Санкт-Петербургский Государственный Университет
Математико-механический факультет

Программная инженерия
Кафедра системного программирования

Шабанов Владимир Сергеевич

Разработка расширяемой системы обработки и анализа разнородных данных

Выпускная квалификационная работа

Научный руководитель:
к.ф.-м.н. ст. преп. Луцив Д. В.

Рецензент:
программист компании ООО «Девяносто один» Карпов Д. Н.

Санкт-Петербург
2019

SAINT-PETERSBURG STATE UNIVERSITY

Software engineering

Vladimir Shabanov

Extensible heterogeneous data processing and analyzing system development

Graduation Thesis

Scientific supervisor:
Ph.D. sen. lecturer Luciv D. V.

Reviewer:
Lead Developer «NineOne» Karpov D. N.

Saint-Petersburg
2019

Оглавление

Введение

К настоящему моменту в больших организациях скопились десятки терабайт разнообразных данных, например сведения о государственных закупках. С одной стороны очевидно, что из такого количества данных можно извлечь полезные сведения. Но с другой стороны встает проблема анализа таких объемов данных за приемлемое время. Ведь для каждого конкретного случая интерес представляет лишь небольшое подмножество накопленных данных, которое нужно быстро выделить и предоставить для анализа человеку. Для принятия решений человеку нужно понимать общую картину по конкретному вопросу, поэтому инструмент должен не только быстро находить все запрошенные данные, но и агрегировать их для удобства восприятия. Это привело к развитию различных способов их обработки и использования. Одна из таких технологий — OLAP (Online Analytical Processing)[2].

Отличительные особенности OLAP—сценария работы это: частые запросы на чтение данных, редкие запросы на добавление данных, запросы на чтения затрагивают большое количество строк и нестрогие требования к согласованности данных. Такой сценарий встречается во многих системах анализа накопленных данных, когда не так важна актуальность данных, как возможность оперативно ознакомится с исторической статистикой. Это обычная ситуация в компаниях, когда менеджер хочет ознакомится с отчетностью за последний год/квартал/-месяц.

Целевая аудитория OLAP—продуктов — менеджеры различного уровня, ответственные за принятие управленческих решений и професиональные аналитики, работающие с большими объемами информации. Благодаря OLAP—инструментам, специалист может получить результаты интересующего его агрегирования большого количества данных с минимальной задержкой. Что увеличивает скорость анализа, принятия решений и привлекательность системы для конечных пользователей.

В Ленинградской области планируется запустить новый аналитический центр, который будет собирать данные из различных источников

в области, например: система 112, единая информационная система в сфере государственных закупок. Но существующие BI—решения не удовлетворяют требованиям области. В этих решениях недостаточно агрегационных функций и слабый функционал фильтров. Таким образом появилась необходимость разработать новую аналитическую систему.

1. Постановка задачи

В рамках данной выпускной квалификационной работы будет разработана новая масштабируемая и стандартизированная OLAP—система анализа данных. Для достижения этой цели были поставлены следующие задачи.

- Провести обзор существующих решений.
- Разработать требования к системе анализа данных.
- Разработать архитектуру системы анализа данных.
- Реализовать подсистемы загрузки, обработки и анализа данных пользователя.

2. Обзор предметной области

2.1. Область применения OLAP—систем

Анализ деятельности организации — важный этап при определении ее эффективности и планировании следующих действий. Такого рода анализ может быть проведен специалистом, который знает, что и как нужно искать, но в простейших случаях может справиться и рядовой менеджер. Но и тому и другому нужен инструмент, который будет аккумулировать полезные данные, и который будет способен быстро предоставить ответы на запросы пользователя. Именно для такого сценария работы был пересмотрен классический OLTP подход к построению архитектуры БД. В общем виде этот сценарий работы можно описать следующими особенностями.

- Данные поступают большими пачками, часто используется автоматизированная загрузка.
- Скорость загрузки данных в систему не так важна, как скорость доступа к данным.
- Абсолютное большинство запросов на чтение данных.
- Запросы на чтение часто затрагивают большое количество строк.
- На запрашиваемых данных активно используются фильтры и агрегации.
- Данные, уже находящиеся в системе не изменяются, или делают это очень редко.

2.2. Описание работы OLAP—систем

Основной особенностью OLAP—систем является способ организации таблиц в базе данных. Используется, так называемая, схема «звезды». В таком представлении выделяется центральная таблица — таблица

фактов (содержащая меры) и таблицы измерений. Для описания структуры такой базы данных используется абстракция OLAP—куб. Ребрами, которого можно представить измерения, а ячейками — меры. Тогда процесс создания аналитического запроса будет состоять из следующих этапов.

1. Пользователь выбирает интересующие его измерения для анализа.
2. Пользователь устанавливает границы и фильтры для этих измерений.
3. Система выбирает и агрегирует все подходящие под условия меры.
4. Система предоставляет пользователю данные в удобном для него формате.

По способу расположения данных при хранении, СУБД можно разделить на строковые и столбцовые. В строковых СУБД значения, относящиеся к одной строке, физически хранятся рядом, поэтому чтение из таких баз возможно только по строчно. Такие системы спроектированы, чтобы эффективно читать строки целиком. Например, когда мы хотим получить всю доступную информацию о человеке в справочнике работников. Примеры строковых СУБД: MySQL, PostgreSQL, MS SQL Server. В столбцовых СУБД значения из разных столбцов хранятся отдельно, а данные одного столбца — вместе. Это позволяет быстрее считывать значения одного или нескольких столбцов для строчек таблицы, потому что нет необходимости считывать строки целиком. В такой схеме хранения операции затрагивающие строки целиком будут медленнее. Однако, такие операции встречается редко в OLAP—системах. В большинстве случаев требуется только небольшое подмножество столбцов. Например, получить информацию о зарплате всех сотрудников, чтобы посчитать среднею зарплату в справочнике сотрудников. Примеры столбцовых СУБД: Vertica, ClickHouse, InfiniDB, MonetDB, Hive. Из—за особенностей структуры базы, таблицы в OLAP—системах получаются очень «широкими», т.е. имеют много столбцов, но аналитиков в каждом

конкретном случае интересует лишь небольшое подмножество столбцов. Поэтому максимальную эффективность здесь показывают столбцовые СУБД.

2.3. Существующие решения

2.3.1. QlikView

Интерфейс слишком упрощен, что затрудняет анализ, т.к. нужные функции часто далеко запрятаны. Есть возможность производить анализ на основе формул, а не только исходных данных, но язык формул не содержит условных выражений, что делает его довольно примитивным. Уникальная функция QlickView — автоматически обнаруживать связи в таблицах из разных источниках данных. После серии экспериментов, становится понятно, что это функция во многом полагается на название колонок в таблицах [4]

2.3.2. Tableau

Популярное решение в сфере бизнесс аналитики. Имеет приятный и удобный интерфейс, встроенные возможности по интеграции с множеством баз данных. Но нет возможности настраивать сложную фильтрацию данных без предварительной обработки. Имеются функции для совместной работы над отчетами. Богатые возможности визуализации.[5]

2.3.3. Power BI

Входит в систему Microsoft SQL. Не используется с другими базами данных, разрабатывается только под ОС семейства Windows. Имеется мобильная версия.

	Tableau	Power BI	QlikView
Фильтры и поиск по измерениям	+	+	+
OLAP функциональность	+	+	+
Кроссплатформенность	+	−	+
Мониторинг состояния системы	+	+	−
Мобильный формат отчетов	−	+	+
Операции над представлениями	−	+	+
Открытый исходный код	−	−	−

Таблица 1: Сравнение BI решений

3. Разработка требований

Требования к системе разрабатывались, учитывая пожелания заказчика, основные сценарии использования аналитиками и лучшие практики проанализированных решений. В процессе анализа рабочего окружения заказчика было установлено, что основными источниками данных будут служить базы данных PostgreSQL, MySQL и файлы формата XLSX, CSV. Были проработаны требования для следующих основных подсистем.

1. Подсистема интеграции, информационного взаимодействия и загрузки данных.

    - Импорт данных из файлов формата: JSON, XLSX, XLS, XML, CSV.
    - Импорт данных из внешних СУБД: Oracle, MS SQL, MySQL, Postgres, с возможностью настройки периодичности.
    - Создание новых столбцов в загружаемых таблица на основе простых арифметических формул и других столбцов.
    - Автоматические подсказки типов данных в столбцах.
    - Предварительный анализ таблицы (количество пустых ячеек, ошибки формата, дублирование).

2. Подсистема многомерной аналитической обработки информации.

3. Подсистема визуализации аналитических панелей с модулем детализации (drill—down).

4. Архитектура системы

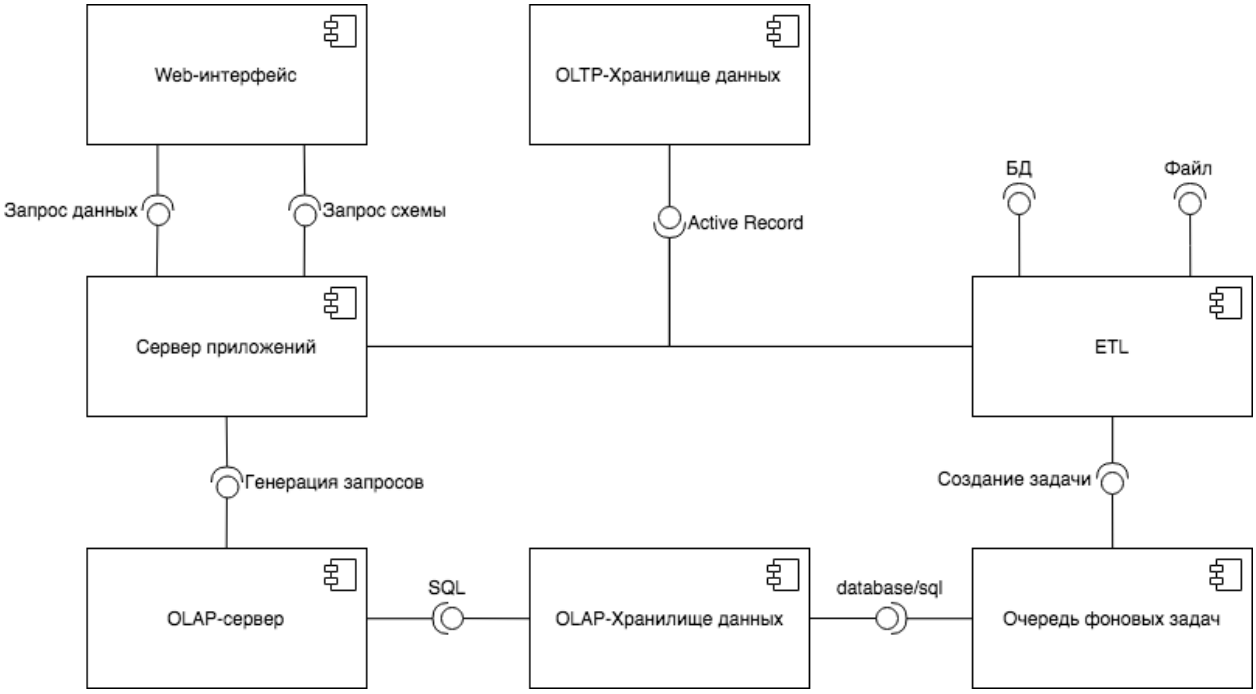

Рис. 1: Архитектура системы

Система состоит из двух функциональных частей: модуля многомерной аналитической обработки (компоненты «OLAP—Хранилище данных» и «OLAP—сервер») и модуля загрузки, преобразования и импорта данных (компоненты «ETL» и «Очередь фоновых задач»). Первый модуль состоит из аналитического хранилища данных, отвечающего за хранение, сжатие, и эффективное чтение данных, и OLAP—сервера, отвечающего за формирование запросов к хранилищу данных и формирование многомерной модели на концептуальном уровне. Второй модуль состоит из коннекторов к распространенным СУБД, парсеров распространенных файловых форматов, сервисов интеграции с внешними информационными системами, сервиса трансформации данных к формату хранилища и сервиса загрузки данных. Оба модуля работают под управлением сервера приложений и взаимодействуют с пользователем посредством веб—интерфейса.

4.1. OLAP—хранилище данных

Чтобы выбрать СУБД было проведено сравнительное исследование столбцовых СУБД (Vertica, ClickHouse, InfiniDB, MonetDB, Hive) на открытых данных ЕИС (единая информационная система в сфере гос. закупок) по контрактам. Сравнивалось время выполнения следующих запросов.

1. `SELECT count(*) FROM contracts`.
2. `SELECT count(*) FROM contracts WHERE oktmo != '14701000001'`.
3. `SELECT sum(price) FROM contracts`.
4. `SELECT uniq(customer_inn) FROM contracts`.
5. `SELECT oktmo, sum(price), count(*) AS c, avg(nmck), uniq(custom FROM contracts GROUP BY oktmo ORDER BY c DESC LIMIT 10`.

Размеры таблиц: 100 млн строк, в формате CSV таблица занимает 220G. Каждый тест проводился 10 раз и бралось среднее значение, чтобы уменьшить значение шумов в результате. Параметры сервера:

`Intel Xeon E5-2697 2,70 GHz`

`128 GB RAM`

Результаты тестирования представлены в Таблице 2. Время приведено в секундах

№ Запроса	Vertica	ClickHouse	InfiniDB	MonetDB	Hive
1	0.044	0.137	1.972	0.029	2.157
2	0.124	0.094	0.801	2.557	58.777
3	0.250	0.442	86.484	87.650	47.942
4	0.927	0.610	28.658	14.286	61.381
5	4.119	1.101	8.221	275.857	103.410

Таблица 2: Результаты тестирования СУБД

По результатам исследования видно, что лучшие результаты показывают Vertica и ClickHouse. Но т.к. ClickHouse — отечественный проект с открытым исходным кодом и лицензией на свободное использова-

ние, было решено выбрать его в качестве основного хранилище данных системы.[7]

4.2. OLAP—сервер

Mondrian: OLAP сервер с открытым исходным кодом, поддерживается и разрабатывается компанией Pentaho. Может быть настроен для работы с большинством распространенных реляционных СУБД. Работает на большинстве современных операционных систем. Лицензия позволяет свободное коммерческое использование. [6]

Среди плюсов Mondrian позволяет использовать различные готовые front—end системы, такие как Saiku, Hitachi Vantara и тд.

Среди минусов стоит отметить обязательное использование языка запросов MDX (Multidimensional Expressions) и стандарта OLAP—схемы Mondrian. Необходимость использования MDX приводит к необходимости необоснованного расширения компетенций, больших сложностей в модернизации и отладке. Поддержка стандарта OLAP—схемы Mondrian приводит к зависимости системы от сторонних разработчиков, к невозможности ее модернизации и меньшему удобству пользователя.

Также Mondrian не приспособлен к частому изменению схемы данных, что необходимо в нашем случае. Чтобы сервер начал обрабатывать запросы согласно с новой схемой, необходимо его полностью перезапустить. Каждый перезапуск приводит к задержке в несколько секунд и сбросу кеша запросов.

В следствие нестандартного синтаксиса джоинов СУБД ClickHouse, Mondrian не может быть выбранным ранее хранилищем данных.

Druid: Столбцовое хранилище данных с открытым исходным кодом. Спроектирован для максимально быстрой обработки запросов на запись большого количества данных. Из плюсов отметили для себя следующее: Как и Mondrian позволяет использовать различные готовые front—end системы, например: Pivot, Caravel и тд. Но их значитель-

но меньше, чем у Mondrian. Но, внедренее Druid еще сложнее, из—за большой инфраструктуры, которая идет вместе с ним. Сложность инфраструктуры: требуются отдельные ноды для получения, обработки и хранения данных, для отказоустойчивости необходимо двукратное количество серверов[8]

Из—за описанных недостатков эти OLAP—сервера невозможно адаптировать под наши задачи. Другие OLAP—сервера не могут быть использованы из—за лицинзионных ограничений. Поэтому было решено реализовать собственный OLAP—сервер.

4.3. Сервер приложений

Сервер приложений нужен для координации основных компонентов системы. Для его быстрой реализации был выбран уже знакомый команде фреймворк Ruby on Rails. К другим преимуществам использования этого фреймворка в данном проекте можно отнести: удобный и продуманный процесс работы с базой данных, простые общепринятые практики по написанию безопасных приложений и возможность масштабировать проект без особых усилий.[1]

4.4. ETL—модуль

Модуль загрузки и обработки информации должен предоставлять возможность пользователю загрузить как информацию из файлов форматов: CSV, XML, XLSX, XLS, так и настроить автоматическую выгрузку из следующих СУБД: Oracle, MS SQL, MySQL, PostgreSQL. Для обеспечения этих возможностей и возможностей расширения была выбрана модельная архитектура. Где для каждого формата файлов или СУБД используется отдельный модуль загрузки, который приводит данные в унифицированный вид. А вся логика преобразований и выгрузки в хранилище данных производится уже над этим унифицированным представлением.[3]

5. Реализация системы

5.1. OLAP—сервер

OLAP—сервер в нашем приложении выполняет две главные функции.

1. Преобразование запросов, поступивших от пользователя в валидные SQL—запросы к OLAP—хранилущу
2. Преобразование результатов, полученных из OLAP—хранилища в формат, удобный для отображения в интерфейсе

Рассмотрим запрос на построение графика на упрощенном примере.

```
{
  element_type: ''bar_chart'', filters: [{
      field1: {id: ''date'',  cube_id: ''notifications'', reg_exp: '%Y'},
      field2: {value: ['2018', '2019']}, condition: ''='',
    }, {
      field1: {id: ''price'', cube_id: ''contracts'', agg: 'sum'},
      field2: {value: 1000000}, condition: ''>='',
    }
  ], dimensions: [{
      agg: ''none'', place: ''column'', args: [
        {id: ''supplier_solo'', cube_id: ''contracts''}
    ]}, {
      agg: ''none'', place: ''column'', args: [
        {id: ''eco_type'', cube_id: ''contracts''}
    ]}
  ], measures: [{
      agg: ''median'', place: ''row'', args: [
        {id: ''price'', cube_id: ''contracts''}
    ]}
  ]
}
```

В данном примере с клиентской части приложения мы получаем запрос на предоставление данных с двумя фильтрами. В фильтре содержится информация о том, из какой колонки и таблицы в БД взять

данные для фильтрации (поле id и $cube_id$), как их обработать (поле reg_exp), с чем сравнивать (поле $value$) и по какому условию (поле $condition$). Из этих данных формируется $WHERE$ часть запроса. Второй фильтр имеет схожую структуру, но является $HAVING$ фильтром на суммарную цену.

Далее идет информация о измерениях, интересующих пользователя. Указывается откуда брать данные и как их агрегировать. И в последней части идет информация о мерах в запросе. Из этого примера будет сформирован SQL—запрос:

```
SELECT
  sum(''contracts.price''),
  ''contracts.supplier\_solo'',
  ''contracts.eco\_type''
FROM (
  SELECT * FROM (
    SELECT
      ''price'' AS ''contracts.price'',
      ''supplier_solo'' AS ''contracts.supplier_solo'',
      ''eco_type'' AS ''contracts.eco_type'',
      ''price'' AS ''contracts.price'',
      ''notif_number'' AS ''notifications_contracts_key''
    FROM ''contracts''
  ) LEFT JOIN (
    SELECT * FROM (
      SELECT ''purchase_number'' AS ''notifications_contracts_key''
      FROM ''notifications''
      WHERE formatDateTime('date', '%Y') in ('2018', '2019')
    )
  ) USING ''notifications_contracts_key''
)
GROUP BY ''contracts.supplier_solo'', ''contracts.eco_type''
HAVING ''contracts.price'' >= 1000000
ORDER BY ''contracts.supplier_solo'', ''contracts.eco_type''
FORMAT JSONCompact
```

Полученные данные будут преобразованы к специальному представлению, перед отправкой клиенту. Например, вот такие результаты могут быть получены. Значение уменьшены для более легкого восприятия

структуры данных.

```
{
    params: [{
        values: ['Да', 'Нет'],
      }, {
        values:['Больше 10%', 'Больше 10% (Отрицательная)',
                'Меньше 10%', 'Меньше 10% (Отрицательная)']
      }]
    values: [
      [
        [18, 23, 17, 18],
        [26, 34, 39, 44]
      ],
      [
        [18, 23, 17, 18],
        [26, 34, 39, 44]
      ]
    ]
}
```

Как можно заметить, сначала идет описание осей будущего графика, а потом его данных. Данные формируются исходя из количества требуемых измерений. Так, при двух измерениях, глубина вложенности массивов в будет 2. Используя эти данные, веб—интерфейс сможет отобразить график (Рис. 2).

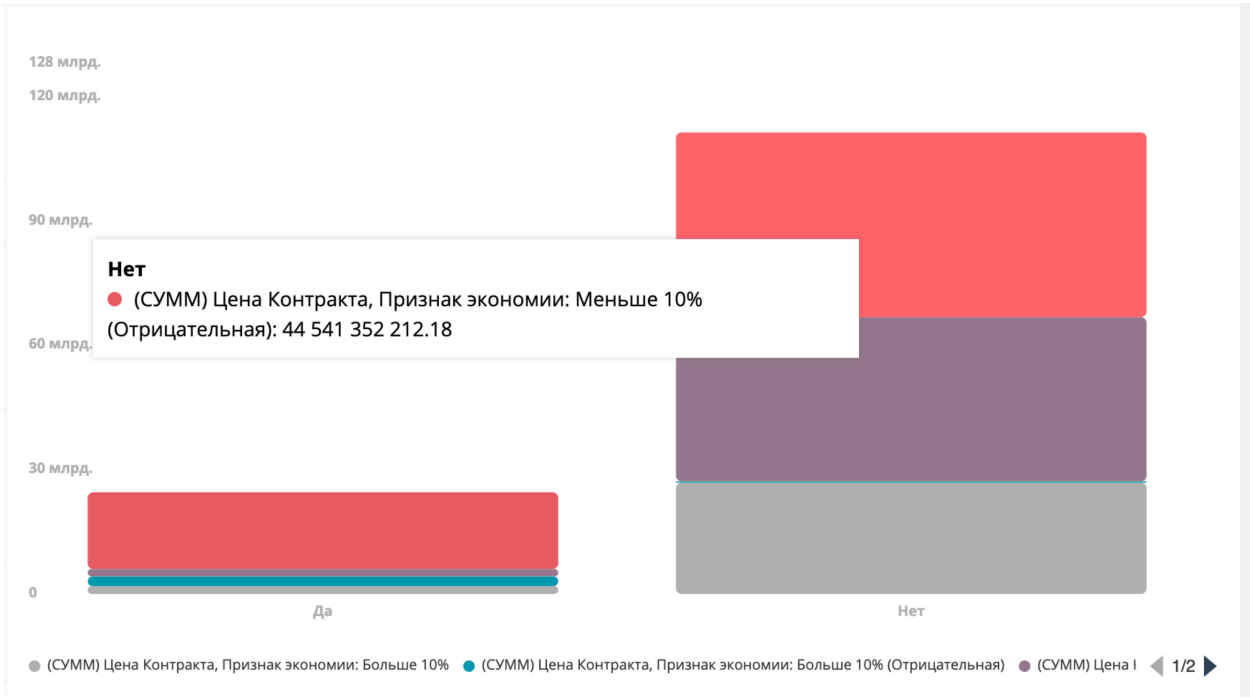

Рис. 2: График из примера

5.2. ETL—модуль

Модуль загрузки был реализован, как отдельный сервис на Go, принимающий запросы по HTTP. Это позволит в случае возрастания нагрузки легко вынести его на отдельный сервер, чтобы увеличить скорость загрузки в хранилище. Процесс происходит в несколько этапов.

1. Сервер приложения получает от пользователя файл или данные для подключения к базе данных.

2. Сервер приложения согласует с пользователем первичные ключи, форматы данных, колонки для выгрузки, новые колонки.

3. Сервер приложения посылает HTTP запрос ETL—сервису, содержащий расположение файла или данные для подключения к БД, данные о том, какие колонки добавить в первичный ключ, какие форматы использовать, какие колонки не загружать, а какие создать, как назвать новую таблицу.

4. ETL—сервис создает временную таблицу, учитывая все параметры и начинает загружать данные пачками по 100’000 строк в нее,

преобразуя данные в соответствующие типы.

5. ETL—сервис при успешной загрузке переименовывает временную таблицу, делая ее постоянной, или удаляет ее, если во время загрузки произошла критическая ошибка.

5.3. Интерфейс

Веб—интерфейс был реализован с использованием технологий React и TypeScript.

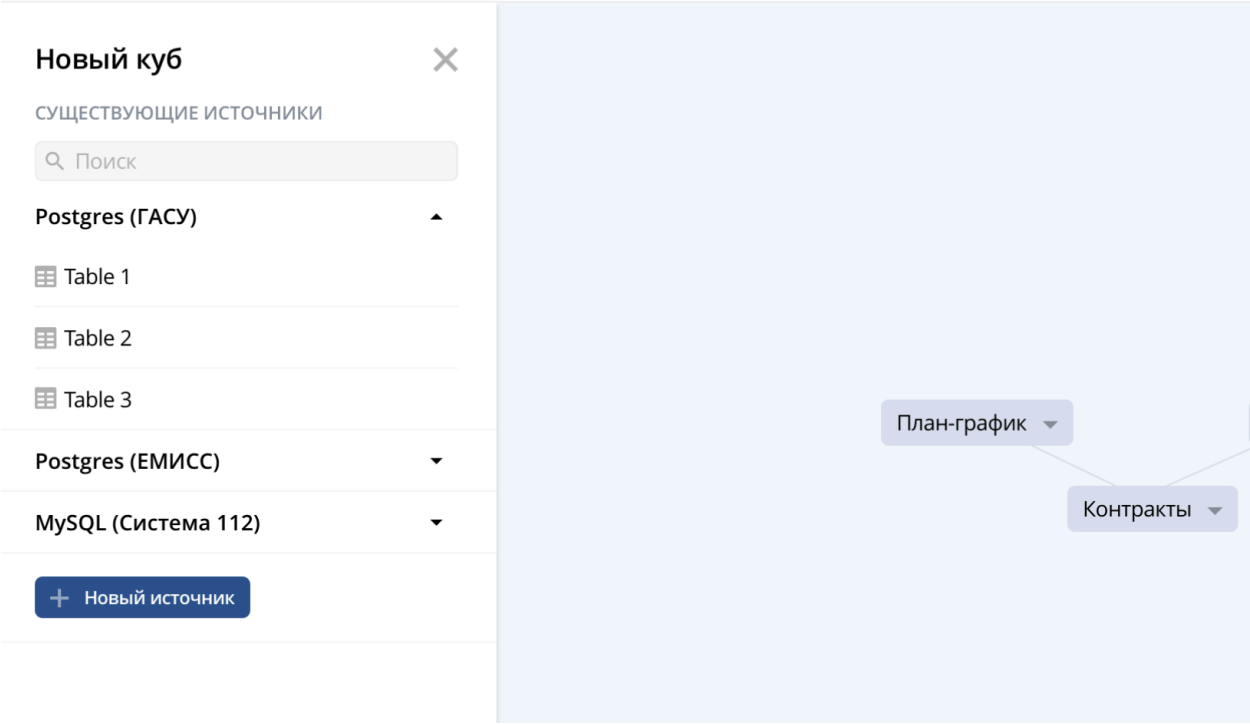

Рис. 3: Интерфейс создания источника данных

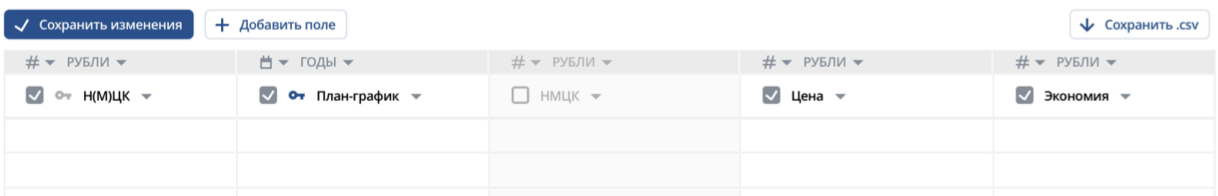

Рис. 4: Элемент интерфейса корректировки источника перед импортом

Заключение

В ходе работы над проектом были достигнуты следующие результаты.

- Проанализированы основные существующие решения: QlikView, Tableau, Power BI. Выделены общие черты интерфейса и сценарии знакомые пользователю.
- Разработаны требования к системе анализа данных. Определены самые популярные СУБД и форматы файлов необходимые заказчику.
- Создана модульная архитектура системы. В качестве хранилища выбран ClickHouse. OLAP—сервер было решено реализовывать с нуля.
- Выполнена реализация системы, а именно разработаны модули загрузки, обработки и анализа данных.

Список литературы

[1] Bächle M. Kirchberg P. Ruby on rails // IEEE Software. — 2008.

[2] Chaudhuri S. Dayal U. An Overview of Data Warehousing and OLAP Technology // SIGMOD Record. — 1997.

[3] Ralph Kimball Joe Caserta. The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data. — 2004.

[4] Sense Qlik. News since Qlik Sense 1.0. — QlikTech International AB, 2016.

[5] Tableau. Tableau Help. — 2019. — URL: `onlinehelp.tableau.com` (дата обращения: 12.01.2019).

[6] William Back Nicholas Goodman, Hyde Julian. Mondrian in Action / Ed. by Susanna Kline. — Manning Publications, 2014.

[7] Yandex. Clickhouse Docs. — 2019. — URL: `clickhouse.yandex/docs` (дата обращения: 22.02.2019).

[8] Yang F. Tschetter E. Léauté X. Ray N. Merlino G. Ganguli D. Druid: A real-time analytical data store // Proceedings of the ACM SIGMOD International Conference on Management of Data. — 2014.