MigOps

Migration validation made easy with Ora2Pg

We are going to cover

- Validation of data type.
- Validation of migrated objects.
- Validation of data.
- Validation of stored procedures.

1.
Introducing

Presentation

Author : Gilles Darold (CTO at MigOps Inc)

Email : gilles@migops.com

MigOps Inc

Company specialized in the migration to PostgreSQL

- ▷ Sponsorize the development of Ora2Pg
- ▷ Commercial support for Ora2Pg and migration.

Contact : https://www.migops.com/contact-us/

Celebration

Happy birthday Ora2Pg !

20 years

First version 05 mai 2001

Version 23.0 released 15 novembre 2021

Migration to PostgreSQL

The Steps

Steps of a migration

Assessment/Analyze	Analysis of the feasibility and the migration effort
Migration	Implementation of tasks deduced from the analysis, migration of the schema, data, SQL, stored procedures and the application
Testing	Testing of migrated objects and data, testing of the application, batches and the complete workflow
Performances	Analyze performance issues and bring fixes, either at SQL, PostgreSQL or application level
Training	Teams must be trained in the new RDBMS according to the needs of the company
Support	24/7 support for incident resolution, operational implementation assistance or response to operational questions

Testing

This is the key to the success of your migration

- Test, test and test again!

Take the opportunity to integrate more unit tests

Validate the steps to switchover in production several times

2.
Tests on the objects

Type of objects

TYPES

SEQUENCES

TABLES

INDEXES

CONTRAINTES

TRIGGERS

VIEWS

MATERIALIZED VIEWS

PARTITIONS

FONCTIONS

PROCEDURES

TABLESPACES

PACKAGES => SCHEMA

DBLINKS => dblink/oracle_fdw

SYNONYMS => VIEWS

JOBS => pgcron/pg_dbms_job

Validation of data type

Loading part of the data makes it possible to detect errors. To load a limited amount of data:

WHERE ROWNUM < 10000

- Problems of BIGINT vs NUMERIC
- RAW(16) ou RAW(32) vs Uuid
- Translation to boolean
- Column varchar() with length limit
- Special case of date vs timestamp

Objects count

```
ora2pg -c config/ora2pg.conf -t TEST > test_objects.log
```

Principle :

- Simultaneous connections on the Oracle and the PostgreSQL database
- Extraction and counting of each type of object
- Comparison between the two extractions and status
- Report errors if there are any

Objects count

- TABLES
- TRIGGERS
- VIEWS
- SEQUENCES with LAST_VALUE check
- Users data types
- EXTERNAL TABLE (ALL_EXTERNAL_TABLE vs FOREIGN TABLE)

Global count of the number of functions:

- PACKAGES
- FONCTIONS
- PROCEDURES

Count per table

- INDEXES
- UNIQUE CONTRAINTS
- PRIMARY KEYS
- CHECK CONTRAINTS
- NOT NULL CONTRAINTS
- COLUMNS with DEFAULT VALUE
- IDENTITY COLUMN
- FOREIGN KEYS
- TRIGGERS
- PARTITIONS

Examples

Example of the TEST action with the migration of the HR database

https://www.ora2pg.com/TEST_example.txt

Some errors generated by the drop of some constraints in the destination database

https://www.ora2pg.com/TEST_example_error.txt

Checking the number of lines

ora2pg -c config/ora2pg.conf -t TEST **--count_rows**

Count the number of rows in each table while counting objects.

Dedicated action to only count the lines:

ora2pg -c config/ora2pg.conf -t **TEST_COUNT**

(useful after a second data import)

Example

```
[TEST ROWS COUNT]
ORACLE:actor:200
POSTGRES:actor:200
ORACLE:address:603
POSTGRES:address:603
ORACLE:film_actor:5462
POSTGRES:film_actor:5462
ORACLE:film_category:1000
POSTGRES:film_category:1000
ORACLE:film_text:1000
POSTGRES:film_text:1000
(...)
[ERRORS ROWS COUNT]
OK, Oracle and PostgreSQL have the same number of rows.
```

3.
Test of views

Checking views

ora2pg -c config/ora2pg.conf -t **TEST_VIEW**

Counts the number of rows returned by each view

No control of the returned data, only the number of lines.

Application-level validation or unitary tests are required.

Example

```
[UNITARY TEST OF VIEWS]
ORACLE:actor_info:200
POSTGRES:actor_info:200
ORACLE:customer_list:599
POSTGRES:customer_list:599
ORACLE:film_list:997
POSTGRES:film_list:997
ORACLE:nicer_but_slower_film_list:997
POSTGRES:nicer_but_slower_film_list:997
ORACLE:sales_by_film_category:16
POSTGRES:sales_by_film_category:16
ORACLE:sales_by_store:2
POSTGRES:sales_by_store:2
ORACLE:staff_list:2
POSTGRES:staff_list:2
```

4.

Test of Data

New since version 23.0 of Ora2Pg

Data migration time

Reduce the cut-off window necessary for the switch to production.

- Test data migration time with options:
  - -P : number of tables exported in parallel
  - -J : number of parallel Oracle process for one table
  - -j : number write process into PostgreSQL per table.

- With and without oracle_fdw (optimum for BLOB with -J)
- Use LOAD action with -j option to import indexes/constraints
- Separate archived data and “live” data for TB databases

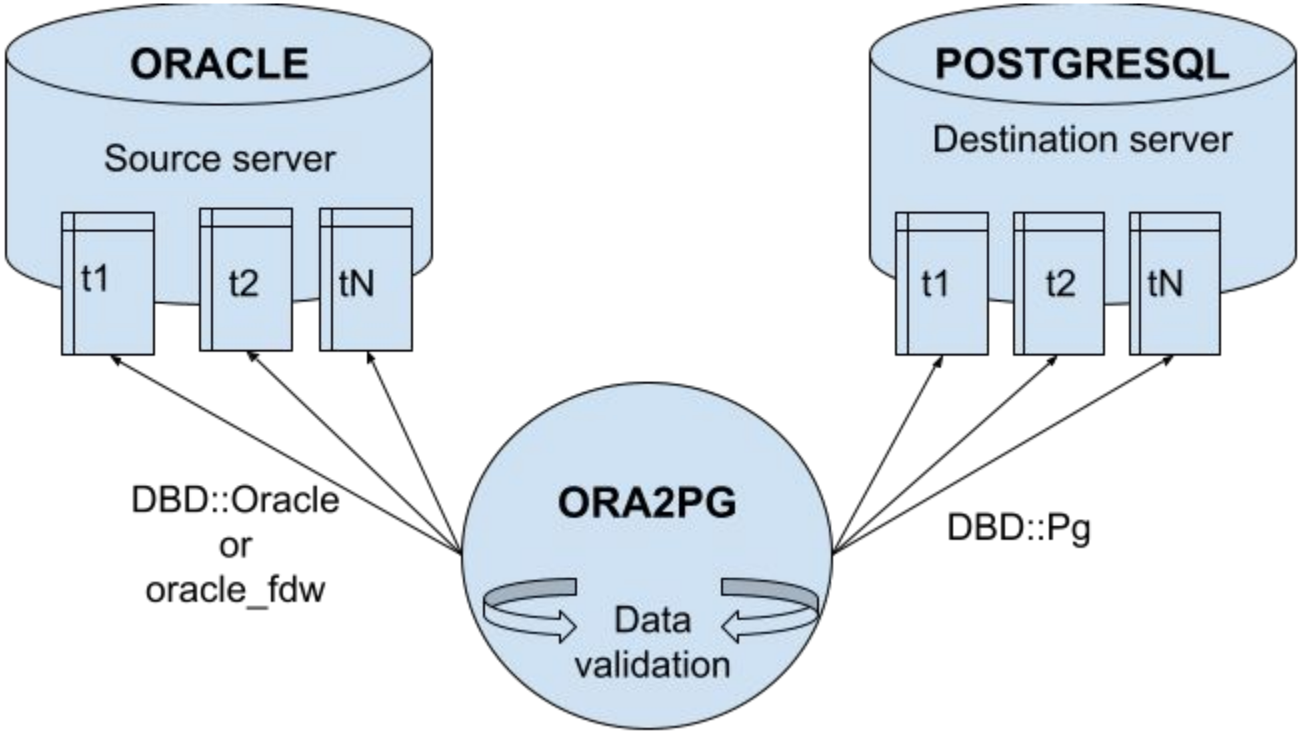
Data validation

ora2pg -c config/ora2pg.conf -t **TEST_DATA**

Checks the values returned by the two DBMSs row by row.

It uses Foreign Data Wrapper or a direct connection.

A WHERE clause can be applied following the imported data



Data validation - TEST_DATA

Prerequisites

Make sure that the columns and their data types in the source and the destination database match.

- Table with primary or unique key for ORDER BY, except initial loading without parallelism
- Collation ‘C’ for non numeric unique keys in PostgreSQL
- No data modification on both side during the check

Data validation

The result of the data validation is stored in a dedicated file : data_validation.log.

In the current directory or in the one specified using option -b | --basedir

The errors reported are limited to 10 before stopping the check for a table in error.

Data validation can be parallelized using option -P | --parallel

Settings

FDW_SERVER	Name of the foreign server to connect to Oracle. If not defined use a direct connection to query the tables.
PG_DSN	Connection settings to the PostgreSQL database
DATA_VALIDATION_ROWS	Maximum number of lines to test. Default: 10000 A value of 0 causes the validation of all rows in the tables
DATA_VALIDATION_ERROR	By default, the data check of a table stops after 10 faults. This number can be increased if you want to treat more error in one pass.
PARALLEL_TABLES	Parallelize data checking by table, uses only 1 process by default.
DATA_VALIDATION_ORDERING	Sorts the data by a unique key, only table with such a key are checked. If disabled, no sorting.

Data validation

Limit:

- ▷ No multi-schema validation, only schema by schema.
- ▷ No user defined type data validation (for the moment)
- ▷ No partition by partition check, only the partitioned table.
- ▷ No data validation of views

5.
Differences in structure

How about definition changes ?

When checking Ora2Pg natively supports changes of:

- Destination schema name (PG_SCHEMA)
- Tables renaming (REPLACE_TABLES)
- Columns renaming (REPLACE_COLS)
- Drop of columns (MODIFY_STRUCT)

Example of definition change

Table renaming :

- ▷ REPLACE_TABLES PRODUCT_TMP:PRODUCT2

Column renaming :

- ▷ REPLACE_COLS RAW_INFO(UID_COL:COL_UID)

Unexported column during the migration :

- ▷ MODIFY_STRUCT RAW_INFO(ID,UID_COL,INFO_COL)

(there is a 4th column named ACTIVE in the source database)

How about data type differences

When checking Ora2Pg natively supports changes of data types:

- To boolean (REPLACE_AS_BOOLEAN and BOOLEAN_VALUES)
- The translation of RAW(16) and RAW(32) in uuid (default)
- Remapping of data types translation (DATA_TYPE)

6.
Stored procedures

Test of procedures

Load functions and procedures one by one, correcting potential syntax errors.

- PostgreSQL check the code at execution time
- No precompiled or invalid code like in Oracle
- Check the stored procedures with plpgsql_check
- Found solution for Oracle DBMS modules

plpgsql_check

```
hr=# CREATE EXTENSION plpgsql_check;
LOAD
hr=# --Check all plpgsql functions in the hr schema
hr=# SELECT p.oid, p.proname, plpgsql_check_function(p.oid)
     FROM pg_catalog.pg_namespace n
     JOIN pg_catalog.pg_proc p ON pronamespace = n.oid
     JOIN pg_catalog.pg_language l ON p.prolang = l.oid
     WHERE l.lanname = 'plpgsql' AND n.nspname = 'hr'
          AND p.prorettype <> 2279; /* no trigger function */
```

Execution performances

Some procedures, best in Oracle, may perform poorly in PostgreSQL.

- Detect the source of performance problems with plprofiler
- Review the logic of the procedure to optimize it.
- pldebugger : PostgreSQL pl/pgsql Debugger API

Unitary tests

Check that the results are identical between the two DBMS

Guarantee the stability of the code during the migration and after.

Tools:

- ▷ Test scripts using psql and sqlplus
- ▷ Test scripts using Perl DBD::Pg and DBD::Oracle
- ▷ Same using JDBC
- ▷ pgTap, Junit, etc.

Perl test script

```
use Test::Simple tests => 1;
use DBI;

# Test function addition(int, int)
my $dbh = DBI->connect("dbi:Pg:dbname=hr;host=192.168.1.10", 'hr', 'pwd');
my $sth = $dbh->prepare( "SELECT addition(100, 45)" );
$sth->execute();
my @row = $sth->fetchrow;
$sth->finish();
ok($row[0] == 145, "Test function addition(int, int)");
```

pgTap

```
\set account_id 32
\set expire_days 60
BEGIN;
SELECT ok(    update_user_account(:account_id::integer, expire_days::integer ),
                          'Call procedure update_user_account' );
-- Check modifications
PREPARE account_expiration_check AS select expire_days, account_id from accounts where account_id
= :account_id::integer;
PREPARE account_expiration_results AS select :expire_days::integer, :account_id::integer;
SELECT results_eq(
   'account_expiration_check',
   'account_expiration_results',
   'Expiration day should be set for account' );
ROLLBACK;
```

Thanks !

Any questions?

http://www.ora2pg.com/

Post your bug reports, feature requests, contribution:

https://github.com/darold/ora2pg