Oracle11*g*

Administration

Razvan Bizoï

Tsoft EDITEUR

EYROLLES

Avant-propos

Oracle est le système de base de données le plus utilisé au monde. Il fonctionne de façon relativement identique sur tout type d'ordinateur. Les connaissances acquises sur une plate-forme sont donc utilisables sur une autre ; les utilisateurs ou développeurs Oracle expérimentés sont des professionnels très demandés.

Pour une bonne compréhension de l'ouvrage, il est souhaitable que le lecteur ait une connaissance suffisante du modèle relationnel et qu'il maîtrise les langages de programmation SQL et PL/SQL.

Conçu sous forme d'un guide de formation, il vous permettra d'acquérir des connaissances solides sur les tâches fondamentales liées à l'administration des bases de données : concevoir, créer et gérer une base de données Oracle 11*g*.

Cet ouvrage est complété par un autre ouvrage du même auteur chez le même éditeur, *Oracle 11g - Sauvegarde et restauration*. Ces deux ouvrages peuvent vous préparer aux examens de certification Oracle :

- 1Z0-052 : Oracle Database 11*g* Administration I.
- 1Z0-053 : Oracle Database 11*g* Administration II.

Ce guide de formation vise surtout à être plus clair et plus agréable à lire que les documentations techniques, exhaustives et nécessaires, mais ingrates, dans lesquelles vous pourrez toujours vous plonger ultérieurement. Par ailleurs, l'auteur a aussi voulu éviter de ne fournir qu'une collection supplémentaire de « trucs et astuces », mais plutôt expliquer les concepts et les mécanismes avant d'indiquer les procédures pratiques.

Dans la mesure où l'on dispose du matériel informatique nécessaire, il est important de réaliser les travaux pratiques qui sont indispensables à l'acquisition d'une compétence réelle, et qui permettent de comprendre réellement la manière dont le système fonctionne.

En fin de chaque module un QCM est proposé, pour vérifier certaines des connaissances acquises. Vous pourrez télécharger les réponses aux QCM depuis le site de l'éditeur www.editions-eyrolles.com : dans la zone <Recherche> saisissez **G12898** et validez par <Entrée>.

Bon courage !

Table des matières

- *Instance et base de données*
- *CREATE DATABASE*
- *Les scripts de création*

6

La création d'une base de données

Objectifs

À la fin de ce module, vous serez à même d'effectuer les tâches suivantes :

- Décrire les étapes de création de la base de données.
- Préparer le système d'exploitation pour l'installation de la base de données.
- Créer une base de données manuellement.
- Changer le nom de l'instance d'une base de données.
- Décrire les fichiers nécessaires pour effectuer une sauvegarde.

Contenu

La base de données

Comme nous avons pu le voir précédemment, la base de données est l'ensemble des trois types de fichiers obligatoires : les fichiers de contrôle, les fichiers de données et les fichiers des journaux.

Les fichiers de la base de données sont des fichiers binaires ne pouvant être lus ou écrits directement.

Les fichiers de données contiennent toutes les informations de votre base dans un format spécifique à Oracle. Il n'est pas possible d'en visualiser le contenu avec un éditeur de texte.

Les fichiers de contrôle sont des fichiers binaires contenant des informations sur tous les autres fichiers constitutifs d'Oracle. Ils décrivent leur nom, leur emplacement et leur taille.

Les fichiers journaux sont des fichiers conservant toutes les modifications successives de votre base de données. L'activité des sessions qui interagissent avec Oracle est consignée en détail dans les fichiers journaux. Il s'agit en quelque sorte des journaux de transactions de la base, une transaction étant une unité de travail soumise au système pour traitement.

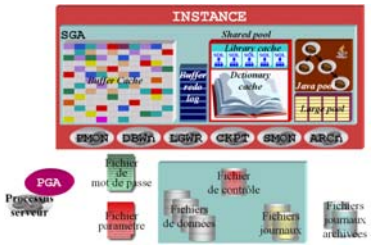
Le fichier des paramètres contient les paramètres de démarrage de la base et d'autres valeurs qui déterminent l'environnement dans lequel la base s'exécute. Lorsqu'elle est démarrée, le fichier des paramètres est lu et plusieurs structures mémoire sont allouées en fonction de son contenu.

Le fichier de mots de passe est utilisé pour établir l'authenticité des utilisateurs privilégiés de la base de données.

Vous pouvez utiliser deux modalités de création de la base de données :

- Manuellement à l'aide de l'instruction SQL « **`ALTER DATABASE`** ».
- Avec l'assistant de configuration de base des données (DBCA **D**ata**B**ase **C**onfiguration **A**ssistant).

La création manuelle



- **Choisissez le nom de l'instance.**
- **Choisissez le nom de la base des données.**
- **Préparez le système d'exploitation.**
- **Créez le fichier des paramètres « PFILE ».**
- **Créez le fichier des paramètres à serveur « SPFILE ».**
- **Démarrez votre instance.**
- **Créez la base de données.**
- **Exécutez les scripts de création du dictionnaire de données.**
- **Exécutez les scripts de création de votre schéma relationnel.**
- **Sauvegardez votre base de données.**

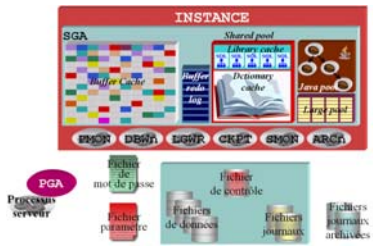
La création de la base est la première étape dans la gestion et l'organisation d'un système de base de données.

La création d'une base de données est une tâche consistant à préparer plusieurs fichiers du système d'exploitation, qu'il n'est nécessaire d'effectuer qu'une fois, quel que soit le nombre de fichiers de données de la base. Il s'agit d'une tâche très importante, l'administrateur de la base de données devant déterminer des paramètres de la base, tels que le nom de la base ou la taille du bloc, qui ne peuvent plus être modifiés après la création.

Vous pouvez créer une base en utilisant de nouveaux fichiers de données ou en effaçant les informations d'une base existante ayant la même structure physique.

Pour créer une base de données manuellement, il faut d'abord préparer et planifier la création de cette base des données.

La création manuelle



- **Choisissez le nom de la base des données.**
- **Choisissez le nom du domaine.**
- **Choisissez le nom unique de la base des données.**
- **Choisissez le nom de l'instance.**

La création de la base est la première étape dans la gestion et l'organisation d'un système de base de données ; il faut bien choisir l'identification de la base de données dans les serveurs sur lesquels on l'installe, mais également dans le système d'information complet de votre entreprise.

Le nom de la base de données

Le nom de la base de données est stocké dans le paramètre « **DB_NAME** » ; il ne doit pas dépasser huit caractères.

Le nom de la base de données est également utilisé dans la commande SQL « **CREATE DATABASE** » ; ainsi il est stocké directement dans le fichier des contrôles. Il n'est plus possible de modifier le nom de la base de données après sa création.

Le nom du domaine

Le nom du domaine « **DB_DOMAIN** » spécifie le domaine de réseau où la base de données est créée. Si la base de données que vous vous apprêtez à créer devait faire partie d'un système de bases de données distribuées, accordez une attention particulière à ce paramètre d'initialisation de la base de données avant la création. Il permet de définir avec un nom unique de la base de données dans votre domaine qui est :

« **DB_NAME** ».« **DB_DOMAIN** »

Le nom unique de la base de données

Dans le cas où plusieurs bases de données ont le même nom à l'intérieur de votre domaine, vous devez préciser le paramètre « **DB_UNIQUE_NAME** » qui représente le nom unique de la base de données dans votre domaine. Le nom unique de la base de données peut atteindre jusqu'à 30 caractères.

Le nom de l'instance

Le nom de l'instance est stocké dans le paramètre « **INSTANCE_NAME** », un paramètre facultatif depuis la version Oracle 10*g*.

Il est préférable de choisir un nom d'instance et un nom de base de données identiques.

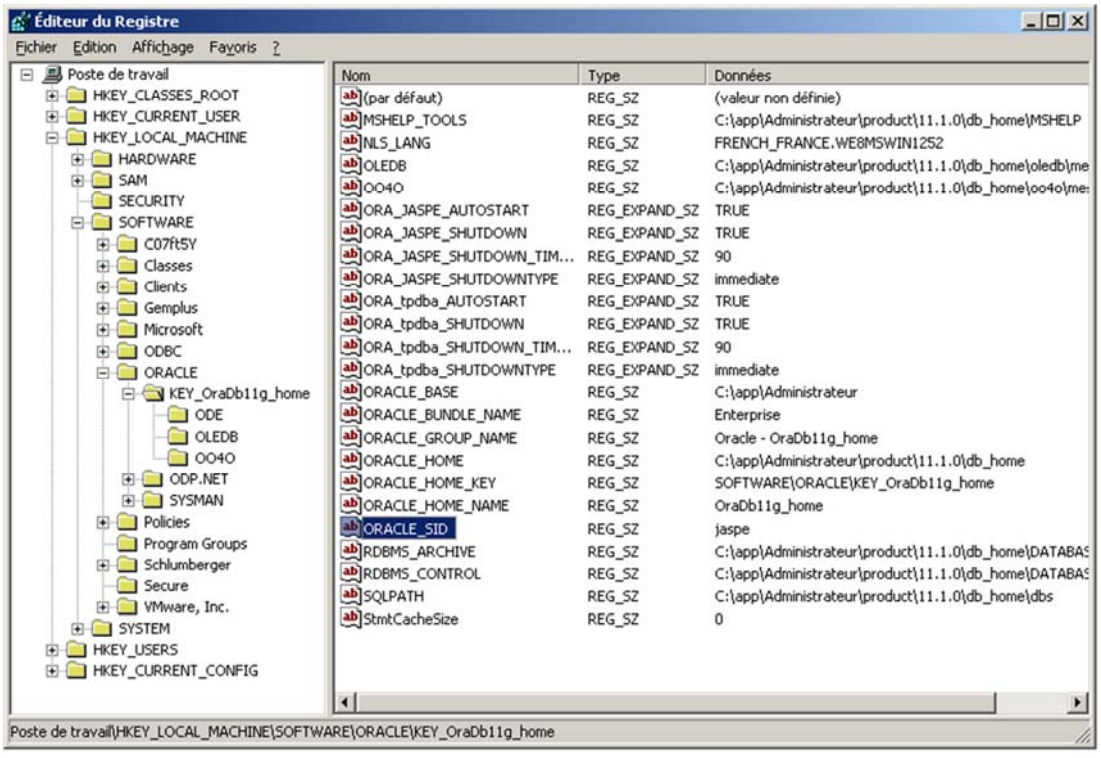
Attention

Pour vous connecter à la base de données à l'aide de SQL*Plus, vous avez deux solutions :

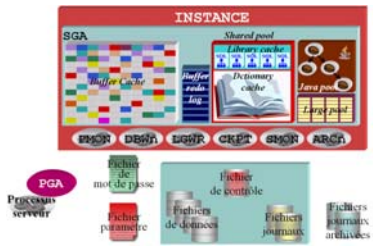
- La connexion à l'aide du réseau Oracle Net décrit dans le module 10.
- La deuxième solution est possible uniquement si vous vous connectez directement à partir du serveur. Il s'agit d'une connexion à l'instance par défaut du serveur. L'instance par défaut du serveur est définie par la variable d'environnement « **ORACLE_SID** ».

Par conséquent, il est impératif de définir cette variable avant de se connecter à SQL*Plus pour pouvoir créer la base de données.

Dans l'environnement Windows, la variable est configurée au sein de la base de registres dans « **HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\KEY_*** ». Vous pouvez la modifier directement ou utiliser une variable d'environnement dans votre environnement de commandes.



La configuration du système



- **L'arborescence de répertoires**
- **La méthode d'authentification**
- **La variable d'environnement « ORACLE_SID »**
- **La création du service**

L'arborescence de répertoires

Rappelez-vous que, pour chaque base de données, vous devez créer une arborescence de répertoires pour les fichiers d'administration et une deuxième arborescence de répertoires pour l'ensemble des fichiers de la base de données.

Le répertoire pour stocker les fichiers d'administration se trouve dans le répertoire suivant :

`$ORACLE_BASE/admin/DB_NAME` ou `%ORACLE_BASE%\admin\DB_NAME`

```
C:\> mkdir C:\app\oracle\admin\tpdba\adump

C:\> mkdir C:\app\oracle\admin\tpdba\dpdump

C:\> mkdir C:\app\oracle\admin\tpdba\pfile

C:\> mkdir C:\app\oracle\cfgtoollogs\dbca\tpdba
```

La méthode d'authentification

Vous devez choisir la méthode que vous souhaitez utiliser pour l'authentification de l'utilisateur administrateur de votre base de données. Il doit disposer de la totalité des privilèges relatifs au système d'exploitation ou utiliser l'authentification par fichier mot de passe. Pour plus d'informations voir le module 9.

La création du service

Dans l'environnement Windows, l'instance ne peut fonctionner que s'il existe un environnement pour faire tourner les processus d'arrière-plan et partager les hommes mémoire.

L'environnement nécessaire est un service Windows. Oracle fournit un utilitaire pour la gestion des différents services Windows.

L'utilitaire a la syntaxe suivante :

```
ORADIM -{NEW | DELETE | EDIT }
       -SID SID
            [-INTPWD mot-de-passe] [-MAXUSERS nombre]
            [-PFILE fichier]
            [-STARTMODE a|m]
            [-SRVCSTART system | demand]
            [-TIMEOUT secs]
```

NEW	L'argument permet la création d'un service Windows.
DELETE	L'argument permet l'effacement d'un service Windows.
EDIT	L'argument permet la modification des paramètres d'un service Windows, à savoir passer du mode de démarrage du service automatique en mode de démarrage manuel et l'inverse.
SID	L'argument permet de préciser le nom de l'instance.
INTPWD	L'argument permet d'initialiser un mot de passe qui sera stocké dans le fichier de mots de passe. En effet « **ORADIM** » vous permet de créer également un fichier de mots de passe positionné dans le répertoire par défaut de la base de données.
MAXUSERS	Le nombre maximum d'utilisateurs gérés par le fichier de mots de passe.
STARTMODE	L'argument détermine si au démarrage de services Windows l'instance est démarrée automatiquement ou manuellement. Si vous utilisez l'option manuelle, vous devez démarrer manuellement votre instance à l'aide de la commande SQL*Plus « **STARTUP** ».
SRVCSTART	L'argument vous permet de définir si le service Windows démarre automatiquement ou manuellement après un redémarrage du serveur.
PFILE	Le nom du fichier des paramètres par défaut utilisés pour démarrer par défaut l'instance.
TIMEOUT	Le temps d'attente maximum, en secondes, pour un service Windows avant l'arrêt final.

Nous allons considérer à présent un exemple mettant en œuvre la création d'un service Windows à l'aide de l'utilitaire « **ORADIM** ». Pour pouvoir utiliser l'environnement de l'instance, il faut d'abord initialiser une variable d'environnement déjà rencontrée auparavant qui s'appelle « **ORACLE_SID** ».

```
C:\>set ORACLE_SID=tpdba

C:\>sqlplus /nolog
SQL>connect / as sysdba
ERROR: ORA-12560: TNS : erreur d'adaptateur de protocole

SQL>exit

C:\>oradim -NEW -SID tpdba -INTPWD password -STARTMODE a
```

```
Instance créée.

C:\>sqlplus / AS SYSDBA
Connecté à une instance inactive.
SQL> $ dir
C:\app\oracle\product\11.2.0\db_home\database\PWDtpdba.ORA
...
 Répertoire de C:\app\oracle\product\11.2.0\db_home\database

16/07/2009  08:21             2 560 PWDtpdba.ORA
```

Après l'initialisation de la variable d'environnement, on se connecte à SQL*Plus sans définir les informations de connexion à la base de données. La connexion à l'instance par défaut « **tpdba** » définie par la variable d'environnement « **ORACLE_SID** » ne peut pas être effectuée car le service Windows n'a pas encore été défini.

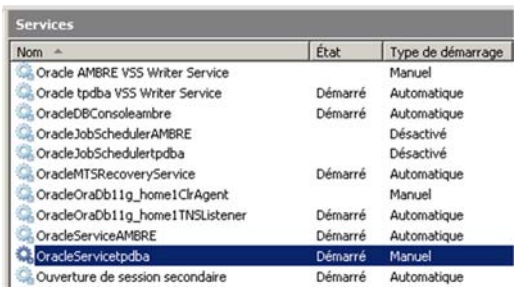
On crée le service Windows pour l'instance appelée « **tpdba** » et en même temps un fichier des mots de passe « **pwdtpdba.ora** » avec le mot de passe « **password** » pour les administrateurs de la base des données. La dernière commande de l'exemple vous permet de visualiser l'emplacement du fichier de mots de passe.

Vous pouvez remarquer que, dès lors que service Windows est créé, il est possible de se connecter normalement.

Attention

Il faut faire la différence entre le service Windows et l'instance de la base de données. Vous pouvez avoir le cas où le service Windows est démarré mais où l'instance n'est pas démarrée pour autant. Dans ce cas vous pouvez démarrer manuellement votre instance.

Toutefois, si le service Windows n'est pas démarré, vous ne pouvez pas accéder à l'instance.



Le service ainsi créé n'est pas démarré automatiquement au démarrage de la machine. Pour que le service démarre automatiquement, il faut exécuter la commande suivante :

```
C:\>oradim -EDIT -SID tpdba -SRVCSTART system
```

Dans l'exemple suivant, vous est présentée la commande pour effacer un service Windows. L'argument « **DELETE** » ne requiert pas d'autres informations que le nom de l'instance.

```
C:\> oradim -DELETE -SID tpdba
Instance supprimée.

C:\>sqlplus /nolog
SQL>connect / as sysdba
ERROR: ORA-12560: TNS : erreur d'adaptateur de protocole
```

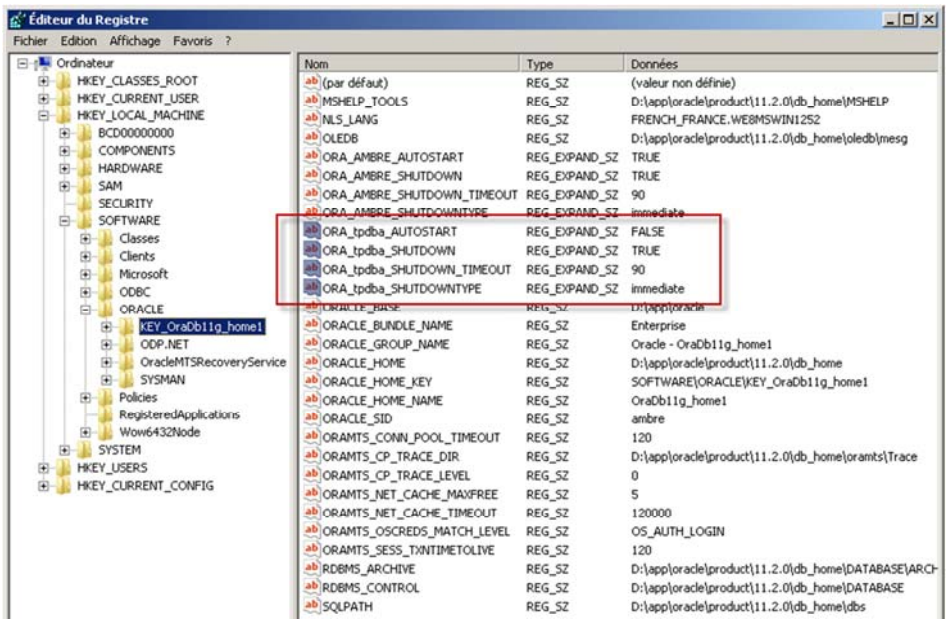
Chaque fois que vous créez un service, quatre clés sont créées qui déterminent le mode de fonctionnement de ce service :

ORA_*_AUTOSTART L'argument détermine si au démarrage du service Windows l'instance est démarrée automatiquement.

ORA_*_SHUTDOWN L'argument détermine si à l'arrêt du service Windows l'instance est arrêtée automatiquement.

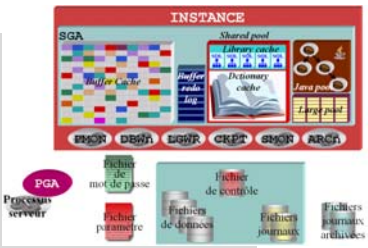
ORA_*_SHUTDOWN_TIMEOUT L'argument définit le temps d'attente maximum en secondes avant que le service ne s'arrête.

ORA_*_SHUTDOWNTYPE L'argument définit le type d'arrêt de l'instance.



Le fichier paramètres

```
compatible=11.2.0.0.0
db_name=tpdba
db_block_size=8192
memory_target=857735168
open_cursors=300
processes=150
undo_tablespace=UNDOTBS1
db_create_file_dest=C:\app\oracle\oradata
db_create_online_log_dest_1=C:\app\oracle\oradata
db_recovery_file_dest=C:\app\oracle\flash_recovery_area
db_recovery_file_dest_size=21474836480
diagnostic_dest=C:\app\oracle
audit_file_dest=C:\app\oracle\admin\tpdba\adump
audit_trail=db
remote_login_passwordfile=EXCLUSIVE
nls_language='FRENCH'
nls_territory='France'
```



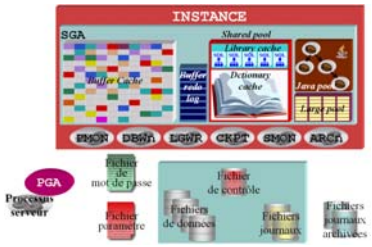
Nous allons créer le fichier paramètres utilisant les différents paramètres nécessaires pour le démarrage et le bon fonctionnement de votre base de données.

Le fichier paramètres SPFILE

Vous devez vous connecter à l'instance et, à l'aide de la commande SQL, « **CREATE SPFILE** », créer le fichier des paramètres serveur.

```
C:\>set ORACLE_SID=tpdba

C:\>sqlplus / as sysdba

Connecté à une instance inactive.

SQL> create spfile from pfile=
  2  'C:\app\oracle\admin\tpdba\pfile\inittpdba.ora';

Fichier créé.

SQL> startup nomount
Instance ORACLE lancée.

Total System Global Area  535662592 bytes
Fixed Size                  1334380 bytes
Variable Size             167773076 bytes
Database Buffers          360710144 bytes
Redo Buffers                5844992 bytes
```

La création de la base




```
CREATE DATABASE tpdba
         MAXINSTANCES 8
         MAXLOGHISTORY 1
         MAXLOGFILES 56
         MAXLOGMEMBERS 3
         MAXDATAFILES 1024
DATAFILE SIZE 300M AUTOEXTEND ON NEXT 10M EXTENT MANAGEMENT LOCAL
SYSAUX DATAFILE SIZE 120M AUTOEXTEND ON NEXT  10M
DEFAULT TEMPORARY TABLESPACE TEMP TEMPFILE SIZE 20M AUTOEXTEND ON NEXT 640K
UNDO TABLESPACE UNDOTBS1 DATAFILE SIZE 200M AUTOEXTEND ON NEXT 5M
CHARACTER SET WE8MSWIN1252 NATIONAL CHARACTER SET AL16UTF16
SET DEFAULT BIGFILE TABLESPACE
LOGFILE GROUP 1 SIZE 50M,GROUP 2 SIZE 50M,GROUP 3 SIZE 50M
USER SYS IDENTIFIED BY &&sysPassword
USER SYSTEM IDENTIFIED BY &&systemPassword;
```

Les fichiers de la base de données sont des fichiers binaires qui ne peuvent pas être lus par d'autres programmes que ceux des processus d'arrière-plan de l'instance.

La création de ces fichiers respecte la même règle, à savoir que l'on a besoin de l'instance pour pouvoir travailler avec les fichiers de données.

La création des fichiers de la base de données est effectuée à partir du SQL*Plus utilisant la commande suivante :

```
CREATE DATABASE nom
          MAXLOGFILES val_int
          MAXLOGMEMBERS val_int
          MAXLOGHISTORY val_int
          MAXDATAFILES valr_int
          MAXINSTANCES val_int
          CONTROLFILE REUSE
          LOGFILE GROUP nom_groupe [fichier] [,...]
          { ARCHIVELOG | NOARCHIVELOG }
DATAFILE [fichier
           [AUTOEXTEND
             {
               OFF |
               ON [NEXT val_int [{K|M|G}]]
                  [MAXSIZE{UNLIMITED | val_int [{K|M|G}]}
             }
           ]
         ] [,...]
SYSAUX DATAFILE
```

```
        [fichier
           [AUTOEXTEND
             {
               OFF |
               ON [NEXT val_int [{K|M|G}]]
                  [MAXSIZE{UNLIMITED | val_int [{K|M|G}]}
             }
           ]
         ] [,...]
UNDO TABLESPACE nom_tablespace
      [DATAFILE
        [fichier
          [AUTOEXTEND
            {
              OFF |
              ON [NEXT val_int [{K|M|G}]]
                 [MAXSIZE{UNLIMITED | val_int [{K|M|G}]}
            }
          ]
        ] [,...]
      ]
DEFAULT TEMPORARY TABLESPACE nom_tablespace
      [TEMPFILE
        [fichier
          [EXTENT MANAGEMENT LOCAL]
          [UNIFORM [SIZE val_int [{K|M|G}]]]
        ] [,...]
      ]
CHARACTER SET val_charset
NATIONAL CHARACTER SET val_charset
SET TIME_ZONE = time_zone
SET DEFAULT [BIGFILE | SMALLFILE] TABLESPACE;
```

MAXLOGFILES Spécifie le nombre maximal de groupes de fichiers redo log en ligne qui peuvent être créés dans la base. Oracle utilise cette valeur pour déterminer la quantité d'espace à allouer au nom des fichiers redo log dans le fichier de contrôle.

MAXLOGMEMBERS Spécifie le nombre maximal de membres, ou de copies identiques, pour un groupe de fichiers redo log. Oracle utilise cette valeur pour déterminer la quantité d'espace à allouer au nom des fichiers redo log dans le fichier de contrôle.

MAXLOGHISTORY	Spécifie le nombre maximal de fichiers redo log archivés pour la récupération de média automatique avec l'option Real Application Clusters.
MAXDATAFILES	Spécifie le dimensionnement initial de la section des fichiers de données dans le fichier de contrôle au moment de l'exécution.
MAXINSTANCES	Spécifie le nombre maximal d'instances qui peuvent simultanément monter et ouvrir la base de données.
CONTROLFILE REUSE	L'argument vous permet de réutiliser les fichiers de contrôle existants identifiés par le paramètre « **CONTROL FILES** ».
LOGFILE	Spécifie un ou plusieurs fichiers à utiliser en tant que fichiers redo log.
GROUP nom_groupe	Identifie de façon unique un groupe de fichiers redo log. Une base de données doit disposer d'au moins deux groupes de fichiers redo log.
ARCHIVELOG	Spécifie que le contenu d'un groupe de fichiers redo log doit être archivé avant d'être réutilisé.
NOARCHIVELOG	Spécifie que le contenu d'un groupe de fichiers redo log ne doit pas être archivé avant d'être réutilisé.
DATAFILE	Spécifie un ou plusieurs fichiers à utiliser en tant que fichier de données. Tous ces fichiers font partie du tablespace « **SYSTEM** ».
SYSAUX	Spécifie un ou plusieurs fichiers à utiliser en tant que fichier de données. Tous ces fichiers font partie du tablespace « **SYSAUX** ».
AUTOEXTEND	L'argument active ou désactive l'extension automatique d'un nouveau fichier de données ou d'un fichier temporaire. Si vous omettez cette clause, ces fichiers ne seront pas automatiquement étendus.
NEXT val_int	Spécifie la taille, en octets, de l'incrément d'espace disque suivant à allouer automatiquement.
K	Valeurs spécifiées pour préciser la taille en kilo-octets.
M	Valeurs spécifiées pour préciser la taille en mégaoctets.
G	Valeurs spécifiées pour préciser la taille en giga-octets.
MAXSIZE	L'argument spécifie l'espace disque maximal autorisé pour l'extension automatique du fichier.
UNLIMITED	Définit une allocation d'espace illimitée pour le fichier.
DEFAULT TEMPORARY	L'argument définit le tablespace temporaire par défaut pour la base de données. Si vous omettez cette clause, le tablespace « **SYSTEM** » servira de tablespace temporaire par défaut.
MANAGEMENT LOCAL	Indique qu'une partie du tablespace est réservée pour un bitmap de gestion des extents.
UNIFORM	Tous les extents d'un tablespace temporaire sont de la même taille.

UNDO TABLESPACE L'argument vous permet de définir le tablespace qui servira à stocker les données undo.

CHARACTER SET Spécifie le jeu de caractères utilisé par la base de données pour stocker les données. Les jeux pris en charge et la valeur par défaut de ce paramètre dépendent de votre système d'exploitation.

NATIONAL Spécifie le jeu de caractères national utilisé pour stocker les données dans des colonnes définies spécifiquement avec les types « **NCHAR** », « **NCLOB** » ou « **NVARCHAR2** ».

SET TIME_ZONE L'argument permet de définir la zone horaire de la base de données.

BIGFILE Indique que par défaut tous les tablespaces sont de type « **BIGFILE** ». Le tablespace est créé avec un seul fichier pouvant contenir jusqu'à 2^{32} blocs. Voir le module 15 pour plus d'informations sur les tablespaces.

SMALLFILE Indique que par défaut tous les tablespaces sont de type « **SMALLFILE** ». Le tablespace peut avoir plusieurs fichiers, chacun pouvant contenir jusqu'à 2^{22} blocs.

L'exemple suivant récapitule l'ensemble des commandes SQL et SQL*Plus jusqu'à la création de la base de données.

```
C:\>set ORACLE_SID=tpdba
C:\>sqlplus / as sysdba
Connecté à une instance inactive.

SQL> create spfile from pfile=
  2  'C:\app\oracle\admin\tpdba\pfile\inittpdba.ora';

Fichier créé.

SQL> startup nomount
Instance ORACLE lancée.

Total System Global Area  535662592 bytes
Fixed Size                  1334380 bytes
Variable Size             167773076 bytes
Database Buffers          360710144 bytes
Redo Buffers                5844992 bytes

SQL> CREATE DATABASE tpdba
  2     MAXINSTANCES 8 MAXLOGHISTORY 1
  3     MAXLOGFILES 56 MAXLOGMEMBERS 3
  4     MAXDATAFILES 1024
  5  DATAFILE SIZE 300M AUTOEXTEND ON NEXT 10M
  6           MAXSIZE UNLIMITED EXTENT MANAGEMENT LOCAL
  7  SYSAUX DATAFILE SIZE 120M AUTOEXTEND ON
  8           NEXT  10M MAXSIZE UNLIMITED
  9  DEFAULT TEMPORARY TABLESPACE TEMP TEMPFILE SIZE 20M
 10           AUTOEXTEND ON NEXT  640K MAXSIZE UNLIMITED
 11  UNDO TABLESPACE UNDOTBS1 DATAFILE SIZE 200M
 12           AUTOEXTEND ON NEXT  5M MAXSIZE UNLIMITED
 13  CHARACTER SET WE8MSWIN1252 NATIONAL CHARACTER SET AL16UTF16
```

```
 14  SET DEFAULT BIGFILE TABLESPACE
 15  LOGFILE GROUP 1 SIZE 50M,
 16          GROUP 2 SIZE 50M,
 17          GROUP 3 SIZE 50M
 18  USER SYS    IDENTIFIED BY &&sysPassword
 19  USER SYSTEM IDENTIFIED BY &&systemPassword;
Entrez une valeur pour syspassword : sys
ancien  18 : USER SYS IDENTIFIED BY &&sysPassword
nouveau  18 : USER SYS IDENTIFIED BY sys
Entrez une valeur pour systempassword : sys
ancien  19 : USER SYSTEM IDENTIFIED BY &&systemPassword
nouveau  19 : USER SYSTEM IDENTIFIED BY sys

Base de données créée.
SQL> select name, open_mode from v$database;

NAME      OPEN_MODE
--------- ----------
TPDBA     READ WRITE

SQL> CREATE TABLESPACE USERS DATAFILE SIZE 5M AUTOEXTEND ON;

Tablespace créé.

SQL> ALTER DATABASE DEFAULT TABLESPACE USERS;

Base de données modifiée.
```

L'ordre SQL de création de la base de données comporte également, comme vous pouvez le voir, les ordres des modifications des mots de passe pour les utilisateurs **SYS** et **SYSTEME**.

Vous pouvez remarquer que l'ordre de création de la base de données a créé la base, mais également ouvert cette base de données.

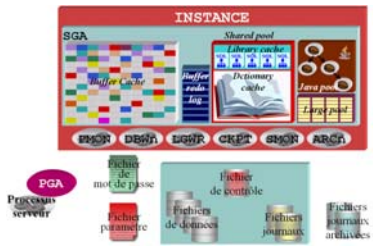
Pour plus d'informations concernant l'emplacement des fichiers de la base de données, voir le module 7.

Nous avons à présent une base de données, mais elle est une coquille vide. Pour pouvoir utiliser cette base de données, il faut, maintenant, passer à l'étape suivante : la création du dictionnaire de données.

La création du dictionnaire

Scripts :

- **catalog.sql**
- **catproc.sql**
- **catclust.sql**
- **cat*.sql**
- **dbms*.sql**
- ...



Répertoire :

$ORACLE_HOME/rdbms

ou

%ORACLE_HOME%\rdbms

Le dictionnaire de données est un élément important d'une base de données Oracle. Le dictionnaire permet de répertorier tous les objets créés dans la base et leur définition.

Le dictionnaire de données est un ensemble de tables et de vues mis à jour par Oracle. Ce dictionnaire contient toutes les informations sur les données de la base, sur les utilisateurs et l'espace physique.

Après la création de la base de données, il faut exécuter les scripts suivants dans le compte SYS.

Nom du script	*Description*
`catalog.sql`	Création des vues sur les tables de base du dictionnaire. Création des vues de performances dynamiques. Création des synonymes.
`catproc.sql`	Ce script permet d'implémenter les fonctionnalités du langage PL/SQL. Plusieurs paquetages sont créés pour étendre les fonctionnalités du noyau du SGBD Oracle.
`catblock.sql`	Crée les vues qui affichent dynamiquement les verrous.
`caths.sql`	Installe les paquetages pour l'administration des services hétérogènes.
`catio.sql`	Crée les vues du dictionnaire de données pour pouvoir tracer les entrées et les sorties table par table.
`caths.sql`	Installe les paquetages pour l'administration des services hétérogènes.

catqueue.sql	Crée les vues du dictionnaire de données pour Advanced Queuing.
catrep.sql	Crée les outils nécessaires pour la réplication de la base de données.
catrman.sql	Crée les vues du dictionnaire de données nécessaires pour l'utilitaire Recovery Manager (RMAN).
catoctk.sql	Crée les outils nécessaires pour Oracle Cryptographic Toolkit.

Les deux premiers scripts sont les scripts de création du catalogue obligatoire, mais, comme vous pouvez le remarquer, les autres scripts sont également très utiles pour le fonctionnement de la base de données. Dans le tableau précédent, seule a été présentée une sélection des scripts nécessaires pour la création de la base de données.

Tous les scripts de création du catalogue ainsi que les scripts utilisés pour l'administration étendue se retrouvent dans le répertoire :

```
$ORACLE_HOME/rdbms
```

ou

```
%ORACLE_HOME%\rdbms
```

Pour gérer une base, il peut être nécessaire au DBA de créer des structures supplémentaires, telles que des tables, des vues et des paquetages. Les scripts administratifs sont séparés en quatre catégories de fichiers se trouvant dans ce répertoire.

Convention	*Description*
cat*.sql	Ils créent des vues du dictionnaire de données et des tables de base du dictionnaire de données.
utl*.sql	Ils créent des vues et des tables additionnelles pour les utilitaires de la base de données.
dbms*.sql	Ils créent des spécifications de paquetage de base de données.
prvt*.plb	Ils créent le corps de paquetage.

```
SQL> connect / as sysdba
Connecté.

SQL> select name, open_mode from v$database;

NAME      OPEN_MODE
--------- ----------
TPDBA     READ WRITE

SQL> spool C:\oracle\admin\tpdba\CreateDBCatalog.log
SQL> @ ?\rdbms\admin\catalog.sql;
...
SQL> @ ?\rdbms\admin\catblock.sql;
...
SQL> @ ?\rdbms\admin\catproc.sql;
...
```

```
SQL> @ ?\rdbms\admin\catoctk.sql;
...
SQL> @ ?\rdbms\admin\owminst.plb;
...
```

Dans l'exemple précédent on commence par la connexion à la base de données, connexion avec les privilèges étendus « **SYSDBA** ».

Les scripts de création du dictionnaire de données sont relativement longs ; il faut par conséquent prendre la précaution que toutes les informations affichées puissent être lues par la suite. C'est pourquoi on ouvre un fichier spool dans le répertoire correspondant à l'architecture OFA. Sans le fichier spool il n'est pas possible de vérifier l'exécution de ces scripts.

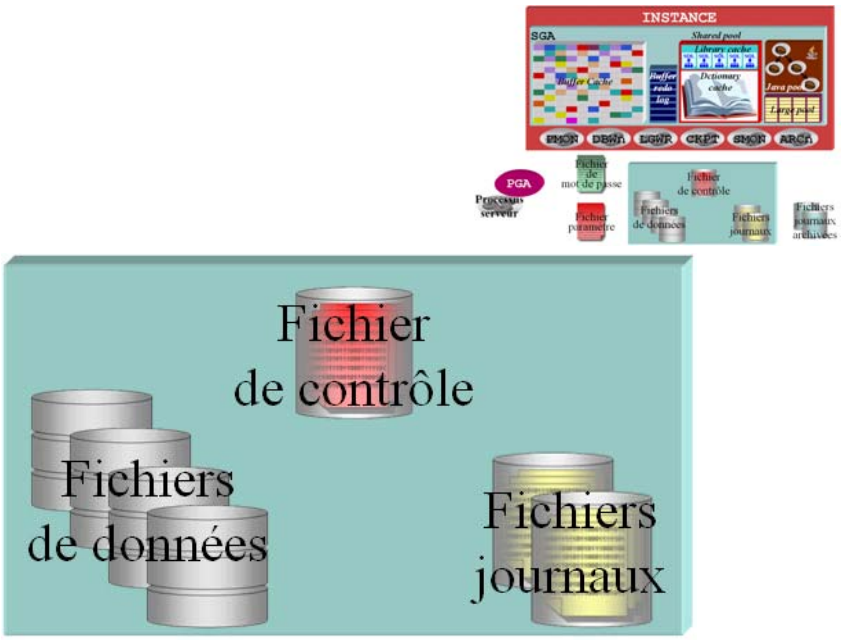
Les scripts peuvent retourner des messages d'erreur dus aux effacements des objets qui n'existent pas.

À la fin du script, il ne faut pas oublier de fermer le fichier spool.

Les scripts de votre schéma relationnel

La base de données qui a été créée ne contient que le dictionnaire de données. Pour pouvoir travailler avec vos propres applications, il faut créer le schéma relationnel nécessaire pour ces applications.

La sauvegarde



À ce stade, il importe d'expliciter la différence entre l'instance et la base de données.

La base de données est, comme nous venons de le voir, l'ensemble des fichiers de données, des fichiers de contrôle et des fichiers journaux, qui sert à la gestion des informations et des données de vos applications. La base de données est identifiée par son nom ; rappelez-vous le paramètre « **DB_NAME** » qui, combiné au nom du domaine « **DB_NAME** », définit un nom unique de la base de données.

L'instance est l'ensemble des zones mémoires et des processus d'arrière-plan, capable de lire et écrire les fichiers de la base de données. On a vu que l'instance est identifiée par son nom « **INSTANCE_NAME** ». Celui-ci peut être modifié après la création de cette instance.

En effet, l'instance est en quelque sorte un programme qui peut être changé ou déchargé en mémoire, et qui utilise les fichiers de la base de données.

Dans l'exemple suivant, nous allons détruire l'instance créée précédemment, tout en laissant à leur place les différents fichiers de la base de données. Ensuite nous allons créer une nouvelle instance qui utilise les mêmes fichiers de données.

```
C:\> oradim -DELETE -SID tpdba
Instance supprimée.
C:\> set ORACLE_SID=tpdbanew

C:\> oradim -NEW -SID tpdbanew -INTPWD password -STARTMODE m
Instance créée.

C:\> sqlplus "/as sysdba"
Connecté à une instance inactive.

SQL> create spfile from pfile=
  2  'C:\app\oracle\admin\tpdba\pfile\inittpdba.ora';

Fichier créé.
```

```
SQL> startup
Instance ORACLE lancée.

Total System Global Area  535662592 bytes
Fixed Size                  1334380 bytes
Variable Size             167773076 bytes
Database Buffers          360710144 bytes
Redo Buffers                5844992 bytes
Base de données montée.
Base de données ouverte.

SQL> select INSTANCE_NAME from V$INSTANCE;

INSTANCE_NAME
----------------
tpdbanew

SQL> select NAME,OPEN_MODE from V$DATABASE;

NAME      OPEN_MODE
--------- ----------
TPDBA     READ WRITE
```

L'outil « **ORADIM** » nous permet d'effacer le service « **tpdba** » et de créer un nouveau service « **tpdbanew** ». La création du service comporte également l'argument pour créer le fichier des mots de passe.

On se connecte à l'instance, authentifiée par le système d'exploitation, avec l'outil SQL*Plus. Comme vous pouvez le remarquer, l'instance n'a pas été démarrée, et l'ordre des créations du service définit le mode de démarrage manuel de l'instance.

On crée un nouveau fichier de paramètres serveur en utilisant le fichier paramètre de l'instance précédente. En effet le fichier paramètre de l'instance précédente ne contient aucune référence au nom de l'instance.

Le démarrage du serveur de base de données s'est effectué sans encombre. Vous pouvez remarquer que le nom de l'instance a été changé, mais que le nom de la base de données est resté le même.

Il faut avoir à l'esprit que toutes les informations du réseau Oracle Net prenant en compte le nom ancien de l'instance « **SID** » sont inutilisables.

Astuce

Dans l'exemple précédent on a pu observer l'effacement de l'instance et la création d'une nouvelle instance qui utilise les fichiers de la base de données, ceux-ci ayant pu être lus par la nouvelle instance.

Alors, si on sauvegarde les fichiers de la base de données puis une copie de ces fichiers, on peut les déployer sur un autre serveur et ensuite créer une instance pour cette base de données. Bien sûr, les fichiers doivent être positionnés dans la même arborescence de répertoires.

En effet, pour avoir une sauvegarde d'une base de données Oracle, il faut d'abord arrêter le serveur et copier l'ensemble des fichiers de données des fichiers de contrôles et des fichiers de journaux. Cette sauvegarde vous donne une image du serveur à l'instant où vous avez effectué votre sauvegarde. Pour plus d'informations concernant les sauvegardes et les restaurations, voir les modules correspondants.

Pour retrouver l'emplacement de l'ensemble des fichiers de données des fichiers de contrôle et des fichiers de journaux, vous pouvez interroger les vues dynamiques sur la performance.

Dans l'exemple suivant, nous allons interroger la base de données pour retrouver l'emplacement des fichiers de la base de données.

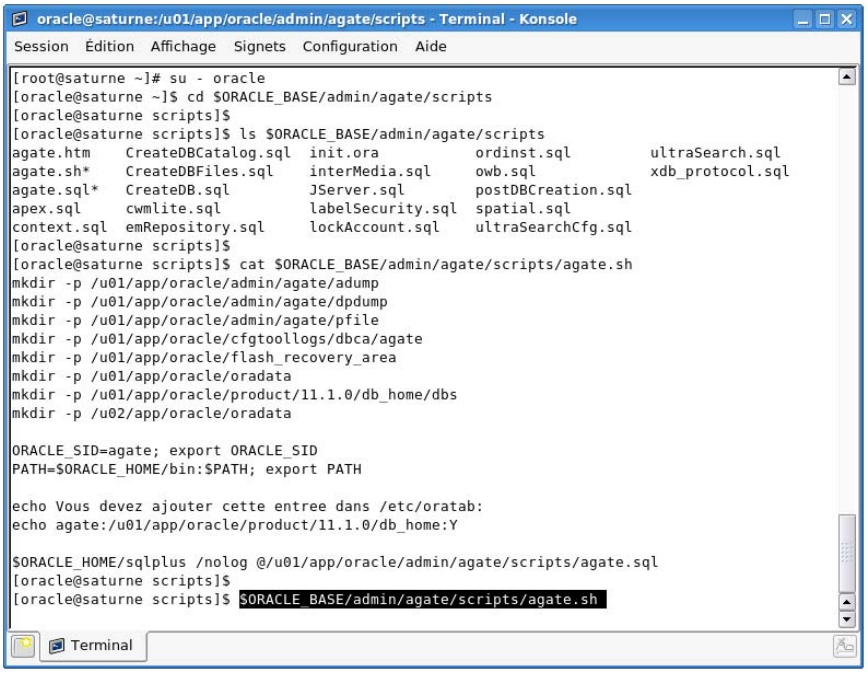
```
C:\> sqlplus / as sysdba

SQL> select *  from (
  2    select 'Données' "Fichier", NAME  "Nom" from V$DATAFILE
  3    union all
  4    select 'Contrôle', NAME   from V$CONTROLFILE
  5    union all
  6    select 'Journal', MEMBER from V$LOGFILE);

Fichier  Nom
-------- ------------------------------------------------------------
Données  C:\APP\ORACLE\ORADATA\TPDBA\DATAFILE\O1_MF_SYSTEM_55WW518S_.DBF
Données  C:\APP\ORACLE\ORADATA\TPDBA\DATAFILE\O1_MF_SYSAUX_55WW61N0_.DBF
Données  C:\APP\ORACLE\ORADATA\TPDBA\DATAFILE\O1_MF_UNDOTBS1_55WW69QX_.DBF
Données  C:\APP\ORACLE\ORADATA\TPDBA\DATAFILE\O1_MF_USERS_55WWFOD6_.DBF
Contrôle C:\APP\ORACLE\ORADATA\TPDBA\CONTROLFILE\O1_MF_55WW4JB8_.CTL
Journal  C:\APP\ORACLE\ORADATA\TPDBA\ONLINELOG\O1_MF_1_55WW4KMJ_.LOG
Journal  C:\APP\ORACLE\ORADATA\TPDBA\ONLINELOG\O1_MF_2_55WW4QNZ_.LOG
Journal  C:\APP\ORACLE\ORADATA\TPDBA\ONLINELOG\O1_MF_3_55WW4VNJ_.LOG
```

Les quatre premières réponses sont des fichiers de données, la suivante un fichier de contrôle et ensuite trois fichiers journaux.

La création manuelle de la base de données vous a permis de voir quelles sont les étapes de création d'une base de données et les différents composants qu'il faut créer.

Dans les exemples utilisés, on a travaillé dans l'environnement Windows parce que la création de la base de données comporte une étape de plus, à savoir la création du service Windows. Dans l'environnement Unix/Linux, tous les exemples peuvent être exécutés avec des modifications minimes concernant les noms des fichiers et leur emplacement.

Les scripts générés



L'assistant de configuration de base de données DBCA qui est détaillé dans le module 7 vous permet de générer les scripts de création de la base de données. Ces scripts peuvent être configurés et personnalisés pour une utilisation répétitive sur plusieurs serveurs avec le même système d'exploitation. Les scripts suivants ont été générés pour la création de la base de données « **agate.etelia.fr** » dans le répertoire « **$ORACLE_BASE/admin/agate/scripts** ».

Attention, tous les fichiers présentés par la suite ne sont pas les fichiers directement crées par DBCA, mais des versions améliorées pour une meilleure présentation, réutilisation et maintenance.

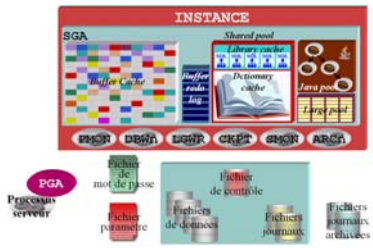
Le fichier paramètres

Le premier fichier à examiner est le fichier paramètres « **INIT.ORA** » contenant tous les paramètres que vous avez sélectionnés dans DBCA. Pour plus d'informations sur les paramètres et la gestion du fichier de paramètres voir le module « La gestion d'une instance ».

```
log_archive_dest_1='LOCATION=/u02/app/oracle/oradata'
log_archive_format=%t_%s_%r.dbf
db_block_size=8192
db_create_file_dest=/u01/app/oracle/oradata
db_create_online_log_dest_1=/u01/app/oracle/oradata
db_create_online_log_dest_2=/u02/app/oracle/oradata
db_recovery_file_dest=/u01/app/oracle/flash_recovery_area
db_recovery_file_dest_size=21474836480
open_cursors=300
compatible=11.2.0.0.0
diagnostic_dest=/u01/app/oracle
memory_target=1429209088
db_domain=etelia.fr
db_name=agate
nls_language="FRENCH"
nls_territory="FRANCE"
processes=150
undo_tablespace=UNDOTBS1
```

```
dispatchers="(PROTOCOL=TCP) (SERVICE=agateXDB)"
audit_file_dest=/u01/app/oracle/admin/agate/adump
audit_trail=db
remote_login_passwordfile=EXCLUSIVE
```

Le démarrage



- **Le script de commande**

  « db_name.sh » ou « db_name.bat »

- **Le script SQL de création**

  « db_name.sql »

Le script de commande

Le fichier de commande qui a le nom de l'instance « **agate.sh** » est le point unique pour le lancement de création de la base de données.

```
mkdir -p $ORACLE_BASE/admin/agate/adump
mkdir -p $ORACLE_BASE/admin/agate/dpdump
mkdir -p $ORACLE_BASE/admin/agate/pfile
mkdir -p $ORACLE_BASE/admin/agate/scripts/logs
mkdir -p $ORACLE_BASE/cfgtoollogs/dbca/agate
mkdir -p $ORACLE_BASE/flash_recovery_area
mkdir -p $ORACLE_BASE/oradata
mkdir -p $ORACLE_HOME/dbs
mkdir -p /u02/app/oracle/oradata

AGATE_SCRIPTS=$ORACLE_BASE/admin/agate/scripts; export AGATE_SCRIPTS
AGATE_HOST=`hostname`; export AGATE_HOST
ORACLE_SID=agate; export ORACLE_SID
PATH=$ORACLE_HOME/bin:$PATH; export PATH

$ORACLE_HOME/bin/sqlplus /nolog @$AGATE_SCRIPTS/agate.sql $AGATE_HOST $ORACLE_HOME

echo 'agate:/u01/app/oracle/product/11.2.0/db_home:N' >> /etc/oratab
```

Le script SQL de création

Le script de commande Shell, une fois qu'il a configuré l'environnement et créé les répertoires nécessaires pour stocker les différents fichiers de la base de données, lance le script SQL « **agate.sql** ». C'est précisément celui-ci qui permettra de contrôler la création de la base de données.

```
set verify off
DEFINE agateHOST = &1
DEFINE agateHOME = &2
```

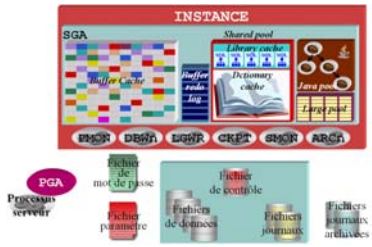
```
ACCEPT sysPassword    CHAR PROMPT "Le mot de passe pour l'utilisateur sys    : " HIDE
ACCEPT systemPassword CHAR PROMPT "Le mot de passe pour l'utilisateur system : " HIDE
ACCEPT sysmanPassword CHAR PROMPT "Le mot de passe pour l'utilisateur sysman : " HIDE
ACCEPT dbsnmpPassword CHAR PROMPT "Le mot de passe pour l'utilisateur dbsnmp : " HIDE

host $ORACLE_HOME/bin/orapwd file=$ORACLE_HOME/dbs/orapwagate password=&&sysPassword
force=y

@$AGATE_SCRIPTS/CreateDB.sql
@$AGATE_SCRIPTS/CreateDBFiles.sql
@$AGATE_SCRIPTS/CreateDBCatalog.sql
@$AGATE_SCRIPTS/JServer.sql
@$AGATE_SCRIPTS/context.sql
@$AGATE_SCRIPTS/xdb_protocol.sql
@$AGATE_SCRIPTS/ordinst.sql
@$AGATE_SCRIPTS/interMedia.sql
@$AGATE_SCRIPTS/cwmlite.sql
@$AGATE_SCRIPTS/spatial.sql
@$AGATE_SCRIPTS/ultraSearch.sql
@$AGATE_SCRIPTS/labelSecurity.sql
@$AGATE_SCRIPTS/emRepository.sql
@$AGATE_SCRIPTS/apex.sql
@$AGATE_SCRIPTS/owb.sql
@$AGATE_SCRIPTS/ultraSearchCfg.sql
@$AGATE_SCRIPTS/lockAccount.sql
@$AGATE_SCRIPTS/postDBCreation.sql
exit;
```

Au début du script il y a des invites pour les mots de passe des quatre comptes « **sys** », « **system** », « **sysman** » et « **dbsnmp** ». Ces valeurs sont utilisées à travers tous les scripts pour renseigner les mots de passe pour ces utilisateurs. Le mot de passe de l'utilisateur « **sys** » est utilisé pour créer un fichier de mots de passe, nécessaire pour la gestion de l'instance. Pour plus d'informations sur le fichier de mots de passe, reportez-vous au module « La gestion d'une instance ».

Le script lance plusieurs autres scripts nécessaires pour la création de toutes les options que vous avez choisies dans **DBCA**. Les seuls scripts présentés ici sont les scripts obligatoires pour la création de la base de données et pour la console d'administration.

La création de la base

« **CreateDB.sql** »



CreateDB

Le script crée le fichier des paramètres serveur et lance l'instance pour pouvoir créer la base de données par la suite.

```
connect / as SYSDBA
set echo on
spool $AGATE_SCRIPTS/logs/CreateDB.log

CREATE SPFILE FROM PFILE =
    '/u01/app/oracle/admin/agate/scripts/init.ora';

STARTUP NOMOUNT

CREATE DATABASE agate
         MAXINSTANCES 8
         MAXLOGHISTORY 1
         MAXLOGFILES 56
         MAXLOGMEMBERS 3
         MAXDATAFILES 1024
DATAFILE SIZE 300M AUTOEXTEND ON NEXT 10M
SYSAUX DATAFILE SIZE 120M AUTOEXTEND ON NEXT 10M
DEFAULT TEMPORARY TABLESPACE TEMP TEMPFILE
        SIZE 20M AUTOEXTEND ON NEXT 640K
UNDO TABLESPACE UNDOTBS1 DATAFILE
        SIZE 200M AUTOEXTEND ON NEXT  5M
CHARACTER SET WE8MSWIN1252
NATIONAL CHARACTER SET AL16UTF16
SET DEFAULT BIGFILE TABLESPACE
LOGFILE GROUP 1  SIZE 51200K,
         GROUP 2  SIZE 51200K,
         GROUP 3  SIZE 51200K,
         GROUP 4  SIZE 51200K,
         GROUP 5  SIZE 51200K,
```

```
         GROUP 6  SIZE 51200K,
         GROUP 7  SIZE 51200K,
         GROUP 8  SIZE 51200K
USER SYS IDENTIFIED BY &&sysPassword
USER SYSTEM IDENTIFIED BY &&systemPassword;

spool off
```

Il est impératif de créer le fichier paramètre avant de créer la base de données ; ainsi, si vous utilisez la gestion automatique des fichiers, les noms des fichiers de contrôle sont mis à jour également dans le fichier paramètre. Voir le module « Le fichier de contrôle » pour plus d'information.

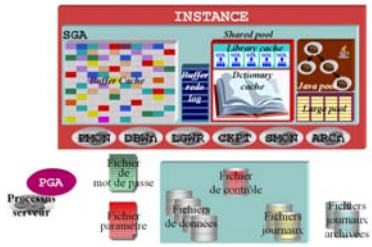
CreateDBFiles

Le script est d'importance mineure ; il crée un petit tablespace « **USERS** » utilisé comme emplacement par défaut pour les objets créés par les utilisateurs.

```
connect / as SYSDBA
set echo on
spool $AGATE_SCRIPTS/logs/CreateDBFiles.log

CREATE TABLESPACE USERS LOGGING DATAFILE SIZE 5M AUTOEXTEND ON NEXT 1M;
ALTER DATABASE DEFAULT TABLESPACE USERS;

spool off
```

La création du dictionnaire

« **CreateDBCatalog.sql** »



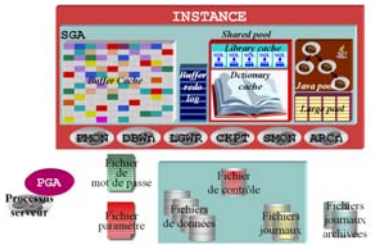
CreateDBCatalog

Le script exécute tous les scripts nécessaires pour la création du dictionnaire de données. Attention, ces scripts sont relativement longs ; il faut par conséquent prendre la précaution que toutes les informations affichées puissent être lues par la suite. C'est pourquoi on ouvre un fichier spool pour pouvoir vérifier l'exécution de ces scripts.

Les scripts peuvent retourner des messages d'erreurs dus à l'effacement d'objets qui n'existent pas.

```
connect / as SYSDBA
set echo on
spool $AGATE_SCRIPTS/logs/CreateDBCatalog.log
@ ?/rdbms/admin/catalog.sql;
@ ?/rdbms/admin/catblock.sql;
@ ?/rdbms/admin/catproc.sql;
@ ?/rdbms/admin/catoctk.sql;
@ ?/rdbms/admin/owminst.plb;
spool off

connect SYSTEM/&&systemPassword
@ ?/sqlplus/admin/pupbld.sql;

connect SYSTEM/&&systemPassword
set echo on
spool $AGATE_SCRIPTS/logs/sqlPlusHelp.log
@ ?/sqlplus/admin/help/hlpbld.sql helpus.sql;
spool off
```

La configuration finale



« **emRepository.sql** »

« **postDBCreation.sql** »

emRepository

Le script permet de créer le référentiel des objets nécessaires à la console d'administration. Voir le module 11 pour plus d'informations sur la console d'administration.

```
connect / as SYSDBA
set echo off
spool $AGATE_SCRIPTS/logs/emRepository.log

@ ?/sysman/admin/emdrep/sql/emreposcre &&agateHOME SYSMAN &&sysmanPassword
TEMP ON;
WHENEVER SQLERROR CONTINUE;

spool off
```

postDBCreation

Le script permet de déverrouiller les utilisateurs « **DBSNMP** » et « **SYSMAN** », comptes utilisés par la console d'administration. Il génère également la console d'administration pour cette instance. Voir le module 9 pour plus d'informations sur la console et sur les utilisateurs « **DBSNMP** » et « **SYSMAN** ».

Il est possible d'effectuer dans ce script les opérations de maintenance de la base de données comme, par exemple, l'activation du mode « **ARCHIVELOG** » ou « **FLASHBACK** ».

```
connect / as SYSDBA
set echo on
spool $AGATE_SCRIPTS/logs/postDBCreation.log
shutdown immediate

connect / as SYSDBA
startup mount
```

```
alter database archivelog;
alter database open;

alter user SYSMAN identified by "&&sysmanPassword" account unlock;
alter user DBSNMP identified by "&&dbsnmpPassword" account unlock;

select 'utl_recomp_begin: ' || to_char(sysdate, 'HH:MI:SS') from dual;
execute utl_recomp.recomp_serial();
select 'utl_recomp_end: ' || to_char(sysdate, 'HH:MI:SS') from dual;

host $ORACLE_HOME/bin/emca -config dbcontrol db -silent -DB_UNIQUE_NAME
agate -PORT 1521 -EM_HOME &&agateHOME -LISTENER LISTENER -SERVICE_NAME
agate.etelia.fr -SYS_PWD &&sysPassword -SID agate -ORACLE_HOME &&agateHOME -
DBSNMP_PWD &&dbsnmpPassword -HOST &&agateHOST -LISTENER_OH &&agateHOME -
LOG_FILE $AGATE_SCRIPTS/emConfig.log -SYSMAN_PWD &&sysmanPassword;

spool off
```

Il suffit à présent de lancer le script de commande Shell pour créer une base de données « **agate.etelia.fr** ».

Atelier 6

Questions

6-1 Quelles sont les privilèges que vous devez avoir pour pouvoir créer une base de données ?

A. DBA

B. SYSDBA

C. SYSOPER

D. RESOURCE

6-2 Quels sont les trois composants qui constituent la base de données ?

A. Table

B. Extent

C. Fichier de données

D. Fichier journaux

E. Segment

F. Tablespace

G. Fichier de contrôle

6-3 Vous voulez créer une nouvelle base de données. Vous ne voulez pas utiliser l'authentification par le système d'exploitation. Quels sont les deux fichiers que vous devez créer avant la création de la base de données ?

A. Fichier de contrôle

B. Fichier de mots de passe

C. Fichier journaux

D. Fichier d'alerte

E. Fichier de paramètres

6-4 Quelles sont les deux variables d'environnement qui doivent être initialisées avant la création de la base de données ?

A. DB_NAME

B. ORACLE_SID

C. ORACLE_HOME

D. SERVICE_NAME

E. INSTANCE_NAME

6-5 Quel est le mode de démarrage de l'instance pour pouvoir créer une base de données ?

A. STARTUP

B. STARTUP NOMOUNT

C. STARTUP MOUNT

D. STARTUP OPEN

- *DICTIONARY*
- *DICT_COLUMNS*
- *DBA_CATALOG*
- *DBA_OBJECTS*

8

Le dictionnaire de données

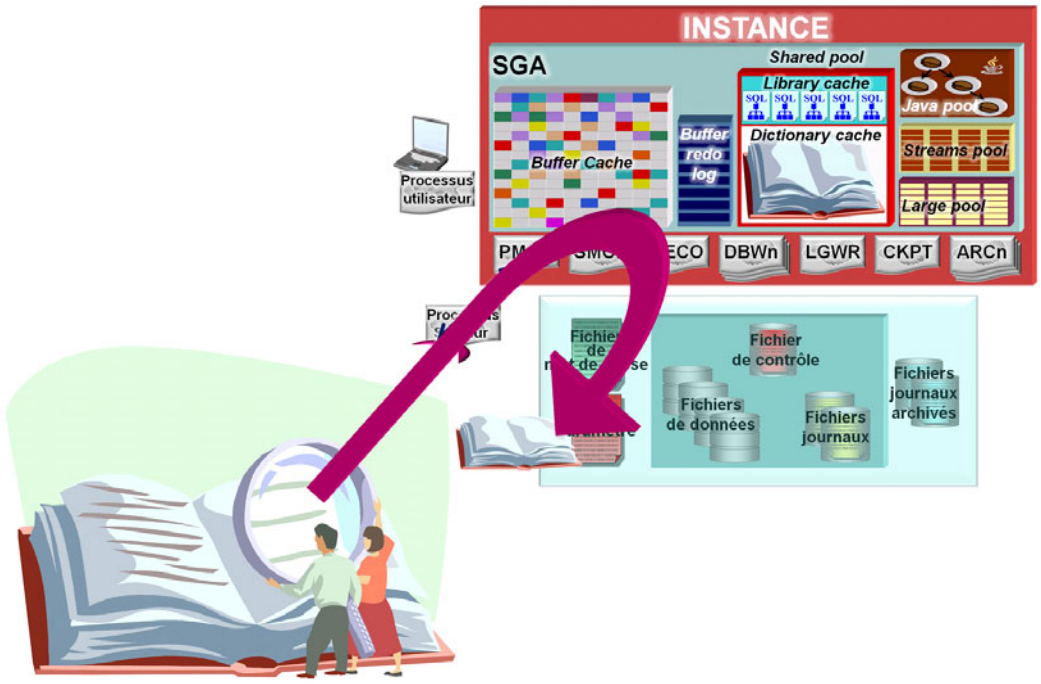
Objectifs

À la fin de ce module, vous serez à même d'effectuer les tâches suivantes :

- Décrire l'environnement du dictionnaire de données.
- Décrire les vues du dictionnaire de données.
- Interroger les vues du dictionnaire de données.

Contenu

Le dictionnaire de données



Le dictionnaire est un ensemble de tables et de vues qui contient toutes les informations concernant la structure de stockage et tous les objets de la base. Toute information concernant la base de données se retrouve dans le dictionnaire de données.

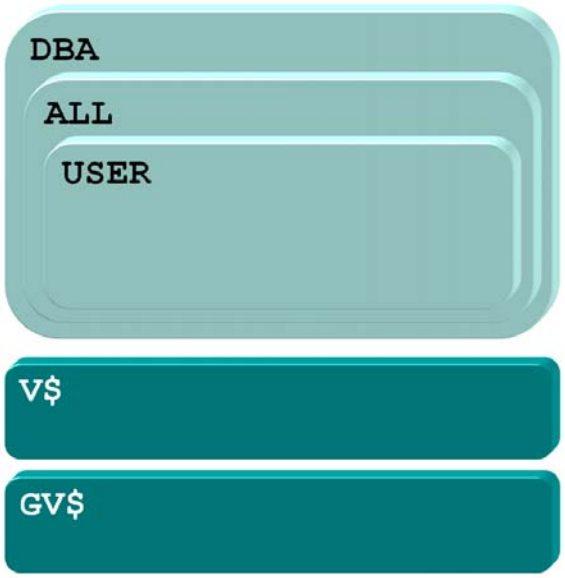
Le dictionnaire de données est automatiquement mis à jour par Oracle lorsque la base de données a été modifiée. Le propriétaire du dictionnaire de données est l'utilisateur « **SYS** ».

Le dictionnaire de données stocke les informations sur :

- La structure logique de la base de données.
- La structure physique de la base de données.
- Les noms et les définitions des objets.
- Les contraintes d'intégrité définies pour les objets d'une base de données.
- Les noms des utilisateurs valides de la base de données et les privilèges attribués à chaque utilisateur de la base de données.
- L'audit sur une base de données.

Le dictionnaire de données Oracle stocke toutes les informations utilisées pour gérer les objets de la base. Ce dictionnaire est généralement exploité par l'administrateur de base de données, mais c'est aussi une source d'information utile pour les développeurs et les utilisateurs.

Les vues du dictionnaire de données



Le dictionnaire est un ensemble de tables et de vues qui contient toutes les informations concernant la structure de stockage et tous les objets de la base. Toute information concernant la base de données se retrouve dans le dictionnaire de données.

Les vues du dictionnaire de données

Les noms des objets dans le dictionnaire de données Oracle débutent par l'un des trois préfixes suivants :

- Les vues DBA contiennent des informations sur les objets de tous les schémas.
- Les vues ALL incluent les enregistrements des vues USER et des informations sur les objets pour lesquels des privilèges ont été octroyés au groupe PUBLIC ou à l'utilisateur courant.
- Les vues USER contiennent des informations sur les objets appartenant au compte qui exécute la requête.

Les vues USER, ALL et DBA sont disponibles pour quasiment tous les objets de base de données.

Les vues dynamiques normales

Les vues dynamiques des performances sont identifiées avec le préfixe V_$, mais le serveur Oracle crée un synonyme public avec le préfixe V$. Elles sont accessibles uniquement pour l'utilisateur « **SYS** » ou pour tout autre utilisateur qui a le privilège « **SYSDBA** ».

La liste complète des vues de performances est disponible à partir de la vue « **V$FIXED_TABLE** ».

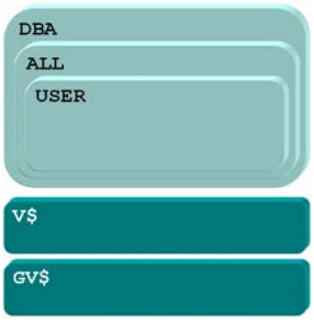
Ces vues sont utilisées pour fournir des données relatives aux performances telles que des informations sur les fichiers de données et les structures de la mémoire.

Les vues des bases en cluster

Les informations fournies par ces vues sont uniquement nécessaires dans une configuration de cluster. Pour surveiller les performances d'instances placées sur des serveurs différents, il est important de disposer des vues identiques aux vues dynamiques mais permettant d'identifier l'instance surveillée. Elles sont identifiées avec le préfixe « GV$ ».

Le guide du dictionnaire

- La vue DICTIONARY
- La vue DICT_COLUMNS



Les descriptions des objets sont accessibles via une vue nommée « **DICTIONARY** ». Cette vue, également accessible via le synonyme public « **DICT** », interroge la base de données pour déterminer à quelles vues du dictionnaire vous pouvez accéder. Elle recherche également les synonymes publics définis pour ces vues.

L'exemple suivant sélectionne dans la vue « **DICT** » les noms des vues du dictionnaire de données qui incluent la chaîne « **VIEWS** ». Cette vue contient uniquement deux colonnes, le nom d'objet et les commentaires associés aux objets du dictionnaire.

```
SQL> DESC DICTIONARY
Nom                                       NULL ?   Type
 ----------------------------------------- -------- ----------------------------
 TABLE_NAME                                         VARCHAR2(30)
 COMMENTS                                           VARCHAR2(4000)
SQL> select * from dict
  2  where TABLE_NAME like '%BASE%' ;

 TABLE_NAME                     COMMENTS
------------------------------ --------------------------------------------
DBA_BASE_TABLE_MVIEWS           All materialized views with log(s) in the
ALL_BASE_TABLE_MVIEWS           All materialized views with log(s) in the
USER_BASE_TABLE_MVIEWS          All materialized views with log(s) owned by
DBA_IAS_OBJECTS_BASE
DBA_CACHEABLE_TABLES_BASE
DBA_CACHEABLE_OBJECTS_BASE
DBA_CAPTURE_PREPARED_DATABASE   Is the local database prepared for
ALL_CAPTURE_PREPARED_DATABASE   Is the local database prepared for
DBA_HIST_DATABASE_INSTANCE      Database Instance Information
DBA_HIST_BASELINE               Baseline Metadata Information
```

```
DBA_DIR_DATABASE_ATTRIBUTES    Database attributes for cluster director
DATABASE_COMPATIBLE_LEVEL      Database compatible parameter set via
NLS_DATABASE_PARAMETERS        Permanent NLS parameters of the database
V$FLASHBACK_DATABASE_LOGFILE   Synonym for V_$FLASHBACK_DATABASE_LOGFILE
V$FLASHBACK_DATABASE_LOG       Synonym for V_$FLASHBACK_DATABASE_LOG
V$FLASHBACK_DATABASE_STAT      Synonym for V_$FLASHBACK_DATABASE_STAT
V$DATABASE                     Synonym for V_$DATABASE
V$DATABASE_BLOCK_CORRUPTION    Synonym for V_$DATABASE_BLOCK_CORRUPTION
V$DATABASE_INCARNATION         Synonym for V_$DATABASE_INCARNATION
GV$FLASHBACK_DATABASE_LOGFILE  Synonym for GV_$FLASHBACK_DATABASE_LOGFILE
GV$FLASHBACK_DATABASE_LOG      Synonym for GV_$FLASHBACK_DATABASE_LOG
GV$FLASHBACK_DATABASE_STAT     Synonym for GV_$FLASHBACK_DATABASE_STAT
GV$DATABASE                    Synonym for GV_$DATABASE
GV$DATABASE_BLOCK_CORRUPTION   Synonym for GV_$DATABASE_BLOCK_CORRUPTION
GV$DATABASE_INCARNATION        Synonym for GV_$DATABASE_INCARNATION

25 ligne(s) sélectionnée(s).
```

Vous pouvez interroger les colonnes des vues du dictionnaire via la vue « **DICT_COLUMNS** ». À l'instar de la vue « **DICTIONARY** », cette vue affiche les colonnes pour les vues du dictionnaire de données. Elle possède trois colonnes, le nom d'objet, le nom de la colonne et les commentaires associés aux objets du dictionnaire. L'interrogation de cette vue permet de déterminer les vues du dictionnaire les plus appropriées à vos recherches.

```
SQL> DESC DICT_COLUMNS
Nom                                       NULL ?   Type
 ----------------------------------------- -------- ----------------
 TABLE_NAME                                         VARCHAR2(30)
 COLUMN_NAME                                        VARCHAR2(30)
 COMMENTS                                           VARCHAR2(4000)

SQL> SELECT TABLE_NAME FROM DICT_COLUMNS
  2  WHERE  COLUMN_NAME LIKE 'BLOCK' ;

TABLE_NAME
------------------------------
DBA_LMT_USED_EXTENTS
DBA_DMT_USED_EXTENTS
V$_LOCK
V$LOCK
V$ENQUEUE_LOCK
V$TRANSACTION_ENQUEUE
GV$ENQUEUE_LOCK
GV$TRANSACTION_ENQUEUE
GV$_LOCK
GV$LOCK
```

Dans l'exemple précédent, nous avons interrogé la vue « **DICT_COLUMNS** », afin d'obtenir la liste de toutes les vues possédant une colonne appelée « **BLOCS** ».

Vous pouvez créer un script interactif qui vous permet de rechercher les vues du dictionnaire de données suivant le nom de la vue ou une colonne bien spécifique.

Pour créer un script interactif, il faut se rappeler que **SQL*Plus** est un environnement de commandes qui vous permet de formater les scripts **SQL**, de les enregistrer sur le disque, et en même temps d'exécuter des fichiers scripts. Ainsi l'on va créer un fichier qui va interroger la base à partir du dictionnaire de données.

Voici un exemple des scripts qui vous permet d'interroger le dictionnaire de données et de créer une liste des différentes vues avec leur description.

```
-------------------------------------------------------------------------------
-- « recherche_dict.sql »
-- Fichier interactif pour trouver les vues du dictionnaire
--                       les plus appropriées à vos besoins.
-------------------------------------------------------------------------------
SET SERVEROUTPUT ON SIZE UNL
SET VERIFY OFF
SET LINESIZE 5000

PROMPT
PROMPT Vous pouvez saisir le nom en entier ou seulement une partie
PROMPT
ACCEPT var_nom_vue CHAR PROMPT "Le nom de la vue : "
PROMPT
ACCEPT var_nom_col CHAR PROMPT "Le nom de la colonne : "
PROMPT

spool c:\recherche_dict.lst

begin
 for var_vues in ( SELECT TABLE_NAME, COMMENTS
                   FROM  DICT
                   WHERE TABLE_NAME  like '%&var_nom_vue%' AND
                         TABLE_NAME  in ( SELECT TABLE_NAME
                                          FROM  DICT_COLUMNS
                                          WHERE COLUMN_NAME like '%&var_nom_col%'))
 loop
   DBMS_OUTPUT.PUT_LINE( '#########################################################'||
                         '#########################################################');
   DBMS_OUTPUT.PUT_LINE( '#    Vue         : '||var_vues.TABLE_NAME);
   DBMS_OUTPUT.PUT_LINE( '#    Description : '||var_vues.COMMENTS);
   DBMS_OUTPUT.PUT_LINE( '#########################################################'||
                         '#########################################################');
   for desc_col in ( SELECT DTC.COLUMN_NAME, DATA_TYPE, DCC.COMMENTS,
                            max( length(DTC.DATA_TYPE  )) over () taille_type
                                   FROM DBA_TAB_COLUMNS DTC, DBA_COL_COMMENTS DCC
                                   WHERE DTC.OWNER = DCC.OWNER
                                            AND DTC.TABLE_NAME = DCC.TABLE_NAME
                                            AND DTC.COLUMN_NAME = DCC.COLUMN_NAME
                                            AND DTC.TABLE_NAME = var_vues.TABLE_NAME)
   loop
      DBMS_OUTPUT.PUT_LINE( RPAD( desc_col.COLUMN_NAME, 30)||' '||
                    RPAD( desc_col.DATA_TYPE, GREATEST( desc_col.taille_type,12))
                              ||' '||desc_col.COMMENTS);
    end loop ;
 end loop ;
end ;
/
spool off
```

Le script précédent interroge la base de données pour obtenir les noms des vues que vous recherchez. Le résultat est affiché formaté et stocké dans le fichier « **c:\recherche_dict.lst** ».

À l'exécution de ce script, vous devez saisir le nom de la vue et le nom de la colonne que vous recherchez. Les deux noms ne sont pas obligatoires, vous n'êtes pas non plus obligé de saisir la description complète de chaque nom. Vous pouvez alors saisir des parties de nom ou ne pas saisir du tout. Voici l'exemple qui génère un fichier script **SQL** et une liste des différentes vues avec leur description.

```
SQL> @C:\recherche_dict.sql

Vous pouvez saisir le nom en entier ou seulement une partie

Le nom de la vue : DBA%STATISTICS

Le nom de la colonne : PARTITION_NAME

################################################################################
#    Vue         : DBA_IND_STATISTICS
#    Description : Optimizer statistics for all indexes in the database
################################################################################
OBJECT_TYPE                    VARCHAR2 Type of the object (INDEX, PARTITION,
SUBPARTITION)
BLEVEL                         NUMBER   B-Tree level
LEAF_BLOCKS                    NUMBER   The number of leaf blocks in the index
DISTINCT_KEYS                  NUMBER   The number of distinct keys in the index
AVG_LEAF_BLOCKS_PER_KEY        NUMBER   The average number of leaf blocks per key
AVG_DATA_BLOCKS_PER_KEY        NUMBER   The average number of data blocks per key
CLUSTERING_FACTOR              NUMBER   A measurement of the amount of (dis)order of
the table this index is for
NUM_ROWS                       NUMBER   The number of rows in the index
AVG_CACHED_BLOCKS              NUMBER   Average number of blocks in buffer cache
AVG_CACHE_HIT_RATIO            NUMBER   Average cache hit ratio for the object
SAMPLE_SIZE                    NUMBER   The sample size used in analyzing this index
LAST_ANALYZED                  DATE     The date of the most recent time this index
was analyzed
GLOBAL_STATS                   VARCHAR2 Are the statistics calculated without merging
underlying partitions?
USER_STATS                     VARCHAR2 Were the statistics entered directly by the
user?
STATTYPE_LOCKED                VARCHAR2 type of statistics lock
STALE_STATS                    VARCHAR2 Whether statistics for the object is stale or
not
OWNER                          VARCHAR2 Username of the owner of the index
INDEX_NAME                     VARCHAR2 Name of the index
TABLE_OWNER                    VARCHAR2 Owner of the indexed object
TABLE_NAME                     VARCHAR2 Name of the indexed object
PARTITION_NAME                 VARCHAR2 Name of the partition
PARTITION_POSITION             NUMBER   Position of the partition within index
SUBPARTITION_NAME              VARCHAR2 Name of the subpartition
SUBPARTITION_POSITION          NUMBER   Position of the subpartition within partition
################################################################################
#    Vue         : DBA_PART_COL_STATISTICS
#    Description :
################################################################################
OWNER                          VARCHAR2
```

```
TABLE_NAME                     VARCHAR2
PARTITION_NAME                 VARCHAR2
COLUMN_NAME                    VARCHAR2
NUM_DISTINCT                   NUMBER
LOW_VALUE                      RAW
HIGH_VALUE                     RAW
DENSITY                        NUMBER
NUM_NULLS                      NUMBER
NUM_BUCKETS                    NUMBER
SAMPLE_SIZE                    NUMBER
LAST_ANALYZED                  DATE
GLOBAL_STATS                   VARCHAR2
USER_STATS                     VARCHAR2
AVG_COL_LEN                    NUMBER
HISTOGRAM                      VARCHAR2
################################################################################
#    Vue         : DBA_SUBPART_COL_STATISTICS
#    Description :
################################################################################
OWNER                          VARCHAR2
TABLE_NAME                     VARCHAR2
SUBPARTITION_NAME              VARCHAR2
COLUMN_NAME                    VARCHAR2
NUM_DISTINCT                   NUMBER
LOW_VALUE                      RAW
HIGH_VALUE                     RAW
DENSITY                        NUMBER
NUM_NULLS                      NUMBER
NUM_BUCKETS                    NUMBER
SAMPLE_SIZE                    NUMBER
LAST_ANALYZED                  DATE
GLOBAL_STATS                   VARCHAR2
USER_STATS                     VARCHAR2
AVG_COL_LEN                    NUMBER
HISTOGRAM                      VARCHAR2
################################################################################
#    Vue         : DBA_TAB_STATISTICS
#    Description : Optimizer statistics for all tables in the database
################################################################################
OWNER                          VARCHAR2 Owner of the object
TABLE_NAME                     VARCHAR2 Name of the table
PARTITION_NAME                 VARCHAR2 Name of the partition
PARTITION_POSITION             NUMBER   Position of the partition within table
SUBPARTITION_NAME              VARCHAR2 Name of the subpartition
SUBPARTITION_POSITION          NUMBER   Position of the subpartition within partition
OBJECT_TYPE                    VARCHAR2 Type of the object (TABLE, PARTITION,
SUBPARTITION)
NUM_ROWS                       NUMBER   The number of rows in the object
BLOCKS                         NUMBER   The number of used blocks in the object
EMPTY_BLOCKS                   NUMBER   The number of empty blocks in the object
AVG_SPACE                      NUMBER   The average available free space in the
object
CHAIN_CNT                      NUMBER   The number of chained rows in the object
AVG_ROW_LEN                    NUMBER   The average row length, including row
overhead
AVG_SPACE_FREELIST_BLOCKS      NUMBER   The average freespace of all blocks on a
```

```
freelist
NUM_FREELIST_BLOCKS             NUMBER   The number of blocks on the freelist
AVG_CACHED_BLOCKS               NUMBER   Average number of blocks in buffer cache
AVG_CACHE_HIT_RATIO             NUMBER   Average cache hit ratio for the object
SAMPLE_SIZE                     NUMBER   The sample size used in analyzing this table
LAST_ANALYZED                   DATE     The date of the most recent time this table
was analyzed
GLOBAL_STATS                    VARCHAR2 Are the statistics calculated without merging
underlying partitions?
USER_STATS                      VARCHAR2 Were the statistics entered directly by the
user?
STATTYPE_LOCKED                 VARCHAR2 type of statistics lock
STALE_STATS                     VARCHAR2 Whether statistics for the object is stale or
not
```

Pour lister tous les noms des vues ainsi que toutes leurs colonnes, vous pouvez lancer le même script sans aucune valeur pour le nom de vues ni pour nom de colonnes.

Lorsque vous ne savez pas où rechercher les informations, interrogez le dictionnaire de données à l'aide du script précédent. Ainsi vous retrouvez les vues, leur description et également l'ensemble des colonnes de ces vues.

```
SQL> @C:\recherche_dict.sql

Vous pouvez saisir le nom en entier ou seulement une partie

Le nom de la vue : DBA_SYNONYMS

Le nom de la colonne :

################################################################################
#    Vue         : DBA_SYNONYMS
#    Description : All synonyms in the database
################################################################################
OWNER                           VARCHAR2 Username of the owner of the synonym
SYNONYM_NAME                    VARCHAR2 Name of the synonym
TABLE_OWNER                     VARCHAR2 Owner of the object referenced by the synonym
TABLE_NAME                      VARCHAR2 Name of the object referenced by the synonym
DB_LINK                         VARCHAR2 Name of the database link referenced in a
remote synonym
```

Les objets utilisateur

- **DBA_CATALOG**
- **DBA_OBJECTS**

L'ensemble des objets appartenant à un utilisateur est désigné par le terme catalogue ; il en existe un seul par utilisateur. Un catalogue affiche tous les objets dont l'utilisateur peut sélectionner les enregistrements.

DBA_CATALOG

La vue « **CATALOG** » liste tous les objets dont l'utilisateur peut sélectionner les enregistrements.

Les colonnes de cette vue sont :

OWNER Le propriétaire de l'objet.

TABLE_NAME Le nom de l'objet.

TABLE_TYPE Le type de l'objet.

```
SQL> DESC DBA_CATALOG
 Nom                                       NULL ?   Type
 ----------------------------------------- -------- ------------
 OWNER                                     NOT NULL VARCHAR2(30)
 TABLE_NAME                                NOT NULL VARCHAR2(30)
 TABLE_TYPE                                         VARCHAR2(11)

SQL> SELECT TABLE_NAME, TABLE_TYPE FROM DBA_CATALOG
  2  WHERE OWNER LIKE 'STAGIAIRE';

TABLE_NAME                     TABLE_TYPE
------------------------------ -----------
CATEGORIES                     TABLE
CLIENTS                        TABLE
EMPLOYES                       TABLE
```

```
FOURNISSEURS                   TABLE
COMMANDES                      TABLE
PRODUITS                       TABLE
DETAILS_COMMANDES              TABLE
```

La vue « **USER_CATALOG** » donne exactement le même affichage pour l'utilisateur courant car elle ne contient pas la colonne « **OWNER** ». La vue « **USER_CATALOG** » peut aussi être désignée par le synonyme public « **CAT** ».

```
SQL> CONNECT STAGIARE/PWD
Connecté

SQL> SELECT TABLE_NAME, TABLE_TYPE FROM CAT ;

TABLE_NAME                     TABLE_TYPE
------------------------------ -----------
CATEGORIES                     TABLE
CLIENTS                        TABLE
EMPLOYES                       TABLE
FOURNISSEURS                   TABLE
COMMANDES                      TABLE
PRODUITS                       TABLE
DETAILS_COMMANDES              TABLE
```

DBA_OBJECTS

La vue « **DBA_OBJECTS** » liste tous les types d'objets : clusters, liens de base de données, fonctions, index, paquetages, corps de paquetages, classes Java, types de données abstraits, plans de ressource, séquences, synonymes, tables, déclencheurs et vues.

Les colonnes de cette vue sont :

OWNER	Le propriétaire de l'objet.
OBJECT_NAME	Le nom de l'objet.
SUBOBJECT_NAME	Le nom d'un composant de l'objet, une partition par exemple.
OBJECT_ID	L'identifiant de l'objet.
DATA_OBJECT_ID	L'identifiant du segment qui contient l'objet.
OBJECT_TYPE	Le type de l'objet, par exemple une table, un index, une table partitionnée.
CREATED	La date et l'heure de création de l'objet.
LAST_DDL_TIME	La date et l'heure de la dernière modification « **DDL** » de l'objet.
TIMESTAMP	La date et l'heure de création de l'objet dans un champ de types de caractère.
STATUS	L'état de l'objet « **VALID** » ou « **INVALID** ».
TEMPORARY	Indicateur signifiant si l'objet est une table temporaire.
GENERATED	Indicateur signifiant si le nom de l'objet a été généré par le système.

SECONDARY Indicateur signifiant si l'objet est un index secondaire créé par un index de domaine.

La vue « **DBA_OBJECTS** » contient plusieurs informations essentielles qui ne sont pas disponibles via d'autres vues du dictionnaire de données. Cette vue consigne la date de création des objets et la date de leur dernière modification.

L'exemple suivant récupère la date de création et la date de dernière modification des objets de l'utilisateur « **STAGIAIRE** ».

```
SQL> SELECT OBJECT_NAME,OBJECT_TYPE,
  2  CREATED,LAST_DDL_TIME FROM DBA_OBJECTS
  3  WHERE OWNER LIKE 'STAGIAIRE';

OBJECT_NAME              OBJECT CREATED  LAST_DDL
------------------------ ------ -------- --------
CATEGORIES               TABLE  13/07/08 13/07/08
PK_CATEGORIES            INDEX  13/07/08 13/07/08
CLIENTS                  TABLE  13/07/08 13/07/08
PK_CLIENTS               INDEX  13/07/08 13/07/08
COMMANDES                TABLE  13/07/08 13/07/08
PK_COMMANDES             INDEX  13/07/08 13/07/08
FK_CLIENTS_COMMANDES     INDEX  13/07/08 13/07/08
FK_EMPLOYES_COMMANDES    INDEX  13/07/08 13/07/08
DETAILS_COMMANDES        TABLE  13/07/08 13/07/08
PK_DETAILS_COMMANDES     INDEX  13/07/08 13/07/08
FK_COM_DET_COM           INDEX  13/07/08 13/07/08
FK_PROD_DET_COM          INDEX  13/07/08 13/07/08
EMPLOYES                 TABLE  13/07/08 13/07/08
PK_EMPLOYES              INDEX  13/07/08 13/07/08
FK_EMPLOYES_REND_COMPTE  INDEX  13/07/08 13/07/08
FOURNISSEURS             TABLE  13/07/08 13/07/08
PK_FOURNISSEURS          INDEX  13/07/08 13/07/08
PRODUITS                 TABLE  13/07/08 13/07/08
PK_PRODUITS              INDEX  13/07/08 13/07/08
FK_CATEGORIES_PRODUITS   INDEX  13/07/08 13/07/08
FK_FOURNISEURS_PRODUITS  INDEX  13/07/08 13/07/08
UTILISATEURS             TABLE  13/07/08 13/07/08
DESCRIPTION              LOB    13/07/08 13/07/08
```

La vue « **USER_OBJECTS** » donne exactement le même affichage pour l'utilisateur courant car elle ne contient pas la colonne « **OWNER** ». La vue « **USER_OBJECTS** » peut aussi être désignée par le synonyme public « **OBJ** ».

La vue « **DBA_OBJECTS** » liste tous les types d'objets d'un utilisateur ; en revanche elle ne fournit pas beaucoup d'information sur leurs attributs. Pour obtenir davantage d'information sur un objet, vous devez examiner la vue spécifique à son type.

À ce stade, l'ensemble des vues ne peut pas être décrit en détail sachant que plusieurs notions n'ont pas encore été vues. Aussi allons-nous présenter un certain nombre de vues qui sont détaillées dans les modules suivants.

Les tables

- **DBA_TABLES**
- **DBA_TAB_COLUMNS**
- **DBA_CONSTRAINTS**
- **DBA_CONS_COLUMNS**

La vue « **DBA_OBJECTS** » liste tous les types d'objets d'un utilisateur ; en revanche elle ne fournit pas beaucoup d'information sur leurs attributs. Pour obtenir davantage d'information sur un objet, vous devez examiner la vue spécifique à son type.

À ce stade, l'ensemble des vues ne peut pas être décrit en détail sachant que plusieurs notions n'ont pas encore été vues. Aussi allons-nous présenter un certain nombre de vues qui sont détaillées dans les modules suivants.

DBA_TABLES

La vue « **DBA_TABLES** » affiche toutes les tables de la base de données. La plupart des outils de reporting tiers qui listent les tables disponibles pour les requêtes obtiennent cette liste en interrogeant cette vue.

Les colonnes de la vue « **DBA_TABLES** » peuvent être classées en quatre catégories principales : identification, espace de stockage, statistiques et autres.

Identification	*Espace de stockage*	*Statistiques*	*Autres*
OWNER	TABLESPACE_NAME	NUM_ROWS	DEGREE
TABLE_NAME	CLUSTER_NAME	BLOCKS	INSTANCES
IOT_NAME	PCT_FREE	EMPTY_BLOCKS	CACHE
LOGGING	INI_TRANS	AVG_SPACE	TABLE_LOCK
BACKED_UP	MAX_TRANS	CHAIN_CNT	BUFFER_POOL
PARTITIONED	INITIAL_EXTENT	AVG_ROW_LEN	ROW_MOVEMENT
IOT_TYPE		SAMPLE_SIZE	DURATION
TEMPORARY		LAST_ANALYZED	SKIP_CORRUPT

SECONDARY		AVG_SPACE_FREELIST_BLOCKS	MONITORING
NESTED		NUM_FREELIST_BLOCKS	CLUSTER_OWNER
STATUS		GLOBAL_STATS	DEPENDENCIES
		USER_STATS	COMPRESSION

Vous pourrez ainsi récupérer la liste des tables, l'information concernant les espaces des disques logiques dans lesquels sont stockées des informations plus détaillées concernant le type de table, les volumes de stockage ainsi que le mode de gestion de la mémoire.

```
SQL> SELECT TABLE_NAME,TABLESPACE_NAME, NUM_ROWS
  2       , BLOCKS, AVG_ROW_LEN, AVG_ROW_LEN
  3  FROM DBA_TABLES
  4  WHERE OWNER like 'STAGIAIRE';

TABLE_NAME           TABLESP NUM_ROWS BLOCKS AVG_ROW_LEN AVG_ROW_LEN
-------------------- ------- -------- ------ ----------- -----------
CATEGORIES           GVDATA         8      5          43          43
CLIENTS              GVDATA        91      5         131         131
COMMANDES            GVDATA       830      5          34          34
DETAILS_COMMANDES    GVDATA      2155     13          17          17
EMPLOYES             GVDATA         9      5          76          76
FOURNISSEURS         GVDATA        29      5         128         128
PRODUITS             GVDATA        77      5          71          71
UTILISATEURS         GVDATA      1000     13         122         122
```

DBA_TAB_COLUMNS

La vue du dictionnaire de données « **DBA_TAB_COLUMNS** » qui affiche des informations sur les colonnes est étroitement liée à la vue « **DBA_TABLES** ».

Les colonnes de la vue « **DBA_TAB_COLUMNS** » peuvent être classées en trois catégories principales :

Identification	*Définition*	*Statistiques*
OWNER	DATA_TYPE	NUM_DISTINCT
TABLE_NAME	DATA_TYPE_MOD	LOW_VALUE
COLUMN_NAME	DATA_TYPE_OWNER	HIGH_VALUE
COLUMN_ID	DATA_LENGTH	DENSITY
	DATA_PRECISION	NUM_NULLS
	DATA_SCALE	NUM_BUCKETS
	NULLABLE	LAST_ANALYZED
	DEFAULT_LENGTH	SAMPLE_SIZE
	DATA_DEFAULT	GLOBAL_STATS
	CHARACTER_SET_NAME	USER_STATS
	CHAR_COL_DECL_LENGTH	V80_FMT_IMAGE
	AVG_COL_LEN	DATA_UPGRADED

	CHAR_LENGTH	HISTOGRAM
	CHAR_USED	

Les colonnes « **OWNER** », « **TABLE_NAME** » et « **COLUMN_NAME** » contiennent l'utilisateur propriétaire des tables, les noms des tables et les colonnes. Les colonnes de définition sont décrites dans le module de création des objets de la base.

```
SQL> SELECT COLUMN_NAME,DATA_TYPE,DATA_LENGTH,
  2  DATA_PRECISION,DATA_DEFAULT
  3  FROM DBA_TAB_COLUMNS
  4  WHERE OWNER LIKE 'STAGIAIRE' AND
  5        TABLE_NAME LIKE 'EMPLOYES';

COLUMN_NAME     DATA_TYPE DATA_LENGTH DATA_PRECISION DATA_DEFAULT
--------------- --------- ----------- -------------- -------------
NO_EMPLOYE       NUMBER            22              6
REND_COMPTE      NUMBER            22              6
NOM              VARCHAR2          20
PRENOM           VARCHAR2          10
FONCTION         VARCHAR2          30
TITRE            VARCHAR2           5
DATE_NAISSANCE   DATE               7
DATE_EMBAUCHE    DATE               7                SYSDATE
SALAIRE          NUMBER            22              8
COMMISSION       NUMBER            22              8
```

La commande **SQL*Plus « DESCRIBE »** permet également d'obtenir les mêmes informations ; toutefois, elle ne permet pas de connaître les valeurs par défaut des colonnes ni leurs statistiques.

DBA_CONSTRAINTS

La vue du dictionnaire de données **« DBA_CONSTRAINTS »** vous permet d'afficher les attributs des contraintes. Elles sont très utiles pour modifier des contraintes ou résoudre des problèmes avec les données d'une application.

Il est essentiel de bien connaître les types de contraintes pour obtenir les informations adéquates.

```
SQL> SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, SEARCH_CONDITION
  2  FROM DBA_CONSTRAINTS
  3  WHERE OWNER = 'STAGIAIRE'
  4    AND TABLE_NAME = 'EMPLOYES';

CONSTRAINT_NAME                C SEARCH_CONDITION
------------------------------ - ----------------------------
SYS_C0011111                   C "NO_EMPLOYE" IS NOT NULL
SYS_C0011112                   C "NOM" IS NOT NULL
SYS_C0011113                   C "PRENOM" IS NOT NULL
SYS_C0011114                   C "FONCTION" IS NOT NULL
SYS_C0011115                   C "TITRE" IS NOT NULL
SYS_C0011116                   C "DATE_NAISSANCE" IS NOT NULL
```

```
SYS_C0011117                    C "DATE_EMBAUCHE" IS NOT NULL
SYS_C0011118                    C "SALAIRE" IS NOT NULL
PK_EMPLOYES                     P
FK_EMPLOYES_EMPLOYES            R
```

DBA_CONS_COLUMNS

La vue du dictionnaire de données « **DBA_CONS_COLUMNS** » vous permet d'afficher les attributs des colonnes associées à des contraintes.

```
SQL> SELECT C.CONSTRAINT_NAME, C.CONSTRAINT_TYPE,
  2         CC.POSITION, CC.COLUMN_NAME
  3  FROM DBA_CONSTRAINTS C, DBA_CONS_COLUMNS CC
  4  WHERE C.OWNER = CC.OWNER
  5    AND C.CONSTRAINT_NAME = CC.CONSTRAINT_NAME
  6    AND C.TABLE_NAME = CC.TABLE_NAME
  7    AND C.OWNER = 'STAGIAIRE'
  8    AND C.TABLE_NAME = 'EMPLOYES'
  9    AND CC.POSITION IS NOT NULL;

CONSTRAINT_NAME      C POSITION COLUMN_NAME
-------------------- - -------- ----------------
FK_EMPLOYES_EMPLOYES R        1 REND_COMPTE
PK_EMPLOYES          P        1 NO_EMPLOYE
```

Les partitions des tables

- **DBA_PART_TABLES**
- **DBA_PART_KEY_COLUMNS**
- **DBA_SUBPARTITION_TEMPLATES**
- **DBA_TAB_PARTITIONS**
- **DBA_TAB_SUBPARTITIONS**

DBA_PART_TABLES

La vue du dictionnaire de données « **DBA_PART_TABLES** » vous permet d'afficher les informations de partitionnement des tables pour toutes les tables partitionnées de la base.

```
SQL> DESC DBA_PART_TABLES
Nom                                       NULL ?   Type
 ----------------------------------- -------- ---------------
 OWNER                                          VARCHAR2(30)
 TABLE_NAME                                     VARCHAR2(30)
 PARTITIONING_TYPE                              VARCHAR2(9)
 SUBPARTITIONING_TYPE                           VARCHAR2(7)
 PARTITION_COUNT                                NUMBER
 DEF_SUBPARTITION_COUNT                         NUMBER
 PARTITIONING_KEY_COUNT                         NUMBER
 SUBPARTITIONING_KEY_COUNT                      NUMBER
 STATUS                                         VARCHAR2(8)
 DEF_TABLESPACE_NAME                            VARCHAR2(30)
 DEF_PCT_FREE                                   NUMBER
 DEF_PCT_USED                                   NUMBER
 DEF_INI_TRANS                                  NUMBER
 DEF_MAX_TRANS                                  NUMBER
 DEF_INITIAL_EXTENT                             VARCHAR2(40)
 DEF_NEXT_EXTENT                                VARCHAR2(40)
 DEF_MIN_EXTENTS                                VARCHAR2(40)
 DEF_MAX_EXTENTS                                VARCHAR2(40)
 DEF_MAX_SIZE                                   VARCHAR2(40)
```

```
 DEF_PCT_INCREASE                                  VARCHAR2(40)
 DEF_FREELISTS                                     NUMBER
 DEF_FREELIST_GROUPS                               NUMBER
 DEF_LOGGING                                       VARCHAR2(7)
 DEF_COMPRESSION                                   VARCHAR2(8)
 DEF_COMPRESS_FOR                                  VARCHAR2(18)
 DEF_BUFFER_POOL                                   VARCHAR2(7)
 REF_PTN_CONSTRAINT_NAME                           VARCHAR2(30)
 INTERVAL                                          VARCHAR2(1000)
```

DBA_PART_KEY_COLUMNS

La vue du dictionnaire de données « **DBA_PART_KEY_COLUMNS** » vous permet d'afficher les informations sur les clés de partitionnement des tables.

```
SQL> DESC DBA_PART_KEY_COLUMNS
Nom                                       NULL ?   Type
 ----------------------------------- -------- -----------------
 OWNER                                             VARCHAR2(30)
 NAME                                              VARCHAR2(30)
 OBJECT_TYPE                                       CHAR(5)
 COLUMN_NAME                                       VARCHAR2(4000)
 COLUMN_POSITION                                   NUMBER

SQL> SELECT T.TABLE_NAME, T.PARTITIONING_TYPE,
  2         T.SUBPARTITIONING_TYPE,C.COLUMN_NAME, C.COLUMN_POSITION
  3  FROM DBA_PART_TABLES T, DBA_PART_KEY_COLUMNS C
  4  WHERE T.OWNER = C.OWNER
  5    and T.TABLE_NAME = C.NAME
  6    and T.OWNER =  'STAGB';

TABLE_NAME        PARTITION SUBPART COLUMN_NAME  COLUMN_POSITION
----------------- --------- ------- ------------ ---------------
DETAILS_COMMANDES HASH      NONE    NO_COMMANDE                1
DETAILS_COMMANDES HASH      NONE    REF_PRODUIT                2
INDICATEURS       RANGE     LIST    COMMANDE                   1
```

DBA_SUBPARTITION_TEMPLATES

La vue du dictionnaire de données « **DBA_SUBPARTITION_TEMPLATES** » vous permet d'afficher les informations sur les modèles de sous-partitions des tables.

```
SQL> DESC DBA_SUBPARTITION_TEMPLATES
Nom                                       NULL ?   Type
 ----------------------------------- -------- ---------------
 USER_NAME                          NOT NULL VARCHAR2(30)
 TABLE_NAME                         NOT NULL VARCHAR2(30)
 SUBPARTITION_NAME                  NOT NULL VARCHAR2(34)
 SUBPARTITION_POSITION                       NUMBER
 TABLESPACE_NAME                             VARCHAR2(30)
 HIGH_BOUND                                  LONG
```

DBA_TAB_PARTITIONS

La vue du dictionnaire de données « **DBA_TAB_PARTITIONS** » vous permet d'afficher les informations au niveau de la partition sur les paramètres de stockage ainsi que les statistiques calculées pour cette partition.

```
SQL> DESC DBA_TAB_PARTITIONS
Nom                                       NULL ?   Type
 ----------------------------------- -------- ------------
 TABLE_OWNER                                       VARCHAR2(30)
 TABLE_NAME                                        VARCHAR2(30)
 COMPOSITE                                         VARCHAR2(3)
 PARTITION_NAME                                    VARCHAR2(30)
 SUBPARTITION_COUNT                                NUMBER
 HIGH_VALUE                                        LONG
 HIGH_VALUE_LENGTH                                 NUMBER
 PARTITION_POSITION                                NUMBER
 TABLESPACE_NAME                                   VARCHAR2(30)
 PCT_FREE                                          NUMBER
 PCT_USED                                          NUMBER
 INI_TRANS                                         NUMBER
 MAX_TRANS                                         NUMBER
 INITIAL_EXTENT                                    NUMBER
 NEXT_EXTENT                                       NUMBER
 MIN_EXTENT                                        NUMBER
 MAX_EXTENT                                        NUMBER
 MAX_SIZE                                          NUMBER
 PCT_INCREASE                                      NUMBER
 FREELISTS                                         NUMBER
 FREELIST_GROUPS                                   NUMBER
 LOGGING                                           VARCHAR2(7)
 COMPRESSION                                       VARCHAR2(8)
 COMPRESS_FOR                                      VARCHAR2(18)
 NUM_ROWS                                          NUMBER
 BLOCKS                                            NUMBER
 EMPTY_BLOCKS                                      NUMBER
 AVG_SPACE                                         NUMBER
 CHAIN_CNT                                         NUMBER
 AVG_ROW_LEN                                       NUMBER
 SAMPLE_SIZE                                       NUMBER
 LAST_ANALYZED                                     DATE
 BUFFER_POOL                                       VARCHAR2(7)
 GLOBAL_STATS                                      VARCHAR2(3)
 USER_STATS                                        VARCHAR2(3)

SQL> SELECT TABLE_NAME, PARTITION_NAME, SUBPARTITION_COUNT,
  2         NUM_ROWS, BLOCKS, SAMPLE_SIZE, LAST_ANALYZED
  3  FROM DBA_TAB_PARTITIONS
  4  WHERE TABLE_OWNER = 'STAGB'
```

```
  5    AND TABLE_NAME  = 'INDICATEURS'
  6    AND PARTITION_NAME  in ('IND_2007', 'IND_2008');

TABLE_NAME  PARTITIO SUBP NUM_ROWS   BLOCKS     SAMPLE_SIZE LAST_ANA
----------- -------- ---- ---------- ---------- ----------- --------
INDICATEURS IND_2007   26      75395        934       75395 18/07/09
INDICATEURS IND_2008   26      68423        888       68423 18/07/09
```

DBA_TAB_SUBPARTITIONS

La vue du dictionnaire de données « **DBA_TAB_SUBPARTITIONS** » vous permet d'afficher les informations au niveau de la sous-partition sur les paramètres de stockage ainsi que les statistiques calculées pour cette partition.

Les statistiques des tables

- **DBA_TAB_STATISTICS**
- **DBA_TAB_COL_STATISTICS**
- **DBA_TAB_STAT_PREFS**
- **DBA_TAB_HISTOGRAMS**
- **DBA_PART_HISTOGRAMS**
- **DBA_SUBPART_HISTOGRAMS**
- **DBA_TAB_MODIFICATIONS**

DBA_TAB_STATISTICS

La vue du dictionnaire de données « **DBA_TAB_STATISTICS** » vous permet d'afficher les informations concernant les statistiques calculées pour toutes les tables dans la base de données.

```
SQL> DESC DBA_TAB_STATISTICS
Nom                                       NULL ?   Type
 ----------------------------------- -------- ---------------
 OWNER                                          VARCHAR2(30)
 TABLE_NAME                                     VARCHAR2(30)
 PARTITION_NAME                                 VARCHAR2(30)
 PARTITION_POSITION                             NUMBER
 SUBPARTITION_NAME                              VARCHAR2(30)
 SUBPARTITION_POSITION                          NUMBER
 OBJECT_TYPE                                    VARCHAR2(12)
 NUM_ROWS                                       NUMBER
 BLOCKS                                         NUMBER
 EMPTY_BLOCKS                                   NUMBER
 AVG_SPACE                                      NUMBER
 CHAIN_CNT                                      NUMBER
 AVG_ROW_LEN                                    NUMBER
 AVG_SPACE_FREELIST_BLOCKS                      NUMBER
 NUM_FREELIST_BLOCKS                            NUMBER
 AVG_CACHED_BLOCKS                              NUMBER
 AVG_CACHE_HIT_RATIO                            NUMBER
 SAMPLE_SIZE                                    NUMBER
 LAST_ANALYZED                                  DATE
```

```
GLOBAL_STATS                                       VARCHAR2(3)
USER_STATS                                         VARCHAR2(3)
STATTYPE_LOCKED                                    VARCHAR2(5)
STALE_STATS                                        VARCHAR2(3)
```

DBA_TAB_COL_STATISTICS

La vue du dictionnaire de données « **DBA_TAB_COL_STATISTICS** » vous permet d'afficher les informations concernant les statistiques calculées informations au niveau de la colonne pour toutes les tables dans la base de données.

```
SQL> DESC DBA_TAB_COL_STATISTICS
 Nom                                       NULL ?   Type
 ----------------------------------------- -------- ----------------
 OWNER                                              VARCHAR2(30)
 TABLE_NAME                                         VARCHAR2(30)
 COLUMN_NAME                                        VARCHAR2(30)
 NUM_DISTINCT                                       NUMBER
 LOW_VALUE                                          RAW(32)
 HIGH_VALUE                                         RAW(32)
 DENSITY                                            NUMBER
 NUM_NULLS                                          NUMBER
 NUM_BUCKETS                                        NUMBER
 LAST_ANALYZED                                      DATE
 SAMPLE_SIZE                                        NUMBER
 GLOBAL_STATS                                       VARCHAR2(3)
 USER_STATS                                         VARCHAR2(3)
 AVG_COL_LEN                                        NUMBER
 HISTOGRAM                                          VARCHAR2(15)

SQL> SELECT T.TABLE_NAME, T.NUM_ROWS, C.COLUMN_NAME, C.NUM_DISTINCT,
  2         C.NUM_BUCKETS, C.HISTOGRAM
  3  FROM DBA_TAB_STATISTICS T, DBA_TAB_COL_STATISTICS C
  4  WHERE T.OWNER       = C.OWNER
  5    AND T.TABLE_NAME  = C.TABLE_NAME
  6    AND T.OWNER       = 'STAGB'
  7    AND T.TABLE_NAME  = 'COMMANDES';

TABLE_NAM   NUM_ROWS COLUMN_NAME   NUM_ NUM_BUCKETS HISTOGRAM
--------- ---------- ------------- ---- ----------- ----------
COMMANDES      18552 ANNULEE          2           2 FREQUENCY
COMMANDES      18552 ACQUITEE         2           2 FREQUENCY
COMMANDES      18552 LIVREE           2           2 FREQUENCY
COMMANDES      18552 PORT           501           1 NONE
COMMANDES      18552 DATE_ENVOI    1122           1 NONE
COMMANDES      18552 DATE_COMMANDE 1096           1 NONE
COMMANDES      18552 NO_EMPLOYE      92          92 FREQUENCY
COMMANDES      18552 CODE_CLIENT     91          91 FREQUENCY
COMMANDES      18552 NO_COMMANDE  18552           1 NONE
```

DBA_TAB_HISTOGRAMS

Les statistiques des colonnes sont conservées sous forme d'histogrammes. Ces histogrammes donnent des estimations précises de la répartition des valeurs stockées dans la colonne. Grace aux histogrammes l'optimiseur Oracle fournit une meilleure sélectivité dans le choix des plans d'exécution. La vue du dictionnaire de données « **DBA_TAB_HISTOGRAMS** » ainsi que « **DBA_PART_HISTOGRAMS** » « **DBA_SUBPART_HISTOGRAMS** » vous permet d'afficher les informations concernant les histogrammes sur la table.

```
SQL> DESC DBA_TAB_HISTOGRAMS
 Nom                                       NULL ?   Type
 ----------------------------------------- -------- ----------------
 OWNER                                              VARCHAR2(30)
 TABLE_NAME                                         VARCHAR2(30)
 COLUMN_NAME                                        VARCHAR2(4000)
 ENDPOINT_NUMBER                                    NUMBER
 ENDPOINT_VALUE                                     NUMBER
 ENDPOINT_ACTUAL_VALUE                              VARCHAR2(1000)

SQL> DESC DBA_PART_HISTOGRAMS
 Nom                                       NULL ?   Type
 ----------------------------------------- -------- ----------------
 OWNER                                              VARCHAR2(30)
 TABLE_NAME                                         VARCHAR2(30)
 PARTITION_NAME                                     VARCHAR2(30)
 COLUMN_NAME                                        VARCHAR2(4000)
 BUCKET_NUMBER                                      NUMBER
 ENDPOINT_VALUE                                     NUMBER
 ENDPOINT_ACTUAL_VALUE                              VARCHAR2(1000)
```

DBA_TAB_MODIFICATIONS

La vue du dictionnaire de données « **DBA_TAB_MODIFICATIONS** » vous permet d'afficher les informations concernant les opérations effectuées sur la table depuis la dernière fois que les statistiques ont été recueillies.

```
SQL> DESC DBA_TAB_MODIFICATIONS
Nom                                       NULL ?   Type
 ----------------------------------------- -------- --------------
 TABLE_OWNER                                        VARCHAR2(30)
 TABLE_NAME                                         VARCHAR2(30)
 PARTITION_NAME                                     VARCHAR2(30)
 SUBPARTITION_NAME                                  VARCHAR2(30)
 INSERTS                                            NUMBER
 UPDATES                                            NUMBER
 DELETES                                            NUMBER
 TIMESTAMP                                          DATE
 TRUNCATED                                          VARCHAR2(3)
 DROP_SEGMENTS                                      NUMBER

SQL> SELECT TABLE_NAME, INSERTS, UPDATES, DELETES
```

```
  2  FROM DBA_TAB_MODIFICATIONS
  3  WHERE TABLE_OWNER = 'STAGB'
  4    AND TABLE_NAME  = 'COMMANDES';

TABLE_NAM    INSERTS    UPDATES    DELETES
--------- ---------- ---------- ----------
COMMANDES       9874      18552       358
```

Les index

- DBA_INDEXES
- DBA_IND_COLUMNS
- DBA_CLUSTERS
- DBA_CLU_COLUMNS

DBA_INDEXES

La vue du dictionnaire de données « **DBA_INDEXES** » vous permet d'afficher tous les index de la base.

DBA_IND_COLUMNS

La vue du dictionnaire de données « **DBA_IND_COLUMNS** » vous permet de déterminer les colonnes qui font partie d'un index.

DBA_CLUSTERS

La vue du dictionnaire de données « **DBA_CLUSTERS** » vous permet d'afficher les paramètres de stockage et de statistiques associés aux clusters.

DBA_CLU_COLUMNS

La vue du dictionnaire de données « **DBA_CLU_COLUMNS** » vous permet de savoir quelles colonnes de tables font partie d'un cluster.

Les objets utilisateur

- DBA_VIEWS
- DBA_SYNONYMS
- DBA_SEQUENCES
- DBA_MVIEWS
- DIMENSIONS
- DBA_TYPES
- DBA_LOBS
- DBA_DBLINK
- DBA_RECYCLEBIN

DBA_VIEWS

La vue du dictionnaire de données « **DBA_VIEWS** » affiche les informations sur les vues traditionnelles.

DBA_SYNONYMS

La vue du dictionnaire de données « **DBA_SYNONYMS** » vous permet d'afficher les attributs des synonymes.

DBA_SEQUENCES

La vue du dictionnaire de données « **DBA_SEQUENCES** » vous permet d'afficher les attributs de séquences.

DBA_MVIEWS

La vue du dictionnaire de données « **DBA_MVIEWS** » vous permet d'afficher les informations sur les vues matérialisées.

DIMENSIONS

La vue du dictionnaire de données « **DIMENSIONS** » vous permet d'afficher les dimensions et les hiérarchies de la base de données.

DBA_TYPES

La vue du dictionnaire de données « **DBA_TYPES** » vous permet d'afficher la liste des types de données abstraits.

DBA_LOBS

La vue du dictionnaire de données « **DBA_LOBS** » vous permet d'afficher les informations sur les grands objets, LOB, stockés dans les tables de la base de données.

DBA_DBLINK

La vue du dictionnaire de données « **DBA_DBLINK** » vous permet d'afficher les liens de base de données.

DBA_RECYCLEBIN

La vue du dictionnaire de données « **DBA_RECYCLEBIN** » vous permet d'afficher les attributs des objets qui peuvent être récupérés à l'aide du paquetage « **DBMS_FLASHBACK** ». La base de données peut récupérer uniquement les objets qui ont été effacés pas les objets tronqués.

La structure de stockage

- DBA_TABLESPACES
- DBA_DATA_FILES
- DBA_TS_QUOTAS
- DBA_SEGMENTS
- DBA_EXTENTS

Vous pouvez utiliser le dictionnaire de données pour déterminer l'espace disponible et l'espace alloué aux objets de la base de données. Les principales vues qui décrivent comment déterminer les paramètres de stockage par défaut des objets, les quotas d'utilisation de l'espace, l'espace libre disponible et la façon dont les objets sont stockés physiquement sont énumérés dans cette partie. Pour la description complète de ces différentes vues, rapportez-vous au module correspondant au stockage.

DBA_TABLESPACES

La vue du dictionnaire de données « **DBA_TABLESPACES** » vous permet d'afficher les espaces de disques logiques et les paramètres de stockage de chacun d'eux.

DBA_DATA_FILES

La vue du dictionnaire de données « **DBA_DATA_FILES** » vous permet d'afficher les fichiers de données ainsi que les espaces de disques logiques auxquels il appartient.

DBA_TS_QUOTAS

La vue du dictionnaire de données « **DBA_TS_QUOTAS** » vous permet d'afficher les quotas de stockage de tous les espaces de disques logiques ; elle se révèle très efficace pour déterminer l'utilisation de l'espace dans l'ensemble de la base de données.

DBA_SEGMENTS

La vue du dictionnaire de données « **DBA_SEGMENTS** » vous permet d'afficher les paramètres de stockage et l'utilisation d'espace pour les segments dans la base de données.

DBA_EXTENTS

La vue du dictionnaire de données « **DBA_EXTENTS** » vous permet d'afficher les paramètres de stockage et l'utilisation d'espace pour les extents des segments.

Les utilisateurs et privilèges

- DBA_USERS
- DBA_ROLES
- DBA_SYS_PRIVS
- DBA_TAB_PRIVS
- DBA_COL_PRIVS
- DBA_ROLE_PRIVS

Les utilisateurs et leurs privilèges sont enregistrés dans le dictionnaire de données. Les principales vues qui décrivent comment obtenir des informations sur les comptes d'utilisateurs, les limites de ressources et les privilèges des utilisateurs, sont énumérées dans cette partie. Pour la description complète de ses différentes vues, rapportez-vous au module correspondant à la gestion des utilisateurs.

DBA_USERS

La vue du dictionnaire de données « **DBA_USERS** » vous permet d'afficher la liste de tous les comptes utilisateurs de la base de données. Elle est utile pour connaître les noms d'utilisateurs disponibles.

DBA_ROLES

La vue du dictionnaire de données « **DBA_ROLES** » vous permet d'afficher les rôles assignés à un utilisateur. Les rôles octroyés au groupe PUBLIC sont également listés dans cette vue.

DBA_SYS_PRIVS

La vue du dictionnaire de données « **DBA_SYS_PRIVS** » vous permet d'afficher les privilèges système octroyés directement à un utilisateur.

DBA_TAB_PRIVS

La vue du dictionnaire de données « **DBA_TAB_PRIVS** » vous permet d'afficher la liste des privilèges d'objets accordés à tous les utilisateurs de la base.

DBA_COL_PRIVS

La vue du dictionnaire de données « **DBA_COL_PRIVS** » vous permet d'afficher tous les privilèges de colonnes octroyés aux utilisateurs de la base.

DBA_ROLE_PRIVS

La vue du dictionnaire de données « **DBA_ROLE_PRIVS** » vous permet d'afficher tous les rôles octroyés aux utilisateurs de la base.

Les audits

- DBA_AUDIT_TRAIL
- DBA_AUDIT_SESSION
- DBA_AUDIT_OBJECT
- DBA_OBJ_AUDIT_OPTS
- DBA_AUDIT_STATEMENT

Dans une base Oracle, on peut activer les fonctionnalités d'audit ; une fois ces fonctionnalités activées, plusieurs vues du dictionnaire de données permettent à tout utilisateur d'accéder au journal d'audit.

DBA_AUDIT_TRAIL

La vue du dictionnaire de données « **DBA_AUDIT_TRAIL** » vous permet d'afficher toutes les entrées de la table de suivi d'audit.

DBA_AUDIT_SESSION

La vue du dictionnaire de données « **DBA_AUDIT_SESSION** » vous permet d'afficher les entrées de la table de suivi d'audit pour les connexions et déconnexions.

DBA_AUDIT_OBJECT

La vue du dictionnaire de données « **DBA_AUDIT_OBJECT** » vous permet d'afficher les entrées de la table de suivi d'audit pour les instructions concernant les objets.

DBA_OBJ_AUDIT_OPTS

La vue du dictionnaire de données « **DBA_OBJ_AUDIT_OPTS** » vous permet d'afficher les entrées de la table de suivi d'audit pour les options d'audit appliquées aux objets.

DBA_AUDIT_STATEMENT

La vue du dictionnaire de données « **DBA_AUDIT_STATEMENT** » vous permet d'afficher les entrées de la table de suivi d'audit pour les commandes « **GRANT** », « **REVOKE** », « **AUDIT** », « **NOAUDIT** » et « **ALTER SYSTEM** » exécutées par un utilisateur.

Atelier 8

Questions

8-1 Quelle est la vue du dictionnaire de données qui vous permet d'afficher la liste de tous les utilisateurs de la base de données et leurs caractéristiques ?

A. DBA_USERS

B. USER_USER

C. ALL_USER

D. V$SESSION

8-2 Quelle est la vue qui vous permet d'afficher le nom de toutes les vues du dictionnaire de données ?

A. DBA_NAMES

B. DBA_TABLES

C. DBA_DICTIONARY

D. DICTIONARY

Exercice n°1

Créez une requête qui interroge la vue du dictionnaire de données « **`DICTIONARY`** ». Elle doit utiliser une variable de substitution pour récupérer uniquement les enregistrements qui correspondent. Le filtre porte sur le nom ou une partie du nom d'une ou plusieurs vues du dictionnaire de données.

Exercice n°2

Affichez l'ensemble des utilisateurs de la base de données ainsi que la date de création de leurs comptes.

- *CREATE TABLESPACE*
- *DB_Nx_CACHE_SIZE*
- *TEMPORARY*
- *UNDO*

15

Les tablespaces

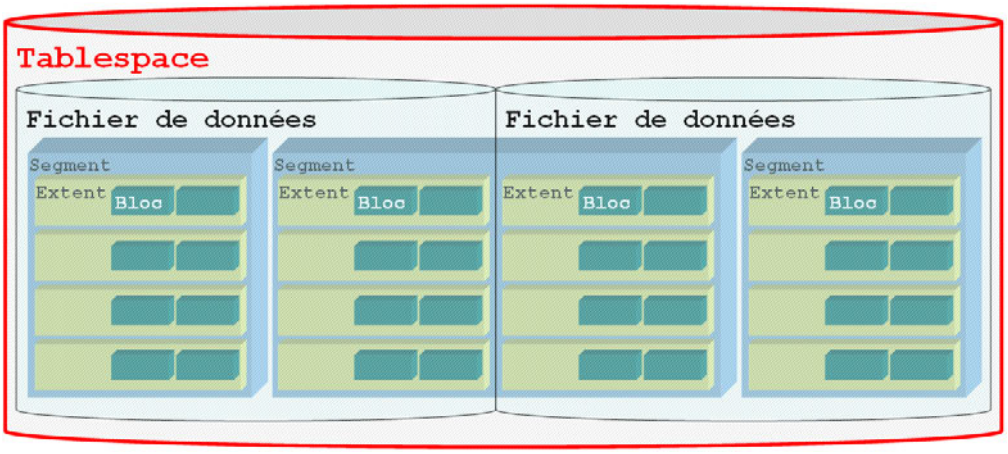
Objectifs

À la fin de ce module, vous serez à même d'effectuer les tâches suivantes :

- Créer un espace logique de stockage permanent.
- Définir le tablespace permanent par défaut pour une base de données.
- Créer un tablespace de type temporaire.
- Créer un groupe de tablespaces temporaires.
- Définir le tablespace temporaire par défaut pour la base de données.
- Créer un tablespace utilisant un stockage crypté.
- Créer un tablespace avec une taille de bloc autre que celle par défaut.

Contenu

Le tablespace



La séparation des structures logique et physique d'une base de données facilite le contrôle poussé de la gestion de l'espace disque. L'administrateur peut configurer les paramètres d'allocation d'espace aux composants physiques et logiques de la base de données.

Pour utiliser efficacement l'espace du disque dur, il est important de connaître les relations entre les composants physiques et logiques de la base de données. Il est important également de savoir comment l'espace est alloué dans la base de données.

Comme on l'a vu précédemment, la base de données est divisée en zones d'espace logique plus petites, appelées tablespaces.

Note

Un tablespace est constitué d'un ou plusieurs fichiers de données (datafiles) ; par contre un fichier de données ne peut appartenir qu'à un seul tablespace.

Un fichier de données est créé automatiquement par le serveur Oracle chaque fois que vous créez le tablespace ou que vous avez ajouté un nouveau fichier au tablespace. La quantité de disque occupé par le fichier de données est spécifiée par l'administrateur de la base de données. Chaque fois que le fichier est créé, l'espace est automatiquement réservé sur disque.

Après la création d'un tablespace, vous pouvez ajouter d'autres fichiers de données. Un fichier de données peut être modifié par l'administrateur de la base de données après sa création.

Dés lors qu'un fichier de données est créé pour un tablespace, il est attaché à ce tablespace, et il va pouvoir être détaché uniquement par la destruction du tablespace.

Un tablespace est constitué de segments. Un segment est l'espace alloué pour un type spécifique de structure de stockage logique dans un tablespace. Les segments d'index, segments temporaires, undo segments et segments de données représentent quelques exemples de segments. Un segment, tel qu'un segment de données, peut être réparti sur plusieurs fichiers appartenant au même tablespace.

Le niveau suivant de la structure logique d'une base de données est l'extent. Un extent est un ensemble de blocs contigus. Chaque segment est constitué d'un ou plusieurs extents. Un extent ne peut pas être stocké sur plusieurs fichiers de données.

Note

Un segment peut être stocké sur un ou plusieurs fichiers appartenant au même tablespace.

Par contre un extent ne peut pas être stocké sur plusieurs fichiers de données ; il doit être absolument contenu dans le même fichier de données.

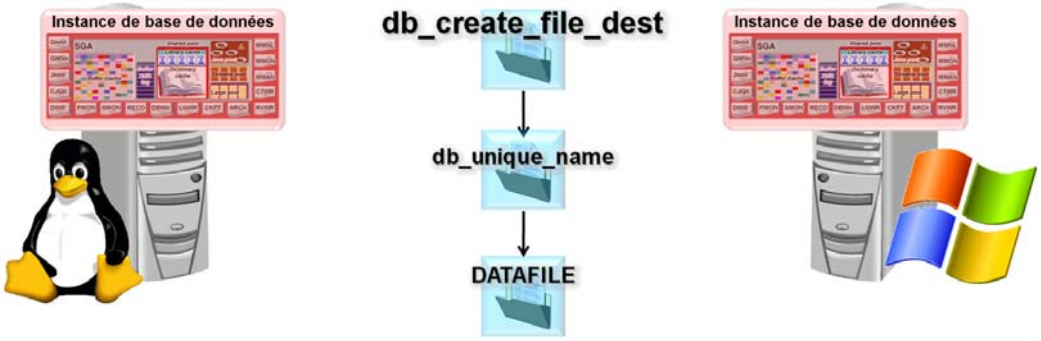
Les blocs de données constituent le dernier niveau de granularité. Les données d'une base de données Oracle sont stockées dans les blocs de données. Un bloc de données correspond à un ou plusieurs blocs de fichiers physiques alloués à partir de fichiers de données existants.

Dans ce module, nous allons voir la gestion des tablespaces et des fichiers de données ; la partie stockage est détaillée dans le module « La gestion du stockage ».

Attention

Un tablespace est un conteneur qui n'a pas de concept de propriété d'objet. Il n'y a aucune relation d'appartenance entre un tablespace et un propriétaire de structure (ou un propriétaire de table). Les objets possédés par un utilisateur peuvent résider dans de multiples tablespaces ou dans un même tablespace.

L'emplacement des fichiers



	hostname	db_name	hostname	db_name
db_create_file_dest	terra	saphir	gaia	ambre
	/u02/donnees/oradata		W:\donnees\oradata	
	saturne	rubis	cronos	jaspe
	+GD_DONNEES		+GD_DONNEES	

La valeur du paramètre « **DB_CREATE_FILE_DEST** » est le nom d'un répertoire existant indiquant à Oracle où créer les fichiers de données et les fichiers temporaires. Chaque fois que vous voulez créer un fichier de données, Oracle, crée automatiquement dans ce répertoire un sous-répertoire avec le nom du paramètre « **DB_UNIQUE_NAME** » s'il n'existe pas déjà. Ensuite, pour stocker le fichier, il crée un autre sous-répertoire « **DATAFILE** » pour les fichiers de données ou pour les fichiers temporaires.

Pour les fichiers stockés dans des répertoires de votre système de fichiers, le format du nom du fichier est le suivant :

« **o1_mf_%t_%u_.dbf** » ou « **o1_mf_%t_%u_.tmp** »

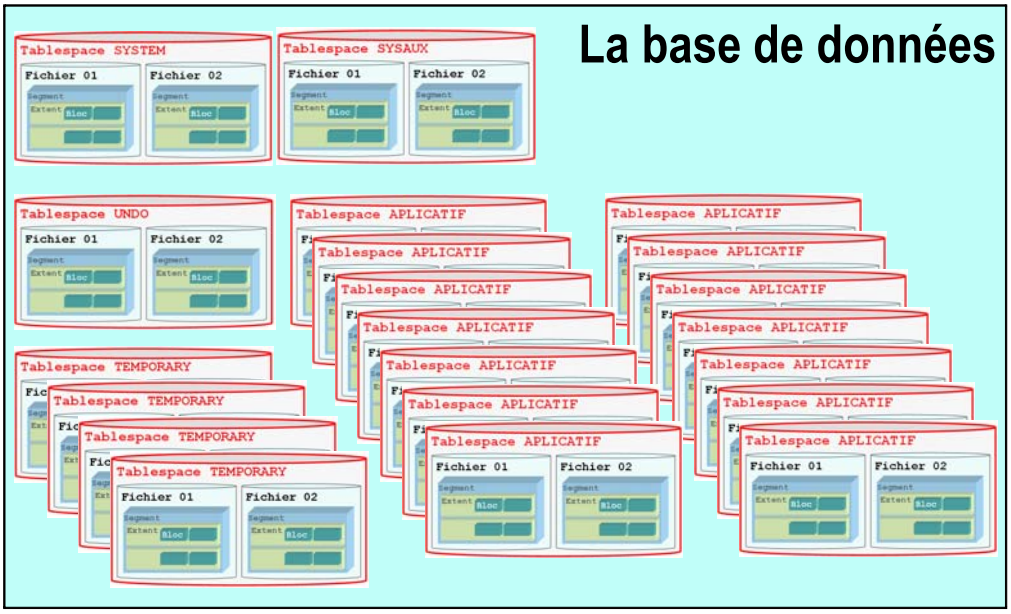
Pour les fichiers stockés dans des groupes de disques, le format du nom du fichier est le suivant : « **nom.fichier.incarnation** ».

Pour les exemples de créations des tablespaces, quatre machines avec deux systèmes d'exploitation Linux et Windows sont utilisées. Dans l'environnement Linux vous vous avez la machine « **terra** » avec la base de données « **saphire** » et la machine « **saturne** » avec la base de données « **rubis** ». Dans l'environnement Windows les deux machines sont « **gaia** » avec la base de données « **ambre** » et « **cronos** » avec sa base de données « **jaspe** ».

Dans l'image précédente vous pouvez voir les valeurs du paramètre « **DB_CREATE_FILE_DEST** » pour chacune de ces bases de données. Ainsi vous pouvez remarquer que « **rubis** » et « **jaspe** » sont stockées chacune dans une groupe de disques d'une instance ASM installée sur les machines respectives.

L'utilisation de plusieurs base de données sur des environnements indépendants montre que la gestion **OMF** (**O**racle **M**anaged **F**iles) utilise les mêmes syntaxes de création et de modification des tablespaces quel que soit l'environnement ou le type stockage de la base.

Les types de tablespaces



Chaque base de données Oracle créée possède un tablespace « **SYSTEM** ». Il s'agit de l'emplacement où Oracle garde toute l'information du catalogue exigée pour le fonctionnement de la base de données.

10g

À partir de la version Oracle 10*g*, un deuxième tablespace « **SYSAUX** » est créé automatiquement à la création de la base de données. Il contient les objets système complémentaires qui permettent ainsi de diminuer le temps d'attente pour les lectures des informations système.

Les deux tablespaces « **SYSTEM** » et « **SYSAUX** » sont utilisés uniquement par Oracle ; il faut prendre soin de ne pas stocker des objets utilisateur dans ces deux tablespaces.

Lors d'importantes opérations de tri (telles que select distinct, union et create index), si la taille de la zone **PGA** du processus utilisateur ne suffit pas, Oracle va stocker dans les tablespaces de la base de données des informations concernant le tri des enregistrements avant de retourner l'information aux utilisateurs. En raison de leur nature dynamique, ces espaces de tris ne devraient pas être stockés avec d'autres types de segments.

Lorsqu'un tablespace temporaire « **TEMPORARY** » est défini, un segment de tri est aussi créé. Celui-ci est capable de croître si nécessaire afin de pouvoir héberger toutes les opérations de tri de données, et il existe jusqu'à ce que la base de données soit fermée puis redémarrée.

Lors de la création de la base de données, vous avez la possibilité de définir un tablespace avec la clause « **DEFAULT TEMPORARY TABLESPACE** » de la commande « **CREATE DATABASE** ».

Certains utilisateurs d'une base de données Oracle peuvent avoir besoin de volumes de stockage temporaires beaucoup plus grands que ceux de tous les autres utilisateurs de l'application. Dans ce cas, vous pouvez créer plusieurs tablespaces temporaires « **TEMPORARY** », pour distribuer les espaces de stockages des utilisateurs ayant des besoins semblables sur les mêmes tablespaces.

Toutes les données d'annulation sont stockées dans un tablespace spécial appelé « **UNDO** ». Lorsque vous créez un tablespace « **UNDO** », Oracle gère le stockage, la rétention et l'emploi de l'espace pour les données de rollback par l'intermédiaire de la fonction SMU (System-Managed Undo). Aucun objet permanent n'est placé dans le tablespace **UNDO**.

Rappelez-vous, la syntaxe SQL de création de la base de données comporte d'abord la création d'un tablespace « **SYSTEM** » et d'un tablespace « **SYSAUX** », ainsi que la création des tablespaces « **TEMP** » et « **UNDO** ».

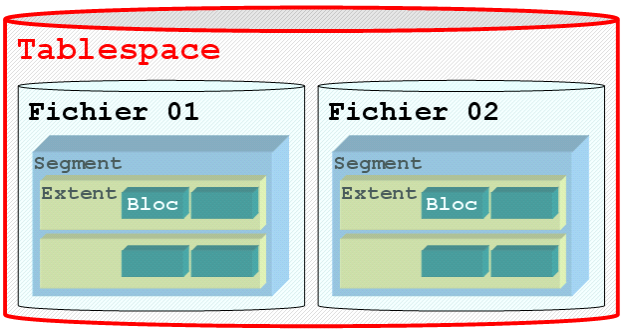
```
SQL> CREATE DATABASE "tpdba"
  2         MAXINSTANCES 8
  3         MAXLOGHISTORY 1
  4         MAXLOGFILES 24
  5         MAXLOGMEMBERS 2
  6         MAXDATAFILES 1024
  7  DATAFILE SIZE 300M AUTOEXTEND ON
  8           NEXT 10240K
  9           MAXSIZE UNLIMITED
 10           EXTENT MANAGEMENT LOCAL
 11  SYSAUX DATAFILE SIZE 120M AUTOEXTEND ON
 12           NEXT 10240K
 13           MAXSIZE UNLIMITED
 14  DEFAULT TEMPORARY TABLESPACE TEMP TEMPFILE SIZE 20M
 15          AUTOEXTEND ON NEXT 640K MAXSIZE UNLIMITED
 16  UNDO TABLESPACE "UNDOTBS1" DATAFILE SIZE 200M
 17          AUTOEXTEND ON NEXT 5120K MAXSIZE UNLIMITED
 18  CHARACTER SET WE8MSWIN1252
 19  NATIONAL CHARACTER SET AL16UTF16
 20  LOGFILE GROUP 1 SIZE 10240K,
 21         GROUP 2 SIZE 10240K,
 22         GROUP 3 SIZE 10240K
 23  USER SYS IDENTIFIED BY "&&sysPassword"
 24  USER SYSTEM IDENTIFIED BY "&&systemPassword";
```

Un tablespace applicatif typique contient tous les objets principaux associés à une application. L'important volume de lectures-écritures dont elles font l'objet justifie l'isolation de ces tables dans leur propre tablespace, séparant ainsi leurs fichiers de données des autres fichiers de données dans la base. La répartition de ces fichiers sur des disques différents peut de plus améliorer les performances (grâce à une réduction de la contention lors des accès au disque) et simplifier leur gestion.

Les index ne devraient pas être stockés dans le même tablespace que les tables de données sur lesquelles ils ont été définis, car ces deux types de structures font l'objet de nombreuses opérations de lectures-écritures concurrentes lors des manipulations de données et des interrogations.

La création d'un tablespace

```
CREATE SMALLFILE TABLESPACE APPLICATION_01
DATAFILE
      SIZE 1G AUTOEXTEND ON NEXT 1G MAXSIZE 5G,
      SIZE 1G AUTOEXTEND ON NEXT 1G MAXSIZE 5G;
```



À la création d'un tablespace, vous indiquez le ou les fichiers de données, leur taille, leur mode d'agrandissement. La création du tablespace permet également de configurer la taille du bloc, les informations de journalisation ou les options de compression… ; ces options seront détaillées au fur et à mesure que nous avancerons dans ce module.

L'instruction SQL qui permet de créer un tablespace permanent est :

```
CREATE {BIGFILE|SMALLFILE} TABLESPACE nom_tablespace
       [ DATAFILE
         ['nom_fichier'] [ SIZE integer {K|M|G|T} ]
         [ AUTOEXTEND
            {OFF |
             ON [ NEXT integer {K|M|G|T}]
                [ MAXSIZE {UNLIMITED | integer {K|M|G|T}]
             }
          ] [,...]
       ]
       [BLOCKSIZE integer [ K ]]
       [{LOGGING | NOLOGGING}]
       [FORCE LOGGING]
       [{ONLINE | OFFLINE}]
       [FLASHBACK {ON | OFF}]
       [ENCRYPTION [ USING 'algorithm']
                   [IDENTIFIED BY mot_de_passe]
                     DEFAULT STORAGE ENCRYPT]
```

```
        [ DEFAULT { COMPRESS
           { BASIC
            | FOR { OLTP
                   |{ QUERY | ARCHIVE } [ LOW | HIGH ]}
          | NOCOMPRESS}] ;
```

BIGFILE	Indique que le tablespace est créé avec un seul fichier pouvant contenir jusqu'à 2^{32} blocs. Pour un tablespace d'un bloc de 8 k, vous pouvez stocker jusqu'à 32 TB.
SMALLFILE	Indique que le tablespace peut avoir un ou plusieurs fichiers de données. Aucun fichier ne peut contenir plus de 2^{22} blocs. Pour un tablespace d'un bloc de 8 k, chaque fichier peut stocker jusqu'à 32 GB.
nom_fichier	Le ou les fichiers de données qui constituent le tablespace.
AUTOEXTEND	L'argument active ou désactive l'extension automatique d'un nouveau fichier de données ou temporaire. Si vous omettez cette clause, ces fichiers ne seront pas automatiquement étendus.
integer	Spécifie une taille en octets si vous ne précisez pas de suffixe pour définir une valeur en K, M, G, T.
K	Valeurs spécifiées pour préciser la taille en kilooctets.
M	Valeurs spécifiées pour préciser la taille en mégaoctets.
G	Valeurs spécifiées pour préciser la taille en gigaoctets.
T	Valeurs spécifiées pour préciser la taille en téraoctets.
AUTOEXTEND	Active ou désactive l'extension automatique du fichier de données, d'un nouveau fichier de données, ou d'un fichier temporaire. Si vous ne spécifiez pas cette clause, ces fichiers ne sont pas automatiquement étendus.
NEXT	Définit la taille en octets du prochain incrément d'espace disque qui doit être alloué automatiquement au fichier lorsque davantage d'espace de stockage est requis.
MAXSIZE	Définit l'espace disque maximal autorisé pour l'extension automatique du fichier de données.
UNLIMITED	Définit une allocation d'espace illimitée pour le fichier.
BLOCKSIZE	Indique une taille de bloc non standard pour le tablespace.
LOGGING	Définit que la base de données effectue les journalisations pour toutes les opérations sur tous les index, tables et partitions contenus dans le tablespace. L'attribut de journalisation de niveau tablespace peut être modifié par les spécifications de journalisation au niveau table, index ou partition.
NOLOGGING	Définit que la base de données n'effectue pas de journalisation pour les « **INSERT /*+APPEND*/** » ou les chargements des données à l'aide de SQL*Loader ainsi que les opérations DDL sur toutes les créations d'index, tables et partitions contenues dans le tablespace. Les modifications de tables avec les options « **MOVE** » ou « **SPLIT** ».

FORCE LOGGING	Force le travail en mode « **LOGGING** » pour toutes les opérations de la base de données, même si l'opération concernée est effectuée dans le mode « **NOLOGGING** ».
ONLINE	Permet de créer un tablespace mis à la disposition des utilisateurs qui ont reçu le droit d'y accéder immédiatement après sa création. Il s'agit du choix par défaut
OFFLINE	Permet de créer un tablespace dans un état indisponible immédiatement après sa création.
FLASHBACK	Indique que le tablespace peut être utilisé dans des opérations de récupération de type « **FLASHBACK** ».
ENCRYPTION	Indique que le tablespace stocke les informations cryptées dans ces fichiers. Cette option doit être accompagnée de la clause de stockage : « **DEFAULT STORAGE ENCRYPT** ».
COMPRESS	Indique que le stockage de segments dans le tablespace est compressé. Pour plus d'informations sur le stockage des segments, voir le module « La gestion du stockage ».

```
SYS@saphir>create tablespace tbsdata datafile
  2  size 10m autoextend on next 10m maxsize 1g,
  3  size 10m autoextend on next 10m maxsize 1g
  4  force logging online flashback on;

Tablespace créé.

SYS@saphir>select file_name
  2  from dba_data_files
  3  where tablespace_name = 'TBSDATA';

FILE_NAME
--------------------------------------------------------------------------------
/u02/donnees/oradata/SAPHIR/datafile/o1_mf_tbsdata_63mdggwt_.dbf
/u02/donnees/oradata/SAPHIR/datafile/o1_mf_tbsdata_63mdgh16_.dbf

SYS@saphir>!ls -l /u02/donnees/oradata/SAPHIR/datafile
...
-rw-r----- 1 oracle oinstall  10493952 jui 11 13:58
o1_mf_tbsdata_63mdggwt_.dbf
-rw-r----- 1 oracle oinstall  10493952 jui 11 13:58
o1_mf_tbsdata_63mdgh16_.dbf
...
```

Dans l'exemple précédent, vous pouvez remarquer la création d'un tablespace avec deux fichiers de données, chacun d'une taille de 10 M. L'espace de stockage défini pour chaque fichier de 10 Mb est réservé sous sur disque.

Attention

À la création du tablespace, il faut veiller à ce que les tailles des fichiers mentionnés ne dépassent pas l'espace libre qui se trouve sur disque.

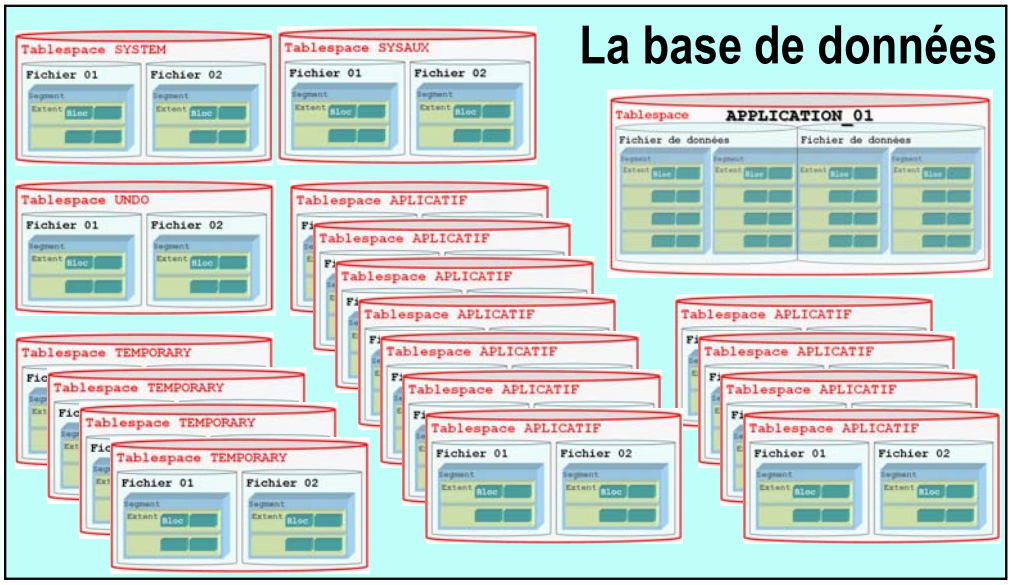
En effet Oracle réserve automatiquement l'espace précisé pour chaque fichier de données. S'il n'y a pas assez d'espace sur disque, Oracle projette la création du tablespace.

Vous pouvez également définir si le tablespace créé est accessible tout de suite après la création, comme celui créé dans l'exemple précédent. Le tablespace créé « **OFFLINE** » n'est pas accessible immédiatement après sa création ; il faut modifier son état pour pouvoir travailler avec un tel type de tablespace.

```
SYS@saphir>create table table_01 tablespace tbsdata
  2  as select * from cat;

Table créée.

SYS@saphir>create tablespace tbsdata02 datafile
  2  size 10m autoextend on next 10m maxsize 1g
  3  force logging offline flashback on;

Tablespace créé.

SYS@saphir>create table table_02 tablespace tbsdata02 as
  2  select * from cat;
select * from cat
              *
ERREUR à la ligne 2 :
ORA-01542: tablespace 'TBSDATA02' hors ligne ; impossible de lui
affecter de l'espace

SYS@saphir>alter tablespace tbsdata02 online;

Tablespace modifié.

SYS@saphir>create table table_02 tablespace tbsdata02 as
  2  select * from cat;

Table créée.
```

Le tablespace par défaut

```
ALTER DATABASE DEFAULT TABLESPACE APPLICATION_01;
```



Tous les objets qui ne comportent pas, dans la syntaxe de création, les mentions de stockage dans un tablespace, sont stockés automatiquement dans le tablespace permanent par défaut.

Il faut prendre l'habitude de définir un tablespace permanent par défaut qui reçoive l'ensemble des objets créés sans mention de stockage.

Vous pouvez définir le tablespace permanent par défaut de la base de données à l'aide de la commande SQL suivante :

```
ALTER DATABASE
      DEFAULT TABLESPACE nom_tablespace ;
```

```
SYS@saphir>alter database default tablespace tbsdata;

Base de données modifiée.

SYS@saphir>select property_value from database_properties
  2  where property_name like 'DEFAULT_PERMANENT_TABLESPACE';

PROPERTY_VALUE
------------------------------------------------------------------
TBSDATA

SYS@saphir>create table table_03 as select * from cat;

Table créée.

SYS@saphir>select table_name, tablespace_name
  2  from dba_tables
  3  where tablespace_name = 'TBSDATA';
```

```
TABLE_NAME                     TABLESPAC
------------------------------ ---------
TABLE_01                       TBSDATA
```

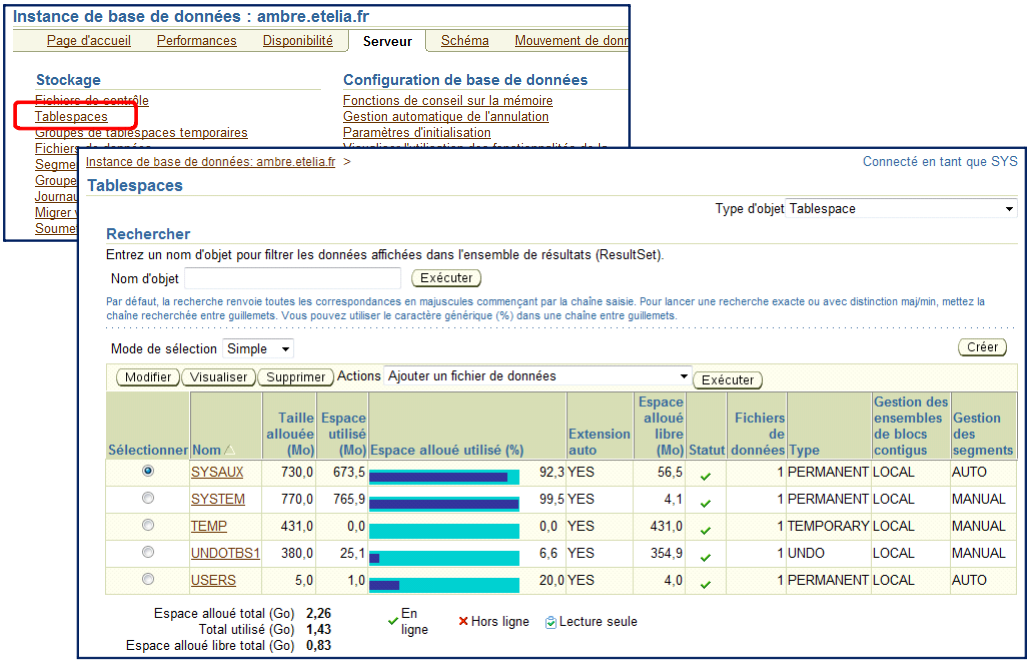
La vue « **DATABASE_PROPERTIES** » permet d'afficher toutes les informations sur les propriétés par défaut d'une base de données comme : le nom global de la base de données, le tablespace permanent, le tablespace temporaire, le type de tablespace, le fuseau horaire, la langue, le territoire…

Le tablespace permanent par défaut peut être défini lors de la création de la base de données, grâce à l'argument « **DEFAULT TABLESPACE** ».

```
SQL> create database "tpdba"
...
  7  datafile size 300m autoextend on
  8            next 10240k
  9            maxsize unlimited
 10            extent management local
 11  sysaux datafile size 120m autoextend on
 12            next 10240k
 13            maxsize unlimited
 14  default temporary tablespace temp tempfile size 20m
 15           autoextend on next 640k maxsize unlimited
 16  undo tablespace "UNDOTBS1" datafile size 200m
 17           autoextend on next 5120k maxsize unlimited
 16  default tablespace app_users datafile size 200m
 17           autoextend on next 5120k maxsize unlimited
...
```

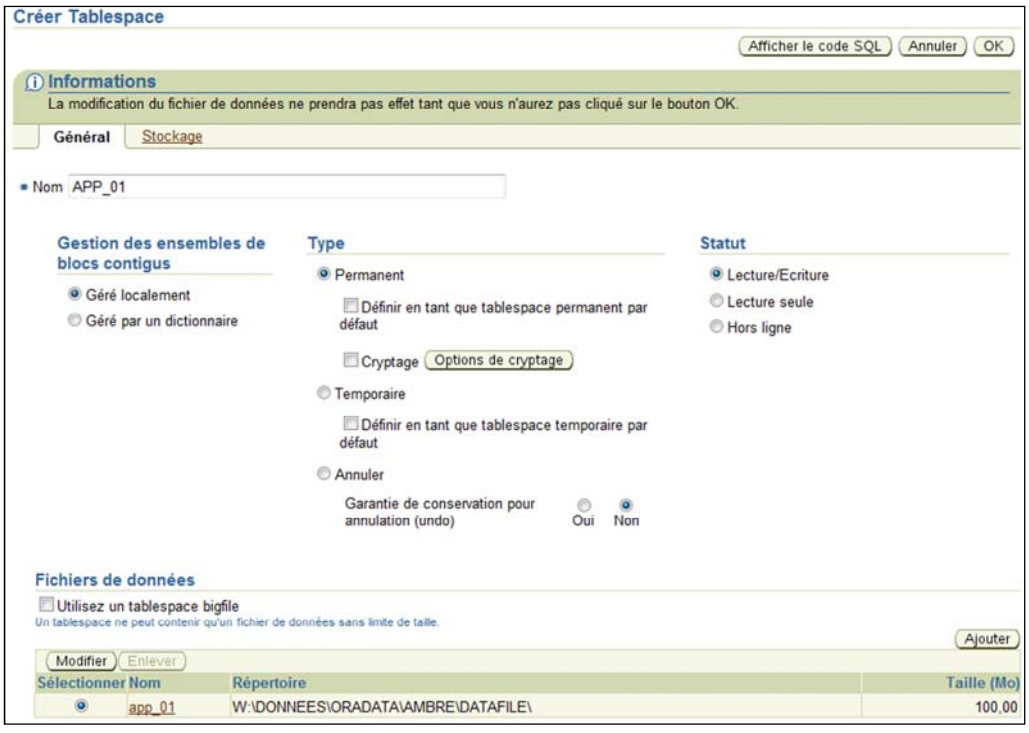
Comme vous pouvez le remarquer dans l'exemple précédent, la création de la base de données permet de créer en plus des quatre tablespaces « **SYSTEM** », « **SYSAUX** », « **UNDOTBS1** » et « **TEMP** », un autre tablespace permanent « **APP_USERS** » pour devenir le tablespace permanent par défaut de la base.

La console d'administration

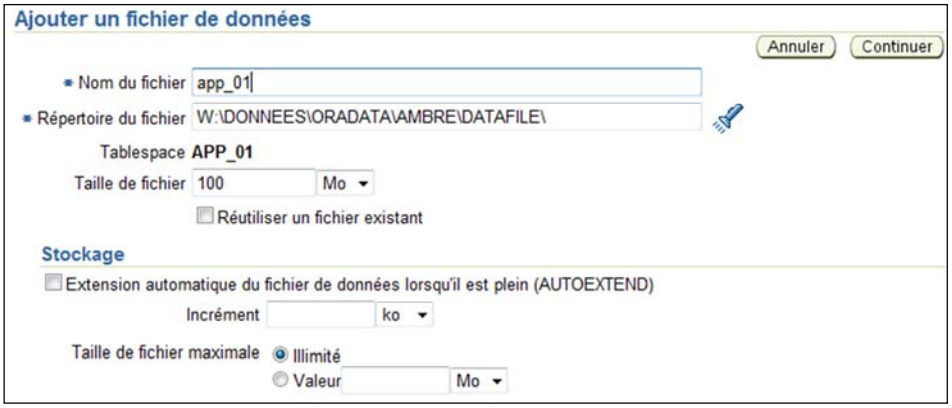


Vous pouvez également utiliser la console **OEM** (**O**racle **E**nterprise **M**anager) pour créer un tablespace.

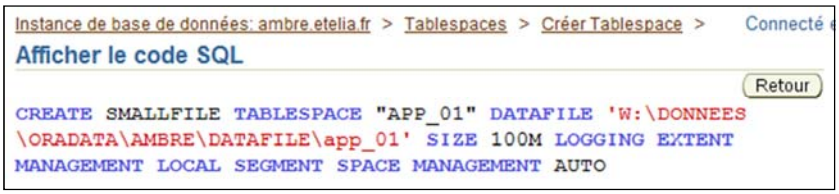
Choisissez l'onglet ***Serveur*** sur la page d'accueil puis sur le lien ***Tablespaces*** pour accéder à la page de gestion des tablespaces. Vous pouvez ainsi créer un tablespace, définir le type et le statut, ainsi que le mode de gestion du tablespace.



Vous pouvez saisir toutes les informations relatives aux fichiers de données définis pour le tablespace, comme l'emplacement, la taille et la quantité d'espace utilisée dans le fichier de données.



Pour toutes les opérations de création d'objets et des composants de la base de données que vous exécutez à travers la console, vous pouvez obtenir le code SQL de l'opération de création.

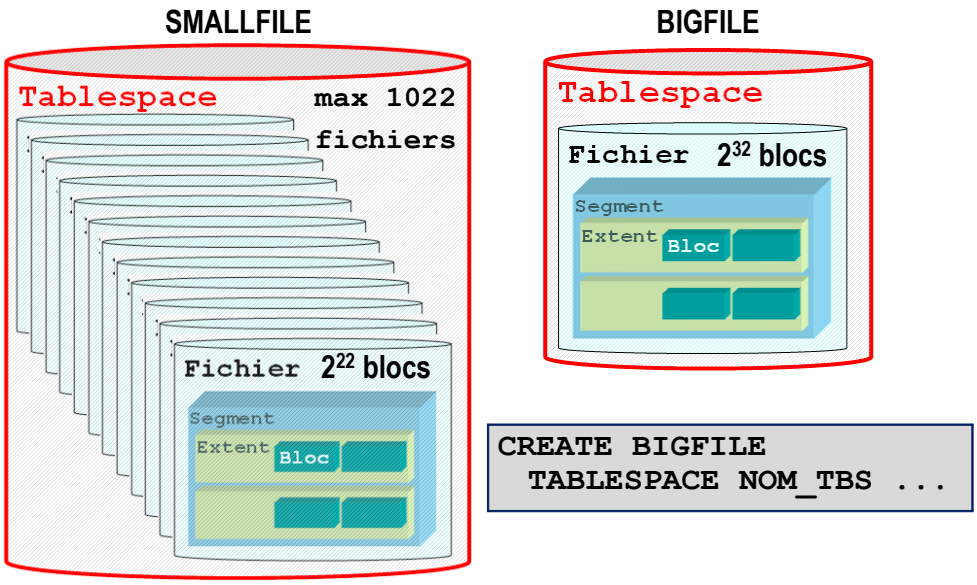


Attention

Pour toutes les opérations d'administration que vous effectuez à travers la console, vous devez impérativement sauvegarder le script SQL correspondant.

En effet, plusieurs opérations d'administration de la base de données nécessitent le script SQL pour pouvoir recréer les objets ou tout simplement effectuer des opérations de maintenance.

Le tablespace BIGFILE



Le tablespace de type « **SMALLFILE** » peut avoir plusieurs fichiers de données, au maximum **1022** fichiers, qui peuvent chacun contenir jusqu'à 2^{22} blocs de données. Ainsi pour un talespace construit avec un bloc de **32 Kb**, vous pouvez avoir un espace de stockage par fichier de **128 Gb**. La taille maximale de stockage pour un tablespace « **SMALLFILE** » est d'approximativement **32 Tb** (il faut **1024** fichiers pour une taille exacte de **32 Tb**).

L'inconvénient de ce type de tablespace, c'est que lors d'un point de contrôle (checkpoint), le processus « **DBWn** » écrit les blocs modifiés dans les fichiers de données correspondants et le processus « **CKPT** » met à jour l'en-tête de tous les fichiers de la base de données qui sont en mode lecture écriture. Ainsi, plus il y a de fichiers, plus les checkpoints durent longtemps même si ces fichiers n'ont pas de blocs de données modifiés.

Par ailleurs, quand vous utilisez un gestionnaire de volume logique comme Oracle ASM, les fichiers dans ces architectures sont uniquement des informations logiques.

Le tablespace type « **BIGFILE** » est un tablespace avec un seul fichier de données qui peut contenir jusqu'à 2^{32} blocs de données. Ainsi un fichier de données pour un tablespace construit avec des blocs de **32 Kb** peut stocker jusqu'à **32 Tb**. Ainsi ce type de tablespace réduit le nombre de fichiers de données en occurrence avec le temps des checkpoints.

Lorsque vous créez votre base de données, vous pouvez spécifier le type de tablespace par défaut que vous souhaitez, à l'aide de la syntaxe :

```
… SET DEFAULT {BIGFILE | SMALLFILE} TABLESPACE …
```

Sans précision explicite, le type du talespace « SYSTEM » est considéré comme le type par défaut. Après la création de la base de données vous pouvez également changer le type par défaut à l'aide de la syntaxe :

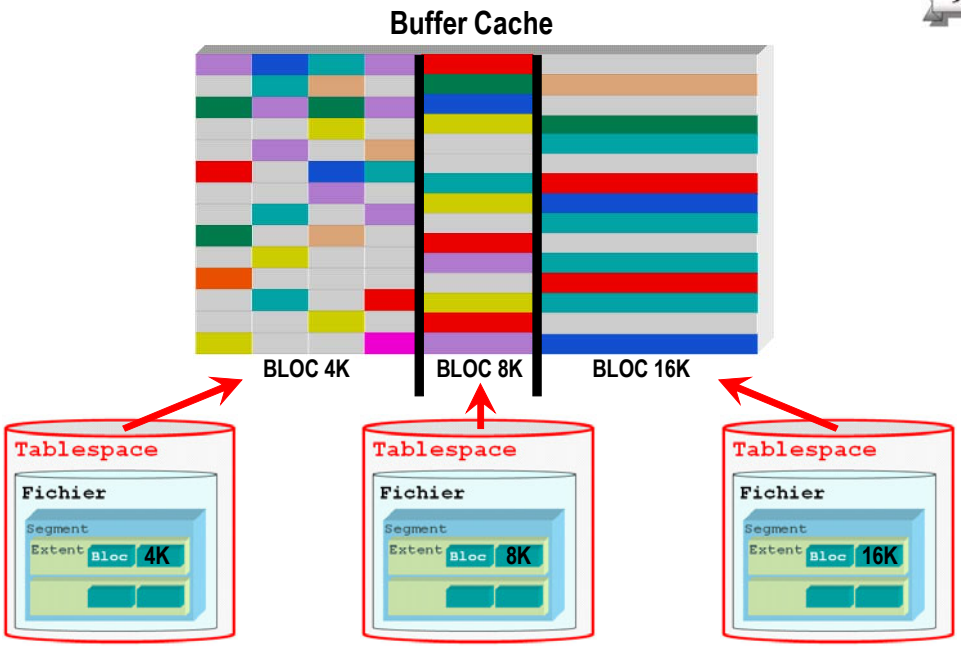
```
ALTER DATABASE SET
        DEFAULT {BIGFILE | SMALLFILE} TABLESPACE ;
```

La vue « **DATABASE_PROPERTIES** » vous permet d'afficher le type de tablespace par défaut de votre base de données.

```
SYS@rubis>alter database set default bigfile tablespace;

Base de données modifiée.

SYS@rubis>select property_name, property_value
  2  from database_properties
  3  where property_name = 'DEFAULT_TBS_TYPE';

PROPERTY_NAME      PROPERTY_VALUE
------------------ ------------------
DEFAULT_TBS_TYPE   BIGFILE

SYS@rubis>create tablespace tbsbig datafile
  2  size 10m autoextend on next 10m;

Tablespace créé.

SYS@rubis>create tablespace tbssmall datafile
  2  size 10m autoextend on next 10m ,
  3  size 10m autoextend on next 10m;
create tablespace tbssmall datafile
*
ERREUR à la ligne 1 :
ORA-32774: plusieurs fichiers ont été indiqués pour le tablespace
BIGFILE TBSSMALL

SYS@rubis>create smallfile tablespace tbssmall datafile
  2  size 10m autoextend on next 10m ,
  3  size 10m autoextend on next 10m;

Tablespace créé.

SYS@rubis>select tablespace_name, bigfile, file_name
  2  from dba_data_files join
  3       dba_tablespaces using ( tablespace_name)
  4  where tablespace_name in ( 'TBSBIG', 'TBSSMALL');

TABLESPAC BIG FILE_NAME
--------- --- --------------------------------------------------
TBSBIG    YES +GD_DONNEES/rubis/datafile/tbsbig.330.724090169
TBSSMALL  NO  +GD_DONNEES/rubis/datafile/tbssmall.329.724090417
TBSSMALL  NO  +GD_DONNEES/rubis/datafile/tbssmall.328.724090419
```

Dans l'exemple précédent, vous pouvez voir la configuration de la base de données pour créer par défaut des tablespaces de type « **BIGFILE** ». Par la suite, la création du tablespace « **TBSBIG** » est effectuée avec succès, mais la création d'un tablespace « **TBSSMALL** » avec plusieurs fichiers échoue. En précisant qu'il s'agit d'un tablespace de type « **SMALLFILE** » le tablespace est cette fois-ci créé.

La taille du bloc



Pour lire ou écrire les données de la base de données, Oracle charge d'abord les blocs correspondants dans le buffer cache (cache de tampon). Ainsi il faut d'abord paramétrer le buffer cache pour pouvoir recevoir les blocs respectifs.

Le paramètre qui vous permet de réserver de l'espace pour les blocs différents des blocs par défaut est :

DB_nK_CACHE_SIZE

Les valeurs autorisées pour le « **n** » sont 2, 4, 8, 16 et 32 (certains systèmes d'exploitation n'acceptent pas cette valeur). La valeur du bloc par défaut est « **DB_BLOCK_SIZE** », paramètre défini à la création de la base de données.

```
SYS@rubis>show parameter db_block_size

NAME                                 TYPE        VALUE
------------------------------------ ----------- -----
db_block_size                        integer     8192
SYS@rubis>show parameter k_cache_size

NAME                                 TYPE        VALUE
------------------------------------ ----------- -----
db_16k_cache_size                    big integer 0
db_2k_cache_size                     big integer 0
db_32k_cache_size                    big integer 0
db_4k_cache_size                     big integer 0
db_8k_cache_size                     big integer 0
SYS@rubis>alter system set db_32k_cache_size=16m;

Système modifié.

SYS@rubis>alter system set db_4k_cache_size=16m;

Système modifié.
```

```
SYS@rubis>show parameter k_cache_size

NAME                                 TYPE        VALUE
------------------------------------ ----------- -----
db_16k_cache_size                    big integer 0
db_2k_cache_size                     big integer 0
db_32k_cache_size                    big integer 16M
db_4k_cache_size                     big integer 16M
db_8k_cache_size                     big integer 0
```

L'exemple précédent montre le paramétrage de la mémoire pour pouvoir créer des tablespaces qui utilisent une taille de bloc de 4 Kb et 16 Kb en plus de la taille par défaut de 8 Kb.

```
SYS@rubis>create tablespace data_4k datafile
  2  size 10m autoextend on next 10m ,
  3  blocksize 4k;

Tablespace créé.

SYS@rubis>create tablespace data_32k datafile
  2  size 10m autoextend on next 10m
  3  blocksize 32k;

Tablespace créé.

SYS@rubis>create tablespace data_2k datafile
  2  size 10m autoextend on next 10m
  3  blocksize 2k;
create tablespace data_2k datafile
*
ERREUR à la ligne 1 :
ORA-29339: la taille de bloc de tablespace 2048 ne correspond pas
aux tailles de blocs configures

SYS@rubis>select tablespace_name, block_size/1024 "bs K", file_name
  2  from dba_data_files join
  3       dba_tablespaces using ( tablespace_name)
  4  where tablespace_name like 'DATA%K';

TABLESPAC bs K FILE_NAME
--------- ---- ----------------------------------------------------
DATA_4K      4 +GD_DONNEES/rubis/datafile/data_4k.327.724092783
DATA_32K    32 +GD_DONNEES/rubis/datafile/data_32k.326.724092799
```

Les deux premières créations de tablespace sont acceptées par la base de données, et le paramétrage du buffer cache est déjà effectué. Par contre la dernière création est rejetée ; il faut d'abord paramétrer le buffer cache pour pouvoir recevoir les blocs de 2 Kb.

```
SYS@rubis>create table tdata_32k tablespace data_32k as
  2  select * from cat;

Table créée.

SYS@rubis>alter system set db_32k_cache_size=0;
```

```
Système modifié.

SYS@rubis>select * from tdata_32k;
select * from tdata_32k
              *
ERREUR à la ligne 1 :
ORA-00379: aucun tampon libre dans le pool de tampons DEFAULT pour
la taille de bloc (32 ko)


SYS@rubis>alter system set db_32k_cache_size=16m;

Système modifié.

SYS@rubis>select count(*) from tdata_32k;

  COUNT(*)
----------
      4854
```

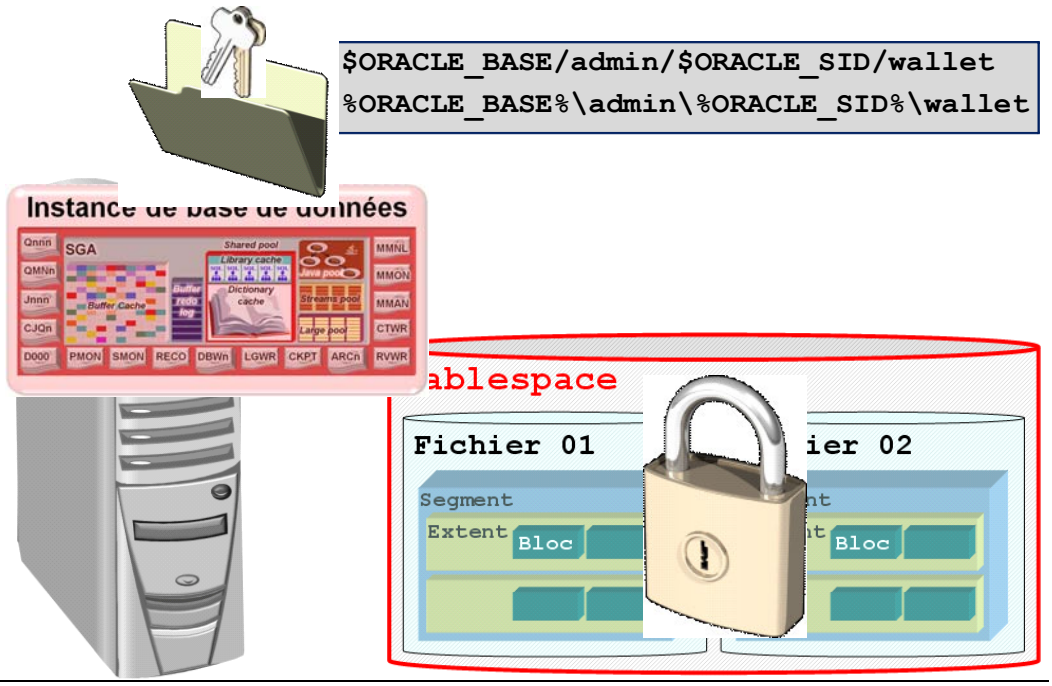
Attention

Pour lire ou écrire les données de la base de données, Oracle charge d'abord les blocs correspondants dans le buffer cache (cache de tampon).

Si une des zones d'une taille de bloc non standard vient d'être mise à zéro, alors tous les objets des tablespaces qui ont cette taille de bloc ne peuvent plus être lus, comme c'est le cas dans l'exemple précédent.

Une fois que la zone mémoire a été initialisée, les données sont à nouveau accessibles.

Le cryptage transparent



Pour pouvoir mettre en place le cryptage transparent, il faut d'abord configurer « Oracle Encryption Wallet » s'il n'a pas déjà été initialisé pour votre base de données.

Initialiser consiste à créer un fichier « **WALLET** » portefeuille dans le répertoire par défaut pour ce fichier de la base de données :

```
$ORACLE_BASE/admin/$ORACLE_SID/wallet
```

ou

```
%ORACLE_BASE%\admin\%ORACLE_SID%\wallet
```

Le répertoire doit exister, sinon la création par défaut du fichier est impossible.

Vous pouvez visualiser l'emplacement du fichier « **WALLET** » portefeuille et le statut en interrogeant la vue « **V$ENCRYPTION_WALLET** ».

La syntaxe de création du portefeuille est la suivante :

```
ALTER SYSTEM SET ENCRYPTION KEY
                    IDENTIFIED BY "mot de passe";
```

mot de passe Le mot de passe pour l'activation du cryptage une fois le fichier portefeuille créé.

Vous pouvez visualiser les informations concernant les algorithmes de cryptage en interrogeant la vue « **V$RMAN_ENCRYPTION_ALGORITHMS** ».

```
SYS@saphir>select algorithm_name "Nom",
  2  algorithm_description "Description",is_default "Déf"
  3  from v$rman_encryption_algorithms;

Nom      Description      Déf
-------- ---------------- ---
AES128   AES 128-bit key  YES
AES192   AES 192-bit key  NO
```

```
AES256   AES 256-bit key  NO

SYS@saphir>!mkdir /u01/app/oracle/admin/saphir/wallet

SYS@saphir>!ls -la /u01/app/oracle/admin/saphir
...
drwxr-xr-x 2 oracle oinstall 4096 jui 11 18:31 wallet


SYS@saphir>select * from v$encryption_wallet;

WRL_TYPE WRL_PARAMETER                          STATUS
-------- ------------------------------------ ------
file     /u01/app/oracle/admin/saphir/wallet  CLOSED

SYS@saphir>alter system set encryption key identified by "P#sw0rd3";

Système modifié.

SYS@saphir>select * from v$encryption_wallet;

WRL_TYPE WRL_PARAMETER                          STATUS
-------- ------------------------------------ -------
file     /u01/app/oracle/admin/saphir/wallet  OPEN

SYS@saphir>!ls -la /u01/app/oracle/admin/saphir/wallet
...
-rw-r--r-- 1 oracle oinstall 1573 jui 11 18:36 ewallet.p12
```

Attention

Il est impératif, après chaque démarrage de la base, d'ouvrir le fichier portefeuille pour pouvoir bénéficier des fonctionnalités de cryptage.

La syntaxe d'ouverture ou de la fermeture du fichier portefeuille est la suivante :

```
ALTER SYSTEM SET WALLET
  { OPEN | CLOSE }
  IDENTIFIED BY "mot de passe";
```

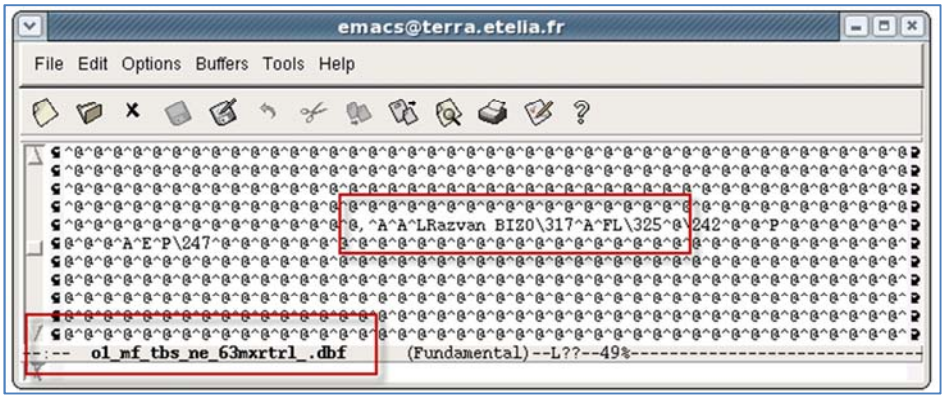
```
 SYS@saphir>startup force
Instance ORACLE lancée.
...
SYS@saphir>select * from v$encryption_wallet;

WRL_TYPE WRL_PARAMETER                          STATUS
-------- ------------------------------------ ------------------
file     /u01/app/oracle/admin/saphir/wallet  CLOSED

SYS@saphir>alter system set wallet open identified by "P#sw0rd3";

Système modifié.
```

```
SYS@saphir>create tablespace tbs_ne datafile size 256k;

Tablespace créé.

SYS@saphir>create tablespace tbs_e datafile size 256k
  2  encryption using 'AES256' default storage ( encrypt);

Tablespace créé.

SYS@saphir>create table nom01 ( nom varchar2(50)) tablespace tbs_ne;

Table créée.

SYS@saphir>create table nom02 ( nom varchar2(50)) tablespace tbs_e;

Table créée.

SYS@saphir>insert all into nom01 into nom02
  2  select 'Razvan BIZOÏ' from dual;

2 ligne(s) créée(s).

SYS@saphir>commit;

Validation effectuée.

SYS@saphir>select * from nom01, nom02;

NOM            NOM
-------------- --------------
Razvan BIZOÏ   Razvan BIZOÏ

SYS@saphir>alter system set wallet close identified by "P#sw0rd3";

Système modifié.

SYS@saphir>select * from nom02;
select * from nom02
              *
ERREUR à la ligne 1 :
ORA-28365: le wallet n'est pas ouvert


SYS@saphir>select file_name
  2  from dba_data_files join
  3       dba_tablespaces using ( tablespace_name)
  4  where tablespace_name like 'TBS%E';

FILE_NAME
--------------------------------------------------------------------
/u02/donnees/oradata/SAPHIR/datafile/o1_mf_tbs_ne_63mxrtrl_.dbf
/u02/donnees/oradata/SAPHIR/datafile/o1_mf_tbs_e_63mxrx4g_.dbf

SYS@saphir>!mv /u01/app/oracle/admin/saphir/wallet/ewallet.p12
/u01/app/oracle/admin/saphir/wallet/ewallet.p12.sav
```

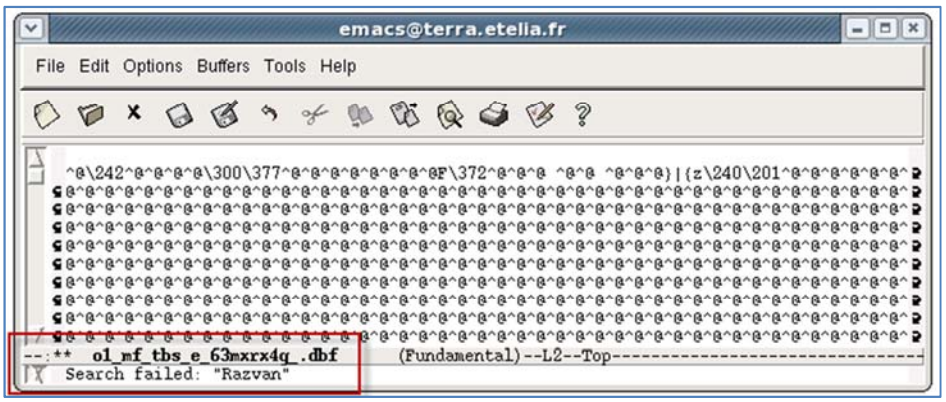
```
SYS@saphir>alter system set wallet close identified by "P#sw0rd3";

Système modifié.

SYS@saphir>alter system set wallet open identified by "P#sw0rd3";
alter system set wallet open identified by "P#sw0rd3"
*
ERREUR à la ligne 1 :
ORA-28367: le wallet n'existe pas

SYS@saphir>alter system set encryption key identified by "P#sw0rd3";
alter system set encryption key identified by "P#sw0rd3"
*
ERREUR à la ligne 1 :
ORA-28374: clé maître indiquée introuvable dans le portefeuille
(wallet)
```

L'utilisation du fichier portefeuille permet de créer des tablespaces cryptés. Ainsi le tablespace **TBS_NE** est un tablespace classique qui contient une table avec un seul champ et un seul enregistrement avec la valeur `'Razvan BIZOÏ'`. Le deuxième tablespace **TBS_E** est un tablespace crypté contenant également une table avec un seul champ et un seul enregistrement. Évidemment, si vous perdez le fichier portefeuille, vous n'avez plus aucune possibilité de lire les données stockées dans les tablespaces.

En utilisant un éditeur qui permet de lire des contenus hexadécimaux, vous pouvez remarquer que pour le tablespace non crypté l'information est visible comme c'est le cas dans l'image suivante.

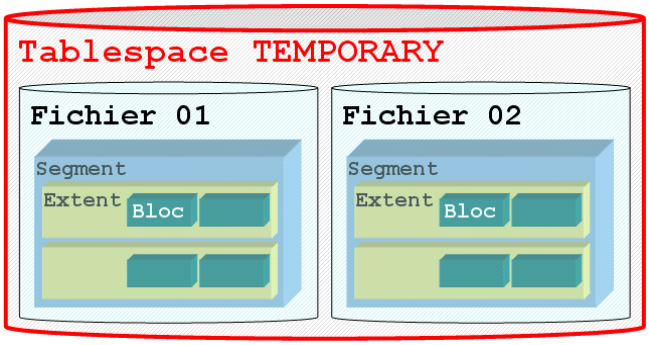


Dans le cas d'un tablespace crypté il n'est pas possible de trouver l'information.



Le tablespace temporaire

```
CREATE TEMPORARY TABLESPACE TEMP TEMPFILE
SIZE 1G AUTOEXTEND ON NEXT 256M,
SIZE 1G AUTOEXTEND ON NEXT 256M;
```



Lors d'importantes opérations de tri (telles que select distinct, union et create index) Oracle a besoin de stocker dans la base de données, des informations concernant le tri des enregistrements, avant de retourner l'information aux utilisateurs. En raison de leur nature dynamique, ces espaces de tris ne devraient pas être stockés avec d'autres types de données.

Lorsqu'un tablespace temporaire « **TEMPORARY** » est défini, un segment de tri est aussi créé. Celui-ci est capable de croître si nécessaire afin de pouvoir héberger toutes les opérations de tri de données, et existe jusqu'à ce que la base de données soit fermée puis redémarrée.

La syntaxe de création d'un tablespace temporaire est :

```
CREATE {BIGFILE|SMALLFILE} TEMPORARY
       TABLESPACE nom_tablespace
       [ TEMPFILE
         ['nom_fichier'] [ SIZE integer {K|M|G|T} REUSE]
         [ AUTOEXTEND
            {OFF |
             ON [ NEXT integer {K|M|G|T}]
                [ MAXSIZE {UNLIMITED | integer {K|M|G|T}]
             }
          ] [,...]
       ]
       [{ONLINE | OFFLINE}] ;
```

TEMPORARY	Indique que le tablespace est de type temporaire.
TEMPFILE	Indique qu'il s'agit d'un fichier temporaire et pas d'un fichier de données.
REUSE	Définit la réutilisation du fichier temporaire s'il existe déjà.

```
SYS@jaspe>create bigfile temporary tablespace temp01 tempfile
  2  size 10m autoextend on next 10m ;

Tablespace créé.

SYS@jaspe>create bigfile temporary tablespace temp02 tempfile
  2  size 10m autoextend on next 10m ;

Tablespace créé.

SYS@jaspe>create bigfile temporary tablespace temp03 tempfile
  2  size 10m autoextend on next 10m ;

Tablespace créé.

SYS@jaspe>select tablespace_name, file_name
  2  from dba_temp_files join
  3       dba_tablespaces using ( tablespace_name)
  4  where tablespace_name like 'TEMP0_';

TABLESPAC FILE_NAME
--------- --------------------------------------------------
TEMP01    +GD_DONNEES/jaspe/tempfile/temp01.283.724094305
TEMP02    +GD_DONNEES/jaspe/tempfile/temp02.281.724094309
TEMP03    +GD_DONNEES/jaspe/tempfile/temp03.280.724094315
```

Dans l'exemple précédent, vous pouvez observer la création de trois tablespaces temporaires de type « **BIGFILE** ». Les fichiers temporaires ainsi créés peuvent s'agrandir automatiquement par des tranches de 10 Mb.

```
SYS@ambre>create temporary tablespace temp04
  2  tempfile 'W:\DONNEES\ORADATA\AMBRE\DATAFILE\TEMP04.TMP'
  3  size 10m autoextend on next 10m;

Tablespace créé.

SYS@ambre>drop tablespace temp04;

Tablespace supprimé.

SYS@ambre>create temporary tablespace temp04
  2  tempfile 'W:\DONNEES\ORADATA\AMBRE\DATAFILE\TEMP04.TMP'
  3  size 10m autoextend on next 10m;
create temporary tablespace temp04
*
ERREUR à la ligne 1 :
ORA-01119: échec de création du fichier de base de données
'W:\DONNEES\ORADATA\AMBRE\DATAFILE\TEMP04.TMP'
ORA-27038: le fichier créé existe déjà
OSD-04010: option <create> indiquée ; le fichier existe déjà

SYS@ambre>create temporary tablespace temp04
  2  tempfile 'W:\DONNEES\ORADATA\AMBRE\DATAFILE\TEMP04.TMP'
  3  size 20m reuse autoextend on next 10m;

Tablespace créé.
```

```
SYS@ambre>$dir W:\DONNEES\ORADATA\AMBRE\DATAFILE\TEMP04.TMP
Le volume dans le lecteur W s'appelle Nouveau nom
 Le numéro de série du volume est C051-C97C

 Répertoire de w:\donnees\oradata\AMBRE\DATAFILE

11/07/2010  18:03        20 979 712 TEMP04.TMP
...
```

L'exemple précédent commence par la création d'un tablespace « **TEMP04** ». Ensuite on détruit le tablespace sans effacer le fichier temporaire. Vous pouvez remarquer que la création d'un tablespace ne peut pas être effectuée si le fichier correspondant existe déjà, et cela est valable pour tous les tablespaces. Vous pouvez utiliser l'argument « **REUSE** » pour demander à Oracle de réutiliser le fichier existant sur disque. En effet Oracle réutilise le nom du fichier temporaire et il le dimensionne au volume demandé.

Cette démarche de rattachement d'un fichier temporaire est utilisée dans le cas de perte du fichier de contrôle et la création à l'aide du script de sauvegarde.

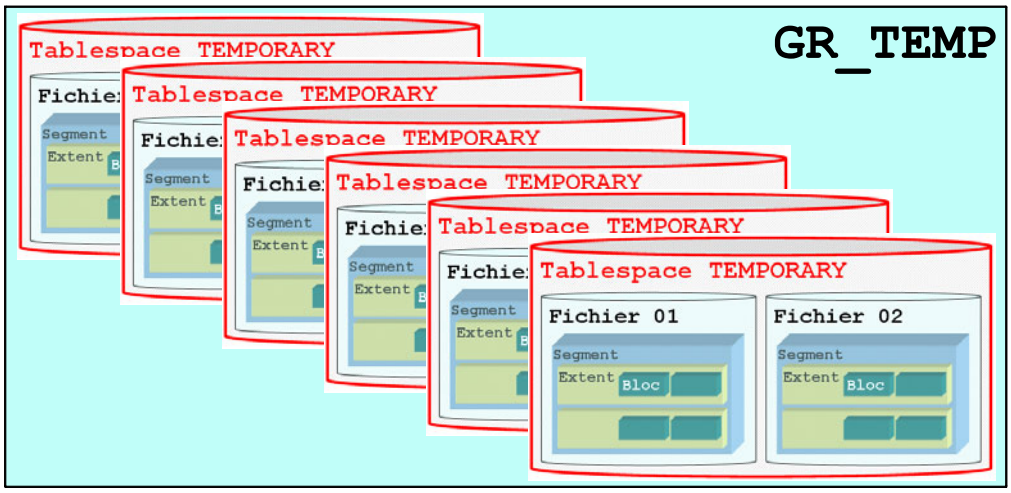
Attention

Vous ne pouvez pas créer de tablespaces temporaires avec une taille de bloc différente de la taille du bloc par défaut de la base de données.

Les tablespaces temporaires ainsi que les tablespaces système sont toujours créés avec une taille de bloc égale au bloc par défaut de la base de données.

Le groupe tablespaces temporaires

```
ALTER TABLESPACE TEMP_n TABLESPACE GROUP GR_TEMP;
...
```



Certains utilisateurs d'une base de données Oracle peuvent avoir besoin de volumes de stockage temporaire beaucoup plus grands que ceux de tous les autres utilisateurs de l'application. Dans ce cas, vous pouvez créer plusieurs tablespaces temporaires « **TEMPORARY** » pour distribuer les espaces de stockages des utilisateurs ayant des besoins semblables sur les mêmes tablespaces.

À partir de la version Oracle 10*g*, il est possible de créer des groupes de tablespaces temporaires pour équilibrer les charges et diminuer les contentions d'accès aux segments temporaires. Un groupe de tablespaces temporaires est géré par la base de données comme un tablespace temporaire ; vous pouvez ainsi définir que c'est le tablespace temporaire par défaut.

La syntaxe de création d'un groupe de tablespaces temporaires est en effet uniquement un assignement d'un tablespace temporaire existant au nom du groupe qui va être créé, comme suit :

```
ALTER TABLESPACE nom TABLESPACE GROUP nom_groupe;
```

```
SYS@jaspe>alter tablespace temp01 tablespace group ngrp_temp;

Tablespace modifié.

SYS@jaspe>alter tablespace temp02 tablespace group ngrp_temp;

Tablespace modifié.

SYS@jaspe>alter tablespace temp03 tablespace group ngrp_temp;

Tablespace modifié.

SYS@jaspe>select * from dba_tablespace_groups;

GROUP_NAME                     TABLESPAC
------------------------------ ---------
```

```
NGRP_TEMP                      TEMP01
NGRP_TEMP                      TEMP02
NGRP_TEMP                      TEMP03
```

Les trois tablespaces temporaires ont été ajoutés au groupe « **NGRP_TEMP** » ; à présent le nom du groupe peut être utilisé à la place du nom d'un tablespace temporaire. Pour retirer un tablespace d'un groupe de tablespaces temporaires, il suffit d'utiliser la syntaxe suivante :

```
ALTER TABLESPACE nom TABLESPACE GROUP '' ;
```

Vous pouvez définir le tablespace temporaire par défaut de la base, à l'aide de la commande SQL suivante :

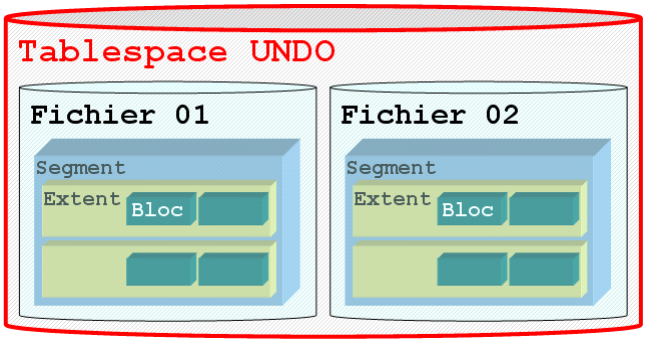
```
ALTER DATABASE
      DEFAULT TEMPORARY TABLESPACE nom_tablespace ;
```

```
SYS@jaspe>alter database default temporary tablespace ngrp_temp;

Base de données modifiée.

SYS@jaspe>select property_value from database_properties
  2  where property_name like 'DEFAULT_TEMP_TABLESPACE';

PROPERTY_VALUE
------------------
NGRP_TEMP
```

Dans l'exemple précédent, vous pouvez remarquer la définition du tablespace temporaire par défaut avec l'utilisation d'un groupe de tablespaces « **NGRP_TEMP** », et l'utilisation de la vue « **DATABASE_PROPERTIES** » pour retrouver la valeur du tablespace temporaire par défaut de la base de données.

Le tablespace undo

```
CREATE UNDO TABLESPACE UNDO DATAFILE
       SIZE 2G AUTOEXTEND ON NEXT 512M,
       SIZE 2G AUTOEXTEND ON NEXT 512M,
       SIZE 2G AUTOEXTEND ON NEXT 512M,
       SIZE 2G AUTOEXTEND ON NEXT 512M;
```



Toutes les données d'annulation sont stockées dans un tablespace spécial appelé « **UNDO** ». Lorsque vous créez un tablespace « **UNDO** », Oracle gère le stockage, la rétention et l'emploi de l'espace pour les données de rollback par l'intermédiaire de la fonction SMU (System-Managed Undo). Aucun objet permanent n'est placé dans le tablespace undo.

Pour pouvoir créer un tablespace undo à la création de la base de données, il faut prendre soin d'initialiser le paramètre

```
UNDO_MANAGEMENT=AUTO
```

À partir de la version Oracle *9i*, il est fortement conseillé d'utiliser la gestion automatique des segments d'annulation. À partir de la version Oracle 10*g*, il est impératif d'utiliser ce mode de gestion, sans quoi une grande partie des fonctionnalités de la base de données sont inutilisables. Voir le module « Les segments UNDO ».

Rappelez-vous, la syntaxe SQL de création de la base de données comporte d'abord la création d'un tablespace « **SYSTEM** » et d'un tablespace « **SYSAUX** », ainsi que la création de tablespaces « **TEMP** » et « **UNDO** ».

Attention

À la création de la base de données, si vous ne précisez pas de tablespace undo, Oracle crée un tablespace undo appelé « **SYS_UNDOTS** ».

Le paramètre « **UNDO_TABLESPACE** » doit avoir la valeur « **SYS_UNDOTS** », sinon la base de données ne pourra pas être créée.

La syntaxe de création d'un tablespace UNDO est :

```
CREATE {BIGFILE|SMALLFILE} UNDO
       TABLESPACE nom_tablespace
       [ DATAFILE
         ['nom_fichier'] [ SIZE integer {K|M|G|T} ]
```

```
        [ AUTOEXTEND
          {OFF |
           ON [ NEXT integer {K|M|G|T}]
              [ MAXSIZE {UNLIMITED | integer {K|M|G|T}]}
         ] [,...]
      ]
      [{ONLINE | OFFLINE}] ;
```

UNDO Indique que le tablespace est de type undo.

Attention, les tablespaces pérennes et temporaires par défaut sont attribués à l'aide de la syntaxe « **ALTER DATABASE DEFAULT** … » ; ce n'est pas le cas pour le tablespace de type « **UNDO** ». Le tablespace « **UNDO** » par défaut est celui qui est précisé dans le paramètre « **UNDO_TABLESPACE** ».

```
SYS@rubis>show parameter undo_tablespace

NAME                                 TYPE        VALUE
------------------------------------ ----------- --------
undo_tablespace                      string      UNDOTBS1

SYS@rubis>create bigfile undo tablespace undo datafile
  2  size 512m autoextend on next 512m;

Tablespace créé.

SYS@rubis>select contents, file_name
  2  from dba_data_files join
  3       dba_tablespaces using ( tablespace_name)
  4  where tablespace_name like 'UNDO';

CONTENTS  FILE_NAME
--------- -------------------------------------------------
UNDO      +GD_DONNEES/rubis/datafile/undo.325.724104703

SYS@rubis>alter system set undo_tablespace =undo;

Système modifié.

SYS@rubis>show parameter undo_tablespace

NAME                                 TYPE        VALUE
------------------------------------ ----------- -------
undo_tablespace                      string      UNDO
```

Attention

Comme pour les tablespaces système et pour les tablespaces temporaires, le tablespace **UNDO** doit être créé avec la taille de bloc par défaut.

La base de données Oracle utilise pour ses fichiers de gestion la taille de bloc par défaut définie lors de la création de cette base de données.

Atelier 15

Exercice n°1

Identifiez le tablespace permanent, temporaire et undo par défaut de votre base de données.

Exercice n°2

Créez deux tablespaces « **TPD01** » et « **TPD02** » chacun avec un seul fichier de données d'une taille de 10 M sans agrandissement automatique.

Définissez le tablespace « **TPD01** » comme étant le tablespace permanent par défaut de votre base de données. Effacez l'ancien tablespace par défaut.

Exercice n°3

Créez un tablespace de type temporaire « **TPT01** » avec un seul fichier de données d'une taille de 10 M sans agrandissement automatique.

Définissez le tablespace « **TPT01** » comme étant le tablespace temporaire par défaut de votre base de données. Effacez l'ancien tablespace par défaut.

Exercice n°4

Créez un tablespace de type undo « **TPU01** » avec un seul fichier de données d'une taille de 512 M en agrandissement automatique jusqu'à une taille de 3 G.

Définissez le tablespace « **TPU01** » comme étant le tablespace temporaire par défaut de votre base de données. Effacez l'ancien tablespace par défaut.

Exercice n°5

Créez un tablespace de type BIGFILE « **TPD03** » avec une taille de bloc de 4 k et un tablespace de type BIGFILE « **TPD04** » avec une taille de bloc de 16 k, chacun avec un fichier de données d'une taille de 10 M, avec agrandissement automatique jusqu'à une taille de 128 M.

Exercice n°6

Affichez le répertoire de destination pour les fichiers des données et pour les fichiers journaux.

Créez un tablespace « **GEST_DATA** » avec deux fichiers de données, chacun d'une taille de 10 M.