ADMINISTRATION AVEC MYSQL

Rappel:

- Connexion à une base MYSQL:
  - Mysql –u root –p
  - Mysql - -host='NomHote' –u utilisateur –p motde passe
- Lister les bases présentes sur le serveur MYSQL
  - Show databases;
- Créer une base de donnée sur le serveur MYSQL
  - Create database NomBD;
- Choix d'une base de données sur le serveur
  - Use NomBD;

- Supression d’une Bd
  - Drop database NomBd;
- Création d’un utilisateur qui peut se connecter à distance depuis n’importe quel client
  - Create user NomUtilisateur@’%’ identified by ‘mot de passe’;
- Création d’un utilisateur qui peut se connecter seulement en local
  - Create user NomUtilisateur@’localhost’ identified by ‘mot de passe’
- Donner tous les priviléges à l’utilsateur toto sur toutes les bases
- Grant all privileges on *.* to toto;

1. Connaître la liste des tables en format InnoDB
   - SELECT table_name from information_schema.tables where engine like ‘InnoDB’ ;
2. Lister les login des utilisateurs ayant le privilège de sélection
   - SELECT DISTINCT user FROM mysql.USER WHERE select_priv = ‘Y’ ;
3. Donner à tout utilisateur anonyme le privilège de sélection sur la table ‘Example’ de la base ‘TEST’
   - GRANT SELECT on TEST.Example TO ‘’@’%’ ;
4. Désactiver la validation automatique des requêtes dans la session en cours
   - SET AUTOCOMMIT = 0 ;

ADMINISTRATION AVEC ORACLE

Présentation

- SGBD relationnel disponible sur MICRO, mini et gros systèmes
- Il dispose d'un:
  - outil de génie logiciel: DESIGNER
  - une interface interactive textuelle: SQL Plus
  - outil de développement Developer
  - outils d'administration: Enterprise Manager
  - outil de communication: SQL STAR

Instance d’une base

- Une base en exploitation est constitués d’une instance (ensemble de processus chargés en mémoire) associée à une base (ensemble de fichiers physiques)
  - **Démarrage d’une base**
    1. Démarrer une instance (**start up**): le démarrage d’une instance crée l’allocation mémoire pour une SGA et les processus associés. L’instance est paramétrée par la lecture du fichier d’initialisation.
    2. Monter une base (**mount**): consiste à associer une base de données à l’instance démarrée. L’instance ouvre les fichiers de contrôle spécifiés par la paramètre CONTROL_FILES. Ces fichiers contiennent les fichiers de données (Data file) et les fichiers de journalisation (redo log file)
    3. Ouvrir la base (**open**) : ouvrir les fichiers de données et de journalisation avec le statut ONLINE. La base est maintenant accessibles aux utilisateurs

Instance d’une base

Arrêt d’une base

1. Fermer la base (**close**): les données présentes dans la SGA sont sauvegardées dans les fichiers physiques du disque. Puis ces fichiers sont fermées et par conséquent la base devient inaccessible.
2. Démonter la base (**dismount**): La base est dissociée de la SGA; les fichiers de contrôle sont fermés.
3. Arrêter l’instance (**shut down**) : la mémoire occupée par l’instance est libérée

Instance et processus

- Base de donnée démarrée implique une création d'une instance
- Instance: ensemble de processus chargé en mémoire composé de :
- SGA: System Global Area espace mémoire qui contient les informations partagées par tous les utilisateurs.
  - Buffer de données (**Database Buffer**) cache mémoire des données de la base pour limiter les accès disques
  - Zone de partage des ordres SQL (**Shared Pool**): cache mémoire qui contient les arbres de parcours et les plans d'exécution des ordres SQL pour économiser la mémoire
  - Cache des modifications (**Redo Log Buffer**) qui contient les modifications sur les données (données avant et après modification, les modifs effectuées, toutes les transactions validées ou non encore validées). Quand le cache est plein, les données sont écrites dans un fichier (**Redo Log File**). Les modifications présentes dans ces deux structures permettent de défaire des opérations

Présentation

- Les processus d'arrière plan (**background processes**) gèrent le fonctionnement interne du SGBD (primitives d'accès aux supports, mise à jour du dictionnaire, sécurité des opérations)
- Les processus utilisateurs (**user processes**) sont associé à chaque utilisateur connecté et exécutent du code SQL via la SGA. Ils communiquent avec le SGBD à travers les processus serveurs.
- Les processus serveurs (**server processes)** sont crées pour traiter les requêtes soumises par les processus utilisateurs connectés à une instance. Leur rôle est d'analyser et exécuter les commandes SQL soumises , de transférer les blocs de données du disque vers la SGA et de communiquer les résultats des requêtes aux applications.

Structure d’une base Oracle

- Structure physique
  - Les fichiers de contrôle (control files)
  - Les fichiers de données (data files)
  - Les fichiers de reprise (redo log files)
- Structure logique
  - Database
    - Espace de disque logique (Tablespace)
      - Segments
        - Extensions
          - Blocs de données

Structure physique

- Les fichiers de contrôle (**control files**)
  - Contient les informations relatives à la structure physique : nom de la base, nom et localisation des fichiers de données et de reprise **(select * from v$controlfile)**
- Les fichiers de données (**data files**)
  - Stockent les objets de la base (tables, vue, index) ainsi que le dictionnaire des données d'oracle (méta-base). Taille minimale 2MO **(select * from v$dbfile)**
- Les fichiers de reprise (**redo log files**)
  - Contienneent les modifications intervenues sur les données de la base. Reprise à chaud: application des mises à jour non appliquées mais sauvegardées **(select * from v$logfile)**

Structure logique

- Cette structure fait le lien entre les objets de la base et la structure physique
- Les éléments sont:
  - **Espace de disque logique** (tablespace) est une unité de sauvegarde qui regroupe les objets logiques d'une application. Une tablespace est associée à un utilisateur durant sa création afin de contrôler l'espace disque alloué à l'utilisateur.
  - **Les segments** (segments): lieu de stockage d'une structure (table, index, partition de table, etc.) (ensemble d'extensions contenant les données d'une structure logique). **Remarque** : seuls les objets physiques peuvent être des segments. Ainsi une vue ou un synonyme n'est pas un segment…
  - **Les extensions** (extents): unité logiques d'allocation d'espace composée d'un ensemble contigu de blocs de données alloués simultanément à un segment
  - **Les blocs de données** (Data blocks): ce sont les plus petites unités d'entrées-sorties utilisées dans la base. La paramétre DB_BLOCK_SIZE défini la taille d'un bloc qui doit être un multiple de la taille d'un bloc physique géré par le SE (cluster ou Block OS)

Tablespace

- Une base est décomposée en tablespaces :
  - de 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.
- On peut également stocker les datas et les index dans un même tablespace, et obtenir ainsi une base minimale peu structurée, peu performante et peu sécurisée :

Tablespace

- Création d'un espace de disque logique nommé TBF contenant un fichier de 10MO et des EXTENTS de 1MO

CREATE TABLESPACE TBF

DATAFILE 'TBF1.dbf' SIZE 10M

DEFAULT STORAGE

(INITIAL 1024K NEXT 1024K PCTINCREASE 0);

Tablespace

- **Mettre en offline une tablespace**

  ALTER TABLESPACE TBF1 OFFLINE;

- **Changement des paramètres d'un tablespace existant**

  ALTER TABLESPACE TBF1
  DEFAULT STORAGE (INITIAL 100K NEXT 100K PCTINCREASE 1);

- **Ajout ajout d'un fichier de 10M**

  ALTER TABLESPACE TBF ADD
  DATAFILE ‘TBF2.dbf' SIZE 10M;

- **Rendre le fichier auto-extensible avec un pas de 5M et un maximum e 100 M**

  ALTER DATABASE DATAFILE ‘TBF2.dbf' AUTOEXTEND ON
  NEXT 5M MAXSIZE 100 M;

Extension tablespace

La taille d'un tablespace est la taille de son (ses) fichier(s) d'origine.
Pour augmenter la taille d'un tablespace, il y a 2 solutions :

- Ajouter un fichier au tablespace, qui sera chainé au premier
  - ALTER TABLESPACE toto ADD DATAFILE ‘toto.dbf’ ;

- Mettre le fichier existant du tablespace en auto-extension
  - ALTER DATABASE DATAFILE ‘toto.dbf’ AUTOEXTEND ON;

- Une table (et tout segment en général) , 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.
- **ATTENTION** :
- la clause AUTOEXTEND spécifie la taille d'extension du fichier d'un tablespace.
- La clause STORAGE INITIAL, NEXT. spécifie la taille d'extension d’un SEGMENT du tablespace (table., index, etc.).
- Ces 2 paramètres sont totalement indépendants. La preuve en est qu'une table (un segment de données) est forcément en allocation dynamique alors qu'un fichier peut avoir une taille fixe (AUTOEXTEND OFF)

Segment

- Segment de données (data segments)
  - stocke les tables systèmes, les tables utilisateurs et les clusters. Une table est associée à un segment de données créé lors de la création de ta table.
- Segment d'index (index segments)
  - Index
- Segment temporaire (temporary segments)
  - utilisé en interne par Oracle, si la zone mémoire de tri est insuffisante
- Segments d’amorçage (Bootstrap segments)
- segment d'annulation (rollback segment)
  - stocke l'image avant modification des données

Segment

- On peut forcer les segments de données et d'index à s'implanter dans un tablespace particulier :
  1. explicitement à la création du segment
  2. implicitement en affectant un tablespace par défaut à l'utilisateur qui va créer le segment.

1. **explicitement : SQL**> create table Compte (numCompte Integer, ...) tablespace TBF;
2. ou bien (implicitement)
   - **SQL**> create user toto identified by passer default tablespace TBF;
   - **SQL**> create table Compte (numCompte Integer, ...)

Allocation dynamique

- L'allocation obéit à des règles définies par une clause STORAGE.
- Oracle applique d'abord
  - la clause STORAGE du segment (définie explicitement dans un create table par exemple),
  - sinon il utilise la clause storage du tablespace (définie explicitement lors du create tablespace),
  - sinon il utilise les valeurs par défaut (implicites) du tablespace.

- Au départ, lors de la création du segment (table, index ou autre) Oracle alloue MINEXTENTS extents de taille INITIAL.
- Ensuite, lorsque le segment se remplit, quand le(s) premier(s) extent(s) est (sont) plein(s), il alloue un extent de taille NEXT. Ensuite il augmente la taille des extents d'un pourcentage fixé par PCTINCREASE. La limite est définie par MAXEXTENTS.
- SQL> create table Compte (numCompte number,...) storage (initial 10K next 10K pctincrease 50 maxextents 100) ;

Séquences et Déclencheurs

```
CREATE SEQUENCE ma_sequence START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
CREATE SEQUENCE ma_sequence START WITH -1 INCREMENT BY -1 MINVALUE -9999;
CREATE SEQUENCE ma_sequence START WITH 1 INCREMENT BY 1 MAXVALUE 9999  NOCYCLE;

select  ma_sequence.nextval from dual;
---intialise la séquence pour la première fois et donne le suivant pour les autres

select ma_sequence.currval from dual;
---affiche  la veleur courante et erreur si la séquence n'est pas encore initialisée


create trigger MyTrigger
before insert on matable for each row
begin
select  ma_sequence.nextval into :new.X from dual;
--- X est le nom du champ à auto-incrémeneter
end;
```