

# Ora2Pg

Presentation, best practices  
and roadmap



# Synopsis

- Historic and general
- Installation
- Best practices
  - Common configuration
  - Schema migration
  - Data migration
  - Stored procedures migration
  - Unitary tests
- PL/SQL to PLPGSQL conversion
- Ora2Pg roadmap

# Historic 1/2

- Created in 2000
- First a data duplication tool from Oracle to PostgreSQL
  - Copy Oracle to PostgreSQL tables (+/- some columns)
- An Oracle database scanner / reverse engineering
  - Difficult to obtain all informations
- Oracletool (<http://www.oracletool.com/>)
  - Perl Web tool for Oracle DBAs - Adam vonNieda

# Historic 2/2

- Oracle to PostgreSQL database migration tool
  - First official release: may 2001
  - 2002 : Ora2Pg was added to the contrib/ repository of PostgreSQL v7.2
  - 2006 : it has been removed from the contrib/ repository of PostgreSQL v8.2
  - 2008 : Ora2Pg moves to PgFoundry
  - 2010 : Ora2Pg web site => <http://ora2pg.darold.net/>
  - 2011 : release are now hosted on SourceForge.net
- Current release: Ora2Pg 8.8

# About Oracle to PostgreSQL migration

- Demystify the Oracle database migration
- Automatic migration are rarely possible
- Compatibility layers are slow
- Other migration tools
  - Orafce (<http://pgfoundry.org/projects/orafce/>)
  - EnterpriseDB Advanced Server Plus
  - Bull (<http://www.bull.us/liberatedb/>)
- No miracle, it need at least some rewrite

# Code design

- **Ora2Pg.pm** - main Perl module used to interfacing with Oracle and allowing all kind of exports.
- **Ora2Pg/PSQL.pm** - module used to convert Oracle PL/SQL code into PLPGSQL code.
- **ora2pg** – Perl script used as frontend to the Perl modules.
- **ora2pg.conf** - configuration file used to define the behaviors of the Perl script ora2pg and the action to do.

# Prerequisite

- Oracle  $\geq$  8i client or server installed
- PostgreSQL  $\geq$  8.4 client or server installed
- Perl 5.8+ and DBI/DBD::Oracle Perl modules
- Windows : Strawberry Perl 5.10+
- Optionals Perl modules:
  - DBD::Pg – for direct import into PostgreSQL
  - Compress::Zlib – to compress output files on the fly
- Multi-threading : Perl compiled with thread support
  - `perl -V | grep "useithread=defined"`

# Installation 1/2

- Oracle / PostgreSQL : follow your system installation documentation.
- Define the ORACLE\_HOME environment variable

```
Export ORACLE_HOME=/usr/lib/oracle/10.2.0.4/client64
```

- File tnsnames.ora

```
cat <<EOF > $ORACLE_HOME/network/admin/tnsnames.ora
```

```
XE = ( DESCRIPTION =  
      (ADDRESS = (PROTOCOL = TCP) (HOST = 192.168.1.10) (port = 1521) )  
      (CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME = XE) )  
    )
```

```
EOF
```



# Installation 2/2

- Verify the Oracle installation using `tnsping` or `sqlplus`.
- Install Perl modules `DBD::Oracle` et `DBD::Pg`
- Unix/Linux Install
  - `perl Makefile.PL`
  - `make && sudo make install`
- Windows install
  - `perl Makefile.PL`
  - `dmake && dmake install`
  - Install manually `ora2pg.pl` et `ora2pg.conf`

# Workspace 1/2

```
mig_project/  
  mig_config/  
    ora2pg.conf  
  mig_schema/  
    users/      tables/      sequences/    views/  
    triggers/   functions/   procedures/  
    types/      packages/    tablespaces/  
  mig_source/  
    oraviews/   oratriggers/  oratypes/  
    orafunctions/  oraprocedures/  
    Orapackages/  
  mig_data/
```

# Workspace 2/2

- Script to create automatically the workspace

```
#!/bin/sh
mkdir mig_project/ && cd mig_project/
for d in users tables sequences views triggers functions
procedures types packages tablespaces
do
    mkdir -p mig_schema/$d
done
for d in oratypes oraviews oratriggers orafunctions oraprocedures
orapackages
do
    mkdir -p mig_source/$d
done
mkdir mig_config/
mkdir mig_data/
cp -n /etc/ora2pg/ora2pg.conf mig_config/
```

# Common configuration 1/4

- Oracle database connection: DataSourceName
  - ORACLE\_DSN dbi:Oracle:host=192.168.1.10;sid=XE
  - ORACLE\_USER hr
  - ORACLE\_PWD mypassphrase
- Oracle connection user: DBA or not
- DBA is mandatory to export GRANT, TYPE and TABLESPACE (need access to DBA\_\* tables)
- If non DBA user, Ora2Pg will need to be informed to look at ALL\_\* tables
  - USER\_GRANTS 1

# Common configuration 2/4

- Oracle schema should be exported into PG ?
  - EXPORT\_SCHEMA 1
  - Oracle schema list : ora2pg -t SHOW\_SCHEMA
- Is there's some tables to exclude from export ?
  - EXCLUDE table1 table2 table3
  - oracle tables list : ora2pl -t SHOW\_TABLE
- Some tables or columns need to be renamed ?
  - REPLACE\_TABLES
  - REPLACE\_COLS
  - Oracle columns of a given table:
    - ora2pl -t SHOW\_COLUMN -x TABLE\_NAME

# Common configuration 3/4

- What is the Oracle database encoding ?
  - `NLS_LANG AMERICAN_AMERICA.UTF8`
  - `ora2pg -t SHOW_ENCODING`
  - The `NLS_LANG` value is obtained by concatenating the `NLS_LANGUAGE`, `NLS_TERRITORY` and `NLS_CHARACTERSETS` values.
    - Example : `FRENCH_FRANCE.WE8ISO8859P1`
- Automatic conversion to PostgreSQL encoding
  - `CLIENT_ENCODING LATIN9`
- The character set in PostgreSQL
  - <http://www.postgresql.org/docs/9.1/static/multibyte.html>

# Common configuration 4/4

- DATA\_LIMIT 10000
- DROP\_FKEY 0
- DISABLE\_TABLE\_TRIGGERS 0
- FILE\_PER\_CONSTRAINT 1
- FILE\_PER\_INDEX 1
- FILE\_PER\_TABLE 1
- FILE\_PER\_FUNCTION 1
- TRUNCATE\_TABLE 1
- PG\_SUPPORTS\_WHEN 1
- PG\_SUPPORTS\_INSTEADOF 1
- STANDARD\_CONFORMING\_STRINGS 1

# Schema migration 1/4

- Different kind of export:
  - TABLESPACE - GRANT - TYPE
  - TABLE - SEQUENCE - VIEW - TRIGGER
  - FUNCTION - PROCEDURE - PACKAGE
- Export choice by modification of the configuration file or the use of INCLUDE
- More flexible with options -t, -o, -b at command line:
  - -t EXPORT\_NAME : kind of export
  - -o FILENAME : output file suffix (output.sql)
  - -b DIRECTORY : output directory of the export files



# Schema migration 2/4

```
export ora2pg_conf=mig_configs/ora2pg.conf
```

```
ora2pg -t TABLE -o table.sql -b mig_schema/tables -c $ora2pg_conf
```

```
ora2pg -t SEQUENCE -o sequences.sql -b mig_schema/sequences -c $ora2pg_conf
```

```
ora2pg -t GRANT -o users.sql -b mig_schema/users -c $ora2pg_conf
```

```
ora2pg -t TABLESPACE -o tablespaces.sql -b mig_schema/tablespaces -c $ora2pg_conf
```

```
ora2pg -p -t TYPE -o types.sql -b mig_schema/types -c $ora2pg_conf
```

```
ora2pg -p -t VIEW -o views.sql -b mig_schema/views -c $ora2pg_conf
```

```
ora2pg -p -t TRIGGER -o triggers.sql -b mig_schema/triggers -c $ora2pg_conf
```

```
ora2pg -p -t FUNCTION -o functions.sql -b mig_schema/functions -c $ora2pg_conf
```

```
ora2pg -p -t PROCEDURE -o procs.sql -b mig_schema/procedures -c $ora2pg_conf
```

```
ora2pg -p -t PACKAGE -o packages.sql -b mig_schema/packages -c $ora2pg_conf
```

```
ora2pg -t TYPE -o types.sql -b mig_schema/oratypes -c $ora2pg_conf
```

```
ora2pg -t VIEW -o views.sql -b mig_schema/oraviews -c $ora2pg_conf
```

```
ora2pg -t TRIGGER -o triggers.sql -b mig_schema/oratriggers -c $ora2pg_conf
```

```
ora2pg -t FUNCTION -o functions.sql -b mig_schema/orafunctions -c $ora2pg_conf
```

```
ora2pg -t PROCEDURE -o procs.sql -b mig_schema/oraprocedures -c $ora2pg_conf
```

```
ora2pg -t PACKAGE -o packages.sql -b mig_schema/orapackages -c $ora2pg_conf
```

# Schema migration 3/4

- Create the Pg database owner:
  - `createuser --no-superuser --no-createrole --no-createdb miguser`
- Working with schema (EXPORT\_SCHEMA)
  - `ALTER ROLE miguser SET search_path TO "migschema",public;`
- Create the Pg database:
  - `createdb -E UTF-8 --owner miguser migdb`
- Create the database objects:
  - `psql -U miguser -f sequences/sequences.sql migdb > create_migdb.log 2>&1`
  - `psql -U miguser -f tables/tables.sql migdb >> create_migdb.log 2>&1`

# Schema migration 4/4

- Look into log file and study the problems
  - Bad encoding in the CKECK constraint values for example
  - Specific Oracle code found into constraints or indexes definition
- PostgreSQL reserved words found into tables or columns names (ex: comment, user)
  - Usage of user defined Oracle types, see TYPE export
- Error in SQL code sample:
  - `CREATE INDEX idx_usage ON user ( to_number(to_char('YYYY', user_age)) );`
  - `CREATE INDEX idx_usage ON «user» ( date_part('year', user_age) );`

# Data migration 1/3

- Export data as COPY statements into text file:
  - `ora2pg -t COPY -o datas.sql -b mig_data/ -c mig_config/ora2pg.conf`
- Import data into PostgreSQL database:
  - `psql -U miguser -f mig_data/datas.sql migdb >> migdb_data.log 2>&1`
- Restore constraints and indexes:
  - `psql -U miguser -f mig_schema/tables/CONSTRAINTS_table.sql migdb >> migdb_data.log 2>&1`
  - `psql -U miguser -f mig_schema/tables/INDEXES_table.sql migdb >> migdb_data.log 2>&1`

# Data migration 2/3

- Exporting Oracle's BLOB into bytea is very slow because of the escaping of all data
- Exclude tables with bytea column from the global data export using EXCLUDE directive
- Activate multi-threading when exporting the bytea tables using the TABLES directive
  - THREAD\_COUNT set to Ncore ( $\leq 5$  above there's no real performance gain)
- DATA\_LIMIT set to 5000 max to not OOMing
- With huge data use an ETL (Kettle for example)

# Data migration 3/3

- Exporting data with composite type

- Inserting sample into oracle:

```
Insert into T_TEST (ID,OBJ) Values (1,"TEST_TYPE_A"(13,'obj'));
```

- Export by Ora2Pg

```
INSERT INTO t_test (id,obj) VALUES (1,ARRAY(0x8772fb8));
```

```
COPY "t_test" ("id","obj") FROM stdin;
```

```
1    ARRAY(0xa555fb8)
```

```
\.
```

- Solving this required to know the columns type of the composite type before proceeding to the data export => Ora2Pg v9.x

# Stores procedures migration 1/2

- Loading of functions and procedures
  - `psql --single-transaction -U miguser -f procedures/procedures.sql migdb`
  - `psql --single-transaction -U miguser -f functions/functions.sql migdb`
- Load packages of functions
  - `psql --single-transaction -U miguser -f packages/packages.sql migdb`
- Import files dedicated to each function of each package (one subdirectory per packages)
- Exit on error: `\set ON_ERROR_STOP ON`

# Stores procedures migration 2/2

- Missing Oracle PL/SQL Code ?
- Use `COMPILE_SCHEMA` configuration directive to force Oracle to validate the PL/SQL code before export.
  - or do it yourself : `DBMS_UTILITY.compile_schema (schema => sys_context('USERENV', 'SESSION_USER'));`
- Activate `EXPORT_INVALID` to export all Oracle PL/SQL code.
  - By default Ora2Pg will export only Oracle code set as `VALID`.
- Ora2Pg preserve comments into function code



# Unitary tests

- It is really important to validate the stored procedure code and that is working the same way as in Oracle
- It is almost possible that the results differ:
  - either slightly, for example with the number of decimals after the dot.
  - either heavily, despite the PL/PGSQL code has been loaded without errors.
- PL/pgsql debugger
  - Edb-debugger (<http://pgfoundry.org/projects/edb-debugger/>)
  - Pavel Stehule's plpgsql\_lint (<http://kix.fsv.cvut.cz/~stehule/download/>)

# PL/SQL to PLPGSQL rewrite 1/5

- Complete rewrite of triggers, functions, procedures and packages headers
- Replacement of NVL by coalesce()
- Replacement of trunc() into date\_trunc('day',...)
- Replacement of SYSDATE by LOCALTIMESTAMP (same as CURRENT\_TIMESTAMP without timezone)
- Remove of FROM DUAL call
- Rewrite calls to sequences (name.nextval → nextval('name'))
- Replace calls to MINUS by EXCEPT
- Replace all Oracle types into variable definitions into corresponding PostgreSQL type

# PL/SQL to PLPGSQL rewrite 2/5

- Replace `dup_val_on_index` by `unique_violation`
- Replace `raise_application_error` by `RAISE EXCEPTION`
- Replace Oracle `DBMS_OUTPUT.(put_line|put|new_line)` into `RAISE NOTICE` call
- Remove of `DEFAULT NULL` which is the default value with PostgreSQL when no default value is given
- Rewrite cursor declaration to make them compatibles with PostgreSQL
- Rewrite of `RAISE EXCEPTION` with concatenation `||` by the format à la `sprintf` used by PostgreSQL

# PL/SQL to PLPGSQL rewrite 3/5

- Add reserved keyword STRICT to the SELECT ... INTO when an EXCEPTION ... NO\_DATA\_FOUND or TOO\_MANY\_ROWS is found
- Remove object's name repeated after the END keyword, for example : "END fct\_name;" is rewritten into "END;"
- Replacement of "WHERE ROWNUM = N" or "AND ROWNUM = N" by "LIMIT N"
- Moves comments into CASE between the WHEN and the THEN keywords at top of the clause, this is not supported by PostgreSQL
- Rewrite the HAVING ... GROUP BY clause order by GROUP BY ... HAVING, inverted under PostgreSQL
- Rewrite calls to functions add\_months et add\_years into "+ 'N months/year'::interval"

# PL/SQL to PLPGSQL rewrite 4/5

- Replacement of conditions IS NULL and IS NOT NULL by instructions based on coalesce (for Oracle, an empty string is the same as NULL)
- Replacement of SQLCODE by his equivalent SQLSTATE under PostgreSQL
- Replacement of TO\_NUMBER(TO\_CHAR(...)) en to\_char(...)::integer, in PostgreSQL the to\_number() needs two parameters
- Replacement of SYS\_EXTRACT\_UTC by AT TIME ZONE 'UTC'
- Reverting min and max limits into the FOR ... IN ... REVERSE min .. max loop

# PL/SQL to PLPGSQL rewrite 5/5

- Replacement of cursor loop exit `EXIT WHEN ...%NOTFOUND` by `IF NOT FOUND THEN EXIT; END IF;`
- Replacement of `SQL%NOTFOUND` by `NOT FOUND`
- Replacement of `SYS_REFCURSOR` by `REFCURSOR`
- Replacement of `INVALID_CURSOR` by `INVALID_CURSOR_STATE`
- Replacement of `ZERO_DIVIDE` by `DIVISION_BY_ZERO`
- Replacement of `STORAGE_ERROR` by `OUT_OF_MEMORY`
  
- This code is into the Perl module `Ora2pg/PLSQL.pm` and the function: `plsqli_to_plpgsql()`

# Ora2Pg roadmap

- Source code should be hosted on [github.org](https://github.com)
- Add an option to create a skeleton of project with export/import scripts creation
- Add an option to evaluate the cost of an Oracle database migration - `ESTIMATE_COST`
- Add an option to only extract a report on the Oracle database content
- Allow to modify data on the fly by calling a function following the type, table or column
- Allow data export of composites type and XML

# Questions ?

- <http://ora2pg.darold.net/>
- <http://sourceforge.net/projects/ora2pg/>
- gilles [ at ] darold [ dot ] net



**[Http://2011.pgDay.eu/](http://2011.pgDay.eu/) , Amsterdam**