SGBD
Enseignant : M. Nakechbnadi mail : nakech@free.fr

Chapitre IV
La Structure physique de la Base de données

IV.1/ Les tablespaces

IV.1.1/ Notion de tablespaces

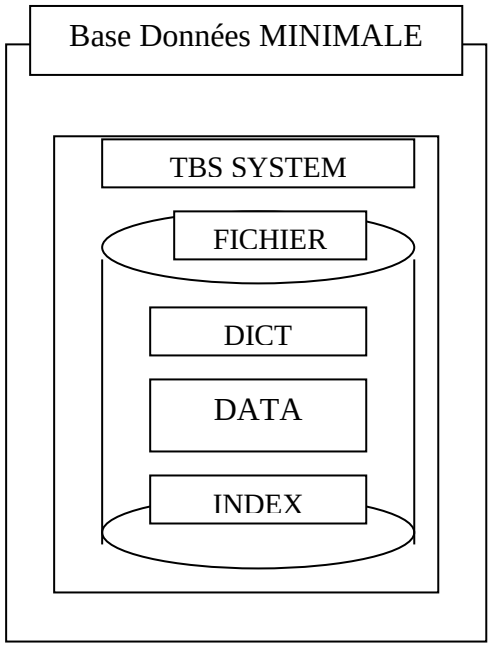
Le données d'une base peut être décomposée en tablespaces : **partitions logiques contenant un ou plusieurs fichiers**. Un fichier appartient à 1 et 1 seul tablespace.

Un tablespace peut s'étendre soit par ajout (on-line) d'un fichier, soit par auto-extension DU fichier du tablespace.

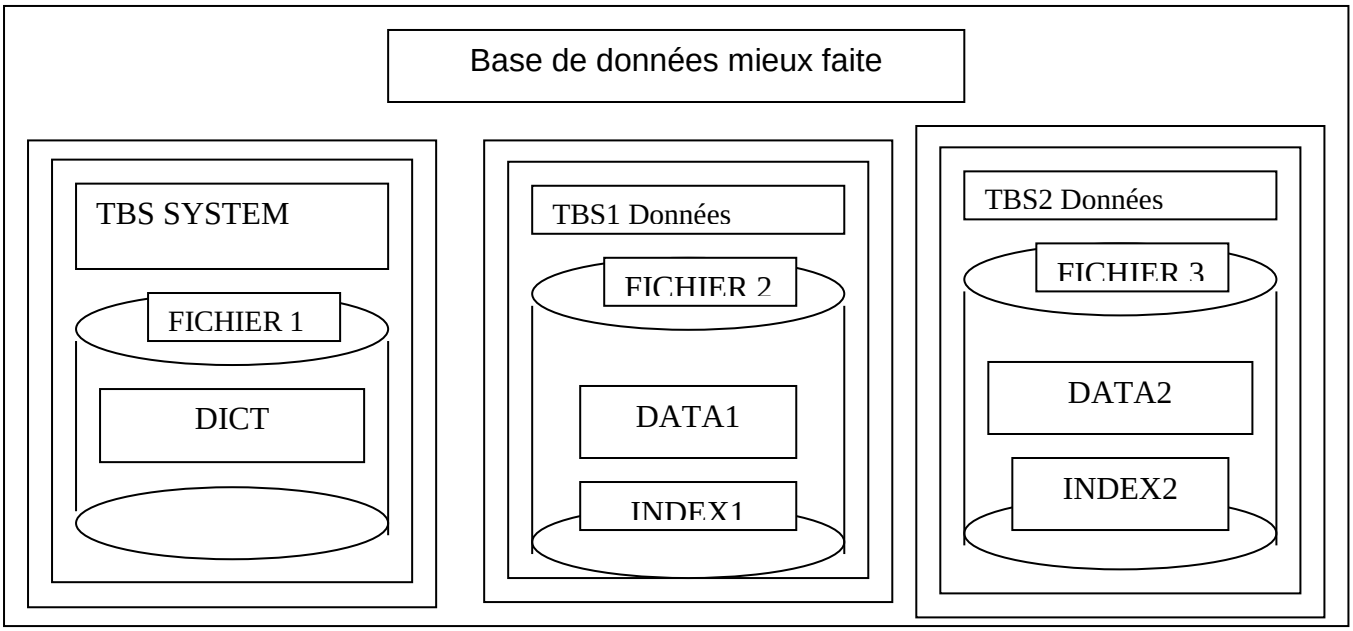
Par défaut il existe toujours un tablespace baptisé SYSTEM qui contient le dictionnaire de données et le rollback segment SYSTEM.

Quand vous allez créer une nouvelle table, celle-ci sera contenue soit dans un **tablespace** existant (SYSTEM par défaut) soit dans un nouveau que vous créerez.

On peut également stocker les datas et les index dans ce même tablespace, et obtenir ainsi une base minimale peu structurée, peu performante et peu sécurisée :



Au contraire on peut répartir les données, les index, mais aussi les images avant (rollback segments) sur un nombre maximum de disques. On y gagnera en performance, en souplesse, et en sécurité :



Quelques conseils

- Séparez les données utilisateurs des données du dictionnaire
- Séparez les données utilisateurs suivant les applications qui les utilisent
- notion de charge : Afin de diminuer la contention E/S,placer les fichiers des tablespaces différents sur des disques distincts.

IV.1.2/ Les Ordres SQL associés aux tablespaces (voir annexe et tp5**):**

```
SQL> CREATE TABLESPACE ...
SQL> DROP TABLESPACE...
SQL> ALTER TABLESPACE...

SQL> select tablespace_name from dba_tablespaces;
```

IV.1.3/ ON LINE/.OFFLINE de un tablespace :

Par défaut un tablespace à la création est ON LINE (et donc accessible), il peut être mis OFFLINE (et les fichiers qu'il continet par conséquent) pour en interdire l'accès ou pour certaines opérations de **maintenance** :

```
ALTER TABLESPACE toto OFFLINE;
```

Description des tablespaces de la base courante dans les vues Dba_tablespaces et Dba_data_files du dictionnaire.

IV.1.4/ Les fichiers du tablespace

Un tablespace contient AU MOINS un fichier. Celui ci est créé lors de la création du tablespace, de manière automatique par Oracle, en fonction des paramètres données par la commande CREATE ou ALTER tablespace (emplacement du fichier, nom et taille).

lors de la suppression du tablespace (DROP TABLESPACE...) les fichiers correspondant ne sont PAS SUPPRIMES par Oracle !! il faut le faire manuellement au niveau du système Unix ou Windows (rm, del...)

IV.1.5/ Quelques exemples SQL pour les tablespoaces et les fichiers

- creation d'un tablespace nommé RBS contenant un fic de 10MO et des EXTENTS de 1MO :
  ```
  CREATE TABLESPACE test DATAFILE '\home\database\test\test1.dbf' SIZE
  10M DEFAULT STORAGE ( INITIAL 1024K NEXT 1024K PCTINCREASE 0);
  ```

- `create table f(N number(10), S number(10)) tablespace test ;`

- `alter user mon_utilisateur default tablespace test;`

-

- ```
  SELECT SUM(BYTES)/1024 "Taille en KO" FROM DBA_FREE_SPACE
                                          WHERE TABLESPACE_NAME= 'test';
  ```

Description de la table systeme : `DBA_FREE_SPACE`

Nom de la colonne	Description
TABLESPACE_NAME	Name of the tablespace containing the extent
FILE_ID ID	number of the file containing the extent
BLOCK_ID	Starting block number of the extent
BYTES	Size of the extent in bytes
BLOCKS	Size of the extent in Oracle blocks

- DROP TABLESPACE *tablespace_name* [INCLUDING CONTENTS [AND DATAFILES][CASCADE CONSTRAINTS]];

- `select file_name from dba_data_files where tablespace_name='test'`
  ```

IV.1.6/ Extension du tablespace

Pour augmenter la taille d'un tablespace, il y a 2 solutions :

- Ajouter un fichier au tablespace, qui sera chaîné au premier :
  ```
  ALTER TABLESPACE toto ADD DATAFILE '\home\database\test\test2.dbf' SIZE 10M ;
  ```

  Une table, peut "s'étaler" sur plusieurs fichiers. Ainsi le fait qu'une table sature un tablespace n'est pas bloquant il suffit d'augmenter la taille du tablespace.

- mettre le fichier du tablespace en AUTO extension
  ```
  ALTER DATABASE DATAFILE test.dbf AUTOEXTEND ON ;
  ```

IV.1.7/ Les différents types de tablespaces spéciaux

Tablespaces en lecture seule (READ ONLY tablespaces)
Ces tablespaces sont utilisés (on s'en serait douté) en lecture seule. Ils permettent de stocker des données statiques (ou variant très peu souvent, éventuellement sur des CDROMS, et ne rentrent pas en ligne de compte dans les sauvegardes / restaurations.

Pour modifier les données d'un Tablespace READ ONLY il est évidemment obligatoire de modifier préalablement son statut.

```
ALTER TABLESPACE toto READ ONLY;
ALTER TABLESPACE toto READ WRITE;
```

Tablespaces temporaires (temporary tablespaces)
On peut (et doit) créer un tablespace temporarire par défaut autre que SYSTEM, où seront stockées toutes les données temporaires (utilisées lors des tris, création d'index, jointures, etc). Ils sont définis lors de la création de la base :

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

En plus de ce tablespace temporaire par défaut, chaque utilisateur peut se voir assigner un tablespace temporaire particulier

```
CREATE TEMPORARY TABLESPACE mon_temp TEMPFILE  '/oracle/data/temp01.dbf'
SIZE 20M EXTENT MANAGEMENT LOCAL UNIFORM SIZE 10M

CREATE USER toto IDENTIFIED BY tutu DEFAULT TABLESPACE data
QUOTA 100M ON data TEMPORARY TABLESPACE temp_ts
```

IV.2/ Les tables, Les indexe, et les clusters

Une table est l'élément constitutif d'un tablespace. La table est constituée de segments.
On n'entre pas dans le détail d'une table (concept supposé acquis).

Syntaxe

```
CREATE TABLE [schema.]table
     (  column datatype [DEFAULT expr] [column_constraint] ...
       | table_constraint
    [,  column datatype [DEFAULT expr] [column_constraint] ...
       | table_constraint ]...)

     [ [PCTFREE  integer] [PCTUSED  integer]
       [INITRANS integer] [MAXTRANS integer]
       [TABLESPACE tablespace]
       [STORAGE storage_clause]
     | [CLUSTER cluster (column [, column]...)] ]
     [ ENABLE   enable_clause
     | DISABLE disable_clause ] ...
     [AS subquery]
```

Exemple :

```
CREATE TABLE  hr.emp
(     empno NUMBER(5)        PRIMARY KEY,
      ename VARCHAR2(15)     NOT NULL,
      job   VARCHAR2(10),
      mgr   NUMBER(5),
      hiredate    DATE       DEFAULT sysdate,
      sal   NUMBER(7,2),
      com   NUMBER(7,2),
      deptno      NUMBER(3)        NOT NULL
           CONSTRAINT dept_fk REFERENCES hr.departments
           (department_id)
)

TABLESPACE    tbs1
STORAGE
(
      INITIAL       50K
      NEXT      50K
      MAXEXTENTS    10
      PCTINCREASE   25
) ;
```

Commentaire :
La table emp est créée dans le schéma `hr`.
La colonne `empno` est définie comme étant la clé primaire.
La colonne `ename` est définie comme obligatoire.
La colonne `hiredate` reçoit par défaut la date du jour.
La colonne `deptno` est obligatoire et le département doit exister dans la table departments.
La table est créée dans le tablespace `tbs1` avec ses informations de stockage soit :
un extent initial de 50 Ko, les extents ultérieurs de 50 Ko également, un maximum de 10 extents possible et un taux d'accroissement de 25% pour chaque nouvel extent créé.

Les **index** sont déjà étudiés en chapitre 3 et TP2. Les **clusters** sont déjà illustrés en chapitre 3 et feront l'objet d'un exercice du TP 5 exercice 7 d'aujourd'hui .

IV.3/ Les segments

Un segment est constitué d'**extents** et rentre dans la constitution de la table. En effet une table est constituée d'au moins deux segments : les data segments et le rollback segment.

```
SELECT segment_name, tablespace_name, extents, blocks FROM DBA_SEGMENTs
```

Deux autres segments peuvent apparaître dans une table : l'index segment et le temporary segment.

Regardons d'un peu plus près à quoi correspondent ces différents segments.

Le data segment : ce segment qui, rappelons le, rentre dans la constitution de la table, sert à stoker toutes les données (les valeurs des différentes lignes) que contient la table.

Le rollback segement : ce segment stocke des données relatives aux transactions. En effet si une transaction ne peut aboutir (on verra plus tard des exemples), la transaction doit être annulée par la commande ROLLBACK. Dans ce cas, on doit être capable de restituer la base dans l'état initiale ou elle était (au départ de la transaction en cours. Pour ce qui est des mécanismes intervenants, nous verrons cela dans un chapitre ultérieur. Dans tous les cas, il a fallut garder des informations qui sont stockées dans ce segment.

```
CREATE ROLLBACK SEGMENT rbs01 TABLESPACE rbs;
DROP ROLLBACK SEGMENT name;
```

L'index segment : ce segment optionnel sert à stocker les informations relatives aux index crées sur la table. Rappelons que les index servent notamment a optimiser les temps d'accès aux données.

Le temporary segment : cet autre segment est utilisé pour stocker les résultats temporaires d'une requête PL/SQL ne pouvant directement d'exécuter en mémoire. Pour ce faire un segment est alloué pour les traitements intermédiaires puis désalloué directement à la fin de la transaction (d'ou son nom).

```
select s.username, u."USER",u.tablespace, u.contents,u.extents, u.blocks
from V$SESSION s, V$SORT_USAGE v
where s.saddr = u.session_addr;;
```

IV.4/ Les extents

L'extent est un ensemble de blocks consécutifs, et rentre dans la constitution du segment. Autrement dit, un segment est constitué de plusieurs extents.

1 block	1 block	1 block	1 block
1 block	1 block	1 block	1 block
1 block	1 block	1 block	1 block
1 block	1 block	1 block	1 block
1 block	1 block	1 block	1 block
1 block	1 block	1 block	1 block

1 extent

Tout comme il existe différents types de segments, il existe différents types d'extents.

L'allocation des extents, constituant le segment, est dynamique. C'est-à-dire que lorsque qu'un extent, contenant par exemple des données, est plein, et que des nouvelles données doivent être ajoutées, un nouvel extent est alloué et ce s'il y a suffisamment de place sur le tablespace courant. En effet, la taille du nouvel extent doit au moins être de la même taille que l'extent précédent.

```
SELECT extent_id, file_id, block_id, blocks FROM DBA_EXTENTSS
```

IV.5/ Les blocks

Le **block** est la plus petite unité logique de stockage que peut manipuler le système. Tout les blocs constituant la BD ont tous la même taille. Cette taille peut soit être celle par défaut (fixée par le SGBD), soit être fixée par l'administrateur de la base. Dans ce dernier cas, le paramètre DB_BLOCK_SIZE du fichier d'initialisation doit contenir cette valeur. Nous reparlerons plus tard de l'utilisation du fichier d'initialisation.

En fait, un bloc est principalement divisé en trois sous-parties : la première contient des informations sur le bloc : c'est le header. Il indique par exemple à quel type de segment il appartient, la table et la ligne, ... La seconde est une zone vide pouvant être utilisée pour ajouter de nouvelles données et la dernière contient les données sur les lignes de la table à laquelle le block appartient.

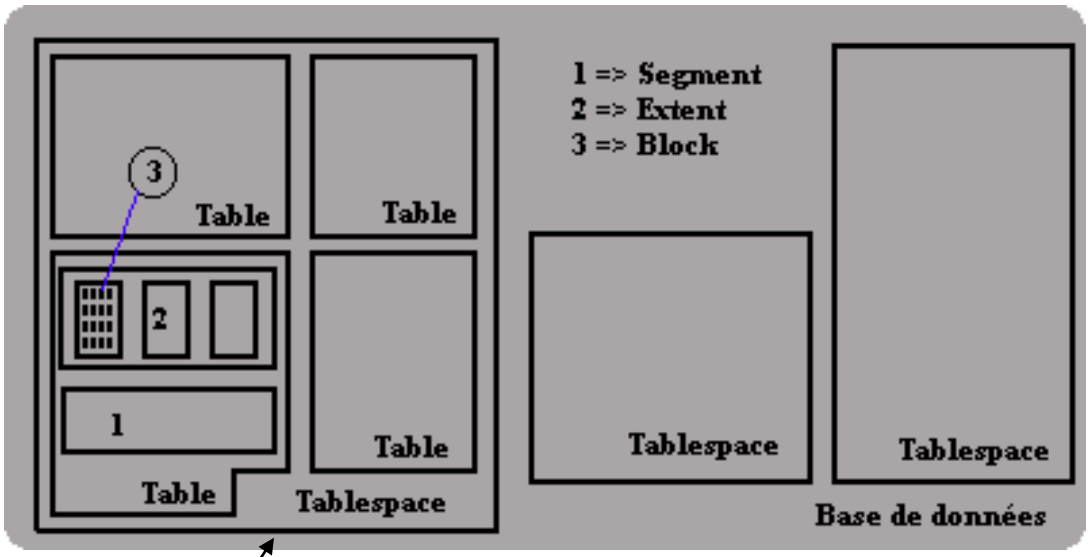
La zone vide est très importante et nous verrons dans un chapitre suivant, qu'en contrôlant la taille de cette dernière, on peut optimiser ou non les temps d'accès à la BD (base de données).

```
SELECT tablespace_name, count(*), max (blocks), sum(blocks)
                                                FROM dba_free_space
```

Exemple : Voir exercice 5.g de TP5

IV.6/ Récapitulation

Le schéma suivant replace, les unes en rapport aux autres, les diverses unités logiques existantes.



Les différentes structures logiques de la base de données.

```
Ce tablespace correspond à un ou
plusieurs fichiers sur disque
```

IV.7/ Création des tables système

La base créée est vide, et il faut exécuter des scripts qui installent des tables et programmes systèmes dans la base:

- catalog.sql crée le dictionnaire de données
- catproc.sql créé les structures pour PL/SQL.

Catalog.sql fait 820 K (sic). L'exécution dure donc une bonne vingtaine de minutes (`set echo off`).

```
connect sys/ the_pass as sysdba
@\...\admin\catalog.sql
@\...\admin\catproc.sql
```

Annexe

syntaxe tablespace

Syntaxe

- CREATE TABLESPACE nom
  DATAFILE file SIZE integer [K|M]
  [autoextend_clause]
  [, file [autoextend_clause]
  [MINIMUM EXTENT integer [K|M]]
  [DEFAULT storage_clause]
  [PERMANENT | TEMPORARY]
  [ONLINE | OFFLINE]

- storage_clause ::= STORAGE (
  [INITIAL integer [K|M]]
  [NEXT integer [K|M]]
  [MINEXTENTS integer]
  [MAXEXTENTS [integer | UNLIMITED]
  [PCTINCREASE integer])

- autoextend_clause ::= AUTOEXTEND
  [OFF | ON
  [NEXT integer [K|M]]
  [MAXSIZE
  [UNLIMITED | integer [K|M]]]]

Paramètres de stockage

- Paramètres qui influent sur l'allocation d'espace de segment
  - INITIAL : taille du 1er extent (min à 2*DB_BLOC_SIZE, par défaut à 5 blocs)
  - NEXT : taille du suivant (min à 1 bloc, par défaut à 5 blocs)
  - MINEXTENTS : nombre d'extents alloués lors de la création du segment (min à 1).
  - MAXEXTENTS : nombre maximum d'extent d'un segment (min à 1, par défaut dépend de la taille du bloc, i.e. DB_BLOC_SIZE, max =UNLIMITED)
  - PCTINCREASE : % de croissance de la taille de l'extent calculée par :

    Taille du nième extent
    = NEXT*(1+PCTINCREASE/100)^(n-2)
    - minimum à 0
    - valeur par défaut à 50
    - valeur calculée arrondie (multiple de 5 blocs).