по протоколу SFTPданные отправляются на сетевое хранилище ipcc-cdr-syslog, далее с помощью парсера данные, по порту 5432 отправляются в БД PostgreSQL, рисунок11, а для ClickHouse данные отправляются по порту 80 на точку балансировкиphone-billing-clickhouse.hq.bc, рисунок12.

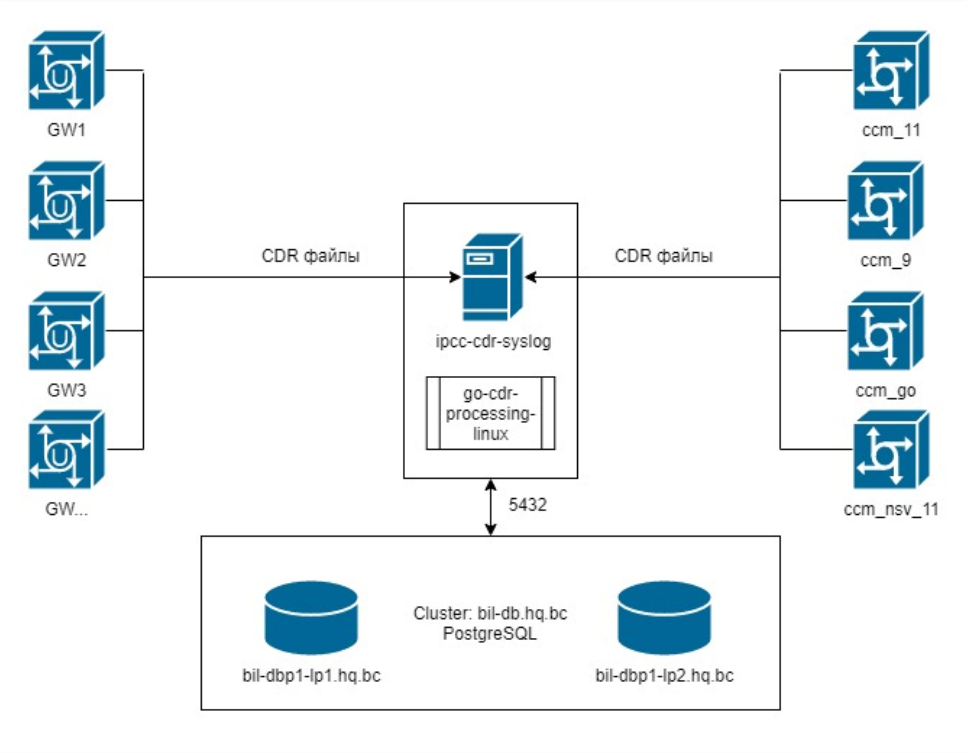

Рисунок 11 - Схема отправки данных в СУБД PostgreSQL

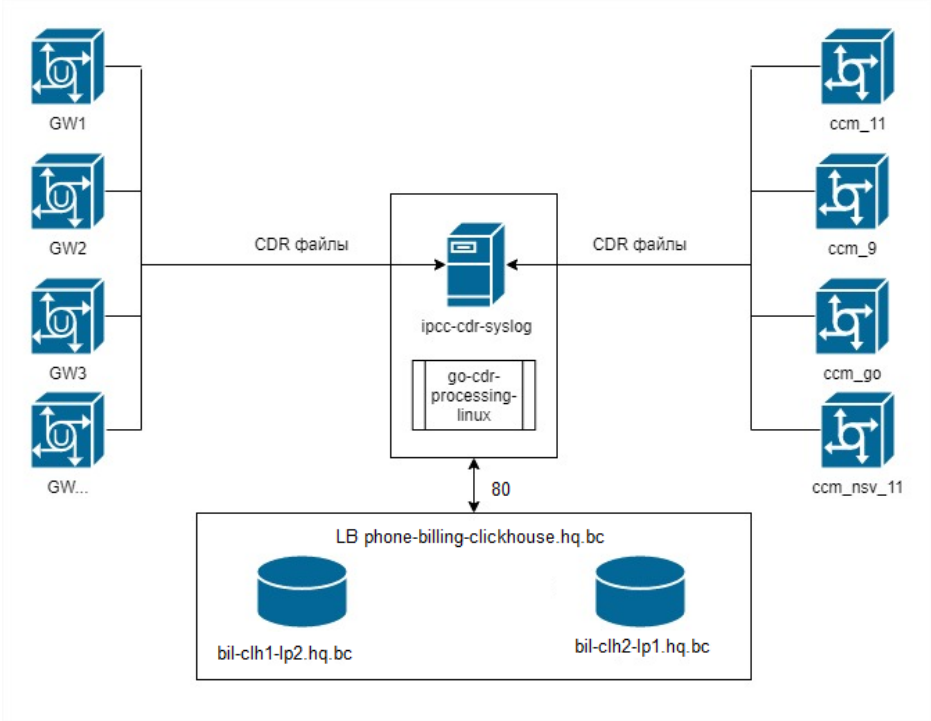

Рисунок 12 - Схема отправки данных в СУБД ClickHouse

Далее, в процессе подготовки данных, используется парсер, который формирует модель и структуру данных, включая создание индексов, определение названий, типов и форматов полей, для последующей передачи данных в СУБД PostgreSQL и СУБД ClickHouse, рисунок 13.

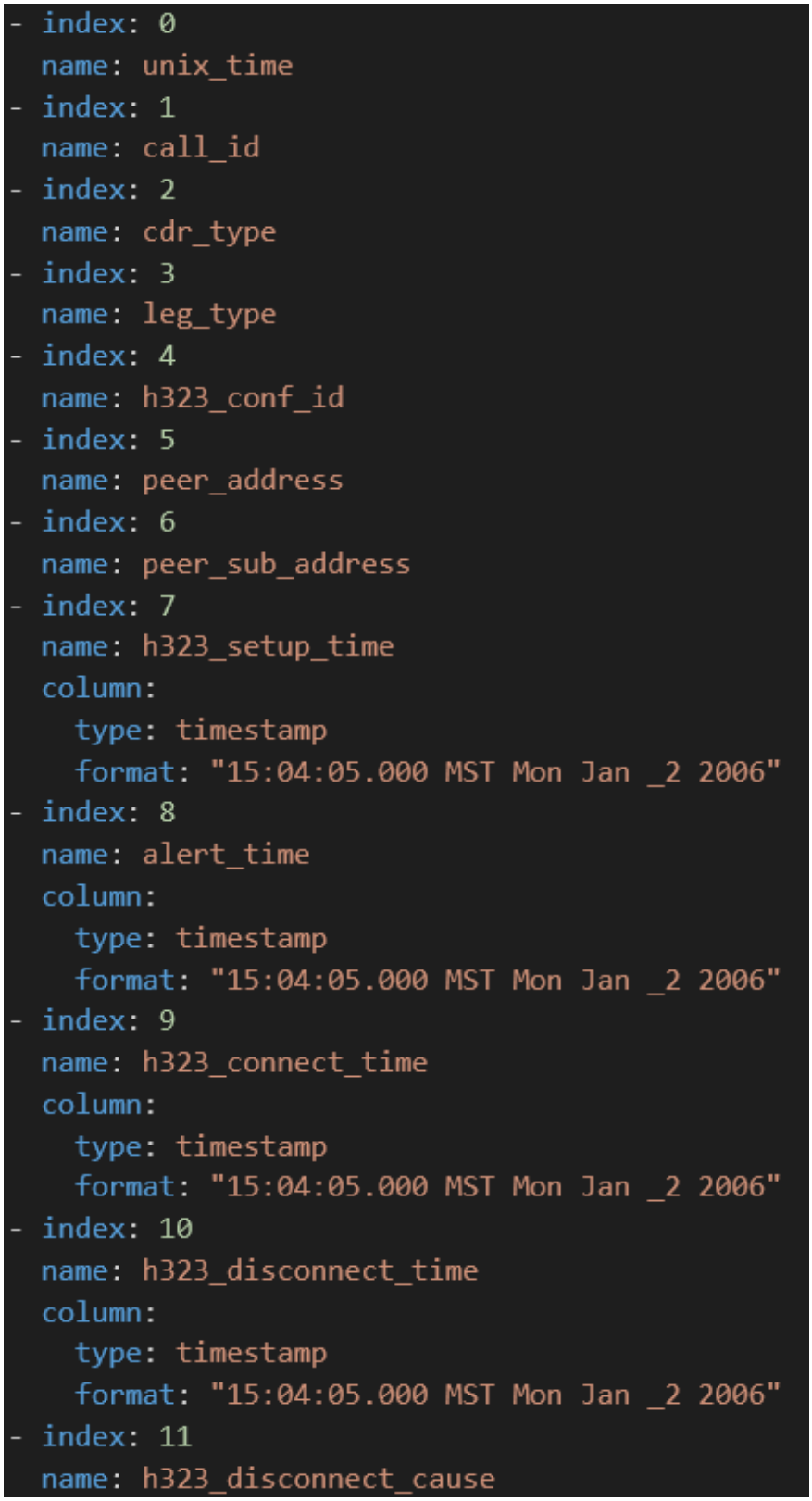

Рисунок 13 - Модель и структура данных для передачи данных в СУБД PostgreSQL и СУБД ClickHouse

Следующий этап в данном исследовании — это формирование DDL (Data Definition Language).Для обоих СУБД данный этап имеет различия. ДляClickHouse, формирование DDL выполняется для создания и изменения схемы базы данных, таблиц и других объектов данныхи позволяет определить структуру данных, их типы, ограничения и другие атрибуты.

В ClickHouse для формирования используются следующие ключевые слова и конструкции, рисунок 14:

```
    create table if not exists bilcdr.cube_new
(
    unix_time             Int32,
    call_id               Int32,
    cdr_type              Int32,
    leg_type              Int32,
    h323_conf_id          String,
    peer_address          String,
    peer_sub_address      String,
    h323_setup_time       DateTime64(3),
    alert_time            DateTime64(3),
    h323_connect_time     DateTime64(3),
    h323_disconnect_time  DateTime64(3),
    h323_disconnect_cause String,
    disconnect_text       String,
    h323_call_origin      String,
    charged_units         Int32,
    info_type             String,
    paks_out              Int32,
    bytes_out             Int32,
    paks_in               Int32,
    bytes_in              Int32,
    username              String,
    clid                  String,
    dnis                  String,
    column23              String,
    column34              String,
    filename              String,
    gateway               String,
    linenum               Int32,
    created               DateTime64(3),
    source                String,
    create_file           String
)
    engine =
ReplicatedReplacingMergeTree('/clickhouse/tables/bilcdr/cube_new
', '{replica}')
        PARTITION BY toYYYYMM(toDate(h323_setup_time))
        ORDER BY (h323_conf_id, h323_setup_time,
h323_connect_time, h323_disconnect_time, clid, dnis, column23,
                  filename)
        SETTINGS index_granularity = 8192;
```

Рисунок 14 – DDLв ClickHouse

В связи с тем, что ClickHouse является колоночной системой управления базами данных, данные со всех устройствCUBE (GW), могут быть эффективно храниться в одной таблице, используя оптимальные параметры для этого. Кроме того, благодаря такому подходу к хранению данных, можно получить значительный прирост производительности при выполнении запросов, так как ClickHouse позволяет быстро анализировать большие объемы данных. Для

эффективного хранения данных в одной таблице могут использоваться следующие параметры:

unix_time: поле, содержащее время звонка в формате Unix time (количество секунд, прошедших с 1 января 1970 года).

call_id: уникальный идентификатор звонка, который помогает идентифицировать каждый отдельный вызов.

cdr_type: тип CDR (Call Detail Record), указывающий на характер вызова (например, входящий, исходящий или внутренний).

leg_type: тип соединения (например, входящий, исходящий или промежуточный).

h323_conf_id: идентификатор конференции H323 (H.323 - стандарт протокола для голосовой и видеосвязи в IP-сетях).

peer_address: IP-адрес (или доменное имя) удаленного узла, с которым установлено соединение.

peer_sub_address: дополнительная информация об удаленном узле, например, номер порта.

h323_setup_time: время установки соединения H.323.

alert_time: время, в течение которого вызываемый абонент слышит звонок перед ответом.

h323_connect_time: время установки физического соединения H.323.

h323_disconnect_time: время, когда соединение H.323 было разорвано.

h323_disconnect_cause: причина разрыва соединения H.323.

disconnect_text: дополнительная информация о причине разрыва соединения.

h323_call_origin: источник вызова H.323 (например, внутренний или внешний).

charged_units: количество единиц, по которым взимается плата за звонок.

info_type: тип информации о звонке.

paks_out: количество пакетов (сетевых пакетов), отправленных исходящей стороной.

bytes_out: количество переданных байтов исходящей стороной.

paks_in: количество пакетов, полученных входящей стороной.

bytes_in: количество полученных байтов входящей стороной.

username: имя пользователя, связанное с вызовом.

clid: номер Calling Line Identification (идентификации вызывающей линии), который отображается на телефоне вызываемого абонента.

dnis: номер Dialed Number Identification Service (идентификации набранного номера), который указывает на набранный вызывающим номер.

column23, column24, column25, ..., column34: эти поля представляют дополнительные столбцы, которые могут быть использованы для хранения дополнительной информации или пользовательских данных. Конкретное назначение этих столбцов зависит от конкретной реализации и потребностей проекта.

filename: имя файла, в котором содержится запись CDR.

gateway: поле, которое является определителем с какого оборудования пришли данные

linenum: номер линии или порта, через который был совершен звонок.

created: время создания записи CDR.

source: источник данных CDR или источник записи.

create_file: имя файла, в котором была создана запись CDR.

engine = ReplicatedReplacingMergeTree('/clickhouse/tables/bilcdr/cube_new', '{replica}': в ClickHouse является типом движка таблицы, который обеспечивает репликацию данных и автоматическую замену записей в случае конфликтов и является комбинацией двух других типов движков.ReplicatedMergeTree обеспечивает репликацию данных между несколькими узлами ClickHouse, чтобы обеспечить отказоустойчивость и устойчивость к сбоям. Он реплицирует данные на основе логических реплик, которые могут быть размещены на разных узлах в кластере ClickHouse. Это позволяет поддерживать реплицированную копию данных и обеспечивать их доступность даже при сбое одного или нескольких узлов.ReplacingMergeTree, с другой стороны, обеспечивает механизм автоматической замены записей в случае конфликтов. Это полезно, когда в таблицу поступают данные с обновлениями, и при возникновении дубликатов или конфликтов записи автоматически заменяются новыми значениями. Комбинируя эти два типа движков,обеспечивает репликацию данных и автоматическую замену записей, также, когда требуется хранить и обрабатывать данные с высокой доступностью и устойчивостью к сбоям, и при этом обеспечивать корректность данных и устранение конфликтов при обновлениях. В параметре /clickhouse/tables/bilcdr/cube_new указывается путь к таблице в файловой системе, {replica} указывается шаблон имени реплики.

PARTITION BY toYYYYMM(toDate(h323_setup_time)):используется для определения разбиения данных на разделы (partitions) в таблице на основе значения поля "h323_setup_time".

ORDER BY: ключ уникальности записи данных, используется для определения порядка сортировки данных в таблице. Оно указывает, по каким полям следует упорядочить данные внутри каждой партиции.

SETTINGS index_granularity:это настройка позволяет задать гранулярность индексов. Гранулярность индексов определяет, насколько большими должны быть группы строк в индексе, чтобы он был эффективным. Чем меньше группы строк, тем более точным и быстрым будет поиск по индексу, но тем больше места индекс будет занимать на диске.По умолчанию значение параметра index_granularity равно 8192 строкам.

В PostgreSQL формирование DDL (Data Definition Language) выполняется для создания и изменения схемы базы данных, таблиц, индексов и других объектов данных. DDL позволяет определить структуру данных, их типы, ограничения и другие атрибуты, главное отличие, это то, что вPostgreSQL хранение данных с оборудования осуществляется в отдельные таблицы идля представления и сложных запросов приходится собирать данные с помощью

VIEW используя команду UNION ALL.Что сильно влияет на производительность.

UNION ALL в PostgreSQL — это оператор для объединения результатов двух или более запросов, которые возвращают данные с одинаковой структурой. Он объединяет результаты всех запросов, сохраняя дубликаты строк в отличие от оператора UNION, который удаляет дубликаты.

В PostgreSQL выражение "PARTITION BY RANGE" используется для создания разделов (partitions) в таблице на основе диапазона значений столбца. Каждый раздел содержит ряд значений из столбца, и это позволяет эффективно управлять и обрабатывать большие объемы данных. В примере PostgreSQLразделение таблиц по патрициямпроизводится по дате:
partition by RANGE (h323_setup_time);

BPostgreSQL для формированияDDL используются следующие ключевые слова и конструкции, рисунок 15.

```
create table if not exists old_cdr.gw2
(
    unix_time             integer,
    call_id               integer,
    cdr_type              integer,
    leg_type              integer,
    h323_conf_id          varchar(100),
    peer_address          varchar(50),
    peer_sub_address      varchar(30),
    h323_setup_time       timestamp,
    alert_time            timestamp,
    h323_connect_time     timestamp,
    h323_disconnect_time  timestamp,
    h323_disconnect_cause varchar(255),
    disconnect_text       varchar(2000),
    h323_call_origin      varchar(30),
    charged_units         integer,
    info_type             varchar(30),
    paks_out              integer,
    bytes_out             integer,
    paks_in               integer,
    bytes_in              integer,
    username              varchar(100),
    clid                  varchar(30),
    dnis                  varchar(30),
    column23              varchar(30),
    filename              varchar(255),
    linenum               integer,
    created               timestamp
```

Рисунок 15 – DDLв PostgreSQL

unix_time: поле, содержащее время звонка в формате Unixtime (целое число).

call_id: уникальный идентификатор звонка (целое число).

cdr_type: тип CDR (CallDetailRecord), указывающий на характер вызова (целое число).

leg_type: тип соединения (целое число).

h323_conf_id: идентификатор конференции H.323 (строка длиной до 100 символов).

peer_address: IP-адрес (или доменное имя) удаленного узла (строка длиной до 50 символов).

peer_sub_address: дополнительная информация об удаленном узле, например, номер порта (строка длиной до 30 символов).

h323_setup_time: время установки соединения H.323 (дата и время).

alert_time: время, в течение которого вызываемый абонент слышит звонок перед ответом (дата и время).

h323_connect_time: время установки физического соединения H.323 (дата и время).

h323_disconnect_time: время, когда соединение H.323 было разорвано (дата и время).

h323_disconnect_cause: причина разрыва соединения H.323 (строка длиной до 255 символов).

disconnect_text: дополнительная информация о причине разрыва соединения (строка длиной до 2000 символов).

h323_call_origin: источник вызова H.323 (строка длиной до 30 символов).

charged_units: количество единиц, по которым взимается плата за звонок (целое число).

info_type: тип информации о звонке (строка длиной до 30 символов).

paks_out: количество пакетов (сетевых пакетов), отправленных исходящей стороной (целое число).

bytes_out: количество переданных байтов исходящей стороной (целое число).

paks_in: количество пакетов, полученных входящей стороной (целое число).

bytes_in: количество полученных байтов входящей стороной (целое число).

username: имя пользователя, связанное с вызовом (строка длиной до 100 символов).

clid: номер CallingLineIdentification (идентификация вызывающей линии), который отображается на телефоне вызываемого абонента (строка длиной до 30 символов).

dnis: номер DialedNumberIdentificationService (идентификация набранного номера), который указывает на набранный вызывающим номер (строка длиной до 30 символов).

column23, column24, column25, ..., column34: эти поля представляют дополнительные столбцы, которые могут быть использованы для хранения дополнительной информации или пользовательских данных. Конкретное назначение этих столбцов зависит от конкретной реализации и потребностей проекта (строки с различными размерами, в зависимости от конкретного столбца).

filename: имя файла, в котором содержится запись CDR (строка длиной до 255 символов).

linenum: номер линии или порта, через который был совершен звонок (целое число).

created: время создания записи CDR (дата и время).

Для каждого устройства CUBEсоздается отдельная таблица, рисунок 16 которая называется по его имени gw2, gw3, gw4…gw15. Также создаются отдельные VIEW, рисунок 17, с помощью которых создаются запросы по подсчету биллинга:

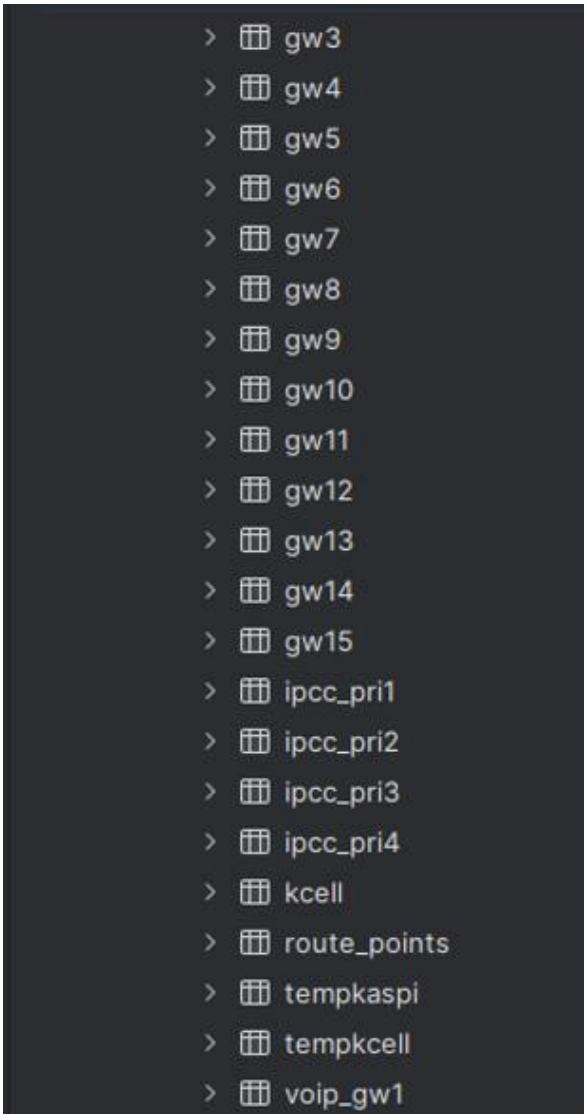

Рисунок 16 - Список всех таблиц в PostgreSQL

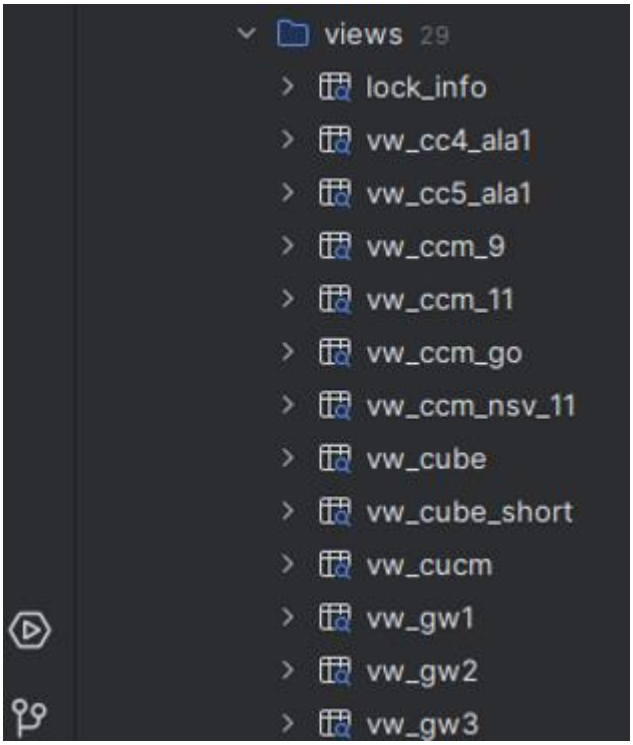

Рисунок 17 - Список всех VIEW

В рамках данного исследования, необходимо протестировать скорость выполнения запроса по определения количества звонков и минут разговора в период времени с 01.03.2023 по 15.03.2023 по исходящим вызовам коллекторского направления, звонящих по префиксам “9000”,“9001”,“9005”, “9006”, “9007”, “8765”,“9012”, “9013”

В ClickHouse получаем результат 7 921307 звонков и 3 574826 минут разговора за период с 01.03.2023 по 15.03.2023. Данный запрос отработал за 25709 миллисекунд (25 секунд), рисунок 18.

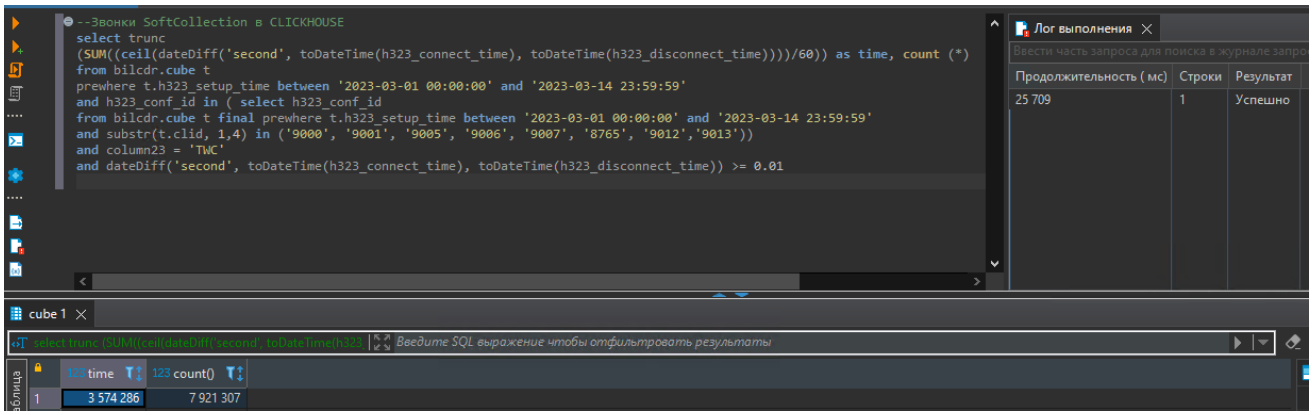

Рисунок 18 – Консоль с результатом запроса по количеству звонков и минут разговора в ClickHouse

В PostgreSQL получаем результат 9 372663 звонков и 3 6598551 минут разговора за период с 01.03.2023 по 15.03.2023.Данный запрос отработал за 292586 миллисекунд (292 секунды), рисунок 19.

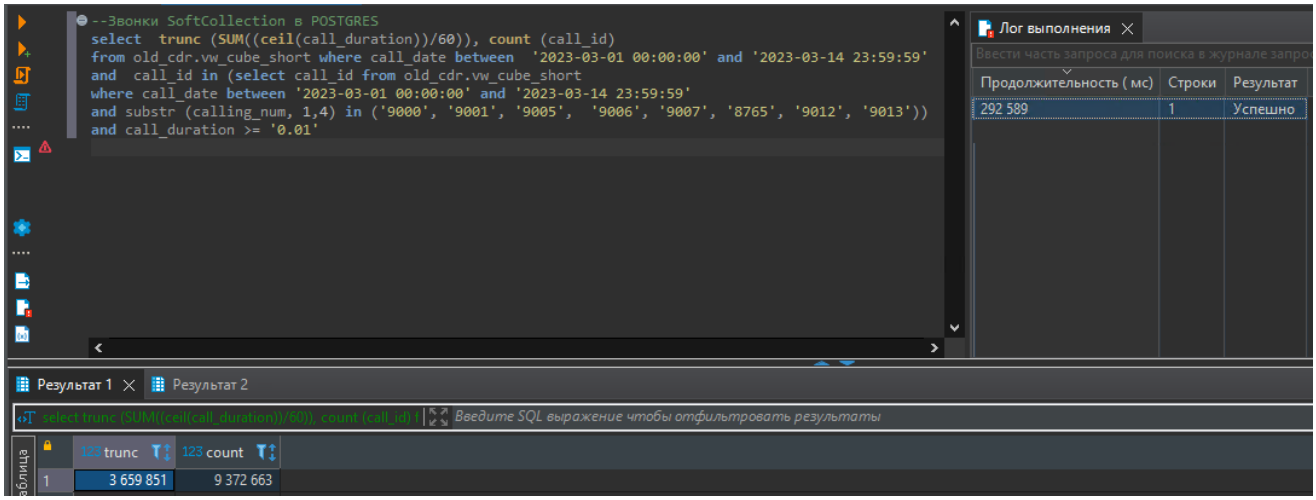

Рисунок 19 – Консоль с результатом запроса по количеству звонков и минут разговора в PostgreSQL

Следует учитывать, что тестирование проводилось в момент минимальной нагрузки на обе тестируемые СУБД. Результат показывает, что время выполнения запроса в ClickHouse 10раз быстрее чем, в PostgreSQL. Также видно, что время разговоров и количество вызов в ClickHouseотличается от данных в PostgreSQL. Это связанно с тем, что в PostgreSQLне учитываются дубли, которые зачастую снижают точность биллинга. При этом избавиться от дублей в PostgreSQLвесьма сложно, т.к. их появление сложно предугадать.В ClickHouseснижение дублей достигается с помощью есть функционала, который расширяет SQL язык, например команды FINAL, PREWHERE позволяют оптимизировать запрос по времени выполнения.

FINAL - это ключевое слово в языке запросов ClickHouse, которое указывает, что запрос должен выполниться непосредственно на данных, а не на виртуальных столбцах или блоках данных, полученных из таблицы. Ключевое слово FINAL используется в различных контекстах запросов, таких как GROUP BY, DISTINCT или ORDER BY, чтобы указать, что операции агрегации или сортировки должны быть выполнены на финальных данных, а не на промежуточных результатах.

PREWHERE — это ключевое слово в ClickHouse, которое позволяет выполнить фильтрацию данных до операции чтения. Оно позволяет уменьшить количество данных, которые будут прочитаны из таблицы, и, следовательно, ускорить выполнение запроса. PREWHERE используется для указания условия фильтрации, которое должно быть выполнено до операции WHERE. Таким образом, оно позволяет отбросить строки, которые не удовлетворяют условию PREWHERE, до чтения данных из таблицы.

Следующий этап исследования — это сравнения объема данных тестируемых СУБД ClickHouseи СУБД PostgreSQL.Для примера, будут взята данные по устройству GW2 за период с 01.03.2023 по 15.03.2023.

Для точности тестирования, необходимо перенести данныеиз таблицы GW2 за период с 01.03.2023 по 15.03.2023 в отдельную таблицуgw2222 и выяснить

количество записей. В PostgreSQL по таблице gw2222 количество записей равно 18 443 139, рисунок 20.

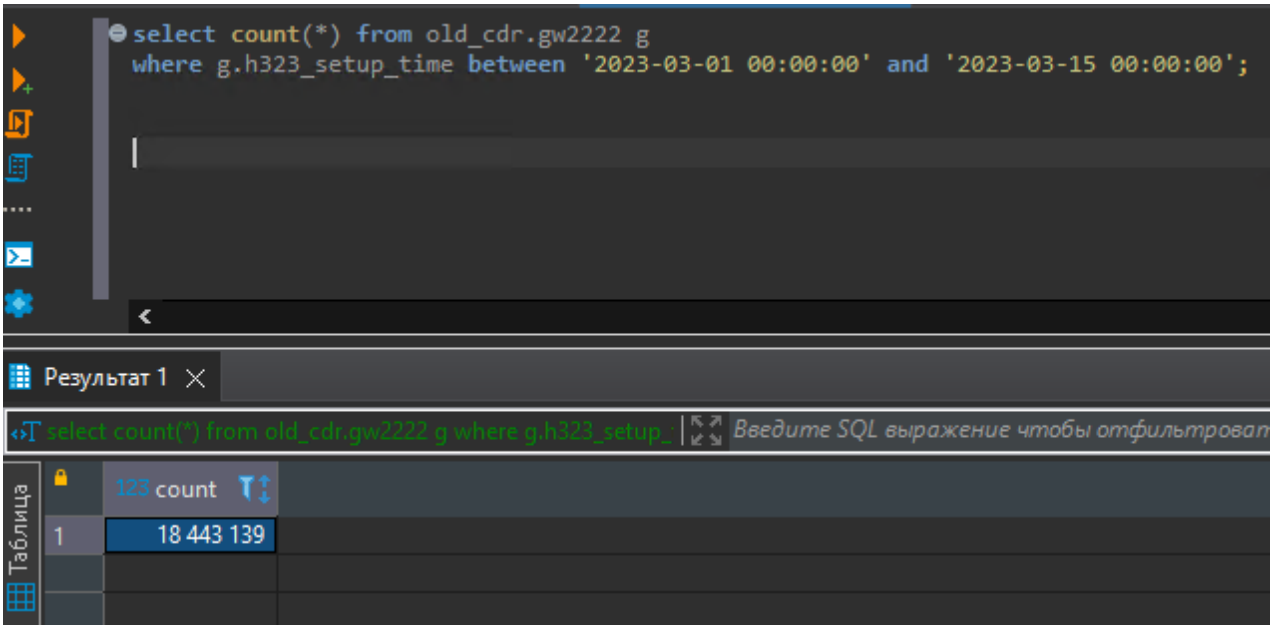

Рисунок 20 – Консоль с результатом по количеству строк за период в PostgreSQL

После этого создаем запрос и выясняем сколько весит таблица gw2222 в схеме oldcdrи получаем результат 10 GB, рисунок 21.

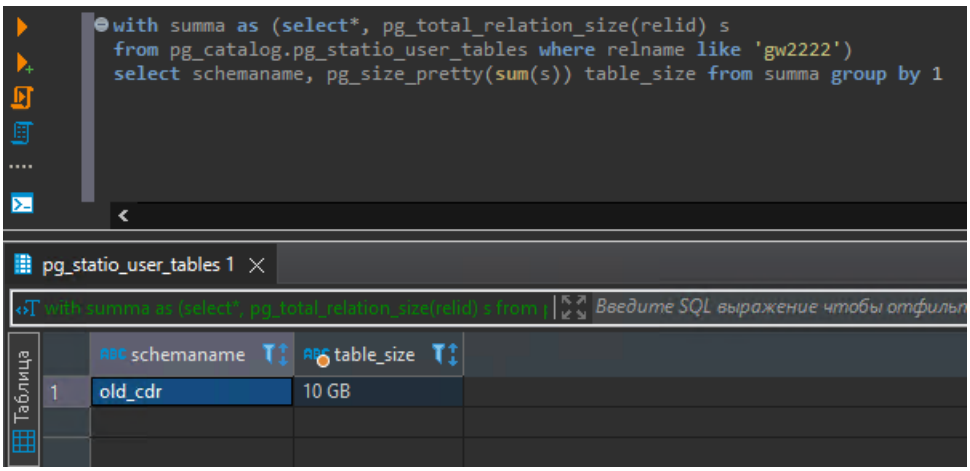

Рисунок 21 – Консоль с результатом по размеру таблицы gw2222 в PostgreSQL

Для ClickHouseаналогично, из общей таблицы ‘cube’ переносим все данные по устройству GW2 за период с 01.03.2023 по 15.03.2023 в отдельную таблицуGW2. В ClickHouseпо таблице GW2 количество записей равно 18 5242 945, рисунок 22.

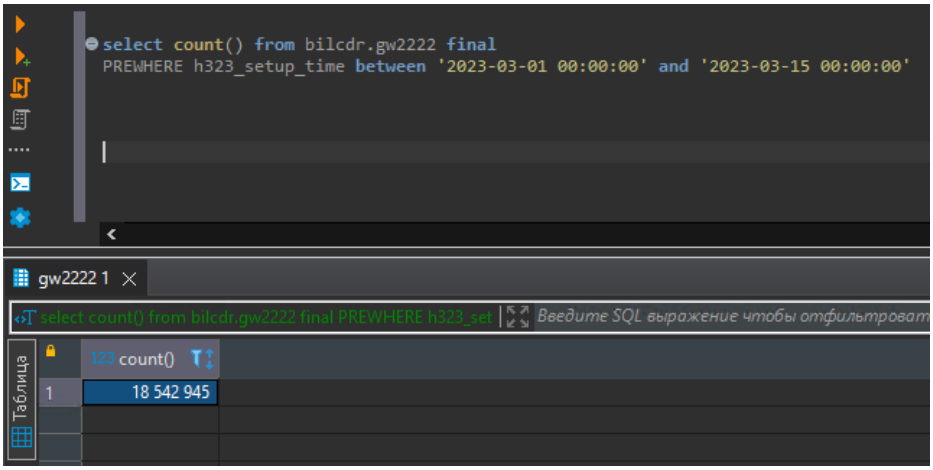

Рисунок 22 – Консоль с результатом по количеству строк за период в ClickHouse

После этого создаем запрос и выясняем сколько весит таблица gw2222 и получаем результат 1.4 GB, рисунок 23.

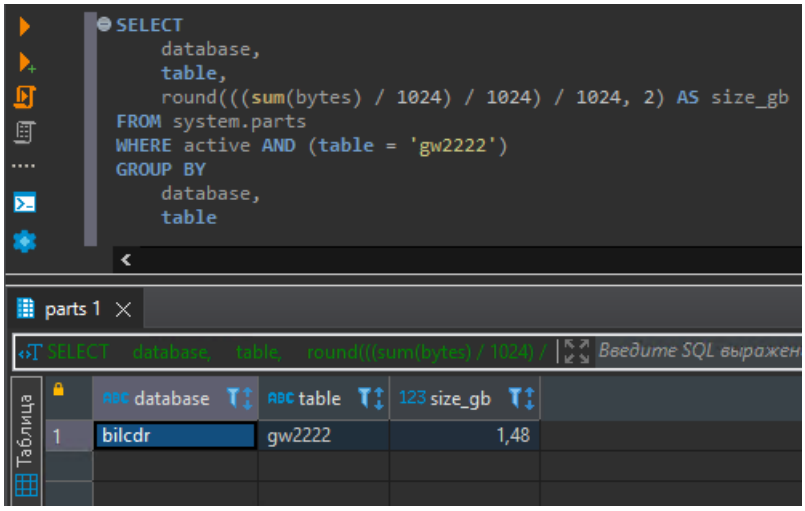

Рисунок 23 – Консоль с результатом по размеру таблицы gw2222 в ClickHouse

Результат тестирования показал, что данные в ClickHouse весят 8 раз меньше по сравнению с СУБДPostgreSQL этосвязанно со следующими причинами:

1) Колоночное хранение данных: ClickHouse использует колоночное хранение данных, что позволяет более эффективно сжимать и хранить информацию. Вместо хранения данных по строкам, как это делают реляционные СУБД, ClickHouse организует данные по столбцам. Это позволяет использовать эффективные алгоритмы сжатия, так как столбцы обычно содержат повторяющиеся значения, что приводит к более компактному хранению.

2) Компрессия данных: ClickHouse имеет встроенную поддержку сжатия данных. При записи данных ClickHouse автоматически применяет сжатие для каждого столбца, используя подходящий алгоритм сжатия, такой

как LZ4 или ZSTD. Это значительно снижает размер данных на диске и уменьшает требования к хранилищу.

3) Оптимизированные структуры индексов: ClickHouse использует специальные структуры индексов, такие как Bloom Filter и Bitmap Index, которые позволяют эффективно фильтровать и искать данные. Эти индексы занимают меньше места по сравнению с традиционными индексами, используемыми в других СУБД.

4) Отсутствие лишних данных и метаданных: ClickHouse сконцентрирован на хранении и обработке аналитических данных, поэтому он может удалять или не хранить некоторые избыточные метаданные или системную информацию, которая не является необходимой для аналитических запросов. Это также способствует снижению общего размера данных.

В результате этих оптимизаций ClickHouse может обеспечить более компактное хранение данных, что позволяет экономить место на диске и улучшает производительность при выполнении аналитических запросов. Также, за счет этих возможностей можно увеличить срок хранения данных. Если, сейчас для хранения данных за полгода в СУБД PostgreSQLтребуется хранилище размеров в 2 ТБ, в ClickHouseэто срок можно увеличить до нескольких лет, при этом учитывая увеличения количества звонков.

OLAPв ClickHouse

Для обработки аналитических запросов в режиме реального времени необходимо создать многомерный куб, при помощи инструмента MicrosoftPowerBI[3]. В связи с тем, что на данный момент отсутствует прямой коннектор подключения MSPowerBI к ClickHouse, потребуется настроить ODBC драйвер, рисунок 24.

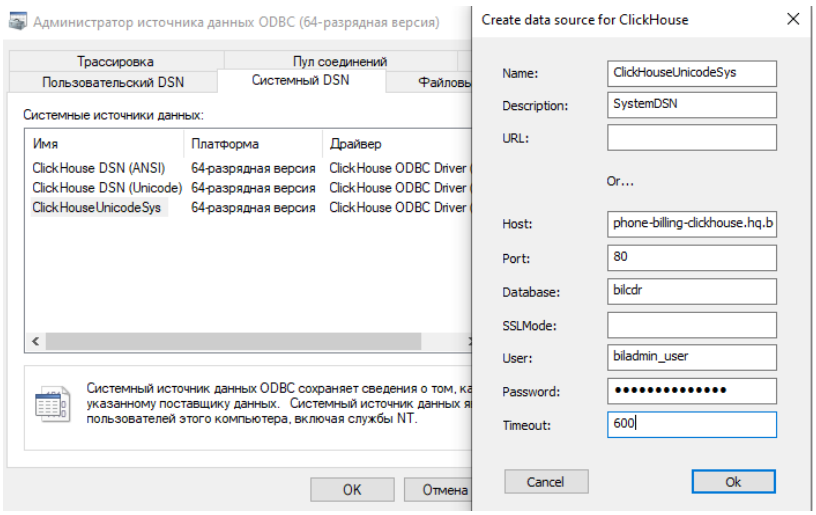

Рисунок 24 – Пример настройки драйвера ODBC дляClickHouse

[3]https://powerbi.microsoft.com/

При подключении MSPowerBIк СУБД ClickHouse,для создания многомерного куба, в первую очередь необходимо указать тело запроса. В данном исследовании выбран запрос по количеству звонков и минут по операторуBeelineдля Удаленного Коллекторского агентства, рисунок 25.

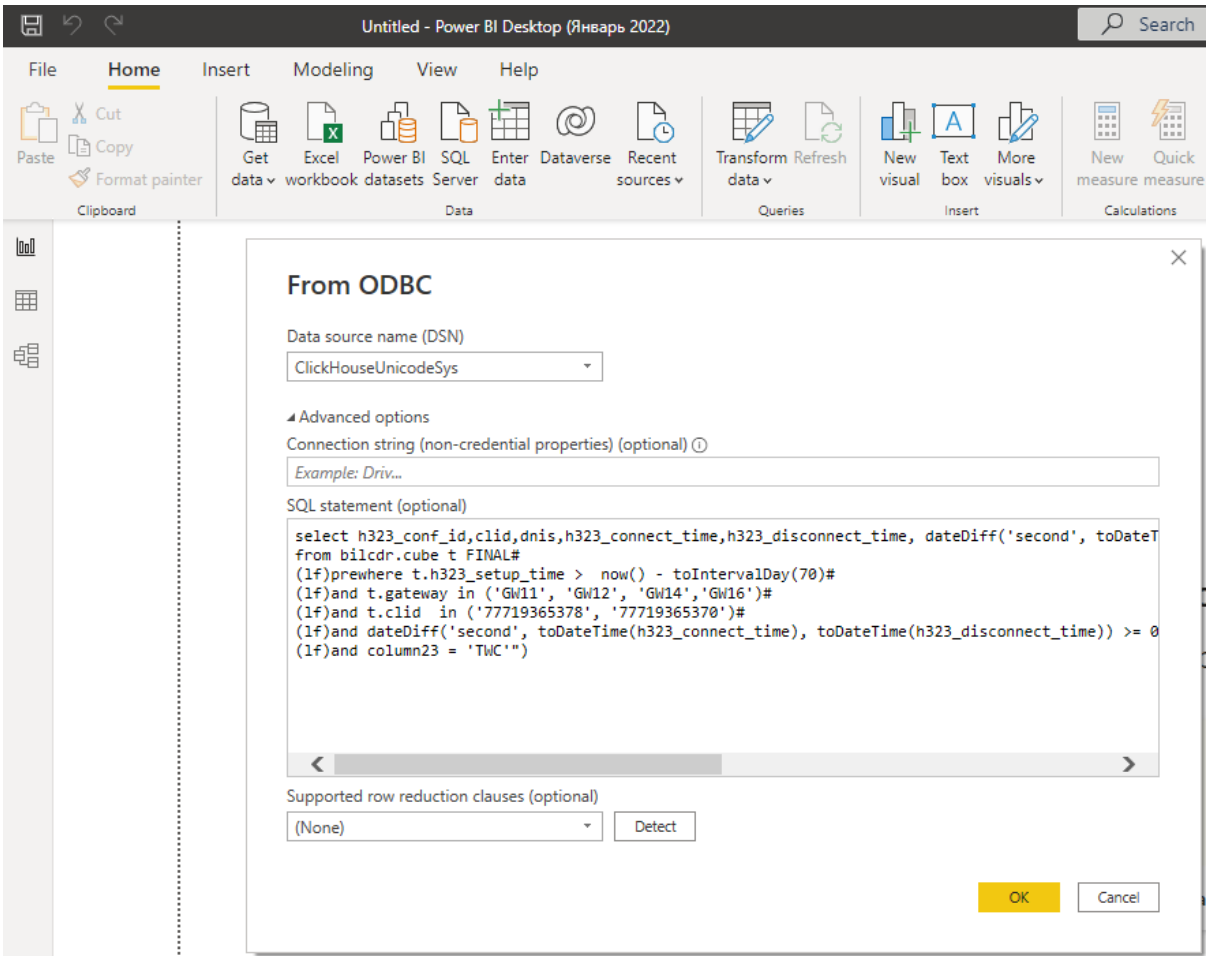

Рисунок 25 – Создание запроса для многомерного куба

После настройки ODBC,авторизации подключения MSPowerBIк базе данных, получаем данные, с которые необходимо обработать, рисунок 26.

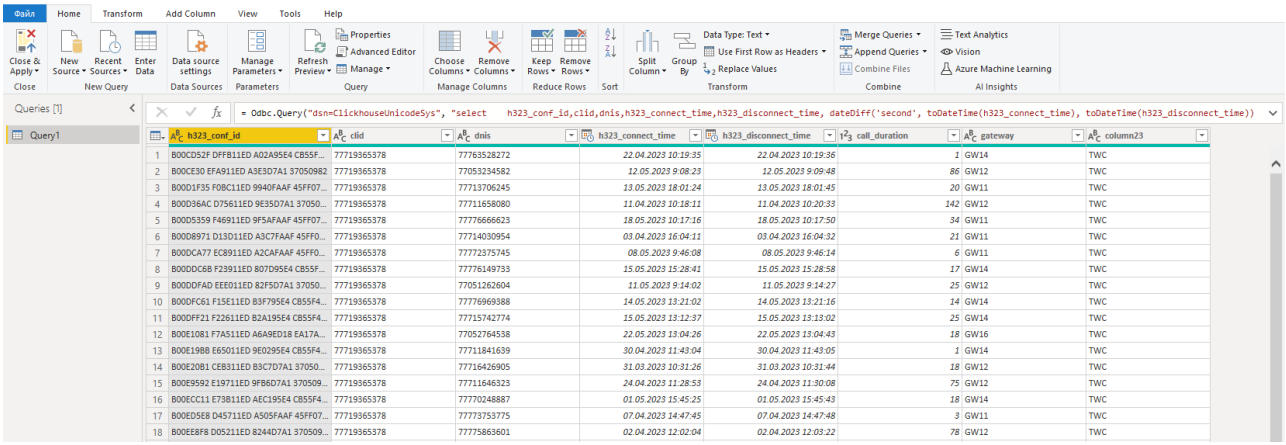

Рисунок 26 – Данные по звонкам оператора Beeline Удаленного Коллекторского агентства

Далее настраиваем необходимые поля, в первую очередь это добавления поля «dur_minut», данное поле необходимо для округления секунд до минут, в связи с тем, что биллинг от операторов приходит в минутах, рисунок 27 и задаем этой колонке числовой формат, рисунок 28.

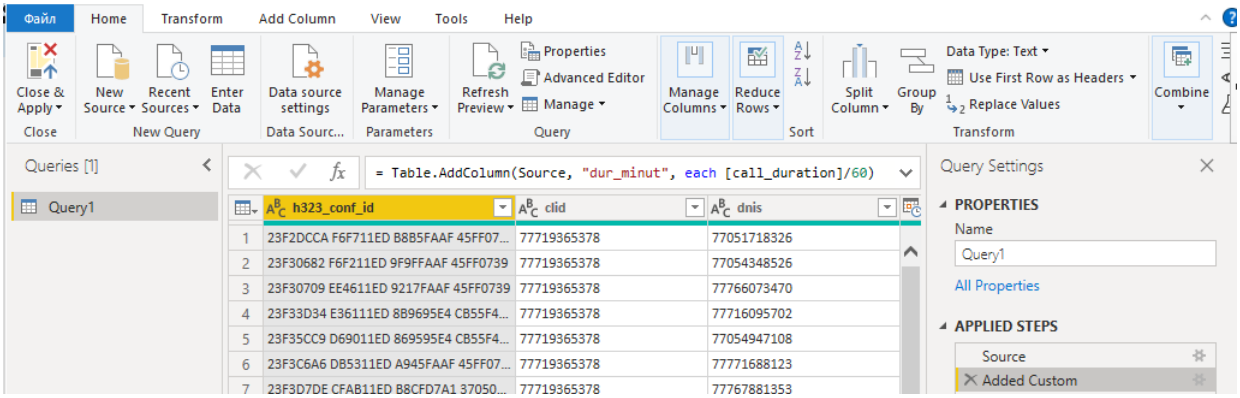

Рисунок 27 – Добавление колонки “dur_minut”

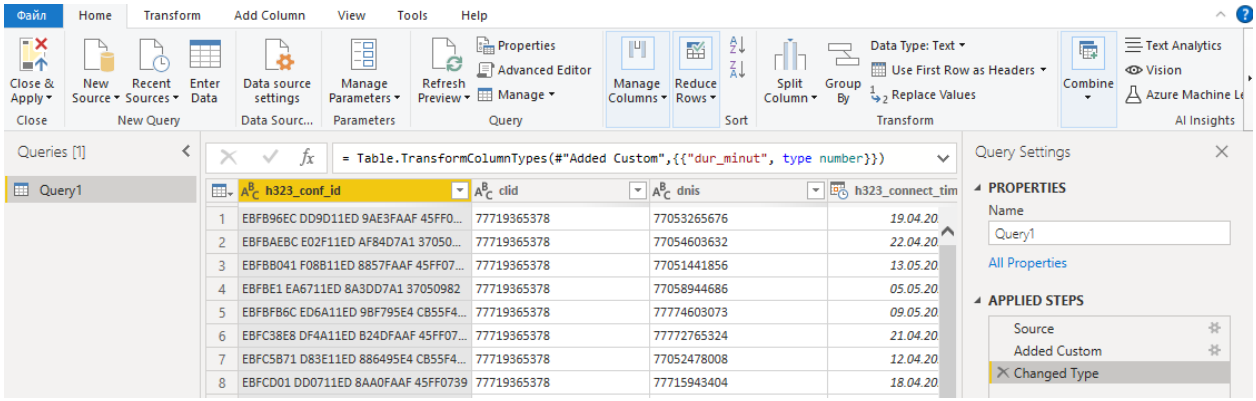

Рисунок 28 – Настройка колонки “dur_minut”, числовой формат.

Заключительный этап, создаем многомерный куб. Для общей таблицы по звонкам выбираем необходимые столбцы с временем по секундам и минутам, номерам абонентов, времени соединения и разъединения и дате, рисунок 29.

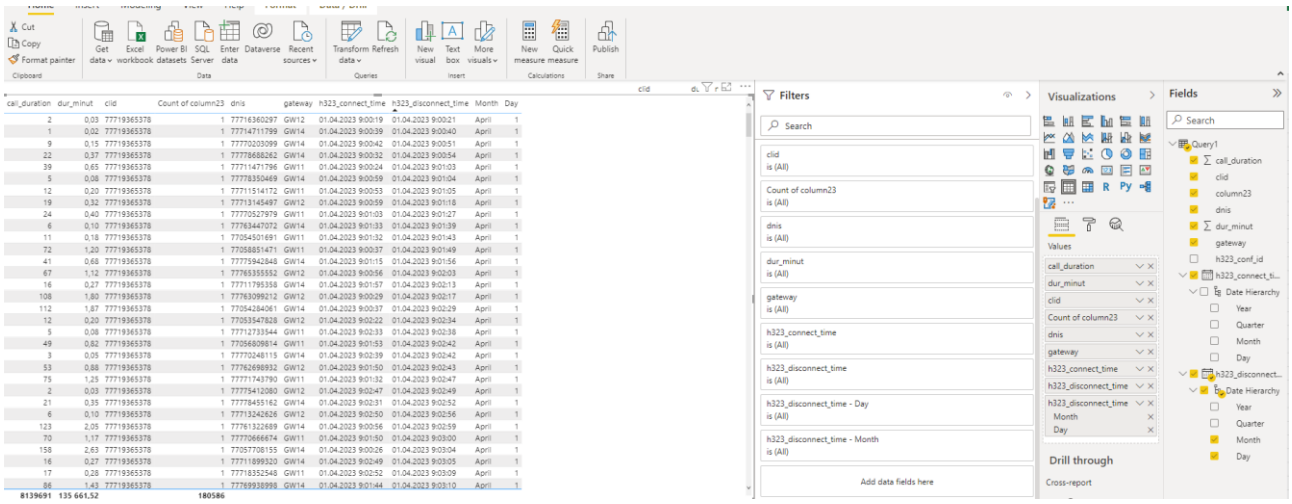

Рисунок 29 – Создание представления по звонкам

Создаем график по количеству звонков в день, на вертикальное оси время соединения, в значениях колонку, определяющую сам звонок, рисунок 30.

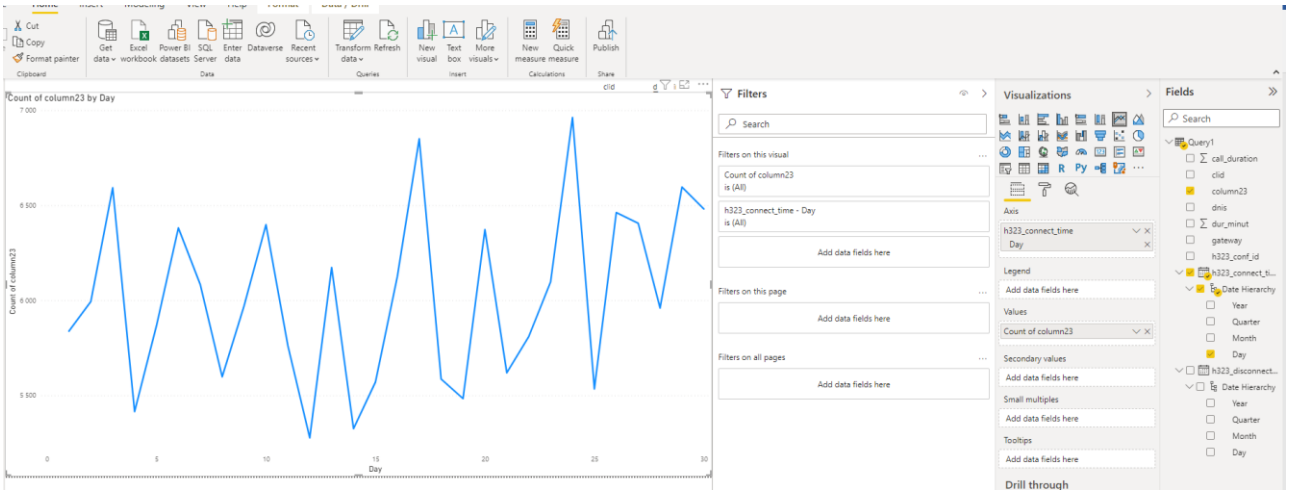

Рисунок 30 – График по количеству звонок в день

Также добавляем в эту визуализацию фильтр по времени и фильтр по итоговым данным по количеству звонков и количеству минут, рисунок 31.

Рисунок 31– Фильтр по времени и экраны с итоговыми данными по количеству звонков и количеству минут

В итоге получаем представление, которое позволят посчитывать биллинг в режиме реального времени по исходящим звонкам на операторе мобильной связи Beeline, рисунок 32.

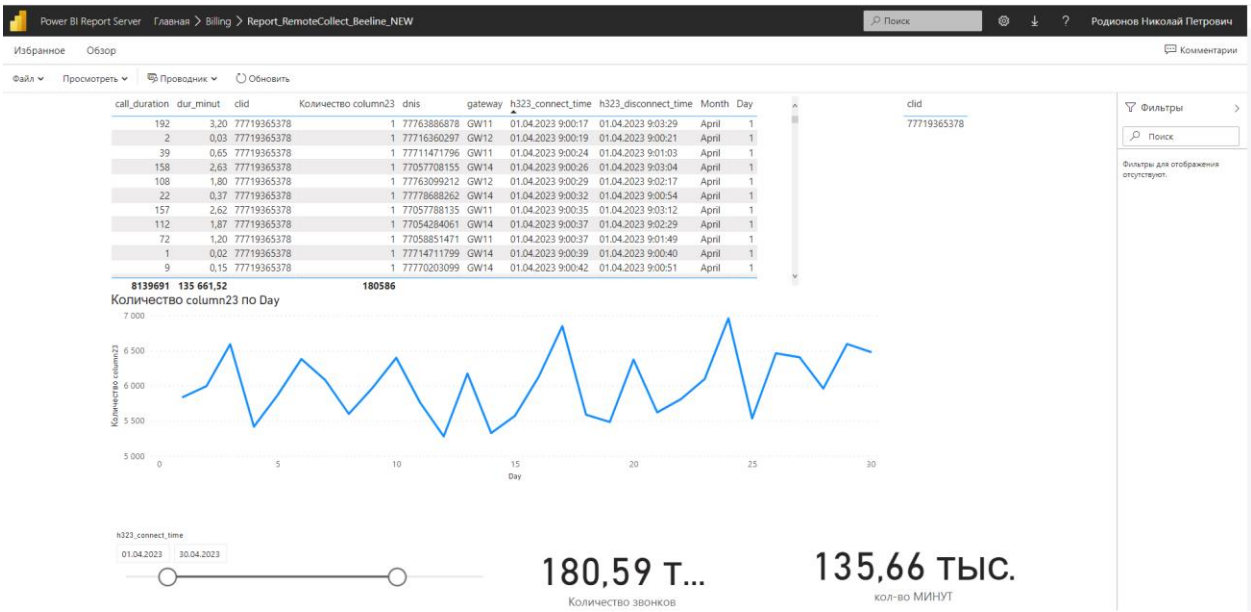

Рисунок 32 – Визуализация данных по подсчету биллинга по оператору Beeline

ЗАКЛЮЧЕНИЕ

В результате исследования было установлено, что ClickHouse, как колоночная СУБД, обладает значительными преимуществами при работе с большими объемами данных и аналитическими запросами. Благодаря своей колоночной структуре хранения данных, ClickHouse демонстрирует высокую производительность и эффективность при агрегировании и анализе данных. Он может эффективно использовать оптимальные параметры для хранения данных и предоставляет быстрый доступ к информации. PostgreSQL также имеет хорошую поддержку масштабирования и репликации, что делает его предпочтительным выбором для проектов, где важны надежность и согласованность данных.Однако при сравнении производительности и скорости выполнения аналитических запросов ClickHouse показал лучшие результаты. Он обрабатывает запросы на миллионы строк данных за секунды благодаря оптимизированной структуре хранения и параллельной обработке запросов. PostgreSQL, в свою очередь, хоть и поддерживает аналитические запросы, может быть менее производительным при работе с большими объемами данных.Выбор между ClickHouse и PostgreSQL зависит от конкретных потребностей проекта. Если требуется обработка больших объемов данных и аналитический аспект является ключевым, то ClickHouse представляет собой более подходящее решение. Однако, если важны транзакционные данные и надежность, PostgreSQL остается привлекательным выбором.В целом, выбор между ClickHouse и PostgreSQL зависит от конкретных потребностей проекта. Если требуется обработка больших объемов данных и аналитический аспект является ключевым, то ClickHouse представляет собой более подходящее решение. Однако, если важны транзакционные данные и надежность, PostgreSQL остается привлекательным выбором.Тем не менее, следует отметить, что обе системы имеют свои ограничения. ClickHouse, например, не поддерживает полный набор функций транзакций, что может быть недостатком для определенных приложений, требующих строгой согласованности данных. PostgreSQL, с другой стороны, может быть менее производительным при работе с большими объемами данных и аналитическими запросами, поскольку его реляционная модель данных не так эффективна для агрегации и анализа.

В данной диссертации было проведено сравнение двух систем управления базами данных - ClickHouse и PostgreSQL - с целью определить их отличия, преимущества и недостатки при обработке больших объемов данных и аналитической обработке. Исследование было выполнено на примере использования ClickHouse в рамках задач, связанных с анализом данных и хранением информации в Контакт Центре АО "Kaspi Bank". Результаты и выводыполученные в ходе исследования, позволяют сделать следующие заключения:

1. *Анализ возможностей колоночной базы данных*: в результате исследования было установлено, что ClickHouse является мощной системой для

аналитической обработки данных. Его колоночная структура хранения позволяет эффективно агрегировать и анализировать большие объемы информации. ClickHouse демонстрирует высокую скорость работы и чтения данных, что является важным преимуществом при выполнении сложных аналитических запросов.

2. *Создание многомерной модели данных*: в рамках диссертации была разработана многомерная модель данных на примере Базы Данных Контакт Центра АО "Kaspi Bank". ClickHouse обеспечивает возможность эффективной работы с такой моделью данных, позволяя выполнять сложные аналитические запросы и проводить анализ информации по различным измерениям.

3. *Исследование способов повышения скорости выполнения аналитических запросов*: было проведено исследование и определены способы оптимизации выполнения аналитических запросов в ClickHouse. Применение правильных индексов, использование оптимальных параметров конфигурации и оптимизация структуры таблиц позволили значительно повысить скорость выполнения запросов и сократить время отклика системы. Тестирование показало, что скорость выполнения запросов в СУБД ClickHouseв 10 раз быстрее, чем СУБД PostgreSQL.

4. *Увеличение объема данных*: Тестирование показало, что в СУБД ClickHouseобъем данных в8 раз меньше, чем СУБД PostgreSQL. Что позволяет увеличить срок хранения в 4 раза с учетом увеличения звонковой активности.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Surajit Chaudhuri, Umeshwar Dayal. “An overview of data warehousing and OLAP technology”. ACM SIGMOD Record Vol. 26 Issue 1,pp 65–74, March 1997.

2 Статья: "Большие данные (Big Data)" на TAdviser.ru.

3 Kimball R., Ross M. "The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling". Wiley, 3rdedition, July 2013.

4 Pedersen T.B., Jensen C., Thomsen C. "Multidimensional Databases and Data Warehousing". Springer Cham, September 2010.

5 Moss L., Atre S. "Business Intelligence Roadmap: The Complete Project Lifecycle for Decision-Support Applications". Addison-Wesley Professional, 1st edition, January 2003.

6 Turban E., Delen D., Sharda R. "Business Intelligence: A Managerial Perspective on Analytics". Pearson, 3rd edition, December 2019.

7 Han J., Kamber M., Pei J. "Data Mining: Concepts and Techniques". Morgan Kaufman, 3rd edition,June 2011.

8 Provost F., Fawcett T. "Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking". O`Reilly Media, 1st edition, September 2013.

9 Chatfield C., Xing H. "The Analysis of Time Series: An Introduction". Chapman and Hall/CRC, 7th edition, May 2019.

10 Hyndman R.J., Athanasopoulos G. "Forecasting: Principles and Practice". OTexts, 3rd edition, May 2021.

11 Longley P., Goodchild M., Maguire D., Rhind D. "Geographic Information Science and Systems". Wiley, 4th edition, February 2015.

12 O'Sullivan D., Unwin D. "Geographic Information Analysis". Wiley, 2nd edition, March 2010.

13 Hastie T., Tibshirani R., & Friedman J. "The Elements of Statistical Learning: Data Mining, Inference, and Prediction". Springer, 2nd edition, January 2016.

14 Bishop C. M. "Pattern Recognition and Machine Learning".Springer, August 2006.

15 Haseeb Abdul, Pattun Geeta.“A review on NoSQL: Applications and challenges” International Journal of Advanced Research in Computer Science., Vol. 8 Issue 1, 203-207, January 2017.

16 Chodorow K., Dirolf M. "MongoDB: The Definitive Guide". O`Reilly Media, 3rd edition, December 2019.

17 Radcliffe D. "NoSQL for Dummies". For Dummies, 1s edition, February 2015.

18 Joshi H. "Redis Cookbook: Practical Techniques for Fast Data Manipulation" O`Reilly Media, 1st edition, August 2011.

19 Brown E. "Riak Handbook". Wiley, Juny 2014.

20 Webber J., Robinson I.,Eifrem E. "Graph Databases: New Opportunities for Connected Data".O`Reilly Media, 2nd edition, July 2015.

21 Smarandache F., Broumi S. "Advances in Data Mining and Database Managment". Engineering Science Reference, October 2019.

22 Lakshman A., Malik P. "Cassandra: A Decentralized Structured Storage System" ACM SIGOPS Operating Systems Review, vol. 44, No. 2, April 2010.

23 George L., Mistretta R. "HBase: The Definitive Guide", O`Reilly Media, 1st edition, October 2011.

24 Sivasubramanian S."Amazon DynamoDB: A Distributed and Highly Available Key-Value Store". ACM SIGOPS Operating Systems Review, vol. 41, No. 6, Issuet 6, pp 205-220,December 2007.

25 Vogels W."Amazon DynamoDB: a seamlessly scalable non-relational database service" ACM SUGMOD International Conference on Management of Data, pp 729-730, May 2012.

26 White T. "Hadoop: The Definitive Guide". Yahoo Press, 3rd edition, July 2015.

27 Zaharia M., Wendell P., Konwinski A., Karau H. "Learning Spark: Lightning-Fast Big Data Analysis". O`Reilly Media, 1st edition, February 2015.

28 Malik P., Lakshman A. "Cassandra: The Definitive Guide". O`Reilly Media, 2nd edition, July 2016.

29 Shkarupin V. "Mastering Apache HBase: Advanced Techniques for High-Performance Hadoop Data Storage".O`Reilly Media, 2nd edition, April2017.

30 Dan Sullivan "NoSQL for Mere Mortals". Addison-Wesley Professional. 1sr edition, April 2015.

31 Cuzzocrea A., Song I.-Y., Davis K. C. "Data Warehousing and Mining: Concepts, Methodologies, Tools, and Applications". Information Science Reference, 2nd edition,May 2012.

32 Mohan C., Haderle D., Lindsay B., Pirahesh H. "Database Management Systems". McGraw-Hill, 2003.

33 Sadalage P. J., Fowler M. "NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence". Addison-Wesley Professional, 1st edition, August 2012.

34 Lakshman A., Malik P. "Cassandra: The Definitive Guide: Distributed Data at Web Scale", O`Reilly Media, 2nd edition, July 2016.

35 Christopher Adamson and Christopher Ilacqua"Star Schema: The Complete Reference". McGraw-Hill, 2010.

36 Gray J., Watson A. “Decision Support in the Data Warehouse”. Pearson, 1st edition, January 2008.

37 Gray J., Bosworth A. “Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals” Data Mining and Knowledge Discovery, 29-53,1997.

38 Russell Bradberry and Eric Lubow. "Practical Cassandra: A Developer's Approach" Addison-Wesley Professional,1st edition, December 2013.

39 Codd, E. F. “Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate”. Technical Report, E. F. Codd & Associates, 1993.

40 МатвеевА. В., ОнищенкоМ. И., КлейменовД. А. “ClickHouse: колоночнаяСУБДдляаналитикибольшихданных. Высокая производительность и открытый исходный код”. В сборнике "Информационно-аналитические системы в науке, образовании и производстве", 2020.

41 Зайцев А. “ClickHouse для аналитики”. Издательство "Питер", 2020.

42 Yandex. ClickHouse Documentationнасайте Clickhouse.com/docs

43 Farouk R., El-Sayed, H., El-Kassas S., El-Den D. “Building OLAP cubes: challenges and approaches”. Journal of Database Management, 31, pp 38-54, 2020.

44 Jensen, C. S., Pedersen T. B., Dyreson, C. E. “Multidimensional database technology”. ACM Computing Surveys, 33, pp 292-326, 2001.

45 Mundy J., Ross M., Kimball R., Thornthwaite W. “The data warehouse toolkit: practical techniques for building dimensional data warehouses”. John Wiley & Sons, 1998.

46 Gray J., Chaudhuri S., Bosworth A., Layman A., Reichart D., Venkatrao M., Pellow F. “Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals”. Data Mining and Knowledge Discovery, 29-53, 1997.

47 Zolotov D., Tkachenko V., Dolgov A., & Vorobev A. “ClickHouse: High-performance column-oriented database”. In 2nd International Workshop on Click-Based Interactive Recommendation and Advertising, 61-68, 2016.

48 Özcan F., Tatbul N., & Zdonik S. “Evaluating the performance of real-time stream processing platforms”.Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, 783-794, 2011.

49 Pavlo A., Angulo D., Arulraj J., Lin H., Lin J., Ma M., Zhan J. “Self-driving database management systems”. Proceedings of the 2017 ACM International Conference on Management of Data, 197-212, 2017.

50 Enrico Pirozzi, Gabriel Hautclocq, and Gregory Smith. "PostgreSQL 10 High Performance". Packt Publishing, April 2018.

ПриложениеА

```
- index: 0
  name: unix_time
- index: 1
  name: call_id
- index: 2
  name: cdr_type
- index: 3
  name: leg_type
- index: 4
  name: h323_conf_id
- index: 5
  name: peer_address
- index: 6
  name: peer_sub_address
- index: 7
  name: h323_setup_time
  column:
    type: timestamp
    format: "15:04:05.000 MST Mon Jan _2 2006"
- index: 8
  name: alert_time
  column:
    type: timestamp
    format: "15:04:05.000 MST Mon Jan _2 2006"
- index: 9
  name: h323_connect_time
  column:
    type: timestamp
    format: "15:04:05.000 MST Mon Jan _2 2006"
- index: 10
  name: h323_disconnect_time
  column:
    type: timestamp
    format: "15:04:05.000 MST Mon Jan _2 2006"
- index: 11
  name: h323_disconnect_cause
- index: 12
  name: disconnect_text
- index: 13
```

  name: h323_call_origin
- index: 14
  name: charged_units
- index: 15
  name: info_type
- index: 16
  name: paks_out
- index: 17
  name: bytes_out
- index: 18
  name: paks_in
- index: 19
  name: bytes_in
- index: 20
  name: username
- index: 21
  name: clid
- index: 22
  name: dnis
- index: 23
  name: column23
- index: 24
  name: column24
- index: 25
  name: column25
- index: 26
  name: column26
- index: 27
  name: column27
- index: 28
  name: column28
- index: 29
  name: column29
- index: 30
  name: column30
- index: 31
  name: column31
- index: 32
  name: column32
- index: 33
  name: column33
- index: 34
  name: column34

ПриложениеБ

```
create table if not exists bilcdr.cube_new
(
    unix_time            Int32,
    call_id              Int32,
    cdr_type             Int32,
    leg_type             Int32,
    h323_conf_id         String,
    peer_address         String,
    peer_sub_address     String,
    h323_setup_time      DateTime64(3),
    alert_time           DateTime64(3),
    h323_connect_time    DateTime64(3),
    h323_disconnect_time DateTime64(3),
    h323_disconnect_cause String,
    disconnect_text      String,
    h323_call_origin     String,
    charged_units        Int32,
    info_type            String,
    paks_out             Int32,
    bytes_out            Int32,
    paks_in              Int32,
    bytes_in             Int32,
    username             String,
    clid                 String,
    dnis                 String,
    column23             String,
```

```
    column24            String,
    column25            String,
    column26            String,
    column27            String,
    column28            String,
    column29            String,
    column30            String,
    column31            String,
    column32            String,
    column33            String,
    column34            String,
    filename            String,
    gateway             String,
    linenum             Int32,
    created             DateTime64(3),
    source              String,
    create_file         String
)
    engine = ReplicatedReplacingMergeTree('/clickhouse/tables/bilcdr/cube_new',
'{replica}')
        PARTITION BY toYYYYMM(toDate(h323_setup_time))
        ORDER BY (h323_conf_id, h323_setup_time, h323_connect_time,
h323_disconnect_time, clid, dnis, column23,
filename)
        SETTINGS index_granularity = 8192;
```

ПриложениеВ

```
create table if not exists old_cdr.gw2
(
    unix_time            integer,
    call_id              integer,
    cdr_type             integer,
    leg_type             integer,
    h323_conf_id         varchar(100),
    peer_address         varchar(50),
    peer_sub_address     varchar(30),
    h323_setup_time      timestamp,
    alert_time           timestamp,
    h323_connect_time    timestamp,
    h323_disconnect_time  timestamp,
    h323_disconnect_cause varchar(255),
    disconnect_text      varchar(2000),
    h323_call_origin     varchar(30),
    charged_units        integer,
    info_type            varchar(30),
    paks_out             integer,
    bytes_out            integer,
    paks_in              integer,
    bytes_in             integer,
    username             varchar(100),
    clid                 varchar(30),
    dnis                 varchar(30),
    column23             varchar(30),
```

```
column24        varchar(30),
column25        varchar(30),
column26        varchar(50),
column27        varchar(50),
column28        varchar(50),
column29        varchar(100),
column30        varchar(50),
column31        varchar(50),
column32        varchar(50),
column33        varchar(50),
column34        varchar(50),
filename        varchar(255),
linenum         integer,
created         timestamp
```