ARENADATA

Корпоративная платформа хранения
и обработки больших данных

Arenadata DB для Разработчиков

Часть 4

ADB – аналитическая СУБД для больших данных

Внешние таблицы в ADB:
Концепция External Table.
Протокол GPFDIST, утилита GPLOAD.
Протокол PXF.
Коннекторы ADB EE.

Пользовательские функции в ADB:
Классы функций ADB.
Языки написания функций.
Пользовательские агрегаты.

ARENADATA

Корпоративная платформа хранения
и обработки больших данных

Внешние таблицы

Доступ к данным: внешние таблицы

- Внешняя таблица (ВТ), `EXTERNAL TABLE` – метаобъект, предназначенный для доступа к данным, хранящимся вне СУБД ADB.
- ВТ не хранят данные. Данные запрашиваются или отправляются при каждом запросе заново.
- ВТ могут использоваться в запросах как обычные таблицы.
- ВТ бывают:
  - `READABLE` и `WRITABLE` – читать и писать в один объект нельзя.
  - `WEB` и обычные.
    - `WEB` – только для протокола `HTTP` и запуска системных команд в окружении ОС кластера.
    - Обычные предназначены для подключения к файлам или другим системам.
- ВТ на чтение всегда вызывают `REDISTRIBUTE MOTION` при запросе данных, на запись – не всегда.
- Доступно логирование ошибочных записей.

Протоколы

Протокол – метод доступа к источникам данных. Для WEB внешних таблиц:

- http:// - доступ к данным на HTTP-сервере. Только чтение. HTTPS не поддерживается.
- *EXECUTE* - Выполнение команды в операционной системе кластера.

Для простых внешних таблиц.

- file:// - доступ к файлам на ФС сегментов. С мастера данные получить нельзя. Для простых внешних таблиц.
- gpfdist:// - доступ к GPFDIST-серверу. Для простых внешних таблиц.
- gpfdists:// - доступ к SSL-версии GPFDIST. Для простых внешних таблиц.
- s3:// - доступ к файлам на Amazon S3 bucket.
- pxf:// - доступ к внешним ресурсам через Platform eXtension Framework.
- Custom – можно создавать свои протоколы с помощью C-библиотек.

Формат – метод работы с потоком данных:

- CSV
- TEXT. Отличается от CSV дефолтными настройками и позволяет установить DELIMITER=OFF для загрузки строк целиком.
- CUSTOM - можно создавать свои форматы с помощью функций. Например, часто применяется с протоколом PXF.

Обычная внешняя таблица на чтение

```
CREATE [READABLE] EXTERNAL TABLE table_name
    ( column_name data_type [, ...] | LIKE other_table )
     LOCATION (protocol://location[options]' [, ...])
     [ON MASTER] -- только для S3 и custom
     FORMAT 'TEXT'
           [( [HEADER] -- игнорировать первую строку. Недоступно для PXF
              [DELIMITER [AS] 'delimiter' | 'OFF']
              [NULL [AS] 'null string']
              [ESCAPE [AS] 'escape' | 'OFF']
              [NEWLINE [ AS ] 'LF' | 'CR' | 'CRLF']
              [FILL MISSING FIELDS] )] -- если не хватает столбцов, заполнять NULL
```

Обычная внешняя таблица на чтение

```
 | 'CSV'
           [( [HEADER]
              [QUOTE [AS] 'quote']
              [DELIMITER [AS] 'delimiter']
              [NULL [AS] 'null string']
              [FORCE NOT NULL column [, ...]]
              [ESCAPE [AS] 'escape']
              [NEWLINE [ AS ] 'LF' | 'CR' | 'CRLF']
              [FILL MISSING FIELDS] )]
| 'CUSTOM' (Formatter=<formatter_specifications>)
```

[ENCODING 'encoding'] -- кодировка

[[LOG ERRORS] SEGMENT REJECT LIMIT count [ROWS | PERCENT]] -- логировать ли ошибки и число допустимых

WEB внешняя таблица на чтение

```
CREATE [READABLE] EXTERNAL WEB [TEMPORARY | TEMP] TABLE table_name
   ( column_name data_type [, ...] | LIKE other_table )
      LOCATION ('http://webhost[:port]/path/file' [, ...])
    | EXECUTE 'command' [ON
                           ALL -- на всех сегментах
                         | MASTER
                         | number_of_segments  -- на N случайных сегментах
                         | HOST -- на одном сегменте на каждом хосте ИЛИ
                         | HOST ['segment_hostname'] -- на всех сегментах одного хоста
                         | SEGMENT segment_id ] -- конкретном сегменте (content)
```

… (форматы всё остальное аналогично обычной ВТ на чтение)

Обычная внешняя таблица на запись

```
CREATE WRITABLE EXTERNAL [TEMPORARY | TEMP] TABLE table_name
    ( column_name data_type [, ...] | LIKE other_table )
     LOCATION (protocol://location[options]' [, ...])
          FORMAT 'TEXT'
               [( [DELIMITER [AS] 'delimiter']
               [NULL [AS] 'null string']
               [ESCAPE [AS] 'escape' | 'OFF'] )]
          | 'CSV'
               [([QUOTE [AS] 'quote']
               [DELIMITER [AS] 'delimiter']
               [NULL [AS] 'null string']
               [FORCE QUOTE column [, ...]] ]
               [ESCAPE [AS] 'escape'] )]
           | 'CUSTOM' (Formatter=<formatter specifications>)
    [ ENCODING 'write_encoding' ]
    [ DISTRIBUTED BY (column, [ ... ] ) | DISTRIBUTED RANDOMLY ]
```

WEB внешняя таблица на запись

```
CREATE WRITABLE EXTERNAL WEB [TEMPORARY | TEMP] TABLE table_name
    ( column_name data_type [, ...] | LIKE other_table )
    EXECUTE 'command' [ON ALL]
```

… (форматы всё остальное аналогично обычной ВТ на запись)

Примеры

Внешняя таблица, читающая по протоколу file все csv-файлы из директорий `/data/expense/` на трех разных сегментных серверах.

```
CREATE REDABLE EXTERNAL TABLE ext_expenses (
   name text, date date, amount float4, category text, desc1 text )
LOCATION ('file://sdw1/data/expense/*.csv',
          'file://sdw2/data/expense/*.csv',
          'file://sdw3/data/expense/*.csv')
FORMAT 'CSV' (HEADER);
```

Примеры

Внешняя таблица, читающая по протоколу file файл hosts на первом сегментном сервере:

```
CREATE REDABLE EXTERNAL TABLE ext_expenses (name text)
LOCATION ('file://sdw1/etc/hosts') FORMAT 'TEXT' (DELIMITER 'OFF');
```

Примеры

Внешняя WEB таблица, читающая по протоколу http csv-файл с удаленного сервера:

```
CREATE READABLE EXTERNAL WEB TABLE fam_rel (parent text,student text)
LOCATION
('http://insight.dev.schoolwires.com/HelpAssets/C2Assets/C2Files/C2ImportFamRelSample.csv')
FORMAT 'CSV' (HEADER);
```

Нюансы

Можно указывать несколько источников (приемников) для одной таблице в LOCATION, но нельзя смешивать разные протоколы в рамках одной ВТ.

При длинный LOCATION нельзя переносить на другую строку.

В зависимости от протокола и опций в работе с внешними данными с одним источником (приемником):

- Может работать один выбранный сегмент.
- Могут работать несколько выбранных сегментов.
- Сразу все сегменты.

EXECUTE BT может создавать только суперпользователь. Для кастомизации команды в них доступны переменные:

Variable	Description
$GP_CID	Command count of the transaction executing the external table statement
$GP_DATABASE	The database in which the external table definition resides
$GP_DATE	The date on which the external table command ran
$GP_MASTER_HOST	The host name of the Greenplum master host from which the external table statement was dispatched
$GP_MASTER_PORT	The port number of the Greenplum master instance from which the external table statement was dispatched
$GP_SEG_DATADIR	The location of the data directory of the segment instance executing the external table command
$GP_SEG_PG_CONF	The location of the postgresql.conf file of the segment instance executing the external table command
$GP_SEG_PORT	The port number of the segment instance executing the external table command
$GP_SEGMENT_COUNT	The total number of primary segment instances in the Greenplum Database system
$GP_SEGMENT_ID	The ID number of the segment instance executing the external table command (same as **dbid** ingp_segment_configuration)
$GP_SESSION_ID	The database session identifier number associated with the external table statement
$GP_SN	Serial number of the external table scan node in the query plan of the external table statement
$GP_TIME	The time the external table command was executed
$GP_USER	The database user executing the external table statement
$GP_XID	The transaction ID of the external table statement

Примеры

Внешняя WEB таблица, читающая файл **`postmaster.opts`** каждого сегмента:

```
CREATE EXTERNAL WEB TABLE pid (config text)
EXECUTE 'cat $GP_SEG_DATADIR/postmaster.opts' ON ALL
FORMAT 'TEXT' (DELIMITER 'OFF');
```

Внешняя WEB таблица, пишущая данные в txt-файлы с датой и временем в названии в папку каждого сегмента:

```
CREATE EXTERNAL WEB TABLE pid (config text)
EXECUTE 'cat > $GP_SEG_DATADIR/myfile_$GP_DATE_$GP_TIME.txt' ON ALL
FORMAT 'TEXT' (DELIMITER ',');
```

Лабораторная работа: Внешние таблицы

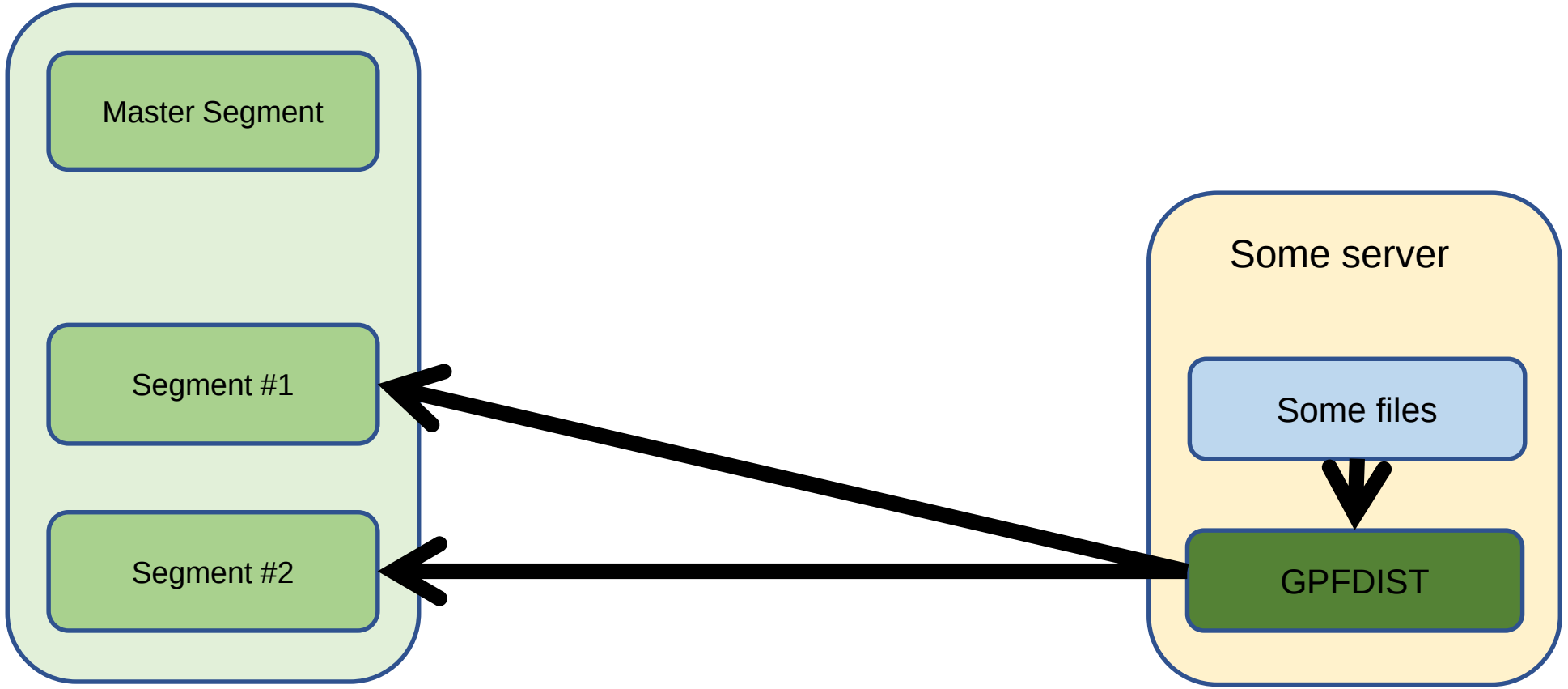
1. Создайте две внешние таблицы, которые выводили бы содержимое файла «/etc/hosts» с каждого сегмент-сервера:
   - С помощью EXECUTE, используя WEB External Table
   - С помощью протокола file://, используя External Table

ARENADATA

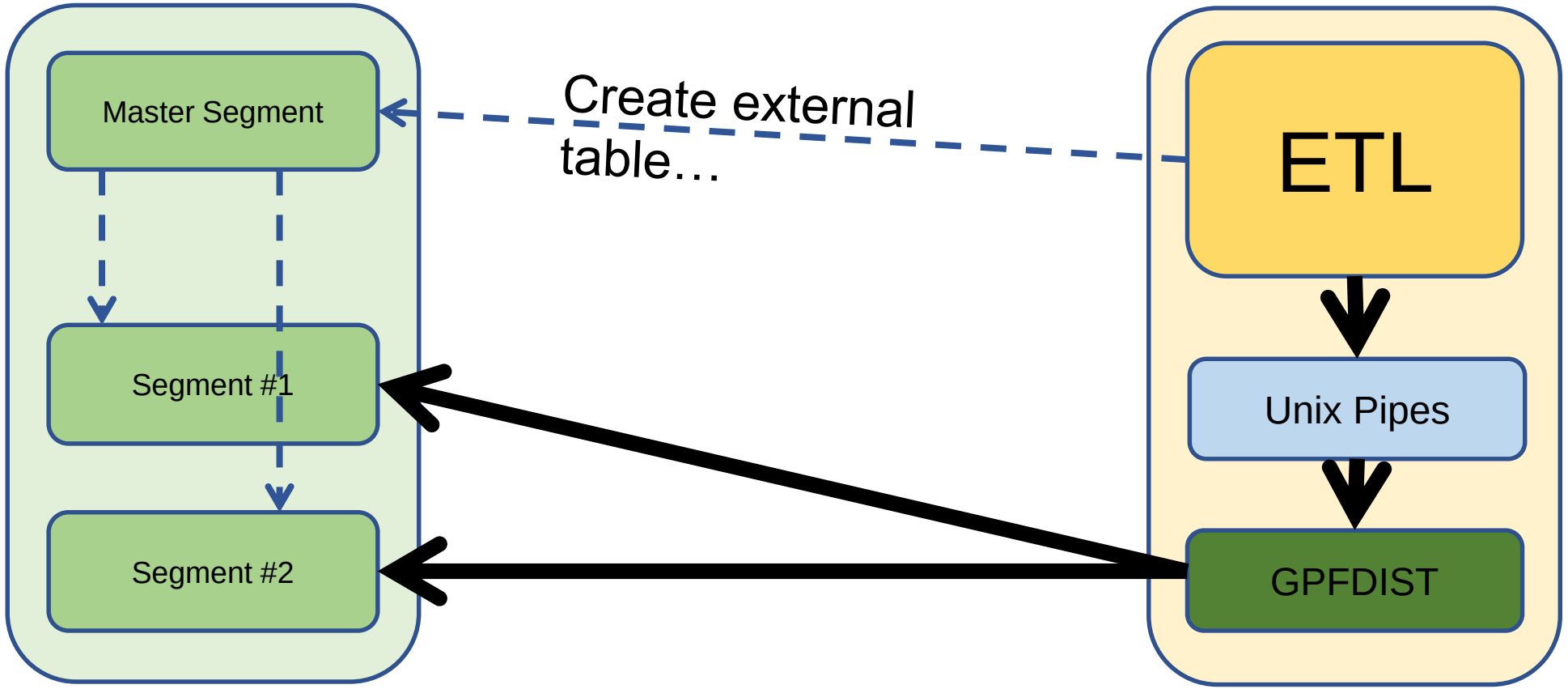
Корпоративная платформа хранения
и обработки больших данных

GPFDIST

GPFDIST. Схема работы



GPFDIST. Схема работы



GPFDIST. Схема работы

- GPFDIST – сервер модифицированного протокола HTTP, предназначенный для загрузки данных в ADB
- GPFDIST – обычно, самый быстрый способ загрузки **несжатых** данных с внешних серверов
- Утилита входит в состав rpm-пакета с СУБД: `/usr/lib/gpdb/bin/gpfdist`. Её можно и нужно ставить на внешние серверы.

GPFDIST. Параметры запуска

[-d *directory*] – директория с файлами.

[-p *http_port*] – слушаемый порт (8080).

[-P *last_http_port*] – в случае, если слушаемый порт занят, перебирать порты до этого значения.

[-l *log_file*] – вывод лога (stdout).

[-t *timeout*] – таймут для установки соединения с ГП, секунд (5).

[-S] – использовать флаг ОС O_SYNC – синхронная запись на диск, не использовать кеш.

[-w *time*] – ожидание перед закрытием файла на запись. В случае PIPES лучше чуть больше (0).

[-v | -V] – verbose | Very verbose.

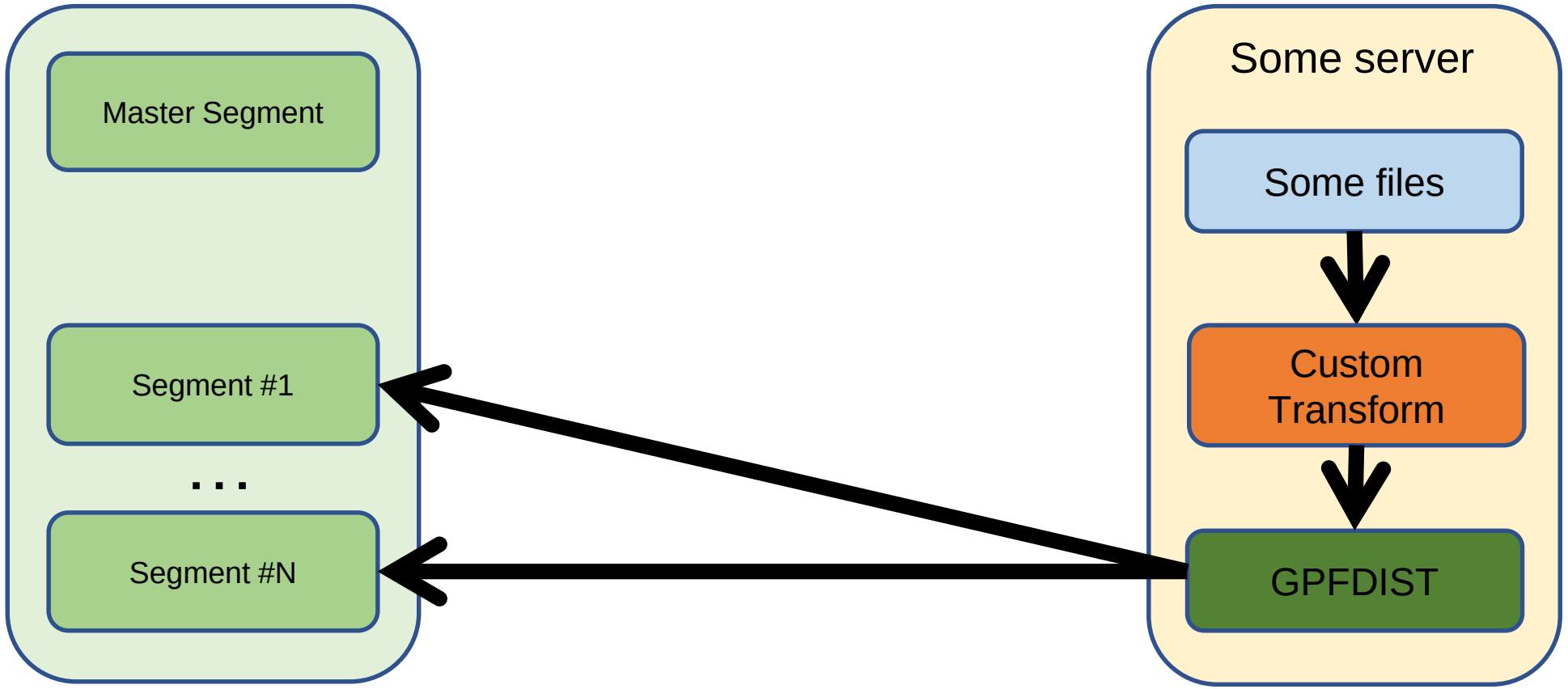
[-s] – упрощённое логирование, только WARN и ERROR

[-m *max_length*] - максимальная длина строки в байтах. Помогает увеличить от ошибок «line too long» (32768).

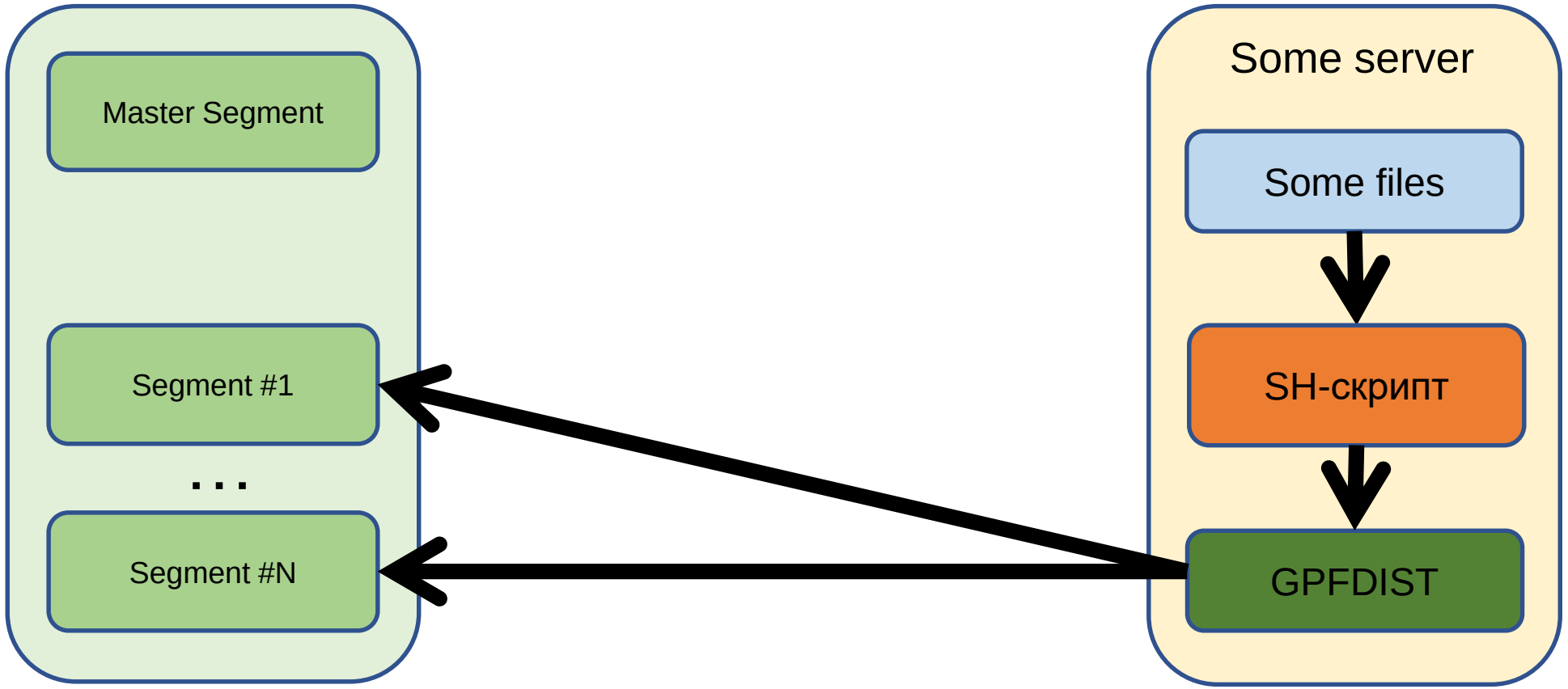
[--ssl *certificate_path* [--sslclean *wait_time*]] – параметры SSL.

[-c *config.yml*] - файл трансформаций.

GPFDIST. Трансформация данных



GPFDIST. Параметры запуска



GPFDIST. Файл трансформаций

- **-c config.yaml –** YAML-файл, описывающий опциональные трансформации.

```
---
VERSION: 1.0.0.1
TRANSFORMATIONS:
  transform_name:
    TYPE:     input
    COMMAND:  /bin/bash transform.sh %filename%
```

- Можно использовать выборочные трансформации:

```
CREATE READABLE EXTERNAL TABLE ext_tab(LIKE tab)
   LOCATION ('gpfdist://hostname:8080/prices.xml#transform=transform_name')
   FORMAT 'TEXT' (DELIMITER '|')
   LOG ERRORS SEGMENT REJECT LIMIT 10;
```

GPFDIST. Системные параметры

- **gp_external_max_segs -** как много сегментов участвуют в работе с gpfdist (64).
- **readable_external_table_timeout –** таймаут на чтение для всех extarnal tables (0 – не отменяем, ждём до конца).
- **verify_gpfdists_cert** – проверять ли сертификаты для gpfdists.
- **writable_external_table_bufsize –** размер буфера внешней таблицы на запись.

GPFDIST. Системные параметры

- HTTP – не параллельный протокол.
- GPFDIST – параллельный протокол благодаря заголовкам:
  - X-GP-SEGMENT-ID:0.
  - X-GP-SEGMENT-COUNT:4.
- GPFDIST и HTTP могут использовать wildcards (ex: gpfdist://filehost:8081/*, gpfdist://filehost:8081/[0-9].txt).
- Как много сегментов участвуют в работе с gpfdist: GUC gp_external_max_segs.
- Gpfdist работает со сжатыми файлами на чтение, НО:
  - В отличии от PXF данные разжимаются на стороне gpfdist и передаются по сети несжатыми.

Лабораторная работа: GPFDIST (1)

Все действия под gpadmin:

1. Создайте директорию на мастер-сервере: `mkdir /tmp/gpfdist_test.`
2. Сгенерируйте два файла с синтетикой:

```
for i in $(seq 1 10000); do echo "$i,foo$i"; done > /tmp/gpfdist_test/sample_1.csv.
for i in $(seq 10001 20000); do echo "$i,foo$i"; done > /tmp/gpfdist_test/sample_2.csv.
```

3. Запустите сервер gpfdist на директории /tmp/gpfdist_test на порту 5555.
4. Создайте внешнюю таблицу `external_table1` с двумя полями (id,gen), которая будет читать оба файла через gpfdist.
5. Прочитайте данные из таблицы.

Лабораторная работа: GPFDIST (2)

(Все действия под gpadmin)

1. Создайте скрипт /tmp/gpfdist_test/foobar.sh: `cat $1 |sed 's/foo/bar/g'.`
2. Создайте файл /tmp/gpfdist_test/config.yaml:

```
---
VERSION: 1.0.0.1
TRANSFORMATIONS:
  foobar:
    TYPE:     input
    COMMAND:  /bin/bash /tmp/gpfdist_test/foobar.sh %filename%
```

1. Запустите сервер gpfdist на директории /tmp/gpfdist_test на порту 5555 с созданным файлом конфигурации.
2. Создайте внешнюю таблицу `external_table2` с двумя полями (id,gen), которая будет читать оба файла через gpfdist с трансформацией `foobar.`
3. Прочитайте данные из таблицы.

ARENADATA

Корпоративная платформа хранения
и обработки больших данных

GPLOAD

GPLOAD

- GPLOAD – обёртка вокруг GPFDIST, которая сама:
  - Поднимает GPFDIST sever.
  - Создаёт external таблицу.
  - Переливает данные в физическую таблицу.
  - Удаляет external таблицу.
- Делает все действия в одной транзакции (опционально).

GPLOAD. Параметры запуска

gpload

-f *control_file* – основной конфиг.

[**-l** *log_file*].

[**-h** *hostname*].

[**-p** *port*].

[**-U** *username*].

[**-d** *database*].

[**-W**] – дополнительно спрашивать пароль.

[**--gpfdist_timeout** *seconds*] – таймаут gpfdist.

[**--no_auto_trans**] – не использовать одну транзакцию.

[[**-v** | **-V**] [**-q**]] – verbose, Very verbose или Quite.

[**-D**] – debug.

```
---
VERSION: 1.0.0.1
DATABASE: db_name
USER: db_username
HOST: master_hostname
PORT: master_port
GPLOAD:
   INPUT:
    - SOURCE:
         LOCAL_HOSTNAME:
           - hostname_or_ip – по какому адресу ADB будет стучаться на хост GPLOAD.
         PORT: http_port – порт GPFDIST
       | PORT_RANGE: [start_port_range, end_port_range] – порты GPFDIST.
         FILE:
           - /path/to/input_file
         SSL: true | false
         CERTIFICATES_PATH: /path/to/certificates
    - FULLY_QUALIFIED_DOMAIN_NAME: true | false
```

```
- COLUMNS:
            - field_name: data_type
    - TRANSFORM: 'transformation' – то же что и в GPFDIST.
    - TRANSFORM_CONFIG: 'configuration-file-path' – то же что и в GPFDIST.
    - MAX_LINE_LENGTH: integer
    - FORMAT: text | csv
    - DELIMITER: 'delimiter_character'
    - ESCAPE: 'escape_character' | 'OFF'
    - NULL_AS: 'null_string'
    - FORCE_NOT_NULL: true | false
    - QUOTE: 'csv_quote_character'
    - HEADER: true | false
    - ENCODING: database_encoding
    - ERROR_LIMIT: integer
    - LOG_ERRORS: true | false
  EXTERNAL:
     - SCHEMA: schema | '%' – отдельная схема для внешних таблиц.
```

OUTPUT:

- TABLE: schema.table_name
- MODE: insert | update | merge **– как именно вставлять данные из внешней таблицы.**

UPDATE – Обновляет UPDATE_COLUMNS по MATCH_COLUMNS где UPDATE_CONDITION true.

MERGE – вставляет новые строки и апдейтит UPDATE_COLUMNS в старых .

- MATCH_COLUMNS: **- список столбцов для соединения.**
    - target_column_name
- UPDATE_COLUMNS: **- список столбцов которые надо обновлять для update/merge.**
    - target_column_name
- UPDATE_CONDITION: 'boolean_condition‘ **– дополнительный фильтр для update/merge.**
- MAPPING:
        target_column_name: source_column_name | 'expression‘ **– кастомный мапиинг столбцов.**

PRELOAD:

- TRUNCATE: true | false **– очистить целевую таблицу перед заливкой данных.**
- REUSE_TABLES: true | false **– не удалять внешнюю таблицу.**

SQL:

- BEFORE: "sql_command“ **– что выполнить до переливки.**
- AFTER: "sql_command" **– что выполнить после переливки.**

Нюансы

- Есть требования:
  - Хост должен был доступен со всех серверов кластера ADB.
  - Должен быть прописан gpfdist в PATH.
  - Python 2.6.2 или больше с пакетами:
    - Pygresql.
    - Pyyaml.

Лабораторная работа: GPLOAD

Все действия под gpadmin:

1. Создайте таблицу table10 с двумя полями (id,gen).
2. Создайте файл /tmp/gpfdist_test/gpload_config.yaml так, чтобы загрузить содержимое файлов sample_1.csv и sample_2.csv в таблицу table10 методом INSERT.
3. Загрузите данные.

ARENADATA

Корпоративная платформа хранения
и обработки больших данных

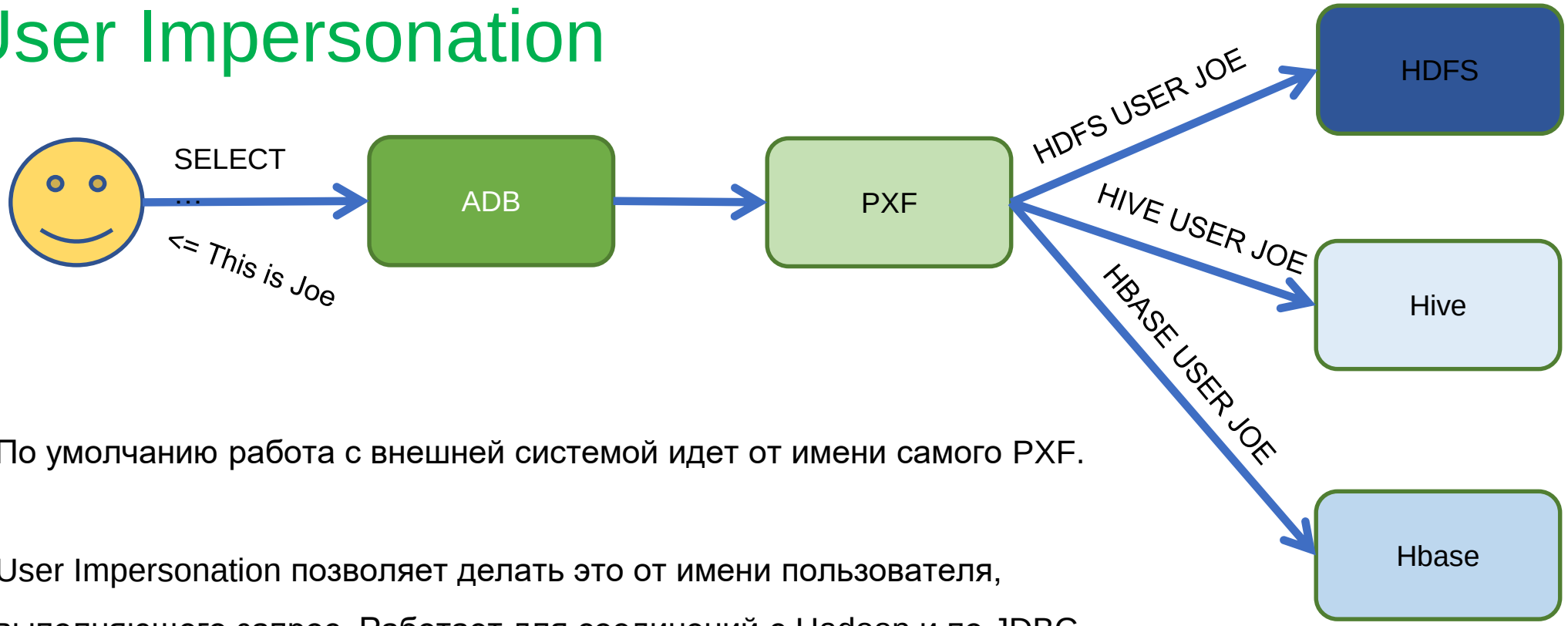
PXF

Platform Extension Framework

Platform Extension Framework (PXF) – это отдельное ПО, выступающее посредником между сегментами кластера и сторонними системами (базами данных, хранилищами).

- Реализован как отдельный JAVA-сервис, работающий на всех сегмент-серверах под своим пользователем (PXF).
- Запускается одна копия PXF на каждом сегментном сервере.
- Сегменты ADB обращаются к своей копии через через REST.
- Содержит подключаемые модули – коннекторы и плагины, необходимые для доступа к внешним системам.
- Возможность чтения и записи данных зависит от плагина. Бывают однонаправленными и двунаправленными.
- Можно разрабатывать свои коннекторы. Корпоративная версия ADB включает коннекторы к ClickHouse и Kafka.

User Impersonation

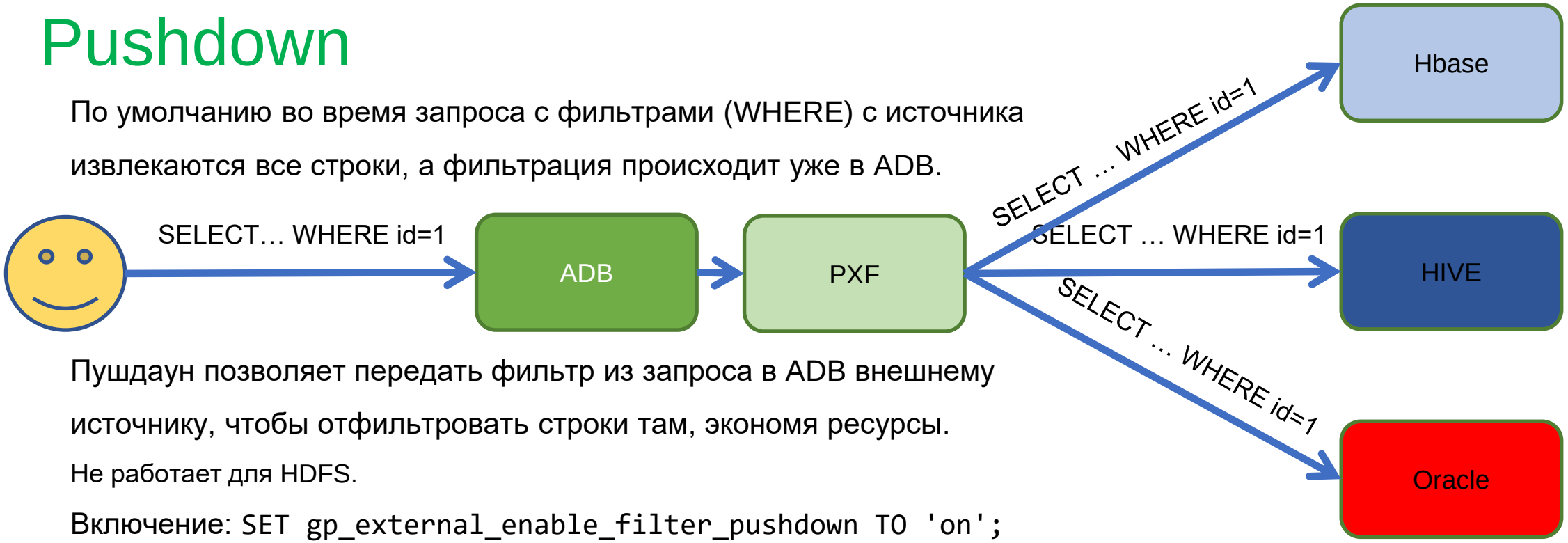


По умолчанию работа с внешней системой идет от имени самого PXF.

User Impersonation позволяет делать это от имени пользователя, выполняющего запрос. Работает для соединений с Hadoop и по JDBC.

Pushdown

По умолчанию во время запроса с фильтрами (WHERE) с источника извлекаются все строки, а фильтрация происходит уже в ADB.



Пушдаун позволяет передать фильтр из запроса в ADB внешнему источнику, чтобы отфильтровать строки там, экономя ресурсы.

Не работает для HDFS.

Включение: `SET gp_external_enable_filter_pushdown TO 'on';`

Типы данных:

- `INT, array of INT`
- `FLOAT`
- `NUMERIC`
- `BOOL`
- `CHAR, TEXT, array of TEXT`
- `DATE, TIMESTAMP` - только для JDBC

Операторы:

- `<, <=, >=, >`
- `<>, =`
- `IN`
- `LIKE` - только для TEXT

Профили

Профиль – это папка с конфигурационными файлами для конкретного подключения к внешней системе.

Она должна присутствовать у каждого экземпляра PXF в директории /var/lib/pxf/servers/

Название этой папки указывается без кавычек в LOCATION в опции SERVER:

- …&SERVER=<name>…;

Необходимо настраивать подключение в следующих случаях:

- Нужен доступ до нескольких кластеров Hadoop.
- Кластер Hadoop использует аутентификацию по Kerberos (необходим файл pxf-site.xml).
- Необходимо скрыть параметры подключения во внешних таблицах для JDBC (необходим файл jdbc-site.xml);
- Если нужно выполнить именованный запрос с помощью JDBC (query:<query_name>) .
- Нужно включить или выключить параметр «User impersonation» для Hadoop или JDBC (файл pxf-site.xml или jdbc-site.xml)).

Профиль по-умолчанию - директория /var/lib/pxf/servers/default/

Для создания профиля необходимо создать новую директорию в папке /var/lib/pxf/servers/. Например:

- /var/lib/pxf/servers/pg_1/
- /var/lib/pxf/servers/ora/

Шаблоны файлов для настройки профилей находятся в /var/lib/pxf/templates/

Профили: Hadoop

Для связи с Hadoop, Hive и Hbase необходимы core-site.xml, hdfs-site.xml, hive-site.xml и hbase-site.xml.

- Скопируйте эти файлы с активной name-ноды вашего Hadoop-кластера в директорию /var/lib/pxf/servers/default/ на всех сегмент-серверах.

Для доступа к нескольким кластерам Hadoop необходимо создать разные директории в /var/lib/pxf/servers/ и скопировать необходимые *-site.xml файлы в эту директорию. Например:

- /var/lib/pxf/servers/hd1
- /var/lib/pxf/servers/hd2

Для настройки аутентификации по Kerberos и параметра User impersonation необходимо использовать файл pxf-site.xml.

LOCATION

При работе с PXF многие параметры задаются непосредственно в строке LOCATION. Строка формируется следующим образом:

- Протокол – pxf.
- Локация ресурса для чтения или записи (директория, файл, таблица).
- Набор опций, часть из которых – общие, а часть зависит от коннектора.

```
LOCATION ('pxf://<path-to-hdfs-file>?[<option>=<value>][&<option>=<value>][…]')
```

Общие опции:

- SERVER – профиль конфигурации подключения.
- PROFILE – указатель на коннектор.

```
LOCATION ('pxf://data/my_file.txt?PROFILE=hdfs:text&SERVER=devHadoop1')
```


HDFS READ

Для создания внешней таблицы, читающей файловые данные из HDFS, необходимо в LOCATION указать:

- Протокол и путь к файлу.
- PROFILE для конкретного типа данных (см. таблицу).
- SERVER (если не используется дефолтный).

Data Format	Profile
delimited single line text	hdfs:text
delimited text with quoted linefeeds	hdfs:text:multi
AVRO	hdfs:avro
JSON	hdfs:json
Parquet	hdfs:parquet
AvroSequenceFile	hdfs:AvroSequenceFile
SequenceFile	hdfs:SequenceFile

В блоке FORMAT такой внешней таблице доступны три варианта:

- TEXT – только для hdfs:text. Настройки как у обычных внешних таблиц (кроме опции HEADER).
- CSV – hdfs:text.
- Custom – для остальных форматов.

Для Custom в качестве `<formatting-properties>` указывается форматтер `pxfwritable_import`.

```
CREATE [REDABLE] EXTERNAL TABLE <table_name> (…)
LOCATION (
'pxf://<path-to-hdfs-file>?PROFILE=<profile>[[SERVER=<server_name>][&<custom-option>=<value>][...]] ')
FORMAT '[TEXT|CSV|CUSTOM]' (<formatting-properties>) … ;
```

HDFS READ: Примеры

```
--TEXT
CREATE EXTERNAL TABLE pxf_hdfs_textsimple(location text, month text, num_orders int, total_sales float8)
LOCATION ('pxf://data/examples/pxf_hdfs_simple?PROFILE=hdfs:text&SERVER=hd1')
FORMAT 'TEXT' (delimiter=E',');

--AVRO
CREATE EXTERNAL TABLE pxf_hdfs_avro(id bigint, username text, followers text)
LOCATION
('pxf://data/examples/pxf_hdfs_avro.avro?PROFILE=hdfs:avro&COLLECTION_DELIM=,&MAPKEY_DELIM=:&RECORDKEY_DELIM=:')
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

HDFS WRITE

Для создания внешней таблицы, пишущей файловые данные в HDFS, необходимо в LOCATION указать:

- Протокол и путь к директории для записи.
- PROFILE для конкретного типа данных (доступна запись в форматах Text, Sequence и Parquet).
- SERVER (если не используется дефолтный).

В блоке FORMAT такой внешней таблице доступны три варианта:

- TEXT – только для hdfs:text. Настройки как у обычных внешних таблиц (кроме опции HEADER).
- CSV – hdfs:text.
- Custom – для остальных форматов.

Для Custom в качестве `<formatting-properties>` указывается форматтер `pxfwritable_export`.

Результат записи – директория, которая содержит по файлу на каждый сегмент + id транзакции.

```
CREATE WRITABLE EXTERNAL TABLE <table_name> (…) LOCATION (
'pxf://<path-to-hdfs-dir>?PROFILE=<profile>[[&SERVER=<server_name>][&<custom-option>=<value>][...]]')
FORMAT '[TEXT|CSV|CUSTOM]' (<formatting-properties>) … ;
```

Copyright arenadata.io

HDFS WRITE: Дополнительные опции

Option	Value Description	Profile
COMPRESSION_CODEC	Класс компрессора. Один из: org.apache.hadoop.io.compress.DefaultCodec org.apache.hadoop.io.compress.BZip2Codec org.apache.hadoop.io.compress.GzipCodec	hdfs:text, hdfs:SequenceFile
COMPRESSION_TYPE	Единица компрессии, RECORD (default) или BLOCK	hdfs:text, hdfs:SequenceFile
DATA-SCHEMA	Класс сериализатора. Обязателен для SequenceWritable	hdfs:SequenceFile
THREAD-SAFE *(опция устарела, доступна в старых сборках)*	Boolean-значение, определяет, параллелить ли запись. По умолчанию включена запись в один поток (true)	hdfs:text, hdfs:SequenceFile

HDFS WRITE: Примеры

--TEXT

```
CREATE WRITABLE EXTERNAL TABLE pxf_hdfs_writabletbl_1(location text, month text)
LOCATION ('pxf://data/pxf_examples/pxfwritable_hdfs_textsimple1?
PROFILE=hdfs:text&COMPRESSION_CODEC=org.apache.hadoop.io.compress.GzipCodec')
FORMAT 'TEXT' (delimiter=',');
```

--Parquet

```
CREATE WRITABLE EXTERNAL TABLE pxf_tbl_parquet (location text, month text, number_of_orders int,
total_sales double precision)
LOCATION ('pxf://data/pxf_examples/pxf_parquet?PROFILE=hdfs:parquet')
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_export');
```

Hive READ

Для создания внешней таблицы, читающей данные из таблицы в Hive, необходимо в LOCATION указать:

- Протокол и таблицу (с указанием базы).
- PROFILE для конкретного типа данных (см. таблицу). Профиль Hive читает все форматы, но делает это медленнее, чем специальные.
- SERVER (если не используется дефолтный).

File Format	Profile
TextFile	Hive, HiveText
SequenceFile	Hive
RCFile	Hive, HiveRC
ORC	Hive, HiveORC, HiveVectorizedORC
Parquet	Hive

В блоке FORMAT такой внешней таблице доступны два варианта:

- TEXT – только для hdfs:text. Можно указать свой разделитель.
- Custom – для остальных форматов.

Форматтер – `pxfwritable_import.`

```
CREATE [READABLE] EXTERNAL TABLE <table_name> (…)
LOCATION (
'pxf://<db>.<table>?PROFILE=<profile >[[&SERVER=<server_name>][&<custom-option>=<value>[...]]')
FORMAT 'CUSTOM|TEXT' (formatter='pxfwritable_import' | delimiter='<delim>') … ;
```

HBase READ

Для создания внешней таблицы, читающей данные из таблицы в Hive, необходимо в LOCATION указать:

- Протокол и таблицу.
- PROFILE – Hbase.
- SERVER (если не используется дефолтный).

В блоке FORMAT такой внешней таблицы доступен только один вариант:

- Custom – для всего.

Для Custom в качестве `<formatting-properties>` указывается форматтер `pxfwritable_import.`

- Имя колонки в Hbase состоит из двух частей: <column-family>:<column-qualifier>.
- Служебная колонка recordkey может использоваться во внешних таблицах в качестве фильтра в условии WHERE.

```
CREATE [READABLE] EXTERNAL TABLE <table_name>(<column-family>:<column-qualifier> <data-type>, …)
LOCATION ('pxf://<hbase-table-name>?PROFILE=HBase[&SERVER=<server_name>]')
FORMAT 'CUSTOM' (formatter='pxfwritable_import') … ;
```

ARENADATA

Корпоративная платформа хранения
и обработки больших данных

PXF JDBC

JDBC READ

JDBC-драйвер целевой СУБД должен быть скопирован в папку /usr/lib/pxf/lib/ и подключен в private-classpath.xml на каждом сервере.

В LOCATION указать:

- Протокол и таблицу (или название файла именованного запроса).
- PROFILE – JDBC.
- SERVER (если не используется дефолтный или не указаны настройки JDBC непосредственно в Location).

В блоке FORMAT – Custom (`pxfwritable_import`).

Поддержка ограниченного набора типов данных:

- INTEGER, BIGINT, SMALLINT.
- REAL, FLOAT8.
- NUMERIC.
- BOOLEAN
- VARCHAR, BPCHAR, TEXT.
- DATE.
- TIMESTAMP.

```
CREATE [READABLE] EXTERNAL TABLE <table_name>(…)
LOCATION ('pxf://<schema-name>.<external-table-name>|query:<query_name>?PROFILE=Jdbc
[[&SERVER=<server_name>][&<custom-option>=<value>][...]]')
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import') … ;
```

JDBC READ: Опции в LOCATION

Option Name	Description
JDBC_DRIVER	Имя класса драйвера (обязательно)
DB_URL	URL СУБД. Зависит от типа внешней СУБД, обычно включает hostname, port, и название БД (обязательно)
USER	Имя пользователя (опционально)
PASS	Пароль (опционально)
PARTITION_BY	Имя столбца, по которому сегменты будут делить данные во внешней системе. Может быть типа: date, int или enum (список). Тип указывается через ":". (опционально, если не указано, данные будет выкачивать один сегмент)
RANGE	Начальное и конечное значение для забора из внешней системы. В случае date и int указываются начальные и конечные значения через ":", в случае enum – перечисление через ":". (обязательно если указан PARTITION_BY)
INTERVAL	Величина интервала забора данных из внешней системы. Для DATE может указываться с типом (day, month, etc.) (обязательно если указан PARTITION_BY)

--Примеры

- `&PARTITION_BY=year:int&RANGE=2011:2013&INTERVAL=1`
- `&PARTITION_BY=createdate:date&RANGE=2013-01-01:2016-01-01&INTERVAL=1:month`
- `&PARTITION_BY=color:enum&RANGE=red:yellow:blue`

JDBC WRITE

JDBC-драйвер целевой СУБД должен быть скопирован в папку /usr/lib/pxf/lib/ и подключен в private-classpath.xml на каждом сервере.

В LOCATION указать:

- Протокол и таблицу.
- PROFILE – JDBC.
- SERVER (если не используется дефолтный или не указаны настройки JDBC непосредственно в Location).

В блоке FORMAT – Custom (`pxfwritable_export`).

Поддержка ограниченного набора типов данных:

- INTEGER, BIGINT, SMALLINT.
- REAL, FLOAT8.
- NUMERIC.
- BOOLEAN
- VARCHAR, BPCHAR, TEXT.
- DATE.
- TIMESTAMP.

```
CREATE WRITABLE EXTERNAL TABLE <table_name>(…)

LOCATION ('pxf://<schema-name>.<external-table-name>?PROFILE=Jdbc [&<custom-option>=<value>[...]]')

FORMAT 'CUSTOM' ('pxfwritable_export');
```

JDBC WRITE: Параметры в LOCATION

Option Name	Description
JDBC_DRIVER	Имя класса драйвера (обязательно)
DB_URL	URL СУБД. Зависит от типа внешней СУБД, обычно включает hostname, port, и название БД (обязательно)
USER	Имя пользователя (опционально)
PASS	Пароль (опционально)
BATCH_SIZE	Использовать метод batch JDBC-драйвера (если поддерживается). Задаётся размер батча в строках (опционально)
POOL_SIZE	Использовать пулинг (переиспользование сессий), задаётся размер пула (опционально)

Профили: JDBC

- Создать директорию в /var/lib/pxf/servers/.
- Скопировать в неё шаблон */var/lib/pxf/templates/jdbc-site.xml* .
- Отредактировать файл *jdbc-site.xml* согласно вашим параметрам подключения.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <property>
        <name>jdbc.driver</name>
        <value>oracle.jdbc.driver.OracleDriver</value>
    </property>
    <property>
        <name>jdbc.url</name>
        <value>jdbc:oracle:thin:@//10.1.2.15/XE</value>
    </property>
    <property>
        <name>jdbc.user</name>
        <value>user</value>
    </property>
    <property>
        <name>jdbc.password</name>
        <value>password</value>
    </property>
</configuration>
```

Профили: User Impersonation JDBC

- Для включения параметра User Impersonation, в файл jdbc-site.xml необходимо добавить параметр:

```
<property>
        <name>pxf.service.user.impersonation</name>
        <value>true</value>
</property>
```

- Для каждого внутреннего пользователя GP, можно настроить свой файл доступа к внешнему источнику:
    - Файл должен называться <greenplum_user_name>-user.xml, где <greenplum_user_name> - это имя пользователя внутри GP.
    - Файл должен лежать в директории профиля.
    - Файл может содержать имя внешнего пользователя, пароль и другие настройки для подключения к внешнему источнику.
    - Приоритет настроек в этом файле будет выше чем в файле *jdbc-site.xml.*

Профили: Именованные запросы JDBC

Аналог внешнего представления. Доступны только для READABLE внешних таблиц.

Когда необходимо использовать Named Query:

- Джойн двух внешних таблиц на стороне источника.
- Запросы с агрегацией на стороне источника.
- Выполнение запроса полностью на стороне внешнего источника.
- Выполнение запроса, синтаксис которого не поддерживается в GP.

Использование:

- Создать директорию в /var/lib/pxf/servers/. Например, /var/lib/pxf/servers/db1/;
- Создать файл с запросом внутри созданной директории;
- При создании внешней таблицы, использовать параметры

  `query:<query_name>?PROFILE=Jdbc&SERVER=<server_name>`, где `query_name` – это имя файла запроса без расширения, а `server_name`- это имя ранее созданной директории.

Внешние таблицы: PXF JDBC примеры (чтение)

```
--Oracle
CREATE EXTERNAL TABLE public.insurance_sample_jdbc_ora_ro(
policyid bigint,
statecode text)
LOCATION ('pxf://default/pxf_user.insurance_test?
    PROFILE=JDBC&
    JDBC_DRIVER=oracle.jdbc.driver.OracleDriver&
    DB_URL=jdbc:oracle:thin:@//35.205.173.31:1521/XE&
    USER=pxf_user&PASS=password&
    PARTITION_BY=policyid:int&
    RANGE=100000:999999&
    INTERVAL=10000')
    FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import’);
```

```
--Oracle with profile
CREATE EXTERNAL TABLE
pxf_oracle (id integer, descr text)
LOCATION ('pxf://gp.companies?PROFILE=JDBC&SERVER=oracle’)
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import’);
```

```
--Oracle Named Query
CREATE EXTERNAL TABLE
pxf_oracle_query (company_hq text, company text)
LOCATION ('pxf://query:query_ora?PROFILE=JDBC& SERVER=oracle’
)
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

```
--PostgreSQL
CREATE EXTERNAL TABLE pxf_tblfrompg(id int)
LOCATION ('pxf://public.forpxf_table1?
    PROFILE=Jdbc&JDBC_DRIVER=org.postgresql.Driver&
    DB_URL=jdbc:postgresql://pserver:5432/pgtestdb&
    USER=pxfuser1&
    PASS=changeme')
    FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import’);
```

Внешние таблицы: PXF JDBC пример (запись)

--PostgreSQL

```
create writable external table postgr_target_w (id1 int, id2 int)
LOCATION ('pxf://public.postgr_target?
    JDBC_DRIVER=org.postgresql.Driver&
    PROFILE=JDBC&
    DB_URL=jdbc:postgresql://postgr/pxf_db&
    USER=pxf_user&PASS=pxf_password&
    POOL_SIZE=3&
    BATCH_SIZE=10')
    FORMAT 'CUSTOM' (FORMATTER='pxfwritable_export')
    DISTRIBUTED BY (id1);
```

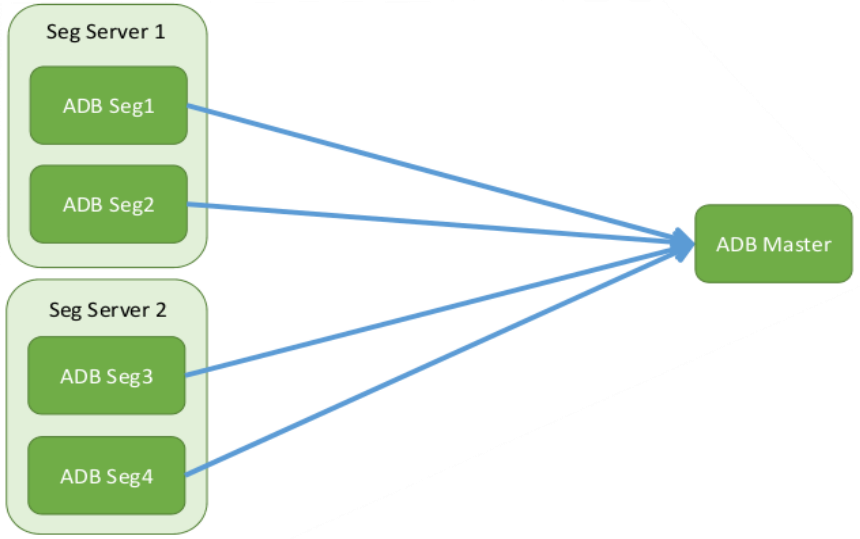
Лабораторная: PXF JDBC (1)

1. Создайте таблицу table11 (колоночная, сжатие `zstd` уровня 1):

```
id1 int,
id2 int,
gen text,
now timestamp without time zone
```

2. Создайте индекс на поле `id1`
3. Вставьте в таблицу данные:

```
insert into table11
    select gen, gen, 'text' || gen::text,now()
    from generate_series(1,4000000) gen;
```



4. Создайте `READABLE` внешнюю таблицу `table_11_pxf_read`, которая:
   - Будет обращаться к серверу `mdw` к таблице `table11.`
   - Использует JDBC-драйвер `PostgreSQL: org.postgresql.Driver`.
   - Использует имя пользователя `gpadmin`.
   - Читает данные в один поток.
5. Изучите результат `EXPLAIN ANALYZE` запроса `select count(1) from table_11_pxf_read.`

Лабораторная: PXF JDBC (2)

6. Создайте аналогичную вторую `READABLE` внешнюю таблицу `table_11_pxf_read_parallel` таким образом, чтобы:
   - Чтение выполнялось параллельно.
   - Шардирование происходило по полю `id1.`
   - Размер одной пачки данных составил 500000 строк.
7. Изучите результат `EXPLAIN ANALYZE запроса select count(1) from table_11_pxf_read_parallel.`

8. Создайте пустую таблицу `table12`, которая полностью повторяет структуры таблицы `table11.`
9. Создайте аналогичную `WRITABLE` таблицу `table_12_pxf_write`, которая:
   - Будет обращаться к серверу mdw к таблице `table12`.
   - Использует `BATCH_SIZE=25`.
   - Использует `POOL_SIZE=2`.
   - Распределена по полю `id1`.
3. Выполните запрос

   `insert into table_12_pxf_write select * from table11 limit 100;`

   и проверьте, загрузились ли данные в таблицу table12.

ARENADATA

Корпоративная платформа хранения
и обработки больших данных

Коннекторы ADB Enterprise для работы с ADS и ADQM

Kafka WRITE

В поставке ADB EE присутствует kafka-connector для PXF, позволяющий производить транзакционную загрузку данных из ADB в кластер брокера сообщений Kafka. Устанавливается как сервис в ADCM.

- Для работы нужно настроить профиль PXF (*/var/lib/pxf/conf/pxf-profiles.xml*):

```
<profiles>
    <profile>
        <name>kafka</name>
        <description>A profile for export data into Apache Kafka</description>
        <plugins>
            <accessor>org.greenplum.pxf.plugins.kafka.KafkaAccessor</accessor>
            <resolver>org.greenplum.pxf.plugins.kafka.KafkaResolver</resolver>
        </plugins>
        <optionMappings>
            <mapping option="BOOTSTRAP_SERVERS" property="kafka.bootstrap.servers"/>
            <mapping option="BATCH_SIZE" property="kafka.batch.size"/>
            <mapping option="TOPIC_AUTO_CREATE_FLAG" property="kafka.topic.auto.create"/>
            <mapping option="AVRO_DEFAULT_DECIMAL_PRECISION" property="avro.decimal.default.precision" />
            <mapping option="AVRO_DEFAULT_DECIMAL_SCALE" property="avro.decimal.default.scale" />
        </optionMappings>
    </profile>
</profiles>
```

Kafka WRITE

Для создания внешней таблицы, читающей данные из Kafka, необходимо в LOCATION указать:

- Протокол и таблицу (с указанием базы).
- Топик Kafka.
- SERVER (если не используется дефолтный).
- Дополнительные опции.

Форматтер – `pxfwritable_export.`

```
CREATE WRITABLE EXTERNAL TABLE <table_name>
    ( <column_name> <data_type> [, ...] | LIKE <other_table> )
LOCATION ('pxf://<kafka_topic>?PROFILE=kafka[&SERVER=<server_name>][&<custom-option>=<value>[...]]')
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_export');
[DISTRIBUTED BY (<column_name> [, ... ] ) | DISTRIBUTED RANDOMLY];
```

Kafka WRITE. Параметры

Настройка	Описание значения	Обязателен
BOOTSTRAP_SERVERS	Список брокеров Kafka через запятую, каждый из которых является хостом или host:port строкой.	Да
BATCH_SIZE	Определяет сколько строк должно складываться в одно сообщение Avro.	Нет
TOPIC_AUTO_CREATE_FLAG	Позволяет топикам создаваться автоматически, когда в них записывают данные. Топик будет создан с 1 партицией с фактором репликации =1. Значение по умолчанию: true.	Нет
AVRO_DEFAULT_DECIMAL_PRECISION	Максимальное количество знаков в числе. Должно быть положительным числом выше нуля. Значение по-умолчанию: 38.	Нет
AVRO_DEFAULT_DECIMAL_SCALE	Количество знаков после запятой для чисел. Например, число 123.45 имеет DECIMAL_PRECISION 5 и DECIMAL_SCALE 2. Должно быть неотрицательным числом меньше или равным предыдущему параметру. Значение по-умолчанию: 18.	Нет

Kafka WRITE. Соответствие типов

Тип PXF	Примитивный тип AVRO	Логический тип AVRO
BOOLEAN	BOOLEAN	
TEXT	STRING	
VARCHAR	STRING	
TIMESTAMP	LONG	timestamp-micros
BIGINT	LONG	
TIME	LONG	time-micros
NUMERIC	DOUBLE	
FLOAT8	DOUBLE	
REAL	FLOAT	
SMALLINT	INT	
INTEGER	INT	
DATE	INT	date

Kafka WRITE. Пример

```
CREATE WRITABLE EXTERNAL TABLE kafka_tbl (a TEXT, b TEXT, c TEXT)
LOCATION ('pxf://data_from_gp?PROFILE=kafka
&BOOTSTRAP_SERVERS=10.92.8.43:9092,10.92.8.38:9092
&BATCH_SIZE=10') FORMAT 'CUSTOM' (FORMATTER='pxfwritable_export');
```

Kafka READ

В поставке ADB EE присутствует kadb-fdw - расширение, позволяющее производить транзакционную загрузку данных из кластера брокера сообщений Kafka. Устанавливается как сервис в ADCM. Особенности:

- AVRO десереализация.
- Хранение смещений Kafka вне кластера Kafka, на стороне потребителя.
- Поддержка транзакций ADB/GPDB.
- Поддержка Kerberos-аутентификации.

Для работы используется механизм foreign data wrapper.

Требуется создание объектов foreign server и foreign table.

Параметры: https://docs.arenadata.io/adb/adbkafka/Kafka2ADB.html#id4

Kafka READ. Пример

```
-- Create a SERVER
DROP SERVER IF EXISTS ka_server;
CREATE SERVER ka_server
FOREIGN DATA WRAPPER kadb_fdw
OPTIONS (
    k_brokers 'localhost:9092'
);

-- Create a FOREIGN TABLE
DROP FOREIGN TABLE IF EXISTS ka_table;
CREATE FOREIGN TABLE ka_table(field1 INT, field2 TEXT)
SERVER ka_server
OPTIONS (
    format 'avro',  -- Data serialization format
    k_topic 'my_topic',  -- Kafka topic
    k_consumer_group 'my_consumer_group',  -- Kafka consumer group
    k_seg_batch '100',  -- Limit on the number of Kafka messages retrieved by each GPDB segment
    k_timeout_ms '1000',  -- Kafka response timeout
    k_initial_offset '42'  -- Initial Kafka offset (for new or unknown partitions)
);
```

Clickhouse WRITE

В поставке ADB EE сервис, позволяющий производить транзакционную загрузку данных из ADB в кластер Clickhouse. Устанавливается как сервис в ADCM.

Состоит из двух компонентов – модуля PXF, непосредственно занимающегося отправкой данных с сегментов, и расширения ADB, предназначенного для того, чтобы сделать операцию загрузки данных более безопасной при отсутствии транзакций в Clickhouse.

Расширение используется в случаях, когда необходимы дополнительные гарантии консистентности вставки. При этом необходимые настройки кластера Clickhouse автоматически проверяются при попытке вставки, используя функцию расширения:

```
function txn(query text, http_port int default 8123, debug boolean default false,
ending_pattern text default '_tmp_$') returns void;
```

Clickhouse WRITE

Для работы необходимо настроить профиль PXF:

```
<profiles>
    <profile>
        <name>Tkh</name>
        <description>Clickhouse</description>
        <plugins>
            <accessor>io.arenadata.tkh.TkhAccessor</accessor>
            <resolver>io.arenadata.tkh.TkhResolver</resolver>
        </plugins>
        <optionMappings>
            <mapping option="send_threads" property="clickhouse.send.threads"/>
            <mapping option="send_delay" property="clickhouse.send.delay"/>
            <mapping option="send_queue_sizeMultiplier" property="clickhouse.send.queue.sizeMultiplier"/>
            <mapping option="net_timeout" property="clickhouse.network.timeout"/>
        </optionMappings>
    </profile>
</profiles>
```

Clickhouse WRITE

Для создания внешней таблицы, необходимо в LOCATION указать:

- Протокол и таблицу (с указанием базы).
- Имя таблицы Clickhouse.
- SERVER (если не используется дефолтный).
- Дополнительные опции.
- Формат – TEXT.

```
CREATE WRITABLE EXTERNAL TABLE <table_name> ( { <column_name> <data_type> [, ...] | LIKE <other_table> } )
LOCATION ('pxf://<clickhouse_table_name>?<pxf_parameters><settings>')
FORMAT 'TEXT'
ENCODING 'UTF8';
```

Где <pxf_parameters>:

```
{ PROFILE=TKH | ACCESSOR=io.arenadata.tkh.TkhAccessor&RESOLVER=io.arenadata.tkh.TkhResolver }]
```

Все опции:

https://docs.arenadata.io/adb/adbClickhouse/Configuration.html#id4

Clickhouse WRITE. Пример

```
create writable external table ext_tct(a int)
location ('pxf://default.d_tct_tmp_$?profile=tkh&url=default@sdch1:8123')
format 'text' encoding 'utf8';
```

ARENADATA

Корпоративная платформа хранения
и обработки больших данных

Команда COPY

COPY

- Самый низкоуровневый способ импортировать и экспортировать данные.
- Работает как на мастере, так и на сегментах.
- Интегрируется со сторонним ПО.
- Текст и BINARY.
- Может экспортировать результат выполнения запроса, представлений и т.д.

COPY FROM – импорт данных

```
COPY table [(column [, ...])] FROM {'file' | PROGRAM 'command' | STDIN}
     [ [WITH]
       [ON SEGMENT] -- локально или на мастере.
       [BINARY] -- текстово или бинарно.
       [OIDS] -- копировать OID – только для таблиц, содержащих OID каждой строки.
       [HEADER] -- игнорировать первую строку.
       [DELIMITER [ AS ] 'delimiter'] – разделитель.
       [NULL [ AS ] 'null string'] – NULL.
       [ESCAPE [ AS ] 'escape' | 'OFF'] -- escape-символ.
       [NEWLINE [ AS ] 'LF' | 'CR' | 'CRLF'] -- символ перевода строки.
       [CSV [QUOTE [ AS ] 'quote'] -- кавычки, дефолт – двойные кавычки.
            [FORCE NOT NULL column [, ...]] -- пусто вместо NULL.
       [FILL MISSING FIELDS] -- если не хватает столбцов, заполнять NULL.
       [[LOG ERRORS] -- логировать записи, на которых возникла ошибка (только со след. опцией).
      SEGMENT REJECT LIMIT count [ROWS | PERCENT] ] -- допустимое количество записей с ошибками.
```

COPY TO – экспорт данных

```
COPY {table [(column [, ...])] | (query)} TO {'file' | PROGRAM 'command' | STDOUT}
      [ [WITH]
        [ON SEGMENT]
        [BINARY]
        [OIDS]
        [HEADER]
        [DELIMITER [ AS ] 'delimiter']
        [NULL [ AS ] 'null string']
        [ESCAPE [ AS ] 'escape' | 'OFF']
        [CSV [QUOTE [ AS ] 'quote']
             [FORCE QUOTE column [, ...]] ] -- брать в кавычки всё
      [IGNORE EXTERNAL PARTITIONS ] -- если в экспортируемой таблице есть внешние партиции, упасть с
                                       ошибкой или игнорировать их.
```

Нюансы

- **gp_enable_segment_copy_checking** – проверять ли соответствие каждой строки сегменту согласно distributed by(). По умолчанию в true. False производительнее, но возможны неконсистентные данные – надо делать REORGANIZE;
- **gp_read_error_log('schema.table')** – вывести список строк, загруженных с ошибками (можно использовать * для всего). Полезно для дебага внешних источников;
- **gp_truncate_error_log('schema.table')** – очистить список строк, загруженных с ошибками (можно использовать * для всего);
- **gp_initial_bad_row_limit** – если первые N строк ошибочные, падать с ошибкой, независимо от REJECT LIMIT. По умолчанию – 1000;
- Для запросов, представлений и внешних объектов: COPY (SELECT * from my_sales) TO …
- На папки, файлы и программы должны быть даны соответствующие права пользователю gpadmin;
- COPY умеет валидировать XML для типа XML;
- Не супер-пользователи могут читать/писать только в клиентосвкий stdin/stdout;
- При локальном (посегментном) импорте и экспорте в пути до файлов или в строке вызова внешней программы необходимо указыват переменную: <SEGID> (подставляется content ID). Также доступна переменная <SEG_DATA_DIR> (абсолютный путь до папки сегмента);.

GPSSH и GPSCP

- **gpssh** – утилита параллельного SSH по серверам кластера;
  - Опция -f – файл со списком сегмент серверов;
  - Примеры:
    - gpssh -f /home/gpadmin/arenadata_configs/arenadata_all_hosts.hosts
    - gpssh -f /home/gpadmin/arenadata_configs/arenadata_segment_hosts.hosts
    - gpssh -f /home/gpadmin/arenadata_configs/arenadata_all_hosts.hosts id gpadmin
- **gpscp** – утилита параллельного копирования файлов между серверами кластера;
  - Символ ‘=‘ копирует файлы на все сервера;
  - Опция –r рекурсивно копирует директорию;
  - Пример:

  gpscp -f /home/gpadmin/arenadata_configs/arenadata_all_hosts.hosts /localfile =:/remotedir/

Для доступа к утилитам ADB из-под root выполните source /usr/lib/gpdb/greenplum_path.sh

ARENADATA

Корпоративная платформа хранения
и обработки больших данных

Пользовательские функции

Функции

- Функция – Объект, в котором содержится исполняемый алгоритм, описанный на одном из доступных языков.
- Функцию можно выполнять с помощью SQL-запроса. Процедур и пользовательских триггеров в GreenPlum нет.
- Функции принимают параметры.
- Функции возвращаю результат в виде одной или нескольких строк.
- В функциях можно выполнять запросы.
- Каждая функция принадлежит одному из классов:
  - **IMMUTABLE** – Значение функции зависит только от её аргументов (пример: string_agg)
  - **STABLE** – Значение функции может меняться от транзакции к транзакции, но не может меняться в рамках одной транзакции.
  - **VOLATILE** (default) – Значение функции может меняться в ходе выполнения. Используйте такие типы функций осторожно. Чаще всего они выполняются только на мастере.
- Если вы уверены, что ваша функция IMMUTABLE или STABLE – обязательно укажите это.
- Функции могут работать на мастере и на сегментах в зависимости от некоторых факторов.

Место выполнения функции

Функции можно выполнять с помощью двух типов команд:

- Select function() ; - чаще всего будет выполняться на мастере, если не используется параметр EXECUTE ON ALL SEGMENTS.
- Select function(field) from table; - чаще всего будет выполняться на сегментах , если не используется параметр EXECUTE ON. MASTER

Параметр EXECUTE ON указывает, где должна исполняться функция:

Имя атрибута	Описание	Дополнительно
EXECUTE ON ANY	Функция может исполняться как на мастере, так и на сегменте. Возвращает один и тот же результат вне зависимости от места исполнения. Является значением по умолчанию	Greenplum сам определяет где должна исполняться функция
EXECUTE ON MASTER	Функция должна исполняться на мастере	Используйте это значение, если внутри функции используется запрос, который обращается к таблице
EXECUTE ON ALL SEGMENTS	Функция должна исполняться на сегментах	

Функция не может работать на сегментах, если:

- Таблица table распределена по кластеру (не из каталога и не DISTRIBUTED REPLICATED).
- Функция читает или изменяет данные в распределённых таблицах.
- Функция возвращает более одной строки **ИЛИ** принимает на вход столбец из распределённой таблицы.

Функция с запросом может исполняться на сегментах если только читает данные из Replicated-таблиц или из каталога.

Функции: доступные языки программирования

- Процедурные языки (Procedural Language, PL/…) могут быть:
  - Trusted – при использовании языка пользователь не может изменять ФС, выходить в SHELL и тд – не может навредить системе.
  - Untrusted – использование языка может нанести вред СУБД. Создание функций на таких языках доступно только суперпользователям, выполнение – всем.
  - У некоторых языков есть две версии – trusted и untrusted.

- Доступны следующие языки:
  - PL/pgSQL – trusted (установлен по-умолчанию).
  - PL/R – untrusted.
  - PL/Python – untrusted (установлен по-умолчанию).
  - PL/Container (доступны Python и R) – trusted (установлен по-умолчанию).
  - PL/Java – trusted и untrusted.
  - PL/Perl – trusted и untrusted.
  - PL/C – trusted.

Создание функции

```
CREATE [OR REPLACE] FUNCTION name ( [ [argmode] [argname] argtype [ { DEFAULT | = } default_expr ] [, ...] ] )
[ RETURNS rettype | RETURNS TABLE ( column_name column_type [, ...] ) ] – Атомарный тип или таблица
{ LANGUAGE langname
| WINDOW
| IMMUTABLE
| STABLE
| VOLATILE
| CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT
| NO SQL | CONTAINS SQL | READS SQL DATA | MODIFIES SQL
| EXECUTE ON { ANY | MASTER | ALL SEGMENTS | INITPLAN }
| COST execution_cost | SET configuration_parameter { TO value | = value | FROM CURRENT }
| AS 'definition' | AS 'obj_file', 'link_symbol' } ...
```

Создание функции

```
CREATE [OR REPLACE] FUNCTION name ( [ [argmode] [argname] argtype [ { DEFAULT | = } default_expr ] [, ...] ] )
[ RETURNS rettype | RETURNS TABLE ( column_name column_type [, ...] ) ]
{ LANGUAGE langname  – Язык функции
| WINDOW
| IMMUTABLE
| STABLE
| VOLATILE
| CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT
| NO SQL | CONTAINS SQL | READS SQL DATA | MODIFIES SQL
| EXECUTE ON { ANY | MASTER | ALL SEGMENTS | INITPLAN }
| COST execution_cost | SET configuration_parameter { TO value | = value | FROM CURRENT }
| AS 'definition' | AS 'obj_file', 'link_symbol' } ...
```

Создание функции

```
CREATE [OR REPLACE] FUNCTION name ( [ [argmode] [argname] argtype [ { DEFAULT | = } default_expr ] [, ...] ] )
[ RETURNS rettype | RETURNS TABLE ( column_name column_type [, ...] ) ]
{ LANGUAGE langname
| WINDOW – Оконная функция
| IMMUTABLE
| STABLE
| VOLATILE
| CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT
| NO SQL | CONTAINS SQL | READS SQL DATA | MODIFIES SQL
| EXECUTE ON { ANY | MASTER | ALL SEGMENTS | INITPLAN }
| COST execution_cost | SET configuration_parameter { TO value | = value | FROM CURRENT }
| AS 'definition' | AS 'obj_file', 'link_symbol' } ...
```

Создание функции

```
CREATE [OR REPLACE] FUNCTION name ( [ [argmode] [argname] argtype [ { DEFAULT | = } default_expr ] [, ...] ] )
[ RETURNS rettype | RETURNS TABLE ( column_name column_type [, ...] ) ]
{ LANGUAGE langname
| WINDOW
| IMMUTABLE – Класс функции
| STABLE – …
| VOLATILE – …
| CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT
| NO SQL | CONTAINS SQL | READS SQL DATA | MODIFIES SQL
| EXECUTE ON { ANY | MASTER | ALL SEGMENTS | INITPLAN }
| COST execution_cost | SET configuration_parameter { TO value | = value | FROM CURRENT }
| AS 'definition' | AS 'obj_file', 'link_symbol' } ...
```

Создание функции

```
CREATE [OR REPLACE] FUNCTION name ( [ [argmode] [argname] argtype [ { DEFAULT | = } default_expr ] [, ...] ] )
[ RETURNS rettype | RETURNS TABLE ( column_name column_type [, ...] ) ]
{ LANGUAGE langname
| WINDOW
| IMMUTABLE
| STABLE
| VOLATILE
| CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT – Допустимы ли NULL на входе
| NO SQL | CONTAINS SQL | READS SQL DATA | MODIFIES SQL
| EXECUTE ON { ANY | MASTER | ALL SEGMENTS | INITPLAN }
| COST execution_cost | SET configuration_parameter { TO value | = value | FROM CURRENT }
| AS 'definition' | AS 'obj_file', 'link_symbol' } ...
```

Создание функции

```
CREATE [OR REPLACE] FUNCTION name ( [ [argmode] [argname] argtype [ { DEFAULT | = } default_expr ] [, ...] ] )
[ RETURNS rettype | RETURNS TABLE ( column_name column_type [, ...] ) ]
{ LANGUAGE langname
| WINDOW
| IMMUTABLE
| STABLE
| VOLATILE
| CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT
| NO SQL | CONTAINS SQL | READS SQL DATA | MODIFIES SQL – Содержит ли функция запросы в теле
| EXECUTE ON { ANY | MASTER | ALL SEGMENTS | INITPLAN }
| COST execution_cost | SET configuration_parameter { TO value | = value | FROM CURRENT }
| AS 'definition' | AS 'obj_file', 'link_symbol' } ...
```

Создание функции

```
CREATE [OR REPLACE] FUNCTION name ( [ [argmode] [argname] argtype [ { DEFAULT | = } default_expr ] [, ...] ] )
[ RETURNS rettype | RETURNS TABLE ( column_name column_type [, ...] ) ]
{ LANGUAGE langname
| WINDOW
| IMMUTABLE
| STABLE
| VOLATILE
| CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT
| NO SQL | CONTAINS SQL | READS SQL DATA | MODIFIES SQL
| EXECUTE ON { ANY | MASTER | ALL SEGMENTS | INITPLAN } – Место исполнения
| COST execution_cost | SET configuration_parameter { TO value | = value | FROM CURRENT }
| AS 'definition' | AS 'obj_file', 'link_symbol' } ...
```

Создание функции

```
CREATE [OR REPLACE] FUNCTION name ( [ [argmode] [argname] argtype [ { DEFAULT | = } default_expr ] [, ...] ] )
[ RETURNS rettype | RETURNS TABLE ( column_name column_type [, ...] ) ]
{ LANGUAGE langname
| WINDOW
| IMMUTABLE
| STABLE
| VOLATILE
| CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT
| NO SQL | CONTAINS SQL | READS SQL DATA | MODIFIES SQL
| EXECUTE ON { ANY | MASTER | ALL SEGMENTS | INITPLAN }
| COST execution_cost | SET configuration_parameter { TO value | = value | FROM CURRENT } – Cost для планов
| AS 'definition' | AS 'obj_file', 'link_symbol' } ...
```

Создание функции

```
CREATE [OR REPLACE] FUNCTION name ( [ [argmode] [argname] argtype [ { DEFAULT | = } default_expr ] [, ...] ] )
[ RETURNS rettype | RETURNS TABLE ( column_name column_type [, ...] ) ]
{ LANGUAGE langname
| WINDOW
| IMMUTABLE
| STABLE
| VOLATILE
| CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT
| NO SQL | CONTAINS SQL | READS SQL DATA | MODIFIES SQL
| EXECUTE ON { ANY | MASTER | ALL SEGMENTS | INITPLAN }
| COST execution_cost | SET configuration_parameter { TO value | = value | FROM CURRENT }
| AS 'definition' | AS 'obj_file', 'link_symbol' } ... – Определение функции (описание или ссылка)
```

Функции: примеры

```
CREATE FUNCTION add(integer, integer) RETURNS integer
AS 'select $1 + $2;'
LANGUAGE SQL
IMMUTABLE
RETURNS NULL ON NULL INPUT;
--
CREATE OR REPLACE FUNCTION t1_calc( name text) RETURNS
integer
AS $$
DECLARE
t1_row table1%ROWTYPE;
calc_int table1.f3%TYPE;
BEGIN
SELECT * INTO t1_row FROM table1 WHERE table1.f1 = $1 ;
calc_int = (t1_row.f2 * t1_row.f3)::integer ;
RETURN calc_int ;
END;
$$ LANGUAGE plpgsql VOLATILE;
--
```

```
CREATE FUNCTION add_two_values(v1 anyelement,v2
anyelement)
RETURNS anyelement AS $$
DECLARE
sum ALIAS FOR $0;
BEGIN
sum := v1 + v2;
RETURN sum;
END;
$$ LANGUAGE plpgsql;
--
CREATE OR REPLACE FUNCTION run_on_segs (text)
RETURNS setof text as $$
BEGIN
RETURN NEXT ($1 || ' - ' || timeofday()::text );
END;
$$ LANGUAGE plpgsql VOLATILE EXECUTE ON ALL SEGMENTS;
```

Функции: анонимные блоки

Анонимные блоки – выполняемый код, не сохраненный в виде объекта.

```
DO $$
DECLARE
t1_row table1%ROWTYPE;
calc_int table1.f3%TYPE;
BEGIN
SELECT * INTO t1_row FROM table1, list WHERE table1.f1 = list.column1 ;
calc_int = (t1_row.f2 * t1_row.f3)::integer ;
RAISE NOTICE 'calculated value is %', calc_int ;
END $$ LANGUAGE plpgsql ;
```

Функции: словари для обмена данных (Python)

- SD – глобальный словарь для обмена данными между вызовами функций
- GD – глобальный словарь для обмена данными между всеми функциями

```
CREATE FUNCTION usesavedplan() RETURNS trigger AS $$
if SD.has_key("plan"):
plan = SD["plan"]
else:
plan = plpy.prepare("SELECT 1")
SD["plan"] = plan
$$ LANGUAGE plpythonu;

CREATE FUNCTION pytest() returns text as $$
if 'mymodule' not in GD:
import mymodule
GD['mymodule'] = mymodule
return GD['mymodule'].sumd([1,2,3])
$$ LANGUAGE plpythonu;
```

Лабораторная 6: функции (1)

1. Создайте функцию:

```
create or replace function get_host_pyt() returns text
as $$
import socket
return 'I am running on host: ' + socket.gethostname()
$$
volatile
language plpythonu execute on all segments;
```

2. Выполните запрос:

```
select get_host_pyt();
```

3. В функции get_host_pyt() поменяйте «execute on all segments» на «execute on master»

4. Выполните запрос:

```
select get_host_pyt();
```

Лабораторная 6: функции (2)

1. Создайте функцию:

```
create or replace function get_host_cont() returns text
as $$
# container: plc_py
import socket
return 'I am running on host: ' + socket.gethostname()
$$
volatile
language plcontainer execute on all segments;
```

2. Выполните запрос:

```
select get_host_cont();
```

3. В функции get_host_cont() поменяйте «execute on all segments» на «execute on master»

4. Выполните запрос:

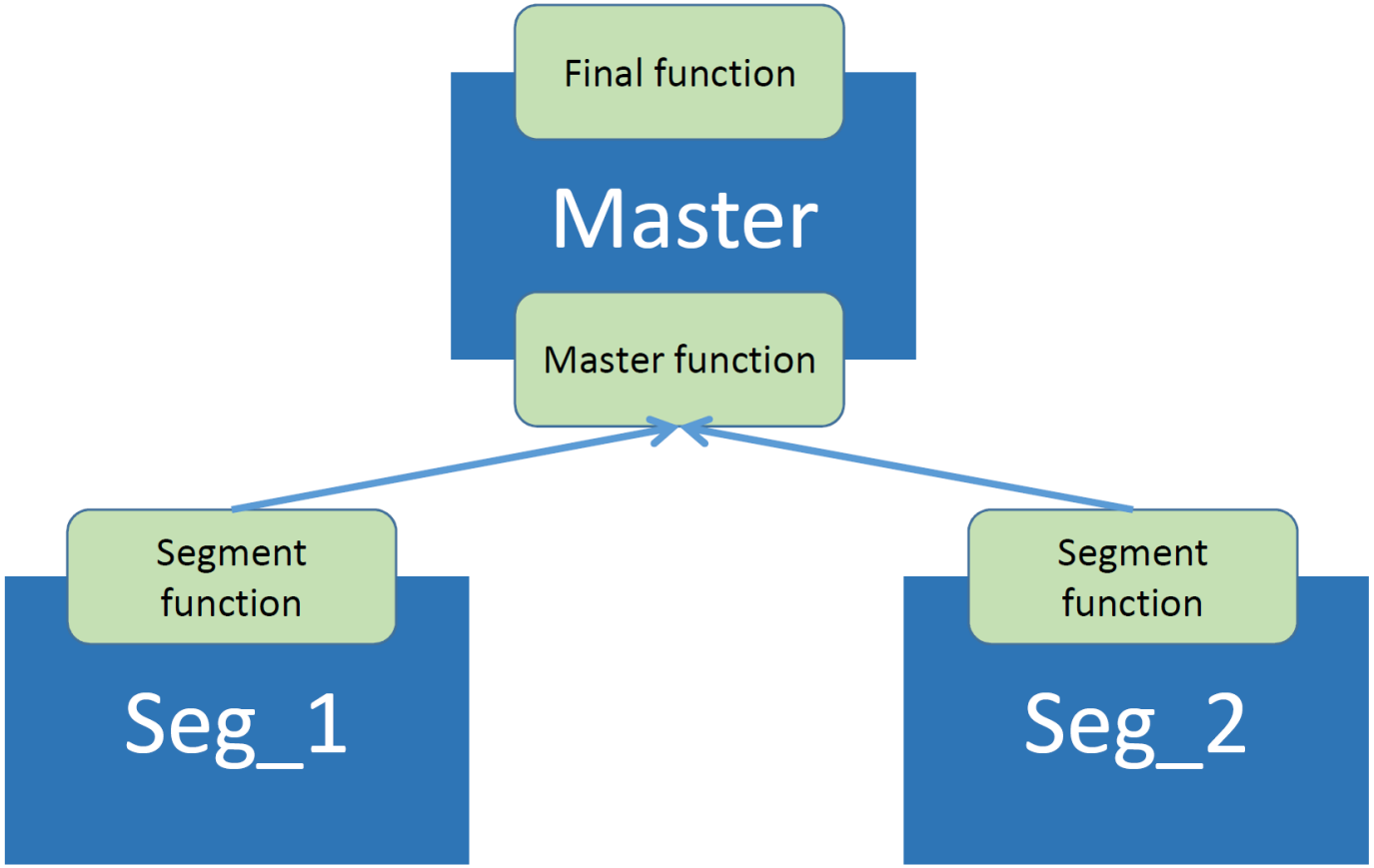
```
select get_host_cont();
```

ARENADATA

Корпоративная платформа хранения
и обработки больших данных

Пользовательские агрегатные функции

Пользовательские агрегатные функции



Пользовательские агрегатные функции

```
drop table if exists dist3;
create table dist3 as (select * from generate_series(1,8) x) distributed randomly;

CREATE OR REPLACE FUNCTION segment_sfunc(int1 int, int2 int)
  RETURNS int
   AS $$
    import socket
    var1=int1
    var2=int2
    if var1 == 0:
        var1 = 1
    if var2 == 0:
        var2 = 1
    return var1 *  var2
   $$
   LANGUAGE plpythonu
   IMMUTABLE
   RETURNS NULL ON NULL INPUT;

CREATE OR REPLACE FUNCTION master_prefunc(int1 int, int2 int)
  RETURNS int
   AS $$
    import socket
    var1=int1
    var2=int2
    return   var1 + var2
   $$
   LANGUAGE plpythonu
   IMMUTABLE
   RETURNS NULL ON NULL INPUT;
```

gp_segment_id	x
0	6
0	2
1	5
1	1
2	8
2	4
3	7
3	3

Пользовательские агрегатные функции

```
CREATE OR REPLACE FUNCTION final_func(int)
  RETURNS int
   AS $$
   select $1*1
   $$
   LANGUAGE SQL
   IMMUTABLE
   RETURNS NULL ON NULL INPUT;

  DROP    AGGREGATE agg_func(int);
  CREATE AGGREGATE agg_func(int) (
   SFUNC = segment_sfunc,
   STYPE = int,
   FINALFUNC = final_func,
   PREFUNC = master_prefunc,
   INITCOND = 0 );

set optimizer=on;
 select agg_func(x::int) as agg, sum(x) as sum from dist3;
```

agg	sum
70	36

В текущей версии ADB базовыми компонентами агрегатной функции являются:

- Базовая функция *statefunc.*
- Опциональная финальная функция *ffunc.*
- Опциональная комбинирующая функция *combinefunc.*

Функции: кастомные агрегаты

```
CREATE AGGREGATE name ( [ argmode ] [ argname ]
arg_data_type [ , ... ] ) (
    SFUNC = statefunc,
    STYPE = state_data_type
    [ , SSPACE = state_data_size ]
    [ , FINALFUNC = ffunc ]
    [ , FINALFUNC_EXTRA ]
    [ , COMBINEFUNC = combinefunc ]
    [ , SERIALFUNC = serialfunc ]
    [ , DESERIALFUNC = deserialfunc ]
    [ , INITCOND = initial_condition ]
    [ , MSFUNC = msfunc ]
    [ , MINVFUNC = minvfunc ]
    [ , MSTYPE = mstate_data_type ]
    [ , MSSPACE = mstate_data_size ]
    [ , MFINALFUNC = mffunc ]
    [ , MFINALFUNC_EXTRA ]
    [ , MINITCOND = minitial_condition ]
    [ , SORTOP = sort_operator ]
  )
```

```
CREATE AGGREGATE name ( [ [ argmode ] [ argname ]
arg_data_type [ , ... ] ]
      ORDER BY [ argmode ] [ argname ] arg_data_type [ , ...
] ) (
    SFUNC = statefunc,
    STYPE = state_data_type
    [ , SSPACE = state_data_size ]
    [ , FINALFUNC = ffunc ]
    [ , FINALFUNC_EXTRA ]
    [ , COMBINEFUNC = combinefunc ]
    [ , SERIALFUNC = serialfunc ]
    [ , DESERIALFUNC = deserialfunc ]
    [ , INITCOND = initial_condition ]
    [ , HYPOTHETICAL ]
  )
```

ARENADATA

Корпоративная платформа хранения
и обработки больших данных

Встроенная аналитика на основе MADLib

MADLIB

- Apache MADLIB – open source набор библиотек, функций и методов для анализа данных.
- MADLIB интегрируется в СУБД – для анализа данные не нужно выгружать и загружать.
- Доступ ко всем функциям MADLIB осуществляется через SQL.
- Внутри – Python и C-функции.

MADLIB. Пример – association rule

```
CREATE TABLE test_data (
   trans_id INT,
   product text
);

INSERT INTO test_data VALUES -- Номер транзакции и купленный товар
   (1, 'beer'),
   (1, 'diapers'),
   (1, 'chips'),
   (2, 'beer'),
   (2, 'diapers'),
   (3, 'beer'),
   (3, 'diapers'),
   (4, 'beer'),
   (4, 'chips'),
   (5, 'beer'),
   (6, 'beer'),
   (6, 'diapers'),
   (6, 'chips'),
   (7, 'beer'),
   (7, 'diapers');
```

MADLIB. Пример – association rule

```
SELECT * FROM madlib.assoc_rules (
   .40,          -- support
   .75,          -- confidence
   'trans_id',   -- transaction column
   'product',    -- product purchased column
   'test_data',  -- table name
   'public',     -- schema name
   false);       -- display processing details

output_schema | output_table | total_rules | total_time
--------------+--------------+-------------+-----------------
public        | assoc_rules  |           2 | 00:00:01.153283

SELECT pre, post, support FROM assoc_rules
   ORDER BY support DESC;

    pre    |  post  |      support
-----------+--------+-------------------
 {diapers} | {beer} | 0.714285714285714
 {chips}   | {beer} | 0.428571428571429
```

ARENADATA

Корпоративная платформа хранения
и обработки больших данных

Работа с географическими данными и объектами с помощью PostGIS

PostGIS

- PostGIS – расширение, позволяющее хранить и обрабатывать географические объекты (координаты, фигуры и т.д.)
- Используются специальные индексы – GIST.

PostGIS. Примеры объектов

```
CREATE TABLE geom_test ( gid int4, geom geometry, name varchar(25) );

INSERT INTO geom_test ( gid, geom, name ) VALUES ( 1, 'POLYGON((0 0 0,0 5 0,5 5 0,5 0 0,0 0 0))', '3D Square');

INSERT INTO geom_test ( gid, geom, name ) VALUES ( 2, 'LINESTRING(1 1 1,5 5 5,7 7 5)', '3D Line' );

INSERT INTO geom_test ( gid, geom, name ) VALUES ( 3, 'MULTIPOINT(3 4,8 9)', '2D Aggregate Point' );

SELECT * from geom_test WHERE geom &&Box3D(ST_GeomFromEWKT('LINESTRING(2 2 0, 3 3 0)'));
```

ARENADATA

Корпоративная платформа хранения
и обработки больших данных

Перечень дополнительных модулей

Дополнительные модули

auto_explain – позволяет логировать планы запросов без выполнения команды EXPLAIN или EXPLAIN ANALYZE;

citext – модуль предоставляет тип данных для строк, нечувствительных к регистру, citext. По сути он сравнивает значения, вызывая внутри себя функцию lower. В остальном он почти не отличается от типа text;

dblink — выполняет запрос в удалённой базе данных;

diskquota – ограничивает общий размер схем и таблиц;

fuzzystrmatch - содержит несколько функций для вычисления схожести и расстояния между строками;

gp_sparse_vector - модуль реализует тип данных базы данных Greenplum и связанные функции, которые используют сжатое хранилище нулей для ускорения векторных вычислений с числами с плавающей запятой

hstore - модуль реализует тип данных hstore для хранения пар ключ/значение внутри одного значения. Это может быть полезно в самых разных сценариях, например для хранения строк со множеством редко анализируемых атрибутов или частично структурированных данных. Ключи и значения задаются простыми текстовыми строками.

orafce - модуль предоставляет возможность использовать функции Oracle в базе данных Greenplum.

Дополнительные модули

pageinspect - предоставляет функции, позволяющие исследовать страницы баз данных на низком уровне, что бывает полезно для отладки. Исключение составляют append-optimized таблицы, которые нельзя исследовать с помощью данного модуля;

pgcrypto – модуль, который предоставляет криптографические функции. Функции pgcrypto позволяют администраторам баз данных хранить определенные столбцы данных в зашифрованном виде. Это добавляет дополнительный уровень защиты для конфиденциальных данных, поскольку данные, хранящиеся в базе данных Greenplum в зашифрованном виде, не могут быть прочитаны кем-либо, у кого нет ключа шифрования, и не могут быть прочитаны непосредственно с дисков;

sslinfo - модуль предоставляет информацию о сертификате SSL, предоставленном текущим клиентом при подключении к Greenplum. Большинство функций в этом модуле возвращают NULL, если текущее соединение не использует SSL

Конец четвертой части