HMIN328 : migration de schémas

1. Objectifs

L'objetif du TP est d'aborder certaines tâches dévolues à l'export de schémas de bases de données et éventuellement de lots de données.

1. Il s'agit d'une part d'exploiter le paquetage DBMS_METADATA pour assurer l'export de schéma dans une stratégie de migration de bases de données (d'Oracle vers PostgreSQL par exemple).
2. Il s'agit d'autre part de compléter l'export de schéma par de l'export de données (au format tabulé) en exploitant au mieux les vues du méta-schéma.

2. Export de schéma

2.1 Préalable

Le paquetage DBMS_METADATA [1] intègre différentes fonctions et procédures qui vont faciliter les travaux de manipulation portant sur les schémas de bases de données. Nous exploitons plus particulièrement ici :

- les fonctions GET_DDL et GET_DEPENDENT_DDL (voir aussi GET_XML et GET_DEPENDENT_XML) qui permettent d'afficher l'ordre de création d'un objet en particulier ou bien d'un schéma utilisateur en particulier. Ces fonctions renvoient une valeur de type CLOB (Character Large Object Binary).
- la procédure SET_TRANSFORM_PARAM qui propose différents variantes à l'affichage : point-virgule final, informations sur le stockage, ...au travers d'une prise en charge de couples propriétés - valeurs.

Des exemples de fonctionnement vous sont donnés :

Table à tout faire DUAL

```
-- gestion CLOB
set long 40000
select DBMS_METADATA.GET_DDL('TABLE','COMMUNE') from DUAL;
```

Vues méta-schéma avec écriture dans un fichier

```
spool emp.sql
set long 40000
set head off echo off
```

1. desc dbms_metadata pour en visualiser le contenu

```
exec
dbms_metadata.set_transform_param(dbms_metadata.session_transform,'SEGMENT_ATTRIBUTES',false);
exec dbms_metadata.set_transform_param(dbms_metadata.session_transform,'STORAGE',true);
exec  dbms_metadata.set_transform_param(dbms_metadata.session_transform,'TABLESPACE',false);
select dbms_metadata.get_ddl('TABLE','REGION', USER) from user_tables;
spool off
```

Toutes les ordres de création de tables

```
exec dbms_metadata.set_transform_param(dbms_metadata.session_transform,'SQLTERMINATOR',true);
SELECT dbms_metadata.get_ddl('TABLE', TABLE_NAME) FROM user_tables;
```

2.2 Exercices

1. Vous construirez une fonction nommée **ToutesTables** qui renvoie les ordres de création de toutes les tables d'un schéma utilisateur dont le nom est passé en paramètres d'entrée (variable de sortie de type CLOB).
2. Vous construirez une nouvelle fonction nommée **ToutesTablesInfos** qui renvoie les ordres de création (plus l'information concernant l'organisation logique et les paramètres associés au stockage physique) de toutes les tables d'un schéma utilisateur dont le nom est passé en paramètres d'entrée.

3. Export XML Schéma et données

3.1 Fonction GET_XML du paquetage DBMS_METADATA

1. Vous construirez une fonction nommée **TableXML** qui renvoie la description XML d'une table en particulier du schéma utilisateur (nom de la table passé en paramètres d'entrée).
2. Vous exploiterez le paquetage DBMS_XMLGEN pour avoir également les feuilles de l'arborescence XML (données) pour une table donnée. Construisez également une fonction à ce sujet **TableDataXML** qui prend en argument le nom d'une table mais aussi une condition de filtre (clause where).

```
set heading off
set pagesize 0
 select dbms_xmlgen.getxml
             (dbms_xmlgen.newcontext
               ('select * from REGION')
             )
      from dual;
```

4. Export des données

Des modules d'import/export sont en général rendus disponibles au sein des SGBDs. Pour Oracle, il s'agit des modules expdp (export datapump) et impdp (import datapump) ainsi que du paquetage DBMS_DATAPUMP, que nous n'utiliserons pas ici. L'idée, lorsque l'on procède à de la migration de bases de données, est de pouvoir gérer ce qui est désigné par dump (chargement) des tables. Nous

allons simplement aborder l'export des données au format tabulaire avec une commande select qui retourne toutes les données d'une table et une utilisation appropriée des variables d'édition de sqlplus. Le paquetage UTL_FILE donne des fonctionnalités pour la création, l'ouverture, l'écriture et la lecture dans des fichiers externes à la base de données. Nous ne l'utiliserons pas non plus.

```
set pages 0
set feedback off
set heading off
set trimspool off
set termout off
set verify off
set colsep ""
set tab off
set linesize 100

SPOOL ON
SPOOL file_out
select *  from region ;
SPOOL OFF
```

Vous réfléchirez à partir de l'exemple suivant à construire une procédure qui permet de prendre en charge un caractère de séparation en bonne et due forme pour le fichier résultat (la tabulation ou bien le point-virgule ou bien la virgule ...). chr(9) et chr(13) valent respectivement pour la tabulation et le retour chariot.

```
set linesize 120
set serveroutput on

create or replace procedure factory_region is
cursor reg is select * from region;
begin
for reg_t in reg
loop
dbms_output.put_line(reg_t.region||chr(9)||reg_t.ncc||chr(9)||reg_t.cheflieu||chr(13)) ;
end loop ;
exception
when others then dbms_output.put_line('Pb sur l''affichage ') ;
end ;
/

spool on
spool file_reg
set serveroutput on
execute factory_region;
spool off
```

Vérifier la validité de votre fichier de sortie en chargeant son contenu tabulé (à l'aide de SQL Loader) dans une table créée pour l'occasion.