18

Les outils d'Oracle 10*g*

Dans ce chapitre :

- présentation des outils d'Oracle 10*g* ;
- Oracle Migration Workbench, pour migrer des bases vers Oracle ;
- Oracle Export/Import ;
- Oracle Data Pump ;
- Oracle SQL*Loader ;
- Oracle SQL*Plus ;
- utilitaires complémentaires aux outils Oracle.

Depuis des années, chaque version d'Oracle est livrée avec un ensemble d'outils standard. Ils sont puissants et performants et on les retrouve sur toutes les machines où Oracle 10*g* existe. Livrés gratuitement, ils rendent l'utilisation de produits complémentaires facultative. Avant de les étudier séparément, nous vous les présentons rapidement.

Présentation des outils d'Oracle 10*g*

Oracle Migration Workbench n'est pas installé avec Oracle. C'est un « kit de migration » permettant de migrer toutes sortes de bases de données vers Oracle. Vous pouvez ainsi migrer des bases Access, Informix, DB2, Sybase, SQL Server, etc. vers Oracle. Vous pouvez télécharger gratuitement le kit concernant le fournisseur et la version exacte de la base à migrer vers Oracle à partir du site *http://technet.oracle.com*. Nous ne développerons pas plus cet utilitaire dans ce chapitre, mais il était important d'en mentionner l'existence.

Export/Import sont deux utilitaires qui fonctionnent l'un avec l'autre. Le premier permet d'exporter une partie ou la totalité du contenu d'une base Oracle 10*g*, le second importe la partie exportée dans une base de données Oracle 10*g*. Export/Import sont principalement utilisés lors de sauvegardes, de réorganisations et de déplacements du contenu d'une base Oracle 10*g* de machine à machine. La portabilité des fichiers exportés permet à un Export réalisé sur une base Oracle 10*g* fonctionnant sous Windows, d'être importé dans une base Oracle 10*g* sous Linux et inversement.

Un nouvel utilitaire d'Oracle 10*g*, Oracle Data Pump, reprend les mécanismes d'import/export. Il offre de meilleures performances qu'import/export et procure plus de souplesse, notamment la possibilité d'interrompre et de reprendre un Data Pump Export ou un Data Pump Import. Oracle Data Pump ne permet des échanges qu'entre des bases 10*g*, alors que l'import/export « classique » permet aussi de transférer des données entre des versions différentes d'Oracle.

SQL*Loader est un utilitaire très puissant de chargement des bases Oracle, à partir de fichiers contenant des données. Le format des données contenues dans les fichiers et leur destination dans la base cible sont très facilement paramétrables. SQL*Loader est fréquemment utilisé lors de l'alimentation de bases d'infocentre pour charger, dans Oracle, des données extraites d'autres bases.

SQL*Plus est l'interface mode caractère d'accès à Oracle 10*g* ; certaines versions, comme iSQL*Plus, peuvent se présenter sous forme semi-graphique via un accès Web. Cet outil est indispensable à tout développeur ou tout administrateur Oracle 10*g*, car c'est l'interface SQL privilégiée pour accéder aux bases Oracle 10*g*.

SQL*Plus permet maintenant de créer, démarrer, arrêter, administrer et effectuer des opérations complexes de restauration sur les bases Oracle. Il remplace l'ancien Server Manager qui assurait ces fonctions. SQL*Plus devient donc l'interface privilégiée d'Oracle par laquelle la totalité des ordres SQL concernant l'administration et l'utilisation d'Oracle peuvent être exécutés. Oracle Enterprise Manager est un outil graphique qui reprend une grande partie des possibilités d'administration offertes par SQL*Plus. Pour plus de détails à son sujet, reportez-vous au chapitre 25, *Oracle Enterprise Manager.*

Outre les outils standard, d'autres sont dédiés au développement et à l'administration. Ils sont proposés par Oracle ou par d'autres sociétés. Nous vous en présentons quelques-uns en distinguant les principales familles.

Oracle Export / Oracle Import

Ce paragraphe traite de l'utilisation pratique de l'utilitaire d'Export/Import. C'est un outil fort apprécié et pourtant négligé lors des formations Oracle. Ses principales utilisations sont les suivantes :

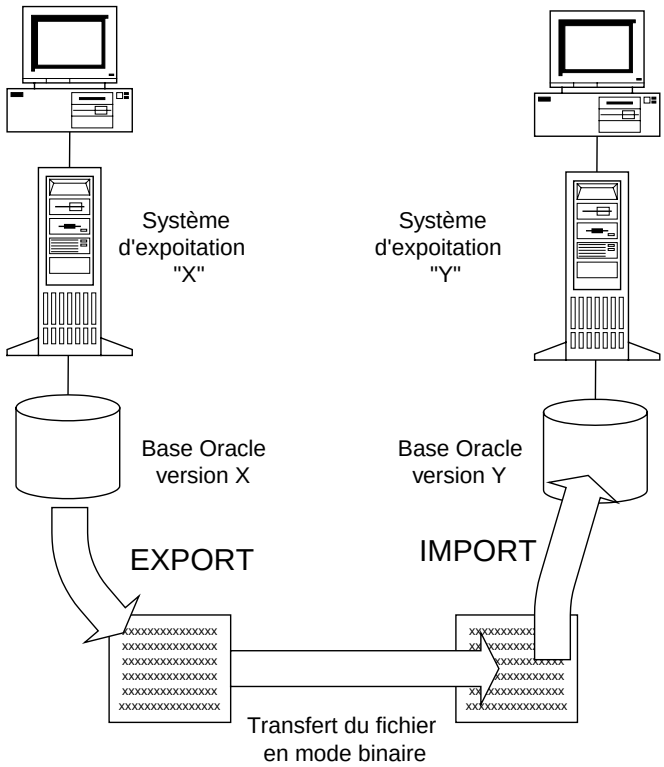
- le déplacement du contenu d'une base vers une autre (regroupement, migrations…) ;
- des sauvegardes partielles.

Principes de l'Export/Import

Le principe de l'Export/Import est d'extraire des données d'une base Oracle dans un format indépendant de tout système d'exploitation et de les réintégrer ailleurs, pour ce qui concerne l'Import. Cet objectif se réalise en deux étapes :

- Extrayez tous les objets (tablespaces, tables, index, procédures…) de la base de données. Ce sont les vues interrogeant le dictionnaire de la base Oracle qui réalisent cette fonction. Les objets sont extraits dans un ordre particulier.
- Extrayez les données contenues dans les tables. Ce sont des ordres SELECT qui interrogent les tables et enregistrent les données dans un format d'Export particulier.

Figure 18-1
Principe de l'Export/Import



L'opération d'Export est graduelle. Elle permet de transférer, par étapes, le contenu de la base :

- full : exportation complète de la base ;
- user : exportation de l'ensemble des objets propriétés d'un utilisateur ;
- table : exportation d'une table particulière.

À l'inverse, l'Import insère le contenu d'un Export dans une base de données. De même que l'Export, il fonctionne suivant plusieurs modes :

- full : importation complète de tous les objets présents dans le fichier export en entrée ;
- user : importation de l'ensemble des objets propriétés d'un utilisateur ;
- table : importation d'une table particulière.

L'Export/Import est-il un outil de sauvegarde ?

Force est de reconnaître que l'on ne peut pas considérer l'Export/Import comme un moyen de sauvegarde fiable à 100%. En effet, dans certains cas, vous pouvez effectuer une exportation complète sans exporter la totalité du contenu de la base. Par exemple, si l'un des tablespaces est inactif (OFFLINE), aucun des objets de ce tablespace ne sera exporté. L'exportation sera partielle au vu du contenu global de la base, ce qui peut être grave.

L'Export/Import est complémentaire des sauvegardes classiques, c'est-à-dire de la totalité des fichiers physiques qui constituent la base de données. Ce type de sauvegarde très puissant peut être étendu et inclure les journaux d'activité (archivage des redo-log). L'Export/Import est intéressant si, par exemple, une table et ses données sont supprimées par erreur. Il est alors possible d'importer cette table (dont les données sont à J-1 si l'exportation a eu lieu la veille), sans perturber la production ni arrêter la base.

Avant de choisir une stratégie de sauvegarde, reportez-vous au chapitre 26, *La sauvegarde d'une base Oracle 10*g.

Où se situe le jeu d'essai de l'Export/Import ?

À la différence de SQL*Loader, PL/SQL, Pro*C, il n'y a pas de fichier d'exemple livré par Oracle concernant l'Export/Import.

Configuration préalable

Pour rechercher dans le dictionnaire de données Oracle l'ensemble des objets existants dans la base, l'Export s'appuie sur des vues. Elles sont créées par le script `catexp.sql`, sous le compte utilisateur SYS (propriétaire du dictionnaire de la base). Il est exécuté durant la création de la base de données, à la fin de l'exécution du script catalog.sql, qui crée le catalogue de la base. Toutefois, le script catexp.sql peut être lancé manuellement, quand vous le souhaitez. Malgré son nom, (*cat*alog *exp*ort) il est aussi utilisé par le logiciel d'Import.

Si le script de création des vues n'est pas exécuté, l'Export/Import cherchera à accéder à des vues inexistantes et générera une erreur.

Les programmes d'Export/Import sont indépendants de la base de données. Ils peuvent être lancés en client-serveur pour accéder à des bases locales ou distantes. Sous Windows, les exécutables se nomment exp (Export) et imp (Import). Ils sont situés dans ORACLE_HOME\bin soit généralement C:\oracle\product\10.1.0\db_1\BIN.

Compatibilité des Export/Import avec les précédentes versions d'Oracle

Puisque les exécutables d'Export/Import sont des outils clients, ils ne se situent pas forcément sur la même machine et au même niveau de version que la base de données qu'ils interrogent. C'est pourquoi certaines versions d'Export/Import incorporent dans leur nom le numéro de la base de données à laquelle ils peuvent accéder (imp, exp, imp7, exp7…).

Une version prévue pour exporter des données d'une base Oracle8*i* permet, par rapport à la version Oracle7, d'exporter les nouvelles possibilités de cette base (extensions objet, partitionnement, etc.). Réaliser l'exportation complète d'une base Oracle8*i* avec la version Export d'Oracle7 conduit à une exportation incomplète, ce qui est grave.

En conséquence, il sera toujours plus facile de réaliser une exportation sur une base Oracle7 et d'en importer le contenu dans une base Oracle 10*g*. Dans ce cas, la compatibilité ascendante est assurée. Dans le cas d'une exportation Oracle 10*g* importée dans une base Oracle7, des difficultés peuvent survenir si vous avez utilisé dans la nouvelle version des possibilités inexistantes dans l'ancienne. Ces difficultés se traduisent par des erreurs lors des opérations.

> Pour éviter tout inconvénient, la règle de base est d'utiliser la même version d'Export/Import que celle livrée avec la base de données. Réservez exclusivement ces opérations aux serveurs qui hébergent vos bases.

Si vous devez exporter vos données d'une base Oracle7, 8, 9*i* pour les importer dans une base Oracle 10*g*, utilisez la version correspondante pour l'exportation et la version 10*g* pour l'importation. Ce cas est fréquemment rencontré lors de l'évolution des versions Oracle.

Que contient un fichier export ?

Un fichier export est d'une taille beaucoup plus réduite que celle de la base de données exportée. Il ne contient que les ordres SQL de création des index, rollback segments, tablespaces, utilisateurs, droits, etc. Pour les autres objets, il contient à la fois leur définition, leur données et le code des programmes (tables, triggers, fonctions, etc.). Cette technique fait gagner beaucoup d'espace disque par rapport au volume de l'ensemble des fichiers qui composent la base de données.

Le format interne d'un fichier export n'est pas lisible facilement. Il faut toujours passer par les programmes d'Export/Import pour en manipuler le contenu. Nous vous déconseillons d'utiliser des éditeurs pour en visualiser le contenu. En effet, leur codage interne est binaire afin d'éviter les problèmes de conversion lors d'un changement de système d'exploitation.

Comment transférer des fichiers Export ?

Il faut toujours utiliser un outil (par exemple ftp) qui permet de transférer les fichiers d'Export de machine à machine sous forme binaire. Cette erreur (transfert en mode texte et non binaire) fréquente se manifeste par l'impossibilité de charger le fichier dans la base cible.

> Attention : un fichier export peut être l'un des éléments de sauvegarde de votre base de données. Transférez-le en mode binaire et soyez prudent avant de supprimer les fichiers Export d'origine : assurez-vous qu'ils n'ont pas été corrompus ou transformés par leur transfert.

Comment lancer un Export/Import ?

Vous pouvez lancer un Export/Import par la commande exp (ou imp), suivie du nom d'utilisateur et du mot de passe. Par exemple :

```
exp SCOTT/TIGER
```

Dans ce cas, l'Export propose un mode interactif et vous guide pour les opérations à effectuer. Vous pouvez aussi commander l'Export en précisant dans sa ligne de commande le paramètre piloté et sa valeur :

```
exp PARAMETRE=valeur ou PARAMETRE=(valeur1, valeur2, ..., valeurN)
```

Par exemple :

```
exp userid=scott/tiger file=14_02_2005.dmp tables=(emp,dept)
```

> Le mode interactif ne propose pas tous les paramètres possibles. Nous vous conseillons d'utiliser le passage de paramètres qui permet d'utiliser toutes les options d'Export/Import.

L'Export/Import en mode incrémental

Vous pouvez créer des exportations incrémentales. Pour cela, il faut procéder comme suit :

1. Effectuer une exportation totale.
2. Exporter les incréments.

Cela permet de synchroniser le contenu des bases de données distantes, sans avoir à importer la totalité de la base à chaque fois.

Les options d'un Export

Pour obtenir les options du programme Export, appelez la commande suivante :

```
exp HELP=Yes
```

> Il est regrettable que chaque outil Oracle ait sa propre syntaxe pour appeler l'aide.

```
Export: Release 9.2.0.1.0 - Production on Ve Aou 23 08:31:41 2002
Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.

Vous pouvez laisser l'export vous demander les param. en tapant la
commande EXP suivie de votre nom utilisateur/mot de passe :
```

```
  Exemple : EXP SCOTT/TIGER

Pour contrôler l'exécution de l'export, entrer la commande EXP suivie de divers arguments.
➥Pour indiquer les paramètres, utiliser des mots-clés :

 Format :  EXP KEYWORD=valeur
           ou KEYWORD=(valeur1,valeur2,...,valeurN)
 Exemple : EXP SCOTT/TIGER GRANTS=Y TABLES=(EMP,DEPT,MGR)
           ou TABLES=(T1:P1,T1:P2), si T1 est une table partitionnée

USERID doit être le premier paramètre de la ligne de commande.

Mot-clé    Description  (Valeur par défaut)
------------------------------------------------------------
USERID     nom utilisateur/mot de passe
FULL       export du fichier entier (N)
BUFFER     taille du tampon de données
OWNER      listes des noms utilisateur propriétaire
FILE       fichiers de sortie (EXPDAT.DMP)
TABLES     liste des noms de table
COMPRESS   import dans un extent (Y)
RECORDLENGTH longueur de l'enregistrement d'E/S
GRANTS     export des autorisations d'accès (Y)
INCTYPE    type d'export incrémentiel
INDEXES    export des index (Y)
RECORD     suivi de l'export incr. (Y)
DIRECT     chemin direct (N)
TRIGGERS   export des déclencheurs (Y)
LOG        fichier journal de sortie d'écran
STATISTICS analyse des objets (ESTIMATE)
ROWS       export des lignes de données (Y)
PARFILE    nom du fichier de paramètres
CONSISTENT cohérence entre les tables(N)
CONSTRAINTS  export des contraintes (Y)

OBJECT_CONSISTENT  transaction définie en lecture seule pendant l'export de l'objet (N)
FEEDBACK           afficher la progression toutes les x lignes (0)
FILESIZE           taille maximale de chaque fichier de vidage
FLASHBACK_SCN      SCN utilisé pour rétablir le cliché de session
FLASHBACK_TIME     temps d'obtention du SCN le plus près du temps indiqué
QUERY              clause SELECT utilisée pour exporter un sous-ensemble d'une table
RESUMABLE          suspension lorsqu'une erreur associée à l'espace se produit(N)
RESUMABLE_NAME     chaîne de texte utilisée pour identifier l'instruction RESUMABLE
RESUMABLE_TIMEOUT temps d'attente pour RESUMABLE
TTS_FULL_CHECK     exécuter un contrôle de dépendance complet ou partiel pour TTS
TABLESPACES        liste des tablespaces à exporter
TRANSPORT_TABLESPACE - export des métadonnées des tablespaces transportables (N)
TEMPLATE           nom de modèle appelant l'export du mode iAS

Procédure d'export terminée avec succès.
```

Les options les plus importantes sont commentées dans ce paragraphe.

Paramètre	Usage	Valeur par défaut
BUFFER	Taille de tampon de données.	Dépend de l'OS
COMPRESS	Exporte la définition d'un objet (table, index…) en une seule extension contiguë.	Yes
CONSISTENT	Cohérence entre les tables exportées.	Sans
CONSTRAINTS	Exporte la définition des contraintes d'intégrité.	Yes
DIRECT	Chemin d'accès direct.	No
FEEDBACK	Affiche l'avancement de l'exportation toutes les n lignes.	0
FILE	Nom du fichier de sortie. Si le fichier est trop volumineux, vous pouvez en créer plusieurs d'une taille inférieure grâce au paramètre FILESIZE.	EXPDAT.DMP
FILESIZE	Pour limiter la taille du fichier exporté, vous pouvez en créer plusieurs de taille réduite. FILESIZE indique la taille maximale au-delà de laquelle un nouveau fichier d'export sera créé.	Sans
FULL	Exportation complète.	Yes
GRANTS	Exporte les autorisations d'accès.	Yes
INCTYPE	Type d'exportation incrémentale.	Sans
INDEXES	Exporte la définition des index.	Yes
LOG	Fichier de journalisation des sorties écran. Utile pour conserver une trace de l'exécution d'un export.	Sans
PARFILE	Fichier de commande contenant les paramètres d'exportation.	Sans
OWNER	Liste des utilisateurs à exporter.	Sans
QUERY	Il est possible d'exporter une partie des enregistrements d'une table en filtrant les lignes désirées par la commande QUERY.	Sans
RECORD	Suivi d'exportation incrémentale.	Sans
RECORDLENGTH	Longueur d'enregistrement.	Dépend de l'OS
RECOVERY_TABLESPACES	Liste des tablespaces à récupérer.	Sans
ROWS	Exporte les lignes de données.	Yes
STATISTICS	Analyse (ANALYZE) des objets exportés.	Sans
TABLES	Liste des tables à exporter.	Sans
TABLESPACE	Tous les objets hébergés dans ce tablespace seront exportés.	Sans
USERID	Nom utilisateur / mot de passe.	Sans

Les possibilités de l'Export en mode USER

Un utilisateur peut exporter plus ou moins d'éléments selon les droits qu'il possède :

- un utilisateur exporte ses propres objets :

```
exp userid=scott/tiger file=scott_export.dmp
```

- un administrateur DBA peut exporter plusieurs utilisateurs :

```
exp userid=system/manager owner=(scott,toto) file=export.dmp
```

Dans les deux cas, les exportations sont considérées de type utilisateur (USER), mais les utilisateurs possèdent des droits différents qui permettent d'exporter les objets de plusieurs d'entre eux ou d'un seul.

Les possibilités de l'Export en mode TABLE

D'une façon similaire à l'Export en mode USER, tout dépend des droits de l'utilisateur qui réalise l'exportation. Certains n'accéderont qu'à leurs propres objets :

```
exp userid=scott/tiger tables=(scott.emp, scott.dept, scott.customer)
```

D'autres, de droits plus étendus, exportent des tables propriétés de plusieurs utilisateurs :

```
exp userid=system/manager tables=(scott.emp, toto.travail)
```

Dans les deux cas, les exportations sont considérées comme des exportations en mode TABLE.

Qui peut réaliser une exportation complète ?

Une exportation complète (FULL) est réalisable par tout utilisateur possédant le rôle EXP_FULL_DATABASE. Par défaut, c'est le cas de tous les utilisateurs DBA. On peut aussi attribuer ce rôle à d'autres utilisateurs pour réaliser des exportations complètes. Toutefois, par prudence, nous vous conseillons de réserver les Exports FULL aux seuls administrateurs DBA de la base de données.

```
exp userid=system/manager full=Yes
```

> Ne vous laissez pas pirater le contenu de votre base en autorisant l'Export à un trop grand nombre de personnes.

Dans quel ordre sont exportés les objets ?

Voici l'ordre dans lequel les objets sont exportés. Suivant les versions Oracle, il peut évoluer :

- tablespaces ;
- profiles ;
- utilisateurs ;

- rôles ;
- System Privilege Grants ;
- autorisations sur les rôles ;
- rôles par défaut ;
- quotas sur les tablespaces ;
- rollback segments;
- database links;
- séquences (avec leurs droits) ;
- snapshots, logs, queues, refresh ;
- cluster ;
- tables (avec droits, autorisations, commentaires, index, contraintes, audit) ;
- intégrité référentielle ;
- synonymes ;
- vues ;
- procédures stockées ;
- triggers.

L'ordre d'Export est très important. Souvent, les objets dépendent les uns des autres. Par exemple, une table est contenue dans un tablespace et elle possède des contraintes liées à d'autres tables situées dans d'autres tablespaces. Si vous importez une table avant son tablespace, il en résultera une erreur. C'est pourquoi l'Import obéit lui aussi à cet ordre très précis.

Qu'est-ce que l'ARRAY FETCH ?

Pour extraire les données des tables, l'Export utilise des ordres SQL SELECT. Les données doivent être transférées de la base au programme Export, qui n'est pas forcément situé sur la même machine. Si l'Export extrait les données ligne à ligne, cela génère une forte consommation réseau. Pour éviter cela, le mécanisme d'ARRAY FETCH extrait les données par paquet. Pour l'Export et l'Import, la taille des paquets dépend des paramètres BUFFER et RECORDLENGTH.

Utilisation des paramètres BUFFER et RECORDLENGTH

Lors d'un Export, les données sont extraites de la base Oracle 10*g*, transmises à Export qui les conserve en mémoire avant de les enregistrer dans le fichier d'Export. La variable BUFFER détermine la taille de cette zone mémoire tampon. On retrouve le même paramètre BUFFER lors d'une importation.

La variable RECORDLENGTH détermine la taille des paquets de données extraits de la zone mémoire fixée par BUFFER, pour les écrire dans le fichier disque.

Dans un premier temps, nous vous conseillons de laisser les valeurs par défaut.

Comment sont exportés les champs de type LONG ?

Les champs de type LONG peuvent atteindre 2 Go par enregistrement. L'Export/Import traite les champs de type LONG au même titre que les autres champs.

Comment utiliser un fichier de commande ?

Pour éviter une ligne de commande trop complexe ou écrire des scripts réutilisables, nous vous conseillons d'utiliser un fichier de commande. Par exemple :

```
exp PARFILE=fichier_de_commande.txt
```

Le fichier de commande contient les paramètres sous la forme PARAMETRE=valeur. Par exemple :

```
USERID=scott/tiger
FULL=yes
FILE=mon_fichier_export.dmp
GRANTS=yes
INDEXES=yes
```

> En environnement Windows, si le nom du fichier cible comporte un ou plusieurs espaces, vous devez l'encadrer avec trois parenthèses : FILE= """C:\temp\mon fichier avec des espaces.dmp"""

Comment utiliser l'option QUERY ?

Pour réaliser un environnement de test, il est souvent nécessaire d'extraire juste une partie d'une table. La méthode « classique » consiste à réaliser une table temporaire, insérer les données provenant de la table complète à l'aide d'un ordre INSERT…AS SELECT puis d'exporter / importer cette table temporaire.

Oracle Export propose une option très intéressante qui évite la création de cette table temporaire. Vous pouvez n'extraire qu'une partie d'une table. Par exemple :

```
exp scott/tiger TABLE=emp QUERY="WHERE job='CLERK' and sal>1000"
```

Dans ce cas, les données extraites depuis la table EMP correspondront au résultat de l'ordre SQL :

```
SELECT * from emp WHERE job='CLERK' and sal>1000
```

> En environnement Windows, pour que le caractère « " » soit correctement interprété, il est nécessaire de le faire précéder par un « \ ». Par exemple :
> exp scott/tiger TABLE=emp QUERY=\"WHERE job='CLERK' and sal>1000\"

Comment limiter la taille d'un fichier export ?

Prenons un exemple : vous devez envoyer rapidement un fichier export à un utilisateur distant. Le fichier est très volumineux, ce qui exclu un transfert par le réseau. Vous disposez d'un graveur de CD-Rom et d'un lecteur de cartouches. Dans le cas d'un CD-Rom ou d'un cartouche de sauvegarde, leur taille respective donne la taille maximale du fichier d'export qu'il serait simple de manipuler.

```
exp user/mot_de_passe FILE=exp1.dmp,exp2.dmp,exp3.dmp FILESIZE=600MB
```

Trois fichiers d'export seront crées d'une taille maximale de 600 Mo. Si la taille du fichier export est supérieure, un nouveau nom de fichier vous sera demandé. Vous pouvez graver les fichiers sur trois CD-Rom et les envoyer. Cette technique est la plus simple.

Une autre technique consiste à compresser le fichier export, le copier puis le décompresser à l'arrivée. Cela impose que l'expéditeur et le destinataire soient équipés d'outils identiques.

Comment récupérer les erreurs rencontrées lors d'un export ?

Vous pouvez rediriger les sorties écran pour conserver dans des fichiers la trace de l'exécution d'un export :

```
exp user/mot_de_passe PARFILE=fichier.txt >fichier1.log 2>fichier2.err
```

Dans cet exemple, les messages d'erreur sont séparés des messages d'exécution de l'export. Le fichier fichier1.log contient le résultat des messages écran et le fichier fichier2.err contient les erreurs.

Les options d'un Import

Pour obtenir les options du programme Import, appelez la commande suivante :

```
imp HELP=Yes

Import: Release 9.2.0.1.0 - Production on Ve Aou 23 08:32:24 2002
Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.

Pour que l'import vous demande les paramètres., entrez la commande
IMP suivie de votre nom utilisateur/mot de passe:

 Exemple : IMP SCOTT/TIGER

Pour contrôler l'exécution de l'import, entrez la commande IMP suivie
de divers arguments. Pour indiquer paramètre, utiliser des mots-clés :

 Format :  IMP KEYWORD=valeur ou KEYWORD=(valeur1,valeur2,...,valeurN)
 Exemple : IMP SCOTT/TIGER IGNORE=Y TABLES=(EMP,DEPT) FULL=N
        ou TABLES=(T1:P1,T1:P2), si T1 est une table partitionnée
```

```
USERID doit être le premier paramètre de ligne de commande.

Mot-clé  Description (Valeur par défaut)
------------------------------------------------
USERID   nom utilisateur/mot de passe
FULL     import du fichier entier (N)
BUFFER   taille du tampon de données
FROMUSER liste noms utilisateur propriétaire
TOUSER   liste des noms utilisateur
FILE     fichier d'entrée (EXPDAT.DMP)
SHOW     afficher seulement contenu du fichier (N)
TABLES   liste des noms de table
IGNORE   ignorer erreurs de création (N)
RECORDLENGTH taille d'enreg. ES
GRANTS   import d'autorisations d'accès (Y)
INCTYPE  type d'import incrémentiel
INDEXES  import d'index (Y)
COMMIT   valider l'insertion de tableau (N)
ROWS     import de lignes de données (Y)
PARFILE  nom fichier de paramètres
LOG      fichier journal de la sortie écran
CONSTRAINTS  import de contraintes (Y)
DESTROY      écraser le fichier de données de tablespace (N)
INDEXFILE    écriture d'infos de table/index dans le fichier indiqué
SKIP_UNUSABLE_INDEXES  sauter la maintenance des index inutilisables (N)
FEEDBACK        afficher la progression toutes les x lignes (0)
TOID_NOVALIDATE sauter la validation des ID de type indiqués
FILESIZE        taille maximale de chaque fichier de vidage
STATISTICS      importer les statistiques pré-calculées (toujours)
RESUMABLE       suspension lorsqu'une erreur associée à l'espace se produit(N)
RESUMABLE_NAME  chaîne de texte utilisée pour identifier l'instruction RESUMABLE
RESUMABLE_TIMEOUT temps d'attente pour RESUMABLE
COMPILE         compiler procédures, packages et fonctions (Y)
STREAMS_CONFIGURATION  import des métadonnées générales des flux de données (Y)
STREAMS_INSTANITATION  import des métadonnées d'instanciation des flux de données (N)

Les mots-clés suivants ne s'appliquent qu'aux tablespaces transportables
TRANSPORT_TABLESPACE - import des métadonnées des tablespaces transportables (N)
TABLESPACES - tablespaces à transporter dans la base de données
DATAFILES - fichiers de données à transporter dans la base de données
TTS_OWNERS - utilisateurs auxquels appartiennent les données contenues dans l'ensemble
➥de tablespaces transportables

Procédure d'import terminée avec succès.
```

Paramètre	Usage	Valeur par défaut
BUFFER	Taille de tampon de données.	Dépend de l'OS
CHARSET	Jeu de caractères du fichier sous lequel a été réalisée l'exportation.	Sans
COMMIT	Valide l'insertion des données.	Yes
COMPILE	Compile tous les objets lors de l'import.	Yes
CONSTRAINTS	Importe les contraintes liées aux tables.	Yes
DESTROY	Remplace le tablespace s'il existe déjà (attention !).	No
FEEDBACK	Affiche l'avancement de l'importation toutes les n lignes.	0
FILE	Nom du fichier d'entrée. Vous pouvez indiquer plusieurs fichiers résultat d'un seul export.	EXPDAT.DMP
FROMUSER/ TOUSER	Permet de changer le propriétaire des données lors de l'importation.	Sans
FULL	Importation complète du fichier d'Export.	Yes
GRANTS	Importation des autorisations d'accès.	Yes
HELP	Aide en ligne.	Sans
IGNORE	Ignore les erreurs lors de la création.	No
INCTYPE	Type d'Export incrémental.	Sans
INDEXES	Importation et création (si inexistants) des index.	Yes
INDEXFILE	Lit le fichier d'entrée et en extrait la définition DDL de création des tables et des index qu'il contient.	No
LOG	Fichier de journalisation des sorties écran. Utile pour conserver une trace de l'exécution d'un import.	Sans
PARFILE	Fichier de commandes contenant les paramètres d'un Import.	Sans
RECORDLENGTH	Longueur d'enregistrement.	Dépend de l'OS
SHOW	Affiche le contenu du fichier sans importer les objets.	No
STATISTICS	Analyse (ANALYZE) les objets exportés.	Sans
TABLES	Liste des tables à importer.	Sans
USERID	Nom utilisateur/mot de passe.	Sans

Les options disponibles lors d'une importation sont plus nombreuses que celles proposées par l'exportation. Les plus importantes sont commentées dans ce paragraphe.

Quelle action réalise le paramètre DESTROY ?

Si vous importez un tablespace, il est créé par la commande CREATE TABLESPACE qui permet de réutiliser les fichiers spécifiés dans la commande s'ils existent déjà sur le disque. DESTROY = Yes peut effacer un fichier déjà existant. Nous vous déconseillons fortement d'utiliser cette option. La valeur par défaut est DESTROY = No.

Comment créer le clone d'une base ?

Si vous voulez effectuer une copie d'une base existante, il faut créer une base cible « minimale » puis procéder comme suit :

- exportez complètement la base source :

```
exp system/manager full=Y file=expfull.dmp
```

- importez ce fichier dans la base cible :

```
imp system/manager full=Y file=expfull.dmp
```

Si vous copiez la base sur la même machine ou si le système d'exploitation change, il faut au préalable créer les tablespaces manuellement sur la base cible, car l'Import cherchera à en créer les fichiers sous le même nom et aux mêmes endroits que les fichiers des tablespaces source.

Peut-on remplacer des données existantes par celles issues d'une importation ?

La réponse est non, avant de réaliser l'importation, il faut supprimer manuellement les données que vous souhaitez mettre à jour. Par défaut, l'Import ajoute les données importées à celles qui existent déjà. C'est le principe de l'importation incrémentale.

Quelles sont les conséquences d'une importation pour les objets propriétés de SYS ?

L'utilisateur SYS est le propriétaire du dictionnaire de données qui gère la vie de chaque base Oracle 10*g*. Sauf à lancer des scripts SQL fournis par Oracle, on ne doit jamais travailler ni créer d'objets sous le compte utilisateur SYS. Toute base cible possède, elle aussi, son dictionnaire interne de données, qui ne peut pas être effacé par celui issu d'une autre base. C'est pourquoi les objets de l'utilisateur SYS ne sont ni exportés, ni importés. De la même façon, les autorisations (GRANT) attribuées aux objets de SYS ne sont pas exportées.

> C'est une autre raison pour laquelle l'Export/Import est l'un des éléments d'une sauvegarde logique, différente de la sauvegarde physique réalisée par la copie des fichiers composant la base.

Les mots de passe des utilisateurs sont-ils automatiquement exportés/importés ?

Les mots de passe des utilisateurs autres que SYS et SYSTEM ne sont pas affectés par un Export/Import. En revanche, les mots de passe SYS et SYSTEM sont mis à jour par un Import en mode FULL. Pour éviter cela, il faut se connecter pour modifier les mots de passe de SYS et SYSTEM.

```
# Connexion sous le compte OS propriétaire d'Oracle
su - oracle

sqlplus /nolog
connect / as sysdba
```

Par sécurité, les mots de passe sont cryptés lorsqu'ils figurent dans un fichier d'Export.

Exporter tous les utilisateurs, est-ce identique à un Export en mode FULL ?

La réponse est non. Lorsque vous exportez le contenu d'une base, les objets sont exportés dans un ordre particulier. Lorsque l'on exporte uniquement les objets des utilisateurs, certains éléments exportés lors d'un Export FULL manquent.

Comment sont importées les contraintes d'intégrité référentielle ?

Les contraintes d'intégrité référentielles sont créées lorsque toutes les tables et leurs données ont été importées.

Si les contraintes d'intégrité existent déjà dans la base cible, l'ordre d'Import des tables, géré lors de l'Export, permet d'éviter tout problème.

Peut-on changer le nom d'une table ?

La réponse est non. Il n'est pas possible d'exporter une table sous un nom et de l'importer sous un autre. Pour cela, il faut renommer la table avant l'Export ou après l'Import.

Quelle répercussion à l'Import sur les index, contraintes et triggers d'une table ?

Aucun, en ce sens que l'Import crée les nouveaux objets s'ils n'existent pas, mais ne supprime pas ceux qui existent déjà.

Des problèmes peuvent survenir si un objet importé entre en conflit avec un autre de même nom, existant déjà dans la base de données cible. Par exemple, deux tables de structure différente portant le même nom.

Comment sont gérés les droits et les synonymes ?

Les droits (GRANT) et les synonymes sont importés suivant l'ordre dans lequel ils ont été créés.

À quoi sert l'option COMPRESS=Yes ?

Cette option est très utile pour réorganiser vos bases de données en évitant toute fragmentation des tables et des index. Une table ou un index possède une taille initiale. Lorsque cet espace alloué est plein, un extent (une extension) est automatiquement créé. La taille du segment initial, du segment suivant et de l'incrément donné aux autres segments sont des paramètres liés à chaque table et index. Ils sont précisés lors de la création de la table et peuvent être modifiés ultérieurement. Par exemple :

```
CREATE TABLE scott.emp (
    empno  NUMBER(4),
    ...
    deptno NUMBER(2) )
STORAGE (
           INITIAL 1 M-- 1 Mo pour l'extent initial
           NEXT 100k-- taille du premier extent
           PCTINCREASE 10 -- % d'incrément des extents suivants
      )
```

Si des `extents` sont alloués lorsque la taille de l'objet augmente, ils perdurent lorsque sa taille diminue.

Il est plus facile et plus rapide pour la base Oracle 10*g* de lire des données contenues dans un même segment, car des segments dispersés au sein d'un tablespace obligent le bras du disque dur à de fréquents déplacements. Le gain est particulièrement sensible pour les index.

Une optimisation de premier niveau de la base de données consiste à regrouper une table ou un index en un seul `extent`.

L'exportation d'une table ou d'un index contient son ordre SQL de création avec les paramètres de STORAGE initiaux. L'option EXP COMPRESS=Y calcule la somme des différents `extents` et l'affecte au segment INITIAL.

L'importation ne modifie pas les paramètres STORAGE si l'objet existe déjà dans la base cible. Pour supprimer les `extents` d'une table ou d'un index il faut supprimer l'objet avant de l'importer pour, qu'à sa création, les nouveaux paramètres STORAGE soient pris en compte.

> Comme cette opération s'effectue souvent sur des bases de production, nous vous conseillons de vous entraîner et de disposer d'une sauvegarde avant de supprimer vos tables et leurs données...

Notez que si la table importée nécessite un `extent` INITIAL de grande dimension, la base de données cible ne dispose peut-être pas d'autant de place en un seul segment contigu. L'annexe 3, *Procédures pour le DBA*, possède des ordres SQL qui vérifient ces aspects.

Comment diminuer la taille d'une table ou d'un index ?

Il est possible qu'une table soit de taille importante mais relativement vide car des enregistrements ont été supprimés. Il est alors préférable de la pré-créer manuellement. Pour cela procédez comme suit :

- exportez la table sans ses données :

```
exp userid=scott/tiger table=(emp)
```

- récupérez dans un fichier texte l'ordre SQL de création de la table en effectuant un Import avec le paramètre :

```
imp userid=scott/tiger table=(emp) INDEXFILE=fichier.txt
```

- modifiez manuellement la clause STORAGE de l'ordre de création ;
- créez la table sous SQL*Plus en exécutant l'ordre SQL modifié.

> Il est aussi possible d'utiliser les procédures standard DBMS_DESCRIBE ou DBMS_METADATA.

Comment récupérer les ordres SQL de création de tous les objets d'une base ?

La méthode précédente, utilisant le paramètre INDEXFILE, permet de récupérer dans un fichier texte l'ensemble des ordres SQL de création des objets de la base de données (tables, vues, index, triggers…).

Ce paramètre doit être utilisé uniquement lors de l'importation. Son utilisation inhibe toute action sur la base de données cible.

Si une table et son index ont des propriétaires différents, que se passe-t-il ?

L'utilisateur SCOTT possède par exemple la table EMP et TOTO a créé un index sur cette table. L'Export suivant ne porte pas sur l'index qui a été créé par l'utilisateur TOTO :

```
exp userid=scott/tiger tables=(emp) indexes=Y
```

> Les règles de bonne programmation imposent que tous les objets d'une application soient créés sous le même compte utilisateur. Des droits ou des rôles (lecture, lecture-écriture...) sont ensuite attribués aux autres utilisateurs.

Qu'en est-il des tablespaces OFFLINE ?

Un tablespace peut être actif (ONLINE) ou inactif (OFFLINE). Lors d'une exportation, les données et objets contenus dans un tablespace OFFLINE ne sont pas exportés. C'est

pourquoi un Export FULL ne peut pas être considéré comme une sauvegarde fiable et complète à 100 %.

Les informations du tablespace SYSTEM sont-elles automatiquement exportées ?

Non, les informations concernant les fichiers du tablespace SYSTEM (emplacement, nombre, taille...) ne sont pas exportées. En effet, pour importer une base, la base cible fonctionne, donc son tablespace SYSTEM et ses fichiers existent déjà.

Comment connaître le résultat d'un Import lancé sous `at`?

`at` est l'outil d'ordonnancement standard de Windows qui permet de planifier des tâches. Il est l'outil idéal pour exécuter automatiquement des exportations et leurs sauvegardes durant la nuit. Pour disposer d'un compte-rendu d'exécution d'exportation, utilisez l'option LOG.

Quelles sont les erreurs rencontrées ?

Un Export/Import peut se terminer de trois façons :

- il peut aboutir avec succès ;
- il peut rencontrer un avertissement (warning) ;
- il peut comporter des erreurs.

Un avertissement peut être mineur : c'est le cas d'une erreur dans le nom de la table à exporter. Mais certaines erreurs peuvent provoquer l'arrêt brutal de Export/Import.

Vous pouvez récupérer dans des fichiers les erreurs rencontrées lors d'un import. Ce point est traité dans la partie « export » de ce chapitre.

Oracle Data Pump

Oracle Data Pump Export et Oracle Data Pump Import sont la nouvelle génération des utilitaires Export/Import apparus avec Oracle 10*g*. Ils offrent de meilleures performances que l'import/export et permettent l'échange à la fois des données et de la structure d'une base.

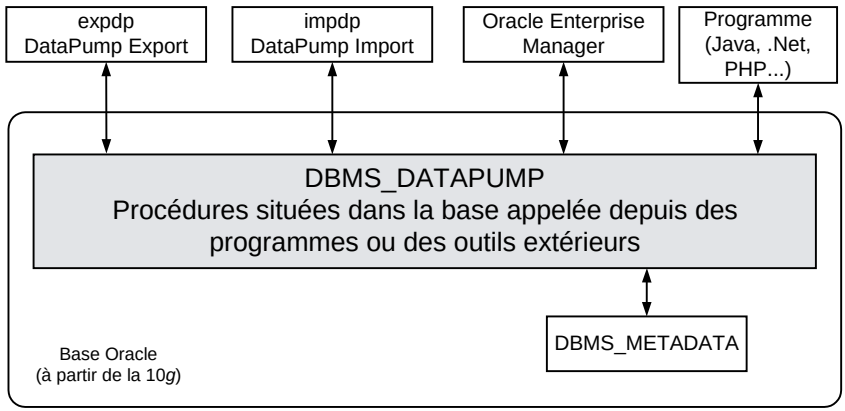
Les nouveautés apportées par Data Pump

Outre les performances, la nouveauté majeure apportée par Data Pump concerne la possibilité d'arrêter ou de relancer un Data Pump Export ou Import. Comme Data Pump cible les gros volumes de données, cette nouveauté est très intéressante.

Les différences entre Data Pump et Export/Import

Tout d'abord, seules des données extraites par Data Pump Export peuvent être importées par Data Pump Import. Il y a donc une rupture avec Export/Import qui permet d'effectuer des transferts de données d'une version d'Oracle vers une autre. Data Pump sera donc à utiliser à partir d'Oracle 10*g* et pour les futures versions.

Figure 18-2
Principaux composants de Data Pump



Autre point, Export et Import peuvent être utilisés en mode client-serveur. Ce mode de fonctionnement fait transiter les données par un réseau et ralentit la durée du traitement : il convient de l'éviter lorsque l'on manipule de gros volumes de données. C'est pourquoi Data Pump ne travaille qu'en « local », sur le serveur où est située la base. Data Pump Export et Import manipulent exclusivement des données situées sur un serveur.

Pouvoir relancer des Data Pump Export ou Import est très intéressant pour les DBA qui manipulent de gros volumes de données ou lorsque les opérations sont longues. Les cas rencontrés peuvent être les manques d'espace disque (pour le Data Pump Export) ou un tablespace sous-dimensionné (pour le Data Pump Import). Lorsque le traitement Data Pump rencontre une erreur, le DBA peut intervenir, corriger l'erreur puis relancer le traitement. Les deux exemples suivants montrent comment relancer un Data Pump Export et Import.

Comment relancer Data Pump Export

Ce premier exemple montre comment relancer un Data Pump Export. Nous allons effectuer un Data Pump Export en précisant une taille de fichier bien inférieure à celle des données à exporter. Une fois l'erreur rencontrée, nous ajouterons un fichier au Data Pump Export puis nous continuerons l'opération.

Tout d'abord, préparons les répertoires où seront exportées les données :

```
create directory mon_repertoire_pump as 'c:\export\datapump';
grant read, write on directory mon_repertoire to system;
```

Ensuite, utilisons `expdp` (*Export Data Pump*) pour lancer l'export, en précisant un nom de job facile à identifier : mon_job.

```
expdp system/manager schemas=gros_schema \
directory= mon_repertoire_pump \
logfile=mon_job.log filesize=100000 dumpfile=mon_job1.dmp \
job_name=mon_job
```

Le signe \ indique que la commande continue sur une autre ligne et qu'elle ne doit pas être immédiatement interprétée.

Le Data Pump Export va démarrer puis indiquer une erreur, le fichier cible étant trop petit. Une fois l'erreur rencontrée, connectons-nous sous SQL*Plus, sous l'utilisateur system.

```
SQL> select job_name, state from dba_datapump_jobs ;

JOB_NAME   STATE
------------------  -------------------
MON_JOBNOT RUNNING
```

Pour corriger le problème, attachons-nous au job MON_JOB en cours et ajoutons un second fichier au fichier initial d'export.

```
expdp system/manager attach=MON_JOB

Export: Release 10.1.0.2.0 - Production le Samedi 4 Septembre 2004
...
...
Job : MON_JOB
    Owner : SYSTEM
    Operation : EXPORT
...
...
```

Nous sommes maintenant dans l'outil Export : ajoutons le fichier.

```
Export> add_file=mon_job2.dmp
```

La commande status nous permet de voir la modification apportée.

```
Export> status
Job=MON_JOB
Owner : SYSTEM
    Operation : EXPORT
    Mode=SCHEMA
    STATE= IDLING
    Bytes Processed: 60,453
    Percent done: 80
    Current Parallelism: 1
    Job Error Count: 0
    Dump file Size: F:\travail\oracle\mon_job1.dmp
           Size: 100,000
           Bytes written: 54,000
    Dump file Size: F:\travail\oracle\mon_job2.dmp
```

Il faut maintenant relancer le job par la commande CONTINUE_CLIENT.

```
Export> continue_client - on peut aussi utiliser START_JOB
```

Comment relancer Data Pump Import

Dans ce second exemple, importons les données provenant d'un Data Pump Export dans un tablespace trop petit pour les contenir. Tout d'abord, créons le petit tablespace.

```
SQL> create tablespace petit_tablespace
  datafile 'D:\database\TEST\petit_tablespace1.dbf'
  size 500k extent management local ;
```

Lançons le Data Pump Import.

```
impdp system/manager dumpfile=gros_fichier.dmp \
remap_tablespace=system:petit_tablespace
logfile=mon_job_import.log job_name=mon_job_import

Import: Release 10.1.0.2.0 - Production le Samedi 4 Septembre 2004
...
...
Processing object type SCHEMA_EXPORT/TABLE/TABLE
...
...
ORA-39171: Job is experiencing a resumable wait.
ORA-01658: unable to create INITIAL extent for segment in tablespace PETIT_TABLESPACE.
```

Le job est en attente. Par défaut, la durée d'attente est de deux heures. Passé ce délai, le job est tué. Cela laisse le temps au DBA d'intervenir et évite de laisser des jobs non terminés.

Le job étant bloqué, interrompons-le par un « control C ».

```
^C
Import> stop_job=immediate
```

Ajoutons un fichier au tablespace.

```
SQL> alter tablespace petit_tablespace
  Add datafile 'D:\database\TEST\petit_tablespace2.dbf'
  size 10M autoextent on maxsize 100M ;
```

Attachons-nous au job et relançons-le.

```
impdp system/manager attach=mon_job_import

Import: Release 10.1.0.2.0 - Production le Samedi 4 Septembre 2004
...
...
Job Error Count: 0
...
Worker 1 Status:
    State: UNDEFINED
    Object Schema: GROS_SCHEMA
    Object name: COMMANDES
```

```
Object type: SCHEMA_EXPORT/TABLE/TABLE
Completed objects: 32
Worker parallelism: 1
```

Il faut alors relancer le job.

```
Import> start_job
```

Tout comme pour l'export, nous pouvons vérifier l'avancement de l'import par la commande `status`. Lorsque le job est terminé, consultez les fichiers log.

Utiliser Data Pump pour récupérer le code d'une procédure

L'Export traditionnel permet de récupérer le code de procédures, mais le script généré contient aussi le code des tables, vues, index… que vous ne désirez peut-être pas. Par exemple, pour recréer une ou plusieurs procédures sur une autre base ou pour un utilisateur particulier, la commande suivante vous permet de récupérer uniquement le code des procédures et rien d'autre : ni tables, ni index, etc.

```
expdp system/manager directory=mon_repertoire_pump \
dumpfile=exp_procedure.bmp include=PROCEDURE
```

Le fichier exp_procedure.bmp est créé au format Data Pump. Pour extraire le code SQL des procédures :

```
Impdp system/manager directory=mon_repertoire_pump \
dumpfile=exp_procedure.dmp sqlfile=code_procedure.sql
```

Quelles sont les autres possibilités de Data Pump ?

Les possibilités offertes par Data Pump Export et Import sont en grande partie similaires à celles de l'Export et Import traditionnels. Pour découvrir les autres possibilités, je vous conseille d'utiliser en ligne expdp help=Y ou impdp help=Y.

Vous pouvez aussi accéder à toutes les possibilités de Data Pump depuis la base de données par la procédure DBMS_DATAPUMP. Vous pouvez ainsi imaginer construire des procédures en PL/SQL, appelables depuis un site Web (Java, .Net, PHP…) ou tout autre programme qui utilisera Data Pump.

Oracle SQL*Loader

Nous avons choisi de vous présenter SQL*Loader par le biais d'une série de questions/réponses et d'exemples.

*Quelle utilisation pour SQL*Loader ?*

SQL*Loader est un utilitaire servant à charger des données provenant de fichiers externes, dans une base Oracle 10*g*. Il permet d'intégrer de nombreux formats de données, à partir d'un ou de plusieurs fichiers, dans une ou plusieurs tables.

C'est l'outil fréquemment utilisé lors de migrations d'une base vers une autre ou lors de la construction d'un datawarehouse ou autre infocentre.

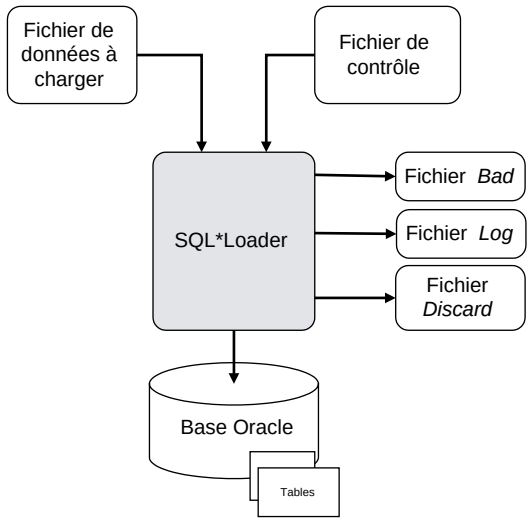
Comment fonctionne SQL*Loader ?

Pour piloter le chargement des données, SQL*Loader utilise un *fichier de contrôle*. Ce dernier détermine les fichiers de données à charger et leur table de destination.

Durant le chargement, SQL*Loader peut produire trois types de fichiers :

- le fichier log, qui effectue un compte rendu complet du chargement ;
- le fichier bad, qui conserve les enregistrements qui n'ont pu être insérés (rejetés lors de l'insertion dans la table cible) ;
- le fichier discard, qui stocke les enregistrements illisibles (provenant du fichier de données à charger, mais non reconnus par SQL*Loader).

Figure 18-3
*Principe de fonctionnement de SQL*Loader*



Où se situe le jeu d'essai de SQL*Loader ?

Oracle 10*g* est livré avec un jeu d'essai SQL*Loader. Les fichiers d'essai se terminent tous avec l'extension .ctl et se situent généralement dans :

```
ORACLE_HOME\rdbms\demo
```

Nous vous conseillons de les étudier en complément de ce chapitre.

Comment utiliser SQL*Loader ?

SQL*Loader est avant tout un utilitaire « mode caractère ». Il peut être piloté par un utilitaire graphique, qui ne fera que transmettre les différents paramètres à la commande en ligne.

Sous Windows, il se nomme sqlldr et se situe dans ORACLE_HOME\bin. Pour accéder à l'aide en ligne :

```
C:\> sqlldr
```

La commande usuelle se présente comme suit :

```
C:\> sqlldr utilisateur/mot_de_passe@alias control=controle.ctl

avec : Utilisateur/mot_de_passe@alias : connexion à la base Oracle 10g locale ou distante.
```

Le fichier de contrôle controle.ctl contient les données suivantes :

```
        load data
1         infile *
2          replace
3          into table COURS
4         ( codeposition (02:05) char(4),
            nomposition (08:27) char(20)
          )
5       begindata
          FRAN  LANGUE FRANCAISE
          ENGL  LANGUE ANGLAISE
          MATH  MATHEMATIQUES
          PHYS  PHYSIQUE
```

La numérotation indiquée ne doit pas figurer, elle sert à commenter l'exemple.

En outre, il faut tenir compte du fait que :

1. les données ne proviennent pas d'un fichier mais sont présentes à la fin du fichier de contrôle ;
2. les données de la table COURS seront remplacées ;
3. la table cible, dans la base Oracle 10*g*, doit être accessible à l'utilisateur de connexion ;
4. il faut insérer des informations dans les colonnes CODE et NOM de la table COURS. Pour les données fournies en entrée, le CODE sera lu entre le deuxième et le cinquième caractère, le NOM entre le huitième et le vingt-septième ;
5. les données suivantes sont à insérer.

Comment générer les fichiers log, bad, discard ?

Ces fichiers sont créés soit à partir de la commande en ligne de SQL*Loader, soit à partir du fichier de contrôle :

```
C:\> sqlldr user/pwd control=test.ctl log=test.log bad=test.bad discard=test.dsc

load data
```

```
infile 'fichier_donnees.dat'
badfile 'test.bad'
discardfile 'test.dsc'
truncate
into table test
 (colonne1,
  colonne2,
  colonne3)
```

Pour faciliter leur repérage, il est préférable d'utiliser le même nom pour les fichiers de contrôle, à savoir : *bad, log, discard*, plutôt que celui de la table que vous chargez.

De même, disposer du plus d'éléments possibles dans les fichiers de contrôle en facilite l'écriture, la mise au point et la maintenance.

Les données à insérer peuvent-elles être de longueur variable ?

La réponse est oui. Dans l'exemple suivant, la première et la seconde colonne sont de longueur fixe :

```
LOAD DATA
   INFILE *
   INTO TABLE test
 1  FIELDS TERMINATED BY "," OPTIONALLY ENCLOSED BY '"'
 2 TRAILING NULLCOLS
   (  colonne1,
    colonne2
   )
   BEGINDATA
   11111,AAAAAAAAAA
   222,"A,B,C,D,"
```

Les données seront séparées par des virgules et peuvent être entre guillemets (").

TRAILING NULLCOLS signifie qu'à la fin de chaque champ inséré, les « blancs » ou « espaces » seront supprimés. On évite de la sorte les colonnes qui contiennent des valeurs de type "XXX " avec le caractère " " qui « remplit » inutilement le champ.

N'oubliez pas qu'avec Oracle, une colonne de type VARCHAR2(5) ne contient que les données stockées et que "XXX" et "XXX " ne sont pas identiques.

Les données à insérer peuvent-elles être de longueur fixe ?

La réponse est oui. L'exemple suivant fait appel à un positionnement absolu des données lues dans le fichier d'entrée :

```
   LOAD DATA
   INFILE *
```

```
INTO TABLE positionnement_absolu
(  data1 POSITION(1:5),
   data2 POSITION(6:15)
)
BEGINDATA
11111AAAAAAAAAA
22222BBBBBBBBBB
```

Attention ! Lors du comptage de la position des champs, il est facile de commettre une erreur en provoquant un décalage. Vérifiez toujours, après un chargement, que les données ont été insérées dans les colonnes souhaitées.

*SQL*Loader crée-t-il la table dans la base Oracle 10g ?*

La réponse est non. La table doit être créée avant le chargement des données.

Quels liens existe-t-il entre la table et le fichier de contrôle ?

Considérons l'exemple suivant. La structure de la table TEST est :

```
CREATE TABLE TEST (
COLONNE1      NUMBER(5) PRIMARY KEY,
COLONNE2      VARCHAR2(10),
COLONNE3      VARCHAR2(10),
COLONNE4      VARCHAR2(10)
);
```

Dans la colonne1, nous souhaitons insérer un nombre et dans les colonnes suivantes, les lettres de l'alphabet, dans l'ordre.

Le fichier de contrôle a pour rôle d'indiquer à SQL*Loader où placer les données lues dans le fichier de données :

```
LOAD DATA
   INFILE *
   INTO TABLE test
   FIELDS TERMINATED BY "," OPTIONALLY ENCLOSED BY '"'
   TRAILING NULLCOLS
   (colonne1,
    colonne2,
    colonne3,
    colonne4
   )
   BEGINDATA
   11111,AAAAAAAAAA,BBB,CCCCCCC
   22222,AAAAAAAAA,BBBBBBBBBB,CCC
   333,AAAAAAAA,BBBBBB,CCCCC
```

Dans cet exemple, l'ordre des colonnes correspond à celui des données à insérer.

```
LOAD DATA
   INFILE *
   INTO TABLE test
   FIELDS TERMINATED BY "," OPTIONALLY ENCLOSED BY '"'
   TRAILING NULLCOLS
   (colonne3,
    colonne2,
    colonne4,
    colonne1
   )
   BEGINDATA
   BBBB,AAAAA,CCCC,11111
   BBBB,AAAAAAA,CCCCCC,222
   BBBBBB,AAAAAAAAAA,CCCCC,3333333
```

Ce sont bien les données à intégrer qui définissent l'ordre des colonnes du fichier de contrôle.

```
LOAD DATA
   INFILE *
   INTO TABLE test
   FIELDS TERMINATED BY "," OPTIONALLY ENCLOSED BY '"'
   TRAILING NULLCOLS
   (colonne2,
    colonne4,
    colonne1
   )
   BEGINDATA
   AAAA,CCCC,11111
   AAAAAAA,CCCCCC,222
   AAAAAAAAAA,CCCCC,3333333
```

Dans l'exemple précédent, toutes les colonnes de la table ne sont pas insérées, car les valeurs « BBB... » sont absentes. En revanche, la colonne1 est obligatoire, car c'est une clé primaire, qui doit donc exister pour chaque enregistrement inséré.

Comment préciser le type des données en entrée ?

Le type des données dans la table cible est bien défini. Souvent, la difficulté est de préciser à SQL*Loader le type des données en entrée pour qu'il les lise correctement.

C'est grâce au fichier de contrôle que nous précisons les types :

```
LOAD DATA
   INFILE *
   INTO TABLE test
   (colonne1       INTEGER EXTERNAL,
    colonne2       CHAR,
    colonne3       INTEGER EXTERNAL,
    colonne4       DECIMAL EXTERNAL
   )
....
```

C'est en précisant ce type que vous pouvez lire n'importe quel format de données en entrée : signé, binaire packé décimal, etc.

Comment charger des données qui ne sont pas codées en ASCII ?

Si les données proviennent de systèmes informatiques dont le codage interne n'est pas en ascii, vous pouvez demander leur conversion dans le fichier de contrôle. Cela vous permet de récupérer correctement tous les caractères spécifiques à la langue française : àéèùç... C'est une possibilité très utile que propose SQL*Loader.

Par exemple, si les données proviennent d'un grand système IBM, elles sont souvent codées en EBCDIC. Transférez alors les fichiers jusqu'à la machine qui héberge la base Oracle 10*g* en mode binaire. Vous obtiendrez en local, un fichier qui sera l'image exacte (en binaire) de celui d'origine. Chargez-le en indiquant son type d'origine dans le fichier de contrôle, de sorte que SQL*Loader convertisse le fichier lors du chargement.

```
LOAD DATA
   CHARACTERSET 'WE8EBCDIC500'
   INFILE fichier_en_EBCDIC
   INTO TABLE test
  (
      colonne1
      ...
      colonneN
   )
```

Vous devrez peut-être effectuer quelques essais pour trouver le jeu de caractères convertissant correctement vos données, mais cela vaut la peine. Sinon, il faut modifier les données une fois chargées dans la base, opération longue et complexe.

Peut-on modifier les données lors de leur insertion ?

La réponse est oui. Vous disposez en outre de pratiquement toutes les fonctions SQL portant sur les nombres, chiffres, dates, etc.

```
   LOAD DATA
   INFILE *
   INTO TABLE test
1  (  numero                        "ma_sequence.nextval",
2     heure_chargement              "to_char(SYSDATE, 'HH24:MI')",
3     colonne2        POSITION(1:5)  ":colonne1/100",
      colonne1        POSITION(6:15) "upper(:colonne2)"
   )
   BEGINDATA
   11111AAAAAAAAAA
   22222BBBBBBBBBB
```

1. Insérez un numéro de séquence qui numérotera les enregistrements en entrée.
2. L'heure de chargement sera l'heure système.

3. Lors de l'utilisation des fonctions SQL, faites précéder la référence à la colonne de « : », en l'encadrant par des « " » .

Peut-on insérer des données dans plusieurs tables ?

La réponse est oui. C'est très utile lorsqu'on récupère des données provenant de fichiers qui ne sont pas normalisés au sens SQL :

```
LOAD DATA
INFILE *
REPLACE
INTO TABLE table1
     WHEN champ_test != ' '
( champ_test POSITION(1:4)   INTEGER EXTERNAL,
  colonne2   POSITION(6:15)  CHAR,
  colonne3   POSITION(17:18) CHAR,
  colonne4   POSITION(20:23) INTEGER EXTERNAL
)
INTO TABLE table2
     WHEN champ_test2 != ' '
( champ_test2 POSITION(25:27) INTEGER EXTERNAL,
  colonne2    POSITION(1:4)   INTEGER EXTERNAL
)
```

Comment charger plus vite avec SQL*Loader ?

Le paramètre DIRECT=TRUE dans la ligne de commande de SQL*Loader permet d'améliorer considérablement la vitesse de chargement de vos fichiers, en court-circuitant une bonne partie des opérations effectuées normalement par la base de données.

Certaines commandes présentes dans le fichier de contrôle (par exemple, "upper(:colonne2)") ne permettent pas d'utiliser le mode DIRECT, car elles font appel aux fonctions du SQL.

Lors du chargement en mode DIRECT, aucune indication ne s'affiche à l'écran. Seule la fin est indiquée, chose surprenante car le mode normal nous habitue à suivre la progression du chargement.

> La recommandation générale est de mettre au point vos fichiers de chargement en mode normal puis de les exécuter en mode DIRECT.

Quelle relation entre SQL*Loader et les index ?

Au fur et à mesure du chargement d'une table, les index qui s'y rapportent se construisent : il faut donc disposer de place pour le tri et la construction de l'index dans le tablespace temporaire.

En mode normal, les données sont progressivement ajoutées dans la table, et donc dans les index. Leur construction ne nécessite pas une importante zone de tri.

En mode DIRECT=TRUE, les index sont construits en une fois, à la fin du chargement de la table. Ils ont besoin de plus de place dans la zone de tri.

De manière générale, il est préférable de commencer par charger les données, puis de construire les index. Cela améliore le temps global des deux opérations.

Comment forcer SQL*Loader à effectuer un commit à la fin du chargement ?

Vous ne pouvez pas réaliser une telle opération. Néanmoins, en attribuant au paramètre ROWS=*valeur* une forte valeur, vous augmentez le nombre d'enregistrements entre deux commits.

Comment sont utilisés les rollback segments ?

Lors d'un chargement normal des données, les rollback segments doivent être suffisamment volumineux pour contenir les données entre deux commits.

Le mode DIRECT court-circuite les rollback segments.

Que contient le fichier log, résultat d'un chargement ?

Le fichier log, résultat d'un chargement, est essentiel pour valider le bon chargement des données. Le fonctionnement standard de SQL*Loader interrompt le chargement d'un fichier au bout de 50 erreurs rencontrées. Il peut donc s'arrêter avant terme.

Création de la table de test sous SQL*Plus :

```
SQL> connect scott/tiger
Connecté.
SQL> CREATE TABLE TEST (
  2  COLONNE1 NUMBER(5) PRIMARY KEY,
  3  COLONNE2 VARCHAR2(10),
  4  COLONNE3 VARCHAR2(10),
  5  COLONNE4 VARCHAR2(10)
  6  );

Table créée.
```

Fichier de contrôle :

```
LOAD DATA
   INFILE *
   INTO TABLE test
   FIELDS TERMINATED BY "," OPTIONALLY ENCLOSED BY '"'
   TRAILING NULLCOLS
   (colonne1,
    colonne2,
    colonne3,
```

```
 colonne4
)
BEGINDATA
11111,AAAAAAAAAA,BBB,CCCCCCC
22222,AAAAAAAAA,BBBBBBBBBB,CCC
333,AAAAAAAA,BBBBBB,CCCCC
```

Exécution du chargement :

```
C:\> sqlldr userid=scott/tiger control=test.ctl log=test.log
```

Voici comment se présente le fichier log, résultat du chargement de la table TEST, piloté par le fichier de contrôle précédent :

```
SQL*Loader: Release 9.0.1 - Production on Di Aug 26 15:43:18 2001

(c) Copyright 1997 Oracle Corporation.  All rights reserved.

Fichier de contrôle :  test.ctl
Fichier de données  :  test.ctl
Fichier défectueux  :  test.bad
Fichier de rebut    :  aucune spécification

 (Allouer tous les rebuts)

Nombre à charger    : ALL
Nombre à sauter     : 0
Erreurs permises    : 50
Tableau de liens    : 64 lignes, maximum de 65536 octets
Continuation        : aucune spécification
Chemin utilisé      : Conventionnel

Table TEST, chargée à partir de chaque enregistrement physique.
Option d'insertion en vigueur pour cette table : INSERT
option TRAILING NULLCOLS effective

Nom de colonne Position   Long.  Séparat. Encadrem. Type de données
------------------------------------------------------------------------
COLONNE1       FIRST      *        ,        O(")    CHARACTER
COLONNE2       NEXT       *        ,        O(")    CHARACTER
COLONNE3       NEXT       *        ,        O(")    CHARACTER
COLONNE4       NEXT       *        ,        O(")    CHARACTER

Table TEST:
  Chargement réussi de 3 Lignes.
  0 Lignes chargement impossible dû à des erreurs de données.
  0 Lignes chargement imposs. car échec de toutes les clauses WHEN.
  0 Lignes chargement imposs. car tous les champs étaient non renseignés.

Espace affecté au tableau de liens:      18944 octets(64 lignes)
Espace alloué à la mémoire en plus du tableau de liens: 0 octets
```

```
Nombre total d'enregistrements logiques ignorés :       0
Nombre total d'enregistrements logiques lus :           3
Nombre total d'enregistrements logiques rejetés :       0
Nombre total d'enregistrements logiques mis au rebut :  0

Le début de l'exécution a été effectué Di Jan 25 15:43:18 2005
La fin de l'exécution a été effectuée Di Jan 25 15:43:25 2005

Temps écoulé (ELAPSED):     00:00:06.61
Temps processeur (CPU):    00:00:00.18
```

En effectuant une lecture attentive, vous constaterez que tout est mentionné dans ce fichier log : le contenu du fichier de contrôle, les paramètres lancés lors de l'exécution de SQL*Loader et le résultat de l'exécution incorporant la durée totale du chargement.

Oracle SQL*Plus

SQL*Plus est un outil en mode caractère permettant l'accès à une base de données Oracle. Il peut être utilisé pour exécuter de façon interactive des ordres SQL unitaires, lancer des fichiers contenant un ensemble d'ordres SQL, exécuter des programmes écrits en PL/SQL, afficher le résultat de requêtes SQL, formater le résultat d'une requête et en améliorer la présentation, accéder à plusieurs bases de données simultanément, etc.

SQL*Plus remplace maintenant l'ancien Server Manager pour créer, démarrer, arrêter, administrer et effectuer des opérations complexes de restauration sur les bases Oracle.

> Server Manager n'est plus livré avec les nouvelles versions depuis Oracle9. SQL*Plus devient dont l'interface privilégiée d'Oracle par laquelle la totalité des ordres SQL concernant l'administration et l'utilisation d'Oracle peuvent être exécutés.

SQL*Plus est un outil généraliste, livré depuis des années avec toutes les versions d'Oracle. Il a l'avantage d'exister sur toutes les plates-formes où Oracle est porté et dispense de l'achat d'un outil complémentaire pour accéder à une base Oracle 10*g*. Il présente l'inconvénient d'une ergonomie en mode caractère qui peut faire préférer pour certains usages des outils graphiques parfois moins performants mais plus agréables d'utilisation.

Ce paragraphe a comme but de vous faire découvrir cet outil puissant et d'en cerner les principales caractéristiques. Nous vous présenterons de nombreux exemples dont l'utilisation de SQL*Plus pour créer des interrogations dynamiques.

*Pourquoi administrer Oracle avec SQL*Plus ?*

Pourquoi se limiter à l'utilisation d'un outil dont l'interface est en mode caractère ? OEM (*Oracle Enterprise Manager*) est un outil graphique qui reprend une partie des possibilités de SQL*Plus. Bien que l'approche graphique proposée par OEM soit très utile, elle ne nous dispense pas d'en connaître et d'en comprendre les concepts et ordres SQL

induits. De plus, dans certaines situations critiques, seul SQL*Plus peut vous permettre de redémarrer votre base. Par ailleurs, un outil mode caractère et des scripts de commande sont indispensables à l'automatisation de tâches comme le démarrage, l'arrêt et les sauvegardes de vos bases. Pour toutes ces raisons, SQL*Plus est un outil que tout administrateur Oracle se doit de connaître.

Les deux modes de connexion à SQL*Plus

Nous avons vu que SQL*Plus peut être utilisé pour deux types d'ordres SQL :

- des ordres SQL de création, démarrage, arrêt, restauration de la base ;
- des ordres SQL d'utilisation de la base.

Le mode de connexion est différent suivant le type d'utilisation car la création ou le démarrage d'une base nécessitent plus de pouvoirs qu'une simple utilisation.

Connexion à SQL*Plus en mode « administration »

Dans chaque outil Oracle, le lancement est immédiatement suivi d'une demande de connexion à une base de données existante. Mais comment opérer sur une base qui n'est ni créée ni démarrée ? À cet instant, la base de données ne peut intervenir dans le processus d'identification et d'autorisation d'accès. À cet effet, l'utilisateur Windows qui exécute SQL*Plus doit bénéficier de droits supplémentaires provenant de privilèges issus du système d'exploitation ou vérifiés dans un fichier mot de passe situé à l'extérieur à la base. Ces points sont détaillés au chapitre 24, *Stratégie de sécurité sous Windows*.

Les privilèges SYSDBA et SYSOPER permettent de se connecter à Oracle avec plus de privilèges. Ils sont décrits au chapitre évoqué précédemment.

L'utilisation de SQL*Plus pour créer, démarrer, arrêter ou restaurer une base Oracle nécessite obligatoirement l'une (ou les deux) conditions suivantes :

– être connecté sous un utilisateur Windows possédant des privilèges OS supplémentaires,
– être un utilisateur Oracle ayant reçu les privilèges SYSDBA ou SYSOPER qui sont vérifiés dans un « fichier mot de passe » conservé à l'extérieur de la base.

Pour un utilisateur, les privilèges Oracle SYSOPER et SYSDBA permettent les actions suivantes :

- CREATE DATABASE ;
- STARTUP ;
- SHUTDOWN ;
- ALTER DATABASE OPEN / MOUNT ;
- ALTER DATABASE BACKUP CONTROLFILE ;
- ALTER TABLESPACE BEGIN / END BACKUP ;

- ARCHIVE LOG ;
- RECOVER.

Certaines de ces commandes sont étudiées aux chapitres 14 et 10, *Création d'une base Oracle 10*g et *Démarrer et arrêter une base Oracle 10*g.

Pour se connecter en mode administration, les syntaxes possibles sont les suivantes. Elles sont décrites au chapitre 24, *Stratégie de sécurité sous Windows*.

```
C:\> sqlplus /nolog

# connexion avec un utilisateur ayant reçu le privilège SYSDBA
connect utilisateur/passwd as SYSDBA

# connexion avec un utilisateur ayant reçu le privilège SYSOPER
connect utilisateur/passwd as SYSOPER

# connexion sous le privilège SYSDBA ou SYSOPER
# pour cette syntaxe, l'utilisateur doit avoir reçu un pouvoir
# supplémentaire hérité du système d'exploitation
connect / as SYSDBA
connect / as SYSOPER

# idem mais en client-serveur
# (nécessite l'utilisation d'un « fichier mot de passe »)
# sur le serveur cible.
connect utilisateur/passwd@alias as SYSDBA
connect utilisateur/passwd@alias as SYSOPER
```

L'usage de l'ancienne syntaxe CONNECT INTERNAL est interdite. Il faut maintenant utiliser l'une des syntaxes précédentes.

Connexion à SQL*Plus en mode « utilisation »

Pour exécuter des ordres SQL autres que ceux mentionnés au paragraphe précédent, la syntaxe de connexion est plus simple. Utilisez la commande suivante pour lancer SQL*Plus :

```
sqlplus
```

Un utilisateur Oracle et son mot de passe vous sont alors demandés. Vous pouvez aussi saisir des options lors du lancement de SQL*Plus :

```
C:\> sqlplus -S utilisateur/mot_de_passe@base_cible @fichier_sql
```

- -S est un mode « silencieux » ;
- l'utilisateur et son mot de passe peuvent être suivis de l'alias Oracle Net de la base cible à laquelle vous souhaitez vous connecter. Dans ce cas, le signe @ est accolé au mot de passe ;

- un fichier de commande contenant des ordres SQL, des commandes SQL*Plus et PL/SQL peut être automatiquement lancé, un espace doit alors précéder le signe @ ;
- Vous pouvez aussi lancer SQL*Plus en indiquant qu'il doit se comporter comme une version précédente, en indiquant la version souhaitée. Vous aurez ainsi une garantie supplémentaire pour vos scripts existants.

> Notez la différence entre les deux exemples précédents :
> *sqlplus utilisateur/pwd@base_cible* se connecte à une base de donnée particulière ;
> *sqlplus utilisateur/pwd@fichier.sql* exécute automatiquement le fichier de commande cité sur la base indiquée par la variable d'environnement ORACLE_SID.

Vous pouvez aussi utiliser les syntaxes :

```
sqlplus /nolog
connect utilisateur/mot_de_passe
connect utilisateur/mot_de_passe@alias_oracle_net
connect utilisateur/mot_de_passe@fichier_a_executer
```

Pour accéder à l'aide décrivant l'ensemble des syntaxes accessibles lors du lancement de SQL*Plus, il faut appeler :

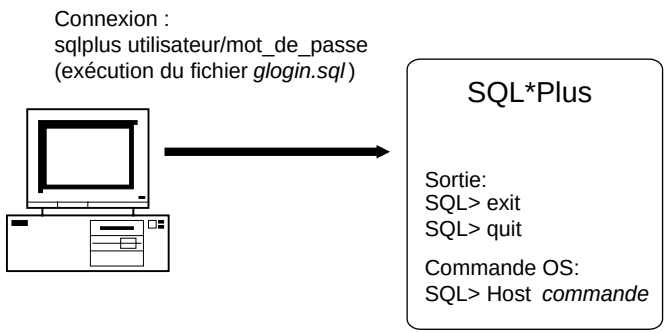
```
sqlplus -
```

La commande EXIT vous permet de sortir de SQL*Plus.

Interactions entre SQL*Plus et son environnement

La figure suivante décrit les liens entre SQL*Plus et son environnement.

Figure 18-4
*Interactions entre SQL*Plus et son environnement*



Les grandes caractéristiques sont les suivantes :

- lors du lancement de SQL*Plus, un nom d'utilisateur, son mot de passe et la base de donnée cible vous sont demandés. Sous certaines conditions, le mot de passe et la base cible sont optionnels ;

- lors de la connexion à la base Oracle 10*g*, le fichier `glogin.sql` est exécuté. Ce fichier, situé sur la machine qui héberge le code exécutable de SQL*Plus (le serveur si vous êtes sur le serveur, un poste Windows si vous êtes en client-serveur), peut contenir toutes sortes d'ordres SQL ;
- par défaut, l'invite d'une session SQL*Plus est *SQL> ;*
- à partir de la session SQL*Plus, vous pouvez lancer des commandes OS par la commande HOST. Celle-ci ne termine pas votre session SQL*Plus qui peut être bloquée ou, au choix, active en attendant la fin de l'exécution de la commande OS ;
- vous pouvez vous déconnecter de la base Oracle cible tout en restant dans SQL*Plus au moyen de la commande DISC (DISCONNECT) ;
- pour se connecter à un autre compte Oracle ou à une autre base de données, utilisez la commande CONNECT ;
- pour se déconnecter et terminer une session SQL*Plus, utilisez EXIT ou QUIT.

Les différentes commandes de SQL*Plus

Différentes commandes peuvent être lancées depuis l'interface SQL*Plus. Par leur diversité, interrogation, débogage, éditeur de texte…, il est souvent facile de les confondre. Les commandes SQL*Plus peuvent être regroupées par famille :

- des ordres SQL ;
- des ordres de création, démarrage, arrêt d'une base ;
- des commandes de type « éditeur de texte » ;
- des commandes de contrôle d'environnement :
- contrôle d'entrée/sortie ;
- dialogue avec l'utilisateur ;
- formatage et présentation d'éditions ;
- définition de l'environnement ;
- contrôle de l'exécution.

Installation d'un jeu d'essai

Si la base de données a été créée par défaut, l'utilisateur SCOTT et son jeu d'essai ont été installés.

Si vous souhaitez placer ce jeu d'essai sur une nouvelle base de données (ou sous un autre compte utilisateur), créez cet utilisateur et exécutez le script sous le compte SYSTEM, en ayant préalablement créé l'utilisateur SCOTT :

```
#sous l'utilisateur SYSTEM
create user SCOTT identified by TIGER;
```

```
# se connecter sous SCOTT
connect SCOTT/TIGER
# Lancer le script
ORACLE_HOME\RDBMS\ADMIN\SCOTT.SQL
```

Chargement de l'aide en ligne SQL*Plus

Sous SQL*Plus, la commande *Help* vous permet de disposer en ligne des principales syntaxes SQL. Pour installer cette aide, il faut définir la variable locale SYSTEM_PASS contenant l'utilisateur et le mot de passe administrateur et lancer un script d'installation :

```
C:\> cd ORACLE_HOME\bin
C:\> set SYSTEM_PASS=system/manager
C:\> helpins
```

Que signifie le message « Product User Profile is not loaded » ?

Ce message apparaît lors du lancement de SQL*Plus dans certaines circonstances.

SQL*Plus est un outil très puissant. Vous pouvez en limiter l'usage grâce à la table PRODUCT_USER_PROFILE, dans laquelle vous précisez les utilisateurs qui peuvent se connecter à SQL*Plus et leurs droits d'utilisation du produit.

Si la base de données a été créée par défaut, ou si vous avez sélectionné l'option *Create Database Object* lors de l'installation de SQL*Plus, cette table a été établie automatiquement sous le compte de l'administrateur SYSTEM.

Si une table d'aide n'a pas été créée en même temps que la base, il faut la créer pour supprimer le message « Product User Profile is not loaded » qui apparaît à chaque lancement de SQL*Plus. Pour cela, la variable locale SYSTEM_PASS doit contenir l'utilisateur et le mot de passe administrateur :

```
C:\> cd ORACLE_HOME\bin
C:\> set SYSTEM_PASS=system/manager
C:\> pupbld
```

Le fichier `ORACLE_HOME\sqlplus\admin\pupbld.sql` est alors lancé.

Les ordres SQL et les commandes SQL*Plus

Sous SQL*Plus, il faut différencier ces types de commande :

- les ordres SQL qui sont adressés à la base de données ;
- les commandes SQL*Plus qui servent à gérer l'affichage, la présentation, en un mot à « habiller » les ordres SQL dans un environnement de travail plus convivial.

Pour chacune de ces commandes, la syntaxe de lancement est différente :

- un ordre SQL est toujours terminé par le signe « ; » ou le signe « / » qui déclenche l'exécution de l'ordre ;
- une commande SQL*Plus est exécutée sans caractère de fin particulier.

Par exemple, l'ordre SQL suivant (qui peut être saisi sur plusieurs lignes) est exécuté par le signe « ; » :

```
SQL> select * from scott.dept
  2  where deptno = 10
  3  ;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
```

Si aucun caractère d'exécution ne vient terminer l'ordre SQL, celui-ci est présent dans le buffer de commande de SQL*Plus, mais n'est pas exécuté.

> Dans l'exemple précédent, si l'ordre SQL s'étend sur plusieurs lignes, les numéros des lignes sont indiqués. C'est la fameuse « erreur 2 » que de nombreux utilisateurs de SQL*Plus signalaient au support Oracle, en voyant s'afficher « 2 » à la fin de la saisie d'un ordre SQL. Il suffit alors d'ajouter « ; » pour lancer l'exécution de l'ordre SQL courant.

Dans le cas d'une commande SQL*Plus, aucun caractère particulier n'est nécessaire :

```
SQL> show user
USER is "SYSTEM"
```

Si vous lancez successivement plusieurs commandes SQL et PL/SQL, seul le dernier ordre SQL reste dans le buffer SQL de SQL*Plus. La commande SQL*Plus `list` ou `l` permet de visualiser la dernière commande SQL saisie et la commande `run` ou `r` exécute l'ordre SQL présent dans le buffer :

```
SQL> l
SQL> select * from scott.dept
  2  where deptno = 10
  3 *
SQL> r

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
```

Utilisation de SQL*Plus avec PL/SQL

Dans le chapitre 17, *Programmer avec PL/SQL*, nous présentons l'utilisation de SQL*Plus pour réaliser des procédures, des fonctions, des triggers et des packages en langage PL/SQL.

Utiliser un éditeur de texte pour modifier vos ordres SQL

Comment faire pour modifier un ordre SQL comportant des erreurs de syntaxe ? SQL*Plus possède un éditeur de texte en mode ligne très rudimentaire, qui se rapproche de l'antique « edlin », sous DOS ! Nous vous déconseillons fortement de l'utiliser…

À la place, la commande ed pour edit permet d'appeler l'éditeur de texte de votre choix et y place l'ordre SQL présent dans le buffer courant. Lors de la sortie de l'éditeur, l'ordre SQL modifié sera rétabli dans le buffer courant.

Pour préciser l'éditeur de texte de votre choix (ici notepad), utilisez la commande SQL*Plus :

```
SQL> def _editor = notepad
```

> Attention, l'éditeur choisi doit être un éditeur en mode ascii, à l'exclusion des logiciels de traitement de texte qui incorporent leurs caractères de contrôle dans le texte ascii.

Les commandes SQL*Plus étant conservées jusqu'à la sortie de l'outil, nous vous conseillons de positionner la commande précédente dans le fichier glogin.sql. L'éditeur appelé par ed sera alors déterminé pour chaque lancement de SQL*Plus. Ce fichier se trouve dans ORACLE_HOME\sqlplus\admin\glogin.sql.

Créer un fichier de commandes SQL

L'utilisation de `ed` permet aussi de travailler avec des fichiers texte contenant à la fois des ordres SQL et des commandes SQL*Plus.

```
SQL> ed mon_fichier.sql
```

> La saisie de l'extension .sql n'est pas obligatoire.

Par exemple, ce fichier peut contenir une suite d'ordres tels que :

```
--
-- Fichier contenant des ordres SQL et SQL*Plus
--
show user

select * from scott.dept
where deptno = 10
;
select ename, job, sal from scott.emp
where deptno = 10
/
```

Exécuter un fichier de commande à partir de SQL*Plus

Exécutez le fichier de commande précédent par la commande suivante :

```
start mon_fichier.sql
```

Pour notre exemple, le résultat est alors :

```
SQL> start mon_fichier.sql

USER is "SYSTEM"
```

```
   DEPTNO DNAME          LOC
--------- -------------- -------------
       10 ACCOUNTING     NEW YORK


ENAME      JOB              SAL
---------- --------- ---------
CLARK      MANAGER         2450
KING       PRESIDENT       5000
MILLER     CLERK           1300
```

SQL*Plus vous propose tout un ensemble de commandes pour améliorer la présentation des données obtenues. Nous vous les présentons ci-après.

Mesurer la durée de vos ordres SQL

L'option SQL*Plus SET TIMING ON|OFF permet de mesurer la durée d'exécution de vos ordres SQL (en millisecondes). Attention à son interprétation, car le temps affiché mesure la durée de l'ordre SQL ainsi que le transport et l'affichage des enregistrements retournés par un ordre SELECT.

```
set timing on

select ename, job, sal from scott.emp
where deptno = 10
/

ENAME      JOB              SAL
---------- --------- ---------
CLARK      MANAGER         2450
KING       PRESIDENT       5000
MILLER     CLERK           1300

real: 520
set timing off
```

Exécuter une commande OS

@nom_de_fichier	Exécute le fichier spécifié.
@@nom_de_fichier	Si un fichier exécuté par *@nom_de_fichier* doit lui aussi lancer ses propres fichiers, ils seront appelés par *@@nom*_de_fichier.
START nom_de_fichier	Exécute le fichier spécifié.

La commande HOST permet de lancer tout type de commande ou de programme présent sur la machine où s'exécute SQL*Plus. Suivant les versions de SQL*Plus, il est possible grâce à HOST d'exécuter en totalité une commande OS, puis de revenir dans SQL*Plus, ou de lancer cette commande et de revenir dans SQL*Plus sans attendre la fin de l'exécution.

```
host commande_à_lancer paramètre1 paramètre2 ...
```

Interrompre un ordre SQL

Pour interrompre un ordre SQL en cours (par exemple un ordre SELECT qui rapatrie trop d'enregistrements), on peut mettre un terme à l'affichage écran en appuyant sur les touches [Ctrl]+[c] pour les systèmes BSD, ou [Delete] pour les System V.

Inclure des commentaires

-- texte	Utilisé pour inclure des commentaires dans vos ordres SQL. Le texte qui suit -- est alors ignoré lors du traitement de l'ordre SQL. Le commentaire se poursuit jusqu'à la fin de la ligne courante.
/* texte */	Le texte encadré est ignoré même s'il s'étend sur plusieurs lignes.
REM texte	Remarque. Utilisé pour inclure des commentaires dans vos ordres SQL. Doit obligatoirement être situé en début de ligne et ne concerne que la ligne courante.

Écrire des commandes interactives avec SQL*Plus

```
ACCEPT variable PROMPT texte
```

Cette commande permet de définir une variable et d'afficher un texte avant sa saisie. La variable saisie peut être utilisée dans vos ordres SQL, préfixée du signe &.

```
SQL> ACCEPT v_nom PROMPT 'Recherche utilisateur : '

Recherche utilisateur : MILLER
old   2: where upper(ename) = '&v_nom'
new   2: where upper(ename) = 'MILLER'

SQL>SELECT ename, job, sal FROM emp
 2  WHERE UPPER(ename) = UPPER( '&v_nom')
 3  /

ENAME      JOB              SAL
---------- --------- ---------
MILLER     CLERK           1300
```

La variable avant et après substitution de la valeur est affichée. Vous pouvez bloquer cet affichage :

```
SQL> SET VERIFY OFF
SQL> ACCEPT v_nom PROMPT 'Recherche utilisateur : '

Recherche utilisateur : MILLER

SQL>SELECT ename, job, sal FROM emp
 2  WHERE UPPER(ename) = UPPER( '&v_nom')
 3  /

ENAME      JOB              SAL
---------- --------- ---------
MILLER     CLERK           1300
```

La commande suivante définit une variable sous SQL*Plus sans qu'il faille en saisir manuellement le contenu :

```
SQL> DEFINE v_nom MILLER

SQL>SELECT ename, job, sal FROM emp
 2  WHERE UPPER(ename) = UPPER( '&v_nom')
 3  /

ENAME      JOB              SAL
---------- --------- ---------
MILLER     CLERK           1300
```

L'utilisation de variables dans vos programmes SQL*Plus permet une meilleure lisibilité ainsi qu'une réutilisation du code.

Décrire la structure des tables et des vues

La possibilité de décrire les tables et les vues contenues dans la base de données est une option très intéressante de SQL*Plus. Son utilisation est simple : seules les tables et les vues auxquelles vous avez accès en lecture sont accessibles par DESC. Par exemple, pour connaître la structure de la table EMP, propriété de l'utilisateur SCOTT :

```
SQL> DESC scott.emp

 Name                          Null?    Type
 ----------------------------- -------- ----
 EMPNO                         NOT NULL NUMBER(4)
 ENAME                                  VARCHAR2(10)
 JOB                                    VARCHAR2(9)
 MGR                                    NUMBER(4)
 HIREDATE                               DATE
 SAL                                    NUMBER(7,2)
 COMM                                   NUMBER(7,2)
 DEPTNO                                 NUMBER(2)
```

Améliorer la présentation de vos résultats

SQL*Plus n'est pas un outil destiné à produire des éditions perfectionnées comme peuvent le faire des outils spécialisés. Pourtant, un certain nombre de commandes rend l'environnement de travail plus convivial. Il est possible de réaliser des éditions minimalistes avec cet outil.

Effectuer des additions sur des colonnes

Pour effectuer une addition sur une colonne, il faut définir le niveau de rupture souhaité et la colonne de la table sur laquelle s'effectue l'addition.

```
SQL>BREAK ON REPORT
SQL>COMPUTE SUM OF sal ON REPORT
SQL>
SQL>select ename, sal from scott.emp
  2  where deptno = 10
  3  /

ENAME            SAL
---------- ---------
CLARK           2450
KING            5000
MILLER          1300
           ---------
sum             8750

SQL>CLEAR BREAK
SQL>CLEAR COMPUTE
```

Afficher un titre avant vos ordres SQL

TTITLE est utilisé pour afficher un titre avant chaque ordre SQL. Il permet de centrer le titre (gauche, droite, milieu) et de jouer avec les polices de caractère (gras, italique...).

```
SQL> -- Le titre est placé à gauche (LEFT)
SQL> -- SKIP 1 saute une ligne
SQL> -- le caractère « - » en fin de ligne indique que TTITLE continue
SQL>
SQL>TTITLE LEFT '#####################################' skip 1 -
>   LEFT 'Employés et salaire'  skip 1 -
>   LEFT '#####################################'
SQL>
SQL> -- L'exécution de l'ordre SQL provoque l'affichage du titre
SQL>
SQL>select ename, sal from scott.emp
  2  where deptno = 10
  3  /

#####################################
Employés et salaire
#####################################

ENAME            SAL
---------- ---------
CLARK           2450
KING            5000
MILLER          1300

SQL> -- annule l'affichage d'un titre pour les prochains ordres SQL
SQL> TTITLE ''
```

Enregistrer dans un fichier texte

Il est possible d'enregistrer dans un fichier tout ce qui se passe à l'écran sous SQL*Plus.

```
SQL> SPOOL fichier_texte.lis
......
SQL> SPOOL OFF  -- force l'écriture dans le fichier spool.
```

L'éditeur *ED fichier_texte.lis* vous permet d'éditer le résultat obtenu.

Afficher un commentaire

La commande PROMPT permet d'afficher un texte lors de l'exécution de vos ordres SQL. Elle sert notamment à commenter le résultat lors de l'exécution d'un script SQL. Ainsi :

```
--
-- Fichier script SQL
--
PROMPT Début du script ...
...
```

donnera comme résultat :

```
Début du script ...
```

Notez la différence entre les commentaires destinés à améliorer la lisibilité du script et *PROMPT* qui sert à en commenter le résultat.

Afficher / cacher les ordres SQL avant leur exécution

Vous pouvez afficher/cacher le texte des ordres SQL avant leur exécution. Ainsi :

```
--
-- Fichier script SQL
--
SET ECHO ON

select ename, sal from scott.emp
where deptno = 10
/
...
```

affichera comme résultat :

```
SET ECHO ON
SQL>select ename, sal from scott.emp
  2  where deptno = 10
  3  /

ENAME            SAL
```

```
---------- ---------
CLARK           2450
KING            5000
MILLER          1300
```

L'utilisation de SET ECHO OFF cachera le texte de l'ordre SQL. Ainsi :

```
--
-- Fichier script SQL
--
SET ECHO OFF

select ename, sal from scott.emp
where deptno = 10
/
```

aura comme résultat :

```
SET ECHO OFF

ENAME            SAL
---------- ---------
CLARK           2450
KING            5000
MILLER          1300
```

Ne pas afficher le nombre d'enregistrements renvoyés

Par défaut, SQL*Plus affiche le nombre d'enregistrements renvoyés dès que ceux-ci sont supérieurs à cinq. Pour que cette information ne s'affiche plus, utilisez SET FEEDBACK OFF.

```
SQL> SET FEEDBACK ON
SQL> select ename from emp;

ENAME
----------
SMITH
MARTIN
CLARK
SCOTT
KING
TURNER
ADAMS
JAMES
FORD
MILLER

10 rows selected.

SQL> SET FEEDBACK OFF
SQL> select ename from emp;
```

```
ENAME
----------
SMITH
MARTIN
CLARK
SCOTT
KING
TURNER
ADAMS
JAMES
FORD
MILLER
SQL>
```

Les commandes de présentation SET

Un grand nombre de commandes *SET variable paramètres* est proposé par SQL*Plus. Nous en avons déjà rencontré quelques-unes :

```
SET ECHO [ON | OFF]
SET FEEDBACK [ON | OFF]
SET TIMING [ON | OFF]
...
```

Reportez-vous à la documentation SQL*Plus pour disposer de l'ensemble des commandes SET.

Création de requêtes dynamiques

Vous pouvez utiliser une option très puissante de SQL*Plus pour créer automatiquement des ordres SQL avec des données issues de la base Oracle. Vous souhaitez, par exemple, obtenir la description de l'ensemble des tables et des vues qui sont votre propriété.

La première étape consiste à créer un ordre SQL qui recherche la liste des vues et des tables. Ces données sont précédées et suivies de textes, qui serviront à créer une commande SQL complète encadrant les données issues de la base.

La seconde étape exécutera ces ordres SQL, préalablement conservés dans un fichier de commandes.

Première étape : fichier de commandes

Ce fichier déclenche automatiquement les autres étapes.

```
--
-- Création d'une requête SQL dynamique
--

-- Gestion de la présentation
SET ECHO OFF
SET FEEDBACK OFF
```

```
SET HEADING OFF

-- Création du fichier SQL qui sera exécuté
SPOOL fichier_temporaire.sql

-- Bannière destinée à figurer dans le fichier de commandes
PROMPT SET ECHO ON
PROMPT REM ################################
PROMPT REM Fichier créé automatiquement
PROMPT REM ################################
-- Fichier qui contiendra le resultat
PROMPT SPOOL resultat.lis
select 'DESC '||object_name from user_objects
where object_type in ('TABLE','VIEW')
/
-- arrêt de l'enregistrement du fichier résultat
PROMPT SPOOL OFF

spool off
START fichier_temporaire.sql
```

Deuxième étape : le contenu du fichier temporaire

Le fichier de commande *fichier_temporaire.sql* est créé automatiquement lors de l'exécution du fichier précédent. La liste des tables et des vues provient directement de la base Oracle 10*g*.

```
SET ECHO ON
REM ################################
REM Fichier créé automatiquement
REM ################################
SPOOL resultat.lis

DESC DEPT
DESC EMP
DESC TOTO
DESC TRACE_SALAIRE

SPOOL OFF
```

À la place de DESC NOM_TABLE, vous pouvez placer n'importe quel ordre SQL valide en plaçant à la fin le signe « ; ».

Troisième étape : exécuter le fichier de commandes temporaire

Cette étape est effectuée à la fin du script de commandes principal par :

```
START fichier_temporaire.sql
```

Le résultat du descriptif des tables est enregistré dans le fichier *resultat.lis* dont le contenu est le suivant :

```
SQL>
SQL> DESC BONUS
 Name                          Null?    Type
 ----------------------------- -------- ----
 ENAME                                  VARCHAR2(10)
 JOB                                    VARCHAR2(9)
 SAL                                    NUMBER
 COMM                                   NUMBER

SQL> DESC DEPT
 Name                          Null?    Type
 ----------------------------- -------- ----
 DEPTNO                        NOT NULL NUMBER(2)
 DNAME                                  VARCHAR2(14)
 LOC                                    VARCHAR2(13)

SQL> DESC EMP
 Name                          Null?    Type
 ----------------------------- -------- ----
 EMPNO                         NOT NULL NUMBER(4)
 ENAME                                  VARCHAR2(10)
 JOB                                    VARCHAR2(9)
 MGR                                    NUMBER(4)
 HIREDATE                               DATE
 SAL                                    NUMBER(7,2)
 COMM                                   NUMBER(7,2)
 DEPTNO                                 NUMBER(2)
NUMBER(2)
...

SQL> SPOOL OFF
```

Autre requête dynamique

Les exemples d'utilisation des requêtes dynamiques sont très nombreux. Ici, nous utilisons les requêtes dynamiques pour accomplir une tâche d'administration. Oracle 10*g* utilise des statistiques pour déterminer la manière dont son moteur SQL traite les requêtes qui lui sont soumises. Les statistiques doivent être mises à jour régulièrement. La commande :

```
ANALYZE [TABLE|INDEX] nom COMPUTE STATISTICS...
```

analyse tables et index et met à jour le dictionnaire interne d'Oracle. Rechercher et saisir l'ensemble des tables et index de la base de données, est une opération longue et fastidieuse. Or, le dictionnaire de données d'Oracle9 nous fournit la liste des tables et index contenus. D'où le programme SQL*Plus :

```
--
-- Création d'une requête SQL dynamique:
-- Analyse des tables et index
--

-- Modifications de l'environnement SQL*Plus
SET ECHO OFF
SET FEEDBACK OFF
SET HEADING OFF
SET PAGESIZE 100

-- Création du fichier SQL qui sera exécuté
SPOOL analyze.SQL

select 'analyze '||object_type||' '||object_name
     ||' estimate statistics sample 20 percent;'
from user_objects
where object_type in ('TABLE','INDEX')
order by object_type
/
spool off
-- Modifications de l'environnement SQL*Plus
SET ECHO ON
SET ECHO OFF
SET FEEDBACK ON
SET HEADING ON
SET PAGESIZE 20
-- Exécution de l'analyse de tables et des index
PROMPT ########################################
PROMPT Début de l'analyse des tables et index
PROMPT ########################################
START analyze.sql
PROMPT ###############
PROMPT Fin d'analyse
PROMPT ###############
-- On peut supprimer le fichier temporaire (ici en commentaire)
-- HOST rm analyze.sql
```

Le fichier analyze.sql contient :

```
analyze INDEX SYS_C001227 estimate statistics sample 20 percent;
analyze INDEX SYS_C0012271 estimate statistics sample 20 percent;
analyze INDEX SYS_C001229 estimate statistics sample 20 percent;
analyze TABLE BONUS estimate statistics sample 20 percent;
analyze TABLE DEPT estimate statistics sample 20 percent;
analyze TABLE TOTO estimate statistics sample 20 percent;
analyze TABLE TRACE_SALAIRE estimate statistics sample 20 percent;
analyze TABLE EMP estimate statistics sample 20 percent;
analyze TABLE MLOG$_EMP estimate statistics sample 20 percent;
analyze TABLE SALGRADE estimate statistics sample 20 percent;
analyze TABLE SNAP$_EMP_SNAPSHOT estimate statistics sample 20 percent;
```

Ce fichier est lancé automatiquement à l'issue du script principal.

> La commande *HOST rm analyse.sql* peut être utilisée à l'issue du script pour supprimer toute trace du fichier temporaire.

Les outils complémentaires

En complément de la présentation détaillée des outils standard fournis par Oracle, nous avons choisi de vous présenter d'autres outils qui répondent à des besoins spécifiques, à savoir :

- des besoins non satisfaits par les outils standard Oracle ;
- des offres innovatrices qui comblent des lacunes.

Le principal reproche que l'on puisse adresser aux outils standard d'Oracle est leur manque de convivialité et leur accès complexe. Ce sont des produits en mode caractère destinés à des informaticiens. Ils ont l'avantage d'être présents sur tout type d'environnement, d'être performants, stables et d'offrir des possibilités étendues. Nous sommes loin d'un univers « 100 % graphique », mais ils sont puissants et fiables.

Quelle complémentarité attendre d'Oracle Enterprise Manager ?

Le chapitre 25, *Oracle Enterprise Manager*, est consacré à ce sujet. Souvent, les outils incorporés dans Oracle Enterprise Manager proposent un habillage graphique plus convivial que celui des outils Oracle fournis en standard.

Les modules proposés par Oracle Enterprise Manager proposent toutes sortes de services. Ils administrent vos bases de données, permettent d'ordonnancer et de planifier des tâches sur les systèmes distants, d'arrêter et de démarrer une base, de l'administrer, de créer une table, un utilisateur, etc. C'est à la fois trop et pas assez. Trop, pour un utilisateur néophyte qui est perdu devant un ensemble d'écrans dont il ne comprend pas toujours le sens et l'utilisation (de plus, il est difficile de gérer des profils d'utilisateur). Pas assez, si vous êtes un utilisateur professionnel souhaitant disposer de l'ensemble des possibilités offertes par la base Oracle 10*g*. Par ailleurs, cédant à la mode du tout graphique, certains modules sont proches du gadget et n'ont pas de finalité précise.

Cependant, Oracle Enterprise Manager est livré gratuitement et il convient de l'étudier soigneusement avant d'avoir recours à l'achat d'autres produits. De même, il est nécessaire de le mettre en concurrence avec des outils spécialisés avant d'acquérir des modules supplémentaires (et facturés) d'Oracle Enterprise Manager.

Les limites d'Oracle Export/Import ?

Dans la majorité des cas, tant en termes de fonctionnalités que de performance, l'Export/Import est très satisfaisant. Si vous manipulez de très gros volumes de données, des éditeurs spécialisés comme ex-PLATINUM (acheté depuis par Computer Associates) proposent des outils complémentaires.

Les limites d'Oracle Data Pump ?

Oracle Data Pump Export et Import apportent de nouvelles possibilités aux traditionnels Export et Import. Ils sont annoncés comme une véritable avancée en terme de performance. Sur de petites configurations (serveur mono processeur, faible volume de données), l'utilisation de Data Pump peut s'avérer plus longue qu'Export ou Import. Sur de gros serveurs, avec une forte volumétrie de données (on parle alors en centaines de Go…) les gains seront certainement probants. Je vous conseille d'effecteur préalablement un test pour comparer les deux possibilités dans votre contexte.

Les limites d'Oracle SQL*Loader ?

De même que pour l'Export/Import, Oracle SQL*Loader est très satisfaisant, laissant peu d'opportunités à d'autres outils payants.

Quelles sont les limites d'Oracle SQL*Plus ?

Il faut dissocier l'usage de SQL*Plus pour créer, démarrer, arrêter une base et celui plus large de la « programmation SQL ». Dans le premier cas, SQL*Plus et Oracle Enterprise Manager sont les seuls outils répondant à cet usage. Pour programmer en SQL, le marché des outils tiers est florissant.

Oracle SQL*Plus est un outil de base indispensable, mais qui montre vite ses limites si vous le destinez à des programmeurs habitués à des environnements de développements graphiques performants. Oracle Enterprise Manager propose SQL*Worksheet, mais ce dernier n'est pas à la hauteur de ses concurrents. Parmi les éditeurs fournissant des outils de substitution, axés sur la gestion et la programmation d'une base Oracle, nous pouvons citer :

- Oracle, qui propose ses propres outils de développement,
- Platinum Technology, *http://www.cai.com*,
- Quests Software, *http://www.quests.com* qui a acheté l'excellent TOAD,
- Tora, *http://www.globecom.se/tora* un très bon outil qui existe sous Windows et Linux,
- DataBee, *http://www.databee.com*, un outil très bien conçu autour des opérations d'export et d'import,
- SQL-Programmer édité par BMC, *http://www.bmc.com*,
- SQL*Object Builder de IDB, *http://www.idb-consulting.fr*.

Quels autres outils pour quels besoins ?

Pour découvrir le marché des outils tiers gravitant autour d'Oracle, la meilleure source d'informations sont les publicités qui paraissent dans *Oracle magazine*, magazine papier bimensuel à abonnement gratuit. Le formulaire d'abonnement est accessible à l'adresse *http://www.oramag.com*.

C'est un marché de « niche », fortement spécialisé, où des éditeurs proposent des produits très techniques à forte valeur ajoutée. La palette proposée est large et comprend notamment :

- des outils de conversion automatique d'applications Oracle*Forms vers Oracle Developer 2000 ;
- des outils complémentaires opérant autour d'Oracle Application ;
- des outils de réorganisation de base de données ;
- des utilitaires pour automatiser la sécurité d'accès.

Résumé

Oracle 10*g* est fourni avec un ensemble d'utilitaires puissants, trop souvent méconnus. Ce chapitre, volontairement détaillé, a permis de vous en présenter les champs d'application et d'utilisation ainsi que leurs limites :

- Oracle SQL*Loader pour charger toutes sortes de fichiers dans une base Oracle ;
- Oracle Export/Import pour extraire et transférer des données de base Oracle à base Oracle ;
- Oracle SQL*Plus, l'interface privilégiée pour interagir sur la base de données et pour la création, le démarrage et l'arrêt des bases ;
- des outils tiers, complémentaires aux précédents.