# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ
# BI.QUBE
# METASTAGING

Компонент разработан ООО «БИАЙ КУБ»
Приказ № _____ от «__»_____ 20___ г.
Акт сдачи № _____ от «__»_____ 20___ г.

Москва, 2023

# ОГЛАВЛЕНИЕ

# ВВЕДЕНИЕ

Компонент MetaStaging позволяет консолидировать в стейджинговом слое хранилища данные из гетерогенных источников с поддержанием целостности и унифицированности метаданных, также уменьшает нагрузку на операционные базы при выполнении запросов, а 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.

В документе приведено описание компонента и принципы работы с ним. Рассмотрены примеры загрузки данных с помощью компонента из разных источников.

Изучение данного документа позволит понять принцип работы компонента.

# ГЛОССАРИЙ

| | | |
|---|---|---|
| 1. | MetaStaging - BI.Qube | Инструмент, предназначенный для транспортировки данных. |
| 2. | Хранимая процедура | Объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере |
| 3. | Представление | Виртуальная таблица, содержимое которой определяется запросом |
| 4. | Бизнес-представление | Представление, в котором собраны Hub, Satellite и Link для сущности |
| 5. | Материализация | Процесс сохранения результата запроса бизнес-представления в таблицу для ускорения выборки. |
| 6. | Инкрементальная загрузка (загрузка с параметрами) | Регулярная загрузка данных в Greenplum. Извлекаются актуальные данные с даты последней загрузки. В таблице stg.session базы данных settings.db можно отследить историю всех загрузок. |
| 7. | Полная загрузка (снэпшоты) | Загрузка данных в Greenplum без параметризации. Применяется, когда необходима полная перезагрузка всех данных в таблице на источнике (например, при отсутствии столбца, подходящего для секционриования). |
| 8. | Полная загрузка с сохранением истории | Загрузка данных в Greenplum без параметризации. Но представления на Greenplum перенацеливаются на новые Parquet-файлы, а старые не удаляются из S3. |
| 9. | Профиль | Добавляются в таблице stg.profile |

| | | |
|---|---|---|
| 10. | Экстрактор | Компонент системы для извлечения данных из источников в S3. Исполняемый файл находится в директории LoadingToS3. Вызывается в Airflow в соответствии с командами в настроечной БД (settings.db). |
| 11. | External Table (ET) | Вид таблицы в Greenplum, обеспечивающий доступ к внешним источникам данных, как к объекту самой БД Greenplum. В системе используется для получения доступа к файлам в S3. Используется фреймворк PXF. |
| 12. | Сервисные процедуры | Процедуры, вызываемые автоматически в процессе работы компонента. |

# 1. ЦЕЛИ И НАЗНАЧЕНИЕ METASTAGING

Цель MetaStaging – обеспечить транспортировку данных из систем источников в файловое S3-совместмое хранилище данных (HDFS, ObjectStorage) с автоматической генерацией в СУБД Greenplum объектов типа «представление» на каждый полученный файл хранилищем.

Компонент MetaStaging, предназначен для передачи данных из различных источников, как правило, из учетных систем в целевое корпоративное хранилище данных (КХД) с поддержкой целостности метаданных систем-источников, при формировании промежуточного физического слоя хранения учитываются особенности целевой платформы.

Компонент MetaStaging входит в состав системы BI.Qube и может эксплуатироваться как отдельный компонент, так и в составе системы, так и под управлением компонента MetaOrchestrator, в такой конфигурации использование компонента является наиболее эффективной.

# 2. УСТАНОВКА И ЗАПУСК

Компонент MetaStaging для развертывания, функционирования и настройки использует различные программные инструменты и фреймворки. Обязательным условием является наличие у них открытого исходного кода.

Поддерживаемые операционные системы: Linux (различные дистрибутивы, такие как Ubuntu, Mint, РЕД ОС), другие Unix-подобные системы, а также есть возможность развернуть компонент под Windows.

Настроечные данные компонента могут хранится посредством СУБД: PostgreSQL (9.0 и позднее) / Postgres Pro (10.22 и позднее) / Arenadata Postgres (ADPG) (14.2.1) / Greenplum на выбор заказчика.

Для тестирования корректности загрузки данных в S3 хранилище с помощью файлов «.parquet» использовались инструменты s3-browser и parquet-viewer.

Инструменты разработки DBeaver, Visual Studio Code

Среды выполнения Python, «.Net Core».

В качестве библиотек для взаимодействия с системами источниками и назначениями, а также для обеспечения интеграции данных используются:

- AWSSDK.S3 - Amazon Simple Storage Service для Amazon S3 (nuget.org)
- CommandLineParser - Terse syntax C# command line parser for .NET.
- ExcelDataReader - Lightweight and fast library written in C# for reading Microsoft Excel files
- Google.Cloud.BigQuery.V2 - Recommended Google client library to access the BigQuery API.
- Microsoft.Data.SqlClient - Provides the data provider for SQL Server.

- MySql.Data
- Newtonsoft.Json - Json.NET high-performance JSON framework for .NET
- Npgsql - is the open source .NET data provider for PostgreSQL.
- ParquetSharp -.NET library for reading and writing Parquet files.
- YandexDisk.Client - .NET library wrapper of Yandex Desktop RestAPI.
- Встроенные модуле из стандартной библиотеки Python.
- Psycopg2 – PostgreSQL database adapter for the Python programming language.

В связи с высокой сложностью развертывания компонента в среде целевой СУБД установку компонента осуществляет вендор.

# 3. ИСТОЧНИКИ META STAGING

В компоненте MetaStaging реализована поддержка следующих источников:

- Rest API
- SQL Server
- PostgreSQL
- MySQL
- Excel (файлы, находящиеся в YandexDisk)

В таблице ниже приведены поддерживаемые типы данных на стороне источников данных.

| Источник | Поддерживаемые типы | Неподдерживаемые типы |
|---|---|---|
| SQL Server | tinyint, smallint, int, bigint, smallmoney, money, decimal, numeric, bit, real, float, date, time, smalldatetime, datetimeoffset, datetime, datetime2, char, varchar, nchar, nvarchar, text, ntext, binary, varbinary, image, uniqueidentifier, xml | geometry, geography |
| PostgreSQL | bigint, bigserial, bit varying, boolean, box, bytea, character varying, character, cidr, circle, date, double precision, inet, integer, interval, line, lseg, macaddr, money, numeric, path, point, polygon, real, smallint, text, time without time zone, time with time zone, timestamp without time zone, timestamp with time zone, uuid, xml, json, jsonb | составные типы, диапазонные типы (int8range, datarange), enum,s array, tsquery, tsvector, txid_snapshot |
| MySQL | all | enum (x,y …), set |
| Rest API, Excel | all | |

# 4. АРХИТЕКТУРА METASTAGING

## 4.1. Описание компонентов системы

Принцип работы MetaStaging сводится к взаимодействию программных блоков, которые отображены на рисунке ниже.

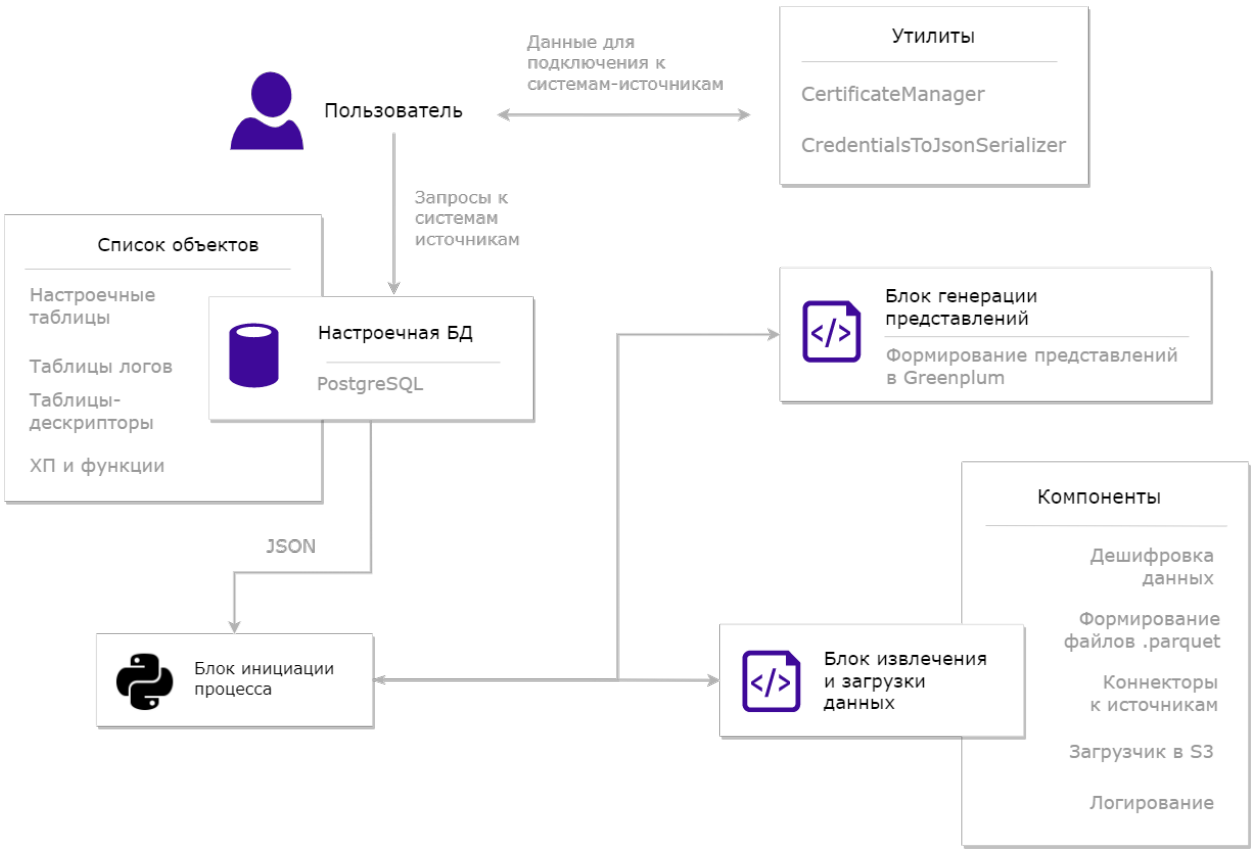



Рисунок 4. Компоненты системы MetaStaging

Краткое описание и назначение основных блоков компонента MetaStaging:

- Блок инициации процесса. Представляет собой Python3-скрипт и отвечает за запуск и координацию остальных блоков для интеграции данных.
- Блок извлечения и загрузки данных (Экстрактор). Представляет собой сборку «.Net Core». Загрузка может осуществляться в S3-совместимое хранилище в файлы «.Parquet».
- Блок генерации представлений. Отвечает за генерацию External tables и представлений в Greenplum, поддерживающих метаданные источников.
- Настроечная БД. Хранит информацию, необходимую для загрузки данных. Также служит интерфейсом для взаимодействия пользователя с MetaStaging. (*см. п. Структура Базы данных*).
- Утилиты. Предназначены для упрощения процесса заполнения настроечных таблиц
    - CertificateManager (Утилита шифрования) (*см. п. Утилита шифрования*).
    - CredentialsToJsonSerializer (Генератор Json для credentials источника) (*см. п. Генерация JSON для поля Credentials в таблице stg.source*).

### *4.1.1. Утилита шифрования*

Шифрование ключей, паролей, строк подключения производится в ручном режиме с помощью программы CertificateManager. Программа расположена в директории */home/itpro_admin/CertificateManager/.* Публичный и приватные ключи находятся в директории */home/itpro_admin/keytabs/*

```
itpro_mgmt@prd-bietl-01:~/CertificateManager$ ./CertificateManager --help
CertificateManager 1.0.0
Copyright (C) 2022 CertificateManager

  encrypt     Зашифровать текст

  decrypt     Расшифровка текста

  generate    Генерация двух ключей

  help        Display more information on a specific command.

  version     Display version information.
```

Рисунок 5. Пример применения утилиты

Пример команды для шифрования текста:

*~/CertificateManager$ ./CertificateManager encrypt --public-key ../keytabs/public.crt --text "text to encrypt"*

```
itpro_mgmt@prd-bietl-01:~/CertificateManager$ ./CertificateManager encrypt --public-key ../keytabs/public.crt --text "te
xt to encrypt"
Зашифрованный текст: 'fePw4my/Ru8Iec0yda4SsAGU1BDSgsBk9obxtVbOASkHRAQP+oRyYAC4qf3Up7LiXbzhIEvOuZwOd0gmn/z+NLuNKllP4mm2w1
9at+HugxJ7AZkGM0LDyaj9NBo5DbaC6UnH621zrpvyyTGAyRR00Q+zvf0ygtPzEnGkPl5CvzT2b1H0ju5VCYdwFzhNVKCPngzy/wqtlNBpC632Oh7UhrReXN
bdUuDJyPmVJnpGwqx/k7T0oXSaPE618VKAwaQKolPmn5lvr7X9M+MPFqGSpP9e73I62EZBu0kXaPv+Vzt0j+i4dqbaNKJZCK7xlmF4xIFSXeOd0FbNYeNWxl
1HkiLF7q7pOXy97o5lxgQJw3WKhSzwIY+xUNtOrStmg7sKptEfuXi+1lHV9j+eg5Wn6j9crNAnKPGh+9UgX1gfZnezlb9mxRVrYzeG/QOZ405gmy2AAKih2+
5Q6ycwtGv3kKVarcU6U5wfDZFyYnsRLQ0OecYRR2BQn+1tIBtCdc4pgZQfwfW/iW/namLq5y07NykFgu0fdqDGOSEHXfRcfHc9BPg8R0EBLc5+CnthW4wA90
6ESKGWCdq6qizbyejAw26o8WF/hWDJS2MyfICNwHIhiRl3GFpEYOBHkRiz9LREdlSKbFk0Z5HNSgNgXaWR++xiVNMCZ2FCA1lgnIgxZ58='
```

Рисунок 6. Пример применения утилиты

Пример команды для расшифровки текста:

*~/CertificateManager$ ./CertificateManager decrypt --private-key ../keytabs/private.key --text "text"*

```
itpro_mgmt@prd-bietl-01:~/CertificateManager$ ./CertificateManager decrypt --private-key ../keytabs/private.key --text "
fePw4my/Ru8Iec0yda4SsAGU1BDSgsBk9obxtVbOASkHRAQP+oRyYAC4qf3Up7LiXbzhIEvOuZwOd0gmn/z+NLuNKllP4mm2w19at+HugxJ7AZkGM0LDyaj9
NBo5DbaC6UnH621zrpvyyTGAyRR00Q+zvf0ygtPzEnGkPl5CvzT2b1H0ju5VCYdwFzhNVKCPngzy/wqtlNBpC632Oh7UhrReXNbdUuDJyPmVJnpGwqx/k7T0
oXSaPE618VKAwaQKolPmn5lvr7X9M+MPFqGSpP9e73I62EZBu0kXaPv+Vzt0j+i4dqbaNKJZCK7xlmF4xIFSXeOd0FbNYeNWxl1HkiLF7q7pOXy97o5lxgQJ
w3WKhSzwIY+xUNtOrStmg7sKptEfuXi+1lHV9j+eg5Wn6j9crNAnKPGh+9UgX1gfZnezlb9mxRVrYzeG/QOZ405gmy2AAKih2+5Q6ycwtGv3kKVarcU6U5wf
DZFyYnsRLQ0OecYRR2BQn+1tIBtCdc4pgZQfwfW/iW/namLq5y07NykFgu0fdqDGOSEHXfRcfHc9BPg8R0EBLc5+CnthW4wA906ESKGWCdq6qizbyejAw26o
8WF/hWDJS2MyfICNwHIhiRl3GFpEYOBHkRiz9LREdlSKbFk0Z5HNSgNgXaWR++xiVNMCZ2FCA1lgnIgxZ58="
Расшифрованный текст: 'text to encrypt'
```

Рисунок 7. Пример применения утилиты

Данные, которые будут зашифрованы утилитой, помещаются в настроечную БД. В экстракторе они дешифруются и используются для подключения к внешним системам.

Шифруются следующая информация:

- Поля key и secret в таблице *stg.subscription* для подключения к S3-хранилищу.
- Отдельные элементы JSON в поле credentials таблицы *stg.source*.

- **Для реляционных источников** (PostgreSQL, MySQL, SQL Server) атрибутами являются ConnectionString и ConnectionStringSecure. В 1 передается нешифрованная часть строки подключения, во 2 – шифрованная.

Например, если необходимо зашифровать только пароль, выполняется следующая команда:

*~/CertificateManager$ ./CertificateManager encrypt --public-key ../keytabs/public.crt --text "**Password=iii435iaf2Ma**"*

Результат передается в элемент ***ConnectionStringSecure*** для поля credentials.

```
"{
    "ConnectionString":"Server=192.168.72.109;Database=Tests;User
        Id=itpro_admin;TrustServerCertificate=true;",
    "ConnectionStringSecure": "<Результат выполнения утилиты>"
}"
```

- **Для других источников** обязательно прописывать флаг, указывающий зашифрован ли атрибут.

Пример заполнения поля credentials для RestAPI (можно зашифровать логин и пароль отдельно):

```
"{
    "User":"user@itprocomp.ru",
    "UserEncryption":false,
    "Password":"<Результат выполнения утилиты>",
    "PasswordEncryption":true,
    "AuthType":1
}"
```

### *4.1.2. Генерация JSON для поля Credentials в таблице stg.source*

Для подключения к источникам и указания учетных данных (логинов, паролей, строк подключения) необходимо заполнить поле Credentials таблицы sourse.

Для упрощения процесса заполнения данного поля необходим воспользоваться программой CredentialsToJsonSerializer, предварительно задав параметры, специфичные для источника. Программа расположена в директории */home/itpro_admin /CredentialsToJsonSerializer/.*

```
itpro_mgmt@prd-bietl-01:~/CredentialsToJsonSerializer$ ./CredentialsToJsonSerializer --help
CredentialsToJsonSerializer 1.0.0
Copyright (C) 2022 CredentialsToJsonSerializer

  sqlserver          SqlServer

  postgresql         PostgreSql

  mysql              MySql

  onedriveexcel      OneDriveExcel

  restapi            RestApi

  yandexdiskexcel    YandexDiskExcel

  bigquery           BigQuery

  help               Display more information on a specific command.

  version            Display version information.

itpro_mgmt@prd-bietl-01:~/CredentialsToJsonSerializer$
```

Рисунок 8. Пример использования программы

Данная программа преобразует текст в формат, который необходим для успешной загрузки из источника. При помощи ключа --help можно получить дополнительную информацию по любому источнику:

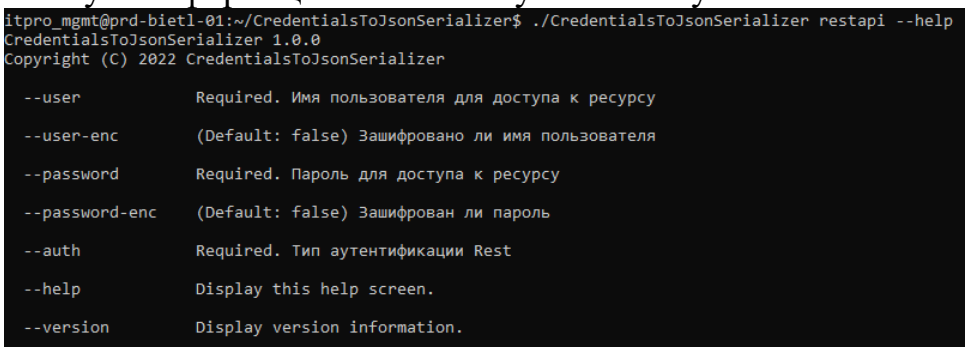



Рисунок 9. Пример использования программы

Пример команды для источника restapi:

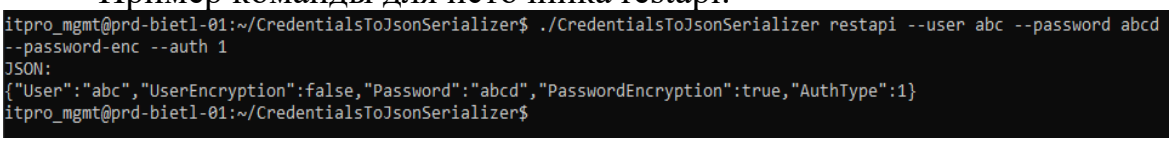



Рисунок 1. Пример использования программы

*~/CredentialsToJsonSerializer$ ./CredentialsToJsonSerializer restapi --user abc --password abcd --password-enc --auth 1*

### *4.1.3. Экстрактор*

При вызове блока инициации процесса (python-скрипт, описанный в начале главы) из БД settings автоматически подтягиваются все включенные запросы для всех включенных источников. На основе этого списка генерируются вызовы исполняемого файла экстрактора *GetFromSourceToParquetConsole*.

Путь к файлу – */home/itpro_admint/LoadingToS3.*

Таким образом, для пользователя нет необходимости взаимодействовать с этим компонентом, вся нагрузка лежит на блоке инициации процесса или на специализированном оркестраторе (например, MetaOrchestrator, входит в состав системы BI.Qube). В некоторых случаях может потребоваться запустить вручную данный файл, например, чтобы протестировать запрос отдельно. Аналогично предыдущим компонентам работает --help в командной строке.

Перечислим параметры, которые необходимо передать скрипту для взаимодействия с источниками и назначениями:

- Source – необходим для указания модулю типа источника, из которого извлекаются данные. *Поле name в таблице stg.source_type.*.
- BatchSize – количество записей при пакетной загрузке данных. *Поле batch_size в таблице stg.command.*
- Command – запрос на получение данных к источнику. Конструкция запроса предполагает SQL-подобную инструкцию Select с

возможностью определения полей и фильтров. *Поле command в таблице stg.command.*

- Credentials – реквизиты для подключения к источнику в формате JSON. Опционально реквизиты можно хранить в зашифрованном виде. *Поле credentials в таблице stg.source.*
- FileName – путь к файлу Parquet в S3-хранилище для записи данных из источников. *Поле sink_filename в таблице stg.command.*
- Key, Secret, Region, BucketName, Address – прочие параметры, специфичные для загрузки в S3 object storage. *Таблицы subscription и bucket.*
- Metadata-json-path – путь к файлам JSON с версиями метаданных запросов.

### *4.1.4. Формирование слоя хранения данных в Greenplum*

Для этой задачи разработан блок генерации внешних таблиц и представлений. Внешние таблицы (external table, ET) позволяют обращаться к файлам формата Parquet как к объектам БД. Представления (view) позволяют поддержать оригинальные наименования и типы данных источников.

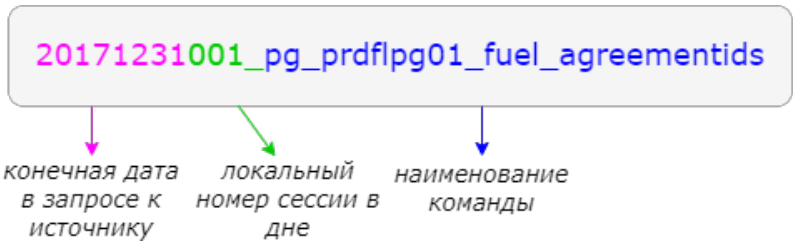
Генератор вызывается на последнем этапе пайплайна (блок инициации процесса). Сборка расположена в директории */home/itpro_admin/DbEntitiesGenerator/*.

В ходе работы генератора на каждый файл в S3-совместимом хранилище создаются внешние таблицы в схеме *"back"*. Если таблицы уже существуют, то пересоздания не происходит.

Загрузка этих таблиц может быть 3 типов (*подробнее в главе 5*):

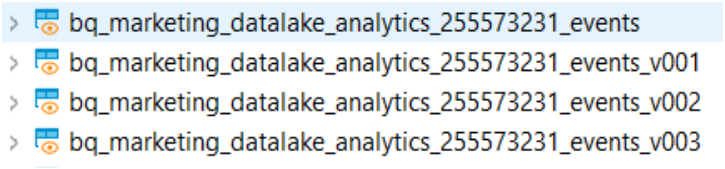
1. инкрементальная,
2. снэпшоты (полная загрузка),
3. снэпшоты (полная загрузка) с сохранением истории.

Каждый запрос из инкрементальной загрузки представлен в Greenplum перечнем ET (1 загрузка в parquet = 1 ET). Наименование для ET состоит из следующих частей:




Поверх ET формируются представления 2 типов:

1. представление, включающее ET, удовлетворяющие последней версии метаданных (к названию добавляется постфикс, указывающий на номер версии "v003");
2. общее представление, включает в себя первую версию данных. Далее дополняется вручную пользователем.




Аналогичный сценарий с версиями используется для снэпшотов. Файлы Parquet либо перезаписывается, либо обновляется с сохранением истории в S3

Пример SQL-запроса для генерации внешней таблицы:

```
CREATE EXTERNAL TABLE
monopolysun.public.20221104002_MS_DocumentsInsurance (
id text,
Sum numeric
)
LOCATION (
'pxf://monopoly-sun-
temp/incremental/*/*/*/DocumentsInsurance_*.parquet?PROFILE=s3:parquet&accesskey=YCAJEcQEeeSmEXJ2wUXJK_NyO&secretkey=YCO9wLy5_O8C9mAzA3CgcV4kXnQrJCdd6ZLklp4&endpoint=storage.yandexcloud.net&SERVER=storage'
) ON ALL
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import')
        ENCODING 'UTF8';
```

Пример SQL-запроса для генерации представлений:

```
CREATE OR REPLACE VIEW public.pg_prdflpg01_fuel_fuelsupplyschemes_v001
AS SELECT q.id,
          q.updatedat,
        q.filledtype,
          q.servicemethod
   FROM
   (
     SELECT
          "20171231001_pg_prdflpg01_fuel_fuelsupplyschemes"."Id"::uuid AS id,
          to_timestamp(("20171231001_pg_prdflpg01_fuel_fuelsupplyschemes"."Updated
          At"/ 1000000)::double precision)::timestamp without time zone AS updatedat,
          "20171231001_pg_prdflpg01_fuel_fuelsupplyschemes"."FilledType"          AS
filledtype,
          "20171231001_pg_prdflpg01_fuel_fuelsupplyschemes"."ServiceMethod"       AS
servicemethod

     FROM back."20171231001_pg_prdflpg01_fuel_fuelsupplyschemes"
        ) q
WHERE
   q.updatedat >= '1900-01-01 00:00:00'::timestamp without time zone
   AND
   q.updatedat < '2017-12-31 21:00:00'::timestamp without time zone;
```

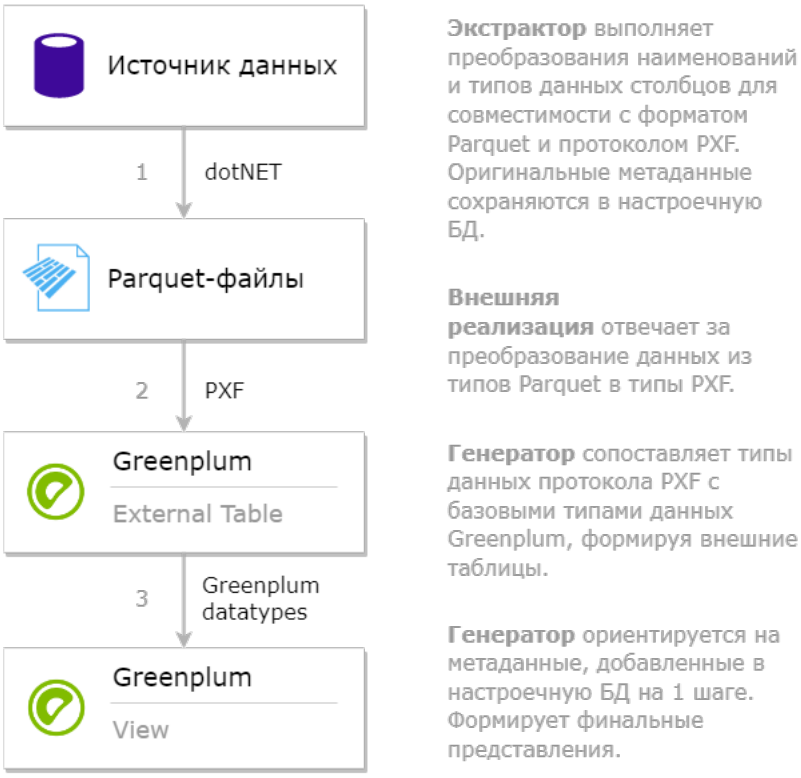



Рисунок 2. Алгоритм формирования слоя в Greenplum

Условие “WHERE” в данном случае помогает оптимизатору Greenplum не сканировать все ET, когда нужно взять из одной конкретной ET. Чтобы данное условие было добавлено необходимо заполнить partition_column.

На рисунке ниже представлен полный путь перекладки данных из источников в назначение.

Внешние таблицы ссылаются на файлы с помощью протокола PXF, рекомендуемого для чтения из S3-хранилища в документации Yandex-Cloud. Список поддерживаемых типов данных PXF и сопоставление с Greenplum можно посмотреть здесь:

Reading and Writing HDFS Parquet Data | Pivotal Greenplum Docs

## 4.2. Структура проекта MetaStaging в файловой системе

Путь к файлам MetaStaging на Windows и на Linux отличается.

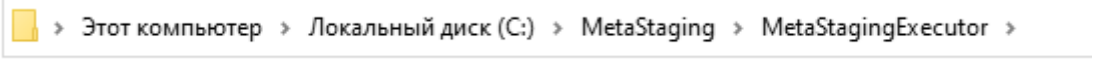



Рисунок 3. Путь в файловой системе Windows

Файлы MetaStaging на Linux находятся в папке home пользователя itpro_admin.

В общем виде структура проекта выглядит следующим образом:

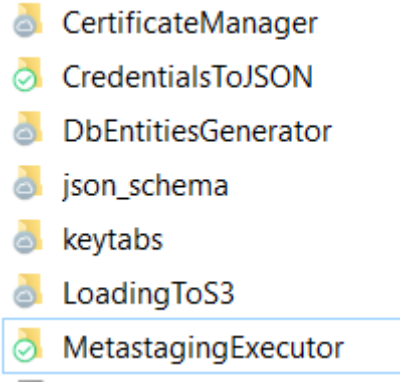



Рисунок 4.Файловая система компонента

CertificateManager – это «Утилита шифрования», описанная в предыдущем разделе.

CredentialsToJSON – содержит сборку утилиты «Генерация JSON для поля Credentials».

DbEntitiesGenerator – это сборка для «Формирования слоя хранения данных на Greenplum».

Json_schema – данный каталог содержит JSON-файлы с метаданными запросов из источника, которые выполнялись в рамках конкретных сессий. Эти данные записываются в БД в таблицу *stg.last_command_metadata.*

Keytabs – данный каталог содержит ключи для шифрования и расшифрования конфиденциальной информации. Каталог используется экстрактором (в файле app.config проекта указан путь к каталогу) и утилитой CertificateManager.

LoadingToS3 – содержит сборку с экстрактором.

MetastagingExecutor – содержит python-скрипт инициации процесса.

## 4.3. Структура Базы данных

Таблицы MetaStaging делятся на три категории:

- *Настроечные таблицы* – данные вносит пользователь в соответствии с правилами заполнения.
- *Таблицы логов (заполняются автоматически)* – данные вносит система по итогам выполнения очередной сессии.
- *Таблицы дескрипторы (заполнены разработчиком)* – данные внесены предварительно в соответствии с реализованным функционалом.

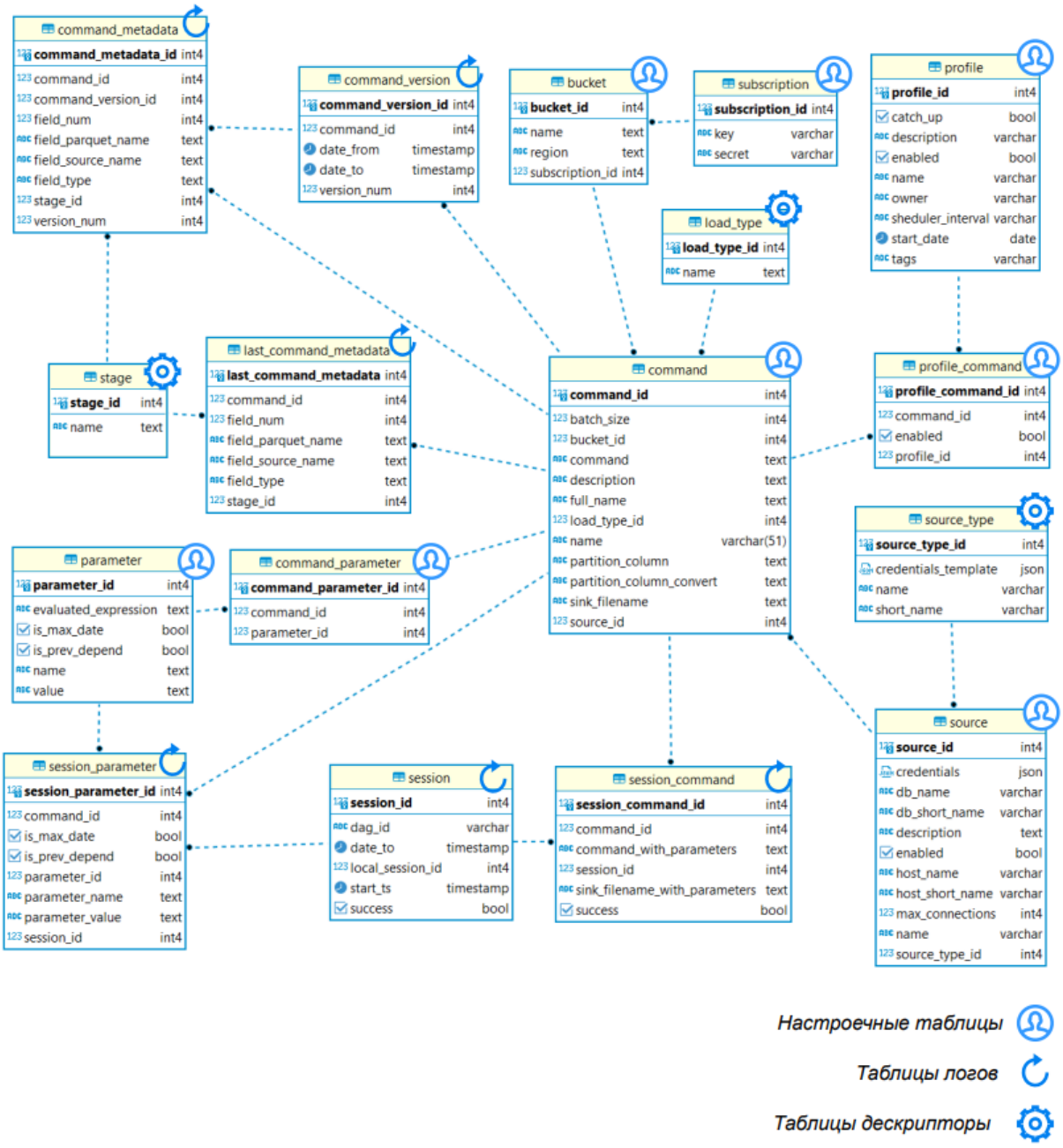



Рисунок 5.Структура таблиц MetaStaging

### *4.3.1. Настроечные таблицы MetaStaging*

1. **«stg.subscription»** – список ключей необходимых для доступа к сервисам S3. Можно записывать как в шифрованном, так и в незашифрованном виде.

**Поля таблицы:**

| Имя столбца | Тип данных | Источник | Назначение |
|---|---|---|---|
| subscription_id | int | Автоинкрементен | Идентификатор ключа для доступа к сервисам |
| key | varchar | Вручную | Ключ |
| secret | varchar | Вручную | Секретный ключ |
| address | text | Вручную | Адрес хоста с установленным S3 MinIO |

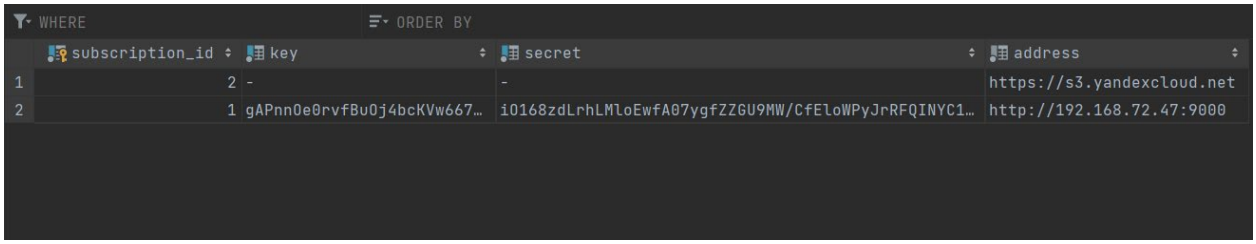



Рисунок 6. Таблица «stg.subscription»

2. **«stg.bucket»** – список каталогов S3-хранилища, доступных для размещения файлов parquet.

**Поля таблицы:**

| Имя столбца | Тип данных | Источник | Назначение |
|---|---|---|---|
| bucket_id | int | Автоинкримент | Идентификатор каталога |
| name | text | Вручную | Имя каталога |
| region | text | Вручную | Регион, в котором создаются бакеты (для MinIO не указывается) |
| subscription_id | int | Вручную | Идентификатор ключа для доступа к сервисам |

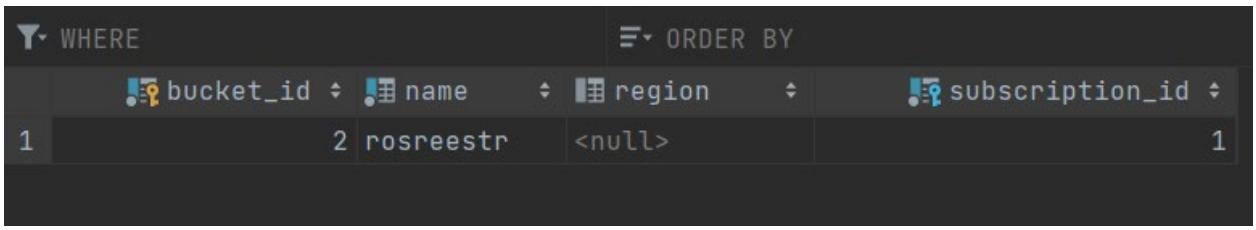



Рисунок 7. Таблица «stg.bucket»

3. **«stg.source»** – информация, необходимая системе для доступа к источникам данных.

**Поля таблицы:**

| Имя столбца | Тип данных | Источник | Назначение |
|---|---|---|---|
| sourse_id | int | Автоинкримент | Идентификатор источника |

| | | | |
|---|---|---|---|
| name | varchar | Вручную | Название источника, не влияет на загрузку данных |
| description | text | Вручную | Описание источника, необязательное поле |
| credentials | json | Вручную | Учетные данные для подключения определенного источника в формате JSON |
| enabled | bool | Вручную | Метка о необходимости выполнения/игнорирования всех запросов к данному источнику |
| max_connections | int | Вручную | Максимальное количество одновременных запросов на выполнение, разрешенных для источника |
| source_type_id | int | Вручную | Ссылка на тип источника |

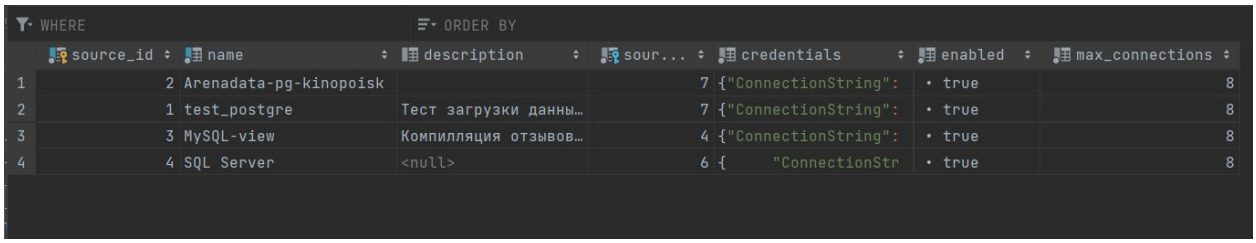
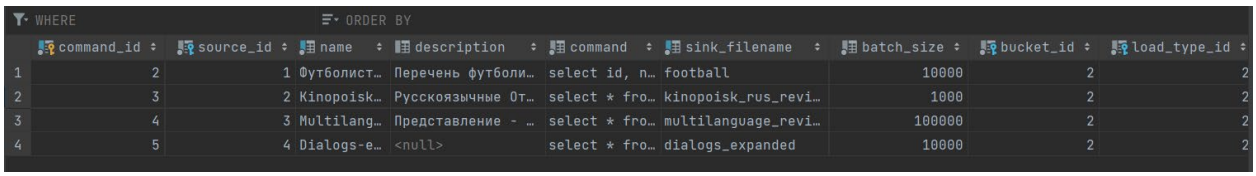



Рисунок 8. Таблица «stg.source»

4. **«stg.command»** – список запросов к источникам.

## Поля таблицы:

| Имя столбца | Тип данных | Источник | Назначение |
|---|---|---|---|
| command_id | int | Автоинкримент | Идентификатор команды |
| batch_size | int | Вручную | Количество строк выгружаемых из источников в файл Parquet за одну итерацию при пакетной загрузке |
| command | text | Вручную | Текст запроса с возможностью параметризации. Используется Select-подобный синтаксис, при работе экстрактора запрос корректируется в соответствии с требованиями источника |
| description | text | Вручную | Описание команды |
| full_name | text | Вручную | Полное наименование команды |
| name | varchar | Вручную | Наименование команды, длина до 51 символа! Влияет на наименование файла Parquet |
| sink_filename | text | Вручную | Путь к Parquet-файлу в S3-совместимом хранилище |
| source_id | int | Вручную | Ссылка на источник |
| partition_column | text | Вручную | 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 | Вручную | Поле содержит логику конвертации для значения в partition_column. Данная логика будет отражена в представлении на Greenplum. *Пример: cast(/*{partition_column}*/ as bigint)* |



| | command_id | source_id | name | description | command | sink_filename | batch_size | bucket_id | load_type_id |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 1 | Футболист… | Перечень футболи… | select id, n… | football | 10000 | 2 | 2 |
| 2 | 3 | 2 | Kinopoisk… | Русскоязычные От… | select * fro… | kinopoisk_rus_revi… | 1000 | 2 | 2 |
| 3 | 4 | 3 | Multilang… | Представление - … | select * fro… | multilanguage_revi… | 100000 | 2 | 2 |
| 4 | 5 | 4 | Dialogs-e… | <null> | select * fro… | dialogs_expanded | 10000 | 2 | 2 |

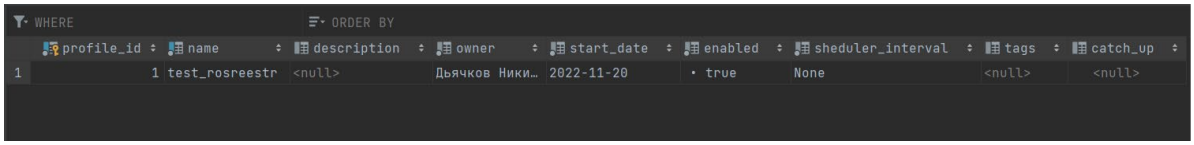
Рисунок 9. Таблица «stg. command»

5. **«stg.profile»** – список профилей загрузки. На основе данной таблицы (совместно с profile_command) запросы к источникам будет разделены на несколько DAG'ов в оркестраторе.

**Поля таблицы:**

| Имя столбца | Тип данных | Источник | Назначение |
|---|---|---|---|
| profile _id | int | Автоинкремент | Идентификатор профиля |
| name | varchar | Вручную | Название профиля |
| description | varchar | Вручную | Описание профиля |
| enabled | bool | Вручную | Метка о необходимости выполнения/ игнорирования данного профиля |
| owner | varchar | Вручную | Владелец графа выполнения *(используется только при наличии оркестратора)* |
| scheduler_interval | varchar | Вручную | Интервал запуска DAG'ов, может принимать в себя явное cron-выражение, алиас (@daily, @monthly и тд) или спец-слово None, что приведет к отключению автоматического планирования и запуска задач по расписанию *(используется только при наличии оркестратора)* |
| start_date | date | Вручную | Дата начала профиля |

○



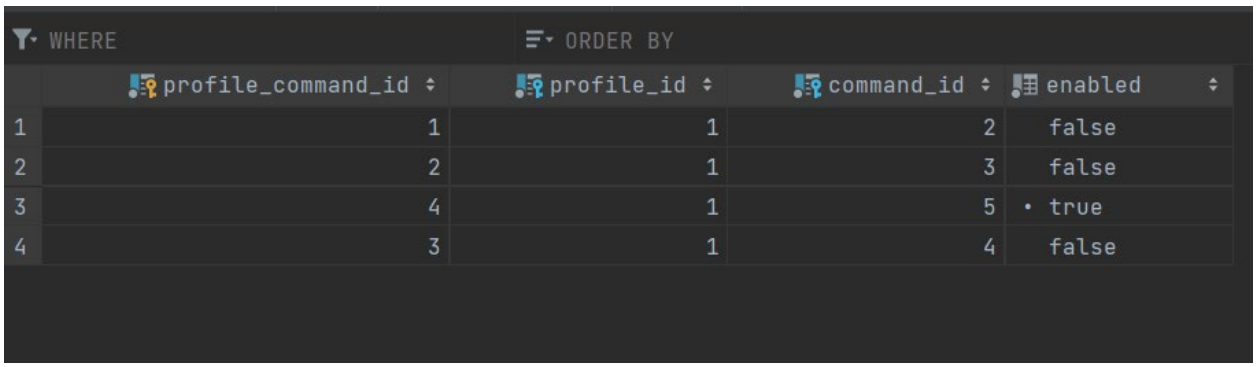
| | profile_id | name | description | owner | start_date | enabled | sheduler_interval | tags | catch_up |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | test_rosreestr | <null> | Дьячков Ники… | 2022-11-20 | true | None | <null> | <null> |

Рисунок 19. Таблица «stg.profile»

6. **«profile_command»** – соответствие выполняемых команд профилям загрузки. Одна команда может использоваться несколькими профилями, один профиль может выполнять несколько команд.

**Поля таблицы:**

| Имя столбца | Тип данных | Источник | Назначение |
|---|---|---|---|
| profile_command_id | int | Автоинкремент | Идентификатор профиля, команды |
| profile_id | int | Вручную | Идентификатор профиля |
| command_id | int | Вручную | Идентификатор команды |
| enabled | bool | Вручную | Метка о необходимости выполнения/игнорирования данного запроса |


WHERE ORDER BY

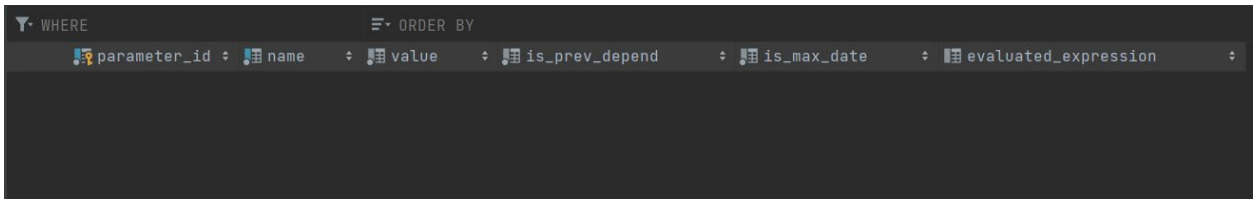
| | profile_command_id | profile_id | command_id | enabled |
|---|---|---|---|---|
| 1 | 1 | 1 | 2 | false |
| 2 | 2 | 1 | 3 | false |
| 3 | 4 | 1 | 5 | true |
| 4 | 3 | 1 | 4 | false |

Рисунок 10. Таблица «profile_command»

7. **«parameter»** – список параметров для использования в запросах к источникам.

**Поля таблицы:**

| Имя столбца | Тип данных | Источник | Назначение |
|---|---|---|---|
| parameter_id | int | Автоинкримент | Идентификатор параметра |
| name | text | Вручную | Наименование параметра, указывается в запросе |
| value | text | Вручную | Значение параметра, подставляется в запрос при работе экстрактора |
| evaluated_expression | text | Вручную | SQL-выражение, которое преобразует value к требуемому формату. |
| is_max_date | bool | Вручную | Указание экстрактору, ссылаться ли на value в запросе или не указывать в нем верхнюю границу (брать актуальные данные, начиная от конкретной даты) |
| is_prev_depend | text | Вручную | Указание экстрактору, ссылаться ли на value или использовать дату, зависящую от последней сессии |


WHERE ORDER BY

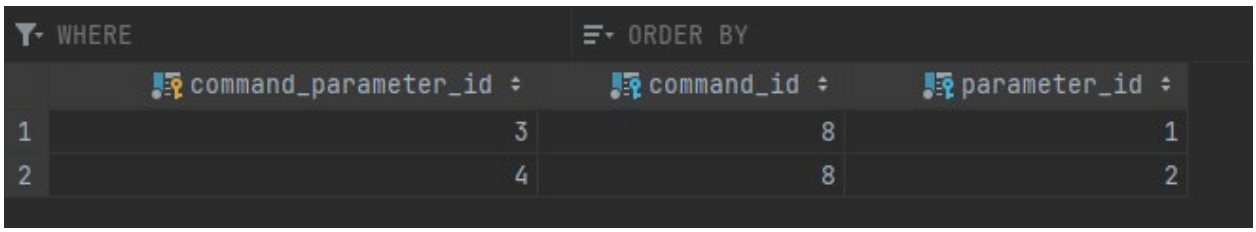
| parameter_id | name | value | is_prev_depend | is_max_date | evaluated_expression |
|---|---|---|---|---|---|

Рисунок 11. Таблица «parameter»

8. **command_parameter** – ссылки на таблицы command и parameter.

**Поля таблицы:**

| Имя столбца | Тип данных | Источник | Назначение |
|---|---|---|---|
| command_parameter_id | int | Автоинкремент | Идентификатор параметра, команды |
| command_id | int | Вручную | Идентификатор команды |
| parameter_id | int | Вручную | Идентификатор параметра |



WHERE ORDER BY

| | command_parameter_id | command_id | parameter_id |
|---|---|---|---|
| 1 | 3 | 8 | 1 |
| 2 | 4 | 8 | 2 |

Рисунок 12 Таблица «command_parameter»

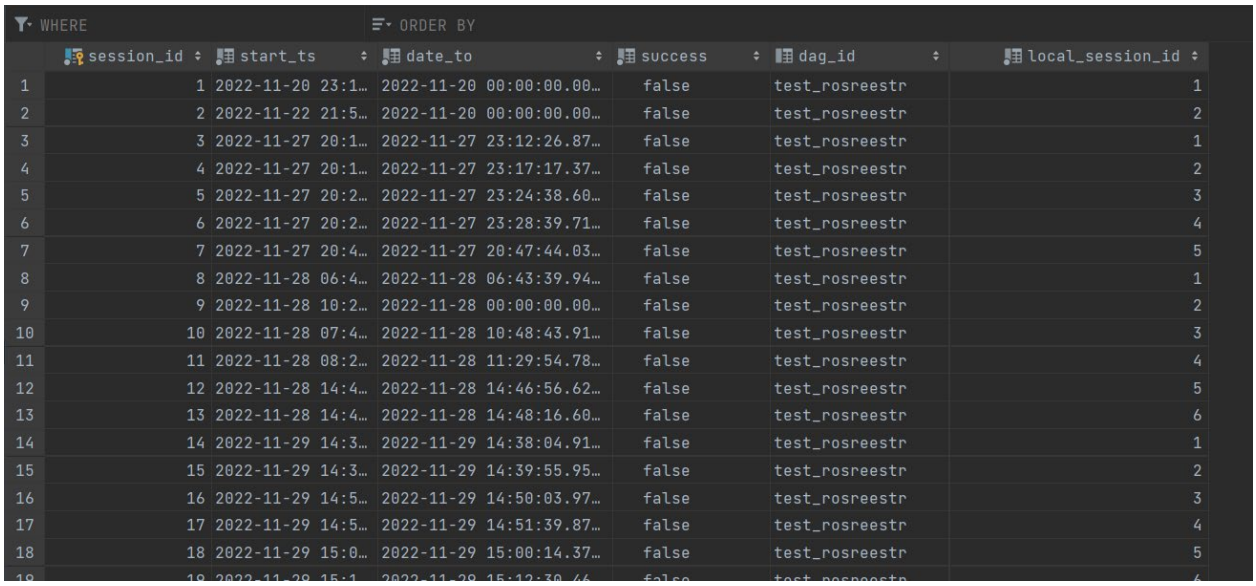
### *4.3.2. Таблицы логов MetaStaging*

Данные таблицы заполняются в процессе работы компонента

1. **«session»** – список запусков пайплайна.

**Поля таблицы:**

| Имя столбца | Тип данных | Источник | Назначение |
|---|---|---|---|
| sessionID | int | Автоинкремент | Идентификатор сессии |
| start_ts | timestamp | Автоматически | Дата и время запуска работы MetaStaging |
| date to | timestamp | Автоматически | Дата окончания сессии |
| success | bool | Автоматически | Идентификатор успешности загрузки в рамках данной сессии |
| dag_id | varchar | Автоматически | Наименования графа загрузки *(используется только при наличии оркестратора)* |
| local_session_id | int | Автоматически | Локальный идентификатор сессии относительно дня (когда запрос выполняется несколько раз за день) |



WHERE ORDER BY

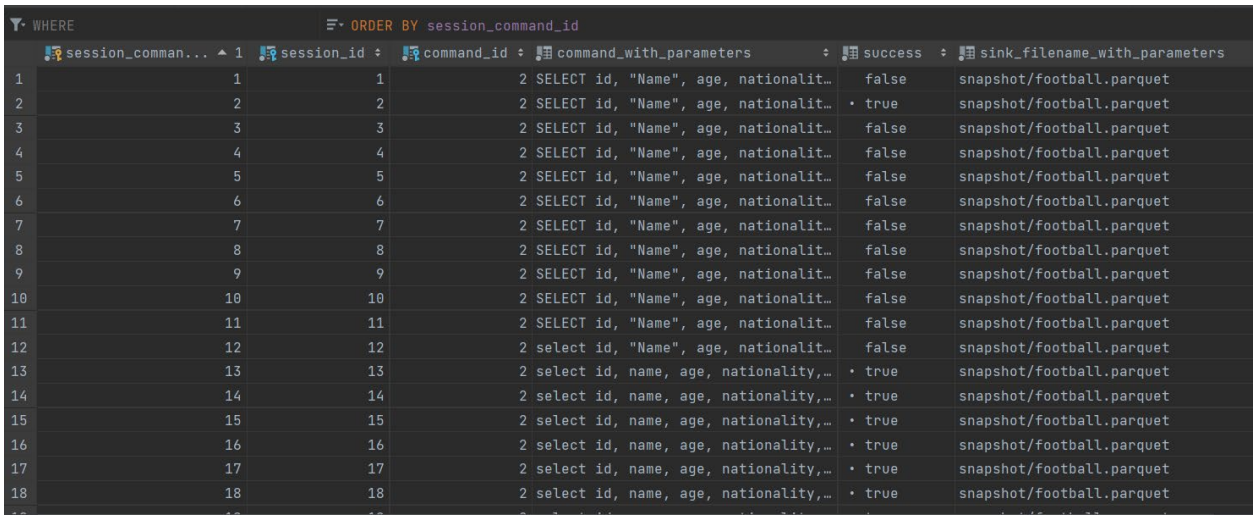
| | session_id | start_ts | date_to | success | dag_id | local_session_id |
|---|---|---|---|---|---|---|
| 1 | 1 | 2022-11-20 23:1… | 2022-11-20 00:00:00.00… | false | test_rosreestr | 1 |
| 2 | 2 | 2022-11-22 21:5… | 2022-11-20 00:00:00.00… | false | test_rosreestr | 2 |
| 3 | 3 | 2022-11-27 20:1… | 2022-11-27 23:12:26.87… | false | test_rosreestr | 1 |
| 4 | 4 | 2022-11-27 20:1… | 2022-11-27 23:17:17.37… | false | test_rosreestr | 2 |
| 5 | 5 | 2022-11-27 20:2… | 2022-11-27 23:24:38.60… | false | test_rosreestr | 3 |
| 6 | 6 | 2022-11-27 20:2… | 2022-11-27 23:28:39.71… | false | test_rosreestr | 4 |
| 7 | 7 | 2022-11-27 20:4… | 2022-11-27 20:47:44.03… | false | test_rosreestr | 5 |
| 8 | 8 | 2022-11-28 06:4… | 2022-11-28 06:43:39.94… | false | test_rosreestr | 1 |
| 9 | 9 | 2022-11-28 10:2… | 2022-11-28 00:00:00.00… | false | test_rosreestr | 2 |
| 10 | 10 | 2022-11-28 07:4… | 2022-11-28 10:48:43.91… | false | test_rosreestr | 3 |
| 11 | 11 | 2022-11-28 08:2… | 2022-11-28 11:29:54.78… | false | test_rosreestr | 4 |
| 12 | 12 | 2022-11-28 14:4… | 2022-11-28 14:46:56.62… | false | test_rosreestr | 5 |
| 13 | 13 | 2022-11-28 14:4… | 2022-11-28 14:48:16.60… | false | test_rosreestr | 6 |
| 14 | 14 | 2022-11-29 14:3… | 2022-11-29 14:38:04.91… | false | test_rosreestr | 1 |
| 15 | 15 | 2022-11-29 14:3… | 2022-11-29 14:39:55.95… | false | test_rosreestr | 2 |
| 16 | 16 | 2022-11-29 14:5… | 2022-11-29 14:50:03.97… | false | test_rosreestr | 3 |
| 17 | 17 | 2022-11-29 14:5… | 2022-11-29 14:51:39.87… | false | test_rosreestr | 4 |
| 18 | 18 | 2022-11-29 15:0… | 2022-11-29 15:00:14.37… | false | test_rosreestr | 5 |

Рисунок 13. Таблица «session»

2. **«session_command»** – список запросов для извлечения данных из источников в S3-хранилище. Запросы разделены на секции, в рамках которых выполнялись.

**Поля таблицы:**

| Имя столбца | Тип данных | Источник | Назначение |
|---|---|---|---|
| session_command ID | int | Автоинкремент | Идентификатор строки |
| commad_with_parameters | text | Автоматически | Окончательный текст запроса, отправляемого на источник данных |
| success | bool | Автоматически | Метка об успешном выполнении запроса |
| sink_filename_with_parameters | text | Автоматически | Полный путь к файлу parquet в рамках S3-совместимого хранилища. |
| session_ID | int | Автоматически | Идентификатор сессии |
| commad_ID | int | Автоматически | Идентификатор команды |



WHERE ORDER BY session_command_id

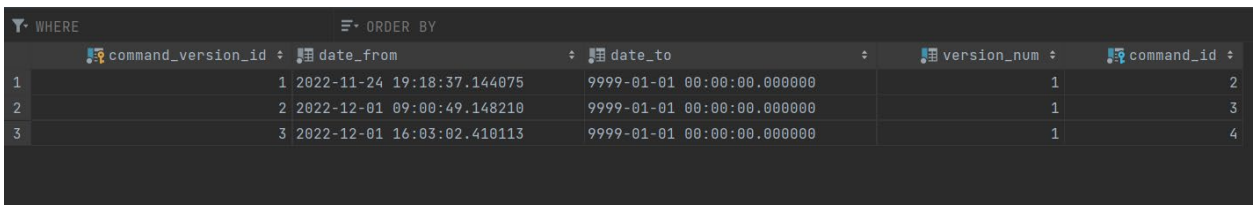
| | session_comman... | session_id | command_id | command_with_parameters | success | sink_filename_with_parameters |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 2 | SELECT id, "Name", age, nationalit… | false | snapshot/football.parquet |
| 2 | 2 | 2 | 2 | SELECT id, "Name", age, nationalit… | true | snapshot/football.parquet |
| 3 | 3 | 3 | 2 | SELECT id, "Name", age, nationalit… | false | snapshot/football.parquet |
| 4 | 4 | 4 | 2 | SELECT id, "Name", age, nationalit… | false | snapshot/football.parquet |
| 5 | 5 | 5 | 2 | SELECT id, "Name", age, nationalit… | false | snapshot/football.parquet |
| 6 | 6 | 6 | 2 | SELECT id, "Name", age, nationalit… | false | snapshot/football.parquet |
| 7 | 7 | 7 | 2 | SELECT id, "Name", age, nationalit… | false | snapshot/football.parquet |
| 8 | 8 | 8 | 2 | SELECT id, "Name", age, nationalit… | false | snapshot/football.parquet |
| 9 | 9 | 9 | 2 | SELECT id, "Name", age, nationalit… | false | snapshot/football.parquet |
| 10 | 10 | 10 | 2 | SELECT id, "Name", age, nationalit… | false | snapshot/football.parquet |
| 11 | 11 | 11 | 2 | SELECT id, "Name", age, nationalit… | false | snapshot/football.parquet |
| 12 | 12 | 12 | 2 | select id, "Name", age, nationalit… | false | snapshot/football.parquet |
| 13 | 13 | 13 | 2 | select id, name, age, nationality,… | true | snapshot/football.parquet |
| 14 | 14 | 14 | 2 | select id, name, age, nationality,… | true | snapshot/football.parquet |
| 15 | 15 | 15 | 2 | select id, name, age, nationality,… | true | snapshot/football.parquet |
| 16 | 16 | 16 | 2 | select id, name, age, nationality,… | true | snapshot/football.parquet |
| 17 | 17 | 17 | 2 | select id, name, age, nationality,… | true | snapshot/football.parquet |
| 18 | 18 | 18 | 2 | select id, name, age, nationality,… | true | snapshot/football.parquet |

Рисунок 14. Таблица «session_command»

**3. «command_version»** – список существующих версий для конкретных команд с указанием периода их действия.

**Поля таблицы:**

| Имя столбца | Тип данных | Источник | Назначение |
|---|---|---|---|
| command_version ID | int | Автоинкремент | Идентификатор версии |
| date_from | timestep | Автоматически | Фактическая дата начала загрузки |
| date_to | timestep | Автоматически | Фактическая дата окончания загрузки |
| version num | int | Автоматически | Номер версии запроса к источнику |
| command__id | int | Автоматически | Ссылка на команду |



WHERE ORDER BY

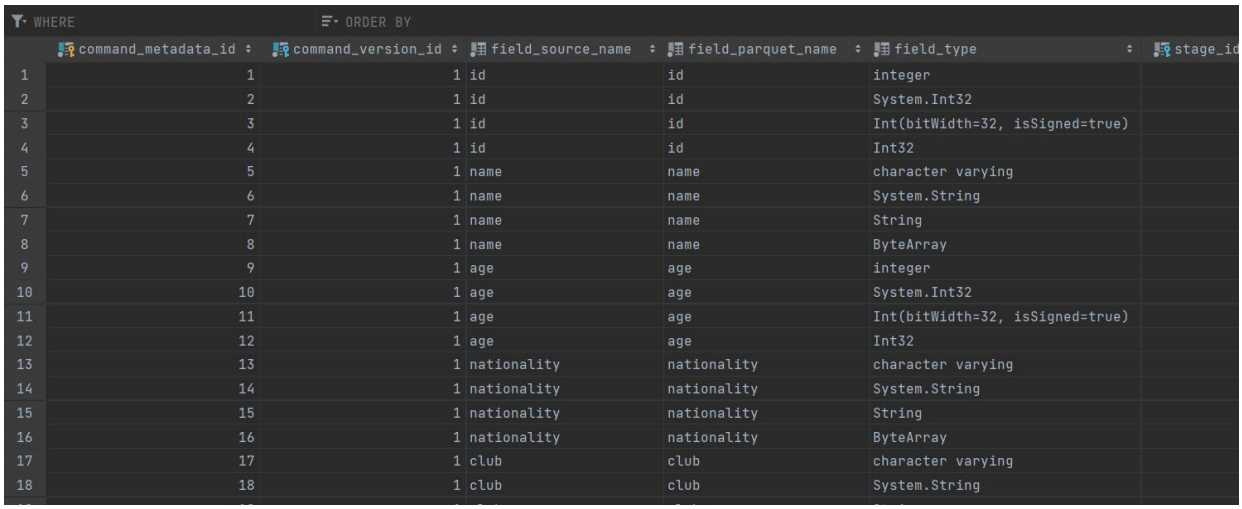
| | command_version_id | date_from | date_to | version_num | command_id |
|---|---|---|---|---|---|
| 1 | 1 | 2022-11-24 19:18:37.144075 | 9999-01-01 00:00:00.000000 | 1 | 2 |
| 2 | 2 | 2022-12-01 09:00:49.148210 | 9999-01-01 00:00:00.000000 | 1 | 3 |
| 3 | 3 | 2022-12-01 16:03:02.410113 | 9999-01-01 00:00:00.000000 | 1 | 4 |

Рисунок 15. Таблица «command_version»

**4. «command_metadata»** – список наименований и типов данных полей, включенных в запросы к источникам за всю историю выполнений этих запросов.

**Поля таблицы:**

| Имя столбца | Тип данных | Источник | Назначение |
|---|---|---|---|
| command_metadata ID | int | Автоинкремент | Идентификатор метаданных команды |
| command_version ID | int | Автоматически | Идентификатор команды, версии |
| field_source_name | text | Автоматически | Наименование поля в источнике |
| field_type | text | Автоматически | Тип данных поля на различных этапах интеграции |
| field_parquet_name | text | Автоматически | Наименование поля, допустимое для записи в Parquet и чтения в Greenplum через External Table (замена пробелов и специальных символов) |
| version_num | int | Автоматически | Версия метаданных запроса. По умолчанию равна 1. При изменении метаданных значение увеличивается |
| field_num | int | Автоматически | Номер поля в запросе к источнику, используется для сохранения порядка полей в Greenplum относительно порядка в источнике |



WHERE ORDER BY

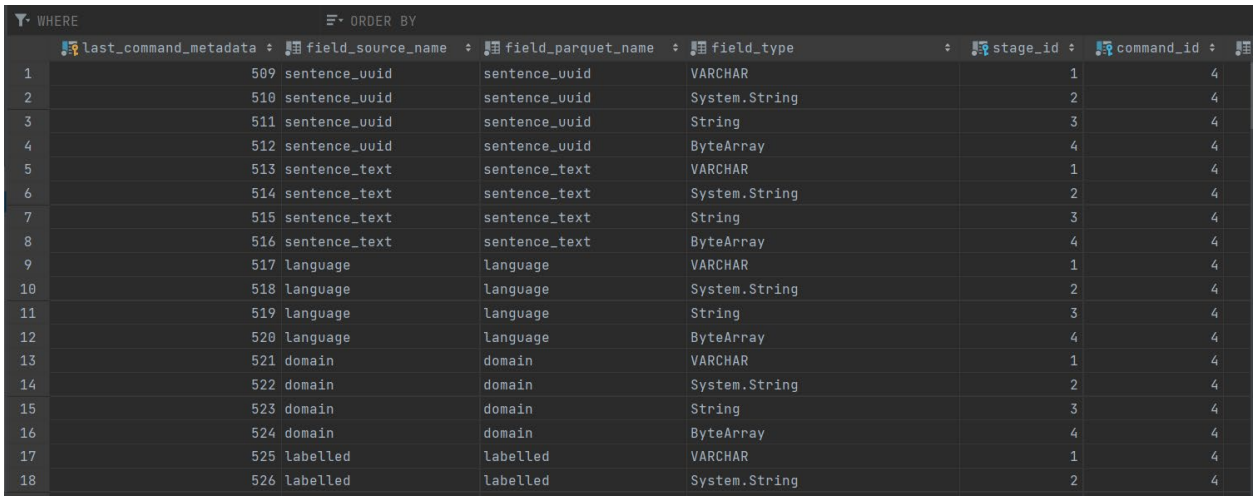
| | command_metadata_id | command_version_id | field_source_name | field_parquet_name | field_type | stage_id |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | id | id | integer | |
| 2 | 2 | 1 | id | id | System.Int32 | |
| 3 | 3 | 1 | id | id | Int(bitWidth=32, isSigned=true) | |
| 4 | 4 | 1 | id | id | Int32 | |
| 5 | 5 | 1 | name | name | character varying | |
| 6 | 6 | 1 | name | name | System.String | |
| 7 | 7 | 1 | name | name | String | |
| 8 | 8 | 1 | name | name | ByteArray | |
| 9 | 9 | 1 | age | age | integer | |
| 10 | 10 | 1 | age | age | System.Int32 | |
| 11 | 11 | 1 | age | age | Int(bitWidth=32, isSigned=true) | |
| 12 | 12 | 1 | age | age | Int32 | |
| 13 | 13 | 1 | nationality | nationality | character varying | |
| 14 | 14 | 1 | nationality | nationality | System.String | |
| 15 | 15 | 1 | nationality | nationality | String | |
| 16 | 16 | 1 | nationality | nationality | ByteArray | |
| 17 | 17 | 1 | club | club | character varying | |
| 18 | 18 | 1 | club | club | System.String | |

Рисунок 16. Таблица «command_metadata»

**5. «last_command_metadata»** – список наименований и типов данных полей при последнем выполнении конкретного запроса. Здесь всегда хранится актуальная версия метаданных. При формировании представлений на Greenplum данные из этой таблицы сравниваются с данными из таблицы *command_metadata* для проверки изменений (добавлено поле, изменен тип и тд).

**Поля таблицы:**

| Имя столбца | Тип данных | Источник | Назначение |
|---|---|---|---|
| **last_command_metadata_id** | int | Автоинкремент | Идентификатор строки |
| field_source_name | text | Автоматически | Наименование поля в источнике |
| field_type | text | Автоматически | Тип данных поля на этапах интеграции |
| field_parquet_name | text | Автоматически | Наименование поля, допустимое для записи в Parquet и чтения в Greenplum через External Table (замена пробелов и специальных символов) |
| field_num | int | Автоматически | Номер поля в запросе к источнику, используется для сохранения порядка полей в Greenplum относительно порядка в источнике |



WHERE ORDER BY

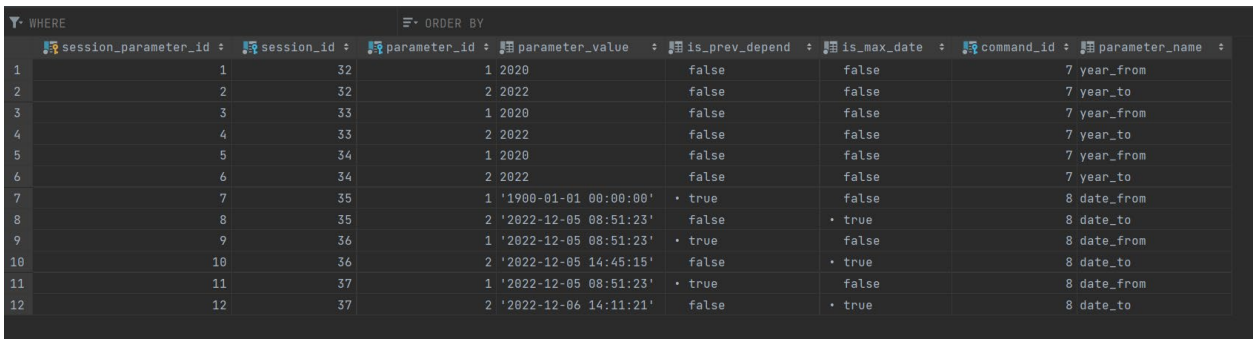
| | last_command_metadata | field_source_name | field_parquet_name | field_type | stage_id | command_id |
|---|---|---|---|---|---|---|
| 1 | 509 | sentence_uuid | sentence_uuid | VARCHAR | 1 | 4 |
| 2 | 510 | sentence_uuid | sentence_uuid | System.String | 2 | 4 |
| 3 | 511 | sentence_uuid | sentence_uuid | String | 3 | 4 |
| 4 | 512 | sentence_uuid | sentence_uuid | ByteArray | 4 | 4 |
| 5 | 513 | sentence_text | sentence_text | VARCHAR | 1 | 4 |
| 6 | 514 | sentence_text | sentence_text | System.String | 2 | 4 |
| 7 | 515 | sentence_text | sentence_text | String | 3 | 4 |
| 8 | 516 | sentence_text | sentence_text | ByteArray | 4 | 4 |
| 9 | 517 | language | language | VARCHAR | 1 | 4 |
| 10 | 518 | language | language | System.String | 2 | 4 |
| 11 | 519 | language | language | String | 3 | 4 |
| 12 | 520 | language | language | ByteArray | 4 | 4 |
| 13 | 521 | domain | domain | VARCHAR | 1 | 4 |
| 14 | 522 | domain | domain | System.String | 2 | 4 |
| 15 | 523 | domain | domain | String | 3 | 4 |
| 16 | 524 | domain | domain | ByteArray | 4 | 4 |
| 17 | 525 | labelled | labelled | VARCHAR | 1 | 4 |
| 18 | 526 | labelled | labelled | System.String | 2 | 4 |

Рисунок 17. Таблица «last_command_metadata»

**6. «session_parameter»** – информация о параметрах запросов, выполняемых в рамках сессии.

## Поля таблицы:

| Имя столбца | Тип данных | Источник | Назначение |
|---|---|---|---|
| **session_parameter_id** | int | Автоинкремент | Идентификатор строки |
| **session_id** | int | Автоматически | Ссылка на сессию |
| Parameter_id | int | Автоматически | Ссылка на параметр |
| Parameter_value | text | Автоматически | Значение параметра |
| Is_prev_depend | bool | Автоматически | Поле, аналогичное полю из *stg.parameter* |
| Is_max_date | bool | Автоматически | Поле, аналогичное полю из *stg.parameter* |
| Command_id | int | Автоматически | Ссылка на команду |
| Parameter_name | text | Автоматически | Наименование параметра из таблицы *stg.parameter* |



WHERE ORDER BY

| | session_parameter_id | session_id | parameter_id | parameter_value | is_prev_depend | is_max_date | command_id | parameter_name |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 32 | 1 | 2020 | false | false | 7 | year_from |
| 2 | 2 | 32 | 2 | 2022 | false | false | 7 | year_to |
| 3 | 3 | 33 | 1 | 2020 | false | false | 7 | year_from |
| 4 | 4 | 33 | 2 | 2022 | false | false | 7 | year_to |
| 5 | 5 | 34 | 1 | 2020 | false | false | 7 | year_from |
| 6 | 6 | 34 | 2 | 2022 | false | false | 7 | year_to |
| 7 | 7 | 35 | 1 | '1900-01-01 00:00:00' | true | false | 8 | date_from |
| 8 | 8 | 35 | 2 | '2022-12-05 08:51:23' | false | true | 8 | date_to |
| 9 | 9 | 36 | 1 | '2022-12-05 08:51:23' | true | false | 8 | date_from |
| 10 | 10 | 36 | 2 | '2022-12-05 14:45:15' | false | true | 8 | date_to |
| 11 | 11 | 37 | 1 | '2022-12-05 08:51:23' | true | false | 8 | date_from |
| 12 | 12 | 37 | 2 | '2022-12-06 14:11:21' | false | true | 8 | date_to |

Рисунок 18. Таблица «session_parameter»

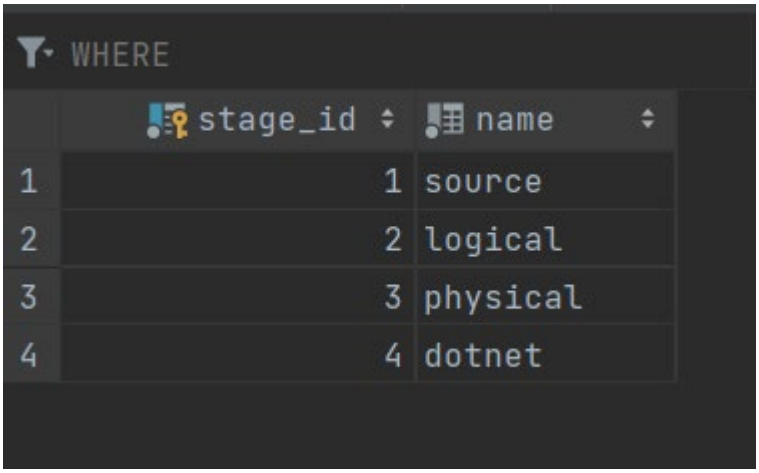
### *4.3.3. Таблицы дескрипторы*

Данные в этих таблицах описывают функциональные части компонента. Они заполняются автоматически при развертывании MetaStaging. И используются только в качестве ссылок при заполнении настроечных таблиц.

1. **«stage»** – список этапов интеграции данных.

**Поля таблицы:**

| Имя столбца | Тип данных | Источник | Назначение |
|---|---|---|---|
| **Stage_ID** | int | Автоинкремент | Идентификатор строки |
| name | text | Автоматически | Наименование этапа интеграции<br>*source – метаданные системы-источника, logical – логический тип хранения в Parquet, physical – физический тип хранения Parquet, dotnet – тип данных «.Net Core» используемый в экстракторе для извлечения из источника* |



WHERE

| | stage_id | name |
|---|---|---|
| 1 | 1 | source |
| 2 | 2 | logical |
| 3 | 3 | physical |
| 4 | 4 | dotnet |

Рисунок 29. Таблица «stage»

2. **«load_type»** – список режимов загрузки данных из источников.

**Поля таблицы:**

| Имя столбца | Тип данных | Источник | Назначение |
|---|---|---|---|
| load_type _id | int | Автоинкремент | Идентификатор строки |

| | | | |
|---|---|---|---|
| name | text | Автоматически | Наименование типа загрузки |

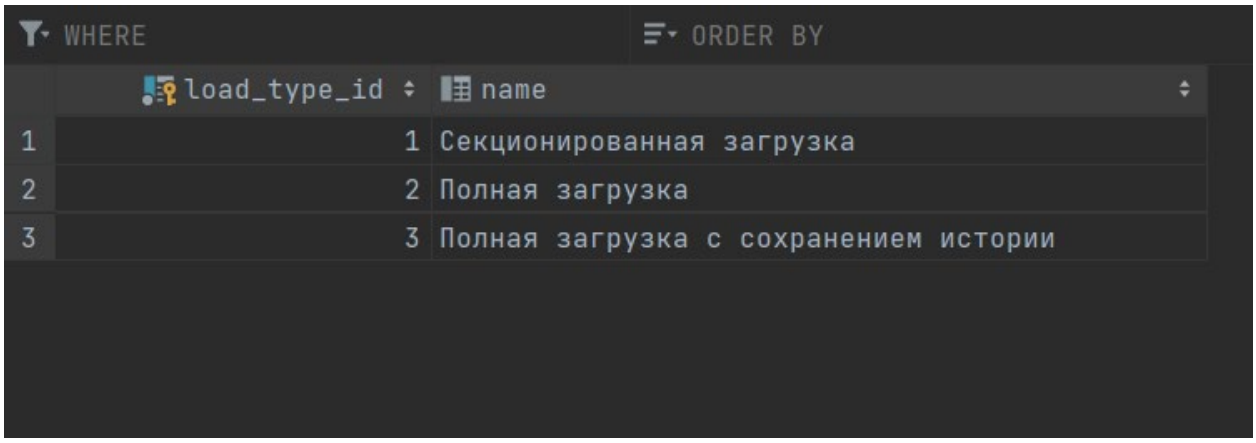



Рисунок 19. Таблица «load_type»

3. **«source_type»** – список поддерживаемых источников.

**Поля таблицы:**

| Имя столбца | Тип данных | Источник | Назначение |
|---|---|---|---|
| source_type_id | int | Автоинкремент | Идентификатор строки |
| name | varchar | Автоматически | Наименование источника |
| credentionals_template | json | Автоматически | Справочная информация – JSON шаблон заполнения поля credentials в *stg.source* |
| short_name | varchar | Автоматически | Краткое наименование |

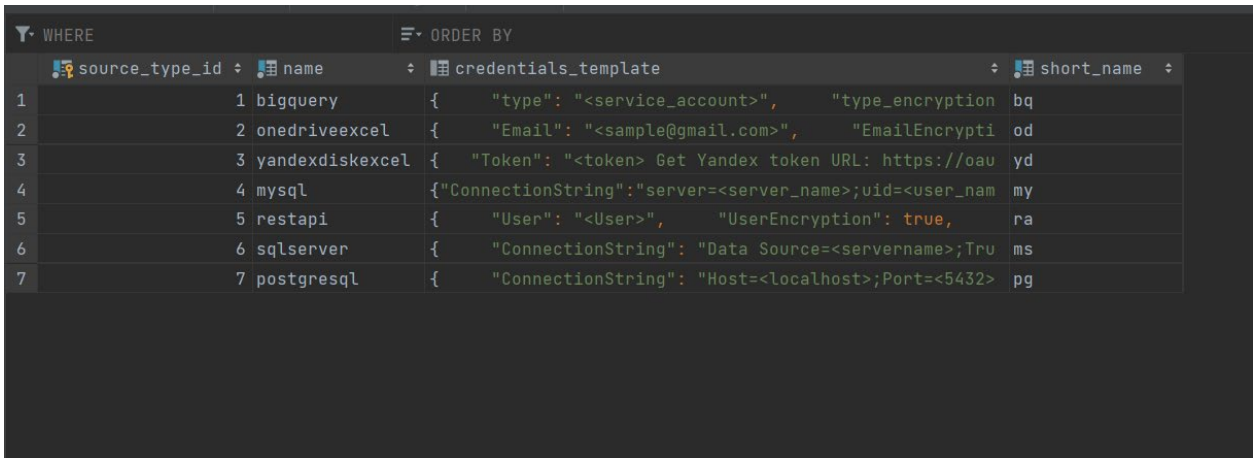



Рисунок 20. Таблица «source_type»

### *4.3.4. Хранимые процедуры и функции MetaStaging*

Представленные далее процедуры и функции являются сервисными, то есть, вызов пользователями запрещен.

**stg.init_session** – инициализирует новую сессию загрузки данных, и выполняет подстановку параметров в запросы к источникам, заполняя таблицы логов (stg.session_command).

Параметры:

- profile_id – идентификатор профиля загрузки.

- start_ts_var – дата запуска сессии.
- dag_id – название профиля (при отсутствии оркестратора, можно передавать любое текстовое значение).

Возвращает:

- JSON с данными, необходимыми для запуска экстрактора (*см. раздел 4.1.3.*), включая id новой сессии и список запросов к источникам (id команды, текст команды, путь к файлу в S3-хранилище).

**stg.get_sample_commands**

Параметры:

- profile_id – идентификатор профиля загрузки.

Возвращает:

- JSON с данными, необходимыми для создания каталога в файловой система «json_schema» (п. Генерация JSON для поля Credentials в таблице stg.source) и для запуска экстрактора (п. Экстрактор).

**stg.compare_metadata –** вызывается для конкретного запроса к источнику с целью проверки изменения метаданных текущей таблицы с таблицей, загруженной в прошлой сессии. Заполняет таблицы stg.command_version и stg.command_metadata.

Параметры:

- command_id – идентификатор запроса к источнику.

Возвращает:

- Номер актуальной версии метаданных.

**stg.mark_command_success –** Помечает запросы, которые были выполнены корректно (включая создание представлений на Greenplum), как успешные: *поле success в stg.session _command = True*.

Параметры:

- command_id

**stg.mark _session_ success -** Помечает сессии, которые были выполнены корректно как успешные: *поле success в stg.session = True.*

Параметры:

- command_id

# 5. СЦЕНАРИИ РАБОТЫ С METASTAGING

MetaStaging предназначен для организации процесса передачи данных из различных источников. На рисунке ниже приведена общая схема движения данных в процессе работы компонента MetaStaging.

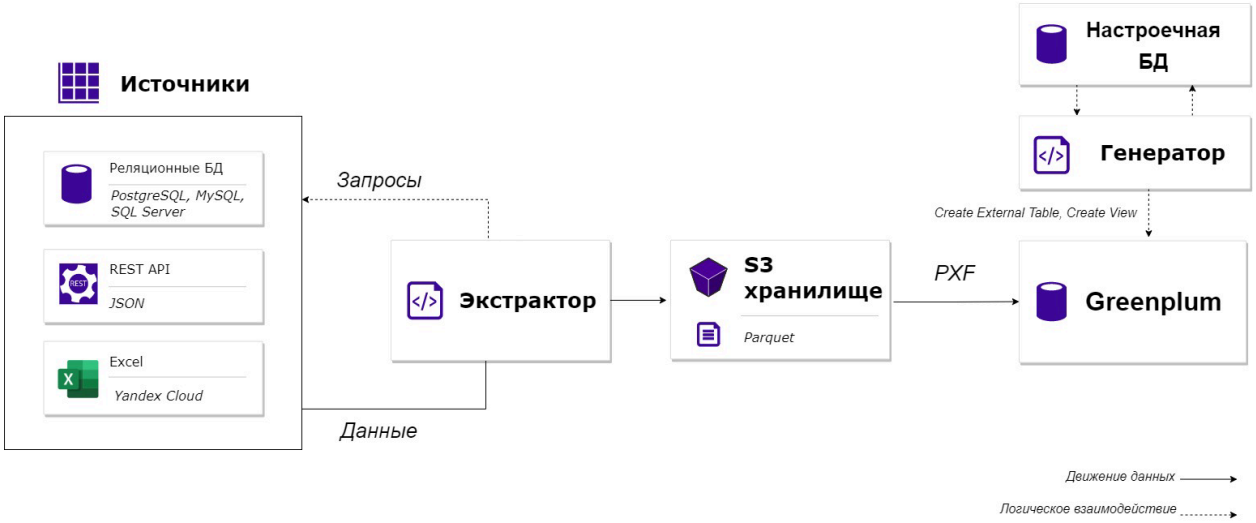



Рисунок 21. Алгоритм работы MetaStaging

Для того, чтобы MetaStaging осуществил указанную выше интеграцию, необходимо заполнить настроечные таблицы. В зависимости от типа загрузки данных (инкрементальная загрузка, полная загрузка, полная загрузка с сохранением истории) алгоритм заполнения этих таблиц меняется:

## 5.1. Предварительная настройка компонента

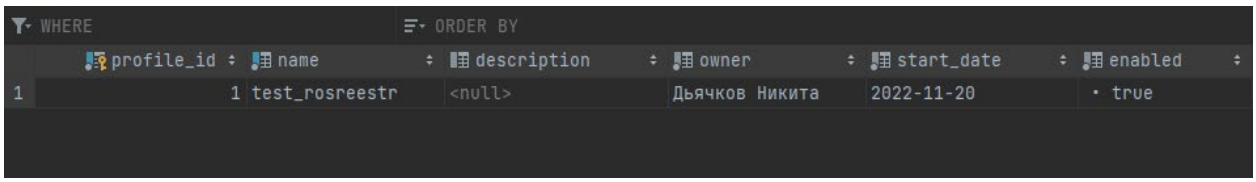
Предварительная настройка компонента (заполнение БД для тестового запуска) осуществляется разработчиком, что позволяет пользователю сразу приступить к решению своих задач, не вдаваясь в подробности настройки.

При необходимости пользователь может протестировать MetaStaging на собственных:

- S3-совместимом хранилище,
- системе источнике
- произвольных запросах к различным системам.

## 5.2. Полная загрузка

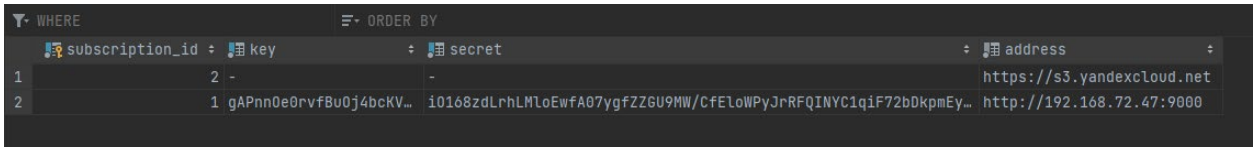
1. Необходимо заполнить таблицу «stg.profile». Поля «name» «description» «owner» должны содержать информацию об имени, описании и владельце профиля соответственно. В поле «start_date» вносится дата начала работы с профилем. Поле «enebled» «включает» тот или иной профиль.



| | profile_id | name | description | owner | start_date | enabled |
|---|---|---|---|---|---|---|
| 1 | 1 | test_rosreestr | <null> | Дьячков Никита | 2022-11-20 | true |

Рисунок 22. Таблица «stg.profile»

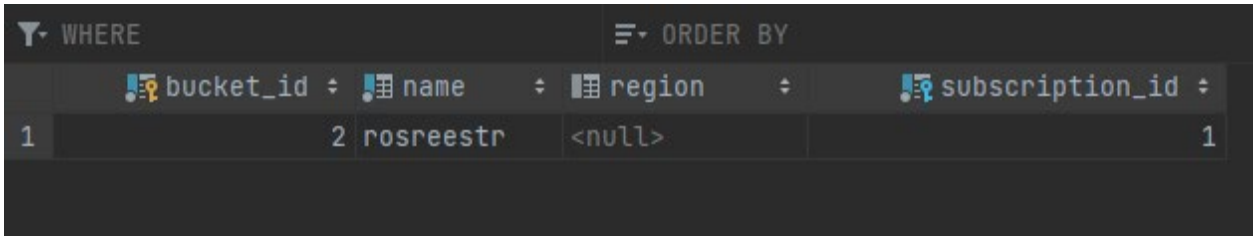
2. Для доступа к S3-хранилищу заполняется таблица «stg.subscription». В поле «key» указывается ключ, в поле «secret» указывается секретный ключ, они хранятся в зашифрованном виде, поле «address» заполняется по адресу сервера, на котором установлен S3. Исходные ключи передаются отдельно по запросу для тестирования метакомпонента, как правило, эти данные хранятся в настройках S3 хранилища.



| | subscription_id | key | secret | address |
|---|---|---|---|---|
| 1 | 2 | - | - | https://s3.yandexcloud.net |
| 2 | 1 | gAPnnOe0rvfBuOj4bcKV… | iO168zdLrhLMloEwfAO7ygfZZGU9MW/CfEloWPyJrRFQINYC1qiF72bDkpmEy… | http://192.168.72.47:9000 |

Рисунок 23. Таблица «stg.subscription»

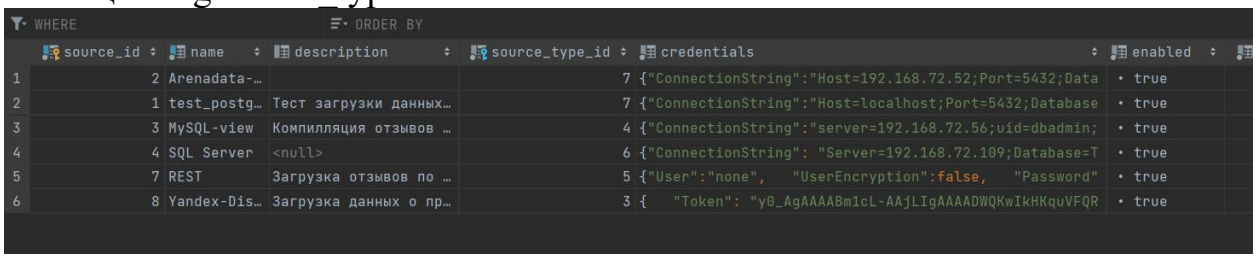
3. Заполняется «name» в таблице «stg.bucket» и в колонке «subscription_id» задается ссылка на таблицу «stg.subscription» (в данном случае в нем указано значение «1»,что в данном случае соответствует идентификатору ключа для доступа к сервисам), при этом «bucket_id» задается автоматически.



| | bucket_id | name | region | subscription_id |
|---|---|---|---|---|
| 1 | 2 | rosreestr | <null> | 1 |

Рисунок 24. Таблица «stg.bucket»

4. Необходимо заполнить таблицу «stg.source». В поле «name» нужно задать имя источника (целесообразно указывать имя источника из таблицы «source_type»), в поле «source_type_id» необходимо указать id источника из таблицы «stg.source_type».



| | source_id | name | description | source_type_id | credentials | enabled |
|---|---|---|---|---|---|---|
| 1 | 2 | Arenadata-… | | 7 | {"ConnectionString":"Host=192.168.72.52;Port=5432;Data | true |
| 2 | 1 | test_postg… | Тест загрузки данных… | 7 | {"ConnectionString":"Host=localhost;Port=5432;Database | true |
| 3 | 3 | MySQL-view | Компилляция отзывов … | 4 | {"ConnectionString":"server=192.168.72.56;uid=dbadmin; | true |
| 4 | 4 | SQL Server | <null> | 6 | {"ConnectionString": "Server=192.168.72.109;Database=T | true |
| 5 | 7 | REST | Загрузка отзывов по … | 5 | {"User":"none", "UserEncryption":false, "Password" | true |
| 6 | 8 | Yandex-Dis… | Загрузка данных о пр… | 3 | { "Token": "y0_AgAAAABm1cL-AAjLIgAAAADWQKwIkHKquVFQR | true |

Рисунок 25. Таблица «stg.source»

Поле «credentials» содержит данные для подключения к системе-источнику, а также указание экстрактору о необходимости шифрования/дешифрования отдельных элементов.

В зависимости от источника формат строки может отличаться. Для преобразования к формату JSON необходимо использовать программу «CredentialsToJsonSerializer» (*см. п. Генерация JSON для поля Credentials в таблице stg.source*).

Чтобы правильно заполнить «credentials» нужно ввести в консоль:

*Host = 192.168.72.52; Port = 5432; Database = postgres; User = postgres; Password = ******

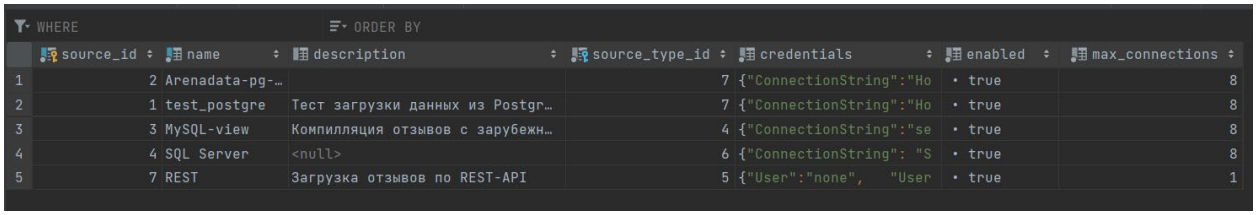
```
itpro_admin@biqube-etl-01:~$ ./CredentialsToJsonSerializer postgresql
--cstring "Host=192.168.72.52;Port=5432;Database=postgres;User=postgre
s;Password=MNBrewq123;" --cstring-secure ""
```

Рисунок 26. Пример работы с консолью

Полученный результат необходимо внести в поле «credentials» в таблицу «stg.source». В зависимости от источника в поле «credentials» могут быть разные записи:

| **Наименование источника (таблица stg.source.type)** | **«credentials»** |
|---|---|
| yandexdiskexcel | *{<br>"Token":"<token> Get Yandex token URL: https://oauth.yandex.ru/authorize?response_type=token&client_id=6105ac8902804c838a5892404c0b39a2",<br>"TokenEncryption":false,<br>"RootFolder":"disk:/<put your path>",<br>"RootFolderEncryption":false<br>}* |
| mysql | *{<br>"ConnectionString":"server=<server_name>;uid=<user_name>;pwd=<password>;database=<db_name>;default command timeout=200;",<br>"ConnectionStringSecure":""<br>}* |
| restapi | *{<br>"User":"<User>",<br>"UserEncryption":true,<br>"Password":"<Password>",<br>"PasswordEncryption":true,<br>"AuthType":1<br>}* |
| sqlserver | *{<br>"ConnectionString":"Data Source=<servername>;TrustServerCertificate=Yes;Initial Catalog=cargoowner;Trusted_Connection=True",<br>"ConnectionStringSecure":""<br>}* |
| postgresql | *{<br>"ConnectionString":"Data Source=<servername>;TrustServerCertificate=Yes;Initial Catalog=cargoowner;Trusted_Connection=True",<br>"ConnectionStringSecure":""<br>}* |

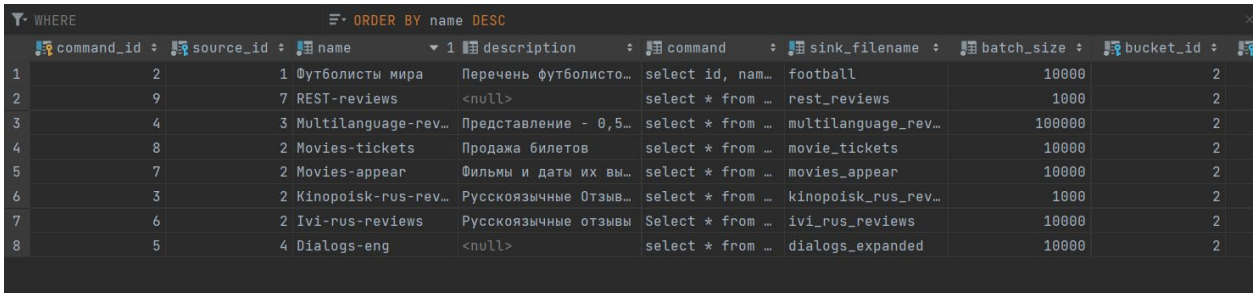
В поле «enabled» необходимо задать «true». В поле «max_connections» задается максимальное желаемое количество запросов к источнику.



| | source_id | name | description | source_type_id | credentials | enabled | max_connections |
|---|---|---|---|---|---|---|---|
| 1 | 2 | Arenadata-pg-… | | 7 | {"ConnectionString":"Ho | true | 8 |
| 2 | 1 | test_postgre | Тест загрузки данных из Postgr… | 7 | {"ConnectionString":"Ho | true | 8 |
| 3 | 3 | MySQL-view | Компилляция отзывов с зарубежн… | 4 | {"ConnectionString":"se | true | 8 |
| 4 | 4 | SQL Server | <null> | 6 | {"ConnectionString": "S | true | 8 |
| 5 | 7 | REST | Загрузка отзывов по REST-API | 5 | {"User":"none", "User | true | 1 |

Рисунок 27.Таблица «stg.source»

5. Необходимо заполнить таблицу «stg.command». Поля «name» и «description» заполняются по наименованию добавляемого объекта и описания соответственно.



| | command_id | source_id | name | description | command | sink_filename | batch_size | bucket_id |
|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 1 | Футболисты мира | Перечень футболисто… | select id, nam… | football | 10000 | 2 |
| 2 | 9 | 7 | REST-reviews | <null> | select * from … | rest_reviews | 1000 | 2 |
| 3 | 4 | 3 | Multilanguage-rev… | Представление - 0,5… | select * from … | multilanguage_rev… | 100000 | 2 |
| 4 | 8 | 2 | Movies-tickets | Продажа билетов | select * from … | movie_tickets | 10000 | 2 |
| 5 | 7 | 2 | Movies-appear | Фильмы и даты их вы… | select * from … | movies_appear | 10000 | 2 |
| 6 | 3 | 2 | Kinopoisk-rus-rev… | Русскоязычные Отзыв… | select * from … | kinopoisk_rus_rev… | 1000 | 2 |
| 7 | 6 | 2 | Ivi-rus-reviews | Русскоязычные отзывы | Select * from … | ivi_rus_reviews | 10000 | 2 |
| 8 | 5 | 4 | Dialogs-eng | <null> | select * from … | dialogs_expanded | 10000 | 2 |

Рисунок 39. Таблица «stg.command»

Для каждого источника правила написания запроса для поля «command» отличаются:

| **Наименование источника (таблица stg.source.type)** | **Запрос для поля «command» таблицы «stg.command»** | |
|---|---|---|
| yandexdiskexcel | *select * from file 'D:\\file.xlsx:[ЛИСТ1]' with (HaveHeader, LoadFirstRow* | |
| mysql | Любой SELECT запрос, который поддерживает СУБД MySQL | |
| restapi | POST-запрос | *Select * from 'Ссылка на ресурс' where body = 'JSON с параметрами'* |
| | *GET-запрос* | *Select * from 'Ссылка на ресурс'* |
| sqlserver | Любой SELECT запрос, который поддерживает СУБД SQL Server | |
| postgresql | Любой SELECT запрос, который поддерживает СУБД PostgreSQL | |

В поле «sink_filename» необходимо задать путь к Parquet-файлу в S3-совместимом хранилище. Поле «bucket_size» необходимо указать количество строк выгружаемых из источников в файл Parquet за одну итерацию при пакетной загрузке. В «bucket_id» необходимо указать из таблицы «stg.bucket». Поле «load_tipe_id» определяет тип загрузки («2» - полная загрузка).

6. Далее необходимо заполнить таблицу «profile_command». В колонке «profile_command_id» указывается id из таблицы «profile_command». При этом в поле «enabled» необходимо выставлять значение «true» только на актуальные загрузки.

Поле «profile_id» необходимо заполнить id профиля, с которого вносится изменение из таблицы «stg.profile».

7. Запуск процесса интеграции данных можно вызвать либо на Windows, либо на Linux.

Процесс запуска с виртуальной машины linux (*см. п.* Запуск компонента на linux), в появившемся окне необходимо ввести python скрипт:

*sudo python MetaStagingExecutor/ MetaStagingExecutor.ru*

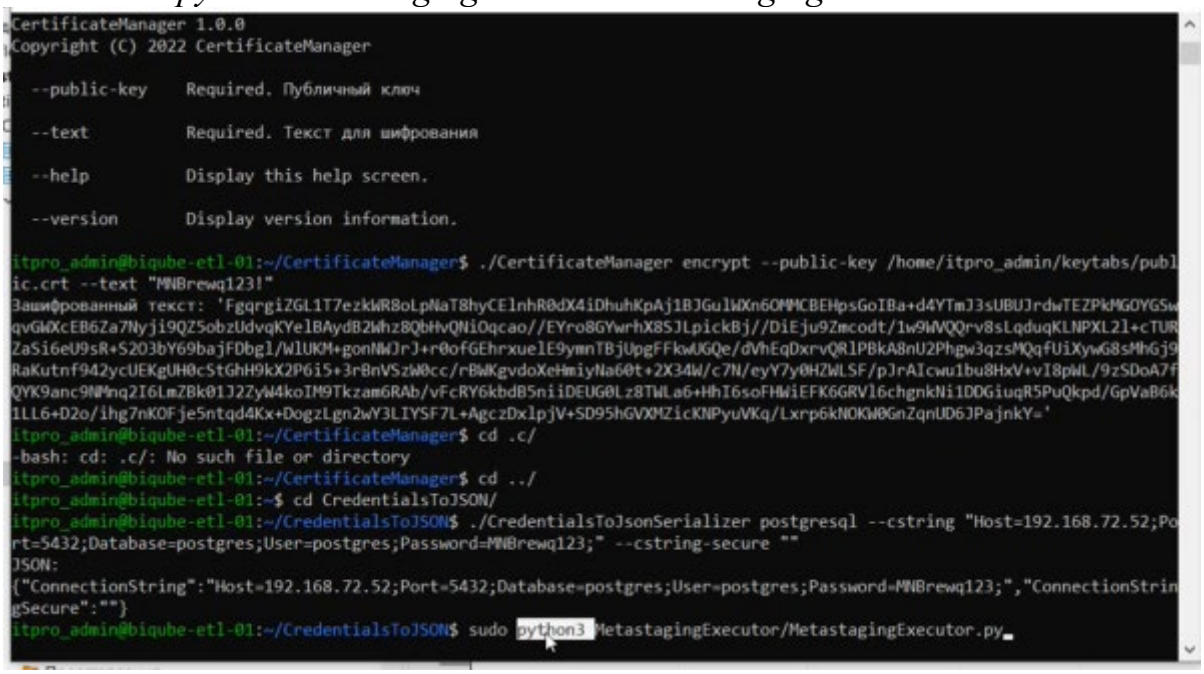



Рисунок 28.Консоль

В результате должны появиться итоговые представления на Greenplum с данными из источников (*см. п. Информация для проверки*).

Процесс запуска с виртуальной машины Windows (*см. п. Запуск компонента на windows),* в появившемся окне необходимо ввести скрипт:

*python3 "C:\MetaStaging\MetaStagingExecutor\MetastagingExecutor.py"*

## 5.3. Полная загрузка с сохранением истории

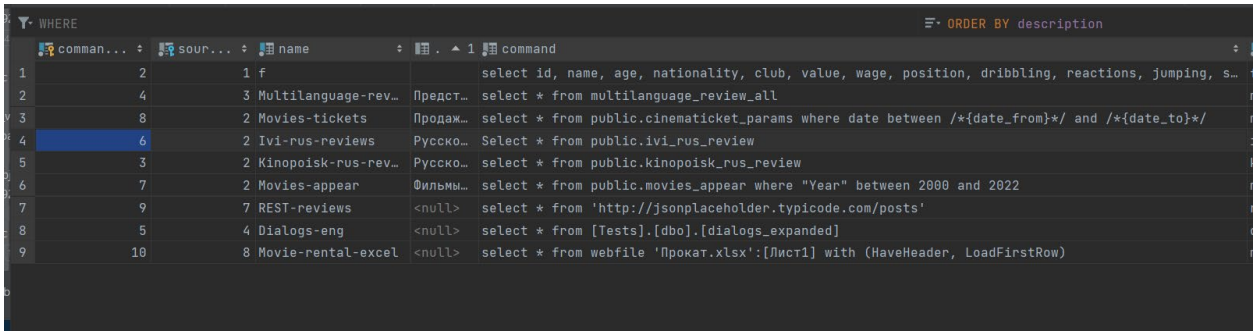
Повторяет алгоритм полной загрузки. Отличается только в части заполнения поля «load_tipe_id», таблицы «stg.command», которое определяет тип загрузки. В данном случае нужно поставить («3» - полная загрузка с сохранением истории).

## 5.4. Инкрементальная загрузка

1. Для инкрементальной загрузки данных следует повторить первых 4 пункта описанный в п. Полная загрузка

2. Запрос для инкрементальной загрузки (или загрузки с параметрами) данных отличается от полной запросом в строке «command» таблицы «stg.command». Любая команда прописываемая с условием для

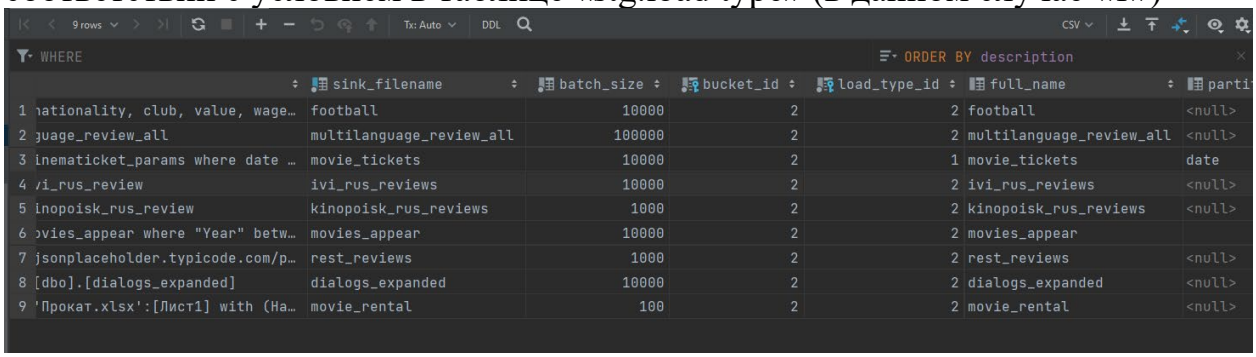
секционированной загрузки должна отражать это же условие в таблице «stg.parameter».



| | comman... | sour... | name | . 1 | command |
|---|---|---|---|---|---|
| 1 | 2 | 1 | f | | select id, name, age, nationality, club, value, wage, position, dribbling, reactions, jumping, s… |
| 2 | 4 | 3 | Multilanguage-rev… | Предст… | select * from multilanguage_review_all |
| 3 | 8 | 2 | Movies-tickets | Продаж… | select * from public.cinematicket_params where date between /*{date_from}*/ and /*{date_to}*/ |
| 4 | 6 | 2 | Ivi-rus-reviews | Русско… | Select * from public.ivi_rus_review |
| 5 | 3 | 2 | Kinopoisk-rus-rev… | Русско… | select * from public.kinopoisk_rus_review |
| 6 | 7 | 2 | Movies-appear | Фильмы… | select * from public.movies_appear where "Year" between 2000 and 2022 |
| 7 | 9 | 7 | REST-reviews | <null> | select * from 'http://jsonplaceholder.typicode.com/posts' |
| 8 | 5 | 4 | Dialogs-eng | <null> | select * from [Tests].[dbo].[dialogs_expanded] |
| 9 | 10 | 8 | Movie-rental-excel | <null> | select * from webfile 'Прокат.xlsx':[Лист1] with (HaveHeader, LoadFirstRow) |

Рисунок 29. Таблица «stg.command»

Так же в таблице command важно указывать «load type id», он задается в соответствии с условием в таблице «stg.load type» (в данном случае «1»)



| | | sink_filename | batch_size | bucket_id | load_type_id | full_name | parti |
|---|---|---|---|---|---|---|---|
| 1 | nationality, club, value, wage… | football | 10000 | 2 | 2 | football | <null> |
| 2 | guage_review_all | multilanguage_review_all | 100000 | 2 | 2 | multilanguage_review_all | <null> |
| 3 | inematicket_params where date … | movie_tickets | 10000 | 2 | 1 | movie_tickets | date |
| 4 | vi_rus_review | ivi_rus_reviews | 10000 | 2 | 2 | ivi_rus_reviews | <null> |
| 5 | inopoisk_rus_review | kinopoisk_rus_reviews | 1000 | 2 | 2 | kinopoisk_rus_reviews | <null> |
| 6 | ovies_appear where "Year" betw… | movies_appear | 10000 | 2 | 2 | movies_appear | |
| 7 | jsonplaceholder.typicode.com/p… | rest_reviews | 1000 | 2 | 2 | rest_reviews | <null> |
| 8 | [dbo].[dialogs_expanded] | dialogs_expanded | 10000 | 2 | 2 | dialogs_expanded | <null> |
| 9 | 'Прокат.xlsx':[Лист1] with (Ha… | movie_rental | 100 | 2 | 2 | movie_rental | <null> |

Рисунок 30. Таблица «stg.command»

3. Необходимо заполнить таблицу «stg.parameter»

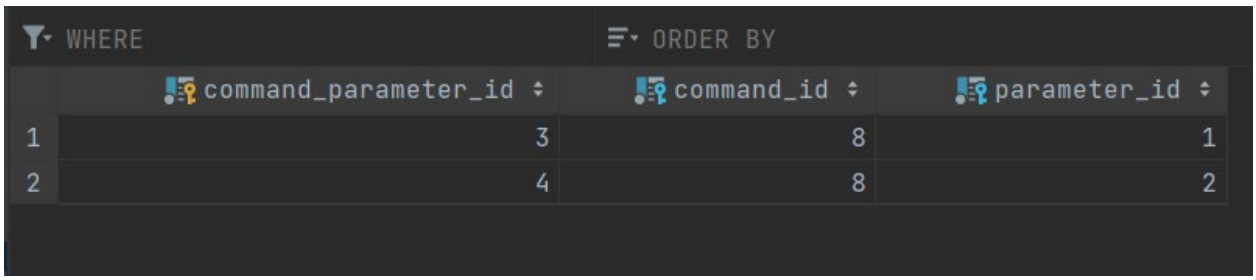
Поле parameter id заполняется автоматически, значение в поле name должно соответствовать плейсхолдеру, который указана в таблице «stg.command», в поле «command».

В случае если у источника есть временные рамки необходимо задать «true» в таблицах «is_prev_depend» и «is_max_date»

| | parameter_id | name | value | is_prev_depend | is_max_date |
|---|---|---|---|---|---|
| 1 | 1 | date_from | 2000 | true | false |
| 2 | 2 | date_to | 2022 | false | true |

Рисунок 31. Таблица «stg.parameter»

4. Далее необходимо связать запросы с источниками с параметрами в таблице «stg.command_parameter». Для этого нужно взять «id command» в таблице «stg.command», а «id parameter» в таблице «stg.parameter».



WHERE ORDER BY

| | command_parameter_id | command_id | parameter_id |
|---|---|---|---|
| 1 | 3 | 8 | 1 |
| 2 | 4 | 8 | 2 |

Рисунок 32. Таблица «stg.command_parameter»

5. Для запуска процесса интеграции данных нужно вызвать перейти на виртуальную машину linux (*см. п. Запуск компонента на linux*), в появившемся окне необходимо ввести python скрипт:

*sudo python MetaStagingExecutor/ MetaStagingExecutor.ru*

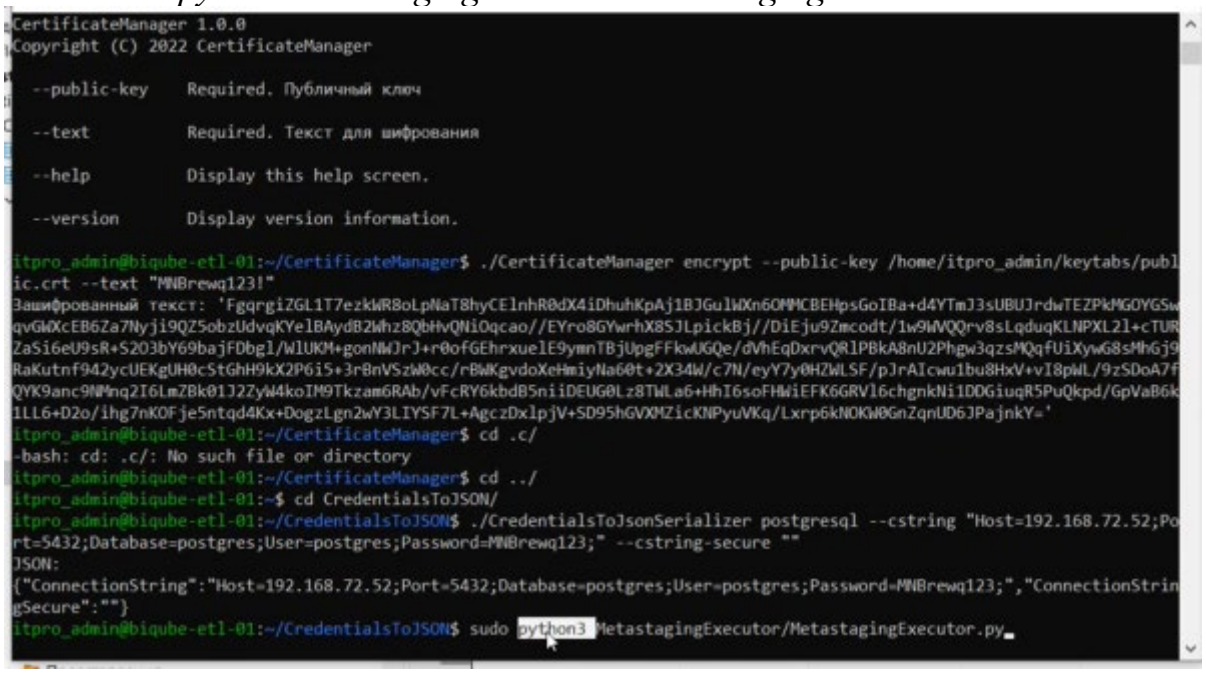



Рисунок 33.Консоль

В результате должны появиться итоговые представления на Greenplum с данными из источников (*см. п. Информация для проверки*).

# 6. ИНФОРМАЦИЯ ДЛЯ ПРОВЕРКИ

**Для проверки корректности заливки в S3-содержимое хранилище** можно использовать программу S3-Browser:

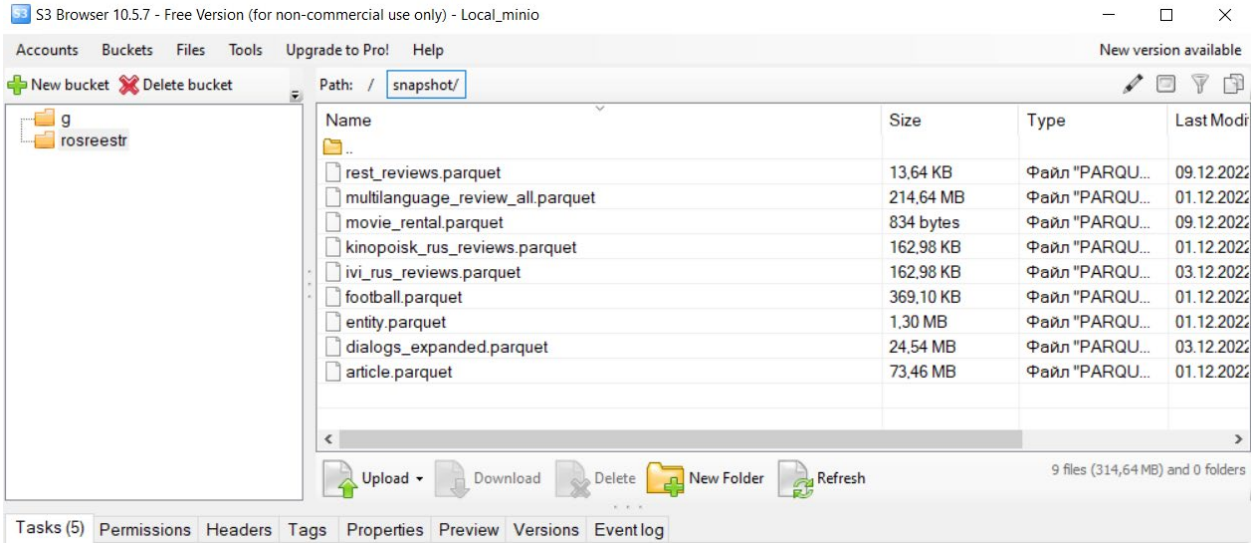



Рисунок 34. S3 хранилище

Отсюда можно скачать любой **файл с данными и проверить его содержимое** в программе ParquetViewer:

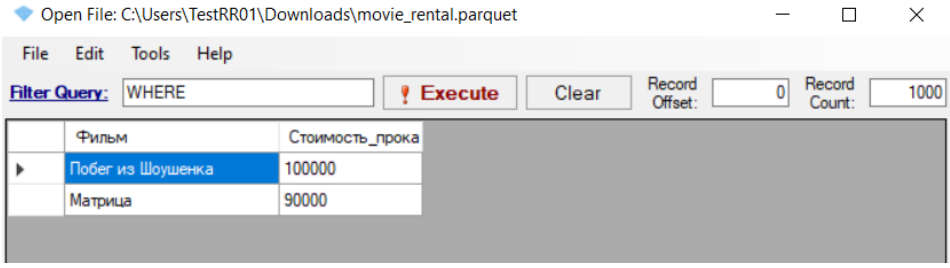



Рисунок 35. ParquetViewer

**Конечные представления в Greenplum** проверяются через DBeaver также, как и настроечная БД. В схеме «back» лежат внешние таблицы, которые через PXF обращаются к файлам в S3, а в схеме «public» лежат итоговые представления с данными последней загрузки и с данными за все загрузки.

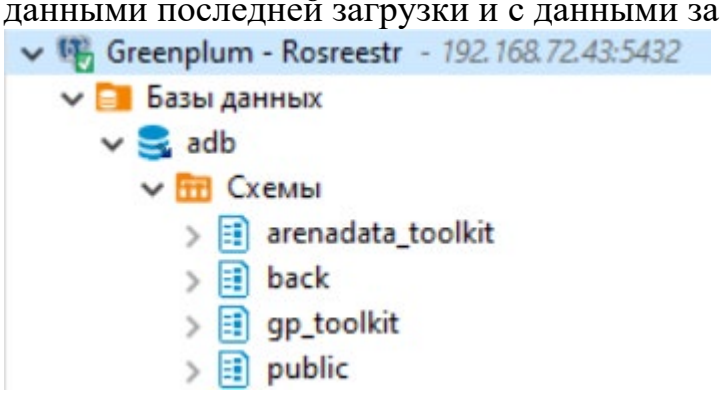



Рисунок 36. Представление в Greenplum

В результате должны появиться **итоговые представления** на Greenplum с данными из источников:


Рисунок 49. Представление в Greenplum