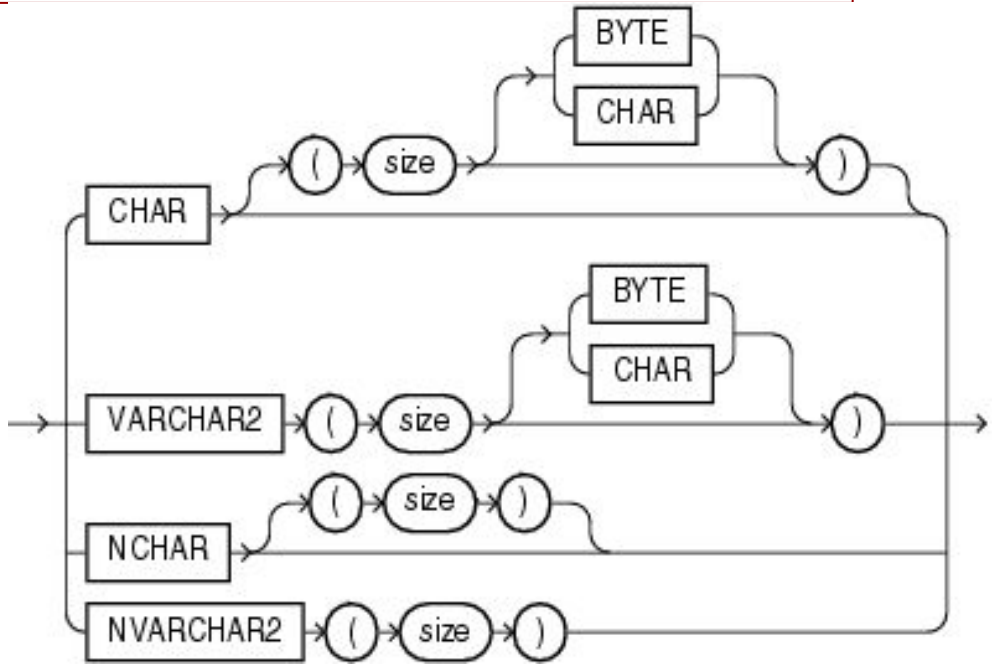
Les types de données sous Oracle CHAR, VARCHAR2

Types	Description	Size
VARCHAR2(n)	Variable-length character string.	From 1 byte to 4KB.
NVARCHAR2(size)	Variable-length Unicode character string having maximum length size characters.	Maximum size is determined by the national character set definition, with an upper limit of 4000 bytes. You must specify size for NVARCHAR2.
Char	Fixed length	



{ **CHAR** [(size [**BYTE** | **CHAR**])]
| **VARCHAR2** (size [**BYTE** | **CHAR**])
| **NCHAR** [(size)]
| **NVARCHAR2** (size)
}

Juin 2015

https://docs.oracle.com/cd/B28359_01/server.111/b28286/sql_elements001.htm#SQLRF0021

Char VS Varchar2

Comme l'indique le nom des types :

- Utiliser char lorsque vous avez une longueur fixe (Exemple numéro de tél sur 10)
- Utiliser char , pour optimiser l'espace de stockage

Table B-2 Mapping ANSI Data Types to Oracle Data Types

ANSI	Oracle
SMALLINT	NUMBER(5)
INTEGER	NUMBER(10)
NUMERIC(p,s)	NUMBER(p,s)
FLOAT	FLOAT(23)
DOUBLE PRECISION	FLOAT(49)
VARCHAR	VARCHAR2
DATE	DATE
TIME	DATE
TIMESTAMP	DATE

Char VS Varchar2

Comme l'indique le nom des types :

- Utiliser char lorsque vous avez une longueur fixe (Exemple numéro de tél sur 10)
- Utiliser varchar , pour optimiser l'espace de stockage

Soit la table **Ora_Type_char** qui contient une colonne dont la **longueur est fixe**

```
CREATE TABLE Ora_Type_char (
    password_Fixe char(15)) ;
```

Et la table **Ora_Type_varchar** qui contient une colonne dont la l**ongueur est variable**

```
CREATE TABLE Ora_Type_char_v (
    col1 varchar2(15)) ;
```

EXERCICE : peuplez les 2 tables avec le même nombre d'enregistrements et comparez leur taille en se servant la requête suivante :

```
select segment_name,segment_type, sum(bytes) Octets
 from USER_SEGMENTS
group by segment_name,segment_type;
```

CHAR VS VARCHAR2

```
declare
i int ;

Begin
    for i in 1 .. 100000
Loop
   insert into Ora_Type_char (tel) select  dbms_random.string('A', 15)  from dual;
end loop;
end;
```

Créer une table char mais nop - avec des longueurs varaiables :

Même espace est réservé.

Créer comme suit une table dont la longueur est variable :

```
declare
i int ;
n_aleatoire int;
Begin
    for i in 1 .. 100000
Loop
  select  dbms_random.value(1, 15) into n_aleatoire  from dual;
   insert into Ora_Type_char_nop (tel) select  dbms_random.string('A', n_aleatoire)  from dual;
end loop;
end;
```

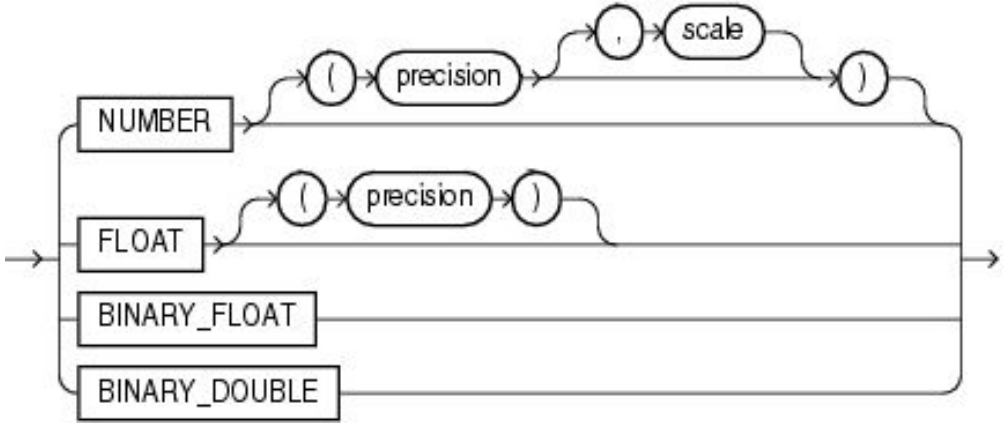
```
CREATE TABLE Ora_Type_char_v2 (  tel varchar2(15)) ;
  insert into Ora_Type_char_v2  select rtrim(tel) from  Ora_Type_char_nop;
```

LES NOMBRES SOUS ORACLE

Table B-2 Mapping ANSI Data Types to Oracle Data Types

ANSI	Oracle
SMALLINT	NUMBER(5)
INTEGER	NUMBER(10)
NUMERIC(p,s)	NUMBER(p,s)
FLOAT	FLOAT(23)
DOUBLE PRECISION	FLOAT(49)
VARCHAR	VARCHAR2
DATE	DATE
TIME	DATE
TIMESTAMP	DATE

LES NOMBRES SOUS ORACLE



x number (6,2) ⟷ p = 6, 1234.56, s = 2

{ NUMBER [(precision [, scale])]
| FLOAT [(precision)]
| BINARY_FLOAT
| BINARY_DOUBLE
}

Un nombre de longueur **p** dont **s** décimaux (au **s**-ième près).

x = 1234.56 ⟶ OK

x = 1234.56 7⟶ OK sera arrondi à 1234.57

Exercices 1

```
  CREATE TABLE Ora_Type_demo ( number_value NUMERIC(13, 2));
--
insert into ora_type_demo (number_value)  values (12345678901.23); -- s < p
select * from ora_type_demo;

---------
insert into ora_type_demo (number_value)  values (12345678901.23); -- s < p
select * from ora_type_demo;

--------------------------
alter table ora_type_demo add number_ps_neg number(5,-2);
insert into ora_type_demo (number_ps_neg) values (12345);
```

CASE

```
CASE SELECTOR
   WHEN EXPRESSION 1 THEN STATEMENT 1;
   WHEN EXPRESSION 2 THEN STATEMENT 2;
   ...
   WHEN EXPRESSION N THEN STATEMENT N;
   ELSE STATEMENT N+1;
END CASE;
```

Exercice : Reprendre l'exercice précédent avec CASE

Demander à l'utilisateur de saisir une date,

- ensuite récupérer le jour : **to_char**(date, 'fmt')
- si LUNDI alors afficher 'Au travail'
- si MARDI alors afficher 'Repose'
- si MERCREDI alors afficher 'Pour les enfants'
- si JEUDI alors afficher 'Déjà fatigué'
- si VENDREDI alors afficher 'Vivement le week end'
- autres afficher week-end.

CASE

Exercice : Demander à l'utilisateur de saisir une date,

- ensuite récupérer le numéro du jour grace à la commande : **to_char**(date, 'D')
- et dire s'il s'agit du week-end ou pas.

SWITCH - CASE

```
SET SERVEROUTPUT ON
DECLARE
    v_date DATE := TO_DATE('&sv_user_date', 'DD/MM/YYYY');
    v_day VARCHAR2(1);
BEGIN
    v_day := TO_CHAR(v_date, 'D');
CASE v_day WHEN '1' THEN DBMS_OUTPUT.PUT_LINE ('LUNDI');
    WHEN '2' THEN DBMS_OUTPUT.PUT_LINE ('MARDI');
    WHEN '3' THEN DBMS_OUTPUT.PUT_LINE ('MERCREDI');
    WHEN '4' THEN DBMS_OUTPUT.PUT_LINE ('JEUDI');
    WHEN '5' THEN DBMS_OUTPUT.PUT_LINE ('VENDREDI');
    WHEN '6' THEN DBMS_OUTPUT.PUT_LINE ('SAMEDI');
    WHEN '7' THEN DBMS_OUTPUT.PUT_LINE ('DIMANCHE');
END CASE;
END;
```

NULLIF

A = NULLIF (expression1, expression2)

A sera null si expression1 = expression2,
si non ce sera la valeur de la 1ere expression.

Exercice : Demander à l'utilisateur de saisir un nombre et afficher s'il est pair ou impair.

```
set SERVEROUTPUT ON

declare
v_num number := &u_Num;
v_Reste number;
Begin
  v_reste := nullif(Mod(v_num, 2),0);
  if v_reste is null then
    dbms_output.put_line ('Le reste est null');
  else
    dbms_output.put_line ('Le reste est : '||v_reste);
  end IF;
end;
```

LES TABLEAUX

Les variables de type TABLE permettent de définir et de manipuler des tableaux dynamiques.

Un tableau est composé d'une clé primaire et d'une colonne (de type scalaire, TYPE, ROWTYPE) pour stocker chaque élément.

Syntaxe

La syntaxe générale pour déclarer un type de tableau et une variable tableau est la suivante :

```
TYPE nomTypeTableau IS TABLE OF

                        {typeScalaire |
                         variable%TYPE | table.colonne%TYPE} [NOT NULL]
                | table.%ROWTYPE
                [INDEX BY BINARY_INTEGER];
declare nomTableau nomTypeTableau
```

Exercice : créer un tableau d'entiers puis afficher le tableau trier

LES BOUCLES

Elles permettent de faire un traitement itératif

```
LOOP
        -- instrcution 1
        -- instrcution 2
        -- instrcution 3
END LOOP
```

LES BOUCLES : EXIT pour finir

```
SET SERVEROUTPUT ON
DECLARE
v_counter BINARY_INTEGER := 0;
BEGIN
    LOOP
        v_counter := v_counter + 1;
        DBMS_OUTPUT.PUT_LINE ('v_counter = '||v_counter);
        IF v_counter = 5 THEN
            EXIT;
        END IF;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE ('Done...');
END;
```

LES BOUCLES

Exercice :
Dans le schéma HR,
créer une table client (id int auto-incrémenté, info varchar2(50))

Insérer 100 enregistrements dans la table client.

(envoyez vos scripts en privé)

LES BOUCLES : EXIT pour finir

```
SET SERVEROUTPUT ON
DECLARE
v_counter BINARY_INTEGER := 0;
BEGIN
LOOP
v_counter := v_counter + 1;
DBMS_OUTPUT.PUT_LINE ('v_counter = '||v_counter);
EXIT WHEN v_counter = 5;
END LOOP;
DBMS_OUTPUT.PUT_LINE ('Done...');
END;
```

LES BOUCLES : while

```
WHILE CONDITION   LOOP
        -- instrcution 1
        -- instrcution 2
        -- instrcution 3
 END LOOP
```

LES BOUCLES : while

```
DECLARE
v_counter NUMBER := 5;
BEGIN
WHILE v_counter < 5 LOOP
DBMS_OUTPUT.PUT_LINE ('v_counter = '||v_counter);
v_counter := v_counter - 1;
END LOOP;
END;
```

LES BOUCLES : while - exit prématuré

```
DECLARE
v_counter NUMBER := 1;
BEGIN
WHILE v_counter <= 5 LOOP
DBMS_OUTPUT.PUT_LINE ('v_counter = '||v_counter);
IF v_counter = 2 THEN
EXIT;
END IF;
v_counter := v_counter + 1;
END LOOP;
END;
```

LES BOUCLES :

Exercice : Donner la sommes des nombres paires et la somme des nombres impairs entre 1 et 100

LES BOUCLES : while - exit prématuré

```
SET SERVEROUTPUT ON
DECLARE
v_counter BINARY_INTEGER := 2;
v_sum NUMBER := 0;
BEGIN
WHILE v_counter <= 100 LOOP
v_sum := v_sum + v_counter;
DBMS_OUTPUT.PUT_LINE ('la somme est : '||v_sum);
-- increment loop counter by two
v_counter := v_counter + 2;
END LOOP;
-- control resumes here
DBMS_OUTPUT.PUT_LINE ('la somme des nombre impairs : '||
'1 et 100 est : '||v_sum);
END;
```

LES BOUCLES - FOR

```
FOR i in v_valeur1 .. v_valeur2
LOOP
  -- instructions ...
END LOOP;
--- autres instructions ...
```

Exercice : Calculer le factoriel de 10

LES BOUCLES : exercice 10!

```
SET SERVEROUTPUT ON
DECLARE
v_factorial NUMBER := 1;
BEGIN
FOR v_counter IN REVERSE 1..10 LOOP
v_factorial := v_factorial * v_counter;
END LOOP;
-- control resumes here
DBMS_OUTPUT.PUT_LINE
('le factoriel de 10 : '||v_factorial);
END;
```

Exercice : En combien de temps votre ordinateur va incrémenter un compteur 900 000 000 000 de fois ?
indication : utiliser DBMS_UTILITY.GET_TIME;

PACKAGE

Un PACKAGE regroupe un ensemble de fonctions, de procédures dans une UNITÉ

Un PACKAGE est composé de deux parties: l'entête (**spécification - déclarative**) et le corps (**définition**).

1- spécification

```
CREATE [OR REPLACE] PACKAGE <NOM_PACKAGE> AS

    <DECLARATION PROCEDURE ET/OU FUCTION>

END;
```

2- corps

CREATE [OR REPLACE] PACKAGE **BODY** <NOM_PACKAGE> AS

<CONTENU PROCEDURE ET/OU FUNCTION>

TP : Package qui contient :

- une procédure stockée qui affiche le nom d'un employé si on lui fournit son ID
- Un fonction qui calcul la somme de 2 nombres

introduction au CURSORS

Un curseur “cursor” permet de parcourir le résultat d’une requête et d’appliquer un traitement à chaque enregistrement

```
CURSOR c_cursor_name IS select statement
```

Les étapes ont :

- La définition du curseur : **CURSOR IS SELECT** …
- Ouverture du curseur : **OPEN CURSOR**
- **FETCH** : pour récupérer l’enregistrement
- **EXIT** : Sortir du curseur
- **CLOSE** pour fermer le curseur

Exemple : un curseur qui calcul la somme des salaires de la table employees.

Exercice

- On veut la somme des salaires pour les employés dont l'id est pair et la somme des salaires pour les employés dont l'id est impair. 10min

RAPPEL - suite

```
select * from article;

declare
CURSOR c_art IS SELECT * FROM ARTICLE;
-- créer une variable dans laquelle, on récupère la position du curseur
vArt c_art%ROWTYPE;
BEGIN
    -- on ouvre le curseur
    OPEN c_art;
    LOOP
        -- récupérer l'enregistrement pointé par le curseur et le mettre dans une variable
        FETCH c_art INTO vArt;
        EXIT WHEN c_art%NOTFOUND;
            -- traitement
            dbms_output.put_line (vArt.Descriptif);
        END LOOP;
    CLOSE c_art;

END;
```

RAPPEL - suite

```
declare
cursor c_recup is select designation, idcat from article ;
v_disg article.designation%type;
v_Idcat article.idcat%type;

begin
   open c_recup;
   loop
      fetch c_recup into v_disg,v_Idcat;
       Exit when c_recup%notfound;
      DBMS_OUTPUT.PUT_LINE('ensemble :'||v_disg||' :'||v_idcat );

   end loop;
   close c_recup;
end;
```

CREATION D'UN UTILISATEUR

Syntax for tables

```
GRANT privilege-type ON [TABLE] { table-Name | view-Name } TO grantees
```

Syntax for routines

```
GRANT EXECUTE ON { FUNCTION | PROCEDURE } routine-designator TO grantees
```

Syntax for sequence generators

```
GRANT USAGE ON SEQUENCE [ schemaName. ] SQL92Identifier TO grantees
```

Lors de la création d'un utilisateur, il faut le grant pour :

- créer les ressources (table, trigger, …)
- droits sur les tablespace (pour insert)

CHARGEMENT DE DONNEES DANS UNE TABLE Oracle

Activité :

Créer une table Departements pour charger la liste des départements

https://www.data.gouv.fr/fr/datasets/liste-des-departements-francais-metropolitains-doutre-mer-et-les-com-ainsi-que-leurs-prefectures/#resources

Ecrire un script PL/SQL pour ce besoin. 15min

CREATION D'UN UTILISATEUR

- https://www.data.gouv.fr/fr/datasets/communes-de-france-base-des-codes-postaux/
- https://docs.oracle.com/cd/B19306_01/server.102/b14215/ldr_concepts.htm EXEMPLE 6
- sqlldr hr/hr control=local.ctl log=local.log
- EXEMPLE 6
- sqlldr hr/hr control=local.ctl log=local.log
-

UPLOAD DATA VIA PLSQL

https://www.data.gouv.fr/fr/datasets/liste-des-films-aides-selectionnes-au-festival-de-cannes-2003-2014/#resources

UPLOAD DATA l'outil UTL_FILE

Avec le package UTL_FILE, les programmes PL/SQL peuvent lire et écrire des fichiers texte du système d'exploitation.

1- Créer un répertoire virtuel

```
drop  DIRECTORY out_dir
CREATE DIRECTORY out_dir AS 'C:\DORANCO';
GRANT READ ON DIRECTORY out_dir TO eranch;

show parameter UTL_FILE_DIR
```

2- ouvrir : FILE_OPEN

3- lire une ligne : GET_LINE

4 - fermer le fichier : FILE_CLOSE

UPLOAD DATA VIA PLSQL

DPT

```
--drop directory rep1;
CREATE or replace DIRECTORY rep1 AS 'C:\ImportOracle';
GRANT READ  ON DIRECTORY rep1 TO PUBLIC;

select directory_name, directory_path
from all_directories;
------------
select grantee  , privilege
   from all_tab_privs
 where table_name = 'REP1';
```

UPLOAD DATA VIA PLSQL

```
DECLARE
                V1 VARCHAR2(32767);
                F1 UTL_FILE.FILE_TYPE ;
                 v_code departement.num_dep%type;
                 v_nom departement.dep_name%type;
                 v_region departement.region_name%type;
                 v_pos_1 int;
                 v_pos_2 int;
BEGIN
                 F1 := UTL_FILE.FOPEN('REP1','departements-region.csv','R',256);
                loop
                                UTL_FILE.GET_LINE(F1,V1,256);
                                 -- code jusuq'a la postion de la virgule = position de la 1ere virgule -1
                                 -- nom : entre premeire et 2 eme virgule
                                 -- region : à partie de la 2ème virgule jusqu'a la fin
                                 -- pour récupérér la position d'un caractère dans un string
                                 -- on utilise la fonction instr(
                                 v_pos_1 := instr(v1,',',1,1);
                                 v_pos_2 := instr(v1,',',1,2);
                                 --substr
                                 v_code := substr(v1,1,v_pos_1-1);
                                 v_nom := substr(v1,v_pos_1+1,v_pos_2-v_pos_1-1);
                                 v_region := substr(v1,v_pos_2+1);

                                   insert into departement values (v_code,v_nom,v_region);
                                dbms_output.put_line(v_code||'#'||v_nom||'#'||v_region);


                end loop;
                 UTL_FILE.FCLOSE(F1);
                 exception when others then
                  dbms_output.put_line('fin du fichier');
END;
```

UPLOAD DATA VIA PLSQL

```
DECLARE
  V1 VARCHAR2(32767);
  F1 UTL_FILE.FILE_TYPE ;
  v_titre varchar(100);
  v_realisateur varchar(100);
  v_section varchar(100);
   v_prix varchar(100);
  v_annee varchar(100);
  v_indice int := 0;
  v_nb_col int := 0;
  v_length int ;
BEGIN


  F1 := UTL_FILE.FOPEN('OUT_DIR','liste-des-films-aides-selectionnes-au-festival-de-cannes-depuis-2003 (4).csv','R');

begin
```

UPLOAD DATA VIA PLSQL

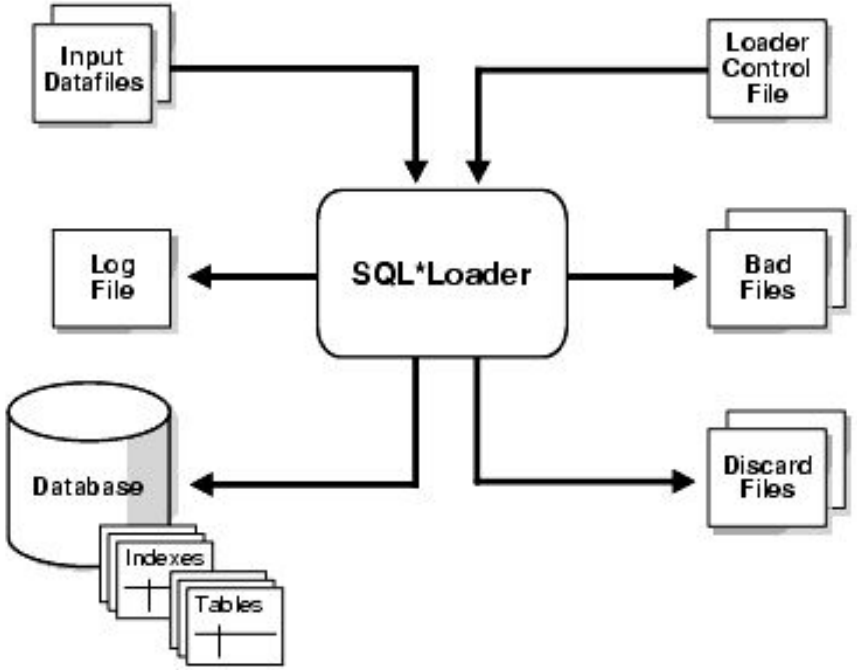
```
  k int;
BEGIN
  F1 := UTL_FILE.FOPEN('IMPORT_DIR','liste-dpt-drom-com-v1.2.csv','R
  loop
    utl_file.get_line(f1, v1);

   i := instr(v1,',',1,1);
    dbms_output.put_line('i :'||i);
    v_code := substr(v1,1,i-1);
    -------------
    k := instr(v1,',',1,2);
    v_typedep := substr(v1,i+1,k-i-1);
    i := k;
    --------------------------
     k := instr(v1,',',1,3);
     v_nom := substr(v1,i+1,k-i-1);
    i := k;
    ----------------
    insert into dep2 (code, type, nom) values (v_code,v_typedep, v_n
     dbms_output.put_line('le code est :'||v_code||' type:  '||v_typ
  end loop;
```

UPLOAD DATA VIA PLSQL

```
begin
loop
  UTL_FILE.GET_LINE(F1,V1);
--  dbms_output.put_line(v1);
  v_titre :=  substr(v1,1,  instr(v1, ';',1,1));
    v_realisateur :=   instr(v1, ';',1,2);
       v_section :=   instr(v1, ';',1,3);
          v_prix :=   instr(v1, ';',1,4);
             v_annee :=   instr(v1, ';',1,5);
                v_indice :=   instr(v1, ';',1,6);
          dbms_output.put_line(v_titre);
end loop;
end;
exception when no_data_found then
  dbms_output.put_line ('fin');
   UTL_FILE.FCLOSE(F1);
END;
```

Chargement (import) de fichiers sous Oracle : SQL LOADER et PL/SQL

SQL LOADER (sqlldr) est un outil Oracle qui permet le chargement de données à partir de fichiers plats et à destination d'Oracle.
Souple et efficace.



LE FICHIER CTL

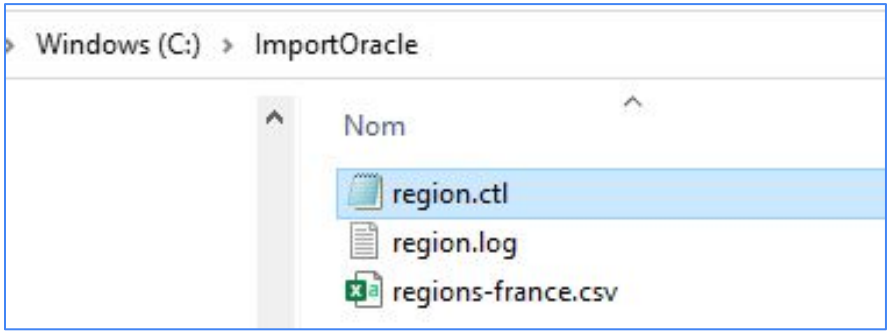
SYNTAXE :

load data
infile ‘fichier text’
into table nom_table
fields terminated by 'séparateur'
(attribut1, attribut2, attribut3, attribut4 …)

TP : Charger le fichier des régions dans une nouvelle table
https://www.data.gouv.fr/fr/datasets/liste-des-films-aides-selectionnes-au-festival-de-cannes-2003-2014/#resources

exemple

```
load data
infile 'regions-france.csv'
into table  region
fields terminated by ','
( code_region ,nom_region)
```




```
C:\ImportOracle>sqlldr hr/hr

control = region.ctl
```

LE FICHIER CTL

Exercice pour la table des departement

1 Télécharger le fichier depuis le site du gouv.data :
(https://www.data.gouv.fr/fr/datasets/departements-et-leurs-regions/)

2 Créer la table departement (regarder la structure dans le fichier téléchargé)
3 Créer le fichier CTL
4 Lancer sqlldr pour charger le fichier text
5 Vérifier le chargement en ouvrant le fichier log
6 Vérifier le même chargement en faisant un select sur la table departement