Intro Oracle et SGBD

Dans oracle il faut distinguer :

- la partie logicielle proprement dite : Le système de gestion de base de données (relationnel) ou SGBDR, qui permet d'accéder à ...
- la partie donnée (la ou les bases de données physiques qui contiennent les données effectives). Ces 2 parties sont installées sur le même serveur et peuvent être accédé à travers le réseau par un grand nombre de poste clients simultanés.

I) Caractéristique d'un vrai SGBD

- gestion de gros volumes de données (en consultation et en mise à jour),
- Sécurité des données, qui se décline en :

- *disponibilité*
  Un SGBDR se doit d'offrir une bonne disponibilité des données. Une disponibilité totale des données est possible (temps de reprise nul) il suffit de s'en donner les moyens logiciels et matériels...

- *fiabilité*
  Des mécanismes de sauvegarde variés (physique, logique, off-line, on-line, totale, partielle, incrémentale), ainsi que des mécanismes de journalisation, et de reprise permettent de restaurer une information sans pratiquement aucune perte, dans tous les cas de problème matériel ou logiciel.

- *confidentialité*
  Tout n'est pas accessible à tout le monde! Se connecter à la base de données, donne un certain nombre de droits et de ressources en fonction d'un profil défini et maintenu par un administrateur. La granularité d'accès peut aller jusqu'à la vision unique d'un champ d'un enregistrement d'une table particulière. Encore une fois on ne manipule plus des fichiers...

- *cohérence*
  Que les données soient réparties ou non –dans ce dernier cas les mécanismes mis en jeux seront plus complexes– elles doivent être cohérentes. Cela sous entend, d'une part que les accès concurrents d'utilisateurs, notamment lors de mises à jour, ne doivent pas compromettre l'intégrité des données et d'autre part que ces dernières satisfassent aux contraintes d'intégrité du modèle, mais aussi aux règles de gestion de l'entreprise.

- *tracabilité*
  On peut, notamment en cas de problème important ou en cas d'attaque du système, recourir à l'analyse de traces ou de logs du SGBD. Le niveau de détail de ces traces est paramétrable, et concerne soit les aspects système, soit réseau, soit l'accès aux données élémentaires elles-mêmes.

- *concurrence d'accès en lecture et écriture* (avec une bonne granularité),
- *gestion (efficace) des transactions*,
- *portabilité sur différents OS, des données et du code*
- *Administrabilité* :
  - existence d'outils d'administration généraux : gestion des données, des utilisateurs, des fichiers physiques, des espaces logiques, des droits, des profils, des ressources systèmes, etc.
  - outils de surveillance
  en temps réel grâce à un moniteur, si possible graphique ou en temps différé grâce à des journaux ou à des traces paramétrables

- *possibilité d'export import* **:**
  de, ou vers des fichiers texte, des logiciels bureautique ou des fichiers Gros systèmes par exemple.

- *performances* **:**
  offrir de bonnes performances et des outils permettant de les mesurer et de les contrôler via des paramètres de configuration. Des processus d'optimisation en temps réel des requêtes complexes sont également souvent présents. Les données peuvent être indexées, de manière souple, dynamique et complète (index simples, concaténés, multiples, tables de « hashage », recherche textuelle, etc.). Un nombre important d'utilisateurs, ainsi qu'un volume conséquent de données peuvent être pris en compte.

II) Caractéristiques du relationnel

- modèle sous-jacent simple, bien formalisé et éprouvé...

- accès simple via un **langage de requêtes SQL**
  normalisé : Comme dans toute norme, elle est enrichie par les différents fournisseurs de logiciel. Mais si l'on se contraint à écrire des procédure respectant la norme ISO/ANSI 92 par exemple, on est quasiment garantie de pouvoir porter ce code sur Oracle, Informix ou Sybase... qui doit être...

- **complet :**
  - opérateur de projection (ou sélection), restriction, produit cartésien, union, différence,
- indépendance entre stockage physique et vision logique des données :
  Les données (le plus souvent des tables) sont référencées de manière logique. Un dictionnaire de données permet de retrouver la correspondance avec l'objet physique désiré. Ceci est bien sûr très utile dans les environnements ouverts, et offre une grande souplesse aussi bien lors du développement, que lors de la mise en production ou dans la phase de maintenance des applicatifs. La conséquence est que l'on pourra déplacer physiquement des données, par exemple les changer de serveur, renommer ou retailler un fichier sans pour autant retoucher le code des applications. Il n'y a pas de correspondance bijective entre un fichier et une table.

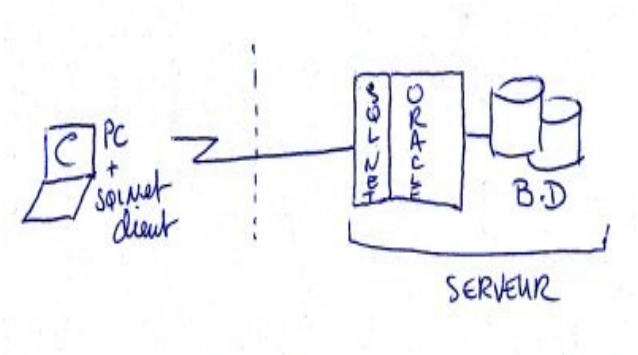
- existence de contraintes d'intégrité
  - intégrité de domaines (contrôle du type de la donnée)
  - intégrité de relation (existence d'une clé primaire : unique et toujours définie (non nulle))
  - intégrité de référence (contrôle de la cohérence d'attributs de tables différentes lors des mises à jour)

Le répertoire d'install est référencé par la variable ORACLE_HOME et la base de données est identifiée par son nom ORACLE_SID. Ces 2 variables d'environnement (ou entrée de la base de registre sur NT) sont obligatoires pour utiliser Oracle, et qu'il retrouve ses petits (fichiers de config réseau, fichier de messages, exécutables, etc.)
L'installation du logiciel Oracle, se termine souvent par la création d'une base de données de test créé dans la foulée ou générée sous forme de script à lancer de manière différée.

III) Client-serveur

Outre un protocole réseau (en général TCP/IP), Oracle requiert SQL*Net entre le client et le serveur. Plus exactement SQL*Net (pour TCP/IP) installé sur le client et un serveur SQL*Net qui tourne (démon Unix ou Service NT) sur le serveur.

remarque ; pour le développement et les test on aura souvent client et serveur Oracle sur la même machine, mais les accès se feront quand même en client-serveur en ré-entrant (LOOPBACK) sur la



machine.

III-1)La partie cliente

Oracle est en général installé sur C:\ORACLE sur un poste Windows.

la partie cliente d'Oracle comprend essentiellement :

- l'outil SQL*Plus : interface simple permettant de faire du SQL et du PL/SQL,
- SQL*Net client et les outils et fichiers de configuration associés (essentiellement tnsnames.ora),

-Depuis la version 11g de l'outil SQL Developper.
- des librairies standards (OCI) (qui peuvent être requise par des programmes tiers),
- de la documentation,
- des fichiers de support ou Required Support Files (fichiers de messages, de config, utilitaires, etc),
- éventuellement des outils d'administration (DBA Studio, Oracle Enterprise Manager)

III-2)Le serveur de données

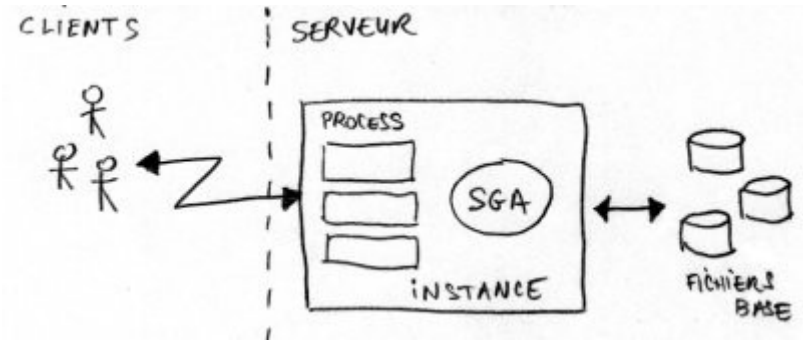
Le serveur de données est constitué :
- d'une **instance** en mémoire (= des programmes actifs d'accès à la base et une zone de données partagée : la SGA)
- et de **la base de données physique** (un ensemble de fichiers contenant les données et les structures internes d'Oracle) accédé par l'instance

III-2-1)L'instance (Voir cour sur l'instance)

Une instance est caractérisée par son identificateur : SID, généralement une variable ORACLE_SID positionnée dans l'environnement.

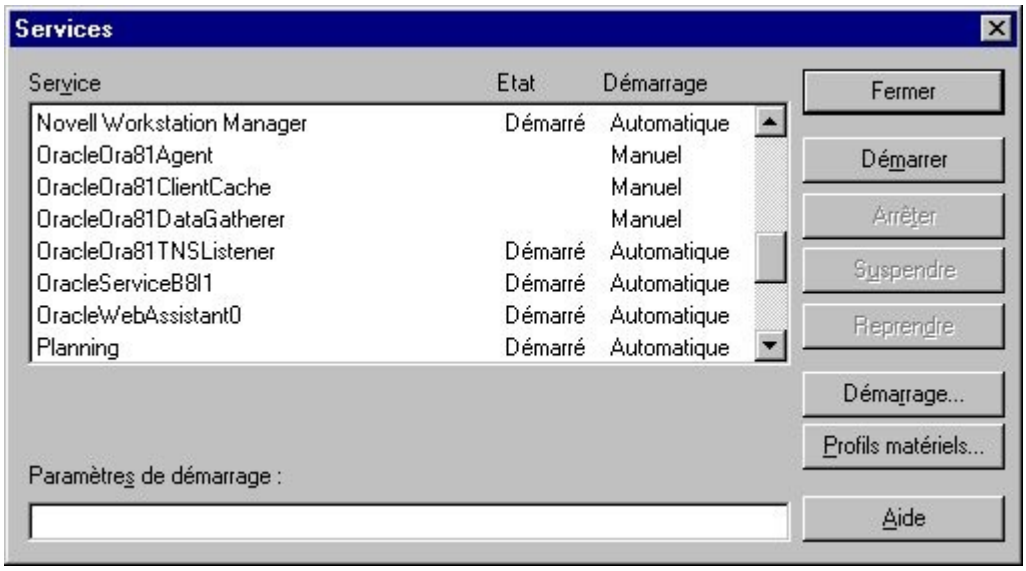
Une instance active, en mémoire ce sont :

- des programmes de fond (services ou processus) qui assurent la maintenance du serveur de données et les entrées / sorties fichiers
- des process server (dédiés ou non à un utilisateur)
- et une zone globale partagée : la SGA, qui contient essentiellement du cache de buffers



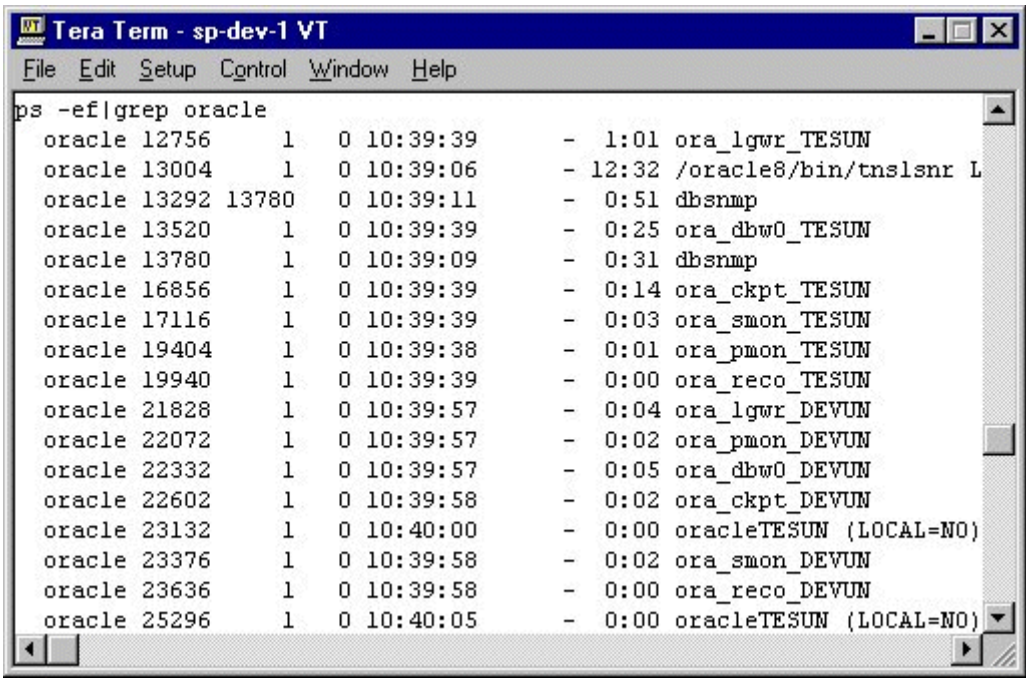
Pour un serveur 10.1 (Oracle 10g) sous Windows on aura par exemple les principaux services suivants :
- Oracle_*nom_home_oracle*_TNSListener : qui indique un serveur SQLNet présent (et actif ici) sur la machine
- OracleService*nom_de_la_base :* qui indique qu'un serveur Oracle est présent (et actif ici) sur la machine

Voir le panneau de configuration NT / services pour info :



ici la base s'appelle b8i1 et le oracle home 'Ora81'

Sur Unix un certain nombre de 'Background processes' tournent sur la machine, le propriétaire est généralement l'utilisateur Oracle (ce qui permet de les identifier). Les noms de process sont également suffixés par le nom de l'instance auquel ils sont attachés



Nous sommes ici sur une machine Unix, avec Oracle 8.0.5 installé. On a filtré les process actifs sur le mot oracle. On remarque ici que 2 bases sont actives : l'une s'appelle TESUN et l'autre DEVUN. On reconnaît les process de fond DBWR, LGWR, PMON, SMON, RECO notamment et deux process utilisateur distants connectés à TESUN...

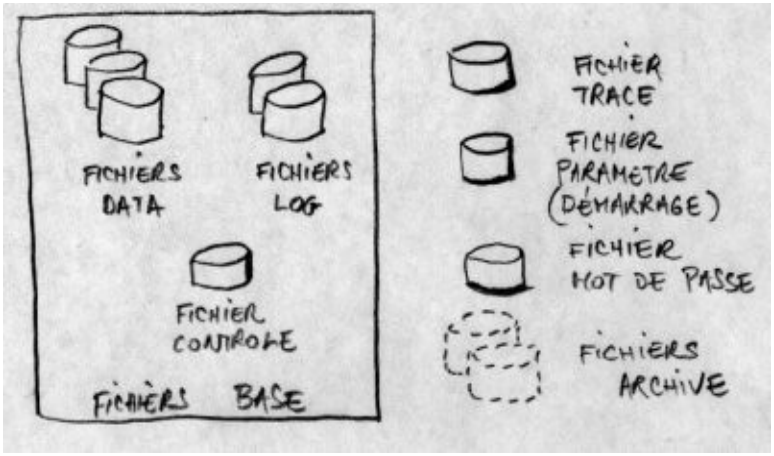
III-2-2) La base physique : les fichiers Oracle

Une base de données complète est constitué d'un certain nombre de fichiers :

- des fichiers de données (datafiles ou databasefiles)
- des fichiers journaux (logfiles ou redologfiles)
- des fichiers de controle (control files)
- un fichier d'initialisation ou de paramétrage (init file)

et de manière optionnelle

- un fichier de mot de passe (password file)
- et des fichiers d'archivage des journaux



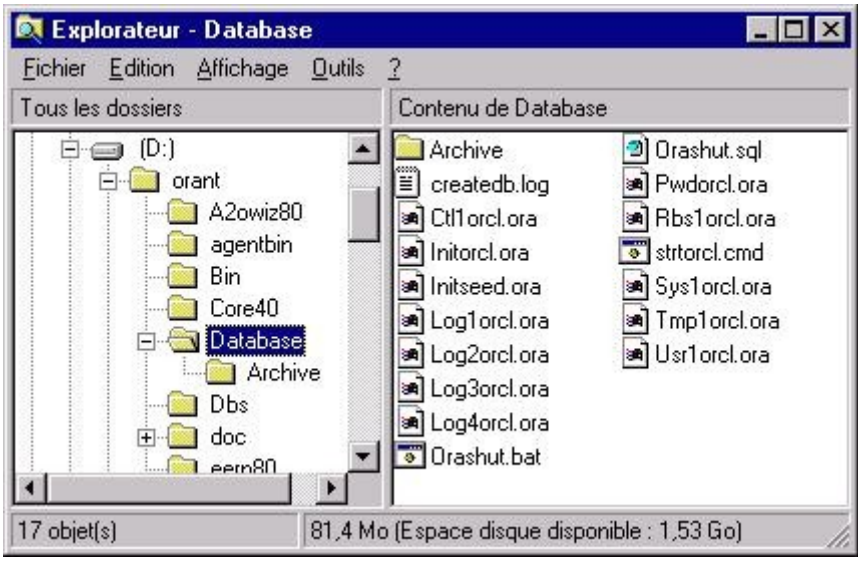
Les fichiers de données initiaux, les fichiers journaux et le control file sont créés par l'ordre SQL 'CREATE DATABASE' :

`exemple`
ici on crée un seul fichier de données et 2 group de logfiles (mirroir) sur 2 disques différents pour des raisons de sécurité.

```
CREATE DATABASE test DATAFILE 'd:\dataora\test_system' SIZE 10M
LOGFILE GROUP 1 ('d:\dataora2\test_log1a', 'd:\dataora2\test_log1b') SIZE 500K,
GROUP 2 ('d:\dataora3\test_log2a', 'd:\dataora3\test_log2b') SIZE 500K;
```

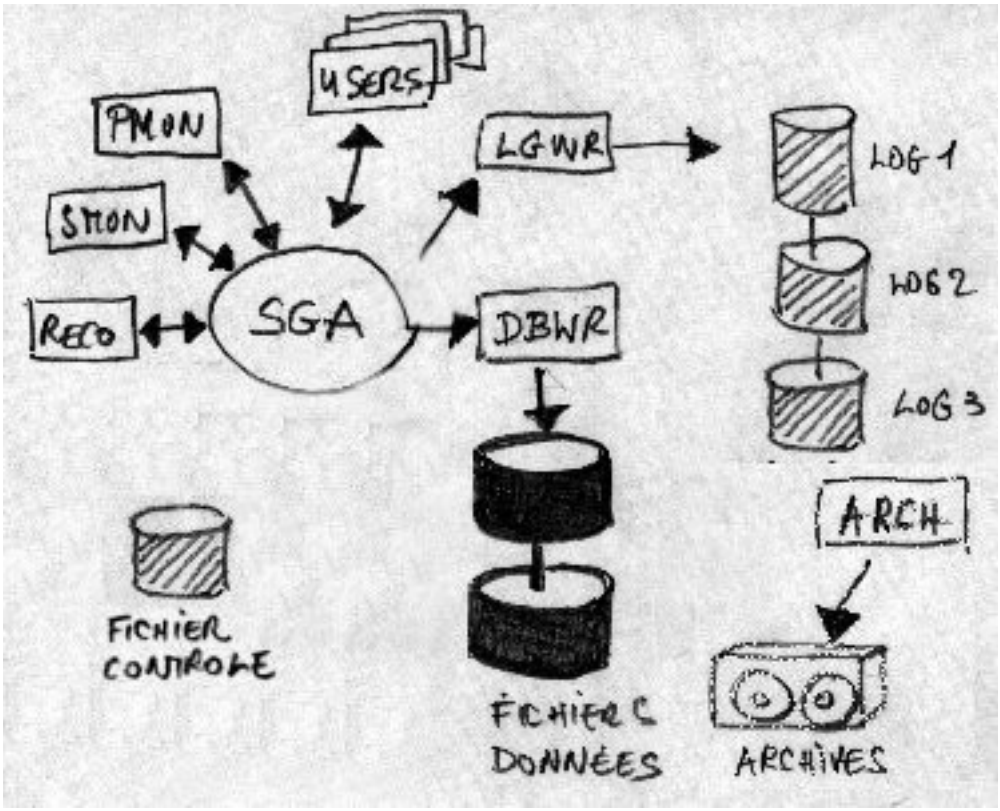
Généralement ces fichiers ont un nom assez explicite et ont le type '.ora' sur NT ou '.dbf' sur Unix.

Voici un exemple des fichiers installés lors de la création par défaut d'une 8.05 sur NT...



III-2-3) Relations entre instance & fichier

Les relations entre process et fichiers de la base physique apparaissent ci-après :



III-2-4) Les fichiers de données (data files)

Les fichiers de données contiennent ... les données proprement dites (essentiellement les tables et leurs lignes),
mais aussi les autres objets Oracle connexes aux tables : index, vues, synonymes, database links, procédures stockées, etc.).

III-2-4-1) Les tables

Contiennent les données utilisateur. Composées de colonnes. Descriptions des tables dans dba_tables.

```
CREATE TABLE salaires
( no_emp NUMBER , salaire NUMBER)
TABLESPACE TEMPORARY
STORAGE (INITIAL 50K NEXT 50K MINEXTENTS 1 MAXEXTENTS 25 PCTINCREASE 0)
```

```
SQL> describe salaires
 Name                              Null?    Type
 --------------------------------- -------- ----
 NO_EMP                                     NUMBER
 SALAIRE                                    NUMBER
```

on peut vérifier les caractéristiques de la table dans le dictionnaire :

```
SQL> select * from dba_tables
  2* where table_name = 'SALAIRES'

OWNER                          TABLE_NAME                     TABLESPACE_NAME
------------------------------ ------------------------------ ------------------------------
```

```
CLUSTER_NAME                   IOT_NAME                        PCT_FREE  PCT_USED INI_TRANS
------------------------------ ------------------------------ --------- --------- ---------
MAX_TRANS INITIAL_EXTENT NEXT_EXTENT MIN_EXTENTS MAX_EXTENTS PCT_INCREASE FREELISTS FREELIST_GROUPS
--------- -------------- ----------- ----------- ----------- ------------ --------- ---------------
LOG B  NUM_ROWS    BLOCKS EMPTY_BLOCKS AVG_SPACE CHAIN_CNT AVG_ROW_LEN AVG_SPACE_FREELIST_BLOCKS
--- - ---------- --------- ------------ --------- --------- ----------- --------------------------
NUM_FREELIST_BLOCKS DEGREE     INSTANCES  CACHE TABLE_LO SAMPLE_SIZE LAST_ANA PAR IOT_TYPE     T NES
------------------- ---------- ---------- ----- -------- ----------- -------- --- ------------ - ---
BUFFER_
-------
SYSTEM                         SALAIRES                      USER_DATA
                                                                    10        40         1
      255          51200       51200           1          25            0         1               1
YES N
                             1          1     N ENABLED                      NO               N NO
DEFAULT
```

Il est plus judicieux comme ici de préciser la clause STORAGE au niveau de chaque table plutôt qu'au niveau du tablespace, les volumes pouvant être très différents d'une table à l'autre....

```
CREATE TABLE mini_emp (empno NUMBER CONSTRAINT pk_emp PRIMARY KEY,
ename VARCHAR2(10) CONSTRAINT upper_ename CHECK (ename = UPPER(ename)),
hiredate DATE DEFAULT SYSDATE,
deptno NUMBER(2) CONSTRAINT fk_deptno REFERENCES scott.dept(deptno) )
```

on voit ici un certain nombre de contraintes d'intégrité qui faciliteront les développements

III-2-4-2) Les index

Accélérateurs. Externes aux tables. Peuvent être créés / détruits à tout moment. Se remplissent dynamiquement au fur et à mesure des mises à jour de la table indexée. Organisés en B-TREE. Description dans dba_indexes;

create [unique] index i1 on t1(col1)

rem : un index peut ne pas être unique. Une clé primaire dans une table est représentée sous forme d'index unique par Oracle.

III-2-4-3) Les clusters

Accélérateurs. Les clusters sont des segments spéciaux qui contiennent plusieurs tables fusionnées, suivant en général une colonne de jointure. On peut les voir comme des jointures physiques. remarque : un certain nombre de tables du dictionnaire sont organisées en cluster.
Voir dba_clusters

les objets qui suivent ne sont pas des segments. Ils sont des références à d'autres objets (segments ou non) dont la description est stocké dans le dictionnaire de données. Il ne prennent donc pas vraiment de place physique

III-2-4-4) Les vues

Une vue est une fenêtre sur une table. Elle ne contient pas de données. Stockée dans le DD sous forme de 'select nommé'.
La mise à jour d'une vue est en fait la mise à jour de la table 'A TRAVERS' la vue. Il n'y a pas de duplication de données.
Une vue peut porter sur plusieurs tables, éventuellement distantes !

Les vues sont décrites dans la vue dba_views du dictionnaire de données.

create view emp_10 as select empno numero, ename nom
where deptno = 10

l'une des colonnes de dba_views est de type LONG. Si on veut visualiser la totalité des informatiions sous SQL*Plus, utiliser la commande 'SET LONG 2000' par exemple pour ne pas tronquer l'affichage !

il existe des **vues paramétrées** qui peuvent rendre de grands services pour restreindre certains type

d'accès

Si on veut restreindre l'accès à certaines périodes horaires par exemple :

```
CREATE VIEW emp_ouvrable
AS SELECT * FROM EMP
WHERE TO_CHAR(SYSDATE,'HH') BETWEEN '08' AND '17'
```

en dehors de la période 8H - 17H le prédicat est faux et la vue ne renvoie aucune données !

Si on veut restreindre l'accès a certains postes de travail :

```
CREATE VIEW EMP_RESTREINTE
AS SELECT * FROM EMP
WHERE USERENV('TERMINAL') IN ('PC1', 'PC3')
```

seuls les postes clients dont l'identification réseau est 'PC1' ou 'PC3' seront autorisés !

III-2-4-5) Synonymes

Type	propriétaire	syntaxe
public	pseudo user 'PUBLIC'	create public synonym nom_syn for [propr.]objet
privé	le créateur (user courant)	create synonym nom_syn for [propr.]objet

rem : seul un DBA peut créer (ou détruire) un synonyme public.

Un synonyme public doit être unique dans la base.h

Le fait que le synonyme soit public ne veut pas dire que l'objet pointé est accessible par tout le monde, il faut en plus donner des droits si nécessaire!

III-2-4-6) Database links

Les database links sont des référence à des comptes utilisateurs distants.
Ceci permet au sein d'une même session SQL d'accéder à différents objets de bases réparties sur le réseau. On peut par exemple définir un database link compta_bordeaux qui référence le schéma compta qui se trouve sur la base distante localisée sur le serveur de bordeaux. Ensuite on pourra accéder à une table ou vue distante via ce database link. Une description des database links se trouve dans la vue USER_DB_LINKS du dictionnaire.

exemple

```
SQL> CREATE DATABASE LINK compta_bordeaux CONNECT TO compta IDENTIFIED BY xyz
USING 'la_base_de_bordeaux' ;
SQL> SELECT * FROM balance@compta_bordeaux ;
-- on peut le rendre transparent grace aux synonymes
SQL> CREATE SYNONYM balance FOR balance@compta_bordeaux ;
SQL> SELECT * FROM balance ;
```

III-2-4-7) Les séquences

Une séquence est un compteur programmable stocké en mémoire par Oracle et utilisable de manière partagé.
Il est en général utilisé pour fournir des no de clé d'enregistrements.
Pour créer une séquence :

```
create sequence nom_seq start with no_debut
increment by saut
```

```
maxvalue valeur_max cycle|nocycle cache|nocache
```

Pour utiliser une séquence :
la valeur courante de la séquence est donnée par `nom_seq.currval` et la suivante par `nom_seq.nextval`.

Exemple

```
SQL> create sequence seq_no_cli
start with 1000 increment by 100;
SQL> insert into clients (no, nom) values (seq_no_cli.nextval, 'Martin');
```

☺ pour vérifier rapidement la valeur à suivre : `select seq_no_cli.nexval from dual`

III-2-4-8) Les programmes stockés

Les programmes stockés (procédures, fonctions ou packages) sont des programmes codés en PL/SQL, nommés, et stockés dans un schéma dee la base.
Nécessite le privilège 'Create procedure' ou 'create any procedure' suivant qu'on est dans son propre schéma ou dans un autre schéma.

III-2-4-9) Procedures

Exemple de procédure de crédit d'un compte (qui prend en entrée le no du compte et le montant à créditer)

```
CREATE PROCEDURE credit (no IN NUMBER, montant IN NUMBER)
AS
BEGIN
UPDATE compte SET solde = solde + montant
WHERE no_compte= no;
END;
```

III-2-4-10) Fonctions

Une fonction est une procédure qui retourne une (ou des) valeur(s).

```
CREATE FUNCTION lit_solde (no IN NUMBER)
RETURN NUMBER
IS solde NUMBER (11,2);
BEGIN
SELECT balance INTO solde FROM compte
WHERE no_compte = no;
RETURN (solde);
END;
```

no est un paramètre d'entrée et solde la valeur du solde du compte fournie en sortie

III-2-4-11)Packages

Un package est un ensemble encapsulé de procédures et ou de fonctions, ainsi que de données dépendant fonctionnellement.
Il est constitué de 2 parties : la spécification qui déclare les objets publics du package et le corps (body) qui définit ces objets.
Pour créer ces 2 entités on utilisera respectivement les ordres :
create package et create package body

III-2-4-12) Triggers

Un trigger est un morceau de code PL/SQL, stocké dans la base, déclenché lors de l'occurrence d'un événement particulier. Il permet notamment de synchroniser des opérations entre plusieurs tables.
La plupart du temps les triggers sont déclenchés par la modification du contenu d'une table.
La liste des événements déclencheurs apparaît ci-après :

Type d'ordre ordre déclencheur : `DELETE, INSERT, UPDATE, CREATE, ALTER, DROP, SERVERERROR, LOGON, LOGOFF, STARTUP, SHUTDOWN`

exemple

```
-- trigger déclenché lors d'une insertion
-- ou d'une modification de la table client
SQL> CREATE OR REPLACE TRIGGER aff_discount BEFORE INSERT OR UPDATE ON clients
FOR EACH ROW WHEN (new.no_cli > 0)
DECLARE evol_discount number;
BEGIN evol_discount := :new.discount - :old.discount;
DBMS_OUTPUT.PUT_LINE(' evolution : ' || evol_discount); END;
/ -- FOR EACH ROW signale qu'une modification de 4 lignes
-- par un seul UPDATE déclenche 4 fois le trigger.
-- Si on ne souhaite qu'un seul déclenchement ,
-- on omet simplement la clause FOR EACH ROW.
```

Il aurait été plus judicieux de les appeler autrement, mais je traduis le terme Oracle 'data file'...

III-2-5) Ce qui n'est pas vraiment des données...dans les fichiers de données :

- Les Vues , synonymes, database links peuvent être considérés comme des données puisque "pointent" sur des données dans des tables.
- Les indexes et les clusters n'en sont évidemment pas, mais plutôt des "accélérateurs" de traitement
- Les procédures stockées (et fonctions et packages) non plus, mais des morceaux de programmes (PL.SQL) stockés dans la base.

et pire encore :

- les Images avant ou "rollback segments" qui sont des zones temporaires qui permettent de "revenir en arrière" sur des mises à jour, ou en d'autres termes d'annuler des transactions

Pourtant tout ce petit monde est stocké dans les data files !

IV) Les fichiers journaux (redo log files)

Les fichiers journaux servent à mémoriser toutes les modifications (validées ou non) faites sur la base, à des fins de reprise en cas de perte de données physiques.

La journalisation (assurée par le process LGWR) est un mécanisme permanent et obligatoire d'Oracle, pour assurer un minimum de sécurité des données.

Les fichiers journaux appartiennent à des groupe, et sont remplis de manière séquentielle et circulaire (quand on a fini de remplir le log du groupe courant, on attaque le log du groupe suivant suivant. Quand on arrive au dernier, on recommence à écrire dans le premier...).

C'est pour cela qu'il faut au minimum 2 (groupes de) redo log.

lorsqu'on boucle un cycle, on perd les premières journalisations qui ont été faites et l'on ne saura

pas toujours restaurer la totalité des données en cas de problème. C'est pour ça qu'on peut "archiver" tout l'historique des redo logs, en utilisant l'archivage optionnel.

Par sécurité on peut multiplexer ou mirrorer l'écriture dans plusieurs redo log en même temps (sur des disques différents bien sur).

Dans ce cas on a plusieurs fichiers par groupe, et LGWR écrit tous les fichiers du groupe courant en même temps.

V) Ma première connexion

Pour se connecter à une base Oracle, il faut :
- du courage ;-)
- le logiciel serveur Oracle installé, (voir exemples d'install sur différents OS)
- un client Oracle (qui peut faire partie du logiciel installé sur le serveur ou être sur être sur une machine distincte),
- un serveur de donnée disponible dont on connaît le nom (ALIAS SQL*Net),
- un compte utilisateur (et son mot de passe) avec lequel on va se connecter à la base de données
- un outil pour se connecter et faire une requête (en général SQL*Plus, dans ORACLE_HOME/bin ou dans le menu programme/oracle/outils de développement du menu démarrer de Windows)

en cas de problème

0) vérifier l'environnement Oracle (base de registre Windows et/ou variables d'environnement)
- ORACLE_SID -> le nom LOCAL de la base (souvent le même que l'Alias SQL*Net)
- ORACLE_HOME -> le répertoire d'install Oracle
- PATH -> qui doit contenir ORACLE_HOME/bin

1) vérifier que le serveur de données est actif et accessible : on va vérifier que l'instance est active et qu'une base de donnée physique est ouverte)
si ce n'est pas le cas essayer de lancer l'instance seule sur le serveur sans ouvrir la base
- positionner si nécessaire les variables ORACLE_SID et ORACLE_HOME
- lancer le server manager (SVRMGRxxx)
puis taper
> CONNECT INTERNAL
> STARTUP NOMOUNT PFILE=...) et
si cela passe , résoudre le problème au niveau des fichiers / répertoires de la base, sinon c'est un problème de démarrage de service ou de process

pour les version récentes d'Oracle. Le 'connect internal' est obsolète utiliser alors
$sqlplus 'system as sysdba'
et un mot de passe vide !

2) vérifier que le serveur SQL*Net (tnslistener) est actif (service NT ou process Unix)

3) vérifier que le nom réseau de la base donné lors de la connexion (Alias SQL*Net) est correct et traduisible par Oracle (par la méthode de résolution de nom Oracle en place, (en général le fichier TNSNAMES.ORA mais pas toujours...)

pour s'affranchir des questions de résolutions de noms on peut tester une connexion directe à Oracle en lui fournissant EXPLICITEMENT la chaîne de connexion complète (c'est à dire en se passant d'un Alias)

VI) Schéma de données

Les données sont stockées sous forme relationnelle, c'est à dire pour simplifier sous forme de tables, constituées de colonnes typées. Toutes les données qui appartiennent à un même utilisateur (table mais aussi vues, index, procédures, synonymes, etc.) s'appellent un schéma.

On accède au schéma (ou au compte utilisateur) en donnant le nom d'utilisateur et un mot de passe, via SQL*Plus par exemple.

Ainsi si je veux consulter ma table 'clients' :

```
sqlplus htondeur/herve@base_de_test
SQL> select * from clients;
```

ou créer ma table 'fournisseur' :

```
sqlplus htondeur/herve@base_de_test
SQL> create table fournisseurs
( no_fournisseur number(2), nom_fournisseur varchar(30),
tel varchar(10) );
```

Toutes les données et structures annexes d'un schéma sont décrites dans le dictionnaire de données en partie accessible à l'utilisateur.

VII) Les droits élémentaires

Ces droits sont par défaut très simples :
un utilisateur qui a le droit de créer des structures de données sur la base (à qui le DBA a donné le rôle 'RESOURCE' par exemple) a tous les droits sur SES données (DROP, DELETE, INSERT, UPDATE, SELECT).
Un utilisateur autre (connecté avec un autre nom/password) n'a AUCUN DROIT.

Pour référencer la table d'un autre utilisateur (s'il n'existe pas de synonyme comme raccourci) on préfixe le nom de la table du nom de l'utilisateur séparé par un '.'

```
sqlplus herve/test@base_de_test
SQL> select * from ddeleglise.clients;
SQL>
```

dans ce cas contraire (si on omet le préfixe) Oracle dit bizarrement que la table ou la vue n'existe pas...

```
sqlplus herve/test@base_de_test
SQL> select * from clients;
SQL>ERREUR a la ligne 1 :
ORA-00942: table or view does not exist
```

On peut (en tant que propriétaire ou créateur d'une donnée) donner des droits sur cette donnée :

SQL> CONNECT scott/tiger@la_base
SQL> GRANT SELECT ON mes_clients TO martin;
SQL> GRANT UPDATE ON mes_fournisseurs TO dupont;
SQL> CONNECT martin/wxdvgrs@la_base
SQL> SELECT * FROM scott.mes_clients;

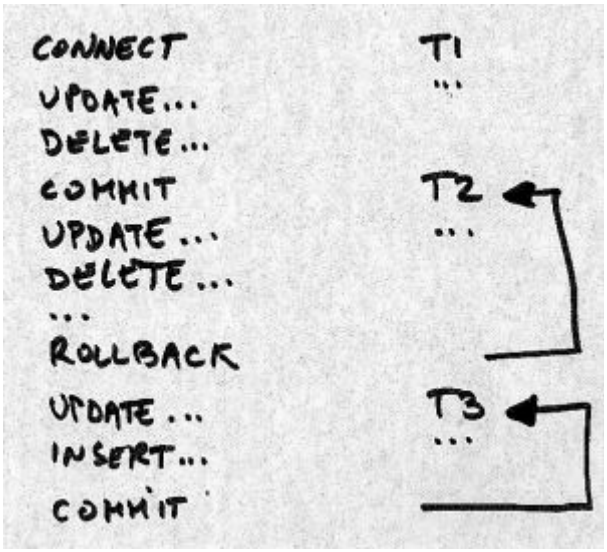
remarque : Le DBA a tous les droits sur tous les objets de la base (mais à la maison il est comme les autres !)

VIII) Les transactions

Un transaction est un ensemble cohérent de modifications faites sur les données. Une transaction est soit entièrement annulée (ordre SQL ROLLBACK) , soit entièrement validée (ordre SQL COMMIT).

Tant qu'il n'y a pas eu COMMIT, seul l'utilisateur courant voit ses mises à jour.

Une transaction débute lorsque l'on se connecte à la base (en début de session donc) ou lorsque la transaction précédente se termine.



.

remarque : la structure permettant de faire des retours en arrière (ROLLBACK) s'appelle un ROLLBACK SEGMENT et est gérée par le DBA.

Les COMMIT et ROLLBACK peuvent être implicites.
COMMIT : Lorsqu'on envoie un ordre SQL de création ou modification de table par exemple, ou plus généralement tout ordre du langage de définition de données.
ROLLBACK : en cas d'interruption anormale du traitement ou d'erreur.