



SYSTÈME DE GESTION DE BASE DE DONNÉES

Responsable : Pr. Samba DIAW
CIBLE: DUT/DST 2 INFORMATIQUE



SOMMAIRE

I. Rappel

- A) Modèle Entité/Association
- B) Modèle relationnel
- C) Langage SQL

II. Administration avec MYSQL

III. Administration avec ORACLE

IV. Le langage PL/SQL



I. RAPPEL



A: Modèle Entité/Association



Le Modèle Entité/Association

- Le modèle Entité/Association est une représentation statique du système d'information de l'entreprise.
- Il a pour but d'écrire de façon formelle les données qui seront utilisées par le système d'information
- Il s'agit donc d'une représentation des données, facilement compréhensible. Cet aspect recouvre les mots qui décrivent le système d'information ainsi que les liens existants entre ces mots.
- Le formalisme adopté par la méthode Merise pour réaliser cette description est basé sur les concepts « entité-association

Les contraintes sur la relation d'héritage en MERISE

- Partition (XT) (Couverture et Disjonction)
 - Une personne est soit un étudiant, soit un enseignant mais pas les deux
- Exclusion (X) (Non couverture et Disjonction)
 - Une personne est soit un étudiant, soit un enseignant mais pas les deux mais il peut exister d'autres catégories de personne (par exemple PATS)

Les contraintes sur la relation d'héritage en MERISE

- Totalité (T) (Couverture et Non disjonction)
 - Une personne est soit un étudiant, soit un enseignant ou bien les deux
- Pas de contrainte (Non couverture et Non Disjonction)
 - Une personne est soit un étudiant, soit un enseignant ou bien les deux et peut il exister d'autres types de personne (PATs, par exemple)



B: Modèle Relationnel

Présentation du Modèle relationnel

- La description conceptuelle a permis de représenter le plus fidèlement possible les réalités de l'univers à informatiser.
- Cette représentation ne peut pas être directement manipulée et acceptée par un système informatique.
- Nécessité de passer du niveau conceptuel à second un niveau (logique) plus proche des capacités des systèmes informatiques

Modèle relationnel

- Modèle relationnel conçu par E.F. Codd en 1970
- Correspond à une définition rigoureuse des données
- Caractérisé par des règles d'intégrité garantissant la cohérence des données
- Utilise la puissance des opérateurs de l'algèbre relationnelle



Objectifs du modèle relationnel

- Permettre l'indépendance des programmes par rapport à la représentation interne des données
- Pouvoir traiter les problèmes de cohérence et de redondance des données
- Développer un langage non procédurale de manipulation des données

Les concepts

- 3 concepts fondamentaux :
 - Domaine : ensemble de valeurs
 - Relation : ensemble de colonnes
 - Attribut : nom d'une colonne
- Chaque attribut prend ses valeurs dans son domaine
- Deux attributs dans une même relation ne peuvent porter le même nom
- Chaque ligne de donnée de la relation est appelée tuple (ou n-uplet ou occurrence)

Transformation du schéma E-A/ schéma UML en un modèle relationnel

1. Toute entité/classe devient une relation
2. L'identifiant de chaque entité/classe devient clé primaire de la relation associée
3. Les attributs de l'entité/classe deviennent des colonnes pour la relation
4. Dans une association de type (1-N), l'identifiant de l'entité/classe la plus forte (mère) migre vers l'entité/classe la plus faible (fille) et devient une clé étrangère (colonne suivi d'un dièse)
5. Toute association/classe-association de type (m-n) donnera naissance à une nouvelle relation formée des identifiants des entités/classes qui participent à l'association en plus des propriétés éventuelles portées par l'association
6. Dans une association de type (1-1), l'identifiant de l'entité/classe la plus faible (cardinalité 0,1) migre vers l'entité/classe la plus forte (cardinalité 1, 1)

Transformation de l'héritage

7. L'héritage est traduit en application des règles suivantes:

- ❑ Décomposition par distinction (on conserve la super-entité (super-classe) mais aussi les sous-entités (sous-classes) dans le modèle relationnel)
- ❑ Décomposition ascendante (push-up) (les données des sous-entités (sous-classes) migrent vers la super-entité (super-classe))
- ❑ Décomposition descendante (push-down)) (les données de la super-entité (super-classe) migrent vers les sous-entités (sous-classes))



C: Le langage SQL

Historique de SQL

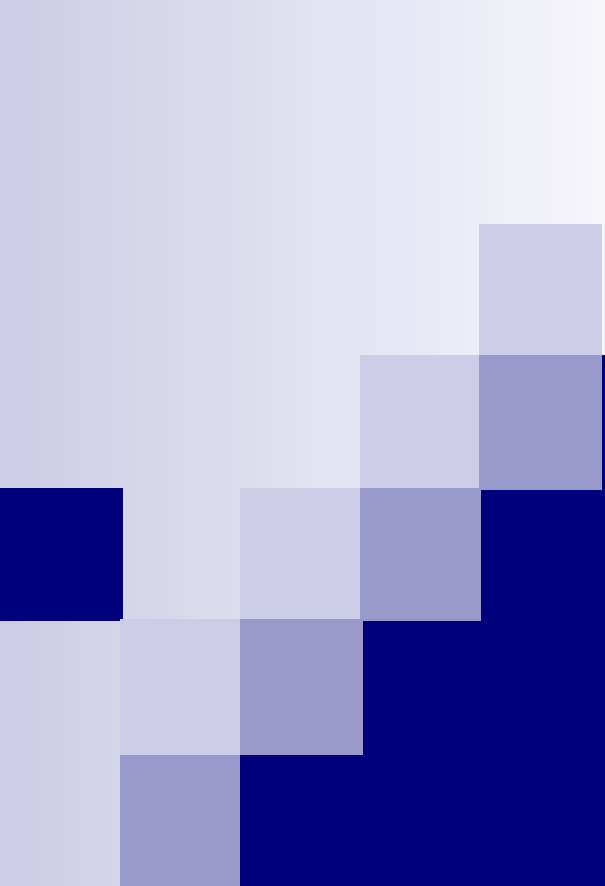
- Le langage SQL (Structured Query Language) est un langage élaboré dans les années 1970 par un informaticien britannique du nom de Edgar Frank Codd
- Oracle fut la première entreprise à miser sur ce langage
- Une requête SQL peut être portée sans modifications (ou modifications de mineures) de MySQL à Oracle ou d'Oracle à SQL Server

Structuration du Langage SQL

- Le langage SQL est composé de trois langages:
 - Le langage de définition des données (LDD) qui permet de créer ou modifier les structures d'une base de données
 - Le Langage de Manipulation des données (LMD) qui permet de consulter ou de modifier le contenu d'une base de données
 - Le langage de contrôle de données (LCD) qui permet de gérer les privilèges ou les différents droits des utilisateurs sur la base de données

Structuration du Langage SQL

SQL		
LDD	LMD	LCD
CREATE	SELECT	GRANT TO
ALTER (ADD, MODIFY, DROP, RENAME)	INSERT	REVOKE FROM
RENAME	DELETE	
DROP	UPDATE	



Langage de Définition des Données (LDD)



Syntaxe de Create Table

CREATE TABLE NOMTABLE (

Colonne1 type1 [DEFAULT valeur1] [NOT NULL] ,

Colonne2 type2 [DEFAULT valeur2] [NOT NULL],

CONSTRAINT nomContrainte1 typeContrainte1,

CONSTRAINT nomContrainte2 typeContrainte2

...);

- ***nomTable*** : peut comporter des lettres majuscules ou minuscules, des chiffres et les symboles, par exemple : `_`, `$` et `#`.
- ***colonnei typei*** : nom d'une colonne et son type (voir les types SQL.). La directive **DEFAULT** fixe une valeur par défaut. La directive **NOT NULL** interdit que la valeur de la colonne soit nulle.
- ***nomContrainte typeContrainte***: nom de la contrainte et son type (clé primaire, clé étrangère, etc.).
- `;` : symbole qui termine une instruction SQL

Les types de contraintes

- UNIQUE (*colonne1* [,*colonne2*]...)
- PRIMARY KEY (*colonne1* [,*colonne2*]...)
- FOREIGN KEY (*colonne1* [,*colonne2*]...
 - REFERENCES *nomTablePere* (*colonne1* [,*colonne2*]...)
 - [ON DELETE { CASCADE | SET NULL }]
- CHECK (*condition*)

- La contrainte **UNIQUE** impose une valeur distincte au niveau de la table (les valeurs nulles font exception à moins que NOT NULL soit aussi appliquée sur les colonnes).
- La contrainte **PRIMARY KEY** déclare la clé primaire de la table. Un index est généré auto- matiquement sur la ou les colonnes concernées. Les valeurs des colonnes clés primaires ne peuvent être ni nulles ni identiques

- La contrainte **FOREIGN KEY** déclare une clé étrangère entre une table enfant (*child*) et une table père (*parent*)..
 - La directive ON DELETE dispose de deux options :
 - CASCADE propagera la suppression de tous les enregistrements fils rattachés à l'enregistrement père supprimé,
 - SET NULL positionnera seulement leur clé étrangère à NULL
- La contrainte **CHECK** impose un domaine de valeurs ou une condition simple ou complexe entre colonnes
 - CHECK (note BETWEEN 0 AND 20),
 - CHECK (Sexe='M' OR Sexe='F')).

TYPE	DESCRIPTION	TAILLE
INTEGER ou int (taille)	Type des entiers relatifs	4 octets
SMALLINT	idem	2 octets
BIGINT	idem	8 octets
FLOAT	Flottant simple précision	4 octets
DOUBLE	Flottant double précision	8 octets
REAL	Synonyme de FLOAT	4 octets
NUMERIC (M,D)	Nombre décimal avec précision fixe.	M octets
DECIMAL (M,D)	idem	M octets
CHAR(M)	Chaînes de longueur fixe	M octets
VARCHAR(M)	Chaînes de longueur variable	$L+1$ avec $L \leq M$
BIT VARYING	Chaînes d'octets	Long. de la chaîne.
DATE	Date (jour, mois, an)	environ 4 octets
TIME	Horaire (heure, minutes, secondes)	idem
DATETIME	Date et Heure	8 octets
YEAR	Année	2 octets

Création de la Table Client

CLIENT				
<u>Numcli</u>	Nom	Prenom	Ville	Telephone

- CREATE TABLE CLIENT (
Numcli INTEGER NOT NULL,
Prenom VARCHAR (25) NOT NULL,
Nom VARCHAR (20) NOT NULL,
Ville VARCHAR (25) NOT NULL,
Telephone VARCHAR (12),
CONSTRAINT pk_Client PRIMARY KEY (Numcli)
);

Création de la Table Article

ARTICLE			
<u>Refart</u>	Designation	Categorie	Prix

- CREATE TABLE ARTICLE (
Refart CHAR (5) NOT NULL,
Designation VARCHAR (20) NOT NULL,
Categorie VARCHAR (25) NOT NULL,
Prix INTEGER NOT NULL ,
CONSTRAINT pk_Article PRIMARY KEY (refart)
);

Création de la Table Achat

ACHAT

Numclient#	Refarticle#	Dateachat	Quantite
------------	-------------	-----------	----------

- CREATE TABLE ACHAT (
Numclient INTEGER NOT NULL,
Refarticle CHAR (5) NOT NULL,
Dateachat DATETIME,
Quantite INTEGER,
Constraint PK_Achat PRIMARY KEY (Numclient,
Refarticle,Dateachat)
*CONSTRAINT fk_Ach_numclient_Cli FOREIGN KEY
(numclient) REFERENCES CLIENT (numcli)
on DELETE CASCADE,*
*CONSTRAINT fk_Ach_refarticle_ART FOREIGN KEY
(refarticle) REFERENCES ARTICLE(refart)*
);

Modification d'une structure d'une table

■ MODIFICATION

- ALTER TABLE CLIENT *ADD*
EMAIL CHAR (25) NOT NULL;
- ALTER TABLE CLIENT *MODIFY*
EMAIL CHAR (20) NOT NULL;
- ALTER TABLE CLIENT ADD
CONSTRAINT un_telephone UNIQUE
(telephone) ;

Suppression et renommage

■ Suppression d'une colonne ou d'une contrainte

- ALTER TABLE CLIENT *DROP COLUMN EMAIL;*
- ALTER TABLE CLIENT *DROP CONSTRAINT un_telephone;*

■ Renommage

- RENAME TABLE AncienNom TO NouveauNom;

■ Suppression d'une table

- DROP TABLE NOM_TABLE;



Langage de Manipulation des Données (LMD)

■ Selection

- SELECT [ALL/DISTINCT] att1, ..., attn FROM Table1,..., Tablen

WHERE condition_recherchée;

- Select [ALL/DISTINCT] * FROM Table1,..., Tablen WHERE condition_recherchée

■ Insertion de données

- INSERT INTO Nom_Table (colonne1,..., colonnen) VALUES (val1,...,valn);

■ Mise à jour des données

- UPDATE Nom_Table SET colonne1=nouvellevaleur WHERE condition;

■ Supression de données

- DELETE FROM Nom_Table WHERE condition;
- DELETE FROM Nom_Table;

EXEMPLE D'APPLICATION

CLIENT				
<u>Numcli</u>	Nom	Prenom	Ville	Telephone
1	DIAW	JEAN	LOUGA	77 442 23 33

INSERT INTO CLIENT VALUES

(1,'DIAW','JEAN','LOUGA','77 442 23 33');

EXEMPLE D'APPLICATION

ARTICLE			
<u>Refart</u>	Designation	Categorie	Prix
1B3F	YOUSSOU NDOUR	CD	3000

```
INSERT INTO ARTICLE VALUES('1B3F','YOUSSOU NDOUR','CD',3000);
```

EXEMPLE D'APPLICATION

ACHAT			
Numclient#	Refarticle#	Dateachat	Quantite
1	1B3F	30/01/2009	5

INSERT INTO ACHAT VALUES(1,'1B3F','2009-01-30 12:20:30 ',5)

Schéma de base de données

■ Gestion des séjours dans des stations balnéaires

- Station (nomStation, capacité, lieu, région, tarif)
- Activite (nomStation, libellé, prix)
- Client (id, nom, prénom, ville, région, solde)
- Séjour (idClient#, Nom station#, début, nbPlaces)

■ Gestion des achats des clients

- Client (Numcli, Nom, Prenom, Ville, Telephone)
- Article(RefArt, Designation, Categorie, Prix)
- Achat(Numclient#, Refarticle#, DateAchat, Quantité)

Requêtes imbriquées

- **EXISTS R**. Renvoie TRUE si R n'est pas vide, FALSE sinon.
- **T IN R** où est t un tuple dont le type est celui de R. TRUE si t appartient à R, FALSE sinon.
- **V cmp ANY R**, où cmp est un comparateur SQL. Renvoie TRUE si la comparaison avec *au moins un* des tuples de la relation unaire renvoie TRUE.
- **V cmp ALL R**, où est un comparateur SQL. Renvoie TRUE si la comparaison avec *tous* les tuples de la relation unair renvoie TRUE.

Requêtes imbriquées

- Les clients qui ont fait un achat à la date du 30 Janvier 2009
- `SELECT Prénom, nom FROM Client, Achat WHERE numcli = numclient AND dateachat = '2009-01-30';`
- Ici, le résultat Prénom, nom, provient de la table Client.
- D'où l'idée de séparer la requête en deux parties :
 - (1) on constitue avec une sous-requête les numéros des clients ayant fait un achat le 30 Janvier 2009,
 - puis (2) on utilise ce résultat dans la requête principale.
- `SELECT Prénom, nom FROM Client WHERE numcli IN (SELECT numclient FROM Achat WHERE dateachat = '2009-01-30');`

Requêtes imbriquées

- *Noms des articles les plus chers?*
 - SELECT designation FROM Article WHERE
prix >= ALL (SELECT prix FROM Article)
- *Dans quelle (s) station (s) pratique-t-on
une activité au même prix qu'à Saly ?*
 - SELECT nomStation, libelle FROM Activite
WHERE prix IN (SELECT prix FROM Activite
WHERE nomStation = 'Saly')

Sous-requêtes corrélées

- Exemple : *Liste des clients dès qu'il existe au moins un qui a fait un achat à la date du 30 Janvier 2009*
- `SELECT nom, prenom FROM Client WHERE EXISTS`
`(SELECT 'x' FROM Achat WHERE dateachat =`
`'2009-01-30' AND numcli = numclient)`
- Le numcli dans la requête imbriquée n'appartient pas à la table Achat mais à la table Client référencée dans le FROM de la requête principale.

Agrégation

- Les fonctionnalités *d'agrégation* de SQL permettent d'exprimer des conditions *sur des groupes de tuples*, et de constituer le résultat par *agrégation de valeurs* au sein de chaque groupe.
- La syntaxe SQL fournit donc :
 - Le moyen de partitionner une relation en *groupes* selon certains critères.
 - Le moyen d'exprimer des conditions sur ces groupes.

Fonctions d'agrégation

- Les fonctions d'agrégation s'appliquent à une colonne en général de type numérique:
 1. COUNT qui compte le nombre de valeurs *non nulles*.
 2. MAX et MIN (Maximum et Minimum)
 3. AVG qui calcule la moyenne des valeurs de la colonne.
 4. SUM qui effectue le cumul.

Exemple d'utilisation

```
SELECT COUNT(refart), AVG(prix), MIN(prix),  
MAX(prix) FROM Article;
```

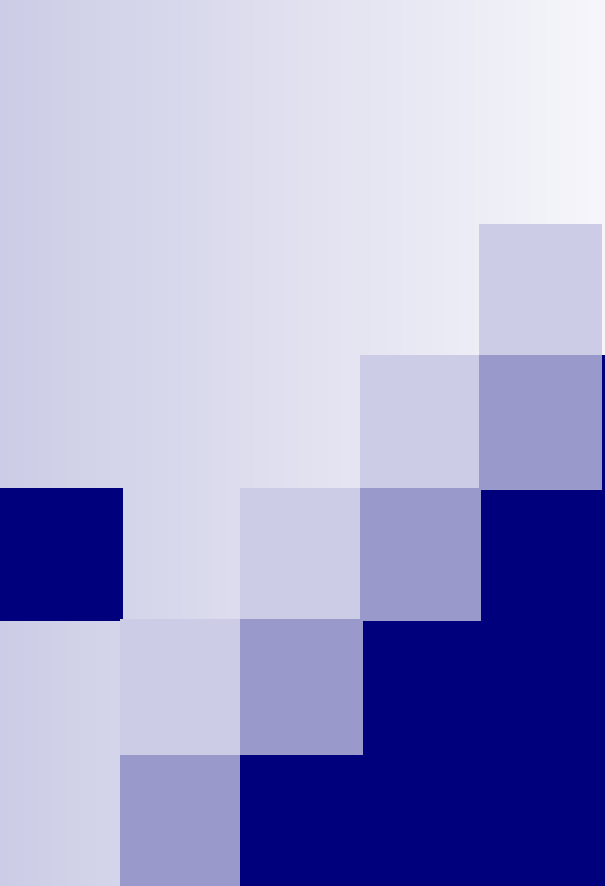
- Remarque importante : on ne peut pas utiliser simultanément dans la clause SELECT des fonctions d'agrégation et des noms d'attributs (sauf dans le cas d'un GROUP BY).
- La requête suivante est « incorrecte » et dénuée de sens.
 - **SELECT Designation, AVG(prix) FROM Article**
- Exemple : *Combien d'achats a effectué le client numéro 1?*
- *Select count (refarticle) as 'Nombre Achats' from Achat
WHERE Numcli= 1;*

Clause Group By

- Dans les requêtes précédentes, on appliquait la fonction d'agrégation à l'ensemble du résultat d'une requête
- Une fonctionnalité complémentaire consiste à *partitionner* ce résultat en groupes, et à appliquer la ou les fonction(s) à chaque groupe.
- On construit les groupes en associant les tuples partageant la même valeur pour une ou plusieurs colonnes.
- Exemple : Afficher les Numéros des clients avec le nombre d'achats effectués.
- `SELECT Numclient, COUNT(refarticle) as 'Nombre Achats' FROM Achat GROUP BY Numclient;`

Clause Group By

- Exemple : on souhaite consulter le nombre d'achats effectués par client, *pour les clients ayant fait plus de 5 achats*
- `SELECT numclient, count(refarticle) as 'Nombre Achats' FROM ACHAT GROUP BY numclient HAVING count(refarticle)>5;`
- L'interprétation est simple :
 - (1) on exécute d'abord la requête `SELECT ... FROM ...`
 - puis (2) on prend le résultat et on le partitionne par numéro de client ,
 - enfin (3) on calcule le résultat des fonctions.



Langage de Contrôle des Données (LCD)

Donner des privilèges

- Donner des permissions à un utilisateur
 - GRANT PERMISSION TO USER
 - GRANT PERMISSION ON NOM_TABLE TO USER;
- Donner des permissions à tout le monde
 - GRANT PERMISSION ON NOM_TABLE TO PUBLIC;
- Permettre à un utilisateur de disposer de droits et de les redistribuer
 - GRANT PERMISSION ON NOM_TABLE TO USER WITH GRANT OPTION;
- Donner tous les privilèges
 - GRANT ALL PRIVILEGES TO USER;

Privilèges sur toutes les bases

1. **Consulter les tables de toutes les bases**
2. GRANT SELECT ON *.* TO TOTO;
3. **Créer une base, par conséquent les tables**
4. GRANT CREATE ON *.* TO TOTO;
5. **Modifier les tables de toutes les bases**
6. GRANT ALTER ON *.* TO TOTO;
7. **Supprimer les tables de toutes les bases**
8. GRANT DROP ON *.* TO TOTO;

Privilèges sur une Base

1. **Consulter les tables de la base Achats**
2. GRANT SELECT ON Achats.* TO TOTO;
3. **Créer les tables dans la base Achats**
4. GRANT CREATE ON Achats.* TO TOTO;
5. **Modifier les tables de la base Achats**
6. GRANT ALTER ON Achats.* TO TOTO;
7. **Supprimer les tables de la base Achats**
8. GRANT DROP ON Achats.* TO TOTO;

Révoquer des privilèges

- **Révoquer les privilèges à un utilisateur sur une table**
 - REVOKE PERMISSION ON NOM_TABLE FROM USER;
- **Révoquer tous les privilèges d'un utilisateur**
 - REVOKE ALL PRIVILEGES FROM USER;

Révoquer des privilèges sur une Base

1. **Consulter les tables de la base Achats**
2. REVOKE SELECT ON Achats.* FROM TOTO;
3. **Créer les tables dans la base Achats**
4. REVOKE CREATE ON Achats.* FROM TOTO;
5. **Modifier les tables de la base Achats**
6. REVOKE ALTER ON Achats.* FROM TOTO;
7. **Supprimer les tables de la base Achats**
8. REVOKE DROP ON Achats.* TO TOTO;



II.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;



■ **Suppression 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 privilèges à l'utilisateur 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 utilisateurs ayant le privilège de sélection sur toutes les bases**
 - `SELECT user FROM mysql.USER WHERE select_priv = 'Y' ;`
- 3. Lister les utilisateurs ayant le privilège de sélection sur base Achats**
 - `SELECT user FROM mysql.DB WHERE select_priv = 'Y' and db='Achats';`
- 4. Donner à tout utilisateur anonyme le privilège de sélection sur la table 'Etudiant' de la base 'TEST'**
 - `GRANT SELECT on TEST.Etudiant TO "@'%';`
- 5. Désactiver la validation automatique des requêtes dans la session en cours**
 - `SET AUTOCOMMIT = 0 ;`



III.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 constituée 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 accessible 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 fermés 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

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

Modifier les paramètres de Oracle

1. Changer le nomenclature des utilisateurs

- SQL> alter session set "_ORACLE_SCRIPT"=true;

2. Changer le format de la date (ex: 12/04/2022 15:30

- alter session set NLS_DATE_FORMAT='DD/MM/YYYY HH24:MI'

3. Changer le format de la date (exe 12 AVR. 2022

- alter session set NLS_DATE_FORMAT='DD MON YYYY'

4. Changer le format de l'heure

- alter session set NLS_TIME_FORMAT= 'HH24:MI:SS'

Structure d'une base Oracle

■ 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 d'une base Oracle

- **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 paramètre DB_BLOCK_SIZE définit 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 'TBF.dbf' SIZE 10M  
DEFAULT STORAGE  
( INITIAL 1024K NEXT 1024K PCTINCREASE 0);
```

Tablespace

Mettre en offline une tablespace

ALTER TABLESPACE TBF OFFLINE;

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 de 100M

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

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 Etudiant (numEtudiant Integer, ...) tablespace DSTI;
- 2. ou bien (implicitement)
 - **SQL>** create user toto identified by passer default tablespace DSTI
 - **SQL>** create table Etudiant (numEtudiant Integer, ...);

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;
```

---initialise la séquence pour la première fois et donne le suivant pour les autres

```
select ma_sequence.currval from dual;
```

---affiche la valeur 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-incrémenter  
end;
```