Ora2Pg

Présentation, bonnes pratiques et futur du projet

Sommaire

- Historique et généralités
- Installation
- Bonnes pratiques
  - Configuration générique
  - Migration du schéma
  - Migration des données
  - Migration des procédures stockées
  - Tests unitaires
- Conversion PL/SQL vers PLPGSQL
- Futur d'Ora2Pg

Historique 1/2

- Créé en 2000
- Outil de duplication de données Oracle vers PostgreSQL
  - Copie table vers table (+/- certaines colonnes)
- Scanner de bases Oracle / reverse engineering
  - Difficulté d'obtenir l'information
- Oracletool (http://www.oracletool.com/))
  - Outil Web en Perl pour DBAs Oracle – par Adam vonNieda

Historique 2/2

- Outil d'aide à la migration d'une base Oracle vers PostgreSQL
  - Première version officielle : mai 2001
  - 2002 : Ora2Pg est ajouté au répertoire contrib/ de PostgreSQL v7.2
  - 2006 : il est supprimé du répertoire contrib/ de PostgreSQL v8.2
  - 2008 : Ora2Pg migre sur PgFoundry
  - 2010 : site d'Ora2Pg => http://ora2pg.darold.net/
  - 2011 : migration des sources sur SourceForge.net
- Version actuelle : Ora2Pg 8.8

A propos de la migration Oracle vers PostgreSQL

- Démystifier la migration des bases Oracle
- Migrations automatiques rarement possibles
- Lenteur des couches de compatibilité
- Autres outils d'aide à la migration
  - Orafce (http://pgfoundry.org/projects/orafce/)
  - EnterpriseDB Advanced Server Plus
  - Bull (http://www.bull.us/liberatedb/)
- Pas de miracle, nécessite un minimum de réécriture

Architecture du code

- **Ora2Pg.pm** - module principal réalisant l'interface avec Oracle et permettant tous les types d'exports.
- **Ora2Pg/PSQL.pm** - module réalisant la conversion du code Oracle PL/SQL en code PLPGSQL.
- **ora2pg** - script Perl servant de frontend aux modules Perl.
- **ora2pg.conf** - fichier de configuration définissant le comportement du script Perl ora2pg et des actions à réaliser.

Prérequis

- Oracle >= 8i client ou serveur installé
- PostgreSQL >= 8.4 client ou serveur installé
- Perl 5.8+ et les modules DBI / DBD::Oracle
- Windows : Strawberry Perl 5.10+
- Modules Perl optionnels :
  - DBD::Pg - import direct dans PostgreSQL
  - Compress::Zlib - compression des fichiers en sortie
- Multi-threading : Perl compilé avec le support des threads
  - perl -V | grep ”useithread=defined”

Installation 1/2

- Oracle / PostgreSQL : suivre la procédure d'installation de votre système.
- Définition de la variable ORACLE_HOME

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

- Fichier 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

- Vérifier l'installation Oracle avec tnsping ou sqlplus.
- Installation de DBD::Oracle et DBD::Pg
- Installation sous Unix/Linux
  - perl Makefile.PL
  - make && sudo make install
- Installation sous windows
  - perl Makefile.PL
  - dmake && dmake install
  - Installer manuellement ora2pg.pl et ora2pg.conf

Espace de travail 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/
```

Espace de travail 2/2

- Script de création de l'arborescence de travail

```
#!/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/
```

Configuration générique 1/4

- Connexion Oracle, le DataSourceName
  - ORACLE_DSN dbi:Oracle:host=192.168.1.10;sid=XE
  - ORACLE_USER hr
  - ORACLE_PWD mypassphrase
- Utilisateur Oracle : DBA ou pas
- DBA obligatoire pour l'export des GRANT, TYPE et TABLESPACE (accès aux tables DBA_*)
- Si l'utilisateur n'est pas DBA utilisation des tables ALL_*
  - USER_GRANTS 1

Configuration générique 2/4

- Le schéma Oracle doit-il être reporté dans PG?
  - EXPORT_SCHEMA 1
  - Liste des schéma : ora2pg -t SHOW_SCHEMA
- Y a-t-il des tables à exclure de l'export ?
  - EXCLUDE table1 table2 table3
  - Liste des tables : ora2pl -t SHOW_TABLE
- Renommage des tables et colonnes
  - REPLACE_TABLES
  - REPLACE_COLS
  - Liste des colonnes d'une table:
    - ora2pl -t SHOW_COLUMN -x TABLE_NAME

Configuration générique 3/4

- Quel est l'encodage de la base Oracle
  - NLS_LANG AMERICAN_AMERICA.UTF8
  - ora2pg -t SHOW_ENCODING
  - Obtenu par concaténation des valeurs de NLS_LANGUAGE, NLS_TERRITORY et NLS_CHARACTERSETS.
  - Exemple : FRENCH_FRANCE.WE8ISO8859P1.
- Conversion de l'encodage par PostgreSQL
  - CLIENT_ENCODING LATIN9
- Jeux de caractères dans PostgreSQL
  - http://www.postgresql.org/docs/9.1/static/multibyte.html

Configuration générique 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

Migration du Schéma 1/4

- Les différents types d'export :
  - TABLESPACE - GRANT - TYPE
  - TABLE - SEQUENCE - VIEW - TRIGGER
  - FUNCTION - PROCEDURE - PACKAGE
- Export par modification du fichier de configuration ou l'utilisation de INCLUDE
- Plus souple avec les options -t, -o, -b en ligne de commande.
  - -t EXPORT_NAME : type d'export
  - -o FILENAME : fichier de sortie (output.sql)
  - -b DIRECTORY : répertoire de stockage des fichiers

Migration du Schéma 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
```

Migration du Schéma 3/4

- Création du propriétaire de la base :
  - createuser --no-superuser --no-createrole --no-createdb miguser
- Utilisation d'un schéma (EXPORT_SCHEMA)
  - ALTER ROLE miguser SET search_path TO "migschema",public;
- Création de la base :
  - createdb -E UTF-8 --owner miguser migdb
- Création des objets de la base :
  - 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

Migration du Schéma 4/4

- Lecture des logs et étude des problèmes
  - Encodage des valeurs de contraintes CKECK
  - Possibilité de code spécifique à Oracle dans les contraintes et les index
  - Mots réservés PostgreSQL dans les noms de tables ou colonnes (ex: comment, user)
  - Utilisation de types Oracle définis par l'utilisateur, voir TYPE
- Exemple de code en erreur :
  - CREATE INDEX idx_userage ON user (to_number(to_char('YYYY', user_age)));
  - CREATE INDEX idx_userage ON «user» (date_part('year', user_age));

Migration des données 1/3

- Création des fichiers de données :
  - ora2pg -t COPY -o datas.sql -b mig_data/ -c mig_config/ora2pg.conf
- Import des données dans PostgreSQL
  - psql -U miguser -f mig_data/datas.sql migdb >> migdb_data.log 2>&1
- Restauration des contraintes et index
  - 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

Migration des données 2/3

- L'export des champs BLOB en bytea est très lent en raison de l'échappement des données
- Exclure les tables avec colonne bytea lors de l'export global en utilisant la directive EXCLUDE
- Activer le multi-threading pour l'export de ces tables en utilisant la directive TABLES
  - THREAD_COUNT à Ncore (<= 5 après on ne gagne pas grand chose en performance)
- DATA_LIMIT à 5000 max sinon risque d'OOM
- Utilisation d'un ETL (par exemple Kettle) pour les très gros volumes de données

Migration des données 3/3

- Problème de l'export des données avec des types composites ou XMLType
  - Exemple d'insertion dans oracle:

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

  - Export par Ora2Pg

```
INSERT INTO t_test (id,obj) VALUES (1,ARRAY(0x8772fb8));
COPY "t_test" ("id","obj") FROM stdin;
1	ARRAY(0xa555fb8)
\.
```

- La solution nécessite de connaître le type des colonnes du type composite avant l'export des données. Ora2Pg >= 9.x

Migration des procédures stockées 1/2

- Chargement des fonctions et procédures
  - psql -U miguser -f procedures/procedures.sql migdb
  - psql -U miguser -f functions/functions.sql migdb
- Chargement des paquets de fonctions
  - psql -U miguser -f packages/packages.sql migdb
- Fichier dédié à chaque fonction de chaque paquet (un sous répertoire par paquet/schema)
- Arrêt sur erreur : \set ON_ERROR_STOP ON

Migration des procédures stockées 2/2

- Il manque du code PL/SQL d'Oracle?
- Activer COMPILE_SCHEMA force Oracle à valider de nouveau le PL/SQL avant l'export
  - Manuellement : DBMS_UTILITY.compile_schema (schema => sys_context('USERENV', 'SESSION_USER'));
- Activer EXPORT_INVALID pour exporter tout le code PL/SQL d'Oracle même invalide.
  - Par défaut Ora2Pg exporte seulement le code défini comme VALID par Oracle
- Ora2Pg préserve les commentaires défini dans le corps des fonctions.

Tests unitaires

- Validation du portage et du fonctionnement à l'identique de la version sous Oracle
- Il est possible que les résultats diffèrent:
  - soit légèrement, par exemple avec le nombre de décimales après la virgule
  - soit fortement, bien que le code PL/PGSQL ait été importé sans erreurs.
- Debugger le code PL/pgsql
  - Edb-debugger (http://pgfoundry.org/projects/edb-debugger/)
  - Pavel Stehule's plpgsql_lint (http://kix.fsv.cvut.cz/~stehule/download/)

Conversion PL/SQL vers PLPGSQL 1/5

- Réécriture complète des en-têtes de triggers, fonctions, procédures et packages
- Remplacement de NVL par coalesce()
- Remplacement de trunc() en date_trunc('day',…)
- Remplacement de SYSDATE en LOCALTIMESTAMP (équivalent de CURRENT_TIMESTAMP sans le fuseau horaire)
- Supprime les appels à FROM DUAL
- Réécrit les appels aux séquences (nom.nextval → nextval('nom'))
- Remplace les appels à MINUS par EXCEPT
- Remplace tous les types Oracle des définitions de variable en type PostgreSQL

Conversion PL/SQL vers PLPGSQL 2/5

- Remplace dup_val_on_index en unique_violation
- Remplace raise_application_error en RAISE EXCEPTION
- Remplace les sorties Oracle DBMS_OUTPUT.(put_line|put|new_line) en RAISE NOTICE
- Supprime les DEFAULT NULL qui est la valeur par défaut sous PostgreSQL lorsqu'aucune valeur par défaut n'est précisée
- Réécrit les déclarations de curseur pour les rendre compatibles avec PostgreSQL
- Réécrit les RAISE EXCEPTION avec concaténation || par le format à la sprintf utilisé par PostgreSQL

Conversion PL/SQL vers PLPGSQL 3/5

- Ajout du mot clé STRICT aux SELECT … INTO lorsqu'il y a un EXCEPTION … NO_DATA_FOUND ou TOO_MANY_ROW
- Suppression des noms d'objets répétés après les END, exemple : ”END fct_name;” est réécrit en ”END;”
- Remplacement des ”WHERE ROWNUM = N” ou ”AND ROWNUM = N” par ”LIMIT N”
- Déplacement des commentaires dans les CASE entre le WHEN et le THEN, non supporté par PostgreSQL
- Réécrit la clause HAVING … GROUP BY en GROUP BY … HAVING, inversée sous PostgreSQL
- Réécrit les appels aux fonctions add_months et add_years en ”+ 'N months/year'::interval”

Conversion PL/SQL vers PLPGSQL 4/5

- Remplacement des conditions IS NULL et IS NOT NULL par des instructions à base de coalesce (pour Oracle, une chaîne vide est équivalente à NULL)
- Remplacement de SQLCODE par le presque équivalent SQLSTATE sous PostgreSQL
- Remplacement des TO_NUMBER(TO_CHAR(…)) en to_char(…)::integer, la fonction to_number() nécessitant deux paramètres sous PostgreSQL
- Remplacement de SYS_EXTRACT_UTC par AT TIME ZONE 'UTC'
- Inversement des bornes min et max dans les boucles FOR … IN … REVERSE min .. max

Conversion PL/SQL vers PLPGSQL 5/5

- Remplacement des sorties de curseur EXIT WHEN ... %NOTFOUND par IF NOT FOUND THEN EXIT; END IF;
- Remplacement de SQL%NOTFOUND par NOT FOUND
- Remplacement de SYS_REFCURSOR par REFCURSOR
- Remplacement de INVALID_CURSOR par INVALID_CURSOR_STATE
- Remplacement de ZERO_DIVIDE par DIVISION_BY_ZERO
- Remplacement de STORAGE_ERROR par OUT_OF_MEMORY

- Le code est dans le module Perl Ora2pg/PLSQL.pm et la fonction : plsql_to_plpgsql() toute contribution est la bienvenue

Futur d'Ora2Pg

- Hébergement du code source sur github.org
- Option pour la création d'un squelette de projet avec création des scripts d'export/import
- Option d'évaluation du coût de migration d'une base Oracle
- Option pour extraire un rapport sur le contenu de la base Oracle
- Appel d'une fonction de modification des données suivant le type, la table et la colonne
- Export de données de type composites et XML

Questions ?

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

Http://2011.pgDay.eu/ , Amsterdam