

Chapitre 6 :

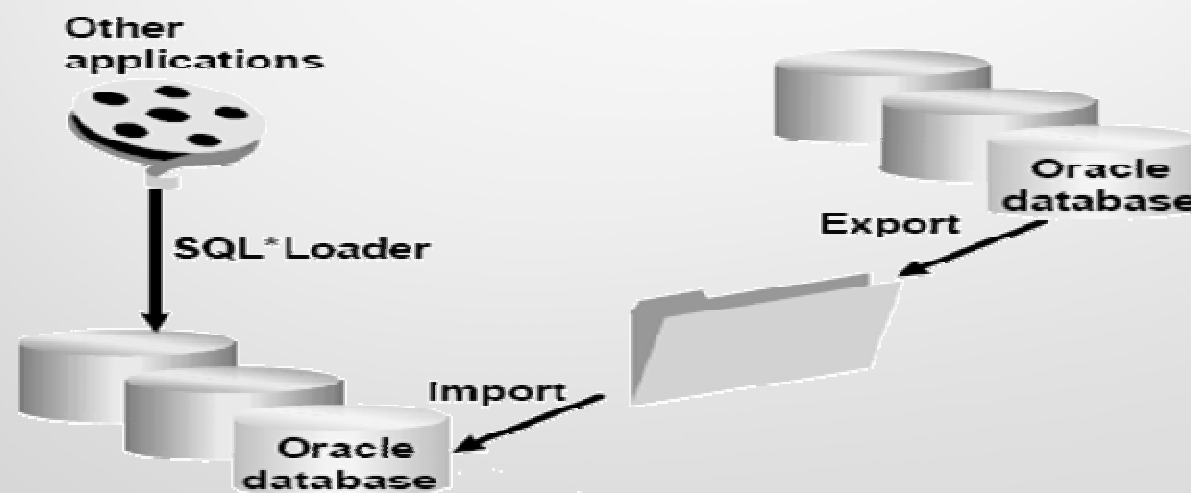
Transfert de données

Déplacer les données

Il existe plusieurs méthodes pour charger des datas dans les tables d'une BD Oracle:

SQL*Loader

Utilitaire d'export/import

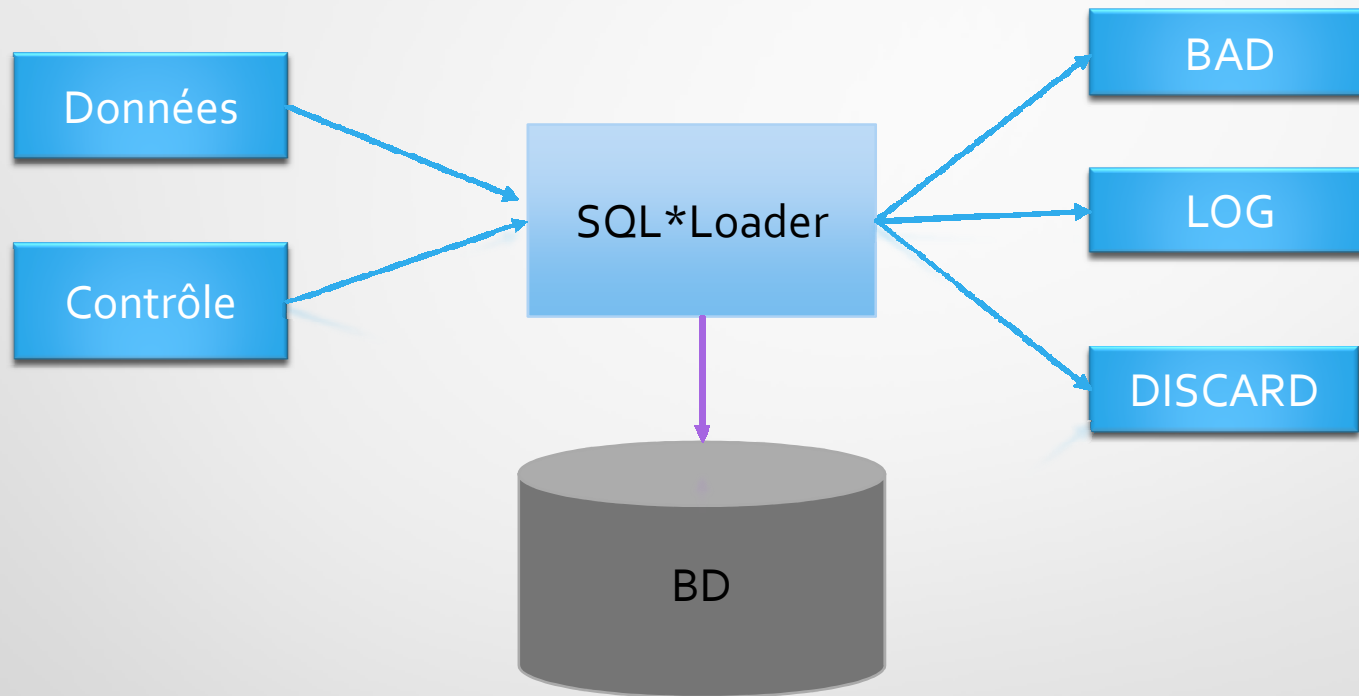


Import/Export

SQL*LOADER

- Permet le chargement d'une BD à partir de fichiers de données obtenus d'un autre environnement.
- Un module externe qui transforme les données externes lues à partir d'un fichier source pour former les commandes INSERT sous contrôle d'un fichier **.ctl**
- SQL*LOADER d'Oracle est le chargeur de données qui utilise 2 fichiers en entrée : le fichier de contrôle et le fichier de données à charger et produit 3 fichiers : .LOG,.BAD,.DSC

SQL*LOADER



Fichiers SQL*LOADER

- Le fichier **.BAD** : ranger les records rejetés par le chargeur comme étant non conformes aux spécification du fichier de contrôle ou non conformes avec la base.
- Le fichier **.LOG** : avoir une trace des activités de chargement:
 - le nombre d'enregistrements chargés
 - le nombre d'enregistrements non chargés vue des erreurs de données
 - le nombre d'enregistrements non chargés faute d'insatisfaction de la clause **When**
- Le fichier **.DSC** : conserver les données non chargées parce qu'elles ne vérifient pas la clause **When** du fichier de contrôle

Modes de chargement

- **Insert** insère les données dans une table vide
- **Append** insère les données à la suite des données existantes
- **Replace** insère les données en remplaçant les données existantes
- **Truncate** insère les données après un TRUNCATE

Exemple 1

etudiant.txt - Bloc-notes

Fichier Edition Format Affichage ?

4-MARAM
5-HAROUN
1-ALI
2-SALAH
3-MOHAMED

etudiant.ctl - Bloc-notes

Fichier Edition Format Affichage ?

LOAD DATA INFILE 'etudiant.txt'
TRUNCATE INTO TABLE etudiant
FIELDS TERMINATED BY '-' (num , nom)

```
E:\ESPRIT\2013-2014\DBA\cours>sqlldr td/td control=etudiant.ctl log=etudiant.log  
bad=etudiant.bad discard=etudiant.dsc;  
  
SQL*Loader: Release 11.2.0.1.0 - Production on Jeu. Déc. 5 18:39:28 2013  
Copyright (c) 1982, 2009, Oracle and/or its affiliates. All rights reserved.  
Point de validation <COMMIT> atteint - nombre d'enregis. logiques 4  
Point de validation <COMMIT> atteint - nombre d'enregis. logiques 5  
  
E:\ESPRIT\2013-2014\DBA\cours>exit  
  
SQL> select * from etudiant;  
  
  NUM NOM  
-----  
    4 MARAM  
    5 HAROUN  
    1 ALI  
    2 SALAH  
    3 MOHAMED
```

Options de chargement

- **Filtre:** On peut appliquer des filtres lors du chargement en utilisant l'instruction **When**

```
LOAD DATA INFILE 'fichier.txt' APPEND INTO TABLE etudiant when  
(3:53) <> 'ALI' FIELDS TERMINATED BY '-' ( num , nom )
```

- **Sauter un nombre de lignes**

```
sqllldr td10/td10 control=etudiant.txt log=etudiant.log bad=etudiant.bad  
discard=etudiant.dsc skip=2
```


Options de chargement

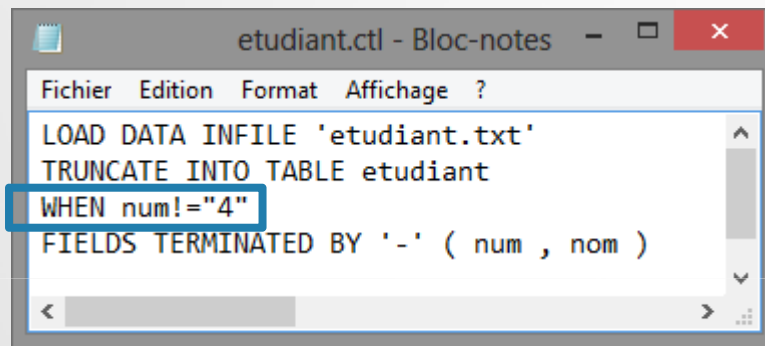
- **Modification des données lors du chargement**

```
LOAD DATA INFILE 'fichier.txt' TRUNCATE INTO TABLE etudiant FIELDS  
TERMINATED BY '-' (num, NOM "upper(:NOM)", moyenne  
":moyenne/100" )
```

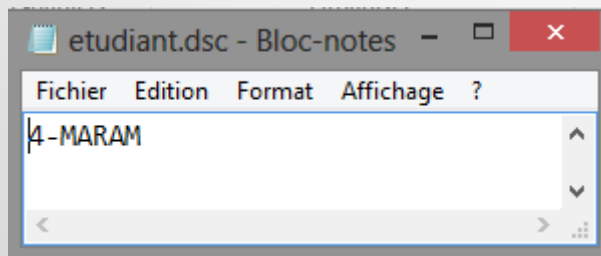
- **Arrêter le chargement après un nombre d'erreurs**

```
Errors = 2
```

Exemple1 (Suite)



```
etudiant.ctl - Bloc-notes
Fichier Edition Format Affichage ?
LOAD DATA INFILE 'etudiant.txt'
TRUNCATE INTO TABLE etudiant
WHEN num!="4"
FIELDS TERMINATED BY '-' ( num , nom )
```

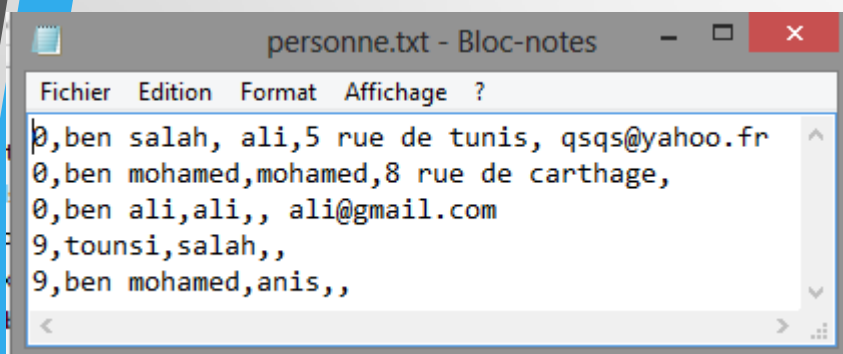


```
etudiant.dsc - Bloc-notes
Fichier Edition Format Affichage ?
4-MARAM
```

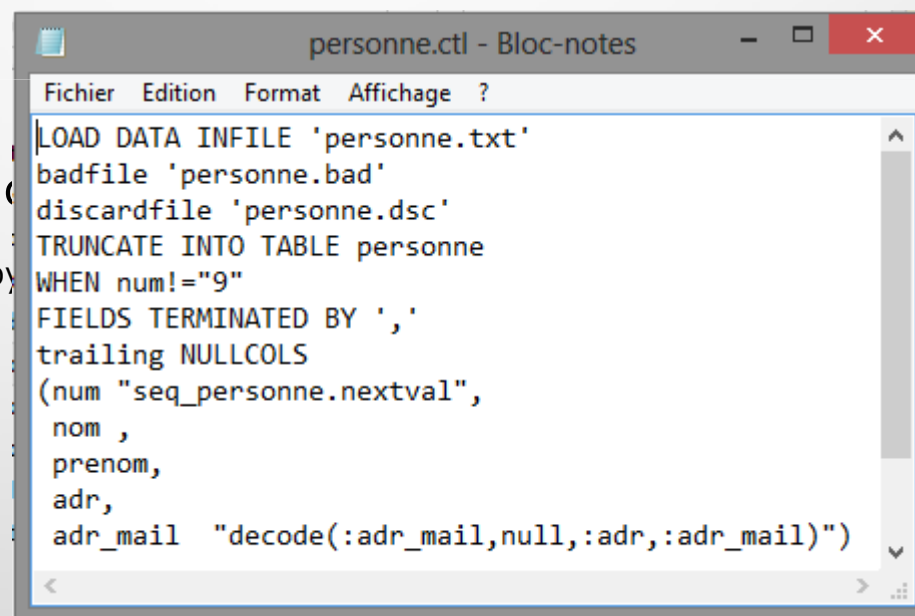
```
SQL> select * from etudiant;
```

NUM	NOM
5	HAROUN
1	ALI
2	SALAH
3	MOHAMED

Exemple 2 (Utilisation de séquence)



```
Fichier  Edition  Format  Affichage  ?
0,ben salah, ali,5 rue de tunis, qsqs@yahoo.fr
0,ben mohamed,mohamed,8 rue de carthage,
0,ben ali,ali,, ali@gmail.com
9,tounsi,salah,,
9,ben mohamed,anis,,
```



```
personne.ctl - Bloc-notes
Fichier  Edition  Format  Affichage  ?
LOAD DATA INFILE 'personne.txt'
badfile 'personne.bad'
discardfile 'personne.dsc'
TRUNCATE INTO TABLE personne
WHEN num!="9"
FIELDS TERMINATED BY ','
trailing NULLCOLS
(num "seq_personne.nextval",
 nom ,
 prenom,
 adr,
 adr_mail "decode(:adr_mail,null,:adr,:adr_mail)")
```

Exemple 2 (Suite)

```
personne.dsc - Bloc-notes
Fichier Edition Format Affichage ?
9,tounsi,salah,,
9,ben mohamed,anis,,
```

```
personne.log; - Bloc-notes
Fichier Edition Format Affichage ?
Table PERSONNE :
Chargement réussi de 3 Lignes.
0 Lignes chargement impossible dû à des erreurs de données.
2 Lignes chargement impossible car échec de toutes les clauses WHEN.
0 Lignes chargement impossible car tous les champs étaient non renseignés.
```

```
select * from personne;
```

Résultats

	NUM	NOM	PRENOM	ADR	ADR_MAIL
1	7	ben salah	ali	5 rue de tunis	qsqs@yahoo.fr
2	8	ben mohamed	mohamed	8 rue de carthage	8 rue de carthage
3	9	ben ali	ali	(null)	ali@gmail.com

Exemple 3: Chargement de plusieurs tables

SAFN.txt - Bloc-notes

Fichier Edition Format Affichage ?

```
SA00001 TITULAIRE
SA00005 OCCASIONNEL
SA00065 CONTRACTUEL
FN00001 CONTROLLEUR
FN00002 DGA
FN00003 OUVRIER
FN00004 AGENT DE SAISIE
```

SQL> select * from fonction;

FN_CODE	FN_LIB
FN00001	CONTROLLEUR
FN00002	DGA
FN00003	OUVRIER
FN00004	AGENT DE SAISIE

SQL> select * from Situation_admin;

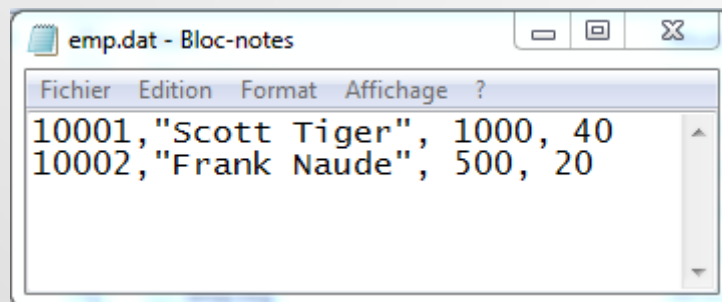
SA_CODE	SA_LIB
SA00001	TITULAIRE
SA00005	OCCASIONNEL
SA00065	CONTRACTUEL

SAFN.ctl - Bloc-notes

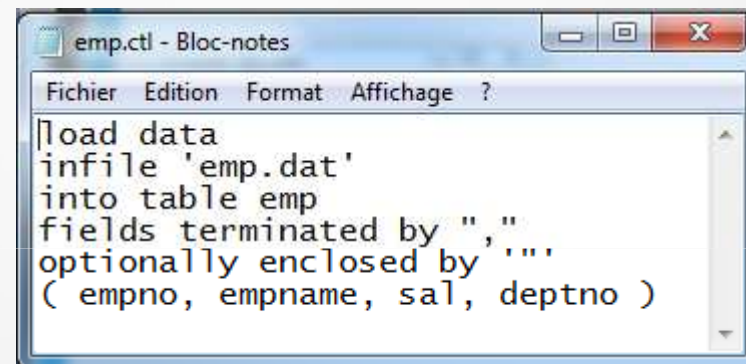
Fichier Edition Format Affichage ?

```
LOAD DATA INFILE 'SAFN.txt'
badfile 'SAFN.bad'
discardfile 'SAFN.dsc'
APPEND
INTO TABLE situation_admin
WHEN (1:2)="SA"
(sa_code position(1:7) char "trim(:sa_code)",
 sa_lib position(10:30) char "trim(:sa_lib)"
)
INTO TABLE fonction
WHEN (1:2)="FN"
(fn_code position(1:7) char "trim(:fn_code)",
 fn_lib position(10:30) char "trim(:fn_lib)"
)
```

Exemple 4



```
emp.dat - Bloc-notes
Fichier Edition Format Affichage ?
10001,"Scott Tiger", 1000, 40
10002,"Frank Naude", 500, 20
```



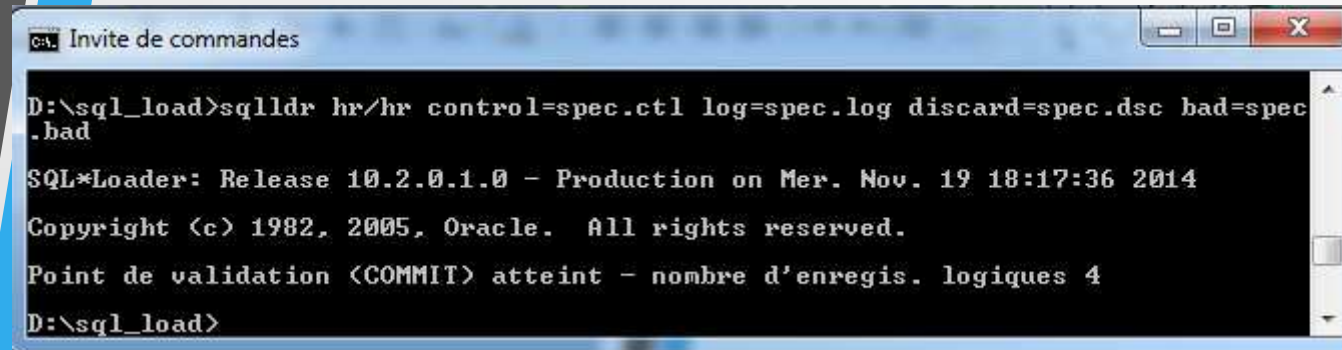
```
emp.ctl - Bloc-notes
Fichier Edition Format Affichage ?
load data
infile 'emp.dat'
into table emp
fields terminated by "\",\"
optionally enclosed by '\"'
( empno, empname, sal, deptno )
```



```
Invite de commandes - sqlplus hr/hr
SQL>
SQL> select * from emp;

  EMPNO EMPNAME      SAL  DEPTNO
-----
 10001 Scott Tiger    1000     40
 10002 Frank Naude     500     20
```

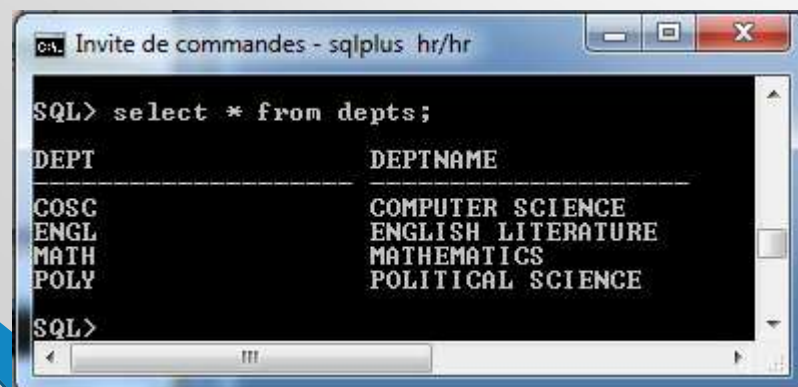
Exemple 5



```

D:\sql_load>sqlldr hr/hr control=spec.ctl log=spec.log discard=spec.dsc bad=spec
.bad
SQL*Loader: Release 10.2.0.1.0 - Production on Mer. Nov. 19 18:17:36 2014
Copyright (c) 1982, 2005, Oracle. All rights reserved.
Point de validation (COMMIT) atteint - nombre d'enregis. logiques 4
D:\sql_load>

```



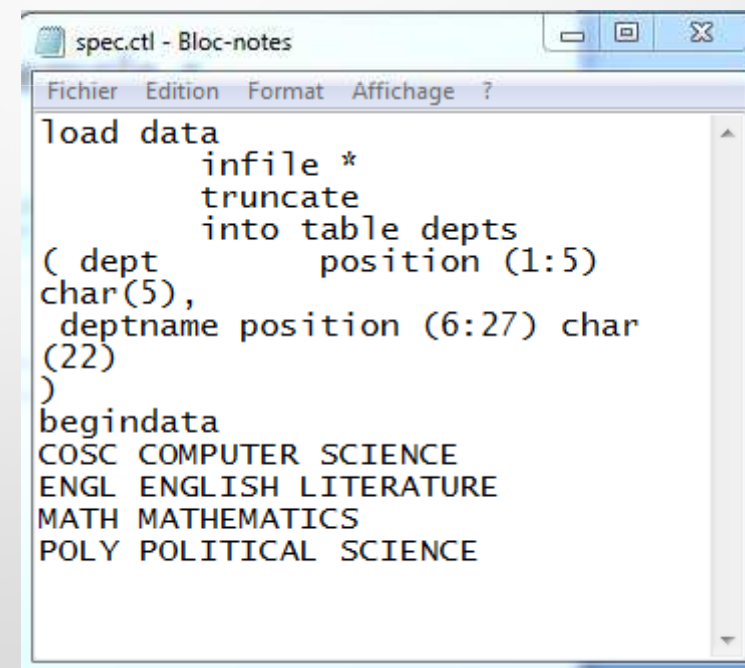
```

SQL> select * from depts;

DEPT          DEPTNAME
-----
COSC          COMPUTER SCIENCE
ENGL          ENGLISH LITERATURE
MATH          MATHEMATICS
POLY          POLITICAL SCIENCE

SQL>

```

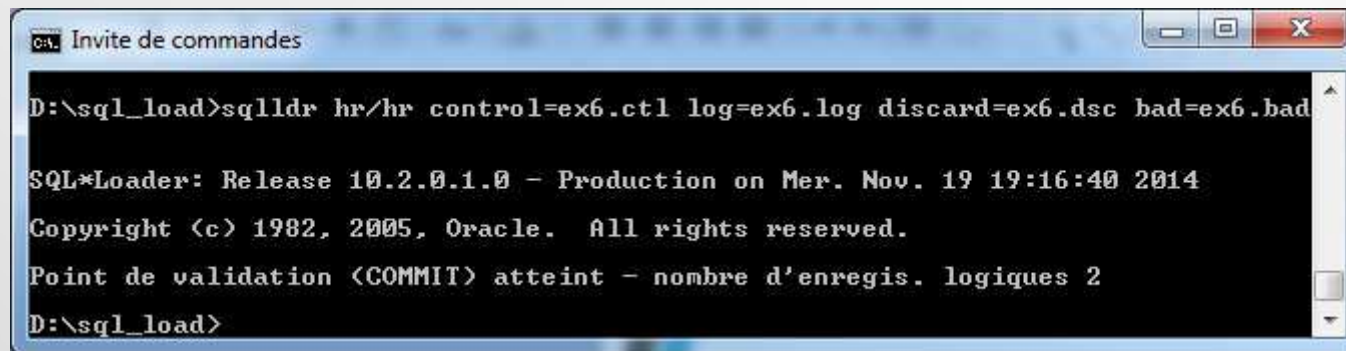


```

load data
  infile *
  truncate
  into table depts
  ( dept          position (1:5)
    char(5),
    deptname position (6:27) char
    (22)
  )
begindata
COSC COMPUTER SCIENCE
ENGL ENGLISH LITERATURE
MATH MATHEMATICS
POLY POLITICAL SCIENCE

```

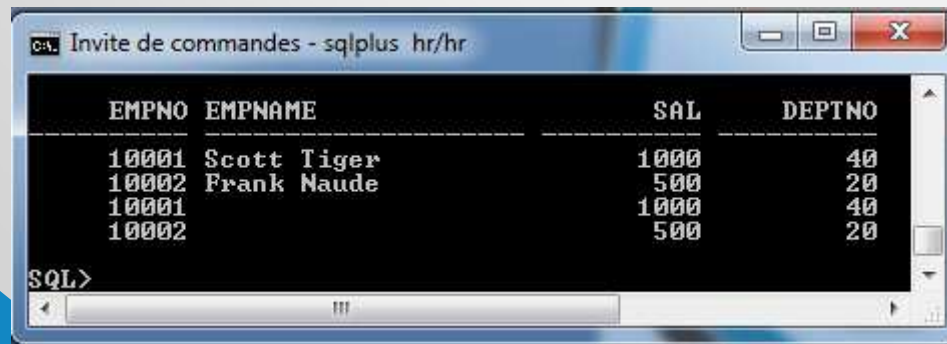
Exemple 6 : Ignorer des colonnes



```
CA\ Invite de commandes

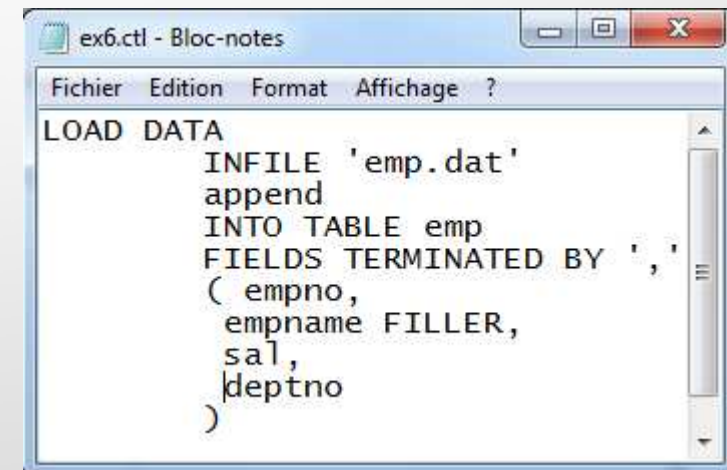
D:\sql_load>sqlldr hr/hr control=ex6.ctl log=ex6.log discard=ex6.dsc bad=ex6.bad

SQL*Loader: Release 10.2.0.1.0 - Production on Mer. Nov. 19 19:16:40 2014
Copyright (c) 1982, 2005, Oracle. All rights reserved.
Point de validation (COMMIT) atteint - nombre d'enregis. logiques 2
D:\sql_load>
```



EMPNO	EMPNAME	SAL	DEPTNO
10001	Scott Tiger	1000	40
10002	Frank Naude	500	20
10001		1000	40
10002		500	20

SQL>



```
ex6.ctl - Bloc-notes
Fichier  Edition  Format  Affichage  ?

LOAD DATA
  INFILE 'emp.dat'
  append
  INTO TABLE emp
  FIELDS TERMINATED BY ','
  ( empno,
    empname FILLER,
    sal,
    deptno
  )
```


Import/Export

- Oracle **Data Pump** est une fonctionnalité qui déplace des données et des méta-données en masse de manière très rapide entre des bases de données Oracle.
- Oracle **Data Pump** fournit deux nouveaux utilitaires d'export et import (respectivement **expdp** et **impdp**) très rapides.

Import/Export

- **Export** : permet de sauvegarder le contenu logique d'une base dans un fichier dump
- Le fichier dump sera relu (**Import**) pour récupérer les objets qu'il contient (indépendant de la plateforme)
- **L'import** et **l'export** influent sur le trafic réseau d'une manière importante
- La version de l'utilitaire **Import** ne peut pas être antérieure à celle **d'export**, on ne pourra pas donc exporter une base de données 10g pour l'importer dans une base 9i

Modes d'export / Import

- **Base Complète (FULL)**
 - Tous les objets de la base sont exportés.
 - Lors de l'import tous les objets sont importés et créés dans la base de destination
- **Niveau Utilisateur**
 - Les objets appartenant à un utilisateur donné sont exportés (tables, procédures, synonymes ...)
- **Niveau table**
 - Lors de l'exportation d'une table, tous les objets qui lui sont associés sont exportés
- **Niveau TABLESPACE**
 - Permet d'exporter les tables d'au moins un espace disque logique.
 - Seules les tables seront exportées, et non les espaces disque logiques eux-mêmes

Import – Export / Privilèges

- **Exporter d'autres schémas**

SYSDBA, EXP_FULL_DATABASE, DBA

- **Exporter la base entière**

EXP_FULL_DATABASE

- **Importer**

IMP_FULL_DATABASE

Répertoire import/export

- Répertoire défini par défaut
 - `select directory_name, directory_path`
`from all_directories;`
- Création d'un répertoire pour stocker les fichiers d'export et les fichiers log : DIRECTORY oracle
 - `create or replace directory DATAPUMP as 'D:\impexp' ;`
 - `grant read, write on directory DATAPUMP` `to hr;`

Import/Export base

- **Export de toute la base**

```
expdp directory=DATAPUMP dumpfile=database.dmp  
logfile=exp_database.log full=y
```

- **Import d'une base de données**

```
impdp directory=DATAPUMP dumpfile=database.dmp  
logfile=imp_database.log full=y
```

Import/Export schéma

- **Export du schéma hr**

```
expdp hr/hr directory=DATAPUMP dumpfile=exp.dmp logfile=exp.log
```

- **Import du schéma hr dans TD**

```
impdp system/oracle directory=DATAPUMP dumpfile=exp.dmp  
logfile=imp.log remap_schema=hr:td
```

Import/Export tablespace

- **EXPORT**

```
expdp td/td directory=DATAPUMP dumpfile=exp_tablespace.dmp  
logfile=exp_tabspace.log tablespaces=tb01
```

- **IMPORT**

```
impdp system/oracle directory=DATAPUMP dumpfile=exp_tablespace.dmp  
logfile=imp_tabspace.log tablespaces=tb01
```

- Remarque: avant l'import, le tablespace doit être créé avec 'create tablespace...'

Import/Export table

- Exporter la table **EMPLOYEES** du schéma **HR**

```
expdp hr/hr directory=DATAPUMP dumpfile=exp_tab.dmp logfile=exp_tab.log  
tables=hr.employees
```

- Il est possible d'importer uniquement quelques lignes d'une table en ajoutant :

```
query = test_q:"where id > 3"
```

- Import de la table **EMPLOYEES** de **HR** dans **TD**

```
impdp system/oracle directory=DATAPUMP dumpfile=exp_tab.dmp  
logfile=imp_tab.log tables=hr.TEST_IMP content=METADATA_ONLY  
remap_schema=hr:td
```

Suivi d'Import/Export

- Nous pouvons vérifier le déroulement du job au cours de l'exécution de l'import ou de l'export, en interrogeant la vue **DBA_DATAPUMP_JOBS**
- `SELECT owner_name, job_name, operation, job_mode, state FROM DBA_DATAPUMP_JOBS ;`

Package DBMS_DATAPUMP

- Export de la table test

```
SET SERVEROUTPUT ON
DECLARE
a NUMBER;
BEGIN
a := DBMS_DATAPUMP.open(operation=>'EXPORT',job_mode=>'TABLE',job_name=>'testEXPORT');
DBMS_DATAPUMP.add_file(handle=>a,filename=>'test.dmp',directory=>'DATAPUMP',filetype=>1);
DBMS_DATAPUMP.add_file(handle=>a,filename=>'test.log',directory=>'DATAPUMP',filetype=>3);
DBMS_DATAPUMP.metadata_filter(handle => a,name=>'SCHEMA_EXPR',value =>'IN ("TEST")');
DBMS_DATAPUMP.start_job(a); --executer job
DBMS_DATAPUMP.detach(a); --stopper session user apres execution job
dbms_output.put_line('Export terminé ');
EXCEPTION
WHEN OTHERS THEN dbms_output.put_line('erreur:'||sqlerrm);
END;
/
```

Package DBMS_DATAPUMP

- Import de la table test

```
SET SERVEROUTPUT ON SIZE 1000000
DECLARE
a NUMBER;
BEGIN
a:=DBMS_DATAPUMP.open(operation=>'IMPORT',job_mode=>'TABLE',job_name=>'testIMPORT');
DBMS_DATAPUMP.add_file(handle=>a,filename=>'test.dmp',directory
=>'DATAPUMP',filetype=>1);
DBMS_DATAPUMP.add_file(handle=>a,filename=>'test_imp.log',directory =>
'DATAPUMP',filetype=>3);
DBMS_DATAPUMP.set_parameter(handle=>
a,name=>'TABLE_EXISTS_ACTION',value=>'REPLACE');
DBMS_DATAPUMP.start_job(a);
DBMS_DATAPUMP.detach(a);
dbms_output.put_line('import terminé ');
EXCEPTION
    WHEN OTHERS THEN dbms_output.put_line('erreur: '||sqlerrm||' Job-
ID: '||a);
END;
/
```