ПРЕДИСЛОВИЕ

Платформа Bi.Qube реализована на технологиях разрешенных минцифры России платформа и отдельные компоненты зарегистрированы в Росреестре:

- BI.Qube 2.0 – Реестровая запись №23535 от 12.08.2024
- MetaStaging – Реестровая запись №17067 от 24.03.2023;
- MetaVault – Реестровая запись №16579 от 13.02.2023;
- MetaControl – Реестровая запись №18132 от 29.06.2023.

Решение разработано на кросс-платформенном .NET.

В качестве системных компонентов используются лидирующие в своём классе продукты с открытым исходным кодом (СУБД PostgreSQL для сохранения метаданных, web-сервер NGINX для Back-end API и пользовательского интерфейса, объектное хранилище для хранения промежуточных данных, система управления учётными данными Keycloak для реализации разграничения доступа), возможно использование сборок отечественных вендоров, входящих в реестр Российского ПО.

Для работы пользователя, помимо развитого API и CLI, предусмотрен адаптивный кросс-браузерный визуальный интерфейс, реализованный на React.

ВВЕДЕНИЕ

BI.Qube 2.0 (далее BI.Qube) – платформа (фреймворк, набор инструментов) предназначена для комплексного анализа данных и метаданных, начиная от извлечения их из источников данных (учетных систем, веб-сервисов, баз данных и так далее) до построения масштабируемой модели данных для хранения и использования в BI аналитики с возможностью обогащения не системными данными, осуществления контроля за качеством данных с организацией представления ролевого доступа к реализованной модели данных.

Применение BI.Qube позволяет существенно снизить требования к уровню подготовки специалистов по построению корпоративных хранилищ данных (КХД) с использованием методологии DataVault, и в большинстве случаев позволяет отказаться от написания программного кода и вести проектирование КХД в подходе no code/ low code.

BI.Qube включает в себя ряд компонентов, позволяющих полноценно решать определенный круг задач, появляющихся при построении КХД не зависимо друг от друга. С другой стороны, каждый компонент предоставляет полноценный интерфейс доступа к данным о своей деятельности. Так, сторонний оркестратор позволяет организовать ETL процесс оптимальным образом с точки зрения временных (ресурсных) затрат. В ряде случаев данные, извлекаемые из источников, в автоматическом режиме укладываются в хранилище в модель DataVault (автоматическое определение бизнес ключей на основании метаданных источника), и пользователю нет необходимости выполнять какие-то дополнительные действия. Концепция построения подсистемы MDM непосредственно в хранилище DataVault существенно снижает трудозатраты, связанные с работой с нормативно справочной информацией (НСИ) в том смысле, что интеграция всех справочников НСИ с хранилищем уже реализованы на уровне системы (хранилища) и не требует от пользователей никакого вмешательства в виде программного кода, что существенно удешевляет разработку хранилища данных в целом, сопровождения его в будущем и самое главное, позволяет бизнес-пользователям самим создавать и настраивать работу с НСИ, «золотой» записью, обогащением новыми данными без привлечения программистов. Построение хранилища и подсистемы MDM на основе модели DataVault существенно расширяют возможности по управлению доступа к данным, одновременной работе с данными, сохранения истории появления и изменения данных.

Продукт **BI.Qube** и его компоненты используют общий подход к организации артефактов разработки. Это позволяет унифицировать процесс разворачивания и тестирования средств разработки и отладки. Это также упрощает перенос и объединение изменений между разными средами разработки.

В состав **BI.Qube** входят следующие компоненты:

- **MetaCommon** – компонент, имеющий развитый визуальный интерфейс, реализующий работу в режиме no code/ low code, предназначенный для выполнения всех необходимых настроек, которые в последующем использует все остальные компоненты;
- **MetaStaging** – компонент, имеющий развитый визуальный интерфейс, реализующий работу в режиме no code/ low code, предназначенный для извлечения данных из источников и доставки их в точку назначения;
- **Data&Model** – компонент, имеющий развитый визуальный интерфейс, реализующий работу в режиме no code/ low code, предназначенный для создания аналитической модели данных, работой со справочниками, обогащения данных. Компонент работает с данными, доставляемыми с использованием компонента MetaStaging. Включает в себя компонент MetaVault и MetaMasterData;
    - **MetaVault** – компонент, имеющий развитый визуальный интерфейс, реализующий работу в режиме no code/ low code, предназначенный для организации хранения данных в модели DataVault. Пользователь может не иметь представления об особенностях модели DataVault, система все необходимые действия выполняет сама и предоставляет доступ к автоматически сгенерированным представлениям;
    - **MetaMasterData** – компонент, имеющий развитый визуальный интерфейс, реализующий работу в режиме no code/ low code, предназначенный для работы с нормативно-справочной информацией, обогащения данными, вводимыми в ручном режиме через веб-интерфейс, создания новых данных. Данный компонент работает только в связке с MetaVault и отдельно работать не может. Компонент реализует возможности MDM систем, и созданные с его помощью объекты не требуют интеграции с объектами MetaVault;
- **MetaControl** – компонент, имеющий развитый визуальный интерфейс, реализующий работу в режиме no code/ low code, предназначенный для создания различных бизнес-правил, например, контроля за ETL-процессами. Компонент выполняет бизнес-правило, которое может быть представлено, например, запросом. Выполняет сопоставление полученного результата с эталонным (ожидаемым), и при обнаружении расхождений (с учетом заданной точности) выполняет рассылку по e-mail или по средствам telegram-канала информации о выполненных действиях всем заинтересованным получателям.

ВВЕДЕНИЕ В ОРГАНИЗАЦИЮ ХРАНИЛИЩ ДАННЫХ

Хранилище данных (англ. Data Warehouse) – предметно-ориентированная информационная база данных, специально разработанная и предназначенная для подготовки отчётов и бизнес-анализа с целью поддержки принятия решений в организации. Строится на базе систем управления базами данных и систем поддержки принятия решений. Данные, поступающие в хранилище данных, как правило, доступны только для чтения.

В общем случае хранилище данных представляет из себя базу(ы) данных, в которой определенным образом организовано хранение данных (чаще всего применяется послойное хранение).

Можно выделить следующие слои хранения данных:

- Staging – слой сырых данных, обычно представляется двумя слоями:
  - Raw – область хранения сырых данных различных форматов (csv, xml, xlsx, json и т.д.).
  - ODS – область хранения данных в едином формате (чаще всего sql-таблицы базы данных). Здесь могут производиться предварительные преобразования, а так же оценка качества данных.
- Core – слой долгосрочного хранения данных, представляется обычно одним слоем:
  - DDS – здесь создается модель данных, выполняются необходимые трансформации данных, очистка, интеграция с мастерданными, обеспечивается отслеживание изменений данных (SCD2).
- Presentation – слой представления данных в виде удобном пользователю, представлен обычно двумя слоями:
  - DataMart – данные подготовлены для использования и оптимизированы под чтение BI-системами.
  - Report – данные представлены в виде отчетов и/или дашбордов.

Чаще всего, разработка хранилища включает в себя использование и комбинирование различных подходов к созданию слоёв. В зависимости от уникальных бизнес-требований проекта можно применять различные техники моделирования (например, использовать Data Vault на слое DDS, а на Data Mart – размерное моделирование).

Слой	**Staging**		**Core**	**Presentation**	
Компонет BI. Qube	**MetaStaging** **MetaControl**		**Data&Model** **(MetaVault, MetaMasterData, MetaFact)**	**MetaCube**	
Подслой	**Raw**	**ODS**	**DDS**	**Data Mart**	**Report**
Описание	• Зона загрузки сырых данных различных форматов (tsv, csv, xml, json и т.д.) из внешних источников **As-Is** • Действует CDC (Change Data Capture)	• Данные приведены в единый формат (стандартизированы) и загружены в первый слой хранилища • Загрузка данных NRT • Начальная очистка данных и применение DQ	• Создается модель данных • Трансформация данных • Добавление мета-данных • Финальная очистка данных • Поддержка историчности (SCD2) • Интеграция с мастер-данными • Оптимизация под запись	• Создается модель данных • Готовый "дата-продукт" для бизнеса • Применена бизнес-логика в трансформации данных • Данные семантически сгруппированы • Действует BI self-service • Оптимизация под чтение	• Готовый отчет /визуализация • Данные агрегированы и объединены в один дата сет для целостного представления
Владельцы/ пользователи	**Data Engineers**	**Data Engineers,** **Data Scientists**	**Operational Analysts,** **Data Scientists**	**BI Analysts,** **Managers,** **Data Scientists**	**Executives, Top** **Management**
Модель данных	**As-Is**	**As-Is /** **Raw Data Vault**	**Data Vault / Business** **Data Vault / 3NF**	**Dimensional** **Modelling** **(Kimball's approach)**	**One big table**

Фреймворк BI.Qube поддерживает возможность послойного построения хранилища и в общем случае предлагает применять архитектуру показанную на рисунке ниже

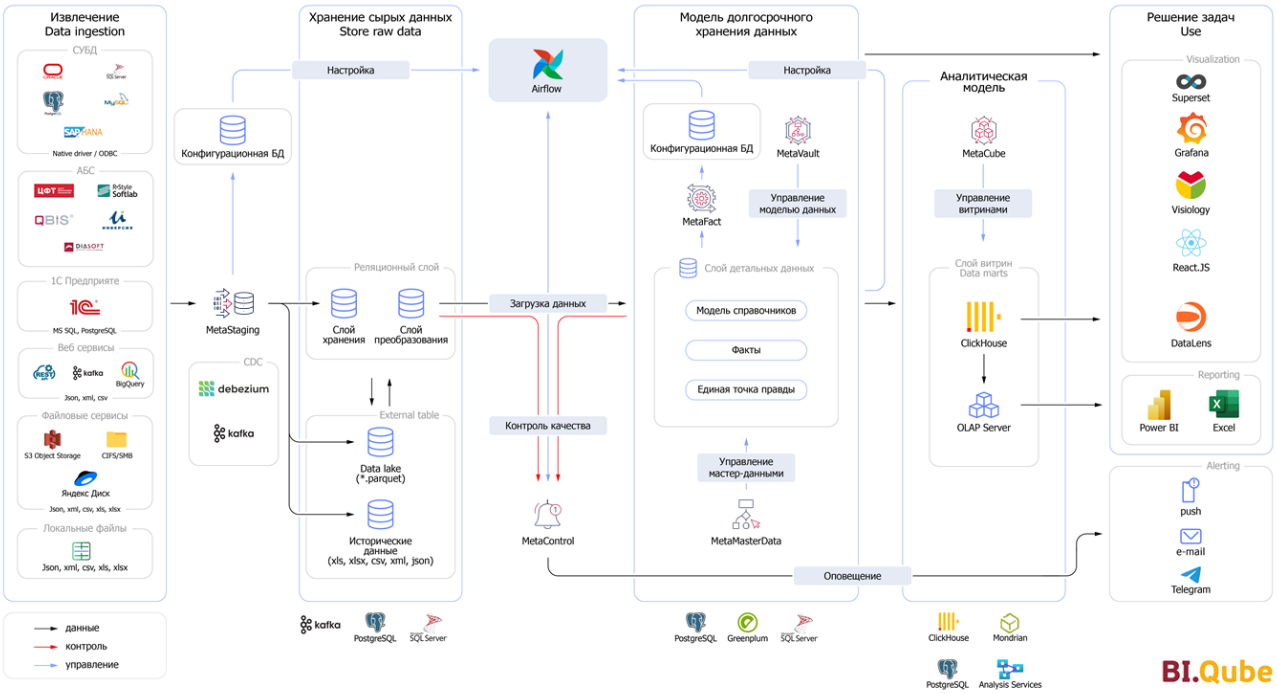


Рисунок. Архитектура организации хранилищ данных

ОРГАНИЗАЦИЯ РАБОТЫ С BI.QUBE 2.0

ОБЩИЕ СВЕДЕНИЯ

Визуальный интерфейс BI.Qube представлен веб-сервисом, организующим диалоговый режим работы с пользователем. Команды сгруппированы в боковом меню, состав каждой группы зависит от решаемых задач этой группой. Такой подход позволяет пользователю быстро находить требуемую функциональность. Состав групп, команд в группах, их расположения внутри группы, а так же некоторых визуальных элементах в командах (на страницах веб-интерфейса) зависит от типа приобретенной лицензии, версии BI.Qube и может отличаться представленного в настоящем руководстве. В руководстве приводится вся функциональность, независимо от типа лицензии. Так же руководство содержит только проверенную функциональность и не содержит описание самых новых версий.

СИСТЕМНЫЕ ТРЕБОВАНИЯ

BI.Qube, являясь фреймворком для используемой платформы СУБД не имеет жестких требований к аппаратному обеспечению, но производительность зависит от инфраструктуры в которой он разворачивается. В связи с этим, для среднего уровня предполагаемого проекта (до сотен гигабайт обрабатываемой информации) рекомендуются следующие характеристики аппаратного обеспечения:

- Количество vCPU 4
- Объём RAM 16 ГБ
- Размер диска 128 ГБ

Данный сайзинг рассчитан для единичного ландшафта, при необходимости реализации дополнительных сред, нужно учесть их количество.
В процессе развития системы, в зависимости от роста объёмов обрабатываемых данных, может потребоваться масштабирование платформы.

СИСТЕМЫ АВТОРИЗАЦИИ И ХРАНЕНИЯ СЕКРЕТОВ В BI.QUBE

Системы авторизации

Фреймворк BI.Qube поддерживает работу с разными системами авторизации. На данный момент реализована поддержка двух систем:

- **ASP.NET Core Identity** – предоставляет инструментарий для работы с пользователями и их аутентификацией. Эта система состоит из библиотеки *Microsoft.AspNet.Identity* с описанием основных классов и абстракций в виде интерфейсов хранилища пользователей и т.д.

- **Keycloak** – продукт с открытым кодом для реализации single sign-on с возможностью управления доступом, нацелен на современные приложения и сервисы.

Выбор системы авторизации осуществляется на этапе развертывания фреймворка и в будущем не может быть изменен.

В выбранной системе авторизации должна быть обязательно создана роль **biqube-admin.** Пользователи с этой ролью в системе будут обладать функциональностью администратора. В системе BI.Qube администратор имеет доступ ко всем объектам, создаваемым в системе и управляет доступом к ним.

Системы хранения секретов

Фреймворк BI.Qube поддерживает работу с разными системами хранения секретов. На данный момент реализована поддержка двух систем:

- **Infisical –** это платформа управления открытыми секретами, которая помогает компаниям управлять секретами своих инженеров и инфраструктуры. Кроме того, Infisical предоставляет возможности автоматического секретного сканирования и предотвращения утечки секретов.
- **Hashicorp Vault –** это инструмент для безопасного управления и хранения секретов, таких как ключи API, пароли и другая конфиденциальная информация.
- **BI.Qube** – хранение секретов во внутренней базе данных.

Выбор системы хранения секретов осуществляется на этапе развертывания фреймворка и в будущем не может быть изменен.

ASP.NET Core Identity

KEYCLOAK

ВХОД В СИСТЕМУ И АВТОРИЗАЦИЯ

Для входа в систему пользователь должен быть зарегистрирован в сервисе Keycloak. Адрес по которому расположен веб-интерфейс BI.Qube. Выдается пользователям лицом, осуществляющим развертывание системы, чаще всего это администратор.

Keycloak – это решение для управления идентификацией и доступом с открытым исходным кодом, предназначенное для использования в ИС, где могут использоваться паттерны микросервисной архитектуры.

В инфраструктуре пользователя сервис должен быть развернут, занесены пользователи и для пользователей BI.Qube должны быть выполнены соответствующие настройки.

В соответствии с настройками в keycloak пользователи могут попасть в систему BI.Qube в режиме **администратора или пользователя.** В первом случае пользователю доступны все настройки системы, нет никакого ограничения доступа к объектам системы. В режиме пользователя доступны объекты, расположенные в доменах, которые подключены к роли пользователя, с которой он авторизуется в системе. Роли пользователя задаются в системе keycloak. У одного пользователя может быть несколько ролей, все они в момент авторизации считываются из keycloak, и к каждой роли добавляется доступ к домену по умолчанию "default". Это не значит, что у пользователя появляется столько доменов по умолчанию, сколько ролей. Одному пользователю в таком случае доступен один домен default.

ТРЕБОВАНИЯ К НАСТРОЙКАМ KEYCLOAK

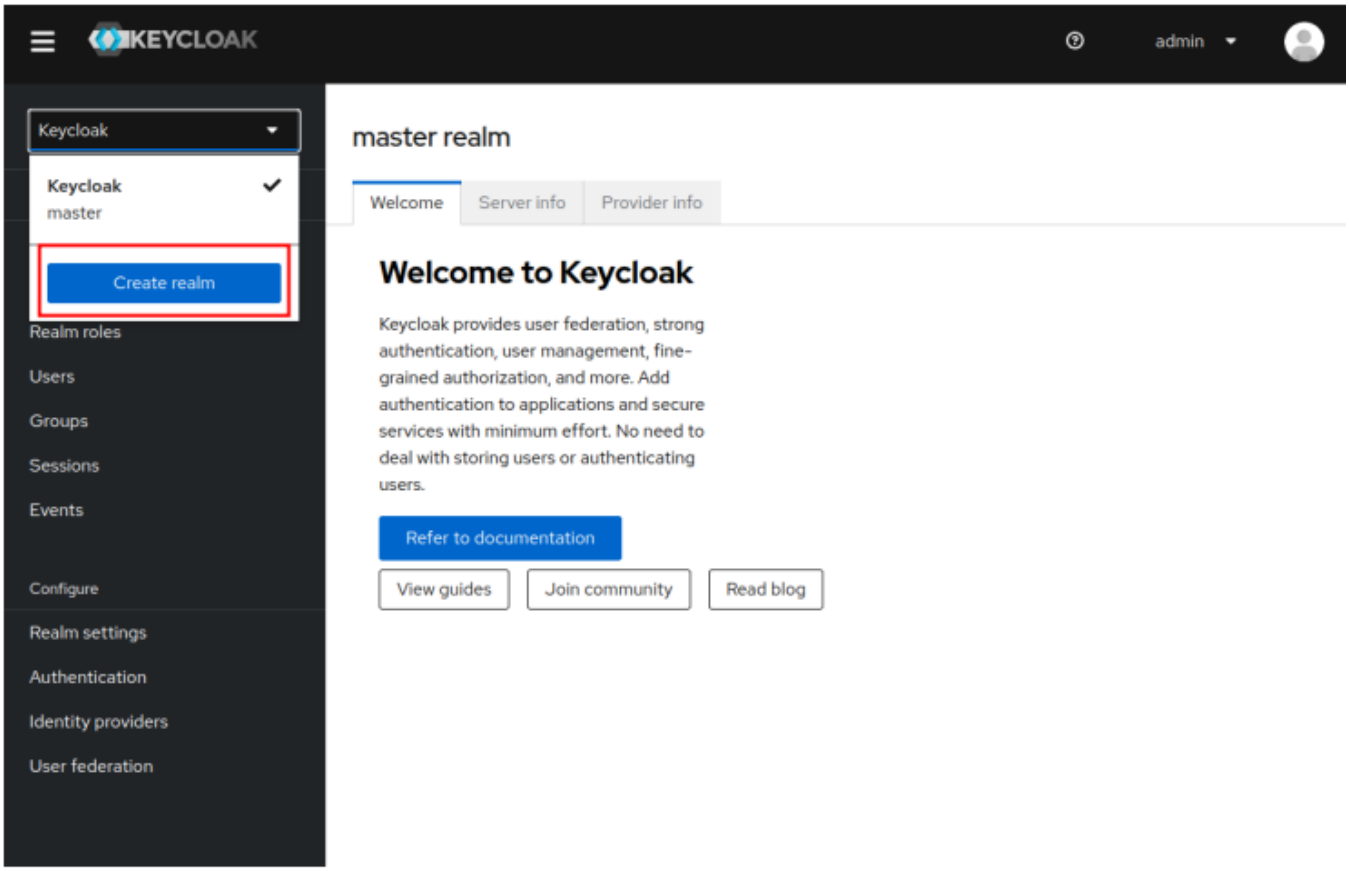
Для того чтобы функционал пользователей и ролей работал корректно, нужно настроить соответствующим образом Keycloak.

Для просмотра списка пользователей необходима учетная запись с ролью realm-management:view-users.

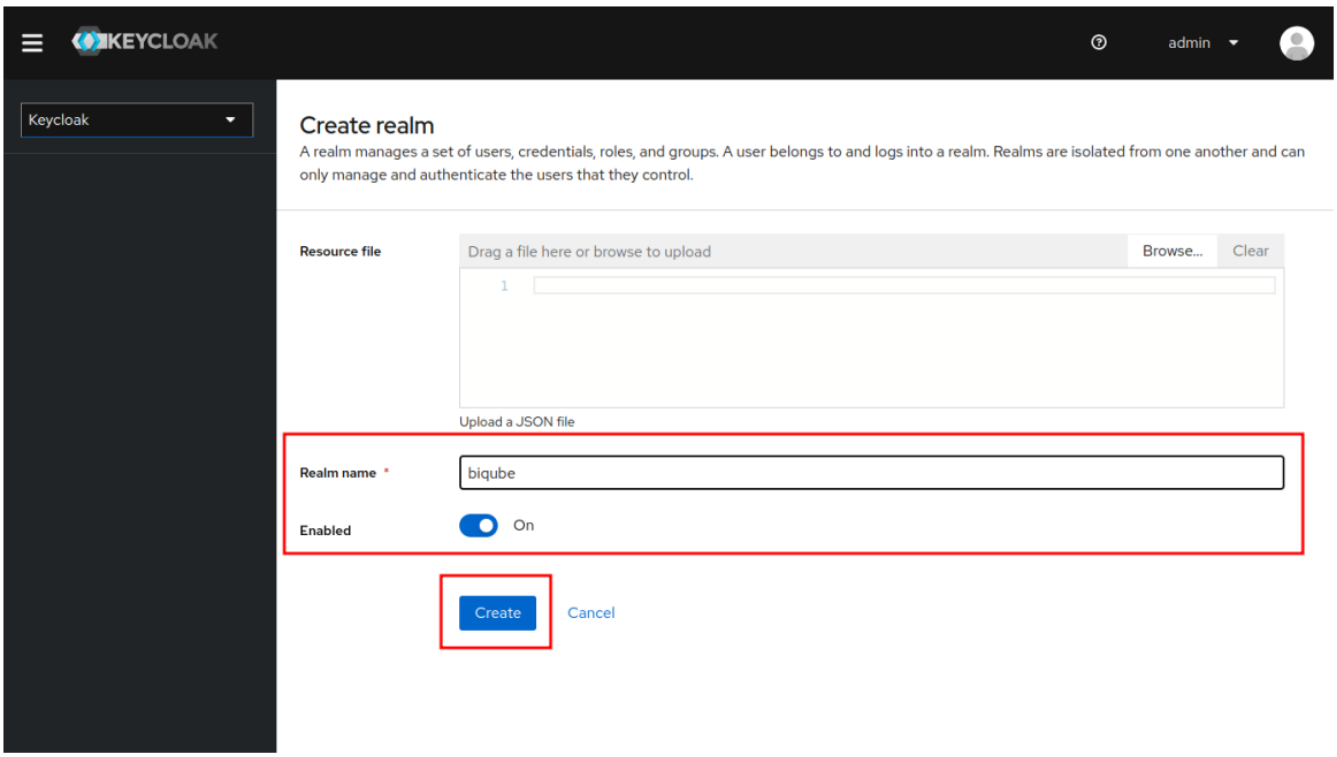
Для работы с ролями необходима учетная запись с ролью realm-management:view-clients.

Чтобы учесть все эти нюансы, нужно завести клиентскую роль (**не реалма**) **biqube-admin** и сделать её композитной, т.е. ассоциировать с ней вышеописанные роли. Дальше эту роль можно выдавать любому пользователю, которому нужны административные привилегии.

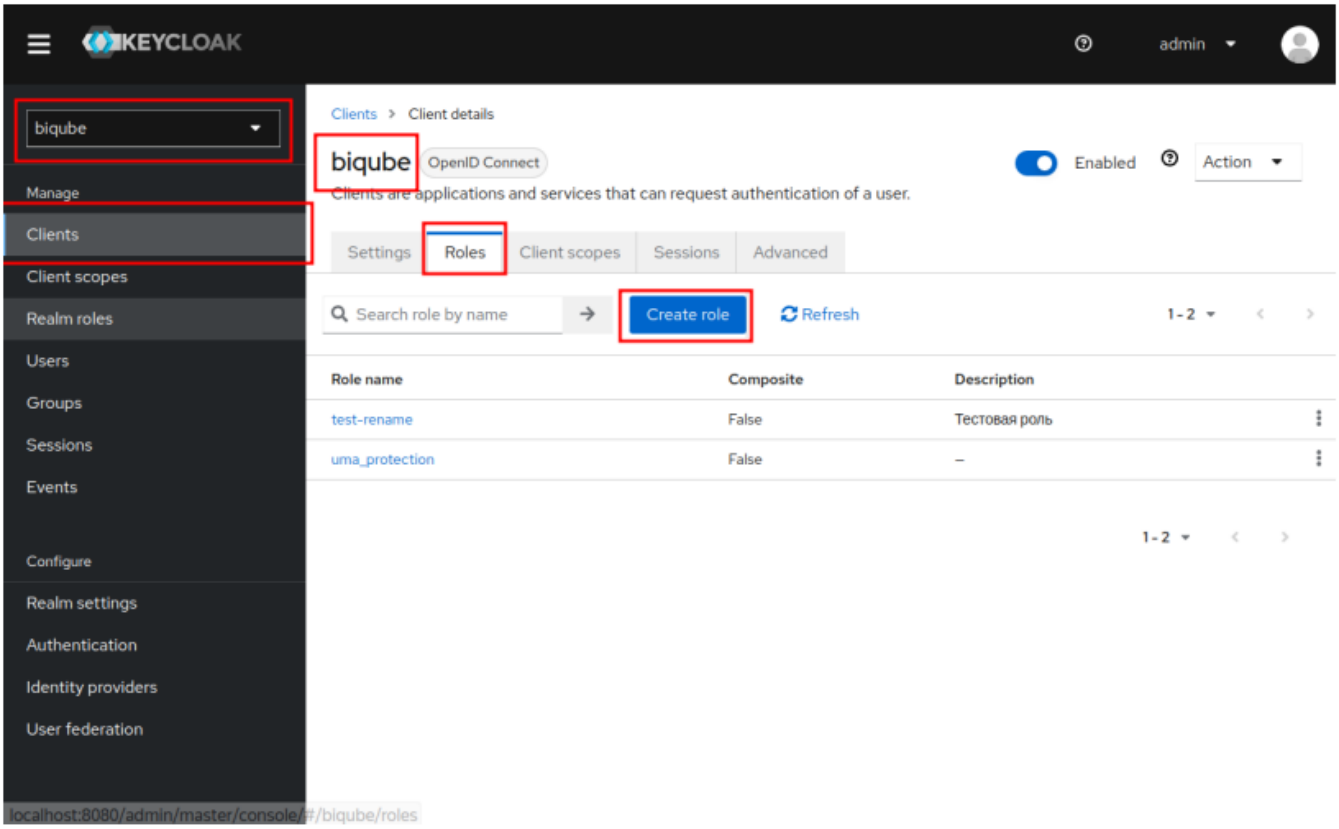
Шаг 1. Создаем realm

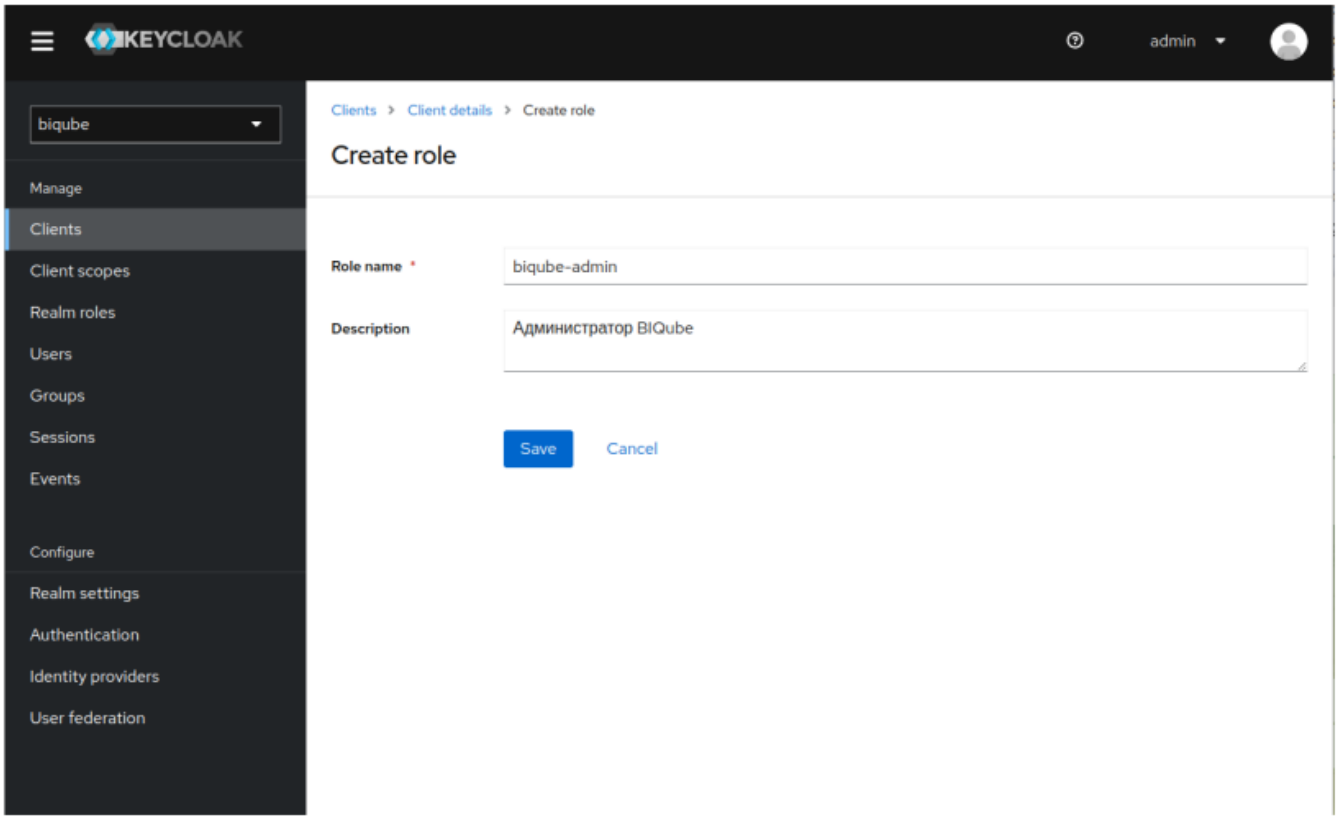


Указать имя realm - biqube.

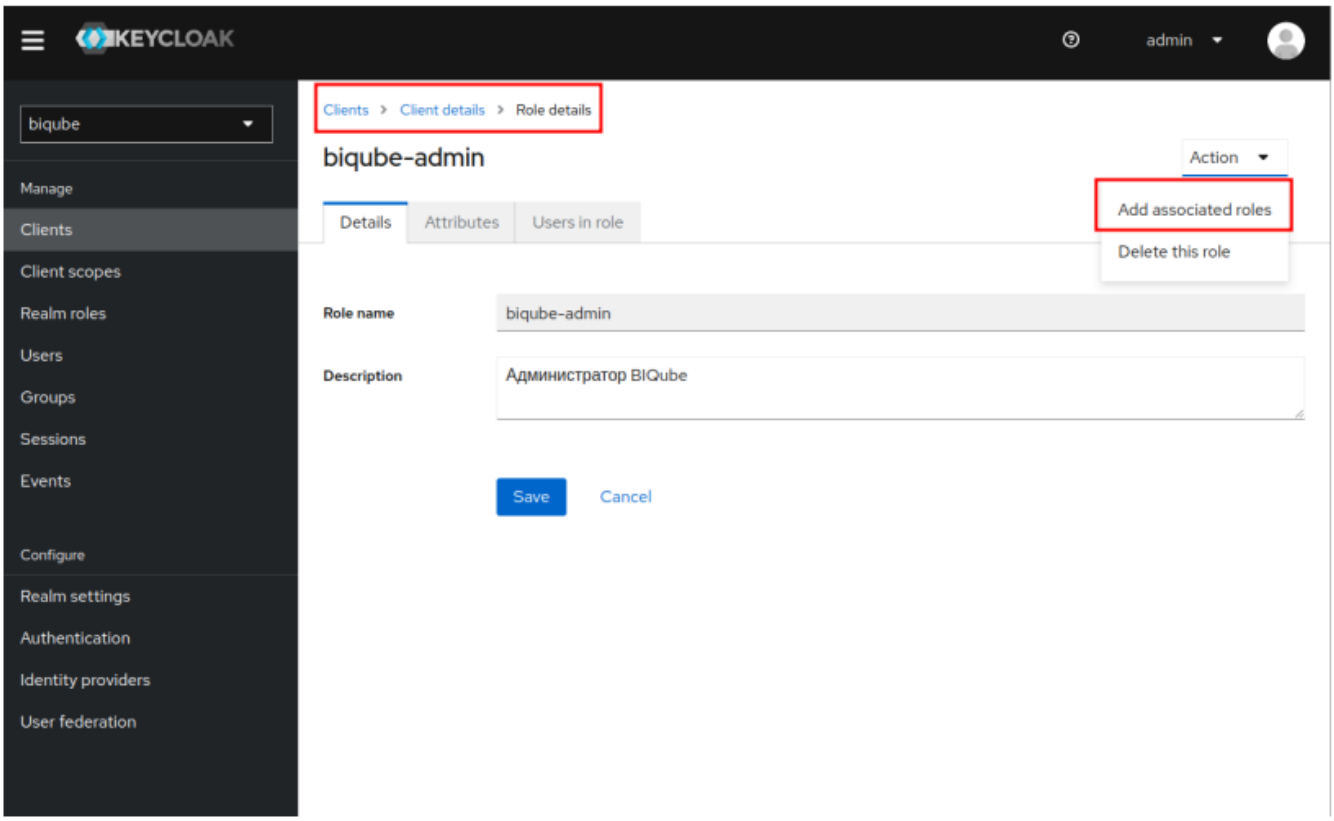


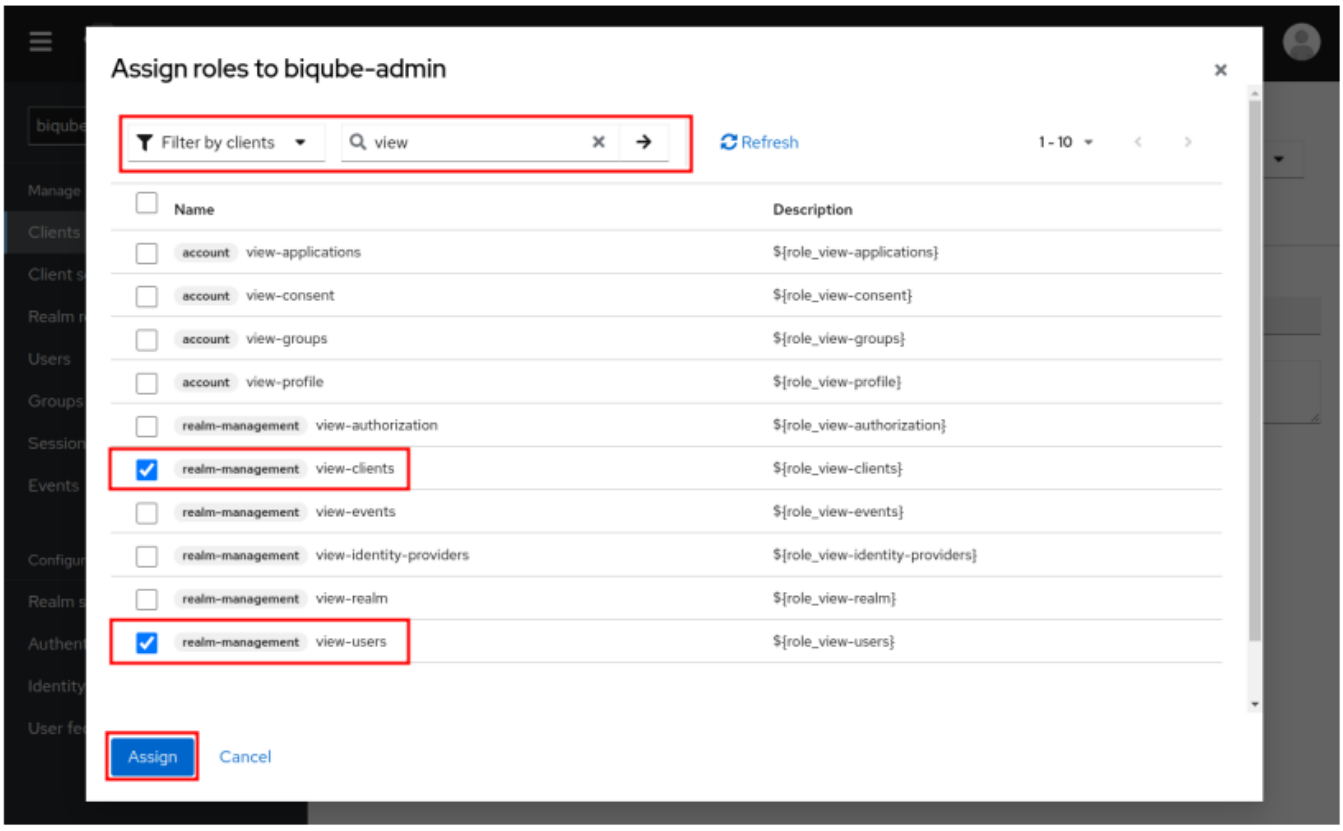
Шаг 2. Создаем роль в выделенном клиенте



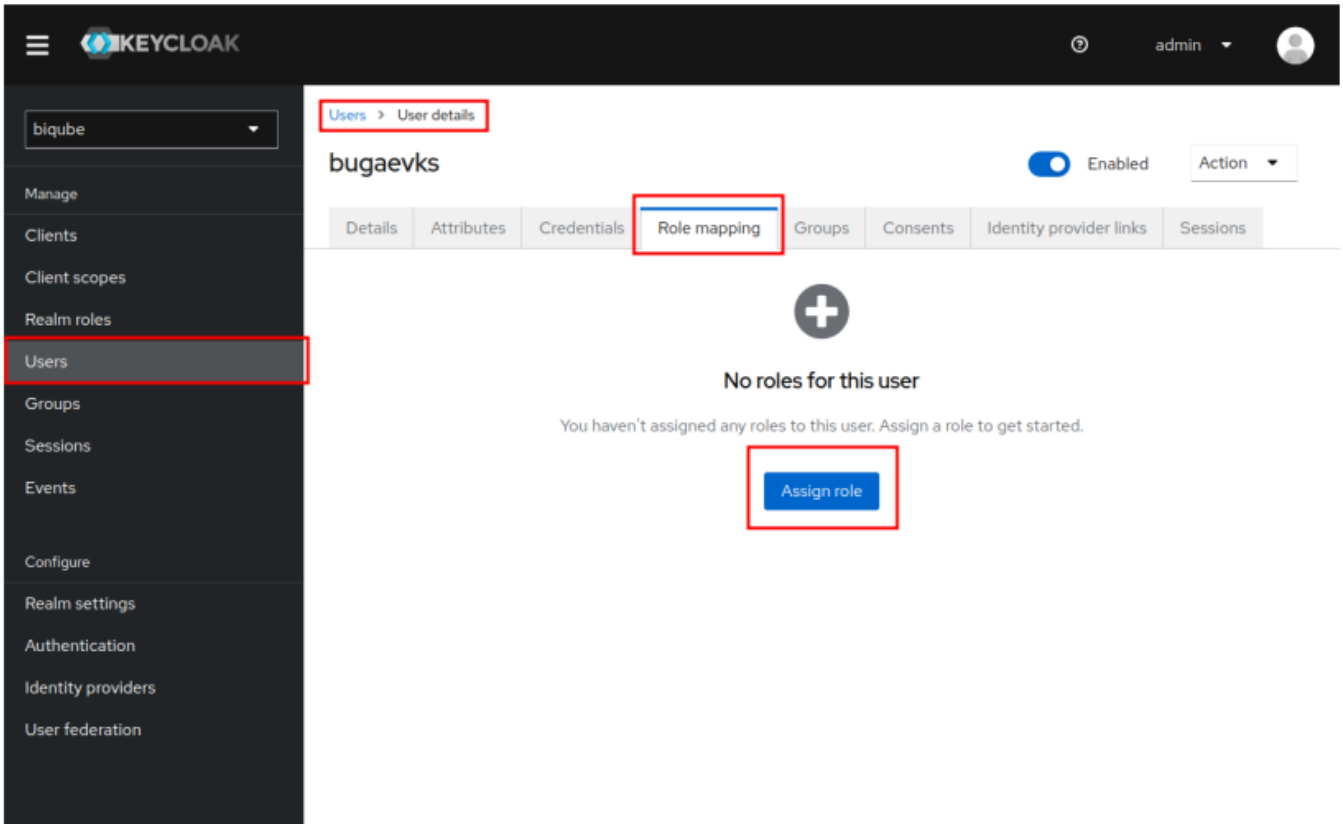


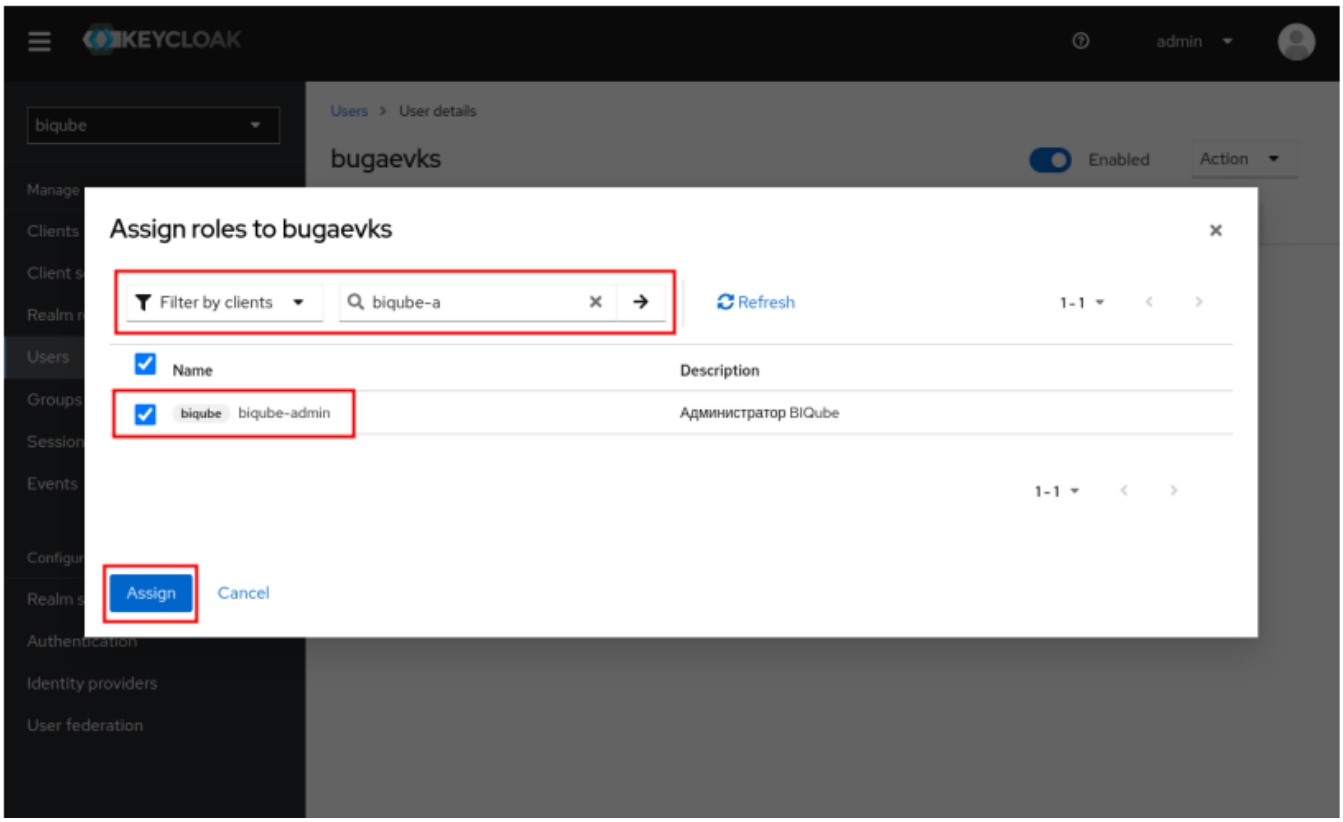
Шаг 3. Ассоциируем дополнительные роли с новой





Шаг 4. Назначаем административную роль пользователю


KEYCLOAK
admin
biqube
Manage
Clients
Client scopes
Realm roles
Users
Groups
Sessions
Events
Configure
Realm settings
Authentication
Identity providers
User federation
Users > User details
bugaevks
Enabled
Action
Details
Attributes
Credentials
Role mapping
Groups
Consents
Identity provider links
Sessions
No roles for this user
You haven't assigned any roles to this user. Assign a role to get started.
Assign role

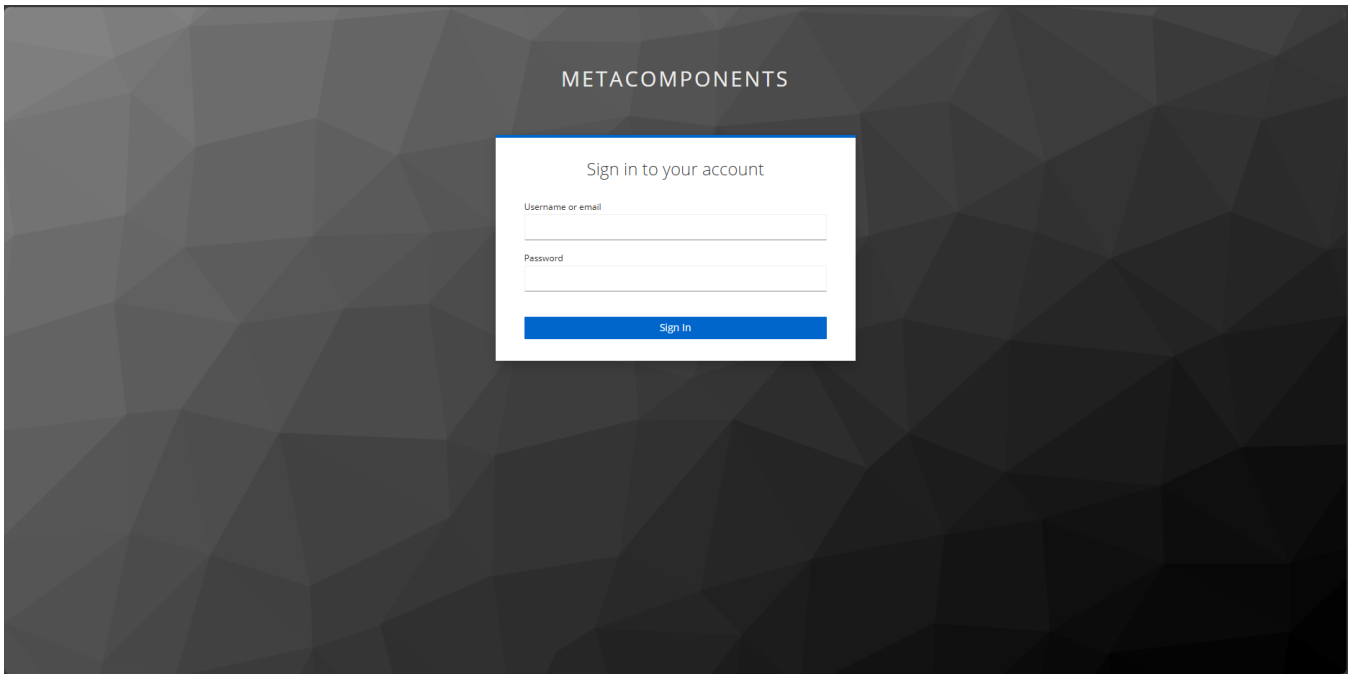

KEYCLOAK
admin
biqube
Users > User details
bugaevks
Enabled
Action
Assign roles to bugaevks
Filter by clients
biqube-a
Refresh
1 - 1
Name
Description
biqube
biqube-admin
Администратор BIQube
1 - 1
Assign
Cancel

INFISICAL

Hashicorp Vault

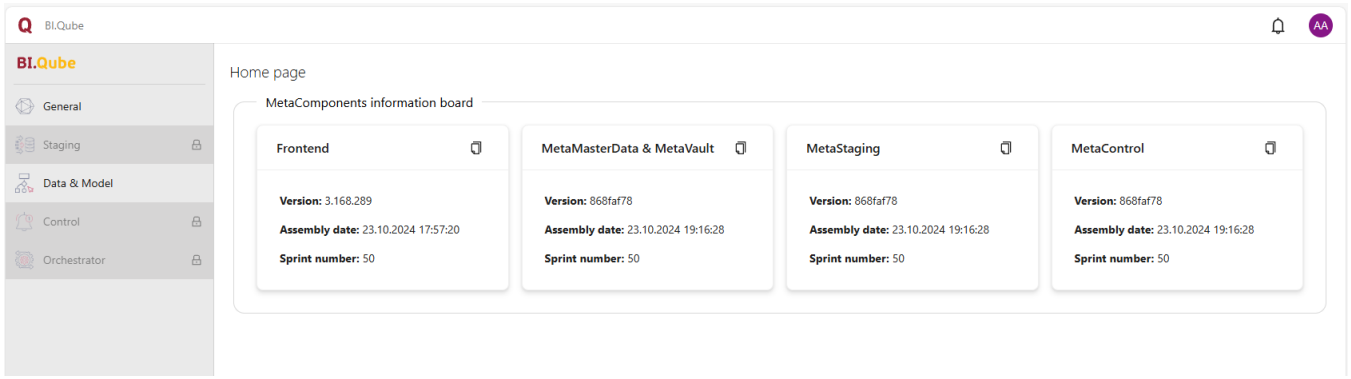
ПЕРВЫЙ ВХОД В СИСТЕМУ

После развёртывания системы и выполнения необходимых настроек в сервисах авторизации и хранения секретов пользователь может зайти на страницы фреймворка. Вход осуществляется через главное окно, в котором выполняется выбор роли, под которой будет выполнен вход. Однако предварительно BI. Qube перенаправит пользователя в сервис авторизации, в интерфейсе которого следует ввести свои учетные данные. Ниже на рисунке показан интерфейс авторизации keycloak.



Выбор роли пользователя перед входом в систему.

После выполнения авторизации пользователю доступна домашняя страница. На данной странице отображается информация о составе текущей версии BI. Qube. номера версий компонентов и даты их сборки.



ОПИСАНИЕ ВЕБ-ИНТЕРФЕЙСА

Все страницы системы BI.Qube имеют похожую структуру и представлены в виде трёхколоночного макета. Левая колонка (1) содержит пункты главного меню, позволяющие осуществить переход на интересующую страницу программы. В средней части (2) размещается основной набор визуальных элементов, позволяющих увидеть все необходимые настройки. В большинстве случаев эта часть представлена в табличном виде. Редактирование осуществляется с использованием правой колонки (3), в которой размещается «скрываемое» окно свойств каждой строки таблицы (Рисунок. Макет типовой страницы).

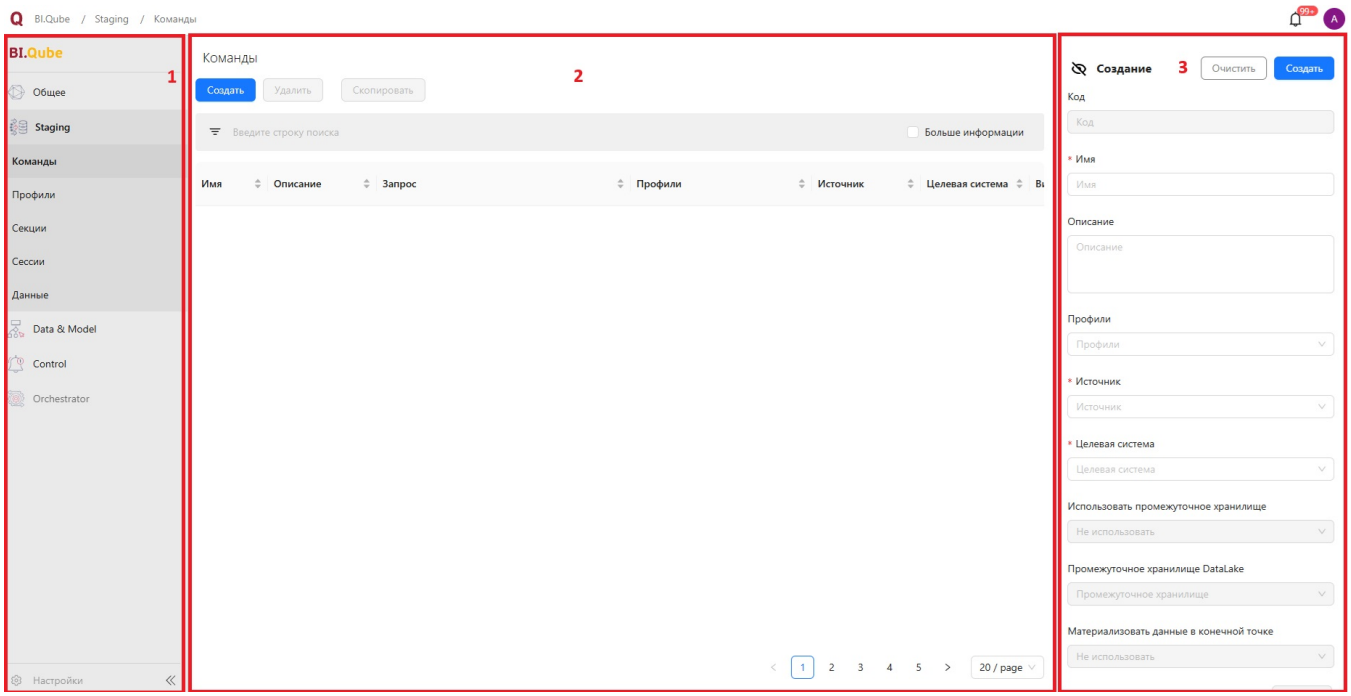


Рисунок. Макет типовой страницы

Переход по страницам программы осуществляется с использованием бокового меню. Наименования страниц имеют логичные названия и позволяют понять, какие настройки могут быть размещены на странице.

В процессе работы с системой могут возникать непредвиденные ошибки. Действия, которые не могут быть обработаны системой, генерируют ошибку. Текст ошибки отображается во всплывающем окне и дополнительно фиксируется в центре уведомлений. Центр уведомлений доступен на любой странице. Вызов осуществляется с помощью нажатия на иконку "звонка" ( ) в правом верхнем углу (Рисунок. Центр уведомлений).

Количество непрочитанных уведомления указывается рядом с иконкой "звонка" цифрами. Пометить всё как прочитанное можно нажатием на иконку с двойной галочкой ( ), очистить все уведомления – нажатием на иконку "кисти" ( ). При наведении курсора на иконку всплывает подсказка о её назначении. Для просмотра списка уведомлений необходимо воспользоваться колёсиком мыши или нажатием на серую полоску – бегунок справа окна свойств. Сортировка осуществляется с указанием даты и времени появления уведомления (Рисунок. Центр уведомлений).

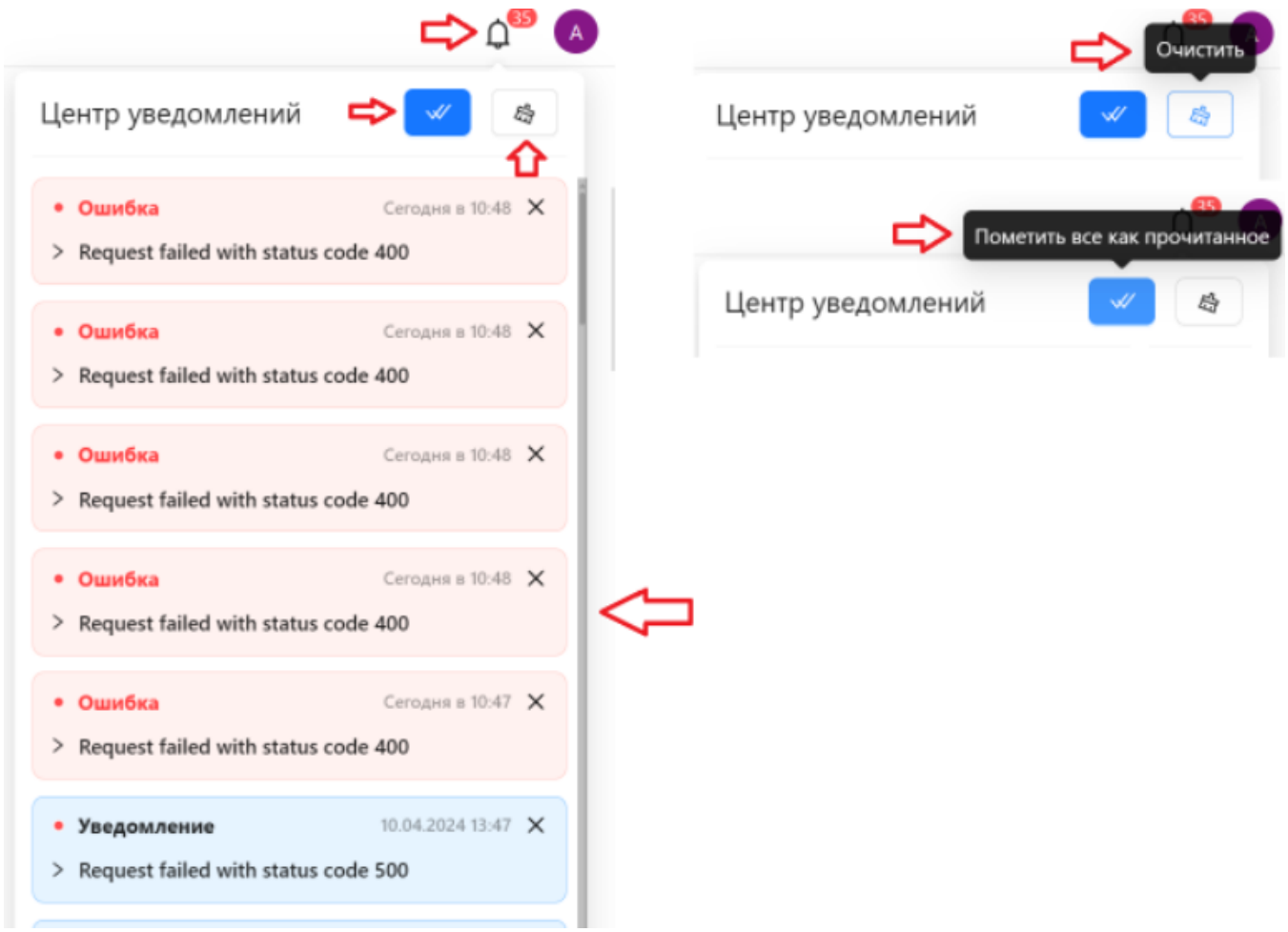


Рисунок. Центр уведомлений

Для лучшего визуального представления выделенные объекты таблицы подсвечивается цветом и шрифт меняет свой стиль на жирный (Рисунок. Выделение таблиц – визуальное отображение).

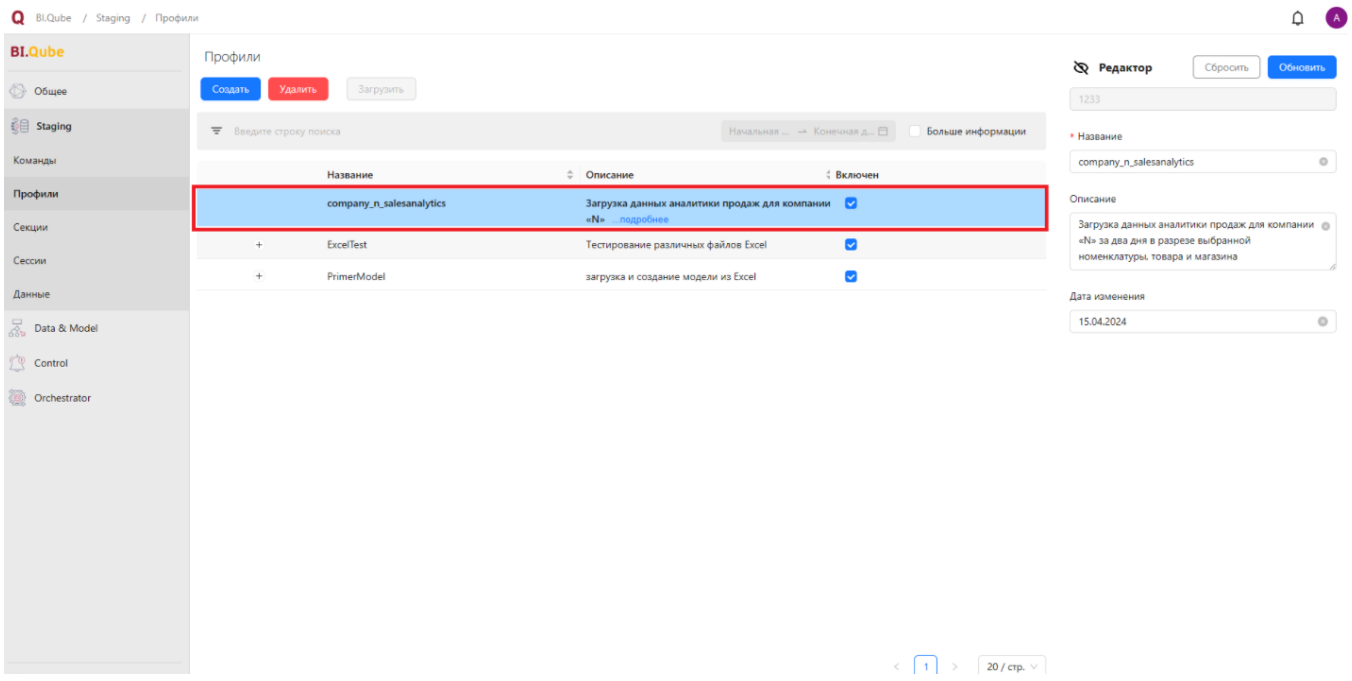


Рисунок. Выделение таблиц (сущностей) – визуальное отображение

Для удобства в интерфейсе есть функция для сворачивания бокового меню слева и окна свойств справа, реализованная с помощью двух кнопок указанных стрелками на рисунке ниже. Для возврата в исходное состояние необходимо снова нажать на указанные кнопки.

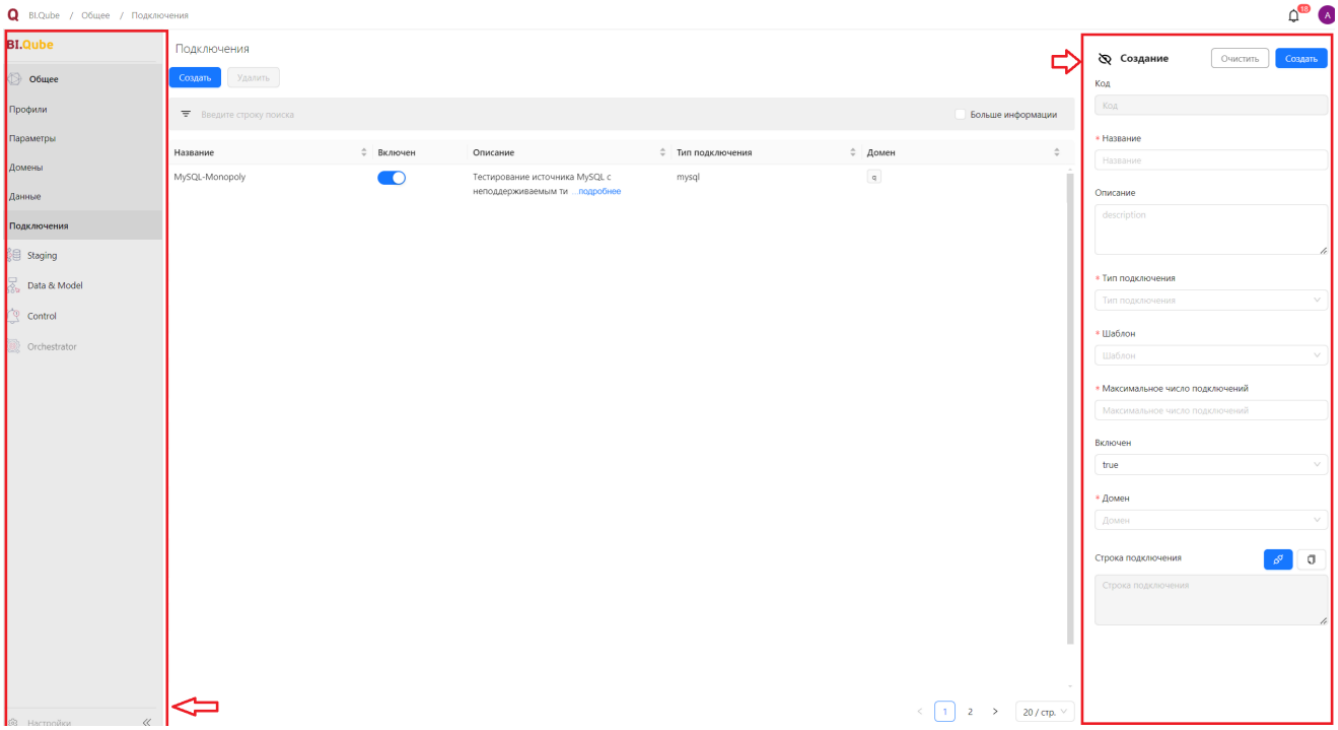


Рисунок. Отмеченные красным области можно свернуть

Смену роли пользователя или языка интерфейса можно осуществить в системном меню в правом верхнем углу

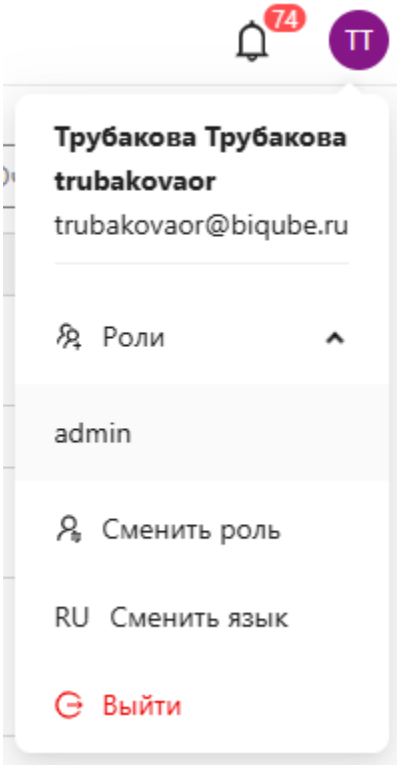


Рисунок. Выбор ролей/языка интерфейса

ПОРЯДОК РАБОТЫ С BI.QUBE

После развертывания системы и добавления пользователей в учетную систему keyklock пользователю в общем случае необходимо выполнить следующие действия.

1. Перейти в раздел "Общие":
    a. Создать "Домен" – для возможности ограничения прав доступа к создаваемым объектам.
    b. Создать "Профиль" – контейнер с командами системы, для возможности запуска их на выполнение.
    c. Создать "Подключения" – подключения как к источникам данных, так и к будущему хранилищу данных.
2. Перейти в раздел MetaStaging:
    a. Создать и настроить команду в подразделе "Команды".
    b. Проверить работоспособность запроса в создаваемой команде к точке подключения.
    c. Перейти на страницу "Профиль", выбрать нужный и запустить профиль на выполнение. Все команды в профиле будут выполнены, и по окончании появится сообщение о результатах.
    d. Перейти на страницу "Данные", выбрать подключение, соответствующее хранилищу данных, и убедиться, что все данные получены. После этих действий можно настроить регулярную загрузку данных с использованием оркестратора.
3. Перейти в раздел MetaVault:
    a. Создать модель данных, указать ее имя и точку подключения, где будет размещена создаваемая модель.
    b. Создать объект Metavault (чаще всего в терминологии Datavault это справочник), в процессе создания указать в качестве источника данных таблицу, в которую ранее были загружены данные.
    c. Выполнить операцию "Собрать", если все сделано верно, то создаться объект системы MetaVault и в него будут загружены данные.
    d. "Провалиться" в созданный объект и убедиться, что данные загружены в объект.
    e. После этих действий можно настроить регулярную загрузку данных с использованием оркестратора.
4. Перейти в раздел MetaControl:
    a. Создать "Список рассылки", указав ранее созданные подключения к почтовому серверу и/или к мессенджеру.
    b. Создать "Категорию" – объект, который позволяет группировать письма по каким-то признакам.
    c. Создать "Проверку" – для это следует настроить запрос и настроить оформления отчета.
    d. Перейти на страницу "Профиль" – выполнить профиль со всеми созданными проверками.
    e. Убедиться, что письма доставлены на указанные адреса, сообщения доставлены в мессенджер.
    f. После этих действий можно настроить регулярную загрузку данных с использованием оркестратора.

В процессе работы с системой последовательность действий может немного отличаться, в зависимости от задач, стоящих перед пользователем, так же в системе доступно много дополнительной функциональности, о которой подробно описано в данном руководстве.

METACOMMON

Общие сведения

Описание компонента

Группа команд Metacommon (Общие) – содержит команды, общие для всех компонентов.

Здесь доступны следующие страницы:

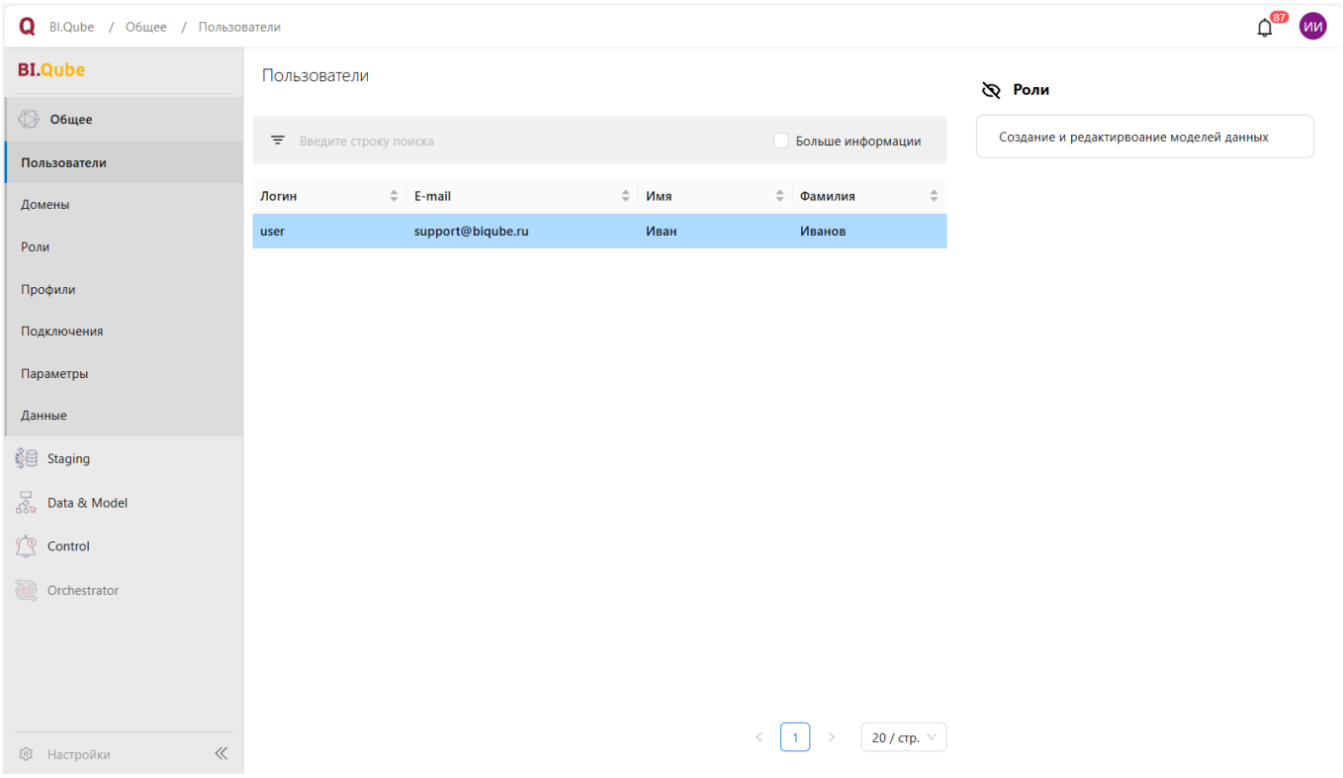
- **Пользователи** – на данной странице от отображается информация о текущем пользователе, если выполнена авторизация с ролью Пользователь. Или информация о всех пользователях, имеющих доступ к системе, если выполнена авторизация с ролью Администратор.
- **Домены** – на странице создаются и настраиваются имена подмножеств объектов системы, к которым будет предоставлен ролевой доступ. Например, доступ к модели данных, доступ к командам и так далее.
- **Роли** – на странице отображаются роли текущего пользователя, полученные из системы авторизации keycloak, а также их привязка к доменам. Для пользователя, авторизованного с ролью Пользователь, отображаются роли текущего пользователя. Для пользователя, авторизованного с ролью Администратор, отображаются все роли, которым предоставлен доступ в учетной системе keycloak.
- **Профили** – страница предназначена для создания объектов типа "Профиль", в которые группируются команды компонентов BI.Qube – контейнеров. Использование контейнеров позволяет группировать задачи и запускать их на выполнение в режиме параллельной обработки.
- **Подключения** – страница предназначена для создания подключений к источникам и получателям данных.
- **Данные** – страница Data (Данные) позволяет пользователю посмотреть визуально загруженные данные в хранилище. Здесь же есть возможность выполнить какие-то простые запросы, на основе которых можно убедиться в качестве полученных данных.
- **Параметры** – страница для настройки параметров – объектов, позволяющих автоматизировать ряд регулярных процессов, выполняемых системой.

ПОЛЬЗОВАТЕЛИ

После авторизации пользователю доступна возможность работать с системой в рамках разрешений, установленных его ролью. На странице Пользователи можно увидеть информацию о себе:

- Логин
- E-mail
- Имя
- Фамилия

если вся эта информация заполнена в системе keykloack. Выделив строку таблицы в зоне свойств, можно увидеть имя роли, с которой был осуществлен вход в систему. При этом, пользователю с ролью "Администратор" выводится список всех пользователей, которым в системе keykloack предоставлен доступ к BI. Qube. Здесь нужно помнить, если в keykloack пользователь добавлен в realm BIQube, но ни одна роль не назначена, то такой пользователь не войдет в систему.



ДОМЕНЫ

Домены - это именованные подмножества объектов различного типа (подключения, команды, и так далее), доступ к которым должен быть разграничен между многочисленными пользователями системы. Объекты, принадлежащие одному домену, доступны пользователю, к роли которого подключен этот домен. Пользователям, к ролям которых домен не подключен, объекты домена не доступны.

Домены может создавать любой пользователь. Созданные пользователем домены автоматически подключаются ко всем ролям текущего пользователя. Отключение домена от какой-то конкретной роли выполняется в разделе управления ролями текущего пользователя.

Страница Домены оформлена в типовом исполнении и выглядит как показано на рисунке ниже. Впервые развернутая система всегда содержит автоматически созданный домен "Default". Этот домен всегда доступен всем ролям пользователей, с которыми они авторизуются через систему keycloak.

Для создания нового домена необходимо нажать кнопку Создать "Create" и в правой части экрана заполнить следующие поля:

- Наименование – имя домена;
- Описание – краткое описание назначения домена.

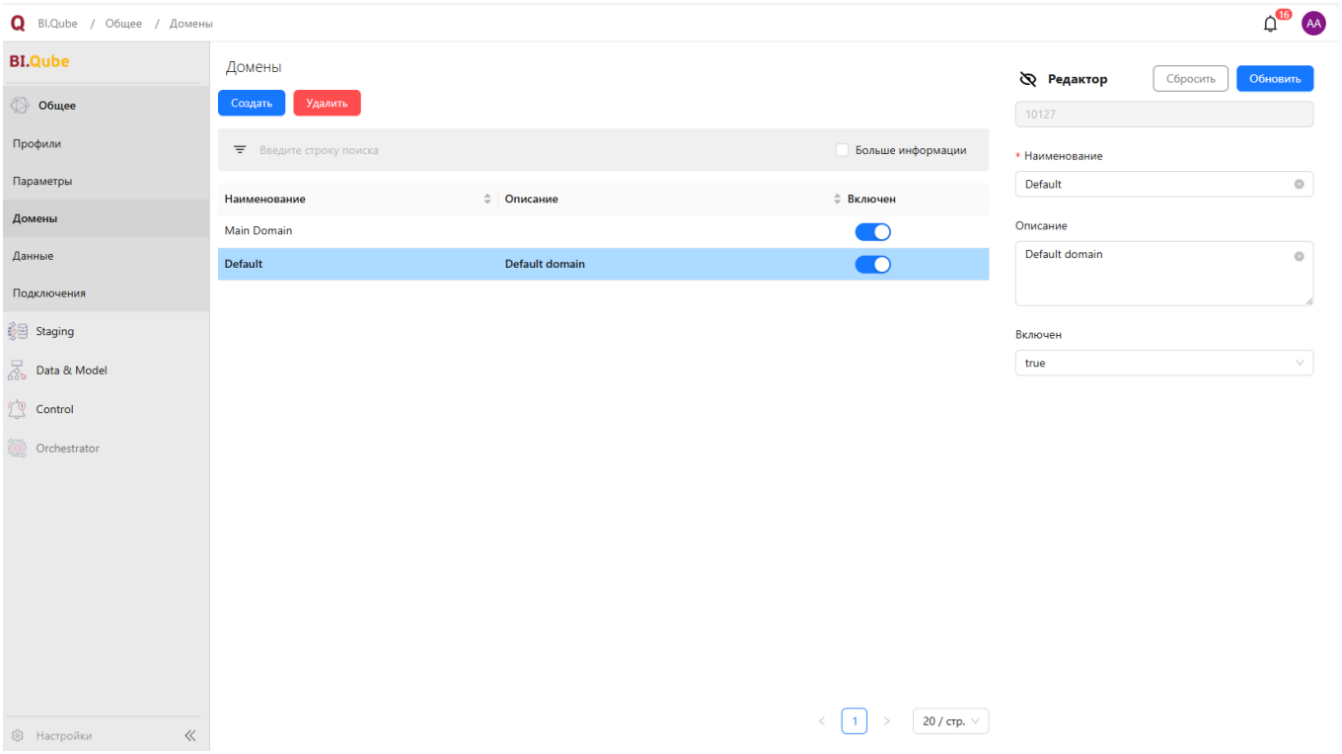
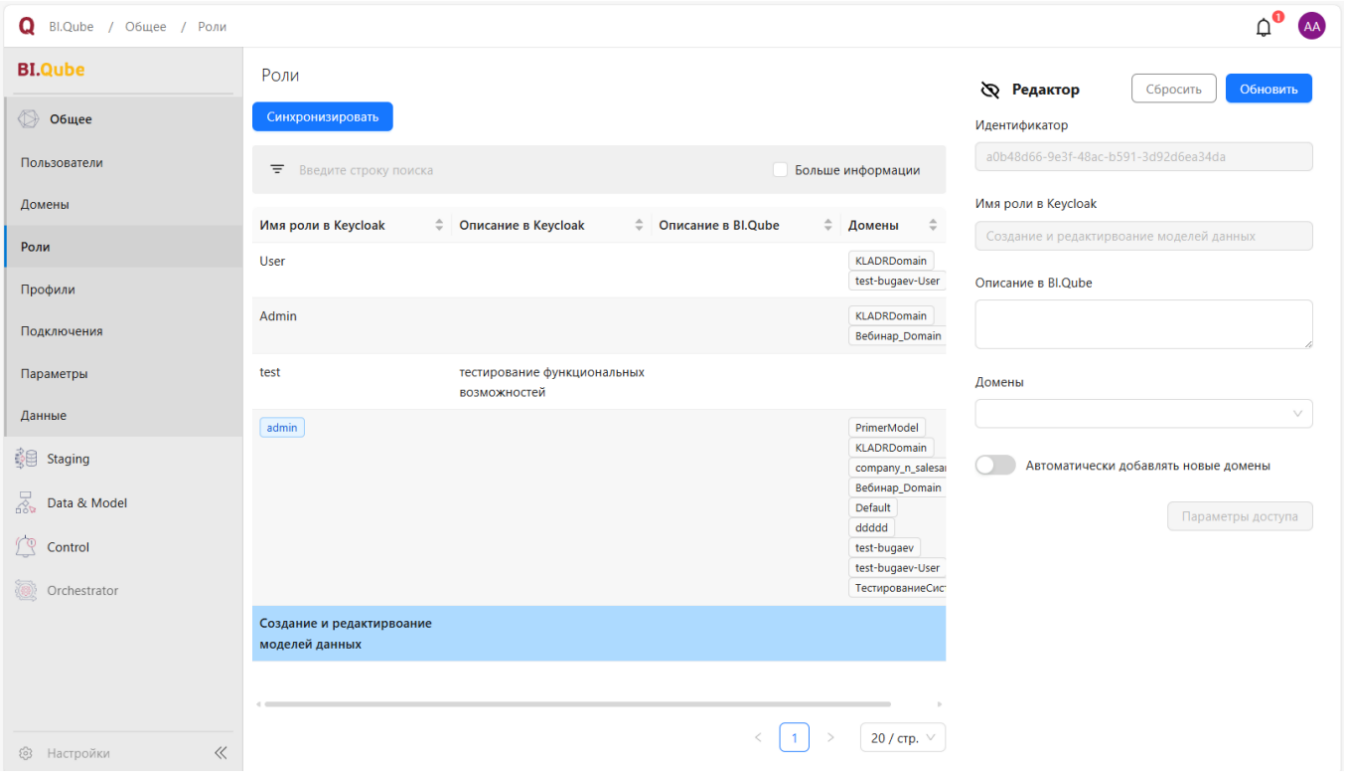


Рисунок. Страница Domains (Домены)

РОЛИ

Страница роли предназначена для контроля за своими ролями у каждого пользователя. Несмотря на то, что в в данный момент времени пользователь может находиться в системе под одной своей ролью, ему на этой странице видны все остальные доступные роли. Это сделано для того, чтобы новые создаваемые объекты пользователь мог привязывать ко всем своим ролям.

В случае, если пользователь авторизован с ролью администратор, то ему доступны все роли всех пользователей в системе. Это сделано для того, чтобы администратор мог предоставлять доступ к объектам системы любым ролям, которым необходим доступ.



В режиме администратор доступна кнопка "Синхронизировать". Она нужна в тех случаях, когда в системе keykloack создаются новые роли и назначаются пользователям. Чтобы новая роль была доступна в системе, администратор сначала должен выполнить синхронизацию.

Все домены, создаваемые пользователями, автоматически привязываются к текущей роли пользователя. Для подключения или отключения домена от роли необходимо выбрать интересующую роль и справа, с зоне свойств, в выпадающем списке "Домены" снять или поставить метку доступности домена в выбранной роли.

ПРОФИЛИ

Создание профиля выполняется на странице "Профили" (Profiles), страница предназначена для создания и редактирования профилей. Профили, созданные здесь доступны во всех компонентах и также отображаются на соответствующих страницах каждого компонента. Удаление профиля доступно только на странице "Профили" в разделе "Общие".

По умолчанию, в только что развернутой системе не создано ни одного профиля, система должна иметь хотя бы один профиль, в который будут сгруппированы команды, без профиля нет возможности запустить выполнение команд из веб интерфейса.

> ***Примечание:*** Не допускается использование кириллицы в наименованиях в случае, если планируется использовать интеграцию с Airflow!

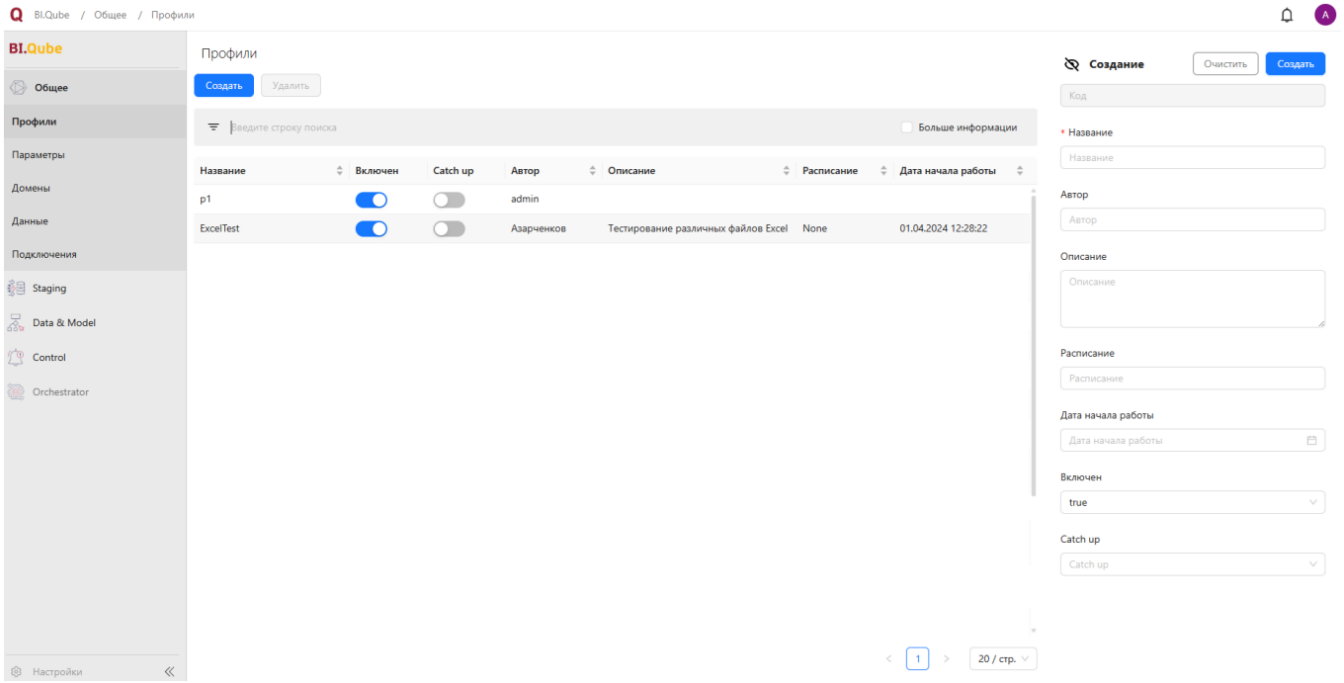
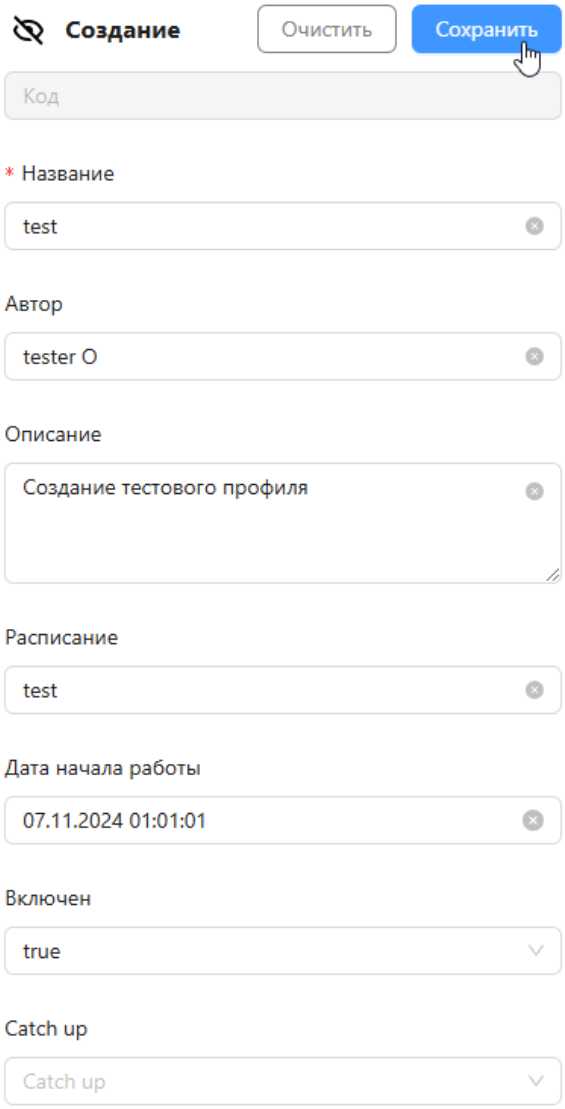


Рисунок. Страница Profiles (Профили)

Создание нового профиля

Для создания нового профиля необходимо нажать на кнопку "Создать" (Create). Справа появится (если оно ранее было скрыто) окно свойств (Рисунок. Заполнение свойств профиля), в котором необходимо заполнить следующие поля:

- Name (Название) – уникальное имя профиля, позволяющее отделять один профиль от другого, как правило, даётся осмысленное имя, поясняющее назначение профиля, не должно содержать пробелов;
- Author (Автор) – заполняется автоматически.
- Description (Описание) – расширенное описание назначения профиля (необязательное поле, введено для удобства пользователей);
- Schedule (Расписание) – это планировщик в формате, приемлемом для Airflow (когда запускать профиль);
- Enabled (Включен) – опция указывает на доступность профиля в системе (представлено в выпадающем списке двумя позициями: true/false);
- Catch up – планировщик по умолчанию запускает запуск DAG для любого интервала данных, который не запускался с момента последнего интервала данных (или был очищен). Эта концепция и называется Catch up;
- Start date (Дата создания) – автоматически создаваемое поле, содержит дату создания поля, при необходимости дата может быть отредактирована.



Поля, указанные со звёздочкой, обязательны для заполнения.

Рисунок. Заполнение свойств профиля

После заполнения всех обязательных полей необходимо нажать кнопку "Сохранить" (Save). Кнопка "Очистить" (Reset) очищает заполненные поля ввода в окне свойств.

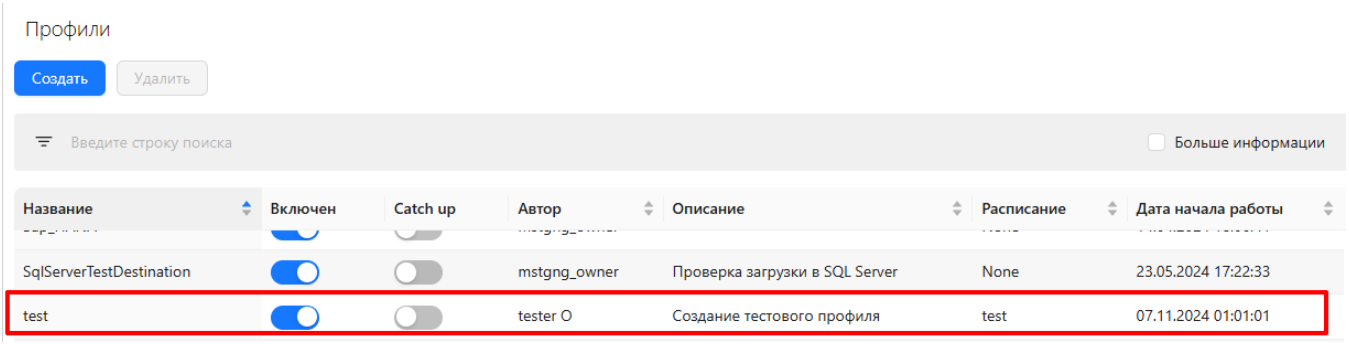
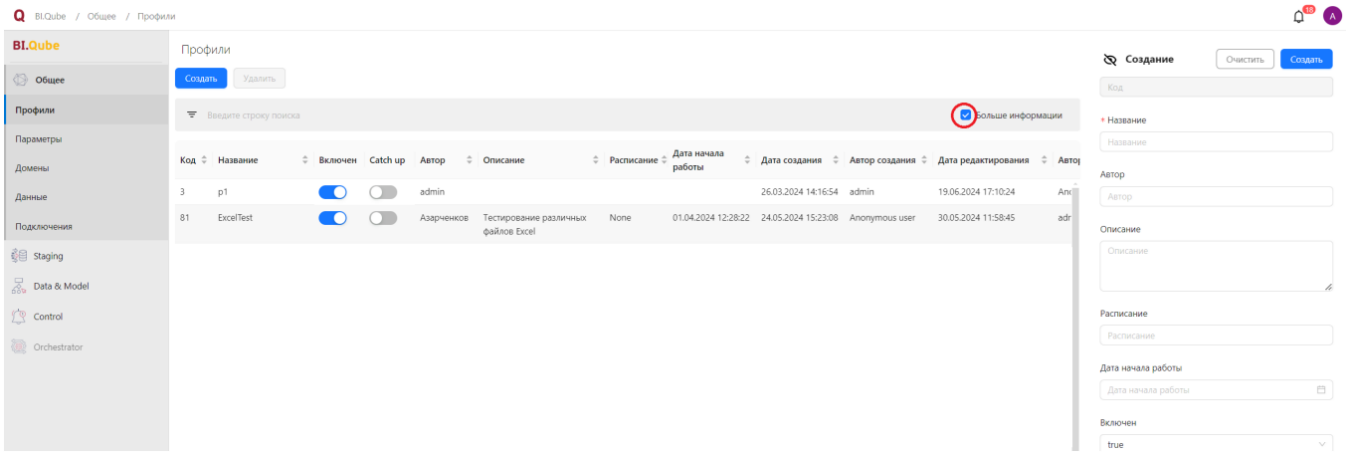


Рисунок. Результат создания тестового профиля

Кроме вышеперечисленных свойств, в базу данных системы автоматически попадают учётные данные о текущем авторизованном пользователе. В базе данных программы хранятся только сведения о последнем внесённом изменении. При редактировании профиля, история внесённых изменений не сохраняется. Дополнительную информацию можно увидеть, нажав на кнопку More info (Больше информации), в строке фильтров над таблицей, в основной части экрана.



Редактирование профиля

Для редактирования свойств профиля необходимо выполнить три шага:

Шаг 1: щёлкнуть левой кнопкой мыши по интересующей строке в таблице;

Шаг 2: внести необходимые изменения в поля свойств в правой части экрана;

Шаг 3: нажать кнопку "Обновить" (Update).

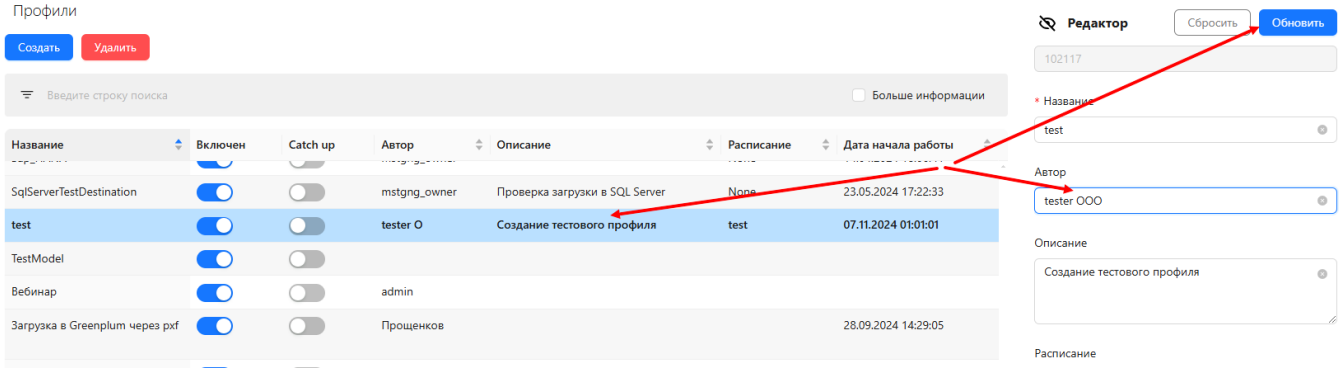
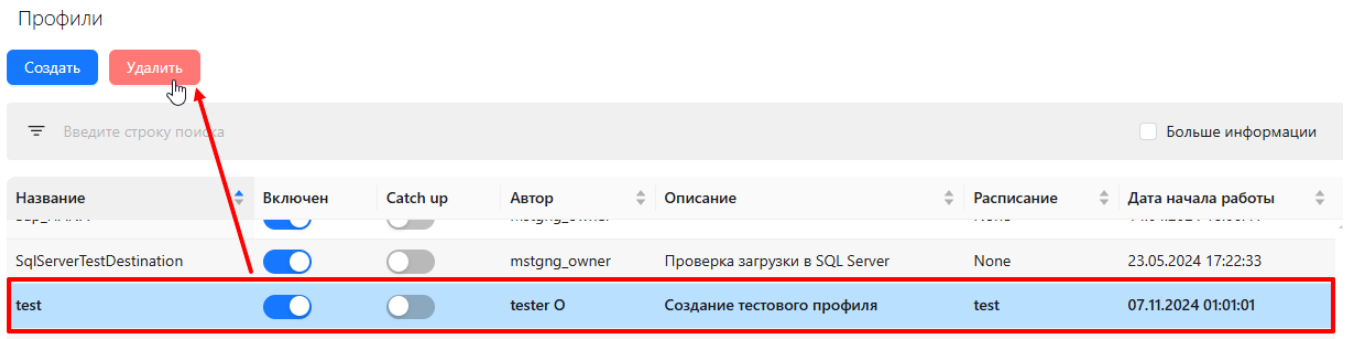


Рисунок. Редактирование свойств профиля

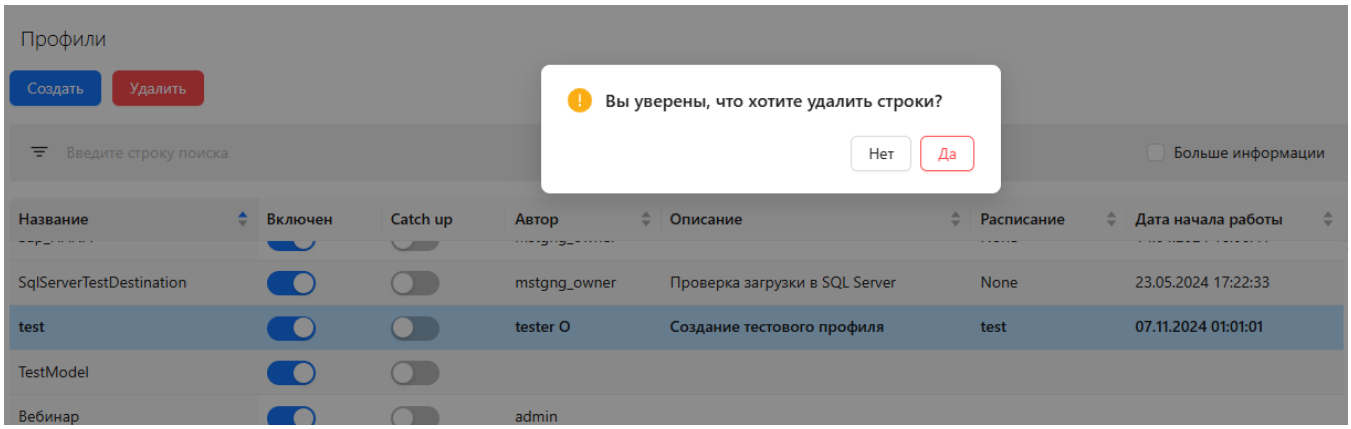
Удаление профиля

Для удаления профиля необходимо выполнить три шага:

Шаг 1: выбрать левой кнопкой мыши интересующую строку в таблице и в верхней панели нажать на кнопку "Удалить" (Delete);



Шаг 2: в появившемся модальном окне подтвердить намерение об удалении, нажав кнопку "Да".



ШАГ 3: проверить результат корректного удаления.

ПОДКЛЮЧЕНИЯ

Общие сведения

Подключения (Endpoints) – служат для установления связи BI.Qube с сервисами работы с данными (СУБД, веб-сервисы, файловые сервисы, каталоги с файлами, другие специализированные сервисы). Некоторые сервисы могут служить только как источник данных для других сервисов, некоторые могут являться и получателем данных. Возможности того или иного сервиса определяются прежде всего администратором этого сервиса и доступной функциональностью BI.Qube. Увидеть, какой сервис работы с данными может быть использован как источник и/или точка назначения, можно в визуальном интерфейсе при выборе шаблона подключения.

Система BI.Qube поддерживает возможность установки подключений к следующим типам сервисов работы с данными:

- Системы управления базами данных (таблицы и представления)
  - ClickHouse
  - PostgreSQL
  - GreenPlum
  - MS SQl Sqerver
  - Oracle
  - MySql
  - SAP Hana
  - другие СУБД на основе драйвера ODBC
- ВЕБ-сервисы(JSON , XML, CSV)
  - Сервисы предоставляющие доступ по протоколу REST API
  - Kafka
- Файловые сервисы (XLS, XLSX, XSV, XML, JSON)
  - Компьютер пользователя
  - Simple Storage Service (S3)
  - Общие папки windows (SMB)
  - YandeDisk
  - OneDrive
- 1С Предприятие (возможны ограничения по доступности к некоторым объектам конфигурации)
  - На базе СУБД MS SQL Server
  - На базе СУБД PostgreSQL
- Почтовые серверы
  - SMTP
- Мессенджеры
  - Telegram

Для всех поддерживаемых подключений реализована поддержка одного и более типа авторизации и/или способа установления связи между BI.Qube и сервисом работы с данными.

По умолчанию в системе не доступно ни одного подключения.

Рекомендации по работе с подключениями

Для возможности настройки извлечения данных из фалов, расположенных на локальных компьютерах пользователей, необходимо создать подключение к сервису S3 (к каждому источнику можно создавать сколько угодно подключений и ограничивать к ним доступ с использованием доменной политики безопасности).

BI.Qube автоматически распознает типы подключений, которые пользователь использует при решении своих задач и строит визуальный интерфейс под тип выбранный пользователем. Так, если пользователю необходимо создать команду извлечения данных из файлов, расположенных на компьютере пользователя, необходимо выбрать подключение к какому-либо сервису S3, система в интерфейсе создаваемой команды предложит сначала скопировать файл в сервис S3, а уже потом из этого сервиса отправит данные в точку назначения.

Общие настройки при создании подключений

Для создания нового подключения (endpoint) – подключения к источнику или создания точки назначения необходимо выбрать Подключения (endpoints) в боковом меню.

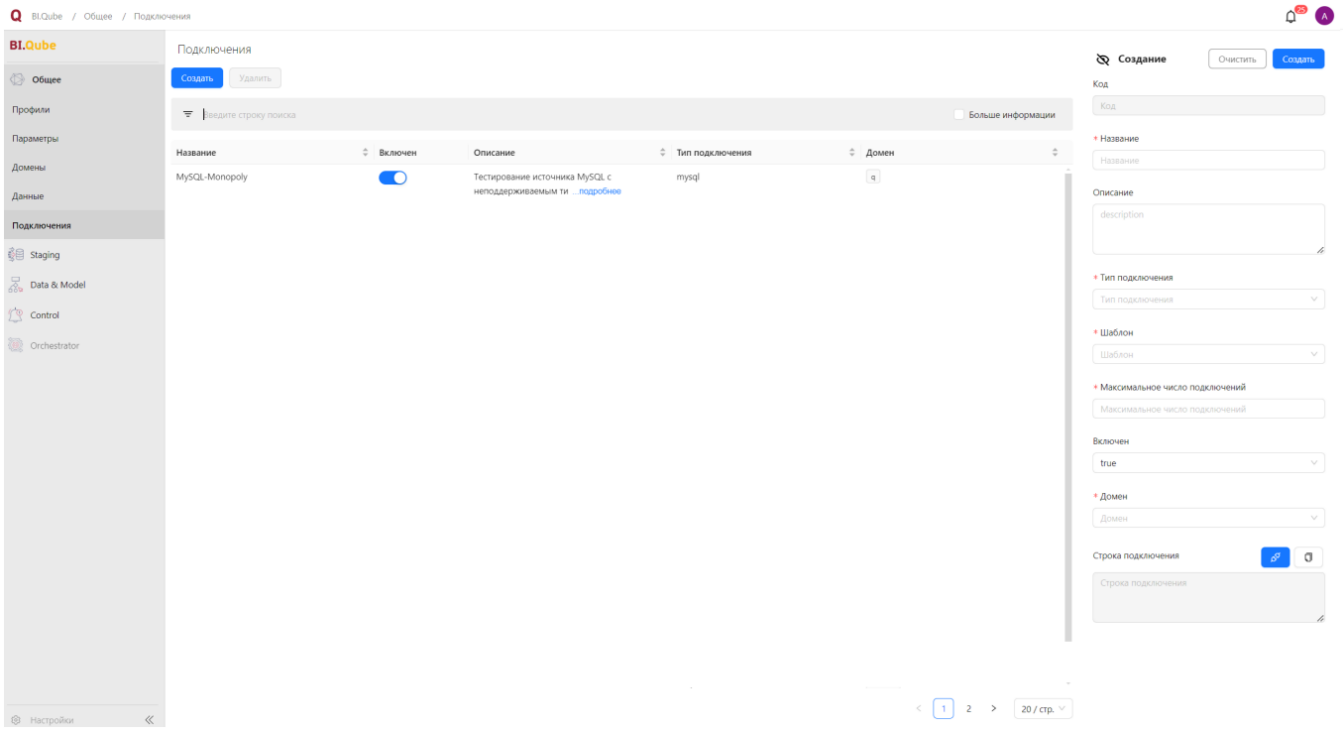


Рисунок. Страница создания подключений

Создание нового подключения осуществляется нажатием кнопки «Создать» (Create). С правой стороны экрана в окне свойств необходимо заполнить следующие поля:

- Code (Код) – уникальный идентификатор записи в базе данных, заполняется автоматически;
- Name (Название) – имя подключения, вводится без пробелов;
- Description (Описание) – бизнес описание подключения;
- Endpoint Type (Тип подключения) – источник данных СУБД, веб-сервис или другая система, к которой настроен коннектор в системе BI.Qube. BI.Qube содержит большой перечень коннекторов к различным источникам и типов источников;
- Template (Шаблон) – шаблон строки подключения, для выбранного типа подключения;
- Max connections (Максимальное число подключений) – максимальное количество одновременных подключений к источнику;
- Enabled (Состояние) – опция указывает на доступность подключения в системе;
- Domain (Домен) – выбор домена;
- Connection string (Строка подключения) – поле из которого можно скопировать автоматически сформированную строку подключения;
- TestConnection (Проверка подключения) – процедура проверки доступности подключения.

Поля, указанные со звёздочкой, обязательны для заполнения.

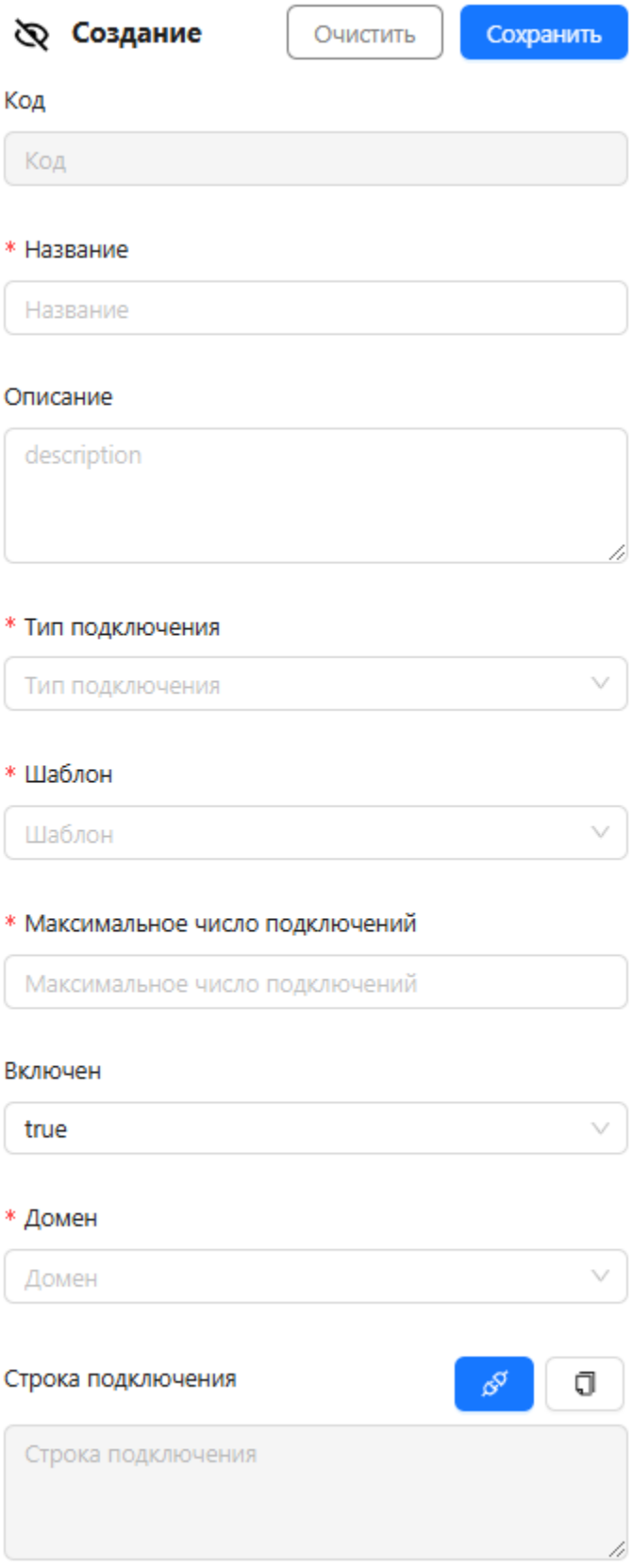
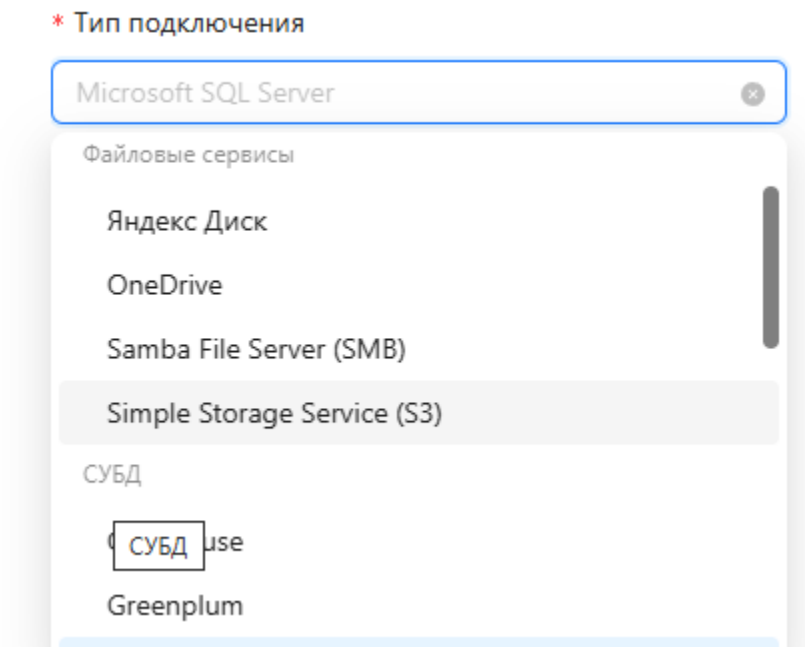
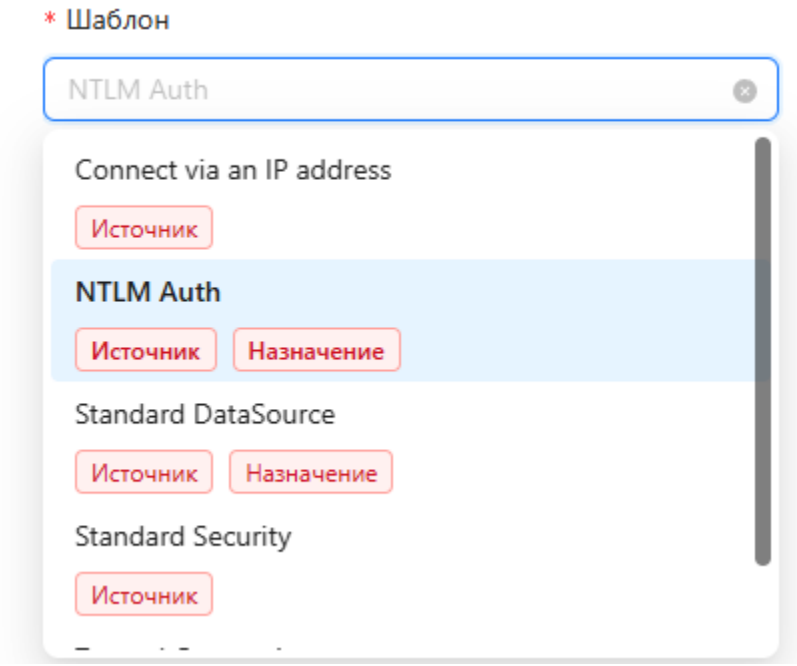


Рисунок. Поля для настройки подключений

В окне свойств справа в поле "Тип подключения" (Endpoint Type) для удобства осуществлена группировка по типам Подключений (Endpoint).



Каждому типу подключения (endpoints) соответствует один или более шаблон настроек. В зависимости от доступных пользователю данных для авторизации на стороне подключения (endpoints) выбирается подходящий шаблон.



В зависимости от выбранного типа подключения автоматически в интерфейсе появляются дополнительные поля, требующие заполнения.

Процедура проверки доступности подключения представлена с помощью кнопки "Проверить подключение" (TestConnection).

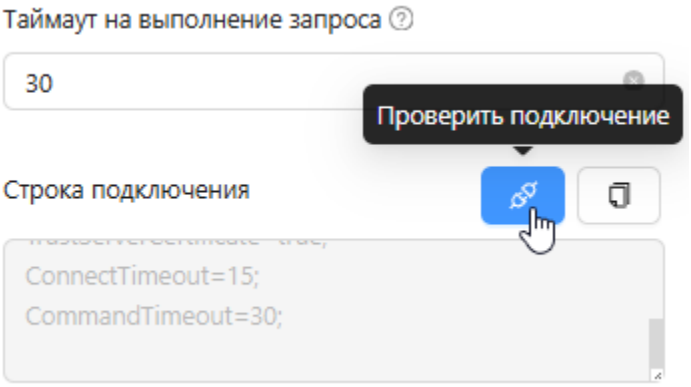


Рисунок. Кнопка "Проверить подключение".

По клику на кнопку происходит проверка подключения, в результате которой появляется сообщение в модальном окне:

- "Не удалось подключиться к конечной точке" – в случае неудачного подключения;
- "Подключение успешно" – в случае успешного подключения.

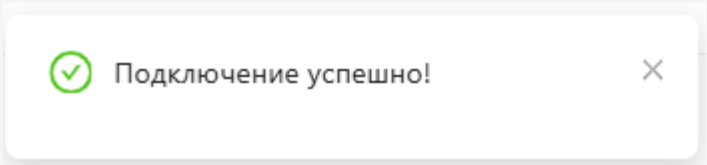
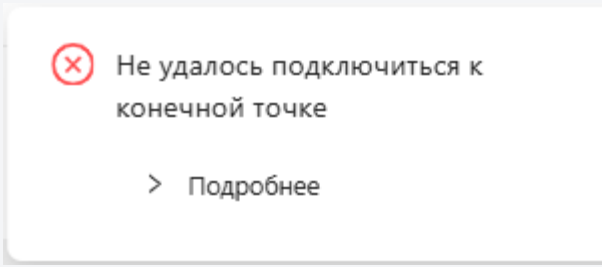
Подключение успешно	Подключение не удалось
	

Рисунок. Состояния подключения.

Кнопка "Скопировать" (Copy) позволяет скопировать автоматически сформированную строку подключения.

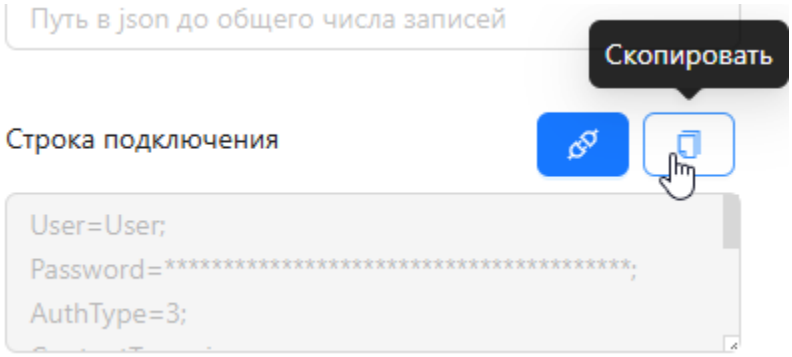


Рисунок. Кнопка "Скопировать".

Редактирование созданного подключения

Для редактирования уже созданного подключения необходимо выполнить три шага:

Шаг 1: щёлкнуть левой кнопкой мыши по интересующей строке в таблице;

Шаг 2: внести необходимые изменения в поля свойств в правой части экрана;

Шаг 3: нажать кнопку "Обновить" (Update).

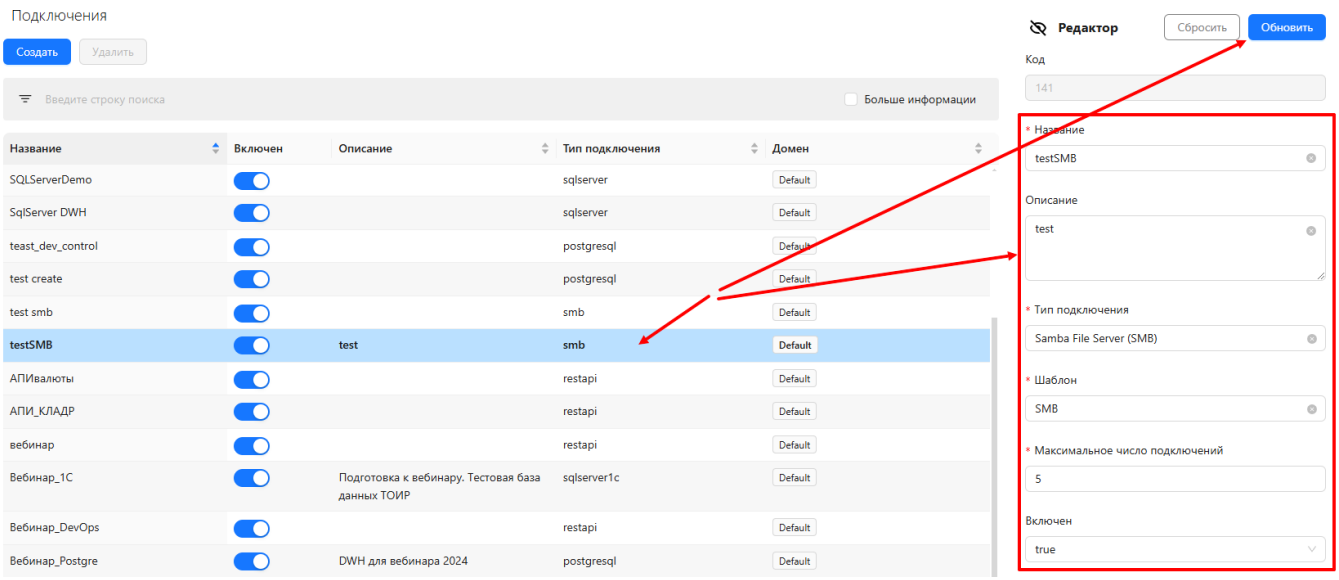
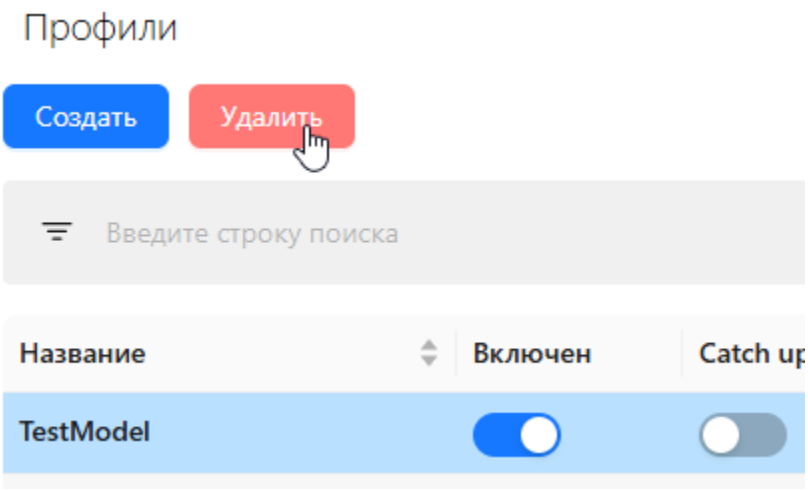


Рисунок. Редактирование созданного подключения

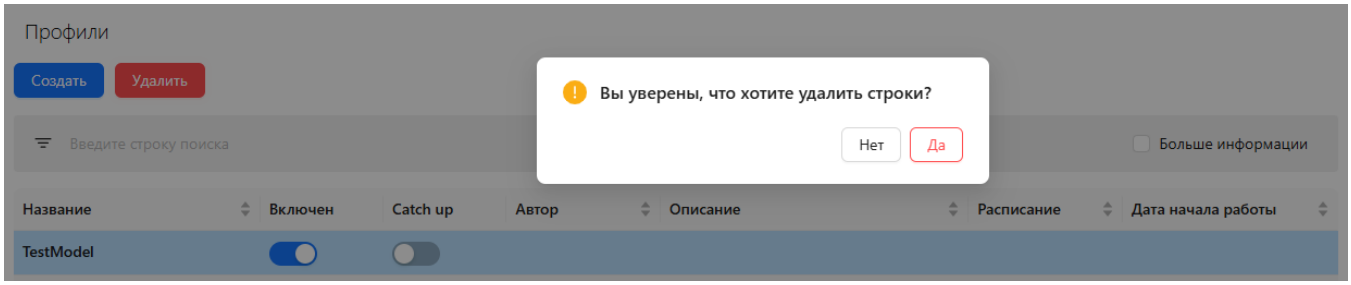
Удаление созданного подключения

Для удаления ранее созданного подключения необходимо выполнить три шага:

Шаг 1: выбрать левой кнопкой мыши интересующую строку в таблице и в верхней панели нажать на кнопку "Удалить" (Delete);



Шаг 2: в появившемся модальном окне подтвердить намерение об удалении, нажав кнопку "Да";



ШАГ 3: проверить результат удаления.

Файловые сервисы

Файловый сервер (ФС) – это выделенный компьютер или устройство в сети, которое предоставляет централизованное хранилище и файловые службы другим устройствам в такой сети. Основное назначение файлового сервера – хранение и защита информации, авторизация доступа и совместное использование файлов между несколькими клиентами по сети. Наличие ФС устраняет необходимость в отдельном локальном хранилище файлов на каждом компьютере.

Простыми словами, файловый сервер – это один или несколько физических компьютеров, ресурсы которых выделены для хранения файлов.

SMB

SMB (сокр. от англ. Server Message Block) – сетевой протокол прикладного уровня для удалённого доступа к файлам, принтерам и другим сетевым ресурсам, а также для межпроцессного взаимодействия (для Windows). А SMB (сокр. от англ. Samba File Server) – это набор софта для Linux. Эти два понятия синонимы, различия только в том, на какой операционной системе они работают.

Для типа подключения **SMB** доступен один вариант строки подключения (шаблонов). В интерфейсе выбор нужного шаблона осуществляется в поле Template (Шаблон).

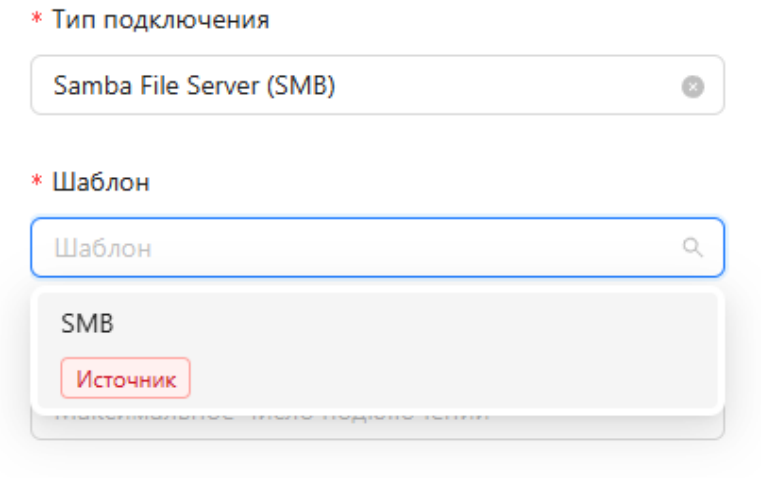


Рисунок. Вариант шаблона для типа подключения SMB.

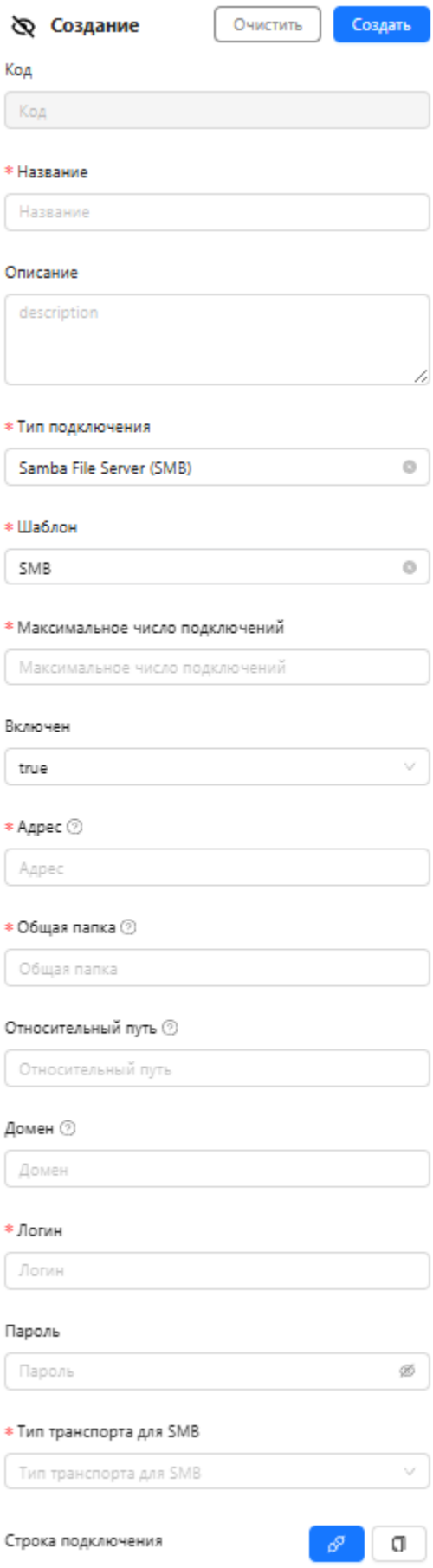

Создание
Очистить
Создать
Код
Код
* Название
Название
Описание
description
* Тип подключения
Samba File Server (SMB)
* Шаблон
SMB
* Максимальное число подключений
Максимальное число подключений
Включен
true
* Адрес
Адрес
* Общая папка
Общая папка
Относительный путь
Относительный путь
Домен
Домен
* Логин
Логин
Пароль
Пароль
* Тип транспорта для SMB
Тип транспорта для SMB
Строка подключения

Рисунок. Строки заполнения шаблона

При выборе данного шаблона в окне свойств появляются следующие строки для заполнения (Рисунок. Строки заполнения шаблона):

- Endpoint Type (Тип подключения) – SMB;
- Template (Шаблон) – smb (выбирается в зависимости от настроек endpoint);
- Max connections (Максимальное число подключений) – максимальное количество одновременных подключений к источнику;
- Enabled (Включен) – опция указывает на доступность подключения в системе (представлено в выпадающем списке двумя позициями: true/false);
- Host (Адрес) – указывается сетевой адрес, где размещен endpoint;
- Relative path (Относительный путь) – путь относительно общей папки. Если нужен корень – не указывать ничего. В конце добавлять слэш;
- Folder (Общая папка) – папка с общим доступом (та, к которой предоставлен общий доступ, указывать без слэшей);
- Domain (Домен) – нужно указывать, если в имени пользователя, которому доступны данные в endpoint, указан домен. В других случаях это поле заполнять не нужно;
- Login (Логин) – указывается имя пользователя, которому доступны данные в endpoint;
- Password (Пароль) – пароль пользователя (сохраняется в базе данных системы и защищен от злоумышленников);
- SMB transport type (Тип транспорта для SMB) – можно выбрать из выпадающего списка.

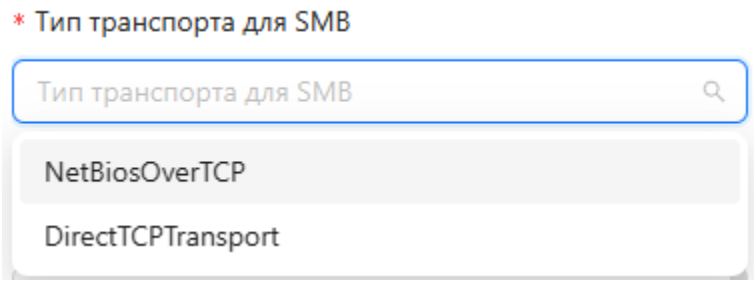


Рисунок. Тип транспорта для SMB

  - NetBIOS over TCP/IP (NetBT) – это адаптация протокола NetBIOS для работы в сетях TCP/IP. Эта адаптация позволяет использовать услуги NetBIOS в сетях IP, расширяя функциональность NetBIOS за пределы локальных сетевых границ.
  - Direct TCP Tansport – (рекомендуется для современных систем) обладает более совершенными механизмами безопасности, такие как сквозное шифрование и алгоритм Advanced Encryption Standard (AES).

- Connection string (Строка подключения) – здесь создаётся строка подключения из заполненных полей выше: Host=/*{Host}*/; Folder=/*{Folder}*/; Domain=/*{Domain}*/; Login=postgres; Password=***; SmbTransportType=/*{SmbTransportType}*/;.

Поля, указанные со звёздочкой, обязательны для заполнения.

S3 (Simple Storage Service)

S3 (Simple Storage Service) – сервис для хранения цифровых данных большого объёма. Работает по одноимённому протоколу.

Это вариант «плоского» (не иерархического) хранилища. С точки зрения системы все объекты равнозначны, поэтому в S3-хранилище удобно долго хранить разнородную информацию и быстро получать к ней доступ.

Для данного типа источника доступные шаблоны подключения приведены в одноименном поле. После выбора подходящего шаблона необходимо заполнить появившиеся поля:

- Endpoint Type (Тип подключения) – файловое хранилище S3;
- Template (Шаблон) – выбрать шаблон строки подключения;
- Access Key (Ключ доступа) – ввести ключ доступа к бакету файлового хранилища;
- Secret Key (Секретный ключ) – секретный ключ;
- Bucket Name (Имя бакета данных) – имя бакета, к которому настраивается доступ;
- IP Endpoint (Адрес сервера) – адрес сервера, где размещено файловое хранилище;
- Region (Регион) – указать, для какого региона выполнены настройки в файловом хранилище.

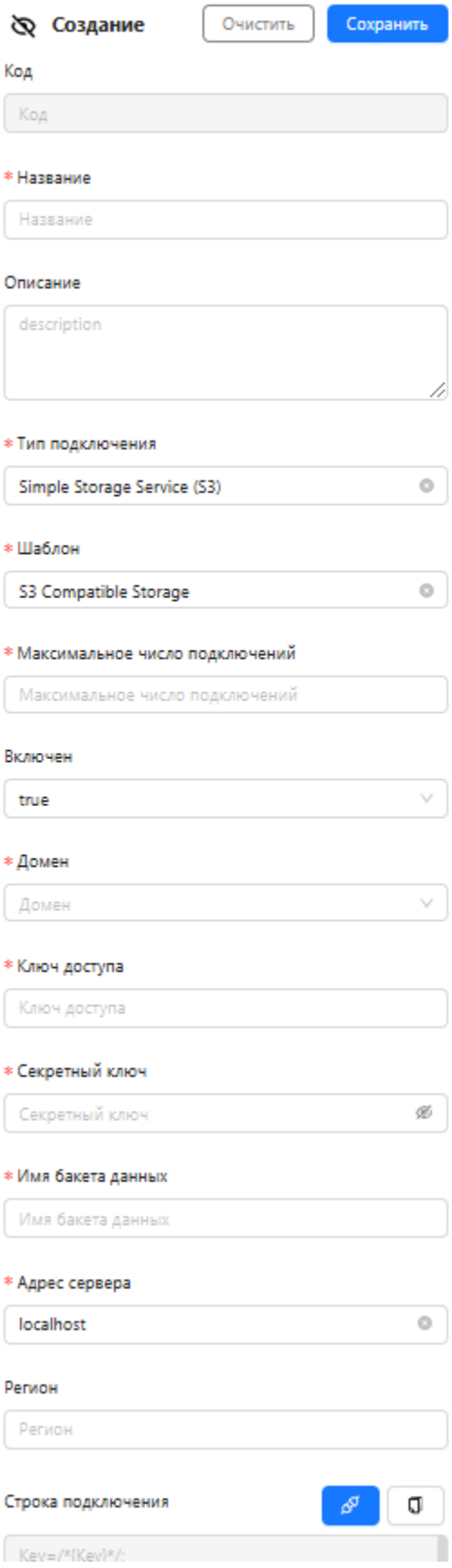


Рисунок. Endpoint Type (Тип подключения) - S3

После создания подключения можно переходить к следующему шагу, например просмотру данных доступных в источнике или перейти к другим компонентам системы.

Поля, указанные со звёздочкой, обязательны для заполнения.

СУБД

СУБД (система управления базами данных) – это комплекс программ, позволяющих создать базу данных и манипулировать данными (вставлять, обновлять, удалять и выбирать).

Система обеспечивает быстродействие, безопасность данных, простоту получения и обновления данных, многопользовательский доступ и способность хранить большое количество данных, а также предоставляет средства для администрирования БД (баз данных).

Бывают реляционные, нереляционные, графовые, документоориентированные, колоночные (столбцовые), базы данных key-value, сетевые и иерархические.

Примеры СУБД: **Oracle, MySQL, Microsoft SQL Server, PostgreSQL**.

Основные функции СУБД:

- управление данными во внешней памяти (на дисках);
- управление данными в оперативной памяти с использованием дискового кэша;
- журнализация изменений (сохранение истории), резервное копирование и восстановление базы данных после сбоев;
- поддержка языков БД (язык определения данных, язык манипулирования данными).

Каждая СУБД основывается на какой-либо модели данных, это является одним из признаков классификации.

Microsoft SQL Server

Microsoft SQL Server (MSSQL) – это система управления реляционными базами данных (СУБД), используемая для хранения и извлечения данных из других программных приложений. Microsoft разработала это программное обеспечение для управления информацией на нескольких компьютерах в одной сети. Используя язык программирования SQL (**S**tructured **Q**uery **L**anguage – «язык структурированных запросов»), SQL Server может выполнять аналитику и обработку транзакций, а также работу с информацией.

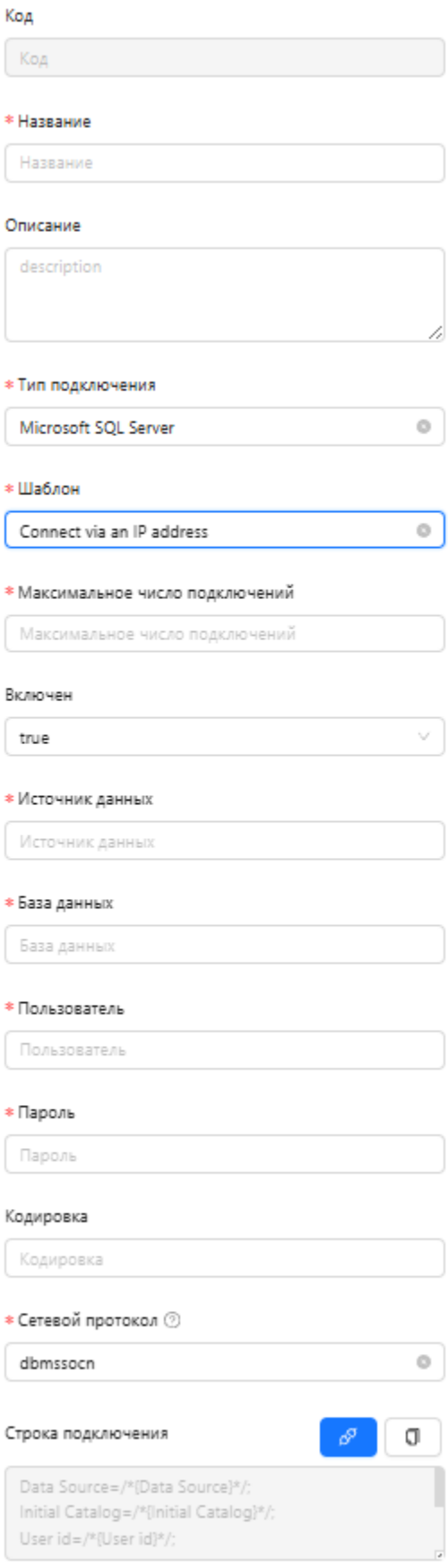


Рисунок. Пример полей для заполнения для Microsoft SQL Server с шаблоном Connect via an IP address

Поля, указанные со звёздочкой, обязательны для заполнения.

Для типа подключения **SQL Server** доступны несколько вариантов строк подключения (шаблонов). В интерфейсе выбор нужного шаблона осуществляется в поле Template (Шаблон).

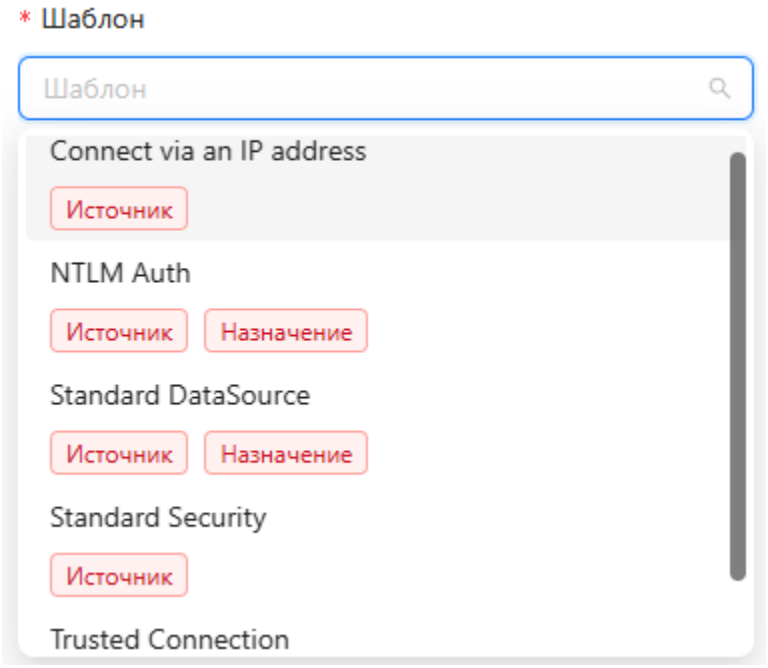


Рисунок. Тип подключения SQL Server и шаблоны

Примеры шаблонов:

Connect via an IP address

- Endpoint Type (Тип подключения) – sqlserver;
- Template (Шаблон) – Connect via an IP address (выбирается в зависимости от настроек endpoint);
- Max connections (Максимальное число подключений);
- Поле Enabled (Включён) – опция указывает на доступность подключения в системе (представлено в выпадающем списке двумя позициями: true/false);
- Data Source (Источник данных) – указывается адрес сервера, где размещен endpoint;
- Initial Catalog (База данных) – указывается база данных;
- User id (Пользователь) – указывается имя пользователя, которому доступны данные в endpoint;
- Password (Пароль) – пароль пользователя (сохраняется в базе данных системы и защищен от злоумышленников);
- TrustServerCertificate (Доверять сертификату сервера) – параметр, влияющий на проверку сертификата в зависимости от выбранного значения: true /false;
- Network Protocol (Сетевой протокол) выбирается из выпадающего списка;

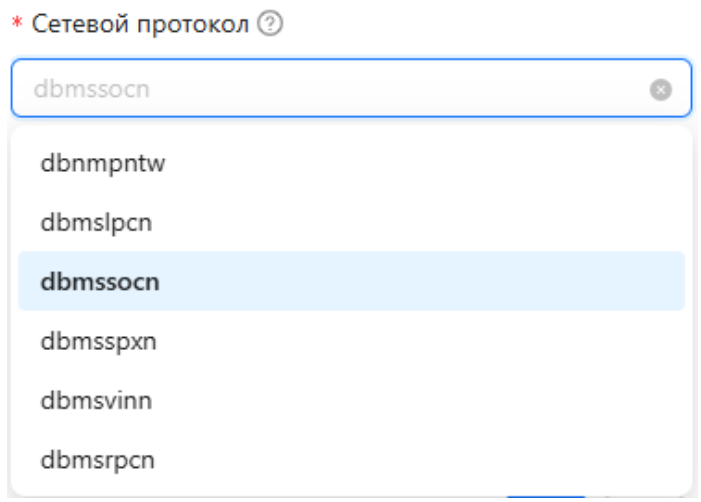


Рисунок. Выдающий список поля Network Protocol (Сетевой протолок)

- Connection string (Строка подключения) – здесь создаётся строка подключения из заполненных полей выше. Пример: User id=sqlserver; Password=***; TrustServerCertificate=/*{TrustServerCertificate}*/; Data Source=192,168,128,1; Initial Catalog=sqlserver;.

NTLM Auth

- Endpoint Type (Тип подключения) – sqlserver;
- Template (Шаблон) – NTLM Auth (выбирается в зависимости от настроек endpoint);
- Max connections (Максимальное число подключений);
- Поле Enabled (Включён) – опция указывает на доступность подключения в системе (представлено в выпадающем списке двумя позициями: true/false);
- Data Source (Источник данных) – указывается адрес сервера, где размещен endpoint;
- Initial Catalog (База данных) – указывается база данных;
- User id (Пользователь) – указывается имя пользователя, которому доступны данные в endpoint;
- Password (Пароль) – пароль пользователя (сохраняется в базе данных системы и защищен от злоумышленников);
- TrustServerCertificate (Доверять сертификату сервера) – параметр, влияющий на проверку сертификата в зависимости от выбранного значения: true /false;
- Connection string (Строка подключения) – здесь создаётся строка подключения из заполненных полей выше. Пример: User id=sqlserver; Password=***; TrustServerCertificate=/*{TrustServerCertificate}*/; Server=localhost; Database=/*{Database}*/;.

Standart DataSource

- Endpoint Type (Тип подключения) – sqlserver;
- Template (Шаблон) – Standart DataSource (выбирается в зависимости от настроек endpoint);
- Max connections (Максимальное число подключений);
- Поле Enabled (Включён) – опция указывает на доступность подключения в системе (представлено в выпадающем списке двумя позициями: true/false);
- Data Source (Источник данных) – указывается адрес сервера, где размещен endpoint;
- Initial Catalog (База данных) – указывается база данных;
- User id (Пользователь) – указывается имя пользователя, которому доступны данные в endpoint;
- Password (Пароль) – пароль пользователя (сохраняется в базе данных системы и защищен от злоумышленников);
- TrustServerCertificate (Доверять сертификату сервера) – параметр, влияющий на проверку сертификата в зависимости от выбранного значения: true /false;
- Connection string (Строка подключения) – здесь создаётся строка подключения из заполненных полей выше. Пример: User id=sqlserver; Password=***; TrustServerCertificate=/*{TrustServerCertificate}*/; Server=localhost; Database=/*{Database}*/;.

Standart Security

- Endpoint Type (Тип подключения) – sqlserver;
- Template (Шаблон) – Standart Security (выбирается в зависимости от настроек endpoint);
- Max connections (Максимальное число подключений);
- Поле Enabled (Включён) – опция указывает на доступность подключения в системе (представлено в выпадающем списке двумя позициями: true/false);
- Server (Источник данных) – указывается адрес сервера, где размещен endpoint;
- Database (База данных) – указывается база данных;
- User id (Пользователь) – указывается имя пользователя, которому доступны данные в endpoint;
- Password (Пароль) – пароль пользователя (сохраняется в базе данных системы и защищен от злоумышленников);
- Trust Server Certificate (Доверять сертификату сервера) – параметр, влияющий на проверку сертификата в зависимости от выбранного значения: true /false;
- Connection string (Строка подключения) – здесь создаётся строка подключения из заполненных полей выше. Пример: User id=sqlserver; Password=***; TrustServerCertificate=/*{TrustServerCertificate}*/; Data Source=192,168,128,1; Initial Catalog=sqlserver;.

Trusted Connection

- Endpoint Type (Тип подключения) – sqlserver;
- Template (Шаблон) – Trusted Connection (выбирается в зависимости от настроек endpoint);
- Max connections (Максимальное число подключений);
- Поле Enabled (Включён) – опция указывает на доступность подключения в системе (представлено в выпадающем списке двумя позициями: true/false);
- Server (Источник данных) – указывается адрес сервера, где размещен endpoint;
- Database (База данных) – указывается база данных;
- Connection string (Строка подключения) – здесь создаётся строка подключения из заполненных полей выше. Пример: User id=sqlserver; Password=***; TrustServerCertificate=/*{TrustServerCertificate}*/; Server=localhost; Database=/*{Database}*/;.

MySQL

MySQL – это реляционная система управления базами данных (СУБД) с открытым исходным кодом, позволяющая хранить, организовывать большие объёмы данных, и манипулировать ими. Использует стандартный язык SQL для обработки данных (Рисунок. Endpoint Type (Тип подключения) – MySQL).

Для типа подключения **MySQL** доступны несколько вариантов строк подключения (шаблонов). В интерфейсе выбор нужного шаблона осуществляется в поле Te mplate (Шаблон) (Рисунок. Варианты шаблонов для подключения mysql).

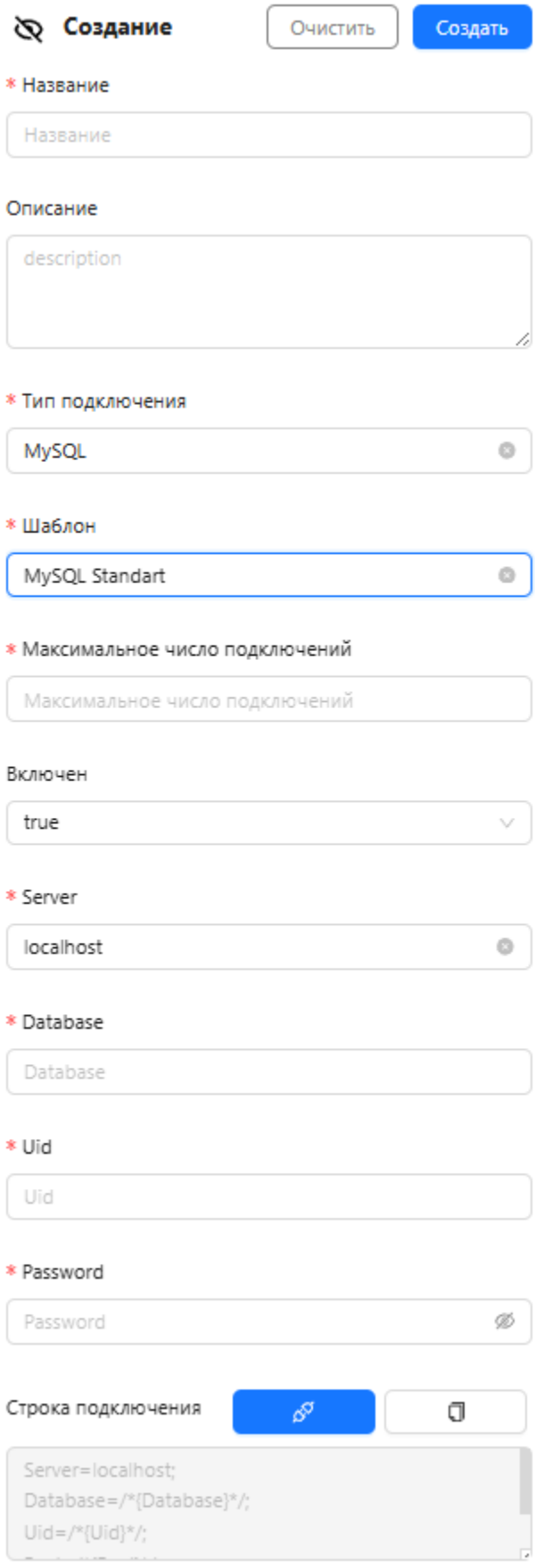

Создание
Очистить
Создать
* Название
Название
Описание
description
* Тип подключения
MySQL
* Шаблон
MySQL Standart
* Максимальное число подключений
Максимальное число подключений
Включен
true
* Server
localhost
* Database
Database
* Uid
Uid
* Password
Password
Строка подключения
Server=localhost;
Database=/*{Database}*/;
Uid=/*{Uid}*/;

Рисунок. Endpoint Type (Тип подключения) – MySQL

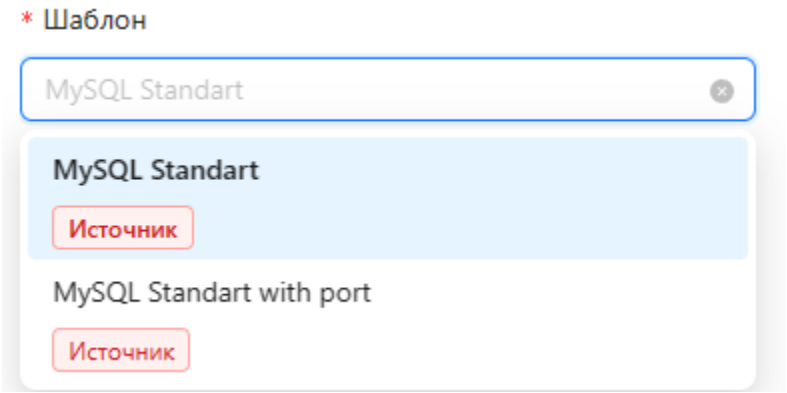


Рисунок. Варианты шаблонов для подключения mysql.

Примеры шаблонов:

MySQL Standart

- Endpoint Type (Тип подключения) – mysql;
- Template (Шаблон) – MySQL Standart (выбирается в зависимости от настроек endpoint);
- Max connections (Максимальное число подключений);
- Поле Enabled (Включён) – опция указывает на доступность подключения в системе (представлено в выпадающем списке двумя позициями: true/false);
- Server (Адрес сервера) – указывается сетевой адрес, где размещен endpoint;
- Database (База данных) – указывается имя базы данных;
- Uid (Пользователь) – указывается имя пользователя, которому доступны данные в endpoint;
- Password (Пароль) – пароль пользователя (сохраняется в базе данных системы и защищен от злоумышленников);
- Connection string (Строка подключения) – здесь создаётся строка подключения из заполненных полей выше. Пример: Server=localhost; Database=/*{Database}*/; Uid=postgres; Pwd=***;.

MySQL Standart with port

- Endpoint Type (Тип подключения) – mysql;
- Template (Шаблон) – MySQL Standart with port (выбирается в зависимости от настроек endpoint);
- Max connections (Максимальное число подключений);
- Поле Enabled (Включён) представлено в выпадающем списке двумя позициями: true/false;
- Server (Адрес сервера) – указывается сетевой адрес, где размещен endpoint;
- Port (Порт) – указывается порт, по которому доступен endpoint;
- Database (Имя базы данных) – указывается имя базы данных;
- User (Пользователь) – указывается имя пользователя, которому доступны данные в endpoint;
- Password (Пароль) – пароль пользователя (сохраняется в базе данных системы и защищен от злоумышленников);
- Connection string (Строка подключения) – здесь создаётся строка подключения из заполненных полей выше. Пример: Server=localhost; Database=/*{Database}*/; Uid=postgres; Pwd=***;.

Поля, указанные со звёздочкой, обязательны для заполнения.

Oracle

Oracle Database – это объектно-реляционная система управления базами данных (СУБД) от компании Oracle. Она используется для создания структуры новой базы, её наполнения, редактирования содержимого и отображения информации. Подходит для работы с высоконагруженными проектами, которые обрабатывают запросы миллионов пользователей (Рисунок. Endpoint Type (Тип подключения) – Oracle).

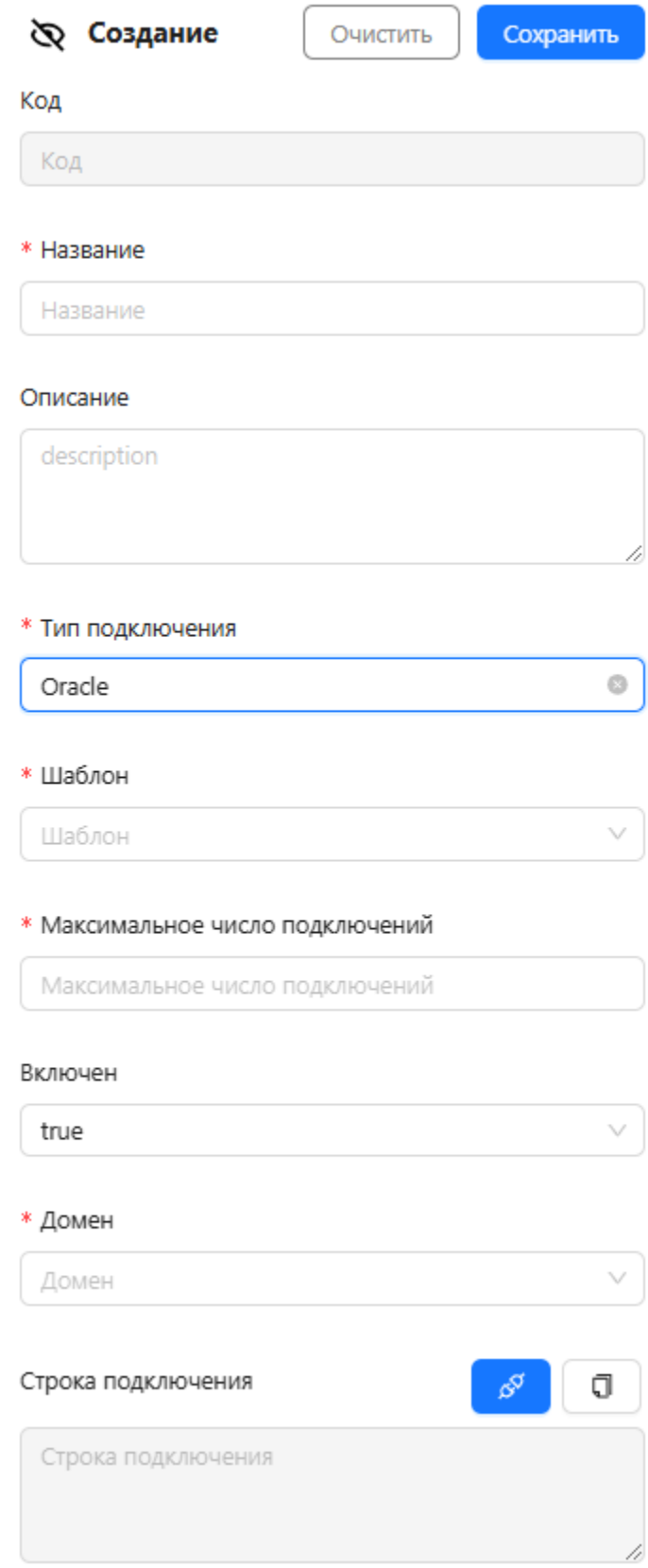


Рисунок. Endpoint Type (Тип подключения) – Oracle

Для типа подключения **Oracle** доступны несколько вариантов строк подключения (шаблонов). В интерфейсе выбор нужного шаблона осуществляется в поле Template (Шаблон) (Рисунок. Виды шаблонов для oracle).

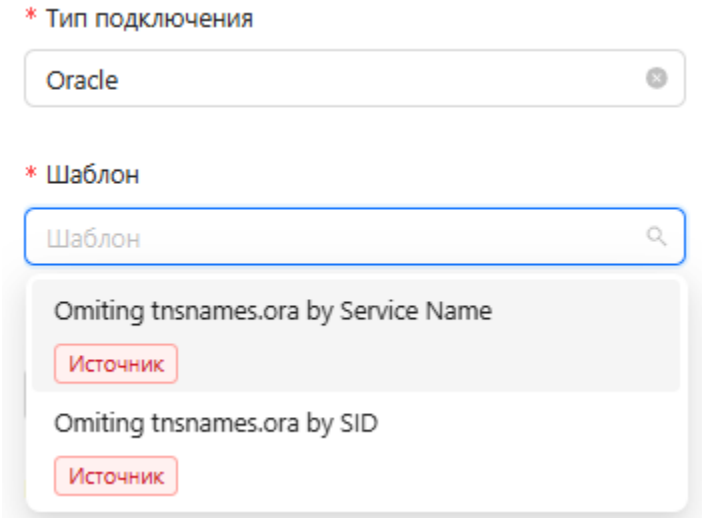


Рисунок. Виды шаблонов для Oracle

Примеры шаблонов:

Omiting tnsnames.ora by Service Name

- Endpoint Type (Тип подключения) – Oracle;
- Template (Шаблон) – Omiting tnsnames.ora by Service Name (выбирается в зависимости от настроек endpoint);
- Max connections (Максимальное число подключений);
- Поле Enabled (Включён) – опция указывает на доступность подключения в системе (представлено в выпадающем списке двумя позициями: true/false);
- Host (Адрес сервера) – указывается сетевой адрес, где размещен endpoint;
- Port (Порт) – указывается порт, по которому доступен endpoint;
- Service Name (Имя целевой службы) – название базы данных;
- User id (Пользователь) – указывается имя пользователя, которому доступны данные в endpoint;
- Password (Пароль) – пароль пользователя (сохраняется в базе данных системы и защищен от злоумышленников);
- Connection Protocol (Протолок подключения);
- Connection string (Строка подключения) – здесь создаётся строка подключения из заполненных полей выше. Пример: Host=localhost; Port=1521; SERVICE_NAME=/*{SERVICE_NAME}*/; User Id=postgres; Password=***; PROTOCOL=TCP;.

Omiting tnsnames.ora by SID

- Endpoint Type (Тип подключения) – Oracle;
- Template (Шаблон) – Omiting tnsnames.ora by SID (выбирается в зависимости от настроек endpoint);
- Max connections (Максимальное число подключений);
- Поле Enabled (Включён) – опция указывает на доступность подключения в системе (представлено в выпадающем списке двумя позициями: true/false);
- Host (Адрес сервера) – указывается сетевой адрес, где размещен endpoint;
- Port (Порт) – указывается порт, по которому доступен endpoint;
- Service Name (Имя целевой службы) – название базы данных;
- User id (Пользователь) – указывается имя пользователя, которому доступны данные в endpoint;
- Password (Пароль) – пароль пользователя (сохраняется в базе данных системы и защищен от злоумышленников);
- Connection Protocol (Протолок подключения);
- Connection string (Строка подключения) – здесь создаётся строка подключения из заполненных полей выше. Пример: Host=localhost; Port=1521; SERVICE_NAME=/*{SERVICE_NAME}*/; User Id=postgres; Password=***; PROTOCOL=TCP;.

Поля, указанные со звёздочкой, обязательны для заполнения.

PostgreSQL

Postgre – это свободно распространяемая объектно-реляционная система управления базами данных (СУБД) с открытым исходным кодом, написанном на языке С.

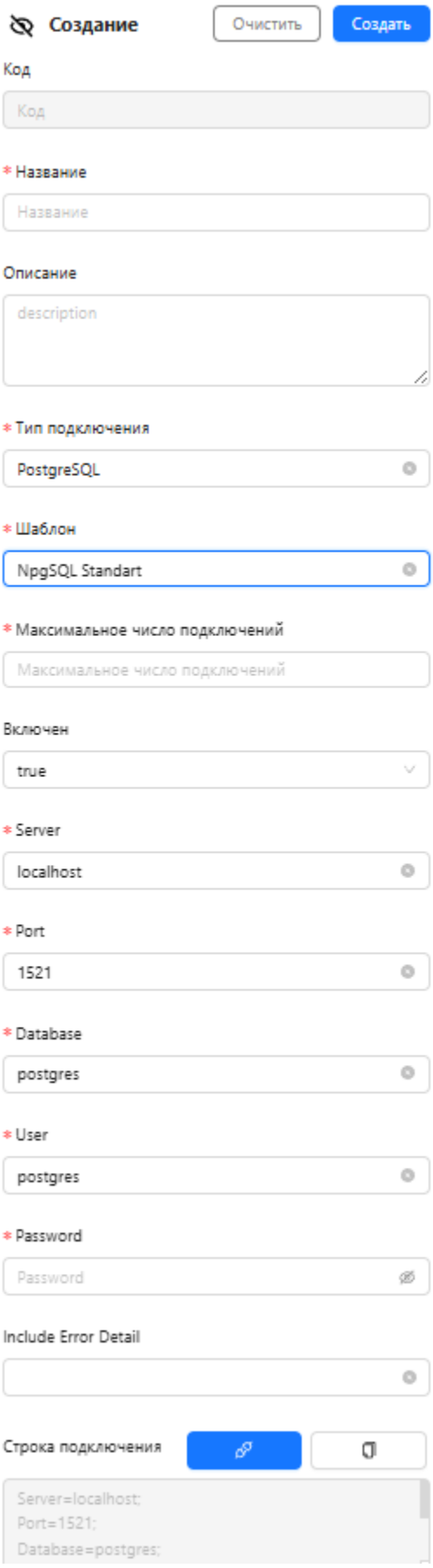

Создание
Очистить
Создать
Код
Код
* Название
Название
Описание
description
* Тип подключения
PostgreSQL
* Шаблон
NpgSQL Standart
* Максимальное число подключений
Максимальное число подключений
Включен
true
* Server
localhost
* Port
1521
* Database
postgres
* User
postgres
* Password
Password
Include Error Detail
Строка подключения
Server=localhost;
Port=1521;
Database=postgres;

Рисунок. Endpoint Type (Тип подключения) – PostgreSQL

Для типа подключения **Postgre** доступны несколько вариантов строк подключения (шаблонов). В интерфейсе выбор нужного шаблона осуществляется в поле Template (Шаблон)

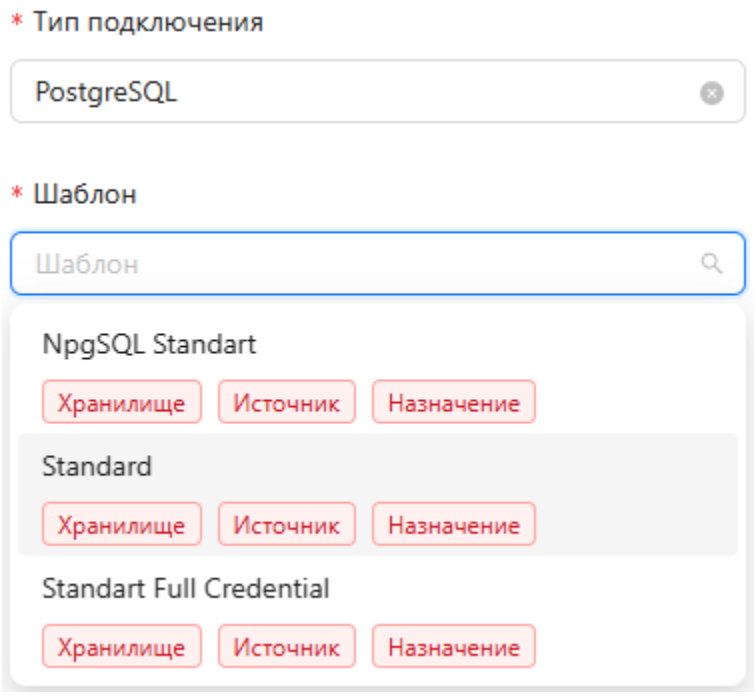


Рисунок. Варианты шаблонов для подключения PostgreSQL.

Примеры шаблонов:

NpgSQL Standart

- Endpoint Type (Тип подключения) – PostgreSQL;
- Template (Шаблон) – NpgSQL Standar*t* (выбирается в зависимости от настроек endpoint);
- Max connections (Максимальное число подключений);
- Поле Enabled (Включён) – опция указывает на доступность подключения в системе (представлено в выпадающем списке двумя позициями: true/false);
- Host (Адрес сервера) – указывается сетевой адрес, где размещен endpoint;
- Port (Порт) – указывается порт, по которому доступен endpoint;
- Database (Имя базы данных) – указывается имя базы данных;
- User (Пользователь) – указывается имя пользователя, которому доступны данные в endpoint;
- Password (Пароль) – пароль пользователя (сохраняется в базе данных системы и защищен от злоумышленников);
- Include Error Detail (Выводить детальный лог) – сохраняет детальный лог в процессе подключения к endpoint;
- Connection string (Строка подключения) – здесь создаётся строка подключения из заполненных полей выше. Пример: Server=192.168.128.1; Port=5432; Database=postgres; User Id=postgres; Password=***; Include Error Detail=false.

Standart

- Endpoint Type (Тип подключения) – PostgreSQL;
- Template (Шаблон) – Standart (выбирается в зависимости от настроек endpoint);
- Max connections (Максимальное число подключений);
- Поле Enabled (Включён) – опция указывает на доступность подключения в системе (представлено в выпадающем списке двумя позициями: true /false);
- Host (Адрес сервера) – указывается сетевой адрес, где размещен endpoint;
- Port (Порт) – указывается порт, по которому доступен endpoint;
- Database (Имя базы данных) – указывается имя базы данных;
- User (Пользователь) – указывается имя пользователя, которому доступны данные в endpoint;
- Password (Пароль) – пароль пользователя (сохраняется в базе данных системы и защищен от злоумышленников);
- Include Error Detail (Выводить детальный лог) – сохраняет детальный лог в процессе подключения к endpoint;
- Connection string (Строка подключения) – здесь создаётся строка подключения из заполненных полей выше. Пример: Server=192.168.128.1; Port=5432; Database=postgres; User Id=postgres; Password=***; Include Error Detail=false.

Standart Full Credential

- Endpoint Type (Тип подключения) – PostgreSQL;
- Template (Шаблон) – Standart Full Credential (выбирается в зависимости от настроек endpoint);
- Max connections (Максимальное число подключений);
- Поле Enabled (Включён) – опция указывает на доступность подключения в системе (представлено в выпадающем списке двумя позициями: true/false);

- Host (Адрес сервера) – указывается сетевой адрес, где размещен endpoint;
- Port (Порт) – указывается порт, по которому доступен endpoint;
- Database (Имя базы данных) – указывается имя базы данных;
- User (Пользователь) – указывается имя пользователя, которому доступны данные в endpoint;
- Password (Пароль) – пароль пользователя (сохраняется в базе данных системы и защищен от злоумышленников);
- Include Error Detail (Выводить детальный лог) – сохраняет детальный лог в процессе подключения к endpoint;
- Connection string (Строка подключения) – здесь создаётся строка подключения из заполненных полей выше. Пример: Server=192.168.128.1; Port=5432; Database=postgres; User Id=postgres; Password=***; Include Error Detail=false.

Поля, указанные со звёздочкой, обязательны для заполнения.

SAP Hana

SAP HANA (High-performance ANalytic Appliance) – это многомодельная база данных, в которой данные хранятся в памяти, а не на диске.

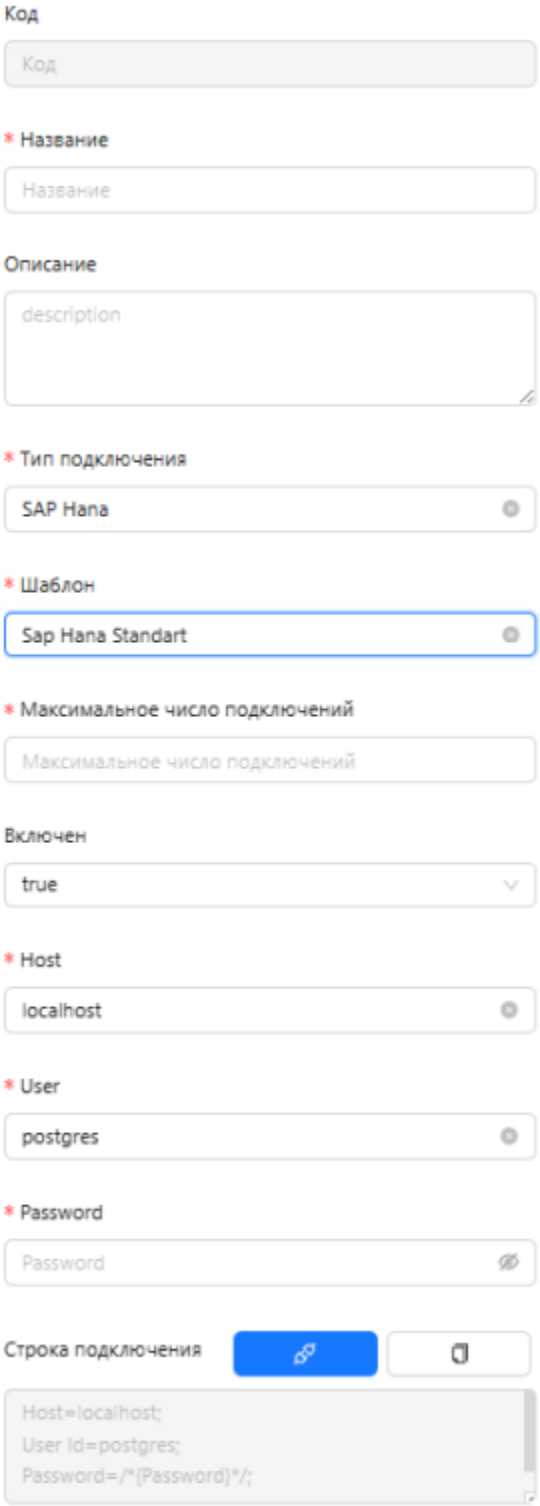


Рисунок. Endpoint Type (Тип подключения) – saphana

Для типа подключения **Saphana** доступны несколько вариантов строк подключения (шаблонов). В интерфейсе выбор нужного шаблона осуществляется в поле T emplate (Шаблон).

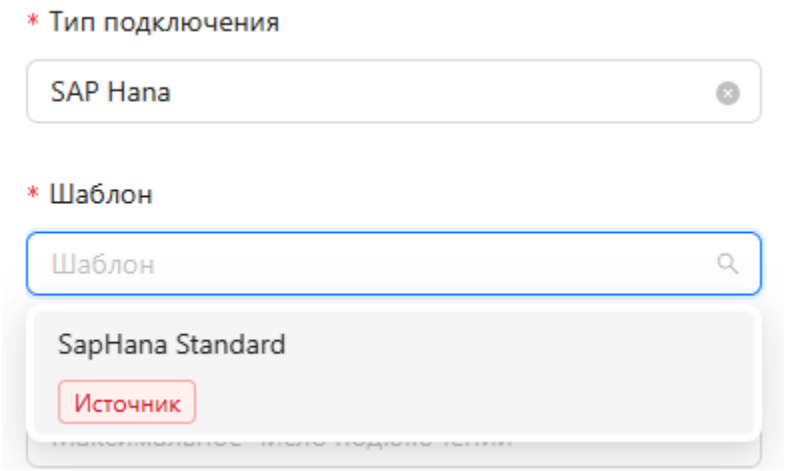


Рисунок. Варианты шаблонов для подключения saphana.

Sap Hana Standart

- Endpoint Type (Тип подключения) – saphana;
- Template (Шаблон) – Sap Hana Standart (выбирается в зависимости от настроек endpoint);
- Max connections (Максимальное число подключений);
- Поле Enabled (Включён) – представлено в выпадающем списке двумя позициями: true/false;
- Host (Адрес сервера) – указывается сетевой адрес, где размещен endpoint;
- User (Пользователь) – указывается имя пользователя, которому доступны данные в endpoint;
- Password (Пароль) – пароль пользователя (сохраняется в базе данных системы и защищен от злоумышленников);
- Keep-Alive query (Keep-Alive запрос) – тип запроса, который делает проверку подключения;
- Connection string (Строка подключения) – здесь создаётся строка подключения из заполненных полей выше. Пример: Host=hxehost:39015; User Id=SYSTEM; Password=***;.

Поля, указанные со звёздочкой, обязательны для заполнения.

Веб-сервисы

Веб-сервисы (или веб-службы) – это технология, позволяющая системам обмениваться данными друг с другом через сетевое подключение. Обычно веб-сервисы работают поверх протокола HTTP или протокола более высокого уровня. Веб-сервис – просто адрес, ссылка, обращение к которому позволяет получить данные или выполнить действие.

Apache Kafka

Apache Kafka – это распределённая система, предназначенная для обработки потоков данных в режиме реального времени. Её можно сравнить с почтой – одни сервисы передают туда сообщения-письма, а другие – получают. Apache Kafka называют брокером сообщений, потому что она выступает в качестве посредника.

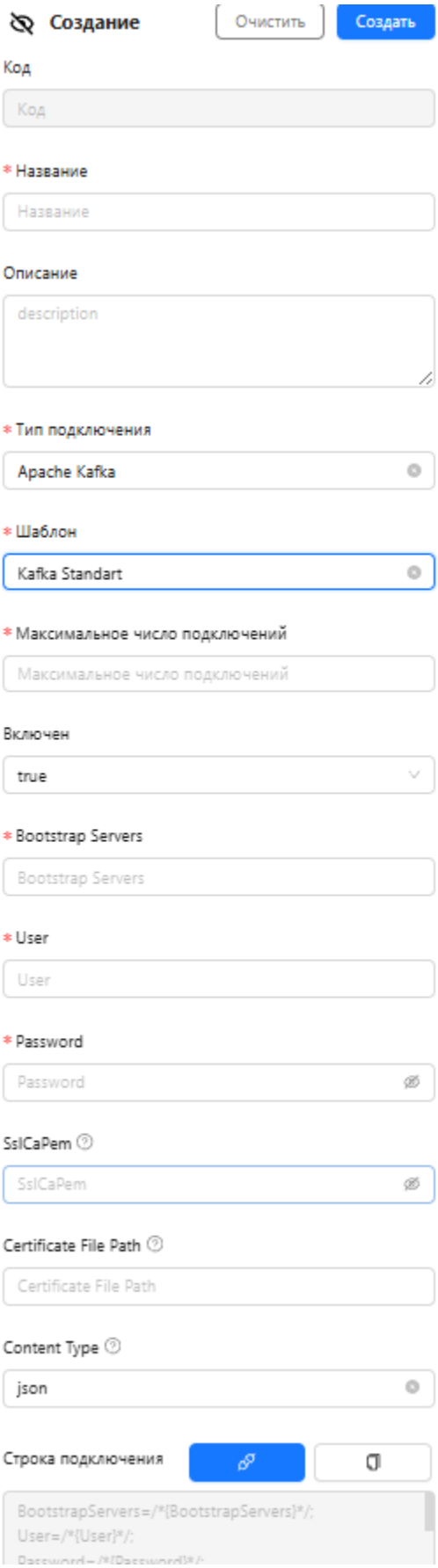


Рисунок. Endpoint Type (Тип подключения) - Apache Kafka

Для типа подключения **Kafka** доступны несколько вариантов строк подключения (шаблонов). В интерфейсе выбор нужного шаблона осуществляется в поле Template (Шаблон).

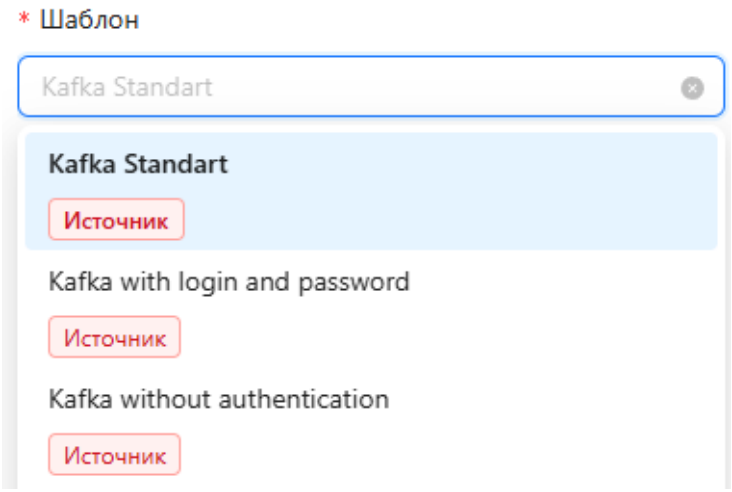


Рисунок. Варианты шаблонов для типа подключения kafka.

Примеры шаблонов:

Kafka Standart

- Endpoint Type (Тип подключения) – kafka;
- Template (Шаблон) – Kafka Standart (выбирается в зависимости от настроек endpoint);
- Max connections (Максимальное число подключений) – максимальное количество одновременных подключений к источнику;
- Поле Enabled (Включён) – опция указывает на доступность подключения в системе (представлено в выпадающем списке двумя позициями: true/false);
- Bootstrap Servers (Адрес кластера) – указывается IP-адрес и порт;
- User (Имя пользователя) – указывается имя пользователя, которому доступны данные в endpoint;
- Password (Пароль) – пароль пользователя (сохраняется в базе данных системы и защищен от злоумышленников);
- SslCaPem (Строка сертификата CA);
- Certificate File Path (Путь к файлу с SslCaPem);
- Content Type (Тип возвращаемых данных) – файл json формата;
- Connection string (Строка подключения) – здесь создаётся строка подключения из заполненных полей выше. Пример: BootstrapServers=198.250.56.15; ContentType=json; User=postgres; Password=***; SslCaPem=***; CertificateFilePath=/*{CertificateFilePath}*/;.

Kafka Standart with login and password

- Endpoint Type (Тип подключения) – kafka;
- Template (Шаблон) – Kafka Standart (выбирается в зависимости от настроек endpoint);
- Max connections (Максимальное число подключений) – максимальное количество одновременных подключений к источнику;
- Поле Enabled (Включён) – опция указывает на доступность подключения в системе (представлено в выпадающем списке двумя позициями: true/false);
- Sasl mechanism (Механизм Sasl);
- Security protocol (Протокол безопасности);
- Bootstrap Servers (Адрес кластера) – указывается IP-адрес и порт;
- User (Имя пользователя) – указывается имя пользователя, которому доступны данные в endpoint;
- Password (Пароль) – пароль пользователя (сохраняется в базе данных системы и защищен от злоумышленников);
- Content Type (Тип возвращаемых данных) – файл json формата;
- Connection string (Строка подключения) – здесь создаётся строка подключения из заполненных полей выше. Пример: BootstrapServers=198.250.56.15; ContentType=json; User=postgres; Password=***; SaslMechanism=Plain; SecurityProtocol=SaslPlaintext;.

Kafka Standart without authentication

- Endpoint Type (Тип подключения) – kafka;
- Template (Шаблон) – Kafka Standart (выбирается в зависимости от настроек endpoint);
- Max connections (Максимальное число подключений) – максимальное количество одновременных подключений к источнику;
- Поле Enabled (Включён) – опция указывает на доступность подключения в системе (представлено в выпадающем списке двумя позициями: true/false);
- Bootstrap Servers (Адрес кластера) – указывается IP-адрес и порт;
- Sasl mechanism (Механизм Sasl);
- Security protocol (Протокол безопасности);
- Content Type (Тип возвращаемых данных) – файл json формата;
- Connection string (Строка подключения) – здесь создаётся строка подключения из заполненных полей выше. Пример: BootstrapServers=198.250.56.15; ContentType=json; SaslMechanism=Plain; SecurityProtocol=SaslPlaintext;.

Sasl UserPassword

- Endpoint Type (Тип подключения) – kafka;
- Template (Шаблон) – Kafka Standart (выбирается в зависимости от настроек endpoint);
- Max connections (Максимальное число подключений) – максимальное количество одновременных подключений к источнику;
- Поле Enabled (Включён) – опция указывает на доступность подключения (представлено в выпадающем списке двумя позициями: true/false);

- Sasl mechanism (Механизм Sasl) – выбирается механизм аутентификации из выпадающего списка (Gssapi/Plain/ScramSha256/ScramSha512/OAuthBearer) ;
- Security protocol (Протокол безопасности) – выбирается протокол безопасности для аутентификации из выпадающего списка (Plaintext/Ssl/SaslPlaintext /SaslSsl);
- Bootstrap Servers (Адрес кластера) – указывается IP-адрес и порт;
- User (Имя пользователя) – указывается имя пользователя, которому доступны данные в endpoint;
- Password (Пароль) – пароль пользователя (сохраняется в базе данных системы и защищен от злоумышленников);
- Certificate File Path (Путь к файлу с SslCaPem) – указывается пусть к файлу доверенного сертификата;
- Content Type (Тип возвращаемых данных) – файл json формата;
- Connection string (Строка подключения) – здесь создаётся строка подключения из заполненных полей выше. Пример: BootstrapServers=198.250.56.15; ContentType=json; User=postgres; Password=***; SaslMechanism=Plain; SecurityProtocol=SaslPlaintext;.

Веб-сервисы (или веб-службы) – это **технология, позволяющая системам обмениваться данными друг с другом через сетевое подключение**. Обычно веб-сервисы работают поверх протокола HTTP или протокола более высокого уровня. Веб-сервис – просто адрес, ссылка, обращение к которому позволяет получить данные или выполнить действие.

Поля, указанные со звёздочкой, обязательны для заполнения.

RestAPI

REST (Representational State Transfer) API – это архитектурный стиль для разработки веб-сервисов, основанный на стандартных HTTP-методах и ресурсоориентированном подходе. Формат данных в REST API может быть разнообразным, включая JSON, XML и другие. REST API широко применяется в веб-приложениях и мобильных приложениях для обеспечения межсистемного взаимодействия и интеграции с различными сервисами и платформами.

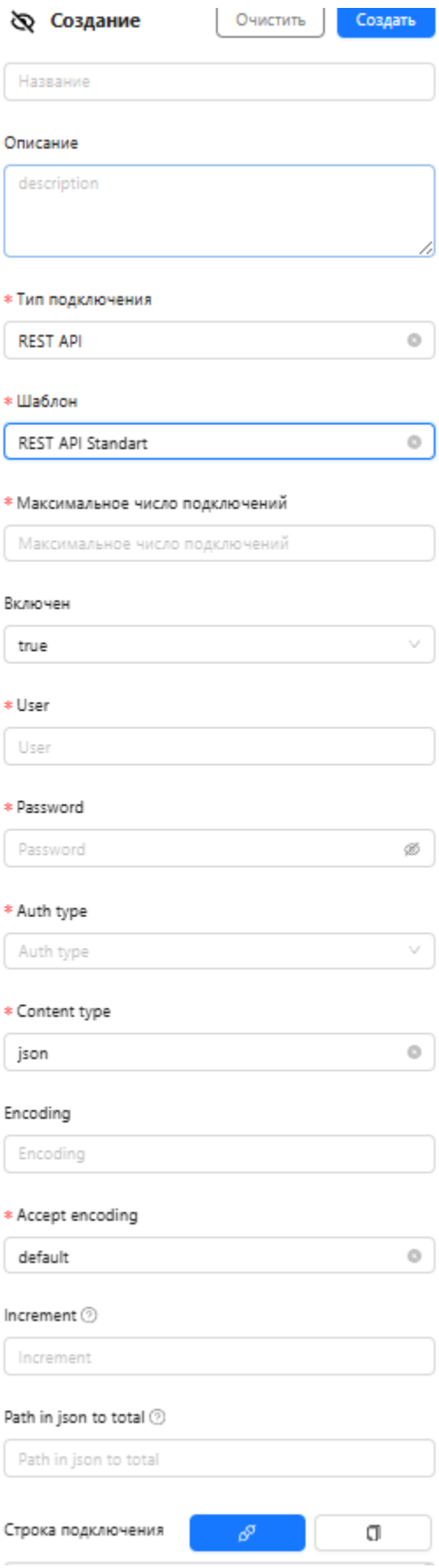


Рисунок. Endpoint Type (Тип подключения) - RestApi

Для типа подключения **RestAPI** доступны несколько вариантов строк подключения (шаблонов). В интерфейсе выбор нужного шаблона осуществляется в поле Template (Шаблон).

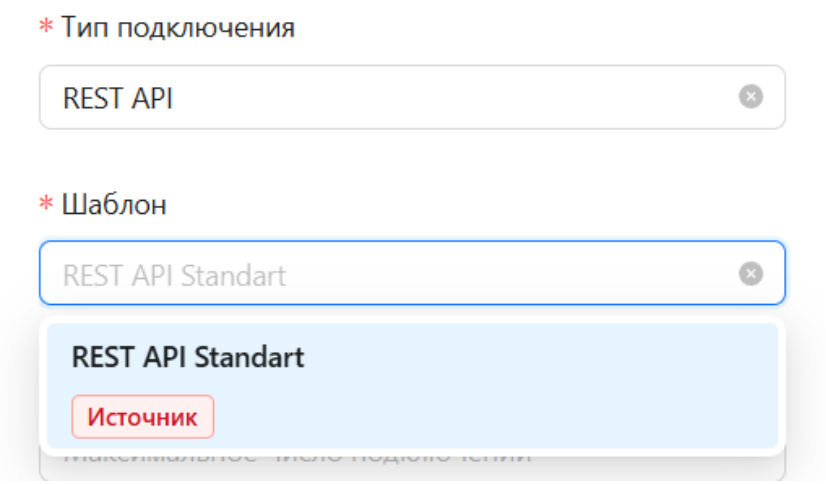


Рисунок. Пример шаблона REST API Standart для Endpoint Type (Тип подключения) – restapi.

Рассмотрим пример заполнения шаблона REST API Standart.

- Endpoint Type (Тип подключения) – restapi;
- Template (Шаблон) – REST API Standart (выбирается в зависимости от настроек endpoint);
- Max connections (Максимальное число подключений);
- Поле Enabled (Включён) – представлено в выпадающем списке двумя позициями: true/false;
- User (Пользователь) – указывается имя пользователя, которому доступны данные в endpoint;
- Password (Пароль) – пароль пользователя (сохраняется в базе данных системы и защищен от злоумышленников);
- AuthType (Тип аутентификации) – выбирается из выпадающего списка (см. рис. Выпадающий список поля AuthType (Тип аутентификации));

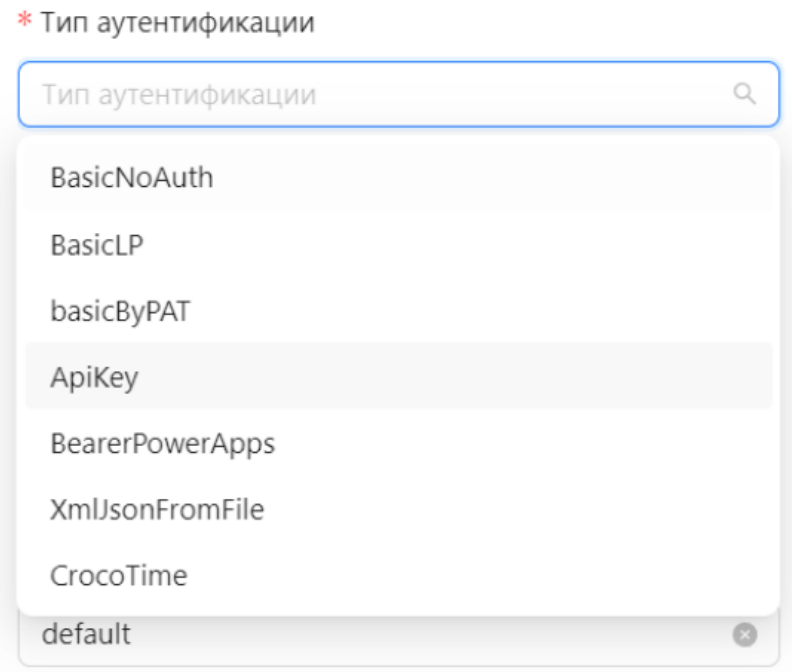


Рисунок. Выпадающий список поля AuthType (Тип аутентификации)

- ContentType (Тип возвращаемого контента) – выбирается из выпадающего списка для файлов типа csv, xml, json;

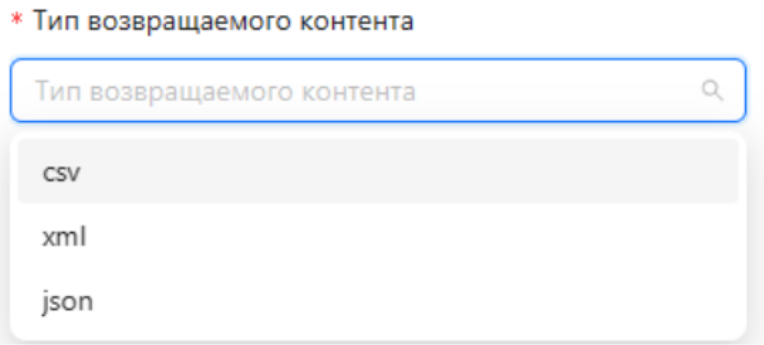


Рисунок. Выпадающий список поля ContentType (Тип возвращаемого контента)

- Encoding (Кодировка);
- AcceptEncoding (Тип декомпрессии);
- Increment (Инкремент) – значение инкремента для подстановки и множественного вызова одного запроса Rest, с записью в один результирующий json;
- Path in json to total (Путь в json до общего числа записей) – путь до общего числа записей, используется для завершения цикла;
- Connection string (Строка подключения) – здесь создаётся строка подключения из заполненных полей выше. Пример: User=postgres; Password=***; AuthType=1; ContentType=csv; Encoding=/*{Encoding}*/; AcceptEncoding=/*{AcceptEncoding}*/; Start=/*{Start}*/;.

1С Предприятие

1С:Предприятие – это полнотекстовая малокодовая платформа, предоставляющая готовую к использованию инфраструктуру и инструменты для быстрой разработки бизнес-приложений, таких как ERP, POS, WMS или другое индивидуальное корпоративное программное обеспечение.

Может быть развёрнута на СУБД: MS SQL Server и PostgreSQL.

1С на базе Microsoft SQL Server

SQLServer1C – серверное программное обеспечение, для работы с базами данных «1С» в клиент-серверном режиме (СУБД). СУБД обрабатывает запрос, который пришел от сервера «1С» и отправляет данные обратно на сервер «1С». Подключение SQL необходимо при работе в 1С в клиент-серверном режиме, это позволяет оптимизировать работу большого количества пользователей с большим объёмом информации, за счет переноса ресурсоёмких операций на сервер.

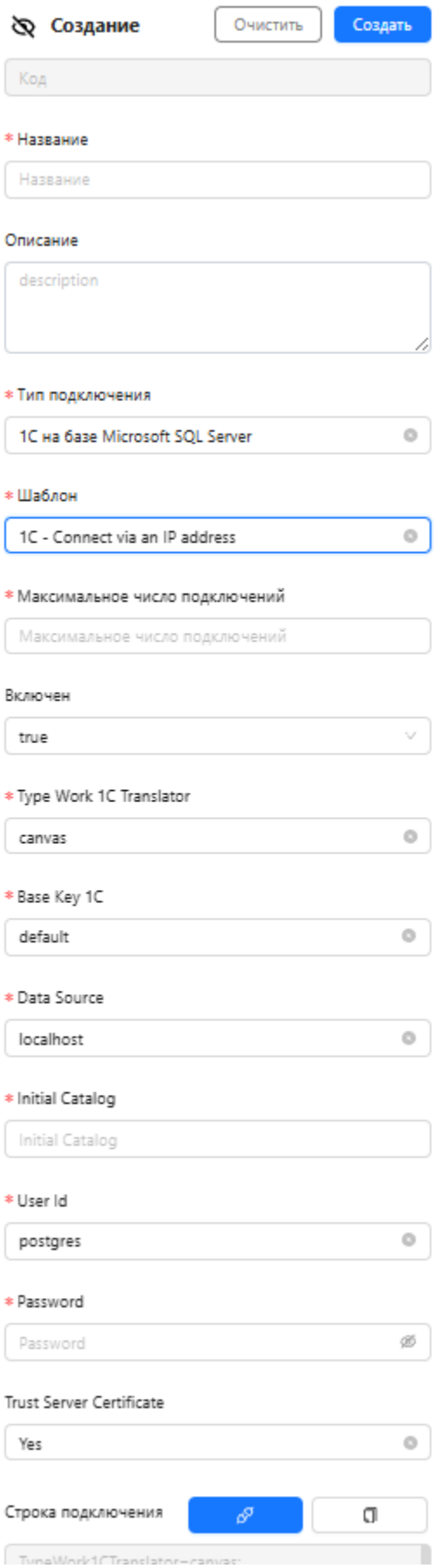

Создание
Очистить
Создать
Код
* Название
Название
Описание
description
* Тип подключения
1С на базе Microsoft SQL Server
* Шаблон
1C - Connect via an IP address
* Максимальное число подключений
Максимальное число подключений
Включен
true
* Type Work 1C Translator
canvas
* Base Key 1C
default
* Data Source
localhost
* Initial Catalog
Initial Catalog
* User Id
postgres
* Password
Password
Trust Server Certificate
Yes
Строка подключения

Рисунок. Тип подключения "1С на базе Microsoft SQL Server"

Для типа подключения **SQLServer1C** доступны несколько вариантов строк подключения (шаблонов). В интерфейсе выбор нужного шаблона осуществляется в поле "Шаблон" (Template).

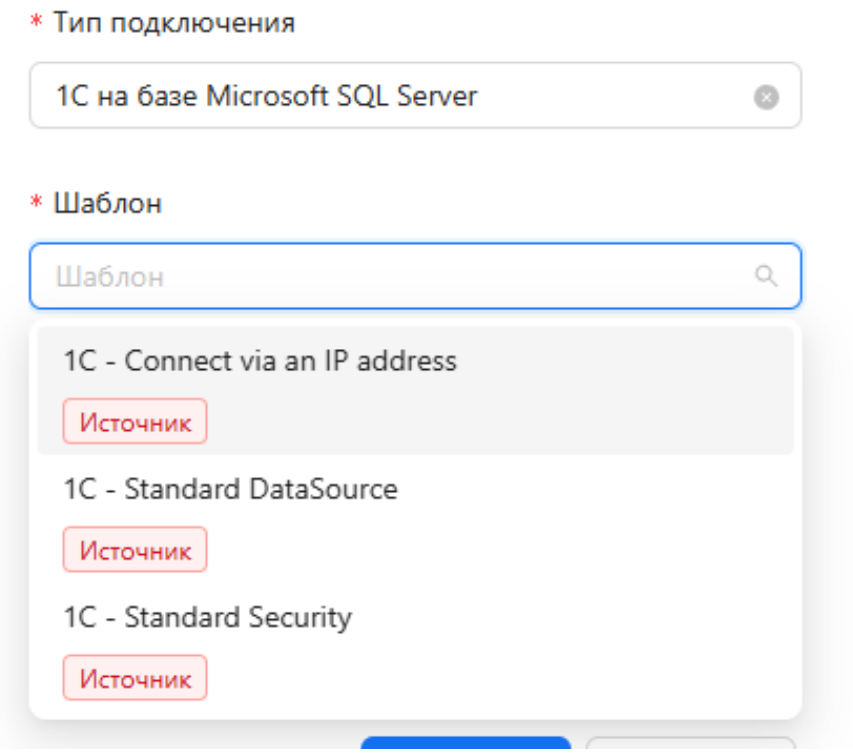


Рисунок. Варианты шаблонов

Примеры шаблонов:

1C - Connect via an IP address

- Endpoint Type (Тип подключения) – sqlserver1c;
- Template (Шаблон) – 1C - Connect via an IP address (выбирается в зависимости от настроек endpoint);
- Max connections (Максимальное число подключений);
- Поле Enabled (Включён) представлено в выпадающем списке двумя позициями: true/false – определяет доступность, создаваемого подключения в командах, если поле имеет состояние false, то команду с этим эндпоинтом создать будет нельзя;
- Type Work 1C Translator (Тип режима работы 1С) представлен двумя позициями: canvas/hard;
- Base Key 1C (Название БД 1С) – указывается название базы данных;
- Data Source (Сервер) – указывается адрес сервера;
- Initial Catalog (База данных) – указывается база данных;
- User Id (Пользователь) – указывается имя пользователя, которому доступны данные в endpoint;
- Password (Пароль) – пароль пользователя (сохраняется в базе данных системы и защищен от злоумышленников);
- Trust Server Certificate (Доверять сертификату сервера) – параметр, влияющий на проверку сертификата в зависимости от выбранного значения: yes/no
- Connection string (Строка подключения) – здесь создаётся строка подключения из заполненных полей выше. Пример: TypeWork1CTranslator=canvas; BaseKey1C=default; Data Source=localhost; Initial Catalog=/*{Initial Catalog}*/; User Id=root; Password=***; TrustServerCertificate=Yes;.

1C - Standart DataSource

- Endpoint Type (Тип подключения) – sqlserver1c;
- Template (Шаблон) – 1C - Standart DataSource (выбирается в зависимости от настроек endpoint);
- Max connections (Максимальное число подключений);
- Поле Enabled (Включён) – представлено в выпадающем списке двумя позициями: true/false;
- Type Work 1C Translator (Тип режима работы 1С) – представлен двумя позициями: canvas/hard;
- Base Key 1C (Название БД 1С) – указывается название базы данных;
- Data Source (Сервер) – указывается адрес сервера;
- Initial Catalog (База данных) – указывается база данных;
- User Id (Пользователь) – указывается имя пользователя, которому доступны данные в endpoint;
- Password (Пароль) – пароль пользователя (сохраняется в базе данных системы и защищен от злоумышленников);
- Trust Server Certificate (Доверять сертификату сервера) – параметр, влияющий на проверку сертификата в зависимости от выбранного значения: yes/no
- Connection string (Строка подключения) – здесь создаётся строка подключения из заполненных полей выше. Пример: TypeWork1CTranslator=canvas; BaseKey1C=default; Data Source=localhost; Initial Catalog=/*{Initial Catalog}*/; User Id=root; Password=***; TrustServerCertificate=Yes;.

1C - Standart Security

- Endpoint Type (Тип подключения) – sqlserver1c;
- Template (Шаблон) – 1C - Standart Security (выбирается в зависимости от настроек endpoint);
- Max connections (Максимальное число подключений);
- Поле Enabled (Включён) представлено в выпадающем списке двумя позициями: true/false;
- Type Work 1C Translator (Тип режима работы 1С) представлен двумя позициями: canvas/hard;
- Base Key 1C (Название БД 1С) – указывается название базы данных;
- Data Source (Сервер) – указывается адрес сервера;
- Initial Catalog (База данных) – указывается база данных;
- User Id (Пользователь) – указывается имя пользователя, которому доступны данные в endpoint;
- Password (Пароль) – пароль пользователя (сохраняется в базе данных системы и защищен от злоумышленников);
- Trust Server Certificate (Доверять сертификату сервера) – параметр, влияющий на проверку сертификата в зависимости от выбранного значения: yes/no
- Connection string (Строка подключения) – здесь создаётся строка подключения из заполненных полей выше. Пример: TypeWork1CTranslator=canvas; BaseKey1C=default; Data Source=localhost; Initial Catalog=/*{Initial Catalog}*/; User Id=root; Password=***; TrustServerCertificate=Yes;.

Поля, указанные со звёздочкой, обязательны для заполнения.

1С на базе PostgreSQL

PostgreSQL 1С – одна из систем управления базами данных, которую поддерживает платформа в клиент-серверном варианте работы. Включает патчи с оптимизациями, выполненными разработчиками платформы 1С:Предприятия, которые учитывают особенности работы платформы 1С:Предприятие и типовых решений фирмы «1С». Используется «1С» в высоконагруженных коммерческих проектах, например, 1С:Fresh.

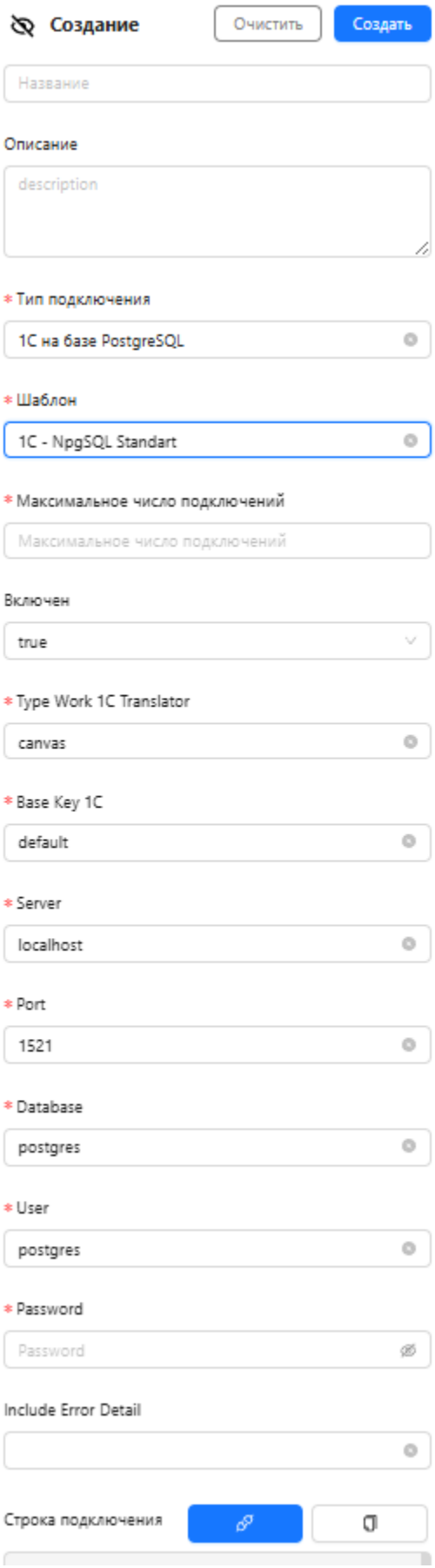

Создание
Очистить
Создать
Название
Описание
description
Тип подключения
1С на базе PostgreSQL
Шаблон
1С - NpgSQL Standart
Максимальное число подключений
Максимальное число подключений
Включен
true
Type Work 1С Translator
canvas
Base Key 1С
default
Server
localhost
Port
1521
Database
postgres
User
postgres
Password
Password
Include Error Detail
Строка подключения

Рисунок. Endpoint Type (Тип подключения) – PostgreSQL1C

Для типа подключения **PostgreSQL 1C** доступны несколько вариантов строк подключения (шаблонов). В интерфейсе выбор нужного шаблона осуществляется в поле Template (Шаблон).

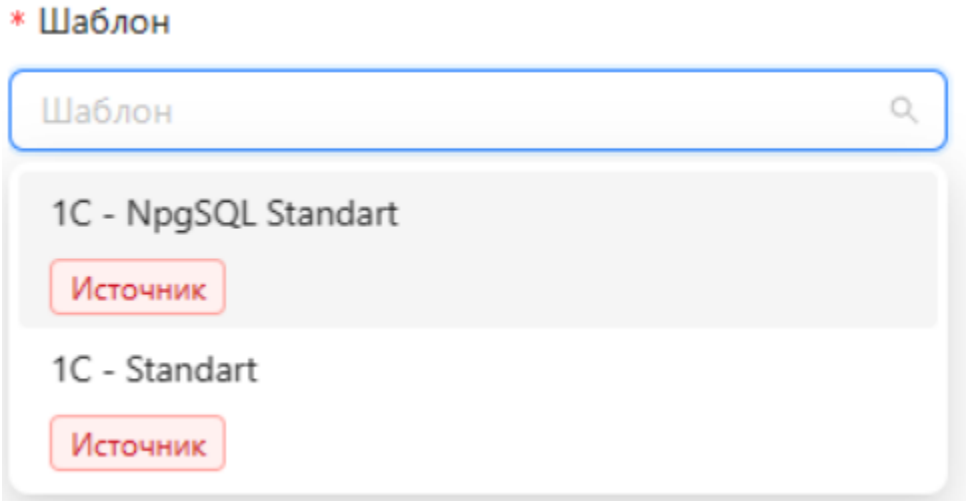


Рисунок. Варианты шаблонов для подключения PostgreSQL1C

Примеры шаблонов:

1C - NpgSQL Standart

- Endpoint Type (Тип подключения) – postgresql1c;
- Template (Шаблон) – 1C - NpgSQL Standart (выбирается в зависимости от настроек endpoint);
- Max connections (Максимальное число подключений);
- Поле Enabled (Включён) представлено в выпадающем списке двумя позициями: true/false;
- Type Work 1C Translator (Тип режима работы 1С) представлен двумя позициями: canvas/hard;
- Base Key 1C (Название БД 1С) – указывается название базы данных;
- Server (Адрес сервера) – указывается сетевой адрес, где размещен endpoint;
- Port (Порт) – указывается порт, по которому доступен endpoint;
- Database (Имя базы данных) – указывается имя базы данных;
- User (Пользователь) – указывается имя пользователя, которому доступны данные в endpoint;
- Password (Пароль) – пароль пользователя (сохраняется в базе данных системы и защищен от злоумышленников);
- Include Error Detail (Выводить детальный лог) – сохраняет детальный лог в процессе подключения к endpoint (представлено в выпадающем списке двумя позициями: true/false);
- Connection string (Строка подключения) – здесь создаётся строка подключения из заполненных полей выше. Пример: TypeWork1CTranslator=canvas; BaseKey1C=default; Server=localhost; Port=3306; Database=postgres; User Id=postgres; Password=***; Include Error Detail=false;.

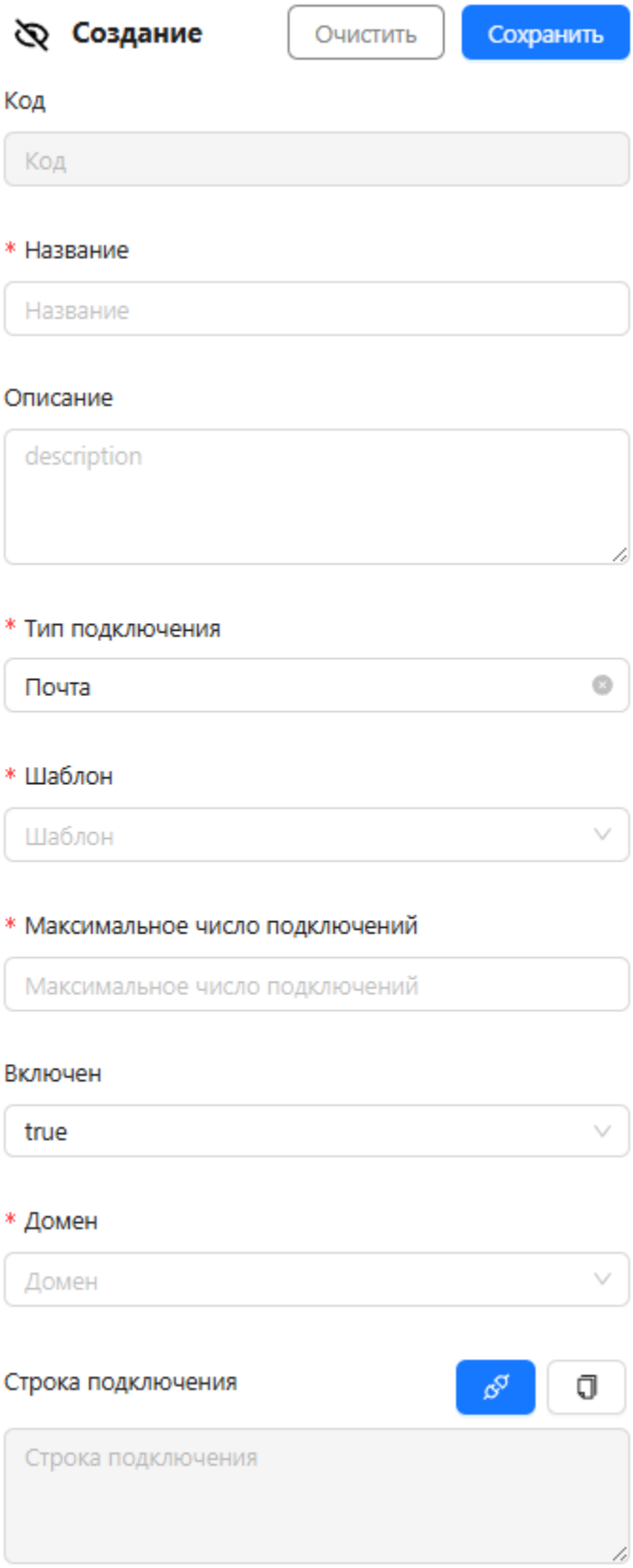
1C - Standart

- Endpoint Type (Тип подключения) – postgresql1c;
- Template (Шаблон) – 1C - Standart (выбирается в зависимости от настроек endpoint);
- Max connections (Максимальное число подключений);
- Поле Enabled (Включён) представлено в выпадающем списке двумя позициями: true/false;
- Type Work 1C Translator (Тип режима работы 1С) представлен двумя позициями: canvas/hard;
- Base Key 1C (Название БД 1С) – указывается название базы данных;
- Host (Адрес сервера) – указывается сетевой адрес, где размещен endpoint;
- Port (Порт) – указывается порт, по которому доступен endpoint;
- Database (Имя базы данных) – указывается имя базы данных;
- User (Пользователь) – указывается имя пользователя, которому доступны данные в endpoint;
- Password (Пароль) – пароль пользователя (сохраняется в базе данных системы и защищен от злоумышленников);
- Include Error Detail (Выводить детальный лог) – сохраняет детальный лог в процессе подключения к endpoint (представлено в выпадающем списке двумя позициями: true/false);
- Connection string (Строка подключения) – здесь создаётся строка подключения из заполненных полей выше. Пример: TypeWork1CTranslator=canvas; BaseKey1C=default; Server=localhost; Port=3306; Database=postgres; User Id=postgres; Password=***; Include Error Detail=false;.

Поля, указанные со звёздочкой, обязательны для заполнения.

Почтовые сервера

Данный тип подключения используется компонентом MetaControl. Для рассылки писем о результатах работы компонента чаще всего достаточно использовать один почтовый сервер от одной технической учетной записи. Для настройки подключения необходимо выбрать соответствующий пункт в списке типов подключения и шаблон строки подключения. В интерфейсе появятся поля, необходимые для заполнения.



- Название (Name) – название подключения
- Описание (Description) – краткое описание назначения подключения
- Тип подключения (Endpoint Type) – Почта

- Шаблон (Template) – Email Standart
- Максимальное число подключений (Max connections) – максимальное одновременное число подключений
- SMTP – адрес smtp сервера
- Порт (Port) – порт, по которому доступен smtp-сервер
- SMTP пользователь (SMTP user) – доменное имя пользователя
- Почта (Mail) – почтовый ящик
- Пароль (Mail password) – пароль от почтового ящика
- Почтовый ящик (Mailbox name) – имя почтового ящика (любое название)
- SMTP Шифрование (SmtpEncryption) – указать если какое-то используется.

Поля, указанные со звёздочкой, обязательны для заполнения.

После выполнения всех настроек можно проверить созданное соединение.

Мессенджеры

Данный тип подключения используется компонентом MetaControl для отправки сообщений о результатах работы компонента. В данный момент реализована поддержка одного мессенджера.

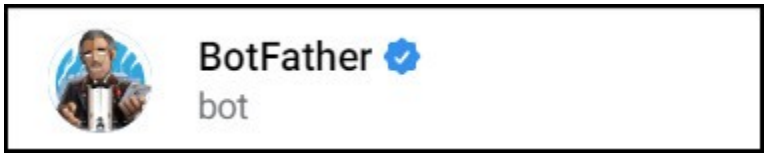
Telegram

Создание Telegram-канала

Чтобы создать Telegram Bot API, нужно найти BotFather в Телеграм. Это отец всех ботов. С помощью него можно создавать и управлять многочисленными чат-ботами.

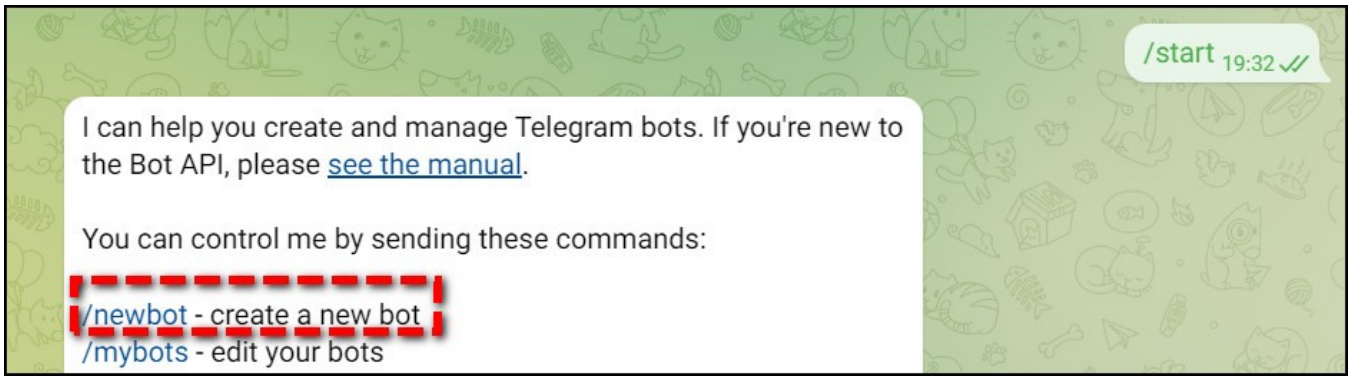
Инструкция, как получить токен в Телеграм:

Вбить в поисковом поле мессенджера @**botfather**.

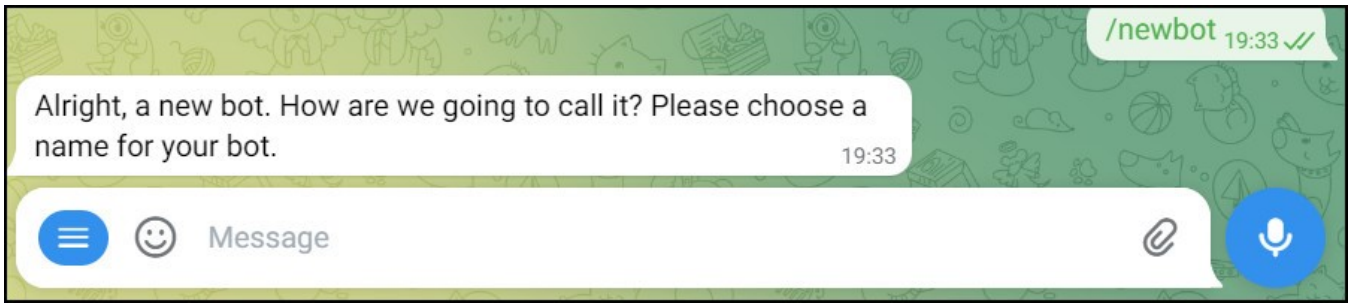


Активировать найденную утилиту. Это делают следующей командой «/start»

Откроется окно, где пользователю нужно найти фразу «/new bot». Кликнуть по ней



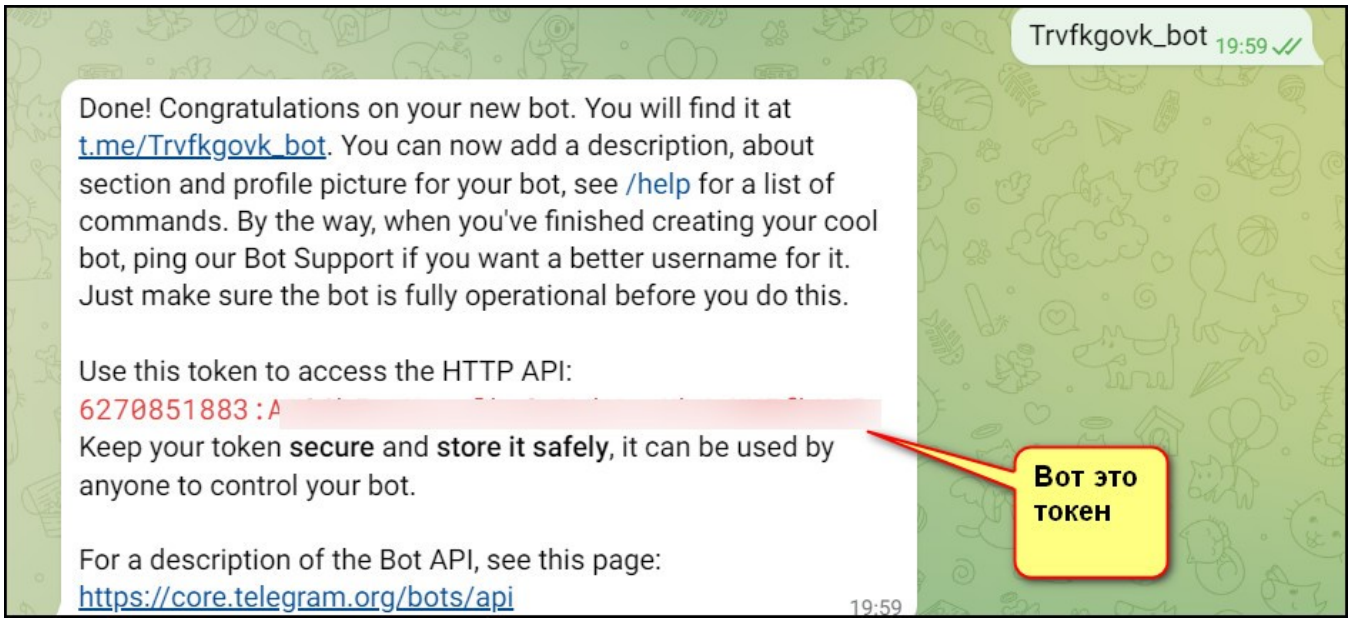
Дать имя будущему роботу, который будет общаться с клиентами



Если юзернейм бота будет занят, то система сообщит об этом

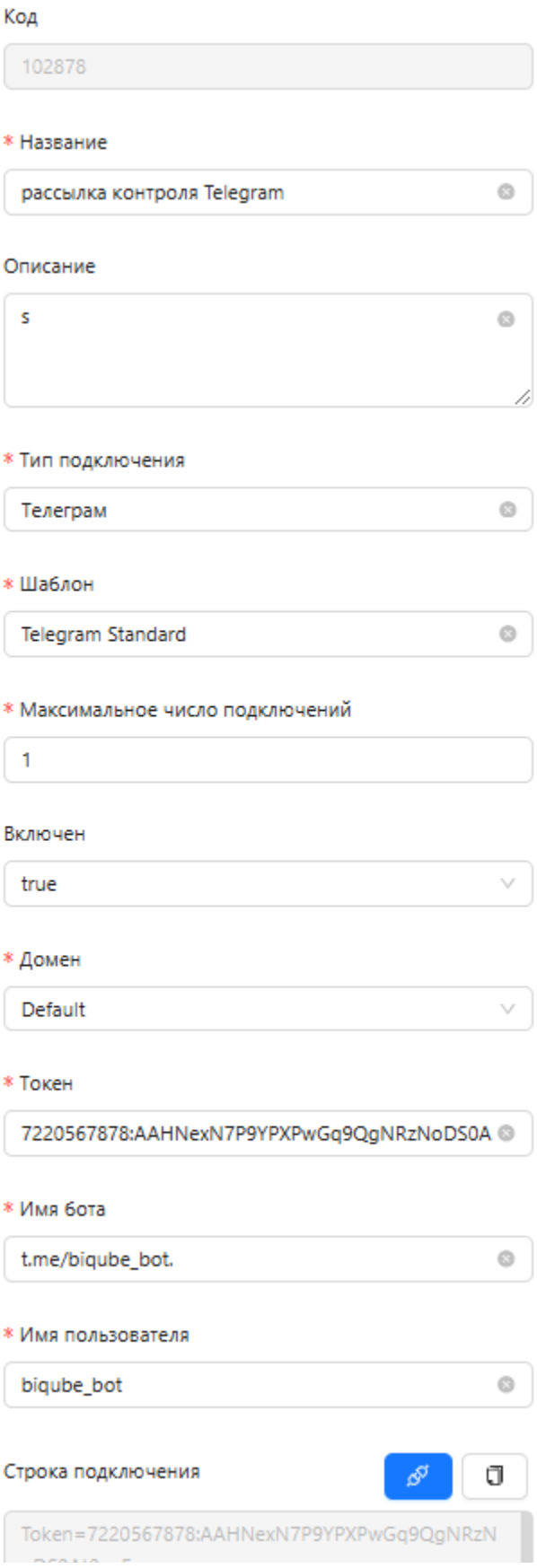
Нужно хорошо подумать и прописать еще раз имя. Следует не забывать после названия вводить «_bot». Иначе система не примет название

После того, как будет создано оригинальное название, система выдаст API токен



Настройка подключения

Для настройки подключения необходимо выбрать соответствующий пункт в списке типов подключения и шаблон строки подключения. В интерфейсе появятся поля, необходимые для заполнения.



- Название (Name) – название подключения
- Описание (Description) – краткое описание назначения подключения
- Тип подключения (Endpoint Type) – Телеграм
- Шаблон (Template) – Telegram Standart
- Максимальное число подключений (Max connections) – максимальное одновременное число подключений
- Включен (Enabled) – опция указывает на доступность подключения в системе (представлено в выпадающем списке двумя позициями: true/false);
- Домен (Domain) – указать домен
- Токен (Token) – уникальный идентификатор, который позволяет вашему боту взаимодействовать с API Telegram
- Имя бота (Bot name) – имя бота, которое указывалось при создании
- Имя пользователя (User name) – имя пользователя, которое указывалось при создании бота
- Строка подключения (Connection string) – здесь создаётся строка подключения из заполненных полей выше.

Поля, указанные со звёздочкой, обязательны для заполнения.

После выполнения всех настроек можно проверить созданное соединение.

ПАРАМЕТРЫ

Создание параметров

В системе BI.Qube пользователи в запросах к источникам данных и в других объектах могут использовать параметры. В системе доступны параметры двух видов:

- пользовательские;
- системные.

Пользовательские параметры, создаются пользователем и могут быть двух типов:

- константа – заранее определенное значение число или текст;
- SQl-запрос – вычисляемый запрос, результатом которого может быть как значение так и двумерная таблица.

Системные параметры подготовлены разработчиками системы и возвращают значения в зависимости от текущего контекста. В системе доступны следующие системные параметры:

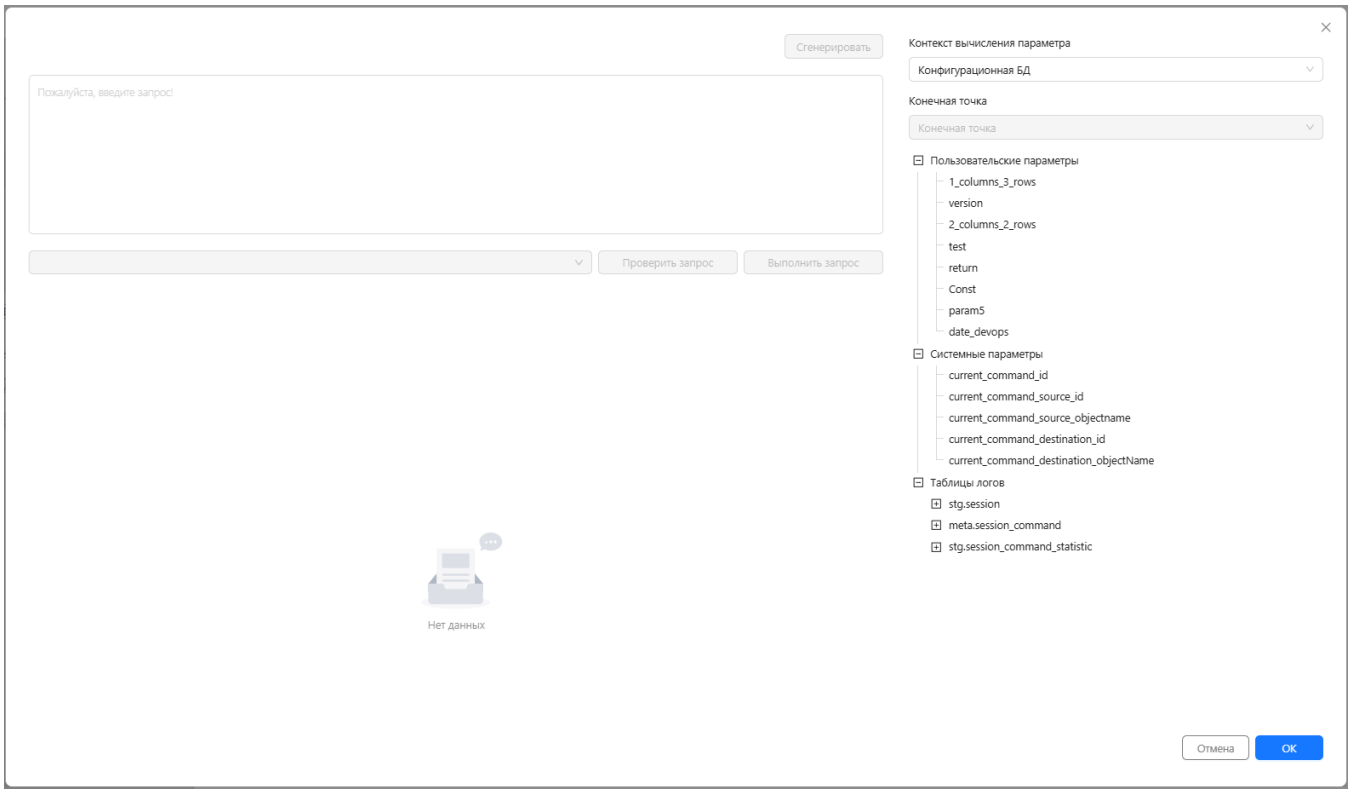
- current_command_id – возвращает числовое значение идентификатора команды извлечения данных, в запросе которой вычисляется значение параметра;
- current_command_source_id – возвращает числовое значения идентификатора источника данных для команды, в запросе которой вычисляется значение параметра;
- current_command_source_objectname – возвращает текстовую строку, содержащую имя объекта (только для sql-запросов к источникам типа СУБД), являющегося источником данных для команды, в запросе которой вычисляется значение параметра;
- current_command_distination_id – возвращает числовое значения идентификатора базы данных, в которую предполагается запись извлеченных данных, команды, в запросе которой вычисляется значение параметра;
- current_command_source_objectname – возвращает текстовую строку, содержащую имя объекта, в которой выполняется запись данных, командой, в запросе которой вычисляется значение параметра;

Для создания параметра необходимо перейти в разделе "Общие" на вкладку "Параметры", нажать кнопку "Создать", чтобы создать новый параметр.

После нажатия кнопки "Создать" появится возможность заполнить поля в правой части экрана:

- Наименование – имя параметра;
- Тип – типа параметра (константа или sql-запрос);
- Значение – в случае если выбран тип константа, то значение вводится с клавиатуры, если выбран тип sql-запрос, то доступна кнопка "Сконструировать", нажатие на которую приведет к появлению нового диалогового окна, позволяющего создать запрос;
- Описание – текстовое описание назначения параметра.

Диалоговое окно создания параметра типа sql-запрос позволяет создавать запросы с использованием ранее созданных пользовательских запросов, системных запросов и других системных объектов. При этом необходимо понимать, что параметр типа sql-запрос может быть выполнен только в контексте какого-то ендпоинта, т.е. под управлением какой-то базы данных, доступ к которой есть у системы. Другие контексты (ендпоинты) в системе не доступны.



Объекты доступные в параметрах

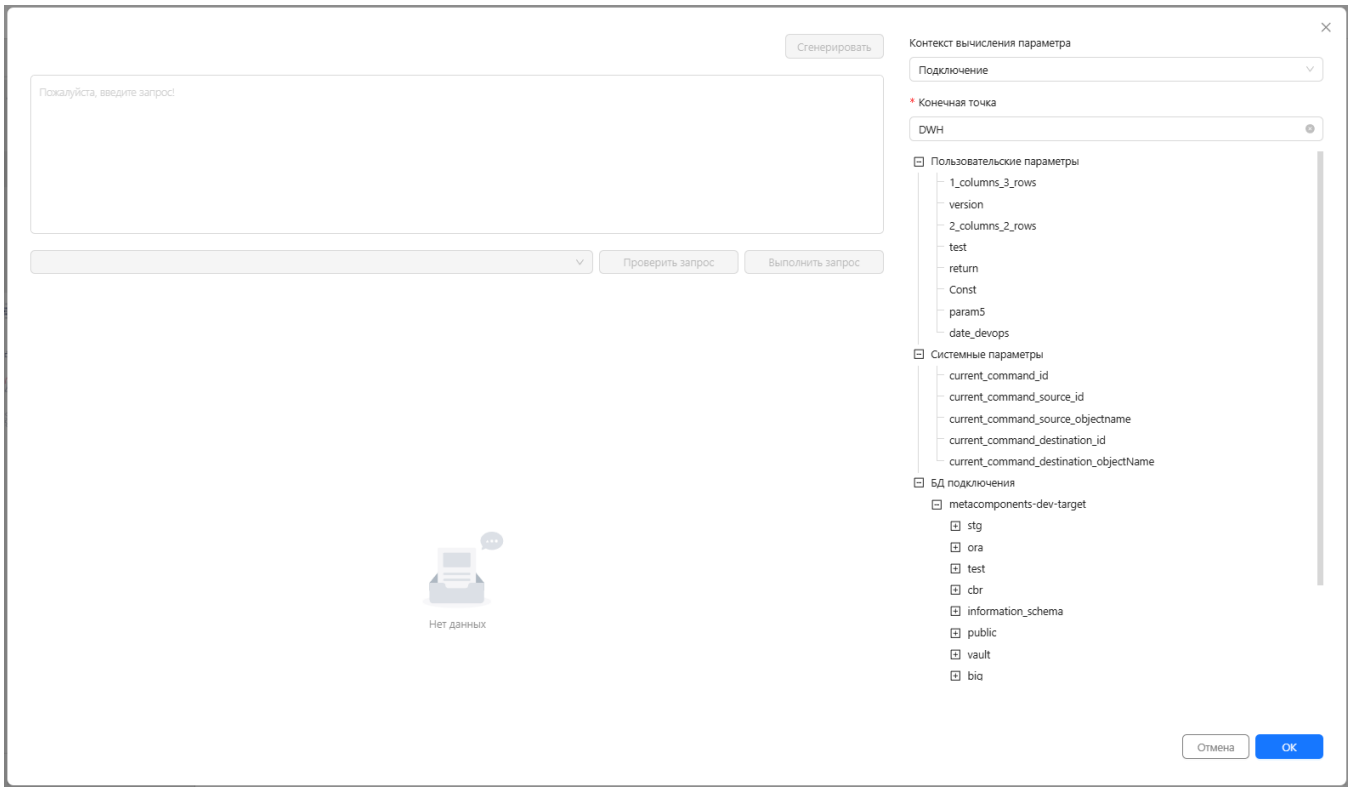
Рисунок. Объектные доступные в параметрах

В окне настройки параметра типа sql-запрос по умолчанию выбран контекст выполнения "Конфигурационная БД" – это база данных в которой размещены все настроечные таблицы и таблицы логов. В ряде случаев пользователю может быть полезна информация из таблиц логов, например дата последней загрузки какой-то команды, или значение какого-то поля. Кроме этого в теле sql-запроса параметра можно использовать ранее созданные параметры.

Система поддерживает работу со следующими типами контекста:

- Конфигурационная БД;
- Источник – параметр будет вычислен в контексте СУБД источника той команды, в запросе которой используется параметр;
- Назначение – параметр будет вычислен в контексте СУБД назначения той команды, в запросе которой используется параметр;
- Подключение – явное указание СУБД, в контексте которой будет вычислен параметр.

В зависимости от выбранного типа контекста изменяется содержимое окно создания параметра типа sql-запрос. Так при выборе контекста типа "Конфигурационная БД" пользователю доступны в запрос параметра все пользовательские параметра, т.е. все параметры созданные ранее, системные параметры и таблицы логов. Если выбран контекст "Назначение" или "Источник", то пользователю в параметре типа sql-запрос доступны все пользовательские параметры и системные параметры, объекты конфигурационной базы данных недоступны. В таких режимах система сама определяет СУБД, в контексте которой будет вычисляться параметр на основании данных той команды, в запросе которой используется параметр. При этом, при создании параметра типа sql-запрос, для проверки его работы, система каждый раз будет предлагать выбрать команду, в которой будет использован параметр. Такой подход позволяет один параметр использовать одновременно в нескольких командах, и для каждой команды значение параметра будет вычисляться индивидуально. При выборе типа контекста "Подключение" у пользователя появляется возможность явно выбрать СУБД, в контексте которой будет вычисляться запрос. В этом случае пользователю доступны пользовательские параметры, системные параметры (кроме тех, которые возвращают информацию об источнике и/или назначении), а так же объекты этого подключения.

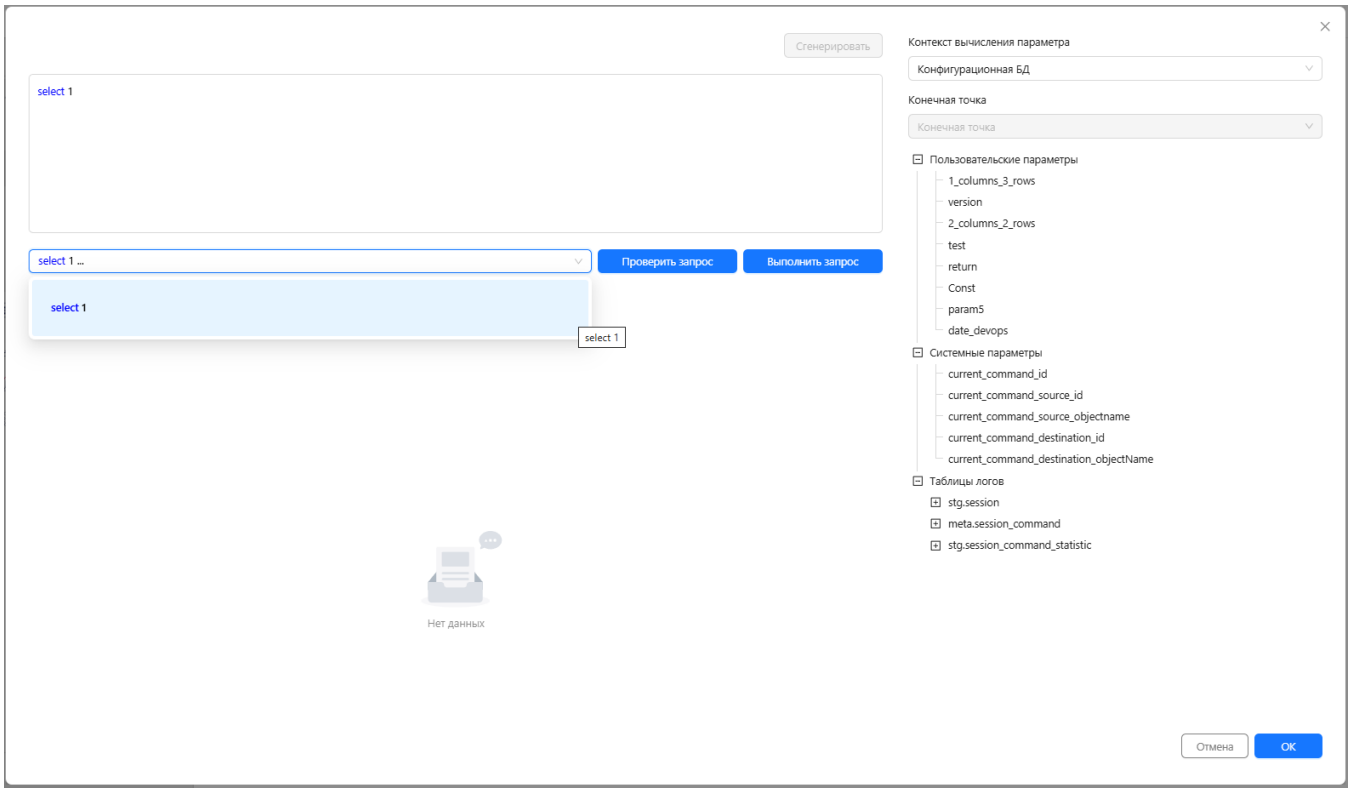


Для создания sql-кода запроса параметра необходимо в поле запрос ввести текст соответствующего запроса. В запросе, в зависимости от контекста, можно использовать параметры (параметр обрамляется специальными символами: /*{имя_параметра}*/). Перед выполнением запроса параметра сначала будет вычислено значение параметра, входящего в запрос, а потом уже сам запрос. При этом, вложенность параметров не ограничена, но следует помнить, что бесконтрольная вложенность может привести к неконтролируемому "размножению" результатов запроса и, как следствие, "размножению" команд, в которых используются параметры. Например, самый простой параметр может выглядеть так:

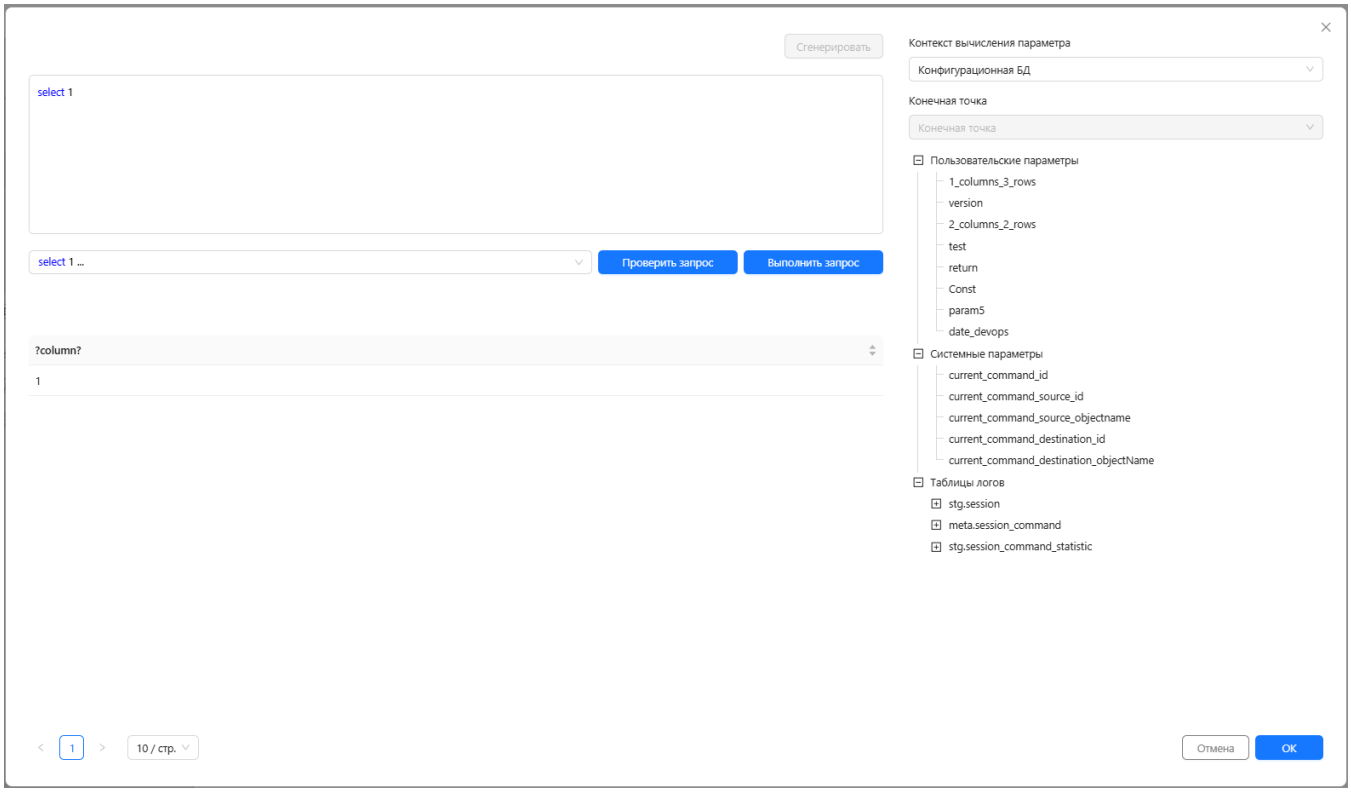
```
Select 1
```

результатом такого запроса является значение 1.

После подготовки кода запроса необходимо нажать кнопку "Проверить запрос". Система выполнит проверку, и если в запрос входит какой-то параметр, то созданный запрос будет размножен в соответствии со списком значений возвращаемых параметром. В данном случае размножения никакого нет.

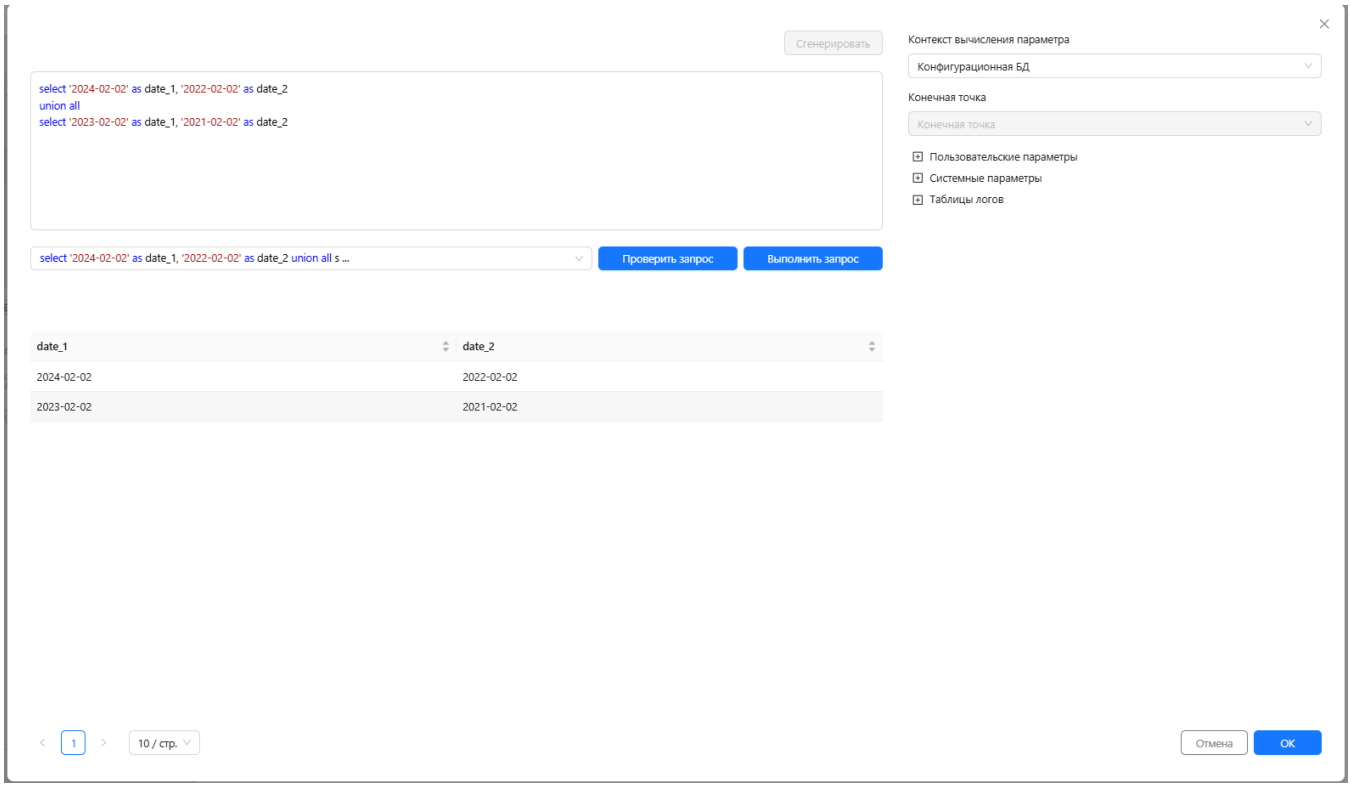


После проверки запроса становится доступной кнопка "Выполнить запрос" и результаты выполнения запроса отобразятся на экране.



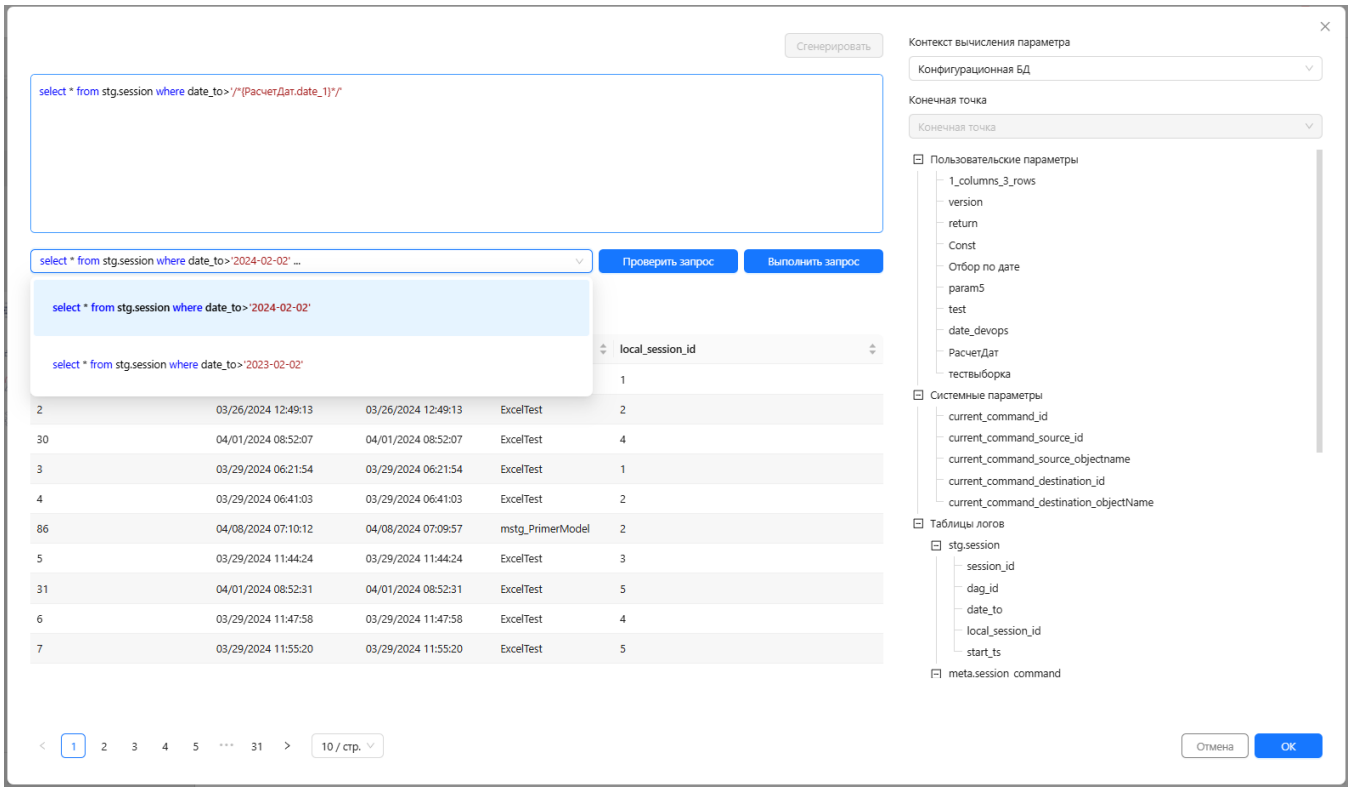
Использование параметров в запросах

Рассмотрим пример создания параметра возвращающего результат в виде двумерной таблицы. Для этого создадим новый параметр с именем "РасчетДат" (пробел в именах не допускается). Контекст "Конфигурационная БД" – цель запроса вернуть набор дат.



Теперь создадим еще один параметр, который будет использовать ранее созданный параметр. Новый параметр извлекает данные из таблицы логов по условию, вычисляемому в ранее созданном параметре. Назначение таблиц логов компонента Metastaging описаны в соответствующем разделе. Вставка имени параметра осуществляется в обрамлении /*{ИмяПараметра}*/, при этом пользователь сам должен понимать особенности синтаксиса языка SQL и добавлять, при необходимости, обрамления в виде кавычек. Кроме этого необходимо явно указать имя атрибута, значения которого требуется использовать в качестве параметров в запросе /*{ИмяПараметра.ИмяАтрибута}*/. В рассматриваемом примере полная запись параметра выглядит следующим образом: '/*{РасчетДат.date_1}*/', где "РасчетДат" – это имя параметра, date_1 – имя атрибута (обратите внимание в данном случае необходимы одинарные кавычки).

После формирования текста кода запроса необходимо нажать кнопку "Проверить запрос", и если параметр в запросе (в данном случае "Расчет дат") возвращает более одной строки, то исходный запрос будет "размножен" в соответствии с количеством строк, возвращаемых параметром. Для проверки работы создаваемого параметра в выпадающем списке нужно выбрать вариант запроса с подходящими подставленными значениями и нажать кнопку "Выполнить запрос". В результате в зоне предварительного просмотра отобразятся результаты работы создаваемого запроса.



Далее, созданный параметр можно использовать в запросах команд метастейджинга и в других объектах, где пользователю доступна возможность создавать SQL-запросы.

ДАННЫЕ

Страница Data (Данные) позволяет пользователю посмотреть визуально загруженные данные в хранилище. Здесь же есть возможность выполнить любые запросы, на основе которых можно убедиться в качестве полученных данных.

Справа в строке необходимо выбрать тот тип загрузки, который выбирали ранее. Затем нажимает на кнопку View (Просмотреть) и раскрываем дерево файлов, нажатием на иконку ( ), находим необходимые данные. Далее нажимаем на конку Form a query (Сформировать запрос) и Run query (Выполнить запрос). В окне снизу появится сформированный запрос (Рисунок. Просмотр загруженных данных).

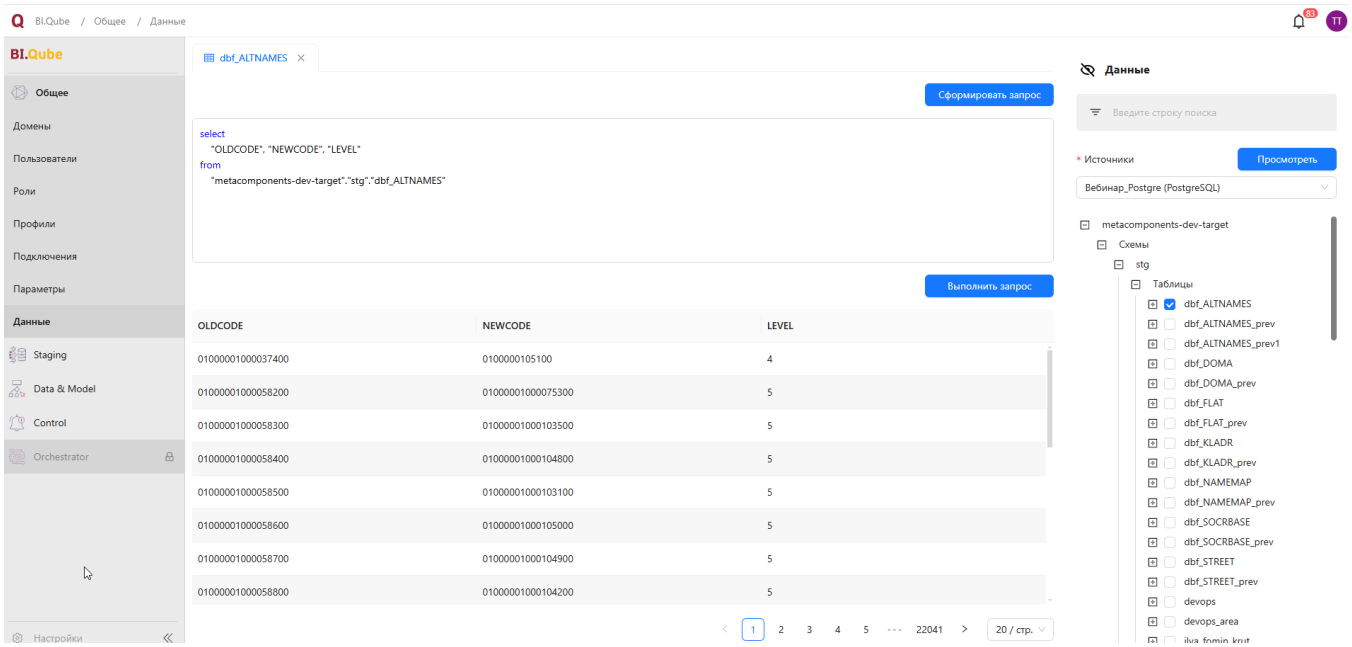


Рисунок. Просмотр загруженных данных

В режиме s3 есть специальная форма запроса, в которой не только можно выбрать сам файл и сформировать запрос, но и выбрать необходимый лист для загрузки и настройки для выбранного файла (Рисунок. Форма запроса для S3).

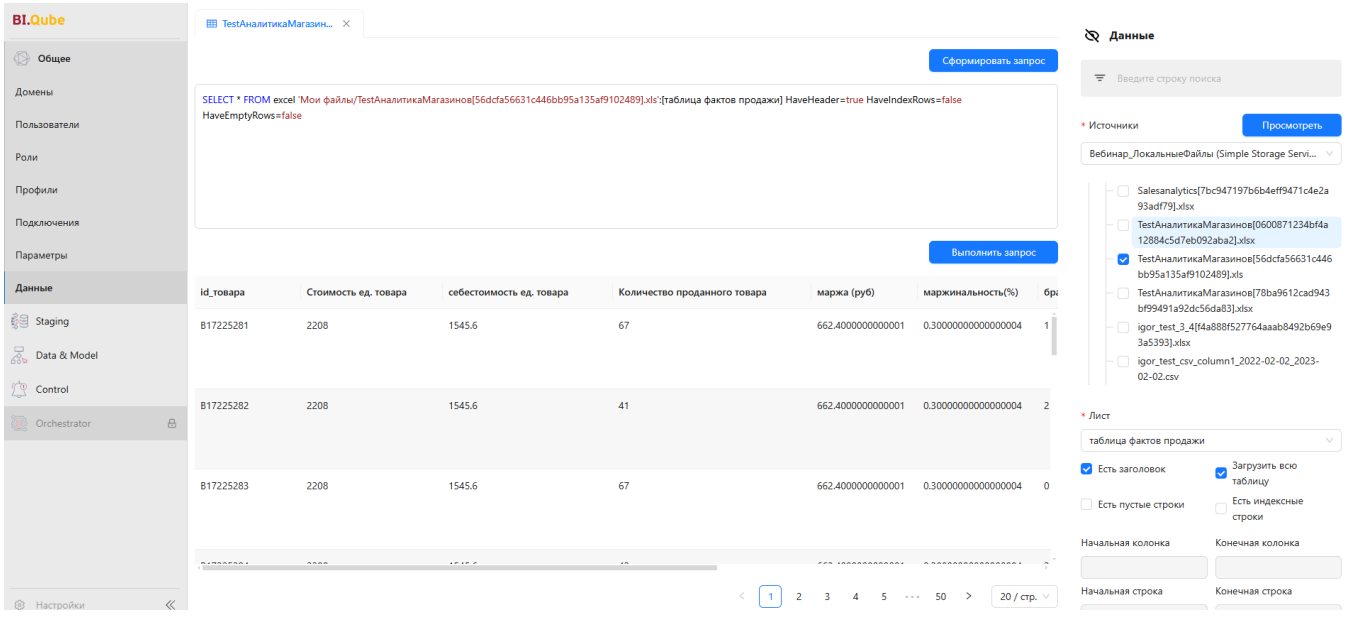


Рисунок. Форма запроса для S3

Здесь же есть возможность создавать хранимые процедуры и другие объекты базы данных, необходимые для поддержки работы хранилища.

METASTAGING

Общие сведения

При построении хранилищ данных наиболее частой задачей является извлечение данных из источника и их копирование в слой, предназначенный для хранения. Под таким слоем в зависимости от целевой архитектуры понимают DataLake, детальный слой данных (DDS), стейджинговый слой – далее обобщенно этот слой называется стейджингом. Более простыми словами можно сказать, что это может быть либо файловое хранилище данных, либо реляционное хранилище данных. При этом, в этом слое данные обычно хранятся в том виде, как они представлены в источнике. MetaStaging поддерживает достаточно сложные сценарии создания детального слоя:

1. детальный слой формируется полностью в реляционном слое – наиболее распространенный подход к организации подготовки детального слоя;
2. формируется озеро данных (data lake) – файловое хранилище, файлы представлены в формате *.parquete, с автоматическим формированием в реляционном слое объектов External Table с возможностью материализации;
3. дополнительно к первым двум сценариям есть возможность сохранения истории загрузок данных в оригинальном формате, представленных на источниках в форматах xls, xlsx, cxv, xml, json.

Другими словами, можно сказать, что MetaStaging предназначен для консолидации данных в стейджинговом слое хранилища данных из гетерогенных источников с поддержанием целостности и унифицированности метаданных. Также уменьшает нагрузку на операционные базы данных при выполнении запросов и, кроме того, обеспечивает надежное подключение различных БД из разнородных источников для помещения данных в единый слой стэйджинга (staging area) с поддержанием целостности метаданных в системе-назначения.

Система поддерживает два режима выполнения команд:

- с использованием веб-интерфейса;
- с использованием планировщика (оркестратора), рекомендуется применять Airflow.

В текущем руководстве рассмотрена работа в режиме веб-интерфейса.

Описание компонента

ПРОФИЛЬ METASTAGING

Для просмотра созданных профилей необходимо зайти в раздел Стейджинг (Staging) во вкладку Профили (Profiles).

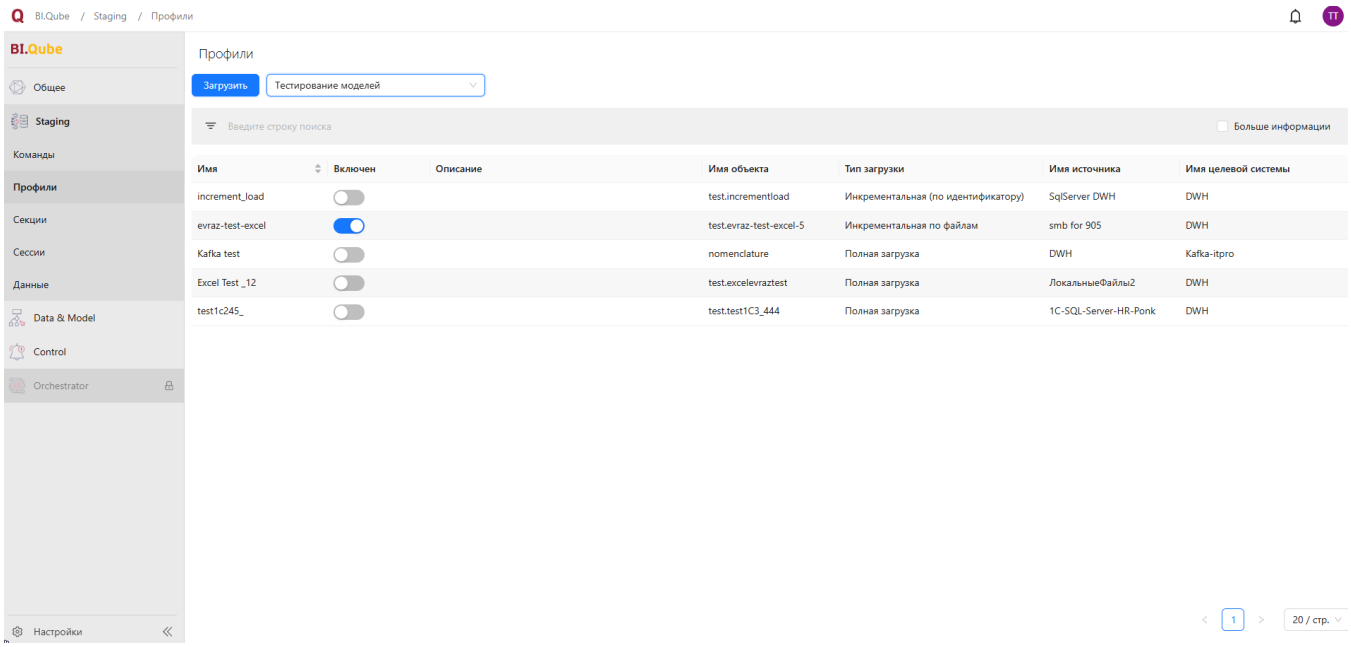


Рисунок. Пример созданного профиля

Для просмотра и выбора, необходимо выбрать нужный профиль в выпадающем списке. (Рисунок. Выбор профиля).

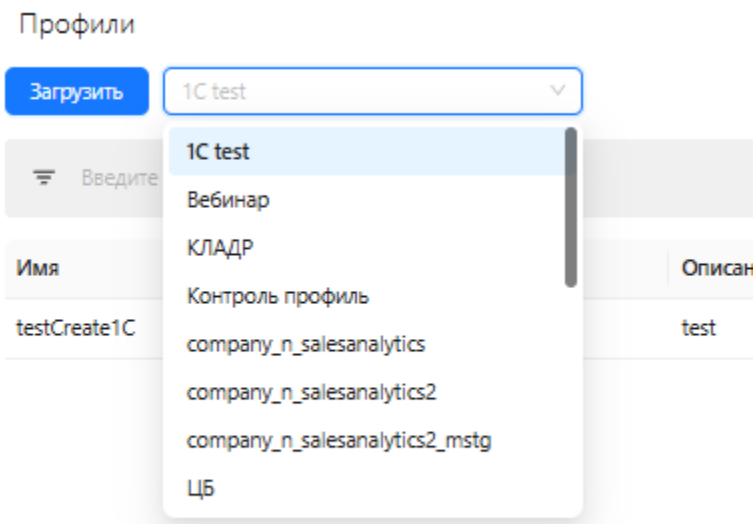


Рисунок. Выбор профиля

После выбора интересующего имени профиля на экране появится перечень команд, включенных в этот профиль. Более детально про работу с профилями будет рассказано в разделе "Запуск на выполнение".

Для просмотра дополнительной информации по сущностям (таблицам) необходимо поставить галочку More info (Больше информации).

СОЗДАНИЕ И РЕДАКТИРОВАНИЕ КОМАНД ДЛЯ ЗАГРУЗКИ ДАННЫХ

Общие настройки

При создании команды для загрузки данных из источника в точку назначения потребуется некоторый опыт работы с СУБД и начальное понимание SQL-запросов. Создание команды выполняется на странице "Команды" (Commands)

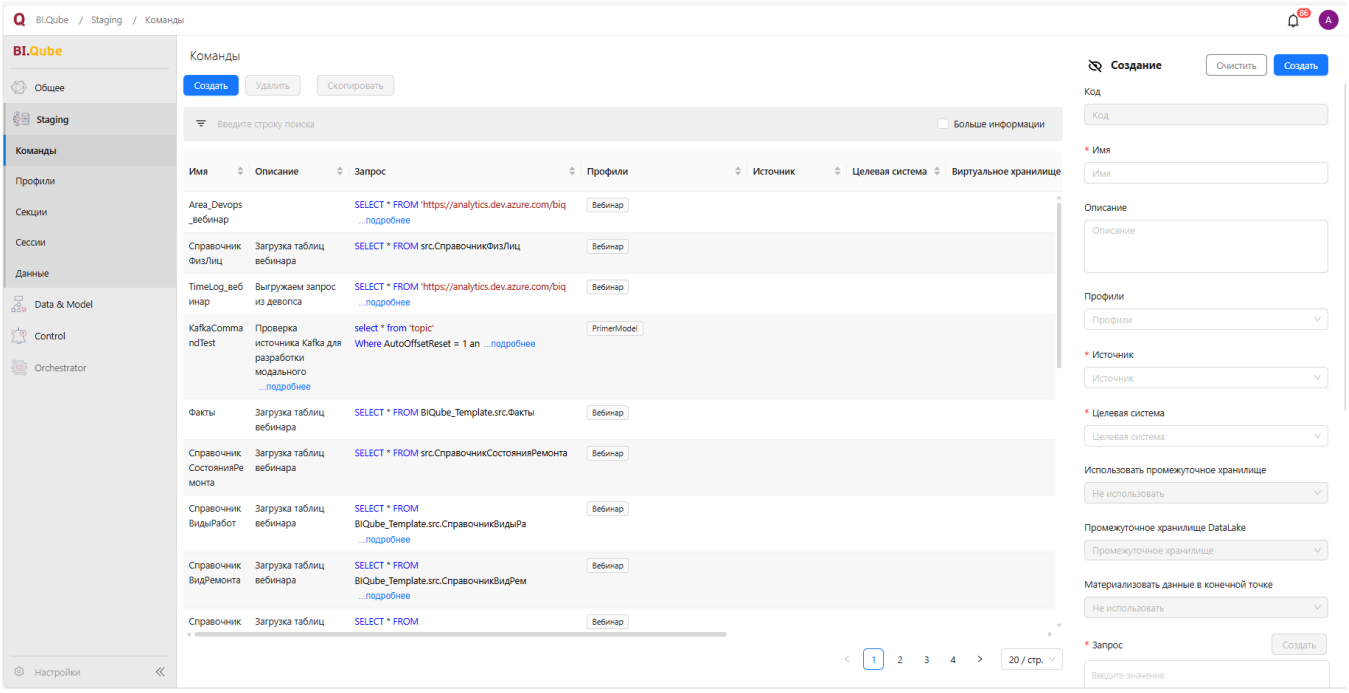


Рисунок. Страница работы с командами

Для создания новой команды необходимо нажать кнопку "Создать" (Create), после чего появится возможность заполнить все необходимые свойства создаваемой команды. Если необходимо создать команду с настройками, которые были созданы ранее в других таблицах, можно нажать кнопку "Скопировать" (Copy) и все свойства для новой команды будут скопированы из той, которая была выбрана в таблице основной части экрана. Здесь нужно быть внимательным и обязательно проверить правильность заполнения всех полей. Команды с одним именем не допускаются.

Создание команды

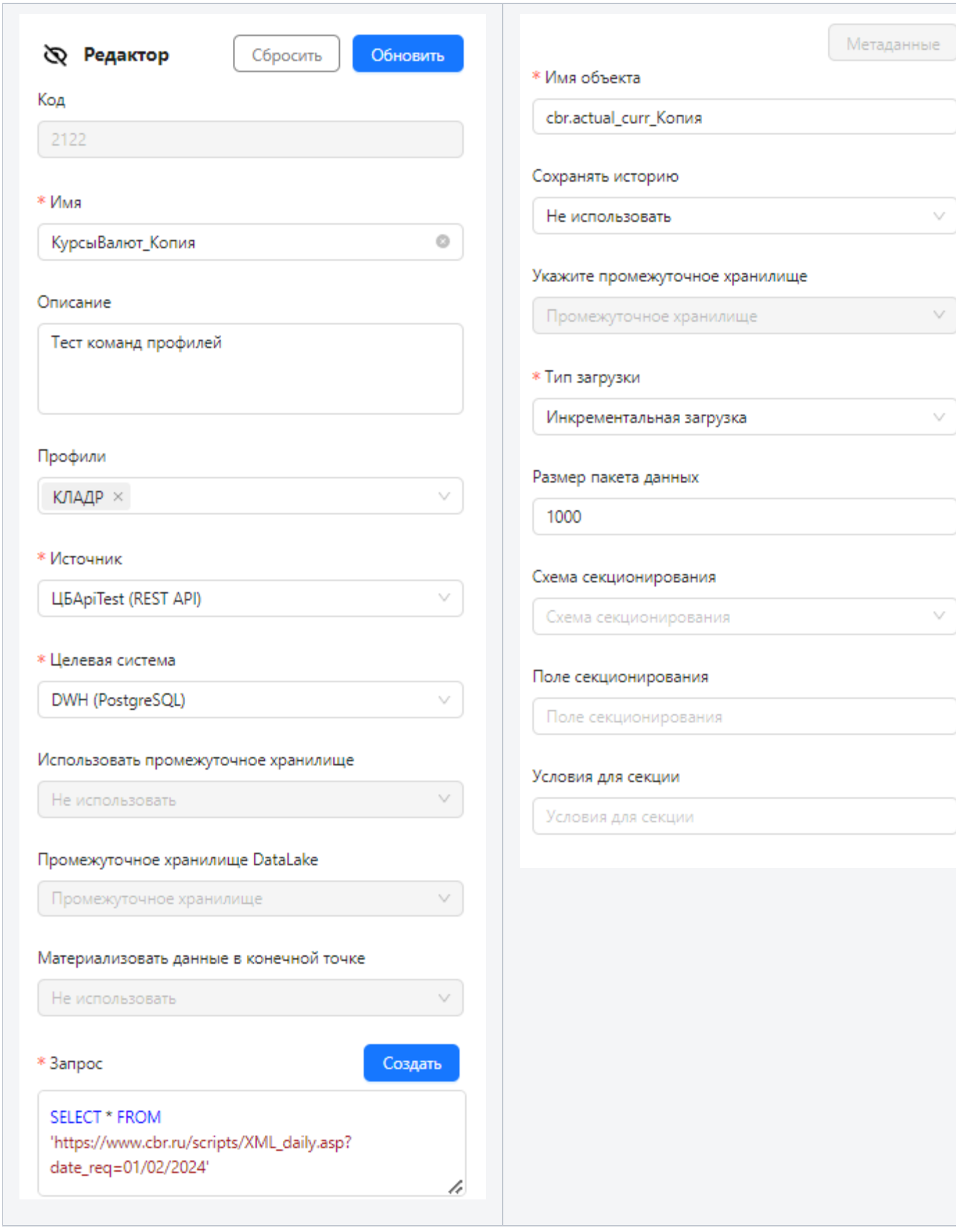


Рисунок. Пример настроенной команды

Для создания настроек параметров команды необходимо заполнить следующие поля:

1. Name (Имя) – уникальное наименование команды без пробелов;
2. Description (Описание) – бизнес-описание команды;
3. Profiles (Профили) – команда помещается в один или более профилей (контейнеров);
4. Source (Источник) – система – источник данных для загрузки (для удобства пользователя осуществлена группировка по типам источников);
5. Destination (Целевая система) – система, в которую планируется загрузить данные из источников (для удобства пользователя осуществлена группировка по типам целевой системы);
6. Use intermediate storage (Использовать промежуточное хранилище) – иногда при построении хранилищ требуется использовать дополнительное промежуточное файловое хранилище, например S3, используя данную опцию можно организовать доставку данных сначала в одно хранилище, а затем в следующее с использованием одной команды;
7. Intermediate storage DataLake (Промежуточное хранилище DataLake) – выбирается endpoint для промежуточного хранилища;
8. Materialize data at endpoint (Материализовать данные в конечной точке) – в случае использования опции «Использовать промежуточное хранилище» есть возможность сгенерировать External Table в конечной точке или генерировать таблицы с данными, которые продублированы в промежуточном хранилище;
9. **Query (Создать)** – запрос к источнику данных (ниже приведено детальное описание возможных вариантов);
10. Destination object (Имя объекта) – наименование объекта в точке назначения, для реляционного слоя задается в формате ИмяСхемы.ИмяТаблицы;
11. Save history (Сохранять историю) – использовать/не использовать;
12. Specify intermediate storage (Укажите промежуточное хранилище) – выбрать из выпадающего списка нужное хранилище, в которое будут сохранятся результаты всех выполнений команд;
13. Load type (Тип загрузки) – тип загрузки (инкрементальная, инкрементальная по дате, инкрементальная по идентификатору, перезагрузка таблицы, полная загрузка, полная с сохранением истории);
14. Batch size (Размер пакета данных) – размер пакета данных;
15. Partition schema (Схема секционирования) – задается исходя из назначения секции и зависит от типа секционирования;
16. Partition column (Поле секционирования) – поле, по которому осуществляется секционирование;
17. Partition column convert (Условия для секции) – условия, характерные для выбранной секции.

После заполнения всех полей ввода необходимо нажать на кнопку Create (Создать) в верхней части меню свойств. В строках «Команды» появиться таблица с создаваемым именем.

Возможность выбора промежуточного хранилища и выбора опции сохранения истории доступны не всем endpoints!

Команда "Создать"

Команда Query (Запрос) открывает диалоговое окно для пользователя, в котором создается запрос (команда), которая будет выполнена на стороне endpoint для извлечения данных. Окно создания запроса зависит от типа endpoint:

- запрос извлечения файлов с компьютера пользователя (xls, xlsx, csv) – подключение «Локальные файлы» создается при развертывании;
- запрос извлечения данных из 1С Предприятие (на основе MS SQL Server, PostgreSQL) – если тип подключения соответствует выбранному типу подключения 1С;
- запрос извлечения данных из СУБД (MS SQL Server, Oracle, MySQL, PG, GP) – если тип подключения соответствует чему-то из перечисленного;
- запрос извлечения данных из веб-сервисов по протоколу REST API (JSON, XML, CSV) – если выбран тип подключения Rest API;
- запрос извлечения данных из брокера сообщений Kafka;
- запрос извлечения данных из общих каталогов windows (xls, xlsx, csv) – если тип подключения соответствует протоколу SMB.

Тип нужного диалогового окна определяется автоматически, на основе выбранного endpoint, используемого в качестве источника данных. Диалект SQL запроса зависит от типа источника данных, запрос будет выполнятся на стороне источника.

Для удаления или копирования уже созданной команды, необходимо нажать на соответствующие кнопки.

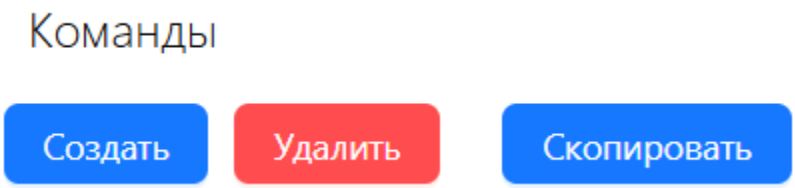
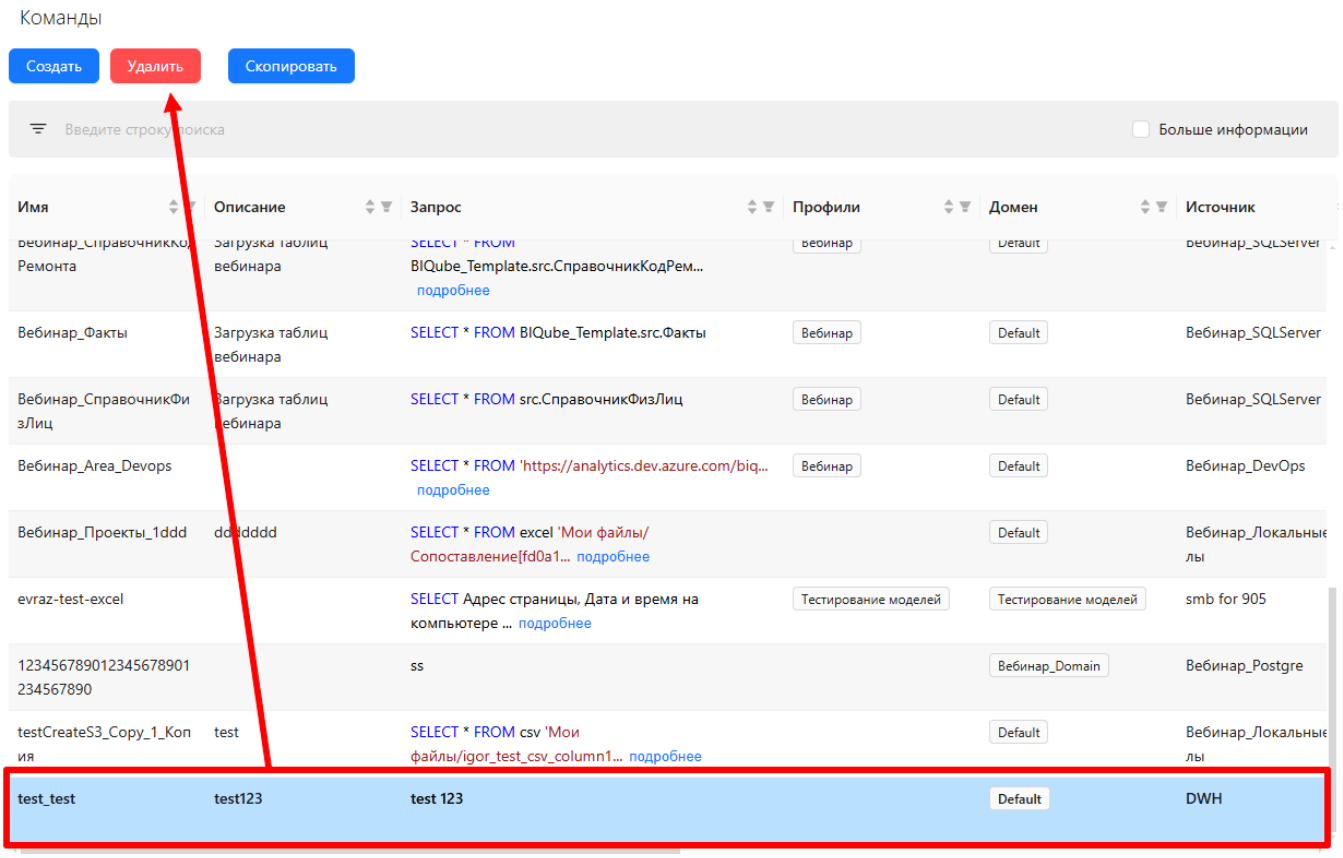


Рисунок. Кнопки создания/удаления/копирования команды

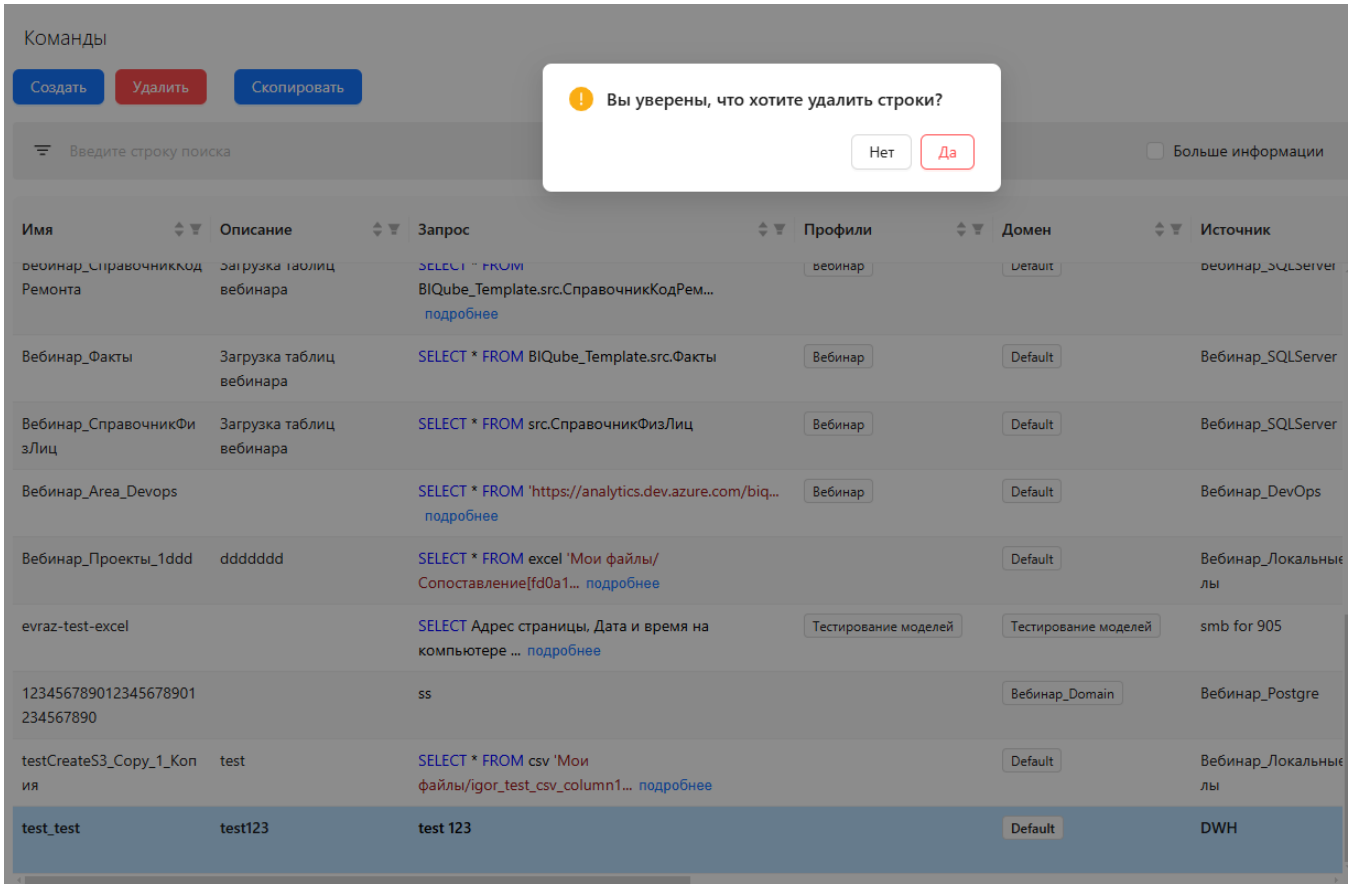
Удаление команды

Для **удаления** команды необходимо выполнить три шага:

ШАГ 1: выделить строку команды и в верхней панели нажать кнопку "Удалить" (Delete);



ШАГ 2: в появившемся модальном окне подтвердить намерение об удалении, нажав кнопку "Да";



ШАГ 3: проверить результат корректного удаления.

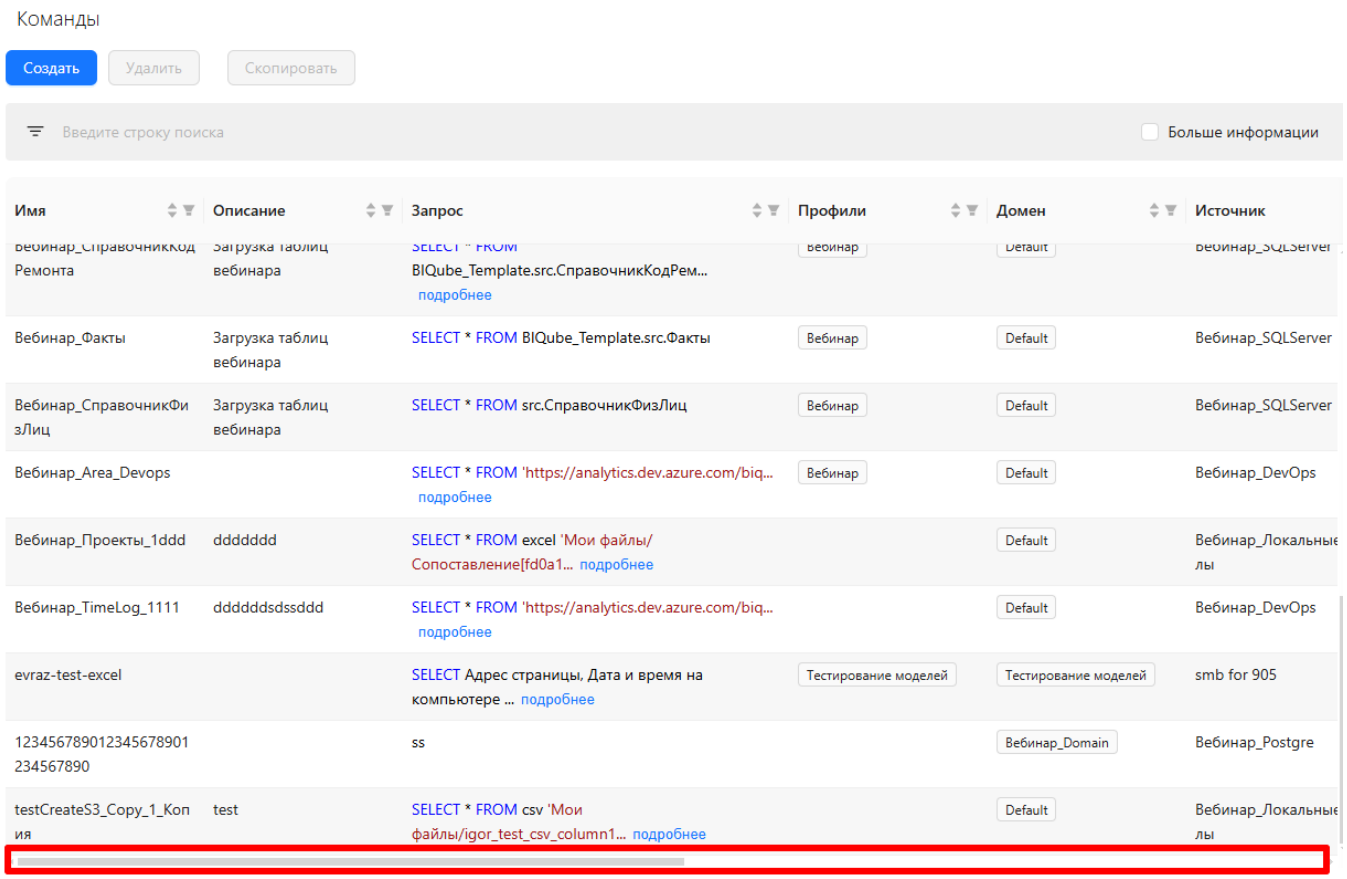


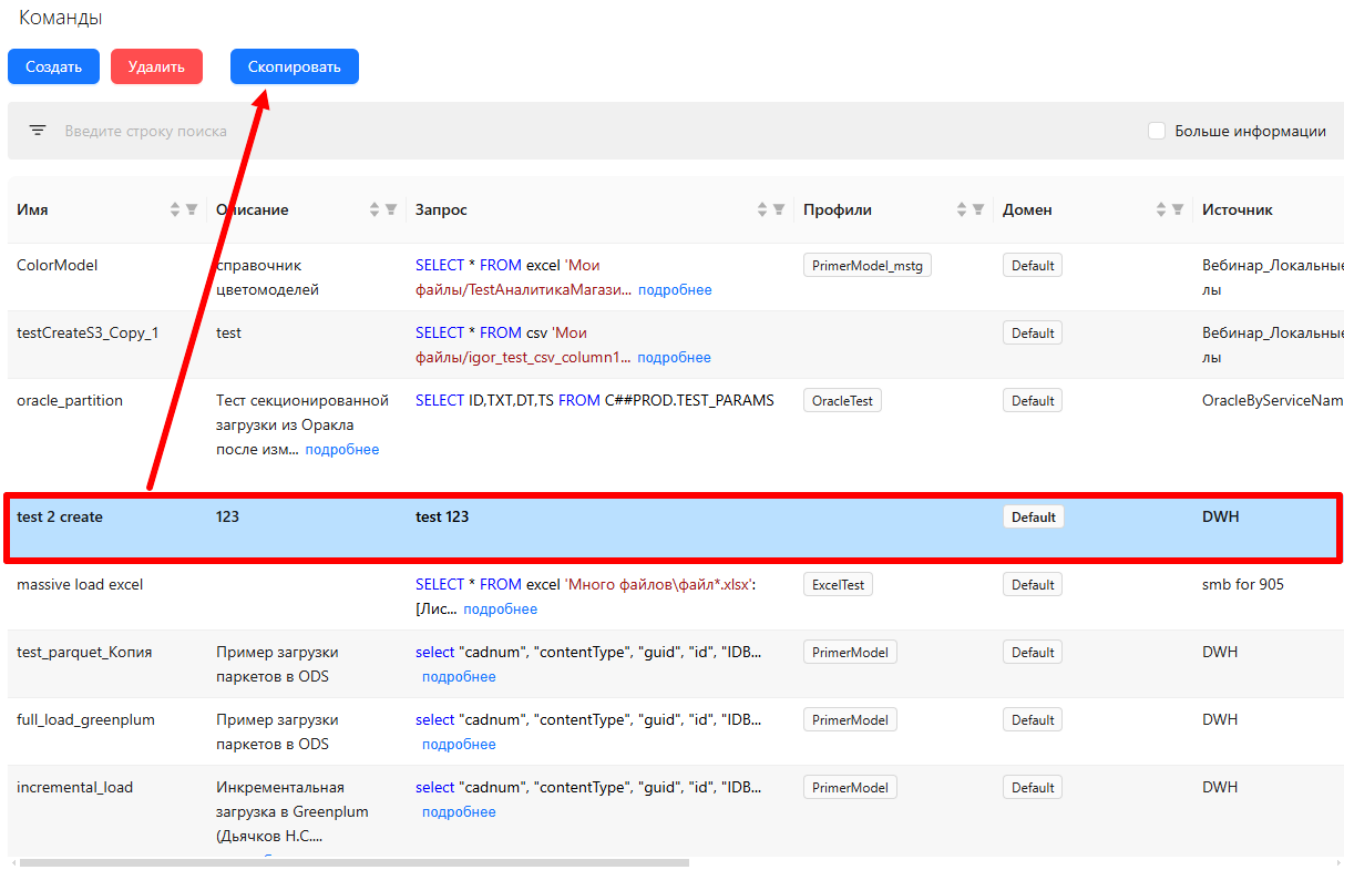
Рисунок. Удаление команды из списка

> Команда удаления не производит физическое удаление из настроечных таблиц, а лишь помечает на удаление и скрывает команду от пользователя.

Копирование команды

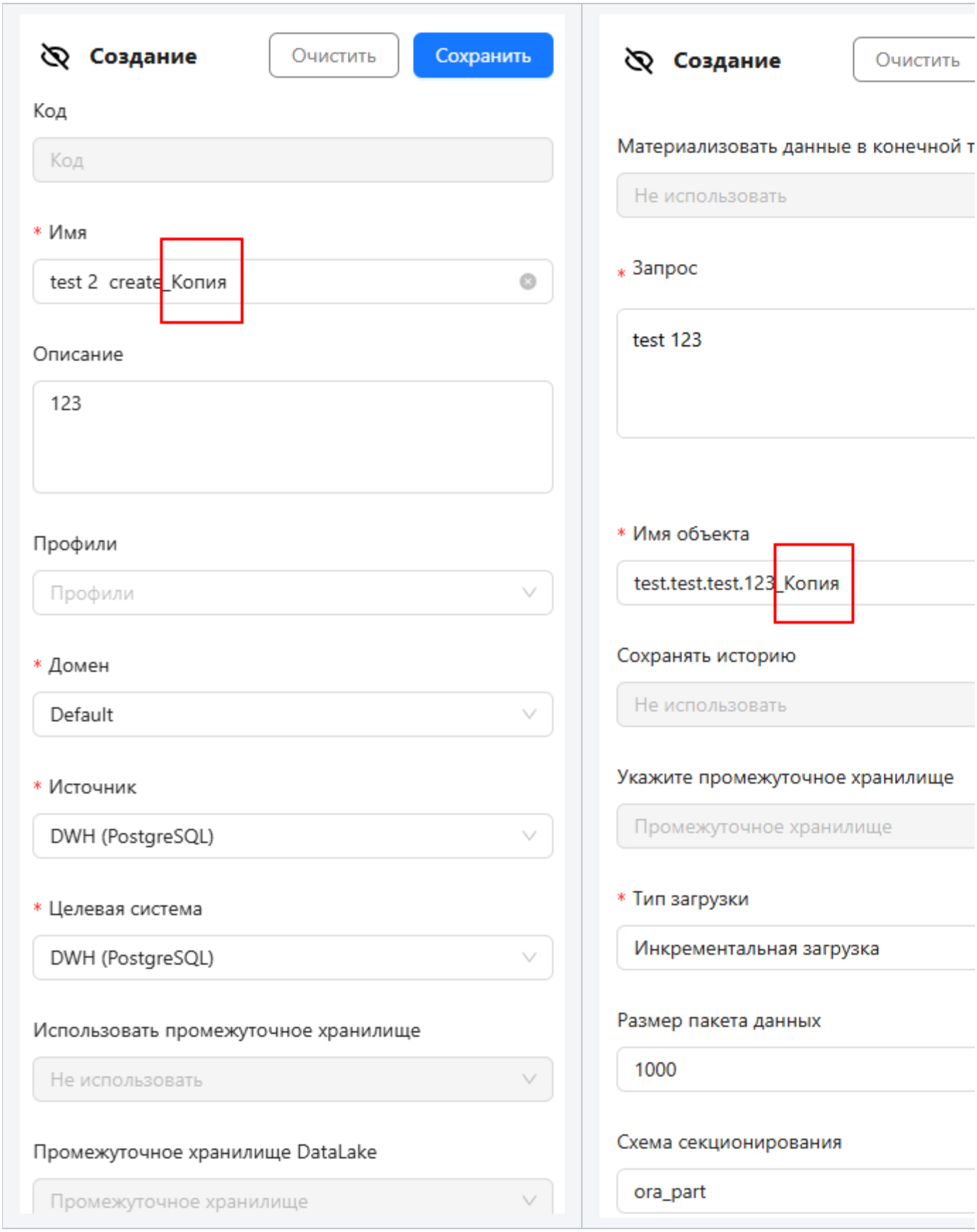
Для **копирования** уже существующей команды необходимо выполнить три шага:

Шаг 1: выделить строку команды с настройками и в верхней панели нажать кнопку "Скопировать" (Copy);

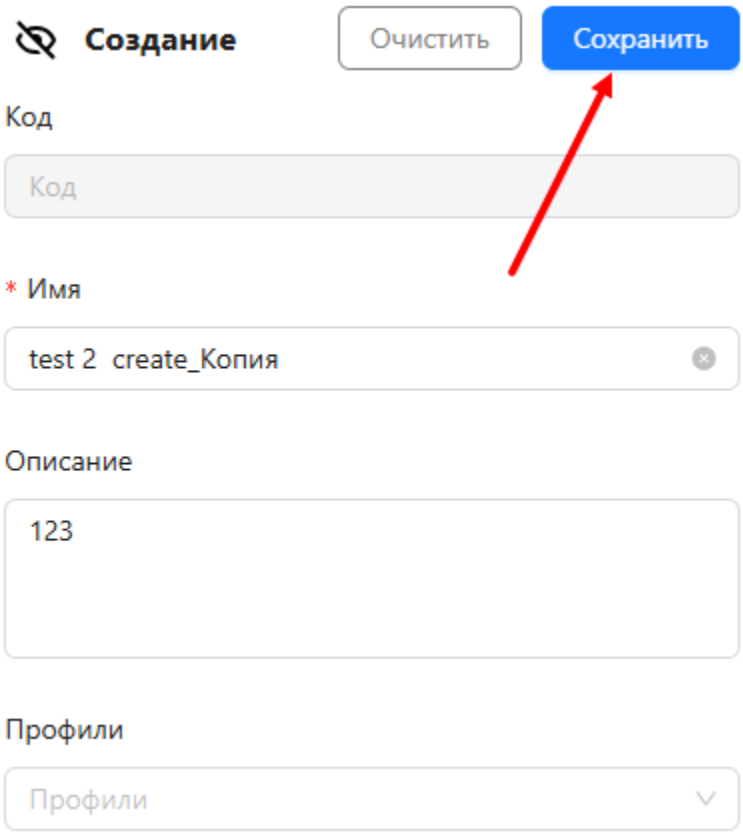


ШАГ 2: при необходимости изменить автоматически заполненные свойства создаваемой команды.

Команды с одним именем не допускаются, поэтому система автоматически дописывает "_Копия" для полей ввода Имя (Name) и Имя объекта (Destination object).



ШАГ 3: сохранить настройки новой команды, нажав кнопку "Сохранить" (Save). Новая команда отобразится в списке всех команд.


Создание
Очистить
Сохранить
Код
Код
* Имя
test 2 create_Копия
Описание
123
Профили
Профили

Запрос извлечения файлов с компьютера пользователя

После нажатия кнопки Create (Создать) появится диалоговое окно, в котором можно оформить запрос на извлечение данных из источника в диалоговом режиме или ввести запрос с клавиатуры на языке источника данных или, в отдельных случаях, на внутреннем языке системы.

Окно создания запроса разделено на две зоны: слева зона отображения кода запроса к источнику, под ним зона предварительного просмотра результата запроса и зона создания кода запроса.

Для загрузки файла с локального компьютера пользователя, файл необходимо обязательно поместить в промежуточное хранилище, чаще всего это хранилище типа S3, endpoint для которого должен быть создан заранее. В правой зоне окна создания запроса отображается файловая структура выбранного хранилища и файлы доступные в хранилище.

Для загрузки файла в хранилище, если нужного файла еще нет, нужно нажать кнопку Upload file (Загрузить файл), в результате откроется стандартное диалоговое окно Windows выбора файла. Далее необходимо выбрать интересующий файл на компьютере пользователя с использованием открытого диалогового окна и щёлкнуть по кнопке «ОК». Выбранный файл автоматически загрузится в хранилище и подсветится в структуре каталогов. С этого момента файл доступен для анализа.

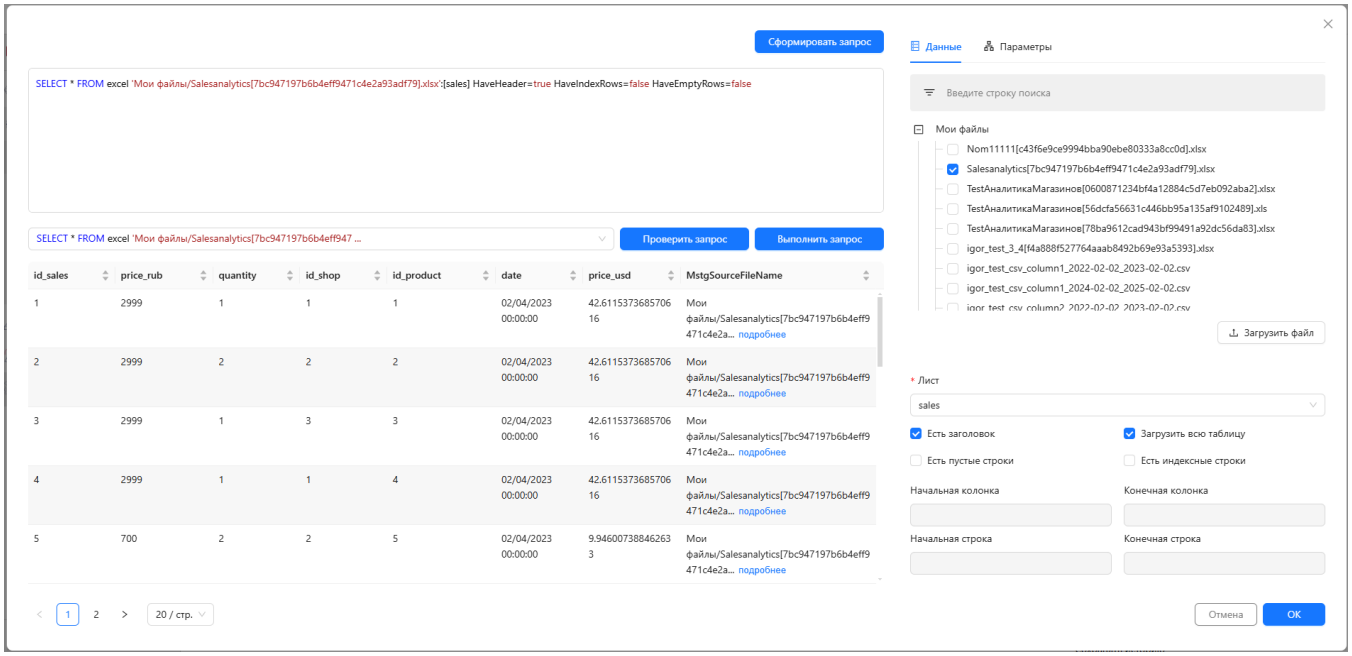


Рисунок. Загрузка файла с компьютера пользователя

Для настройки команды загрузки выбранного файла необходимо в выпадающем списке Sheet (Лист) выбрать лист, данные из которого необходимо будет загрузить, после чего задать нужные опции:

- Have header (Есть заголовок) – опция, позволяющая первую строку диапазона данных использовать как строку заголовков таблицы;
- Load full table (Загрузить всю таблицу) – автоматическое определение диапазона данных;
- Have empty rows (Есть пустые строки) – позволяет из диапазона данных удалять пустые строки;
- Have index rows (Есть индексные строки) – добавляется колонка с номерами строк.

Если опция Load full table (Загрузить всю таблицу) не выбрана, то пользователь может ввести нужный диапазон данных вручную. Для этого указывается Начальная колонка таблицы (Column start), Конечная колонка (Column end), Начальная строка (Have empty rows) и конечная строка (Have index rows). Конечную строку заполнять не обязательно, определяется автоматически. Если опция Load full table (Загрузить всю таблицу) выбрана, данные поля становятся неактивными.

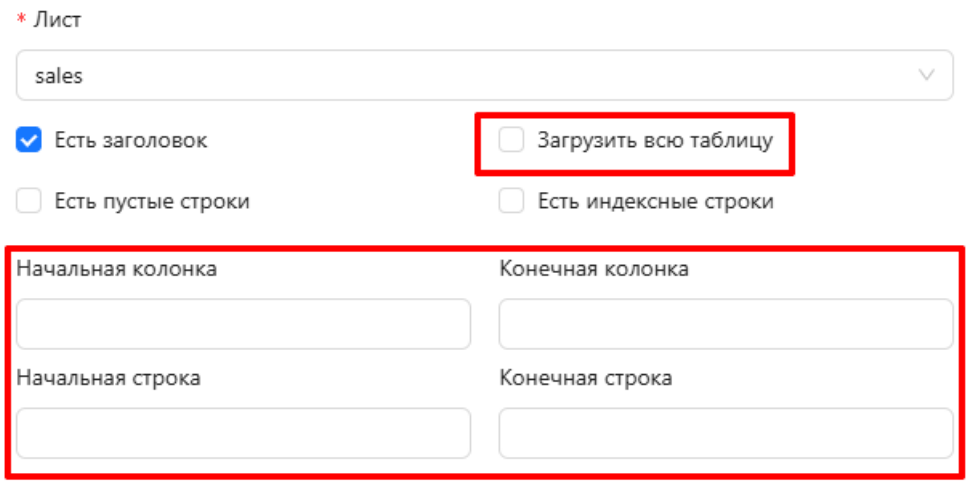


Рисунок. Ручной ввод диапазона данных таблицы

Для просмотра результирующего запроса и результатов его работы необходимо нажать на кнопку Form a query (Сформировать запрос) – программа сформирует простой SQL запрос, а затем необходимо нажать на кнопку Run query (Выполнить запрос), сформируется текст запроса и в зоне предварительного просмотра появятся результаты выполнения этого запроса.

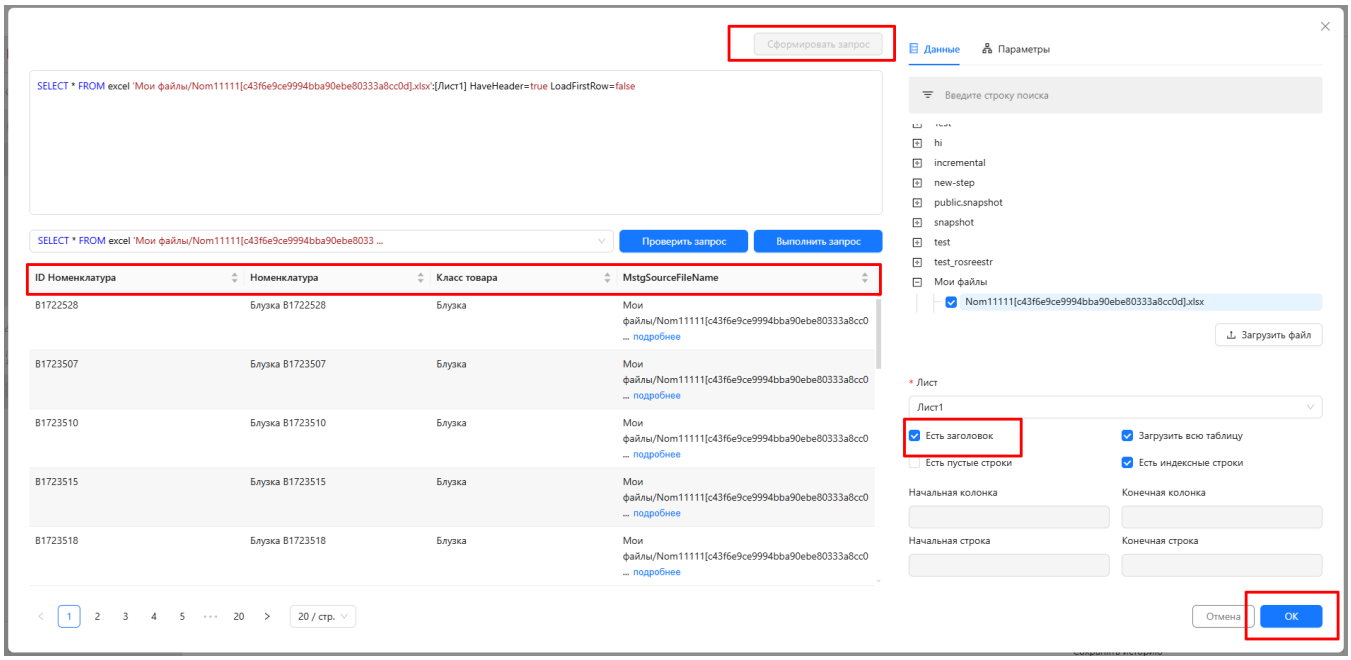


Рисунок. Сверка загруженного файла и добавление заголовков в таблицу

После окончания настройки запроса следует нажать кнопку «ОК», данный запрос загрузит данные из файла в хранилище при запуске команды на выполнение.

Запрос извлечения данных из 1С Предприятие

Создание запроса

Для создания команды загрузки данных из 1С Предприятие, должен быть выбран соответствующий endpoint в выпадающем списке Source (Источник). После нажатия на кнопку Create (Создать), автоматически сформируется окно для настройки команды. В этом окне справа расположено дерево объектов 1С той конфигурации данных, к которой настроен endpoint. Так же над деревом объектом доступны следующие настройки:

- Режим работы (1С или SQL) – определяет на каком языке будет сформирован запрос извлечения данных.
- Режим связей – определяет режим отображения подсказок о том, с чем связано поле. Режим "Все связи" показывает перечень всех возможных связей. Режим "Фактические связи" – позволяет увидеть, какие связи реально есть. Например, в какой-то регистр могут записывать огромный перечень видов документов, однако в текущем учете в организации используется по факту какой-то ограниченный перечень видов документов (определяется по содержимому этого регистра) и во втором режиме только фактический список будет отображен.

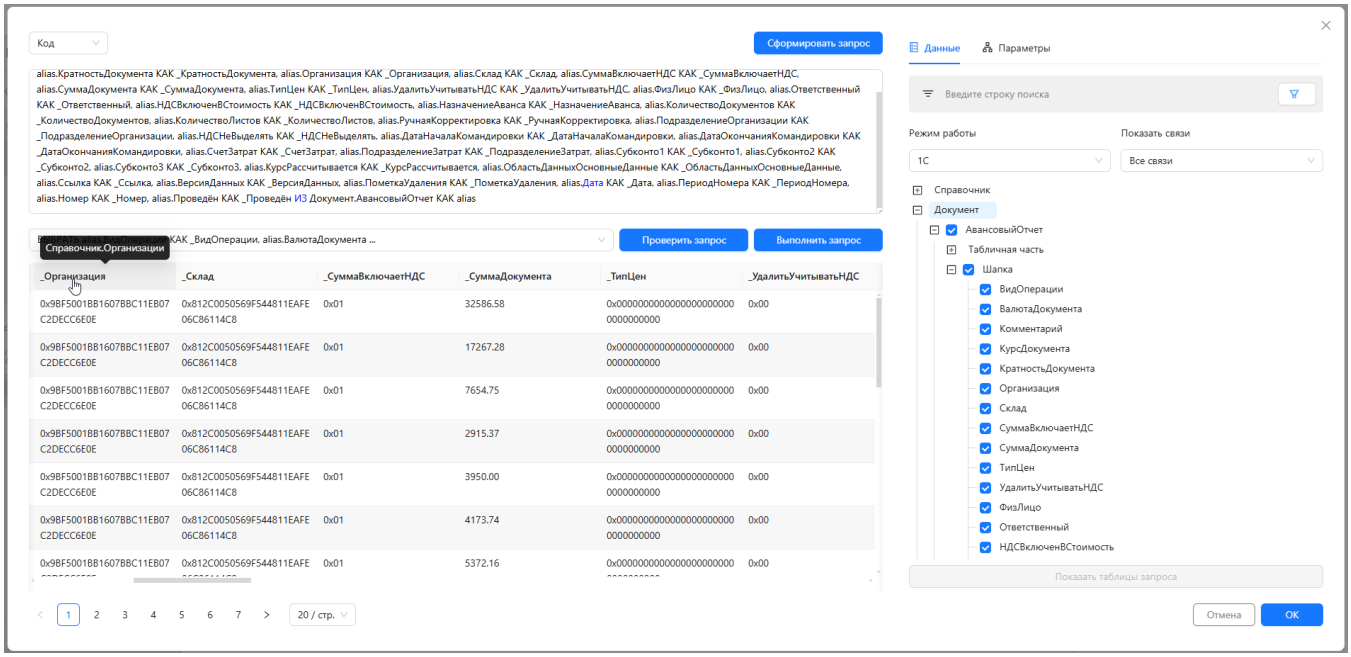


Рисунок. Диалоговое окно при создании команды загрузки данных из 1С

Для автоматического формирования текста запроса необходимо выбрать интересующий объект 1С. При этом следует помнить, что некоторые объекты 1С представлены одной таблицей. Например, справочники некоторые представлены набором таблиц (документы). В связи с этим необходимо понимание данные из какого объекта и связанные с этим объектом нужны пользователю.

После выбора нужного объекта необходимо нажать кнопку Form a query (Сформировать запрос), в результате чего будет сформирован запрос. Затем нажать на кнопку Check request (Проверить запрос) и Run query (Выполнить запрос). Данные из выбранного объекта появятся в зоне предварительного просмотра.

Анализ связей в графическом режиме

Кроме этого, для анализа связей между объектами 1С предусмотрен графический режим работы. В этом режиме пользователь может увидеть с какими объектами связан выбранный объект (простыми словами можно сказать так: из каких связанных таблиц загружаются данные в выбранную таблицу).

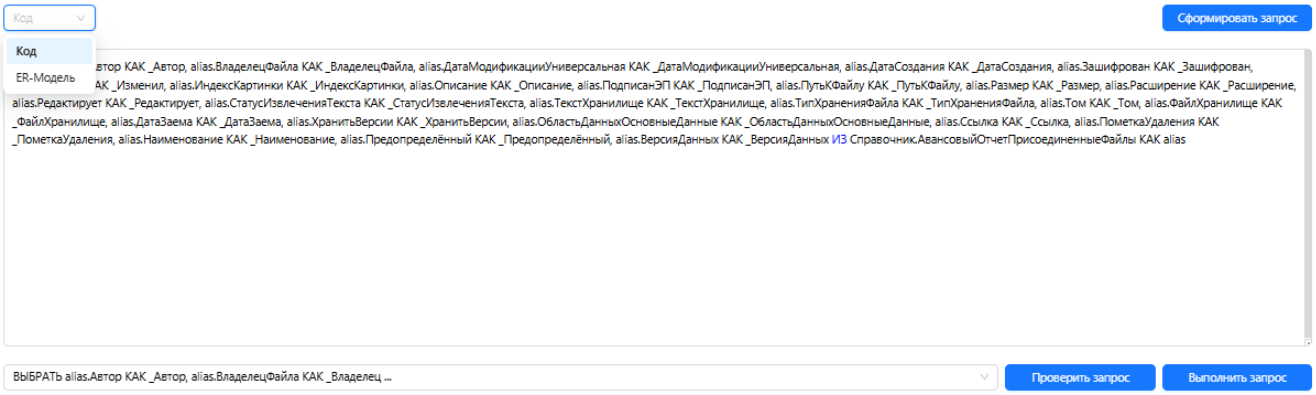


Рисунок. Выбор вариантов отображения

В графическом режиме доступно два вида отображения:

- концептуальный – в этом режиме все объекты, в том числе и сложные (составные) объекты представляются одним графически элементом и отображаются связи между ними, таким образом можно понять, например, на какие справочники ссылается выбранный документ;
- детальный – в этом режиме все объекты отображаются в отдельном графическом объекте, с перечислением атрибутов и указанием по каким атрибутам установлены связи.

Следует помнить, что отображаются все связи выбранного объекта, связи между зависимыми объектами не отображаются. Таким образом для детального анализа связей необходимо исследовать каждый объект по отдельности.

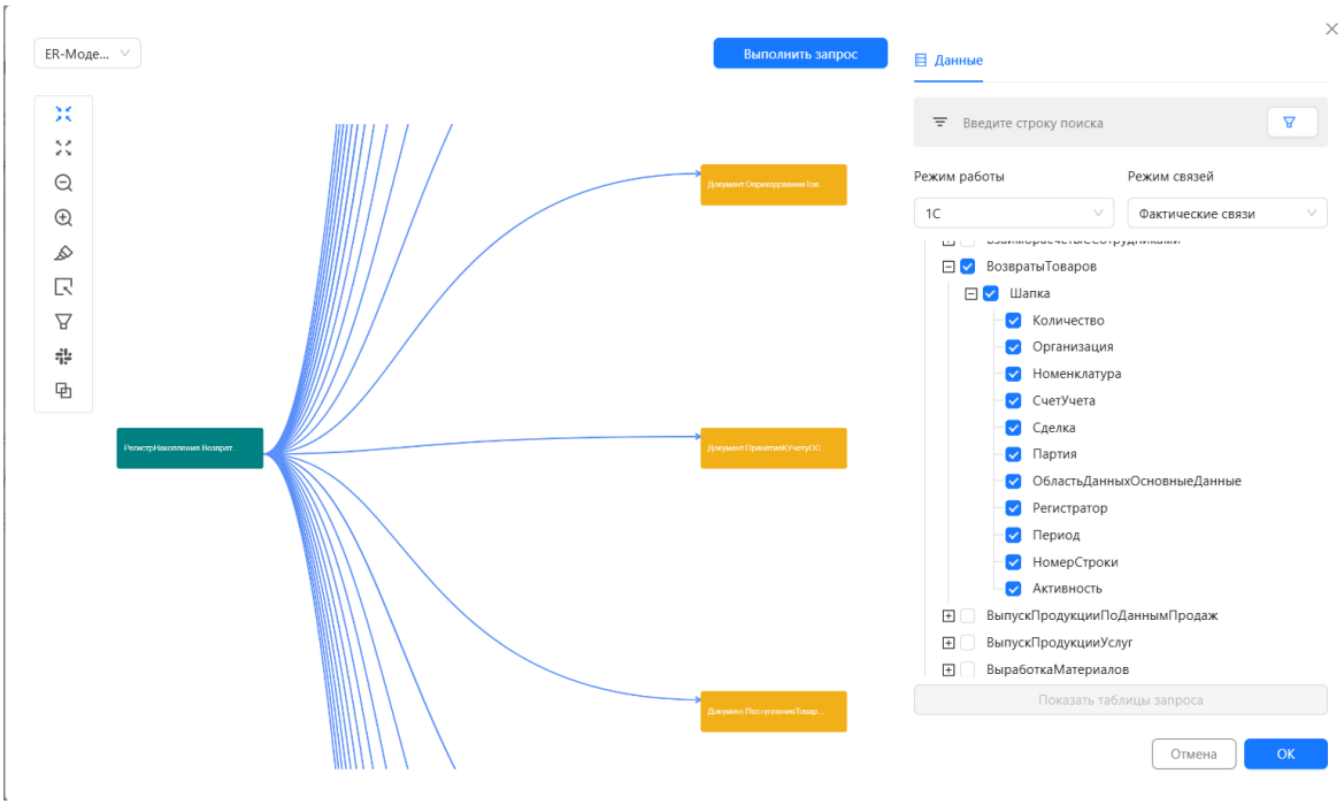


Рисунок. Концептуальная ER – модель отображения данных

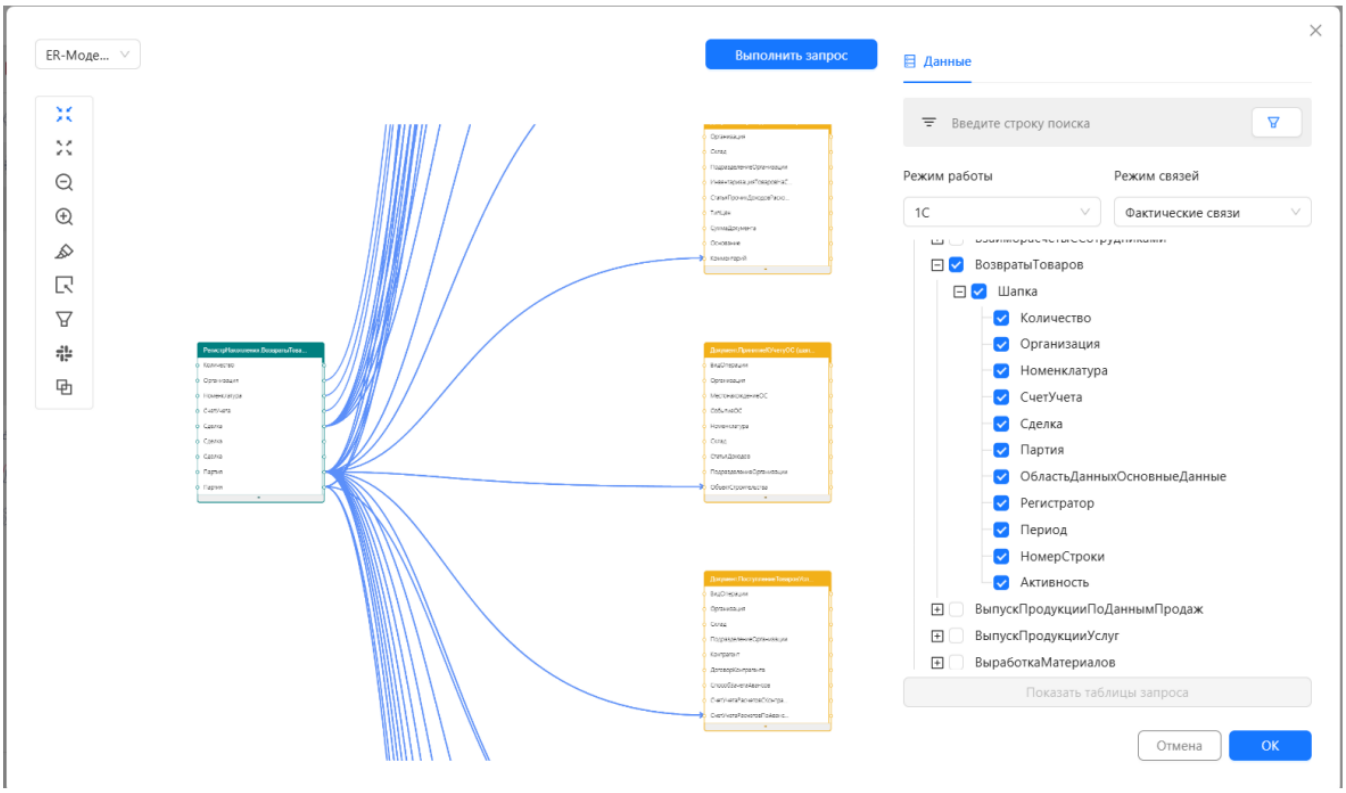


Рисунок. Детальная ER – модель отображения данных

Работа с окном заканчивается после того, как настроен запрос и нажата кнопка «ОК» в правом нижнем углу модального окна.

Поиск и фильтрация объектов конфигурации 1С

В окне справа в самом верху реализована строка поиска и фильтрации, для удобства поиска необходимых таблиц. При нажатии на символ  появляется диалоговое окно для ввода данных, по которым будет проведена фильтрация всего дерева объектов 1С.

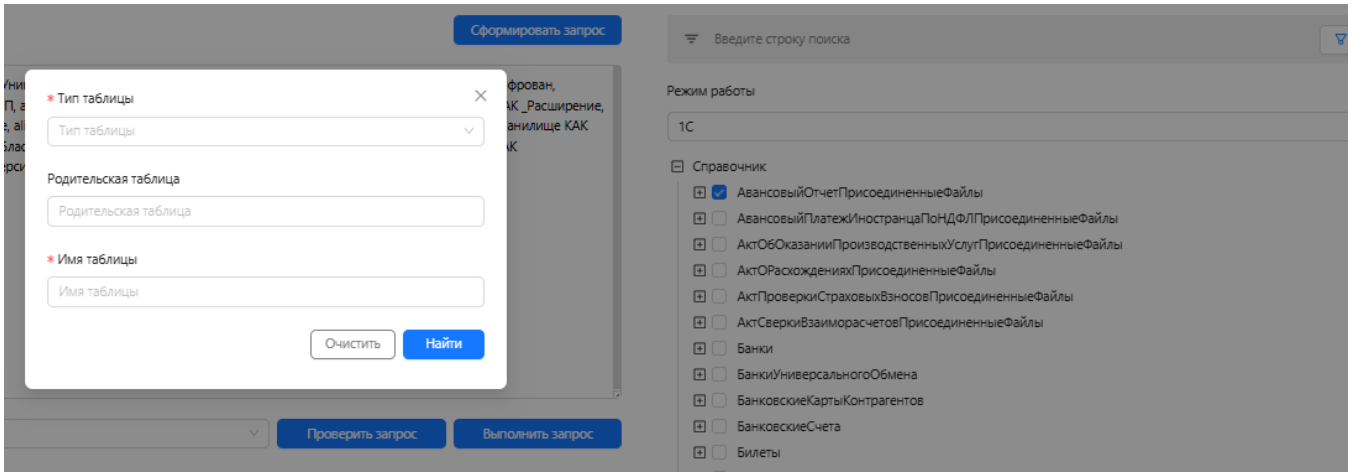


Рисунок. Диалоговое окно фильтрации данных

Запрос извлечения данных из СУБД

Для создания команды загрузки данных из СУБД, должен быть выбран соответствующий endpoint в выпадающем списке Source (Источник). После нажатия на кнопку Create (Создать), автоматически сформируется окно для настройки команды. В этом окне справа расположено дерево объектов СУБД той конфигурации данных, к которой настроен endpoint.

Далее нужно выбрать объект СУБД и нажать кнопку Form a query (Сформировать запрос) – программа сформирует простой запрос на выборку всех данных. Данный запрос можно редактировать, при этом, следует помнить, что нотация SQL запроса зависит от выбранного endpoint. Затем необходимо нажать на кнопку Check request (Проверить запрос) и Run query (Выполнить запрос). В зоне предварительного просмотра появятся результаты выполнения этого запроса.

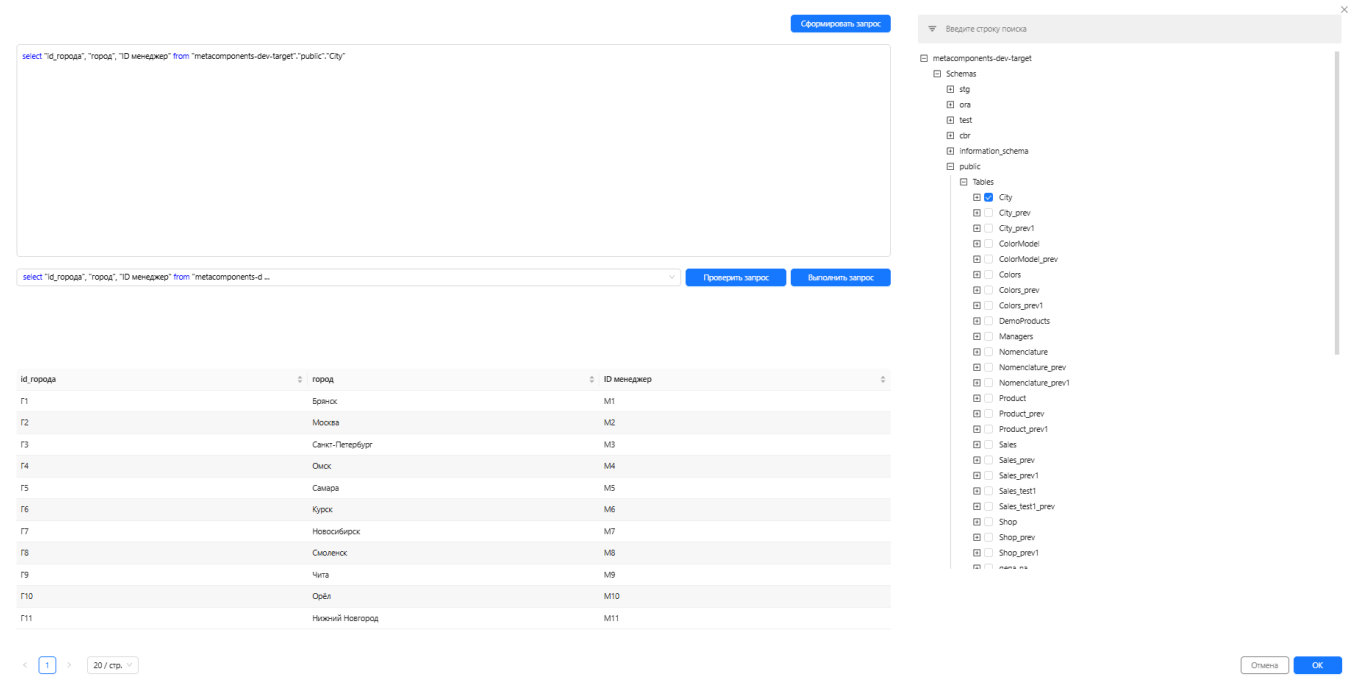


Рисунок. Пример запуска скрипта извлечённых данных из СУБД

После окончания настройки запроса необходимо нажать кнопку «ОК» в правом нижнем углу модального окна.

Запрос извлечения данных из веб-сервисов REST API

Особенности настройки

Для создания запроса извлечения данных из веб-сервиса по протоколам REST API (должен быть выбран соответствующий endpoint), необходимо в окне свойств справа нажать кнопку Create (Создать) (над полем Query (Запрос)). В строку Enter file address (Введите адрес файла) ввести адрес файла размещенного на веб-сервере, данные из которого необходимо загружать (под цифрой 1 на рисунке). Выбрать Method (метод загрузки) (под цифрой 2 на рисунке). Поддерживается 2 метода загрузки: POST, GET (метод определяется настройками веб-сервиса, поэтому перед созданием запроса следует изучить документацию источника данных). После чего нажать кнопку Form a query (Сформировать запрос), затем на кнопку Check request (Проверить запрос) (под цифрой 4 на рисунке). В поле под номером 5 на рисунке появится строка сформированного запроса. В этот момент выполняется проверка запроса и подстановка значений параметров, если они были добавлены. Далее нажать Run query (Выполнить запрос) (под цифрой 6 на рисунке), что приведет к загрузке данных в зону предварительного просмотра. Если данные не получены, то следует выполнить дополнительную настройку запроса, а именно указать заголовки запроса (headers). В заголовках определяется формат передаваемых данных, спецификация и версия протокола обмена и другая информация, необходимая для корректной обработки запроса (в соответствии с документацией веб-сервиса), так же можно указать тело запроса (body) – данные для обработки, в формате JSON.

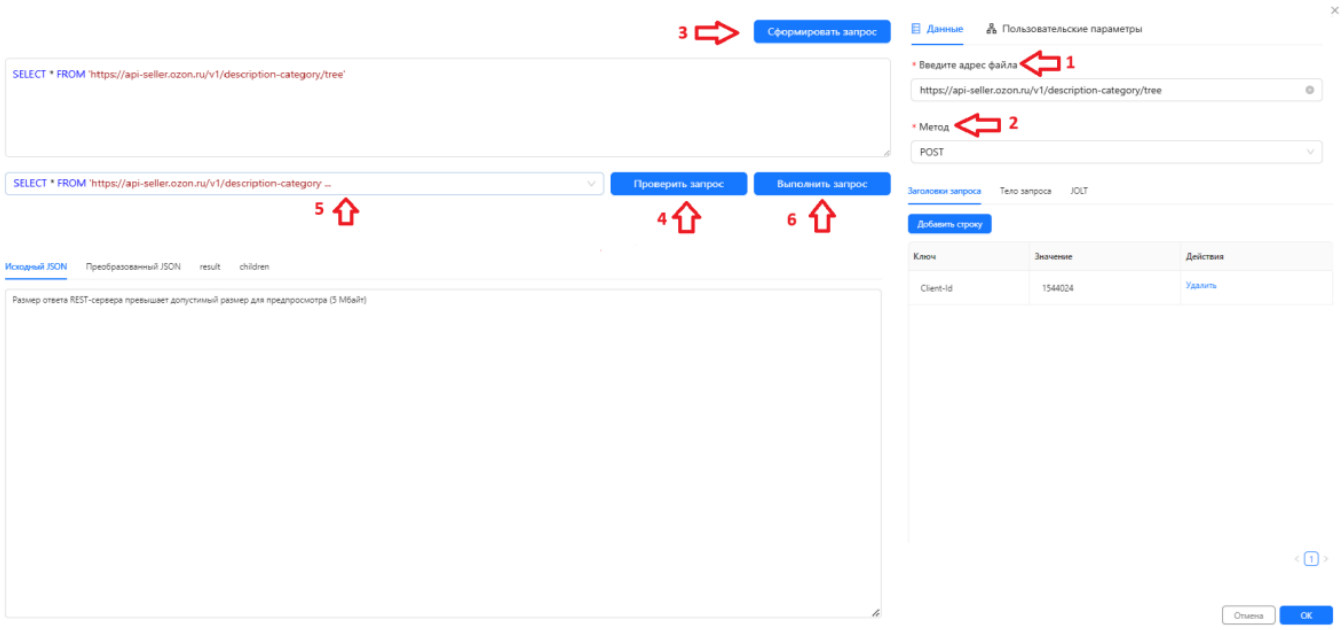


Рисунок. Диалоговое окно для формирования запроса Rest Api

Если и в этом случае система не возвращает ожидаемый результат, рекомендуется подготовить JOLT инструкции, оттестировать их в соответствующем сервисе и использовать в системе BI.Qube.

В зоне предварительного просмотра, отображаются структура исходного JSON (Original JSON), преобразованного инструкциями JOLT (Converted JSON) (если их нет, то показывается исходная структура) и разложенный JSON по таблицам. При этом, если размер файла JSON велик, то он не отображается в браузере.

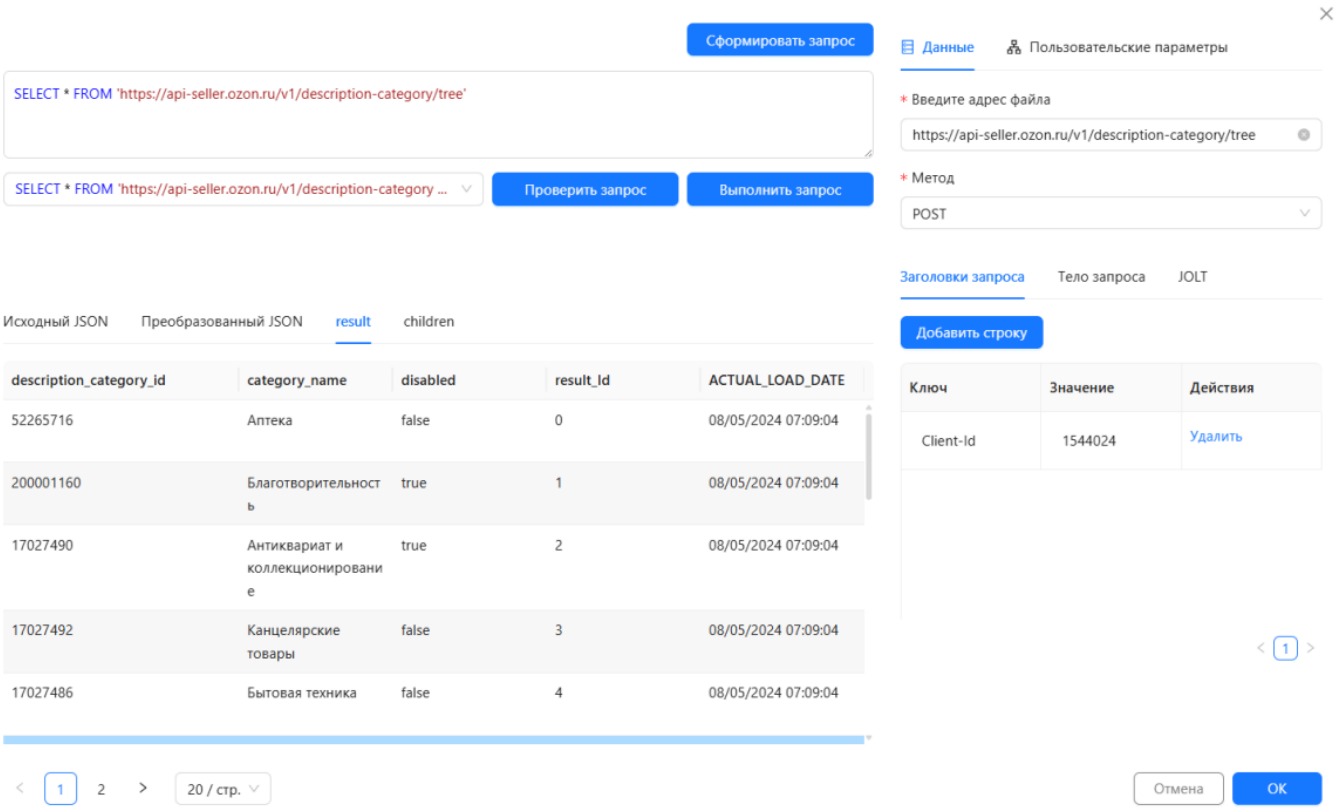


Рисунок. Пример преобразованного исходного JSON к табличному виду в виде: result и children.

Использование параметров в запросе

Помимо стандартной загрузки данных, можно использовать настройки загрузки данных с помощью параметров. Созданные ранее параметры доступны на вкладке User Parameters (Пользовательские параметры). Параметр может быть добавлен в текст запроса без нарушения синтаксиса. При выполнении команды сначала будет выполнен SQL-код параметра, в процессе проверки запроса в него будут подставлены вычисленные значения. Если параметр вернет более одного значения, то исходный запрос будет "размножен". В зоне предварительного просмотра можно посмотреть только результаты выполнения одного "размноженного" запроса. При выполнении регулярной загрузки с использованием оркестратора запрос будет выполнен столько раз, на сколько он был размножен вычисленным параметром.

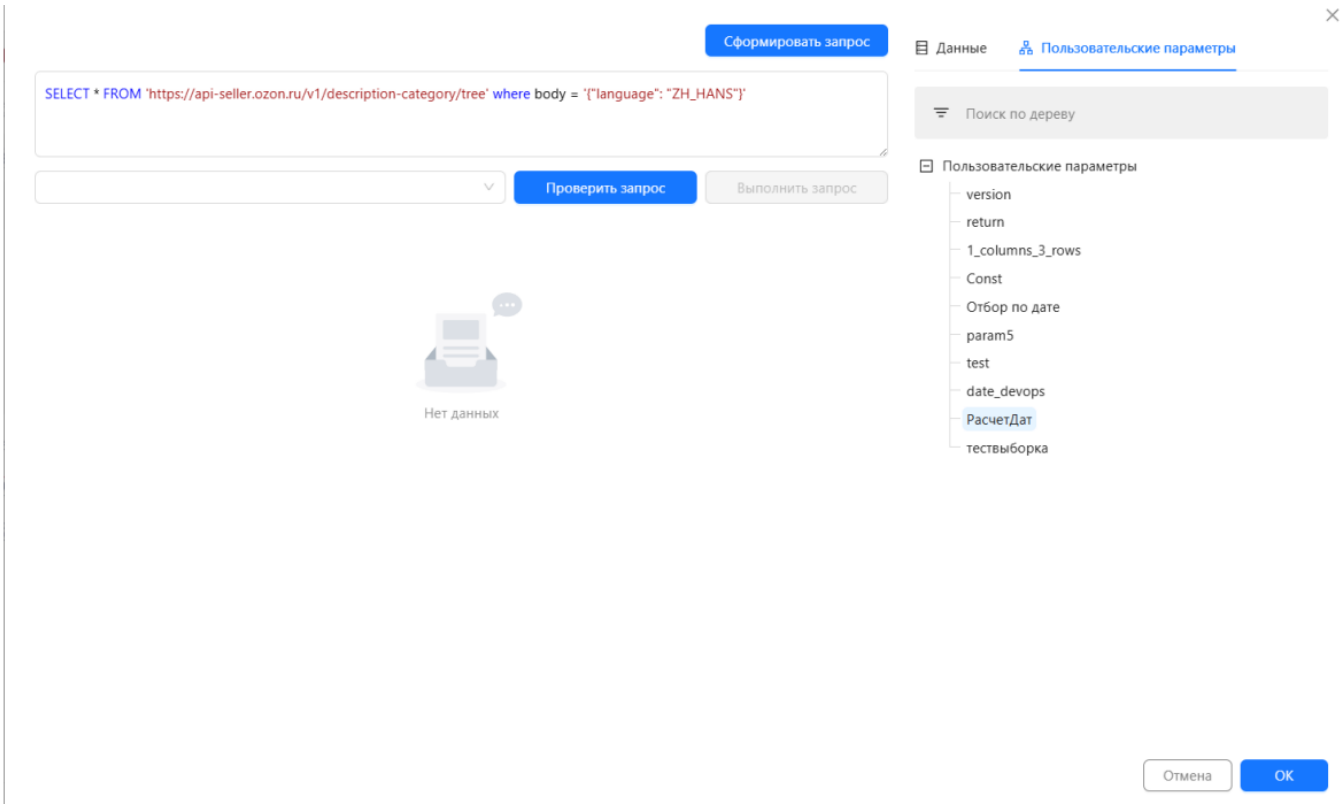


Рисунок. Пользовательские параметры

Пример настройки запроса

Настройка команды загрузки данных по протоколу Rest API с источником json

После создания и заполнения Endpoint (подключения) на странице General (Общее), мы можем во вкладке Staging, Commands (Команды), в окне свойств справа в поле Source (Источник) выбрать созданный Endpoint (Подключение) Rest API и нажать на кнопку Create (Создать).

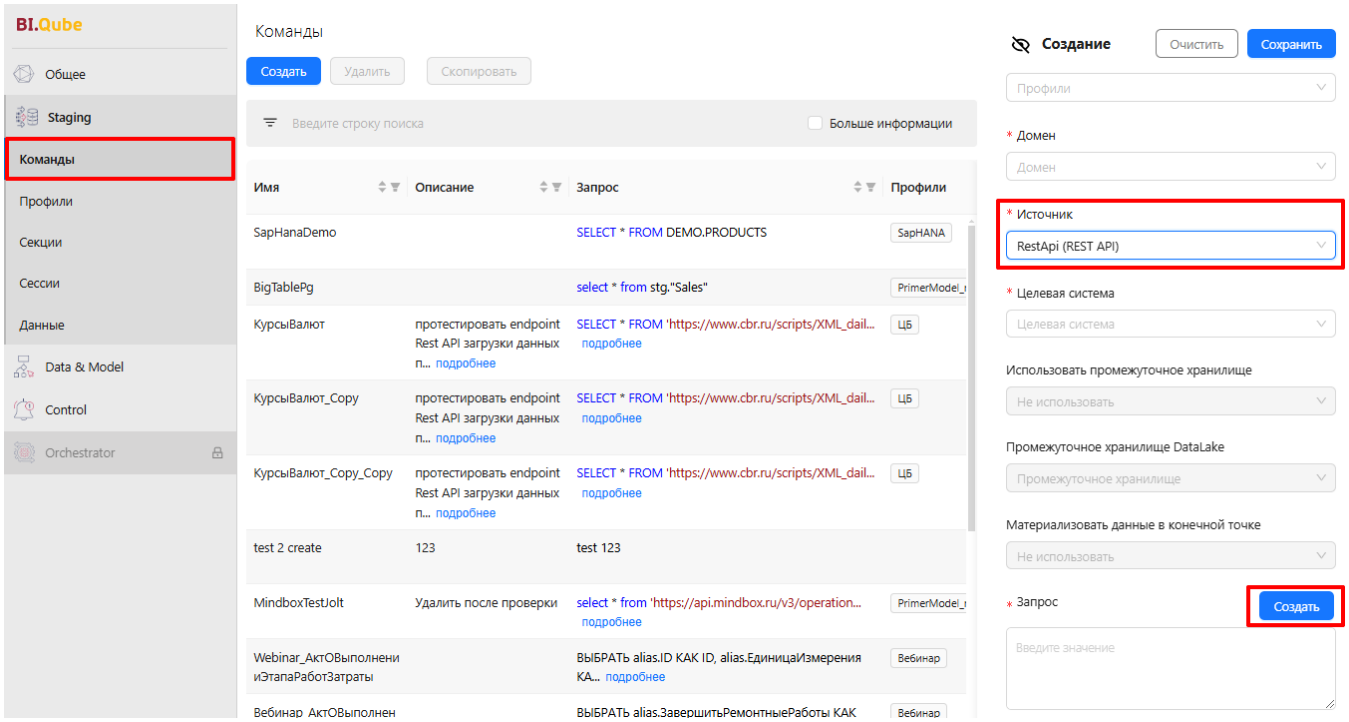


Рисунок. Выбор Endpoint (подключения) Rest API

В появившемся диалоговом окне в поле Enter file address (Введите адрес файла) ввести адрес файла. Выбрать Method (метод загрузки), указанный в документации источника данных.

После нажать на кнопку From a query (Сформировать запрос) – генерируется предварительно простой запрос на выборку данных. Запрос по умолчанию формируется на SQL.

При нажатии на кнопку Check request (Проверить запрос) выполняется проверка запроса с указанными параметрами. В случае удачной проверки появится строка/строки сформированного запроса.

Нажатие на кнопку Run Query (Выполнить запрос) соответственно выполняет сформированный запрос. Во вкладке Исходный JSON выводится содержимое файла json в начальном виде. Преобразованный JSON по умолчанию пустой.

Для дальнейшей работы с данными нажать кнопку ОК.

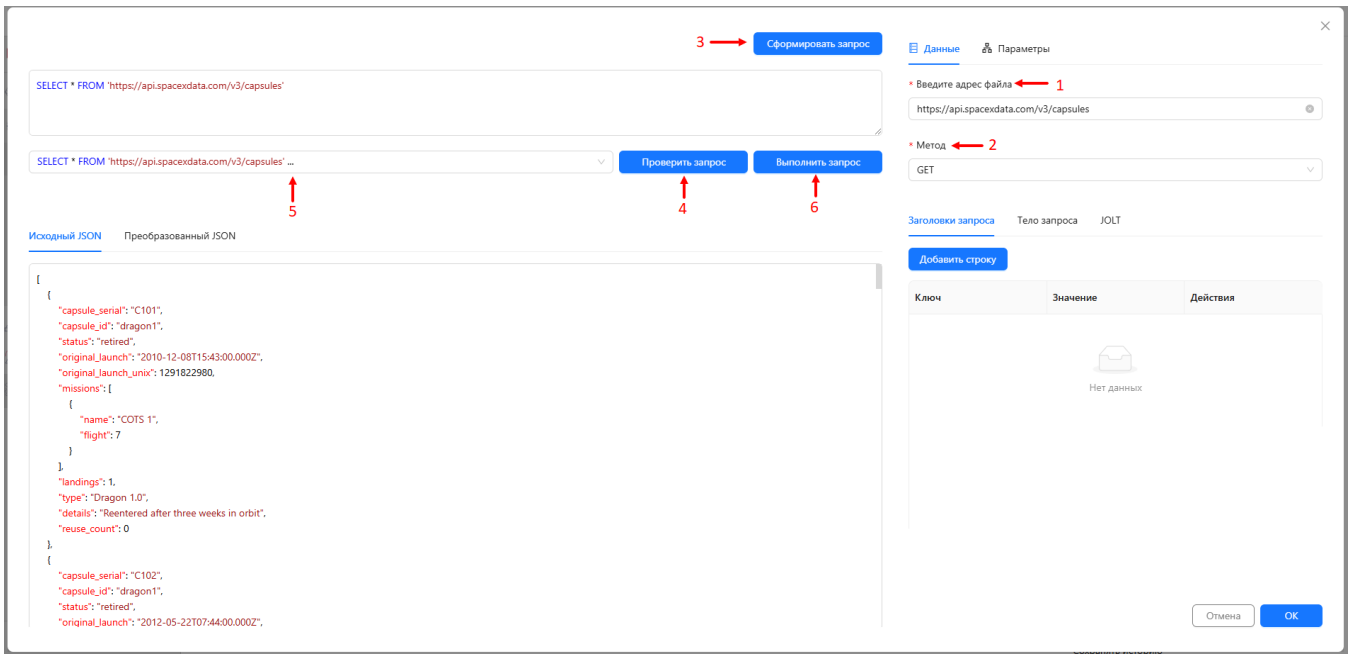


Рисунок. Диалоговое окно загрузки данных из Rest API

Преобразование JSON с помощью JOLT

В случае, если запрос не выполнен, система сообщит об ошибке. Запрос может быть не выполнен, если загружаемый JSON (файл) имеет сложную структуру. Для того, чтобы это исправить, в блоке справа необходимо выбрать вкладку JOLT. В поле для ввода необходимо ввести код JOLT, который преобразует исходный JSON (файл) к более простому виду. Далее нажать кнопку Check request (Проверить запрос), затем Run Query (Выполнить запрос) – исходный файл преобразуется в соответствии с полученными инструкциями JOLT к новому виду и выведется во вкладке Преобразованный JSON.
Если всё успешно, то появятся вкладки с таблицами, которые будут созданы на основании нового созданного JSON (файла). В случае, если новый созданный JSON (файл) алгоритмы системы не смогли обработать, то вкладки не появятся, но преобразованный с помощью JOLT исходный JSON (файл) выведется на экран, чтобы можно было ознакомиться с результатом преобразования. Это позволит подкорректировать текст кода JOLT и выполнить процедуру загрузки и преобразования данных повторно.

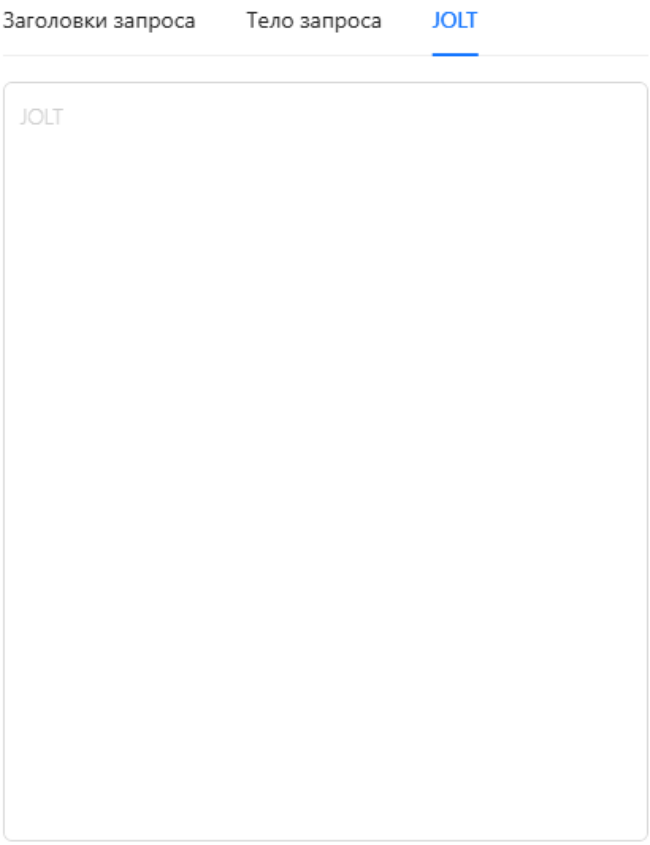


Рисунок. Вкладка JOLT

JOLT – это инструмент для выполнения задач, предназначенный для разработки программного обеспечения. Задачи определяются в скриптах Python. https://jolt.readthedocs.io/en/latest/ – данный сайт может помочь в написании JOLT.

Настройка команды загрузки данных по протоколу Rest API с источником xml/csv

Загрузка файлов по протоколу Rest API с источником xml/csv полностью повторяет уже описанный алгоритм действий для настройка команды загрузки данных по протоколу Rest API с источником json. За исключением того, что для источников xml/csv JOLT не используется.

Создание заголовка запроса

Для дополнительной настройки запроса необходимо указать заголовок запроса (Request headers), который содержит необходимую информацию для корректной обработки запроса.

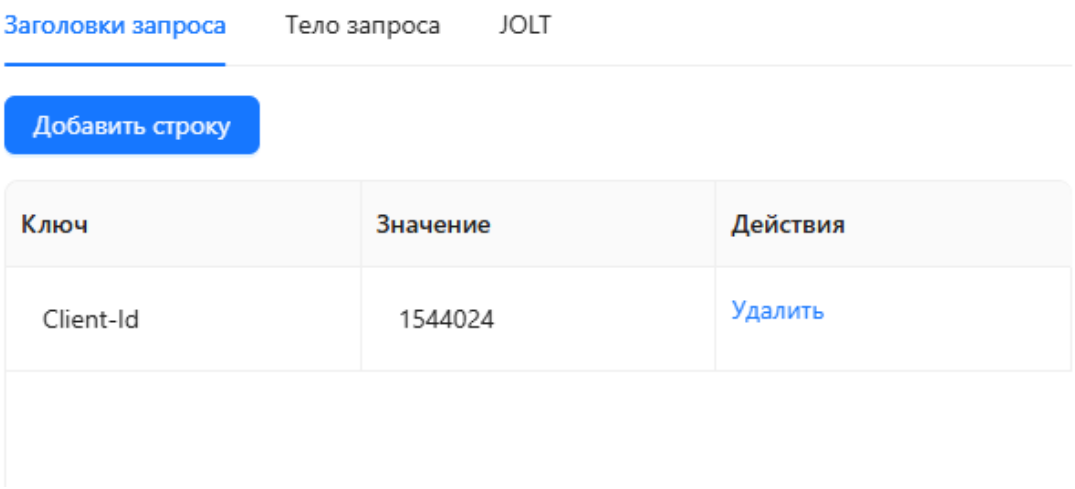


Рисунок. Вкладка Заголовки запроса

В системе, добавление заголовка представлено в виде таблицы. Чтобы добавить заголовок, необходимо нажать на кнопку "Добавить строку" (Add a row).

Создается строка в таблице, состоящая из трех столбцов:

1. Ключ (Key) –
2. Значение (Value) –
3. Действие (Actions) – содержи кнопку "Удалить" (Delete), для возможности удаления всей строки (заголовка). При клике на "Удалить" (Delete), открывается модальное окно с подтверждением удаления строки.

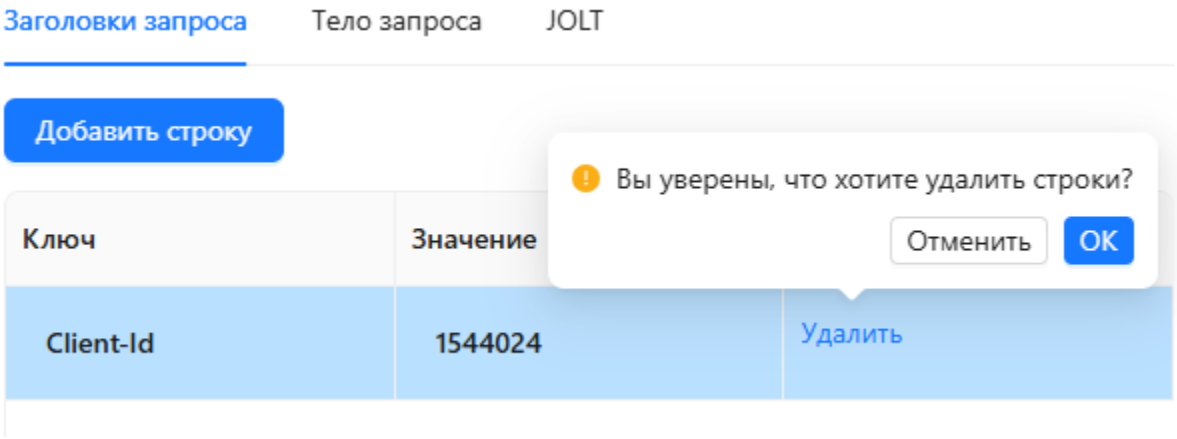


Рисунок. Удаление строки Заголовка запроса

Для добавления следующего заголовка необходимо снова нажать на кнопку "Добавить строку" (Add a row) и повторить все действия.

Для просмотра результирующего запроса и результатов его работы необходимо нажать на кнопку Form a query (Сформировать запрос) – программа сформирует простой SQL запрос. Затем необходимо нажать на кнопку Run query (Выполнить запрос), сформируется текст запроса и в зоне предварительного просмотра появятся результаты выполнения этого запроса.

Создание тела запроса

Чтобы создать Тело запроса (Request body), необходимо перейти во вкладку Тело запроса и заполнить поле ввода следующим образом: {тело запроса}.

Пример: {"language": "ZH_HANS"}

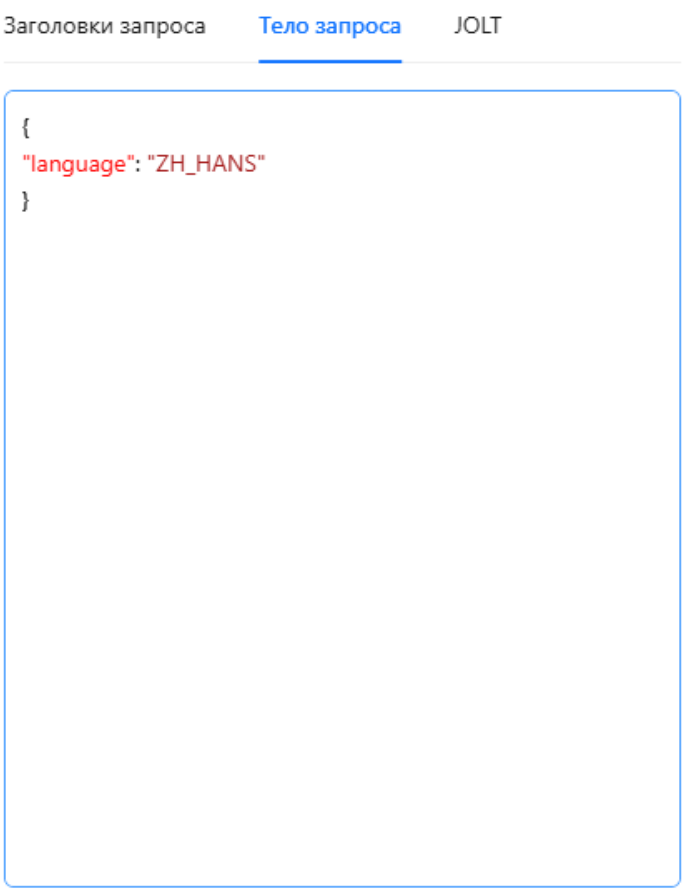


Рисунок. Вкладка Тело запроса

Для просмотра результирующего запроса и результатов его работы необходимо нажать на кнопку Form a query (Сформировать запрос) – программа сформирует простой SQL запрос, добавив условный оператор WHERE и указанное тело запроса (Пример: where body = '{"language": "ZH_HANS"}'). Затем необходимо нажать на кнопку Run query (Выполнить запрос), сформируется текст запроса и в зоне предварительного просмотра появятся результаты выполнения этого запроса.

Запрос извлечения данных из файловых хранилищ S3 и SMB

Загрузка данных из одного файла

После нажатия кнопки Create (Создать) появится диалоговое окно, в котором можно сформировать запрос на извлечение данных из файловых хранилищ типа S3 и SMB в диалоговом режиме или ввести запрос с клавиатуры на языке источника данных или, в отдельных случаях, на внутреннем языке системы.

Окно создания запроса разделено на две зоны. Слева зона отображения кода запроса к источнику, и под ним зона предварительного просмотра результата запроса и зона создания кода запроса. В правой зоне окна создания запроса отображается файловая структура выбранного хранилища и файлы, доступные в хранилище. Под деревом файловой структуры отображаются поля для заполнения, состав которых зависит от типа выбранного файла.

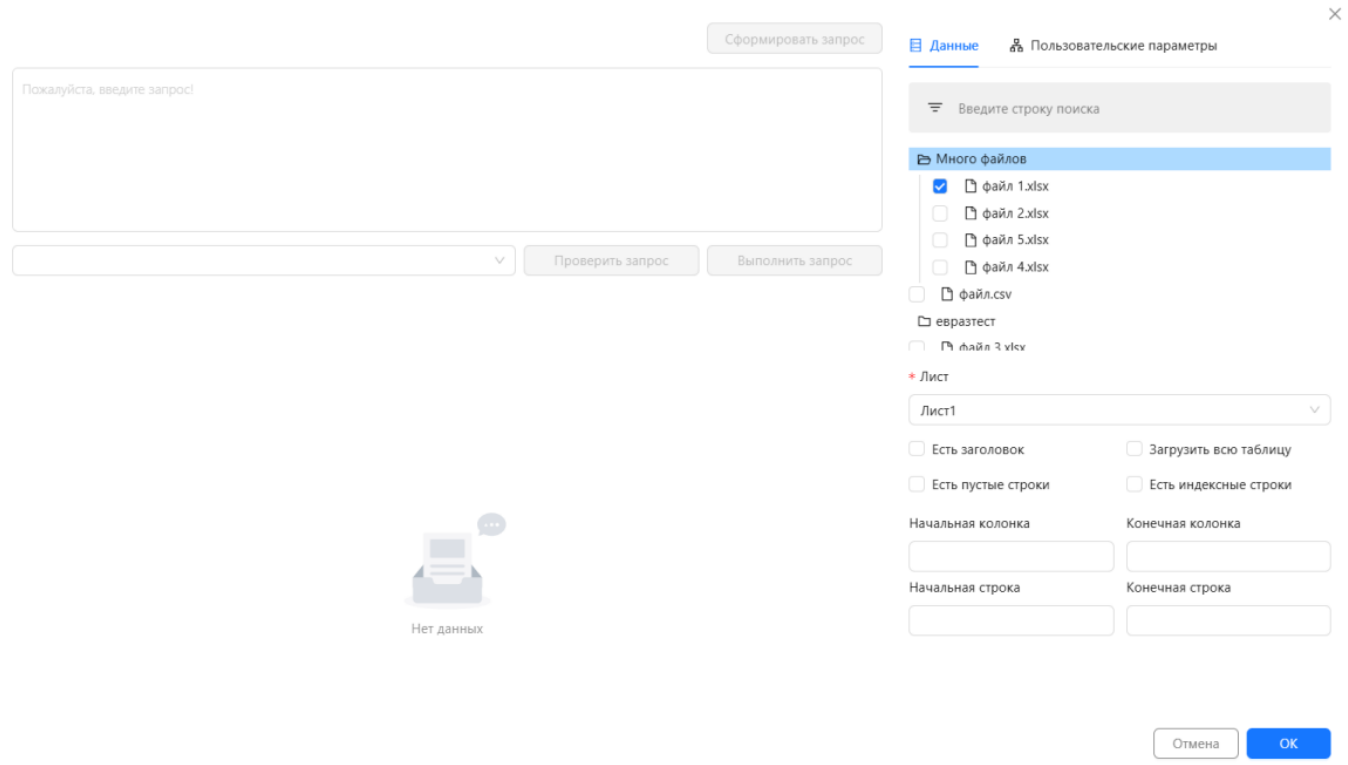


Рисунок. Диалоговое окно настройки запроса извлечения данных из фалов

Рисунок. Диалоговое окно запроса на извлечения данных из файловых хранилищ типа S3 и SMB

Если выбран файл типа xls, xlsx пользователю доступны следующие параметры для создания запроса на извлечения данных:

- В выпадающем списке Sheet (Лист) выбрать лист, данные из которого необходимо будет загрузить, после чего задать нужные опции;
- Have header (Есть заголовок) – опция, позволяющая использовать первую строку диапазона данных как строку заголовков таблицы;
- Load full table (Загрузить всю таблицу) – автоматическое определение диапазона данных;
- Have empty rows (Есть пустые строки) – позволяет из диапазона данных удалять пустые строки;
- Have index rows (Есть индексные строки) – добавляется колонка с номерами строк.

Если опция Load full table (Загрузить всю таблицу) не выбрана, то пользователь может ввести нужный диапазон данных вручную.

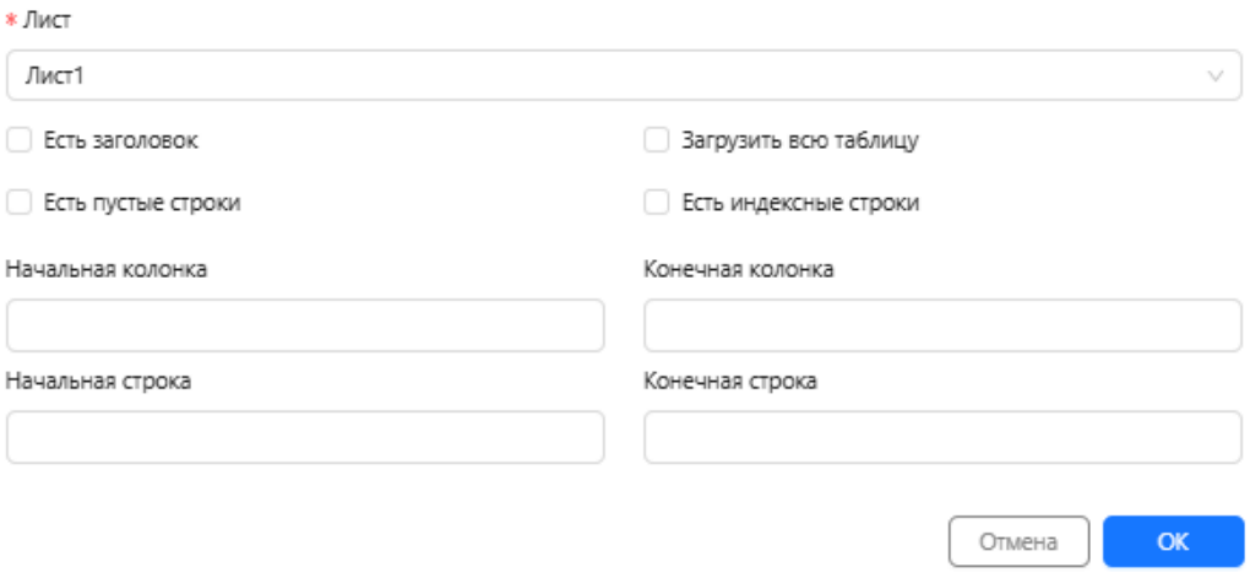


Рисунок. Настройка параметров запроса для извлечения данных из файлов xls, xlsx

Опции:

- есть заголовок – из выбранного диапазона первая строка становится заголовком (шапкой) таблицы назначения (и в предпросмотре), если ячейки первой строки диапазона пустые, то в качестве заголовков подставляем A, B, C...
- загрузить всю таблицу – автоматический выбор диапазона содержащего данные - максимальный охват. Если заполнена ячейка A1 и D10, то диапазон по колонкам от A до D, по строкам от 1 до 10
- есть пустые строки – удаляет пустые строки в выбранном диапазоне. Для предыдущего примера в предпросмотре будет строка 1 и строка 10. Все что между будет удалено из предпросмотра
- есть индексные строки – показываются номера строк из Excel, а не из диапазона. Для предыдущего примера для строки 10 индекс будет 10, хотя таблица в предпросмотре состоит из двух строк.

Если не включена опция "Загрузить всю таблицу", то можно явно задать интересующий диапазон:

- начальная колонка – число, начинается с единицы. Для примера выше A=1 – обязательно для заполнения
- конечная колонка – число, начинается с единицы. Для примера выше D=4 – обязательно для заполнения
- начальная строка – число, начинается с единицы, 1=1 – обязательно для заполнения
- конечная строка – число начинается с единицы 10=10, может быть не заполнено. В этом случае размер таблицы определять автоматически.

Если выбрал файл типа csv пользователю доступны следующие параметры для создания запроса на извлечения данных:

Разделитель – специальный символ, благодаря которому происходит разделение строки на колонки. Чаще всех разделителями являются:

- ";" – самый распространенный
- \t, если разделитель – таб
- ","
- "."

После выбора разделителя необходимо выбрать нужные опции: есть индексные строки; есть заголовок, если того требует запрос.

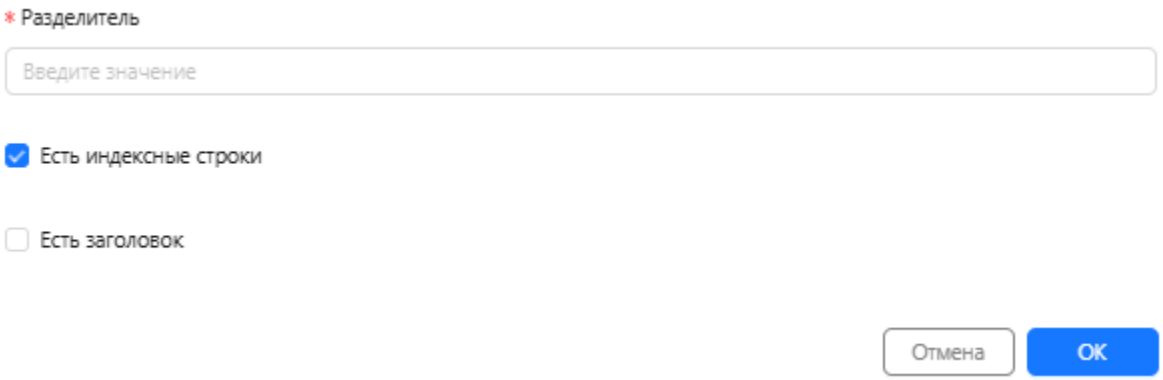


Рисунок. Настройка параметров запроса для извлечения данных из файлов csv

Для просмотра результирующего запроса и результатов его работы необходимо нажать на кнопку Form a query (Сформировать запрос) – программа сформирует запрос. При необходимости в запрос можно внести изменения, например, дописать условия отбора данных. После окончания работы с запросом, необходимо нажать на кнопку проверить, система выполнит валидацию запроса. Если в запросе используются параметры, то система "размножит" запрос в соответствии с используемыми параметрами. Для просмотра результата запроса необходимо в выпадающем списке выбрать нужный вариант запроса и нажать кнопку Выполнить. Результаты запроса появятся в таблице.

Загрузка нескольких файлов

Одной командой может быть настроена одновременная загрузка нескольких файлов, имеющих одинаковую структуру. Все файлы должны размещаться в одном каталоге и иметь единую маску имени. Например, необходимо загрузить одновременно три файла с именами: файл1, файл2, файл3. Для этого в коде запроса, сформированного для любого из этих файлов, вторую часть имени, в данном случае цифровую, заменить на символ "*" (звёздочка). В этом случае, система в процессе выполнения команды загрузит все файлы, попадающие под маску "файл*".

"*" обозначает любые символы.

> Символ "*" поддерживается только в имени файла!

Рассмотрим пример одновременной загрузки четырёх файлов, имя которых соответствует следующей маске "файл*.xlsx". Выбираем в дереве объектов первый файл, указываем опции: в выпадающем списке выбираем "лист1", включаем опцию "есть заголовки", "загрузить всю таблицу". В сформированном запросе в имени файла меняем номер файла на символ "*". Нажимаем кнопку "Проверить запрос". Проверяем в выпадающем списке, что запрос "размножился" на нужное количество файлов, выбираем любой вариант сформированного запроса и нажимаем кнопку "Выполнить". В зоне предварительного просмотра отобразятся данные выбранного запроса.

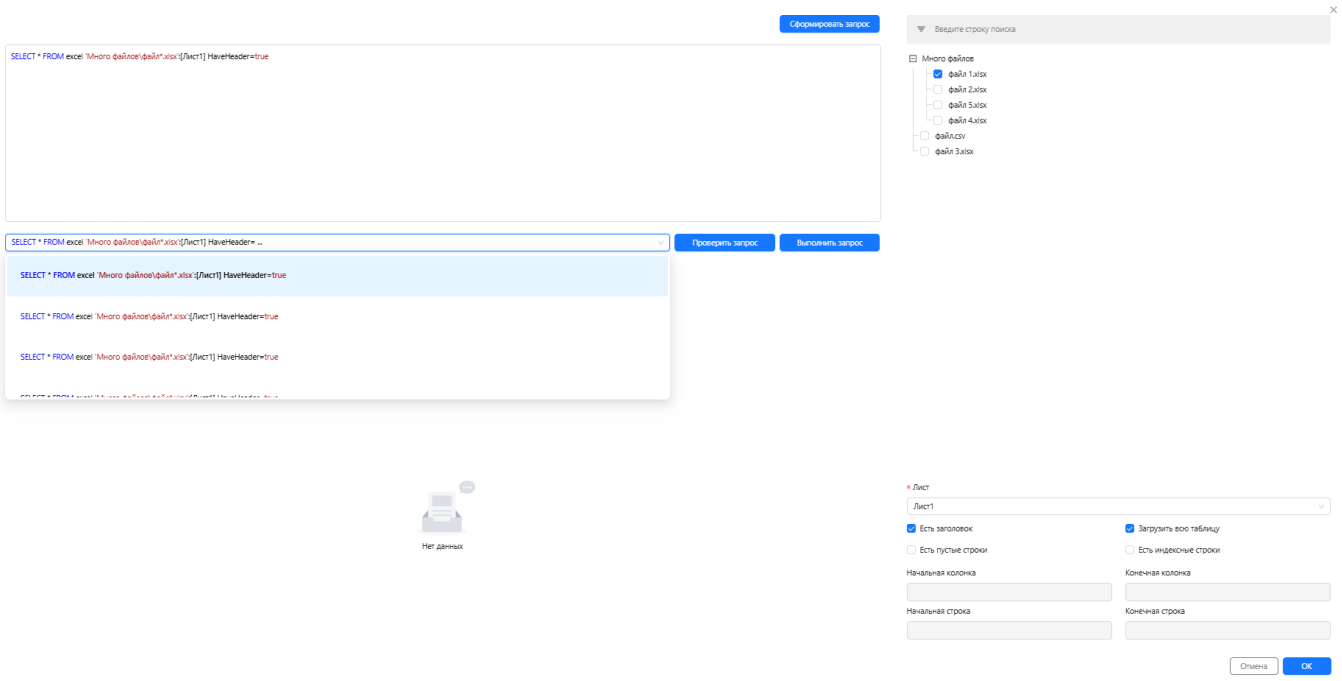


Рисунок. Выбор варианта "размноженного" запроса

Данный режим работы запроса имеет наименование "инкрементальная по файлам". В таком режиме каждый новый запуск на выполнение команды будет загружать в хранилище данные только из ранее незагруженных файлов, имя которых соответствует маске в запросе. При каждой смене Типа загрузки данных в настройках свойств команды, например, на Тип загрузки "Полная", будет приводить к полной перезагрузке данных из всех файлов, имя которых соответствует маске.

В данном режиме есть ограничение – за один запуск выполнения команды может быть загружено не более 500 файлов. В случае, если требуется загрузить больше, следует запустить команду без изменения ее настроек повторно, и так до загрузки всех файлов. Каждый новый запуск проверяет каталог с файлами, сверяет по именам файлов с загруженными ранее и догружает новые в хранилище.

Запрос извлечения данных из брокера сообщений Apache Kafka

Терминология Apache Kafkaa

Брокер – система, преобразующая сообщение от источника данных (продюсера) в сообщение принимающей стороны (консьюмера). Брокер выступает проводником и состоит из серверов, объединенных в кластеры.

Apache Kafka – масштабируемый кластер со множеством взаимозаменяемых серверов, в которые добавляются новые брокеры, распределяющие задачи между собой. Сообщения хранятся на узлах-брокерах.

Для извлечения данных из данного брокера пользователю необходимо знать имя "топика", в котором появляются интересующие сообщения.

Topic – принцип деления потока данных, базовая и основная сущность Apache Kafka. В топик складывается стрим данных, единая очередь из входящих сообщений.

Partition – для ускорения чтения и записи топики делятся на партиции. Происходит параллелизация данных. Это конфигурируемый параметр, сообщения могут отправлять несколько продюсеров и принимать несколько консьюмеров.

Потребитель Apache Kafka – это клиентское приложение (в данном случае BI.Qube), которое подписывается на весь топик или его отдельный раздел, чтобы считывать события, публикуемые туда приложением-продюсером. Потребление сообщений реализуется в цикле опроса, когда потребитель отправляет брокерам Kafka запросы на выборку к лидерам разделов с данными. Смещение потребителя указывается в логе при каждом запросе и сообщается потребителю, который контролирует эту позицию и может изменить ее для повторного считывания данных.

Стратегия управления смещением в потребителе Kafka определяется двумя конфигурациями:

- *auto.commit* – автоматическая фиксация, по умолчанию *true*;
- *offset.reset* – политика сброса смещения.

По умолчанию, когда потребитель читает сообщения из Kafka, он периодически фиксирует свое текущее смещение для разделов, из которых он читает, обратно в Kafka. Потребитель автоматически фиксирует смещения периодически с интервалом, заданным в конфигурации *auto.commit.interval.ms* (по умолчанию 5 секунд).

Политика сброса смещения *auto.offset.reset* определяет поведение потребителя, когда нет зафиксированной позиции, например, при первой инициализации группы потребителей или когда смещение выходит за пределы диапазона. Kafka поддерживает три политики смещения, задаваемые в значении конфигурации потребителя *auto.offset.reset* :

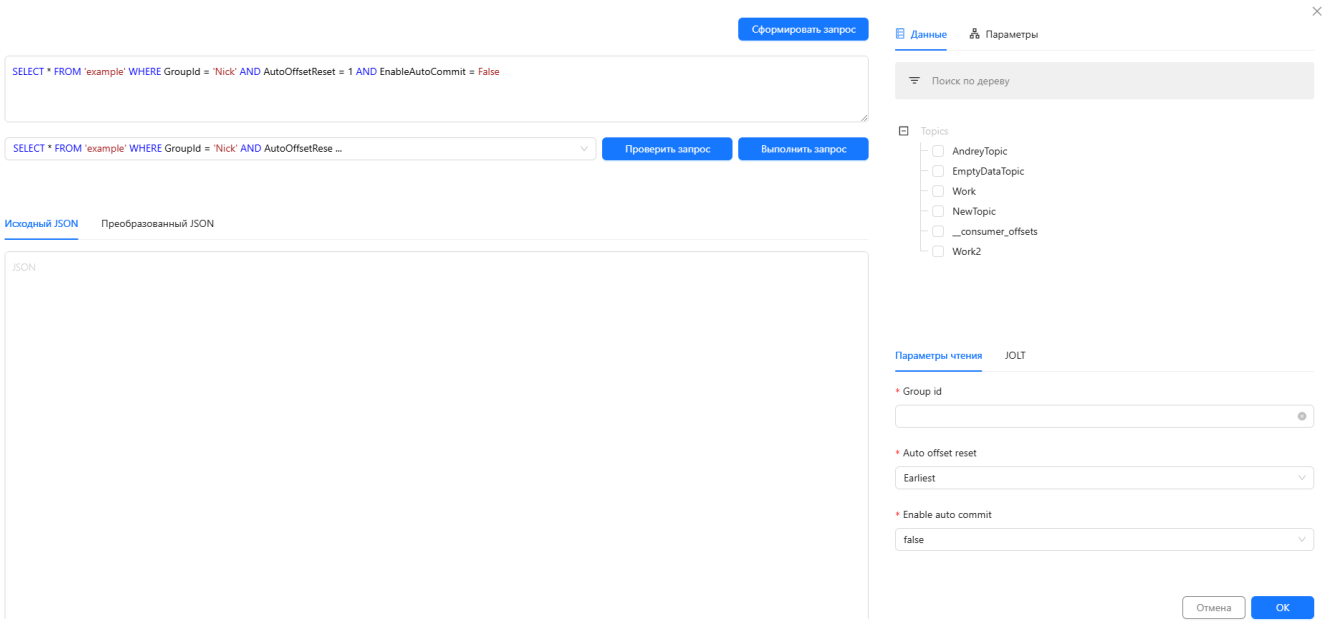
- *самое раннее (earliest),* когда приложению-потребителю необходимо получить все имеющиеся в топике сообщения с самого начала;
- *последнее (latest),* когда приложению-потребителю не нужно получать все сообщения с самого начала, а достаточно считать только данные, поступившие в топик после того момента, как потребитель на него подписался. Или же из последнего зафиксированного смещения, когда потребитель повторно присоединился к кластеру Kafka, например, после восстановления после сбоя.
- *не задано (none),* когда надо устанавливать начальное смещение самостоятельно и обрабатывать ошибки выхода за пределы диапазона вручную.

Потребительская группа – это объединение потребителей для многопоточного (многопользовательского) использования топиков Kafka. Потребительские группы в Kafka имеют следующие особенности:

- id – номер группы, который присваивается ей при создании для возможности подключения потребителей, использующих в качестве параметра соединения этот идентификатор (id). Следовательно, для параллельного использования группы, потребители используют один и тот же group.id;
- брокер Kafka назначает разделы топика потребителю в группе таким образом, что каждый раздел потребляется ровно одним потребителем в группе;
- потребители видят сообщение в том порядке, в котором они были сохранены в журнале, независимо от того, в какой момент времени они подключились к группе;
- максимальный параллелизм группы достигается лишь тогда, когда в топике нет разделов.

Настройка запроса

Создание запроса на извлечение данных из топика Kafka выполняется в соответствующем окне. В первую очередь необходимо выбрать топик, из которого будет осуществляться чтение данных. Список топиков автоматически подгружается в соответствии с выбранным подключением.



После выбора топика система попытается подгрузить имеющиеся потребительские группы и, если они есть, то можно выбрать любую доступную, если нет, то при формировании запроса группа бет сгенерирована автоматически. Затем необходимо выбрать требуемое значение для параметра *auto_offset_reset,* напри мер, значение latest, т.е. чтение с последнего доступного смещения. А *enable_auto_commit,* установленное в True, означает автоматическую фиксацию смещения, это позволит избежать получение дублей в точке назначения.

После установки параметров следует нажать кнопку "Сформировать запрос". При необходимости в сформированный запрос могут быть внесены корректировки. А затем нажать "Выполнить запрос". Система автоматически обработает массив данных представленный в формате JSON и выведет его текст без каких либо преобразований и автоматически предпримет попытку распарсить содержимое и разложить данные по отдельным таблицам. Если результат не получен или не соответствует ожиданию, необходимо разработать JOLT инструкции, с помощью которых система сможет выполнить парсинг исходного JSON.

⚠ При выполнении запроса в диалоговом окне создания параметр enable auto commut всегда устанавливается в значение false

ИСПОЛЬЗОВАНИЕ ПАРАМЕТРОВ В КОМАНДАХ

В зависимости от версии системы использование параметров может отличаться.

> Параметры, созданные в визуальном интерфейсе в ранних версиях Metastaging могут быть включены в запрос, но область их действия распространяется только для режима предварительного просмотра и не будет доступна при запуске на выполнение команды, как при запуске из веб-интерфейса, так и при запуске с использованием внешнего приложения. Для возможности применения параметров в запросах MetaStaging их следует создавать непосредственно в СУБД, в таблице. Параметры созданные в этой таблице будут вычисляться до выполнения команды metastaging и рассчитанные значения будут подставляться в запрос команды.

Для организации более гибкого процесса извлечения данных из источников в запросах команд можно использовать параметры, которые могут ограничивать объемы загружаемых данных или управляют процессом загрузки. Синтаксис вставки параметра в запрос выглядит следующим образом /*{ИмяПараметра}*/

Пример.

```
Select * from Table
where id>/*{number}*/
```

При этом следует помнить, что синтаксис SQL-запроса должен соответствовать той СУБД, на которой планируется волнение команды Metastaging

ТИПЫ ЗАГРУЗКИ

Внимание. В данный момент готовится новая версия управления типами загрузки

Виды загрузок

При работе с загрузкой данных часто возникают разные сценарии загрузки данных. В компоненте MetaStaging поддержаны разные сценарии и методы загрузки:

- Инкрементальная (по идентификатору)
- Инкрементальная загрузка
- Инкрементальная по 2 значениям
- Инкрементальная по единственному значению
- Инкрементальная по файлам
- Перегрузка таблицы
- Полная загрузка
- Полная загрузка с сохранением истории

Секционированная загрузка

При загрузке больших объемов данных, которые могут быть физически разделены по определённым правилам, для некоторых типов точек назначения доступна загрузка с разделением на секции.

Полная загрузка

Полная загрузка – это загрузка данных без параметризации.

- Применяется, когда необходима полная перезагрузка всех данных в таблице на источнике (например, при отсутствии столбца, подходящего для секционирования).
- Может использоваться с источниками:....(не может использоваться с такими источниками как:....
- Сопоставление с типом endpoint

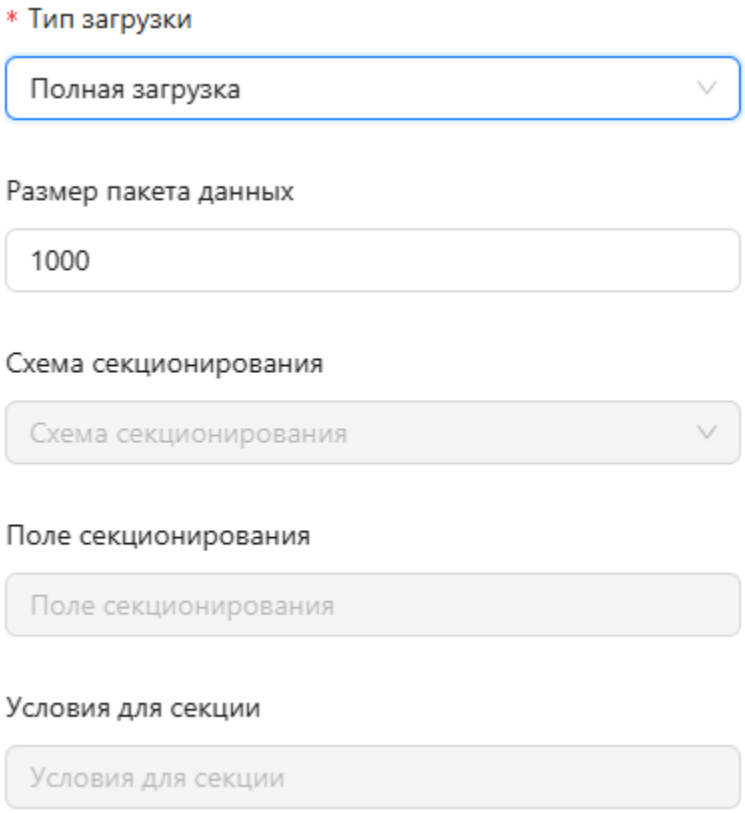


Рисунок. Выбор полной загрузки

Полная загрузка с сохранением истории

Полная загрузка с сохранением истории – это загрузка данных без параметризации.

- Применяется для перенацеливания представлений на новые Parquet-файлы, при этом старые не удаляются из хранилища.
- Может использоваться с источниками:...(не может использоваться с такими источниками как:....
- Сопоставление с типом endpoint:

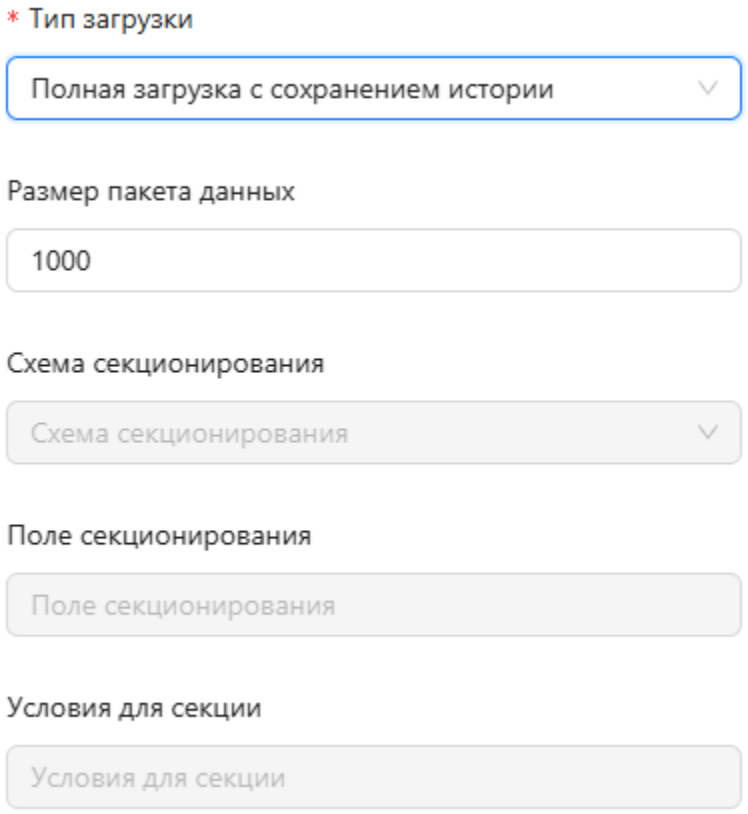


Рисунок. Выбор полной загрузки с сохранение истории

Инкрементальная загрузка

Инкрементальная загрузка (загрузка с параметрами) – это регулярная загрузка данных.

- Применяется для извлечения всех актуальных данных с даты последней загрузки.
- Может использоваться с источниками:...(не может использоваться с такими источниками как:....
- Сопоставление с типом endpoint

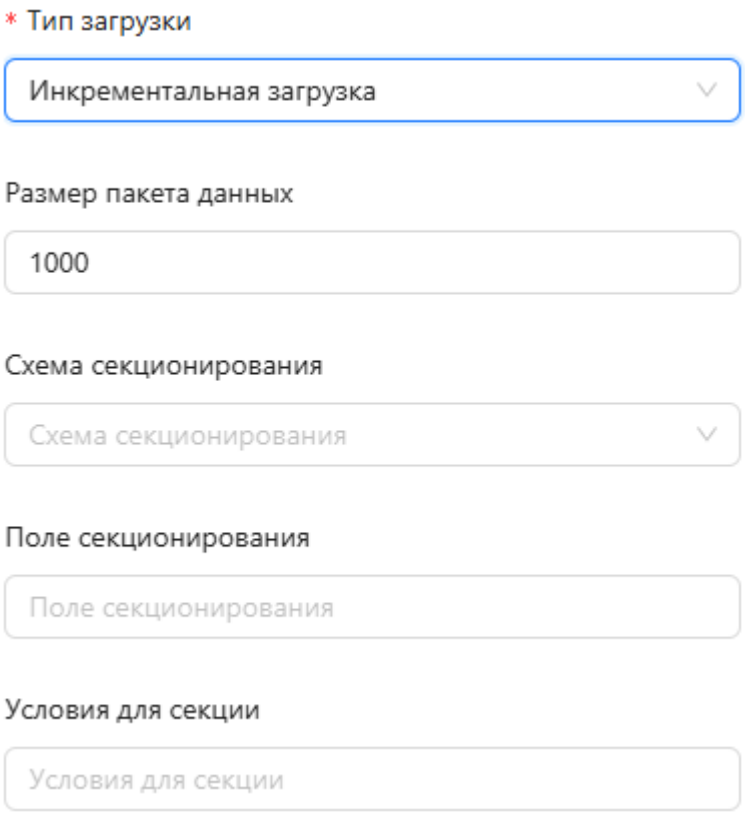


Рисунок. Выбор инкрементальной загрузки

Запрос для инкрементальной загрузки (или загрузки с параметрами) данных отличается от полной и требует настройки дополнительных параметров:

- Partition schema (Схема секционирования) – задается исходя из назначения секции и зависит от типа секционирования. Выбирается из выпадающего списка. Создание секции производится на странице Partition (Создание секций);
- Partition column (Поле секционирования) – поле, по которому осуществляется секционирование;
- Partition column convert (Условия секционирования) – условия, характерные для выбранной секции.

В поле Partition schema (схема секционирования) существует два выбора: ora_part и test.

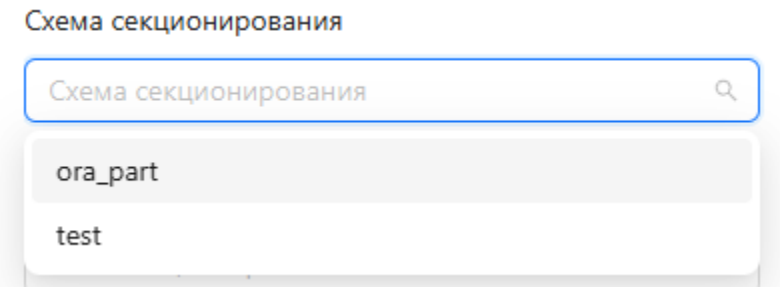


Рисунок. Выпадающий список в поле Partition schema (схема секционирования)

Инкрементальная загрузка (по идентификатору)

Инкрементальная загрузка (по идентификатору) – это загрузка данных, при которой извлекаются только новые и изменённые данные.

- Применяется, когда необходимо загрузить часть определённых данных. Поиск необходимых данных для этого типа загрузки будет осуществляться с помощью идентификатора (уникальный признак объекта, позволяющий отличать его от других объектов, то есть идентифицировать).
- Может использоваться с источниками:....(не может использоваться с такими источниками как:....
- Сопоставление с типом endpoint

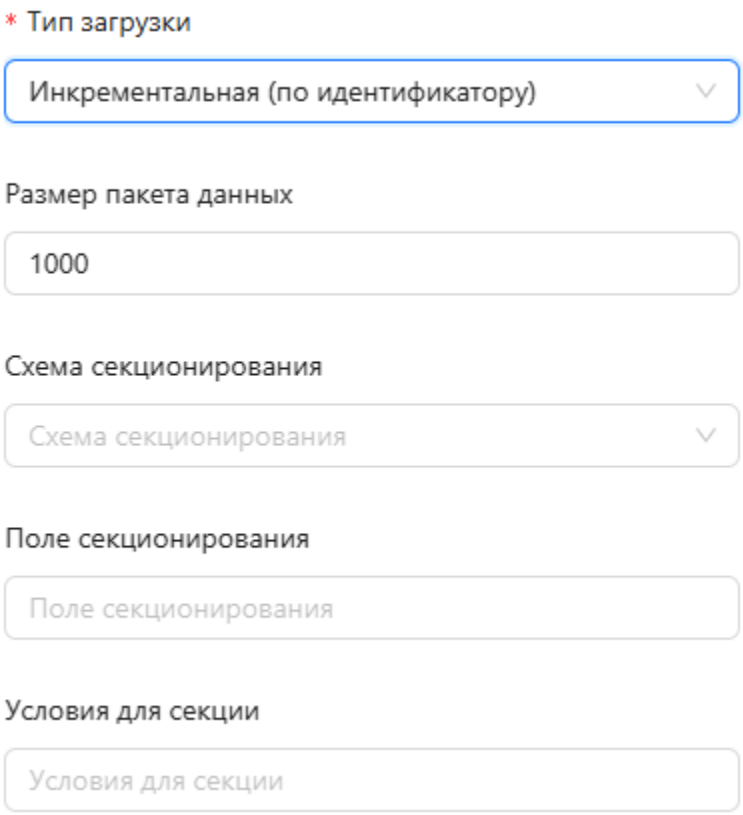


Рисунок. Выбор инкрементальной загрузки (по идентификатору)

Запрос для инкрементальной загрузки (или загрузки с параметрами) данных отличается от полной и требует настройки дополнительных параметров:

- Partition schema (Схема секционирования) – задается исходя из назначения секции и зависит от типа секционирования. Выбирается из выпадающего списка. Создание секции производится на странице Partition (Создание секций);
- Partition column (Поле секционирования) – поле, по которому осуществляется секционирование;
- Partition column convert (Условия секционирования) – условия, характерные для выбранной секции.

В поле Partition schema (схема секционирования) существует два выбора: ora_part и test.

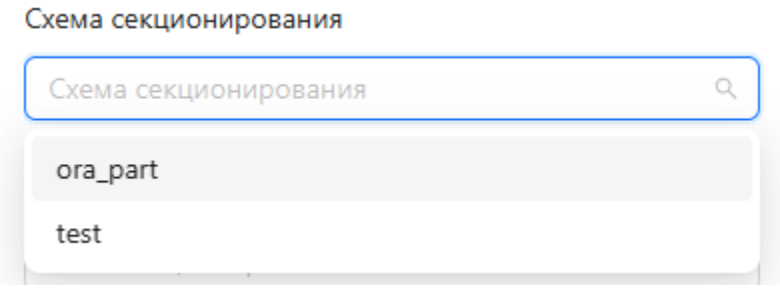


Рисунок. Выпадающий список в поле Partition schema (схема секционирования)

Инкрементальная загрузка по двум значениям: глубина вниз и вверх

Инкрементальная загрузка по двум значениям: глубина вниз и вверх - это тип загрузки данных на основе существующих таблиц.

- Для создания новой базы данных используется подход "сверху вниз", а для старой "снизу вверх". Все объекты моделируются на логическом уровне, затем применяются к дизайну физической базы данных, соответственно загрузка данных также должна подчиняться этим принципам.
- Может использоваться с источниками:....(не может использоваться с такими источниками как:....
- Сопоставление с типом endpoint

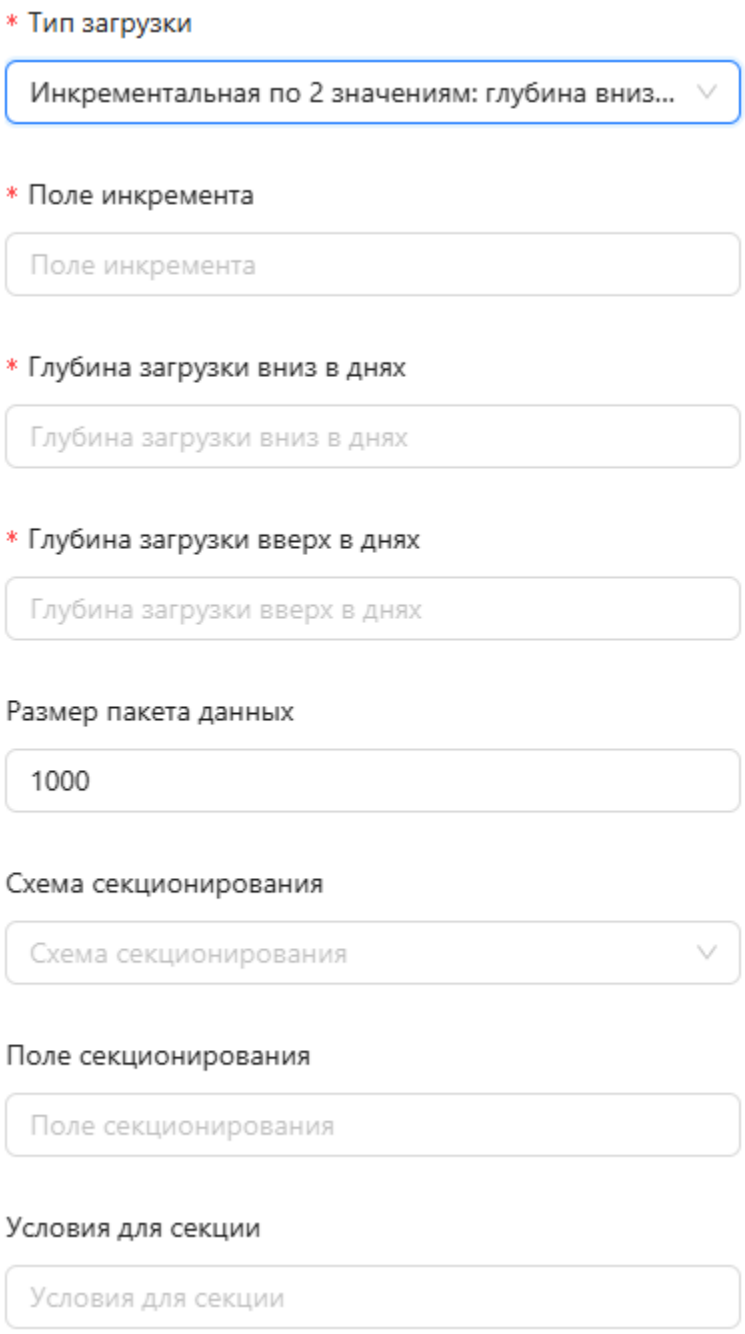


Рисунок. Выбор инкрементальной загрузки по двум значениям: глубина вниз и вверх

Запрос для инкрементальной загрузки по двум значениям: глубина вниз и вверх отличается настройками дополнительных параметров:

- Increment field (Поле инкремента) –
- Depth of down load in days (Глубина загрузки вниз в днях) –
- Depth of upward loading in days (Глубина загрузки вверх в днях) –

- Partition schema (Схема секционирования) – задается исходя из назначения секции и зависит от типа секционирования. Выбирается из выпадающего списка. Создание секции производится на странице Partition (Создание секций);
- Partition column (Поле секционирования) – поле, по которому осуществляется секционирование;
- Partition column convert (Условия секционирования) – условия, характерные для выбранной секции.

В поле Partition schema (схема секционирования) существует два выбора: ora_part и test.

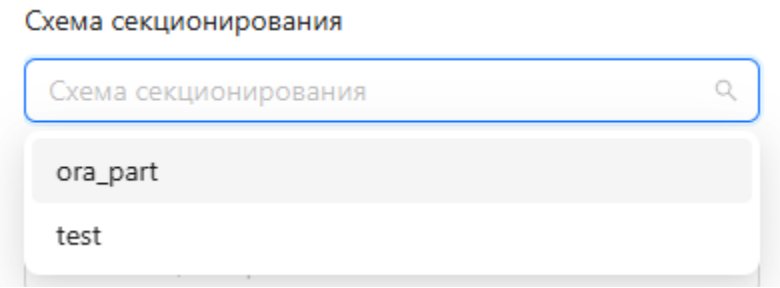


Рисунок. Выпадающий список в поле Partition schema (схема секционирования)

Инкрементальная загрузка по единственному значению

Инкрементальная загрузка по единственному значению – это загрузка данных, при которой извлекаются только новые и изменённые данные.

- Применяется для загрузки определённых данных, поиск которых будет определяться уникальным (единственным значением).
- Может использоваться с источниками:....(не может использоваться с такими источниками как:....
- Сопоставление с типом endpoint

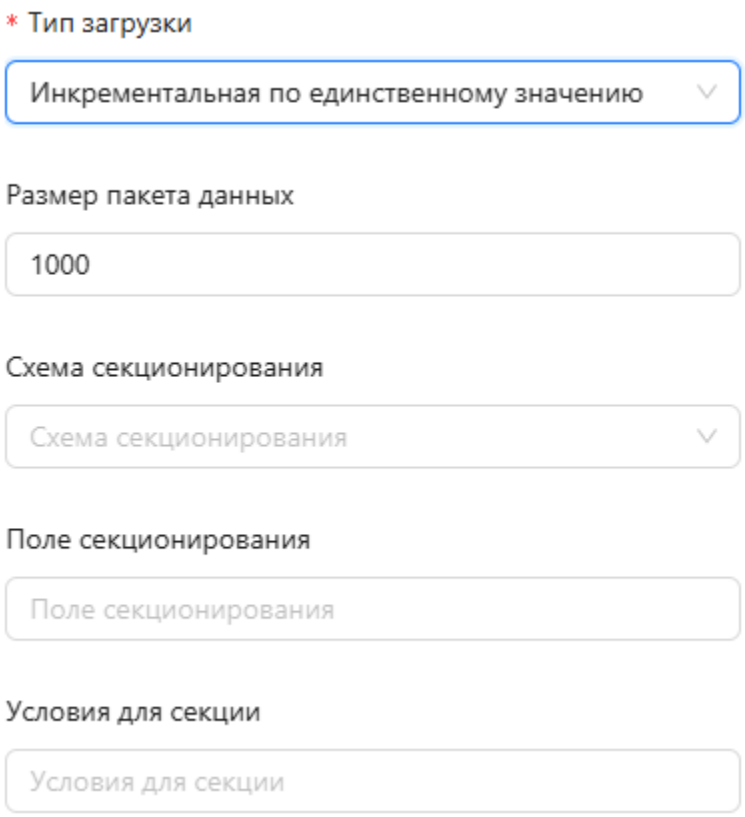


Рисунок. Выбор инкрементальной загрузки по единственному значению

Запрос для инкрементальной загрузки (или загрузки с параметрами) данных отличается от полной и требует настройки дополнительных параметров:

- Partition schema (Схема секционирования) задается исходя из назначения секции и зависит от типа секционирования. Выбирается из выпадающего списка. Создание секции производится на странице Partition (Создание секций);
- Partition column (Поле секционирования) - поле, по которому осуществляется секционирование;
- Partition column convert (Условия секционирования) - условия, характерные для выбранной секции.

В поле Partition schema (схема секционирования) существует два выбора: ora_part и test.

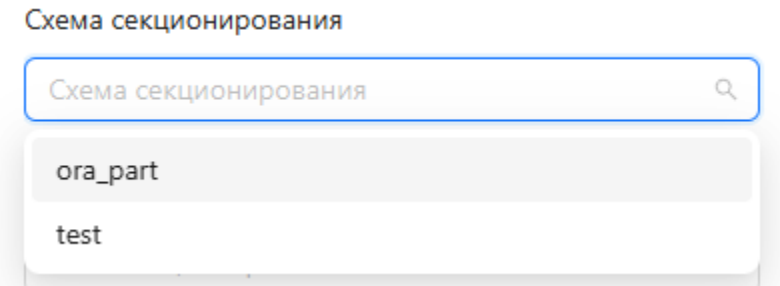


Рисунок. Выпадающий список в поле Partition schema (схема секционирования)

Инкрементальная загрузка по файлам

Инкрементальная загрузка по файлам – это загрузка данных, при которой из внешнего источника загружаются только обновлённые данные, а основная масса неизменившихся данных загружается из внутреннего хранилища.

- Применяется для загрузки определённых данных, поиск которых будет определяться выбором нужного файла или файлов для загрузки.
- Может использоваться с источниками:....(не может использоваться с такими источниками как:....
- Сопоставление с типом endpoint

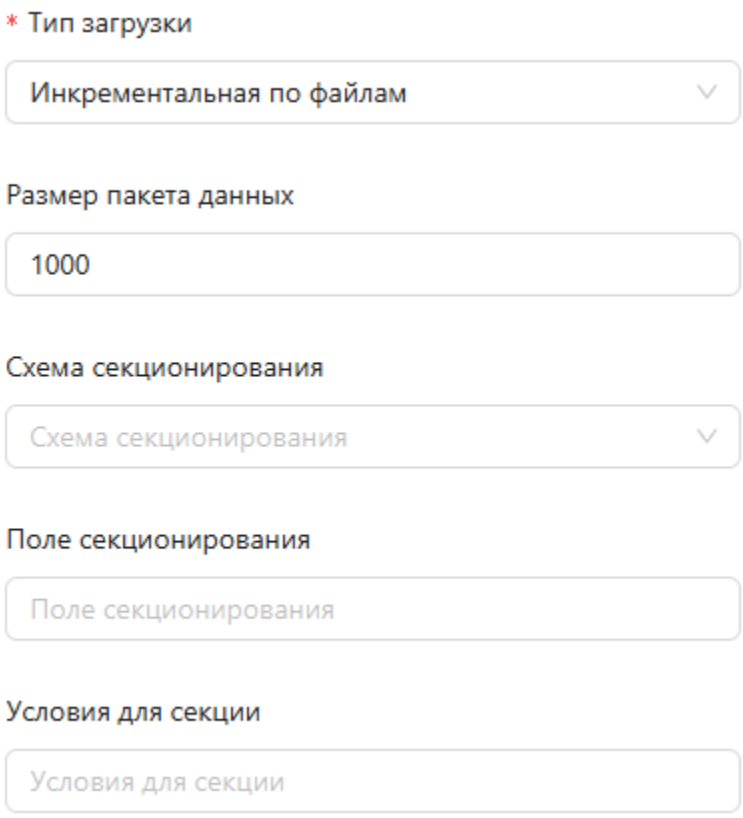


Рисунок. Выбор инкрементальной загрузки по файлам

Запрос для инкрементальной загрузки (или загрузки с параметрами) данных отличается от полной и требует настройки дополнительных параметров:

- Partition schema (Схема секционирования) задается исходя из назначения секции и зависит от типа секционирования. Выбирается из выпадающего списка. Создание секции производится на странице Partition (Создание секций);
- Partition column (Поле секционирования) - поле, по которому осуществляется секционирование;
- Partition column convert (Условия секционирования) - условия, характерные для выбранной секции.

В поле Partition schema (схема секционирования) существует два выбора: ora_part и test.

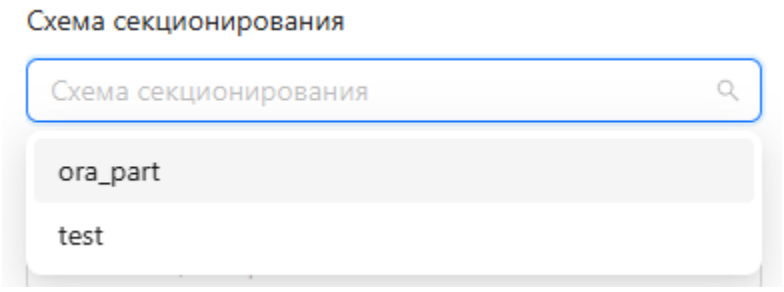


Рисунок. Выпадающий список в поле Partition schema (схема секционирования)

Перегрузка таблицы

- Данный тип загрузки применяется при изменении набора полей в источнике (таблице).
- Может использоваться с источниками:....(не может использоваться с такими источниками как:....
- Сопоставление с типом endpoint

Осуществляется выбор необходимого файла или файлов для загрузки.

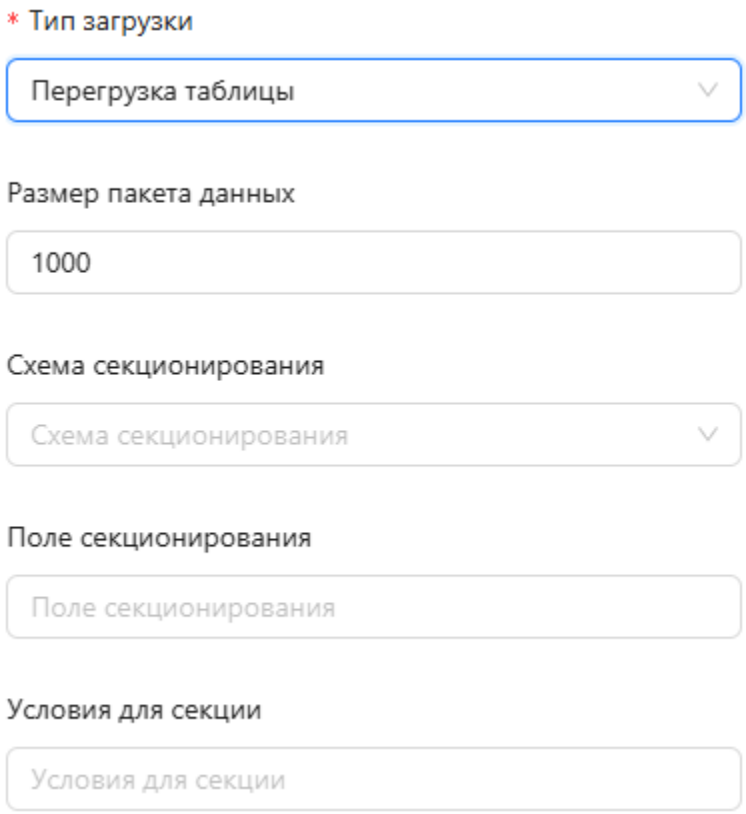


Рисунок. Выбор перезагрузки таблицы

Запрос для инкрементальной загрузки (или загрузки с параметрами) данных отличается от полной и требует настройки дополнительных параметров:

- Partition schema (Схема секционирования) задается исходя из назначения секции и зависит от типа секционирования. Выбирается из выпадающего списка. Создание секции производится на странице Partition (Создание секций);
- Partition column (Поле секционирования) - поле, по которому осуществляется секционирование;
- Partition column convert (Условия секционирования) - условия, характерные для выбранной секции.

В поле Partition schema (схема секционирования) существует два выбора: ora_part и test.

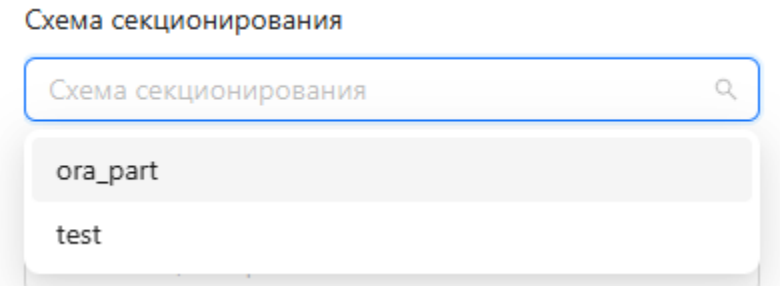


Рисунок. Выпадающий список в поле Partition schema (схема секционирования)

ЗАПУСК НА ВЫПОЛНЕНИЕ

Для того, чтобы запустить созданные команды на выполнение (загрузить данные в хранилище), необходимо на странице Profiles (Профили) выбрать интересующий профиль и нажать кнопку Load (Загрузить), затем в появившемся диалоговом окне подтвердить действия нажатием кнопки Yes (Да).

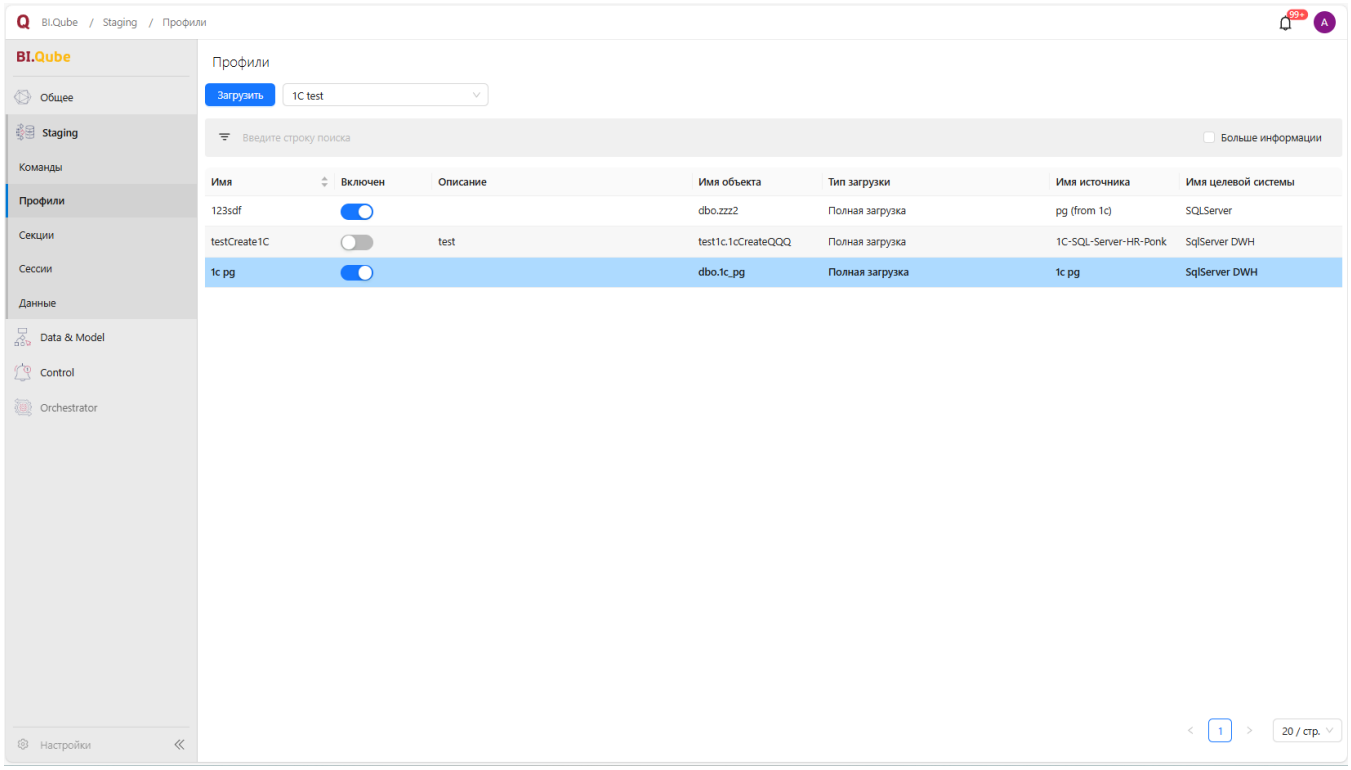


Рисунок. Страница "Профили" компонента Metastaging

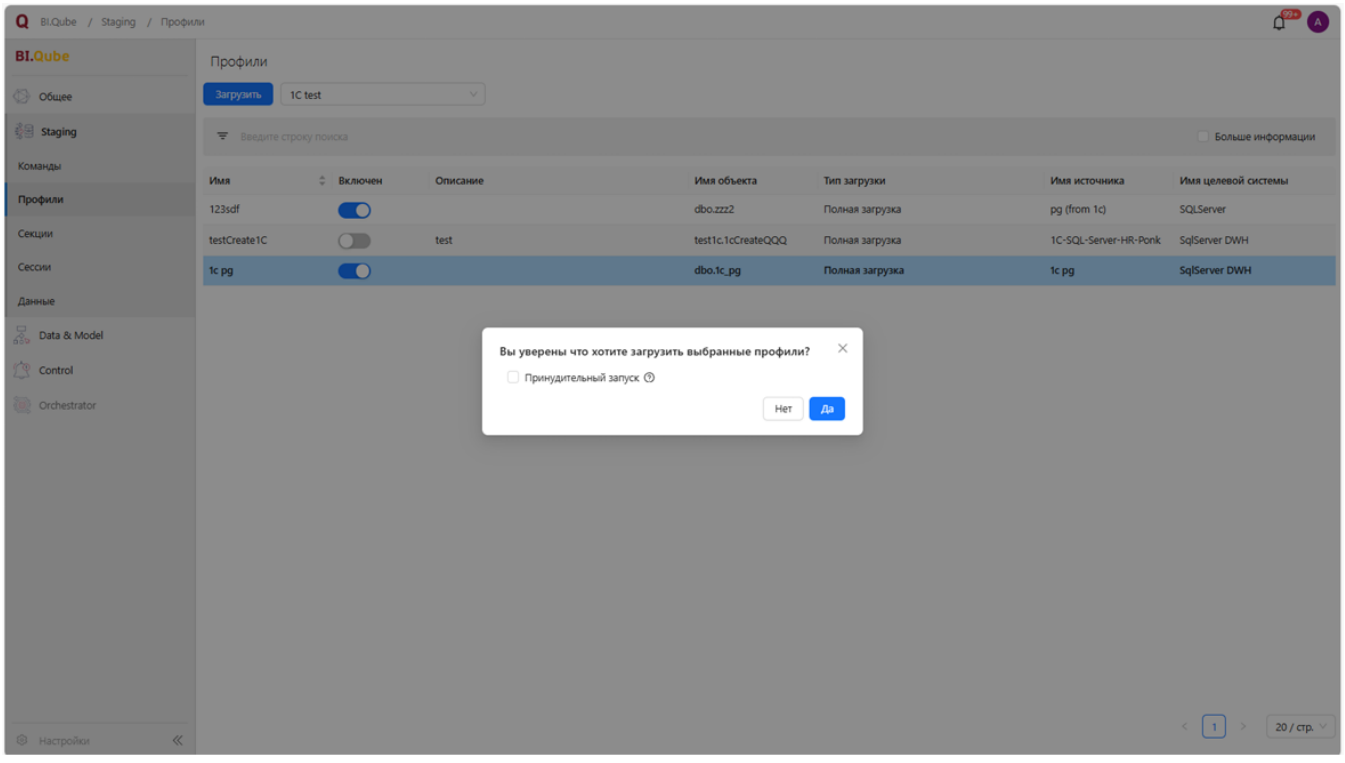


Рисунок. Подтверждение запуска команд профиля на выполнение

В появившемся окне следует нажать "Да", после чего запустится процесс выполнения команд. В некоторых случаях команды не могут быть запущены на выполнение, так как их статус после предыдущего запуска не позволяет выполнить загрузку данных. В этом случае рекомендуется проверить данные в таблицах назначения, уточнить соответствуют ли они ожидаемым. Провести анализ логов, записанных по результатам предыдущих запусков команд и, если все соответствует ожиданиям, запустить профиль на выполнение, а в появившемся окне включить опцию "Принудительный запуск". Статус выполнения команд можно посмотреть в разделе: СЕССИИ - BI.Qube 2.0 Руководство пользователя - Confluence (itprocomp.ru)

Если необходимо в данный момент времени выполнить не все команды профиля, то можно отключить команды, данные из которых не нужны в текущей загрузке. Для этого в таблице следует для нужной команды поле "Включен" необходимо отключить.

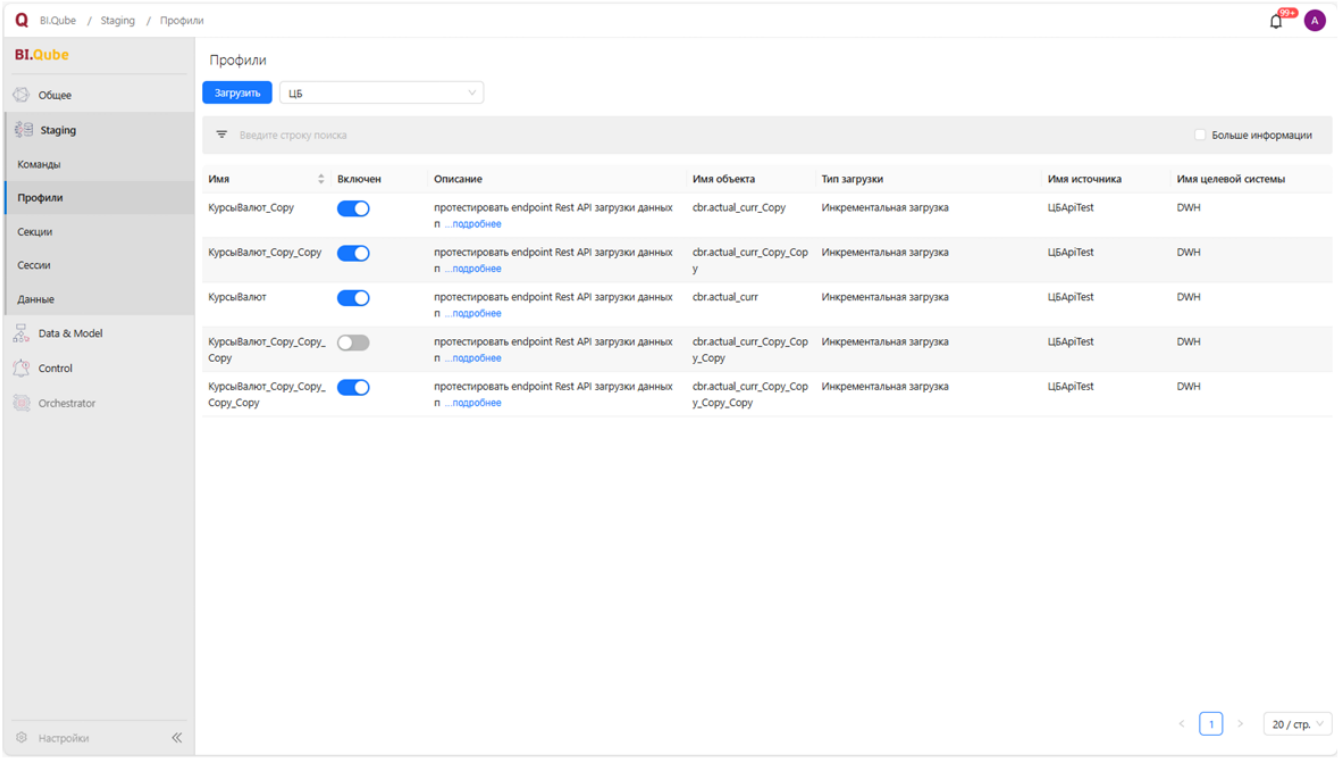


Рисунок. Выбор загруженных данных

СЕКЦИИ

Страница Partition (Секции) предназначена для создания схем секционирования, необходимых для работы инкрементальной загрузки данных.

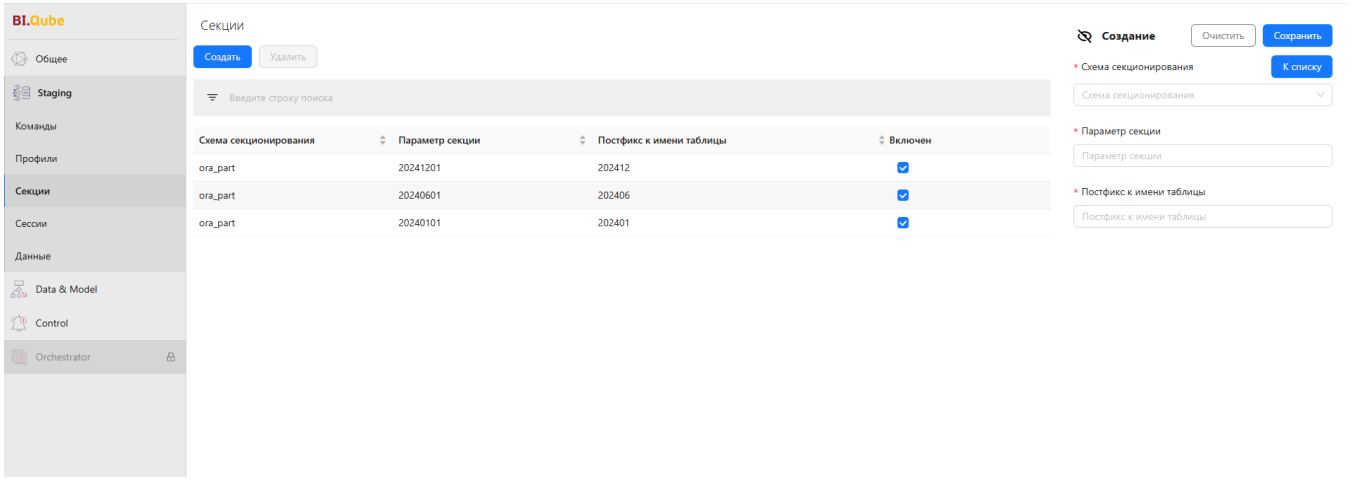


Рисунок. Страница Partition (Секции)

Данные новой схемы заполняются нажатием на кнопку Create (Создать). Далее следует заполнить поля в правой части экрана:

- Partition schema (Схема секционирования) – заполняется выбором из выпадающего списка;
- Partition value (Параметр секции);
- Partition postfix (Постфикс к имени таблицы).

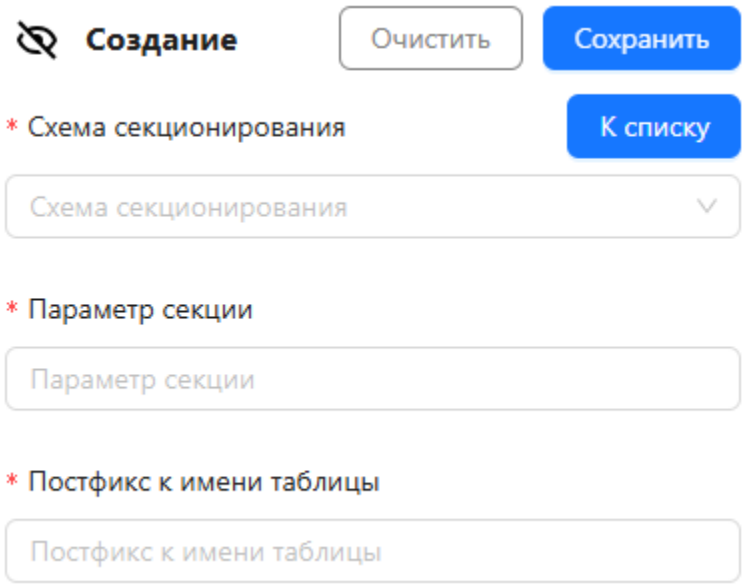


Рисунок. Параметры секционирования

Секция может редактироваться и настраиваться под потребности пользователя, для этого необходимо нажать на кнопку To the list (К списку).

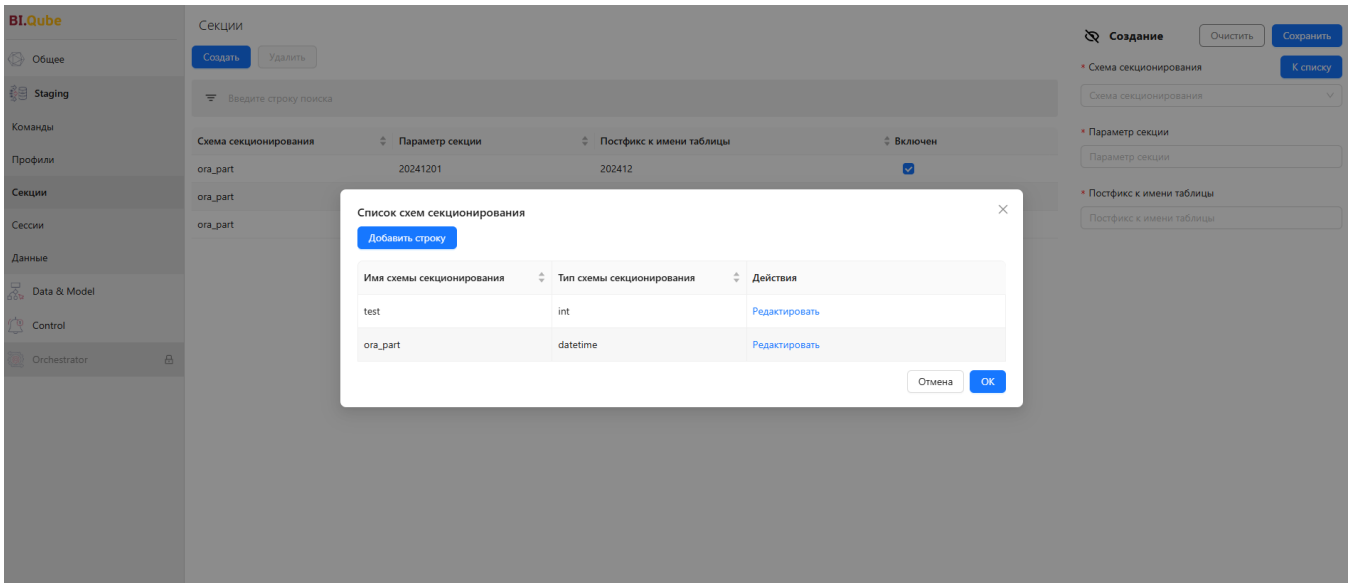


Рисунок. Модальное окно «Список схем секционирования»

Появится модальное окно, в нем три колонки:

- Partition schema name (Имя схемы секционирования);
- Partition schema column type (Тип схемы секционирования);
- Operations (Действия). При нажатии в данной колонке на кнопку Редактировать (Edit) появляется возможность изменить название схемы секционирования, а также задать тип схемы секционирования из выпадающего списка. По окончанию редактирования следует нажать Сохранить (Save), чтобы сохранить внесенные изменения и Ок (OK).

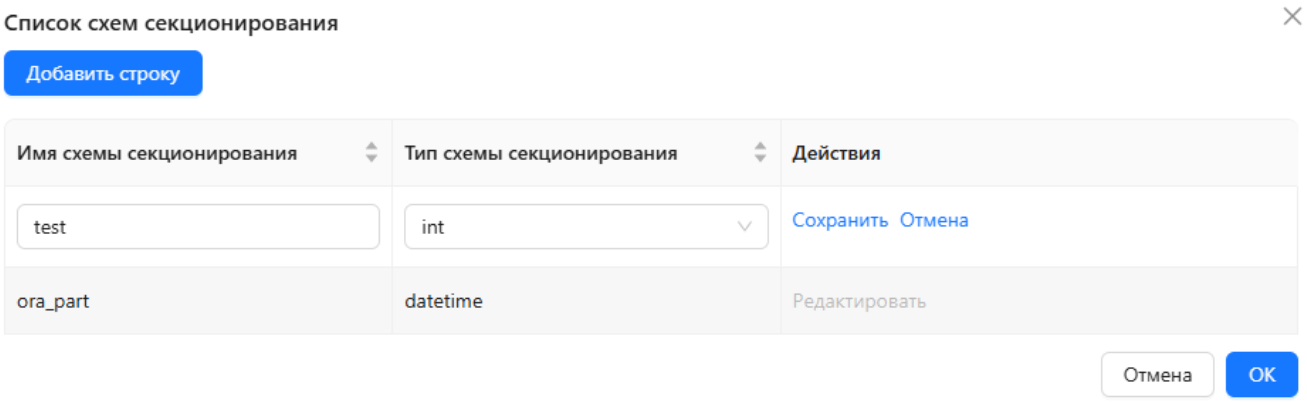


Рисунок. Список схем секционирования. Редактирование

Помимо редактирования можно добавить схему, для этого следует нажать кнопку Add a row (Добавить строку). Далее действия аналогичны процессу редактирования.

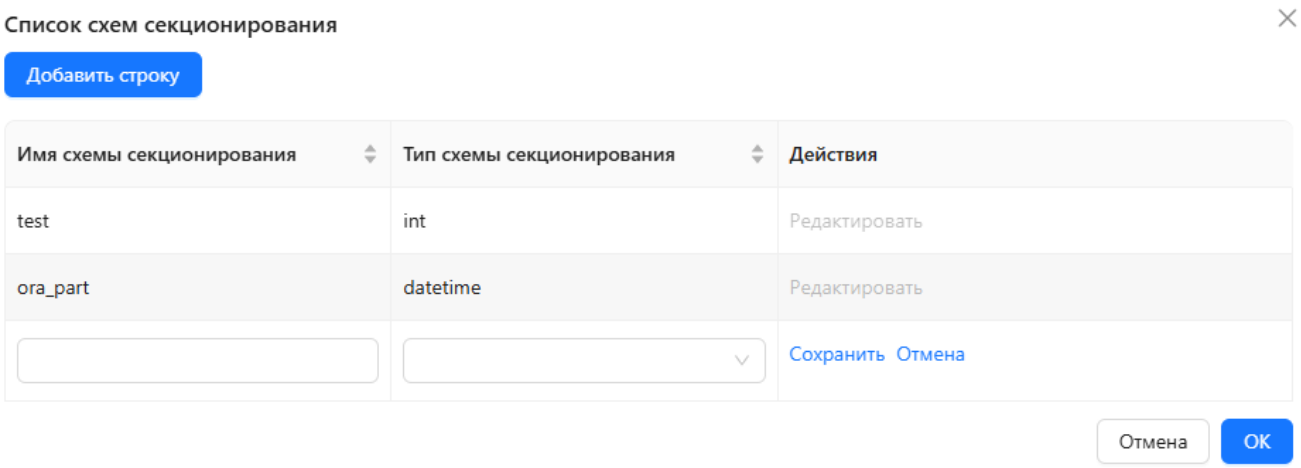


Рисунок. Список схем секционирования. Добавление строки

СЕССИИ

На странице «Сессии» отображаются все сессии загрузки данных (важно не путать с предварительным просмотром при создании команды загрузки). Каждая загрузка подробно логируется и для каждой команды доступна вся история загрузок.

Для просмотра детализации сессии, просмотра какие команды выполнялись в рамках этой сессии, нужно раскрыть знак «+».

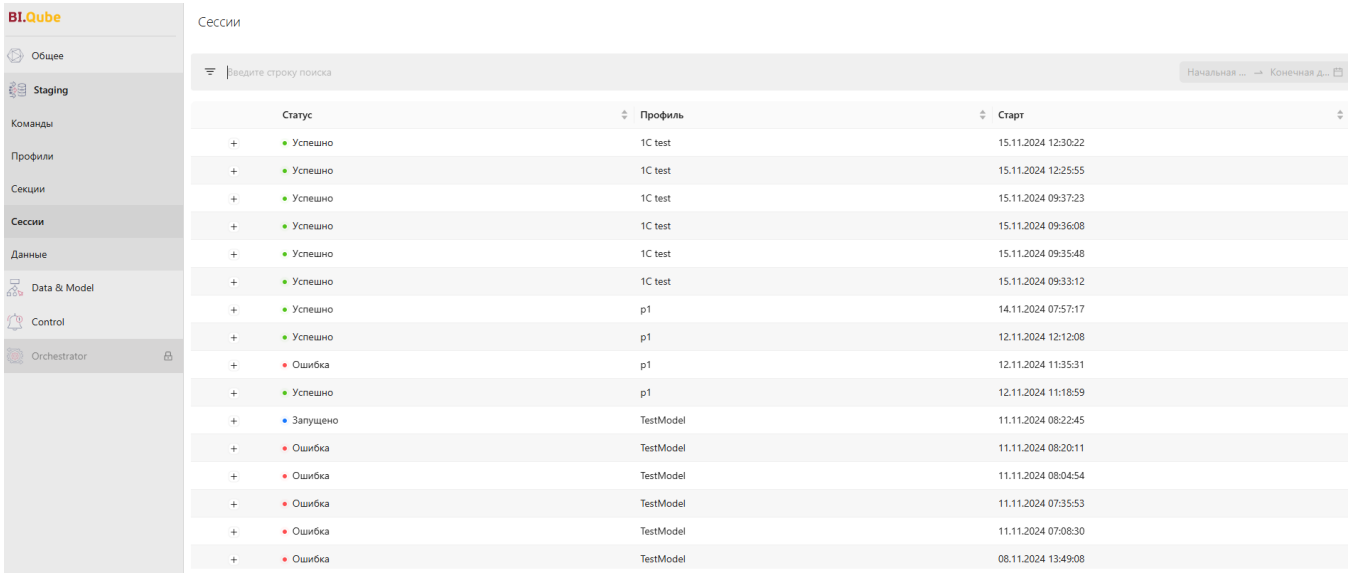


Рисунок. Страница Session (Сессии)

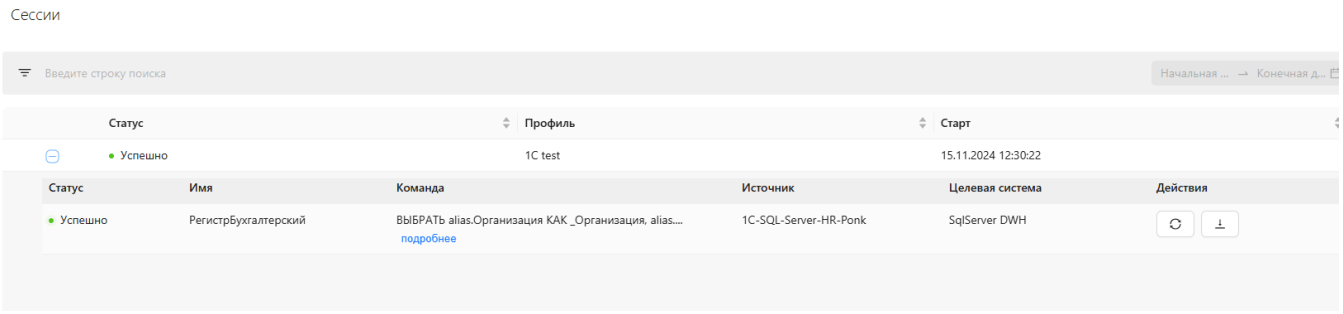


Рисунок. Состав сессии

После раскрытия детальной информации сессии появляется столбец Действия (Actions) с двумя кнопками:

- Перезапустить (Restart)  . По клику на кнопку выбранная команда перезапускается.
- Перейти к командам (Go to command)  . По клику на кнопку происходит переход на страницу Команды.

Для просмотра команды с текущими параметрами следует дважды щелкнуть мышкой по интересующей команде. Таблица на данной странице также содержит кнопку Перезапустить (Restart)  в столбце Действия (Actions).

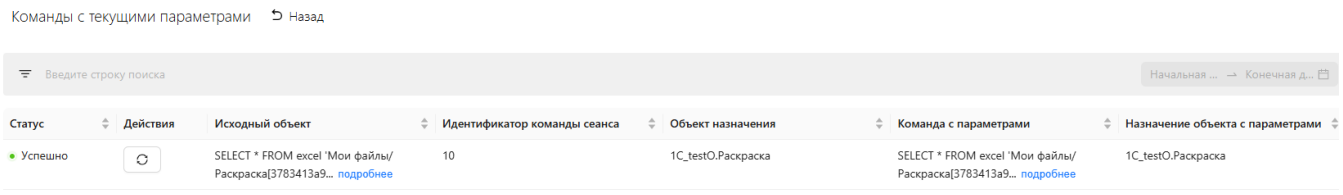
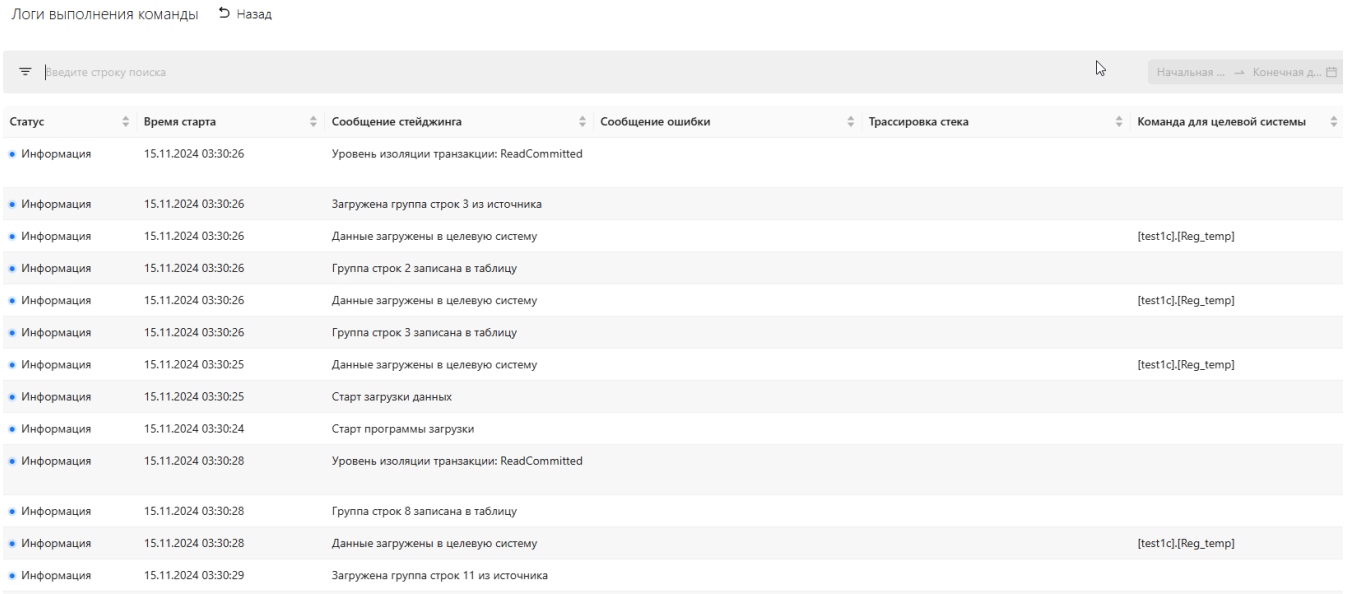


Рисунок. Страница Команды с текущими параметрами

Для просмотра деталей выполнения команды следует снова дважды щелкнуть мышкой по интересующей команде.


Логи выполнения команды Назад

Введите строку поиска

Начальная ... Конечная д...

Статус	Время старта	Сообщение стейджинга	Сообщение ошибки	Трассировка стека	Команда для целевой системы
Информация	15.11.2024 03:30:26	Уровень изоляции транзакции: ReadCommitted			
Информация	15.11.2024 03:30:26	Загружена группа строк 3 из источника			
Информация	15.11.2024 03:30:26	Данные загружены в целевую систему			[test1c].[Reg_temp]
Информация	15.11.2024 03:30:26	Группа строк 2 записана в таблицу			
Информация	15.11.2024 03:30:26	Данные загружены в целевую систему			[test1c].[Reg_temp]
Информация	15.11.2024 03:30:26	Группа строк 3 записана в таблицу			
Информация	15.11.2024 03:30:25	Данные загружены в целевую систему			[test1c].[Reg_temp]
Информация	15.11.2024 03:30:25	Старт загрузки данных			
Информация	15.11.2024 03:30:24	Старт программы загрузки			
Информация	15.11.2024 03:30:28	Уровень изоляции транзакции: ReadCommitted			
Информация	15.11.2024 03:30:28	Группа строк 8 записана в таблицу			
Информация	15.11.2024 03:30:28	Данные загружены в целевую систему			[test1c].[Reg_temp]
Информация	15.11.2024 03:30:29	Загружена группа строк 11 из источника			

Рисунок. Страница, демонстрирующая детальные сведения о выполнении команды

Статусы выполнения команд

Статусы проставляются в таблицу в соответствии с перечислением:

- Skipped - команда не была выполнена из-за завершения сессии, т.е. выполнение команды даже не началось. По завершению сессии все команды со статусом Queued переводятся в Skipped через вызов ХП при использовании оркестратора, а при запуске через Backend через ProfileController;
- **Success - команда отработала без ошибок, данные загружены;**
- **Running - команда в процессе загрузки (нельзя запускать данную команду в других профилях);**
- **Failed - команда отработала с ошибками (см. подробные логи);**
- **Queued - команда в очереди на загрузку (нельзя запускать данную команду в других профилях);**
- **Debug - отладка команды (для внутренних задач, в т.ч. значение по умолчанию в БД);**
- **NoData - команда отработала без ошибок, но данные из источника не загружены (возможно их нет в источнике).**

Все статусы, кроме Skipped проставляются внутри экстрактора.

Статусы сессий загрузки

Статусы сессия не хранятся в БД, а являются вычисляемыми.

- **Success** - ни одна команда в сессии не имеет статус Running, Failed, Queued, Skipped;
- **Running - как минимум одна команда имеет статус Running;**
- **Failed - как минимум одна команда имеет статус Failed.**

ДАННЫЕ METASTAGING

Страница Data (Данные) позволяет пользователю посмотреть визуально загруженные данные в хранилище, здесь же есть возможность выполнить любые запросы, на основе которых можно убедиться в качестве полученных данных.

Справа в строке необходимо выбрать тот тип загрузки, который выбирали ранее. Затем раскрываем дерево файлов, нажатием на плюсик, и находим данные.

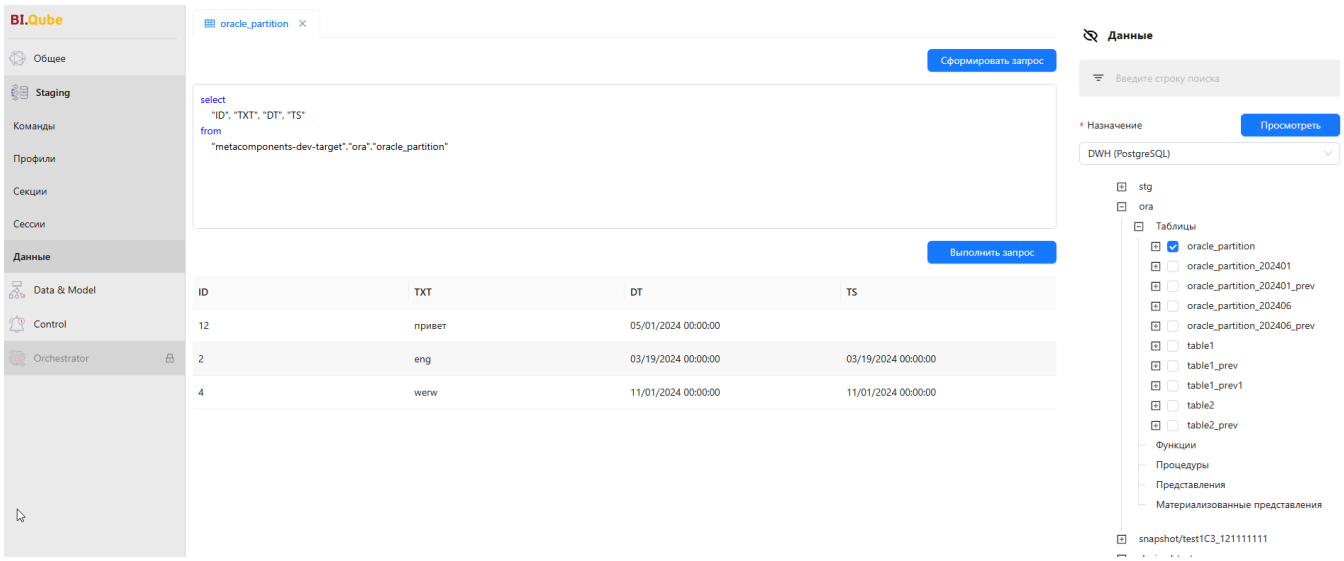


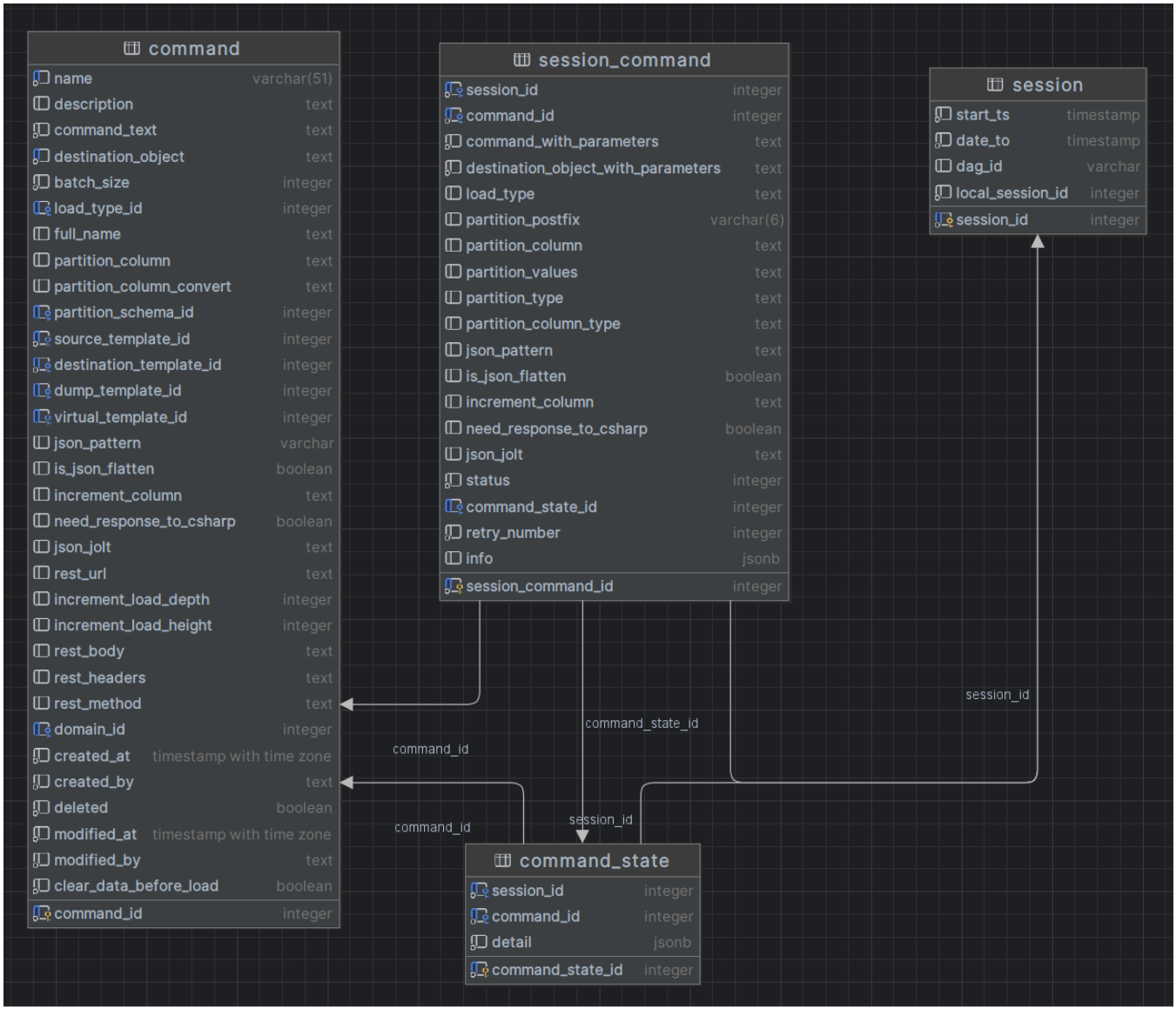
Рисунок. Просмотр загруженных данных

В разработке. Здесь же есть возможность создавать хранимые процедуры и другие объекты базы данных необходимые для поддержки работы хранилища.

ТАБЛИЦЫ ЛОГОВ КОМПОНЕНТА

Общие сведения о правилах записи логов в базу данных

Все действия, выполняемые командой извлечения данных из источника, записываются в таблицы логов. При необходимости они могут быть использованы, например, при подготовке параметров. Ниже на диаграмме приведены связи между таблицами логов. Таблица command хранит информацию о свойствах команды, системные параметры, например current_command_id=command_id, берут информацию как раз из этой таблицы. При выполнении команды создается объект типа сессия session_id в таблице, настройками запущена команда на выполнение. Команда может быть "размножена" на подкоманды в зависимости от значений, подставленных в параметры. Запрос с фактически подставленными параметрами хранится в таблице session_command в поле command_object_with_ parameter. Выполнение каждой команды разбивается на отдельные шаги, и запуск каждого шага фиксируется в таблице extractor_log с указанием статуса выполнения шага. В таблице command_state хранится состояние команды на момент ее запуска (далее, данные из этой таблицы будут использоваться для перезапуска команды).



Описание таблиц логов

В таблицах ниже приведено описание назначения полей таблиц логов и служебных таблиц. Описание дано только для полей, которые можно использовать пользователю при создании SQL-кода параметров или в каких-то других запросов.

Таблица Session

Имя столбца	Тип данных	Назначение
sessionID	int	Идентификатор сессии
start_ts	timestamp	Дата и время запуска сессии
date to	timestamp	Дата окончания сессии
success	bool	Идентификатор успешности загрузки в рамках данной сессии (статусы)
dag_id	varchar	Имя профиля, выполняемого в сессии (с использованием скриптов оркестрации передается в качестве имени Dag в airflow)
local_session_id	int	Локальный идентификатор сессии относительно дня (когда запрос выполняется несколько раз за день)

Таблица Session_command

Имя столбца	Тип данных	Назначение
session_command ID	int	Идентификатор строки
commad_with_parameters	text	Окончательный текст запроса, отправляемого на источник данных (подставлены конкретные значения параметров запроса, если таковые есть)
destination_object_with_parameters	text	Полный путь к файлу parquet в рамках S3-совместимого хранилища
session _ID	int	Идентификатор сессии
commad_ ID	int	Идентификатор команды
partition_postfix	varchar(6)	Строка используется для наименования таблиц-секций при секционированной загрузке. Чаще всего это дата.
partition_column	text	Название столбца, по которому нужно параметризовать запрос к источнику. В конец запроса добавляется WHERE partition_column > 'partition_value1' AND partition_coumn < 'partition_value2'
partition_values	text	Значения через запятую для секционирования. Например ДатаС, ДатаПо
partition_type	text	Тип секционирования (поддерживается только range – это внутренняя настройка для Postgres)
partition_column_type	text	Тип столбца секционирования (datetime, int, string)
json_pattern	text	Служебное поле
is_json_flatten	text	Служебное поле
increment_column	text	Название поля, по которому осуществляется инкрементальная загрузка
need_response_to_csharp	bool	Служебное поле
json_jolt	text	Служебное поле
status	int4	Статус команды в рамках выполнения сессии загрузки данных
command_state_id	int	Идентификатор состояния команды
retry_number	int	Номер перезапуска

Таблица Extractor_log

log_level	**категория события: Debug, Trace, Info, Warning, Error, Fatal**
timestamp	Дата и время добавления записи о событии, записывается в utc
session_command_id	Идентификатор команды сессии, к которой принадлежит записываемое событие
message	Произвольное сообщение
exception	Исключение, которое произошло в процессе выполнения команды, содержит в себе полную информацию об ошибке, в т.ч. стектрейс
command	Текст выполняемой команды (для некоторых сообщений может отсутствовать), может содержать как команду к источнику, так и команду к целевой базе

processed_rows	Количество обработанных строк
processed_bytes	Количество обработанных байт
retry_number	Номер перезапуска размноженной команды

Таблица Command

command_id	int	Идентификатор команды
name	text	Просто Имя
description	text	Описание команды
command	text	Текст запроса с возможностью параметризации через связь с таблицей stg.parameter. Секционировать можно только простые запросы, без условий WHERE и тд. Для этого команда должна ссылаться на табл. stg.partition_schema
destination_object	text	Название целевого объекта. В процессе выполнения команды может ИЗМЕНИТЬСЯ. Есть схемка, которая это демонстрирует.
batch_size	int4	Количество строк, выгружаемых из источников в файл Parquet за одну итерацию при пакетной загрузке
load_type_id	int4	Тип загрузки
full_name	text	Служебное поле
partition_column	text	Если грузим в GP инкрементально, то обязательно задавать это поле. Если грузим в PG секционировано, то обязательно задавать это поле. Иначе можно NULL. (Поле в запросе, по которому выполняется секционирование на представлениях Greenplum. Если данное поле задано (например, UpdatedAt), в запросе можно писать так /*{partition_column}*/ >= /*{datefrom}*/)
partition_column_convert	text	Поле содержит логику конвертации для значения в partition_column. Данная логика будет отражена в представлении на Greenplum. Пример: cast(/*{partition_column}*/ as bigint)
partition_schema_id	int4	Ссылка на схему секционирования – табл. stg.partition_schema. Если грузим в PG секционировано, то обязательно задавать это поле. Иначе можно NULL
source_template_id	int4	Идентификатор подключения, выступающего источником для команды
destination_template_id int4 NOT NULL,	int4	Служебное поле
dump_template_id int4 NULL,	int4	Служебное поле
virtual_template_id int4 NULL,	int4	Служебное поле
json_pattern	varchar	Служебное поле
is_json_flatten bool NULL,	bool	Служебное поле
increment_column text NULL,	text	Служебное поле

need_response_to_csharp	bool	Служебное поле
json_jolt	text	Служебное поле
rest_url	text	Служебное поле
increment_load_depth	int4	Служебное поле
increment_load_height	int4	Служебное поле
rest_body	text	Служебное поле
rest_headers	text	Служебное поле
rest_method	text	Служебное поле
domain_id	int4	Служебное поле

Таблица CommandState

command_state_id	int	Идентификатор команды
session_id	int	Идентификатор сессии
detail	json	Настройки команды (строка подключения назначения и источника, тип загрузки, запрос и т. д.)
command_id	int	Идентификатор команды

Связь системных параметров с таблицами логов и служебными таблицами

Ниже дан перечень системных параметров с указанием источника данных для них, конкретное значение определяется в зависимости от контекста вычисления параметра.

- current_command_id – возвращает числовое значение идентификатора команды извлечения данных, в запросе которой вычисляется значение параметра. current_command_id=stg.command.command_id;
- current_command_source_id – возвращает числовое значения идентификатора источника данных для команды, в запросе которой вычисляется значение параметра. current_command_source_id=stg.command.source_template_id;
- current_command_source_objectname – возвращает текстовую строку, содержащую имя объекта (только для sql-запросов к источникам типа СУБД), являющегося источником данных для команды, в запросе которой вычисляется значение параметра. current_command_source_objectname=stg.command.command – из запроса извлекается имя таблицы на источнике данных;
- current_command_distination_id – возвращает числовое значения идентификатора базы данных, в которую предполагается запись извлеченных данных, команды, в запросе которой вычисляется значение параметра. current_command_distination_id=stg.command.destination_template_id;
- current_command_source_objectname – возвращает текстовую строку, содержащую имя объекта, в который выполняется запись данных, командой, в запросе которой вычисляется значение параметра. current_command_source_objectname=stg.command.destination_object;
- `last_command_load -  .`

Расширенный инструмент работы с логами

Все действия, происходящие в системе, детально логируются. Если выше был описан подход логирования выполнения команды, то в этом разделе речь идет о логировании всех значимых действий пользователя. Данные логи нужны для отладки работы системы, и в ряде случаев без этой информации разработчику не представляется возможным понять причины произошедшей ошибки.

Напрямую к логам у пользователя нет доступа, они доступны администратору системы и располагаются в файлах по адресу: рабочая папка приложения, внутри папка **logs,** файл с названием **ItPro.Metastaging.Backend-текущая_дата.log**.

Кроме этого логи дублируются в таблицах конфигурационной базы данных: для MetaStaging схема **stg**, таблица **logs**.

В случае если в системе происходят какие-то ошибки, то при обращении в службу поддержки обязательно присылать файл с логами, соответствующий дате появления ошибок.

DATA&MODEL

Общие сведения

Data&Model – компонент предназначен для создания масштабируемой модели данных, работой со справочниками, обогащения данных. Компонент работает с данными, получаемыми с использованием компонента MetaStaging. Включает в себя компонент MetaVault и MetaMasterData;

- **MetaVault** – компонент, имеющий развитый визуальный интерфейс, реализующий работу в режиме no code/ low code и предназначенный для организации хранения данных в модели DataVault. Пользователь может не иметь представления об особенностях модели DataVault. Система все необходимые действия выполняет сама и предоставляет доступ к автоматически сгенерированным представлениям;
- **MetaMasterData** – компонент, имеющий развитый визуальный интерфейс, реализующий работу в режиме no code/ low code и предназначенный для работы с нормативно-справочной информацией, обогащения данными, вводимыми в ручном режиме через веб интерфейс, создания новых данных. Данный компонент работает только в связке с MetaVault и отдельно работать не может. Компонент реализует возможности MDM систем и создание с его помощью объекты не требуют интеграции с объектами MetaVault;

Основным понятием при работе компонента является понятие **модель данных.**

Модель данных — это абстрактное, самодостаточное, логическое определение объектов, операторов и прочих элементов, в совокупности составляющих абстрактную машину доступа к данным, с которой взаимодействует пользователь. Эти объекты позволяют моделировать структуру данных, а операторы — поведение данных.

Основными объектами модели данных, используемыми в компоненте **Data&Model** является Сущность и связь. Сущность, простыми словами является классической двумерной таблицей и используется для хранения данных. Связь – это некоторое логическое соединение данных из разных сущностей. Для предоставления большей гибкости при работе с данными на физическом уровне одна сущность представляется несколькими таблицами, такими как Хаб (hub) и Сателит. Связи между сущностями создаются с использование отдельной таблицей, называемой Линком. Все эти термины заимствованы из модели данных DataVault.

Компонент **Data&Model** работает с двумя видами сущностями (внутренняя терминология BI.Qube):

- Сущности, созданные на основе данных
- Сущности, создаваемые пользователем

К сущностям первого вида относятся таблицы, которые имеют источник данных, представленный, в самом простом случае, таблицей в базе данных. Сущности второго типа создаются средствами BI.Qube. В первом случае данные в создаваемую сущность попадают из таблиц источников (таблиц базы данных). Во втором случае сущности заполняются пользователем с клавиатуры.

Кроме этого, доступен так называемый гибридный тип, когда к сущностям первого типа можно добавлять новые поля и редактировать данные в таких полях. Редактировать или удалять поля и данные, созданные автоматически на основе метаданных источников, невозможно.

Описание компонента

ПРОФИЛЬ DATA & MODEL

Для просмотра созданных профилей необходимо зайти в "DATA & MODEL" во вкладку Profiles (Профили). (Рисунок. Пример созданного профиля)

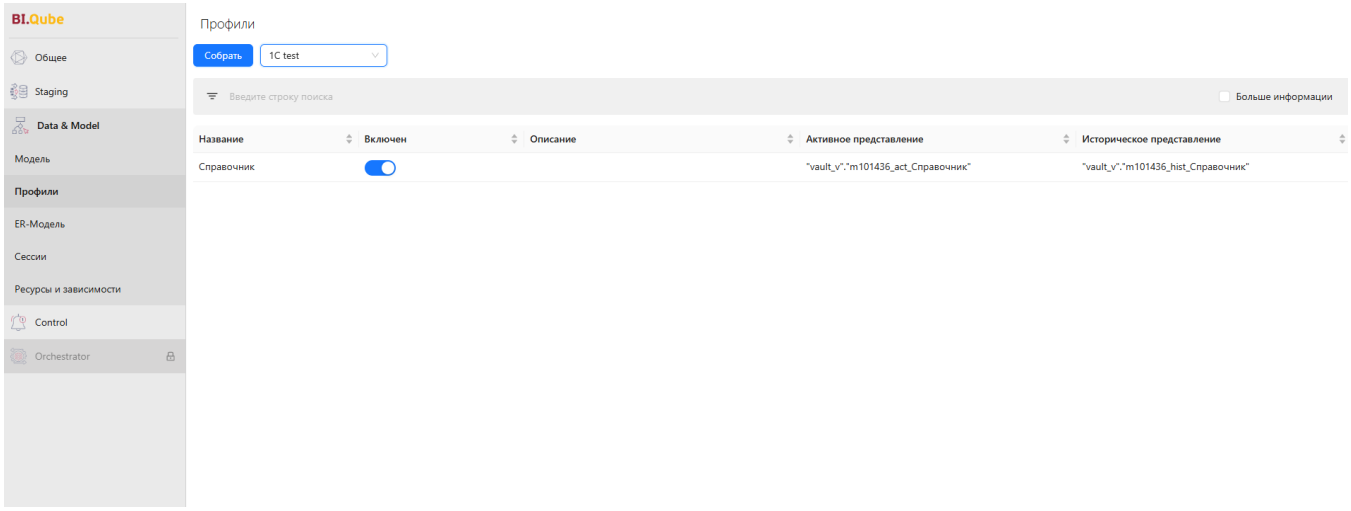


Рисунок. Пример созданного профиля

Для просмотра и выбора необходимо указать нужный профиль в выпадающем списке. (Рисунок. Выбор профиля).

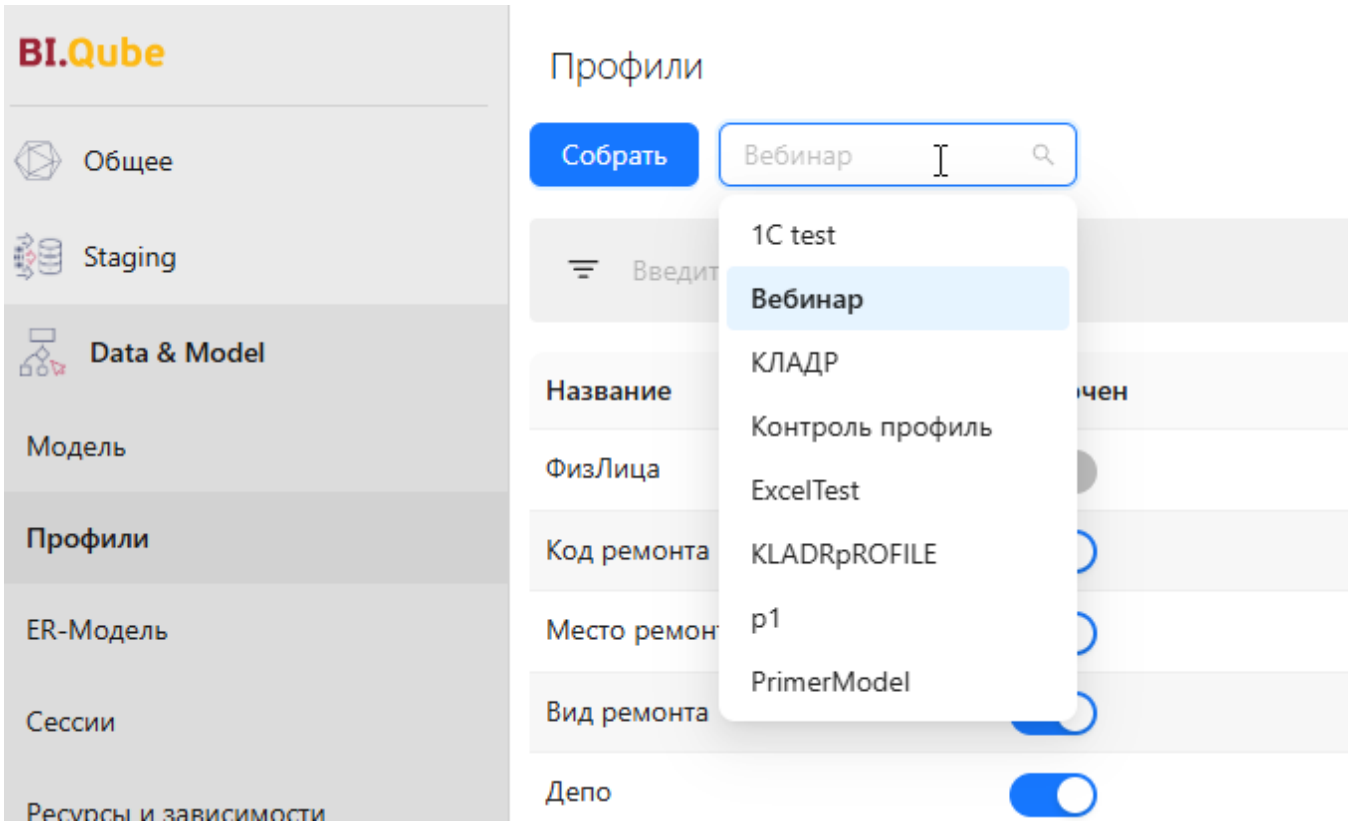


Рисунок. Выбор профиля

Для того, чтобы загрузить необходимые сущности (таблицы) достаточно перевести ползунки в столбце Включен в положение "True" и нажать на кнопку Load (Загрузить). В появившемся диалоговом окне нажать на кнопку ОК (Ок). Также в появившемся диалоговом окне есть возможность указать промежуток времени для сборки (выбрать начальную дату и конечную). Дата выбирается с помощью календаря или вручную по маске yyyy-mm-dd.

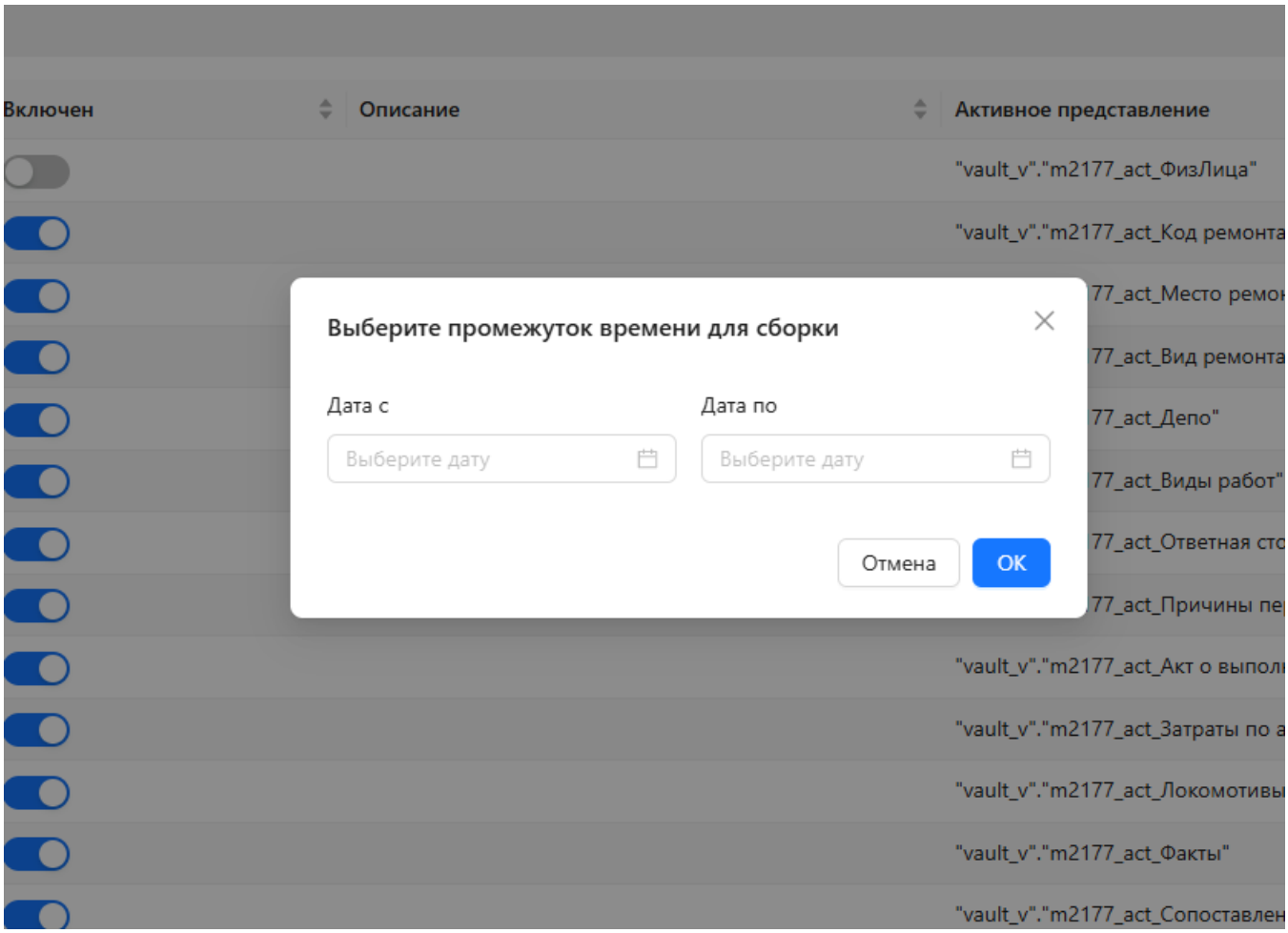


Рисунок. Выбор и загрузка сущностей (таблиц)

Для просмотра дополнительной информации по сущностям (таблицам) необходимо поставить галочку More info (Больше информации)

СОЗДАНИЕ МОДЕЛИ

Создание модели данных осуществляется на странице Models (Модель), нажатием на кнопку Create (Создать) создаются поля для заполнения в правой части экрана.

Необходимо заполнить поля:

- Name (Имя) – имя модели данных;
- Endpoint (Подключение) – представлен выпадающим списком с доступными подключениями;
- Description (Описание) – бизнес-описание модели данных, как правило, дается описание назначения модели данных.
- Default schema (Базовая схема) – схема, которая будет использоваться для справочника. Выбор схем идет из тех, которые существуют в подключении, выбранном для модели.

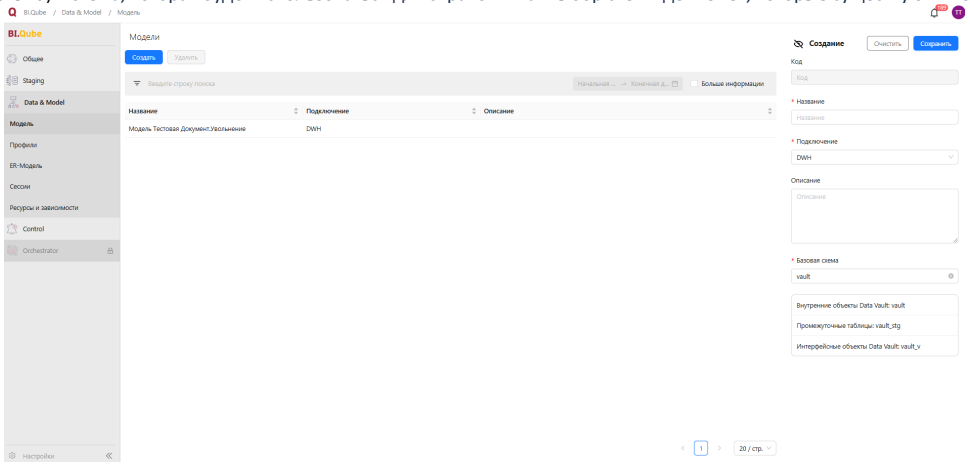


Рисунок. Создание модели данных

При выборе базовой схемы (Default schema) появляется следующая информация:

- Внутренние объекты Data Vault (Inner Data Vault objects): название – схема для таблиц: хабов (hub), линков (link), сателлит (satellite) (слой Raw Data Vault);
- Промежуточные таблицы (Intermediate tables): название – схема, в которой будут созданы таблицы с данными, вводимыми пользователем;
- Интерфейсные объекты Data Vault (Data Vault business objects): название – активные и историчные представления, результат работы Meta Vault (слой Business Data Vault).

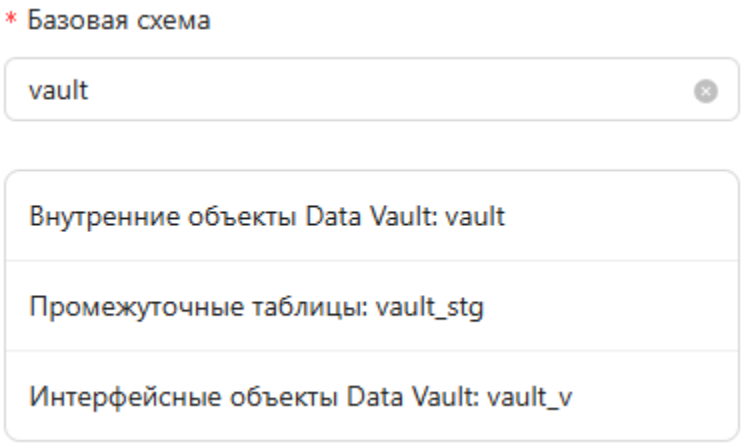


Рисунок. Пример выбора базовой схемы

Редактирование имени и описание выполняется аналогичным образом, щелкнуть левой кнопки мыши по строке модели в центральной части экрана, внести в правой части, в окне свойств необходимые изменения и нажать кнопку Update (Обновить).

Для просмотра содержимого модели данных необходимо дважды щелкнуть левой кнопкой мыши по строке модели после чего на экране появится список сущностей входящих в эту модель.

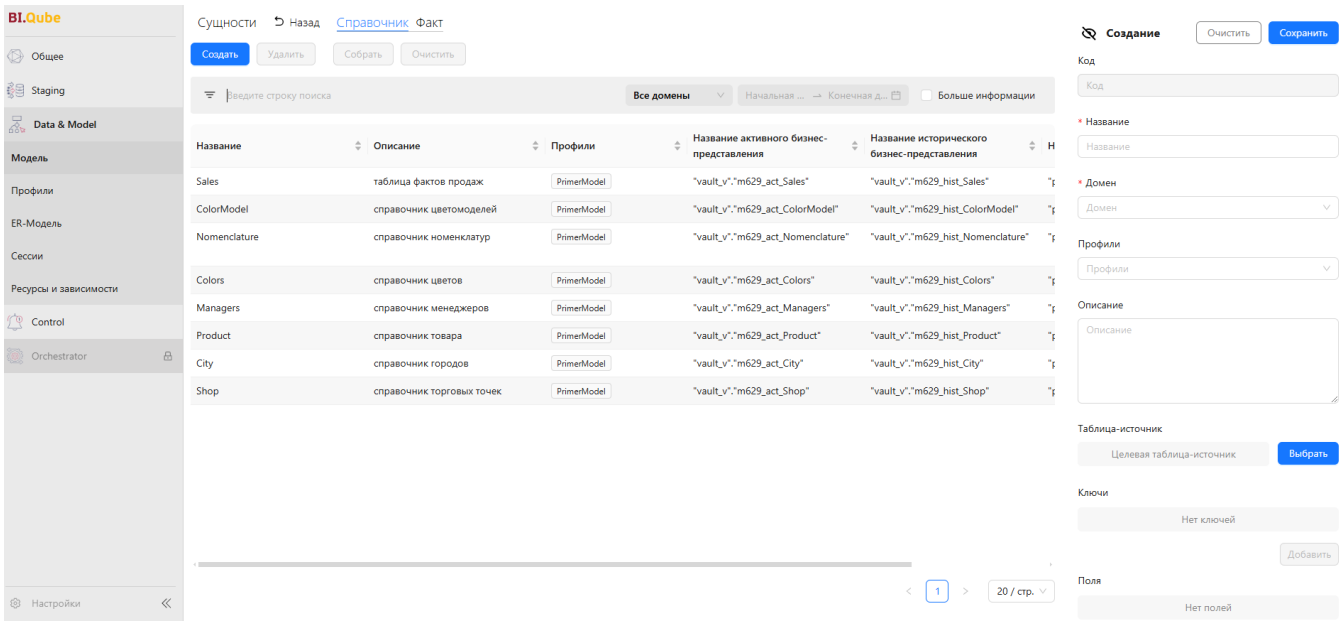


Рисунок. Сущности модели данных

В списке сущностей есть дополнительные поля: первые два – это целевые объекты, которые создаются нашими метакомпонентами.

Первое поле – это актуальные данные, второе – это вся история изменений, третье – это источник данных для сущности, если он есть.

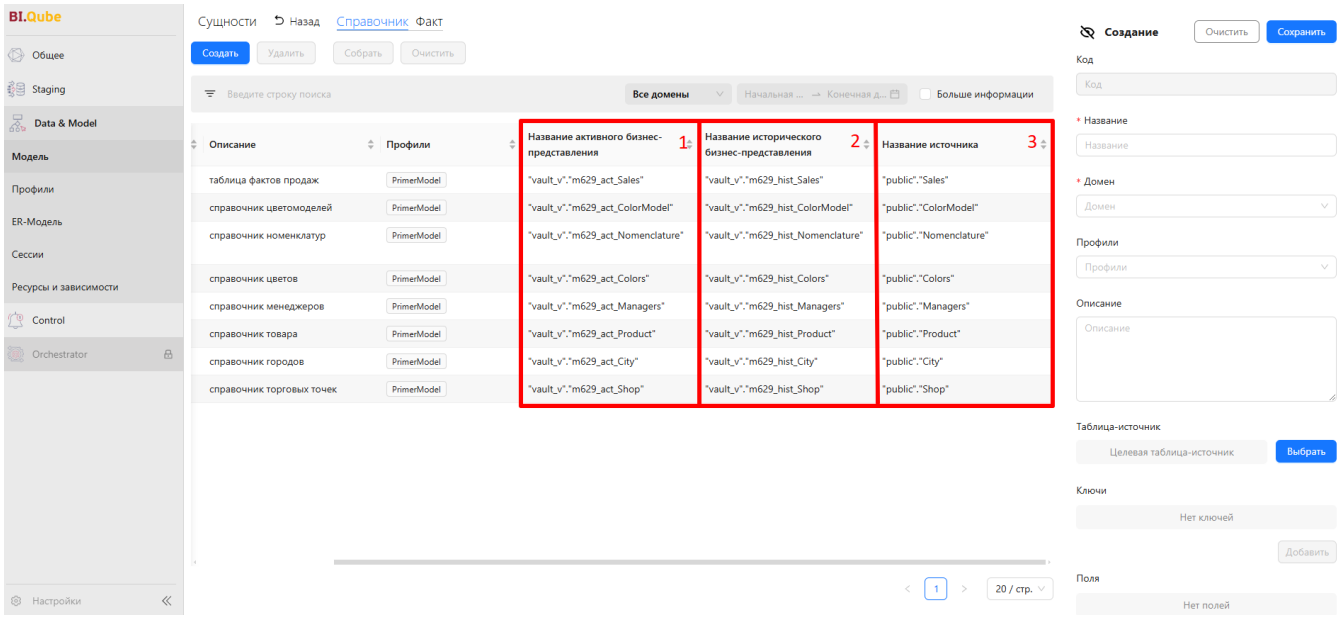


Рисунок. Дополнительные поля для сущности в Data&Model

Создание сущности "Справочник" в модели

Создание сущности "Справочник"

Создание сущности происходит стандартным образом. Необходимо, находясь в модели, нажать на кнопку Create (Создать). Справа в окне свойств появится перечень свойств которые нужно заполнить:

- Name (Название) – имя сущности;
- Domen (Домен) – выбрать домен, в который будет входить сущность. Одна сущность может принадлежать только одному домену;
- Profile (Профиль) – выбрать профиль, к которому будет принадлежать сущность. Сущность может принадлежать нескольким профилям;
- Description (Описание) – бизнес описание назначения сущности;
- Source table (Таблица-источник) – ссылка на таблицу (привязка источника данных к создаваемой сущности), из которой данные будут попадать в создаваемую сущность;
- Keys (Ключи) – создание ключевых полей сущности (доступно, только при наличии источника данных);
- Attributes (Поля) – поля создаваемой сущности. Поля создаются либо путем выбора команды «Add manual» (Создать ручной) для создания нового атрибута или команды «Add» (Добавить) скопировать атрибут из источника привязанного к создаваемой сущности;
- Links (Ссылки) – инструмент создания связей между сущностями.

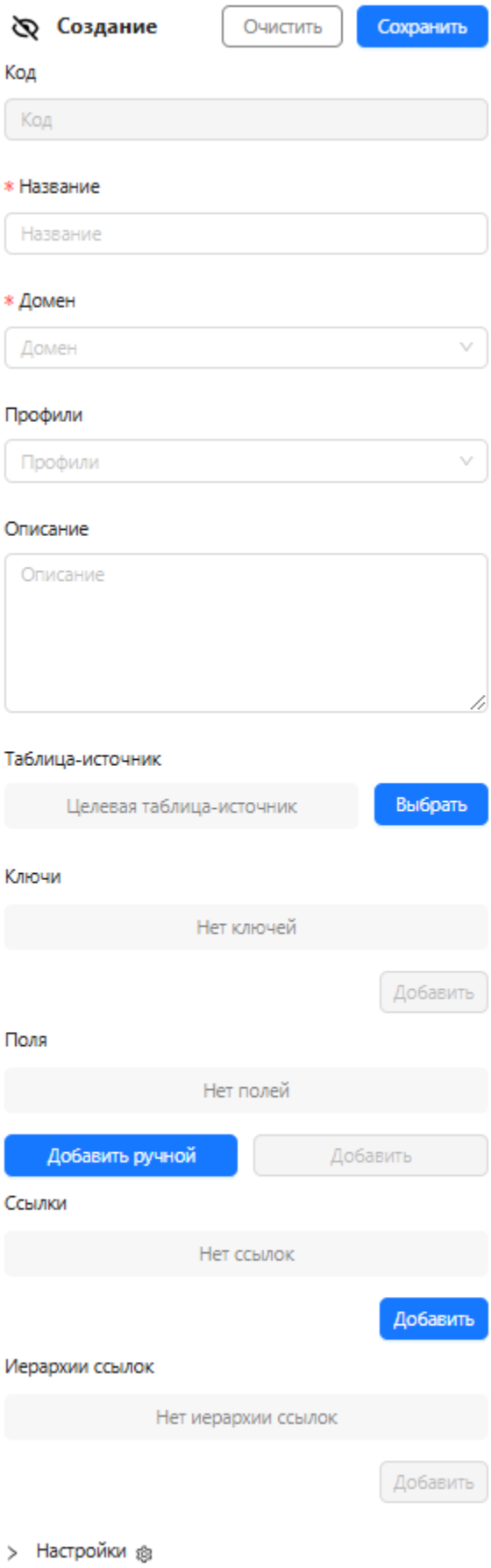

Создание
Очистить
Сохранить
Код
Код
* Название
Название
* Домен
Домен
Профили
Профили
Описание
Описание
Таблица-источник
Целевая таблица-источник
Выбрать
Ключи
Нет ключей
Добавить
Поля
Нет полей
Добавить ручной
Добавить
Ссылки
Нет ссылок
Добавить
Иерархии ссылок
Нет иерархии ссылок
Добавить
Настройки

Рисунок. Свойства сущности

Нажав на кнопку Settings (Настройки), дополнительно можно указать следующие свойства (Рис. Настройки сущности):

- Materialize active view (Активное представление) – материализация данных в бизнес-представлении;
- Materialize historical view (Историчное представление) – материализация данных изменений в бизнес-представлении;
- Type of surrogate key (Тип суррогатного ключа) – тип искусственно созданного ключевого поля;
- Attributes in linked entities (Атрибуты в связанных сущностях) – представлен выпадающим списком указанных атрибутов с возможностью множественного выбора;
- Default schema (Базовая схема) – схема, которая будет использоваться для справочника. Выбор схем идет из тех, которые существуют в подключении, выбранном для модели.

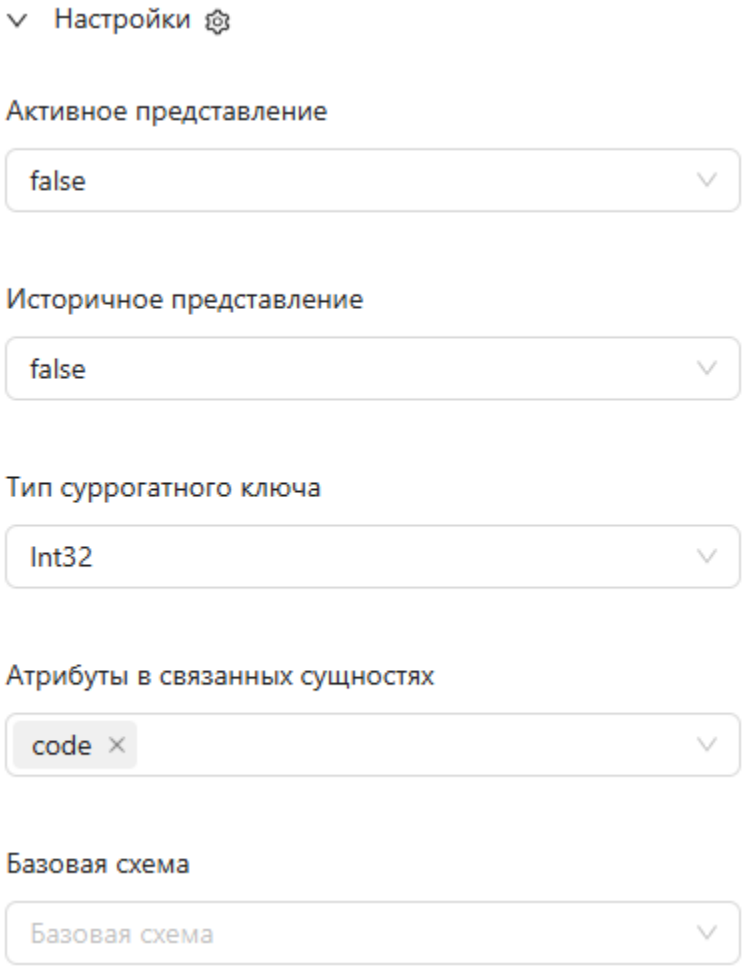


Рисунок. Настройки сущности

После создания сущности необходимо нажать кнопку Save (Сохранить), новая запись о созданной сущности появится в списке сущностей текущей модели данных.

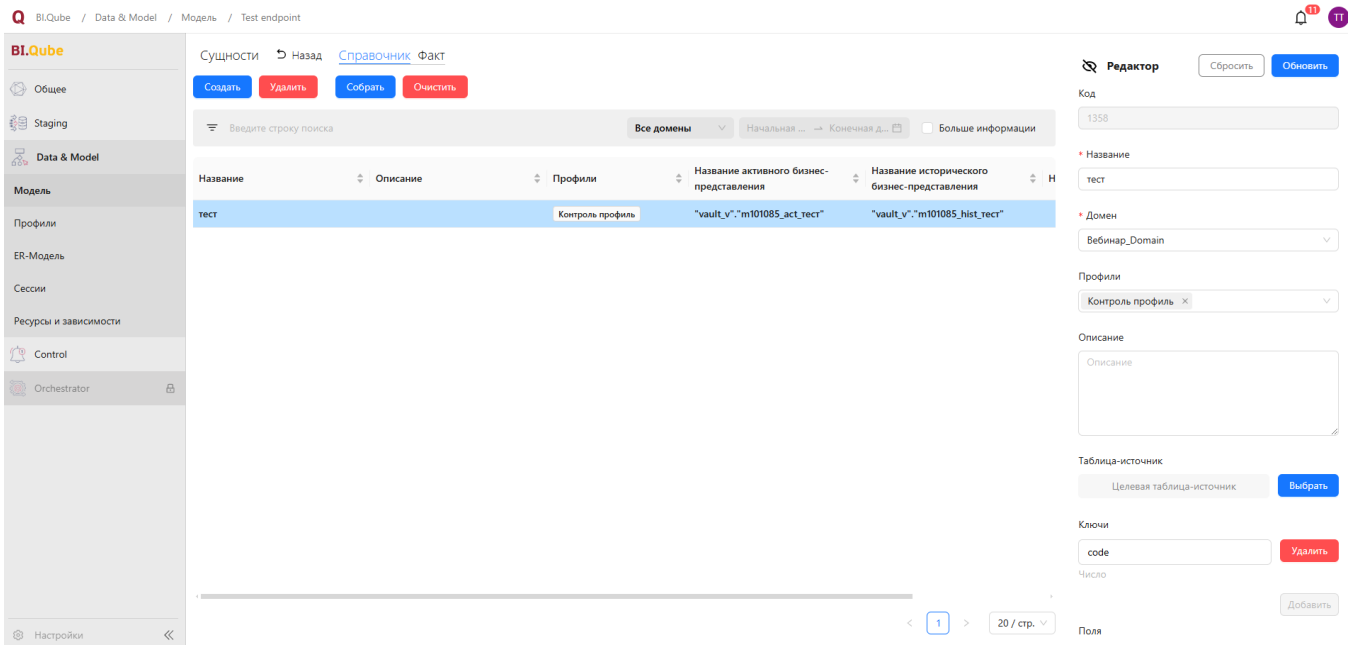


Рисунок. Созданная сущность "тест"

Редактирование метаданных сущности

При выборе любой сущности (таблицы) можно выбирать любой атрибут для редактирования щёлкнув по нему в окне свойств справа. В открывшемся диалоговом окне в поле Description (Описание) пользователь может задать любое описание атрибута и изменять его название в интерфейсе. Это возможно для всех атрибутов.

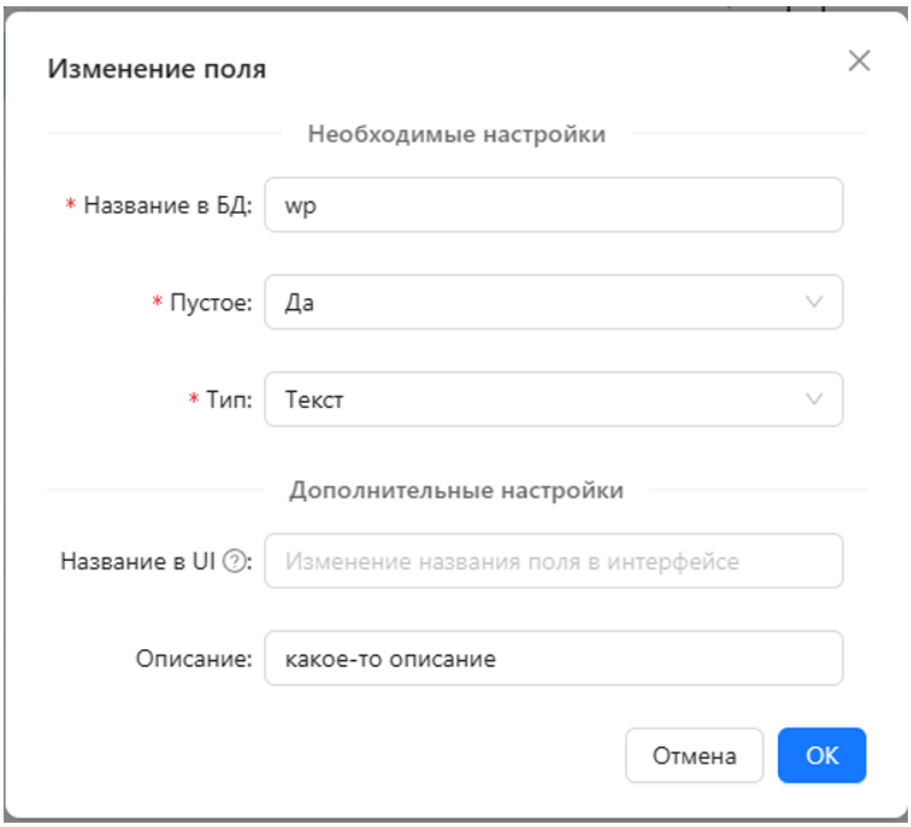


Рисунок. Диалоговое окно для внесения изменений в выбранный атрибут

Создание сущности

Создание пустой сущности в хранилище чаще всего происходит для создания новых данных (справочников), нормативно-справочной информации (НСИ), которых нет в имеющихся учетных системах. Такие сущности (справочники) обычно заполняются в ручном режиме, с клавиатуры оператором системы и используются как «центр правды» для всех остальных учетных систем.

Для пустых сущностей не нужно выбирать таблицу-источник и создавать ключи, необходимо сразу нажать на кнопку Add manual (Добавить ручной) в результате на экране появится окно Creating attribute (Добавление поля) в котором ввести имя создаваемого атрибута, выбрать свойство Nullable (Обязательный), выбрать тип данных и, если доступно, указать свойства выбранного типа данных.

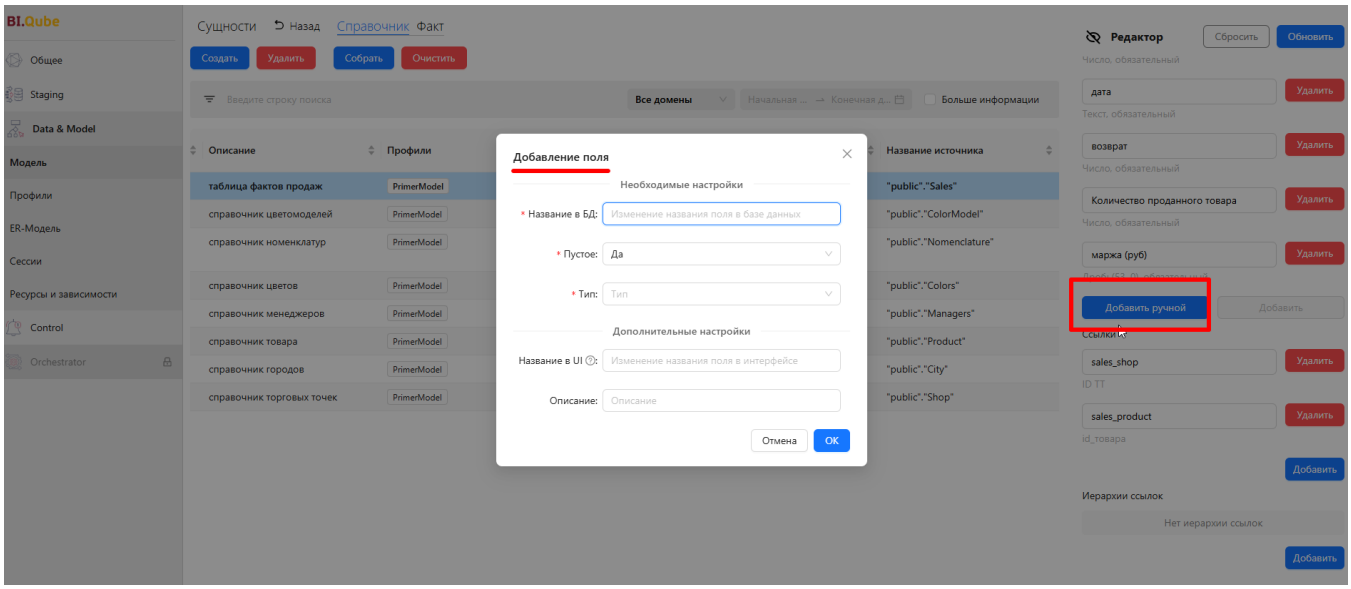


Рисунок. Создание атрибутов сущности

Система поддерживает достаточно разнообразный набор типов данных, который зависит от месторасположения хранилища (PostgreSQL, MS SQL).

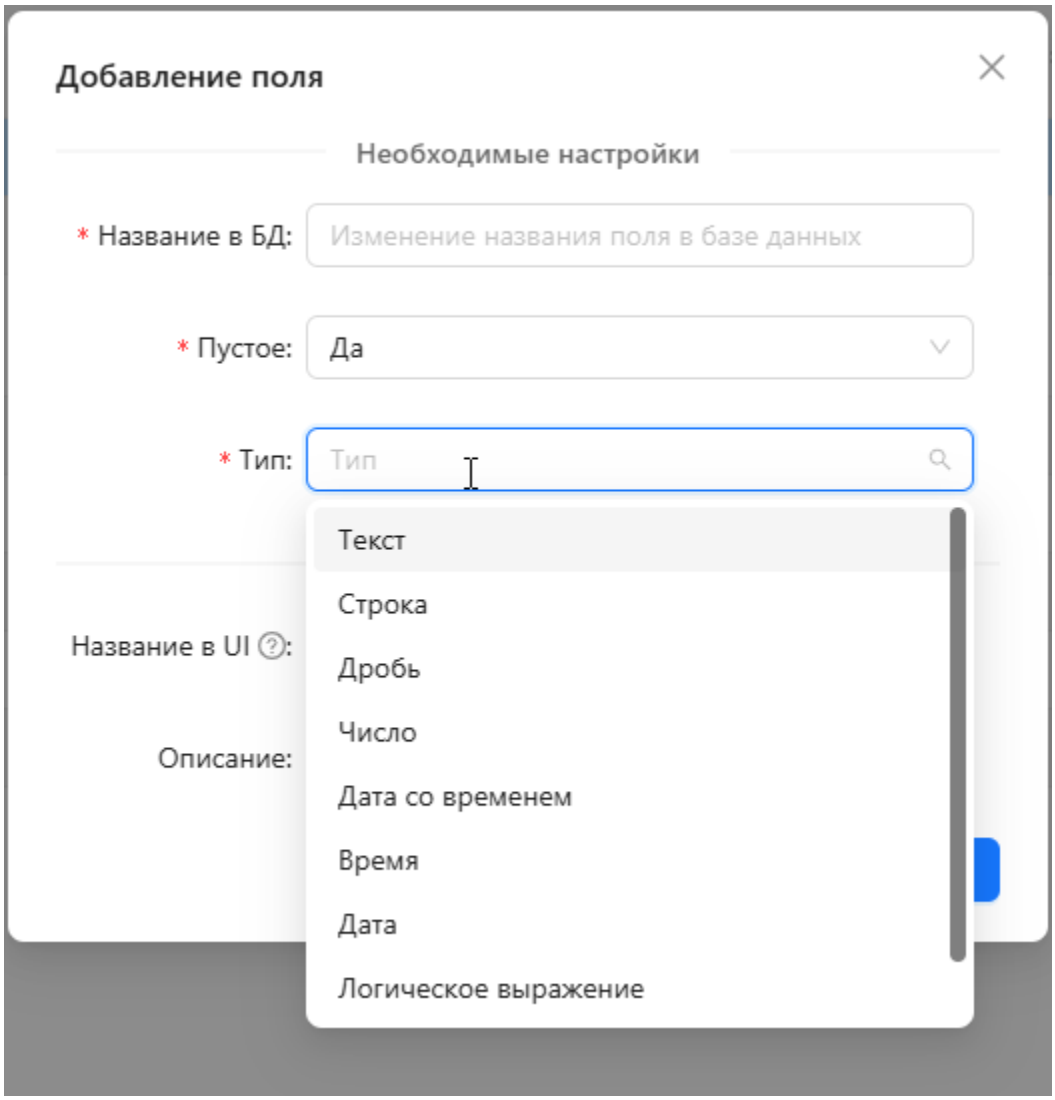


Рисунок. Доступные типы данных

Создание сущности на основе источника данных в БД

Для создания сущности, которая может быть заполнена данными из источника данных необходимо указать источник данных (Таблица-источник) для это нужно нажать кнопку Set (Выбрать), появляется диалоговое окно. Данное окно настроено по умолчанию на определённую базу данных, в которой могут находиться таблицы источники данных. Вверху в выпадающем списке выбрать схемы данных базы данных, после чего указать таблицу, данные из которой будут загружаться в создаваемую сущность. После сделанных настроек нажать кнопку Set (Выбрать).

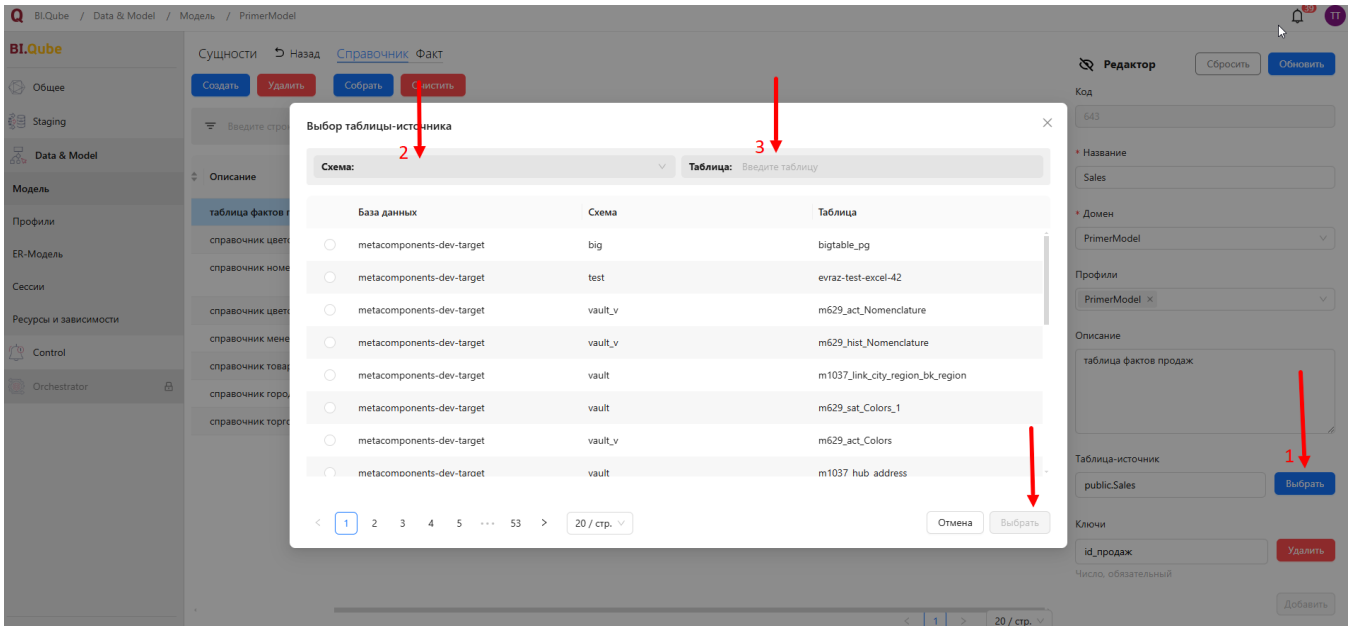


Рисунок. Описание заполнения сущности

После указания таблицы-источника появляется возможность создать ключевые поля сущности. Для этого для поля Keys (Ключи) следует нажать кнопку Add (Добавить) и в появившемся диалоговом окне выбрать тот ключ, который является уникальным для создаваемой сущности – поставить галочку напротив него и нажать на кнопку Add (Добавить).

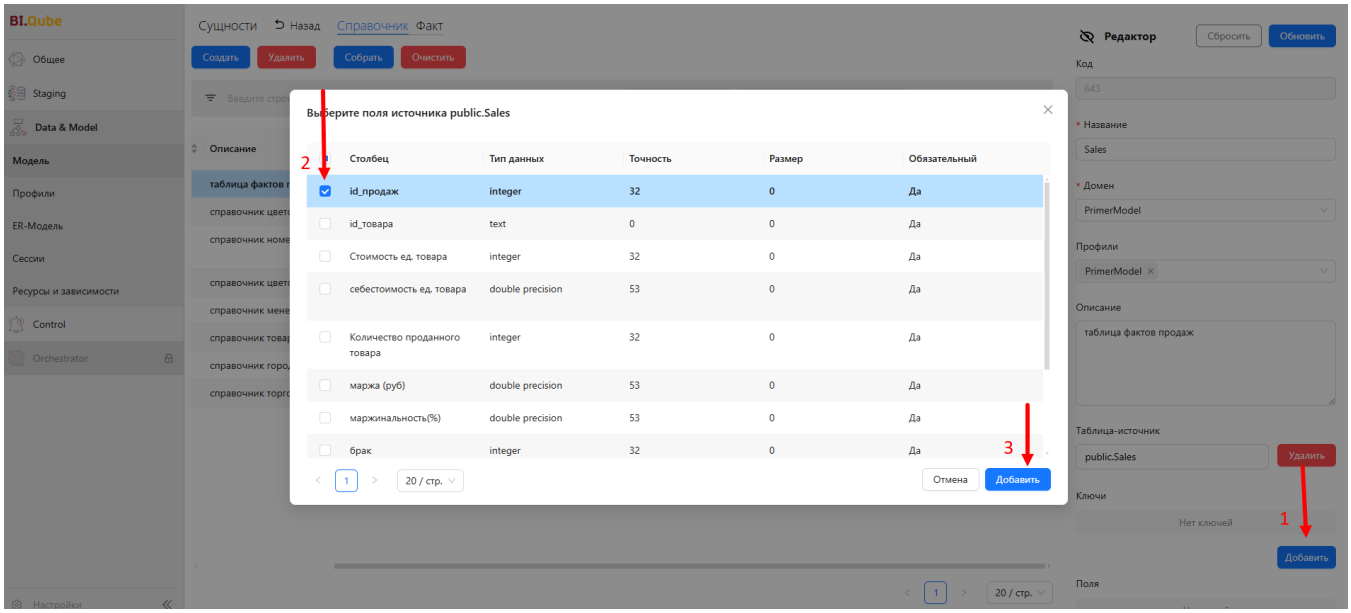


Рисунок. Заполнение поля «Ключи»

Посла создания ключа, который может быть составным, т.е. состоять более чем из одного поля, можно создать остальные поля, при этом не все поля из источника могут попасть в создаваемую сущность. Выбрать команду Add (Добавить), появится диалоговое окно Selecting source attributes (Выбор поля источника), в котором будут перечислены поля таблицы, ранее привязанной к создаваемой сущности и доступные для добавления. Затененные поля недоступны для выбора, так как они уже добавлены в качестве ключа.

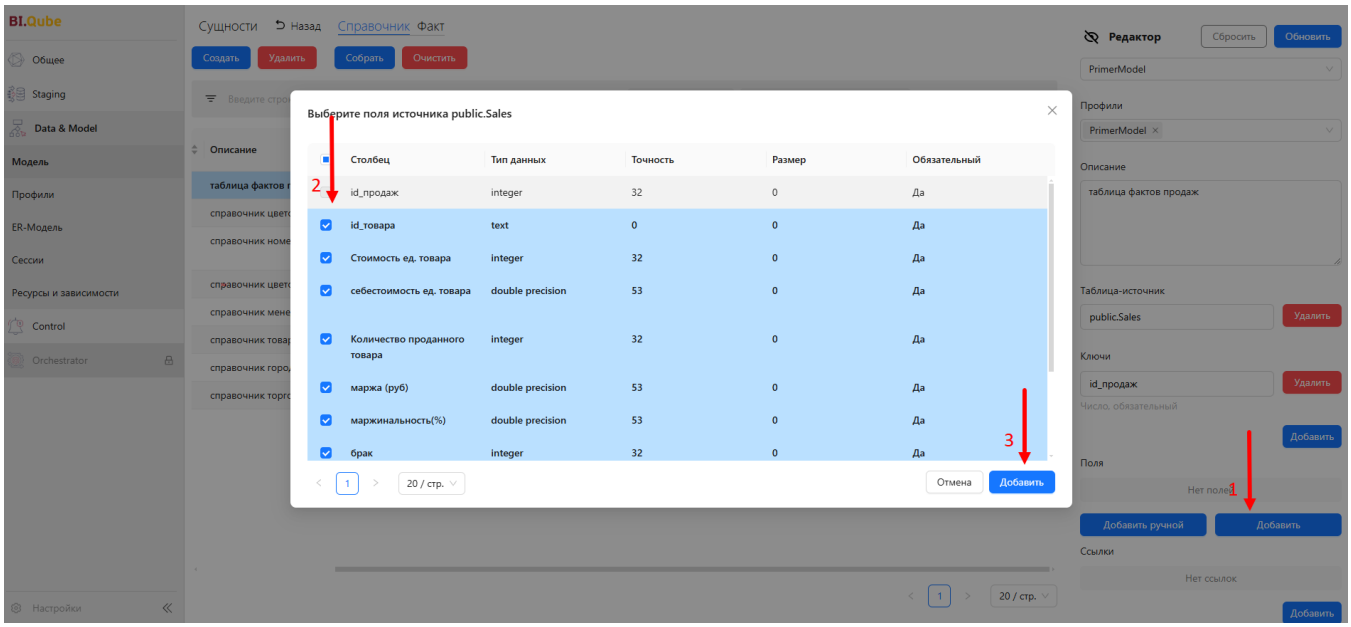


Рисунок. Поэтапное добавление полей

За один шаг можно создать сразу все нужные поля, для каждого поля в окне свойств будут созданы отдельные записи.

Просмотр и редактирование данных

Добавление и редактирование данных

Просмотр содержимого сущности – данных, осуществляется после двойного нажатия левой кнопки мыши по строке сущности в модели. Происходит проваливание в сущность (таблицу) для просмотра данных и редактирования полей в созданных атрибутах.

В этом режиме доступно построчное редактирование, при этом данные, которые попали из таблицы-источника не могут быть изменены. Редактировать или заполнять можно только те поля, которые не привязаны к таблице-источнику. При этом система сохраняет всю историю изменений, выполняемых пользователем. После заполнения полей выбранной строки в окне свойств необходимо нажать кнопку Update (Обновить).

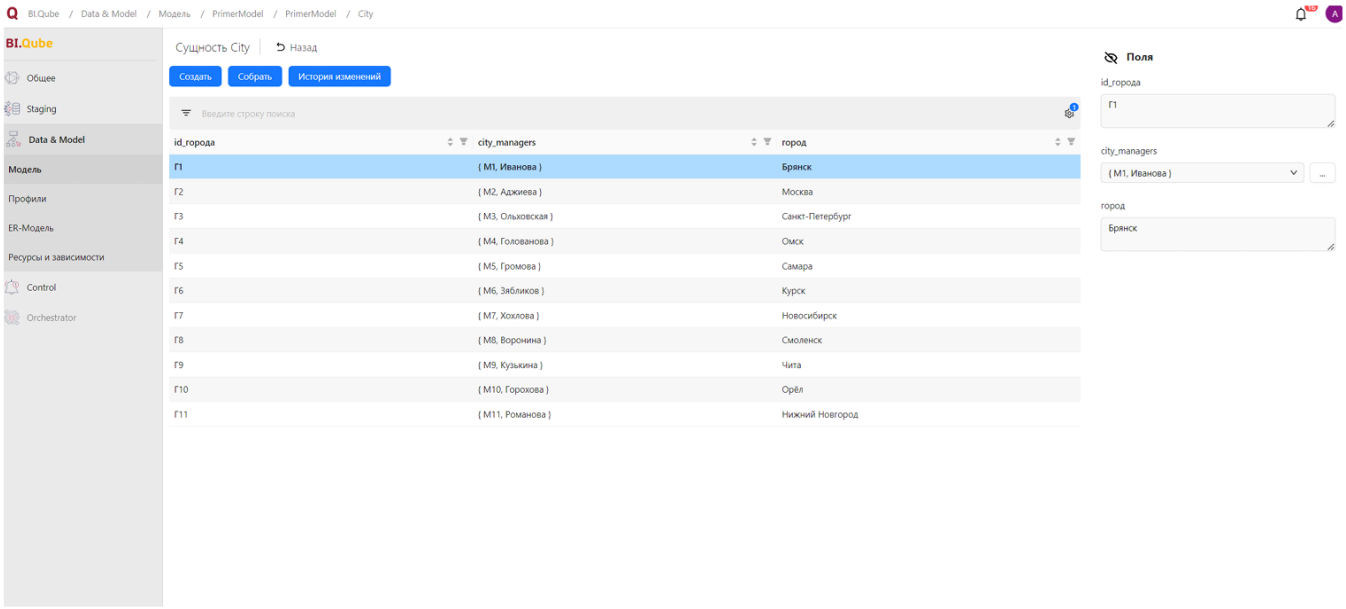


Рисунок. Заполнение добавленного поля в сущности (таблице)

Для просмотра изменений в строке необходимо нажать кнопку History changes (История изменений). Появится окно, в котором отобразятся все изменения данных этой строки, при этом каждое последующее изменение относительно предыдущего выделяется цветом.

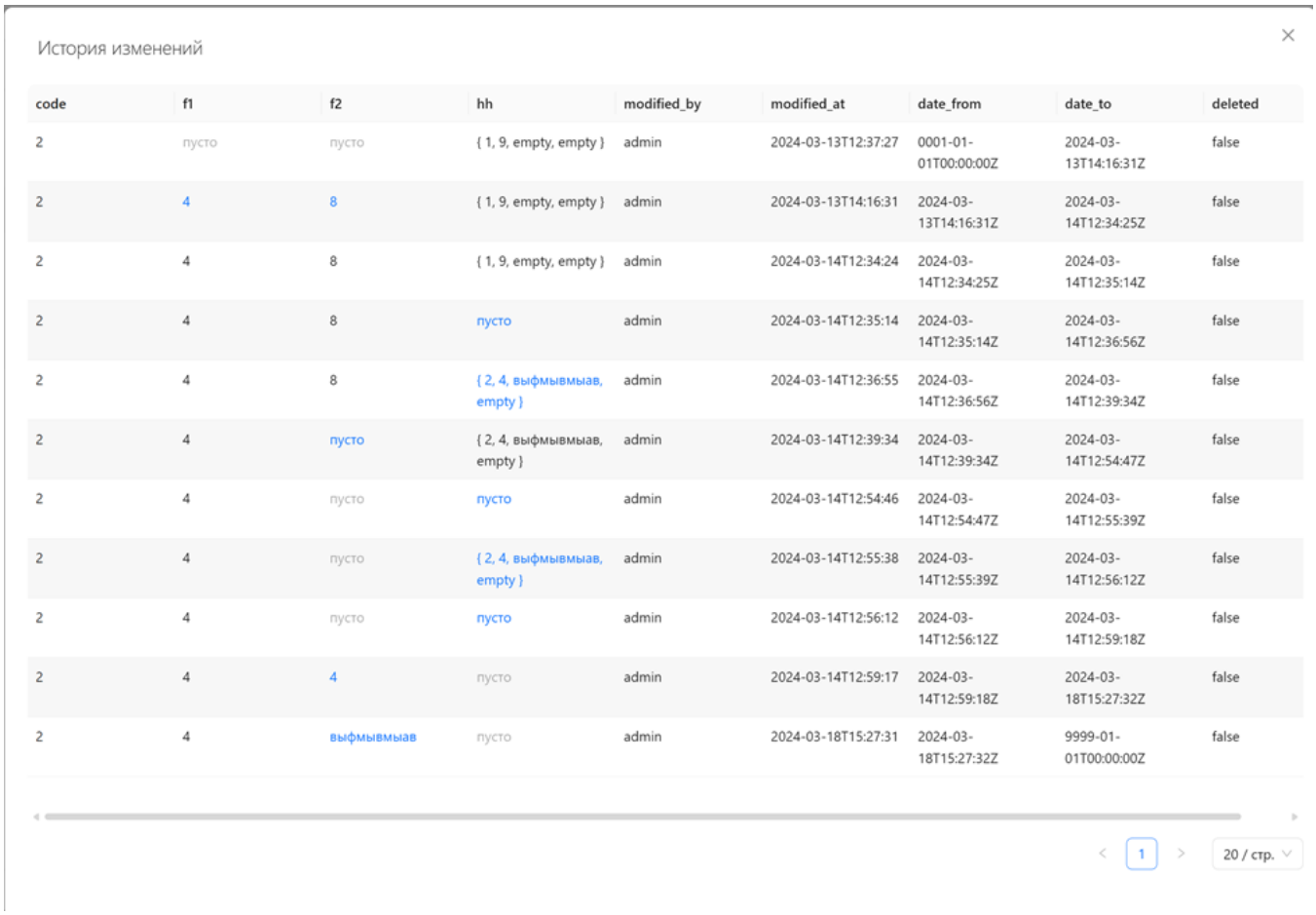


Рисунок. Просмотр истории изменения записей

Фильтрация и поиск данных

При необходимости данные могут быть отфильтрованы. Окно настройки фильтра вызывается нажатием на иконку фильтр ( ), расположенный в заголовке каждой колонки таблицы.

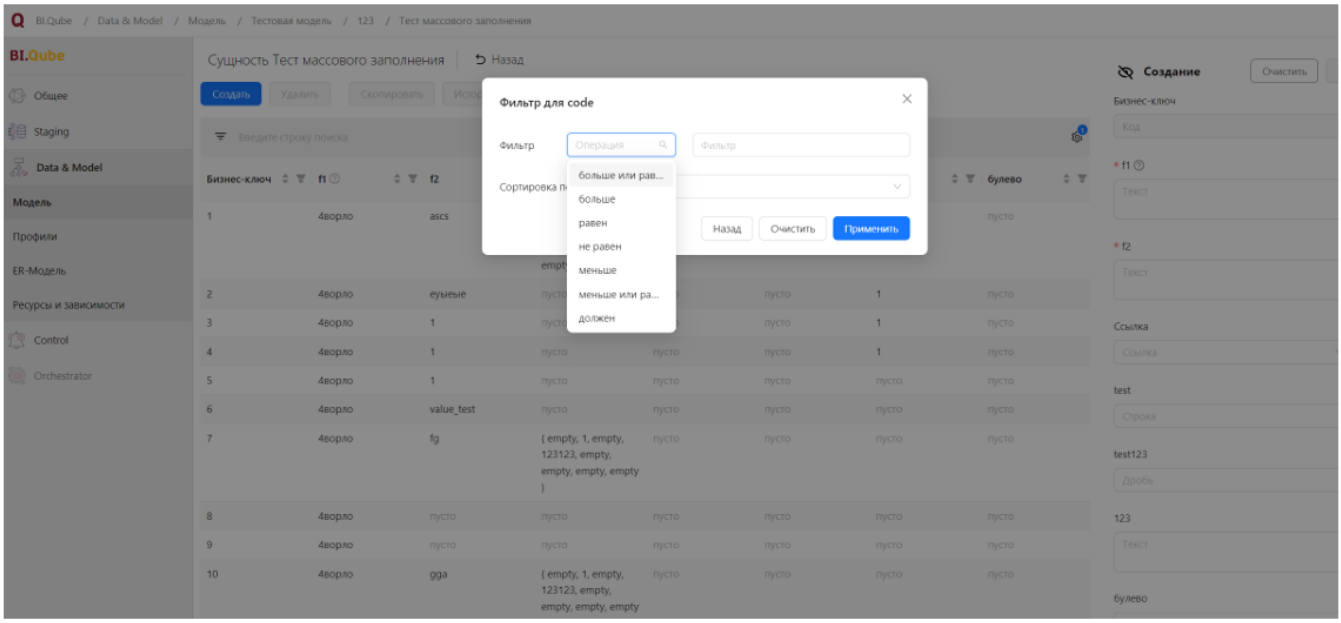


Рисунок. Окно настройки фильтра

Диалоговое окно настройки фильтра позволяет для выбранного поля Filter (Фильтр) использовать следующие операции:

- больше или равен;
- больше;
- равен;
- не равен;
- меньше;
- меньше или равен;
- содержит;
- должен.

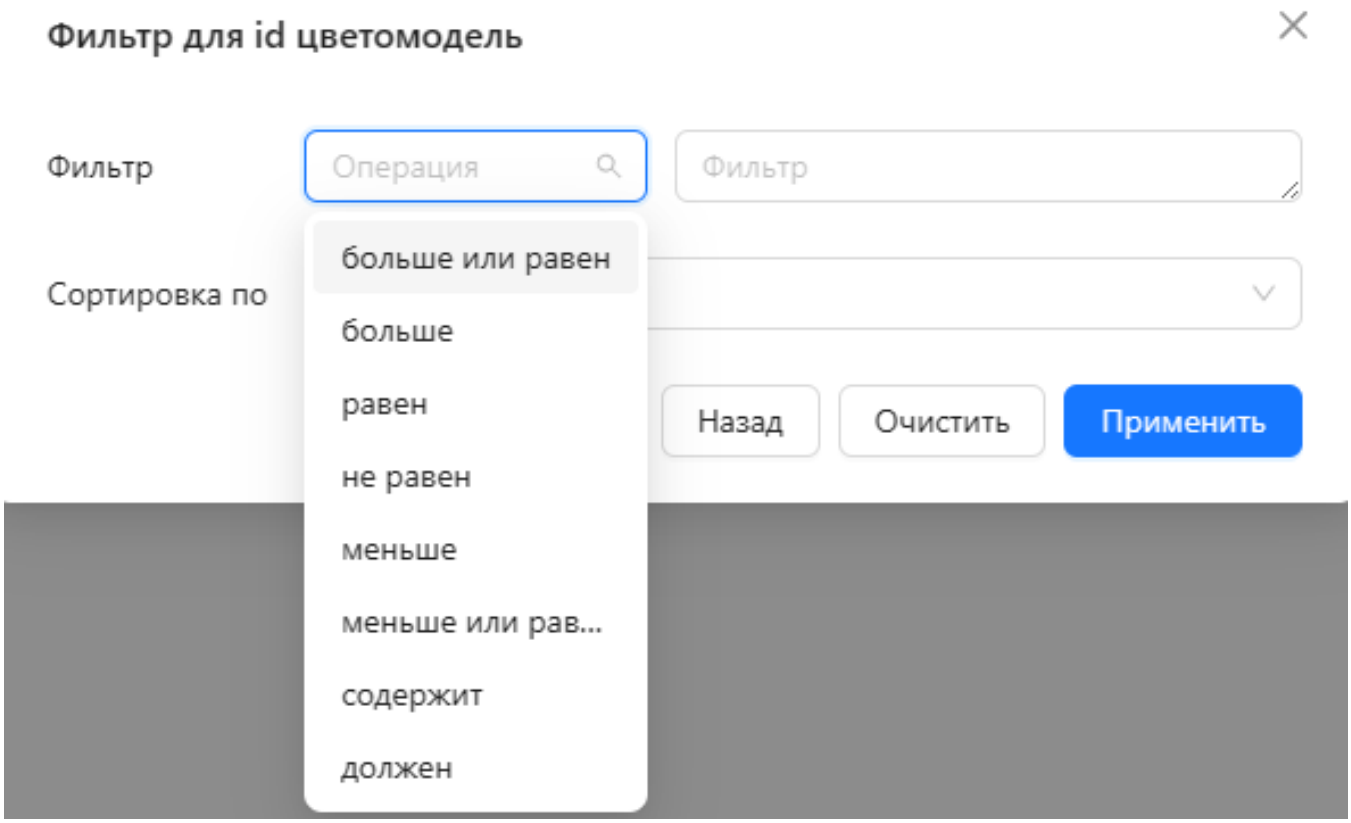


Рисунок. Выпадающий список в диалоговом окне фильтра по полю Filter (Фильтр)

Диалоговое окно настройки фильтра позволяет для выбранного поля Sorting by (Сортировка по) использовать следующие операции:

- по возрастанию;
- по убыванию.

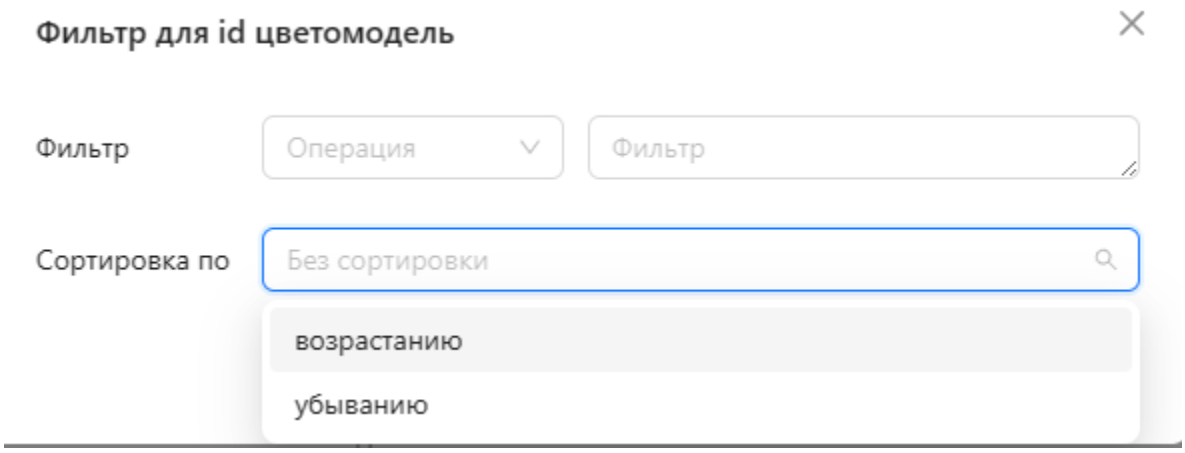


Рисунок. Выпадающий список в диалоговом окне фильтра по полю Sorting by (Сортировка по)

Важно! Поиск по словам осуществляется с учётом регистра. Перечень доступных команд в фильтре зависит от типа данных атрибута

Групповое редактирование данных в таблице

Для заполнения нескольких полей одновременно одинаковыми значениями можно выделить несколько строк в сущности (таблице). В окне свойств в нужное поле внести необходимые изменения, затем нажать кнопку Update (Обновить). И одновременно во всех выделенных столбцах появятся внесённые изменения, при этом следует помнить, что если в каких-то строках, в этом поле были данные, то они будут заменены новыми.

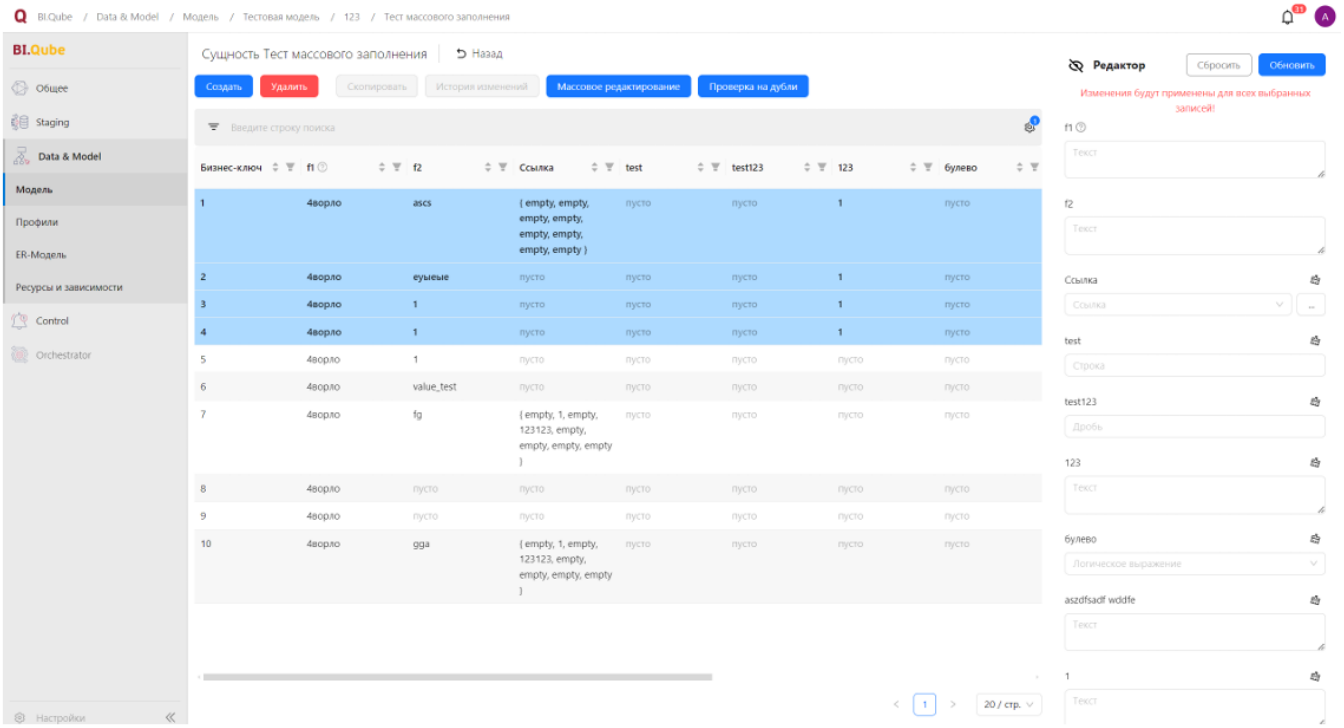


Рисунок. Редактирование нескольких записей одновременно в сущности (таблице)

При включении фильтра вносимые изменения применяются ко всей отфильтрованной выборке по установленным параметрам. После нажатия на кнопку Apply (Применить), фильтр произведёт фильтрацию всей сущности (таблицы). В окне свойств справа появится уведомление о вносимых изменениях. После нажатия на кнопку Update (Обновить) появится диалоговое окно Confirmation of the update (Подтверждение обновления), в котором указаны те записи, для которых будут произведены изменения. Для подтверждения необходимо нажать на кнопку "ОК".

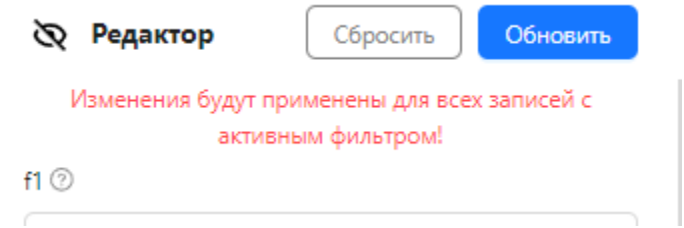


Рисунок. Предупреждение о вносимых изменениях в окне свойств

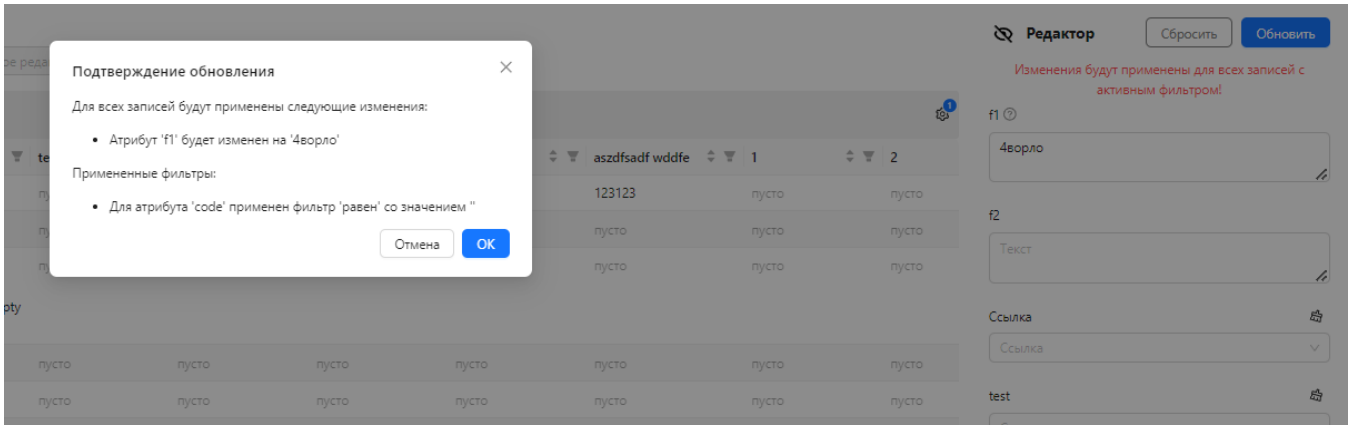
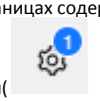


Рисунок. Диалоговое окно подтверждения обновления.

Если необходимо очистить данные в каком-то поле сразу для нескольких строк, то можно выделить эти строки или воспользоваться фильтром, затем в окне свойств, в интересующем поле нажать на иконку "кисточка" (), после чего нажать кнопку Update (Обновить). Данные в этом поле для всех выделенных строк будут удалены.

Настройка режимов отображения данных

на всех страницах содержащих данные доступны различные настройки с режимами отображения данных, найти их можно нажав на кнопку в форме шестерёнки(). Если таблица очень большая то на экране можно настроить фиксацию нескольких колонок с левой стороын, при горизонтальной прокрутке они не будут скрываться. Возможно отключить отображение не нужных в данный момент столбцов.

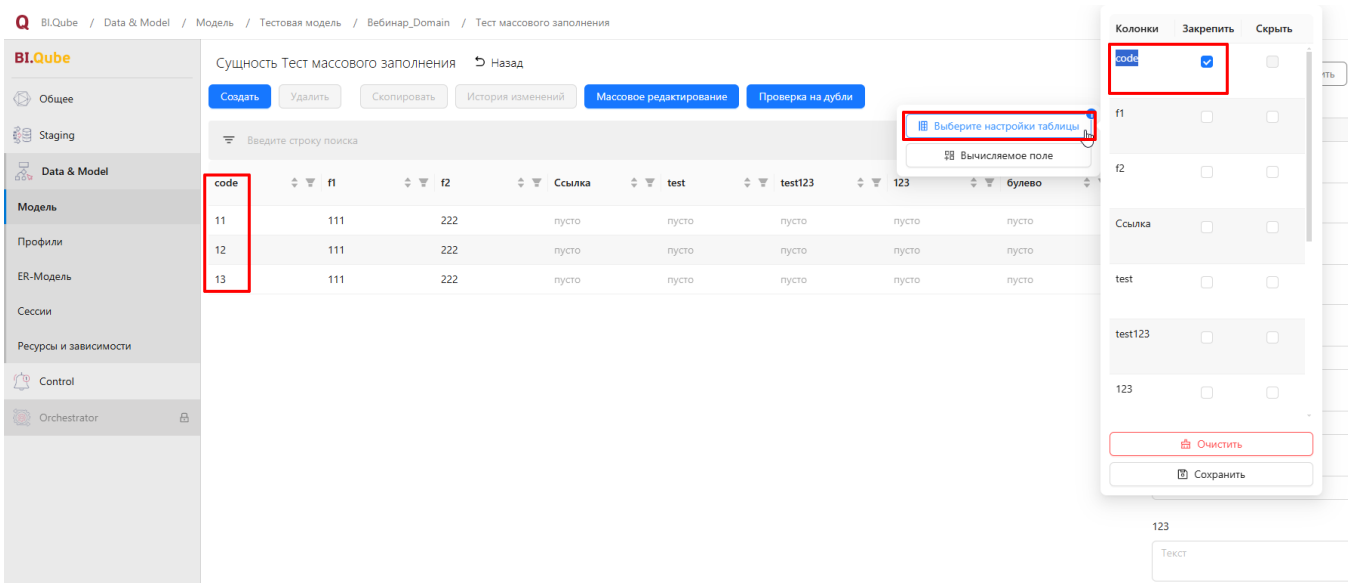
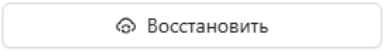


Рисунок. Выбор зафиксированных колонок

При выборе зафиксированных колонок, данные колонки также фиксируются в окне свойств справа. С помощью кнопки Восстановить(

) можно сбросить выполненные настройки.

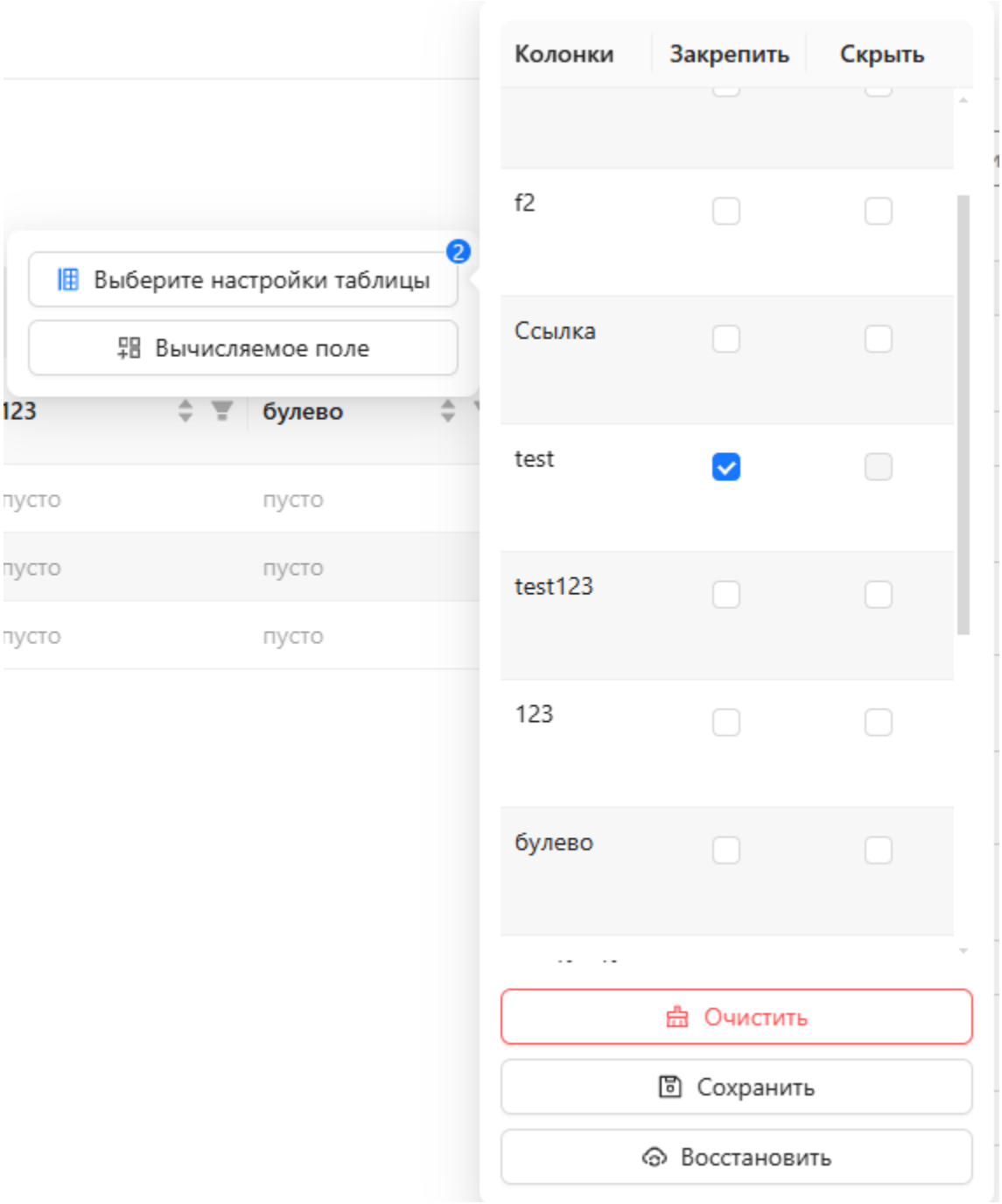


Рисунок. Зафиксированные колонки

Работа со списками

В визуальных элементах типа выпадающий список может содержаться огромное количество данных. Чтобы предотвратить замедление в работе, данные в списках подгружаются частями. С помощью прокрутки можно выполнять просмотр всех данных. Если известно, что требуется найти можно начать писать

искомый текст. Кроме этого для отбора данных в списке доступен расширенный режим поиска нужных данных, для этого нужно нажать на иконку (...) справа от поля, открывается диалоговое поле, в котором можно осуществлять поиск, фильтрацию и сортировку данных, которая при нажатии на кнопку "ОК" будет применена ко всей сущности и её отображению в исходном выпадающем списке.

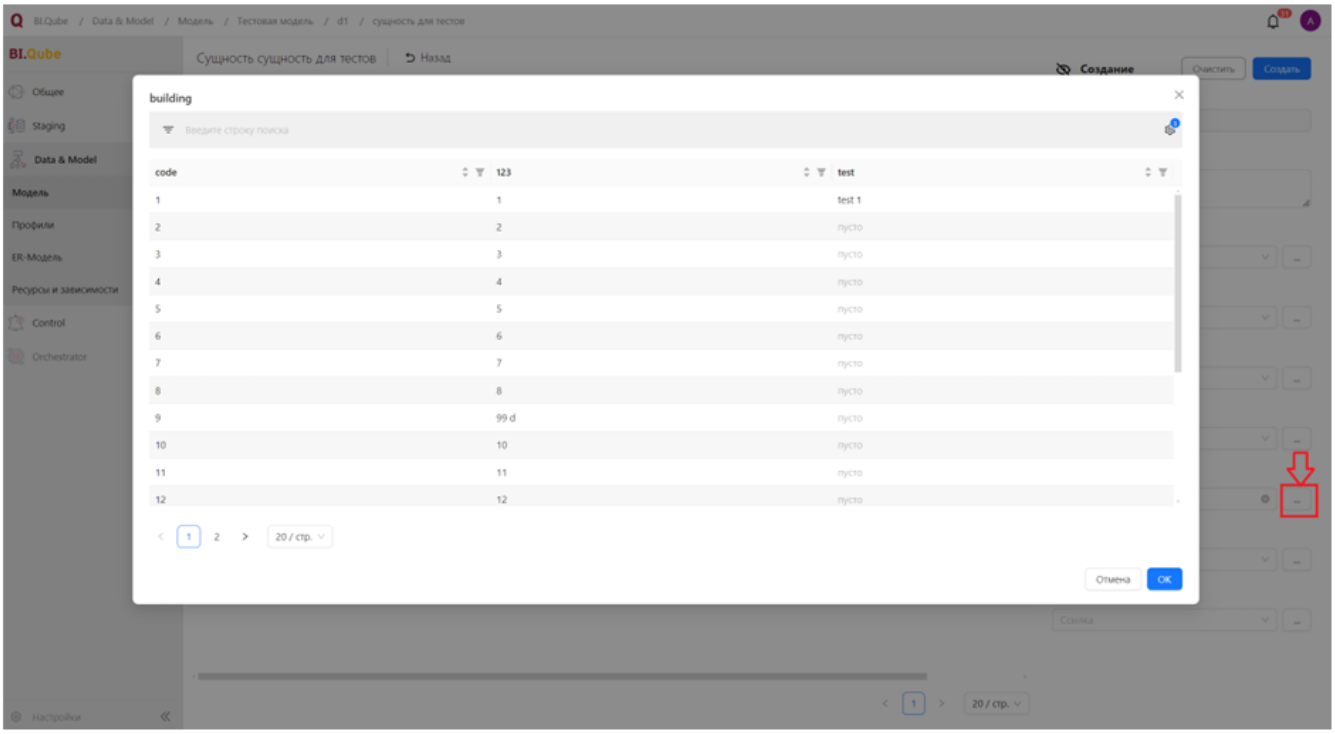


Рисунок. Поиск и фильтрация в связанной таблице

Также есть способ поиска и фильтрации данных в самом поле в окне свойств справа, выбрав нужное из выпадающего списка, либо через ввод данных в само поле.

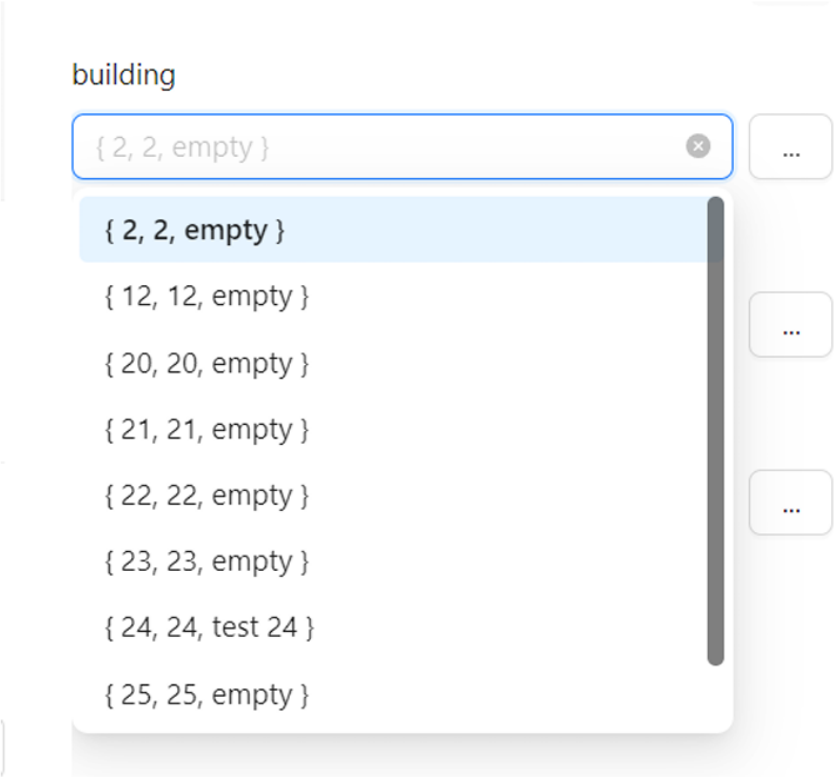
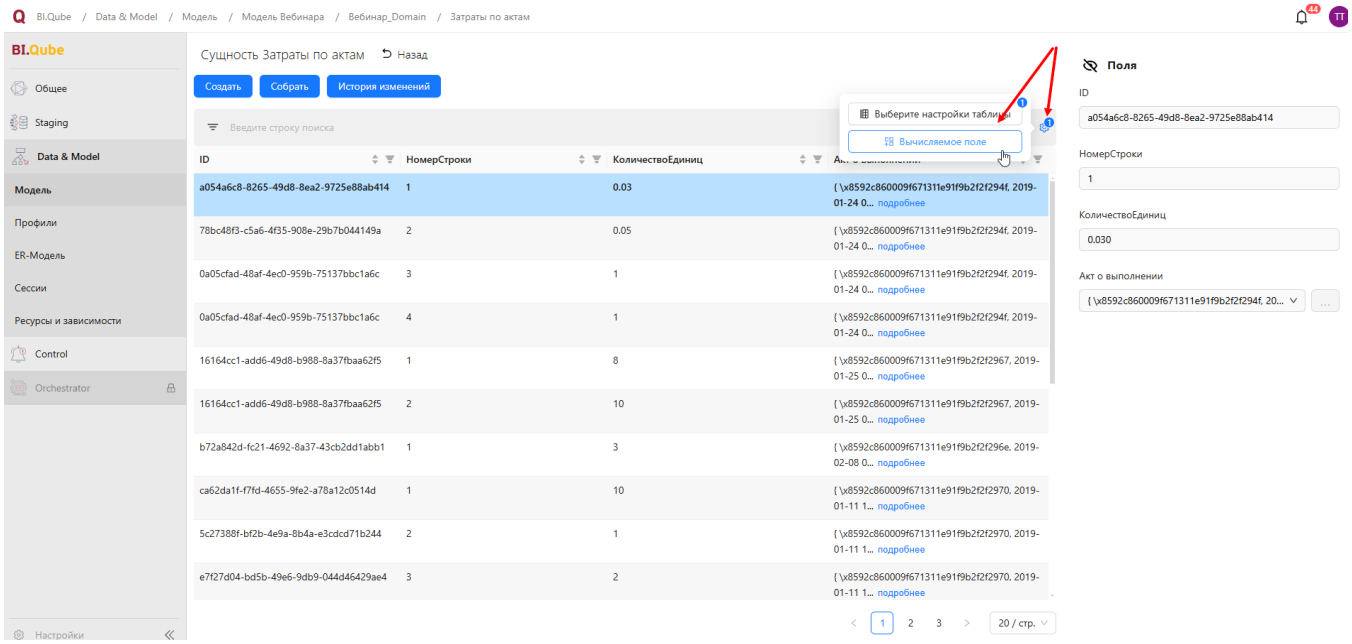


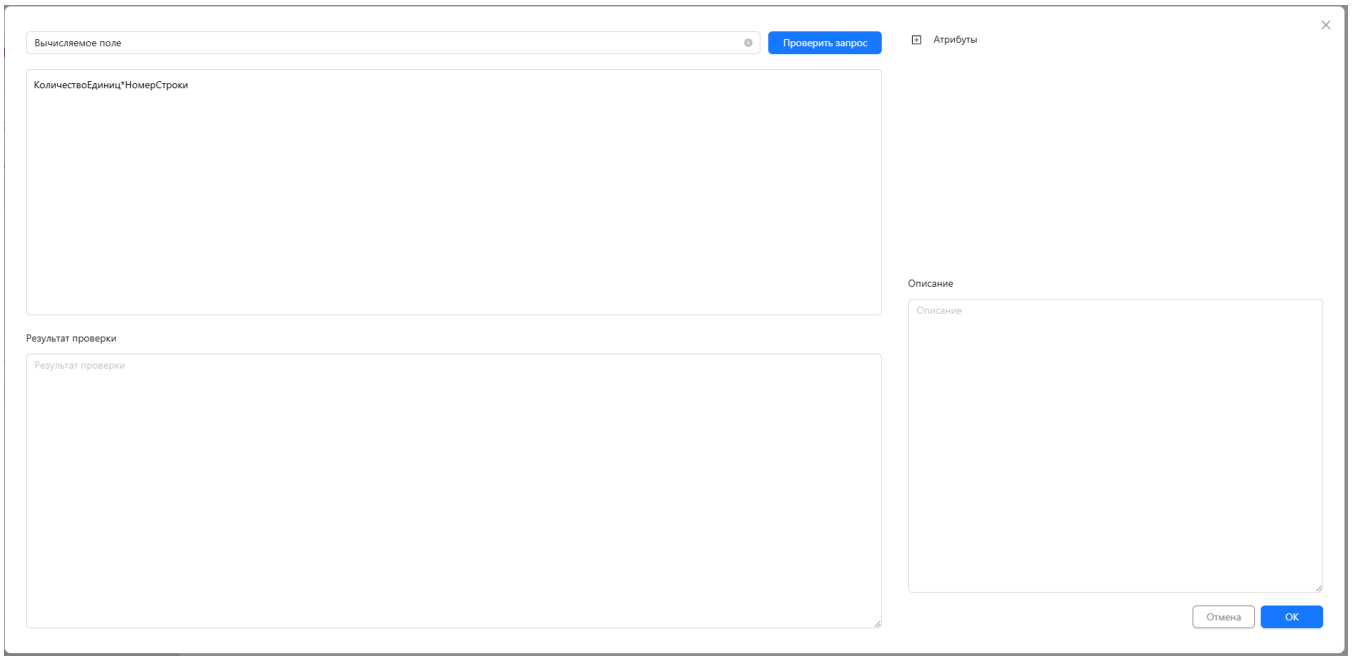
Рисунок. Поиск данных в поле окна свойств (один из вариантов пагинации)

Простейшие вычисления в справочниках

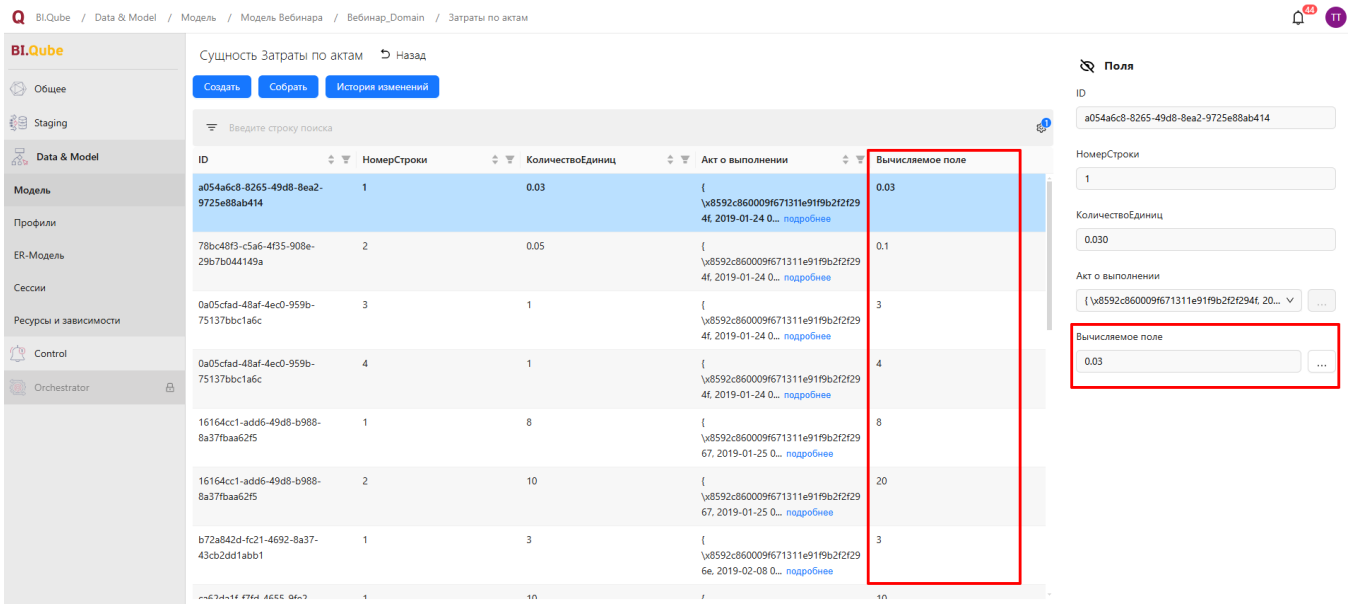
В процессе работы со справочниками возможно создавать новые атрибуты (поля) с использованием простых вычислений, на основе простых выражений поддерживаемых в языке SQL. Для этого в настройках (кнопка с изображением шестеренки) следует выбрать соответствующую команду - "Вычисляемое поле".



Появится диалоговое окно в котором необходимо указать имя нового поля, при необходимости дать краткое описание и написать вычисляемое выражение.



После ввода выражения, можно выполнить проверку. Результат проверки будет выведен в поле рядом, список атрибутов текущего справочника, которые могут участвовать в выражении, также виден пользователю. Атрибуты других справочников при создании вычисляемого поля недоступны. После нажатия на кнопку "Ок" в справочник добавляется вычисляемое поле и автоматически заполняется данными.



Выражение в любой момент может быть изменено, история редактирования не сохраняется.

Создание связей между сущностями

Создание связей

Для создания связей («линков») между сущностями необходимо зайти в свойства сущности, к которой будут привязываться другие сущности.

В зоне Links (Связи) необходимо выбрать команду Add (Добавить) и заполнить появившееся диалоговое окно.

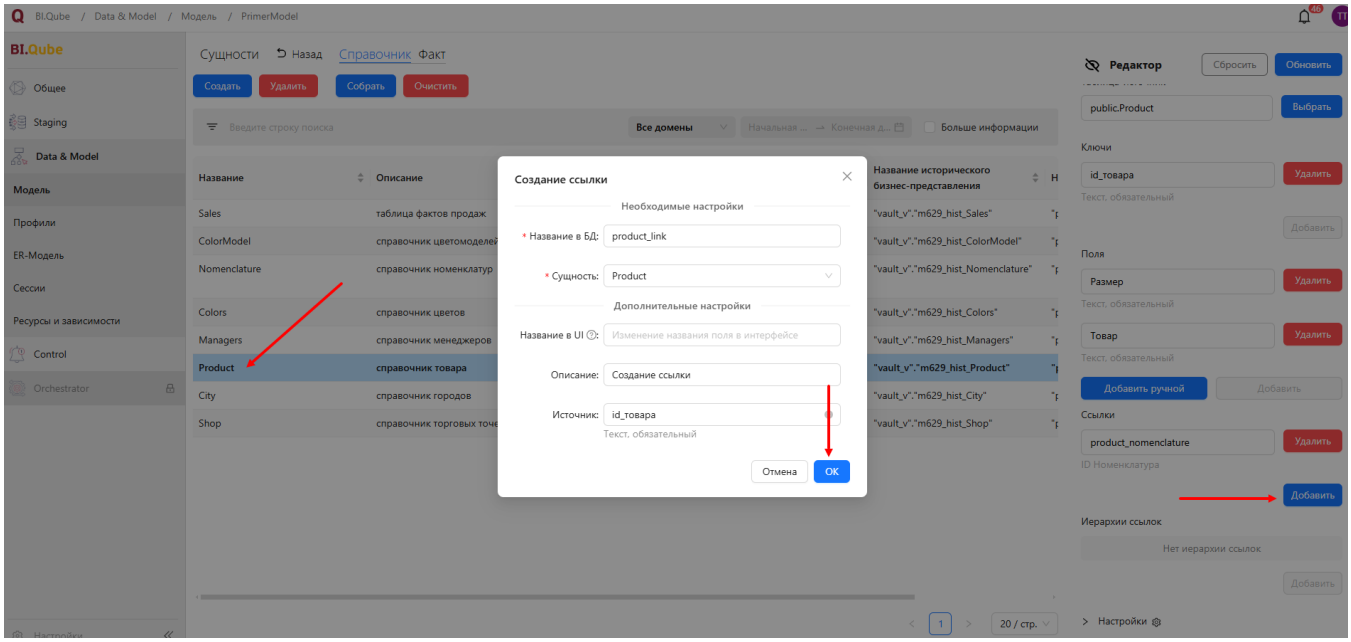


Рисунок. Выбор сущности для создания связи и диалоговое окно создания линка (связи)

Следует указать имя связи, выбрать связываемый объект, и выбрать атрибут текущей сущности, к которому привязываются данные сторонней сущности. Следует отметить, что связываемая сущность (таблица) связывается с текущей по бизнес ключу об этом, следует помнить при создании бизнес-ключей. В поле "Сущность" можно выбирать нужное из выпадающего списка, либо не выбирать ничего, и оно будет заполнено автоматически. Далее необходимо перейти в настройки отображения линка.

Для просмотра дополнительных настроек для линка необходимо нажать на иконку в форме шестерёнки. Появятся дополнительные поля:

- Активное представление представлено в выпадающем списке двумя значениями: true/false;
- Историчное представление представлено в выпадающем списке двумя значениями: true/false;
- Тип суррогатного ключа;
- Атрибуты в связанных сущностях можно выбирать по своему усмотрению из выпадающего списка.

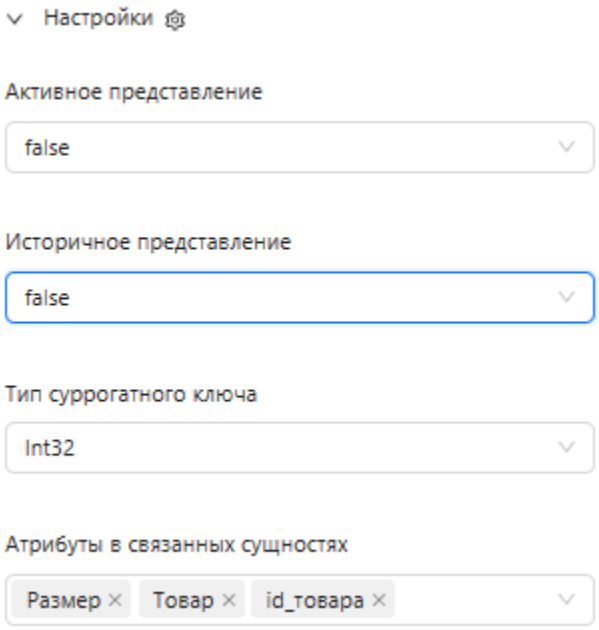


Рисунок. Выбрана все атрибуты

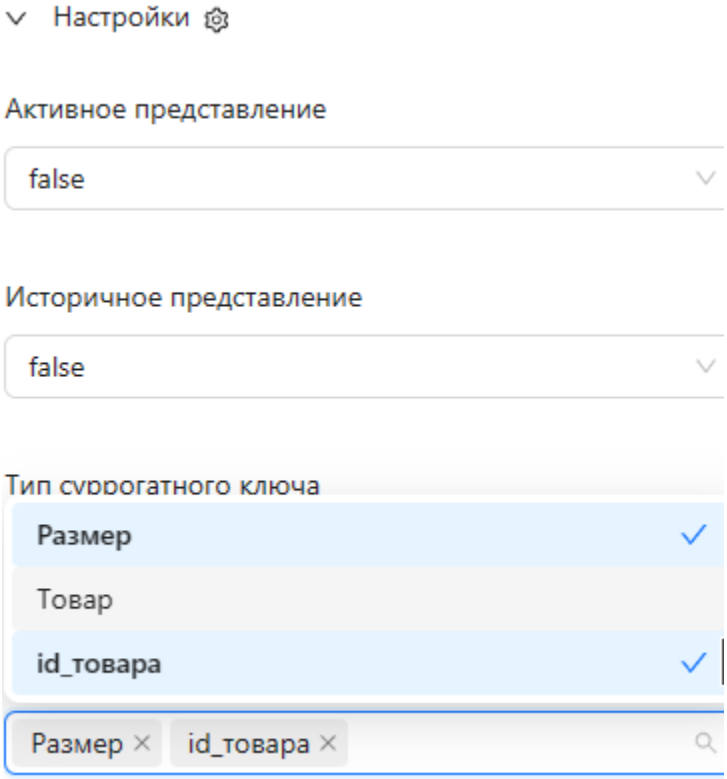


Рисунок. Выбрана только часть атрибутов

Создание иерархий

Если в модели данных справочники связаны между собой, организуя при этом логические иерархии, то линки можно настроить так, чтобы поддерживать работу таких иерархий.

Например, если в модели есть справочники: город, регион, улица, и эти справочники связанны данными, а между ними настроены связи в модели, то для нового справочника адрес между справочниками, образующими запись адреса, можно настроить иерархическую связь. При этом глубина иерархий ничем не ограничена.

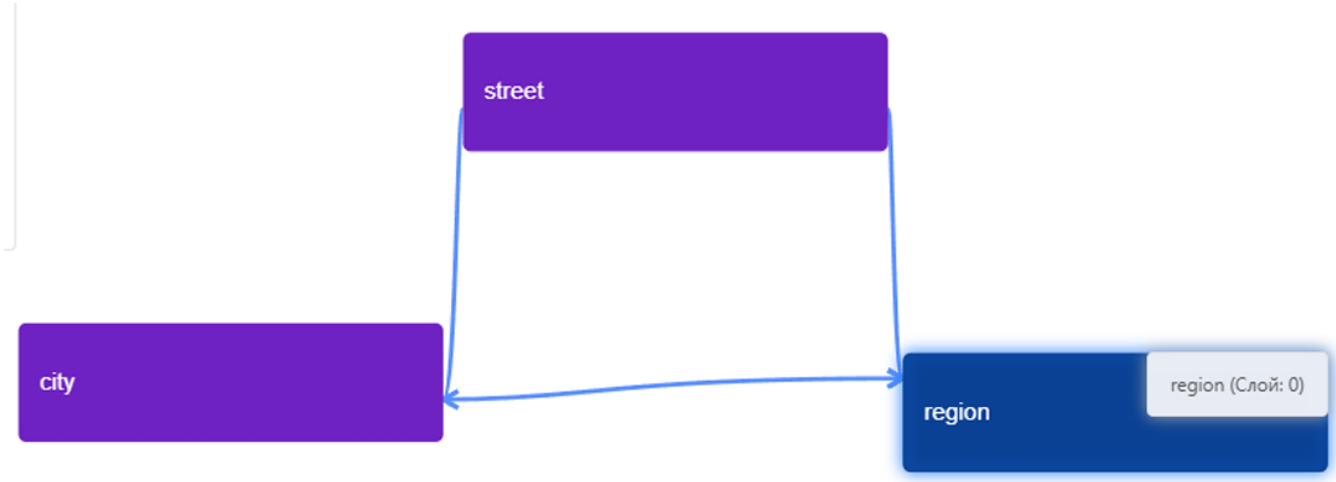


Рисунок. Модель связей между справочниками (связь между city и street. В данном примере связь избыточна и может отсутствовать)

Для создания в таблице Адрес иерархии адреса (при выборе региона, автоматически фильтруются города выбранного региона; при выборе города, автоматически фильтруются улицы для выбранного города) необходимо в таблица Адрес добавить линки на три справочника: города, регионы, улицы.

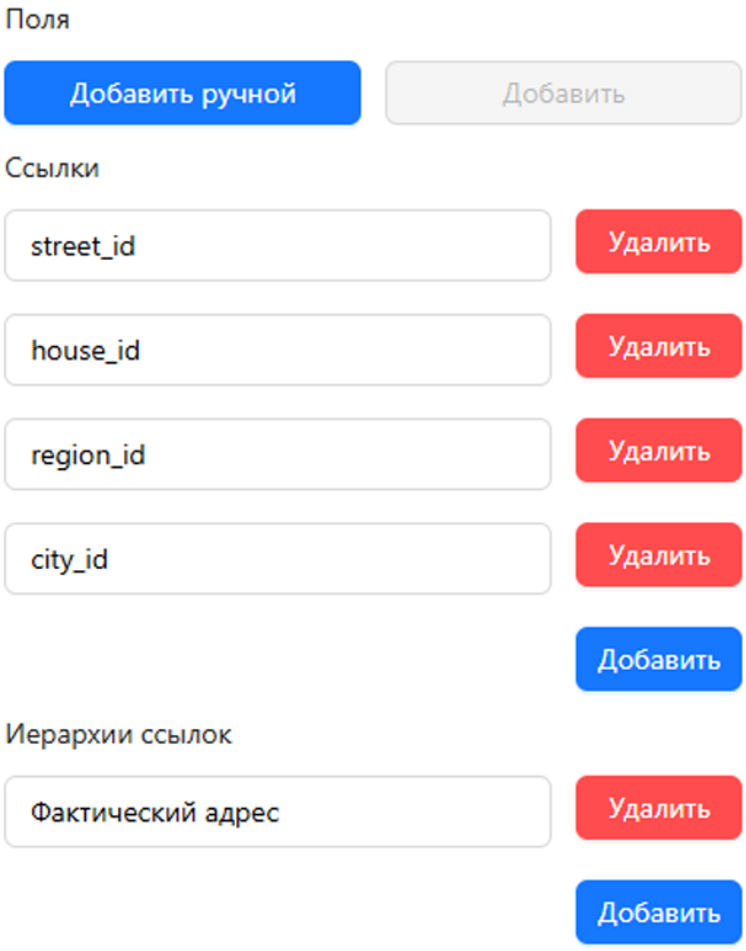


Рисунок. Созданы связи со справочниками (улица, дом, регион, город)

Чтобы эти связи организовать в иерархии нужно в разделе Иерархии ссылок нажать кнопку "Добавить" и в появившемся окне настроить правила формирования иерархий.

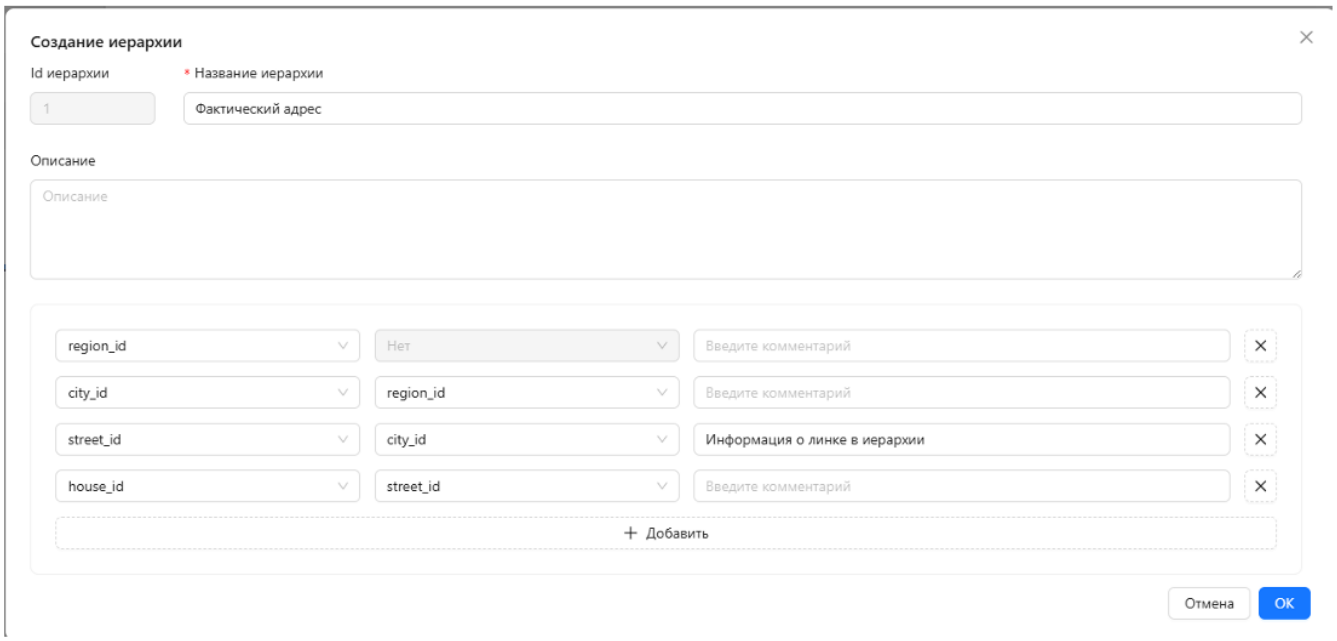


Рисунок. Окно настройки правил формирования иерархий

В первую очередь необходимо указать "Имя иерархии" (используется для отображения группы визуальных элементов в интерфейсе пользователя), добавить при необходимости краткое описание назначения иерархии, затем в первом выпадающем списке выбрать родителя, затем добавить наследника и для наследника явно указать родителя и так далее (ограничений по глубине иерархии нет). Для каждого уровня можно добавлять комментарий. Таким образом можно создавать достаточно сложные зависимости. В визуальном интерфейсе выглядит это следующим образом.

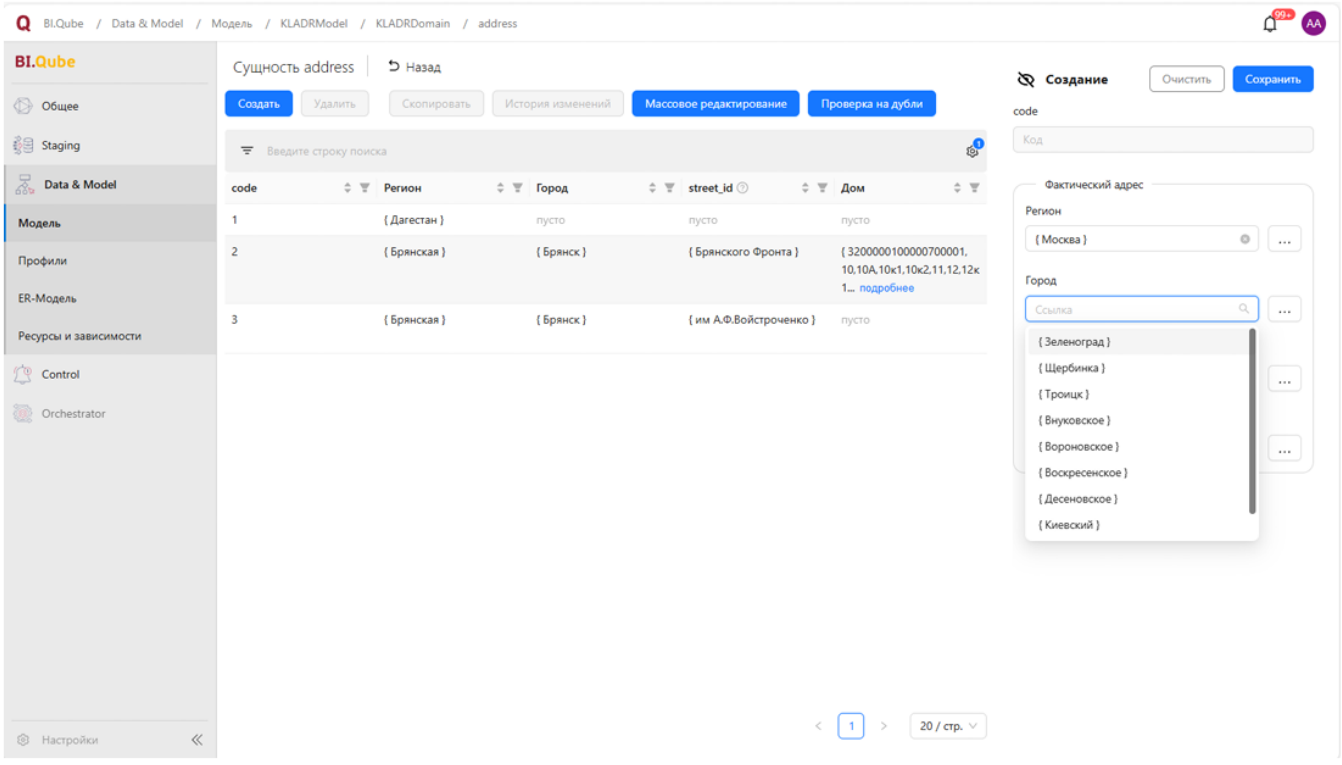


Рисунок. Отображение иерархий в визуальном интерфейсе

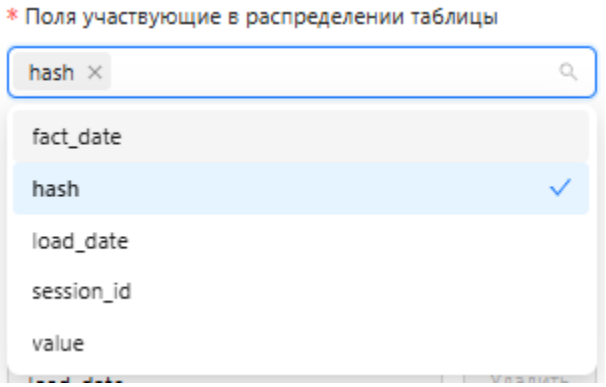
Визуальные иерархии объединены в один блок. Если какие-то связи нужно использовать во второй и/или последующих элементах, нужно добавить еще одну связь (линк) на справочник и использовать в новой иерархии. Другими словами, один справочник может использоваться только в одной иерархии.

Создание сущности "Факт" в модели

Создание сущности "Факт"

Сущность "Факт" служит для преобразования типов данных таблицы. Создание факта происходит стандартным образом. Необходимо, находясь в модели, нажать на кнопку Create (Создать). Справа в окне свойств появится перечень свойств которые нужно заполнить:

- Name (Название) – имя сущности;
- Description (Описание) – бизнес описание назначения сущности;
- Domen (Домен) – выбрать домен, в который будет входить сущность. Одна сущность может принадлежать только одному домену;
- Profile (Профиль) – выбрать профиль, к которому будет принадлежать сущность. Сущность может принадлежать нескольким профилям;
- Table Schema (Схема таблицы) – схема, в которой будет храниться таблица с результатами преобразований. Выбор схемы осуществляется с помощью выпадающего списка.
- Table name (Название таблицы) – название таблицы, в которой будут храниться выполненные преобразования.
- Fields involved in table distribution hash (Поля участвующие в распределении таблицы)



- Table attributes (Атрибуты таблицы) – имя полей записи;
- Sectionalization period (Период секционирования) – промежуток времени, в который будут созданы секции помесячно (заполняются поля От и До, месяц и год)
- Table parameters (Параметры таблицы)
- Table variables (Переменные таблицы)
- Business logic of the procedure (Бизнес логика процедуры) – поле для создания скрипта, преобразующего типы данных полей таблицы.

Также есть две кнопки:

- Copy the table creation code (Скопировать код создания таблицы) – копирует в буфер обмена код создания таблицы.
- Copy the table filling script (Скопировать скрипт заполнения таблицы) – копирует в буфер обмена скрипт заполнения таблицы.

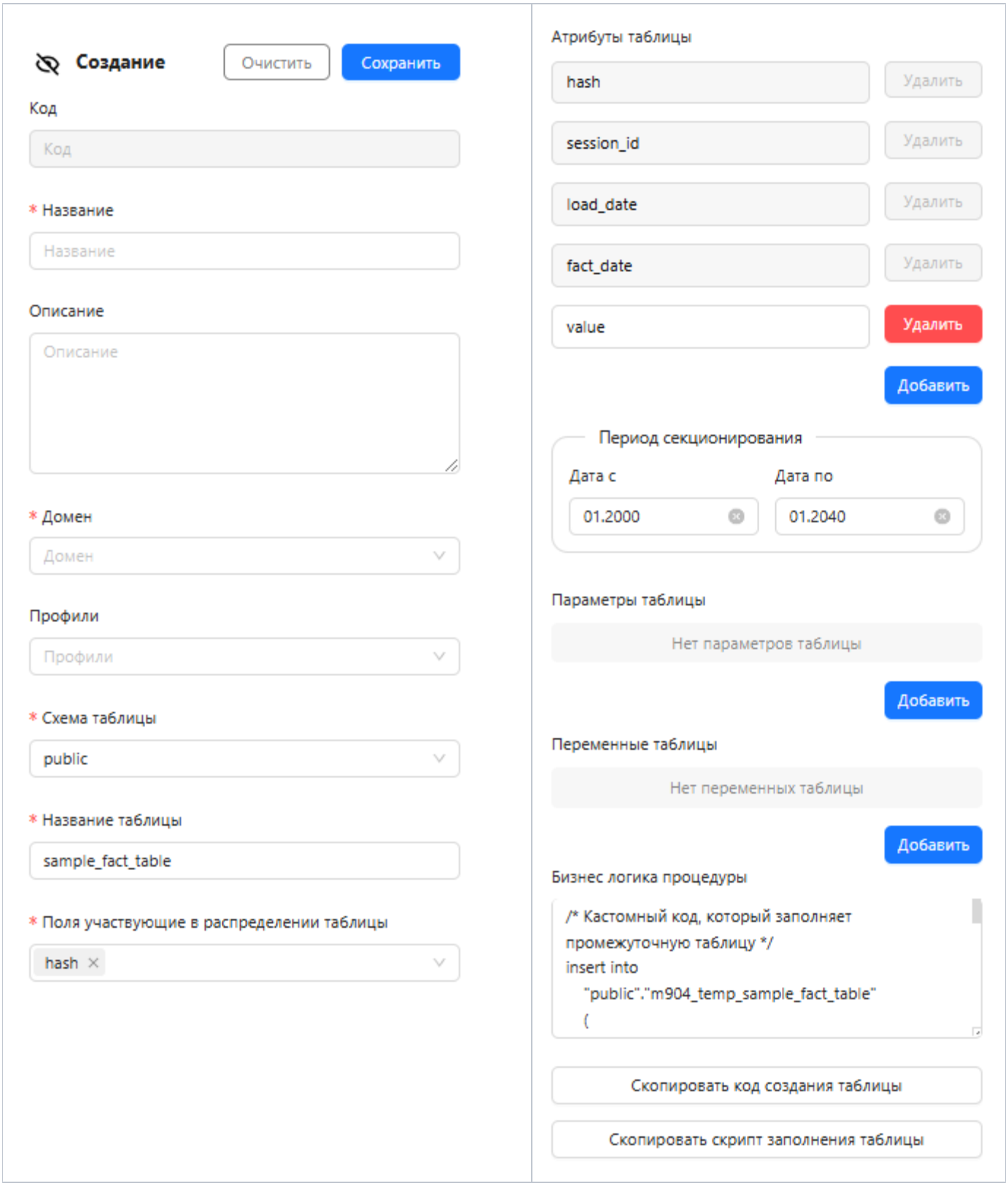


Рисунок. Свойства сущности "Факт"

Атрибуты таблицы представляют собой заголовки таблицы. Добавленные по умолчанию атрибуты нельзя изменить или удалить. Можно добавить атрибуты, требующие преобразования, с помощью кнопки Добавить (Add).

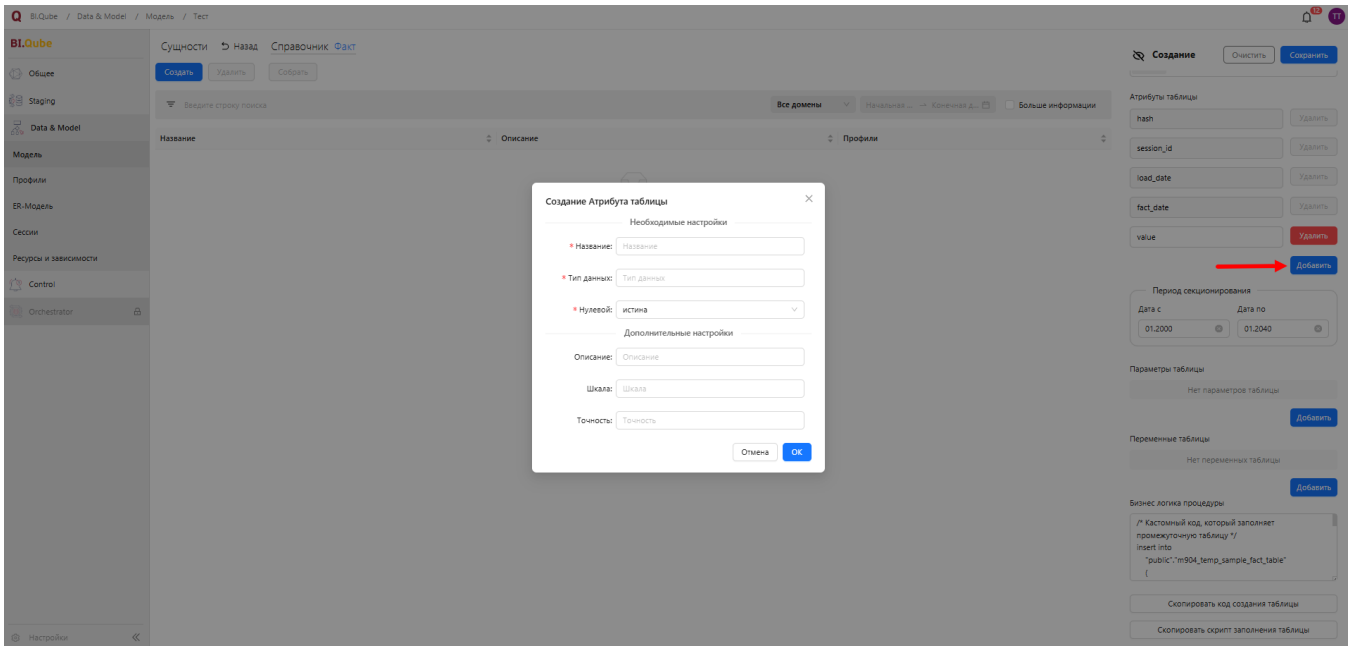


Рисунок. Модальное окно создания атрибута таблицы

При клике на кнопку Добавить (Add) откроется модальное окно со следующими полями для заполнения:

- Название – название атрибута;
- Типы данных – тип данных атрибута, к которому необходимо привести;
- Нулевой – представлен с помощью выпадающего списка: истина, ложь;
- Описание – короткое описание.

Сборка сущности

После создания сущности необходимо её собрать. Операция сборки запускает ряд внутренних процедур, связанных с формированием большого количества программного кода, представления сущности в модель DataVault. Так же для сущностей, созданных на основе таблицы-источника происходит загрузка данных из источника.

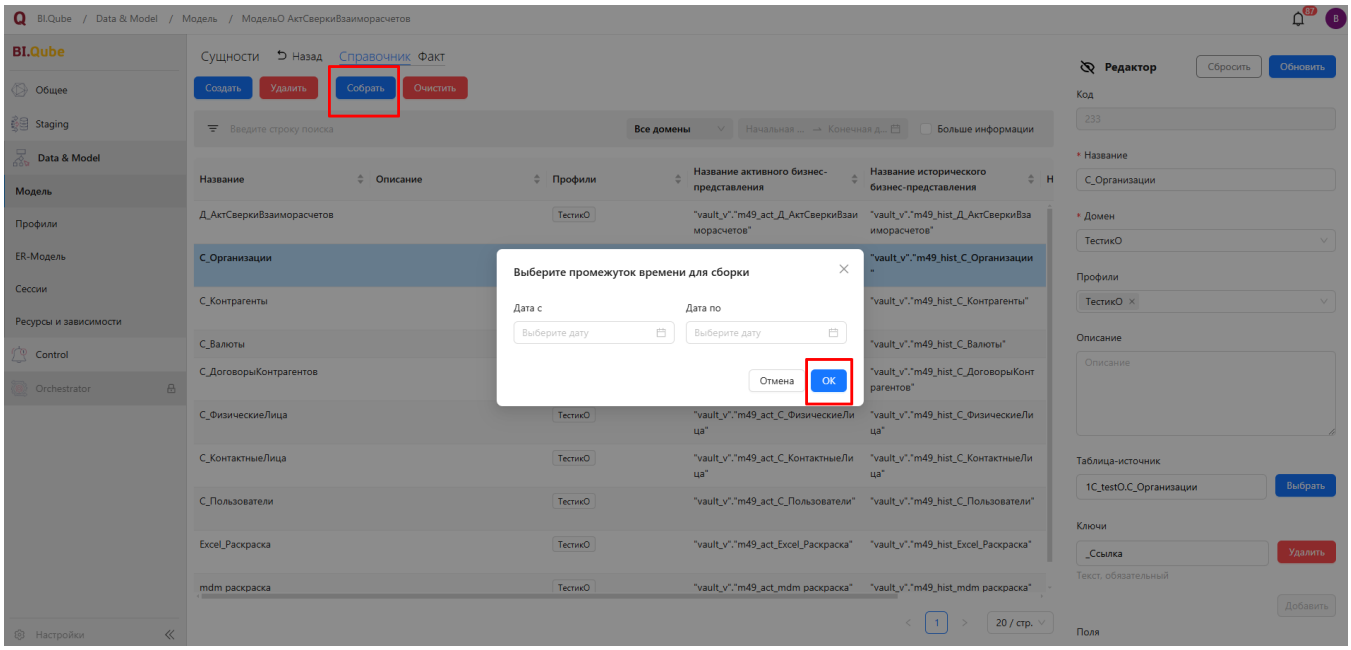


Рисунок. Сборка сущности

В основном окне нажать кнопку Собрать (Assembly), затем в появившемся диалоговом окне нажать ОК. Содержание можно посмотреть двойным щелчком левой кнопки мыши.

В сущность, созданную на основе таблицы-источника, можно добавлять атрибуты, не привязанные к таблице источнику. Такие атрибуты в последующем можно будет заполнить данными в ручном режиме.

Работа с моделью в графическом режиме

Система BI.Qube предоставляет пользователю возможность работы в графическом режиме. В таком режиме можно визуально увидеть весь состав модели, все связи, свойства созданных сущностей и создать новые. Здесь же можно увидеть, как раскладываются метаданные на объекты модели DataVault, зависимости объектов, какой объект создан на основе чего.

Для всех этих задач используются страницы:

- ER-model (Er-модель).
- Ресурсы и зависимости.

ER-модель

Для просмотра графического представления модели необходимо выбрать модель, а также один или более доменов, нажать кнопку Load ER-model (Загрузить ER-модель). В данной модели каждый графический объект несет определенный смысл, так прямоугольниками показаны сущности, внутри прямоугольников могут быть перечислены атрибуты сущности (зависит от режима отображения), линии между прямоугольниками символизируют связи.

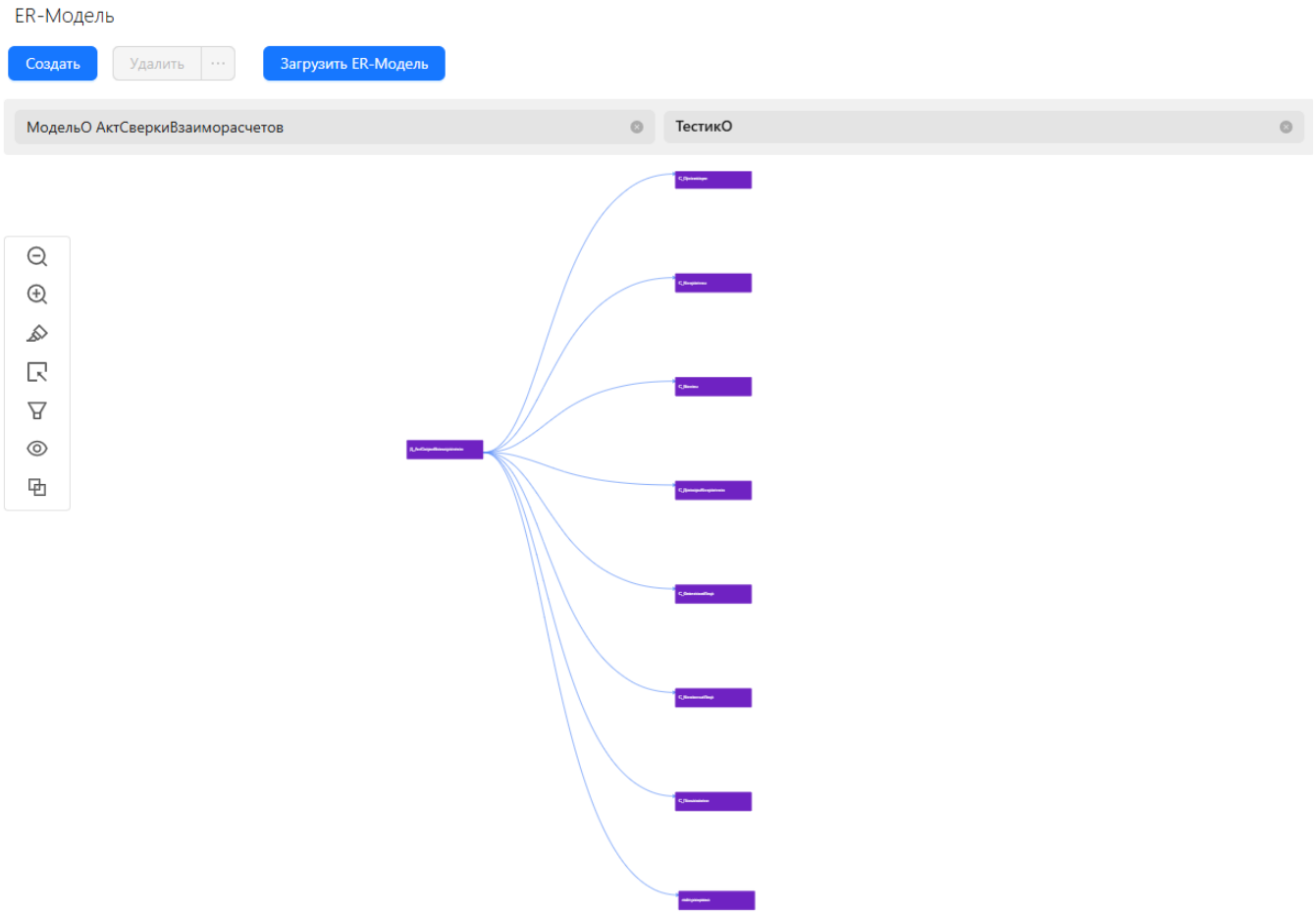


Рисунок. Выбор ER – модели в BI.Qube

При нажатии на любой графический объект в окне свойств справа появляется возможность редактирования свойств выбранного объекта.

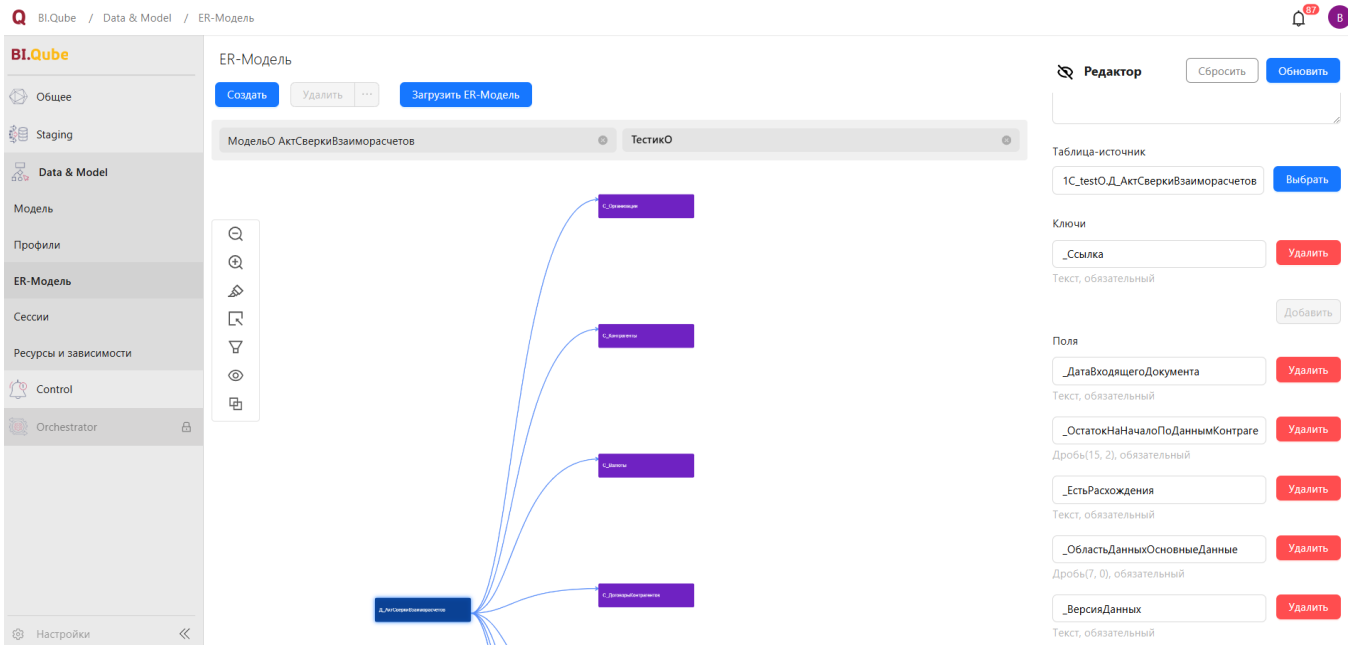


Рисунок. Редактирование таблицы в ER – модели

Рисунок. Панель инструментов (1 и 2 – лупа (уменьшить/увеличить); 3 – подсветка связей; 4 – фильтр слоёв; 5 – фильтр связей; 6 – режим отрисовки ER – модели; 7 – макет)

При выборе в панели инструментов фильтра связи предлагается возможность выбора объектов для отображения в ER – модели. Формат отображения зависит от режима отрисовки (концептуальная, детальная модель, DataVault).

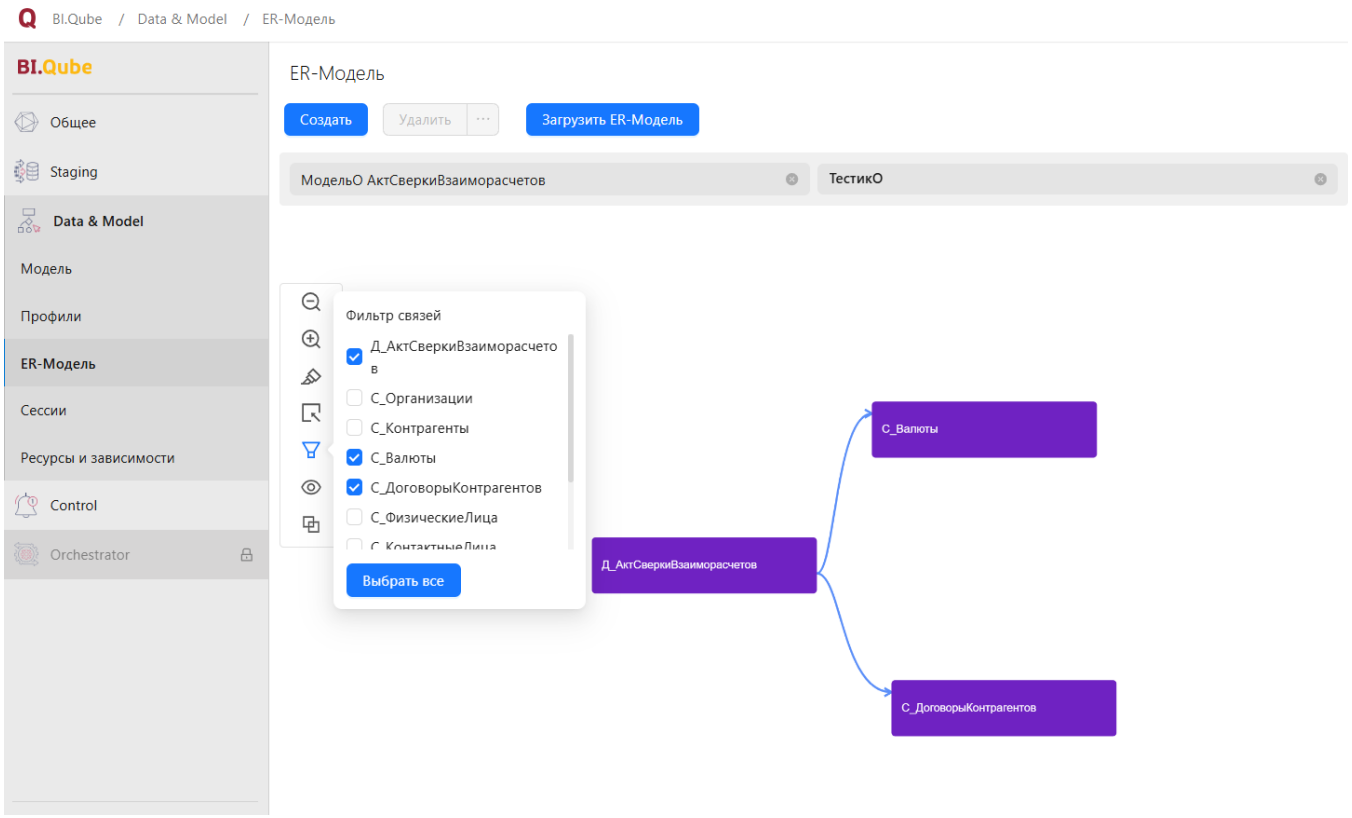


Рисунок. Работа с фильтром связей в режиме концептуальная модель

При визуализации ER – модель имеет три режима отображения модели:

- Conceptual model (концептуальная модель) – в этом режиме отображаются все сущности в виде прямоугольников со связями;
- Detail model (детальная модель) – в этом режиме в прямоугольниках отображаются атрибуты сущности;
- Data Vault – в этом режиме отображаются все объекты модели DataVault.

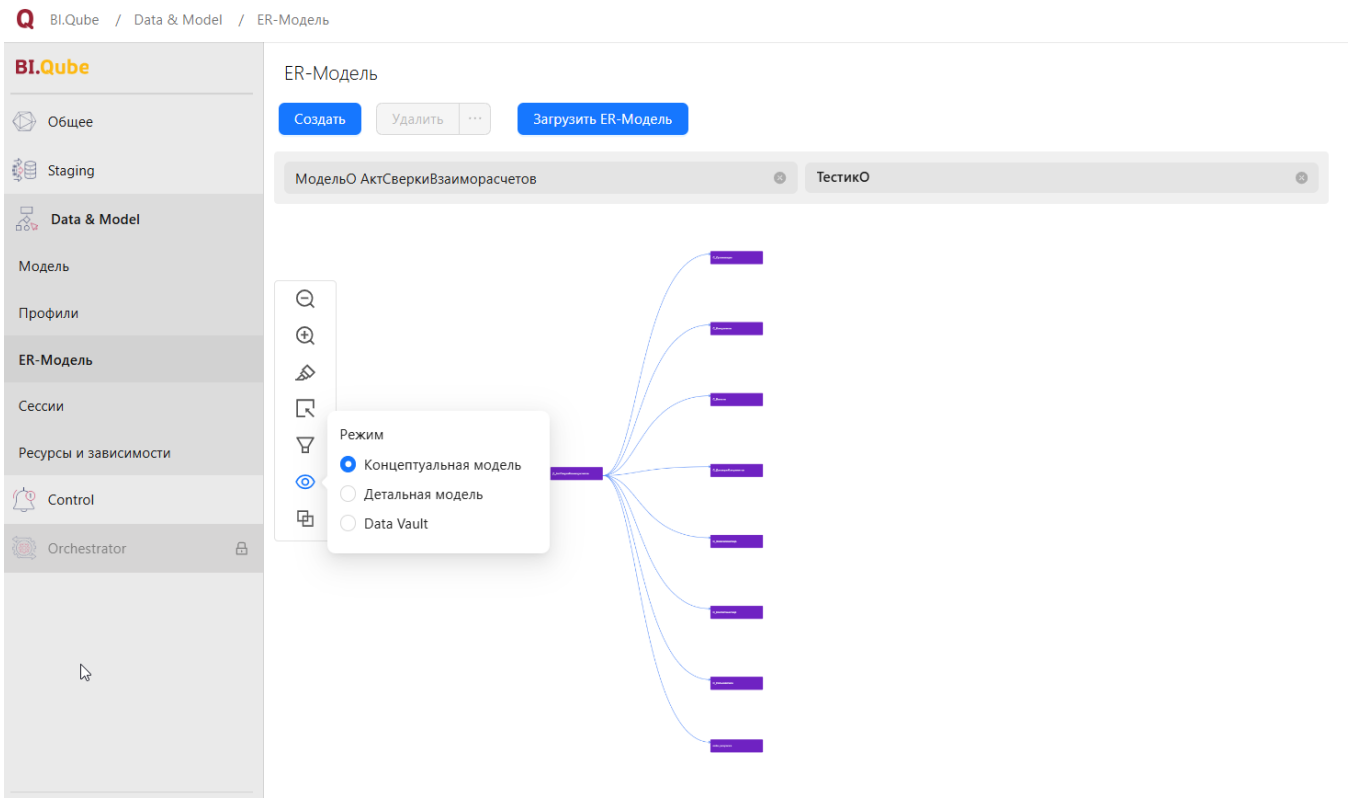


Рисунок. Пример отображения связей в концептуальной модели

Концептуальная модель данных удобна для отображения большого количества сущностей. Она определяет структуру моделируемой системы, свойства её элементов и причинно-следственные связи, присущие системе и существенные для достижения цели моделирования.

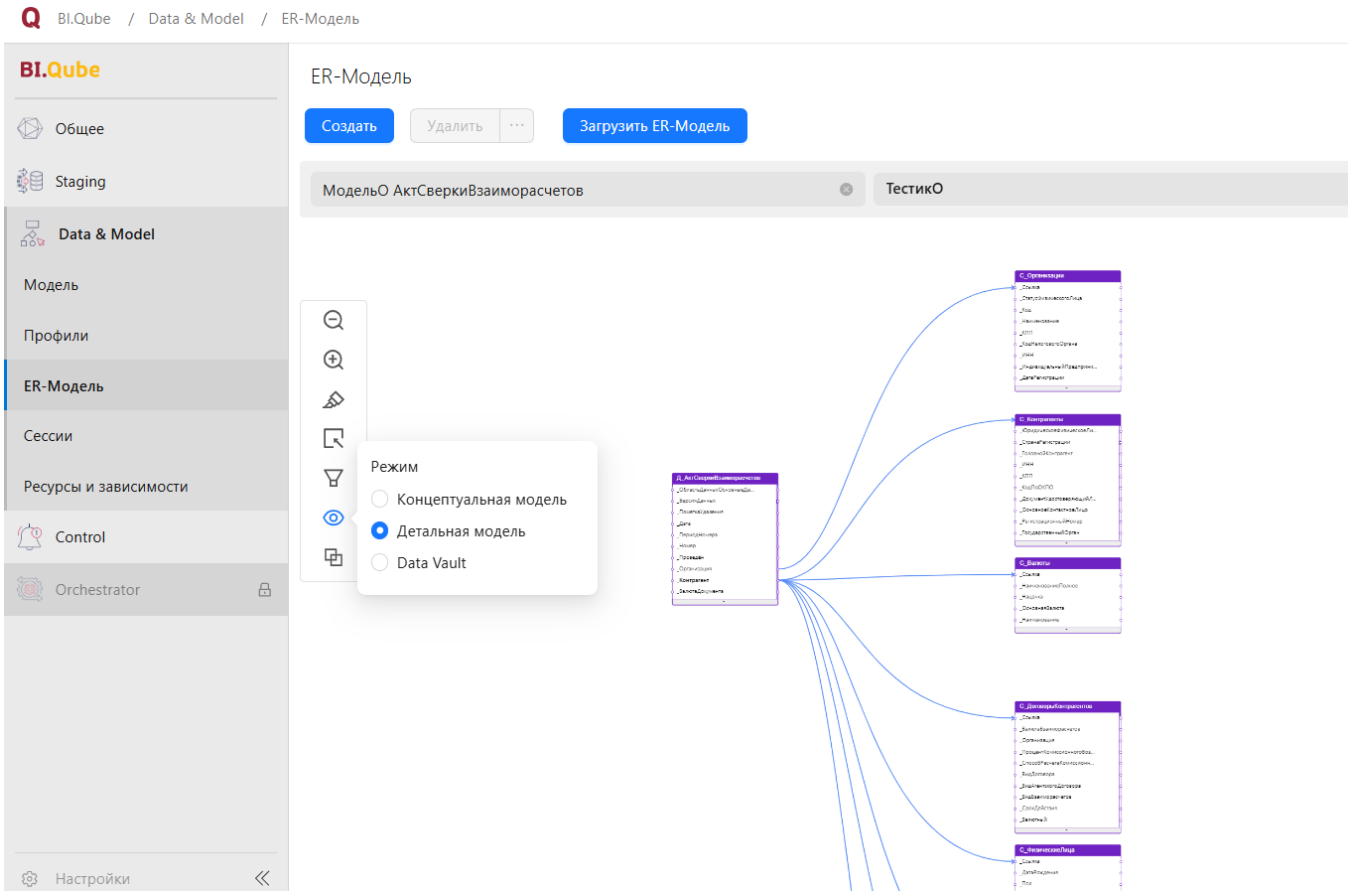


Рисунок. Пример отображения связей в детальной модели

Детальная модель позволяет пользователю более конкретизировано рассмотреть связи между полями сущностей для решения поставленных задач.

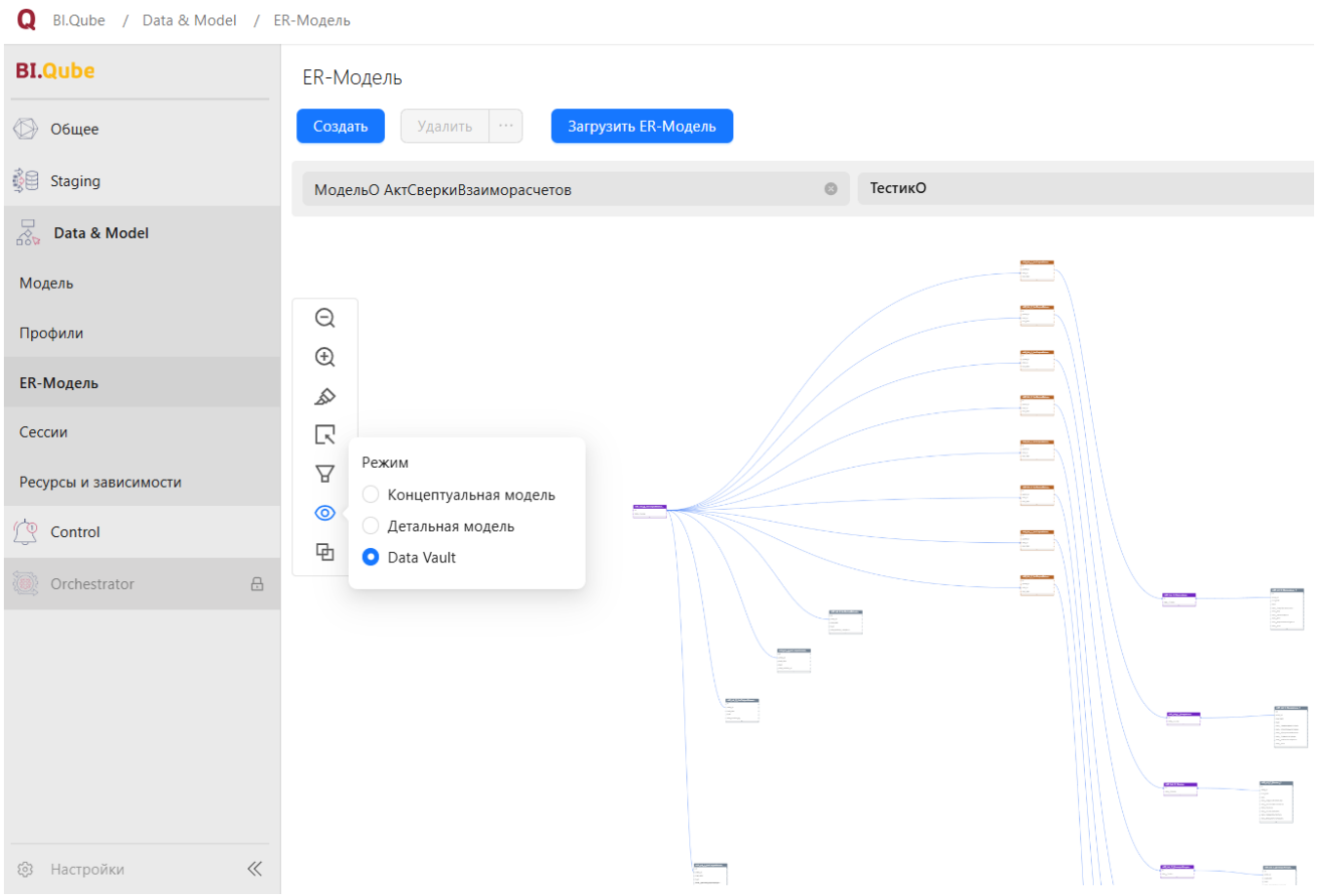


Рисунок. Пример отображения DataVault

При работе с ER – моделью возможен выбор двух вариантов отображения макета (слои, концентрический). А также выбор направлений отображения ER - модели: RL (справа налево), LR (слева направо), TB (сверху вниз), BT (снизу вверх).

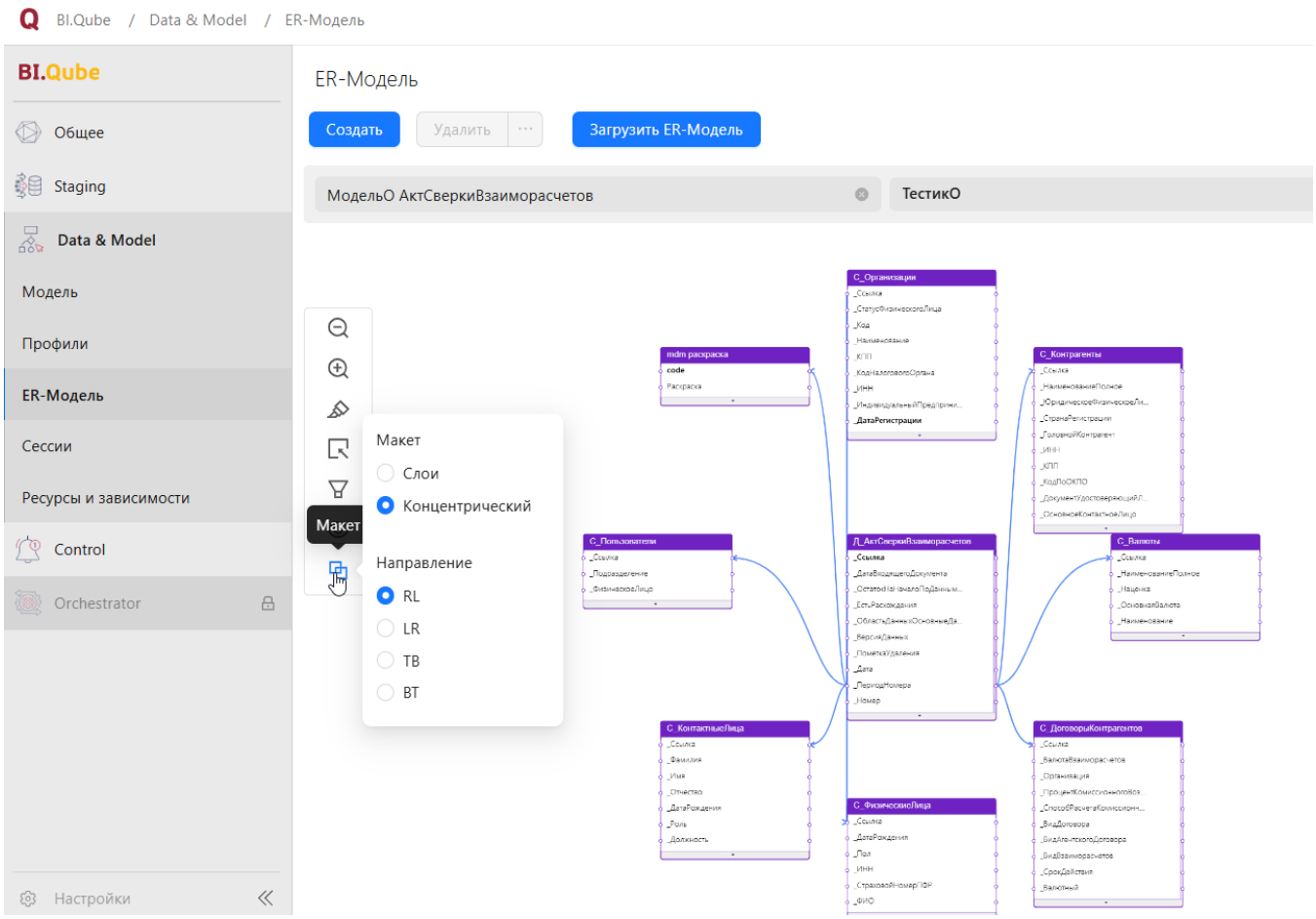


Рисунок. Концентрический макет

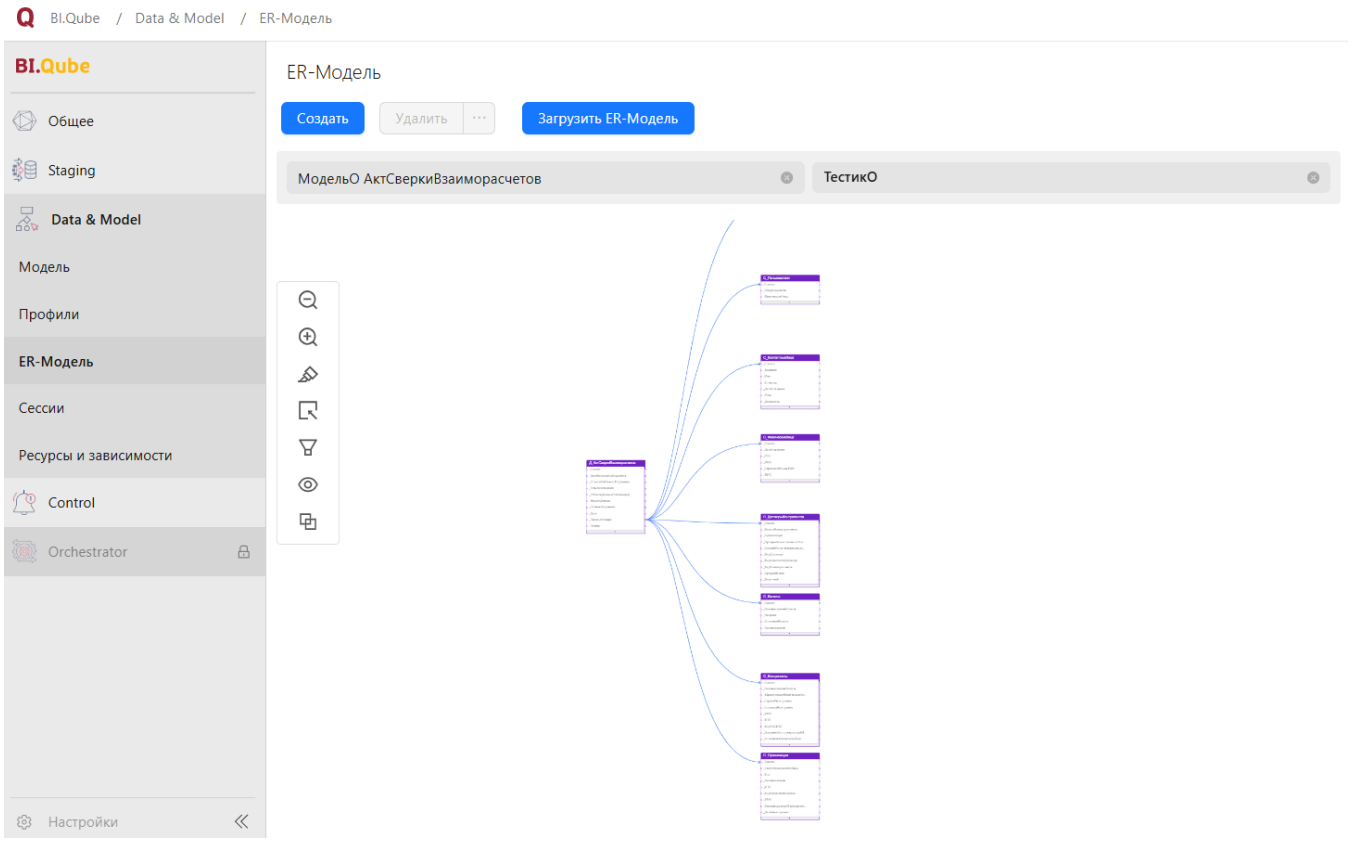
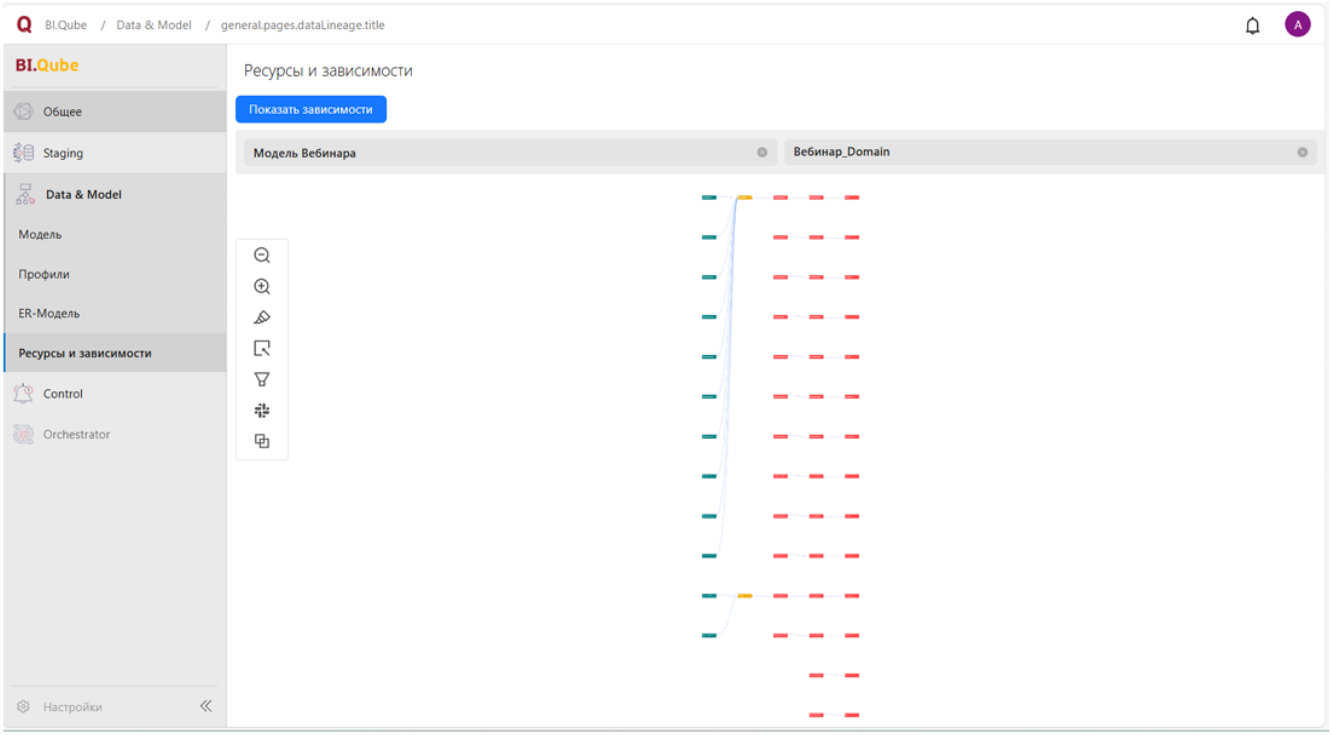


Рисунок. Макет в режиме «слои»

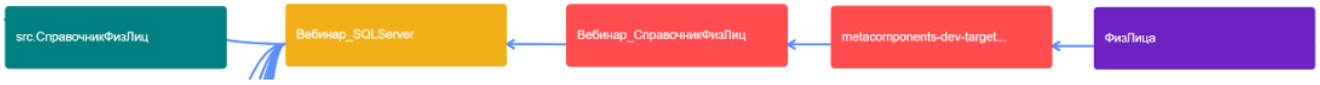
Ресурсы и зависимости

Для возможности отслеживания зависимостей между объектами системы BI.Qube используется страница "Ресурсы и зависимости". Здесь нужно выбрать модель, которую необходимо проанализировать и один или более доменов, если права доступа позволяют, щелкнуть по кнопке "Показать зависимости". В результате на экране отобразится графическое представление всех зависимостей.



По умолчанию фильтры отображают содержимое всей модели, доступна возможность управлять правилами отображения с использованием стандартных возможностей графического редактора. Каждый объект содержит имя типа объекта. Для рисунка ниже слева направо приведены следующие объекты:

- Таблица или файл с исходными данными
- Подключение к источнику данных
- Команда извлекающая данные из источника
- Таблица или представление в стейджинговом слое
- Объект модели данных (хаб, факт. золотая запись)



Существует два варианта отрисовки ER-модели: conceptual model (концептуальная модель) и detail model (детальная модель).

- conceptual model (концептуальная модель) – отображаются только свойства объекта;
- detail model (детальная модель) – к свойства объекта добавляются атрибуты объекта.

На рисунках ниже представлены обе модели (идёт разработка)

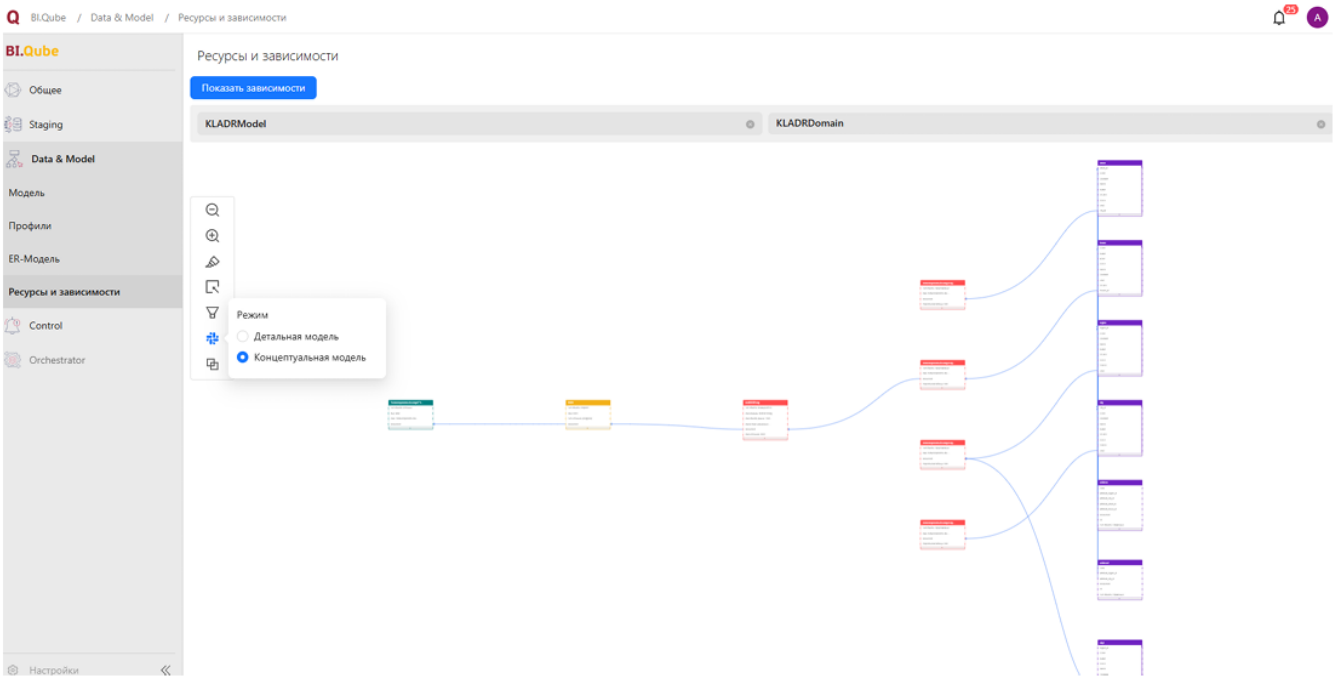


Рисунок. Концептуальное представление зависимостей

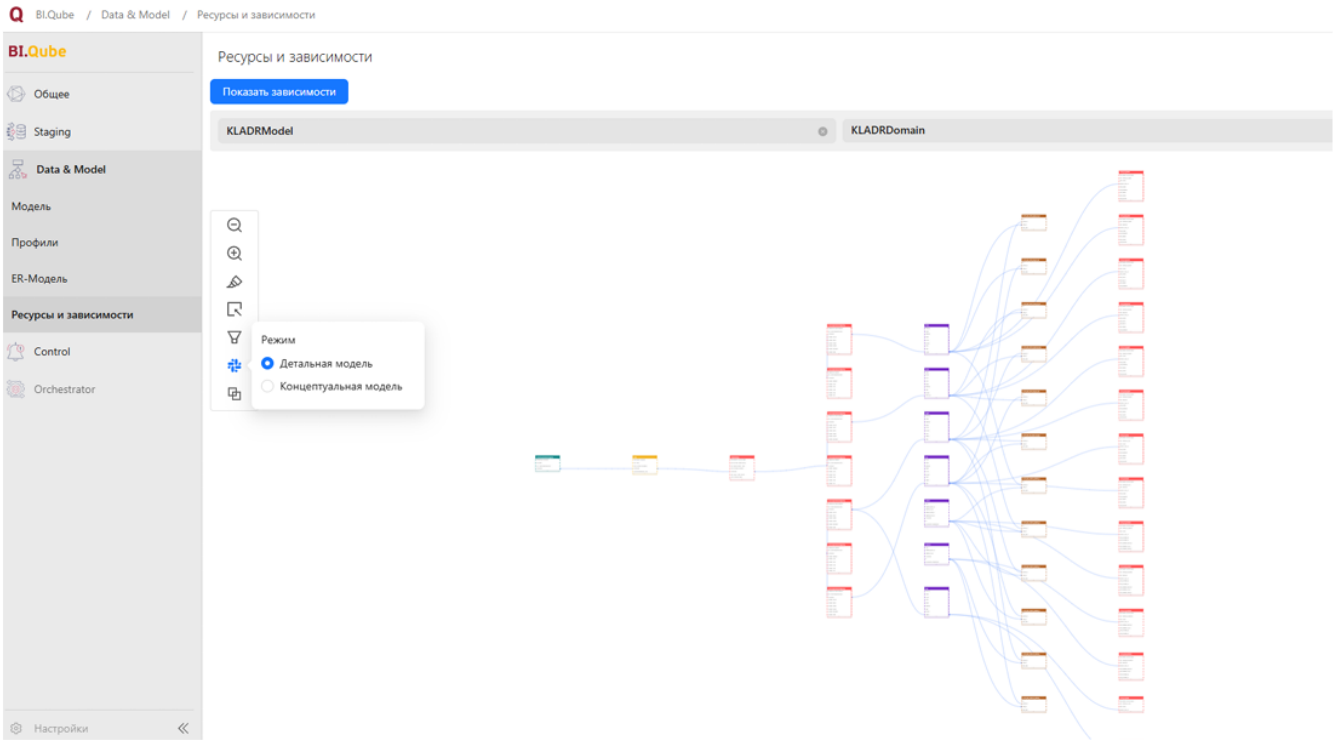


Рисунок. Детальное представление зависимостей

METACONTROL

Общие сведения

Основной целью компонента является предоставление возможности пользователю выполнять любые запросы на любом эндопоинте. Результат выполнения запроса отправляется на указанную почту и/или в настроенный заранее telegram-канал.

Компонент MetaControl входит в состав системы BI.Qube и может эксплуатироваться как отдельный компонент, так и в составе системы.

Наибольший эффект от применения компонента MetaControl можно получить в сочетании с использованием параметров. Вычисляя значения параметров на разных эндпоинтах позволит получить в сводном запросе компонента Metacontrol необходимый результат.

Описание компонента

Компонент имеет развитый визуальный веб-интерфейс и состоит из трех страниц:

- Списки рассылок – страница, предназначена для создания списков рассылок – перечень адресов, на которые будут отправляется сообщения с результатами выполнения запросов, созданных в компоненте.
- Категории – объединение результатов рассылок в одно письмо (категорию).
- Проверка – страница предназначена для создания запроса (контроля, бизнес-правила), формы и вида рассылки сообщений о результатах выполнения запроса.
- Профили – страница предназначена для запуска на выполнение запросов.
- Сессии – страница с результатами работы компонента.

СПИСОК РАССЫЛКИ

Страница Mailing list (Список рассылки) предназначена для создания и редактирования справочника адресов. Страница имеет стандартный интерфейс.

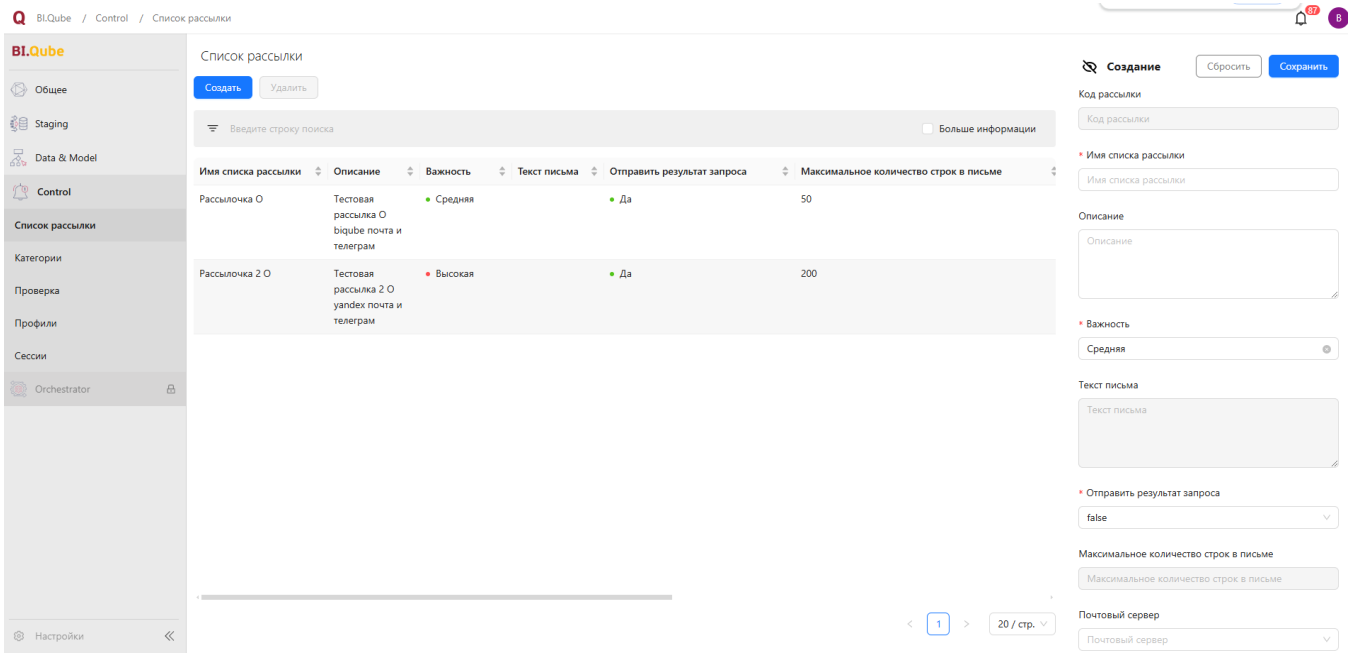
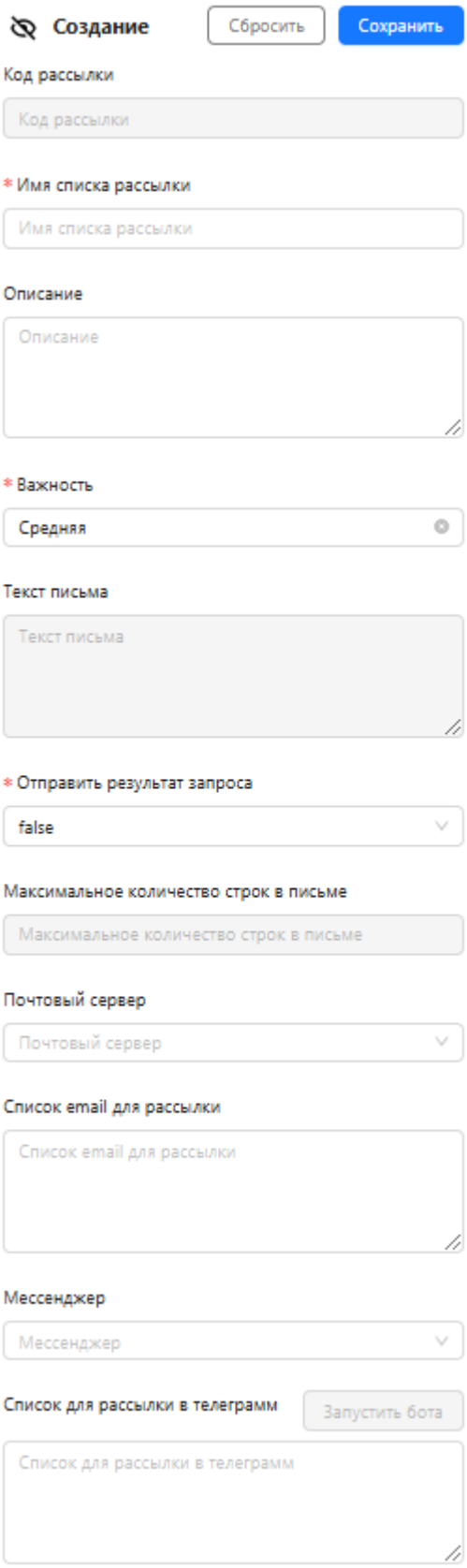


Рисунок. Страница для создания Mailing list (Список рассылки)

Для создания списка адресов необходимо нажать на строку таблицы Create (Создать) двойным щелчком.



Заполнить данные создаваемого источника в таблице Mailing list (Список рассылки):

- Имя списка рассылки – имя списка.
- Описание – краткое описание назначения списка рассылки.

- Важность – метка важности письма.
- Отправить результат запроса – отправляет в письме данные, полученные запросом.
- Максимальное количество строк в письме – ограничение на число строк в письме (иногда контроль может возвращать огромный результат, который может не прогрузиться в письме). Поле становится активным, если в предыдущем поле "Отправить результат запроса" стоит значение True.
- Почтовый сервер – выбор эндпоинта с почтовым сервером.
- Список Email для рассылки – список адресов, на которые будет выполнена рассылка, перечисляются через запятую.
- Мессенджер – выбор эндпоинта, содержащего мессенджер.
- Список для рассылки в Телеграм – формирование списка получателей сообщений в мессенджере Телеграм, имена пользователей перечисляются через запятую.

Далее нажать на значок Run bot (Запустить бота) (https://t.me/MetaControlComponentTGBot) и написать с указанного в поле Telegram_list (Список для рассылки в Телеграм) аккаунта слово «start». Нажать кнопку Save (Сохранить), чтобы завершить создание списка рассылки.

Редактирование осуществляется аналогично созданию списка рассылки – работа по редактированию производится в правом окне свойств. Для этого нужно выделить список адресов нажатием на строку.

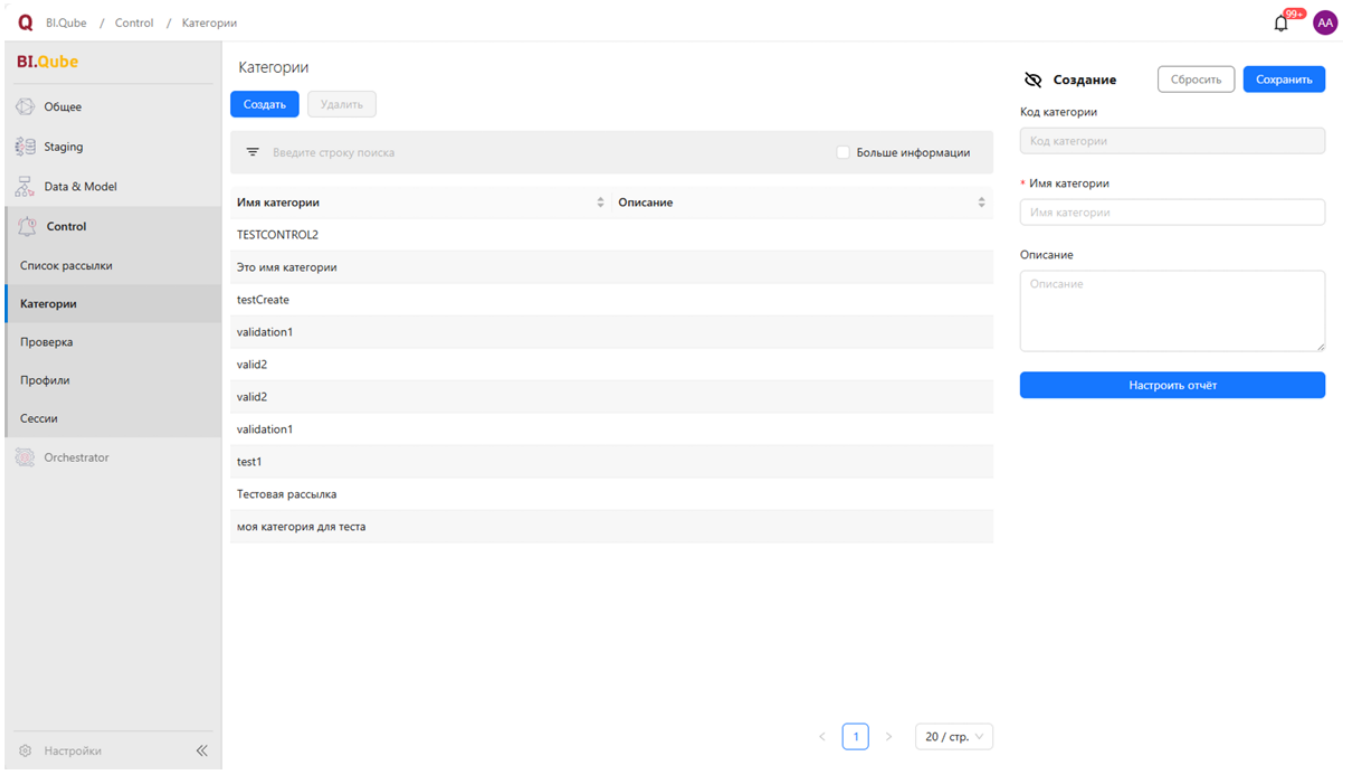
КАТЕГОРИЯ

Создание категории

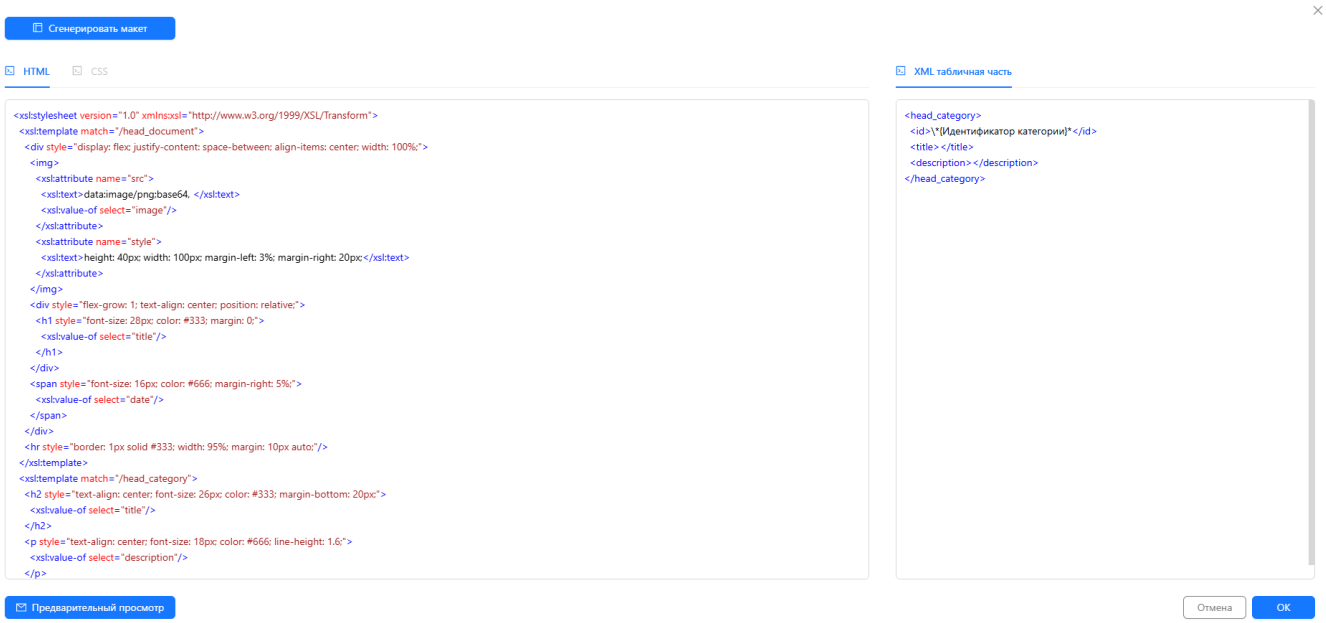
Страница категории предназначена для создания категорий. Категория позволяет объединить в одно сообщение (письмо) результаты нескольких проверок (контролей).

Для создания категории необходимо нажать кнопку "Создать" (Create) и ввести имя новой категории. Также можно заполнить необязательное поле "Описание" – краткое описание категории.

Внешний вид сообщения оформляется с использованием редактора HTML/XSLT, доступного по кнопке "Настроить отчет". Появится диалоговое окно, в котором можно настроить внешний вид письма. В системе поддерживается один вариант макета, внешний вид которого может быть настроен пользователем.



В окне настройки макета пользователю доступны поля свойств категории в формате xml. Пользователь имеет возможность вывести эти поля в отчет либо вовсе отказаться от их отображения в отчете. Отчет генерируется автоматически с минимальным оформлением. Пользователь имеет возможность удалить "лишнюю" информацию из отчета, включить стили оформления и наполнить html-тегами нужные информационные блоки. В случае нарушения глобальной структуры, система не сможет сгенерировать отчет. Посмотреть внешний вид отчета можно нажав на кнопку "Сгенерировать отчет", в брайзере откроется новая страница с отчетом – в данном случае заголовок отчета и имя категории с описанием.



HTML код разбит на две основные секции (см рисунок, строки кода между инструкциями <xsl:template match="/head_document">... </xsl:template>). В первой секции выводится строка заголовка отчета, здесь же рекомендуется размещать блоки с описанием стилей оформления (классы css). Вторая секция используется для вывода информации о категории: имя категории и описание, эти данные берутся из автоматически подготовленного XML, структура которого приведена справа от окна подготовки отчета. Идентификатор id – это номер категории в таблице базы данных системы BI.Qube, title – имя категории, description – описание категории. Если какие-то поля в интерфейсе настройки категории не заполнены, то в соответствующих строках отчета ничего не будет.

Если в процессе настройки отчета верстка макета оказалась нарушена, то привести к исходному состоянию можно нажав на кнопку "Сгенерировать макет". Все введенные изменения пользователя будут удалены и загружен макет по умолчанию.

Описание шаблона отчета

Шаблон отчета состоит из трех секций

Секция 1 – Заголовок письма (содержит логотип BI.Qube, надпись «отчет о работе Metacontrol", дата и время генерации отчета). Заголовки H1 В эту секцию разрешено добавлять описание классов CSS, отдельно стили нигде больше не настраиваются.
Секция 2 – Заголовок категории (содержит имя категории, описание категории). Заголовки H2.
Секция 3 – Отчеты. Заголовки H3, остальные теги.

Рисунок. Пример макета

Работа с секциями 1 и 2 описана выше, работа по настройке внешнего вида секции 3 приведена в разделе настройки контролей.

КОНТРОЛЬ

Создание контроля

Страница Validation (Проверка) предназначена для создания и редактирования запросов (контролей, бизнес-правил).

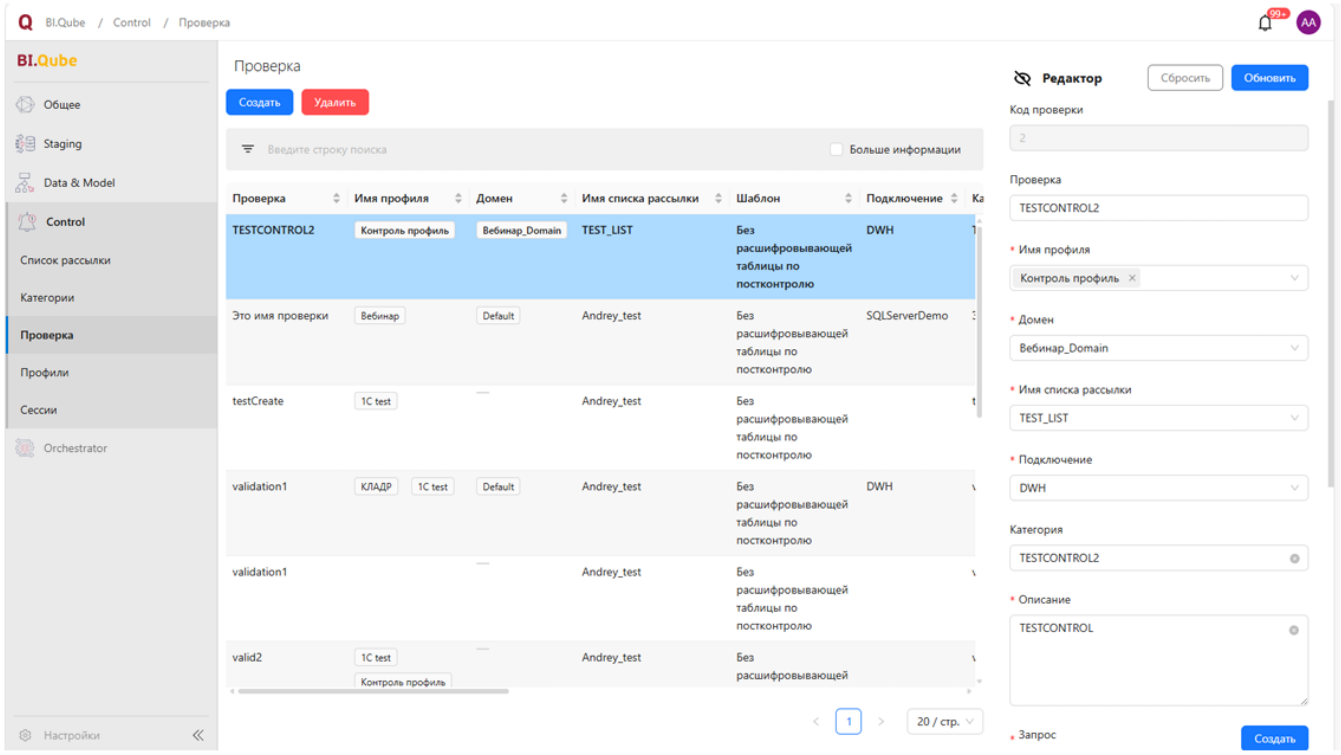


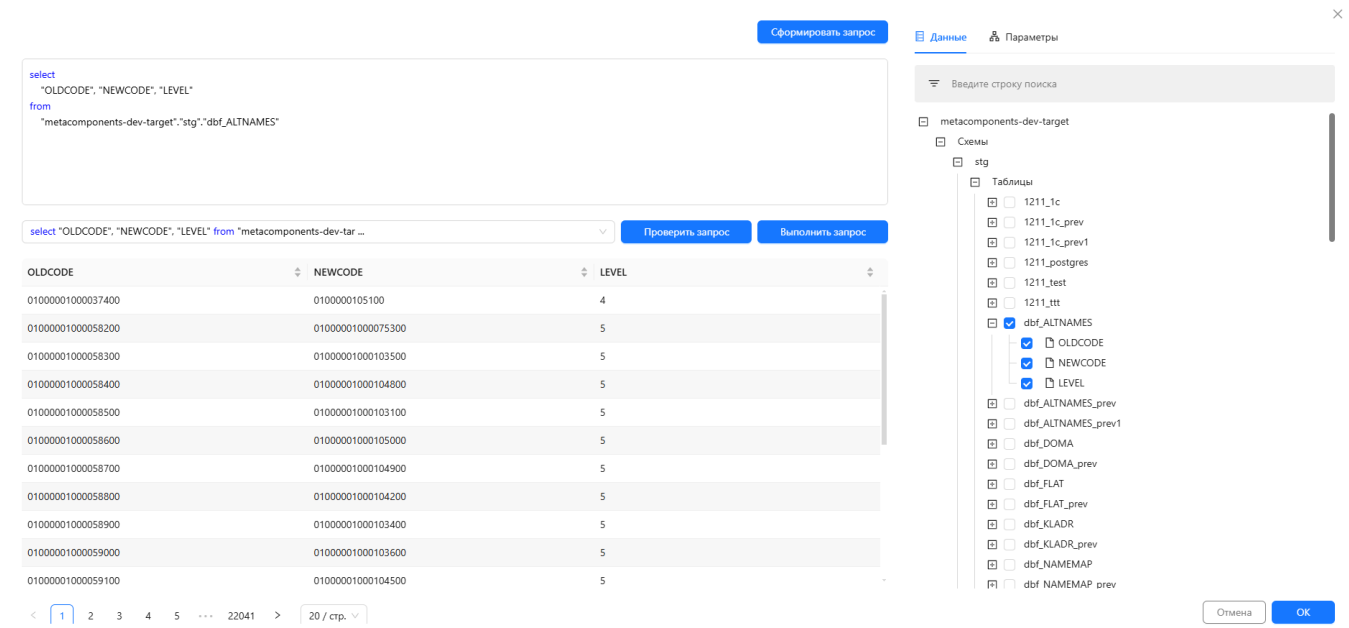
Рисунок. Содержание страницы Validation (Проверка)

Нажать кнопку Save (Сохранить), чтобы завершить создание запроса (контроля). Для удаления контроля его необходимо выделить и нажать кнопку Delete (Удалить).

Редактирование осуществляется аналогично созданию, при выделении нужного контроля в таблице.

Создание запроса

После выбора Endpoint, кнопка Queru (Запрос) становится доступна. Нажав на кнопку открывается диалоговое окно создания запроса. В правой части окна доступен список объектов выбранного Endpoint Data (Данные), к которым может быть составлен запрос. Простейший запрос на выборку может быть составлен в диалоговом режиме. Для этого в дереве объектов необходимо выбрать нужную таблицу, атрибуты и нажать кнопку Form a query (Сформировать запрос), после чего будет сформирован запрос. В него пользователь может вносить любые изменения или создавать собственные запросы, при этом нужно помнить, на каком endpoint будет выполнятся запрос и использовать синтаксис SQL-запроса выбранного Endpoint. После подготовки текста запроса следует нажать на кнопку Check request (Проверить запрос). В этот момент будет выполнена проверка запроса (не всегда возможно выполнить проверку, поэтому нет гарантии, что данная проверка обнаружит какие-то ошибки), затем нажать на кнопку Run query (Выполнить запрос). В зоне предварительного просмотра появятся результаты запроса, если запрос был корректен.



В теле запроса можно использовать пользовательские параметры, перечень которых доступен на вкладке UserParameters (Параметры). Ниже показан пример использования параметра в запросе.

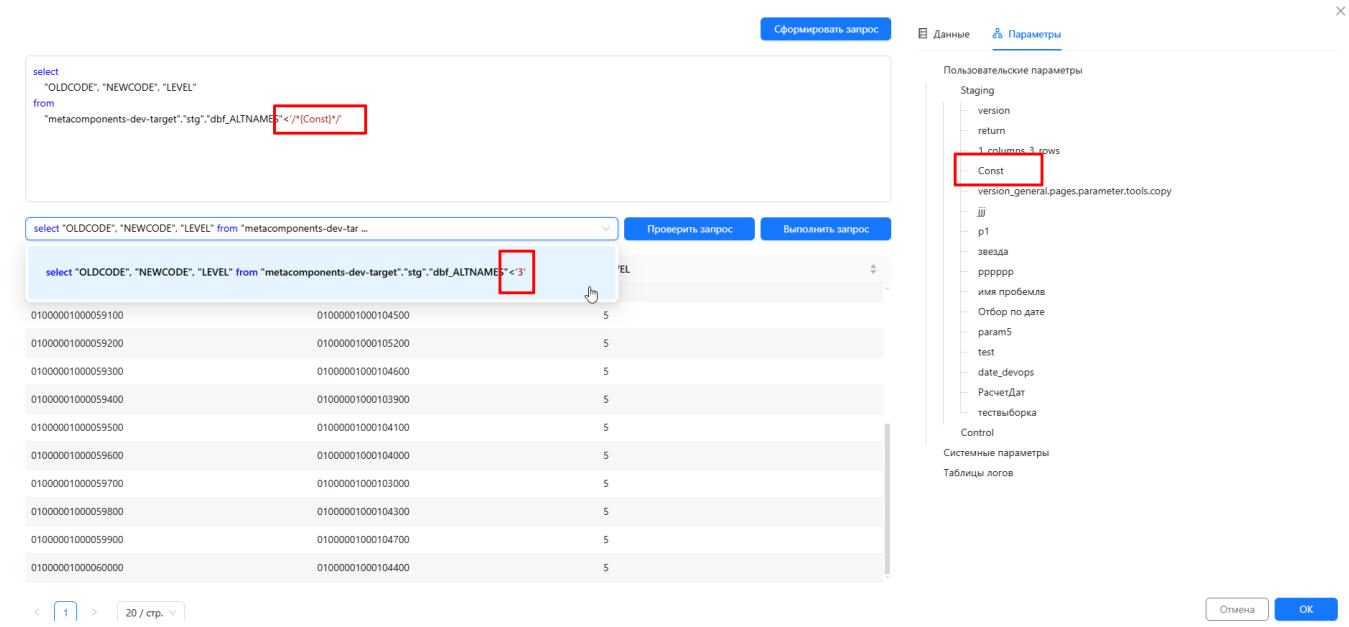


Рисунок. Добавление параметра в тело запроса

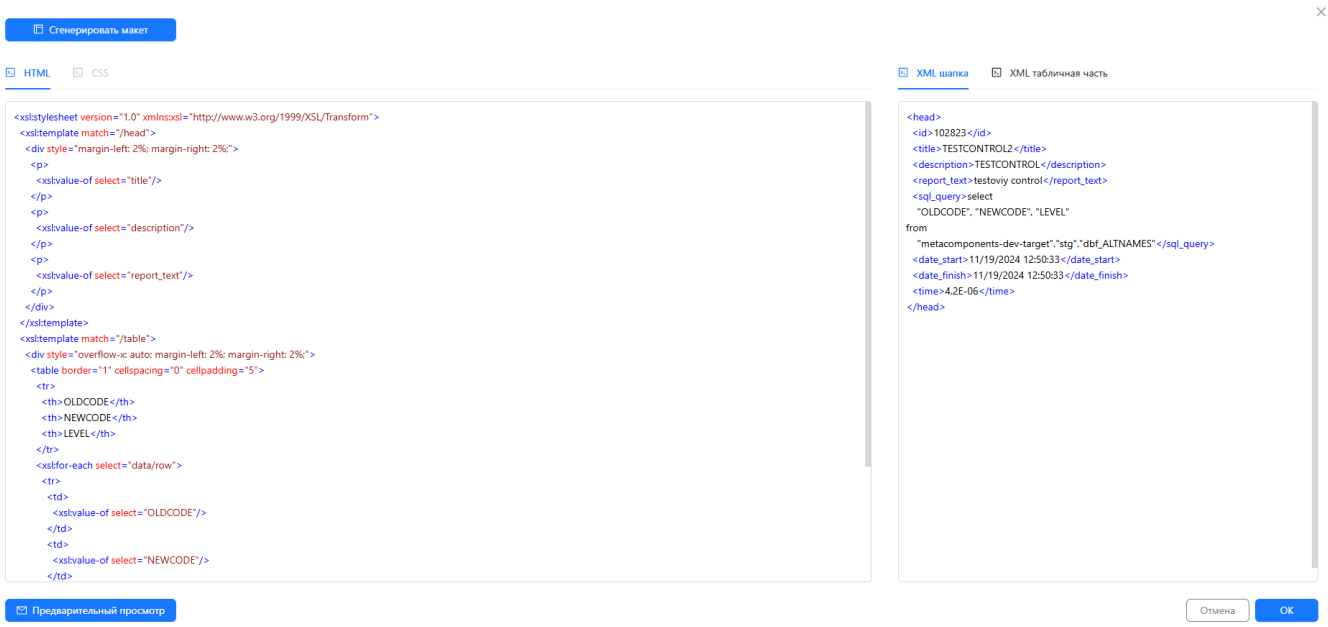
На рисунке показан синтаксис добавления параметра в тело запроса. После нажатия на кнопку Проверить запрос (Check request) происходит подстановка значения параметра в запрос, и если результат выполнения параметра состоит более чем из одного значения, происходит "размножение" запроса, в данном примере параметр вернул только одно значение, которое автоматически подставлено в запрос.

Настройка внешнего вида отчета

После нажатия на кнопку "Настроить отчет" в настройках контроля на экран выводится диалоговое окно настройки внешнего вида отчета. В соответствии с общим макетом здесь настраивается внешний вид секции контента, стили, созданные при настройке категории, применятся для данной секции.

Здесь все действия точно такие же, как и в случае настройки внешнего вида заголовка отчета и категории. Отличием является то, что автоматически может быть создано несколько xml. xml-шапка – это поля настройки контроля, xml-табличная часть – это результат выполнения запроса контроля.

При этом секция разбита на два блока. Первый блок содержит данные из xml-шапка, второй блок – из xml-табличная часть.



Если в категорию объединяются несколько контролей, то все они размещаются друг под другом.

ПРОФИЛЬ METACONTROL

На данной странице доступна единственная возможность запустить созданный запрос (контроль) на выполнение. Для этого необходимо нажать кнопку Start (Начать) после выбора профиля, запросы которого необходимо выполнить.

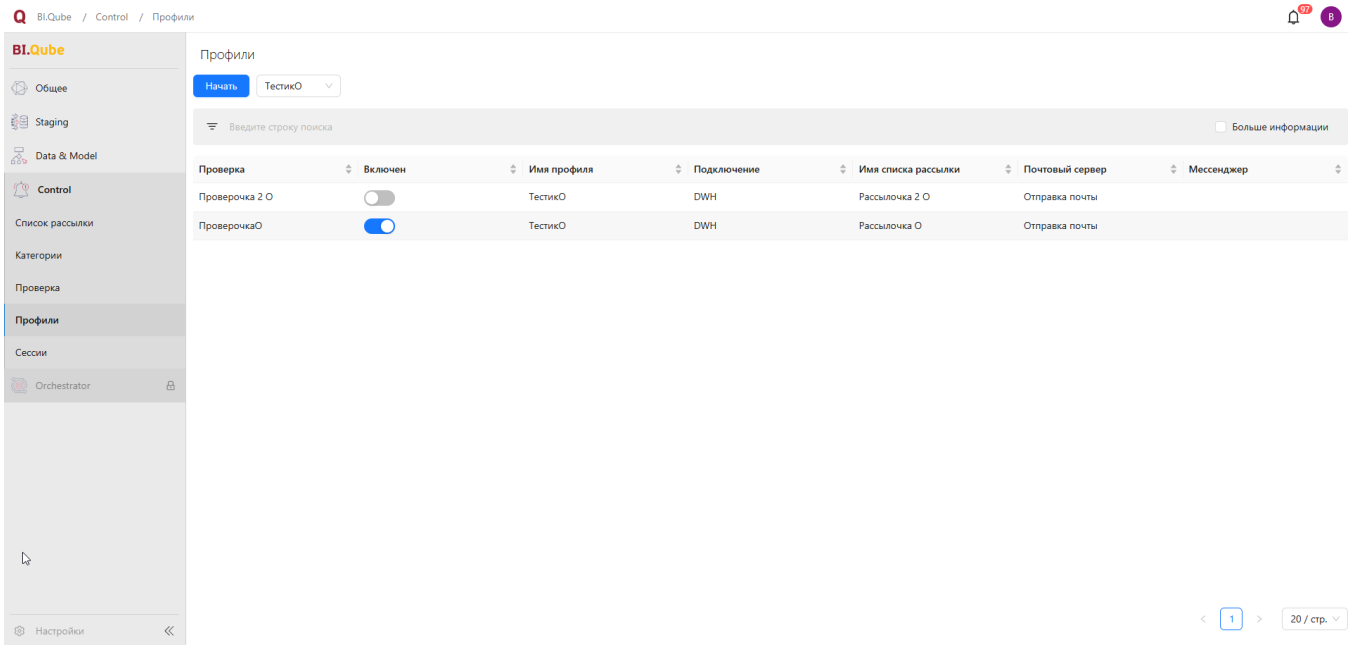


Рисунок. Страница Профили в разделе Control

Результатом работы запроса (контроля) будет сообщение в почте и/или telegram-канале.

СЕССИИ (METACONTROL)

На данной странице можно посмотреть результаты выполнения проверки, статистику, успешность выполнения.

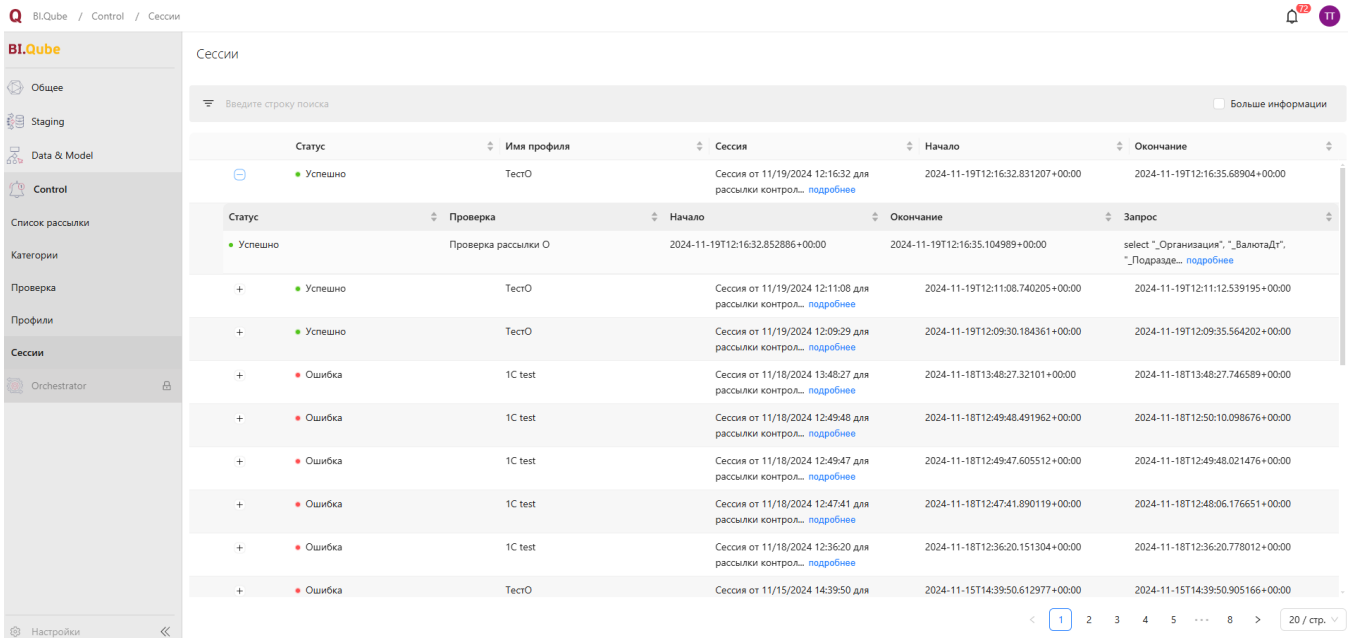


Рисунок. Страница Сессии в разделе Control

Детальная информация о выполнении проверки доступна после двойного щелчка по имени проверки.

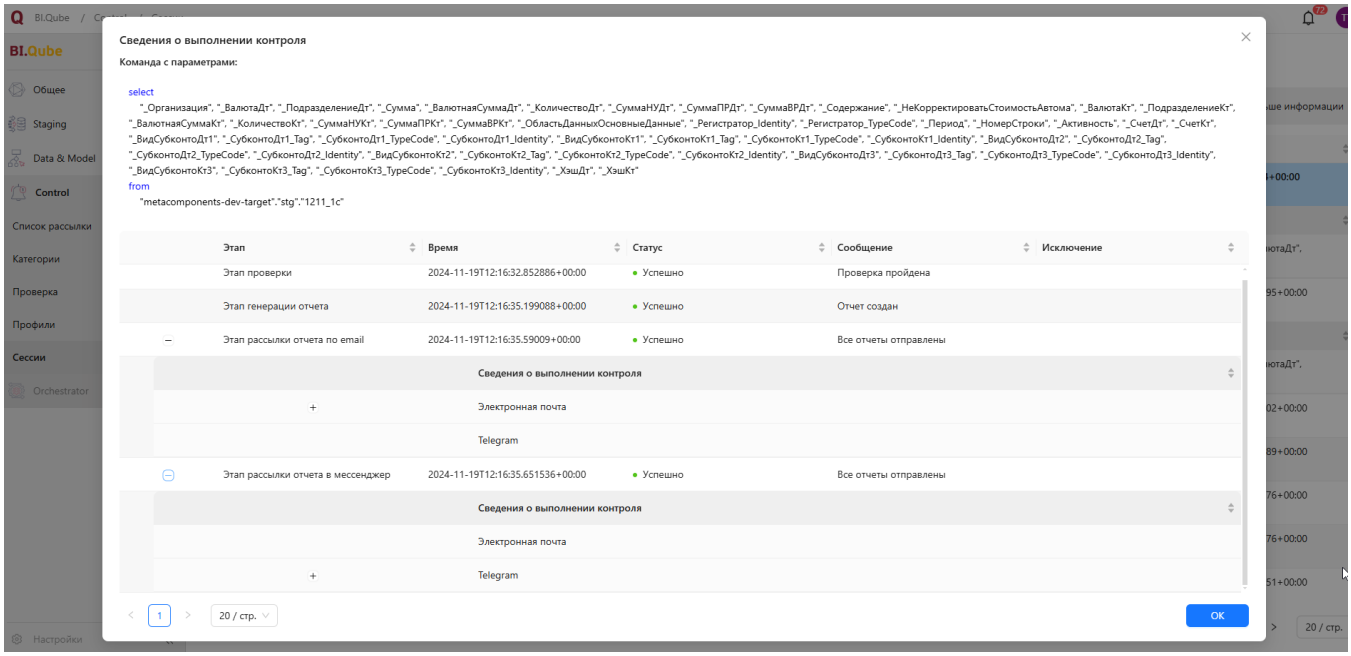


Рисунок. Подробная информация о выполнении контроля

КОНСОЛЬНОЕ ПРИЛОЖЕНИЕ

Консольное приложение может быть использовано для выполнения запуска команд на выполнение.

Синтаксис

Для запуска команды используется следующий синтаксис:

Пример команды

```
BIQube.Cli [command] [option] [argument]
```

Например, вызов команды выполнения профиля в PowerShell:

Выполнение профиля

```
./BIQube.Cli staging execute-profile --profile-id 3
```

Посмотреть все параметры команды (или дополнительную информацию) можно, вызвав параметр --help или -h перед командой:

```
PS C:\Users\ibish\Desktop\work\metacomponents\BIQube.Cli\bin\Debug\net6.0> ./BIQube.Cli staging execute-profile -h
Description:
  Выполнить профиль

Usage:
  BIQube.Cli staging execute-profile [options]

Options:
  -p, --profile-id <profile-id> (REQUIRED)            Id профиля
  --configuration-file <configuration-file>           Файл с настройками конфигурации (.ini)
  --configuration-parameter <configuration-parameter> Параметры конфигурации (ключ=значение)
  -?, -h, --help                                      Show help and usage information
```

Настройка конфигураций

Способы конфигурирования консольного приложения (чем меньше номер, тем больше приоритет):

1. Параметр при вызове: --parameter-configuration "НАИМЕНОВАНИЕ_ПАРАМЕТРА=ЗНАЧЕНИЕ_ПАРАМЕТРА"
2. .ini файл (https://ru.wikipedia.org/wiki/.ini): --configuration-file"путь"
3. Переменные окружения (https://ru.wikipedia.org/wiki/Переменная_среды

Пример структуры ini файла:

sample.ini

```
MDM_TARGET_DATABASE = db_name
MDM_TARGET_DATABASE_TYPE = db_type
MDM_TARGET_VIEW_SCHEMA = shema
MDM_TARGET_STORAGE_SCHEMA = shema

MDM_SETTINGS_DATABASE = sb_name

MDM_SETTINGS_SCHEMA = shema
METAVAULT_SETTINGS_SCHEMA = shwma
COMMON_SETTINGS_SCHEMA = shema

[Keycloak]
Realm = realm_value
AuthServerUrl = auth_server_url_value
SslRequired = ssl_value
Resource = resources_value
PublicClient = public_client_value
VerifyTokenAudience = vertify_token_audience_value
UseResourceRoleMappings = use_resource_value
ConfidentialPort = 0

[KeycloakApi]
ServiceUrl = <url>
Realm = <realm>
ClientUuid = <client uuid>
ApiVersion = 22

[BIQube]
AuthEnabled = true
UseRoleModelMapping = false

[ConnectionStrings]
Settings = User ID=postgres;Password=1234;Host=localhost;Port=5435;Database=db_name

[KeyVault]
Address = adress_value
Prefix =
Token = token_value
Mount = mount_value
```

Команды

Консольное приложение BIQube.Cli имеет следующие команды:

- common – отвечает за компонент MetaCommon
- control – отвечает за компонент MetaControl
- staging – отвечает за компонент MetaStaging
- data-and-model – отвечает за компонент MetaMasterData

Команда common

Под-команды:

- create-session – создание сессии
- get-session – получить сессию(и)

Команда create-session

Параметры:

- -n, --name – название сессии
- -d, --description – описание сессии (не обязательное)

В качестве результата получаем id сессии. Пример:

Создание сессии

```
./BIQube.Cli common create-session --name "new_session" --configuration-file "C:\BIQube.
Cli\Configuration\config.ini"

17
```

Команда get-session

Параметры:

- -i, --id – идентификатор сессии (не обязательное)
- -n, --name – название сессии (не обязательное)

Если использовать данную команду без параметров, получим все сессии. Пример:

Получить сессию

```
./BIQube.Cli common get-session --name "new_session" --configuration-file "C:\BIQube.
Cli\Configuration\config.ini"

Id: 17
: new_session
:
 : 26.07.2024 12:13:02
```

Команда control

Подкоманды:

- execute-profile – выполнить все проверки профиля
- execute-validation – выполнение проверки

Команда execute-profile

Параметры:

- -p, --profile-id – идентификатор профиля

Пример:

Выполнение профиля

```
./BIQube.Cli control execute-profile -p 3 --configuration-file "C:\BIQube.Cli\Configuration\config.ini"

  MetaControl
 : ibishov.tural20@mail.ru
         .pdf     Turalskiu (chatId: 5838428053)
 sdc.pdf   pdf .
 sdc.pdf (sdc)  (chatId: 5838428053)
```

Команда execute-validation

Параметры:

- -v, --validation-id – идентификатор проверки

Пример:

Выполнение проверки

```
./BIQube.Cli control execute-validation -v 3 --configuration-file "C:\BIQube.Cli\Configuration\config.ini"

  MetaControl
 : ibishov.tural20@mail.ru
         .pdf     Turalskiu (chatId: 5838428053)
 sdc.pdf   pdf .
 sdc.pdf (sdc)  (chatId: 5838428053)
```

Команда staging

Подкоманды:

- init-session – инициализация сессии
- execute-profile – выполнение всех команд профиля
- execute-load-command – выполнить команду загрузки

Команда init-session

Параметры:

- -p, --profile-id – идентификатор профиля
- -d, --dag-id – идентификатор dag
- -d-s, --data-start – время начала (не обязательное)

Пример:

Инициализация сессии

```
./BIQube.Cli staging init-session --profile-id 3 --dag-id "dag" --date-start "2023-03-03" --configuration-
file "C:\BIQube.Cli\Configuration\config.ini"

{"session_id":33,"commands":[{"session_command_id":32,"command_id":4,"command":"select \"migration_id\", \"
product_version\" from \"metacomponents\".\"meta_control\".\"migration_history\"","destination_object":"
meta_control.migration_history","partition_postfix":null,"partition_column":null,"partition_values":null,
"partition_type":null,"partition_column_type":null,"load_type":null,"json_pattern":null,"is_json_flatten":
null,"need_response_to_csharp":null,"json_jolt":null}]}
```

Команда execute-profile

Параметры:

- -p, --profile-id – идентификатор профиля

Пример:

Выполнение профиля

```
./BIQube.Cli staging execute-profile -p 3 --configuration-file "C:\BIQube.Cli\Configuration\config.ini"


[26.07.2024 16:00:37] (  postgresql  postgresql)
[26.07.2024 16:00:37] (  postgresql  postgresql)
[26.07.2024 16:00:38] (  postgresql  postgresql)
[26.07.2024 16:00:38] (  postgresql  postgresql)      PostgreSQL
[26.07.2024 16:00:38] (  postgresql  postgresql)
    :
 create schema if not exists "meta_control";
 drop table if exists "meta_control"."migration_history_temp"; create table "meta_control"."
migration_history_temp" ( "migration_id" text, "product_version" text
)


 create schema if not exists "meta_control";
 drop table if exists "meta_control"."migration_history_temp"; create table "meta_control"."
migration_history_temp" ( "migration_id" text, "product_version" text
)
[26.07.2024 16:00:38] (  postgresql  postgresql)
[26.07.2024 16:00:38] (  postgresql  postgresql)   ...
[26.07.2024 16:00:38] (  postgresql  postgresql)    1
[26.07.2024 16:00:38] (  postgresql  postgresql)   :

ReadCommitted
copy "meta_control"."migration_history_temp" (
"migration_id", "product_version"
) from stdin (format binary)
[26.07.2024 16:00:38] (  postgresql  postgresql)   1
[26.07.2024 16:00:38] (  postgresql  postgresql)      temp-
[26.07.2024 16:00:38] (  postgresql  postgresql)
,     :
do language plpgsql $$ begin if exists (
               select * from information_schema.tables where table_name = 'migration_history_prev' and
table_schema = 'meta_control'
        ) then
        drop table if exists "meta_control"."migration_history_prev1";
        alter table "meta_control"."migration_history_prev" rename to "migration_history_prev1"; end if;
if exists (             select * from information_schema.tables where table_name = 'migration_history' and
table_schema = 'meta_control' ) then     alter table "meta_control"."migration_history" rename to
"migration_history_prev"; raise notice '   depr'; else raise notice '  '; end if; alter table
"meta_control"."migration_history_temp" rename to "migration_history"; raise notice '  '; end $$;
do language plpgsql $$ begin if exists (
               select * from information_schema.tables where table_name = 'migration_history_prev' and
table_schema = 'meta_control'
        ) then
        drop table if exists "meta_control"."migration_history_prev1";
        alter table "meta_control"."migration_history_prev" rename to "migration_history_prev1"; end if;
if exists (             select * from information_schema.tables where table_name = 'migration_history' and
table_schema = 'meta_control' ) then     alter table "meta_control"."migration_history" rename to
"migration_history_prev"; raise notice '   depr'; else raise notice '  '; end if; alter table
"meta_control"."migration_history_temp" rename to "migration_history"; raise notice '  '; end $$;
[26.07.2024 16:00:38] (  postgresql  postgresql)
[26.07.2024 16:00:38] (  postgresql  postgresql)
```

Команда execute-load-command

Параметры:

- -i, --session-command-id – идентификатор команды

Пример:

Выполнение команды

```
./BIQube.Cli staging execute-load-command --session-command-id 3 --configuration-file "C:\BIQube.
Cli\Configuration\config.ini"


[26.07.2024 16:00:37] (  postgresql  postgresql)
[26.07.2024 16:00:37] (  postgresql  postgresql)
[26.07.2024 16:00:38] (  postgresql  postgresql)
[26.07.2024 16:00:38] (  postgresql  postgresql)      PostgreSQL
[26.07.2024 16:00:38] (  postgresql  postgresql)
    :
 create schema if not exists "meta_control";
 drop table if exists "meta_control"."migration_history_temp"; create table "meta_control"."
migration_history_temp" ( "migration_id" text, "product_version" text
)


 create schema if not exists "meta_control";
 drop table if exists "meta_control"."migration_history_temp"; create table "meta_control"."
migration_history_temp" ( "migration_id" text, "product_version" text
)
[26.07.2024 16:00:38] (  postgresql  postgresql)
[26.07.2024 16:00:38] (  postgresql  postgresql)   ...
[26.07.2024 16:00:38] (  postgresql  postgresql)    1
[26.07.2024 16:00:38] (  postgresql  postgresql)   :

ReadCommitted
copy "meta_control"."migration_history_temp" (
"migration_id", "product_version"
) from stdin (format binary)
[26.07.2024 16:00:38] (  postgresql  postgresql)   1
[26.07.2024 16:00:38] (  postgresql  postgresql)      temp-
[26.07.2024 16:00:38] (  postgresql  postgresql)
,     :
do language plpgsql $$ begin if exists (
               select * from information_schema.tables where table_name = 'migration_history_prev' and
table_schema = 'meta_control'
        ) then
        drop table if exists "meta_control"."migration_history_prev1";
        alter table "meta_control"."migration_history_prev" rename to "migration_history_prev1"; end if;
if exists (             select * from information_schema.tables where table_name = 'migration_history' and
table_schema = 'meta_control' ) then     alter table "meta_control"."migration_history" rename to
"migration_history_prev"; raise notice '   depr'; else raise notice '  '; end if; alter table
"meta_control"."migration_history_temp" rename to "migration_history"; raise notice '  '; end $$;
do language plpgsql $$ begin if exists (
               select * from information_schema.tables where table_name = 'migration_history_prev' and
table_schema = 'meta_control'
        ) then
        drop table if exists "meta_control"."migration_history_prev1";
        alter table "meta_control"."migration_history_prev" rename to "migration_history_prev1"; end if;
if exists (             select * from information_schema.tables where table_name = 'migration_history' and
table_schema = 'meta_control' ) then     alter table "meta_control"."migration_history" rename to
"migration_history_prev"; raise notice '   depr'; else raise notice '  '; end if; alter table
"meta_control"."migration_history_temp" rename to "migration_history"; raise notice '  '; end $$;
[26.07.2024 16:00:38] (  postgresql  postgresql)
[26.07.2024 16:00:38] (  postgresql  postgresql)
```

Команда data-and-model

Подкоманды:

- assembly-profile – сборка профиля
- assembly-link – сборка ссылки
- assembly-hub – сборка хаба
- assembly-business-view – сборка представления
- assembly-satellite – сборка satellite
- assembly-fact – сборка факта

Команда assembly-profile

Параметры:

- -p, --profile-id – идентификатор профиля
- -s, --session-id – идентификатор сессии (не обязательное)
- -f, --date-from – дата С для инкрементальной загрузки (не обязательное)
- -t, --date-to – дата По для инкрементальной загрузки (не обязательное)

Пример:

Сборка профиля

```
./BIQube.Cli data-and-model assembly-profile --profile-id 3 --session-id 15 --configuration-file "C:\BIQube.Cli\Configuration\config.ini"

 15  26.07.2024 13:14:50
```

Команда assembly-satellite

Параметры:

- -s, --session-id – идентификатор сессии (не обязательное)
- -s, --satellite-id – идентификатор satellite
- -f, --date-from – дата С для инкрементальной загрузки (не обязательное)
- -t, --date-to – дата По для инкрементальной загрузки (не обязательное)
- --filter – фильтр (не обязательное)

Пример:

Сборка satellite

```
./BIQube.Cli data-and-model assembly-satellite --satellite-id 3 --session-id 15 --configuration-file "C:\BIQube.Cli\Configuration\config.ini"

 15  26.07.2024 13:14:50
```

Команда assembly-business-view

Параметры:

- -s, --session-id – идентификатор сессии (не обязательное)
- -s, --business-view-id – идентификатор представления
- --filter – фильтр (не обязательное)

Пример:

Сборка представления

```
./BIQube.Cli data-and-model assembly-business-view --business-view-id 3 --session-id 15 --configuration-file "C:\BIQube.Cli\Configuration\config.ini"

 15  26.07.2024 13:14:50
```

Команда assembly-link

Параметры:

- -s, --session-id – идентификатор сессии (не обязательное)
- -s, --link-id – идентификатор ссылки
- -f, --date-from – дата С для инкрементальной загрузки (не обязательное)
- -t, --date-to – дата По для инкрементальной загрузки (не обязательное)
- --filter – фильтр (не обязательное)

Пример:

Сборка ссылки

```
./BIQube.Cli data-and-model assembly-link --link-id 3 --session-id 15 --configuration-file "C:\BIQube.
Cli\Configuration\config.ini"

 15  26.07.2024 13:14:50
```

Команда assembly-fact

Параметры:

- -fact, --fact-id – идентификатор факта
- -s, --session-id – идентификатор сессии (не обязательное)
- -f, --date-from – дата С для инкрементальной загрузки (не обязательное)
- -t, --date-to – дата По для инкрементальной загрузки (не обязательное)
- -p, --parameters – словарь наименований параметров и их значений (ключ=значение)

Команда assembly-hub

Параметры:

- -s, --session-id – идентификатор сессии (не обязательное)
- -s, --hub-id – идентификатор хаба
- -f, --date-from – дата С для инкрементальной загрузки (не обязательное)
- -t, --date-to – дата По для инкрементальной загрузки (не обязательное)
- --filter – фильтр (не обязательное)

Пример:

Сборка хаба

```
./BIQube.Cli data-and-model assembly-hub --hub-id 3 --session-id 15 --configuration-file "C:\BIQube.
Cli\Configuration\config.ini"

 15  26.07.2024 13:14:50
```