ФУНКЦИОНАЛЬНЫЕ ХАРАКТЕРИСТИКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ BI.QUBE METASTAGING

Москва, 2023

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Компонент MetaStaging позволяет консолидировать в стейжинговом 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, принцип хранения данных в модели Data Vault, а также при необходимости отслеживать ошибки допущенные в процессе настройки работы с компонентом.

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	Загрузка данных в Greenplum без параметризации. Применяется, когда необходима полная перезагрузка всех данных в таблице на источнике (например, при отсутствии столбца, подходящего для секционирования).

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. ЦЕЛЬ 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.

1.1. Описание компонентов системы

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

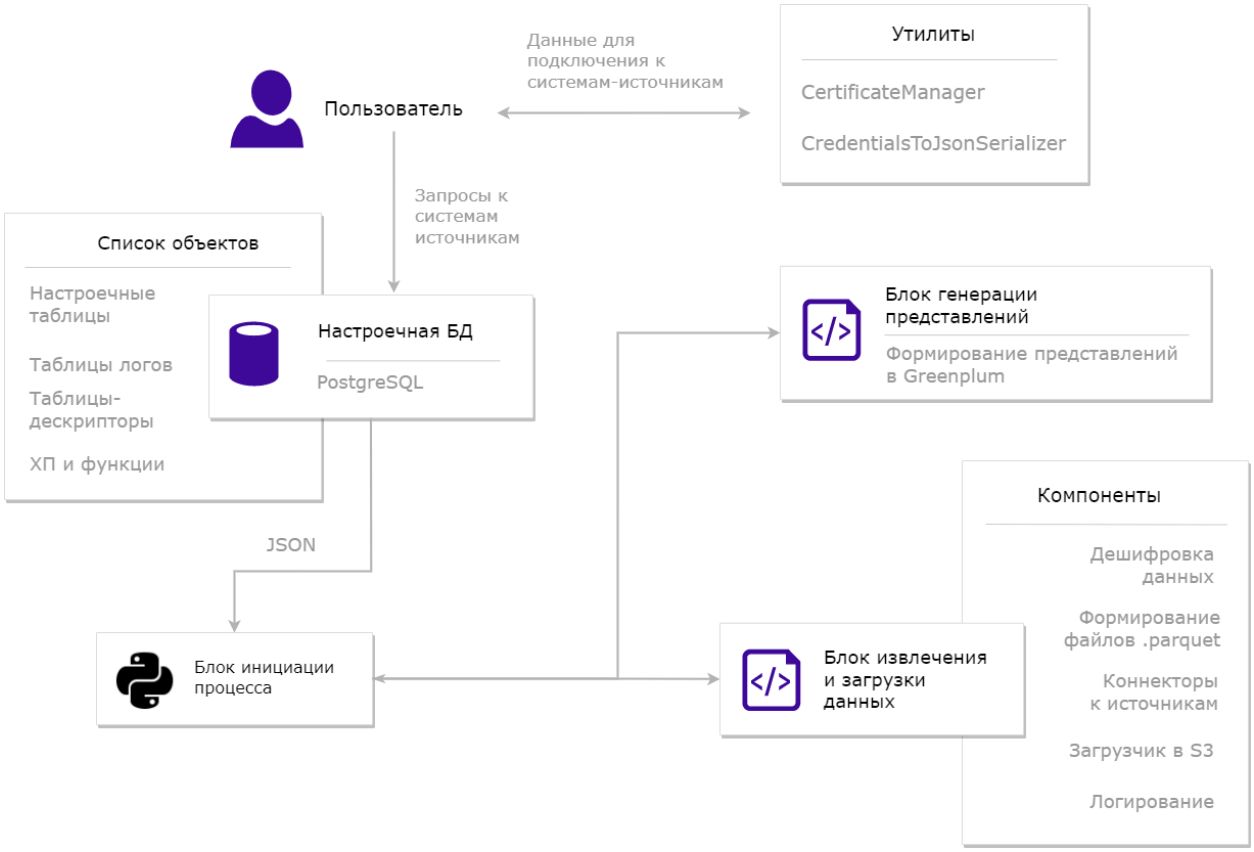


Рисунок 1. Компоненты системы MetaStaging

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.* ***Ошибка! Источник ссылки не найден.***).
- 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*).

1.1.1. 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/*

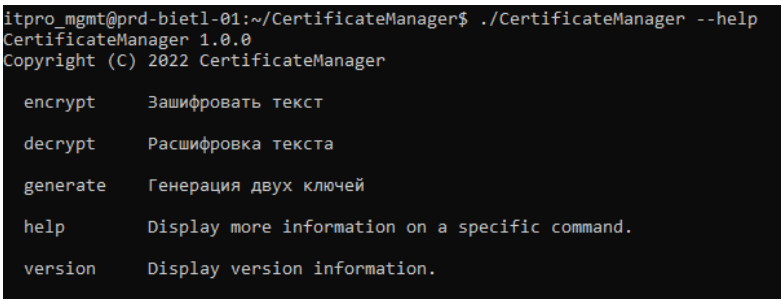


Рисунок 2. Пример применения утилиты

Пример команды для шифрования текста:

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

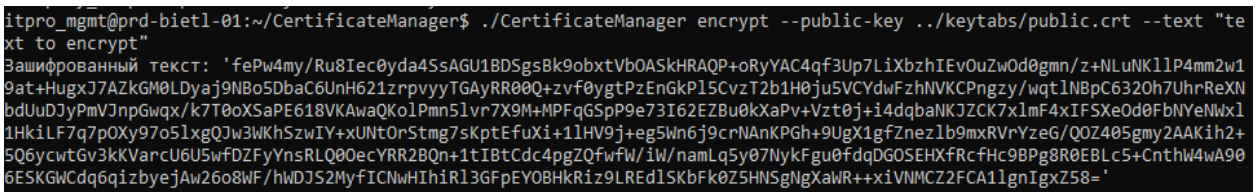


Рисунок 3. Пример применения утилиты

Пример команды для расшифровки текста:

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

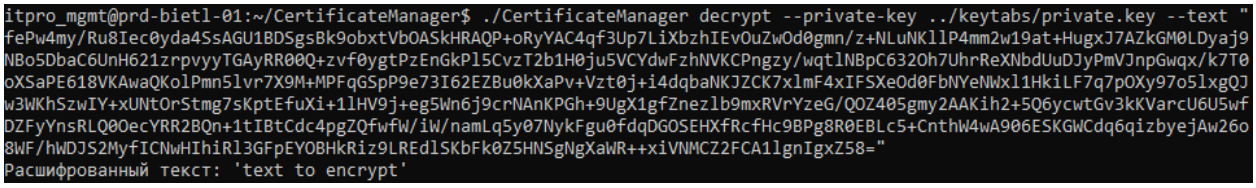


Рисунок 4. Пример применения утилиты

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
}"
```

1.1.2. *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/.*

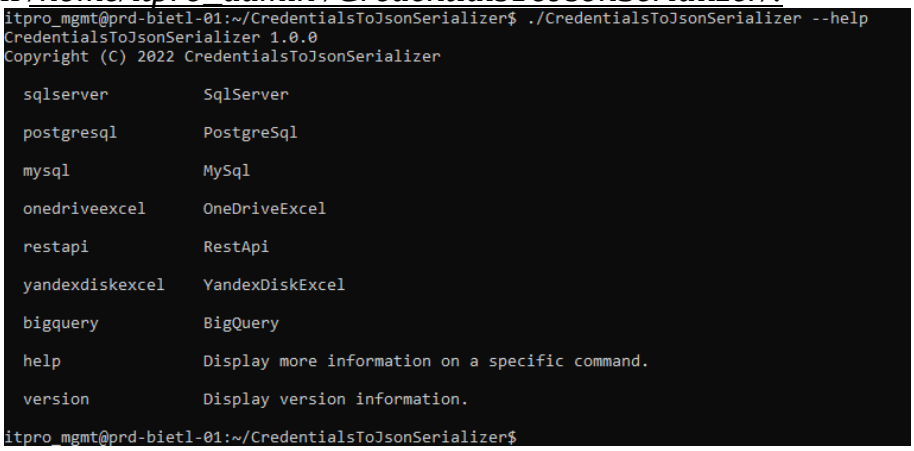


Рисунок 5. Пример использования программы

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

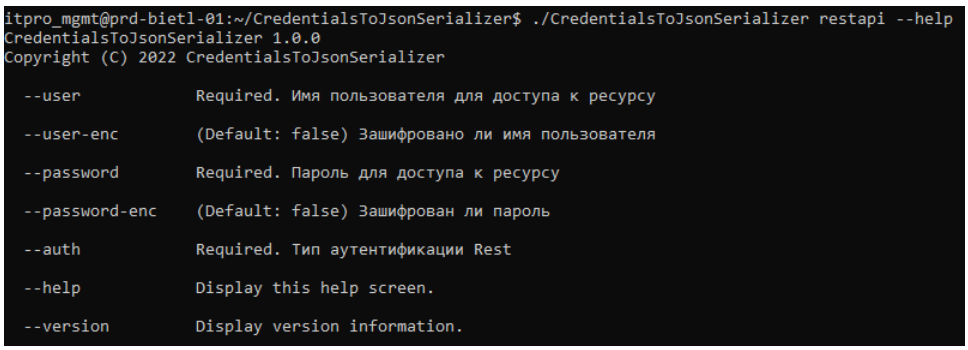


Рисунок 6. Пример использования программы

Пример команды для источника restapi:

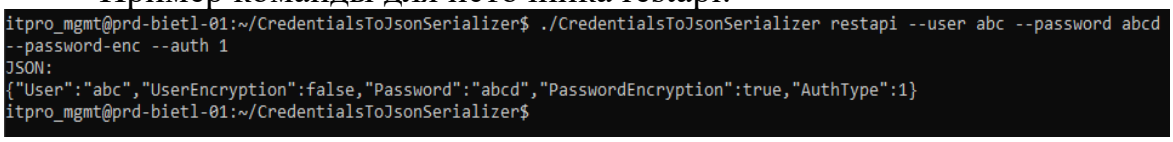


Рисунок 7. Пример использования программы

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

1.1.3. 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.

1.1.4. 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.

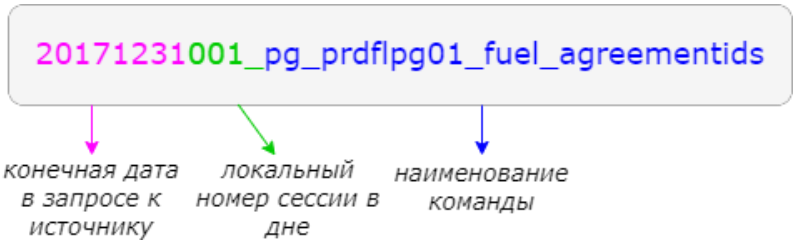
Генератор вызывается на последнем этапе пайплайна (блок инициации процесса). Сборка расположена в директории */home/itpro_admin/DbEntitiesGenerator/*.

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.

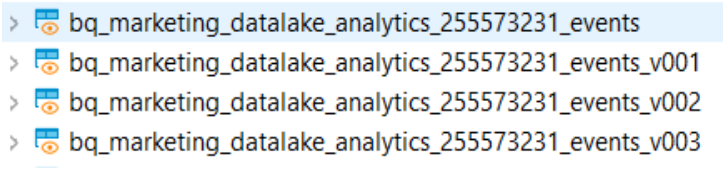
Каждый запрос из инкрементальной загрузки представлен в Greenplum перечнем ET (1 загрузка в parquet = 1 ET). Наименование для ET состоит из следующих частей:



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&a
ccesskey=YCAJEcQEeeSmEXJ2wUXJK_NyO&secretkey=YCO9wLy5_O8C9mAzA3CgcV4kXnQ
rJCddd6ZLklp4&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"."U
      pdatedAt"/ 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;
```

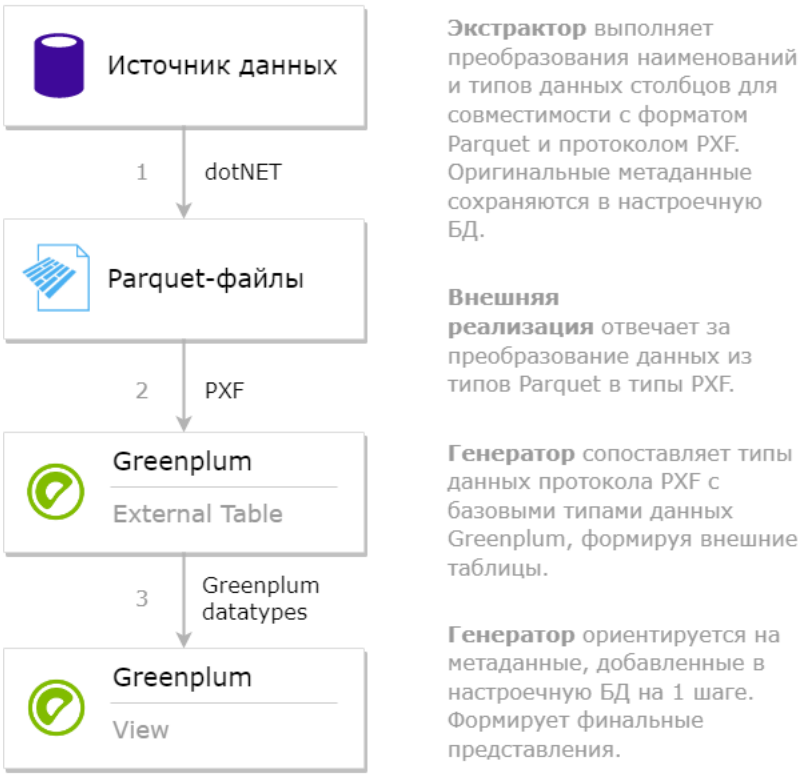


Рисунок 8. Алгоритм формирования слоя в Greenplum

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

2. ТРЕБОВАНИЯ К ПРОГРАММНОМУ И АППАРАТНОМУ ОБЕСПЕЧЕНИЮ METASTAGING

2.1. Требования к ПО

Все процессы компонент осуществляет на основе информации из настроечной БД, которую заполняет пользователь. Взаимодействие с компонентом осуществляется через веб-интерфейс или работая напрямую с БД с помощью доступной среды разработки (например, DBeaver).

Компонент MetaStaging работает под управлением СУБД: PostgreSQL (9.0 и позднее), Postgres Pro (10.22 и позднее), Arenadata Postgres (ADPG) (14.2.1), Greenplum.

2.2. Требования к аппаратному обеспечению

Минимальные аппаратные требования для установки серверной части (процессинг и БД):

- Процессор с тактовой частотой более 2.0 ГГц
- Оперативная память- 2GB
- Свободное место на жестком диске 350 Мб для исходного кода в процессе компиляции и 60 Мб для каталога инсталляции

2.3. Поддерживаемые системы-источники

Компонент поддерживает наиболее востребованные источники, среди которых брокеры сообщений, реляционные, документно-ориентированные и облачные БД и другие:

- Big Query (На источнике данные должны быть в представлениях или таблицах.).
- Rest API Данные должны быть переданы через HTTP-запросы (GET и POST) в формате JSON и XML.).
- SQL Server (На источнике данные должны быть в представлениях или таблицах.).
- PostgreSQL (На источнике данные должны быть в представлениях или таблицах.).
- MySQL (На источнике данные должны быть в представлениях или таблицах.).
- Excel (Поддерживаются файлы в формате «*.xls», «.xlsx», «.xlsm»* и могут быть расположены на локальной машине, в OneDrive и в YandexCloud.).