23

Gestion de l'espace disque et des fichiers

Dans ce chapitre :

- les différences entre base de données physique et logique ;
- le principe du tablespace ;
- les tablespaces temporaires, d'annulation et de données ;
- le DB_BLOCK_SIZE ;
- la notion de segment, d'extent et de bloc ;
- les différents types de segments ;
- la gestion des fichiers de données ;
- la gestion des fichiers redo-log ;
- la gestion des fichiers de contrôle ;
- la gestion automatique des fichiers par Oracle ;
- les différents types de fichiers d'initialisation.

Une base Oracle est conçue pour stocker des données, les modifier, les supprimer, etc. Toutes ces actions sont conservées dans les fichiers qui composent la base. Mais comment Oracle gère-t-il l'espace disque alloué au cours de toutes ces opérations ? Comment déterminer la taille de la base de données ? Comment contrôler l'évolution des différents constituants et anticiper les problèmes ? De quelle nature peuvent être ces

problèmes ? Ce chapitre répond à toutes ces questions par une approche théorique et pratique, pour vous aider à comprendre et agir au cœur de la gestion de l'espace disque des bases Oracle 10*g*.

> Avant toute action, effectuez une sauvegarde de la base de données et entraînez-vous au préalable sur une base test, sans utilisateur actif ni données stratégiques.

Les apports d'Oracle Enterprise Manager

Oracle Enterprise Manager propose des outils graphiques puissants qui couvrent très bien la gestion de l'espace disque et des fichiers ainsi que toutes les possibilités présentées dans ce chapitre.

Pourquoi alors ne pas les avoir présentés ici ? Nous avons volontairement fait l'impasse sur eux, pour vous faire découvrir les entrailles d'Oracle, de façon à comprendre, au-delà d'une simple interface graphique, la richesse et la puissance des options proposées.

> Je vous conseille donc de lire ce chapitre pour comprendre le fonctionnement interne d'Oracle et d'utiliser en parallèle Oracle Enterprise Manager pour son ergonomie. Vous serez ainsi capable de maîtriser à la fois la forme et le fond.

Gestion de l'espace disque

La classification des fichiers décrite au chapitre 9, *Les fichiers d'une base Oracle 10*g, permet de réduire le nombre de ceux qui requièrent une attention particulière de la part de l'administrateur Oracle. De plus, comprendre l'utilisation d'un fichier de données, de contrôle ou redo-log revient à les maîtriser dans leur ensemble, car les fichiers d'une même famille ont un fonctionnement identique.

Base de données physique et logique

L'ensemble des fichiers (données, redo-log, contrôle) visibles par un administrateur Windows constitue la base de données *physique* (figure 23-1).

Un développeur Oracle ou un utilisateur non DBA ne se soucie pas de ces fichiers physiques. De leur point de vue, ils manipulent des tables, des vues ou des index, qui constituent la base de données *logique*. C'est à l'administrateur Oracle qu'incombe le soin de faire le lien entre la base physique et la base de données logique. C'est la notion du tablespace Oracle qui permet d'établir ce lien.

Principe du tablespace

Le tablespace est un concept fondamental du stockage des données dans une base Oracle. Une table ou un index appartient obligatoirement à un tablespace.

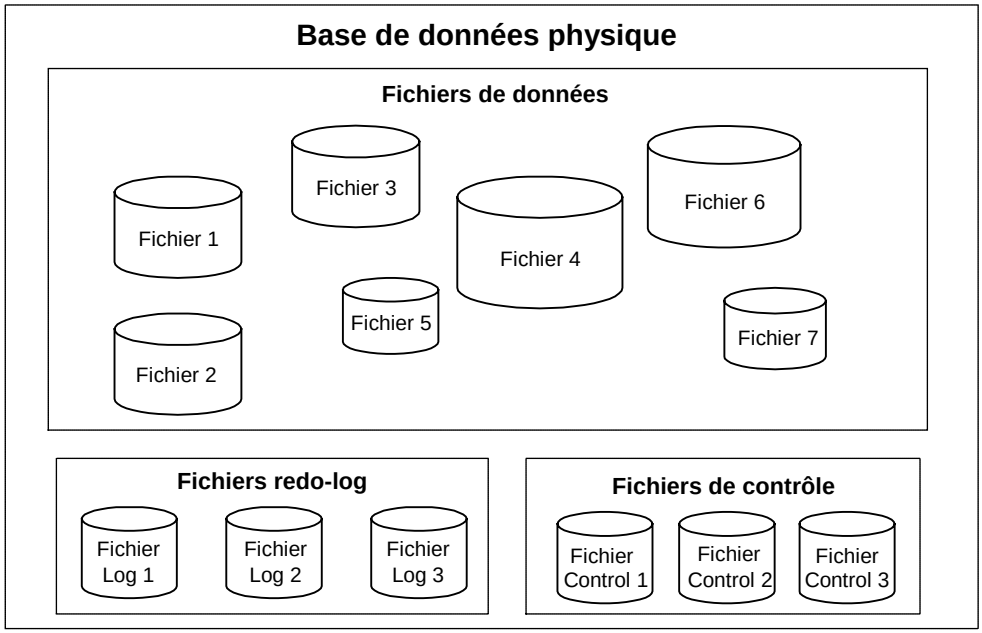


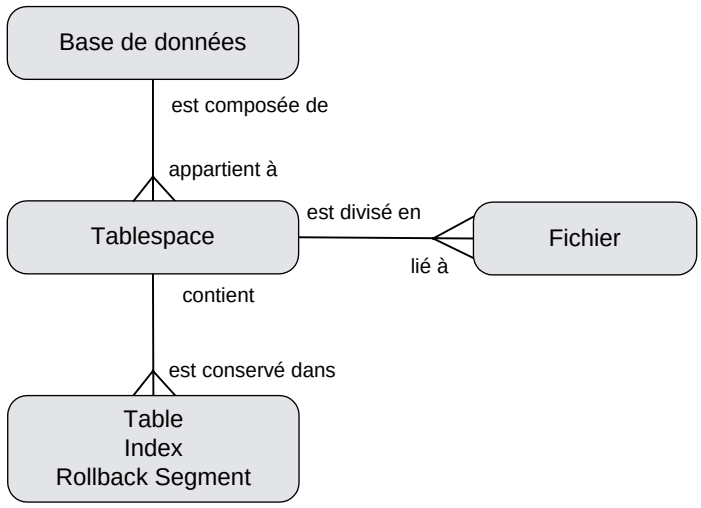
Figure 23-1

Base de données physique

À chaque tablespace sont associés un ou plusieurs fichiers. Tout objet (table, index) est placé dans un tablespace, sans précision du fichier de destination, le tablespace effectuant ce lien. La figure ci-après décrit le modèle conceptuel associé au principe du tablespace :

Figure 23-2

Schéma conceptuel du principe des tablespaces



Lorsque vous créez une table dans un tablespace, vous ne connaissez pas le fichier du tablespace dans lequel elle se trouve. Beaucoup d'administrateurs sont troublés par cette perte de contrôle de l'emplacement des données. Pourtant, ce n'est pas important, et ce pour deux raisons : premièrement, en tant qu'administrateur de base de données, vous aurez suffisamment de travail et il est inutile de vous encombrer de détails relatifs à l'emplacement physique des données au sein d'un fichier. Deuxièmement, vous pouvez toujours créer des tablespaces n'ayant qu'un seul fichier.

À sa création, une base Oracle possède obligatoirement un tablespace appelé SYSTEM, qui contient le dictionnaire de données et le segment de rollback SYSTEM. À ce tablespace est attaché au moins un fichier base. Pour garantir un bon fonctionnement de la base, il faudra créer d'autres tablespaces.

SYSAUX est un nouveau type de tablespace nécessaire à chaque base Oracle 10*g*. Précédemment, le dictionnaire de données hébergé dans le tablespace SYSTEM contenait aussi toutes les structures des différents outils ou options d'Oracle (RMAN, Text, Ultra Search, Data Mining, etc.). Le dictionnaire de la base est maintenant dans SYSTEM et tous les utilitaires Oracle dans SYSAUX. Le tablespace SYSAUX est obligatoirement créé pendant la création de la base.

En fin, le tablespace utilisateur est indiqué comme support à cet exemple. Il peut y en exister un par application, plusieurs par application, etc.

La figure 23-3 résume cette architecture minimale.

Deux critères essentiels déterminent le nombre et le contenu des tablespaces : la *performance* et l'*organisation*.

Le critère de performance veille à répartir les fichiers de la base de données sur différents disques. Cette répartition vise à limiter les problèmes de contention disque.

Le critère d'organisation vise à ordonner et ranger les données en créant plusieurs tablespaces afin de :

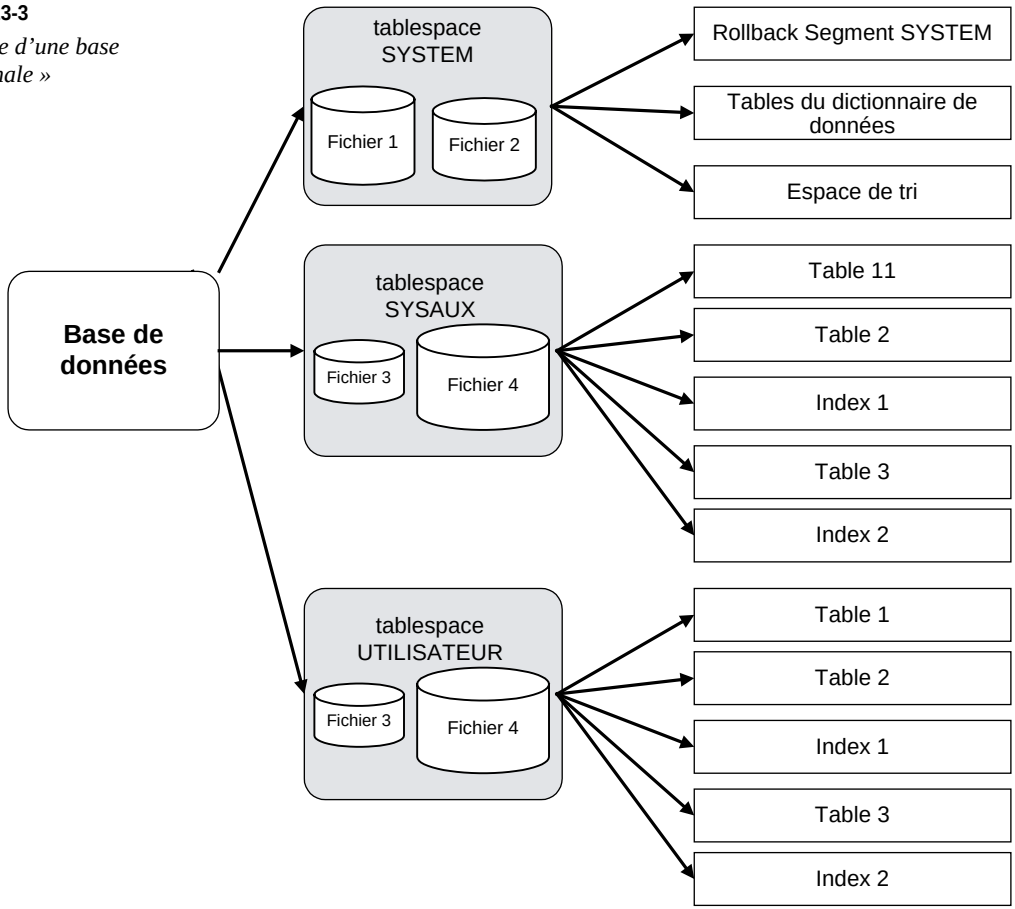
- faciliter les opérations de maintenance : sachant que la plus petite entité cohérente de sauvegarde est le tablespace, le DBA peut en créer plusieurs. Chacun d'eux contient des tables qui nécessitent d'être sauvegardées à la même fréquence ;
- gérer des données utilisées par des applications différentes dans une seule base de données. Cela permet de démarrer un seul jeu de processus Oracle, et d'allouer une seule zone mémoire partagée SGA pour des applications différentes ;
- spécialiser des tablespaces pour des tâches techniques internes à Oracle : l'espace où seront stockés les segments d'annulation (UNDO) et celui où Oracle effectuera ses tris (TEMPORARY).

On peut ainsi se diriger vers l'architecture suivante (figure 23-4) :

Dans cet exemple, la base de données est composée :

- du tablespace SYSTEM (obligatoire) ;

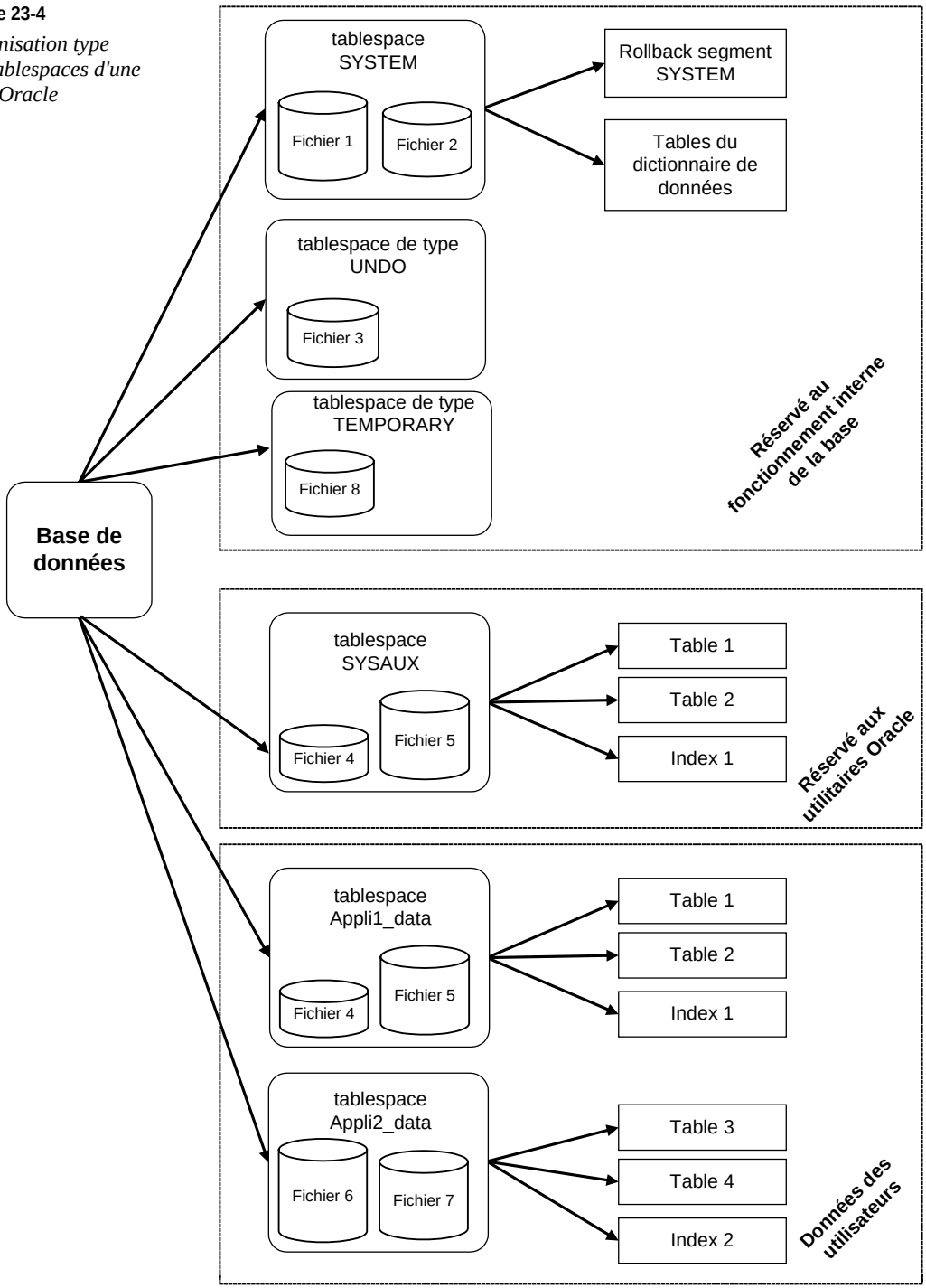
Figure 23-3
Principe d'une base « minimale »



- d'un tablespace de type UNDO pour gérer les données en attente de validation ou d'annulation ;
- d'un tablespace de type TEMPORARY où Oracle disposera d'un espace disque pour effectuer des tris ;
- d'un tablespace SYSAUX réservé à la structure des données des utilitaires Oracle ;
- les deux tablespaces *Appli1_data* et *Appli2_data* sont destinés à contenir les données de deux applications différentes. On peut en imaginer d'autres ayant pour rôle de contenir les index des deux applications.

Depuis Oracle9*i*, vous n'avez plus besoin de gérer des rollback segments dans un tablespace spécifique. Cette gestion est automatique dans le tablespace de type UNDO. De même, le tablespace de type TEMPORARY est affecté par défaut pour la gestion interne des tris de tous les utilisateurs.

Figure 23-4
Organisation type des tablespaces d'une base Oracle



La gestion des tablespaces normaux, SYSAUX, UNDO et TEMPORARY, est étudiée dans ce chapitre.

Le DB_BLOCK_SIZE

Le bloc Oracle est une unité exprimée en octets qui sert d'unité d'échange entre les fichiers, la mémoire et les processus. C'est un multiple de la taille des blocs manipulés par le système Windows. La taille d'un bloc Oracle est précisée lors de la création de la base de données. Sa valeur, une fois la base créée, ne peut plus être modifiée.

> Le paramètre définissant la taille du bloc Oracle est le paramètre DB_BLOCK_SIZE, contenu dans le fichier d'initialisation de la base initSID.ora.

Une fois l'instance démarrée, la valeur peut être consultée par la vue V$PARAMETER :

```
col name for a30
col value for a30
select name, value from v$parameter
     where name = 'db_block_size' ;

NAME            VALUE
--------------- -------------
db_block_size   4096
```

Attribuer au DB_BLOCK_SIZE une taille importante permet de limiter les accès disques, car chaque opération de lecture « monte » plus de données en mémoire. En cas de modification d'un seul enregistrement situé dans un bloc, celui-ci est écrit dans son intégralité sur disque, générant ainsi une forte charge sur les entrées/sorties disque pour de faibles modifications.

> En règle générale, on limitera la taille du DB_BLOCK_SIZE à 2 048 pour des applications privilégiant la mise à jour de données (transactionnelles), on l'augmentera (8 192 et au-delà) pour des applications nécessitant beaucoup de lecture de données (décisionnelles, infocentre, datawarehouse). La valeur 4 096 convient à la majorité des applications.

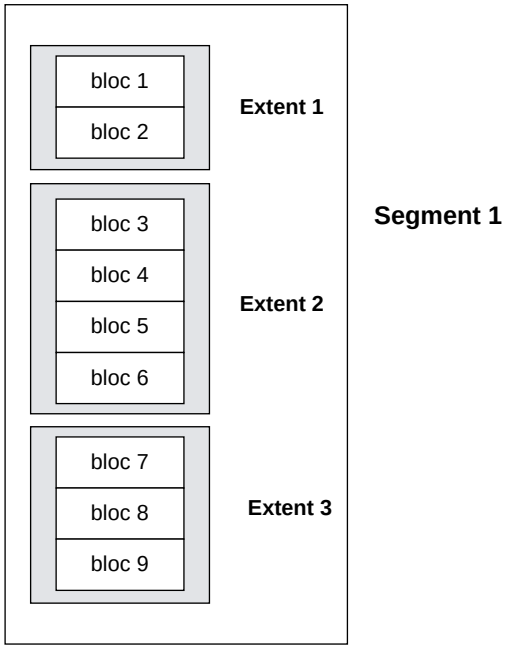
Pour permettre de déplacer des tablespaces de base à base, Oracle permet maintenant d'accueillir jusqu'à quatre DB_BLOCK_SIZE différents pour une base. Par défaut, tous les tablespaces ont un DB_BLOCK_SIZE identique à celui du tablespace SYSTEM sauf si cette valeur est modifiée lors de la création d'un nouveau tablespace. Cette possibilité dépasse le cadre de ce livre.

Segments, extents et blocs

Toute la gestion de l'espace disque des fichiers de données d'une base Oracle repose sur les concepts de segment, d'extent et de bloc.

Un ensemble de blocs contigus est nommé un extent (ou extension). C'est un ensemble logique. Un ensemble d'extents compose un segment. Le segment est un ensemble logique supérieur à l'extent.

Figure 23-5
Liens entre bloc, extent, segment



Dans une base Oracle 10*g*, il existe différents types de segments. Par exemple, l'ensemble des données d'une table est conservé dans un segment de données. De même, un segment d'index contient un index. Les autres types de segment sont utilisés pour le fonctionnement interne d'Oracle. Ce sont les segments temporaires où sont effectués les tris, les segments d'annulation utilisés pour la lecture consistante des données, et les bootstrap segments ou segments d'amorçage.

> Lors de la création d'une table ou d'un index, un extent initial est alloué au segment. Si la taille de l'objet augmente, des extents sont ajoutés au segment. Ce mécanisme est la clé de toute la gestion de l'espace disque d'Oracle 10*g*.

Le premier bloc du premier extent contient le segment header (l'en-tête de segment). Ainsi, lors de la création de la table EMP, le premier bloc du premier extent de la table possède l'en-tête de segment. Entre autres choses, l'en-tête contient la liste de tous les extents du segment ainsi que leur taille. En effet, si les blocs d'un extent sont forcément contigus, les différents extents d'un même segment peuvent être situés n'importe où dans les fichiers du tablespace. On comprend alors l'importance de la notion d'extent pour les performances d'une base de données. Les opérations les plus lentes d'un système étant

les lectures disque, il faut éviter de multiplier le nombre d'extents d'un segment. Cela garantit alors que tous les blocs du segment sont contigus.

Lorsque l'on conçoit une base de données, il faut être attentif à ces considérations de segment et d'extent, pour anticiper l'évolution de la base et disposer des meilleures performances possibles.

Les différents types de fichiers

Oracle peut créer des fichiers sur des systèmes de fichiers différents :

1. « classique » (NTFS sous Windows, ext3 sous Linux…) ;
2. « raw device » pour bénéficier de très bonnes performances d'entrées / sorties disque et pour permettre à deux serveurs d'accéder simultanément à des fichiers communs (cas des clusters) ;
3. Automatic Storage Location (ASM), pour mettre directement à disposition d'Oracle non plus des partitions, des systèmes de fichiers, mais directement des disques ! Oracle s'assure alors de toutes les opérations de bas niveau (mirroring, stripping…) et gère à 100 % l'espace alloué.

Ces points sont présentés au chapitre 27, *Configurer les disques pour Oracle*.

La gestion des tablespaces en mode LOCAL

Depuis Oracle9*i*, une nouvelle possibilité est apparue pour gérer les blocs et les segments. Auparavant, le paramétrage précis de la clause *storage* liée aux tablespaces, tables, index, etc. vous permettait de « piloter » Oracle pour optimiser l'allocation des blocks et des segments. Même si les paramètres de gestion d'espace étaient modifiables par la suite, c'était une opération délicate à assurer car elle nécessitait de connaître la taille des objets et de prévoir leur évolution au moment de leur création.

Maintenant, Oracle propose de gérer automatiquement tout ce fonctionnement interne. Le choix du nouveau mode de gestion ou de l'ancien est fait lors de la création du tablespace et tous les objets qu'il accueillera obéiront à ce mode de gestion. L'ancien mode s'appuie sur le dictionnaire de données, le nouveau mode sur une gestion locale au sein du tablespace.

C'est pourquoi l'ordre CREATE TABLESPACE a été modifié. Les options EXTENT MANAGEMENT LOCAL et EXTENT MANAGEMENT DICTIONARY sont apparues :

```
CREATE TABLESPACE test1
 DATAFILE 'c:\oracle\oradata\TEST\test1_01.dbf' SIZE 20M
 EXTENT MANAGEMENT LOCAL ;

CREATE TABLESPACE test2
 DATAFILE 'c:\oracle\oradata\TEST\test2_01.dbf' SIZE 20M
 EXTENT MANAGEMENT DICTIONARY
```

```
DEFAULT STORAGE ( INITIAL 100 K
                  NEXT 100 K
                  MINEXTENTS 2
                  MAXEXTENTS 100
                  PCTINCREASE 0 );
```

Si aucune indication n'est précisée, Oracle choisi par défaut le mode LOCAL. Seul le tablespace SYSTEM doit obligatoirement être en gestion DICTIONARY. Une fois créé, il est impossible de modifier le mode de gestion d'un tablespace.

Différences entre la gestion locale et dictionnaire

Apparue discrètement avec Oracle8*i* ; la gestion locale est généralisée avec Oracle 10*g*. Auparavant, la gestion interne d'allocation d'espace était centralisée au sein du dictionnaire de données de chaque base.

La gestion locale stocke tous les aspects d'allocation de segments à l'intérieur de chaque tablespace. Les informations sont codées au format *bitmap* dans l'en-tête de chaque tablespace. Cette évolution apporte les avantages suivants :

- La gestion locale (interne à chaque tablespace) facilite les opérations lors du déplacement de tablespaces de base à base. Les accès au dictionnaire de données sont limités.
- Le travail de l'administrateur Oracle est facilité, les fastidieuses opérations d'allocation d'espace ne sont plus nécessaires.
- La gestion centralisée dans le dictionnaire imposait des écritures dans les tables correspondantes lors de chaque allocation ou libération d'espace. L'écriture de données *bitmap* est plus rapide et évite ce goulot d'étranglement. Oracle peut supporter encore plus d'utilisateurs.
- Lors de la libération d'espace, le stockage dans les tables du dictionnaire ne permettait pas à Oracle de détecter que deux espaces libres contigus pouvaient être agrégés. Cette opération devait être effectuée par l'ordre ALTER TABLESPACE… COALESCE. La gestion locale automatise cette opération.
- Vous n'avez plus besoin de paramétrer la clause *storage* pour les tablespaces, tables et index. Cette opération est automatique en gestion locale.
- Vous pouvez maintenant déplacer un tablespace d'une base pour la raccrocher à une autre. C'est très utile pour créer très rapidement un environnement de test. La composition du tablespace étant locale, elle se déplace en même temps que le tablespace.

Les options des tablespaces en gestion locale

Ces options ne s'appliquent qu'aux tablespaces en mode EXTENT MANAGEMENT LOCAL.

Lorsqu'un tablespace local est créé, le mode de gestion d'espace pour l'allocation de nouveaux segments peut être précisé :

- AUTOALLOCATE : c'est l'option par défaut. Oracle gère automatiquement toutes les allocations d'espace dans le tablespace. Elles peuvent être de tailles différentes ;
- UNIFORM SIZE : toute allocation d'espace sera de cette taille (minimum 1 Mo). Le tablespace TEMPORARY utilise cette option par défaut. Le tablespace UNDO ne peut pas l'utiliser. Cette option est optimale en cas de fréquentes créations / suppressions de tables et d'index. Vous êtes certain que l'espace alloué ou désalloué sera utilisable par n'importe quel autre objet. Le remplissage du tablespace est optimum.

Pour chaque cas, la syntaxe utilisée est alors ;

```
CREATE TABLESPACE test1
 DATAFILE 'c:\oracle\oradata\TEST\test1_01.dbf' SIZE 20M
 EXTENT MANAGEMENT LOCAL AUTOALLOCATE;

CREATE TABLESPACE test1
 DATAFILE 'c:\oracle\oradata\TEST\test1_01.dbf' SIZE 20M
 EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M;
```

En gestion locale des tablespaces, nous vous recommandons d'utiliser l'option UNIFORM SIZE. Si vous voulez piloter l'allocation d'espace, utilisez des tablespaces en mode AUTOALLOCATE, voire DICTIONARY car la clause storage est adaptée à cet usage.

Les options précédentes concernent l'allocation d'espace, c'est-à-dire lorsque des objets sont créés ou augmentent de taille. Un autre aspect que doit gérer une base de données est la récupération d'espace libéré. C'est le cas lors de la suppression d'enregistrements dans une table et de l'impact sur ses index.

Pour un tablespace en mode local, la gestion des espaces libres peut être manuelle ou automatique. Si elle est manuelle, des seuils sont définis. Ils indiquent à Oracle quand il doit effectuer des regroupements d'espace contigus. Si elle est automatique, aucun seuil n'est à préciser.

Par défaut, Oracle propose l'ancienne option manuelle tout en recommandant fortement d'utiliser l'option automatique. D'où vient cette contradiction ? Actuellement, les données de type LOB ne peuvent être utilisées dans un tablespace local dont la gestion de l'espace libre est automatique. Oracle propose donc par défaut l'option la plus générale, celle qui permet de stocker tous les types de données possibles. Cette limite sera certainement levée dans une future version.

Si vous avez des doutes concernant le type de données que contiendra le tablespace, utilisez l'ancien mode de gestion manuelle car il a fait ses preuves. Si vous êtes certain qu'aucune donnée de type LOB ne sera dans le tablespace, Oracle recommande le mode automatique.

Dans le cas d'une gestion manuelle, les valeurs PCTUSED, FREELIST, FREELISTS GROUPS peuvent être précisées. Nous vous recommandons de ne pas les modifier et de laisser les valeurs par défaut. Comme cette option MANUAL est actuellement choisie par défaut, vous pouvez aussi tout simplement ignorer cette clause ;

```
CREATE TABLESPACE test1
 DATAFILE 'c:\oracle\oradata\TEST\test1_01.dbf' SIZE 20M
 EXTENT MANAGEMENT LOCAL AUTOALLOCATE
 SPACE MANAGEMENT MANUAL ;
```

En cas d'une gestion automatique des espaces libres, la syntaxe est alors :

```
CREATE TABLESPACE test1
 DATAFILE 'c:\oracle\oradata\TEST\test1_01.dbf' SIZE 20M
 EXTENT MANAGEMENT LOCAL AUTOALLOCATE
 SPACE MANAGEMENT AUTO ;
```

Grâce aux tablespaces à gestion locale, Oracle a supprimé la cause de nombreuses opérations de réorganisation de données. Les administrateurs Oracle ont ainsi supprimé les nuits ou les week-ends passé à effectuer ces tâches. Profitez-en !

Comment déterminer le mode de gestion des tablespaces ?

Le mode de gestion des différents tablespaces est accessible depuis la vue SYS.DBA _TABLESPACES.

```
select tablespace_name, contents, extent_management
  from dba_tablespaces

TABLESPACE_NAME                CONTENTS  EXTENT_MANAGEMENT
------------------------------ --------- -----------------
SYSTEM                         PERMANENT DICTIONARY
UNDOTBS                        UNDO      LOCAL
DRSYS                          PERMANENT LOCAL
EXAMPLE                        PERMANENT LOCAL
INDX                           PERMANENT LOCAL
TEMP                           TEMPORARY LOCAL
TOOLS                          PERMANENT LOCAL
USERS                          PERMANENT LOCAL
```

La gestion des objets des tablespaces en mode DICTIONARY

Ce paragraphe décrit en détail le travail que l'administrateur Oracle doit effectuer pour un tablespace créé avec le mode EXTENT MANAGEMENT DICTIONARY. Vous pouvez l'ignorer si vous décidez d'utiliser des tablespaces en gestion locale. Nous vous conseillons pourtant de le lire car il présente les actions effectuées automatiquement par Oracle avec la gestion LOCAL. C'est aussi le mode de fonctionnement obligatoire du tablespace SYSTEM.

Oracle recommande fortement de gérer les tablespaces avec le mode LOCAL, vu précédemment. Le mode DICTIONARY est obligatoire pour le tablespace SYSTEM. Ce doit être le seul dans ce cas. Nous conservons la description du mode DICTIONARY, car c'était la seule possibilité des versions précédant Oracle 9i. Vous serez à même de juger la simplicité de ce nouveau mode de gestion par rapport à l'ancien.

Création d'un tablespace en mode DICTIONARY

Un tablespace créé en mode de gestion dictionnaire doit contenir la clause EXTENT MANAGEMENT DICTIONARY.

```
CREATE TABLESPACE test2
 DATAFILE 'c:\oracle\oradata\TEST\test2_01.dbf' SIZE 20M
 EXTENT MANAGEMENT DICTIONARY
 DEFAULT STORAGE ( INITIAL 100 K
                   NEXT 100 K
                   MINEXTENTS 2
                   MAXEXTENTS 100
                   PCTINCREASE 0 );
```

La clause *storage* fixe les valeurs par défaut qui seront reprises pour tous les segments de table et d'index créés dans le tablespace.

La clause STORAGE

Oracle gère de façon identique les extents des segments de données et d'index. La clause de STORAGE permet de piloter le mécanisme des extents d'un segment.

> La clause STORAGE peut être spécifiée lors de la création du segment ou modifiée après. Dans ce cas, elle n'agira pas sur les segments déjà créés, mais uniquement sur les segments futurs.

Par exemple, pour la création de la table EMP, nous pouvons utiliser la syntaxe suivante :

```
CREATE TABLE emp (
    ...
    liste des colonnes de la table...
    ...
    )
    STORAGE (
           INITIAL 100K
           NEXT 100 K
           PCTINCREASE 0 )
    TABLESPACE user_data;
```

- INITIAL précise la taille de la première extension. Tout segment possède une taille initiale à sa création.
- NEXT précise la taille de la deuxième extension. Lorsque la première est remplie, Oracle alloue automatiquement une seconde de NEXT octets. Toutes les extensions futures se fondent sur cette valeur.
- PCT_INCREASE est un paramètre que nous positionnons toujours à zéro, mais que certaines personnes utilisent. Ce pourcentage dirige l'incrément de taille des extensions suivantes. Par exemple, INITIAL = 1 Mo, NEXT = 1 Mo et PCT_INCREASE = 50 %. La première extension sera de 1 Mo, la deuxième de 1 000 + 50 % (1 000) = 1,5 Mo, la troisième de 2,25 Mo, etc. Calculez la taille nécessaire pour la vingtième extension !
- TABLESPACE indique le tablespace cible dans lequel sera créée la table EMP.

C'est grâce à cette clause STORAGE que vous pouvez piloter le comportement des segments de données et d'index.

Quelle que soit l'extension, elle ne peut être d'une taille supérieure à la plus grande place libre contiguë, au sein du tablespace cible. Même si vous disposez d'une taille libre importante au total, vous pouvez être bloqué si elle n'est pas contiguë. La vérification de l'espace libre contigu dans un tablespace est abordée dans ce chapitre, au paragraphe traitant des tablespaces.

Que se passe-t-il lorsque aucune clause STORAGE n'est définie ?

Dans le cas où aucune clause STORAGE n'est définie, Oracle lui attribue automatiquement des valeurs par défaut. En effet, un objet est toujours créé sous un compte utilisateur et, dans ses caractéristiques, un utilisateur indique nécessairement un tablespace par défaut pour créer les segments. Ce tablespace possède toujours des valeurs de STORAGE par défaut.

Dans l'exemple suivant, nous interrogeons la vue SYS.DBA_USERS pour connaître les tablespaces par défaut et les tablespaces de tri des utilisateurs.

```
select username, default_tablespace, temporary_tablespace
 from sys.dba_users;

USERNAME             DEFAULT_TABLESPACE  TEMPORARY_TABLESPACE
-------------------- ------------------ ----------------------
SYS                  SYSTEM              SYSTEM
SYSTEM               SYSTEM              TEMP
SCOTT                USERS               TEMP
PASCAL               USERS               TEMP
JEAN                 USERS               TEMP
GILLES               USERS               TEMP
ISABELLE             USERS               TEMP
OEM204               OEM_REPOSITORY      TEMP
```

Pour connaître les détails de la clause STORAGE d'un tablespace particulier, il faut consulter la vue SYS.DBA_TABLESPACES.

```
select tablespace_name, initial_extent, next_extent,
 min_extents, max_extents, pct_increase
 from sys.dba_tablespaces;

Tablespace     Ini.Ext  Nex.Ext  Min.Ext   Mext  Pct.In
------------- -------- -------- -------- ------ -------
SYSTEM           10240    10240        1    121      50
USERS            10240    10240        1    121       0
RBS              10240    10240        1    121       0
TEMP             10240    10240        1    121       0
OEM_REPOSITORY   10240    10240        1    121      50
```

Les caractéristiques de la clause STORAGE existent pour chacun des tablespaces.

Comment contrôler le nombre d'extensions d'un segment ?

Nous avons vu que le segment d'une table ou d'un index peut posséder plusieurs extensions. Afin de les visualiser, utilisez la vue SYS.DBA_SEGMENTS pour une lecture globale et SYS.DBA_EXTENTS pour une vue détaillée.

Par exemple, créons une table possédant des extensions :

```
 -- connexion sous l'utilisateur SCOTT
connect scott/tiger

-- création de la table TEST
create table test as select * from scott.emp ;

-- création d'enregistrements (double à chaque fois le nombre)
insert into test select * from test;

14 ligne(s) créée(s).

insert into test select * from test;

28 ligne(s) créée(s).

... continuer...

insert into test select * from test;

7168 ligne(s) créée(s).

 -- validation
commit;
```

L'ordre SQL suivant visualise le nombre d'extensions de tous les segments propriété de l'utilisateur SCOTT.

```
-- visualisation des extensions sous un compte administrateur
connect system/manager

ttitle center "Place utilisée par les segments de SCOTT " skip -
     center "(tri par type de segment)" skip 2

set pagesize 150
col owner for a8
col segment_name for a17
col segment_type for a9 heading "Type Seg"
col extents for 9999 heading "Ext"
col max_extents for 99999 heading "Mext"

select segment_name,tablespace_name,
 segment_type,extents,max_extents,bytes,owner
 from sys.dba_segments
 where owner = 'SCOTT'
```

```
 order by segment_type desc, segment_name
/

           Place utilisée par les segments de SCOTT
                 (tri par type de segment)

Nom.Seg   Nom Tbs   Type Seg   Ext   Mext     Bytes  Owner
--------- --------- --------- ----- ------ --------- -------
ACCOUNT   USERS     TABLE         1    121     10240 SCOTT
BONUS     USERS     TABLE         1    121     10240 SCOTT
DEPT      USERS     TABLE         1    121     10240 SCOTT
EMP       USERS     TABLE         1    121     10240 SCOTT
RECEIPT   USERS     TABLE         1    121     10240 SCOTT
SALGRADE  USERS     TABLE         1    121     10240 SCOTT
TEST      USERS     TABLE        10    121    880640 SCOTT
PK_DEPT   USERS     INDEX         1    121     10240 SCOTT
PK_EMP    USERS     INDEX         1    121     10240 SCOTT
```

Dans cet ordre SQL, SCOTT ne possède que des segments de type table et index. Ils possèdent tous une seule extension, sauf la table TEST, d'une taille totale de 880 Ko, qui en possède dix. Cet ordre SQL, utilisé à l'annexe 3, *Procédures pour le DBA*, permet de visualiser très rapidement les tables et index qui possèdent des extensions.

L'ordre SQL suivant fournit le détail de chacune des extensions de la table TEST.

```
ttitle center "Détail des extents du segment SCOTT.TEST" skip 2

select owner, segment_name, segment_type, tablespace_name,
  extent_id, bytes
  from sys.dba_extents
  where segment_name = 'TEST'
  and owner = 'SCOTT'
  order by extent_id
/

           Détail des extents du segment SCOTT.TEST

Owner    Nom. Seg.  Type Seg  Tablespace  EXTENT_ID   BYTES
-------- ---------- --------- ---------- ---------- -------
SCOTT    TEST       TABLE     USERS      0          10240
SCOTT    TEST       TABLE     USERS      1          10240
SCOTT    TEST       TABLE     USERS      2          20480
SCOTT    TEST       TABLE     USERS      3          30720
SCOTT    TEST       TABLE     USERS      4          40960
SCOTT    TEST       TABLE     USERS      5          61440
SCOTT    TEST       TABLE     USERS      6          92160
SCOTT    TEST       TABLE     USERS      7         133120
SCOTT    TEST       TABLE     USERS      8         194560
SCOTT    TEST       TABLE     USERS      9         286720
```

L'ordre SQL précédent montre le détail de chacune des dix extensions de la table TEST. Leur taille totale est de 880 Ko. La dimension de chaque extent évolue, cela signifie que le paramètre PCTINCREASE de la clause STORAGE n'est pas nul. L'ordre SQL suivant donne le détail de la clause STORAGE de la table TEST.

```
ttitle center "Paramètres de storage de SCOTT.TEST" skip 2

select owner, table_name,
    initial_extent, next_extent,
    min_extents, max_extents, pct_increase
  from sys.dba_tables
  where table_name = 'TEST'
  and owner = 'SCOTT'
/

             Paramètres de storage de SCOTT.TEST

Owner    Table  Ini. Ext. Next.Ext.  Min. Ext.  Mext Pct.Increase
-------- ------- ------- ----------- -------- ------ ------------
SCOTT    TEST      10240      430080        1    121           50
```

On constate que le segment initial est de 10 Ko, que le PCTINCREASE est de 50 % et que la prochaine extension (la onzième) sera de 430 Ko.

Modifier les caractéristiques de la clause STORAGE

Vous pouvez modifier les caractéristiques de la clause STORAGE pour un tablespace, une table ou un index. Ainsi, si l'on souhaite que la prochaine extension de la table TEST soit de 200 Ko et que le paramètre PCTINCREASE soit à zéro, l'ordre SQL suivant permet de modifier ces caractéristiques liées à la table.

```
alter table scott.test
 storage ( next 200k
           pctincrease 0);
```

> Notez que la clause INITIAL ne peut être modifiée une fois le segment créé.

La vérification des paramètres de stockage de la table TEST indique les nouvelles valeurs.

```
             Paramètres de storage de SCOTT.TEST

Owner    Table  Ini. Ext. Next.Ext.  Min. Ext.  Mext Pct.Increase
-------- ------- ------- ----------- -------- ------ ------------
SCOTT    TEST      10240      200080        1    121            0
```

Comme évoqué précédemment, la modification des clauses de storage n'affecte pas les extensions existantes, elle influe uniquement sur les futures extensions.

La fragmentation est-elle pénalisante ?

Oui, la fragmentation est pénalisante. En effet, les extensions sont disséminées au sein du tablespace et les déplacements du disque dur sont beaucoup plus fréquents que dans le cas où les extensions sont contiguës. Il faut toutefois rapprocher la taille du segment du nombre d'extensions. Une table de 100 Mo possédant cinq extensions est moins pénalisante qu'un index d'1 Mo doté lui aussi de cinq extensions, car les index ont pour but d'accélérer la recherche des données.

> Heureusement, les nouvelles options proposées par Oracle apportent beaucoup de souplesse et évitent beaucoup de travail aux administrateurs.

Défragmenter une table

> Avant toute opération de défragmentation, effectuez une sauvegarde de la base. Même s'il s'agit d'une opération mineure, cela assure une sécurité parfaite, quelle que soit la situation rencontrée. Ces opérations de maintenance doivent toujours être effectuées offline, sans utilisateur actif sur la base.

Les techniques utilisées dépendent du type de segment. Pour une table, vous pouvez effectuer les tâches suivantes :

1. Renommer la table.
2. Créer une nouvelle table à partir de la table renommée.

Par exemple, pour notre table TEST :

```
-- connexion sous l'utilisateur SCOTT
connect scott/tiger

-- renommer la table TEST
rename TEST to TEST_OLD ;

-- créer une nouvelle table avec une clause STORAGE
-- les données provenant de TEST_OLD sont insérées dans TEST.
-- La structure de la table TEST est issue de TEST_OLD.

create table TEST
  storage (initial 800K
           next 100K
           pctincrease 0 )
  as select * from TEST_OLD
/
```

Visualisation des extensions sous un compte administrateur :

```
connect system/manager

ttitle center "Place utilisée par les segments de SCOTT " skip -
       center "(tri par type de segment)" skip 2
```

```
set pagesize 150
col owner for a6 heading "Owner"
col tablespace_name for a10 heading "Tablespace"
col table_name for a10 heading "Table"
col segment_name for a8 heading "Segment"
col segment_type for a9 heading "Type Seg"
col extents for 999 heading "Ext"
col max_extents for 999 heading "Mext"

select segment_name,tablespace_name,segment_type,
      extents,max_extents,bytes,owner
  from sys.dba_segments
      where owner = 'SCOTT'
      order by segment_Type desc, segment_name
/

          Place utilisée par les segments de SCOTT
                  (tri par type de segment)

Segment  Tablespace Type Seg   Ext Mext     BYTES Owner
-------- ---------- --------- ---- ---- --------- ------
...
TEST     USERS      TABLE        1  121    819200 SCOTT
TEST_OLD USERS      TABLE       10  121    880640 SCOTT
...
```

La table TEST_OLD a toujours dix extensions alors que TEST n'en a qu'une. Visualisons maintenant les caractéristiques de storage de la nouvelle table TEST :

```
ttitle center "Paramètres de storage de SCOTT.TEST" skip 2
col initial_extent for 999999 heading "Initial"
col next_extent    for 999999 heading "Next"
col min_extents    for 999999 heading "Min.Ext"
col max_extents    for 999999 heading "Max.Ext"
col pct_increase   for 999999 heading "Pct.Incr"

select owner, table_name,
      initial_extent, next_extent,
      min_extents, max_extents, pct_increase
   from sys.dba_tables
      where table_name = 'TEST'
      and owner = 'SCOTT'
/

             Paramètres de storage de SCOTT.TEST

Owner  Table      Initial    Next Min.Ext Max.Ext Pct.Incr
------ ---------- ------- ------- ------- ------- --------
SCOTT  TEST        819200  102400       1     121        0
```

> Cette première méthode est utilisable si la table concernée n'est pas trop volumineuse et si la base est d'une taille suffisante pour héberger à la fois l'original et une copie.

La deuxième technique pour défragmenter une table consiste à recourir à l'Export/Import. Elle est abordée au chapitre 18, *Les outils d'Oracle 10*g.

Défragmenter un index

Pour défragmenter un index, vous pouvez le détruire puis de le créer à nouveau. Comme pour une table, vous pouvez utiliser l'Export/Import, traité au chapitre 18, *Les outils d'Oracle 10*g.

Vous pouvez aussi utiliser des ordres SQL de reconstruction comme ALTER INDEX REBUILD ou ALTER INDEX COALESCE.

Gestion des tablespaces et des fichiers de données

En complément des présentations théoriques du chapitre 9, *Les fichiers d'une base Oracle 10*g, et du paragraphe précédent, nous présentons maintenant les outils de manipulation des tablespaces des fichiers de données associés, utilisés aussi pour dimensionner correctement les différents objets.

Le mode de gestion des extensions LOCAL est proposé par défaut par Oracle. Nous l'utiliserons donc dans tous les exemples de ce paragraphe.

> Avant toute action, effectuez une sauvegarde de la base et entraînez-vous sur une base test, sans utilisateur actif.

Création d'un tablespace

La création d'un nouveau tablespace implique d'y associer obligatoirement un fichier de données en précisant sa taille. La commande CREATE TABLESPACE permet d'établir un nouveau tablespace avec ses propres paramètres de stockage.

```
create tablespace user_data
datafile 'c:\oracle\oradata\TEST\user01.dbf' size 50M ,
         'D:\oracle\oradata\TEST\user02.dbf' size 100M
extent management local ;
```

Dans cet exemple, le tablespace est créé avec deux fichiers situés sur des disques différents.

La gestion automatique des fichiers proposée par Oracle 10*g* limite les informations à donner. Cette possibilité est abordée en fin de chapitre.

Définir un tablespace par défaut

« Ne créez jamais d'objets dans le tablespace SYSTEM ! » C'est l'une des premières recommandations faites à tout administrateur Oracle. Par analogie, il ne faut jamais créer

d'objets « utilisateurs » sous le compte administrateur d'un serveur. Le tablespace SYSTEM contient le dictionnaire de données ainsi que d'autres objets nécessaires à Oracle. Pour éviter de placer malencontreusement des objets dans le tablespace SYSTEM, les administrateurs devaient tout d'abord créer un nouveau tablespace pour les données utilisateurs puis créer les utilisateurs :

```
CREATE USER nouvel_utilisateur
IDENTIFIED BY son_mot_de_passe
DEFAULT TABLESPACE tablespace_utilisateur
QUOTA LIMITED ON tablespace_utilisateur ;
```

Malheureusement, il était trop facile d'oublier d'indiquer un tablespace par défaut lorsqu'on créait un nouvel utilisateur et, en absence d'information, Oracle plaçait par défaut les objets de cet utilisateur dans le tablespace SYSTEM. Oracle 10*g* apporte la notion de tablespace par défaut pour toute la base. Il suffit de l'indiquer une seule fois au plus haut niveau :

```
ALTER DATABASE
DEFAULT TABLESPACE tablespace_utilisateur;
```

Tout nouvel utilisateur créé qui n'aura pas de tablespace par défaut utilisera alors celui précisé au plus haut niveau. C'est un avantage qui limite les erreurs lors de la création d'utilisateurs.

Les tablespaces temporaires

Oracle effectue de nombreux tris à la demande des utilisateurs ou pour son fonctionnement interne. Ils sont principalement effectués en mémoire, dans la zone SORT_AREA _SIZE de la SGA. Si cette zone s'avère trop petite, un espace disque est réservé à cet effet : c'est le tablespace temporaire ou TEMPORARY TABLESPACE. De même qu'il existe des segments de données et d'index, le segment de tri est un *sort segment*.

Si un tablespace temporaire n'est pas créé, les tris sont effectués dans le tablespace SYSTEM qui contient le dictionnaire. Cela n'est pas son rôle et il convient de toujours créer un tablespace temporaire.

Une fois créé, un tablespace temporaire ne peut pas être modifié pour contenir des données. Par contre, comme pour tous les autres tablespaces, on peut le mettre en ligne, hors ligne et lui ajouter des fichiers.

Un tablespace temporaire peut être géré en mode local ou dictionnaire. Oracle recommande fortement de le gérer en mode local. La syntaxe utilisée est la suivante :

```
create temporary tablespace temp_data
tempfile 'c:\oracle\oradata\TEST\temp_data01.dbf' size 50M
extent management local ;
```

Remarquez qu'un tablespace temporaire n'utilise pas comme fichier un *datafile* mais un *tempfile*.

Une fois le tablespace temporaire créé, Oracle l'utilisera pour tous les tris des utilisateurs, sans qu'il soit nécessaire d'indiquer pour chaque utilisateur quel est son tablespace de tri.

Les vues V$TEMPFILE et DBA_TEMP_FILES permettent d'accéder à toutes les caractéristiques des tablespaces temporaires.

Avant Oracle9*i*, pour utiliser un tablespace temporaire, vous deviez le préciser pour chaque utilisateur :

```
CREATE USER nouvel_utilisateur
IDENTIFIED BY son_mot_de_passe
TEMPORARY TABLESPACE temp;
```

Si vous oubliez d'indiquer la clause TEMPORARY TABLESPACE lors de la création d'un utilisateur, le tablespace SYSTEM était utilisé pour toutes les actions de tri, de groupement, de jointure et autres activités temporaires sur disque.

Cela occasionnait alors une activité disque supplémentaire sur le tablespace SYSTEM qu'il convenait d'éviter. Ces opérations de tri s'ajoutaient alors à l'activité propre du tablespace SYSTEM et pouvaient ainsi créer des points de contention sur les accès disque. Depuis Oracle9*i*, vous pouvez indiquer un tablespace par défaut pour toute la base de données :

```
ALTER DATABASE DEFAULT TEMPORARY TABLESPACE temp;
```

Après avoir exécuté cette commande, tous les utilisateurs créés **après** utiliseront le tablespace temporaire indiqué :

```
ALTER USER toto DEFAULT TEMPORARY TABLESPACE temp;
```

Si des utilisateurs ont été créés, vous pouvez corriger cela simplement en créant automatiquement une série d'ordres SQL :

```
select 'ALTER USER '||username||' DEFAULT TEMPORARY TABLESPACE temp;'
from DBA_USERS ;
```

L'accès simultané de tous les utilisateurs à cet espace de tri pouvant créer un point de contention disque, Oracle permet de répartir les accès en utilisant la notion de tablespaces temporaires groupés.

Les tablespaces UNDO ou d'annulation

Un tablespace de type UNDO (Oracle9*i*) est un nouveau type de tablespaces destiné à remplacer l'ancien mode de gestion des segments d'annulation ou rollback segments. C'est dans cet espace qu'Oracle gère les données en attente de validation ou d'annulation. L'utilisation d'un tablespace UNDO est beaucoup plus simple que l'ancien mode. Ils sont tous deux abordés plus loin dans ce chapitre.

Les tablespaces BIGFILES

Les tablespaces BIGFILES permettent d'utiliser les possibilités des systèmes 64 bits de gérer de très gros fichiers. Un tablespace BIGFILE ne contient qu'un seul fichier qui peut comporter jusqu'à 2^{32} blocs Oracle, soit environ 8 exaoctets ou à peu près 8 millions de teraoctets…

On peut s'interroger sur l'intérêt de créer de tels fichiers. Leur taille gigantesque dépasse les possibilités techniques actuelles. Imaginez-vous comment sauvegarder un fichier d'une taille dépassant celle de tous les médias de sauvegarde ? Ce type d'annonce ressemble à une annonce marketing. La taille d'un fichier : *no limit* !

Modification d'un tablespace

La commande ALTER permet de modifier les caractéristiques d'un tablespace existant.

```
alter tablespace user_data online ;
```

Renommer un tablespace

Vous l'avez longtemps attendu, c'est maintenant chose faite ! Vous pouvez renommer vos tablespaces en une seule commande, à l'exclusion des tablespaces SYSTEM et SYSAUX réservés par Oracle.

```
alter tablespace tbs_test rename tbs_production ;
```

> Les règles de nommage de fichiers recommandent que le nom du fichier indique le tablespace auquel il se rattache. En changeant le nom du tablespace, vous perdez cette indication.

C'est très intéressant lors du transport d'un tablespace d'une base à une autre.

> Autre point, vous devez sauvegarder votre fichier de contrôle sitôt la modification faite, le nom du tablespace et des fichiers attachés y figurant. C'est important pour la validité de vos sauvegardes.

Ajouter un fichier de données

Il existe deux possibilités : la première consiste à créer un nouveau tablespace et à lui associer le nouveau fichier, la seconde à ajouter un nouveau fichier à un tablespace existant.

```
alter tablespace user_data
  add datafile 'D:\oracle\oradata\TEST\user02.dbf' size 50M;

alter temporary tablespace temp_data
  add tempfile 'c:\oracle\oradata\TEST\temp_data02.dbf' size 50M

alter tablespace undotbs
  add datafile 'c:\oracle\oradata\TEST\undotbs_02.dbf' size 50M
```

Il est possible d'ajouter un fichier à un tablespace mais il est impossible de supprimer un seul fichier d'un tablespace. Il faut alors supprimer le tablespace entier. En effet, dans un tablespace, une table importante peut être contenue dans plusieurs fichiers. Supprimer l'un des fichiers conduirait à un résultat imprévisible.

Compacter et réduire les segments en ligne

Le Segment Advisor (Oracle 10*g*) indique la liste des segments à réorganiser pour récupérer de la place dans les tablespaces. Il examine la fragmentation des objets ainsi qu'une estimation du taux de remplissage prévu. Il est accessible depuis Oracle Enterprise Manager (OEM) : *Centre de conseil* puis *Fonctions de conseils sur les segments*.

Une fois les segments à réorganiser détectés, la clause SQL SHRINK de récupération d'espace peut être lancée manuellement ou automatiquement depuis OEM. Elle optimise le stockage lorsque l'espace libre dans les blocs est trop important. C'est très utile pour :

- des tables qui font l'objet de beaucoup d'insertions / suppressions de lignes ;
- des tables utilisées temporairement dont l'espace libéré doit être désalloué.

Le compactage d'une table nécessite que le déplacement d'une table soit autorisé :

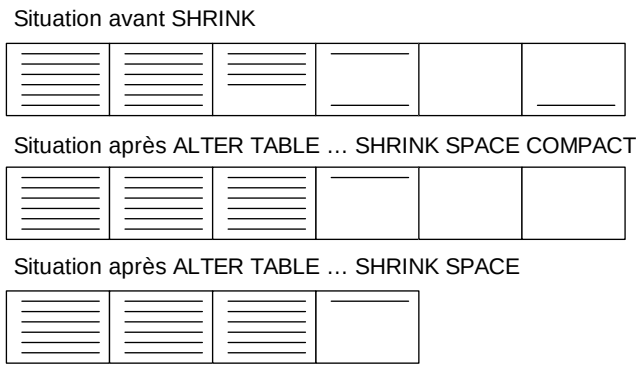
```
ALTER TABLE nom_table ENABLE ROW MOVMENT ;
```

La syntaxe SQL utilisée est :

```
ALTER [TABLE | INDEX] nom_table_index SHRINK SPACE [COMPACT][CASCADE]
```

> Cette option de récupération d'espace s'effectue en ligne et n'est utilisable que pour des tablespaces dont la gestion d'espace est en mode AUTO.

Figure 23-6
Utilisation du compactage d'une table ou d'un index



Sans préciser d'option, les données sont compactées dans les segments des tables et des index et l'espace libre est désalloué.

L'option COMPACT permet de compacter les données dans les segments mais sans désallouer l'espace occupé.

L'option CASCADE permet d'effectuer le compactage pour tous les segments dépendants (index, partitions de tables et d'index, vues matérialisées).

Certaines restrictions existent, comme l'impossibilité d'utiliser le SHRINK pour des tables contenant des données binaires de type Large Objects (LOB), LONG ou LONG ROW.

Extension automatique d'un fichier de données

Auparavant, l'un des problèmes les plus difficiles à résoudre pour les administrateurs Oracle était la gestion de la taille des tablespaces. Que faire lorsqu'un segment (de données, d'index, de tri, de rollback) cherche à s'étendre dans un tablespace et ne dispose plus de place dans les fichiers pour y créer un nouvel extent ? L'utilisateur qui lançait l'opération se heurtait alors à un blocage.

La principale difficulté était de prévoir le segment qui aurait besoin de place, ce qui était pratiquement impossible pour les segments de tri ou de rollback, car on ne peut prévoir a priori la place nécessaire à tous les utilisateurs qui exécutent simultanément des ordres SQL. La seule solution était de « voir large » et de dimensionner les fichiers de sorte que les tablespaces aient toujours beaucoup d'espace disponible. Le blocage était ainsi limité, au prix d'une surconsommation de l'espace disque.

Oracle permet que les fichiers des tablespaces augmentent automatiquement de taille :

```
alter database
    datafile 'c:\oracle\oradata\TEST\user01.dbf'
    autoextend on
    next 25M
    maxsize 2000M;
```

Dans cet exemple, la taille du fichier c :\oracle\oradata\TEST\user01.dbf peut augmenter automatiquement, par palier de 25 Mo jusqu'à la limite maximale de 2 000 Mo. Observez que cette option ne s'applique qu'à des fichiers et non au tablespace auquel appartient ce fichier, ce qui est tout à fait normal, car suivant l'emplacement physique des différents fichiers d'un tablespace, leurs points de montage ne disposent pas tous du même espace.

L'augmentation automatique de la taille des fichiers est possible pour tous les types de tablespaces (données, index, temporaire, UNDO, SYSTEM) qu'ils soient en mode de gestion locale ou dictionnaire.

```
create temporary tablespace temp_data
  tempfile 'c:\oracle\oradata\TEST\temp_data01.dbf' size 50M
  autoextend on
  next 20M
  maxsize 200M
  extent management local ;

create tablespace user_data
  datafile 'c:\oracle\oradata\TEST\user_data01.dbf' size 50M
  autoextend on
  next 100M
  maxsize 2000M
  extent management local ;

create tablespace undotbs_2
  datafile 'D:\oracle\oradata\TEST\undotbs_2_01.dbf' size 50M
  autoextend on
  next 100M
  maxsize 200M
  extent management local ;
```

Il est indispensable de fixer une taille limite pour que les fichiers ne puissent dépasser vos capacités de sauvegarde.

La vue SYS.DBA_DATA_FILES permet de contrôler les caractéristiques d'extension des fichiers.

```
select tablespace_name, file_name, autoextensible,
       maxbytes, increment_by
  from dba_data_files
       order by tablespace_name;

Tablespace  FILE_NAME              AUTOEXTENS  MAXBYTES  INCREMENT_BY
----------- ---------------------- ---------- ----------- ----------
INDX        c:\oradata\TEST\INDX01.dbf    YES  157286400   2560
OEM_REPOSIT c:\oradata\TEST\OEMREP01.dbf   NO  157286400   2560
RBS         c:\oradata\TEST\RBS01.dbf     YES  157286400   2560
SYSTEM      c:\oradata\TEST\SYSTEM01.dbf  YES  524288000   5120
TEMP        c:\oradata\TEST\TEMP01.dbf    YES  157286400   2560
USERS       c:\oradata\TEST\USERS01.dbf   YES  157286400   2560
```

Cet exemple montre que seul le tablespace OEM_REPOSITORY n'est pas en mode auto-extensible.

Les alertes d'utilisation des tablespaces

Vous pouvez maintenant définir des seuils permettant d'être automatiquement alerté en cas de dépassement. Pour les tablespaces, la signification des seuils est différente suivant le type de tablespace :

- UNDO : espace utilisé contenant des données en attente de validation ou d'annulation ;
- temporaire : espace utilisé par les sessions (tris, etc.) ;
- fichiers auto-extensibles : à l'approche de la taille maximale du fichier, dépendant de l'espace disponible sur le disque ou encore du système d'exploitation.

L'interface graphique d'OEM est bien utile pour piloter la syntaxe SQL. Par exemple, pour définir un seuil sur le tablespace utilisateur TBS_USERS, la syntaxe SQL utilisée est :

```
begin
DBMS_SERVER_ALERT.SET_THRESHOLD(9000,4,'90','98',1,1,NULL ,5,'TBS_USERS') ;
end ;
/
```

Estimation de la taille d'un nouveau segment

Avec Oracle Enterprise Manager, lors de la création d'une table ou d'un index, vous disposez d'un utilitaire estimant l'espace nécessaire à partir du nombre de lignes que contiendra la table.

Liens entre fichiers et tablespaces

Pour connaître le lien entre les tablespaces et les fichiers, utilisez la vue SYS.DBA _DATA_FILES du dictionnaire de données :

```
desc sys.dba_data_files
 Nom                           Type
 ----------------------------- -------------
 FILE_NAME                     VARCHAR2(513)
 FILE_ID                       NUMBER
 TABLESPACE_NAME               VARCHAR2(30)
 BYTES                         NUMBER
 BLOCKS                        NUMBER
 STATUS                        VARCHAR2(9)
 RELATIVE_FNO                  NUMBER
 AUTOEXTENSIBLE                VARCHAR2(3)
 MAXBYTES                      NUMBER
 MAXBLOCKS                     NUMBER
 INCREMENT_BY                  NUMBER
 USER_BYTES                    NUMBER
 USER_BLOCKS                   NUMBER

col file_name format a40
select file_name, tablespace_name, bytes, auto-extensible
   from dba_data_files ;

FILE_NAME                          TABLESPACE_NAME  BYTES     AUT
---------------------------------- ---------------- --------- -----
c:\oracle\oradata\TEST\USERS01.DBF   USERS_DATA      118489088  YES
c:\oracle\oradata\TEST\USERS02.DBF   USERS_DATA        8489088  YES
c:\oracle\oradata\TEST\OEMREP01.DBF  SYSAUX            5242880  YES
c:\oracle\oradata\TEST\INDX01.DBF    INDX              2097152  YES
c:\oracle\oradata\TEST\RBS01.DBF     RBS              26214400  YES
c:\oracle\oradata\TEST\TEMP01.DBF    TEMP              2097152  YES
c:\oracle\oradata\TEST\SYSTEM01.DBF  SYSTEM          146800640  YES
```

Dans l'exemple précédent, seul le tablespace USERS_DATA possède deux fichiers. La taille totale de ce tablespace correspond à la somme de ces deux fichiers. La vue SYS.DBA_TABLESPACES permet de connaître la taille totale de tous les tablespaces.

Caractéristiques des tablespaces et fichiers associés

Pour connaître les caractéristiques de chaque tablespace, les vues utiles du dictionnaire de données sont :

- SYS.DBA_DATA_FILES : détail des fichiers de données :
- SYS.DBA_TEMP_FILES : détail des fichiers des tablespaces temporaires :
- SYS.DBA_TABLESPACES : détail des tablespaces :
- SYS.DBA_FREE_SPACE : espace disponible restant dans les tablespaces.

Par exemple, les ordres SQL suivants permettent de connaître la taille totale des tablespaces, puis l'espace disponible restant dans les tablespaces.

```
ttitle center "Taille totale des tablespaces" skip 2
col tablespace_name for a16 heading "Tablespace"

select tablespace_name, sum(bytes)/1024 "Place totale en Ko"
  from sys.dba_data_files
  group by tablespace_name
  order by tablespace_name
/

               Taille totale des tablespaces

Tablespace    Place totale en Ko
------------- ------------------
INDX                        2048
OEM_REPOSITORY              5120
RBS                        25600
SYSTEM                    143360
TEMP                        2048
USERS                     115712


ttitle center "Espace disponible dans les tablespaces" skip 2
col TBS for a15
select tablespace_name, sum(bytes)/1024 "Espace disponible en Ko"
 from sys.dba_free_space
 group by tablespace_name
/

                Place libre dans les tablespaces

Tablespace    Espace disponible en Ko
------------- -----------------
INDX                         2046
OEM_REPOSITORY               3268
RBS                         13698
SYSTEM                       5670
TEMP                         1976
USERS                      114200
```

L'ordre SQL ci-dessous, plus complexe, reprend les éléments issus des deux ordres SQL précédents. Il permet d'afficher, tablespace par tablespace, la taille totale, l'espace disponible et le pourcentage d'occupation au sein du tablespace. C'est ce dernier ordre qui figure à l'annexe 3, *Procédures pour le DBA*. La syntaxe utilisée pour écrire cet ordre montre que le SQL peut être employé pour des requêtes très évoluées.

> Oracle Enterprise Manager vous permet d'accéder à ces informations sans la lourdeur d'une requête SQL dont il est impossible de se souvenir !

```
ttitle center "Taille et occupation des tablespaces" skip 2
col "% occupé" for 99.9

select A.tablespace_name,
       A.total_size "Taille totale (Ko)",
       B.free_size  " Espace disponible(Ko)",
      ((A.total_size - B.free_size)*100 / A.total_size) "% occupé"
  from
    (select tablespace_name,
            sum(bytes)/1024 total_size
            from sys.dba_data_files
            group by tablespace_name) A ,
    (select tablespace_name,
           sum(bytes)/1024 free_size
           from sys.dba_free_space
           group by tablespace_name) B
 where a.tablespace_name = b.tablespace_name
/

                Taille et occupation des tablespaces

Tablespace  Taille totale (Ko) Espace disponible(Ko) % occupé
----------- ------------------ ----------------      --------
INDX                     2048             2046            .1
OEM_REPOSITO             5120             3268          36.2
RBS                     25600            13698          46.5
SYSTEM                 143360             5670          96.0
TEMP                     2048             1976           3.5
USERS                  115712           114200           1.3
```

Le dernier ordre SQL est très intéressant ; il permet de visualiser en un seul coup d'œil l'occupation de vos tablespaces.

Positionner un tablespace online/offline

Un tablespace peut être online (actif, accessible à tous les utilisateurs) ou offline (inactif, inaccessible aux utilisateurs, même administrateur).

Mettre un tablespace offline permet de sauvegarder les fichiers qui le composent.

```
alter tablespaces users offline ;

-- sauvegarder les fichiers
-- qui composent le tablespace.

alter tablespaces users online ;
```

Sauf dans le cas d'une base en mode ARCHIVELOG, ne sauvegardez jamais les fichiers lorsque la base est ouverte et les tablespaces online ; votre sauvegarde n'aurait aucune valeur. En effet, entre le début et la fin de votre sauvegarde, le processus Database Writer d'Oracle continue à écrire dans le fichier.

Déplacer/renommer les fichiers d'un tablespace

La marche à suivre est la suivante :

1. Positionnez le tablespace concerné offline ;

```
alter tablespace users offline;

Espace de tables (TABLESPACE) modifié.
```

2. Copiez/renommez sous Windows les anciens fichiers dans leur nouvelle destination.
3. Indiquez au dictionnaire de la base ce changement d'emplacement :

```
alter tablespace users rename datafile
    'c:\oracle\oradata\TEST\user_old.dbf'
 to 'c:\oracle\oradata\TEST\user_new.dbf' ;
```

4. Mettez le tablespace concerné online :

```
alter tablespace users online;

Espace de tables (TABLESPACE) modifié.
```

5. Vérifiez par la vue DBA_DATA_FILES la prise en compte des modifications, puis supprimer sous Windows l'ancien fichier.

Fragmentation des tablespaces

> Les nouvelles options d'Oracle 10*g* limitent l'intérêt de ce type de mesure. La fragmentation existe, Oracle la gère maintenant très bien !

Les tablespaces contiennent des segments constitués de plusieurs extensions. À ce titre, l'espace disponible au sein d'un tablespace est morcelé. L'ordre SQL suivant permet de mesurer cette fragmentation.

```
ttitle center "Espace disponible en nombre de blocs contigus" skip -
      center " (1 bloc vaut db_block_size octets)"      skip 2

select tablespace_name,
       round(avg(blocks)) "En Moyenne",
       min(blocks) "Minimum",
       max(blocks) "Maximum",
       count(*) "Nombre"
 from sys.dba_free_space
 group by tablespace_name
/

        Espace disponible en nombre de blocs contigus
        (1 bloc vaut db_block_size octets)
```

```
TABLESPACE_NAME             En Moyenne   Minimum   Maximum    Nombre
--------------------------- ---------- --------- --------- ---------
INDX                              1023      1023      1023         1
OEM_REPOSITORY                    1634      1634      1634         1
RBS                                381       125      1474        18
SYSTEM                             118         5      1485        24
TEMP                              1023      1023      1023         1
USERS                            18757       400     55440         3
```

Dans cet exemple, l'espace disponible au sein du tablespace est exprimé en nombre de blocs.

Cet ordre SQL présente la plus importante extension contiguë que votre tablespace acceptera de créer, avant d'augmenter la taille d'un des fichiers qui le compose.

Modifier la taille d'un fichier

Nous avons vu que la taille d'un tablespace pouvait augmenter automatiquement par la clause AUTOEXTEND ON. Vous pouvez aussi augmenter manuellement la taille d'un fichier :

```
alter database datafile 'c:\oracle\oradata\TEST\user_01.dbf'
 resize 100M
```

Suivant la dimension actuelle du fichier, cette opération en augmente ou diminue sa taille. Dans le cas d'une diminution, cette opération est parfois impossible. Dans ce cas, tentez de compacter les tables et index contenus (voir précédemment).

Supprimer un tablespace

C'est une opération simple mais dangereuse ! Avant toute suppression d'un tablespace, assurez-vous de disposer d'une sauvegarde complète de la base. Pour supprimer un tablespace ne contenant aucun segment de données ou d'index, l'ordre DROP TABLESPACE suffit :

```
drop tablespace test ;
```

Si le tablespace contient des tables et des index, vous devez le préciser sinon l'ordre SQL précédent échoue.

```
drop tablespace test including contents ;
```

Si des contraintes d'intégrité lient les données de ce tablespace avec d'autres tablespaces, l'ordre CASCADE CONTRAINTS doit être indiqué.

Enfin, vous pouvez lier la suppression des fichiers composant le tablespace à celui du tablespace :

```
drop tablespace test including contents and datafiles ;
```

Les tablespaces transportables

Cette caractéristique permet de transférer un tablespace d'une base de données Oracle 10*g* vers une autre base de données Oracle 10*g*. Les utilisations sont multiples : déplacer l'ensemble des données d'une application d'une base à une autre, créer rapidement un environnement de test identique à celui de production, transférer des tablespaces d'une base de production vers une base infocentre. Comme les tablespaces transportables peuvent provenir de bases configurées avec des DB_BLOCK_SIZE différents, Oracle accepte maintenant des tablespaces possédant une valeur différente de celle définie pour son tablespace SYSTEM. L'utilisation des tablespaces transportables est limitée entre bases Oracle 10*g*.

Comparées aux moyens antérieurs, tel l'Export/Import ou le chargement de données via SQL*Loader, les performances sont exceptionnelles. Lorsque l'on transporte un tablespace, les données et les index sont également déplacés, ce qui évite tout chargement de données et toute construction d'index. Maintenant, il suffit de copier des fichiers et d'intégrer les méta-données du tablespace dans la base cible.

Les tablespaces READ-ONLY

Vous pouvez utiliser des fichiers de données écrits sur des médias non réinscriptibles, comme des disques optiques ou des CD-Rom. Pour cela, vous pouvez utiliser des tablespaces en mode READ-ONLY.

Il est en revanche impossible d'avoir tous les fichiers d'une base de données dans ce format. Le tablespace SYSTEM, les fichiers de contrôle, les fichiers redo-log, les tablespaces de tri doivent être placés sur des disques « normaux ».

Gestion des tablespaces d'annulation

Un tablespace de type UNDO est un nouveau type de tablespace destiné à remplacer l'ancien mode, les rollback segments. C'est dans cet espace que sont gérées par Oracle les données en attente de validation ou d'annulation. L'utilisation d'un tablespace UNDO est beaucoup plus simple que celle des rollback segments. Ils sont tous deux abordés dans ce paragraphe.

> Pour la gestion interne du dictionnaire, Oracle utilise toujours un rollback segment SYSTEM, quel que soit le mode de gestion des annulations choisi. Aucune opération n'est nécessaire sur cet objet.

Les nouveautés des tablespaces UNDO

Les tablespaces de type UNDO apportent des fonctionnalités nouvelles très intéressantes :

- la gestion est simplifiée par rapport à celle qu'exigeaient les rollback segments ;
- des erreurs complexes à résoudre concernant la durée de rétention des informations ne sont plus rencontrées ;

- le *Flashback Query* permet d'interroger le contenu de la base de données tel qu'il était plusieurs heures auparavant.

La création d'un tablespace UNDO

Le choix entre un tablespace de type UNDO ou les rollback segments doit être fait lors de la création de la base car il est difficile de passer d'un mode à l'autre. Les deux modes ne peuvent pas fonctionner simultanément.

Un tablespace de type UNDO peut être créé en même temps que la base ou après.

```
CREATE DATABASE TEST
MAXINSTANCES 1
MAXLOGHISTORY 1
MAXLOGFILES 5
MAXLOGMEMBERS 5
MAXDATAFILES 100
DATAFILE 'c:\oracle\oradata\TEST\system01.dbf' SIZE 100M REUSE
         AUTOEXTEND ON
         NEXT  10240K
         MAXSIZE 1000M
UNDO TABLESPACE "UNDOTBS"
DATAFILE 'c:\oracle\oradata\TEST\undotbs01.dbf' SIZE 50M,
         'D:\oracle\oradata\TEST\undotbs02.dbf' SIZE 50M REUSE
         AUTOEXTEND ON
         NEXT  5120K
         MAXSIZE 1000M
CHARACTER SET WE8ISO8859P15
NATIONAL CHARACTER SET AL16UTF16
LOGFILE  GROUP 1 ('c:\oracle\oradata\TEST\redo01.log') SIZE 100M,
         GROUP 2 ('c:\oracle\oradata\TEST\redo02.log') SIZE 100M,
         GROUP 3 ('c:\oracle\oradata\TEST\redo03.log') SIZE 100M
;
```

Avant de lancer cette création, le fichier d'initialisation de la base doit préciser le mode de gestion choisi. Le paramètre UNDO_MANAGEMENT accepte les valeurs :

- UNDO_MANAGEMENT=AUTO : la gestion par tablespace UNDO est retenue. Le paramètre UNDO_TABLESPACE doit alors indiquer quel tablespace de type UNDO il doit utiliser ;
- UNDO_MANAGEMENT=MANUAL : la gestion par rollback segment est retenue.

Pour l'exemple précédent, les paramètres figurant dans le fichier d'initialisation persistant ou non seraient :

```
UNDO_MANAGEMENT=AUTO
UNDO_TABLESPACE=UNDOTBS
```

Les paramètres du fichier d'initialisation peuvent être interrogés par la vue V$PARAMETER.

La période de rétention d'un tablespace UNDO

Le *Flashback Query* permet d'interroger les données dans l'état où elles étaient plusieurs heures auparavant. Pour cela, on indique au tablespace UNDO une durée de rétention des informations. Il est ainsi possible d'interroger les données de la base telles qu'elles étaient 12 heures auparavant. L'utilisation de cette possibilité sur une aussi longue période nécessite de disposer d'espace disque dans le tablespace UNDO car tout l'historique de modification des données est conservé. Les performances de la base sont peu affectées par cette possibilité.

Le package PL/SQL DBMS_FLASHBACK permet d'effectuer des interrogations dans le passé. La durée de rétention est précisée par le paramètre d'initialisation UNDO _RETENTION. Sa valeur par défaut est de 900 secondes mais elle peut être modifiée dynamiquement ou dans le fichier d'initialisation.

Ces données conservées nécessitant de l'espace disque, l'utilitaire Undo Tablespace Advisor d'Oracle Enterprise Manager permet d'en déterminer la taille nécessaire en se basant sur l'historique d'utilisation fourni dans les vues V$UNDOSTAT.

Actions possibles sur un tablespace UNDO

Comme pour les autres tablespaces, vous pouvez ajouter des fichiers à un tablespace UNDO existant.

```
alter tablespace undotbs
  add datafile 'E:\oracle\oradata\TEST\undotbs03.dbf' SIZE 50M
  autoextend on
  next  10M
  maxsize 100M;
```

Il est possible de changer le tablespace UNDO d'une base. Pour cela :

1. Créez un nouveau tablespace UNDO ;
2. Arrêtez la base et modifiez les paramètres d'initialisation pour choisir le nouveau tablespace UNDO ;
3. Démarrez la base ;
4. Supprimez l'ancien tablespace par la commande DROP UNDO TABLESPACE.

La gestion des rollback segments

La suite de ce paragraphe décrit la gestion des rollback segments. C'est l'ancienne méthode proposée par Oracle pour gérer les segments en attente de validation ou d'annulation. Leur théorie a été abordée au chapitre 11, *Transactions et accès concurrents*.

> Oracle recommande fortement d'utiliser la nouvelle méthode de gestion des segments d'annulation dans un tablespace de type UNDO. Ce point est détaillé précédemment.

Création d'un rollback segment

L'ordre SQL CREATE ROLLBACK SEGMENT assure leur création.

```
create public rollback segment rb1
  storage(initial 200K
          next 200K
          optimal 400K)
  tablespace rbs ;
```

Dans cet exemple, la clause STORAGE contient un nouveau paramètre : OPTIMAL. Ce paramètre est exclusivement réservé aux rollback segments. Il détermine la taille optimale que doit posséder le rollback segment. Si un utilisateur lance une transaction lourde, le rollback segment de celle-ci occupera la place nécessaire pour contenir toutes les anciennes données. Cette place peut être importante ; le mécanisme des extensions sera alors mis en œuvre. Dès que la transaction est terminée (COMMIT ou ROLLBACK), la place occupée par les différents segments devient inutile. Le paramètre OPTIMAL précise la taille à laquelle doit revenir le rollback segment.

Dès qu'il est créé, le rollback segment est OFFLINE, c'est-à-dire qu'aucune transaction ne peut l'utiliser. L'ordre SQL suivant le positionne ONLINE.

```
alter rollback segment RB1 online ;
```

Visualiser les rollback segments

L'ordre SQL suivant permet de visualiser la liste des rollback segments ainsi que leur statut.

```
ttitle center "Liste des rollback segments" skip 2
col nomsegment for a10
col iniext for 999999999
col nextex for 99999999
col pctin  for 9999
col status for a8
col tablespace_name for a14 heading "Tablespace"

select segment_name "NomSegment", owner, tablespace_name,
  initial_extent "IniExt",
  next_extent "NextEx",pct_increase "PctIn",status
  from sys.dba_rollback_segs
/

                Liste des rollback segments

NomSegment OWNER  Tablespace         IniExt    NextEx PctIn STATUS
---------- ------ -------------- ---------- --------- ----- --------
SYSTEM     SYS    SYSTEM              51200     51200     0 ONLINE
RB_TEMP    SYS    SYSTEM             102400    102400     0 OFFLINE
RB1        PUBLIC RBS                102400    256000     0 ONLINE
RB2        PUBLIC RBS                102400    256000     0 ONLINE
RB3        PUBLIC RBS                102400    256000     0 ONLINE
```

Dans cet exemple, les rollback segments SYSTEM et RBS_TEMP sont placés dans le tablespace SYSTEM. Seul le rollback segment SYSTEM est actif. Il est indispensable au fonctionnement de la base.

Suppression d'un rollback segment

Pour pouvoir supprimer un rollback segment, il ne doit pas être utilisé par une transaction. La méthode la plus simple est de le mettre offline puis de le supprimer :

```
alter rollback segment RB1 offline ;

drop rollback segment RB1 ;
```

Nombre de rollback segments à créer

Chaque transaction utilise un rollback segment, mais celui-ci peut l'être simultanément par plusieurs transactions. Pour disposer d'un nombre de rollback segments cohérent par rapport aux nombre d'utilisateurs connectés à une base, Oracle conseille la valeur *nombre d'utilisateurs connecté/3*. Cette valeur s'avère très satisfaisante dans la pratique.

Gestion des fichiers redo-log

Les fichiers redo-log assurent deux objectifs : la performance et la sécurité. La performance, car ils permettent à la base de données de traiter plus rapidement un gros volume d'écritures disque. La sécurité, car en cas d'arrêt brutal de votre machine, comme une coupure de courant, les fichiers redo-log permettent de valider les transactions valides et d'annuler celles qui étaient en cours. Ils peuvent aussi être archivés sur le serveur hébergeant la base et sur un serveur distant, grâce au mode ARCHIVELOG. Leur principe a été abordé au chapitre 9, *Les fichiers d'une base Oracle 10*g.

> Avant toute action, effectuez une sauvegarde de la base de données et entraînez-vous sur une base test, sans utilisateurs actifs.

Création d'un fichier redo-log

Les fichiers redo-log initiaux sont créés en même temps que la base de données.

```
CREATE DATABASE TEST
MAXINSTANCES 1
MAXLOGHISTORY 1
MAXLOGFILES 5
MAXLOGMEMBERS 5
MAXDATAFILES 100
DATAFILE 'c:\oracle\oradata\TEST\system01.dbf' SIZE 100M REUSE
         AUTOEXTEND ON
         NEXT  10240K
         MAXSIZE 1000M
```

```
UNDO TABLESPACE "UNDOTBS"
DATAFILE 'c:\oracle\oradata\TEST\undotbs01.dbf' SIZE 50M,
         'D:\oracle\oradata\TEST\undotbs02.dbf' SIZE 50M REUSE
         AUTOEXTEND ON
         NEXT  5120K
         MAXSIZE 1000M
CHARACTER SET WE8ISO8859P15
NATIONAL CHARACTER SET AL16UTF16
LOGFILE  GROUP 1 ('c:\oracle\oradata\TEST\redo01.log') SIZE 100M,
         GROUP 2 ('c:\oracle\oradata\TEST\redo02.log') SIZE 100M,
         GROUP 3 ('c:\oracle\oradata\TEST\redo03.log') SIZE 100M
;
```

Caractéristiques des fichiers redo-log

Les vues V$LOGFILE et V$LOG détaillent les caractéristiques des fichiers redo-log. La vue V$LOGFILE fournit la liste des fichiers redo-log de la base.

```
select * from v$logfile ;

Group# STATUS  MEMBER
------ ------ ---------------------------------
     1        c:\oracle\oradata\TEST\redo01.log
     2        c:\oracle\oradata\TEST\redo02.log
     3        c:\oracle\oradata\TEST\redo03.log
```

La vue V$LOG fournit le détail de ces fichiers et précise le fichier redo-log actif, c'est-à-dire celui dans lequel le processus Log Writer (DBWR) écrit.

```
select group#, sequence#, bytes, archived, status
  from v$log ;

   GROUP# SEQUENCE#     BYTES ARC STATUS
--------- --------- --------- --- ------------
        1       269   1048576 NO  INACTIVE
        2       270   1048576 NO  INACTIVE
        3       271   1048576 NO  CURRENT
```

Actuellement, le groupe numéro 3 (c'est-à-dire le fichier c :\oracle\oradata\TEST\redo03.log) est actif. Aucun fichier n'a été archivé.

Basculer la base en mode ARCHIVELOG

Nous avons abordé le principe et les contraintes du mode ARCHIVELOG au chapitre 9, *Les fichiers d'une base Oracle 10*g.

Pour mettre la base en mode ARCHIVELOG :

1. Éditez le fichier d'initialisation *initSID.ora* pour y préciser les paramètres suivants :

```
...
log_archive_start = true
```

```
log_archive_dest  = 'C:\oracle\admin\TEST\arch'
log_archive_format=ARC%S_%T.arc
...
```

Le paramètre LOG_ARCHIVE_DEST peut pointer vers un système de fichiers local ou vers un répertoire d'une machine distante. Sa syntaxe sera alors :

```
log_archive_dest='LOCATION=\repertoire\du\serveur\local'
log_archive_dest='SERVICE=label_du_service_a_utiliser'
```

2. Arrêtez la base. Le processus Log Writer (LGWR) sera lancé aux prochains démarrages.
3. Démarrez la base en mode exclusive avec SQL*Plus et positionnez-la en mode archivelog :

```
sqlplus /nolog
connect sys/change_on_install as SYSDBA
startup mount exclusive;

alter database archivelog ;

shutdown immediate ;
startup ;
```

4. Vérifiez :

```
select name, log_mode from v$database ;

NAME      LOG_MODE
--------- ------------
TEST      ARCHIVELOG
```

5. Identifiez le fichier redo-log actif :

```
select group#, sequence#, bytes, archived, status
 from v$log ;

   GROUP# SEQUENCE#     BYTES ARC STATUS
--------- --------- --------- --- ------------
        1       269   1048576 NO  INACTIVE
        2       270   1048576 NO  INACTIVE
        3       271   1048576 NO  CURRENT
```

6. Forcez la création d'un fichier log archivé :

```
alter system switch logfile;

-- vérification
select group#, sequence#, bytes, archived, status
from v$log ;

   GROUP# SEQUENCE#     BYTES ARC STATUS
--------- --------- --------- --- ------------
        1       269   1048576 NO  CURRENT
        2       270   1048576 NO  INACTIVE
        3       271   1048576 NO  INACTIVE
```

7. Vérifiez que le fichier d'archive a été créé au moment du « switch logfile » en regardant, sous Windows, le répertoire pointé par le paramètre LOG_ARCHIVE_DEST.

```
C:\> cd c:\oracle\admin\TEST\arch
C:\> dir A*
C:\>  ARC001_272.arc
```

8. Fermez votre base et effectuez une sauvegarde complète. Elle servira de base sur laquelle tous les fichiers redo-log archivés seront appliqués en cas de restauration. N'oubliez pas de surveiller le remplissage de votre système de fichier dans lequel vous placez les fichiers redo-log archivés.

Basculer la base en mode NOARCHIVELOG

La démarche est similaire, il suffit de modifier la commande précédente.

```
alter database noarchivelog ;
```

Il est conseillé de ne pas mettre la base en mode ARCHIVELOG lorsque vous effectuez des opérations de maintenance comme des Export/Import ou des programmes batch lourds, car dans ce cas le volume des fichiers d'archives créés est directement lié au volume des données manipulées.

> Si vous passez du mode ARCHIVELOG au mode NOARCHIVELOG, soyez toujours certain du mode dans lequel vous êtes. Rien n'est pire que de penser avoir un système de haute disponibilité opérationnel alors qu'il n'est pas actif…

Sauvegarde base ouverte, utilisateurs actifs

Cette option est très intéressante en cas de haute disponibilité. La base de données est alors nécessairement en mode ARCHIVELOG.

La sauvegarde base ouverte est traitée au chapitre 26, *La sauvegarde d'une base Oracle 10*g.

Informations sur le mode ARCHIVELOG

Les vues du dictionnaire de données qui contiennent des informations sur le mode ARCHIVELOG sont :

- V$DATABASE : identifie si la base de données est en mode ARCHIVELOG ou NOARCHIVELOG ;
- V$ARCHIVED_LOG : affiche des informations issues du fichier de contrôle concernant les fichiers redo-log archivés ;
- V$ARCHIVE_DEST : décrit la destination des fichiers archivés ;
- V$LOG : décrit les fichiers redo-log et indique celui qui est actif ;

- V$LOG_HISTORY : contient l'historique de tous les passages d'un fichier redo-log à un autre.

La vue V$LOG_HISTORY, couplée à l'étude du fichier d'alerte de votre base, permet de vous assurer que les permutations de fichiers redo-log ne sont pas trop fréquentes. Dans le cas contraire, il faut augmenter la taille de ces fichiers. C'est une opération de tuning qui donne souvent d'excellents résultats.

Ajouter un fichier redo-log

Comme pour la création des fichiers redo-log initiaux, cette opération s'effectue au niveau de la base de données.

```
alter database add logfile ('c:\oracle\oradata\TEST\redo04.log' ,
                            'c:\oracle\oradata\TEST\redo05.log' ,
                            'c:\oracle\oradata\TEST\redo06.log' )
size 30M ;
```

Cette opération ajoute trois fichiers redo-log, de 30 Mo chacun, à la base de données.

Supprimer un fichier redo-log

Le plus simple est de supprimer un fichier redo-log inactif. Identifiez d'abord le fichier redo-log actif.

```
 select group#, sequence#, bytes, archived, status
  from v$log ;

   GROUP# SEQUENCE#     BYTES ARC STATUS
--------- --------- --------- --- ------------
        1       269   1048576 NO  CURRENT
        2       270   1048576 NO  INACTIVE
        3       271   1048576 NO  INACTIVE
        4       271   3145728 NO  INACTIVE
        5       271   3145728 NO  INACTIVE
        6       271   3145728 NO  INACTIVE
```

Placez le fichier redo-log actif sur un fichier que vous ne voulez pas supprimer.

```
alter system switch logfile ;
```

Supprimez les fichiers redo-log inactifs.

```
alter database drop logfile
member ('c:\oracle\oradata\TEST\redo01.log',
        'c:\oracle\oradata\TEST\redo02.log',
        'c:\oracle\oradata\TEST\redo03.log' );
```

Vérification :

```
select group#, sequence#, bytes, archived, status

  from v$log ;
```

```
   GROUP# SEQUENCE#     BYTES ARC STATUS
--------- --------- --------- --- ------------
        4       271   3145728 NO  ACTIVE
        5       271   3145728 NO  INACTIVE
        6       271   3145728 NO  INACTIVE
```

N'oubliez pas que le fichier « disparaît » du dictionnaire Oracle mais qu'il existe encore sur le système de fichiers Windows. Il faut supprimer les anciens fichiers manuellement.

Déplacer un fichier redo-log

La procédure à suivre est identique à la création/suppression des fichiers redo-log.

Performances et redo-log

L'impact des fichiers redo-log sur les performances est importante. S'ils sont sous-dimensionnés, ils vont se remplir rapidement puis basculer de l'un vers l'autre, ce qui entraîne une attente le temps que toutes les écritures présentes dans le premier fichier soient écrites dans les fichiers de la base.

Ce goulot d'étranglement étant fréquemment rencontré et facile à éviter, Oracle impose maintenant des fichiers redo-log d'une taille minimale de 4 Mo.

Pour tout savoir de l'historique des mouvements entre fichiers redo-log sur une période d'activité, consultez la vue V$LOG_HISTORY.

Oracle conseille une durée minimale de 20 min pour remplir un fichier redo-log. Cela évite des bascules de fichier à fichier trop fréquentes.

Cas d'usage de Log Miner

Oracle propose un outil d'analyse du contenu des fichiers redo-log d'une base : le Log Miner. Cet utilitaire analyse de manière séquentielle l'ensemble des opérations réalisées sur une base. Vous pouvez ainsi répondre à des questions telles que :

- Comment visualiser sous forme d'ordres SQL le contenu des fichiers redo-log, c'est-à-dire des actions qui ont modifié les données de la base ?
- Quelles sont les modifications de structure qui ont été apportées à une base ?
- Quels sont les derniers ordres SQL enregistrés dans le fichier redo-log juste avant de rencontrer un problème disque ?
- Quelles sont les modifications des données d'une table ?
- Quelles sont les modifications apportées par un utilisateur, sur plusieurs tables ou sur une table particulière ?

On se rend compte que son usage appartient à deux grandes familles : aller le plus loin possible dans la récupération des actions effectuées sur une base suite à un problème technique d'une part et la surveillance et le contrôle d'accès aux données d'autre part.

L'avantage procuré par Log Miner est que rien ne doit être prévu a priori, tout est analysé a posteriori.

Principe d'utilisation de Log Miner

Pour transformer le contenu des fichiers redo-log en une suite d'ordres SQL compréhensibles, le Log Miner s'appuie sur le dictionnaire de données présent dans la base ou stocké sous forme d'un fichier plat ou de redo-log à l'extérieur de la base. Ce sont ensuite les procédures PL/SQL des packages DBMS_LOGMNR et DBMS_LOGMNR_D associées aux vues V$LOGMNR_CONTENTS et V$LOGxx du dictionnaire qui permettent l'analyse complète.

On peut ainsi analyser tous les ordres SQL effectués sur une base en ayant à sa disposition les fichiers redo-log à analyser, le dictionnaire de données dans un fichier ASCII et une base de données. Log Miner peut être utilisé sur une machine différente de celle sur laquelle fonctionne la base.

C'est un moyen de surveillance très puissant et très discret qui ne manquera pas d'être mis en œuvre dans des environnements où la traçabilité et l'accès à l'information sont sensibles. Bienvenue dans le monde de Big Brother !

Enfin, une interface graphique intégrée à Oracle Enterprise Manager permet d'accéder à quelques informations issues de Log Miner. Les cas de figures à étudier étant très variés, un outil graphique ne peut pas en couvrir tous les aspects.

Gestion des fichiers de contrôle

Ce paragraphe précise comment gérer les fichiers de contrôle. Ce sont les fichiers Oracle les plus simples et les plus rapides à administrer. En revanche, leur perte peut s'avérer désastreuse. Nous abordons des opérations de base comme ajouter, renommer, déplacer et supprimer des fichiers de contrôle.

Les fichiers de contrôle

Les fichiers de contrôle contiennent principalement :

- Le nom de la base ;
- Le nom et l'emplacement des fichiers liés à la base et aux redo-log ;
- La séquence identifiant le redo-log actif ;
- Des informations internes concernant l'état de la base, son statut lors du dernier arrêt, etc.

Ces informations sont essentielles pour démarrer et restaurer une base.

Dans le fichier d'initialisation d'une instance, qu'il soit sous la forme d'un fichier initSID.ora ou d'un fichier persistant SPFILE, le paramètre CONTROL_FILES donne le nom et l'emplacement de tous les fichiers de contrôle de la base de données. Durant le démarrage de l'instance Oracle, la lecture de ce paramètre fournit à la base la liste de tous les fichiers de contrôle. Seul le premier d'entre eux sera lu. En revanche, dès qu'un changement intervient, les mises à jour ont lieu simultanément dans tous les fichiers de contrôle.

Dans le fichier *initSID.ora*, ce paramètre ressemblera à la syntaxe suivante :

```
...
control_files = ('c:\oracle\oradata\TEST\control01.ctl',
                 'D:\oracle\oradata\TEST\control02.ctl')
...
```

Pour connaître dynamiquement cette valeur, vous pouvez interroger la base à partir de SQL*Plus :

```
show parameters control_files
```

Si vous utilisez la gestion automatique des fichiers par Oracle, cette valeur ne figurera pas dans le fichier d'initialisation (voir plus loin dans ce chapitre).

Visualiser les fichiers de contrôle existants

La vue V$CONTROLFILE permet de connaître la liste des fichiers de contrôle d'une instance en cours de fonctionnement.

```
select * from v$controlfile ;

STATUS       NAME
------       ---------------------------
             c:\oracle\oradata\TEST\control01.ctl
             D:\oracle\oradata\TEST\control02.ctl
```

Dans notre cas, la base de données comporte deux fichiers de contrôle, situés sur deux disques différents. C'est le minimum de sécurité en cas de perte de disque.

> Il faut toujours disposer de plusieurs fichiers de contrôle, leur perte pouvant engendrer de nombreuses difficultés. Oracle 10*g* permet de créer un fichier de contrôle même si tous les autres ont disparu. C'est une opération délicate qui nécessite de connaître parfaitement les caractéristiques de la base.

Ajouter un fichier de contrôle

La seule contrainte est de fermer la base de données. Vous devez ensuite effectuer les actions suivantes :

Arrêter, par un SHUTDOWN NORMAL ou IMMEDIATE, la base de données.

Sous Windows, copiez un fichier de contrôle existant sous un autre nom :

```
copy c:\oracle\oradata\TEST\control01.ctl E:\oracle\oradata\TEST\control03.ctl
```

Dans le fichier *initSID.ora*, indiquez ce nouveau fichier :

```
...
control_files = ('c:\oracle\oradata\TEST\control01.ctl',
                 'D:\oracle\oradata\TEST\control02.ctl',
                 'E:\oracle\oradata\TEST\control03.ctl'  )
...
```

Démarrez l'instance et contrôlez le résultat :

```
select * from v$controlfile ;

STATUS       NAME
------       --------------------------
             c:\oracle\oradata\TEST\control01.ctl
             D:\oracle\oradata\TEST\control02.ctl
             E:\oracle\oradata\TEST\control03.ctl
```

Sauvegarder un fichier de contrôle base « en marche »

La solution précédente implique la fermeture de la base pour ajouter un fichier de contrôle. Heureusement, cela n'est pas nécessaire pour en sauvegarder un exemplaire base ouverte.

```
alter database backup controlfile to 'c:\temp\backup.ctl' ;
```

Tous les fichiers de contrôle étant identiques, il n'est pas nécessaire de préciser celui à sauvegarder.

Pour sauvegarder un fichier de contrôle base arrêtée, il suffit de le copier.

Déplacer et supprimer un fichier de contrôle

Pour supprimer un fichier de contrôle, il faut procéder de la même façon : le retirer du fichier *initSID.ora*, fermer puis redémarrer la base et supprimer physiquement le fichier sous Windows.

Déplacer un fichier de contrôle est une opération identique à sa création.

Que faire lorsque tous les fichiers de contrôle sont perdus ?

Jusqu'à la version Oracle8, la perte de tous les fichiers de contrôle entraînait celle de la base et il fallait repartir d'une sauvegarde. Maintenant, la commande CREATE CONTROLFILE permet de créer un fichier de contrôle mais cela nécessite de lui fournir l'ensemble des renseignements qu'il contient.

Cette opération est très périlleuse : si elle échoue, vous risquez d'obtenir un fichier de contrôle inopérant et les modifications apportées à la base la rendront inexploitable. Une sauvegarde sérieuse s'impose avant tout essai.

L'ensemble des informations nécessaires à un CREATE CONTROLFILE peut être généré automatiquement par l'ordre :

```
alter database backup controlfile to trace ;
```

Cela ne peut s'effectuer que si la base est démarrée. Le fichier trace généré contiendra alors l'ensemble des informations nécessaires à la création d'un nouveau fichier de contrôle.

Gestion automatique des fichiers par Oracle

La gestion des fichiers proposée par Oracle offre de multiples options pour en faciliter l'administration : accroissement automatique de leur taille, gestion interne, etc.

Oracle 10*g* va plus loin en vous proposant d'assurer à 100 % la gestion de l'ensemble des fichiers composant une base, y compris leur création, suppression, nom, etc. Les fichiers concernés par cette nouveauté sont :

- les fichiers liés aux tablespaces ;
- les fichiers redo-log ;
- les fichiers de contrôle.

Les autres fichiers de trace, audit, alerte, etc. ne sont pas concernés.

En utilisant cette possibilité, on peut arriver à l'ordre de création d'une base suivant :

```
CREATE DATABASE TEST ;
```

La syntaxe est alors très simple, surtout si on la compare à celle utilisée au chapitre 14, *La création d'une base Oracle 10*g. Mais attention, tous les fichiers qui composent la base existent toujours : leurs nom, dimensionnement, gestion, création, suppression, emplacement, etc., sont entièrement délégués à Oracle 10*g*.

Pourquoi utiliser la gestion automatique des fichiers ?

La gestion automatique des fichiers peut s'avérer très pratique pour ces deux cas :

- une base de test à construire rapidement ;
- pour une Standby Database.

L'intérêt pour une base de test est évident : cela permet de se focaliser sur les tests à effectuer en limitant au maximum la gestion de la base.

Une Standby Database est une base de secours, placée sur un serveur distinct de celui hébergeant la base de production. Régulièrement, les fichiers redo-log sont transmis et appliqués à la Standby Database. Elle est ainsi prête à remplacer la base de production si celle-ci s'avère défaillante.

Lorsqu'une opération impacte un fichier sur la base d'origine, le fichier redo-log répercute cette action à la Standby Database. Cette action nécessite une intervention humaine sauf si la base est en gestion automatique des fichiers. Cela limite fortement la surveillance à apporter à ce type de base.

Quels inconvénients présente la gestion automatique des fichiers ?

Pour l'administrateur Oracle, le plus grand inconvénient provient d'une perte de contrôle sur les fichiers et le paramétrage de la base. Voici quelques éléments de réflexion :

- L'espace disque peut-il être mis à disposition d'un dispositif automatique ?
- Êtes-vous certain de disposer de suffisamment de place disque pour permettre la création automatique de plusieurs fichiers de 100 Mo ?
- Acceptez-vous que la taille de certains fichiers puisse dépasser celle de vos médias de sauvegarde ?
- Pour créer les fichiers, Oracle utilise des options par défaut. Connaissez-vous ces options ? Vous conviennent-elles ?
- La mise en place de cette fonctionnalité vous apporte-t-elle des avantages ?

> Vous l'avez compris, nous déconseillons cette option pour une base de production.

Pourtant, pour une société éditrice de logiciels, l'avantage est certain : la base s'administre toute seule et n'engendre aucun appel de la part de ses clients, même si tout cela peut conduire à un blocage grave. D'un autre côté, pour un client final ne disposant pas de compétences informatiques, il est difficile de rester bloqué pour le simple ajout d'un fichier à un tablespace.

> Dernier point, il est possible de mélanger création manuelle et automatique de fichiers. La simple présence de deux paramètres dans le fichier d'initialisation de la base décide de l'utilisation ou non de la gestion automatique des fichiers.

Les paramètres d'initialisation

Les paramètres d'initialisation concernés sont :

- DB_CREATE_FILE_DEST ;
- DB_CREATE_ONLINE_LOG_DEST_*n*.

DB_CREATE_FILE_DEST précise le répertoire où Oracle 10*g* créera les fichiers de tablespaces temporaires ou de données.

DB_CREATE_ONLINE_LOG_DEST_*n* précise les répertoires où Oracle 10*g* créera les fichiers de contrôle et redo-log. Comme il est recommandé de multiplexer ces fichiers, il est possible d'indiquer plusieurs répertoires.

Par exemple, on peut avoir les valeurs suivantes dans le fichier d'initialisation :

```
db_create_file_dest='c:\oracle\oradata\AUTO'
db_create_online_log_dest_1='c:\oracle\oradata\AUTO'
db_create_online_log_dest_2='d:\oracle\oradata\AUTO'
db_create_online_log_dest_3='e:\oracle\oradata\AUTO'
```

Attention, le simple fait de préciser l'une de ces valeurs dans le fichier d'initialisation implique le transfert de la gestion automatique des fichiers à Oracle !

Les options utilisées par défaut

La gestion automatique obéit néanmoins à des règles. Elles sont très nombreuses car les cas de figure ne manquent pas : création d'une base, passage d'une base existante en mode automatique, cas des fichiers de données, des fichiers temporaires, de contrôle, redo-log, etc.

Pour simplifier, le fichier d'un tablespace temporaire ou de données créé automatiquement aura les caractéristiques suivantes : taille de 100 Mo, fichier *auto-extensible* et taille maximale illimitée.

Pour un fichier redo-log, sa taille sera de 100 Mo. Aucune caractéristique particulière n'est précisée pour un fichier de contrôle car sa gestion est dans tous les cas assurée automatiquement par la base.

Toute la difficulté consiste à déléguer la gestion des fichiers à Oracle tout en connaissant les avantages, limites et risques qu'elle apporte.

Exemple d'une base créée avec ces options

Dans l'exemple suivant, la base AUTO sera créée le plus simplement possible.

1. Création des répertoires OFA destinés à recevoir les fichiers de données, de contrôle de redo-log ainsi que les fichiers d'administration de la base. L'utilisateur Windows doit posséder les droits de lecture/écriture sur ces répertoires :

```
# Répertoires pour les fichiers de la base
c:\oracle\oradata\AUTO
d:\oracle\oradata\AUTO
e:\oracle\oradata\AUTO

# Répertoires pour les fichiers d'administration
c:\oracle\admin\AUTO\udump
c:\oracle\admin\AUTO\cdump
c:\oracle\admin\AUTO\bdump
c:\oracle\admin\AUTO\exp
c:\oracle\admin\AUTO\scripts
```

2. Création d'un fichier d'initialisation `initAUTO.ora`. Les paramètres db_create_file_dest et db_create_online_log_dest sont renseignés :

```
# Fichier initAUTO.ora

db_block_size=8192
db_cache_size=15428800
open_cursors=300

db_create_file_dest='c:\oracle\oradata\AUTO'
db_create_online_log_dest_1='c:\oracle\oradata\AUTO'
db_create_online_log_dest_2='d:\oracle\oradata\AUTO'
db_create_online_log_dest_3='e:\oracle\oradata\AUTO'

#############################################
# Diagnostics et statistiques
#############################################
background_dump_dest=c:\oracle\admin\AUTO\bdump
core_dump_dest=c:\oracle\admin\AUTO\cdump
user_dump_dest=c:\oracle\admin\AUTO\udump
timed_statistics=TRUE

#############################################
# Distribué, réplication et cliché
#############################################
db_domain=gilles.fr
compatible=10.0.0
db_name=AUTO
instance_name=AUTO

#############################################
# Fichier de journalisation et récupération
#############################################
fast_start_mttr_target=300

#############################################
# Pools
#############################################

large_pool_size=1048576
shared_pool_size=10428800

#############################################
# Resource Manager
#############################################
resource_manager_plan=SYSTEM_PLAN

#############################################
# Segments d'annulation (Undo et Rollback) gérés par le système
#############################################
undo_management=AUTO
```

```
undo_tablespace=UNDOTBS

###############################################
# Tri, jointures par hachage, index bitmap
###############################################
sort_area_size=524288
```

3. Création de la base de données. L'ordre CREATE DATABASE a été simplifié au maximum.

```
connect SYS/change_on_install as SYSDBA

startup nomount pfile="c:\oracle\ora10\database\initAUTO.ora"

CREATE DATABASE AUTO;
```

4. Les fichiers créés automatiquement par Oracle sont :

```
Fichiers des tablespaces                    Tablespace    Mo STATUS
------------------------------------------- ---------- ----- ---------
c:\oracle\oradata\AUTO\ora_system_xqc29gob.dbf   SYSTEM    127 AVAILABLE
c:\oracle\oradata\AUTO\ora_sys_undo_xqc2bpnc.dbf SYS_UNDOTS   129 AVAILABLE

Fichiers redo-log
----------------------------------------
c:\oracle\oradata\AUTO\ora_1_xqc28jss.log
d:\oracle\oradata\AUTO\ora_1_xqc28rlp.log
e:\oracle\oradata\AUTO\ora_1_xqc28mqp.log
c:\oracle\oradata\AUTO\ora_2_xqc28zo2.log
d:\oracle\oradata\AUTO\ora_2_xqc2968d.log
e:\oracle\oradata\AUTO\ora_1_xqc28erd.log

Fichiers de controle
---------------------------------
c:\oracle\oradata\AUTO\ora_xqc28hym.ctl
d:\oracle\oradata\AUTO\ora_xqc28jcy.ctl
e:\oracle\oradata\AUTO\ora_xqc28jms.ctl
```

La base créée est minimale. Elle contient le tablespace SYS_UNDOTS qui est de type UNDO, mais pas de tablespace temporaire de type TEMPORARY. De plus, aucun tablespace supplémentaire destiné à héberger les schémas des applications n'existe et aucun catalogue n'est créé. Vous devez lancer les catalogues :

```
connect SYS/change_on_install as SYSDBA
@c:\oracle\ora92\rdbms\admin\catalog.sql;
@c:\oracle\ora92\rdbms\admin\catexp7.sql;
@c:\oracle\ora92\rdbms\admin\catblock.sql;
@c:\oracle\ora92\rdbms\admin\catproc.sql;
@c:\oracle\ora92\rdbms\admin\catoctk.sql;
@c:\oracle\ora92\rdbms\admin\catobtk.sql;
@c:\oracle\ora92\rdbms\admin\caths.sql;
@c:\oracle\ora92\rdbms\admin\owminst.plb;
connect SYSTEM/manager
@c:\oracle\ora92\sqlplus\admin\pupbld.sql;
```

```
connect SYSTEM/manager
@c:\oracle\ora92\sqlplus\admin\help\hlpbld.sql helpus.sql;
exit;

# Lancez maintenant la création des tablespaces applicatifs
CREATE TABLESPACE TBS_1 ;
# les fichiers seront créés par Oracle
```

En fournissant plus de renseignements lors du CREATE DATABASE, il est possible de créer une base minimale plus aboutie. Si vous fournissez trop de paramètres lors de la création de la base, revenez à la méthode utilisant l'assistant DBCA décrite au chapitre 14, *Création d'une base Oracle 10*g.

Conclusion sur la gestion automatique des fichiers

La gestion automatique des fichiers est une avancée proposée par Oracle. Si les avantages sont certains à court terme, des inconvénients majeurs peuvent se présenter à moyen ou long terme.

Oracle possède déjà de nombreuses options facilitant la gestion de l'espace disque : fichiers de type *auto-extensible*, taille maximale d'un fichier, tablespaces TEMPORARY, UNDO, etc. Ces paramètres permettent déjà de limiter très fortement les interventions d'un administrateur Oracle et nous vous conseillons de les utiliser.

Toute la difficulté de cette gestion automatique consiste à l'utiliser intelligemment en connaissant les avantages, limites et risques qu'elle apporte.

Gestion du fichier d'initialisation

Historiquement, toutes les versions d'Oracle ont utilisé un simple fichier texte pour y conserver les paramètres d'initialisation. Avec Oracle 10*g*, ces valeurs peuvent maintenant être gérées automatiquement dans un fichier binaire persistant.

Comme de plus en plus de paramètres d'initialisation sont modifiables dynamiquement, cela permet de conserver les valeurs modifiées et de les réutiliser au prochain démarrage de la base. Comparé à la gestion manuelle, cela évite de devoir modifier le fichier d'initialisation pour prendre en compte des modifications effectuées base lancée par la commande ALTER SYSTEM.

Le fichier persistant ou SPFILE (*System Parameter File*) est créé par la commande CREATE SPFILE.

Démarrage d'une base

Même si un fichier d'initialisation persistant existe, vous pouvez démarrer une base en indiquant un fichier texte d'initialisation.

```
sqlplus /nolog
connect system/manager as sysdba
startup pfile=c:\oracle\oradata\TEST\scripts\init.ora
```

Avec un fichier persistant, vous n'avez aucun fichier à indiquer.

```
sqlplus /nolog
connect system/manager as sysdba
startup
```

L'ordre CREATE DATABASE fait toujours référence à un fichier d'initialisation texte. Ensuite, le fichier SPFILE est créé.

Création d'un fichier persistant

Le fichier d'initialisation persistant est créé par la commande CREATE SPFILE d'après un fichier texte d'initialisation. Il s'applique à une base précise et se place sur le serveur en ORACLE_HOME\database. Cela permet à Oracle de le situer, surtout si la commande est effectuée à distance.

```
# l'utilisateur doit être connecté sous SYSDBA
set ORACLE_SID=TEST
sqlplus /nolog
connect SYS/change_on_install as SYSDBA

create spfile='c:\oracle\ora92\database\spfileTEST.ora'
   FROM pfile='c:\oracle\admin\TEST\pfile\init.ora';
```

Le nom donné au fichier SPFILE est important, il doit respecter la syntaxe *spfileORACLE_SID.or*a, par exemple spfile*TEST*.ora dans l'exemple précédent.

Pour identifier l'emplacement d'un fichier SPFILE existant, utilisez la commande :

```
sqlplus /nolog
connect system/manager

show parameters spfile
```

Créer un fichier d'initialisation

L'opération normale consiste à créer un fichier SPFILE d'après un fichier texte, mais l'inverse est possible par la commande CREATE PFILE.

```
sqlplus /nolog
connect system/manager as SYSDBA

create pfile='c:\oracle\admin\TEST\pfile\init.ora'
   FROM spfile='c:\oracle\ora92\database\spfileTEST.ora';
```

Modifier des valeurs par l'ordre ALTER SYSTEM

Pour que la modification d'un paramètre d'initialisation soit prise en compte, il fallait arrêter la base puis la redémarrer. Cela forçait Oracle à prendre en compte la nouvelle valeur. C'était une contrainte majeure pour les bases nécessitant une haute disponibilité.

Sont alors apparus les paramètres dynamiques, qui peuvent être modifiés par l'ordre ALTER SYSTEM sans qu'il soit nécessaire d'arrêter puis relancer la base. De nombreux paramètres sont devenus dynamiques avec Oracle 10*g*.

La commande ALTER SYSTEM permet de modifier les paramètres en mémoire, dans le fichier SPFILE ou simultanément en mémoire et dans le SPFILE.

La modification uniquement en mémoire (MEMORY) est utile si vous voulez tester un paramétrage sans le rendre permanent dans le SPFILE ou si vous n'utilisez pas de fichier SPFILE.

La modification uniquement dans le fichier persistant (SPFILE) est utile pour les commandes qui ne sont pas modifiables dynamiquement mais qui seront prises en compte au prochain démarrage. Enfin, les deux actions peuvent être simultanées (BOTH).

La syntaxe est la suivante :

```
# le paramètre SCOPE
# peut prendre les valeurs MEMORY, SPFILE, BOTH

ALTER SYSTEM SET nom_du_paramètre = valeur
             COMMENT='commentaire libre'
             SCOPE=MEMORY;
```

Par exemple, pour modifier le paramètre DB_CACHE_SIZE :

```
ALTER SYSTEM SET db_cache_size=15M
             COMMENT='Passage de 20M a 15M pour diminuer le swap'
             SCOPE=BOTH;
```

Pour forcer une valeur à reprendre sa valeur par défaut, affectez-lui une valeur nulle :

```
ALTER SYSTEM SET nom_du_paramètre='' ;
```

Visualiser les valeurs d'initialisation

Les ordres SQL suivants permettent de visualiser les caractéristiques des paramètres d'initialisation :

```
sqlplus /nolog
connect system/manager

# Liste des paramètres
show parameters

# vue accédant l'ensemble des paramètres
select * from v$parameter ;

# vues complémentaires
select * from v$parameter2 ;
select * from v$spparameter ;
```

Outils pour modifier les paramètres d'initialisation

Pour modifier l'ensemble des paramètres d'initialisations, les deux outils possibles sont SQL*Plus pour passer les commandes en mode ligne et Oracle Enterprise Manager qui habille graphiquement ces commandes.

Résumé

Ce chapitre a abordé la façon dont Oracle 10*g* gère l'espace disque alloué au cours des opérations de création, d'insertion et de modification opérées sur les données. La gestion des différents fichiers qui composent une base Oracle 10*g* a été abordée. Partant d'éléments théoriques comme le tablespace ou la gestion de l'espace au sein de la base de données, de nombreux exemples mettent en pratique ces notions.

> Une nouvelle fois, n'oubliez pas de vous entraîner sur une base de test avec Oracle Enterprise Manager avant d'effectuer ces opérations sur une base de production.