Microsoft

Oracle to Azure Database for PostgreSQL Migration Cookbook

Prepared by

DM Jumpstart Engineering Team (askdmjfordmtools@microsoft.com)

Disclaimer

The High-Level Architecture, Migration Dispositions and guidelines in this document is developed in consultation and collaboration with Microsoft Corporation technical architects. Because Microsoft must respond to changing market conditions, this document should not be interpreted as an invitation to contract or a commitment on the part of Microsoft.

Microsoft has provided generic high-level guidance in this document with the understanding that MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THE INFORMATION CONTAINED HEREIN.

This document is provided "as-is". Information and views expressed in this document, including URL and other Internet Web site references, may change without notice.

Some examples depicted herein are provided for illustration only and are fictitious. No real association or connection is intended or should be inferred.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2019 Microsoft. All rights reserved.

Note: The detail provided in this document has been harvested as part of a customer engagement sponsored through the Azure Data Services Jumpstart Program.

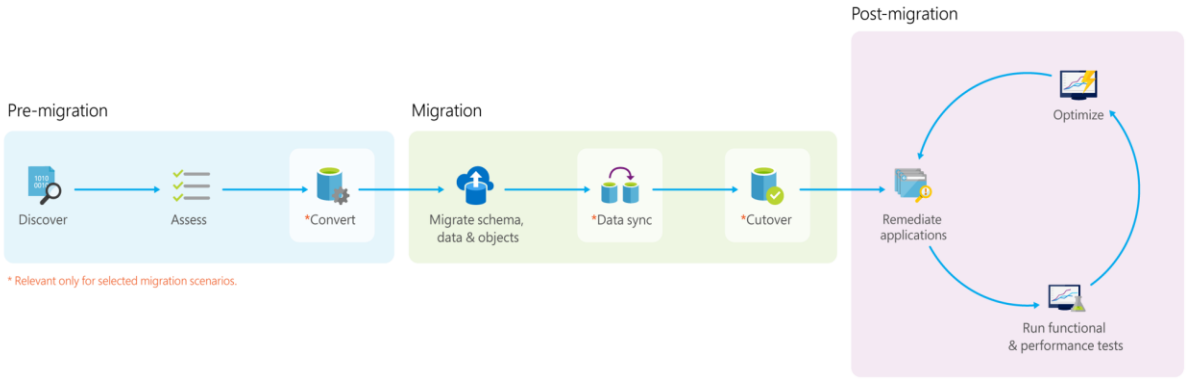
Table of Contents

Introduction

This document purpose is to provide Architects, Consultants, DBAs and related roles with a guide for quick migrating workloads from Oracle to Azure Database for PostgreSQL using ora2pg tool.

1 High Level Migration Approach

Replacing an Oracle database backend with an Azure Database for PostgreSQL backend is a simple exercise that can be executed using the following steps:



Discover: Customers discover all the databases they own, instances hosting them, source database type and version, other database metadata, etc.

Assess: In this phase, the goal is to identify which Azure database target a customer can migrate their database to and how much work is involved.

1. Target and SKU recommendation – Identifies which Azure database target and SKU is right for a database.
2. Readiness assessment – Compares the features used on the source vs the target and give an overview of work involved in migrating a database to Azure.
3. Performance assessment – Compares the query correctness and performance of a database on premise and in Azure.

Convert: Convert the schema from the source database type to target database type. This phase is valid only for heterogenous migrations.

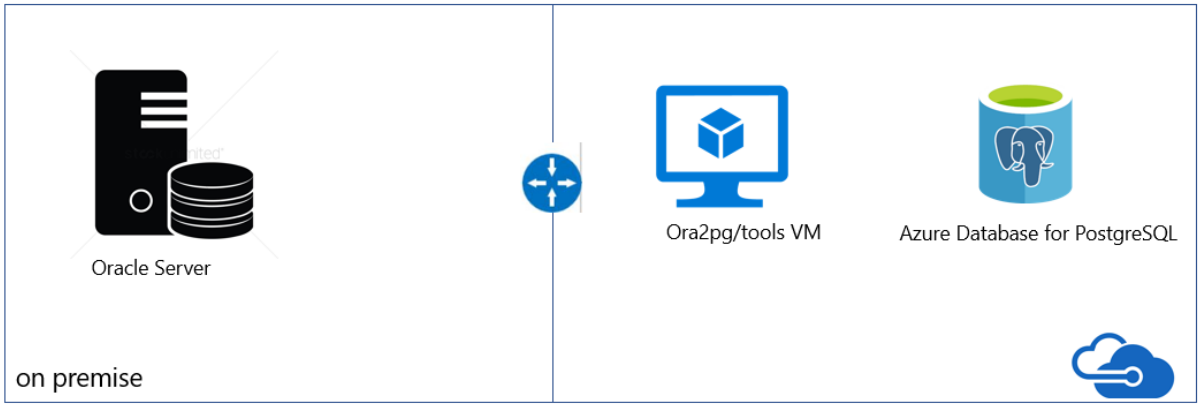
Migrate: Migrating a database involves migrating schema, data and objects usually in that order. There are 2 types of database migration methods:

1. Offline migrations – In this method, customers plan a downtime for the database, mark it read-only, perform the migration and switch over their applications to the new database.

2. Online migrations – Also known as minimal downtime, this method requires a short or no downtime for migrations. This method involves doing a front load of data to the target, enabling data sync so that data continues to be replicated from source to target until customer is ready to cut-over.

Validate: After the migration is complete, it is important to ensure that target database has the same schema, data and objects as source database. Also, it is important to validate if applications using the database behaves the same way on both source and target databases.

2 Typical ora2pg Migration Architecture



After provisioning the VM and Azure Database for PostgreSQL, two configurations are needed for enabling connectivity between them: "Allow Azure Services" and "Enforce SSL Connection", depicted as follow:

"Connection Security" blade -> Allow access to Azure Services -> ON
"Connection Security" blade -> SSL Settings -> Enforce SSL Connection -> DISABLED

3 Steps to Migrate from Oracle to Azure Database for PostgreSQL

3.1 Pre-Migration: Discovery

Microsoft Oracle pre-assessment scripts run against the Oracle database. The Pre-assessment script are a set of queries that hits the Oracle metadata and provides the following:

- Inventory of the database counts of objects by schema, type and status
- Gives a rough estimate of Raw Data in each schema – this is based on statistics
- Provides sizing of tables in each schema
- Provides how many lines of code per Package, Function, Procedure, etc.

Download the copy of the inventory script from DataMigrationTeam GitHub repository by visiting this url.

Download the related scripts from the ora2pg website or alternatively, if you are using ora2pg v20, you can download a sample configuration file from our GitHub Repo. This cookbook is prepared targeting ora2pg v20.

Guide to install ora2pg using script on Windows and Linux is available on Data Migration Team GitHub repository. Please refer to reference section for download url. Please note that the script used to install ora2pg on Windows and Linux is installs the latest version of ora2pg, however, this cookbook is prepared targeting ora2pg v20.

3.2 Pre-Migration: Assessment

After understanding the inventory of the Oracle database and having an idea of the database size and what the challenges are, the next step is to run the Assessment.

Estimating the cost of a migration process from Oracle to PostgreSQL is not easy. To obtain a good assessment of this migration cost, Ora2Pg will inspect all database objects, all functions and stored procedures to detect if there's still some objects and PL/SQL code that cannot be automatically converted by Ora2Pg.

Ora2Pg has a content analysis mode that inspect the Oracle database to generate a text report on what the Oracle database contains and what cannot be exported.

To activate the "analysis and report" mode, use the export de type SHOW_REPORT like in the following command:

```
ora2pg -t SHOW_REPORT
```

Once the database can be analysed, Ora2Pg, by his ability to convert SQL and PL/SQL code from Oracle syntax to PostgreSQL, can go further by estimating the code difficulties and estimate the time necessary to operate a full database migration.

To estimate the migration cost in man-days, Ora2Pg allow you to use a configuration directive called ESTIMATE_COST that you can also enabled at command line:

```
ora2pg -t SHOW_REPORT --estimate_cost
```

The default migration unit represent around five minutes for a PostgreSQL expert. If this is your first migration you can get it higher with the configuration directive COST_UNIT_VALUE or the --cost_unit_value command line option:

The last line of the report shows the total estimated migration code in man-days following the number of migration units estimated for each object. This migration unit represent around five minutes for a PostgreSQL expert. If this is your first migration you can get it higher with the configuration directive COST_UNIT_VALUE or the --cost_unit_value command line option. Find below some variations of assessment a) tables assessment; b) columns assessment c) schema assessment using default cost_unit (5 min) d) schema assessment using 10 min as cost unit. Run the following command to generate the report and estimates. Update the path to your configuration file path as needed.

```
ora2pg -c /etc/ora2pg/ora2pg.conf
```

The output of the schema assessment is illustrated as below:

Migration level: B-5

```
  Migration levels:
       A - Migration that might be run automatically
       B - Migration with code rewrite and a human-days cost up to 5 days
       C - Migration with code rewrite and a human-days cost above 5 days
    Technical levels:
       1 = trivial: no stored functions and no triggers
       2 = easy: no stored functions but with triggers, no manual rewriting
       3 = simple: stored functions and/or triggers, no manual rewriting
       4 = manual: no stored functions but with triggers or views with code rewriting
       5 = difficult: stored functions and/or triggers with code rewriting
```

The assessment consists in a letter A or B to specify whether the migration needs manual rewriting or not, and a number from 1 to 5 to indicate the technical difficulty level. You have an additional option --human_days_limit to specify the number of human-days limit where the migration level should be set to C to indicate that it need a huge amount of work and a full project management with migration support. Default is 10 human-days. You can use the configuration directive HUMAN_DAYS_LIMIT to change this default value permanently.

This feature has been developed to help deciding which database could be migrated first and what is the team that need be mobilized.

3.3 Pre-Migration: Convert

In this step of the migration, the conversion or translation of the Oracle Code + DDLS to PostgreSQL occurs. The ora2pg tool exports the Oracle objects in a PostgreSQL format automatically. For those objects generated, some won't compile in the PostgreSQL database without manual changes.

The process of understanding which elements need manual intervention consists in compiling the files generated by ora2pg against the PostgreSQL database, checking the log and making the necessary changes until all the schema structure is compatible with PostgreSQL syntax.

During the compilation of files, check the logs and correct the necessary syntaxes that ora2pg was unable to convert out of the box.

Refer to this guide for support on working around issues (Link)

3.3.1 ORA2PG configuration v20

ORA2PG could be run from a configuration file or a command line. The command line input is optional. Keep in mind, that If you specify any configuration variable when using command line, it will take precedence over variables in configuration file.

You can either edit the default configuration file /etc/ora2pg/ora2pg.conf (the one you have created during the installation phase) or you can create your own file (Refer to section 3.1). The default file /etc/ora2pg/ora2pg.conf will be used by ora2pg unless you specify a different one. When updating the configuration file, please keep in mind Oracle have many modules installed which may not be useful, so to make sure that these unwanted modules are not extracted from your oracle instance, please find this section in the configuration file and update as required.

```
# Allow to add a comma separated list of system user to exclude from
# Oracle extraction. Oracle have many of them following the modules
# installed. By default, it will suppress all object owned by the following
# system users:
#        CTXSYS, DBSNMP, EXFSYS, LBACSYS, MDSYS, MGMT_VIEW, OLAPSYS, ORDDATA, OWBSYS,
#        ORDPLUGINS, ORDSYS, OUTLN, SI_INFORMTN_SCHEMA, SYS, SYSMAN, SYSTEM, WK_TEST,
#        WKSYS, WKPROXY, WMSYS, XDB, APEX_PUBLIC_USER, DIP, FLOWS_020100,FLOWS_030000,
#        FLOWS_040100, FLOWS_010600, FLOWS_FILES, MDDATA, ORACLE_OCM, SPATIAL_CSW_ADMIN_USR,
#        SPATIAL_WFS_ADMIN_USR, XS$NULL, PERFSTAT, SQLTXPLAIN, DMSYS, TSMSYS, WKSYS,
#        APEX_040200, DVSYS, OJVMSYS, GSMADMIN_INTERNAL, APPQOSSYS
# Other list of users set to this directive will be added to this list.
```

```
#SYSUSERSOE, HR
```

1. Find the following four Oracle related parameters and change them accordingly:

```
ORACLE_HOME     /u01/app/oracle/product/12.1/dbhome_1
ORACLE_DSN      dbi:Oracle:host=localhost;sid=xe;port=1521
ORACLE_USER     system
ORACLE_PWD      oracle
```

2. Find the following parameters and set them as described:

```
TYPE TABLE PACKAGE COPY VIEW GRANT SEQUENCE TRIGGER FUNCTION PROCEDURE TABLESPACE TYPE PARTITI
ON
DISABLE_SEQUENCE 1
DISABLE_TRIGGERS USER
TRUNCATE_TABLE 1
DROP_FKEY 1
```

3. Set parameter *OUTPUT* to the desired value.
4. Set parameter *SCHEMA* to the source DB schema(s)
5. Set parameter *PG_SCHEMA* to the target schema name
6. Connect back to non-root OS user

3.4 Migration: Schema

After the fixes are in place, a stable build of the database is ready for deployment.

At this point, all that is required is to execute the *psql* import commands, pointing to the files containing the modified code in order to compile the database objects against the PostgreSQL database and import the data.

In this step, some level of parallelism on importing the data can be implemented. Use the following command to import DDL to PostgreSQL.

```
psql "dbname=dbhere host=hosthere user=userhere password=pwhere port=5432 sslmode=require tabl
es_output.sql”
```

* sslmode=require is mandatory (encryption)

3.5 Migration: Data Copy

As of May 2019, to do data copy, you can use Attunity Replicate for Microsoft Migrations or Striim should be considered.

You can also use ora2pg to migrate your data from source Oracle to destination PostgreSQL using following command

1) Copy the data
   To copy data from specific tables, use:

```
ora2pg -d -t COPY -a table1, table2,...,tableN
```

To copy all the tables from the database, use:

```
ora2pg -d COPY -j 8 -J 8 [-c /etc/ora2pg/ora2pg.conf]
```

* j 8 specifies the number of parallel copies to Destination PostgreSQL

* J 8 specifies the number of parallel copies from Source Oracle

* [-c /etc/ora2pg/ora2pg.conf] is optional

3.6 Migration: Cutover

With online (minimal downtime) migrations, the source you are migrating continues to change, drifting from the target in terms of data and schema, after the one-time migration occurs. During the Data cutover phase, you need to ensure that all changes in the source are captured and applied to the target in near real time. After you verify that all changes in source have been applied to the target, you can cutover from the source to the target environment.

As of May 2019, if you want to perform an online migration, consider using Attunity Replicate for Microsoft Migrations or Striim.

For "delta/incremental" migration using ora2pg, the technique consists in applying for each table a query that applies a filter (cut) by date or time, etc., and after that finalizing the migration applying a second query which will migrate the rest of the data (leftover). Refer to section 3.5.

In the source data table, migrate all the historical data first. An example of that is:

```
select * from table1 where filter_data < 01/01/2019
```

You can query the changes made since the initial migration by running a command similar to the following:

```
select * from table1 where filter_data >= 01/01/2019
```

In this case it is recommended that the validation is enhanced by checking data parity on both sides, source and target.

For online migrations, with Azure Data Migration Services you can migrate your oracle databases hosted on-premises or on a virtual machine to Azure Database for PostgreSQL. It enables resilient migrations of Oracle databases at scale and with high reliability. Provision an instance of Database Migration Service from the Azure portal or via Azure PowerShell and create a migration project to perform the migration.

3.7 Post-Migration: Perform Tests

To guarantee source and target are properly migrated the manual data validation scripts are run against the Oracle source and PostgreSQL target databases.

Ideally, if source and target databases have a networking path, ora2pg can should be used for data validation. Using the type of action called TEST allows to check that all objects from Oracle database have been created under PostgreSQL. The command run as following:

```
ora2pg -t TEST -c config/ora2pg.conf > migration_diff.txt
```

4 Migration Recommendations

- To improve the performance of the assessment or export operations in the Oracle server, collect statistics as following:

```
BEGIN
        DBMS_STATS.GATHER_SCHEMA_STATS
        DBMS_STATS.GATHER_DATABASE_STATS
        DBMS_STATS.GATHER_DICTIONARY_STATS
        END;
```

- Export data using COPY command instead of INSERT
- Avoid exporting tables with their FKs, constraints and indexes – it will make the data import into PostgreSQL slower
- Create materialized views using the "no data clause" and refresh it later
- If possible, implement unique indexes in materialized views, this will make the refresh faster with the syntax "REFRESH MATERIALIZED VIEW CONCURRENTLY"

5 References

https://docs.microsoft.com/en-gb/azure/postgresql/

https://ora2pg.darold.net/documentation.html

https://datamigration.microsoft.com/

https://www.postgresql.org/

https://www.striim.com/

https://www.attunity.com/

http://blog.dalibo.com/2016/08/19/Autonoumous_transactions_support_in_PostgreSQL.html

https://docs.microsoft.com/en-us/azure/dms/tutorial-oracle-azure-postgresql-online

https://github.com/microsoft/DataMigrationTeam

Steps to install ora2pg on Windows & Linux

Scripts to install ora2pg on Windows & Linux

Feedback and suggestions

If you have feedback or suggestions for improving this data migration asset, please contact the Data Migration Jumpstart Team (askdmjfordmtools@microsoft.com). Thanks for your support!

Note: For additional information about migrating various source databases to Azure, see the Azure Database Migration Guide.