NAME

Ora2Pg − Oracle to PostgreSQL database schema converter

DESCRIPTION

Ora2Pg is a free tool used to migrate an Oracle database to a PostgreSQL compatible schema. It connects your Oracle database, scan it automatically and extracts its structure or data, it then generates SQL scripts that you can load into your PostgreSQL database.

Ora2Pg can be used from reverse engineering Oracle database to huge enterprise database migration or simply to replicate some Oracle data into a PostgreSQL database. It is really easy to used and doesn't need any Oracle database knowledge than providing the parameters needed to connect to the Oracle database.

FEATURES

Ora2Pg consist of a Perl script (ora2pg) and a Perl module (Ora2Pg.pm), the only thing you have to modify is the configuration file ora2pg.conf by setting the DSN to the Oracle database and optionaly the name of a schema. Once that's done you just have to set the type of export you want: TABLE with constraints, VIEW, TABLESPACE, SEQUENCE, INDEXES, TRIGGER, GRANT, FUNCTION, PROCEDURE, PACKAGE, PARTITION, TYPE, and DATA.

By default Ora2Pg exports to a file that you can load into PostgreSQL with the psql client, but you can also import directly into a PostgreSQL database by setting its DSN into the configuration file. With all configuration options of ora2pg.conf you have full control of what should be exported and how.

Features included:

- Export full database schema (tables, views, sequences, indexes), with unique, primary, foreign key and check constraints.
- Export grants/privileges for users and groups.
- Export range and list partition.
- Export a table selection (by specifying the table names).
- Export Oracle schema to a PostgreSQL 7.3+ schema.
- Export predefined functions, triggers, procedures, packages and package bodies.
- Export full datas or following a WHERE clause.
- Export Oracle views as PG tables.
- Export Oracle user defined types.
- Provide basic help for converting PLSQL code to PLPGSQL.
- Works on any plateform.

Ora2Pg do its best to automatically convert your Oracle database to PostgreSQL but there's still manual works to do. The Oracle specific PL/SQL code generated for functions, procedures, packages and triggers has to be reviewed to match the PostgreSQL syntax. You will find some useful recommandations on porting Oracle PL/SQL code to PostgreSQL PL/PGSQL at ''Converting from other Databases to PostgreSQL'', section: Oracle (http://wiki.postgresql.org/wiki/Main_Page).

INSTALLATION

All Perl modules can always be found at CPAN (http://search.cpan.org/). Just type the full name of the module (ex: DBD::Oracle) into the search input box, it will brings you the page for download.

Releases of Ora2Pg stay at PgFoundry (http://pgfoundry.org/projects/ora2pg/).

Requirement

You need a modern Perl distribution (perl 5.6 and more), the DBI and DBD::Oracle Perl modules to be installed. These are used to connect to the Oracle database. To install DBD::Oracle and have it working you need to have the Oracle client libraries installed and the ORACLE_HOME environment variable must be defined.

Optional

By default Ora2Pg dumps export to flat files, to load them into your PostgreSQL database you need the PostgreSQL client (psql). If you don't have it on the host running Ora2Pg you can always transfer these files to a host with the psql client installed. If you prefer to load export 'on the fly', the perl module DBD::Pg is required.

Ora2Pg allow to dump all output int a compressed gzip file, to do that you need the Compress::Zlib Perl module or if you prefer using bzip2 compression, the program bzip2 must be available in your

PATH.

Installing Ora2Pg

Like any other Perl Module Ora2Pg can be installed with the following commands:

```
tar xzf ora2pg-6.x.tar.gz
cd ora2pg-6.x/
perl Makefile.PL
make && make install
```

This will install Ora2Pg.pm into your site Perl repository, ora2pg into /usr/bin/ and ora2pg.conf into /etc/ora2pg/.

Packaging

If you want to build binary package for your preferred Linux distribution take a look at the packaging/ directory of the source tarball. There's everything to build RPM, Slackware and Debian packages. See README file in that directory.

CONFIGURATION

Ora2Pg configuration can be as simple as choose the Oracle database to export and choose the export type. This can be done in the minute.

By reading this documentation you will also be able to:

```
- Select only certain tables and/or column for export.
- Rename some tables and/or column during export.
- Select datas to export following a WHERE clause per table.
- Delay database constraints during data loading.
- Compress exported data to save disk space.
- and much more.
```

The full control of the Oracle database migration is taken though a single configuration file named ora2pg.conf. The format of this file consist in a directive name in upper case followed by tab character and a value. Comments are lines beginning with a #.

Ora2Pg usage

By default Ora2Pg will look for /etc/ora2pg/ora2pg.conf configuration file, if the file exist you can simply execute:

```
/usr/bin/ora2pg
```

If you want to call another configuration file, just give the path as command line argument:

```
/usr/bin/ora2pg --config /etc/ora2pg/new_ora2pg.conf
```

Here are all command line parameters available since version 6.2:

Usage: ora2pg [-dhvp] [--option value]

```
-d | --debug      : Enable verbose output.
-h | --help       : Print this short help.
-v | --version    : Show Ora2Pg Version and exit.
-c | --conf file  : Used to set an alternate configuration file than the
                    default /etc/or2pg/ora2pg.conf.
-l | --log file   : Used to set a log file. Default is stdout.
-o | --out file   : Used to set the path to the output file where SQL will
                    be written. Default: output.sql in running directory.
-t | --type export: Used to set the export type. It will override the one
                    given in the configuration file (TYPE).
-p | --plsql      : Enable PLSQL to PLPSQL code conversion.
-s | --source dsn : Allow to set the Oracle DBI datasource.
-u | --user  user : Used to set the Oracle database connection user.
-w | --password pass: Used to set the password of the Oracle database user.
-n | --namespace schema: Used to set the Oracle schema to extract from.
```

Previous version do not accept any command line parameter than the path to the configuration file.

Oracle database connection

There's 5 configuration directives to control the access to the Oracle database.

ORACLE_HOME

Used to set ORACLE_HOME environment variable to the Oracle libraries required by the DBD::Oracle Perl module.

ORACLE_DSN

This directive is used to set the data source name in the form standard DBI DSN. For example:

```
dbi:Oracle:host=oradb_host.mydom.dom;sid=TEST
```

or

```
dbi:Oracle:DB_SID
```

The SID must be declared in the `$ORACLE_HOME`/network/admin/tnsnames.ora file.

ORACLE_USER et ORACLE_PWD

These two directives are used to define the user and password for the Oracle database connection. Note that if you can it is better to login as Oracle super admin to avoid grants problem during the database scan and be sure that nothing is missing.

USER_GRANTS

Set this directive to 1 if you connect the Oracle database as simple user and do not have enough grants to extract things from the DBA_... tables. It will use tables ALL_... instead. Note that this will not works with GRANT export.

TRANSACTION

This directive may be used if you want to change the default isolation level of the data export transaction. Default is now to set the level to a serializable transaction to ensure data consistency. The allowed values for this directive are:

```
readonly: 'SET TRANSACTION READ ONLY',
readwrite: 'SET TRANSACTION READ WRITE',
serializable: 'SET TRANSACTION ISOLATION LEVEL SERIALIZABLE'
committed: 'SET TRANSACTION ISOLATION LEVEL READ COMMITTED',
```

Releases before 6.2 used to set the isolation level to READ ONLY transaction but in some case this was breaking data consistency so now default is set to SERIALIZABLE.

Testing

Once you have set the Oracle database DSN you can execute ora2pg to see if it works. By default the configuration file will export the database schema to a file called 'output.sql'. Take a look in it to see if the schema has been exported.

Take some time here to test your installation as most of the problem take place here, the other configuration step are more technical.

Trouble shooting

If the output.sql file has not exported anything else than the Pg transaction header and footer there's two possible reasons. The perl script ora2pg dump an ORA-XXX error, that mean that you DSN or login information are wrong, check the error and your settings and try again. The perl script says nothing and the output file is empty: the user has not enough right to extract something from the database. Try to connect Oracle as super user or take a look at directive USER_GRANTS above and at next section, especiallly the SCHEMA directive.

Oracle schema to export

The Oracle database export can be limited to a specific Schema or Namespace, this can be mandatory following the database connection user.

SCHEMA

This directive is used to set the schema name to use during export. Take care that this directive is case sensitive. For example:

```
SCHEMA  APPS
```

will only extract objects associated to the APPS schema.

EXPORT_SCHEMA

By default the Oracle schema is not exported into the PostgreSQL database and all objects are created under the default Pg namespace. If you want to also export this schema and create all objects under this namespace, set the EXPORT_SCHEMA directive to 1. This will set the schema search_path at top of export SQL file to the schema name set in the SCHEMA directive with the default pg_catalog schema. If you want to change this path, use the directive PG_SCHEMA.

PG_SCHEMA

Allow you to defined/force the PostgreSQL schema to use. The value can be a coma delimited list of schema name. By default if you set EXPORT_SCHEMA to 1, the PostgreSQL schema search_path will be set to the schema name set as value of the SCHEMA directive plus the default pg_catalog schema as follow:

```
SET search_path = $SCHEMA, pg_catalog;
```

If you set PG_SCHEMA to something like ‘‘user_schema, public’’ for example the search path will be set like this:

```
SET search_path = $PG_SCHEMA;
-- SET search_path = user_schema, public;
```

This will force to not use the Oracle schema set in the SCHEMA directive.

SYSUSERS

Without explicit schema, Ora2Pg will export all objects that not belongs to system schema or role: SYS, SYSTEM, DBSNMP, OUTLN, PERFSTAT. Following your Oracle installation you may have several other system role defined. To append these users to the schema exclusion list, just set the SYSUSERS configuration directive to a coma separated list of system user to exclude. For example:

```
SYSUSERS        INTERNAL,SYSDBA
```

will add users INTERNAL and SYSDBA to the schema exclusion list.

Export type

The export action is perform following a single configuration directive ’TYPE’, some other add more control on what should be really exported.

TYPE

Here are the different values of the TYPE directive, default is TABLE:

```
- TABLE: Extract all tables with indexes, primary keys, unique keys,
  foreign keys and check constraints.
- VIEW: Extract only views.
- GRANT: Extract roles converted to Pg groups, users and grants on all
  objects.
- SEQUENCE: Extract all sequence and their last position.
- TABLESPACE: Extract storage space, need PostgreSQL >= v8.
- TRIGGER: Extract triggers defined following actions.
- FUNCTION: Extract functions.
- PROCEDURES: Extract procedures.
- PACKAGE: Extract packages and package bodies.
- DATA: Extract datas as INSERT statement.
- COPY: Extract datas as COPY statement.
- PARTITION: Extract range and list Oracle partitioning.
- TYPE: Extract user defined Oracle type.
```

Only one type of export can be perform at the same time so the TYPE directive must be unique. If you have more than one only the last found in the file will be registered.

Some export type can not or should not be load directly into the PostgreSQL database and still require little manual editing. This is the case for GRANT, TABLESPACE, TRIGGER, FUNCTION, PROCEDURE, TYPE and PACKAGE export types especially if you have PLSQL code or Oracle specific SQL in it.

For TABLESPACE you must ensure that file path exist on the system.

Note that you can chained multiple export by giving to the TYPE directive a coma separated list of export type.

The PARTITION export is a work in progress as table partition support is not yet implemented into PostgreSQL. Ora2Pg will convert Oracle partition using table inheritence, trigger and function workaround. See document at Pg site: http://www.postgresql.org/docs/current/interactive/ddl−partitioning.html This new feature in Ora2Pg has not been widly tested so feel free to report any bug and patch.

The TYPE export allow export of user defined Oracle type. If you don't use the −−plsql command line parameter it simply dump Oracle user type asis else Ora2Pg will try to convert it to PostgreSQL syntax.

SHOWTABLEID
Display table extraction indice and exit program (do not perform any export) if you set it to 1. Default is 0 disable. Use this directive if you want to later export from table at indice N1 to indice N2 with other directives MIN and MAX).

Limiting object to export

You may want to export only a part of an Oracle database, here are a set of configuration directives that will allow you to control what parts of the database should be exported.

TABLES
This directive allow you to set a list of tables on witch the export must be limited, excluding all other tables. The value is a space separated list of table name to export.

EXCLUDE
This directive is the opposite of the previous, it allow you to define a space separated list of table name to exclude from the export.

WHERE
This directive allow you to specify a WHERE clause filter when dumping the contents of tables. Value is construct as follow: TABLE_NAME[WHERE_CLAUSE], or if you have only one where clause for each table just put the where clause as value. Both are possible too. Here are some examples:

```
# Global where clause applying to all tables included in the export
WHERE  1=1

# Apply the where clause only on table TABLE_NAME
WHERE  TABLE_NAME[ID1='001']

# Applies two different clause on tables TABLE_NAME and OTHER_TABLE
# and a generic where clause on DATE_CREATE to all other tables
WHERE  TABLE_NAME[ID1='001' AND ID1='002] DATE_CREATE > '2001−01−01' O
```

Any where clause not included into a table name bracket clause will be applied to all exported table including the tables defined in the where clause. These WHERE clauses are very useful if you want to archive some datas or at the opposite only export some recent data.

MIN
Extract will begin at table with indice number set as value. You should use SHOWTABLEID before to know the corresponding table ids. Default value 0, disabled.

MAX
Extract will ended at table with indice number set as value. Default value 0, disabled.

Modifying object structure

One of the great usage of Ora2Pg is its flexibility to replicate Oracle database into PostgreSQL database with a different structure or schema. There's three configuration directives that allow you to map those differences.

MODIFY_STRUCT
This directive allow you to limit the columns to extract for a given table. The value consist in a space separated list of table name with a set of column between parenthesis as follow:

```
MODIFY_STRUCT   NOM_TABLE(nomcol1,nomcol2,...) ...
```

for example:

```
MODIFY_STRUCT   T_TEST1(id,dossier) T_TEST2(id,fichier)
```

This will only extract columns 'id' and 'dossier' from table T_TEST1 and columns 'id' and 'fichier' from the T_TEST2 table.

REPLACE_TABLES

This directive allow you to remap a list of Oracle table name to a PostgreSQL table name during export. The value is a list of space separated values with the following structure:

```
REPLACE_TABLES  ORIG_TBNAME1:DEST_TBNAME1 ORIG_TBNAME2:DEST_TBNAME2
```

Oracle tables ORIG_TBNAME1 and ORIG_TBNAME2 will be respectively renamed into DEST_TBNAME1 and DEST_TBNAME2

REPLACE_COLS

Like table name, the name of the column can be remapped to a different name using the following syntaxe:

```
REPLACE_COLS    ORIG_TBNAME(ORIG_COLNAME1:NEW_COLNAME1,ORIG_COLNAME2:N
```

For example:

```
REPLACE_COLS    T_TEST(dico:dictionary,dossier:folder)
```

will rename Oracle columns 'dico' and 'dossier' from table T_TEST into new name 'dictionary' and 'folder'.

PostgreSQL Import

By default conversion to PostgreSQL format is written to file 'output.sql'. The command:

```
psql mydb < output.sql
```

will import content of file output.sql into PostgreSQL mydb database.

DATA_LIMIT

When you are performing DATA or COPY export Ora2Pg proceed by chunk of 1000 tuples for speed improvement. Tuples are stored in memory before being written to disk, so if you want speed and have enough system resources you can grow this limit to an upper value for example: 100000 or 1000000. A value of 0 mean no limit so that all tuples are stored in memory before being flushed to disk.

OUTPUT

The Ora2Pg output filename can be changed with this directive. Default value is output.sql. if you set the file name with extension .gz or .bz2 the output will be automatically compressed. This require that the Compress::Zlib Perl module is installed if the filename extension is .gz and that the bzip2 system command is installed for the .bz2 extension.

BZIP2

This directive allow you to specify the full path to the bzip2 program if it can not be found in the PATH environment variable.

FILE_PER_TABLE

Allow data export to be saved in one file per table/view. The files will be named as tablename_OUTPUT. Where OUTPUT is the value of the corresponding configuration directive. You can still use .gz xor .bz2 extension in the OUTPUT directive to enable compression. Default 0 will save all data in one file, set it to 1 to enable this feature. This is usable only during DATA or COPY export type.

If you want to import data on the fly to the PostgreSQL database you have three configuration directives to set the PostgreSQL database connection. This is only possible with 'COPY' or 'DATA' export type as for database schema there's no real interest to do that.

PG_DSN

Use this directive to set the PostgreSQL data source namespace using DBD::Pg Perl module as follow:

```
dbi:Pg:dbname=pgdb;host=localhost;port=5432
```

will connect to database 'pgdb' on localhost at tcp port 5432.

PG_USER and PG_PWD
: These two directives are used to set the login user and password.

Taking export under control

The following other configuration directives interact directly with the export process and give you fine granuality in database export control.

SKIP
: For TABLE export you may not want to export all schema constraints, the SKIP configuration directive allow you to specify a space separated list of constraints that should not be exported. Possible values are:

```
- fkeys: turn off foreign key constraints
- pkeys: turn off primary keys
- ukeys: turn off unique column constraints
- indices: turn off all other index types
- checks: turn off check constraints
```

For example:

```
SKIP    indices,checks
```

will removed indexes ans check constraints from export.

KEEP_PKEY_NAMES
: By default names of the primary key in the source Oracle database are ignored and key names are created in the target PostgreSQL database with the PostgreSQL internal default naming rules. If you want to preserve Oracle primary key names set this option to 1.

FKEY_DEFERRABLE
: When exporting tables, Ora2Pg normally exports constraints as they are, if they are non-deferrable they are exported as non-deferrable. However, non-deferrable constraints will probably cause problems when attempting to import data to Pg. The FKEY_DEFERRABLE option set to 1 will cause all foreign key constraints to be exported as deferrable.

DEFER_FKEY
: In addition, when exporting data the DEFER_FKEY option set to 1 will add a command to defer all foreign key constraints during data export. Constraints will then be checked at the end of each transaction.

DISABLE_TABLE_TRIGGERS
: This directive is used to disables triggers on all tables in COPY or DATA export modes during data migration. The possible values are 0 to enable triggers, USER to disable userdefined triggers and ALL to disable userdefined triggers as well as includes RI system triggers.

DISABLE_SEQUENCE
: If set to 1 disables alter of sequences on all tables during COPY or DATA export mode. This is used to prevent the update of sequence during data migration. Default is 0, alter sequences.

NOESCAPE
: By default all datas exported as INSERT statement are escaped, if you experience any problem with that set it to 1 to disable character escaping during data export.

PG_NUMERIC_TYPE
: This directive set to 1 replace portable numeric type into PostgreSQL internal type as numeric(p,s) type is much slower than the different PostgreSQL numeric types. Oracle data type NUMBER(p,s) is approximatively converted to smallint, integer, bigint, real and float PostgreSQL numeric type following the precision. If you have lot of monetary fields you should preserve the numeric(p,s) Pg data type if you need very good precision. NUMBER without precision are set to float unless you redefine it with the DEFAULT_NUMERIC configuration option.

DEFAULT_NUMERIC
: NUMBER without precision are converted by default to float if PG_NUMERIC_TYPE is true. You can overwrite this value to any PG numeric type, like integer or bigint.

DATA_TYPE

If you're experiencing any problem in data type schema conversion with this directive you can take full control of the correspondence between Oracle and PostgreSQL types to redefine data type translation used in Ora2pg. The syntax is a coma separated list of ''Oracle datatype:Postgresql datatype''. Here are the default list used:

```
DATA_TYPE       DATE:timestamp,LONG:text,LONG RAW:text,CLOB:text,NCLOB
```

Note that the directive and the list definition must be a single line.

CASE_SENSITIVE

By default Ora2P convert all object names to lower case as PostgreSQL is case insensitive. If you want to preserve the case of Oracle object name set this directive to 1. I do not recommand this unless you always quote object names on all your scripts.

ORA_SENSITIVE

Since version 4.10 you can export Oracle databases with case sensitive table or view names. This requires the use of quoted table/view names during Oracle querying. Set this configuration option to 1 to enable this feature. By default it is off.

ORA_RESERVED_WORDS

Allow escaping of column name using Oracle reserved words. Value is a list of coma separated reserved word. Default is audit,comment.

GEN_USER_PWD

Set this directive to 1 to replace default password by a random password for all extracted user during a GRANT export.

PG_SUPPORTS_ROLE

By default Oracle roles are translated into PostgreSQL groups. If you have PostgreSQL 8.1 or more consider the use of ROLES and set this directive to 1 to export roles.

PG_SUPPORTS_INOUT

If set to 0, all IN, OUT or INOUT parameters will not be used into the generated PostgreSQL function declarations (disable it for PostgreSQL database version lower than 8.1), This is now enable by default. Please note that things like default parameters aren't supported by PostgreSQL and will not be exported.

Special options to handle character encoding

NLS_LANG

If you experience any issues where mutibyte characters are being substituted with some replacement characters during the export try to set the NLS_LANG configuration directive to the Oracle encoding. This may help a lot especially with UTF8 encoding. For example:

```
NLS_LANG        AMERICAN_AMERICA.UTF8
```

This will set $ENV{NLS_LANG} to the given value.

BINMODE

If you experience the Perl warning: ''Wide character in print'', it means that you tried to write a Unicode string to a non-unicode file handle. You can force Perl to use binary mode for output by setting the BINMODE configuration option to the specified encoding. If you set it to 'utf8', it will force printing like this: binmode OUTFH, '':utf8''; By default Ora2Pg opens the output file in 'raw' binary mode.

PLSQL to PLPSQL convertion

Automatic code convertion from Orable PLSQL to PostgreSQL PLPSQL is a work in progress in Ora2Pg and surely you will always have manual work. The Perl code used for automatic conversion is all stored in a specific Perl Module named Ora2Pg/PLSQL.pm feel free to modify/add you own code and send me patches. The main work in on function, procedure, package and package body headers and parameters rewrite.

PLSQL_PGSQL

Enable/disable PLSQL to PLPSQL convertion. Default disabled.

Other configuration directives

DEBUG

Set it to 1 will enable verbose output.

IMPORT

You can define common Ora2Pg configuration directives into a single file that can be imported into other configuration files with the IMPORT configuration directive as follow:

```
IMPORT  commonfile.conf
```

will import all configuration directives defined into commonfile.conf into the current configuration file.

SUPPORT

Author / Maintainer

Gilles Darold <gilles AT darold DOT net>

Please report any bugs, patches, help, etc. to <gilles AT darold DOT net>.

Feature request

If you need new features let me know at <gilles AT darold DOT net>. This help a lot to develop a better/useful tool.

How to contribute ?

Any contribution to build a better tool is welcome, you just have to send me your ideas, features request or patches and there will be applied.

LICENSE

Copyright (c) 2000−2010 Gilles Darold − All rights reserved.

```
This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see < http://www.gnu.org/licenses/ >.
```

ACKNOWLEDGEMENT

I must thanks a lot all the great contributors:

```
Guillaume Lelarge
Stephane Schildknecht
Jean−Paul Argudo
Jan Kester
Paolo Mattioli
Mike Wilhelm−hiltz
Jefferson Medeiros
Ian Boston
Thomas Wegner
Andreas Haumer
Marco Lombardo
Adam Sah and Zedo Inc
Antonios Christofide and National Technical University of Athens
Josian Larcheveque
Stephane Silly
David Cotter − Alatto Technologies Ltd
Wojciech Szenajch
Richard Chen
Sergio Freire
```

```
Matt Miller
Rene Bentzen
Schnabl Andrea
Ugo Brunel - Bull
Bernd Helmle - credativ GmbH
Peter Eisentraut
Marc Cousin
Daniel Scott
Luca DallOlio
Ali Pouya
Olivier Mazain
Brendan Richards
Andrea Agosti
```

and all others who help me to build a useful and reliable product:

```
Jason Servetar
Jean-Francois Ripouteau
Octavi Fors
Adriano Bonat
Thomas Reiss
Bozkurt Erkut from SONY
Igor MII
```