**Query replicator tool for Greenplum: User guide**

Prepared by : EMEA PDE team

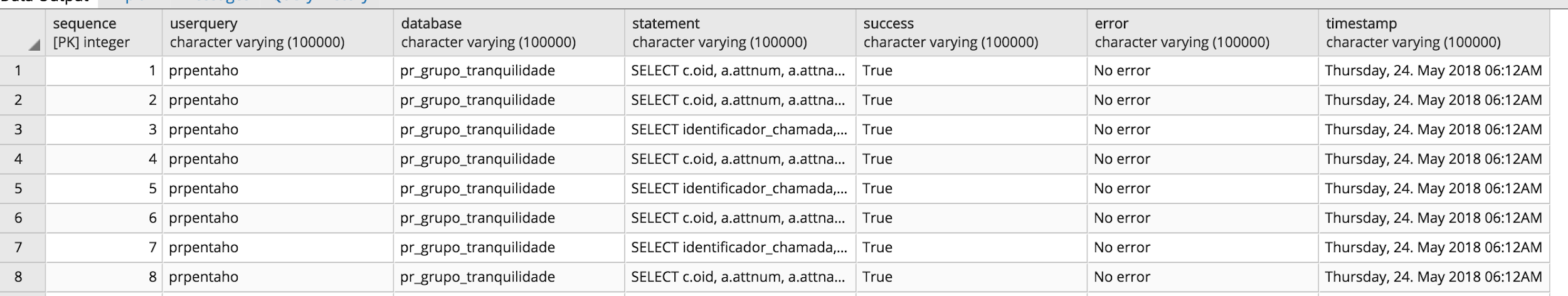
Date : May 2018

Version: 1.0

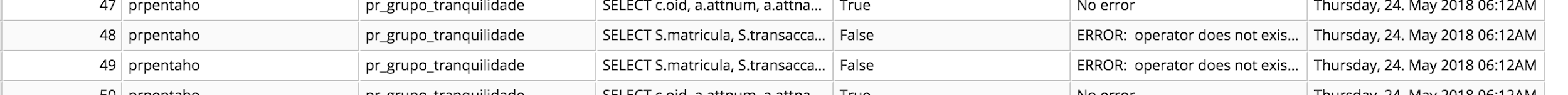
# **Description**

This tool, written in Python (<https://gitlab.com/DanielePalaia/QueryReplicator>) was created by the EMEA PDE team to help clients migrate from different Greenplum versions. The tool is a query replicator tool: From logs of a previous version of Greenplum it is able to re-inject the queries executed to another newer instance of Greenplum.

From a set of Greenplum log files it is able to produce an output table like this:

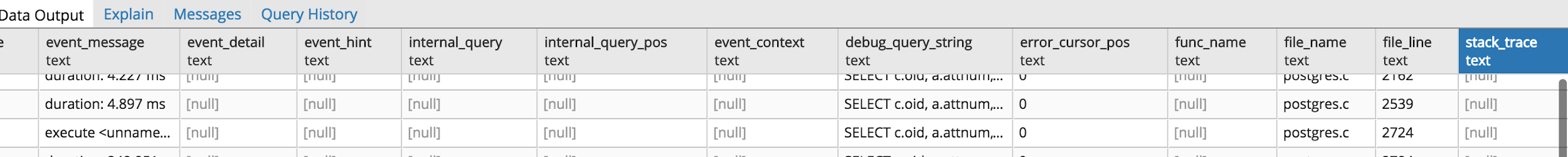


Where for every replicated query we have informations about: the user and the database against which the query was executed, if it was a success or not, and in this last case the error produced like in this case:



The timestamp provide information on the time when the execution was submitted.

**Important note**: The tool does an intermediary step: From log files, it creates an input table in this format, in a way that log informations are parsed and stored as a table:



The tool lets you filter databases you don’t want to parse (you can choice the database to be scanned), also let you define queries that you don’t want to be scanned. As will be explained better before these queries need to be put inside another table of the excluded queries.

# **Filtering**

At the moment the tool is scanning just SELECT and CREATE tmp queries. This was done to avoid testing destructive queries and to start with something simple, other kind of queries can be added. In the database.py file there is at the moment

**statement = '\'select%\''**

**statement2 = '\'create temp''%\''**

It is possible to add other queries to be executed like ALTER, DROP ecc…

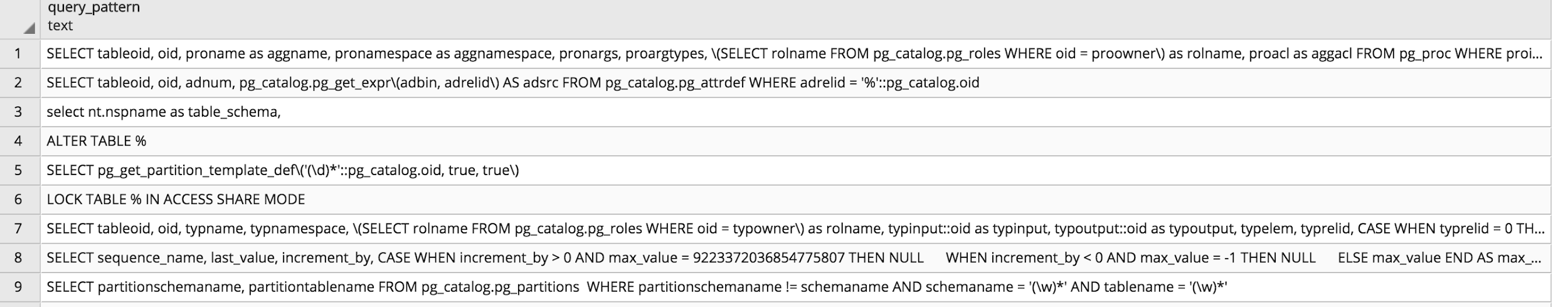
Input log databases can also be filtered. In the file properties.py there is an option

**filterInputDatabases='database1, database2'**

That you can fill putting just the database you want to parse from the input table separated by coma, all the other will be excluded.

There is also the possibility to exclude queries similar to others. During the initialization of the tool (explained afterward), a table which contains queries we want to exclude by similarity will be created. You can just add this table with the queries you want to exclude like this.

If the table is empty no exclusions will be made.



Next paragraph of this document will explain how to set-up the environment and run the tool

## **Set-up the properties configuration files**

In the file properties.py there are some informations which needs to be provided:

*# the location where Greenplum logs are stored (to select all files \*.cvs can be put, zip files are supported as well)*

**str\_Input\_File**=**'/datassdb/gpdb5/master/gpseg-1/pg\_log/gpdb-2018-05-27\_000000.csv'**

*# False if log files are .cvs, True if they are compressed*

**flag\_Compress**=False

*# name of the input table to create and fill from logs*

**logDatabaseTable** = **'public.read\_gpdb\_log\_replay'**

*# input database info*

**logDatabaseIPAddress**= **'10.91.51.23'**

**logDatabaseUsername**= **'gpadmin'**

**logDatabasePort** = 5533

**logDatabaseName** = **'test\_gpdb5'**

*# in this table we need to put the table where we put queries that needs to be excluded from the scanner*

**excludedQueryTable** = **'public.t\_list\_excluded\_query'**

*# output database info (where to inject the input queries)*

**outputDatabaseIPAddress**= **'10.91.51.23'**

**outputDatabaseUsername**= **'gpadmin'**

**outputDatabasePort** = 5533

*# input database filtering (just execute queries for the databases specified separated by coma)*

**filterInputDatabases**=**''**

*# name of the table where you want to store the results of the Query replicator tool computation*

**outputTableName**=**"results"**

## **Create the infrastructure**

From the log files we have we need to:

1. Create the input table as already described which is a table which represent parsed logs and contains log information, the log location needs to be put in the properties.py file
2. Create or Reset the output table which will contain the result after the scanning process
3. A table where if you want you can insert similar queries which can be excluded from the scanning (accordingly with the **similar** Greenplum command)

This can be done with a single parameter **-i** that stands for “initialize”: Run the tool in this way

**python queryreplicator.py -i**

After this operation the 3 tables should have been created.

## **Run the scanning process**

You can just run:

**python queryreplicatortool.py**

The scanning should start, you should see in the terminal the query tested and if it was a success or not

SELECT c.oid, a.attnum, a.attname, c.relname, n.nspname, a.attnotnull OR (t.typtype = 'd' AND t.typnotnull), pg\_catalog.pg\_get\_expr(d.adbin, d.adrelid) LIKE '%nextval(%' FROM pg\_catalog.pg\_class c JOIN pg\_catalog.pg\_namespace n ON (c.relnamespace = n.oid) JOIN pg\_catalog.pg\_attribute a ON (c.oid = a.attrelid) JOIN pg\_catalog.pg\_type t ON (a.atttypid = t.oid) LEFT JOIN pg\_catalog.pg\_attrdef d ON (d.adrelid = a.attrelid AND d.adnum = a.attnum) JOIN (SELECT 133305763 AS oid , 10 AS attnum UNION ALL SELECT 133305763, 2 UNION ALL SELECT 133305763, 5) vals ON (c.oid = vals.oid AND a.attnum = vals.attnum)

**SUCCESS**

After the process is terminated, in the table you specified in **outputTableName** of the properties.py file you should be able to see the results of the computation, you can use pgadmin to have a good view.

## **Statistics**

You can use the option -s (statistics) to see the results in **outputTableName**

**STATISTICS BASED ON OUTPUT TABLE: resultdemo**

**TOTAL QUERY EXECUTED: 224**

**SUCCESSED QUERIES: 206**

**UNSUCCESSED QUERIES 18**

And that’s it but there are other useful options.

## **Other options:**

**Python queryreplicator.py -o**: **Output**, create or reset the output result table **outputTableName**

**Python queryreplicator.py -l**: **Load**, destroy the input table and load logs file into a new input table

**Python queryreplicator.py -r:** **Retry** just the failed queries inside **outputTableName** and retest them

**Python queryreplicator.py -e:** **Excluded**. Creates the excluded query table (where we want to skip the similar query accordingly the SIMILAR query command)

**Python queryreplicator.py -h:** **Help** command