HMIN328 : chargement de données et migration de schémas (à rendre pour le 4/12/20)

1. Chargement de données

L'objet de la section est d'explorer les mécanismes (module de chargement SQL Loader) qui permettent de charger les données provenant d'un fichier au format tabulé (CSV pour Comma Separated Value) au sein de tables d'une base de données relationnelle.

1.1 Schéma partiel de la base de données Communes

Le schéma relationnel de la table **Population** est donné, vous créerez la table associée. Cette table vient compléter les données, concernant les communes, déjà exploitées lors des TPs précédents. Les attributs supportant la contrainte de clé primaire sont en gras. Les types des attributs vous sont également indiqués.

— Population(**codeinsee varchar(6), annee integer**, val_population integer)

1.2 Construction de l'ensemble des tables et chargement

L'alimentation de la table **Population** se fera au travers de l'utilitaire de chargement Oracle nommé SQL Loader. Vous disposez, pour ce faire, d'un fichier de données au format tabulé, fourni sur moodle (tuples.csv). À noter, que les données sur les communes ne concernent que la France métropolitaine (et la Corse). Par contre, les données sur les régions et départements portent également sur les ROM-COM. Un fichier de contrôle est à construire, en vous inspirant des contenus de fichiers présentés ci-dessous. Ces fichiers de contrôle vous permettent de définir les modalités d'insertion des données dans les tables.
Un schéma informel de chargement des données au travers de SQL Loader vous est fourni.

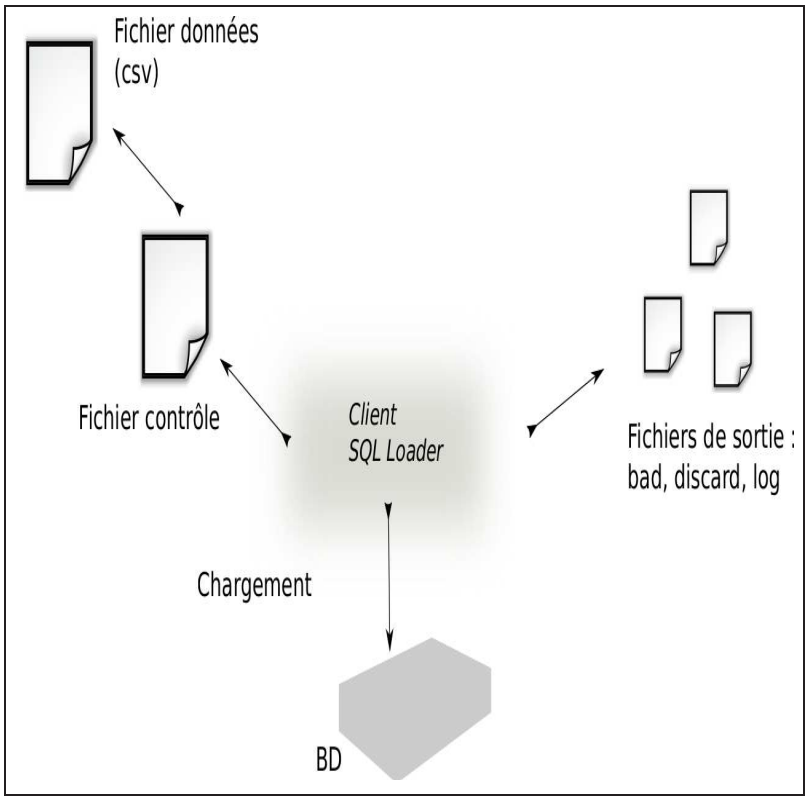

FIGURE 1 – Diagramme illustrant le module SQL Loader

A titre d'illustration, deux fichiers de contrôle sont présentés ci-dessous. Ils permettent de charger des données depuis un fichier csv dans une table donnée. Le caractère de séparation dans le fichier tabulé est soit la tabulation, ici représentée par X'9' ; soit le point-virgule.

```
LOAD DATA
CHARACTERSET UTF8
INFILE 'Region.csv'
APPEND
INTO TABLE REGION
FIELDS TERMINATED BY X'9'
(reg  "to_char(:reg)",
chef_lieu "to_char(:chef_lieu)",
nom_reg "to_char(:nom_reg)"
)
```

```
LOAD DATA
CHARACTERSET UTF8
INFILE 'isf_2002.csv'
TRUNCATE
INTO TABLE Impot
FIELDS TERMINATED BY ';'  trailing nullcols
(  CodeImp "impot_cpt.nextval",
CodeInsee "to_char(:CodeInsee)",
```

```
nbreredevables "to_number(:nbreredevables)",
annee "to_number(:annee)"
)
```

L'appel à sql loader en ligne de commande est de la forme :

```
sqlldr userid=nomUser/mdpUser@oracle.etu.umontpellier.fr:1523/pmaster control=fichier_ctl.txt
```

1.3 Eléments à rendre dans le devoir

1. script de création de la table
2. fichier de contrôle construit
3. sortie du résultat de la requête sur la vue user_tables indiquant en particulier le nombre de tuples inséré et le nombre de blocs de données alloués

2. Migration de schémas

L'objectif de la second section, est d'aborder certaines tâches dévolues à l'export de schémas de bases de données et éventuellement de lots de données. Vous pouvez définir des paquetages PL/SQL qui vous aideront à répondre à l'ensemble des questions posées.

1. Il s'agit d'exploiter le paquetage DBMS_METADATA pour assurer l'export de schéma dans une stratégie de migration de bases de données (d'un serveur de données Oracle vers un serveur de données PostgreSQL par exemple).
2. Il s'agit de compléter l'export de schéma par de l'export de données (au format tabulé) en exploitant au mieux les vues du méta-schéma.
3. Il s'agit également de revenir sur les notions associées à l'organisation logique et physique des données (étudiées dans le cours *architecture Oracle*)

2.1 Export de schéma

2.1.1 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 différentes 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 :

1. desc dbms_metadata pour en visualiser le contenu

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
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
```

Ordres de définition de l'ensemble des tables

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

2.1.2 Code PL/SQL à rendre

1. Vous construirez une procédure nommée **UneTable** qui affiche l'ordre de création d'une table (nom passé en paramètres d'entrée) de votre schéma utilisateur.
2. Vous construirez une fonction nommée **ToutesTables** qui renvoie les ordres de création de toutes les tables (sans les informations concernant le stockage) d'un schéma utilisateur dont le nom est passé en paramètres d'entrée (variable de sortie de type CLOB). Vous utiliserez, pour ce faire, une requête de la forme (ici schéma utilisateur HR) :

   ```
   select dbms_metadata.get_ddl('TABLE',TABLE_NAME,'HR') FROM DBA_TABLES WHERE OWNER ='HR';
   ```

3. 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 paramètre d'entrée.

2.1.3 Informations associées à l'organisation physique de la table

Le schéma de la table est exportable avec toutes les informations concernant le stockage des données de la table et présence éventuelle d'index et de contraintes. Un exemple vous est donné. Vous essaierez de comprendre tous les paramètres qui président à l'organisation de vos tables, à partir de l'exemple mais aussi des explications concernant les différents paramètres liés au stockage.
À rendre : Vous rendrez une version commentée de la table population à partir de toutes les indications qui vous sont données dans cette section.

```
CREATE TABLE "ISA"."REGION"
   ( "REG" VARCHAR2(4),
"NOMC" VARCHAR2(50),
```

```
"CHEF_LIEU" VARCHAR2(46),
"NOM_REG" VARCHAR2(30),
 CONSTRAINT "REGION_PK" PRIMARY KEY ("REG")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
  TABLESPACE "USERS"  ENABLE
   ) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
  TABLESPACE "USERS" ;
```

Les informations associées à la clause de stockage (storage) concernent, en premier lieu, les segments de table ou d'index par le biais du dimensionnement des extensions (ou extents) qui sont amenées à faire évoluer la structure de stockage d'une table ou d'un index. Pour rappel, l'organisation logique d'une base de données se compose de plusieurs espaces de tables (tablespaces) qui sont structurés en segments, lesquels sont organisés en extents qui à leur tour sont des collections de blocs.
Paramètres de stockage : les valeurs sont choisies de façon à minimiser le nombre d'extents affecté à chaque objet.

- INITIAL (integer) taille en octets du premier "extent"
- NEXT (integer) taille en octets du "extent" suivant
- MINEXTENTS (integer) : nombre d'extents alloués à la création
- MAXEXTENTS (integer) : nb maximum d'extents
- PCTINCREASE (integer) : pourcentage d'augmentation entre 2 extents (par defaut 50)
- PCTFREE (integer) : pourcentage d'espace réservé dans chaque bloc pour des mises à jour à venir (par défaut 10%)
- PCTUSED (integer) : pourcentage minimum d'espace utilisé dans un bloc (40% par défaut) avant de pouvoir insérer de nouveaux tuples après des suppressions (somme PCTFREE + PCTUSED doit être inférieure à 100)
- INITRANS (integer) : nombre initial d'entrées de transactions pré-allouées à un bloc (1-255) ; chaque transaction qui modifie un bloc demande une entrée dans le bloc (par défaut 1 pour un segment de table, 2 pour un segment d'index)
- MAXTRANS integer : nombre maximum de transactions concurrentes qui peuvent modifier un bloc alloué à une table (1-255)
- FREELIST GROUPS integer
- FREELIST integer

2.1.4 Informations sur les organisations logique/physique

Vous tirerez parti des vues : dba_segments, dba_extents, dba_objects et du paquetage dbms_rowid, pour retourner le plus d'informations possibles sur l'espace utilisé par la table Population et l'organisation de cet espace. **À rendre :** Vous pouvez construire une ou différentes procédures à ce sujet.

```
select owner, segment_type, count(segment_name), sum(bytes) octets, sum(blocks) blocs,
sum(extents) extensions from dba_segments group by owner, segment_type order by octets asc;
```

2.2 Export XML Schéma et données

2.2.1 Fonction GET_XML du paquetage DBMS_METADATA

À rendre : les deux fonctions sont à rendre

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 paramètre 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 nom_reg from REGION')
             )
      from dual;
```

3. 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
```

À rendre : Vous définirez une procédure qui permet de renvoyer les données correspondant au codeinsee, nom de la commune, valeur de la population en 2000, et valeur de la population en 2010 au format tabulé.