**Описание Fias\_Loader**.

Состав документа.

1. Назначение.
2. Описание архитектуры.
3. Типовые варианты использования, часть 1. Последовательное выполнение.
4. Управление функционированием. Описание управляющих yaml-файлов, csv-файлов.
5. Развёртывание кластера БД.
6. Типовые варианты использования, часть 2. Микросервисное решение.

7. Контроль за функционированием.

Замечание: каждая глава является самостоятельным документом, должна быть снабжена иллюстрациями. Допускается повторение текста из смежных глав.

**1. Назначение.**

Пакет "Fias\_Loader" предназначен для обновления и дополнения адресной базы CRM

Смородина Диалог. Источником данных является сайт "https://fias.nalog.ru/Updates".

В настоящее время скачиваются и обрабатываются только полные версии адресных данных.

Все действия по обновлению и дополнению адресных данных выполняются последовательно, и состоят из трёх основных этапов:

- Загрузка адресного архива с сайта налоговой инспекции. Далее, распаковка ZIP-архива и parsing полученных XML-данных. В процессе парсинга выполняется запоминание данных в соответствующих таблицах БД.

- Обработка загруженных адресных адресных данных. В свою очередь этап обработки разделяется на несколько подэтапов:

- проверка целостности данных, их агрегация в структуру, соответствующую админи-

стративно-территориальному делению.

- обновление следующих объектов: "Гео-регионы ", "Элементы дорожно-планировочной структуры", "Адреса (дома, здания и пр. строения)".

- Перенос обновлённых данных в целевую БД. Перенос данных в целевую БД может быть выполняться двумя способами:

- обновление адресной схемы в целевой БД, выполняемое с использованием FDW (Foreign Data Wrapper), такой вид переноса, возможен только при наличии сетевого соединения между процессинговым хостом и целевой БД.

- выгрузка инкрементальных обновлений в набор текстовых файлов, с последующей их загрузкой в целевую БД.

Отдельно будет рассмотрен рассмотрен этап настройки справочников типов адресных объектов. Все модели: сущность-связь, диаграммы потоков, диаграммы последовательностей созданы в CASE-средстве SAP PowerDesigner.

Далее остановимся на структуре адресной базы.

Адресный архив после раскрытия состоит из примерно из 92 папок, каждая из которых содержит адресные данные по конкретному региону, в каждой из таких региональных папок находится 18 файлов.

Общие данные, т.е справочники типов, документов и т.п, единые для всех адресных регионов, содержатся в 10 XML-файлов и расположены в корне раскрытого архива.

В приложении на рис 1 и 2 показаны примеры файловых структур, содержащей адресные данные, на рис\_3 показана физическая модель адресных данных.

Структура адресного архива, скачанного с сайта и структура процессингового кластера, (множество БД и программ, выполняющих обработку адресных данных) находятся в соотношении 1:1, т.е каждому адресному региону, соответствует одна БД. При этом parsing, загрузка и последующая обработка распределяются по отдельным базам. Такая схема позволяет равномерно распределить нагрузку, достигнуть высокой степени параллелизма при обработке данных, быстро локализовать ошибки, как в данных, так и в программном коде.

**2. Описание архитектуры.**

Пакет Python программ "load\_main". В программах пакета сосредоточена основная логика работы загрузчика. Действия большинства программ описываются сценариями, оформленных в виде CSV-файлов. CSV-файл является базисной структурой, используемой в многих случаях. Коммуникационные параметры и параметры для так называемых «больших команд», описаны в виде YAML-файлов.

Каждое действие, выполняемое в базе данных Python\_программами пакета основано на хранимых функциях/процедурах, написанных на PLpg/SQL. Принято такое разделение функциональности:

- все действия, связанные с преобразованием данных — хранимые функции/процедуры.

- управление процессами, операции с файловой системой — методы и функции в Python-программах.

- управление мультипроцессной работой, обмен данными между процессами, отложенная обработка — программы на Python и хранимые функции.

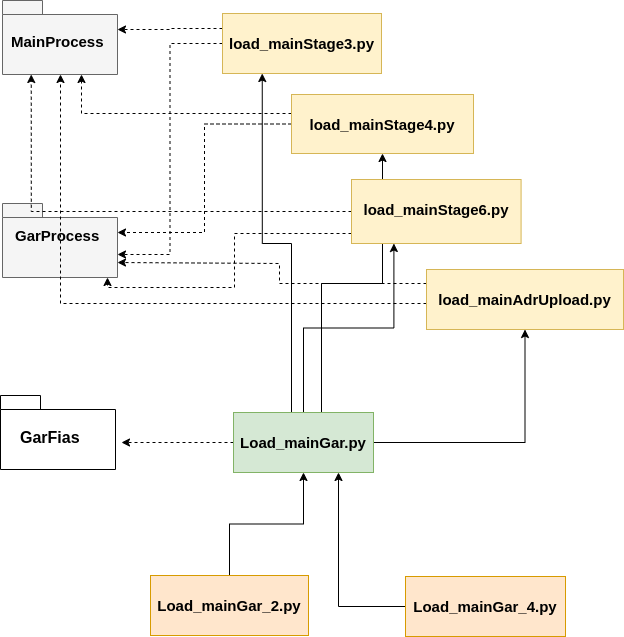
Все программы вызываются из командной строки. GUI нет. Разработка, тестирование и эксплуатация ведутся по OS семейства Linux (Ubuntu 18.4, CentOS 7).

**2.1. Программы пакета load\_main**

| **№ пп** | **Имя программы/пакета** | **Описание** |
| --- | --- | --- |
| 1 | GarFias | Пакет, содержит парсеры. Парсеры не являются универсальными, один парсер соответствует одному адресному XML-файлу и одной хранимой процедуре в базе данных. |
| 2 | GarProcess | Пакет содержащий вспомогательные классы, обеспечивающих выполнение так называемых «больших команд», выполняемых на различных этапах обработки адресных данных. Каждой «большой» команде соответствует множество хранимых функций/процедур. |
| 3 | MainProcess | Пакет с базисными классами. Они используются во всех остальных программах пакета. |
| 4 | load\_mainStage3.py | «Большая команда 3». Реализация этапа предобработ-ки. Управляется файлом «stage\_3.yaml». |
| 5 | load\_mainStage4.py | «Большая команда 4». Реализация этапа основной обработки. Управляется файлом «stage\_4.yaml». |
| 6 | load\_mainStage6.py | «Большая команда 6». Реализация этапа постоб-работки. Управляется файлом «stage\_6.yaml». |
| 7 | load\_mainAdrUpload.py | Дополнительная функциональность, реализует выгрузку инкрементальных обновлений в тестовый файл. Может быть выполнена как в составе «Большая команда 6», так и самостоятельно. |
| 8 | load\_mainGar.py | Основная программа пакета «load\_main». Управляется CSV-сценарием и набором параметров, передаваемых в командной строке, работает с одним хостом и одной БД. Наследует все классы и методы из описанных выше пакетов. |
| 9 | load\_mainGar\_2.py | Управляется csv-сценарием. Последовательная мно-жественная обработка, хосты и базы описаны в управляющем YAML-файле. Наследуют классы и методы из load\_mainGar.py. |
| 10 | load\_mainGar\_4.py | Полностью управляется YAML-файлом, в котором описаны хосты, базы и исполняемые CSV сценарии. В настоящее время используется только для parsing исходных XML-данных, либо на этапе развёртывания кластера БД. Наследуют классы и методы из load\_mainGar.py. |
| 11 | load\_mainBuild.sh | SHELL-оболочка над программой load\_mainGar.py, управляется CSV-сценарием и набором параметров, передаваемых в командной строке. Используется ТОЛЬКО при сборке программных пакетов из хранимых процедур/функций, написанных на PLpg/SQL. |
| 12 | load\_mainCrtScripts.py | Используется перед обработкой нового адресного архива либо перед начальной инициализацией кластера БД. Создаёт sql-скрипты. Управляется csv - файлом «pattern\_xx.csv» и набором параметров, передаваемых в командной строке. |
| 13 | load\_mainCrtYaml.py | Создаёт коллекцию YAML-файлов, Используется только при первоначальном развороте кластера Управляется csv - файлом «pattern\_xx.csv» и набором параметров, передаваемых в командной строке. |
| 14 | load\_mainCrtFscripts.py | Создаёт набор скриптов, при выполнении которых будут созданы внешние сервера и внешние таблицы. УСТАРЕЛА. Была использована при реализации секционированной адресной базы, использующей технологию FDW. При этом каждая секция являлась сторонней таблицей. Управляется csv - файлом «pattern\_xx.csv» и набором параметров, передаваемых в командной строке. |
| 15 | load\_mainTemp.py | Создаёт заготовку csv-сценария, описывающего действия, выполняемые в БД. |
| 16 | r\_scan\_zip.py | Распаковщик zip-архивов. При многопоточном функционировании, должна являться инициатором выполнения цепочки процессов. |
| 17 | load\_mainS\_tr.py | LEGASY от предыдущих проектов, перекодировщик текстовых файлов. |

На Рисунке 6, приводится диаграмма, отображающая основные зависимости между классами и пакетами.

Все программы пакета «load\_main», будучи вызванными без параметров, выводят информационное сообщение об использовании программы, наборе допустимых параметров. (USE CASE программы). Программы, отображённые в правой верхней части диаграммы классов, допускают автономное выполнение со своим собственным набором параметров (блоке кода *if \_\_name\_\_* ***==*** *'\_\_main\_\_'****:*** реализовано создание экземпляров основных классов и вызов соответствующих методов). Что бывает полезным для работы в нестандартных ситуация.

 Рис.6 Диаграмма классов.

**2.2. Описание кластера БД.**

Описание структуры кластера БД начинается с перечня адресных регионов.

| **№ пп** | **Имя базы** | **Имя региона** | **Версия** |
| --- | --- | --- | --- |
| 1 | unsi\_test\_01 | Адыгея Респ | 2022-12-26 |
| 2 | unsi\_test\_02 | Башкортостан Респ | 2022-12-26 |
| 3 | unsi\_test\_03 | Бурятия Респ | 2022-12-26 |
| 4 | unsi\_test\_04 | Алтай Респ | 2022-12-26 |
| 5 | unsi\_test\_05 | Дагестан Респ | 2022-12-26 |
| 6 | unsi\_test\_06 | Ингушетия Респ | 2022-12-26 |
| 7 | unsi\_test\_07 | Кабардино-Балкарская Респ | 2022-12-26 |
| 8 | unsi\_test\_08 | Калмыкия Респ | 2022-12-26 |
| 9 | unsi\_test\_09 | Карачаево-Черкесская Респ | 2022-12-26 |
| 10 | unsi\_test\_10 | Карелия Респ | 2022-12-26 |
| 11 | unsi\_test\_11 | Коми Респ | 2022-12-26 |
| 12 | unsi\_test\_12 | Марий Эл Респ | 2022-12-26 |
| 13 | unsi\_test\_13 | Мордовия Респ | 2022-12-26 |
| 14 | unsi\_test\_14 | Саха /Якутия/ Респ | 2022-12-26 |
| 15 | unsi\_test\_15 | Северная Осетия - Алания Респ | 2022-12-26 |
| 16 | unsi\_test\_16 | Татарстан Респ | 2022-12-26 |
| 17 | unsi\_test\_17 | Тыва Респ | 2022-12-26 |
| 18 | unsi\_test\_18 | Удмуртская Респ | 2022-12-26 |
| 19 | unsi\_test\_19 | Хакасия Респ | 2022-12-26 |
| 20 | unsi\_test\_20 | Чеченская Респ | 2022-12-26 |
|  |  |  |  |
| 21 | unsi\_test\_21 | Чувашская Респ | 2022-12-26 |
| 22 | unsi\_test\_22 | Алтайский край | 2022-12-26 |
| 23 | unsi\_test\_23 | Краснодарский край | 2022-12-26 |
| 24 | unsi\_test\_24 | Красноярский край | 2022-12-26 |
| 25 | unsi\_test\_25 | Приморский край | 2022-12-26 |
| 26 | unsi\_test\_26 | Ставропольский край | 2022-12-26 |
| 27 | unsi\_test\_27 | Хабаровский край | 2022-12-26 |
| 28 | unsi\_test\_28 | Амурская обл. | 2022-12-26 |
| 29 | unsi\_test\_29 | Архангельская обл. | 2022-12-26 |
| 30 | unsi\_test\_30 | Астраханская обл. | 2022-12-26 |
| 31 | unsi\_test\_31 | Белгородская обл. | 2022-12-26 |
| 32 | unsi\_test\_32 | Брянская обл. | 2022-12-26 |
| 33 | unsi\_test\_33 | Владимирская обл. | 2022-12-26 |
| 34 | unsi\_test\_34 | Волгоградская обл. | 2022-12-26 |
| 35 | unsi\_test\_35 | Вологодская обл. | 2022-12-26 |
| 36 | unsi\_test\_36 | Воронежская обл. | 2022-12-26 |
| 37 | unsi\_test\_37 | Ивановская обл. | 2022-12-26 |
| 38 | unsi\_test\_38 | Иркутская обл. | 2022-12-26 |
| 39 | unsi\_test\_39 | Калининградская обл. | 2022-12-26 |
| 40 | unsi\_test\_40 | Калужская обл. | 2022-12-26 |
| 41 | unsi\_test\_41 | Камчатский край | 2022-12-26 |
| 42 | unsi\_test\_42 | Кемеровская обл. | 2022-12-26 |
| 43 | unsi\_test\_43 | Кировская обл. | 2022-12-26 |
| 44 | unsi\_test\_44 | Костромская обл. | 2022-12-26 |
|  |  |  |  |
| 45 | unsi\_test\_45 | Курганская обл. | 2022-12-26 |
| 46 | unsi\_test\_46 | Курская обл. | 2022-12-26 |
| 47 | unsi\_test\_47 | Ленинградская обл. | 2022-12-26 |
| 48 | unsi\_test\_48 | Липецкая обл. | 2022-12-26 |
| 49 | unsi\_test\_49 | Магаданская обл. | 2022-12-26 |
| 50 | unsi\_test\_50 | Московская обл. | 2022-12-26 |
| 51 | unsi\_test\_51 | Мурманская обл. | 2022-12-26 |
| 52 | unsi\_test\_52 | Нижегородская обл. | 2022-12-26 |
| 53 | unsi\_test\_53 | Новгородская обл. | 2022-12-26 |
| 54 | unsi\_test\_54 | Новосибирская обл. | 2022-12-26 |
| 55 | unsi\_test\_55 | Омская обл. | 2022-12-26 |
| 56 | unsi\_test\_56 | Оренбургская обл. | 2022-12-26 |
| 57 | unsi\_test\_57 | Орловская обл. | 2022-12-26 |
| 58 | unsi\_test\_58 | Пензенская обл. | 2022-12-26 |
| 59 | unsi\_test\_59 | Пермский край | 2022-12-26 |
| 60 | unsi\_test\_60 | Псковская обл. | 2022-12-26 |
| 61 | unsi\_test\_61 | Ростовская обл. | 2022-12-26 |
| 62 | unsi\_test\_62 | Рязанская обл. | 2022-12-26 |
| 63 | unsi\_test\_63 | Самарская обл | 2022-12-26 |
| 64 | unsi\_test\_64 | Саратовская обл. | 2022-12-26 |
|  |  |  |  |
| 65 | unsi\_test\_65 | Сахалинская обл. | 2022-12-26 |
| 66 | unsi\_test\_66 | Свердловская обл. | 2022-12-26 |
| 67 | unsi\_test\_67 | Смоленская обл. | 2022-12-26 |
| 68 | unsi\_test\_68 | Тамбовская обл. | 2022-12-26 |
| 69 | unsi\_test\_69 | Тверская обл. | 2022-12-26 |
| 70 | unsi\_test\_70 | Томская обл. | 2022-12-26 |
| 71 | unsi\_test\_71 | Тульская обл. | 2022-12-26 |
| 72 | unsi\_test\_72 | Тюменская обл. | 2022-12-26 |
| 73 | unsi\_test\_73 | Ульяновская обл. | 2022-12-26 |
| 74 | unsi\_test\_74 | Челябинская обл. | 2022-12-26 |
| 75 | unsi\_test\_75 | Забайкальский край | 2022-12-26 |
| 76 | unsi\_test\_76 | Ярославская обл. | 2022-12-26 |
| 77 | unsi\_test\_77 | Москва г. | 2022-12-26 |
| 78 | unsi\_test\_78 | Санкт-Петербург г. | 2022-12-26 |
| 79 | unsi\_test\_79 | Еврейская Аобл | 2022-12-26 |
| 80 | unsi\_test\_83 | Ненецкий АО | 2022-12-26 |
| 81 | unsi\_test\_86 | Ханты-Мансийский АО | 2022-12-26 |
| 82 | unsi\_test\_87 | Чукотский АО | 2022-12-26 |
| 83 | unsi\_test\_89 | Ямало-Ненецкий АО | 2022-12-26 |
| 84 | unsi\_test\_91 | Крым Респ | 2022-12-26 |
| 85 | unsi\_test\_92 | Севастополь г. | 2022-12-26 |

**2.2. Описание структуры БД.**

Описание структуры БД, данное в табличной форме, включает в себя перечень основных схем и краткое описание их назначения.

| **№ пп** | **Имя схемы** | **Описание** |
| --- | --- | --- |
| 1 | gar\_fias | Данные, импортированных из адресного архива, и обработанные парсерами. Физическая модель данных приведена на Рис\_3. |
| 2 | gar\_fias\_pcg\_load | Функциональная схема, обеспечивает запоминание результатов работы парсеров. |
| 3 | gar\_link | Обеспечение коммуникаций, в частности с FOREIGN SERVERS. Расширение dblink здесь. |
| 4 | gar\_tmp | Промежуточные структуры, переход от структуры ГАР-ФИАС к структуре ЕС-НСИ. |
| 5 | gar\_tmp\_pcg\_trans | Основная функциональная схема |
| 6 | gar\_version | Версионирование импорта данных из адресного архива |
| 7 | gar\_version\_pcg\_support | Функциональная схема для поддержки версионирования импорта. |
| 8 | export\_version | Версионирование экспорта данных в адресную базу. |
| 9 | unnsi | Адресная схема на отдалённом сервере (Только Foreign Tables) |
| 10 | unsi | DEPRICATED. Адресная схема на отдалённом сервере (Только Foreign Tables) |
| 11 | public | Разная вспомогательная мелочь. |

**2.3. Индексное покрытие и критерии уникальности.**

Обновления адресных данных затронут три таблицы из схемы «unnsi»:

unnsi.adr\_area — адресные пространства,

unnsi.adr\_street — элементы дорожно планировочной структуры

unnsi.adr\_house — дома, строения

Каждая из таблиц имеет своё собственное индексное покрытие, обеспечивающее приемлимую производительность типовых запросов (единичных), поступающих из приложений к адресной базе. Назовём это индексное покрытие эксплуатационным. В противоположность ему индексное покрытие, разворачиваемое в процессе обновления на региональных фрагментах адресных баз должно обеспечивать запросы совершенно другого типа, возникающие в процессе пакетной (массовой) обработки данных. Такое покрытие назовём процессинговым. Сравним два типа покрытия на примере таблицы «adr\_area».

Пример эксплуатационных индексов для таблицы «adr\_area».

C**REATE UNIQUE INDEX adr\_area\_ak1**

**ON unnsi.adr\_area USING btree**

**(id\_country ASC NULLS LAST, id\_area\_parent ASC NULLS LAST, id\_area\_type ASC NULLS LAST, upper(nm\_area::text) ASC NULLS LAST) WHERE id\_data\_etalon IS NULL;**

**CREATE INDEX adr\_area\_i1**

**ON unnsi.adr\_area USING btree (id\_area\_parent ASC NULLS LAST);**

**CREATE INDEX adr\_area\_i2**

**ON unnsi.adr\_area USING btree (id\_area\_type ASC NULLS LAST) ;**

**CREATE INDEX adr\_area\_i3**

**ON unnsi.adr\_area USING btree (id\_country ASC NULLS LAST);**

**CREATE INDEX adr\_area\_i4**

**ON unnsi.adr\_area USING btree (kd\_timezone ASC NULLS LAST);**

**CREATE INDEX adr\_area\_i5**

**ON unnsi.adr\_area USING btree (id\_data\_etalon ASC NULLS LAST);**

**CREATE INDEX adr\_area\_i6**

**ON unnsi.adr\_area USING btree (id\_country ASC NULLS LAST, upper(nm\_area\_full::text)**

**ASC NULLS LAST);**

**CREATE INDEX adr\_area\_i7**

**ON unnsi.adr\_area USING btree (nm\_fias\_guid ASC NULLS LAST);**

**CREATE INDEX adr\_area\_i8**

**ON unnsi.adr\_area USING btree (id\_country ASC NULLS LAST, upper(nm\_area::text)**

**ASC NULLS LAST);**

**CREATE INDEX adr\_area\_i9**

**ON unnsi.adr\_area USING btree (kd\_okato ASC NULLS LAST);**

**CREATE INDEX index\_nm\_area\_on\_adr\_area\_trigram**

**ON unnsi.adr\_area USING gin (nm\_area gin\_trgm\_ops);**

Пример процессинговых индексов для таблицы «adr\_area»

**CREATE UNIQUE INDEX \_xxx\_adr\_area\_ak1**

**ON gar\_tmp.adr\_area USING btree**

**(id\_country ASC NULLS LAST, id\_area\_parent ASC NULLS LAST, id\_area\_type ASC NULLS LAST, upper(nm\_area::text) ASC NULLS LAST)**

**WHERE id\_data\_etalon IS NULL AND dt\_data\_del IS NULL;**

**CREATE UNIQUE INDEX \_xxx\_adr\_area\_ie2**

**ON gar\_tmp.adr\_area USING btree**

**(nm\_fias\_guid ASC NULLS LAST)**

**WHERE id\_data\_etalon IS NULL AND dt\_data\_del IS NULL;**

Набор индексов для целей обработки короче, необходимы всего лишь два, но оба уникальных. Первый процессинговый уникальный индекс **\_xxx\_adr\_area\_ak1** отличается от эксплуатационного **adr\_area\_ak1** условием WHERE, при этом не рассматриваются записи с признаком отложенного удаления (**dt\_data\_del IS NULL)** поскольку обновление совершается данными актуальными на дату загрузки (вводим допущение, что разность между датой загрузки и максимальной из дат отложенного удаления является достаточно больной) и все отложенные удаления находятся в прошлом. Уникальность второго процессингового индекса **\_xxx\_adr\_area\_ie2** необходима для однозначного определения экземпляров записей в данных «Гар-Фиас» по значению атрибута «object\_guid».

Индексное покрытие в региональных секциях возвращается к эксплуатационному в три этапа в течении процесса постобработки: 1) создание неуникального эксплуатационного индексного покрытия 2) поиск артефактов нарушающих эксплуатационную уникальность, 3) создание уникального эксплуатационного индексного покрытия. После этого обновления, загруженные в целевую адресную базу, не нарушат общей уникальности.

Аналогично для таблиц «adr\_street» и «adr\_house».

Эксплуатационные индексы для «adr\_street».

**CREATE UNIQUE INDEX adr\_street\_ak1**

**ON unnsi.adr\_street USING btree**

**(id\_area ASC NULLS LAST, upper(nm\_street::text) ASC NULLS LAST, id\_street\_type ASC NULLS LAST)**

**WHERE id\_data\_etalon IS NULL;**

**CREATE INDEX adr\_street\_i1**

**ON unnsi.adr\_street USING btree (id\_area ASC NULLS LAST);**

**CREATE INDEX adr\_street\_i2**

**ON unnsi.adr\_street USING btree (id\_street\_type ASC NULLS LAST);**

**CREATE INDEX adr\_street\_i3**

**ON unnsi.adr\_street USING btree (id\_data\_etalon ASC NULLS LAST);**

**CREATE INDEX adr\_street\_i4**

**ON unnsi.adr\_street USING btree (nm\_fias\_guid ASC NULLS LAST);**

Процессинговые индексы для «adr\_street».

**CREATE UNIQUE INDEX \_xxx\_adr\_street\_ak1**

**ON gar\_tmp.adr\_street USING btree**

**(id\_area ASC NULLS LAST, upper(nm\_street::text) COLLATE pg\_catalog."default" ASC NULLS LAST, id\_street\_type ASC NULLS LAST)**

**WHERE id\_data\_etalon IS NULL AND dt\_data\_del IS NULL;**

**CREATE UNIQUE INDEX \_xxx\_adr\_street\_ie2**

**ON gar\_tmp.adr\_street USING btree (nm\_fias\_guid ASC NULLS LAST)**

**WHERE id\_data\_etalon IS NULL AND dt\_data\_del IS NULL;**

Эксплуатационные индексы для «adr\_house».

**CREATE UNIQUE INDEX adr\_house\_ak1**

**ON unnsi.adr\_house USING btree**

**(id\_area ASC NULLS LAST, upper(nm\_house\_full::text) ASC NULLS LAST, id\_street ASC NULLS LAST)**

**WHERE id\_data\_etalon IS NULL;**

**CREATE INDEX adr\_house\_i1**

**ON unnsi.adr\_house USING btree (id\_area ASC NULLS LAST);**

**CREATE INDEX adr\_house\_i2**

**ON unnsi.adr\_house USING btree (id\_street ASC NULLS LAST);**

**CREATE INDEX adr\_house\_i3**

**ON unnsi.adr\_house USING btree (id\_house\_type\_1 ASC NULLS LAST);**

**CREATE INDEX adr\_house\_i4**

**ON unnsi.adr\_house USING btree (id\_house\_type\_2 ASC NULLS LAST);**

**CREATE INDEX adr\_house\_i5**

**ON unnsi.adr\_house USING btree (id\_house\_type\_3 ASC NULLS LAST);**

**CREATE INDEX adr\_house\_i7**

**ON unnsi.adr\_house USING btree (nm\_fias\_guid ASC NULLS LAST);**

**CREATE INDEX adr\_house\_idx1**

**ON unnsi.adr\_house USING btree (id\_area ASC NULLS LAST)**

**WHERE id\_street IS NULL;**

Процессинговые индексы для «adr\_house».

**CREATE UNIQUE INDEX \_xxx\_adr\_house\_ak1**

**ON gar\_tmp.adr\_house USING btree**

**(id\_area ASC NULLS LAST, upper(nm\_house\_full::text) ASC NULLS LAST, id\_street ASC NULLS LAST, id\_house\_type\_1 ASC NULLS LAST)**

**WHERE id\_data\_etalon IS NULL AND dt\_data\_del IS NULL;**

**CREATE UNIQUE INDEX \_xxx\_adr\_house\_ie2**

**ON gar\_tmp.adr\_house USING btree**

**(nm\_fias\_guid ASC NULLS LAST)**

**WHERE id\_data\_etalon IS NULL AND dt\_data\_del IS NULL;**

**2.4. Особенности типизации адресных объектов в ГАР-ФИАС .**

В ЕС НСИ используется так называемая «плоская» типизация. Все адресные объекты имеют перечислимое и однозначно определённое множество типов. В качестве примера рассмотрим справочники типов адресных объектов, улиц и домов.

Рис.7 Справочник адресных объектов.

Аналогично типизируются элементы улично-дорожной сети и дома. Ниже на рисунках 8 и 9 приведены соответствующие таблицы.



Рис.8 Справочник элементов улично-дорожной сети.

****

Рис.9 Справочник домов.

В отличие от ЕС НСИ, в ГАР ФИАС ведено понятие уровня адресного объекта (которого нет в ЕС НСИ) и различных уровнях объекты с совпадающими наименованиями будут иметь различные идентификаторы типов. Кроме того элементы уровней, отмеченные как устаревшие, отмечены текстом и датой окончания актуальности меньшей, чем текущая дата (в таблице выделены жёлтым цветом). Могут быть связаны с актуальными элементами. Пример тому районы в городах Дагестана отнесённые к уровню 14 были отброшены при обработке, как устаревшие. Результат — десятки не найденных адресов.

Рис.10 Уровни адресных объектов.

Далее, приведён фрагмент списка городских районов Республики Дагестан. Выделены желтым цветом актуальные записи, соотнесённые с устаревшим, неактуальным типом.

Другой проблемой, возникающей при загрузке данных из ГАР-ФИАС в ЕС НСИ является то объекты имеющие одинаковое название типов, могут иметь различные идентификаторы типов и различные идентификаторы уровней. Пример приводится ниже, на рисунке 12. Видно что районы к ID типа = 151 относятся к уровню «элементы дорожно-транспортной сети», в то-же время остальные типы районов – нет.

Для разрешения таких неопределённостей и для однозначного разделения адресных объектов и элементов дорожно-транспортной сети, было разработано семейство вспомогательных таблиц «gar\_tmp.xxx»

Рис.11 Фрагмент списка районов Республики Дагестан.

****

Рис.12 Фрагмент списка типов районов Республики Дагестан.

**2.4.1. Семейство вспомогательных таблиц “gar\_tmp.xxx\_”.**

Семейство таблиц необходимо для поддержки преобразования структры данных ГАР-ФИАС в структру ЕС НСИ. Остановимся на трёх из них: “gar\_tmp.xxx\_adr\_area\_type”, “gar\_tmp.xxx\_adr\_street\_type”, “gar\_tmp.xxx\_adr\_house\_type”.

 Рис.13 Структура таблицы «gar\_tmp.\_xxx\_adr\_area\_type

 Рис.14 Структура таблицы «gar\_tmp.\_xxx\_adr\_street\_type

 Рис.15 Структура таблицы «gar\_tmp.\_xxx\_adr\_house\_type

Их назначение заключается преобразовании множества типов ГАР-ФИАС в тип ЕС НСИ. Каждая таблица имеет первичный ключ «fias\_row\_key» (выделен зелёным цветом), с помощью которого выполняется связь с соответствующим справочником ЕС НСИ. Дополнительным назначением таблиц «gar\_tmp.xxx\_adr\_area\_type», «gar\_tmp.xxx\_adr\_street\_type» является хранение однозначно определённых правил для фильтрации и разделение данных из таблицы «gar\_fias.as\_addr\_obj» («Классификатор адресообразующих элементов») на собственно адресообразующие элементы и элементы дорожно-транспортной структуры.

В качестве примера рассмотрим результат выполнения сервисной функции:

 Рис.16 Данные из таблиц «gar\_tmp.xxx\_adr\_area\_type» и «gar\_tmp.adr\_area\_type»

Данные из двух таблиц связаны между собой по столбцу «fias\_row\_key», видна реализация отношения 1:N между типами адресных объектов ЕС НСИ (столбец «id\_area\_type») и ГАР-ФИАС (столбец «fias\_ids»). На его основе выполняется фильтрация данных и преобразование типов.

Аналогично могут быть представлены данные по отношениям типов элементоа дорожно-транспортной сети.

****

Рис.17 Данные из таблиц «gar\_tmp.xxx\_adr\_street\_type» и «gar\_tmp.adr\_street\_type».

**2.5. Дата-версия.**

Существует схема “gar\_version” в которой регистрируются факты поступления данных на обработку, рассмотрим две таблицы ”garfias\_version” и “garfias\_files\_by\_region”

 Рис.18 Данные из таблицы «garfias\_version».

Это единственное место, где столбец «nm\_garfias\_version» несёт определённую смысловую нагрузку, на него наложено ограничение уникальности. Одна запись связана с группой загруженных и обработанных XML-файлов. Во всех остальных случаях она носит информационный характер. Обычно «Дата-версия» берётся с сайта GAR-FIAS.

**3. Типовые варианты использования, часть 1. Последовательное выполнение.**

В разделе будут рассмотрены типовые варианты использования. Все варианты будут пояснены на последовательном выполнении заданий, описанных в CSV-сценарии. Это будет касаться как работы с одним хостом, так и с группой хостов.

**3.1) CSV-сценарий**

CSV-сценарий является основной и единственной структурой, посредством которой описываются задания. Поскольку «fiad\_loader» является пакетной программой, то CSV-сценарий может по другому называться как: «пакет команд», «batch». Как его создать такой сценарий? Какие правила и ограничения должны при этом соблюдаться ?

Отдельно хочется остановиться на применимости и распространении CSV-формата. Широко используемая структура, имеется семейство инструментов, работающих с ним. Например «miller» - инструмент для обработки данных в CSV-формате с индексированием имен. <https://github.com/johnkerl/miller>. Этот продукт ранее применяется в «fias\_loader», сейчас я упомянул его в качестве примера.

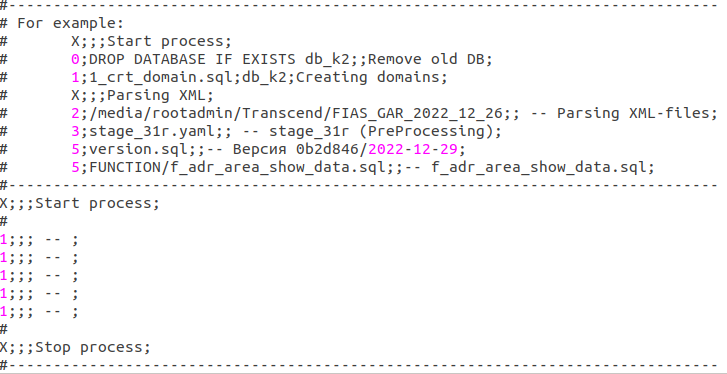


Рис 3.1 Шаблон для CSV-сценариев.

Утилита «load\_mainTemp.py» создаёт шаблон CSV-сценария. Пример вызова утилиты:

$>load\_mainTemp.py > test.csv

Рассмотрим структуру CSV-сценария. Номера колонок начинаются с 0. Созданный шаблон содержит комментарии, облегчающие создание нового CSV-сценария.

| Номер колонки | Описание |
| --- | --- |
| 0 | Выполняемое действие:  0 — непосредственное выполнение SQL-КОМАНДЫ.  1 — выполнение SQL-сценария из внешнего файла  **2 — путь к каталогу с XML-данными**. Далее выполняются: внешний parser и вызов хранимки, запоминающей обработанные данные в схеме GAR\_FIAS.  3 - выполнить Большую команду «Предобработка данных».  4 - выполнить Большую команду «Выполнить обработку данных».  6 - выполнить Большую команду «Выполнить постобработку данных».  5 - Собрать командный SQL-пакет. |
| 1 | Команда, имя файла, либо путь к каталогу |
| 2 | Имя БД — опция. Обычно отсутствует, но если указана, то отменяет назначение БД, сделанное в параметрах командной строки. |
| 3 | Описание действия (в процессе выполнения сценария выводится на консоль и в LOG-файл). |

Рис 3.2 Определение действий, допустимых в CSV-сценарии.

Определение термина «Большая команда» будет дано ниже. Более подробно, вопросы связанные с управлением «Большими командами» будут рассмотрены в следующей главе. Далее приступим к первому CSV-сценарию.

Задача: необходимо собрать статистику с кластера БД, размещённого на 127.0.0.1 (локальный хост). Ранее на этом хосте был выполнен parsing адресных данных для двух регионов. Переходим в каталог «/home/rootadmin/example»:

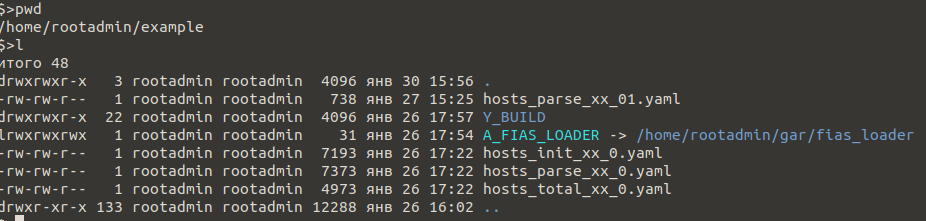
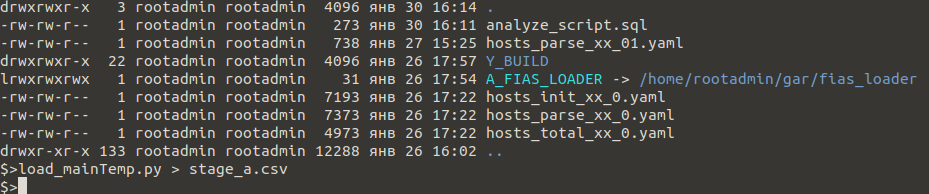


Рис 3.3 Каталог с демонстрационным примером.

Переносим в каталог файл «analyze\_script.sql» и создаём CSV-сценарий.

 Рис 3.4 Создание CSV-сценария.

Вновь созданный CSV-сценарий будет состоять из двух строк: в первой из них выполняется SQL-команда (выполняемое действие - 0), во второй SQL-скрипт (выполняемое действие - 1).

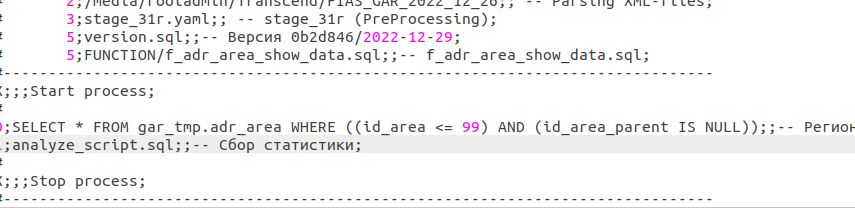


Рис 3.5 Созданный сценарий.

$>load\_mainGar\_2.py hosts\_parse\_xx\_01.yaml stage\_a.csv . '2022-12-26'

... Адыгея Респ, id\_region: 2

... Start process

... -- Регион

... -- Сбор статистики

... Stop process

... Бурятия Респ, id\_region: 3

... Start process

... -- Регион

... -- Сбор статистики

... Stop process

$>

Смотрим результаты, они всегда в файлах process.log/process.out/process.err .

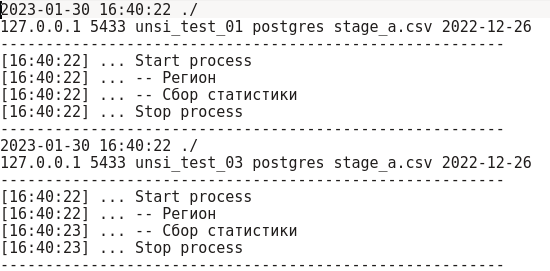


Рис 3.6 process.log.

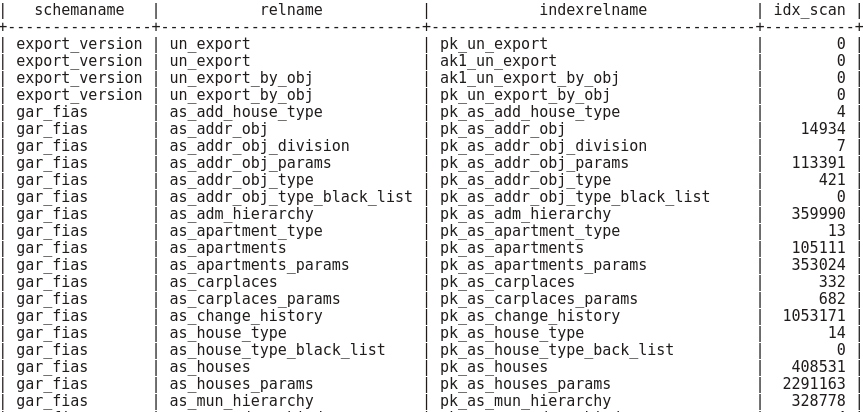


Рис 3.7 Фрагмент process.out.

**3.2) Большие команды.**

Исходя из написанного выше, видно, что одна строка в CSV-сценарии, инициирует одно и более действий в БД. Такое действие может быть единичным SQL-запросом, либо выполнением SQL-сценария, либо вызовом хранимой функции/процедуры. При увеличении размера сценария, сложность управления им будет нарастать. Например: как выключить строку? Без особых размышлений можно ответить: Закомментировать. А если строк много и среди них есть обращения к хранимым процедурам/функциям, параметры которых нужно периодически изменять ?? Задача становится громоздкой. Первые варианты сценария, выполняющего предобработку адресных данных имели более 30 строк и имели тенденцию к росту. С одной стороны их немного, а с другой: …. как легко и быстро выключить этап реиндексации сторонней базы перед повторным запуском сценария ?? Задача управления CSV-сценарием с типами действий 0, 1 становится трудно разрешимой, если в сценарий добавить XML-parsing или операции с внешними данными (файлы, каталоги).

Необходимо структурировать множество действий, объединив часть их в структуры, называемые «Большая команда». При этом базовый функционал оформлен в виде хранимых процедур ти SQL-сценариев, управляющий функционал оформлен в виде методов из Python-программ, управляющие параметры находятся в специальных YAML-файлах. Сейчас сществует 4 больших команды, каждая из которых идентифицируются по коду действия в структуре CSV-сценария.

| Команда | Обращение в сценарии | Описание |
| --- | --- | --- |
| 2 | 2;/media/rootadmin/FIAS\_GAR\_2022\_12\_26/01;;-- Region XML-file; (НЕТ YAML- файла). | Parsing адресных данных. |
| 3 | 3;stage\_3.yaml;; -- Пред-обработка; | Пред-обработка |
| 4 | 4;stage\_4.yaml;; -- Обработка; | Обработка |
| 6 | 6;stage\_6.yaml;; -- Постобработка и Выгрузка инкрементальных обновлений; | Постобработка и Выгрузка инкрементальных обнов-лений |

Рис 3.8 Список Больших команд.

Большая команда №2 была первой, она практически не параметризуема, для каждого XML-файла, встреченного в указанном каталоге, имя которого соответствовало вычисленному ключу, выполнялся parsing, результаты которого запоминались в схеме GAR\_FIAS. Следующие три больших команды управлялись набором параметров, указанных в YAML-файлах. В следующей главе будет изложено управление такими командами.

**3.3) Выполнение команды на одном хосте.**

В этом разделе рассмотрим выполнение таких операций как: «Parsing», «Предобработка», «Обработка», «Обновление пакетов» и «Накатка патчей». Все операции будут выполняться на одном хосте. На практике обычно применяются множественные операции, при которой выбирается группа хостов, но проще объяснить применение «Fias\_Loader» на одном, отдельно взятом хосте, при этом все особенности операций будут сохранены и при множественных операциях.

**3.3.1) Parsing.**

Хост 127.0.0.1, порт 5433, база «unsi\_test\_02» 'Башкортостан Респ. Дата-версия = «2022-12-26». Переходим в каталог «~/example», для выполнения примера необходим каталог «02» (смотри имя базы).

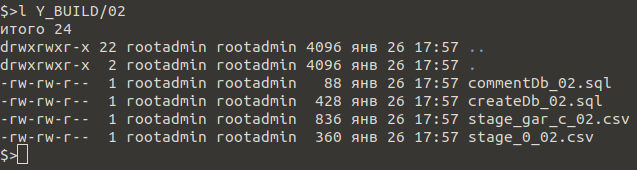


Рис 3.3.1 Состояние перед запуском процесса Parsing. Каталог «Y\_BUILD/02».

Точка старта находится здесь:

$>pwd

/home/rootadmin/example

Для выполнения процесса необходим сценарий «stage\_gar\_c\_02.csv». Ниже, на рис 3.3.2. приводится фрагмент этого сценария.

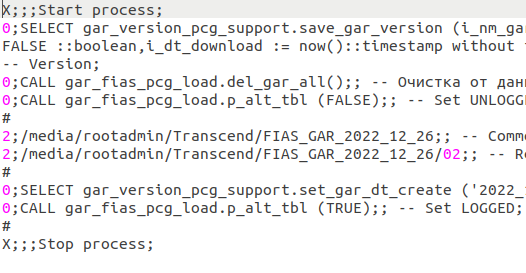


Рис 3.3.2 Фрагмент сценария Parsing для базы «unsi\_test\_02».

Сценарий состоит из команд типа «0» и «2». Команды типа «0» вызывают хранимые поцедуры и функции, смотрим на рисунок, это - SELECT gar\_version\_pcg\_support.save\_gar\_version (), CALL gar\_fias\_pcg\_load.del\_gar\_all(); и т. п. Две больших команды типа «2» выполняют parsing XML-данных. Команды этого типа не зависят от точки старта сценария. **Необходимо запомнить, что в общем случае команды зависят от точки старта процесса (точки старта сценария).**  Следующий пример будет посвящён именно таким командам. Вернёмся к запуску процесса parsing. Загрузчик, будучи, вызванным без параметров, выводит список необходимых параметров.

$>load\_mainGar.py

Version 0.5.5 Build 2023-01-13

Usage: /usr/local/bin/load\_mainGar.py <Host\_IP> <Port> <DB\_name> <User\_name> <Batch\_file\_name> <Path> <Version>

$>

Устанавливаем параметры и запускаем процесс:

$>load\_mainGar.py 127.0.0.1 5433 unsi\_test\_02 postgres Y\_BUILD/02/stage\_gar\_c\_02.csv . '2022-12-26'

... Start process

... -- Version

... -- Очистка от данных

... -- Set UNLOGGED

Все параметры являются позиционными, параметр «Path» принимает значение «.», т.е текущий каталог. Этот параметр необходимо устанавливать только для сценариев, зависящих от точки старта. Процесс будет достаточно длительным, так как «02 - Башкортостан Респ» является большим регионом, по его завершению рассмотрим файлы «process.log», «process.out» и «process.err», которые формируются при каждом запуске загрузчика.

Файл «process.log» - содержит отчёт о выполненном задании. В заголовке файла дата и время старта, точка старта процесса «..». Ниже — параметры соединения с БД, имя сценария, породившего процесс, дата-версия. Стартовая точка процесса, параметр <Path>, устанавливается равной «.». Далее смотрим во что превратились две больших команды типа «2», для наглядности, они скопированы из CSV-сценария и выделены красными.

2023-01-31 12:20:10 ./

127.0.0.1 5433 unsi\_test\_02 postgres Y\_BUILD/02/stage\_gar\_c\_02.csv 2022-12-26

--------------------------------------------------------

[12:20:10] ... Start process

[12:20:10] ... -- Version

[12:20:11] ... -- Очистка от данных

[12:20:13] ... -- Set UNLOGGED

**2;/media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26;; -- Common XML-file;**

[12:20:19] ... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/AS\_ADDHOUSE\_TYPES\_20221226\_2626f853-8a92-4d83-ad73-4539ce780eb0.XML

[12:20:19] ... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/AS\_ADDR\_OBJ\_TYPES\_20221226\_04f2dc05-9918-4f6e-9c4e-2fa06c6cce61.XML

[12:20:19] ... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/AS\_APARTMENT\_TYPES\_20221226\_8dd9eb09-e026-47fe-bc42-2398f92cad5c.XML

[12:20:19] ... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/AS\_HOUSE\_TYPES\_20221226\_c399871c-42a6-49d3-9cae-09766d4d965b.XML

[12:20:19] ... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/AS\_NORMATIVE\_DOCS\_KINDS\_20221226\_5eac7f6d-be77-40d6-825b-6daac06070a7.XML

[12:20:19] ... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/AS\_NORMATIVE\_DOCS\_TYPES\_20221226\_99c46464-14cb-456e-98c1-a32cd4f2833d.XML

[12:20:19] ... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/AS\_OBJECT\_LEVELS\_20221226\_6e97d7bb-9815-410e-81dc-37d79b5ab350.XML

[12:20:19] ... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/AS\_OPERATION\_TYPES\_20221226\_72458699-c8e7-4357-a1b2-510ccf0ce0b7.XML

[12:20:19] ... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/AS\_PARAM\_TYPES\_20221226\_dbbd7cc3-cb69-4472-86f4-94d3baa9f3d3.XML

[12:20:19] ... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/AS\_ROOM\_TYPES\_20221226\_5df4f33f-a479-4e68-a8f3-61ec8a948227.XML

**Одна команда типа 2 инициировала parsing 10-ти общих XML-файлов.**

----------------------------------------------------------------

**Начинается процесс parsing адресных XML-файлов**:

**2;/media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/02;; -- Region XML-file; # Башкортостан Респ**

[12:20:19] ... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/02/AS\_ADDR\_OBJ\_20221226\_9975ed16-195b-414a-84b7-82072e182842.XML

[12:20:53] ... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/02/AS\_ADDR\_OBJ\_DIVISION\_20221226\_d4fd5cee-bf11-463b-a182-65f5f638ca97.XML

[12:20:53] ... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/02/AS\_ADDR\_OBJ\_PARAMS\_20221226\_9c487fc7-f5dc-4365-bbaf-8a05a9dc7643.XML

[12:24:42] ... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/02/AS\_ADM\_HIERARCHY\_20221226\_4e85d608-e3ee-4300-a473-2130257d9ab9.XML

[12:54:41] ... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/02/AS\_APARTMENTS\_20221226\_b7a9746c-a658-4b51-ad0a-88184249a440.XML

[13:04:19] ... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/02/AS\_APARTMENTS\_PARAMS\_20221226\_a5a5e4ff-b547-44ed-949d-c0486978cf35.XML

[13:30:29] ... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/02/AS\_CARPLACES\_20221226\_b2eaa60c-925c-4b8c-a2e4-6fb439cb7470.XML

[13:30:32] ... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/02/AS\_CARPLACES\_PARAMS\_20221226\_d467a028-2973-4bab-a732-7856587d641d.XML

[13:30:39] ... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/02/AS\_CHANGE\_HISTORY\_20221226\_cb59a491-037f-48a3-a13a-0dde6a1dd9f2.XML

[14:10:48] ... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/02/AS\_HOUSES\_20221226\_da8338e8-ef71-4d25-b5a4-dffdcf63c06c.XML

[14:32:48] ... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/02/AS\_HOUSES\_PARAMS\_20221226\_d1d00429-5a47-4ef9-9cac-2cd963773cd0.XML

[15:53:08] ... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/02/AS\_MUN\_HIERARCHY\_20221226\_a6df5108-0f40-43ed-b956-bf680bb9ffe6.XML

[16:17:26] ... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/02/AS\_NORMATIVE\_DOCS\_20221226\_619b4b6e-2592-4d7a-b167-012316935eb1.XML

[16:21:54] ... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/02/AS\_REESTR\_OBJECTS\_20221226\_85eaeee7-2453-45e9-9fd7-1503fd9c668e.XML

[16:40:10] ... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/02/AS\_ROOMS\_20221226\_cf0fff9b-c983-488b-9fbd-f2b2ee0cd23c.XML

[16:40:23] ... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/02/AS\_ROOMS\_PARAMS\_20221226\_b1b1ffeb-1f49-4114-8915-df90cc5c5d32.XML

[16:41:00] ... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/02/AS\_STEADS\_20221226\_746b0502-7528-49ca-aa34-5bd10536f1fb.XML

[16:49:40] ... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/02/AS\_STEADS\_PARAMS\_20221226\_784b7bc5-2a44-4827-ad99-ec80fb6c7f18.XML

**Процесс заверщён, распарсены 18 региональных адресных файлов.**

[18:04:37] ... -- Data finish

[18:04:39] ... -- Set LOGGED

[18:20:28] ... Stop process

--------------------------------------------------------

Адресная база регион «02» имеет большой объём. Обработка XML-файлов заняла более 5-ти часов на локальном стенде.

**3.3.2 Обновление функциональных схем и накатка патчей.**

Этот раздел необходим для понимания зависимости команд в CSV-сценарии от точки старта процесса. Находимся в каталоге «~/example/1», обновляем функции и процедуры, загружающие XML-данные в базу «unsi\_test\_10». Смотрим git-репозиторий:

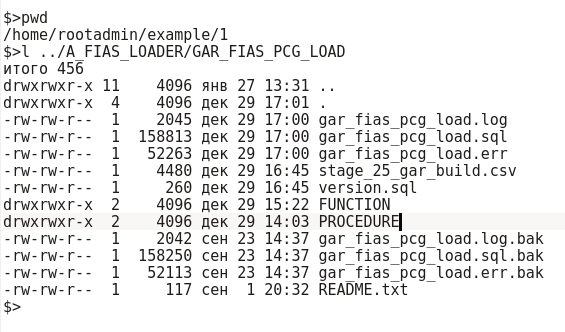


Рис 3.3.3 Фрагмент git-репозитория. Пакет «gar\_fias\_pcg\_load.sql».

База «unsi\_test\_10», необходим CSV-сценарий, создаём его (вообще-то существует набор небольших сервисных сценариев на многие случаи жизни, но учёбы ради создаём новый).

$>load\_mainTemp.py > stage\_x.csv

$>l

итого 12

-rw-rw-r-- 1 1760 янв 31 13:51 stage\_x.csv

drwxrwxr-x 2 4096 янв 31 13:51 .

drwxrwxr-x 4 4096 янв 31 13:39 ..

$>

Далее в любом текстовом редакторе создаём строку с командой типа «1», которая вызовет запуск процесса обновления функциональной схемы.

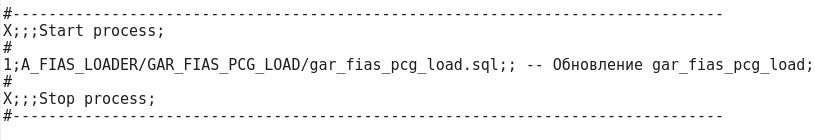


Рис 3.3.4 Фрагмент CSV-сценария. Пакет «gar\_fias\_pcg\_load.sql».

Важен только один факт, пакет «gar\_fias\_pcg\_load.sql» находится в git-репозитории, в папке «GAR\_FIAS\_PCG\_LOAD». Ссылка на git-репозиторий находится на один уровень выше точки старта процесса, это необходимо помнить, ведь команда типа «1», выполняющая накат SQL-скрипта всегда зависит от точки старта процесса. Запускаем обновление:

$>l

drwxrwxr-x 2 rootadmin rootadmin 4096 янв 31 14:15 .

-rw-rw-r-- 1 rootadmin rootadmin 1811 янв 31 14:14 stage\_x.csv

$>load\_mainGar.py 127.0.0.1 5433 unsi\_test\_10 postgres stage\_x.csv .. '2022-12-26'

... Start process

... -- Обновление gar\_fias\_pcg\_load

... Stop process

$>

В течении процесса всегда формируются три файла «process.log», «process.out» и «process.err». Файл «process.sql» рассматривать не будем, это анахронизм, который исчезнет в следующих версиях.

Файл «process.log» - содержит основной отчёт о выполненном задании. Ниже приводится полный вариант такого файла из текущего примера.

2023-01-31 14:16:37 **../**

127.0.0.1 5433 unsi\_test\_10 postgres stage\_x.csv 2022-12-26

--------------------------------------------------------

[14:16:37] ... Start process

[14:16:37] ... -- Обновление gar\_fias\_pcg\_load

[14:16:39] ... Stop process

--------------------------------------------------------

В заголовке дата и время старта, точка старта процесса «..». Ниже — параметры соединения с БД, имя сценария, породившего процесс, дата-версия.

Резюме: при использовании в сценарии команд типа «1» (исполнение sql-файлов), необходимо помнить о стартовой точке процесса, это параметр <Path>.

Файл «process.out» - это снимок всего, что выводит клиент Postgres, программа psql, в поток stdout. Файл «process.err» - соответственно — поток stderr.

**3.3.3) Предоработка и обработка.**

Необходимо обработать загруженные в схему «gar\_fias» адресные данные. Диаграмма потоков данных, иллюстрирующая процесс обновления, приложена к настоящему документу («Рис\_4\_Процесс\_обработки.png»).

CSV-сценарий «stage\_34.csv» является очень простым, он состоит всего лишь из трёх команд. Ниже приведён текст сценария:

X;;;Start process, Обработка;

#

0;SELECT gar\_link.f\_server\_is();; -- gar\_link.f\_server\_is;

3;stage\_31r.yaml;; -- stage\_31r (Предобработка данных);

4;stage\_41r.yaml;; -- stage\_41r (Обработка данных);

#

X;;;Stop process;

Состав сценария:

1) Простая команда типа «0», выполняющая идентификацию стороннего сервера.

2) Большая команда типа «3», выполняющая предобработку адресных данных, её параметры

и переключатели, находятся в управляющем YAML-файле «stage\_31r.yaml».

3) Большая команда типа «4», выполняющая обработку, всё, что управляет её функционированием находится в YAML-файле «stage\_41r.yaml».

По поводу идентификации стороннего сервера. Обработка адресных данных требует связи с целевой БД, содержащей обновляемую адресную схему, такая связь оформлена в виде FDW (Foreign Data Wrapper). Поэтому в начале цикла обработки выполняем запрос, дающий ответ на вопрос «С каким внешним сервером работаем сейчас ?».

Ниже приведены результаты выполнения функции и представления из схемы «gar\_link»:

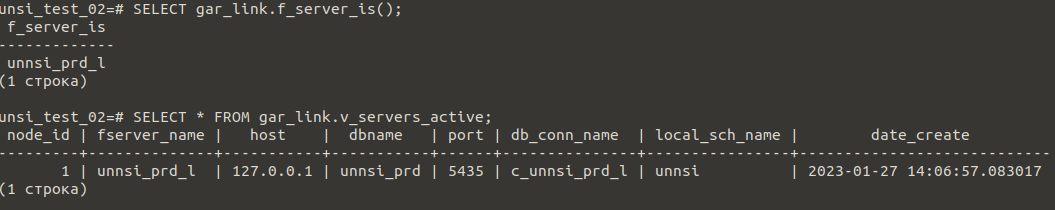


Рис 3.3.5 Список сторонних серверов.

Помним, что текущим активным сервером является «unnsi\_prd\_l», ID сервера = 1. Во всех база демонстрационного кластера сторонние таблицы связанные с сервером №1, находятся в схеме «unnsi».

**3.3.3.1) Описание управляющего YAML-файла «stage\_31r.yaml».**

Ниже приведён текст файла:

#

# 2022-03-11 - Stage\_3. Новая реализация, только управление процессами, вызовы функций

# в коде "stage\_3\_proc.py".

# Version 0.3.0 Build 2022-12-06.

#

control\_params:

stage\_3\_I: True

stage\_3\_9: True

stage\_3\_0: True

stage\_3\_1: True

stage\_3\_2: False

stage\_3\_3: True

#

mogrify\_3\_I: False

mogrify\_3\_9: False

mogrify\_3\_0: False

mogrify\_3\_1: False

mogrify\_3\_2: False

mogrify\_3\_3: False

global\_params:

g\_region\_id: 04 # Башкирия

g\_fserver\_nmb: 1 # На 5435 unnsi\_prd

#

g\_adr\_area\_sch: unnsi

g\_adr\_street\_sch: unnsi

g\_adr\_house\_sch: unnsi

#

g\_adr\_area\_sch\_l: gar\_tmp

g\_adr\_street\_sch\_l: gar\_tmp

g\_adr\_house\_sch\_l: gar\_tmp

#

g\_adr\_hist\_sch: gar\_tmp

#

# stage\_3\_I

#

gar\_fias\_crt\_idx:

descr: Создание рабочих индексов в схеме gar\_fias. False-удаление, True-удаление,создание.

params:

p\_skip: False

p\_sw: True

#

# stage\_3\_9

#

gar\_fias\_set\_gap:

descr: Заполнение таблицы с активными предшественниками

params\_adr\_area:

p\_skip: False

params\_adr\_house:

p\_skip: True

#

gar\_fias\_update\_children:

descr: Обработка таблицы с активными предшественниками, переподчинение и деактивация

param\_adr\_area:

p\_skip: False

p\_date\_1: DATE(current\_date) # НЕ ИСПОЛЬЗУЕТСЯ.

p\_obj\_level: [6,7,8]

p\_date\_2: DATE(current\_date) # Дата, на которую выбираются данные !!!

#

# stage\_3\_0

#

gar\_tmp\_set\_logged:

descr: Установка признака LOGGED/UNLOGGED у таблиц в схеме gar\_tmp

params:

p\_skip: False

p\_sw: True # False - unlogged

#

# 2022-12-15 Изменилась сигнатура функции, константы, управляющие процессом очистки.

# ----------------------------------------------------------------------------------

# 1 - Только временные/вспомогательный таблицы. ВСЕГДА

# 2 - Только таблицы-секции ТОЛЬКО ПЕРЕД ЗАГРУЗКОЙ ИЗ ОТДАЛЁННОЙ БАЗЫ

# 6 - logging выгрузки.

# 7 - GAP-таблицы

# 8 - Исторические таблицы В СЛУЧАЕ КРАЙНЕЙ НЕОБХОДИМОСТИ.

# 9 - Прототипы справочников В СЛУЧАЕ КРАЙНЕЙ НЕОБХОДИМОСТИ.

gar\_tmp\_clear\_tbl:

descr: Очистка данных во временной схеме.

params:

p\_skip: False

p\_sw: [1,2]

#

gar\_tmp\_switch\_indexies:

descr: Смена индексного покрытия в отдалённых таблицах

params:

p\_skip: True

p\_skip\_adr\_object: True

params\_street:

p\_uniq\_sw: True

params\_house:

p\_uniq\_sw: True

params\_objects:

p\_uniq\_x2: True

#

# stage\_3\_1

#

unload\_data\_adr\_area\_type:

descr: Загрузка данных из таблицы ADR\_AREA\_TYPE

params:

p\_skip: False

#

unload\_data\_adr\_street\_type:

descr: Загрузка данных из таблицы ADR\_STREET\_TYPE

params:

p\_skip: False

#

unload\_data\_adr\_house\_type:

descr: Загрузка данных из таблицы ADR\_HOUSE\_TYPE

params:

p\_skip: False

#

unload\_data\_adr\_area:

descr: Загрузка регионального фрагмента из таблицы ADR\_AREA

params:

p\_skip: False

#

unload\_data\_adr\_street:

descr: Загрузка регионального фрагмента из таблицы ADR\_STREET

params:

p\_skip: False

#

unload\_data\_adr\_house:

descr: Загрузка регионального фрагмента из таблицы ADR\_HOUSE

params:

p\_skip: False

p\_skip\_adr\_object: True

#

seq\_settings:

descr: Установка последовательностей

params:

p\_skip: False

p\_seq\_name: gar\_tmp.obj\_seq

p\_seq\_hist\_name: gar\_tmp.obj\_hist\_seq

p\_init\_value: 100000000

#

sq\_adr\_area\_sch: gar\_tmp

sq\_adr\_street\_sch: gar\_tmp

sq\_adr\_house\_sch: gar\_tmp

#

# stage\_3\_2

#

dict\_upgrading:

descr: Актуализация справочников (промежуточные XXX структуры)

dict\_params:

p\_schemas: [] # Список обновляемых схем (Здесь может быть эталон).

p\_op\_type: [1] # Список выполняемых операций.

#

params:

descr\_1: Промежуточный справочник типов адресных пространств

p\_skip\_1: True

p\_stop\_list\_1: []

p\_add\_query\_1:

p\_control\_query\_1: SELECT \* FROM gar\_tmp.xxx\_adr\_area\_type ORDER BY 2;

#

descr\_2: Промежуточный справочник типов улиц

p\_skip\_2: True

p\_stop\_list\_2: []

p\_add\_query\_2:

p\_control\_query\_2: SELECT \* FROM gar\_tmp.xxx\_adr\_street\_type ORDER BY 2;

#

descr\_3: Промежуточный справочник типов домовладений

p\_skip\_3: True

p\_stop\_list\_3: []

p\_add\_query\_3:

p\_control\_query\_3: SELECT \* FROM gar\_tmp.xxx\_adr\_house\_type ORDER BY 2;

#

#

# stage\_3\_3

#

data\_aggregation:

descr: Агрегация данных

agg\_params:

p\_skip\_agg: False

p\_descr\_agg: Агрегация параметров

p\_param\_list: [5,6,7,8,10,11]

#

agg\_adr\_area:

p\_skip\_adr\_area: False

p\_descr\_adr\_area: Агрегация адресных пространств

params\_aa:

# "p\_date" - Не ИСПОЛЬЗОВАТЬ СЕЙЧАС.!!!

p\_date: DATE(current\_date) # Дата, на которую выбираются данные НЕТ

p\_obj\_level: 10 # Уровень объекта

p\_oper\_type\_ids: Null # ARRAY [1,2,3,5] # Типы операций НЕТ

#

agg\_adr\_house:

p\_skip\_adr\_house: False

p\_descr\_adr\_house: Агрегация домовладений

params\_ah:

# "p\_date" - Не ИСПОЛЬЗОВАТЬ СЕЙЧАС.!!!

p\_date: DATE(current\_date) # Дата, на которую выбираются данные НЕТ

p\_parent\_obj: Null # Родительский объект НЕТ

#

obj\_fias:

p\_skip\_obj\_fias: False

# 2022-09-28

p\_switch\_adr\_area\_sch: True # True - Локальная схема, False - Отдалённая

p\_switch\_adr\_street\_sch: True # True - Локальная схема, False - Отдалённая

# 2022-09-28

p\_switch\_adr\_house\_sch: True # True - Локальная схема, False - Отдалённая

p\_descr\_obj\_fias: Заполнение управляющей таблицы "gar\_tmp.xxx\_obj\_fias"

Процесс предобработки адресных данных, запускаемый большой командой «3;stage\_31r.yaml;» управляется набором параметров, сгруппированных в 8 главных секций:

| **Номер секции** | **Наименование** | **Объединяет** | **Описание** |
| --- | --- | --- | --- |
| 1 | control\_params: |  | Управляет выполнением секций «stage\_3\_x». |
| 2 | global\_params: |  | Глобальные параметры, общие для всех секций. |
| 3 | stage\_3\_I | gar\_fias\_crt\_idx: | Переиндексация схемы «gar\_fias». |
| 4 | stage\_3\_9 | gar\_fias\_set\_gap:  gar\_fias\_update\_children: | Заполнение и обработка таблицы с активными предшественниками. |
| 5 | stage\_3\_0 | gar\_tmp\_set\_logged:  gar\_tmp\_clear\_tbl:  gar\_tmp\_switch\_indexies: | Установки в таблицах схемы «gar\_tmp», очистка их, управление индексным покрытием в адресной схеме отдалённой базы (при необходимости). |
| 6 | stage\_3\_1 | unload\_data\_adr\_area\_type:  unload\_data\_adr\_street\_type:  unload\_data\_adr\_house\_type:  unload\_data\_adr\_area:  unload\_data\_adr\_street:  unload\_data\_adr\_house:  seq\_settings: | **Загрузка справочников и региональных фрагментов из отдаленной схемы.** |
| 7 | stage\_3\_2 | **DEPRICATED** | Импорт и обновление справочников из схемы GAR\_FIAS. НЕ ИСПОЛЬЗОВАТЬ |
| 8 | stage\_3\_3 | agg\_params:  agg\_adr\_area:  agg\_adr\_house:  obj\_fias: | Агрегация данных и параметров, подготовка к обновлению адресной базы. |

Рис 3.3.6 Описание основных частей управляющего файла «stage\_3.yaml».

Процесс, запущенный большой командой «3;stage\_31r.yaml;» фактически состоит из 6-ти этапов (секций), выполняемых строго последовательно. Следует отметить, что при выполнении секции «stage\_3\_1», необходимо откуда-то получить и передать в запускаемый процесс id адресного региона (в терминах ЕС НСИ). При запуске процесса на одном хосте id адресного региона берётся из файла «stage\_32\_r.yaml, отсюда-же забирается ID стороннего сервера. Они используются в секции «stage\_3\_1», описание секции **выделено красным.**

global\_params:

g\_region\_id: 04 # Башкирия

g\_fserver\_nmb: 1 # На 5435 unnsi\_prd

По поводу управления секциями, если в «control\_params:» у соответствующей секции стоит значение «False», то она выполняться не будет. Смотрим на заголовок файла, видим что stage\_3\_2: False, поэтому секция «stage\_3\_2» не будет выполняться ни при каких условиях. Кроме того, почти в каждой секции есть параметр содержащий фразу «Skip». Это логический параметр, принимающий значения True либо False. Смотрим на заголовок файла:

control\_params:

…………….

stage\_3\_0: True

stage\_3\_1: True – Секция выполняется ВСЕГДА, но в теле секции есть следующее:

unload\_data\_adr\_house:

descr: Загрузка регионального фрагмента из таблицы ADR\_HOUSE

params:

p\_skip: False

p\_skip\_adr\_object: True Выгрузка регионального фрагмента из таблицы «adr\_objects» будет пропускаться.

В завершении краткого обзора структуры управляющего файла ещё раз остановимся на программах, обеспечивающих настройку и запуск процесса предобработки (смотрим диаграмму классов, «Рис\_6\_Диаграмма\_классов.png»):

| #пп | Наименование | Описание |
| --- | --- | --- |
| 1 | load\_mainStage3.py | «Большая команда 3». Запуск процесса предобработ-ки. Управляется «stage\_3.yaml». |
| 2 | GarProcess/stage\_3\_yaml.py | Разбор YAML-файла. Параметры из него, становятся свойствами соответствующих методов. |
| 3 | GarProcess/stage\_3\_proc.py | Patterns вызовов хранимых функций и процедур, используемых в «load\_mainStage3.py». |

Рис 3.3.7 Основные программы, используются на этапе предобработки.

Запускаем сценарий, предварительно выключив последнюю команду. Включение — Выключение команд выполняется при помощи текстового редактора.

X;;;Start process, Обработка;

#

0;SELECT gar\_link.f\_server\_is();; -- gar\_link.f\_server\_is;

3;stage\_31r.yaml;; -- stage\_31r (Предобработка данных);

**### 4;stage\_41r.yaml;; -- stage\_41r (Обработка данных);**

#

X;;;Stop process;

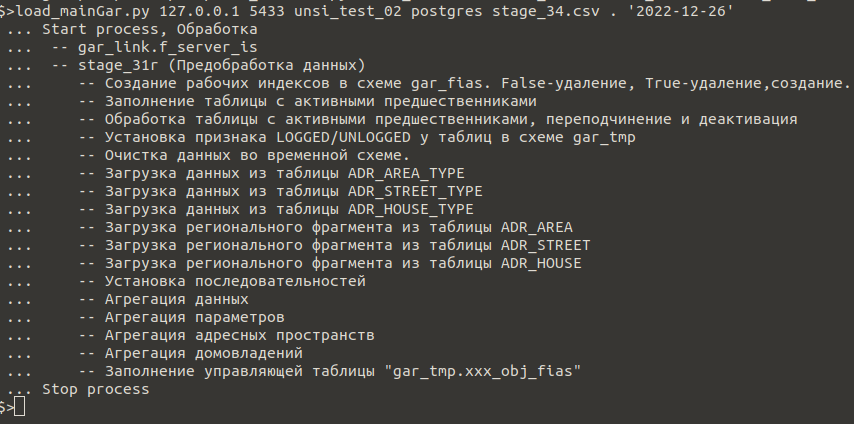


Рис 3.3.8 Запуск и выполнение процесса предобработки.

Процесс будет достаточно длительным, «02 - Башкортостан Респ» является большим регионом, по его завершению опять смотрим файлы «process.log», «process.out» и «process.err», которые формируются/дополняются при каждом запуске загрузчика.

$>l

итого 100

-rw-rw-r-- 1 2080 фев 1 17:58 process.log

-rw-rw-r-- 1 51790 фев 1 17:58 process.out

-rw-rw-r-- 1 9228 фев 1 17:57 process.err

-rw-rw-r-- 1 8043 фев 1 17:04 stage\_31r.yaml

-rw-rw-r-- 1 1874 фев 1 17:03 stage\_34.csv

-rw-rw-r-- 1 6083 дек 16 16:16 stage\_41r.yaml

Файл process.log — отчёт о выполненном задании. Общее время выполнения 50 мин 20 сек.

Все активные этапы/секции были выполнены. Смотрим фрагмент YAML-файла

p\_skip\_adr\_area: False

p\_descr\_adr\_area: Агрегация адресных пространств

Соответствующее сообщение в process.log выделено красным цветом.

2023-02-01 17:08:21 ./

127.0.0.1 5433 unsi\_test\_02 postgres stage\_34.csv 2022-12-26

--------------------------------------------------------

[17:08:21] ... Start process, Обработка

[17:08:21] ... -- gar\_link.f\_server\_is

[17:08:21] ... -- stage\_31r (Предобработка данных)

[17:08:21] ... -- Создание рабочих индексов в схеме gar\_fias. False-удаление, True-

удаление,создание.

[17:12:21] ... -- Заполнение таблицы с активными предшественниками

[17:12:33] ... -- Обработка таблицы с активными предшественниками, переподчинение и деактивация

[17:39:45] ... -- Установка признака LOGGED/UNLOGGED у таблиц в схеме gar\_tmp

[17:39:46] ... -- Очистка данных во временной схеме.

[17:39:46] ... -- Загрузка данных из таблицы ADR\_AREA\_TYPE

[17:39:47] ... -- Загрузка данных из таблицы ADR\_STREET\_TYPE

[17:39:47] ... -- Загрузка данных из таблицы ADR\_HOUSE\_TYPE

[17:39:47] ... -- Загрузка регионального фрагмента из таблицы ADR\_AREA

[17:39:50] ... -- Загрузка регионального фрагмента из таблицы ADR\_STREET

[17:40:03] ... -- Загрузка регионального фрагмента из таблицы ADR\_HOUSE

[17:51:33] ... -- Установка последовательностей

[17:51:34] ... -- Агрегация данных

[17:51:34] ... -- Агрегация параметров

**[17:53:44] ... -- Агрегация адресных пространств**

[17:54:17] ... -- Агрегация домовладений

[17:57:54] ... -- Заполнение управляющей таблицы "gar\_tmp.xxx\_obj\_fias"

[17:58:41] ... Stop process

Далее рассмотрим файл «process.out». Он содержит результаты выполнения хранимых функций/процедур, выполняемых на каждом шаге процесса предобработки.

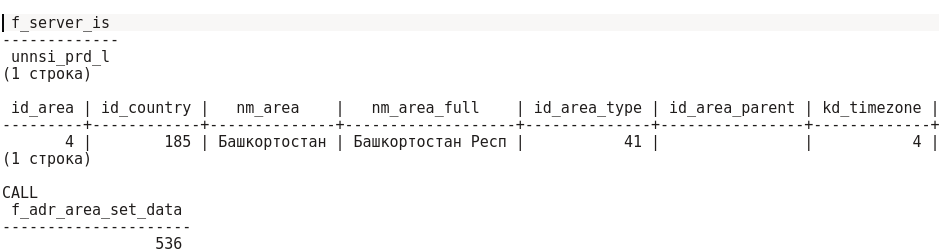


Рис 3.3.9 Первая часть файла «process.out».

Были выполнены этапы «stage\_3\_I» и «stage\_3\_9». Проиндексирована схема «gar\_fias» и выполнена обработка так называемых «активных предшественников» в таблице «gar\_fias.as\_addr\_obj». Активный предшественник (дубликат), это запись являющаяся двойником в таблице «gar\_fias.as\_addr\_obj». Такие записи различаются значениями «ID» - идентификатор записи и «CHANGE\_ID» - идентификатор транзакции изменившей данные. Обе записи активны и актуальны, но активный предшественник (дубликат) имеет значение «CHANGE\_ID» меньшее, чем у основной записи. Цель обработки заключается в деактивации и деактулизации предшественника и переподчинении в в основную запись его дочерних записей.

Далее смотрим результаты выполнения этапов «stage\_3\_0» и «stage\_3\_1».

CALL

CALL

CALL

CALL

CALL

CALL

CALL

CALL

f\_xxx\_obj\_seq\_crt

-------------------

400131799

10400131799

(2 строки)

Цепочка вызовов хранимых процедур, выполняющих загрузку сторонних данных в схему «gar\_tmp». В завершении этапов «stage\_3\_1» выполняется установка последовательности. У каждого адресного региона своё значение последовательности, генерирующей значения для ID вновь создаваемых записей. Актуально сейчас только первое значение «400131799», в следующих релизах второе значение «0400131799» будет убрано. При развёртывании кластера БД значение последовательность устанавливается так: <ТЕКУЩЕЕ ЗНАЧЕНИЕ ПОСЛЕДОВАТЕЛЬНОСТИ> = <НОМЕР РЕГИОНА> \* <МАСШТАБНЫЙ МНОЖИТЕЛЬ> + < НАЧАЛЬНАЯ ВЕЛИЧИНА>.

f\_set\_params\_value

--------------------

3921445

(1 строка)

f\_xxx\_adr\_area\_set\_data

-------------------------

46720

(1 строка)

f\_xxx\_adr\_house\_set\_data

--------------------------

1051045

(1 строка)

f\_xxx\_obj\_fias\_set\_data

-------------------------

9650

37070

1051045

(3 строки)

Далее смотрим результаты выполнения этапа «stage\_3\_3», адресные данные агрегируются в соответствии с отношением административной подчинённости, определённым в таблице «as\_adm\_hierarchy» и сравниваются с региональными адресными данными, загруженными в начале процесса из целевой БД, критерий сравнения UUID адресного объекта.

К моменту завершения процесса к дальнейшей обработке будут готовы 9650 адресных объектов (республики, города, населённые пункты), 37070 элементов дорожной структуры и 1051045 домов и строений. Отдельно следует остановиться на параметрах адресных объектов. Функция «f\_set\_params\_value ()» агрегирует данные из всех таблиц, хранящих параметры адресных объектов, таких как «as\_house\_params», «as\_addr\_obj\_params», и т.д в единую структуру «Ключ-Значение». К параметрам относим: почтовый индекс, код ОКАТО, кадастровый номер и т.п.

Назначение группы переключателей «mogrify\_3\_x»:

#

mogrify\_3\_I: False

mogrify\_3\_9: False

mogrify\_3\_0: False

mogrify\_3\_1: False

mogrify\_3\_2: False

mogrify\_3\_3: False

Эти переключатели, размещённые в заголовке YAML-файла, помогают ответить на вопрос «Какие хранимые функции и процедуры» будут выполнены в течении процесса. Если включить их, то в файл «process.err» будут выведены тексты вызов процедур и функций в той последовательности, в которой они будут реально выполняться. Никакого выполнения команд при этом не будет. Такой функционал является аналогом метода «mogrify()» из библиотеки «psycopg2».

Устанавливаем переключатели:

#

mogrify\_3\_I: True

mogrify\_3\_9: True

mogrify\_3\_0: True

mogrify\_3\_1: True

mogrify\_3\_2: True

mogrify\_3\_3: True

И запускаем процесс предобработки:

$>load\_mainGar.py 127.0.0.1 5433 unsi\_test\_02 postgres stage\_34.csv . '2022-12-26'

... Start process, Обработка

... -- gar\_link.f\_server\_is

... -- stage\_31r (Предобработка данных)

... -- Создание рабочих индексов в схеме gar\_fias. False-удаление, True-удаление,создание.

... -- Заполнение таблицы с активными предшественниками

... -- Обработка таблицы с активными предшественниками, переподчинение и деактивация

... -- Установка признака LOGGED/UNLOGGED у таблиц в схеме gar\_tmp

... -- Очистка данных во временной схеме.

... -- Загрузка данных из таблицы ADR\_AREA\_TYPE

... -- Загрузка данных из таблицы ADR\_STREET\_TYPE

... -- Загрузка данных из таблицы ADR\_HOUSE\_TYPE

... -- Загрузка регионального фрагмента из таблицы ADR\_AREA

... -- Загрузка регионального фрагмента из таблицы ADR\_STREET

... -- Загрузка регионального фрагмента из таблицы ADR\_HOUSE

... -- Установка последовательностей

... -- Агрегация данных

... -- Агрегация параметров

... -- Агрегация адресных пространств

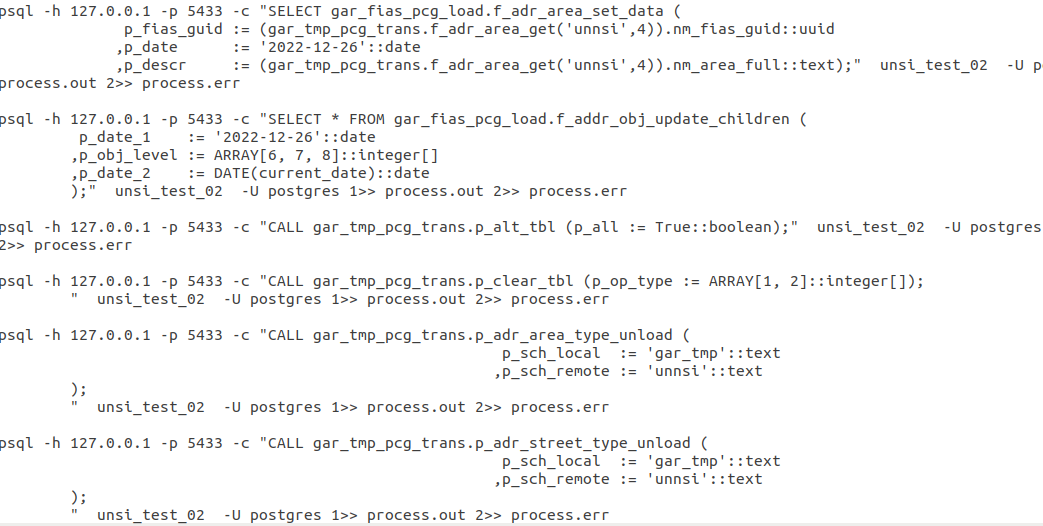
... -- Агрегация домовладений

... -- Заполнение управляющей таблицы "gar\_tmp.xxx\_obj\_fias"

... Stop process

$>

Внешне всё выглядит аналогично предыдущему запуску, но выполнилось практически мгновенно. Смотрим файл «process.err» (фрагмент):

 Рис 3.3.10 Фрагмент файла «process.err». Mogrify = True

В заключении следует отметить, что использование программ-обёрток над psql достаточно затратно с точки зрения на потребление ресурсов. На каждый процесс тратится один Shell-процесс «Python», на каждую команду один Shell-процесс «psql», на каждый вызов хранимки — отдельный connect к базе.

Далее переходим к описанию процесса обработки.

**3.3.3.2) Описание управляющего YAML-файла «stage\_41r.yaml».**

#

# 2022-03-11/2022-06-09

# VERSION\_STR = " Version 0.3.0 Build 2022-12-06"

#

control\_params:

stage\_4\_1: True

stage\_4\_2: True

stage\_a\_p: False

#

stage\_4\_3: True

stage\_4\_4: True

stage\_s\_p: False

#

stage\_4\_5: True

stage\_4\_6: True

stage\_h\_p: False

#

mogrify\_4\_1: False

mogrify\_4\_2: False

mogrify\_a\_p: False

#

mogrify\_4\_3: False

mogrify\_4\_4: False

mogrify\_s\_p: False

#

mogrify\_4\_5: False

mogrify\_4\_6: False

mogrify\_h\_p: False

global\_params:

g\_history\_sch: gar\_tmp

adr\_area:

#

# stage\_4\_1

#

op\_ins:

p\_descr: Адресные регионы, дополнение

params:

p\_schema\_data: gar\_tmp # Обновляемая схема с данными ОТДАЛЁННЫЙ/ЛОКАЛЬНЫЙ СЕРВЕР

p\_schema\_etl: gar\_tmp # Схема эталон, обычно локальный сервер, копия p\_schema\_data

p\_sw\_hist: TRUE # Создаётся историческая запись.

p\_check\_query: SELECT \* FROM gar\_tmp.adr\_area\_hist WHERE (id\_region = 0) AND . . .

#

# stage\_4\_2

#

op\_upd:

p\_descr: Адресные регионы, обновление

params:

p\_schema\_data: gar\_tmp # Обновляемая схема с данными ОТДАЛЁННЫЙ/ЛОКАЛЬНЫЙ СЕРВЕР

p\_schema\_etl: gar\_tmp # Схема эталон, обычно локальный сервер, копия p\_schema\_data

p\_sw\_hist: TRUE # Создаётся историческая запись.

p\_check\_query: SELECT \* FROM gar\_tmp.adr\_area\_hist WHERE (id\_region = 0) AND . . .

#

# stage\_a\_p

#

post\_proc:

p\_descr: Постобработка списка адресных регионов

p\_query\_1: ../../A\_FIAS\_LOADER/GAR\_TMP\_PCG\_TRANS/DO/adr\_area\_post\_proc.sql # Имя сценария

#

adr\_street:

#

# stage\_4\_3

#

op\_ins:

p\_descr: Элемент улично-дорожной сети, дополнение

params:

p\_schema\_data: gar\_tmp # Обновляемая схема с данными ОТДАЛЁННЫЙ СЕРВЕР

p\_schema\_etl: gar\_tmp # Схема эталон, обычно локальный сервер, копия p\_schema\_data

p\_sw\_hist: TRUE # Создаётся историческая запись

p\_check\_query: SELECT \* FROM gar\_tmp.adr\_street\_hist WHERE (id\_region = 0) AND . . .

#

# stage\_4\_4

#

op\_upd:

p\_descr: Элемент улично-дорожной сети, обновление

params:

p\_schema\_data: gar\_tmp # Обновляемая схема с данными ОТДАЛЁННЫЙ СЕРВЕР

p\_schema\_etl: gar\_tmp # Схема эталон, обычно локальный сервер, копия p\_schema\_data

p\_schema\_hist: gar\_tmp # Схема для хранения исторических данных

p\_sw\_hist: TRUE # Создаётся историческая запись

p\_sw\_twin: FALSE # Обязательное выявление дубликатов

p\_check\_query: SELECT \* FROM gar\_tmp.adr\_street\_hist WHERE (id\_region = 0) AND . . .

#

# stage\_s\_p

#

post\_proc:

p\_descr: Постобработка списка улиц

p\_query\_1: ../../A\_FIAS\_LOADER/GAR\_TMP\_PCG\_TRANS/DO/adr\_street\_post\_proc.sql # Имя сценария

#

adr\_house:

#

# stage\_4\_5

#

op\_ins:

p\_descr: Здания (сооружения), дополнение

params:

p\_schema\_data: gar\_tmp # Обновляемая схема с данными ОТДАЛЁННЫЙ СЕРВЕР

p\_schema\_etl: gar\_tmp # Схема эталон, обычно локальный сервер, копия p\_schema\_data

p\_sw: FALSE # Включить дополнение/обновление adr\_objects

p\_sw\_twin: FALSE # Включается поиск двойников

p\_check\_query: SELECT \* FROM gar\_tmp.adr\_house\_hist WHERE (id\_region = 0) AND . . .

#

# stage\_4\_6

#

op\_upd:

p\_descr: Здания (сооружения), обновление

params:

p\_schema\_data: gar\_tmp # Обновляемая схема с данными ОТДАЛЁННЫЙ СЕРВЕР

p\_schema\_etl: gar\_tmp # Схема эталон, обычно локальный сервер, копия p\_schema\_data

p\_sw\_hist: TRUE # Создаётся историческая запись.

p\_sw\_twin: FALSE # Обязательное выявление дубликатов

p\_del: FALSE # Убираю дубли при обработки EXCEPTION

p\_sw: FALSE # Включить дополнение/обновление adr\_objects

p\_check\_query: SELECT \* FROM gar\_tmp.adr\_house\_hist WHERE (id\_region = 0) AND . . .

#

# stage\_h\_p

#

post\_proc:

p\_descr: Постобработка списка домов

p\_query\_1: ../../A\_FIAS\_LOADER/GAR\_TMP\_PCG\_TRANS/DO/adr\_house\_post\_proc.sql # Имя сценария

#

Процесс, запускаемый большой командой «4;stage\_41r.yaml;» управляется набором параметров, сгруппированных в 5 главных секций:

| **Номер секции** | **Наименование** | **Объединяет** | **Описание** |
| --- | --- | --- | --- |
| 1 | control\_params: |  | Управляет выполнением секций «stage\_4\_x». |
| 2 | global\_params: |  | Один глобальный параметр «g\_history\_sch» имя схемы для исторических данных. |
| 3 | adr\_area: | op\_ins: **stage\_4\_1**  op\_upd: **stage\_4\_2**  post\_proc: **stage\_a\_p** | Дополнение и Обновление таблицы «adr\_area». |
| 4 | adr\_street: | op\_ins: **stage\_4\_3**  op\_upd: **stage\_4\_4**  post\_proc: **stage\_s\_p** | Дополнение и Обновление таблицы «adr\_street». |
| 5 | adr\_house: | op\_ins: **stage\_4\_5**  op\_upd: **stage\_4\_6**  post\_proc: **stage\_h\_p** | Дополнение и Обновление таблицы «adr\_house». |

Рис 3.311 Описание основных частей управляющего файла «stage\_4.yaml».

Управление секциями гораздо проще, чем на предыдущем этапе, если в «control\_params:» у соответствующей секции стоит значение «False», она не будет выполняться ни при каких обстоятельствах. В частности выключены все секции, связанные с частичной постобработкой, это «stage\_a\_p», «stage\_s\_p», и «stage\_h\_p». Параметров «Skip» нет. Далее остановимся на программах, обеспечивающих настройку и запуск процесса предобработки:

| **#пп** | **Наименование** | **Описание** |
| --- | --- | --- |
| 1 | load\_mainStage4.py | «Большая команда 4». Запуск процесса предобработ-ки. Управляется «stage\_4.yaml». |
| 2 | GarProcess/stage\_4\_yaml.py | Разбор YAML-файла. Параметры из него, становятся свойствами соответствующих методов. |
| 3 | GarProcess/stage\_4\_proc.py | Patterns вызовов хранимых функций и процедур, используемых в «load\_mainStage4.py». |

Рис 3.3.12 Основные программы, используются на этапе обработки.

Запускаем сценарий, предварительно выключив команду «3;stage\_31r.yaml;».

X;;;Start process, Обработка;

#

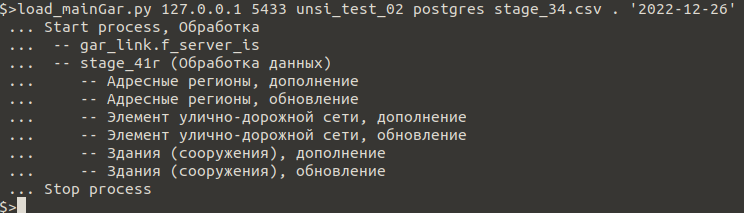
0;SELECT gar\_link.f\_server\_is();; -- gar\_link.f\_server\_is;

### 3;stage\_31r.yaml;; -- stage\_31r (Предобработка данных);

4;stage\_41r.yaml;; -- stage\_41r (Обработка данных);

#

X;;;Stop process;

 Рис 3.3.13 Выполнение процесса обработки адресных данных

Смотрим на то, что получилось в рабочем каталоге.

$>pwd

/home/rootadmin/example/3

$>l

итого 52

drwxrwxr-x 2 4096 фев 2 17:28 .

-rw-rw-r-- 1 886 фев 2 17:13 process.log

-rw-rw-r-- 1 13755 фев 2 17:13 process.out

-rw-rw-r-- 1 0 фев 2 16:58 process.err

-rw-rw-r-- 1 1874 фев 2 16:55 stage\_34.csv

-rw-rw-r-- 1 6083 фев 2 16:54 stage\_41r.yaml

-rw-rw-r-- 1 8037 фев 2 16:54 stage\_31r.yaml

drwxrwxr-x 6 4096 фев 2 16:53 ..

Файл «process.log» содержит отчёт о выполненном задании

**2023-02-02 16:58:00 ./**

**127.0.0.1 5433 unsi\_test\_02 postgres stage\_34.csv 2022-12-26**

--------------------------------------------------------

[16:58:00] ... Start process, Обработка

[16:58:00] ... -- gar\_link.f\_server\_is

[16:58:00] ... -- stage\_41r (Обработка данных)

[16:58:00] ... -- Адресные регионы, дополнение

[16:58:01] ... -- Адресные регионы, обновление

[16:58:28] ... -- Элемент улично-дорожной сети, дополнение

[16:58:28] ... -- Элемент улично-дорожной сети, обновление

[16:59:20] ... -- Здания (сооружения), дополнение

[16:59:21] ... -- Здания (сооружения), обновление

[17:13:42] ... Stop process

Результаты выполнения обновления локальной региональной адресной базы содержатся в файле «process.out». Ниже приведены фрагменты из «process.out».

f\_server\_is

------------- **<<-- Foreign Server**

unnsi\_prd\_l

(1 строка)

total\_row | ins\_row | upd\_row **<<-- Таблица "adr\_area"**

-----------+---------+---------

0 | 0 | 0

(1 строка)

id\_area | id\_country | nm\_area

---------+------------+----------

(0 строк)

total\_row | upd\_row

-----------+---------

9645 | 4

(1 строка)

total\_row | ins\_row | upd\_row **<<-- Таблица "adr\_street"**

-----------+---------+---------

9 | 9 | 0

(1 строка)

id\_street | id\_area | nm\_street

-----------+---------+-----------

(0 строк)

total\_row | upd\_row

-----------+---------

37056 | 9

(1 строка)

total\_row | ins\_row | upd\_row **<<-- Таблица "adr\_house"**

-----------+---------+---------

1 | 1 | 0

(1 строка)

id\_house | id\_area | id\_street

----------+---------+-----------

(0 строк)

total\_row | upd\_row

-----------+---------

1051044 | 3

(1 строка)

**3.3.4 Сборка программных пакетов.**

В заключении рассмотрим сборку программных пакетов. В настоящее время собираются пакеты из текстов хранимых процедур и функций, написанных на PLpg/SQL либо на SQL. Сборка Python-пакетов планируется в среднесрочной переспективе. PLpg/SQL-пакеты, в дальнейшем изложении просто пакеты, расположены в git-репозитории проекта. Дальнейшее изложение построим на примере построения пакета для схемы «gar\_link», пакет получается компактным и легко обозримым.

drwxrwxr-x 5 4096 янв 30 11:26 .

-rw-rw-r-- 1 2745 янв 30 11:26 stage\_25\_link\_build.csv

-rw-rw-r-- 1 293 янв 30 11:26 version.sql

-rw-rw-r-- 1 84881 янв 30 11:26 gar\_link\_pcg.sql

-rw-rw-r-- 1 1010 янв 30 11:26 gar\_link\_pcg.log

-rw-rw-r-- 1 9926 янв 30 11:26 gar\_link\_pcg.err

-rw-rw-r-- 1 1010 янв 30 11:26 gar\_link\_pcg.log.bak

-rw-rw-r-- 1 10111 янв 30 11:26 gar\_link\_pcg.err.bak

-rw-rw-r-- 1 84881 янв 30 11:26 gar\_link\_pcg.sql.bak

drwxrwxr-x 2 4096 янв 30 11:26 PROCEDURE

drwxrwxr-x 2 4096 янв 30 11:26 FUNCTION

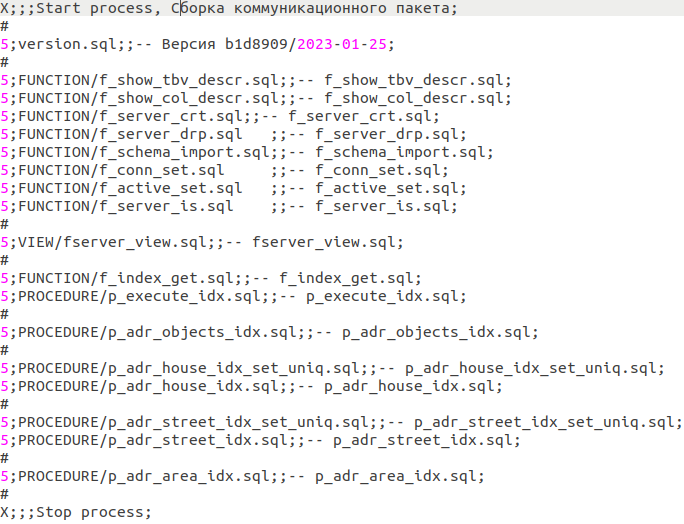
drwxrwxr-x 2 4096 сен 1 20:32 VIEW

-rw-rw-r-- 1 6877 янв 30 11:26 CreateTableGarLink\_1.sql

-rw-rw-r-- 1 3955 янв 30 11:26 CreateTableGarLink.sql

-rw-rw-r-- 1 113 сен 1 20:32 README.txt

Основой является CSV-сценарий, его текст приводится ниже.

 Рис 3.4.1 Сценарий для сборки программного пакета «gar\_link\_pcg».

Пакет собирается с использованием утилиты «load\_mainBuild.sh», пример её вызова,

«unsi\_old2» - сборочная база.

$>

$>load\_mainBuild.sh 127.0.0.1 5433 unsi\_old2 postgres stage\_25\_link\_build.csv . gar\_link\_pcg

Файлы формируемые в процессе сборки:

gar\_link\_pcg.sql - Непосредственно сам пакет

gar\_link\_pcg.log - Протокол сборки

gar\_link\_pcg.err - Зависимости пакета.

Протокол сборки является стандартным для всех программ из семейства «load\_main». Ниже приведён его текст:

**2023-01-25 18:13:23 ./**

**127.0.0.1 5433 unsi\_old2 postgres stage\_25\_link\_build.cs**v

--------------------------------------------------------

[18:13:23] ... Start process, Сборка коммуникационного пакета

[18:13:23] ... -- Версия b1d8909/2023-01-25

[18:13:23] ... -- f\_show\_tbv\_descr.sql

[18:13:23] ... -- f\_show\_col\_descr.sql

[18:13:23] ... -- f\_server\_crt.sql

[18:13:23] ... -- f\_server\_drp.sql

[18:13:23] ... -- f\_schema\_import.sql

[18:13:23] ... -- f\_conn\_set.sql

[18:13:24] ... -- f\_active\_set.sql

[18:13:24] ... -- f\_server\_is.sql

[18:13:24] ... -- fserver\_view.sql

[18:13:24] ... -- f\_index\_get.sql

[18:13:24] ... -- p\_execute\_idx.sql

[18:13:24] ... -- p\_adr\_objects\_idx.sql

[18:13:24] ... -- p\_adr\_house\_idx\_set\_uniq.sql

[18:13:24] ... -- p\_adr\_house\_idx.sql

[18:13:24] ... -- p\_adr\_street\_idx\_set\_uniq.sql

[18:13:25] ... -- p\_adr\_street\_idx.sql

[18:13:25] ... -- p\_adr\_area\_idx.sql

[18:13:25] ... Stop process

--------------------------------------------------------

При сборке пакетов всегда используются функции расширения «plpgsql\_check», которое установлено в сборочной базе. Результаты работы функций «plpgsql\_check» попадают в файл «gar\_link\_pcg.err». Ниже приведён фрагмент этого файла.

./FUNCTION/f\_show\_col\_descr.sql:136: ЗАМЕЧАНИЕ: RELATION | 2612 | pg\_catalog | pg\_language

./FUNCTION/f\_show\_col\_descr.sql:136: ЗАМЕЧАНИЕ: RELATION | 2995 | pg\_catalog | pg\_largeobject\_metadata

./FUNCTION/f\_show\_col\_descr.sql:136: ЗАМЕЧАНИЕ: RELATION | 2615 | pg\_catalog | pg\_namespace

./FUNCTION/f\_show\_col\_descr.sql:136: ЗАМЕЧАНИЕ: RELATION | 1255 | pg\_catalog | pg\_proc

./FUNCTION/f\_show\_col\_descr.sql:136: ЗАМЕЧАНИЕ: RELATION | 6104 | pg\_catalog | pg\_publication

./FUNCTION/f\_show\_col\_descr.sql:136: ЗАМЕЧАНИЕ: RELATION | 3596 | pg\_catalog | pg\_seclabel

./FUNCTION/f\_show\_col\_descr.sql:136: ЗАМЕЧАНИЕ: RELATION | 12161 | pg\_catalog | pg\_seclabels

./FUNCTION/f\_show\_col\_descr.sql:136: ЗАМЕЧАНИЕ: RELATION | 2224 | pg\_catalog | pg\_sequence

./FUNCTION/f\_show\_col\_descr.sql:136: ЗАМЕЧАНИЕ: RELATION | 3592 | pg\_catalog | pg\_shseclabel

./FUNCTION/f\_show\_col\_descr.sql:136: ЗАМЕЧАНИЕ: RELATION | 6100 | pg\_catalog | pg\_subscription

./FUNCTION/f\_show\_col\_descr.sql:136: ЗАМЕЧАНИЕ: RELATION | 1213 | pg\_catalog | pg\_tablespace

./FUNCTION/f\_show\_col\_descr.sql:136: ЗАМЕЧАНИЕ: RELATION | 1247 | pg\_catalog | pg\_type

./FUNCTION/f\_server\_crt.sql:67: ЗАМЕЧАНИЕ: RELATION | 13385 | information\_schema | foreign\_servers

./FUNCTION/f\_server\_crt.sql:67: ЗАМЕЧАНИЕ: RELATION | 13378 | information\_schema | \_pg\_foreign\_servers

./FUNCTION/f\_server\_crt.sql:67: ЗАМЕЧАНИЕ: RELATION | 1260 | pg\_catalog | pg\_authid

./FUNCTION/f\_server\_crt.sql:67: ЗАМЕЧАНИЕ: RELATION | 2328 | pg\_catalog | pg\_foreign\_data\_wrapper

./FUNCTION/f\_server\_crt.sql:67: ЗАМЕЧАНИЕ: RELATION | 1417 | pg\_catalog | pg\_foreign\_server

./FUNCTION/f\_server\_drp.sql:30: ЗАМЕЧАНИЕ: RELATION | 74235 | gar\_link | foreign\_servers

./FUNCTION/f\_schema\_import.sql:138: ЗАМЕЧАНИЕ: RELATION | 74235 | gar\_link | foreign\_servers

./FUNCTION/f\_schema\_import.sql:138: ЗАМЕЧАНИЕ: RELATION | 13382 | information\_schema | foreign\_server\_options

./FUNCTION/f\_schema\_import.sql:138: ЗАМЕЧАНИЕ: RELATION | 13385 | information\_schema | foreign\_servers

./FUNCTION/f\_schema\_import.sql:138: ЗАМЕЧАНИЕ: RELATION | 13378 | information\_schema | \_pg\_foreign\_servers

./FUNCTION/f\_schema\_import.sql:138: ЗАМЕЧАНИЕ: RELATION | 1260 | pg\_catalog | pg\_authid

./FUNCTION/f\_schema\_import.sql:138: ЗАМЕЧАНИЕ: RELATION | 2328 | pg\_catalog | pg\_foreign\_data\_wrapper

./FUNCTION/f\_schema\_import.sql:138: ЗАМЕЧАНИЕ: RELATION | 1417 | pg\_catalog | pg\_foreign\_server

./FUNCTION/f\_conn\_set.sql:1: ЗАМЕЧАНИЕ: функция gar\_link.f\_conn\_set(text,text) не существует, пропускается

./FUNCTION/f\_conn\_set.sql:53: ЗАМЕЧАНИЕ: FUNCTION | 73684 | gar\_link | dblink\_connect | (text,text)

./FUNCTION/f\_conn\_set.sql:53: ЗАМЕЧАНИЕ: FUNCTION | 73721 | gar\_link | dblink\_get\_connections | ()

./FUNCTION/f\_conn\_set.sql:53: ЗАМЕЧАНИЕ: RELATION | 74235 | gar\_link | foreign\_servers

./FUNCTION/f\_index\_get.sql:63: ЗАМЕЧАНИЕ: RELATION | 126481 | gar\_link | adr\_indecies

./PROCEDURE/p\_execute\_idx.sql:28: ЗАМЕЧАНИЕ: warning extra:00000:6:DECLARE:never read variable "\_mess"

./PROCEDURE/p\_execute\_idx.sql:28: ЗАМЕЧАНИЕ: FUNCTION | 73701 | gar\_link | dblink | (text,text)

./PROCEDURE/p\_adr\_objects\_idx.sql:157: ЗАМЕЧАНИЕ: FUNCTION | 73701 | gar\_link | dblink | (text,text)

Строятся зависимости и выполняется проверка кода.

**3.4 Выполнение пакета команд группе хостов. Коммуникационное параметры и набор операций.**

Выполнение обработки на группе хостов по сути своей не отличается от операций, выполняемых на одном хосте. В случае работы на группе хостов, коммуникационные параметры задаются специальными адресными YAML-файлами, образуя при этом адресное пространство. CSV-сценарий выполняется последовательно на каждом адресного пространства. Имеются две разновидности адресных YAM-файлов. Первый и самый распространённый этот «host\_total\_xxx»

Ниже на рис. 3.5.1. приведён фрагмент файла «host\_total\_xxx». Файл представляет собой массив «hosts» описывающий адреса и логины кластера БД., состоит из четырёх параметров, из которых последний является комплексным. «name» - имя узла, «descr» - описание, «conninfo» - строка соединения, включает имя БД, последний параметр «params» является комплексным, в настоящее время состоит только из одной части «id\_region» - идентификатор адресного региона (в терминах ЕС-НСИ).

С коммуникационным YAML-файлом типа «total» работает только программа «load\_mainGar\_2.py». Повторяем выполненный ранее пример, накатывание обновлённого пакета «gar\_fias\_pcg\_load», но на этот раз на всех хостах, включённых в коммуникационный YAML-файл «../hosts\_total\_xx\_0.yaml»

$>pwd

/home/rootadmin/example/1

$>l

итого 12

drwxrwxr-x 2 4096 фев 3 18:38 .

drwxrwxr-x 7 4096 фев 2 18:59 ..

-rw-rw-r-- 1 1811 янв 31 14:14 s**tage\_x.csv**

$>l ../\*yaml

-rw-rw-r-- 1 738 янв 27 15:25 ../hosts\_parse\_xx\_01.yaml

-rw-rw-r-- 1 7193 янв 26 17:22 ../hosts\_init\_xx\_0.yaml

-rw-rw-r-- 1 7373 янв 26 17:22 ../hosts\_parse\_xx\_0.yaml

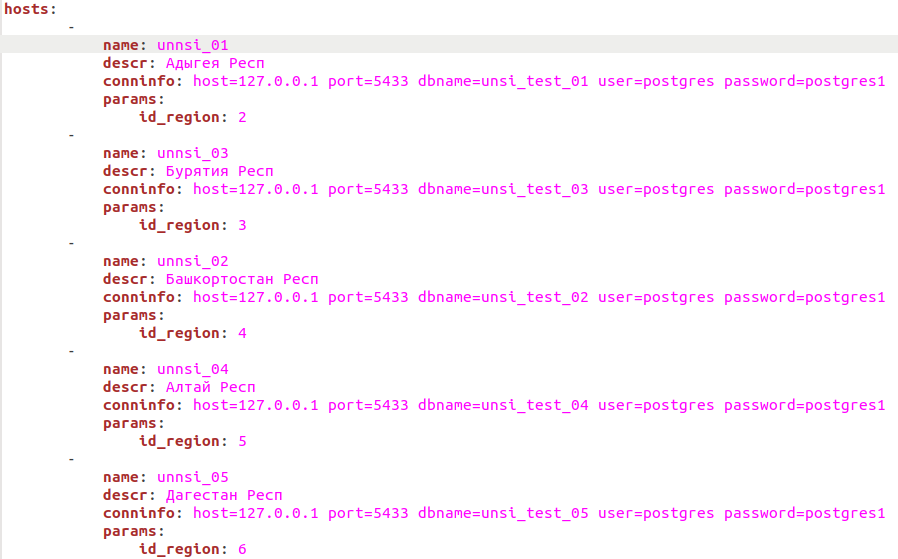
**-rw-rw-r-- 1 4973 янв 26 17:22 ../hosts\_total\_xx\_0.yaml**

$>load\_mainGar\_2.py

Version 0.5.5 Build 2023-01-13

Usage: /usr/local/bin/load\_mainGar\_2.py <Yaml\_file> <Batch\_file> <Path> <Version>

**$>**

****

**/** Рис 3.5.1 Коммуникационный YAML-файл «hosts\_total\_xx\_0.yaml» (фрагмент).

Программе необходимо передать имя YAML-файла, CSV-сценарий (Batch\_file), указать относительный путь к точке, в которой находится ссылка на git-репозиторий. И дату-версию.

$>load\_mainGar\_2.py ../hosts\_total\_xx\_0.yaml stage\_x.csv .. '2022-12-26'

... Адыгея Респ, id\_region: 2

... Start process

... -- Обновление gar\_fias\_pcg\_load

... Stop process

... Бурятия Респ, id\_region: 3

... Start process

... -- Обновление gar\_fias\_pcg\_load

... Stop process

... Башкортостан Респ, id\_region: 4

... Start process

... -- Обновление gar\_fias\_pcg\_load

... Stop process

... Алтай Респ, id\_region: 5

... Start process

... -- Обновление gar\_fias\_pcg\_load

... Stop process

... Дагестан Респ, id\_region: 6

... Start process

... -- Обновление gar\_fias\_pcg\_load

... Stop process

... Ингушетия Респ, id\_region: 7

... Start process

... -- Обновление gar\_fias\_pcg\_load

На каждой итерации по коммуникационному YAML-файлу (массив «hosts») выполняется вызов программы «load\_mainGar.py», в чём можно убедиться, взглянув на отчёт «process.log».

**2023-02-03 19:09:36 ../**

**127.0.0.1 5433 unsi\_test\_01 postgres stage\_x.csv 2022-12-26 ←--- Первый запуск load\_maimnGar.py**

--------------------------------------------------------

[19:09:36] ... Start process

[19:09:36] ... -- Обновление gar\_fias\_pcg\_load

[19:09:39] ... Stop process

--------------------------------------------------------

**2023-02-03 19:09:39 ../**

**127.0.0.1 5433 unsi\_test\_03 postgres stage\_x.csv 2022-12-26**  **←--- Второй запуск load\_maimnGar.py**

--------------------------------------------------------

[19:09:39] ... Start process

[19:09:39] ... -- Обновление gar\_fias\_pcg\_load

[19:09:41] ... Stop process

--------------------------------------------------------

**2023-02-03 19:09:41 ../**

**127.0.0.1 5433 unsi\_test\_02 postgres stage\_x.csv 2022-12-26 ←--- . . . . . . . .**

--------------------------------------------------------

[19:09:41] ... Start process

[19:09:41] ... -- Обновление gar\_fias\_pcg\_load

[19:09:43] ... Stop process

--------------------------------------------------------

**2023-02-03 19:09:43 ../**

**127.0.0.1 5433 unsi\_test\_04 postgres stage\_x.csv 2022-12-26**

--------------------------------------------------------

[19:09:43] ... Start process

[19:09:43] ... -- Обновление gar\_fias\_pcg\_load

[19:09:45] ... Stop process

--------------------------------------------------------

**2023-02-03 19:09:45 ../**

**127.0.0.1 5433 unsi\_test\_05 postgres stage\_x.csv 2022-12-26**

--------------------------------------------------------

В приведённом примере параметр «region\_id» из адресного (коммуникационного) YAML-файла был проигнорирован, он будет использован только при обработке сценариев, содержащих большие команды типа «3» и «4». Прежде, чем перейти к рассмотрению коммуникационных файлов двух других типов, следует отметить, что отдельные базы из адресного кластера можно разместить на различных хостах, в этом случае параметр «conninfo» каждой базы будет различаться.

Существуют ещё два типа коммуникационных YAML-файлов, это файлы типов «INIT» и «PARSE». Файлы этого типа очень близки по структуре, различаются только значениями своих параметров «name» и «params/exec». Если с коммуникационными файлами типа «TOTAL», работает только программа «load\_mainGar\_2.py», то с файлами типов «INIT» и «PARSE» работает соответственно «load\_mainGar\_4.py». На рисунках приведённых ниже приведенф фрагменты файлов типов «INIT» и «PARSE», далее — пример выполнения программы «load\_mainGar\_4.py».



Рис 3.5.2 Коммуникационный YAML-файл «hosts\_parse\_xx\_0.yaml» (фрагмент).

Запуск процесса parsing адресных данных для региона «02 Адыгея».

$>load\_mainGar\_4.py hosts\_parse\_xx\_01.yaml '2022-12-26'

... Адыгея Респ, id\_region: 2, Execute: /home/rootadmin/example/Y\_BUILD/01/stage\_gar\_c\_01.csv

... Start proces

... Start process

... -- Version

... -- Очистка от данных

... -- Set UNLOGGED

... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/AS\_ADDHOUSE\_TYPES\_20221226\_2626f853-8a92-4d83-ad73-4539ce780eb0.XML

gar\_fias.as\_add\_house\_type: 4

... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/AS\_ADDR\_OBJ\_TYPES\_20221226\_04f2dc05-9918-4f6e-9c4e-2fa06c6cce61.XML

gar\_fias.as\_addr\_obj\_type: 421

... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/AS\_APARTMENT\_TYPES\_20221226\_8dd9eb09-e026-47fe-bc42-2398f92cad5c.XML

gar\_fias.as\_apartment\_type: 13

... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/AS\_HOUSE\_TYPES\_20221226\_c399871c-42a6-49d3-9cae-09766d4d965b.XML

gar\_fias.as\_house\_type: 14

... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/AS\_NORMATIVE\_DOCS\_KINDS\_20221226\_5eac7f6d-be77-40d6-825b-6daac06070a7.XML

gar\_fias.as\_norm\_docs\_kinds: 4

... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/AS\_NORMATIVE\_DOCS\_TYPES\_20221226\_99c46464-14cb-456e-98c1-a32cd4f2833d.XML

gar\_fias.as\_norm\_docs\_types: 25

... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/AS\_OBJECT\_LEVELS\_20221226\_6e97d7bb-9815-410e-81dc-37d79b5ab350.XML

gar\_fias.as\_object\_level: 17

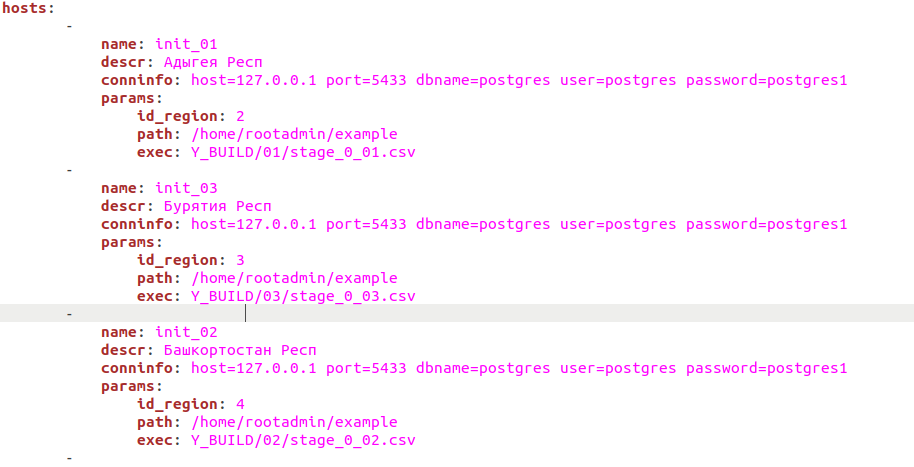


Рис 3.5.3 Коммуникационный YAML-файл «hosts\_init\_xx\_0.yaml» (фрагмент).

Ранее предполагалось создать ещё один файл: процессинговый YAML-файл, который имел бы секцию глобальных параметров. От этой идеи пришлось отказаться в пользу больших команд типа «3», «4» и «6».

**4. Управление функционирование.**

Раздел посвящён оперированию параметрами управляющих yaml-файлов, csv — сценариями, типовые случаи применения и их правильная настройка.

**4.1) Параметризация CSV-сценариев.**

В существующей версии пакета «load\_main» CSV-сценарии не принимают внешних параметров, предполагалось создание механизма передачи значений в соответствующие места CSV-сценария, отмеченные символами $1 .. $n. Но этот механизм не был реализован. Сейчас единственным методом управления CSV-сценарием является комментирование строк при помощи символа «#», либо исправление их в любом доступном текстовом редакторе.

Пример комментирования CSV- сценария:

0;CALL gar\_link.p\_adr\_house\_idx (gar\_link.f\_conn\_set(11), 'unnsi', true, false);; -- "adr\_house" Убираю эксплуатационные;

**##0;CALL gar\_link.p\_adr\_objects\_idx (gar\_link.f\_conn\_set(11), 'unnsi', true, false);; -- "adr\_objects" Убираю эксплуатационные;**

#

0;CALL gar\_link.p\_adr\_area\_idx (gar\_link.f\_conn\_set(11), 'unnsi', false, true);; -- "adr\_area\_id" Создание процессинговых;

0;CALL gar\_link.p\_adr\_street\_idx (gar\_link.f\_conn\_set(11), 'unnsi', false, true);; -- "adr\_street" Создание процессинговых;

0;CALL gar\_link.p\_adr\_house\_idx (gar\_link.f\_conn\_set(11), 'unnsi', false, true);; -- "adr\_house" Создание процессинговых;

**## 0;CALL gar\_link.p\_adr\_objects\_idx (gar\_link.f\_conn\_set(11), 'unnsi', false, true, false);; -- "adr\_objects" Создание процессинговых;**

Выше приведён фрагмент старого сценария «stage\_3\_50.csv», который использовался до появления больших команд типа «3». Строки управляющие процессами реиндексирования сторонней таблицы «unnsi.adr\_objects» выключены двойным символом «##». Выделено **синим.**

**4.2) Параметризация больших команд (3, 4, 6).**

Управляющие YAML-файлы были описаны в предыдущих разделах. Сейчас опишем этапы процесса, запускаемого большой командой, после этого оценим влияние параметров YAML-файла на выполнение процесса.

**4.2.1) Большая команда типа «3».** Описание этапов процесса Предобработки.

| **Номер секции** | **Наименование** | **Объединяет** | **Описание** |
| --- | --- | --- | --- |
| 1 | control\_params: |  | Управляет выполнением секций «stage\_3\_x». |
| 2 | global\_params: |  | Глобальные параметры, общие для всех секций. |
| 3 | stage\_3\_I | gar\_fias\_crt\_idx: | Переиндексация схемы «gar\_fias». |
| 4 | stage\_3\_9 | gar\_fias\_set\_gap:  gar\_fias\_update\_children: | Заполнение и обработка таблицы с активными предшественниками. |
| 5 | stage\_3\_0 | **gar\_tmp\_set\_logged:**  gar\_tmp\_clear\_tbl:  gar\_tmp\_switch\_indexies: | Установки в таблицах схемы «gar\_tmp», очистка их, управление индексным покрытием в адресной схеме отдалённой базы (при необходимости). |
| 6 | stage\_3\_1 | **unload\_data\_adr\_area\_type:**  **unload\_data\_adr\_street\_type:**  **unload\_data\_adr\_house\_type:**  **unload\_data\_adr\_area:**  **unload\_data\_adr\_street:**  **unload\_data\_adr\_house:**  seq\_settings: | Загрузка справочников и региональных фрагментов из отдаленной схемы. |
| **7** | **stage\_3\_2** | **DEPRICATED** | **Импорт и обновление справочников из схемы GAR\_FIAS. НЕ ИСПОЛЬЗОВАТЬ** |
| 8 | stage\_3\_3 | agg\_params:  agg\_adr\_area:  agg\_adr\_house:  obj\_fias: | Агрегация данных и параметров, подготовка к обновлению адресной базы. |

Рис 4.1 Описание основных частей управляющего файла «stage\_3.yaml».

* Отображение названия и идентификатора обрабатываемого региона. Идентификатор региона передаётся в процесс либо из коммуникационного YAM-файла, либо из управляющего YAML-файла.
* Реиндексация схемы GAR-FIAS, необходимо, потому что загрузка распарсенных данных, выполняется в таблицы без индексного покрытия.
* Выполнения поиска и обработки активных предшественников. Активный предшественник (дубликат), это запись являющаяся двойником в таблице «gar\_fias.as\_addr\_obj». Такие записи различаются значениями «ID» - идентификатор записи и «CHANGE\_ID» - идентификатор транзакции изменившей данные. Обе записи активны и актуальны, но активный предшественник (дубликат) имеет значение «CHANGE\_ID» меньшее, чем у основной записи. Цель обработки заключается в деактивации и деактулизации предшественника и переподчинении в в основную запись его дочерних записей.
* **Установка признака «Logged» в таблицах в схеме «gar\_tmp». Устаревшая опция, Ulogged таблицы легко теряют данные.**
* В схеме «gar\_tmp» будут очищены от данных: временные таблицы: «gar\_tmp.xxx\_adr\_area», «gar\_tmp.xxx\_adr\_house», «gar\_tmp.xxx\_obj\_fias», «gar\_tmp.xxx\_type\_param\_value» и вспомогательные таблицы (содержат иденификаторы обновлённых/дополненных записей): «gar\_tmp.adr\_area\_aux», «gar\_tmp.adr\_house\_aux», «gar\_tmp.adr\_street\_aux».
* Смену индексного покрытия в отдалённых таблицах выполняться не будет. Сейчас это устаревшая опция. Данные загружаются в соответствующую региональную секцию и обрабатываются в ней.
* **Загрузка данных из целевой, сторонней базы. Загружаются справочники типов, соответствующие региональные фрагменты из таблиц «adr\_area», «adr\_street », «adr\_house». Этот функционал абсолютно необходим в том случае, если адресные данные поступают в целевую базу несколькими путями, загрузка их из «Гар-Фиас» является одним из них.**
* **Актуализация справочников. Не выполняется, выключено всегда.**
* Агрегация данных. Отношения подчинённости приводятся в виду, принятому в ЕС НСИ (административно-территориальное деление).

Что нужно сделать, для, того, что-бы на этапе предобработки не выполнялись пункты «gar\_tmp\_set\_logged:» и весь «stage\_3\_1:» за исключением «gar\_tmp\_set\_logged:». Выклю-чение «stage\_3\_1:» пункта даст некоторый выигрыш в производительности. Смотрим обновлённый файл «stage\_31r.yaml», включенные параметры «skip» выделены **синим**:

#

control\_params:

stage\_3\_I: True

stage\_3\_9: True

stage\_3\_0: True

stage\_3\_1: True

stage\_3\_2: False

stage\_3\_3: True

#

mogrify\_3\_I: False

mogrify\_3\_9: False

mogrify\_3\_0: False

mogrify\_3\_1: False

mogrify\_3\_2: False

mogrify\_3\_3: False

global\_params:

g\_region\_id: 04 # Башкирия

g\_fserver\_nmb: 1 # На 5435 unnsi\_prd

#

g\_adr\_area\_sch: unnsi

g\_adr\_street\_sch: unnsi

g\_adr\_house\_sch: unnsi

#

g\_adr\_area\_sch\_l: gar\_tmp

g\_adr\_street\_sch\_l: gar\_tmp

g\_adr\_house\_sch\_l: gar\_tmp

#

g\_adr\_hist\_sch: gar\_tmp

#

# stage\_3\_I

#

gar\_fias\_crt\_idx:

descr: Создание рабочих индексов в схеме gar\_fias. False-удаление, True-удаление,создание.

params:

p\_skip: False

p\_sw: True

#

# stage\_3\_9

#

gar\_fias\_set\_gap:

descr: Заполнение таблицы с активными предшественниками

params\_adr\_area:

p\_skip: False

params\_adr\_house:

p\_skip: True

#

gar\_fias\_update\_children:

descr: Обработка таблицы с активными предшественниками, переподчинение и деактивация

param\_adr\_area:

p\_skip: False

p\_date\_1: DATE(current\_date) # НЕ ИСПОЛЬЗУЕТСЯ.

p\_obj\_level: [6,7,8]

p\_date\_2: DATE(current\_date) # Дата, на которую выбираются данные !!!

#

# stage\_3\_0

#

gar\_tmp\_set\_logged:

descr: Установка признака LOGGED/UNLOGGED у таблиц в схеме gar\_tmp

params:

p\_skip: True

p\_sw: True # False - unlogged

gar\_tmp\_clear\_tbl:

descr: Очистка данных во временной схеме.

params:

p\_skip: False

p\_sw: [1,2]

#

gar\_tmp\_switch\_indexies:

descr: Смена индексного покрытия в отдалённых таблицах

params:

p\_skip: True

p\_skip\_adr\_object: True

params\_street:

p\_uniq\_sw: True

params\_house:

p\_uniq\_sw: True

params\_objects:

p\_uniq\_x2: True

#

# stage\_3\_1

#

unload\_data\_adr\_area\_type:

descr: Загрузка данных из таблицы ADR\_AREA\_TYPE

params:

p\_skip: True

#

unload\_data\_adr\_street\_type:

descr: Загрузка данных из таблицы ADR\_STREET\_TYPE

params:

p\_skip: True

#

unload\_data\_adr\_house\_type:

descr: Загрузка данных из таблицы ADR\_HOUSE\_TYPE

params:

p\_skip: True

#

unload\_data\_adr\_area:

descr: Загрузка регионального фрагмента из таблицы ADR\_AREA

params:

p\_skip: True

#

unload\_data\_adr\_street:

descr: Загрузка регионального фрагмента из таблицы ADR\_STREET

params:

p\_skip: True

#

unload\_data\_adr\_house:

descr: Загрузка регионального фрагмента из таблицы ADR\_HOUSE

params:

p\_skip: True

p\_skip\_adr\_object: True

#

seq\_settings:

descr: Установка последовательностей

params:

p\_skip: False

p\_seq\_name: gar\_tmp.obj\_seq

p\_seq\_hist\_name: gar\_tmp.obj\_hist\_seq

p\_init\_value: 100000000

#

sq\_adr\_area\_sch: gar\_tmp

sq\_adr\_street\_sch: gar\_tmp

sq\_adr\_house\_sch: gar\_tmp

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

#

# stage\_3\_3

#

data\_aggregation:

descr: Агрегация данных

agg\_params:

p\_skip\_agg: False

p\_descr\_agg: Агрегация параметров

p\_param\_list: [5,6,7,8,10,11]

#

agg\_adr\_area:

p\_skip\_adr\_area: False

p\_descr\_adr\_area: Агрегация адресных пространств

params\_aa:

# "p\_date" - Не ИСПОЛЬЗОВАТЬ СЕЙЧАС.!!!

p\_date: DATE(current\_date) # Дата, на которую выбираются данные НЕТ

p\_obj\_level: 10 # Уровень объекта

p\_oper\_type\_ids: Null # ARRAY [1,2,3,5] # Типы операций НЕТ

#

agg\_adr\_house:

p\_skip\_adr\_house: False

p\_descr\_adr\_house: Агрегация домовладений

params\_ah:

# "p\_date" - Не ИСПОЛЬЗОВАТЬ СЕЙЧАС.!!!

p\_date: DATE(current\_date) # Дата, на которую выбираются данные НЕТ

p\_parent\_obj: Null # Родительский объект НЕТ

#

obj\_fias:

p\_skip\_obj\_fias: False

# 2022-09-28

p\_switch\_adr\_area\_sch: True # True - Локальная схема, False - Отдалённая

p\_switch\_adr\_street\_sch: True # True - Локальная схема, False - Отдалённая

# 2022-09-28

p\_switch\_adr\_house\_sch: True # True - Локальная схема, False - Отдалённая

p\_descr\_obj\_fias: Заполнение управляющей таблицы "gar\_tmp.xxx\_obj\_fias"

Изменится описание процесса, активных, выполняемых этапов будет на два меньше. Соответственно изменится и сам процесс.

Далее сразу перейдём к команде типа «6», как к наиболее актуальной и малоосвещённой возможности «fias\_loader».

**4.2.2) Большая команда типа «6».** Описание этапов процесса Постобработки. Поскольку структура YAML-файла не описывалась раннее, то здесь дадим её описание и сразу перейдём к параметризации процесса. Важно рассмотреть такие варианты:

1. Как выполнить Постообработку и выгрузить обновления в текстовые файлы.
2. Как выполнить только выгрузку данных.
3. Как переключить процесс с выгрузки в тестовые файлы на прямое обновление целевой БД, выполяемое посредством FDW.

**4.2.2.1. Описание файла «stage\_6.yaml».**

#

# 2022-05-19. Stage\_6. Новая реализация, только управление процессами, вызовы функций

# в коде "stage\_6\_proc.py"

# " Version 0.3.0 Build 2023-01-28"

#

control\_params:

stage\_6\_0: True # Начальная отметка в log (master - запись).

stage\_6\_1: True # adr\_area - постобработка и выгрузка + отметка в LOG

stage\_6\_2: True # adr\_street - постобработка и выгрузка + отметка в LOG

stage\_6\_3: True # adr\_house - постобработка и выгрузка + отметка в LOG

#

mogrify\_6\_0: False

mogrify\_6\_1: False

mogrify\_6\_2: False

mogrify\_6\_3: False

global\_params:

g\_region\_id: 2 # Адыгея -- p\_node\_id

g\_fserver\_nmb: 13 # На 5435 unnsi\_prdl -- p\_id\_region

g\_kd\_export\_type: False # True: с использованием FDW, False: текстовый файл

g\_seq\_name: gar\_tmp.obj\_seq # Региональная последовательность

g\_file\_path: ../tmp/up\_prd\_ # Путь для хранения экспортируемых файлов

g\_git\_path: ../abr\_upload/A\_FIAS\_LOADER # Источник для копирования IDX-файлов, SH-скриптов.

#----------------------------------------------------

# ,p\_dt\_gar\_version := {0}::date

#----------------------------------------------------

#

g\_adr\_area\_sch: unnsi # Удалённые/Общие схемы. Общая база.

g\_adr\_street\_sch: unnsi

g\_adr\_house\_sch: unnsi

#

g\_adr\_area\_sch\_l: gar\_tmp # Локальные схемы, каждая для своей региональной базы.

g\_adr\_street\_sch\_l: gar\_tmp

g\_adr\_house\_sch\_l: gar\_tmp

#

g\_adr\_hist\_sch: gar\_tmp

#

# stage\_6\_0

#

unnsi\_save\_version:

descr: Сохранение записи в журнале выгрузок (master-запись).

# -- ,p\_nm\_object := {2}::text

# -- ,p\_qty\_main := {3}::integer

# -- ,p\_qty\_aux := {4}::integer

#

# stage\_6\_1

#

unnsi\_adr\_area\_upload:

descr: Постанализ и выгрузка адресных пространств.

params\_proc:

p\_bound\_date: 2022-01-01 #

p\_post\_proc\_skip: False # Пропустили постобработку,

p\_post\_script: ../../A\_FIAS\_LOADER/GAR\_TMP\_PCG\_TRANS/DO/check\_parent\_less.sql

p\_post\_upload\_skip: False # Пропустили выгрузку, (в любом из двух видов).

p\_sch\_type: True # Только для выгрузки в файл, при "g\_kd\_export\_type": False

# False - использование локальной схемы (g\_adr\_area\_sch\_l)

# True - -- "" --удалённой/общей -- (g\_adr\_area\_sch)

#

p\_drop\_remote\_idxs: True # Только при "g\_kd\_export\_type: True"

# Не забыть создать индесное покрытие после обработки.

#

# stage\_6\_2

#

unnsi\_adr\_street\_upload:

descr: Постанализ и выгрузка элементов уличной-дорожной сети.

params\_proc:

p\_bound\_date: 2022-01-01

p\_post\_proc\_skip: False # Пропустили постобработку.

p\_post\_script:

p\_post\_upload\_skip: False # Пропустили выгрузку, (в любом из двух видов).

p\_sch\_type: True # Только для выгрузки в файл, при "g\_kd\_export\_type": False

# False - использование локальной схемы (g\_adr\_street\_sch\_l)

**#** True - -- "" --удалённой/общей -- (g\_adr\_street\_sch)

#

p\_drop\_remote\_idxs: True # Только при "g\_kd\_export\_type: True"

# Не забыть создать индесное покрытие после обработки.

#

# stage\_6\_3

#

unnsi\_adr\_house\_upload:

descr: Постанализ и выгрузка строений (дома и пр.).

params\_proc:

p\_bound\_date: 2022-01-01

p\_post\_proc\_skip: False # Пропустили постобработку.

p\_post\_script:

p\_post\_upload\_skip: False # Пропустили выгрузку, (в любом из двух видов).

p\_sch\_type: True # Только для выгрузки в файл, при "g\_kd\_export\_type": False

# False - использование локальной схемы (g\_adr\_house\_sch\_l)

# True - -- "" --удалённой/общей -- (g\_adr\_house\_sch)

#

p\_drop\_remote\_idxs: True # Только при "g\_kd\_export\_type: True"

# Не забыть создать индесное покрытие после обработки.

| **Номер секции** | **Наименование** | **Описание** |
| --- | --- | --- |
| 1 | control\_params: | Управляет выполнением секций «stage\_6\_0» - «stage\_6\_3». |
| 2 | global\_params: | Глобальные параметры, общие для всех секций. Направление экспорта обновлений зависит от состояния глобального переключателя «g\_kd\_export\_type». При «g\_kd\_export\_type» = True обновляется целевая БД, посредством обновления сторонних таблиц, размещённых в схеме «unnsi». При «g\_kd\_export\_type» = False, все обновления выгружаются в тестовые файлы, для последующего обновления целевой базы. Каталог для выгрузки обновлений определён в глобальном параметре «g\_file\_path» , в текущей конфигурации имя каталога = «../tmp/up\_prd\_», к нему добавляется значение «дату-версии». Например «../tmp/up\_prd\_20221226». В каталоге всегда существует папка «DATA», в которой размещены файлы с обновлениями соответствующего адресного региона и shell-сценарий «load\_upd\_wi.sh» запускающий процесс обновления. |
| 3 | stage\_6\_0 | Сохранение записи в журнале выгрузок (master-запись). |
| 4 | stage\_6\_1 | Постанализ и выгрузка адресных пространств. |
| 5 | stage\_6\_2 | Постанализ и выгрузка элементов уличной-дорожной сети. |
| 6 | stage\_6\_3 | Постанализ и выгрузка строений (дома и пр.). |

Рис 4.2 Описание основных частей управляющего файла «stage\_6.yaml».

Далее, перечислим этапы из которых состоит процесс Постобработка.

* Сохранить главную запись в журнале выгрузок.
* Постобработка, постанализ и экспорт обновлений адресных пространств.
* Постобработка и экспорт обновлений элементов улично-дорожной сети.
* Постобработка и экспорт обновлений домов и строений.

Этапы постобработки связаны с переходом от процессингового индексного покрытия к экплуатационному и фильтрацией артефактов, нарушающих уникальность эксплуатационного индексного покрытия.

Что нужно сделать для того, чтобы выполнив Постобработку не выполнять выгрузку обновлений ??

Выключаем этап «stage\_6\_0», далее в этапах «stage\_6\_1», «stage\_6\_2», «stage\_6\_3»

включаем опции «p\_post\_upload\_skip»: True, и запускаем процесс:

$> load\_mainGar\_2.py hosts\_total\_xx\_0123.yaml stage\_6.csv „2022-12-26

Файл «stage\_6.yaml» должен быть таким (Изменения выделены **ЧЁРНЫМ**):

#

# 2022-05-19. Stage\_6. Новая реализация, только управление процессами, вызовы функций

# в коде "stage\_6\_proc.py"

# " Version 0.3.0 Build 2023-01-28"

#

control\_params:

**stage\_6\_0: False** # Начальная отметка в log (master - запись).

stage\_6\_1: True # adr\_area - постобработка и выгрузка + отметка в LOG

stage\_6\_2: True # adr\_street - постобработка и выгрузка + отметка в LOG

stage\_6\_3: True # adr\_house - постобработка и выгрузка + отметка в LOG

#

mogrify\_6\_0: False

mogrify\_6\_1: False

mogrify\_6\_2: False

mogrify\_6\_3: False

global\_params:

g\_region\_id: 2 # Адыгея -- p\_node\_id

g\_fserver\_nmb: 13 # На 5435 unnsi\_prdl -- p\_id\_region

g\_kd\_export\_type: False # True: с использованием FDW, False: текстовый файл

g\_seq\_name: gar\_tmp.obj\_seq # Региональная последовательность

g\_file\_path: ../tmp/up\_prd\_ # Путь для хранения экспортируемых файлов

g\_git\_path: ../abr\_upload/A\_FIAS\_LOADER # Источник для копирования IDX-файлов, SH-скриптов.

#----------------------------------------------------

# ,p\_dt\_gar\_version := {0}::date

#----------------------------------------------------

#

g\_adr\_area\_sch: unnsi # Удалённые/Общие схемы. Общая база.

g\_adr\_street\_sch: unnsi

g\_adr\_house\_sch: unnsi

#

g\_adr\_area\_sch\_l: gar\_tmp # Локальные схемы, каждая для своей региональной базы.

g\_adr\_street\_sch\_l: gar\_tmp

g\_adr\_house\_sch\_l: gar\_tmp

#

g\_adr\_hist\_sch: gar\_tmp

#

# stage\_6\_0

#

unnsi\_save\_version:

descr: Сохранение записи в журнале выгрузок (master-запись).

# -- ,p\_nm\_object := {2}::text

# -- ,p\_qty\_main := {3}::integer

# -- ,p\_qty\_aux := {4}::integer

#

# stage\_6\_1

#

unnsi\_adr\_area\_upload:

descr: Постанализ и выгрузка адресных пространств.

params\_proc:

p\_bound\_date: 2022-01-01 #

p\_post\_proc\_skip: False # Пропустили постобработку,

p\_post\_script: ../../A\_FIAS\_LOADER/GAR\_TMP\_PCG\_TRANS/DO/check\_parent\_less.sql

**p\_post\_upload\_skip: True**  # Пропустили выгрузку, (в любом из двух видов).

p\_sch\_type: True # Только для выгрузки в файл, при "g\_kd\_export\_type": False

# False - использование локальной схемы (g\_adr\_area\_sch\_l)

# True - -- "" --удалённой/общей -- (g\_adr\_area\_sch)

#

p\_drop\_remote\_idxs: True # Только при "g\_kd\_export\_type: True"

# Не забыть создать индесное покрытие после обработки.

#

# stage\_6\_2

#

unnsi\_adr\_street\_upload:

descr: Постанализ и выгрузка элементов уличной-дорожной сети.

params\_proc:

p\_bound\_date: 2022-01-01

p\_post\_proc\_skip: False # Пропустили постобработку.

p\_post\_script:

**p\_post\_upload\_skip: True** # Пропустили выгрузку, (в любом из двух видов).

p\_sch\_type: True # Только для выгрузки в файл, при "g\_kd\_export\_type": False

# False - использование локальной схемы (g\_adr\_street\_sch\_l)

**#** True - -- "" --удалённой/общей -- (g\_adr\_street\_sch)

#

p\_drop\_remote\_idxs: True # Только при "g\_kd\_export\_type: True"

# Не забыть создать индесное покрытие после обработки.

#

# stage\_6\_3

#

unnsi\_adr\_house\_upload:

descr: Постанализ и выгрузка строений (дома и пр.).

params\_proc:

p\_bound\_date: 2022-01-01

p\_post\_proc\_skip: False # Пропустили постобработку.

p\_post\_script:

**p\_post\_upload\_skip: True** # Пропустили выгрузку, (в любом из двух видов).

p\_sch\_type: True # Только для выгрузки в файл, при "g\_kd\_export\_type": False

# False - использование локальной схемы (g\_adr\_house\_sch\_l)

# True - -- "" --удалённой/общей -- (g\_adr\_house\_sch)

#

p\_drop\_remote\_idxs: True # Только при "g\_kd\_export\_type: True"

# Не забыть создать индесное покрытие после обработки.

При необходимости выполнить только обновление целевой базы меняем снова исходный YAML-файл: ,включаем опцию прямого обновления целевой базы, включаем пропуск каем этапов постобработки и запускаем процесс:

$> load\_mainGar\_2.py hosts\_total\_xx\_0123.yaml stage\_6.csv „2022-12-26

При этом YAML-файл должен быть таким: (изменения выделены **ЧЁРНЫМ**):

#

# 2022-05-19. Stage\_6. Новая реализация, только управление процессами, вызовы функций

# в коде "stage\_6\_proc.py"

# " Version 0.3.0 Build 2023-01-28"

#

control\_params:

**stage\_6\_0: True**  # Начальная отметка в log (master - запись).

stage\_6\_1: True # adr\_area - постобработка и выгрузка + отметка в LOG

stage\_6\_2: True # adr\_street - постобработка и выгрузка + отметка в LOG

stage\_6\_3: True # adr\_house - постобработка и выгрузка + отметка в LOG

#

mogrify\_6\_0: False

mogrify\_6\_1: False

mogrify\_6\_2: False

mogrify\_6\_3: False

global\_params:

g\_region\_id: 2 # Адыгея -- p\_node\_id

g\_fserver\_nmb: 13 # На 5435 unnsi\_prdl -- p\_id\_region

**g\_kd\_export\_type: True**  # True: с использованием FDW, False: текстовый файл

g\_seq\_name: gar\_tmp.obj\_seq # Региональная последовательность

g\_file\_path: ../tmp/up\_prd\_ # Путь для хранения экспортируемых файлов

g\_git\_path: ../abr\_upload/A\_FIAS\_LOADER # Источник для копирования IDX-файлов, SH-скриптов.

#----------------------------------------------------

# ,p\_dt\_gar\_version := {0}::date

#----------------------------------------------------

#

g\_adr\_area\_sch: unnsi # Удалённые/Общие схемы. Общая база.

g\_adr\_street\_sch: unnsi

g\_adr\_house\_sch: unnsi

#

g\_adr\_area\_sch\_l: gar\_tmp # Локальные схемы, каждая для своей региональной базы.

g\_adr\_street\_sch\_l: gar\_tmp

g\_adr\_house\_sch\_l: gar\_tmp

#

g\_adr\_hist\_sch: gar\_tmp

#

# stage\_6\_0

#

unnsi\_save\_version:

descr: Сохранение записи в журнале выгрузок (master-запись).

# -- ,p\_nm\_object := {2}::text

# -- ,p\_qty\_main := {3}::integer

# -- ,p\_qty\_aux := {4}::integer

#

# stage\_6\_1

#

unnsi\_adr\_area\_upload:

descr: Постанализ и выгрузка адресных пространств.

params\_proc:

p\_bound\_date: 2022-01-01 #

**p\_post\_proc\_skip: True** # Пропустили постобработку,

p\_post\_script: ../../A\_FIAS\_LOADER/GAR\_TMP\_PCG\_TRANS/DO/check\_parent\_less.sql

p\_post\_upload\_skip: False # Пропустили выгрузку, (в любом из двух видов).

p\_sch\_type: True # Только для выгрузки в файл, при "g\_kd\_export\_type": False

# False - использование локальной схемы (g\_adr\_area\_sch\_l)

# True - -- "" --удалённой/общей -- (g\_adr\_area\_sch)

#

p\_drop\_remote\_idxs: True # Только при "g\_kd\_export\_type: True"

# Не забыть создать индесное покрытие после обработки.

#

# stage\_6\_2

#

unnsi\_adr\_street\_upload:

descr: Постанализ и выгрузка элементов уличной-дорожной сети.

params\_proc:

p\_bound\_date: 2022-01-01

**p\_post\_proc\_skip: True** # Пропустили постобработку.

p\_post\_script:

p\_post\_upload\_skip: False # Пропустили выгрузку, (в любом из двух видов).

p\_sch\_type: True # Только для выгрузки в файл, при "g\_kd\_export\_type": False

# False - использование локальной схемы (g\_adr\_street\_sch\_l)

**#** True - -- "" --удалённой/общей -- (g\_adr\_street\_sch)

#

p\_drop\_remote\_idxs: True # Только при "g\_kd\_export\_type: True"

# Не забыть создать индесное покрытие после обработки.

#

# stage\_6\_3

#

unnsi\_adr\_house\_upload:

descr: Постанализ и выгрузка строений (дома и пр.).

params\_proc:

p\_bound\_date: 2022-01-01

**p\_post\_proc\_skip: True** # Пропустили постобработку.

p\_post\_script:

p\_post\_upload\_skip: False # Пропустили выгрузку, (в любом из двух видов).

p\_sch\_type: True # Только для выгрузки в файл, при "g\_kd\_export\_type": False

# False - использование локальной схемы (g\_adr\_house\_sch\_l)

# True - -- "" --удалённой/общей -- (g\_adr\_house\_sch)

#

p\_drop\_remote\_idxs: True # Только при "g\_kd\_export\_type: True"

# Не забыть создать индесное покрытие после обработки.

**4.3) Управление отладкой.**

Отладка загрузчика адресных данных выполняется в случае возникновения ошибок в данных, либо внутренних ошибок в коде. Следует отметить, что в процессах загрузчика потоки команд и данных всегда делятся на две группы: процессы в Python-программах и процессы в хранимых процедурах/функциях. В настоящем время опции трассировки Python-кода практически отсутствуют и если появятся, то далеко не в следующем релизе. Основной акцент сделан на работу с хранимыми процедурами/функциями написанных на PLpg/SQL. И ещё один важный аспект, касающийся работы Python-программ с объектами БД, необходимо запомнить что:

* результаты выполнения SQL-запросов всегда попадут в файл «process.out» (поток stdout), и никогда не вернутся в инициировавшую запрос Python-программу,
* то-же самое относится и к выполнению хранимых процедур/функций.

Поэтому дальнейшем изложении будут рассмотрены два разделах:

1. Проверка потока SQL-команд, подаваемых из Python-программ в базу данных. Сейчас это относится только к большим командам, управляемых YAML-файлами (команды типов «3», «4», «6»), для этого используется опция MOGRIFY.
2. Автономная трассировка и отладка выделенного фрагмента кода, соответствующие примеры с отладочными сценариями будут рассмотрены. Для выполнения этого пункта необходимо установленное расширение «plpgsql\_check».

**4.3.1) Проверка потока SQL-команд.**

Выполним CSV-сценарий содержащий одну команду типа «4». К примеру цель заключается в локализации ошибки в функциях, выполняющих обновление элементов дорожно-планировочной структуры. С этой целью включим опции «Mogrify» в файле «stage\_41r.yaml».

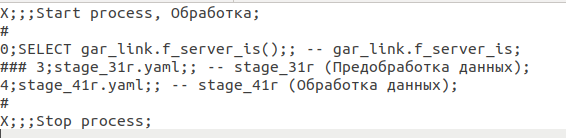


Рис 4.3 Сценарий, содержащий команду типа 4 «stage\_41r.yaml».

#

# 2022-03-11/2022-06-09

# VERSION\_STR = " Version 0.3.0 Build 2022-12-06"

#

control\_params:

stage\_4\_1: False

stage\_4\_2: False

stage\_a\_p: False

#

stage\_4\_3: True

stage\_4\_4: True

stage\_s\_p: False

#

stage\_4\_5: False

stage\_4\_6: False

stage\_h\_p: False

#

mogrify\_4\_1: False

mogrify\_4\_2: False

mogrify\_a\_p: False

#

**mogrify\_4\_3: True**

**mogrify\_4\_4: True**

mogrify\_s\_p: False

#

mogrify\_4\_5: False

mogrify\_4\_6: False

mogrify\_h\_p: False

Что произойдёт в случае запуска сценария ? Выполняться только этапы «stage\_4\_3» и «stage\_4\_4». В файл «process.err» запишутся только вызовы хранимых функций.

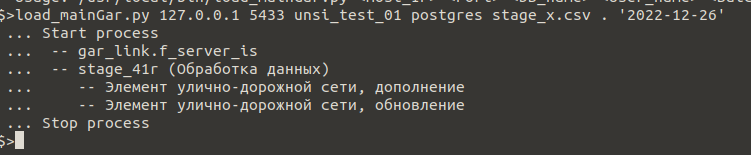


Рис 4.4 Запуск процесса.

Смотрим файл «process.err» в любом доступном текстовом редактор. **Синим** выделено обращение к функции, которую необходимо оттрассировать.

psql -h 127.0.0.1 -p 5433 -c "

SELECT \* FROM gar\_tmp\_pcg\_trans.f\_adr\_street\_ins (

p\_schema\_data := 'gar\_tmp'::text -- Обновляемая схема с данными

,p\_schema\_etl := 'gar\_tmp'::text -- Схема эталон

,p\_schema\_hist := 'gar\_tmp'::text -- Схема для хранения исторических данных

,p\_nm\_guids\_fias := NULL::uuid[] -- Список обрабатываемых GUID, NULL - все.

,p\_sw\_hist := True::boolean -- TRUE -> Создаётся историческая запись.

);" unsi\_test\_01 -U postgres 1>> process.out 2>> process.err

psql -h 127.0.0.1 -p 5433 -c "

SELECT \* FROM gar\_tmp.adr\_street\_hist WHERE (id\_region = 0) AND (date\_create >= (now() - INTERVAL '1 DAY')) ORDER BY date\_create;;

" unsi\_test\_01 -U postgres 1>> process.out 2>> process.err

psql -h 127.0.0.1 -p 5433 -c "

**SELECT \* FROM gar\_tmp\_pcg\_trans.f\_adr\_street\_upd (**

**p\_schema\_data := 'gar\_tmp'::text -- Обновляемая схема с данными**

**,p\_schema\_etl := 'gar\_tmp'::text -- Схема эталон,**

**,p\_schema\_hist := 'gar\_tmp'::text -- Схема для хранения исторических данных**

**,p\_nm\_guids\_fias := NULL::uuid[] -- Список обрабатываемых GUID, NULL - все.**

**,p\_sw\_hist := True::boolean -- TRUE -> Создаётся историческая запись.**

**,p\_sw\_duble := False::boolean -- TRUE -> Обязательное выявление );**" unsi\_test\_01 -U postgres 1>> process.out 2>> process.err

psql -h 127.0.0.1 -p 5433 -c "

SELECT \* FROM gar\_tmp.adr\_street\_hist WHERE (id\_region = 0) AND (date\_create >= (now() - INTERVAL '1 DAY')) ORDER BY date\_create;;

" unsi\_test\_01 -U postgres 1>> process.out 2>> process.err

Далее выполняем трассировку «gar\_tmp\_pcg\_trans.f\_adr\_street\_upd()». Cоздаём SQL-сценарий, оформленный в виде файла «\_TEST\_stage\_4.4.sql». На рисунке 4.5 приведён его фрагмент:

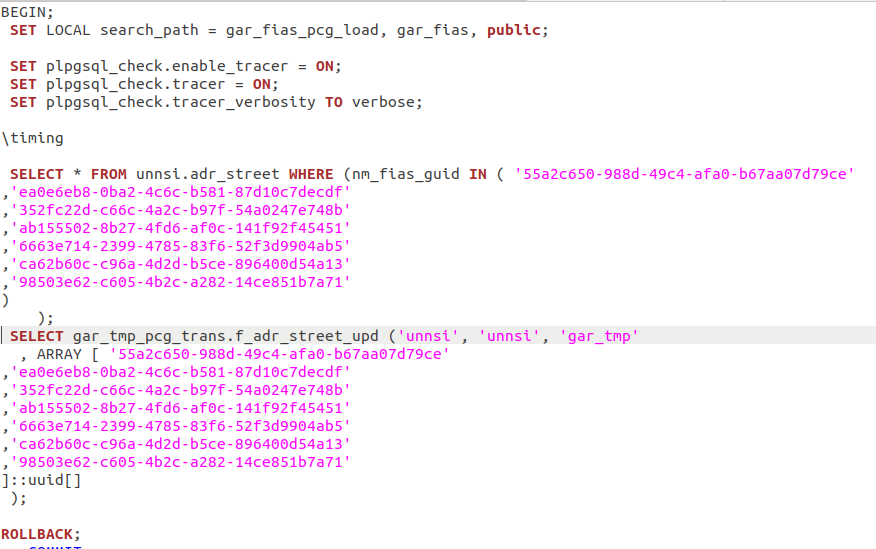
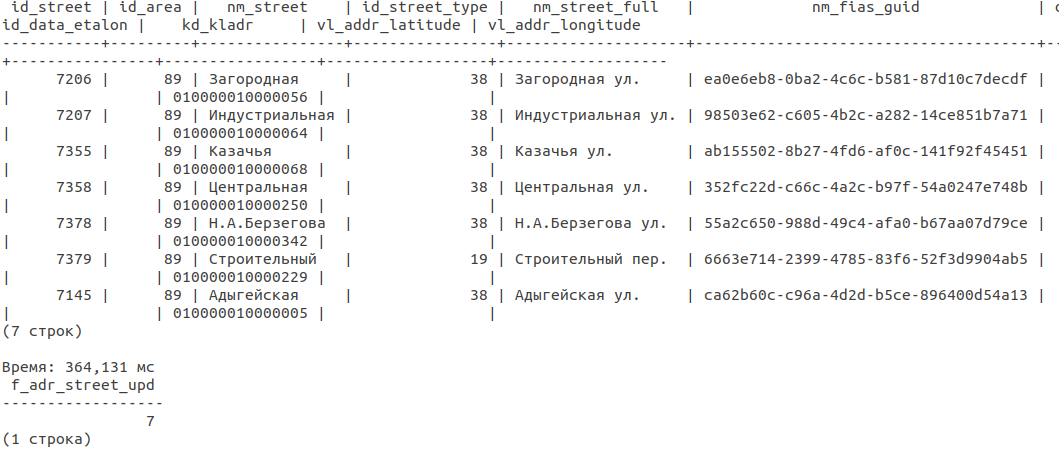


Рис 4.5 Фрагмент SQL-сценарий.

Функция будет выполнять обновление выбранных элементов дорожно-планировочной структуры, их GUID собраны в массиве, передаваемом в функцию в последнем фактическом аргументе. Выше было упомянуто про расширение «plpgsql\_check», на данном этапе оно выполняет ключевую роль. Запускаем отладочный SQL — сценарий

$> psql -h 127.0.0.1 -U postgres -p 5433 -f \_TEST\_stage\_4.4.sql 1>> process.out 2>> process.err

По его завершению смотрим файлы «process.out» и «process.err». В первом из них находятся результаты выполнения функции во втором — трасса. Управление трассировщиком «plpgsql\_check» является темой для отдельного документа и здесь не рассматривается. Анализ трассы выполнения функции оказывает помощь в процессе поиска ошибок.

 Рис 4.6 Результаты выполнения SQL-сценария.

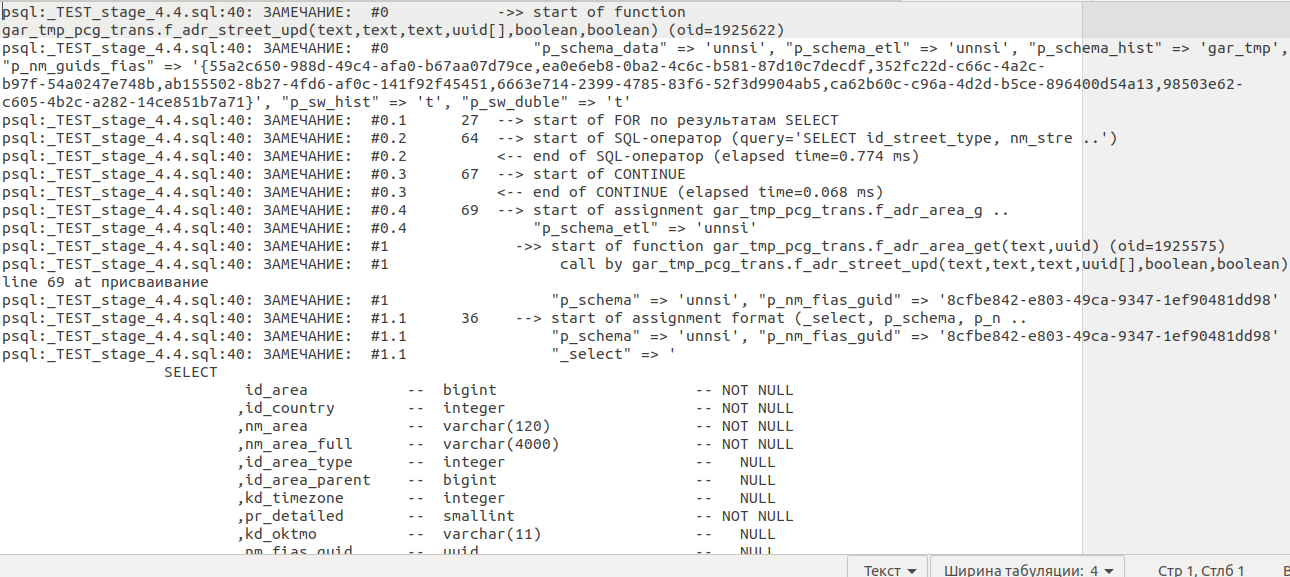


Рис 4.7 Трасса выполнения функции «f\_adr\_street\_upd ()».

**5. Развёртывание кластера БД.**

С чего начинается развёртывание Fias\_Loader"? Первое: необходимо определить, для каких регионов будут поставляться обновления и дополнения адресных данных. Для это цели из таблицы unnsi.adr\_area делается выборка и на её формируется CSV-файл. Тест запроса:

SELECT \* FROM unnsi.adr\_area WHERE (id\_area\_parent IS NULL) AND

(id\_area <= 99) AND (dt\_data\_del IS NULL) AND (id\_data\_etalon IS null) ORDER BY id\_area;

Исключив г. Байконур получаем 85 строк. Ниже представлен фрагмент такой таблицы.



Рис 4 Список регионов в исходном состоянии.

Если нет возможности экспортировать данные из базы в книгу MS Excel, таблицу придётся создать вручную. Структура таблицы:

Первый столбец - номер региона из ГАР-ФИАС,

Второй - номер региона из unnsi.adr\_area (ЕС НСИ),

Третий - код страны,

Четвёртый — название региона,

Пятый содержит полное название региона.

Номера адресных регионов из ГАР-ФИАС и номера регионов из ЕС НСИ как правило не совпадают. Когда-то решили, что система кодирования регионов в ГАР-ФИАС не совершенна и … изобрели свою. Поэтому нужно свести обе кодировки в едином документе, вручную, для этого необходим хороший XML-редактор с функциями поиска и много терпения.

Номера адресных регионов из ГАР-ФИАС используется для формирования имён кластерных БД, папок в файловой системе, они являются внешними идентификаторами. Номера регионов из ЕС НСИ является внутренними идентификаторами, которые необходимы при обновлении адресных данных и выгрузке инкрементальных обновлений..



Рис 5 Список регионов, ID Гар-Фиас и ЕС НСИ связаны.

Еxcel таблица содержащая список адресных регионов, является основой на которой создаётся кластер БД и управляющие YAML-файлы. В настоящее время есть 4 CSV-файла, которые являются эталонами. Без них вся работа остановится.

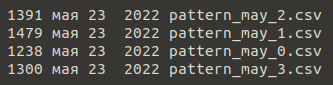


Рис 6 Список эталонных файлов.

Что ещё будет необходимым: git-репозиторий http://gitlab.abr-region.ru/ccrm/gar, в котором, в папке «fias\_loader» находятся сценарии «stage\_1.csv» и «stage\_2.csv». Они понадобятся на завершающем этапе развёртывания адресной базы.

Следующий этап заключаетсяв создании скриптов для развёртывания БД и управляющих YAML-файлов. Выполнять их можно в произвольной последовательности, важна правильность каждого этапа.

| 12 | load\_mainCrtScripts.py | Используется перед обработкой нового адресного архива либо перед начальной инициализацией кластера БД. Создаёт sql-скрипты. Управляется csv - файлом «pattern\_xx.csv» и набором параметров, передаваемых в командной строке. |
| --- | --- | --- |
| 13 | load\_mainCrtYaml.py | Создаёт коллекцию YAML-файлов, Используется только при первоначальном развороте кластера Управляется csv - файлом «pattern\_xx.csv» и набором параметров, передаваемых в командной строке. |

Начинаем всегда с "load\_mainCrtScripts.py", утилита маленькая, но основополагающая.

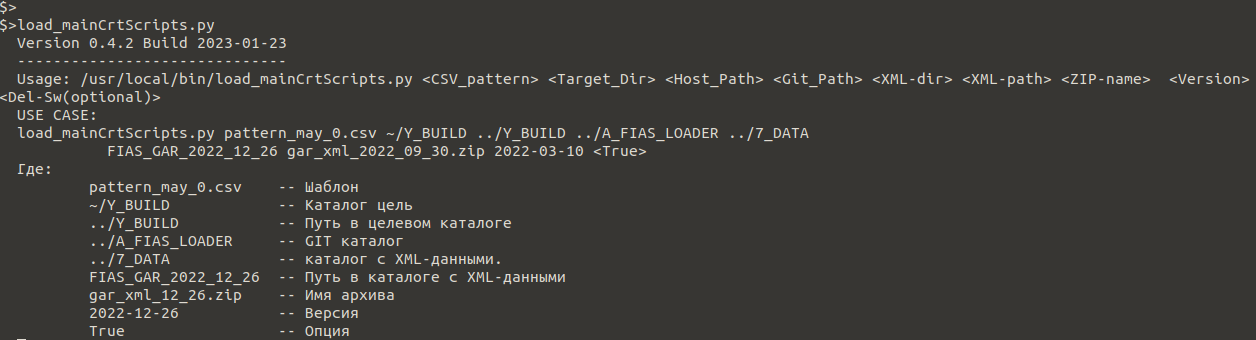


Рис 7 Вызов утилиты load\_mainCrtScripts.py

.

Вызов утилиты без параметров выводит подсказку. Для учебных целей принимаем следующий набор параметров:

| **№ пп** | **Наименование** | **Значение** |
| --- | --- | --- |
| 1 | Точка старта утилиты | «/home/rootadmin/abr\_upload» |
| 2 | CSV-шаблон | «pattern\_may\_0.csv». Сейчас находится в точке старта. |
| 3 | Каталог-цель | «/home/rootadmin/example/Y\_BUILD» |
| 4 | Путь в целевом каталоге | «Y\_BUILD» Дубль, (будет устранено в следующем релизе) |
| 5 | GIT-каталог | A\_FIAS\_LOADER (ссылка на /home/rootadmin/gar/fias\_loader) |
| 6 | Каталог с XML-данными | «/media/rootadmin/Transcend» |
| 7 | Путь в каталоге с XML-данными | «FIAS\_GAR\_2022\_12\_26» |
| 8 | Имя архива: | «gar\_xml\_12\_26.zip» |
| 9 | Версия: | «2022\_12\_26» |
| 10 | Опция: | True |

Выполняем несколько команд.

$>mkdir ~/example

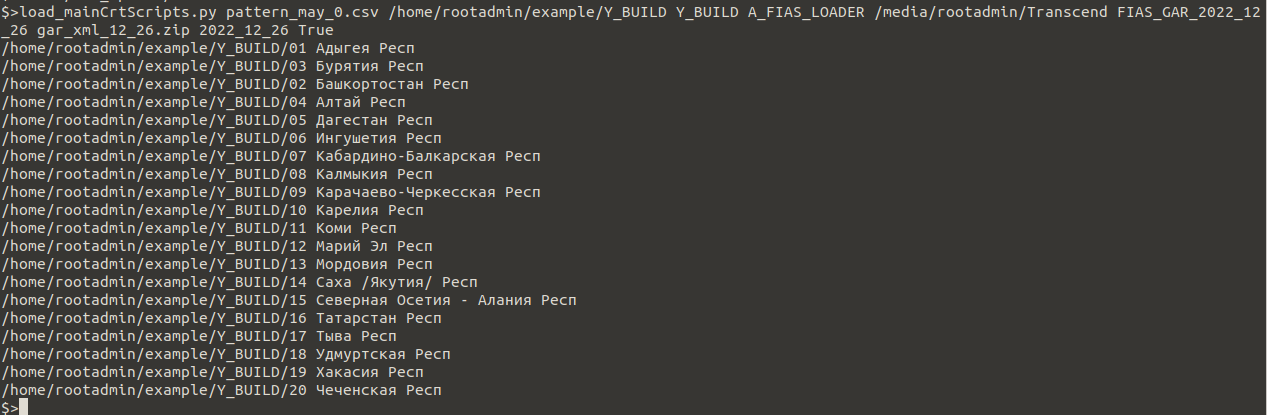
**$>ln -s /home/rootadmin/gar/fias\_loader A\_FIAS\_LOADER <<-– Необходима.**

$>mkdir ~/example/Y\_BUILD

$>cd ~/example/Y\_BUILD

$>l

Сейчас там пусто. Далее выполняем команду, результаты на рисуноке 8.

 Рис 8 Выполнена утилита load\_mainCrtScripts.py

Смотрим, что творится в каталоге «~/example/Y\_BUILD»

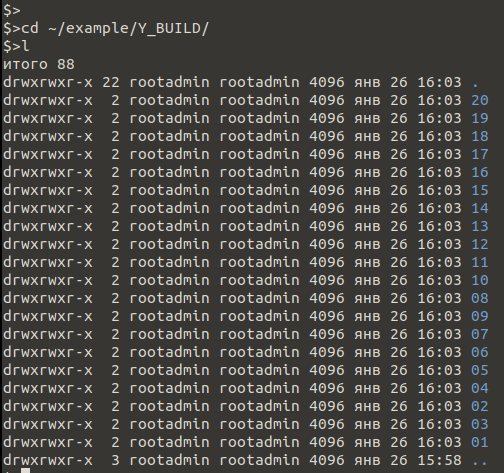


Рис 9 Результат выполнения утилита load\_mainCrtScripts.py

Каждому региону соответствует своя папка. Выбираем папку 05 и смотрим её содержимое.

$>l 05

итого 24

drwxrwxr-x 22 rootadmin rootadmin 4096 янв 26 16:03 ..

drwxrwxr-x 2 rootadmin rootadmin 4096 янв 26 16:03 .

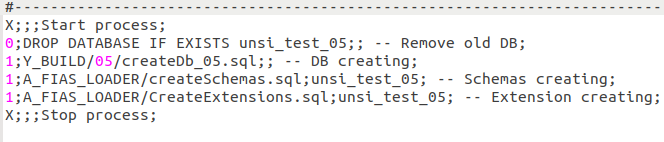
-rw-rw-r-- 1 rootadmin rootadmin 80 янв 26 16:03 commentDb\_05.sql

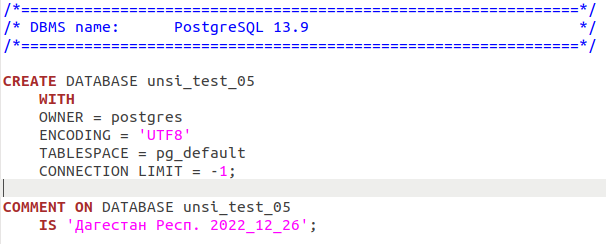
-rw-rw-r-- 1 rootadmin rootadmin 420 янв 26 16:03 createDb\_05.sql

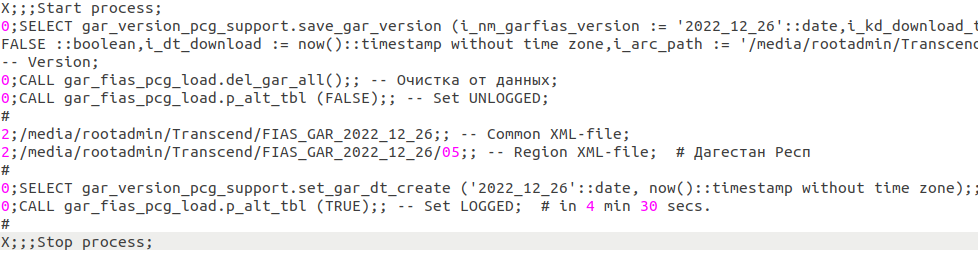
-rw-rw-r-- 1 rootadmin rootadmin 396 янв 26 16:03 stage\_0\_05.csv

-rw-rw-r-- 1 rootadmin rootadmin 828 янв 26 16:03 stage\_gar\_c\_05.csv

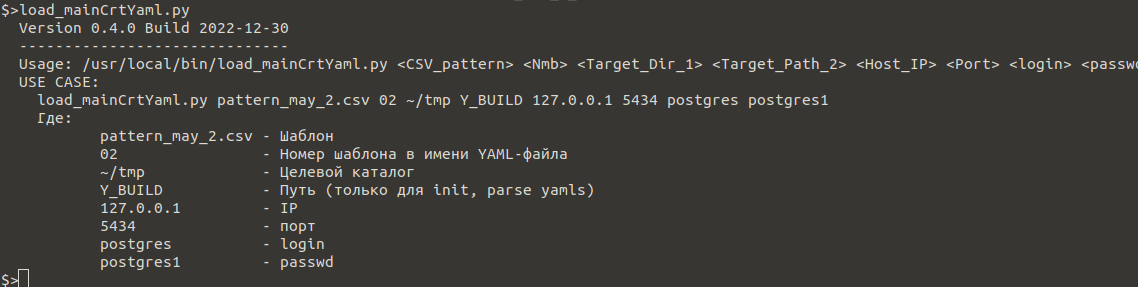
Первый идёт скрипт с комментарием к БД, содержащий наименование региона и дату актуальности загружаемых данных. Вторым скрипт создающий БД. Третим CSV-сценарий выполняющий равёртывние БД. Четвёртым следует CSV-сценарий, участвующий в процессе parsing XML-данных. Далее рассмотрим содержание полученных файлов.

 Рис 10 CSV-сценарий, обеспечивающий создание БД для 05 региона

 Рис 11 SQL-сценарий, созда.obq БД для 05 региона

 Рис 12 CSV-сценарий, обеспечивающий parsing XML-данных для 05 региона (Республика Дагестан).

Далее создаются управляющие YAML-файлы. Создаваемый кластер баз будет размещён на локальном хосте 127.0.0.1 порт 5434. При вызове без параметра утилита выдаёт свою версию с вариантами использования.

 Рис 13 Вызов утилиты load\_mainCrtYaml.py.

Используем следующий набор параметров:

| **№ пп** | **Наименование** | **Значение** |
| --- | --- | --- |
| 1 | Точка старта утилиты | «/home/rootadmin/abr\_upload» |
| 2 | CSV-шаблон | «pattern\_may\_0.csv». Сейчас находится в точке старта. |
| 3 | Номер в именах YAML-файлов | 0 |
| 4 | Каталог-цель | «/home/rootadmin/example» Сюда упадут создаваемые YAML-файлы. |
| 5 | Путь в целевом каталоге | «Y\_BUILD» |
| 6 | IP хоста | «127.0.0.1» |
| 7 | Порт | «5433» |
| 8 | Имя пользователя | «postgres» |
| 9 | Пароль | «postgres1» |

 Рис 14 Выполнение утилиты load\_mainCrtYaml.py.

Смотрим результаты работы утилиты:

$>pwd

/home/rootadmin/abr\_upload

$>l ~/example

drwxrwxr-x 3 4096 янв 26 17:22 .

-rw-rw-r-- 1 7193 янв 26 17:22 hosts\_init\_xx\_0.yaml

-rw-rw-r-- 1 7373 янв 26 17:22 hosts\_parse\_xx\_0.yaml

-rw-rw-r-- 1 4973 янв 26 17:22 hosts\_total\_xx\_0.yaml

drwxrwxr-x 22 4096 янв 26 17:22 Y\_BUILD

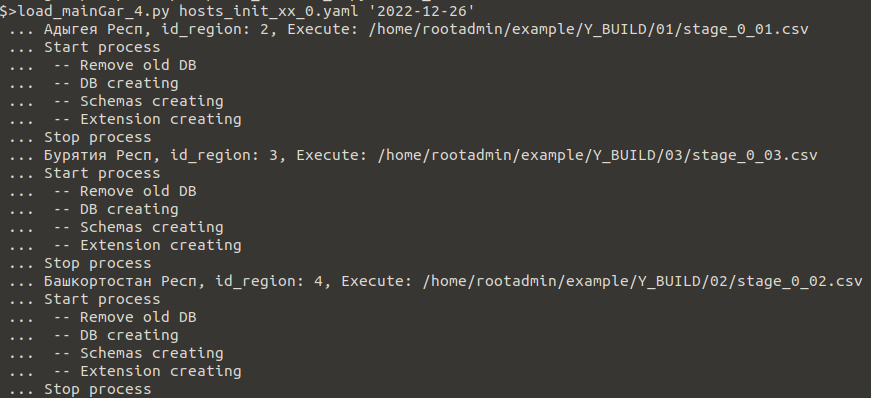
drwxr-xr-x 133 12288 янв 26 16:02 ..

Далее приводится фрагмент созданного YAML-файла, выполняющего создание кластера БД.



Рис 15 Фрагмент файла hosts\_init\_xx\_0.yaml.

Далее создаём кластер. Используем программу load\_mainGar\_4.py использующую yaml-файлы содержащие описания как строк соединений к БД, так и выполняемые действия.

 Рис 16 Процесс создания кластера БД.

$>load\_mainGar\_4.py

Version 0.5.5 Build 2023-01-13

Usage: /usr/local/bin/load\_mainGar\_4.py <Yaml\_file> <Version>

$>load\_mainGar\_4.py hosts\_init\_xx\_0.yaml '2022-12-26'

Кластер баз создался, развёрнуты схемы и расширения:

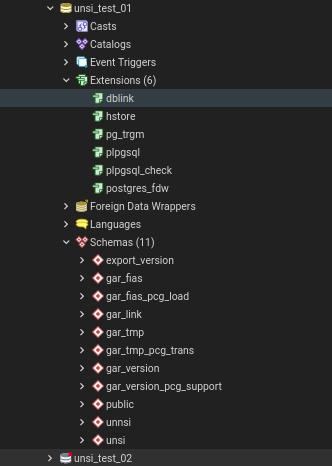


Рис 17 Фрагмент развёрнутого кластера БД.

Созданные базы не имеют структур хранения и хранимых функций/процедур. На следующих

двух этапах будет выполнено создание структуры хранения (таблицы и типы) и развертывание функциональных пакетов. Используем два CSV-сценария:

-- stage\_1.csv - создание структур хранения

-- stage\_1.csv — развёртывание функциональных пакетов.

Следует помнить, что каталог «**A\_FIAS\_LOADER**» является soft-link на git-репозиторий проекта.

$>load\_mainGar\_2.py hosts\_total\_xx\_0.yaml A\_FIAS\_LOADER/stage\_1.csv **A\_FIAS\_LOADER** '2022-12-26'

... Адыгея Респ, id\_region: 2

... Start process

... -- gar\_fias\_types\_crt.sql

... -- createTablesGarFias.sql

... -- gar\_fias\_crt\_tbl\_gap.sql

... -- gar\_fias\_crt\_tbl\_black\_list.sql

... -- gar\_fias\_ins\_tbl\_as\_adr\_objects\_type\_black\_list.sql

... -- gar\_fias\_ins\_tbl\_as\_house\_type\_black\_list.sql

... -- CreateTablesGarVersion.sql

... -- CreateTableGarLink

... -- CreateTableGarLink\_1.sql

... -- CreateTablesUnnsiVersion.sql

... -- CreateTypesGarTmp.sql

... -- CreateTypesGarTmp\_2.sql

... -- CreateTableGarTmp.sql

... -- xxx\_adr\_area\_type.sql

... -- xxx\_adr\_street\_type.sql

... -- CreateTableGarTmpHist.sql

... -- CreateTableGarTmpAux.sql

... -- CreateTableGarTmp\_1.sql

... -- CreateTableGarTmp\_2.sql

... -- CreateTableGarTmp\_3.sql

... -- ins\_xxx\_adr\_house\_type.sql

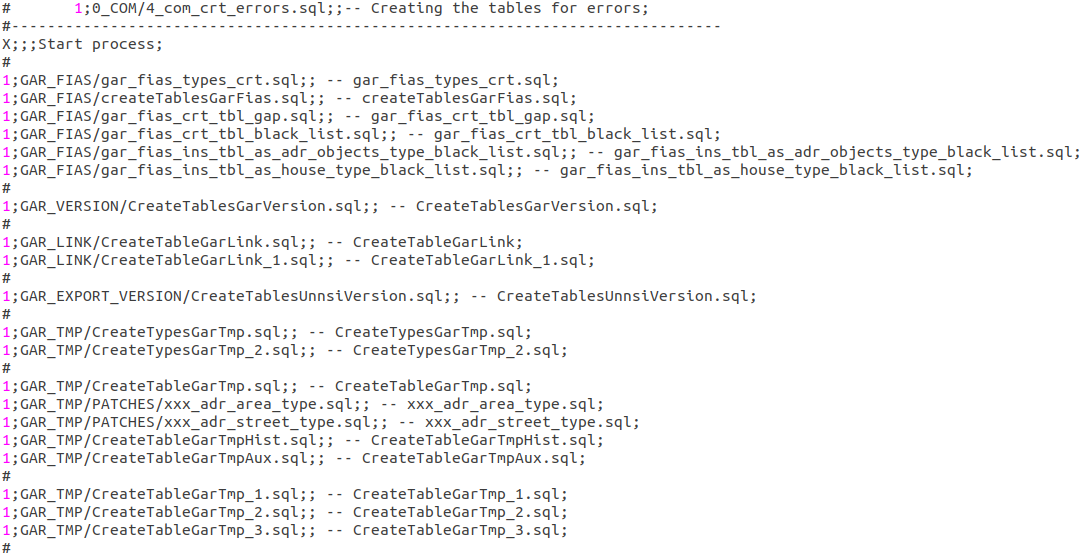
... -- ins\_xxx\_adr\_street\_type.sql

... -- ins\_xxx\_adr\_area\_type.sql

... Stop process

... Бурятия Респ, id\_region: 3

Процесс развёртывания структур хранения (таблицы и типы) запустился. Следует помнить, что практически все CSV-сценарии, работающие с SQL-файлами, используют относительные пути, поэтому ссылка на git-репозиторий, указывается в третьем параметре (выделено красным). Контроллируем успешность завершения процесса. Смотрим файлы "process.log", "process.out", "process.err", создаваемые загрузчиком в процессе развёртывания структуры хранения. Ниже приведён фрагмент файла «stage\_1.csv».

 Рис 18 Фрагмент файла «stage\_1.csv».

Таблицы создались, далее накатываем функциональные пакеты. В проекте используются 6 функциональных пакетов:

| № пп | Имя пакета | Назначение | Схема развёртывания. |
| --- | --- | --- | --- |
| 1 | gar\_misc\_pcg | Функции общего назначения, используются везде. | public |
| 2 | gar\_link\_pcg | Обеспечение коммуникаций с FOREIGN SERVERS. | gar\_link |
| 3 | gar\_tmp\_trans\_pcg | Основной функционал (одна хра-нимка, одно действие). | gar\_tmp\_pcg\_trans |
| 4 | gar\_fias\_pcg\_load | Обеспечивает работу парсеров. | gar\_fias\_pcg\_load |
| 5 | gar\_version\_pcg | Версионирование импорта | gar\_version\_pcg\_support |
| 6 | gar\_version\_pcg | Версионирование экспорта | export\_version |

Рис 19 Перечень функциональных пакетов.

В каждой базе пакета будет выполнено развёртывание пакетов и создание стороннего сервера.

$>load\_mainGar\_2.py hosts\_total\_xx\_0.yaml A\_FIAS\_LOADER/stage\_2.csv A\_FIAS\_LOADER '2022-12-26'

... Адыгея Респ, id\_region: 2

... Start process

... -- gar\_misc\_pcg.sql

... -- gar\_link\_pcg.sql

... -- gar\_tmp\_trans\_pcg.sql

... -- gar\_fias\_pcg\_load.sql

... -- gar\_version\_pcg.sql

... -- gar\_export\_logging\_pcgsql

... -- DROP unnsi\_dev

... -- DROP unnsi\_m2l

... -- DROP unnsi\_prd\_s

... -- DROP unnsi\_prd

... -- DELETE FROM gar\_link.foreign\_servers

... -- gar\_link.f\_server\_crt()

... -- gar\_link.f\_conn\_set()

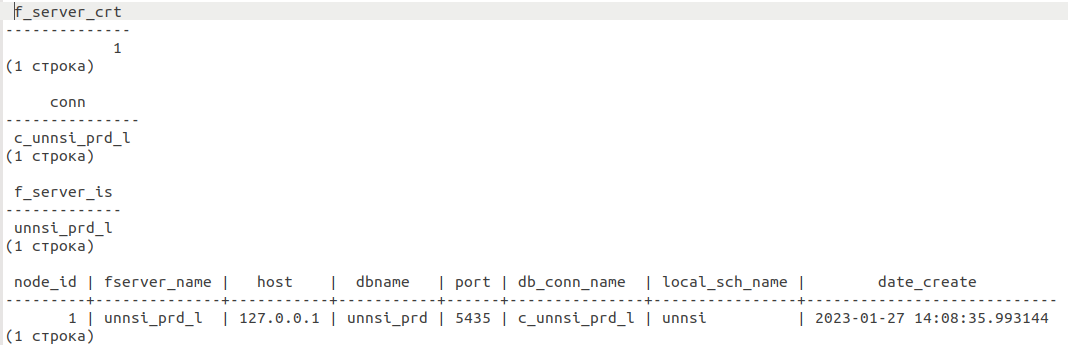
... -- gar\_link.f\_server\_is()

... -- gar\_link.v\_servers\_active

... Stop process

... Бурятия Респ, id\_region: 3

Смотрим "process.out", на каждом хосте, во созданном кластере отработали создание стороннего сервевра и импорт сторонней схемы.

 Рис 20 Фрагмент файла «process.out».

Внешний сервер «unnsi\_prd\_l», ему соответствует именованное db\_link соединение «c\_unnsi\_prd\_l», сторонняя БД «unnsi\_prd», размещённая на другом кластере. Ниже приводится контрольный запрос, сделанный в БД «unsi\_test\_01» к сторонней таблице.

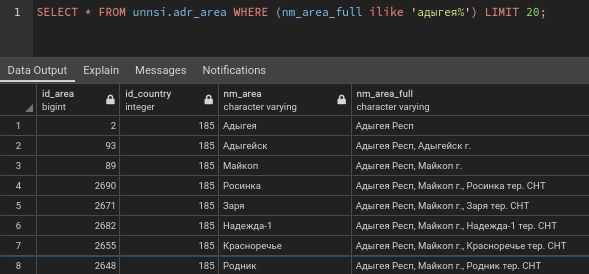


Рис 21 Контрольный запрос к сторонней таблице..

Процесс развёртывания завершён. В заключении, для того, что бы убедить читающего в реальности происходящего, выполним parsing двух адресных регионов «2» и «3», результаты соответственно попадут в базы «unsi\_test\_01» и «unsi\_test\_03». Для этого сделав копию файла «hosts\_parse\_xx\_0.yaml», оставим в нём только первые два региона.

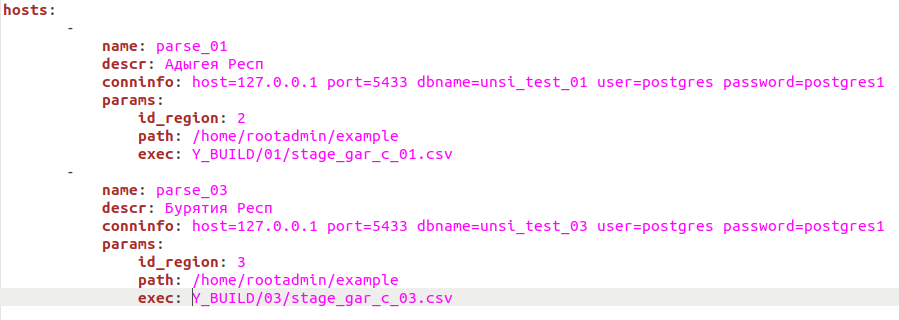


Рис 22 Усечённый файл «hosts\_parse\_xx\_01.yaml».

Запускаем процесс parsing.

$>load\_mainGar\_4.py hosts\_parse\_xx\_01.yaml '2022-12-26'

... Адыгея Респ, id\_region: 2, Execute: /home/rootadmin/example/Y\_BUILD/01/stage\_gar\_c\_01.csv

... Start process

... -- Version

... -- Очистка от данных

... -- Set UNLOGGED

... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/AS\_ADDHOUSE\_TYPES\_20221226\_2626f853-8a92-4d83-ad73-4539ce780eb0.XML

gar\_fias.as\_add\_house\_type: 4

... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/AS\_ADDR\_OBJ\_TYPES\_20221226\_04f2dc05-9918-4f6e-9c4e-2fa06c6cce61.XML

gar\_fias.as\_addr\_obj\_type: 421

... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/AS\_APARTMENT\_TYPES\_20221226\_8dd9eb09-e026-47fe-bc42-2398f92cad5c.XML

gar\_fias.as\_apartment\_type: 13

... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/AS\_HOUSE\_TYPES\_20221226\_c399871c-42a6-49d3-9cae-09766d4d965b.XML

gar\_fias.as\_house\_type: 14

... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/AS\_NORMATIVE\_DOCS\_KINDS\_20221226\_5eac7f6d-be77-40d6-825b-6daac06070a7.XML

gar\_fias.as\_norm\_docs\_kinds: 4

... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/AS\_NORMATIVE\_DOCS\_TYPES\_20221226\_99c46464-14cb-456e-98c1-a32cd4f2833d.XML

gar\_fias.as\_norm\_docs\_types: 25

... /media/rootadmin/Transcend/FIAS\_GAR\_2022\_12\_26/AS\_OBJECT\_LEVELS\_20221226\_6e97d7bb-9815-410e-81dc-37d79b5ab350.XML

gar\_fias.as\_object\_level: 17

Развертывание кластера БД, завершилось успехом.

**6. Типовые варианты использования, часть 2. Микросервисное решение.**

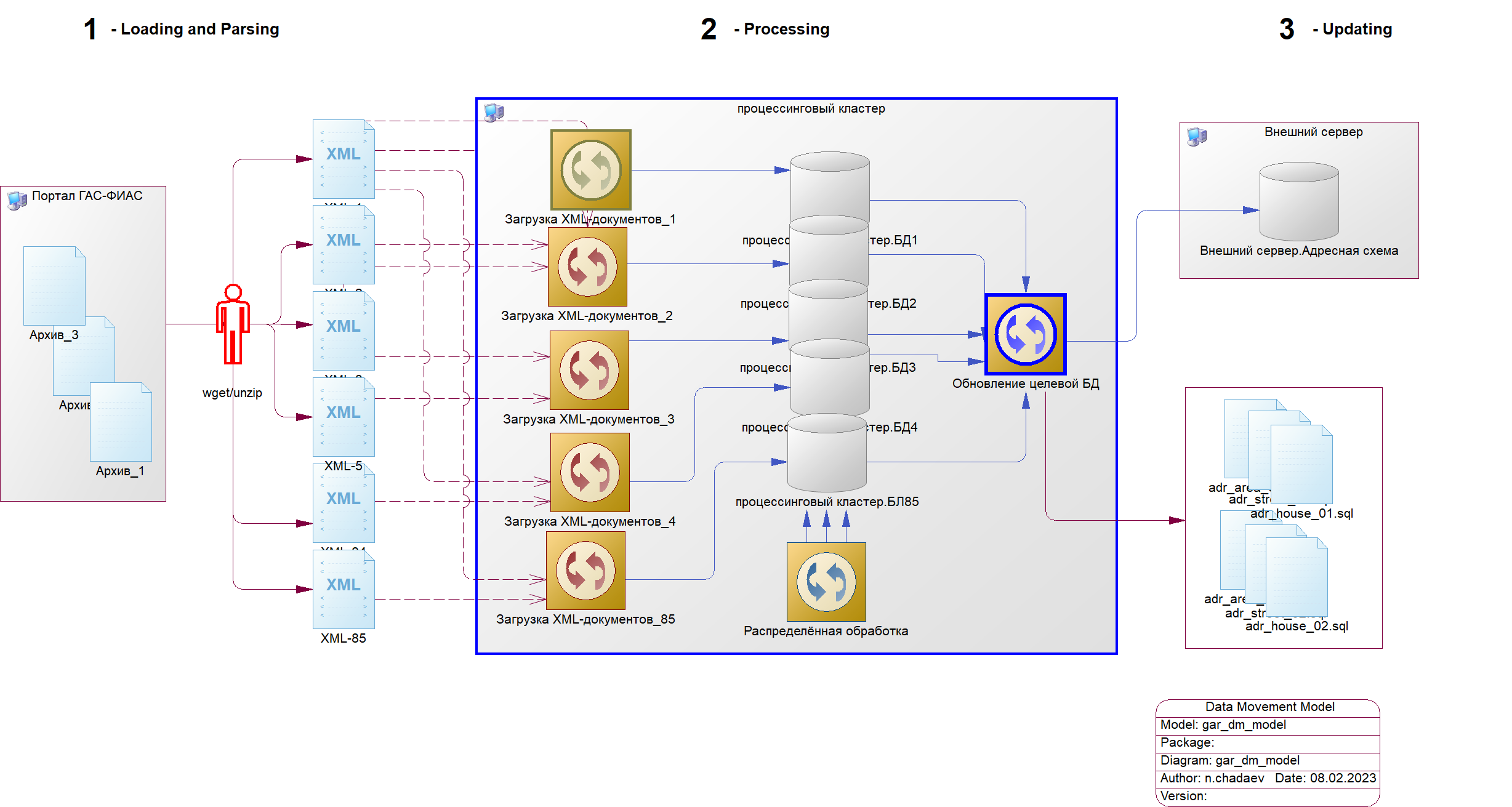
****

Рис 6.1 Общая диаграмма потоков данных.

На рис 6.1 показана основная диаграмма потоков данных в процессинговом кластере. Центром её является процесс «Распределённая обработка». Последовательная реализация этого процесса была описана в предыдущих главах. Именно это решение использовалось в течении первого года регулярной эксплуатации загрузчика адресных данных, хотя данные в адресных архивах и региональных БД является независимыми и допускают параллельную обработку.

**6.1. Требования к параллельной обработке адресных данных:**

1. Процессы должны быть двух типов: выполняющие parsing XML-данных и выполняющую их обработку.

2. Количество процессов обоих типов фиксировано. В том случае, если для процесса нет данных, он переходит из активного состояния в состояние «сна».

3. Однотипные процессы выполняются независимо. Взаимодействие между разнотипными процессами выполняется асинхронно с помощью системы очередей. Для взаимодействия процессов используются две очереди: «Queue\_parse» и «Queue\_proc».

4. В очередях хранятся задания, но не сами данные. Структура очередей должна быть основана на таблице «pgq.event\_template». Функционал расширения pgq не используется. Загрузка адресных данных является мизерной нагрузкой для него.

5. Взаимодействие процессов с очередями выполняется с помощью «pull» и «push» запросов.

Процесс, выполняющий parsing XML-данных выполняет «pull-request» к очереди «Queue\_parse» за получением нового задания. Выполнив его выполняет «push-request» в очередь «Queue\_proc». Процесс, выполняющий обработку данных всегда делает только «pull-request» к очереди «Queue\_proc».

6. Каждый процесс сохраняет текущее выполняемое задание в структуре «In Flight Event» для возможности повторения выполнения задания в случае программной ошибки либо «падения» сервера.

7. Должен быть один процесс, работающий по событию «Завершение обработки»

8. Основа для управления процессами: <https://github.com/polymorphm/pg-perfect-ticker>,

9. Для выполнения заданий должны быть использованы только пакеты и программы «load\_main\_xxx», описанные в предыдущих главах.

Теперь ещё раз вернёмся к процессу «Распределённая обработка» (рис. 6.1). Детализируем его и рассмотрим диаграмму последовательностей каждой его части. Этот процесс состоит из трёх частей: «Инициатор», «Brocker», «PostProcesor». Смотрим на рис. 6.2.

**6.2. Процесс Инициатор.**

Предусловие для успешного выполнения Инициатора:

-- Развёрнутый кластер БД

-- Развёрнутая структура в файловой системе

-- Задания для входной очереди.

Основной сценарий выполнения ( успешный сценарий) :

-- обновлена дата релиза кластера

-- обновлены сценарии для parsing

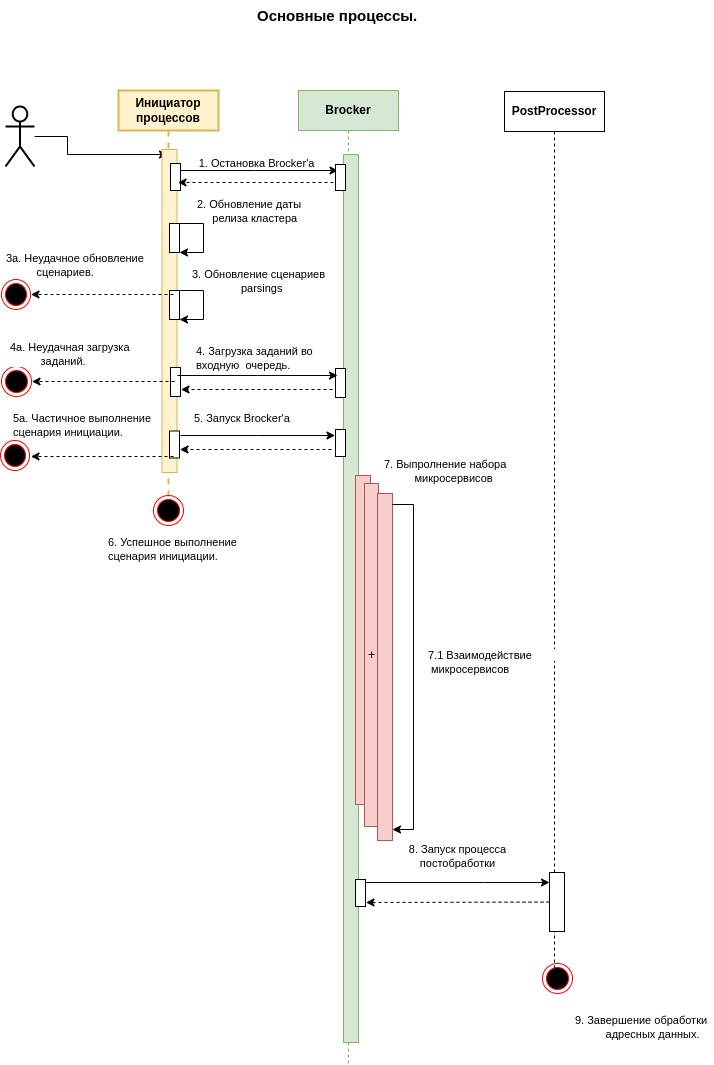
-- загружены задания во входную очередь

-- выполнен перезапуск Брокера очередей (сервис pg-perfect-ticker@exchange)

Альтернативный сценарий выполнения ( частично успешный):

– выполнены все этапы за исключением 5а. В это случае необходимо понять причины неудачного старта Brocker“a. Задания подготовлены, но далее процессы обработки адресных данных выполняться не будут.

Процесс Инициатор является обычным shell-скриптом. Команды остановки и запуска сервиса остались со времени опытной эксплуатации и не являются необходимыми. Микросервисы, управляемые Brocker начнут работу сразу после успешного выполнения пункта 4 «Загрузка заданий во входную очередь.

 Рис 6.2 Детализация процесса распределённой обработки..

**#!/bin/bash**

**# -----------------------------------------------------------------**

**# 2023-01-26 Nick. Подготовка к обработке новой порции данных.**

**# 2023-06-16 Nick. Добавлено вызов функции, формирующей задания**

**# Для очередей.**

**# 2023-07-14 Nick. Добавлен перезапусе Ticker'a**

**# 2023-11-28 Nick. Новый инициатор pg-perfect-ticker**

**# -----------------------------------------------------------------**

**if [ $# -ne 3 ]**

**then**

**echo Использование: new\_data.sh \<Дата-folder\> \<Дата-zip\> \<Дата-версия\>**

**exit 1**

**fi**

**#**

**sudo service pg-perfect-ticker@exchange stop**

**if [ $? -ne 0 ]**

**then**

**echo 1.Ошибка при остановке сервиса.**

**exit 1**

**fi**

**load\_mainCrtScripts.py pattern\_may.csv ~/Y\_BUILD . ../A\_FIAS\_LOADER /home/n.chadaev@abrr.local/7\_DATA $1 $2 $3 True**

**if [ $? -ne 0 ]**

**then**

**echo 2.Ошибка при выполнении обновления CSV-сценариев**

**exit 2**

**fi**

**load\_mainGar.py 127.0.0.1 5432 postgres postgres stage\_z.csv . $3**

**if [ $? -ne 0 ]**

**then**

**rm process.sql**

**echo 3.Ошибка при выполнении обновления описаний БД**

**exit 3**

**fi**

**rm process.sql**

**# 2023-06-16/2023-11-28**

**load\_mainGar\_pt.py 127.0.0.1 5433 db\_exchange postgres hosts\_common\_xxx\_all\_pt.yaml $3**

**if [ $? -ne 0 ]**

**then**

**rm process.sql**

**echo 4.Ошибка при формировании заданий для очередей \(подготовка запуска процессов\).**

**exit 4**

**fi**

**sudo service pg-perfect-ticker@exchange start**

**if [ $? -ne 0 ]**

**then**

**echo 5.Ошибка при запуске сервиса.**

**exit 5**

**fi**

**echo 9.Выполнен процесс**

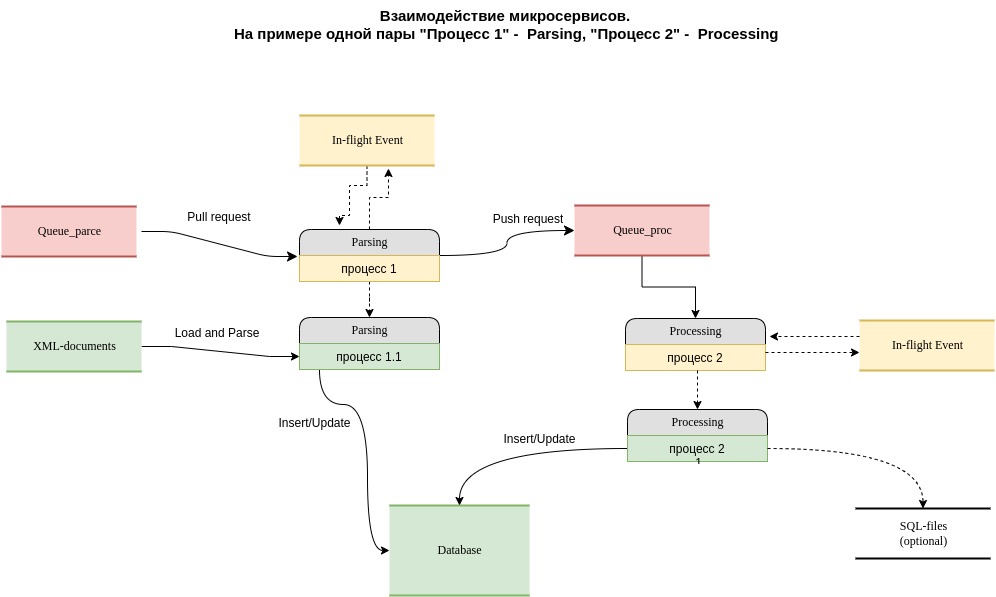
**echo**

Рис 6.3 Текст скрипта-инициатора (new\_data.sh).

**6.3. Процесс Brocker.**

Сама программа, реализующая процесс Brocker написанна на Python c использованием пакетов «asincio», «concurrent».

Рассмотрим DFD-диаграмму, детализирующую пункт 7.1, (взаимодействие между отдельными микросервисами).

 Рис 6.4 Детализация взаимодействия между микросервисам.

Микросервисы могут быть двух типов «P - parsing» и «R — processing». Первый выполняет загрузку XML-файлов, второй их обработку. Каждый состоит из двух частей: управляющей и исполнительной. Управляющая часть обозначена на диаграмме жёлтым цветом, исполнительная — зелёным. Очереди обозначены красным цветом. Управляющая и исполнительная части, вместе компилируются в байт-код в момент старта Brocker'a.

Исполнительная часть это та-же самая load\_mainGar.py, работающая на выбранном хосте и исполняющий CSV-сценарий (эти данными, которые она получает от управляющей части микросервиса).

«In-flight Event» контекст содержит копию исполняемого задания, у в момент успешного завершения микросервиса. Необходим для защиты от программных ошибок и «падений» сервера.

Микросервис типа «P», управляющая часть, . Описание алгоритма работы:

-1) «Pull request» к очереди «Queue parse». При отсутствии заданий переход

в состояниие «сна» на nn-секунд.

-2) При наличии задания в очереди, формирование «In-fligth Event» контекста,

запуск исполняемой части.

-3) При успешном завершении исполняемой части стирание «In-fligth Event» контекста, далее «Push request» в очередь «Queue proc», далее

выполнение шага 1.

Микросервис типа «R», управляющая часть, . Описание алгоритма работы:

-1) «Pull request» к очереди «Queue proc». При отсутствии заданий переход

в состояниие «сна» на nn-секунд.

-2) При наличии задания в очереди, формирование «In-fligth Event» контекста,

запуск исполняемой части.

-3) При успешном завершении исполняемой части стирание «In-fligth Event» контекста, далене выполнение шага 1.

**6.4. PostProcessing.**

Микросервис работает по событию L0 «Данные по всем адресным регионам успешно обработаны». Подсчёт количества обработанных адресных регионов, выполняется в управляющей части. При истинности утверждения L0, начинает работать исполняемая часть, представляющая собой обычный shell-сценарий. Ниже на рисунке 6.5 представлен его текст.

**#!/bin/bash**

**# -----------------------------------------------------------------**

**# 2023-06-30 Nick. Постобработка, финальная стадия**

**# 2023-07-11 Модификация для процессов с правами root**

**# -----------------------------------------------------------------**

**if [ $# -ne 4 ]**

**then**

**echo Использование: post\_proc\_l.sh \<Path\> \<Updates-dir\> \<Owner\> \<Group\>**

**exit 1**

**fi**

**## ----------------------------------------------------- ##**

**## Общие переменные устанавливать только вручную ##**

**## ----------------------------------------------------- ##**

**TD=/tmp # Рабочий каталог, сюда должен попасть архив с обновлениямим**

**WD=$(pwd)**

**PATH\_L=$1**

**DIR\_L=$2**

**OWNER\_L=$3**

**GROUP\_L=$4**

**cd $PATH\_L**

**chown -R $OWNER\_L $DIR\_L**

**if [ $? -ne 0 ]**

**then**

**echo 1.Ошибка при изменении Владельца каталога**

**exit 1**

**fi**

**sudo chgrp -R $GROUP\_L $DIR\_L**

**if [ $? -ne 0 ]**

**then**

**echo 2.Ошибка при изменении Группы каталога**

**exit 2**

**fi**

**tar -cvz $DIR\_L > $DIR\_L.tar.gz**

**if [ $? -ne 0 ]**

**then**

**echo 3.Ошибка при создании архива**

**exit 3**

**fi**

**md5sum $DIR\_L.tar.gz > $DIR\_L.tar.gz.md5**

**if [ $? -ne 0 ]**

**then**

**echo 4.Ошибка при создании контрольной суммы**

**exit 4**

**fi**

**chown -R $OWNER\_L $DIR\_L.tar.gz\***

**if [ $? -ne 0 ]**

**then**

**echo 5.Ошибка при изменении Владельца архива**

**exit 5**

**fi**

**chgrp -R $GROUP\_L $DIR\_L.tar.gz\***

**if [ $? -ne 0 ]**

**then**

**echo 6.Ошибка при изменении Группы архива**

**exit 6**

**fi**

**cp -v $DIR\_L.tar.gz\* $TD/.**

**if [ $? -ne 0 ]**

**then**

**echo 7.Ошибка при копировании файлов**

**exit 7**

**fi**

**chown -R $OWNER\_L $TD/$DIR\_L.tar.gz\***

**if [ $? -ne 0 ]**

**then**

**echo 8.Ошибка при изменении Владельца архива**

**exit 8**

**fi**

**chgrp -R $GROUP\_L $TD/$DIR\_L.tar.gz\***

**if [ $? -ne 0 ]**

**then**

**echo 9.Ошибка при изменении Группы архива**

**exit 9**

**fi**

**cd $WD**

**echo 10.Выполнен процесс**

**echo**

Рис 6.5 Текст сценария, выполняющего post-обработку.

**6.5. Запуск параллельной обработки, необходимые условия.**

Выше было отмечено, что для запуска процесса инициатора необходимо выполнить shell-сценарий «new\_data.sh»

Вызов сценария без параметров.

**$>./new\_data.sh**

**Использование: new\_data.sh <Дата-folder> <Дата-zip> <Дата-версия>**

**$>**

Последовательно: первый параметр — это папка с адресными данными, второй — имя архива, скачанного с сайта ГАР-ФИАС. Третий – дата-версия, дата на которую выполняется обработка. «Дата-версия» описана в разделе «Архитектура».

Пример вызова сценария **new\_data.sh** с параметрами.

**$>./new\_data.sh FIAS\_GAR\_2023\_12\_21 gar\_xml\_12\_22.zip 2023-12-21**

Будучи запущенным, сценарий, в конце своего выполнения, выдаст следующее:

**2023-12-22T12:40:05.356874** ← Текущее время старта Brocker“a

**Redirecting to /bin/systemctl start pg-perfect-ticker@exchange.service**

**9.Выполнен процесс**

Контрольный запрос, выполненный в любом SQL-клиенте даст текущее состояние микросервисов:

**SELECT \* FROM uio.v\_workers\_context;**

**'p3' |'parsing 3'|28795|'2023-12-22 12:40:06'|Y\_BUILD/01/stage\_gar\_c\_01.csv,**

**'p15'|'parsing 15'|28796|'2023-12-22 12:40:06'|Y\_BUILD/03/stage\_gar\_c\_03.csv,**

**'p7' |'parsing 7'|28797|'2023-12-22 12:40:06'|Y\_BUILD/02/stage\_gar\_c\_02.csv,**

**'p4' |'parsing 4'|28798|'2023-12-22 12:40:06'|Y\_BUILD/04/stage\_gar\_c\_04.csv,**

**'p17'|'parsing 17'|28799|'2023-12-22 12:40:06'|Y\_BUILD/05/stage\_gar\_c\_05.csv,**

**'p11'|'parsing 11'|28800|'2023-12-22 12:40:06'|Y\_BUILD/06/stage\_gar\_c\_06.csv,**

**'p18'|'parsing 18'|28801|'2023-12-22 12:40:06'|Y\_BUILD/07/stage\_gar\_c\_07.csv,**

**'p10'|'parsing 10'|28802|'2023-12-22 12:40:06'|Y\_BUILD/08/stage\_gar\_c\_08.csv,**

**'p13'|'parsing 13'|28803|'2023-12-22 12:40:06'|Y\_BUILD/09/stage\_gar\_c\_09.csv,**

**'p0' |'parsing 0'|28804|'2023-12-22 12:40:06'|Y\_BUILD/10/stage\_gar\_c\_10.csv,**

**'p1' |'parsing 1'|28805|'2023-12-22 12:40:06'|Y\_BUILD/11/stage\_gar\_c\_11.csv,**

**'p8' |'parsing 8'|28806|'2023-12-22 12:40:06'|Y\_BUILD/12/stage\_gar\_c\_12.csv,**

**'p14'|'parsing 14'|28807|'2023-12-22 12:40:06'|Y\_BUILD/13/stage\_gar\_c\_13.csv,**

**'p16'|'parsing 16'|28808|'2023-12-22 12:40:06'|Y\_BUILD/14/stage\_gar\_c\_14.csv,**

**'p6' |'parsing 6'|28809|'2023-12-22 12:40:06'|Y\_BUILD/15/stage\_gar\_c\_15.csv,**

**'p19'|'parsing 19'|28810|'2023-12-22 12:40:06'|Y\_BUILD/18/stage\_gar\_c\_18.csv,**

**'p12'|'parsing 12'|28811|'2023-12-22 12:40:06'|Y\_BUILD/17/stage\_gar\_c\_17.csv,**

**'p9' |'parsing 9'|28812|'2023-12-22 12:40:06'|Y\_BUILD/19/stage\_gar\_c\_19.csv,**

**'p2' |'parsing 2'|28813|'2023-12-22 12:40:06'|Y\_BUILD/16/stage\_gar\_c\_16.csv,**

**'p5' |'parsing 5'|28814|'2023-12-22 12:40:06'|Y\_BUILD/20/stage\_gar\_c\_20.csv,**

Но вернёмся к сценарию-инициатору, рассмотрим его текст и остановливаемся на пункте 4 сценария, в котором выполняется программа «load\_mainGar\_pt.py», именно она формирует задания на обработку на основе YAML-файла «hosts\_common\_xxx\_all\_pt.yaml» и текущей даты-версии

**load\_mainGar\_pt.py 127.0.0.1 5433 db\_exchange postgres hosts\_common\_xxx\_all\_pt.yaml 2023-12-21**

**6.6. Описание YAML-файла «hosts\_common\_xxx\_all\_pt.yaml»**

Это ещё более простая структура, чем описанные в предыдущих главах, состоящая из общей части и массива адресов хостов с региональными БД.

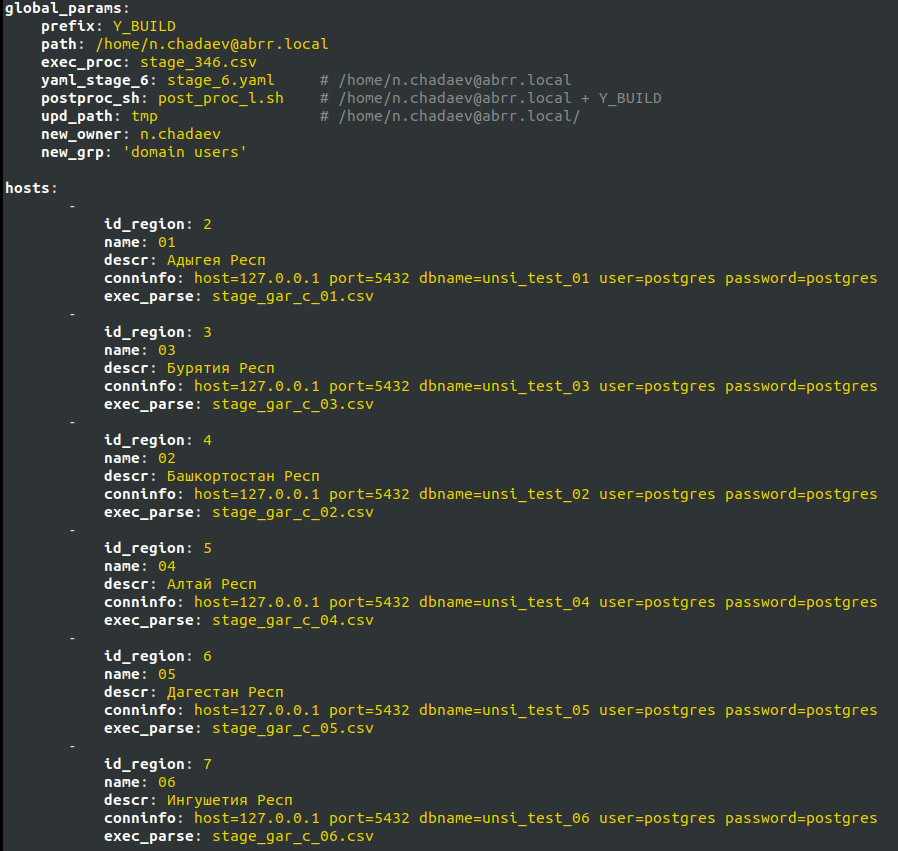
****

Рис 6.6 Фрагмент YAML-файла «hosts\_all\_common\_xxx\_pt.yaml»

Глобальный блок содержит всё, что используется на микросервисами типа «R» (обработка), в адресном массиве «hosts» содержатся наименования сценариев, выполняющих parsing, наименования регионов и строка соединения с региональной БД. Все эти структуры были описаны выше, в предыдущих главах и используются только в исполнительном блоке микросервиса.

**7. Контроль за функционированием.**

В данном разделе будут описаны контроль за функционированием, обработка ошибочных ситуаций, применительно к микросервисной версии загрузчика адресных данных.

Что будет освещено далее :

- Текущая конфигурация микросервисов.

- Текущая конфигурация LOG-файлов.

- Функция и представление для контроля за микросервисами.

- Текущая конфигурация рабочей файловой системы, связь региональных микросервисных протоколов (xxx\_process.log, xxx\_process.err, xxx\_process.out) с LOG-файлами Брокера «pg-perfect-ticker»

**7.1. Текущая конфигурация.**

Конфигурационный файл задач, описывающий перечень и свойства микросервисов, находится, как правило, в каталоге «/opt». В текущей реализации это «/opt/pg-perfect-ticker/exchange-tasks.cfg», где «pg-perfect-ticker» - имя брокера, управляющего функционированием набора микросервисов. Ниже приведены два фрагмента конфигурационного файла (список и свойства задач):

**task\_list =**

**set\_tick**

**p0**

**p1**

**p2**

**p3**

**p4**

**p5**

**p6**

**p7**

**p8**

**p9**

**. . . . . . . . . . . .**

**r0**

**r1**

**r2**

**r3**

**r4**

**r5**

**r6**

**r7**

**r8**

**r9**

**. . . . . . . . . . . .**

**l0**

**fg\_thr\_pool\_t.max\_workers = 2**

**bg\_thr\_pool\_p.max\_workers = 20**

**bg\_thr\_pool\_r.max\_workers = 10**

В списке приведены имена, далее, приведён фрагмент с их параметрами:

**set\_tick.disabled = False**

**set\_tick.sql = SELECT pgq.ticker();**

**set\_tick.timer = 1**

**set\_tick.thread\_pool = fg\_thr\_pool\_t**

**set\_tick.db\_con = exchange\_db\_con**

**p0.disabled = False**

**p0.script = exchange-scripts/p0.py**

**p0.timer = 44.79**

**p0.thread\_pool = bg\_thr\_pool\_p**

**p0.db\_con = exchange\_db\_con**

**p1.disabled = False**

**p1.script = exchange-scripts/p1.py**

**p1.timer = 16.79**

**p1.thread\_pool = bg\_thr\_pool\_p**

**p1.db\_con = exchange\_db\_con**

Задачи и их параметры относятся к управляющей части микросервиса (смотри диаграмму потоков данных, приведённую выше). Важно помнить, что микросервис — это сочетание управляющей и исполнительной частей.

**7.2. Функция и представления для контроля за микросервисами.**

Для отображения текущего состояния микросервисов используется представление «uio.v\_workers\_context». Ниже частично приведён результат его работы



Рис 6.6 Результат работы представления «uio.v\_workers\_context» (Часть 1).

Десять процессов выполняют parsing XML-файлов, один — обработку адресных данных. Идентификатор «ev\_type» содержит номер адресного региона (по классификации ГАР-ФИАС) и ключевые слова «parse\_», «proc\_». Их сочетание означает, что микросервис выполнит parsing, и сформирует задание в очередь на обработку. В качестве примера рассмотрим микросервис «p17». Он уже выполнил «pull-запрос» к очереди заданий на парсинг, получил заданиен, выполняет парсинг (ev\_type = «parse\_91»), по его завершению процесс выполнит «push-запрос» в очередь заданий на обработку (ev\_type = «proc\_91»). Это задание из очереди «выхватит» любой другой свободный микросервис типа «r».

 Рис 6.7 Результат работы представления «uio.v\_workers\_context» (Часть 2).

На рисунке 6.7 представлена следующая часть данных, сформированных представлением «uio.v\_workers\_context», столбец «ev\_data», содержащий строку подключения к региональной БД, далее данные разнятся. Для микросервисов типа «p» после строки подключения следуют два задания на парсинг и на обработку, для микросервисов типа «r» только задание на обработку. Продолжим рассмотрение выделенной строки: используется региональная база «unsi\_test\_91», сценарий на парсинг «stage\_gar\_c\_91.csv» находится в папке «~/Y\_BUILD/91», атрибут «Дата-версия» имеет значение «2023-12-25», код адресного региона для ЕС НСИ равен 84.

Представление «uio.v\_event\_waits» - Отображает список заданий в различных состояниях. Ниже приведён список обработанных заданий (proc\_descr = «terminated»), с идентификатором региона (ЕС НСИ), наименованием региона, последней выполненной командой и идентификатором микросервиса завершившего обработку (столбец выделен жёлтым цветом).

 Рис 6.8 Результат работы представления «uio.v\_event\_waits».

**7.3. Текущая конфигурация LOG-файлов pg-perfect\_ticker (Брокер сообщений).**

На рисунке показано содержимое каталога с log-файлами брокера сообщений. Таких файлов два «exchange.log» и «exchange.warn.log». В первом из них содержатся сообщения о старте, ходе выполнения и завершении работы микросервисов. Во втором из них – сообщения об ошибках, регистрируемых брокером.

**>l /var/log/pg-perfect-ticker/**

**итого 5212**

**-rw-r--r--. 1 root root 162272 дек 27 10:53 exchange.log**

**drwxr-xr-x. 2 root root 161 дек 27 10:20 .**

**-rw-r--r--. 1 root root 999955 дек 27 10:20 exchange.log.1**

**-rw-r--r--. 1 root root 999964 дек 26 14:01 exchange.log.2**

**-rw-r--r--. 1 root root 999936 дек 26 12:52 exchange.log.3**

**-rw-r--r--. 1 root root 999906 дек 26 11:43 exchange.log.4**

**-rw-r--r--. 1 root root 999979 дек 26 10:34 exchange.log.5**

**drwxr-xr-x. 11 root root 4096 дек 24 03:31 ..**

**-rw-r--r--. 1 root root 116576 дек 12 12:24 exchange.warn.log**

Если дата обновления «exchange.warn.log» сравнялась с датой обновления «exchange.log» и размер первого файла продолжает расти, это свидетельствует о наличии ошибок в коде, либо в данных. Необходимо останавливать сервис и искать причину.

Фрагмент файла «exchange.log» в отсутствии нагрузки:

**2023-12-27 20:25:04,671 INFO pg-perfect-ticker ticker task ('r0', 'bg\_thr\_pool\_r', 'exchange\_db\_con'): execute**

**2023-12-27 20:25:04,675 INFO pg-perfect-ticker ticker task ('r0', 'bg\_thr\_pool\_r', 'exchange\_db\_con'): sleep 239.9942608140409**

**2023-12-27 20:25:05,237 INFO pg-perfect-ticker ticker task ('p4', 'bg\_thr\_pool\_p', 'exchange\_db\_con'): execute**

**2023-12-27 20:25:05,241 INFO pg-perfect-ticker ticker task ('p4', 'bg\_thr\_pool\_p', 'exchange\_db\_con'): sleep 43.99324584007263**

**2023-12-27 20:25:05,314 INFO pg-perfect-ticker ticker task ('p15', 'bg\_thr\_pool\_p', 'exchange\_db\_con'): execute**

**2023-12-27 20:25:05,318 INFO pg-perfect-ticker ticker task ('p15', 'bg\_thr\_pool\_p', 'exchange\_db\_con'): sleep 9.993800964206457**

**2023-12-27 20:25:05,570 INFO pg-perfect-ticker ticker task ('p2', 'bg\_thr\_pool\_p', 'exchange\_db\_con'): execute**

**2023-12-27 20:25:05,574 INFO pg-perfect-ticker ticker task ('p2', 'bg\_thr\_pool\_p', 'exchange\_db\_con'): sleep 16.994985926896334**

**2023-12-27 20:25:05,860 INFO pg-perfect-ticker ticker task ('p8', 'bg\_thr\_pool\_p', 'exchange\_db\_con'): execute**

**2023-12-27 20:25:05,863 INFO pg-perfect-ticker ticker task ('p8', 'bg\_thr\_pool\_p', 'exchange\_db\_con'): sleep 12.994904313236475**

**2023-12-27 20:25:07,399 INFO pg-perfect-ticker ticker task ('p7', 'bg\_thr\_pool\_p', 'exchange\_db\_con'): execute**

**2023-12-27 20:25:07,401 INFO pg-perfect-ticker ticker task ('p7', 'bg\_thr\_pool\_p', 'exchange\_db\_con'): sleep 48.99454786628485**

**2023-12-27 20:25:07,763 INFO pg-perfect-ticker ticker task ('p5', 'bg\_thr\_pool\_p', 'exchange\_db\_con'): execute**

**2023-12-27 20:25:07,767 INFO pg-perfect-ticker ticker task ('p5', 'bg\_thr\_pool\_p', 'exchange\_db\_con'): sleep 10.99504916742444**

**2023-12-27 20:25:09,001 INFO pg-perfect-ticker ticker task ('p3', 'bg\_thr\_pool\_p', 'exchange\_db\_con'): execute**

**2023-12-27 20:25:09,003 INFO pg-perfect-ticker ticker task ('p3', 'bg\_thr\_pool\_p', 'exchange\_db\_con'): sleep 28.994521103799343**

Фрагмент файла «exchange.warn.log», выполняются задания (есть нагрузка) присутствует ошибка в обрабатывающем модуле «load\_mainGar.py»:

**2023-12-12 12:08:17,596 WARNING pg-perfect-ticker ticker task ('r3', 'bg\_thr\_pool\_r', 'exchange\_db\_con'): error <class 'SystemExit'>: 256**

**2023-12-12 12:08:18,041 WARNING pg-perfect-ticker ticker task ('r1', 'bg\_thr\_pool\_r', 'exchange\_db\_con'): error <class 'SystemExit'>: 256**

**2023-12-12 12:09:57,520 WARNING pg-perfect-ticker ticker task ('r3', 'bg\_thr\_pool\_r', 'exchange\_db\_con'): error <class 'SystemExit'>: 256**

**2023-12-12 12:10:18,153 WARNING pg-perfect-ticker ticker task ('r1', 'bg\_thr\_pool\_r', 'exchange\_db\_con'): error <class 'SystemExit'>: 256**

**2023-12-12 12:11:38,534 WARNING pg-perfect-ticker ticker task ('r3', 'bg\_thr\_pool\_r', 'exchange\_db\_con'): error <class 'SystemExit'>: 256**

**2023-12-12 12:12:19,154 WARNING pg-perfect-ticker ticker task ('r1', 'bg\_thr\_pool\_r', 'exchange\_db\_con'): error <class 'SystemExit'>: 256**

**2023-12-12 12:12:33,596 WARNING pg-perfect-ticker ticker task ('r2', 'bg\_thr\_pool\_r', 'exchange\_db\_con'): error <class 'SystemExit'>: 256**

**2023-12-12 12:13:17,236 WARNING pg-perfect-ticker ticker task ('r3', 'bg\_thr\_pool\_r', 'exchange\_db\_con'): error <class 'SystemExit'>: 256**

**2023-12-12 12:14:17,269 WARNING pg-perfect-ticker ticker task ('r1', 'bg\_thr\_pool\_r', 'exchange\_db\_con'): error <class 'SystemExit'>: 256**

**2023-12-12 12:14:56,966 WARNING pg-perfect-ticker ticker task ('r3', 'bg\_thr\_pool\_r', 'exchange\_db\_con'): error <class 'SystemExit'>: 256**

**2023-12-12 12:16:17,124 WARNING pg-perfect-ticker ticker task ('r1', 'bg\_thr\_pool\_r', 'exchange\_db\_con'): error <class 'SystemExit'>: 256**

**2023-12-12 12:16:36,656 WARNING pg-perfect-ticker ticker task ('r3', 'bg\_thr\_pool\_r', 'exchange\_db\_con'): error <class 'SystemExit'>: 256**

**2023-12-12 12:17:02,635 WARNING pg-perfect-ticker ticker task ('r2', 'bg\_thr\_pool\_r', 'exchange\_db\_con'): error <class 'SystemExit'>: 256**

SystemExit: 256 — код ошибки, выданный обрабатывающим модулем load\_mainGar.py. Отсутствовал sql-файл «A\_FIAS\_LOADER/GAR\_TMP\_PCG\_TRANS/DO/adr\_area\_post\_proc\_90.sq» для большой команды типа 4 «stage\_41r.yaml».

**7.4. Локализация ошибок, связь log-файлы брокера и** **log-файлов, исполнительных модулей.**

В качестве примера будет рассмотрен фрагменты LOGs отладочного instance обработчика адресных данных. Конфигурационный файл микросервисов «/opt/pg-perfect-ticker/exchange-tasks.cfg» имеет структуру, идентичную, описанной выше. Разница заключается в yaml-файле «hosts\_common\_xxx\_all\_pt\_test.yaml», описывающем задания для обработчика адресных данных, таких заданий гораздо меньше, чем в действительности и они содержат ошибки. Ниже приведён его текст:

**global\_params:**

**prefix: Y\_BUILD**

**path: /home/rootadmin/abr\_upload**

**exec\_proc: stage\_346.csv**

**yaml\_stage\_6: stage\_6.yaml # /home/n.chadaev@abrr.local**

**postproc\_sh: post\_proc\_l.sh # /home/n.chadaev@abrr.local + Y\_BUILD**

**upd\_path: tmp # /home/n.chadaev@abrr.local/**

**new\_owner: rootadmin**

**new\_grp: rootadmin**

**hosts:**

**-**

**id\_region: 5**

**name: 04**

**descr: Алтай Респ**

**conninfo: host=127.0.0.1 port=5434 dbname=unsi\_test\_04 user=postgres password=postgres**

**exec\_parse: stage\_gar\_c\_04.csv**

**-**

**id\_region: 6**

**name: 05**

**descr: Дагестан Респ**

**conninfo: host=127.0.0.1 port=5434 dbname=unsi\_test\_05 user=postgres password=postgres**

**exec\_parse: stage\_gar\_c\_05.csv**

**-**

**id\_region: 7**

**name: 06**

**descr: Ингушетия Респ**

**conninfo: host=127.0.0.1 port=5434 dbname=unsi\_test\_06 user=postgres password=postgres**

**exec\_parse: stage\_gar\_c\_06.csv**

Используются три региональные БД «unsi\_test\_04», «unsi\_test\_05», «unsi\_test\_06». В каждой из которых допущена ошибка в определении FDW (Foreign Data Wrapper). Кроме того при первом запуске процесса была указана неправильное значение атрибута «Дата-версия»: 2023-10-29 вместо 2023-10-26. В рабочем сегменте файловой представляют интерес следующие три каталога «04», «05», «06».

**$>pwd**

**/home/rootadmin/abr\_upload/Y\_BUILD**

**$>**

**$>l 04 05 06**

**04:**

**-rw-rw-r-- 1 rootadmin rootadmin 822 дек 22 18:07 stage\_gar\_c\_04.csv**

**-rw-rw-r-- 1 rootadmin rootadmin 74 июл 20 13:33 commentDb\_04.sql**

**-rw-rw-r-- 1 rootadmin rootadmin 414 июл 20 13:33 createDb\_04.sql**

**-rw-rw-r-- 1 rootadmin rootadmin 360 июл 20 13:33 stage\_0\_04.csv**

**06:**

**-rw-rw-r-- 1 rootadmin rootadmin 831 дек 22 18:07 stage\_gar\_c\_06.csv**

**-rw-rw-r-- 1 rootadmin rootadmin 82 июл 20 13:33 commentDb\_06.sql**

**-rw-rw-r-- 1 rootadmin rootadmin 422 июл 20 13:33 createDb\_06.sql**

**-rw-rw-r-- 1 rootadmin rootadmin 360 июл 20 13:33 stage\_0\_06.csv**

**05:**

**-rw-rw-r-- 1 rootadmin rootadmin 829 дек 22 18:07 stage\_gar\_c\_05.csv**

**-rw-rw-r-- 1 rootadmin rootadmin 80 июл 20 13:33 commentDb\_05.sql**

**-rw-rw-r-- 1 rootadmin rootadmin 420 июл 20 13:33 createDb\_05.sql**

**-rw-rw-r-- 1 rootadmin rootadmin 360 июл 20 13:33 stage\_0\_05.csv**

Красным выделены скрипты, участвующие в процессе парсинга. Запускаем процесс обработки данных и смотрим на состояние log-файлов в каталоге «/var/log/pg-perfect-ticker»:

**$>l /var/log/pg-perfect-ticker/**

**итого 10360**

**drwxrwxr-x 17 root syslog 4096 дек 28 00:06 ..**

**-rw-r--r-- 1 root root 488003 дек 26 17:45 exchange.log**

**-rw-r--r-- 1 root root 29434 дек 26 17:44 exchange.warn.log**

**drwxr-xr-x 2 rootadmin rootadmin 4096 дек 26 17:24 .**

**-rw-r--r-- 1 root root 999959 дек 26 17:24 exchange.log.1**

**-rw-r--r-- 1 root root 999297 дек 26 16:57 exchange.warn.1**

Файл «exchange.warn.log» растёт с такой-же скоростью, как и «exchange.log». Это характерный признак ошибки. Рассмотрим фрагмент его содержимого (микросервис «p19»).

**2023-12-26 16:58:10,197 WARNING pg-perfect-ticker ticker task ('p19', 'bg\_thr\_pool\_p', 'exchange\_db\_con'): error <class 'AttributeError'>: module 'psycopg2' has no attribute 'NotNullViolation'**

**Traceback (most recent call last):**

**File "/opt/pg-perfect-ticker/load\_mainGar.py", line 268, in save\_file\_info**

**cur.execute (CALL\_PROC\_F, (self.version, r\_nmb, p\_file\_name))**

**psycopg2.errors.NotNullViolation: ОШИБКА: значение NULL в столбце "id\_garfias\_version" отношения "garfias\_files\_by\_region" нарушает ограничение NOT NULL**

**DETAIL: Ошибочная строка содержит (1945, null, null, /media/rootadmin/Transcend/FIAS\_GAR\_2023\_10\_26/AS\_ADDHOUSE\_TYPES...).**

**CONTEXT: SQL-оператор: "INSERT INTO gar\_version.garfias\_files\_by\_region AS v**

**(**

**id\_garfias\_version**

**,id\_region**

**,file\_path**

**)**

**VALUES (**

**\_id\_garfias\_version**

**,i\_id\_region**

**,i\_file\_path**

**)**

**ON CONFLICT (file\_path) DO**

**UPDATE SET**

**id\_garfias\_version = excluded.id\_garfias\_version**

**,id\_region = excluded.id\_region**

**WHERE (V.file\_path = excluded.file\_path)**

**RETURNING v.id\_file\_version"**

**функция PL/pgSQL save\_gar\_files\_by\_region(date,integer,text), строка 16, оператор SQL-оператор**

**During handling of the above exception, another exception occurred:**

**Traceback (most recent call last):**

**File "/opt/pg-perfect-ticker/lib\_pg\_perfect\_ticker/pg\_perfect\_ticker.py", line 234, in blocking\_ticker\_task\_process**

**'awake\_task': awake\_task\_func,**

**File "/opt/pg-perfect-ticker/exchange-scripts/p19.py", line 132, in <module>**

**p\_first\_message = first\_mess\_parse)**

**File "/opt/pg-perfect-ticker/load\_mainGar.py", line 630, in to\_do**

**rc1 = self.save\_file\_info (p\_host\_ip, p\_port, l\_db\_name, p\_user\_name, file\_name)**

**File "/opt/pg-perfect-ticker/load\_mainGar.py", line 283, in save\_file\_info**

**except psycopg2.NotNullViolation as e2:**

**AttributeError: module 'psycopg2' has no attribute 'NotNullViolation'**

Ошибка была вызвана недоработкой модуля «load\_mainGar.py», в части обрабатывающей значения атрибута «Дата-версия». Далее рассмотрим содержимое каталога **«/home/rootadmin/abr\_upload/Y\_BUILD/06»**. Исполняемая часть микросервиса «p19» наплодила целую кучу собственных log-файлов, рассмотрим три из них.

**$>l p19\_process\_20231226\_165737.out p19\_process\_20231226\_165737.log p19\_process\_20231226\_165737.err**

**-rw-r--r-- 1 root root 620 дек 26 16:58 p19\_process\_20231226\_165737.log**

**-rw-r--r-- 1 root root 114 дек 26 16:58 p19\_process\_20231226\_165737.out**

**-rw-r--r-- 1 root root 0 дек 26 16:57 p19\_process\_20231226\_165737.err**

Файл типа «.err» пустой, он не успел нормально закрыться. Файл типа «log» содержит записи об исполненном действии, файл типа «out» - о результатах выполнения.

Содержимое файла «p19\_process\_20231226\_165737.log»:

**2023-12-26 16:57:37 home/rootadmin/abr\_upload**

**127.0.0.1 5434 unsi\_test\_06 postgres /home/rootadmin/abr\_upload/Y\_BUILD/06/stage\_gar\_c\_06.csv 2023-10-29**

**--------------------------------------------------------**

**[16:57:37] ... Ингушетия Респ, id\_region: 7, Execute: /home/rootadmin/abr\_upload/Y\_BUILD/06/stage\_gar\_c\_06.csv, process\_id:16652**

**[16:57:37] ... Start process**

**[16:57:37] ... -- Version**

**[16:57:37] ... -- Очистка от данных**

**[16:57:42] ... -- Set UNLOGGED**

**[16:57:42] ... /media/rootadmin/Transcend/FIAS\_GAR\_2023\_10\_26/AS\_ADDHOUSE\_TYPES\_20231026\_1b8ca463-4b89-4a94-96e9-2567cd8bae55.XML**

Ошибка случилась во время парсинга первого файла. Что конкретно произошло, смотрим файл «p19\_process\_20231226\_165737.out»:

**save\_gar\_version**

**------------------**

**71**

**(1 строка)**

**CALL**

**CALL**

**gar\_fias.as\_add\_house\_type: 4**

Ошибка возникла в процессе постобработки результатов парсинга файла «AS\_ADDHOUSE\_TYPES».

Рассмотрим второй пример, ошибка возникшая во время выполнения микросервиса типа «R». Ещё раз рассмотрим файл «exchange.warn.log».

**2023-12-26 17:20:02,291 WARNING pg-perfect-ticker ticker task ('r5', 'bg\_thr\_pool\_r', 'exchange\_db\_con'): error <class 'SystemExit'>: 256**

**2023-12-26 17:26:04,006 WARNING pg-perfect-ticker ticker task ('r5', 'bg\_thr\_pool\_r', 'exchange\_db\_con'): error <class 'SystemExit'>: 256**

**2023-12-26 17:26:23,840 WARNING pg-perfect-ticker ticker task ('r3', 'bg\_thr\_pool\_r', 'exchange\_db\_con'): error <class 'SystemExit'>: 256**

**2023-12-26 17:28:36,196 WARNING pg-perfect-ticker ticker task ('r3', 'bg\_thr\_pool\_r', 'exchange\_db\_con'): error <class 'SystemExit'>: 256**

**2023-12-26 17:31:01,402 WARNING pg-perfect-ticker ticker task ('r3', 'bg\_thr\_pool\_r', 'exchange\_db\_con'): error <class 'SystemExit'>: 256**

**2023-12-26 17:32:13,998 WARNING pg-perfect-ticker ticker task ('r5', 'bg\_thr\_pool\_r', 'exchange\_db\_con'): error <class 'SystemExit'>: 256**

**2023-12-26 17:33:16,318 WARNING pg-perfect-ticker ticker task ('r3', 'bg\_thr\_pool\_r', 'exchange\_db\_con'): error <class 'SystemExit'>: 256**

**2023-12-26 17:35:35,750 WARNING pg-perfect-ticker ticker task ('r3', 'bg\_thr\_pool\_r', 'exchange\_db\_con'): error <class 'SystemExit'>: 256**

**2023-12-26 17:37:53,312 WARNING pg-perfect-ticker ticker task ('r3', 'bg\_thr\_pool\_r', 'exchange\_db\_con'): error <class 'SystemExit'>: 256**

**2023-12-26 17:38:16,665 WARNING pg-perfect-ticker ticker task ('r5', 'bg\_thr\_pool\_r', 'exchange\_db\_con'): error <class 'SystemExit'>: 256**

**2023-12-26 17:40:12,883 WARNING pg-perfect-ticker ticker task ('r3', 'bg\_thr\_pool\_r', 'exchange\_db\_con'): error <class 'SystemExit'>: 256**

**2023-12-26 17:42:32,894 WARNING pg-perfect-ticker ticker task ('r3', 'bg\_thr\_pool\_r', 'exchange\_db\_con'): error <class 'SystemExit'>: 256**

**2023-12-26 17:44:23,903 WARNING pg-perfect-ticker ticker task ('r5', 'bg\_thr\_pool\_r', 'exchange\_db\_con'): error <class 'SystemExit'>: 256**

**2023-12-26 17:44:51,145 WARNING pg-perfect-ticker ticker task ('r3', 'bg\_thr\_pool\_r', 'exchange\_db\_con'): error <class 'SystemExit'>: 256**

Во время исполнения микросервисов «r3» «r5» в их прикладных частях возникает ошибка, код возврата 256. Микросервис «r5» работает с БД «unsi\_test\_06»

**$>pwd**

**/home/rootadmin/abr\_upload/Y\_BUILD/06**

**$>**

**$>l r5\_process\_20231226\_174419.log r5\_process\_20231226\_174419.err r5\_process\_20231226\_174419.out**

**-rw-r--r-- 1 root root 2128 дек 26 17:44 r5\_process\_20231226\_174419.log**

**-rw-r--r-- 1 root root 142 дек 26 17:44 r5\_process\_20231226\_174419.err**

**-rw-r--r-- 1 root root 1589 дек 26 17:44 r5\_process\_20231226\_174419.out**

Содержимое файла типа «log»

**2023-12-26 17:44:19 home/rootadmin/abr\_upload**

**127.0.0.1 5434 unsi\_test\_06 postgres /home/rootadmin/abr\_upload/Y\_BUILD/stage\_346.csv 2023-10-26**

**--------------------------------------------------------**

**[17:44:19] ... Ингушетия Респ, id\_region: 7, Execute: /home/rootadmin/abr\_upload/Y\_BUILD/stage\_346.csv, process\_id:18035**

**[17:44:19] ... Start process, Обработка**

**[17:44:19] ... -- gar\_link.f\_server\_is**

**[17:44:19] ... -- stage\_31r (Предобработка данных)**

**[17:44:19] ... -- Создание рабочих индексов в схеме gar\_fias. False-удаление, True-удаление,создание.**

**[17:44:22] ... -- Заполнение таблицы с активными предшественниками**

**[17:44:22] ... -- Обработка таблицы с активными предшественниками, переподчинение и деактивация**

**[17:44:23] ... -- Установка признака LOGGED/UNLOGGED у таблиц в схеме gar\_tmp**

**[17:44:23] ... -- Очистка данных во временной схеме.**

**[17:44:23] ... -- Загрузка данных из таблицы ADR\_AREA\_TYPE**

**[17:44:23] ... -- Загрузка данных из таблицы ADR\_STREET\_TYPE**

**[17:44:23] ... -- Загрузка данных из таблицы ADR\_HOUSE\_TYPE**

**[17:44:23] ... -- Загрузка регионального фрагмента из таблицы ADR\_AREA**

**[17:44:23] ... [17:44:23]: Error, rc = 256**

**[17:44:23] ... psql -h 127.0.0.1 -p 5434 -c "CALL gar\_tmp\_pcg\_trans.p\_adr\_area\_unload (**

**p\_schema\_name := 'unnsi'::text**

**,p\_id\_region := 7::bigint**

**,p\_conn := (gar\_link.f\_conn\_set(~~13::numeric(3~~)))::text -- Именованное dblink-соединение**

**);**

**" unsi\_test\_06 -U postgres 1>> /home/rootadmin/abr\_upload/Y\_BUILD/06/r5\_process\_20231226\_174419.out 2>> /home/rootadmin/abr\_upload/Y\_BUILD/06/r5\_process\_20231226\_174419.err**

В схеме «unnsi» реиональной БД, находятся только сторонние таблицы. Далее смотрим файл типа «err».

**ОШИБКА: "fserver\_name" IS NULL**

**КОНТЕКСТ: функция PL/pgSQL f\_conn\_set(numeric), строка 16, оператор RAISE**

Ошибка в определении идентификатора внешнего сервера.